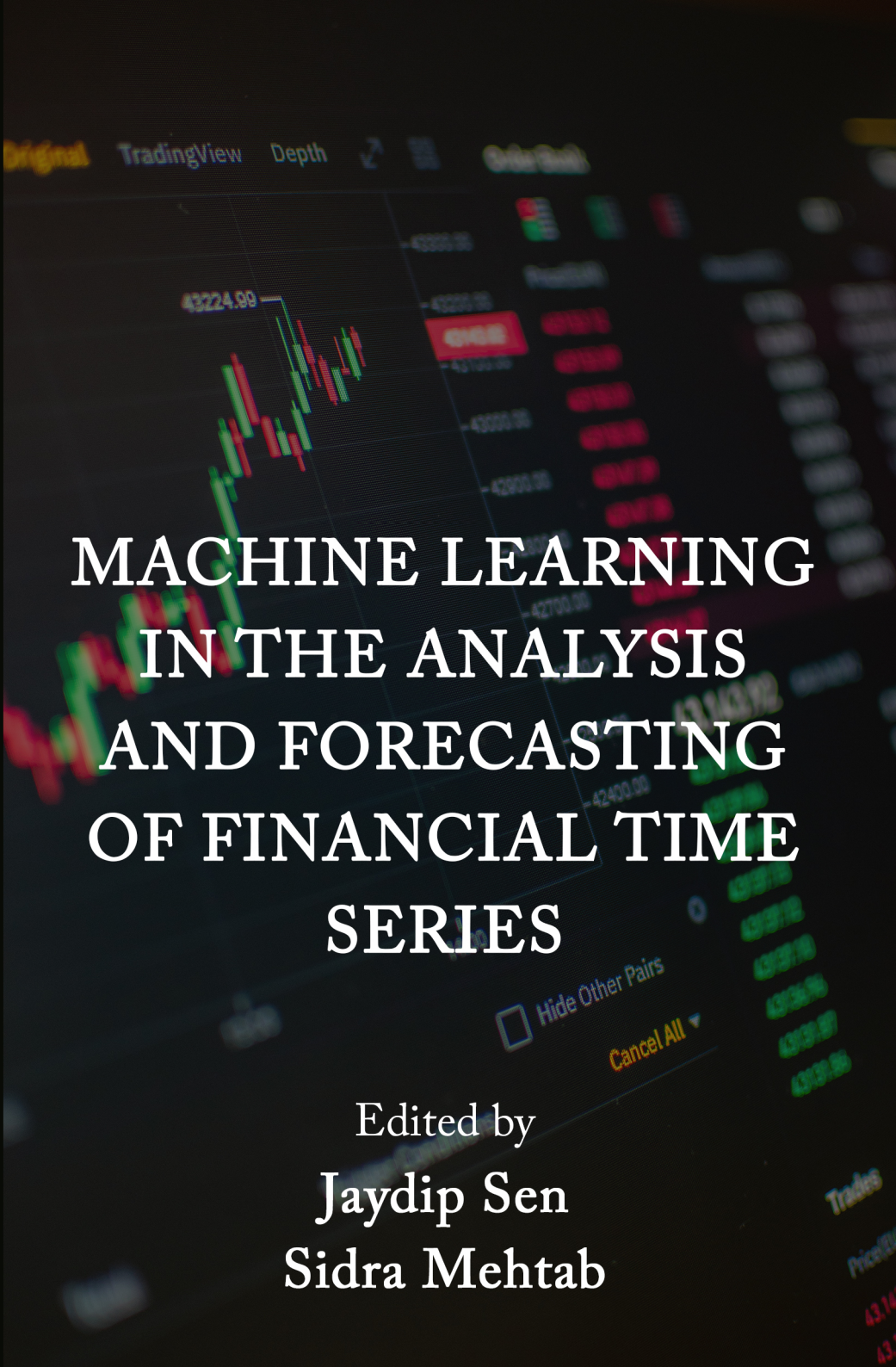

MACHINE LEARNING IN THE ANALYSIS AND FORECASTING OF FINANCIAL TIME SERIES

Edited by

Jaydip Sen

Sidra Mehtab

Machine Learning in the Analysis and Forecasting of Financial Time Series

Machine Learning in the Analysis and Forecasting of Financial Time Series

Edited by

Jaydip Sen and Sidra Mehtab

Cambridge
Scholars
Publishing

Machine Learning in the Analysis and Forecasting of Financial Time Series

Edited by Jaydip Sen and Sidra Mehtab

This book first published 2022

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Copyright © 2022 by Jaydip Sen, Sidra Mehtab and contributors

All rights for this book reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owner.

ISBN (10): 1-5275-8324-4
ISBN (13): 978-1-5275-8324-5

Dedicated to my sister Nabanita who left us on July 27, 2021.
—Jaydip

TABLE OF CONTENTS

List of Figures..... ix

List of Tables..... xiii

Preface..... xvii

Chapter 1 1
Stock Price Prediction using Deep Learning and Natural Language Processing
Jaydip Sen & Sidra Mehtab

Chapter 2 29
Machine Learning and Deep Learning in Stock Price Prediction
Jaydip Sen

Chapter 3 68
Stock Price Prediction using Convolutional Neural Networks
Jaydip Sen & Sidra Mehtab

Chapter 4 102
Robust Predictive Models for the Indian IT Sector using Machine Learning and Deep Learning
Abhishek Dutta & Jaydip Sen

Chapter 5 197
A Causality Analysis between the Indian Information Technology Sector Index and the DJIA Index
Ashmita Paul & Jaydip Sen

Chapter 6 235
Stock Price Prediction using Machine Learning and Deep Learning Algorithms and Models
Sidra Mehtab & Jaydip Sen

Chapter 7 304
Analysis of Different Sectors of the Indian Economy for Robust Portfolio Construction
Jaydip Sen

Contributors 361

LIST OF FIGURES

Figure 2-1. The ANN model for classification (*Case II)* 47
Figure 2-2. The decision tree regression model (*Case III*) 52
Figure 2-3. The loss convergence in the LSTM model (*Case III*) 56
Figure 3-1. The architecture of the CNN#1 model 79
Figure 3-2. The architecture of the CNN#2 model 81
Figure 3-3. The architecture of the CNN#3 model 83
Figure 3-4. The architecture of the CNN#4 model 84
Figure 3-5. The variation of RMSE for CNN#1 model 90
Figure 3-6. The variation of RMSE for CNN#2 model 91
Figure 3-7. The variation of RMSE for CNN#3 model 92
Figure 3-8. The variation of RMSE for CNN#4 model 93
Figure 4-1(a). Bagging – actual and predicted values (*Case I*) 114
Figure 4-1(b). Bagging- actual and predicted relationship (*Case I*) 114
Figure 4-2. Bagging – actual and predicted values (*Case II*)..... 115
Figure 4-3 (a). Bagging – the residual plot (*Case I*) 116
Figure 4-3 (b). Bagging – the residual plot (*Case II*) 116
Figure 4-4 (a). Boosting – actual and predicted values (*Case I*)..... 118
Figure 4-4 (b). Boosting – actual and predicted relationship (*Case I*)..... 119
Figure 4-5. Boosting – actual and predicted relationship (*Case II*) 119
Figure 4-6 (a). Boosting – the residual plot (*Case I*) 120
Figure 4-6 (b). Boosting – the residual plot (*Case II*)..... 120
Figure 4-7 (a). The decision tree regression model (*Case I*) 122
Figure 4-7 (b). Decision tree – actual and predicted values (*Case I*)..... 123
Figure 4-7 (c). Decision tree – actual and predicted relation (*Case I*)..... 123
Figure 4-8 (a). Decision tree – actual and predicted values (*Case II*)..... 125
Figure 4-8 (b). Decision tree – actual and predicted relation (*Case II*) ... 125
Figure 4-9 (a). Decision tree – the residual plot (*Case I*) 126
Figure 4-9 (b). Decision tree – the residual plot (*Case II*)..... 126
Figure 4-10 (a). Random forest – actual and predicted values (*Case I*) .. 128
Figure 4-10 (b). Random forest – act and pred relation (*Case I*)..... 128
Figure 4-11. Random forest – actual and predicted (*Case II*)..... 130
Figure 4-12(a). Random forest – the residual plot (*Case I*)..... 131
Figure 4-12(b). Random forest – the residual plot (*Case II*) 131
Figure 4-13. The architecture of the ANN regression model (*Case I*)..... 133
Figure 4-14 (a). ANN – actual and predicted values (*Case I)* 134

Figure 4-14 (b). ANN – actual and predicted relation (*Case I*) 134
Figure 4-15. The architecture of the ANN regression model (*Case II*) ... 135
Figure 4-16. ANN regression– actual and predicted values (*Case II*) 136
Figure 4-17 (a). ANN regression – the residual plot (*Case I*)..... 137
Figure 4-17 (b). ANN regression – the residual plot (*Case II*) 137
Figure 4-18 (a). Multivariate regression – act and pred values (*Case I*).. 140
Figure 4-18 (b). Multivariate reg – act. and pred. relation (*Case I*)..... 140
Figure 4-19. Multivariate regression – act. and pred. values (*Case II*).... 141
Figure 4-20 (a). Multivariate regression – the residual plot (*Case I*)..... 142
Figure 4-20 (b). Multivariate regression – the residual plot (*Case II*) 142
Figure 4-21. MARS regression – post model tuning (*Case I*) 145
Figure 4-22 (a). MARS – actual and predicted values (*Case I*)..... 147
Figure 4-22 (b). MARS – actual and predicted relation (*Case I*) 146
Figure 4-23. MARS – actual and predicted values (*Case II*)..... 147
Figure 4-24 (a). MARS regression – the residual plot (*Case I*)..... 148
Figure 4-24 (b). MARS regression – the residual plot (*Case II*) 148
Figure 4-25. SVM model - a hyperplane separating two classes..... 150
Figure 4-26 (a). SVM regression – act and pred values (*Case I*)..... 151
Figure 4-26 (b). SVM – actual and predicted relation (*Case I*) 151
Figure 4-27. SVM regression – act and pred values (*Case II*)..... 153
Figure 4-28 (a). SVM regression – the residual plot (*Case I*)..... 154
Figure 4-28 (b). SVM regression – the residual plot (*Case II*) 154
Figure 4-29 (a). Logistic regression – lift curve (*Case I*) 159
Figure 4-29 (b). Logistic regression – ROC curve (*Case I*)..... 160
Figure 4-30 (a). Logistic regression – act and pred prob (*Case II*)..... 162
Figure 4-30 (b). Logistic regression – lift curve (*Case II*)..... 163
Figure 4-30 (c). Logistic regression – ROC curve (*Case II*) 164
Figure 4-31 (a). Bagging – actual and predicted probabilities (*Case I*)... 166
Figure 4-31 (b). Bagging – actual and predicted probabilities (*Case II*).. 167
Figure 4-32. KNN – actual and predicted probabilities (*Case I*) 169
Figure 4-33. Decision tree classification model (*Case I*)..... 171
Figure 4-34. Architecture of the ANN classification model (*Case I*) 173
Figure 4-35 (a). ANN – actual and predicted probs (*Case I*)..... 174
Figure 4-35 (b). ANN – actual and predicted probs (*Case II*) 174
Figure 4-36. Boosting – actual and predicted probs (*Case I*) 177
Figure 4-37 (a). Random forest – act and pred probabilities (*Case I*) 179
Figure 4-37 (b). Random forest – actual and predicted probs (*Case II*) .. 179
Figure 4-38 (a). SVM – actual and predicted probs (*Case I*)..... 181
Figure 4-38 (b). SVM – actual and predicted probabilities (*Case II*) 181
Figure 4-39. Indian IT sector daily index values for the training LSTM ... 184
Figure 4-40. The convergence of the loss for the LSTM model..... 185

Figure 4-41. Training and validation accuracies of the LSTM model..... 186
Figure 4-42. The architecture of the LSTM regression model 187
Figure 4-43. Actual vs the predicted values of the LSTM model..... 188
Figure 5-1. Multivariate *IT_DJIA* – act and pred values (*Case I*) 207
Figure 5-2. Multivariate *IT_DJIA* – act and pred values (*Case II*) 207
Figure 5-3. MARS regression *IT_DJIA* – act vs. pred values (*Case I*).... 209
Figure 5-4. MARS regression *IT_DJIA* – act vs. pred values (*Case II*) .. 210
Figure 5-5. Bagging *IT_DJIA* – act and pred values (*Case I*)..... 211
Figure 5-6. Bagging *IT_DJIA* – act and pred values (*Case II*) 212
Figure 5-7. Boosting *IT_DJIA* – act and the pred values (*Case I*)..... 213
Figure 5-8. Boosting *IT_DJIA* – act and pred relation (*Case I*)..... 213
Figure 5-9. Boosting *IT_DJIA* – the residual plot (*Case I*)..... 213
Figure 5-10. Boosting *IT_DJIA* – act and pred values (*Case II*) 214
Figure 5-11. Boosting IT_DJIA – act and pred relation (*Case II*)..... 214
Figure 5-12. Boosting *IT_DJIA*–the residual plot (*Case II*) 214
Figure 5-13. Random forest *IT_DJIA*– act and pred values of (*Case I*) .. 216
Figure 5-14. Random forest *IT_DJIA* – act and pred relation (*Case I*).... 216
Figure 5-15. Random forest *IT_DJIA*–the residual plot (*Case I*) 216
Figure 5-16. Random forest *IT_DJIA*– act and pred values (*Case II*) 217
Figure 5-17. Random forest *IT_DJIA*– act and pred relation (*Case II*) ... 217
Figure 5-18. Random forest *IT_DJIA*– the residual plot (*Case II*) 217
Figure 5-19. Architecture of ANN regression model *IT_DJIA* (*Case I*).. 219
Figure 5-20. ANN *IT_DJIA* – actual and the predicted values (*Case I*) .. 219
Figure 5-21. ANN *IT_DJIA* – act and the pred values (*Case II*) 220
Figure 5-22. ANN regression model architecture *IT* 220
Figure 5-23. SVM *IT_DJIA* model – act and the pred values (*Case I*).... 222
Figure 5-24. SVM *IT_DJIA* model – act and the pred values (*Case II*) .. 222
Figure 5-25. Indian IT sector index for training *IT_DJIA* (*Case I*) 225
Figure 5-26. The *IT_DJIA LSTM* model loss (*Case I*)..... 226
Figure 5-27. The *IT_DJIA LSTM* model loss (*Case II*) 226
Figure 5-28. The IT LSTM model loss (*Case I*)..... 227
Figure 5-29. The *IT LSTM* model loss (*Case II*)..... 227
Figure 6-1. The logistic regression – act vs. pred probs (*Case III*)..... 259
Figure 6-2. The logistic regression–the *lift* curve (*Case III*)..... 259
Figure 6-3. The logistic regression – the *ROC curve*..... 259
Figure 6-4. The decision tree for classification (*Case III*)..... 262
Figure 6-5. The bagging classification – act vs. pred probs (*Case III*).... 263
Figure 6-6 The boosting classification – act vs pred probs (*Case III*)..... 265
Figure 6-7. The ANN classification model (*Case III*) 268
Figure 6-8. The ANN classifier – act vs pred probs (*Case III*)..... 268
Figure 6-9. Multivariate regression – act and pred values (*Case III*) 271

Figure 6-10. Multivariate regression – act and pred relation (*Case III*) .. 271
Figure 6-11. Multivariate regression – the residual plot (*Case III*) 272
Figure 6-12. The MARS regression – act and pred values (*Case III*)..... 274
Figure 6-13. The MARS regression – act and pred relation (*Case III*).... 275
Figure 6-14. The MARS regression – the residual plot (*Case III*) 275
Figure 6-15. Decision tree regression model (*Case III*) 276
Figure 6-16. Decision tree – act and pred values (*Case III*) 277
Figure 6-17. Decision tree – act and pred relation (*Case III*) 277
Figure 6-18. Decision tree – the residual plot (*Case III*) 277
Figure 6-19. Bagging – actual and predicted values (*Case III*) 279
Figure 6-20. Bagging – act and pred relation (*Case III*)..... 279
Figure 6-21. Bagging – the residual plot (*Case III*)..... 279
Figure 6-22. Boosting – actual and predicted values (*Case III*) 281
Figure 6-23. Boosting – act and pred relation (*Case III*)..... 281
Figure 6-24. Boosting – the residual plot (*Case III*)..... 282
Figure 6-25. Random forest – act and pred values (*Case III*)..... 283
Figure 6-26. Random forest – act and pred relation (*Case III*)..... 283
Figure 6-27. Random forest – the residual plot (*Case III*)..... 284
Figure 6-28. ANN regression model architecture (*Case III*) 285
Figure 6-29. ANN regression – act and pred values (*Case III*) 285
Figure 6-30. ANN regression – act and pred relation (*Case III*) 286
Figure 6-31. ANN regression – the residual plot (*Case III*) 286
Figure 6-32. SVM regression – act and pred values (*Case III*) 287
Figure 6-33. SVM regression – act and pred relation (*Case III*) 288
Figure 6-34. SVM regression – the residual plot (*Case III*) 288
Figure 6-35. The stock data of Godrej consumers for 2013 (*Case I*) 289
Figure 6-36. LSTM model architecture (*Cases I*, *II*, and *III*) 289
Figure 6-37. LSTM regression model loss (*Case I*) 290
Figure 6-38. The stock data of Godrej Consumers for 2014 (*Case II*) 291
Figure 6-39. LSTM model loss (*Case II*) 291
Figure 6-40. Godrej Consumers stock price for 2013-14 (*Case III*)..... 292
Figure 6-41. LSTM model loss (*Case III*) 292
Figure 7-1. Decomposition of Indian banking sector index time series .. 314
Figure 7-2. Actual and predicted Indian IT index using *Method I*..... 340
Figure 7-3. Actual and predicted Indian IT index using *Method II* 342
Figure 7-4. First-order difference of the Indian IT index series 345
Figure 7-5. The ACF plot of the first-order difference series..... 346
Figure 7-6. The PACF plot of the first-order difference series..... 346
Figure 7-7. The ACF plot of the residuals for the best-fit ARIMA 348
Figure 7-8. Actual and predicted Indian IT index using *Method III*..... 349
Figure 7-9. Actual and predicted IT sector index using Method IV 351

LIST OF TABLES

Table 1-1. Logistic regression classification results 16
Table 1-2. KNN classification results..... 16
Table 1-3. Decision tree (CART) classification results 16
Table 1-4. Bagging classification results..... 16
Table 1-5. Boosting classification results..... 17
Table 1-6. Random forest classification results..... 17
Table 1-7. ANN classification results..... 17
Table 1-8. SVM classification results..... 17
Table 1-9. Multivariate regression results 18
Table 1-10. Decision tree regression results..... 18
Table 1-11. Bagging regression results 18
Table 1-12. Boosting regression results..... 18
Table 1-13. Random forest regression results 18
Table 1-14. ANN regression results 19
Table 1-15. SVM regression results 19
Table 1-16. LSTM regression results 19
Table 1-17. Granger test results at different lags..... 20
Table 1-18. Performance of sentiment augmented model 20
Table 2-1. Logistic regression classification results..... 41
Table 2-2. KNN classification results..... 42
Table 2-3. Decision tree (CART) classification results 42
Table 2-4. Bagging classification results..... 44
Table 2-5. Boosting (AdaBoost) classification results 44
Table 2-6. Random forest classification results..... 46
Table 2-7. ANN classification results..... 46
Table 2-8. SVM classification results..... 48
Table 2-9. Multivariate regression results 50
Table 2-10. MARS regression results..... 50
Table 2-11. Decision tree regression results..... 51
Table 2-12. Bagging regression results 53
Table 2-13. Boosting regression results..... 53
Table 2-14. Random forest regression results 54
Table 2-15. ANN regression results 54
Table 2-16. SVM regression results 55
Table 2-17. LSTM regression results 57

Table 2-18. The best performing classification models 58
Table 2-19. The best performing regression models 58
Table 3-1. Logistic regression classification results 86
Table 3-2. KNN classification results 86
Table 3-3. Decision tree (CART) classification results 86
Table 3-4. Bagging Classification results 86
Table 3-5. Boosting classification results 87
Table 3-6. Random forest classification results 87
Table 3-7. ANN classification results 87
Table 3-8. SVM classification results 87
Table 3-9. Multivariate regression results 88
Table 3-10. Decision tree regression results 88
Table 3-11. Bagging regression results 88
Table 3-12. Boosting regression results 88
Table 3-13. Random forest regression results 88
Table 3-14. ANN regression results 88
Table 3-15. SVM regression results 89
Table 3-16. The performance results of CNN#1 model 89
Table 3-17. The performance results of CNN#2 model 90
Table 3-18. The performance results of CNN#3 model 91
Table 3-19. The performance results of CNN#4 model 93
Table 3-20. The best performing ML classification models 93
Table 3-21. The best performing ML regression models 93
Table 4-1. Bagging regression results 117
Table 4-2. Boosting regression results 121
Table 4-3. Decision tree regression results 127
Table 4-4. Random forest regression results 132
Table 4-5. ANN regression results 138
Table 4-6. Multivariate regression results 143
Table 4.7. MARS regression results 149
Table 4.8. SVM regression results 155
Table 4.9. Logistic regression classification results 165
Table 4-10. Bagging classification results 168
Table 4-11. KNN classification results 170
Table 4-12. Decision tree classification results 172
Table 4-13. ANN classification results. 175
Table 4-14. Boosting classification results 178
Table 4-15. Random forest classification results 180
Table 4-16. SVM classification results 182
Table 4-17. The performance results of the LSTM regression model 189
Table 5-1. Granger causality test with DJIA index as a predictor 205

Table 5-2. Multivariate regression results 208
Table 5-3. MARS regression results..... 210
Table 5-4. Bagging regression results 212
Table 5-5. Boosting regression results..... 215
Table 5-6. Random forest regression results 218
Table 5-7. ANN regression results 221
Table 5-8. SVM regression results 223
Table 5-9. Summary of the IT_DJIA regression models (*Case II*)..... 228
Table 5-10. Summary of the IT regression models (*Case II*) of IT 228
Table 6-1. The performance results of the logistic regression models 258
Table 6-2. The performance results of the KNN models..... 260
Table 6-3. The performance results of the decision tree classifiers..... 261
Table 6-4. The performance results of the bagging classifiers 263
Table 6-5. The performance results of the boosting classifiers 264
Table 6-6. The performance results of the random forest classifiers..... 266
Table 6-7. The performance results of the ANN classifiers 267
Table 6-8. The performance results of the SVM classifiers 268
Table 6-9. The performance of the multivariate regression models 271
Table 6-10. The performance results of the MARS regression models... 274
Table 6-11. The performance of the decision tree regression models 278
Table 6-12. The performance of the bagging regression models..... 280
Table 6-13. The performance of the boosting regression models..... 281
Table 6-14. The performance of the random forest regression..... 283
Table 6-15. The performance of the ANN regression models..... 284
Table 6-16. The performance of the SVM regression models..... 286
Table 6-17. Summary of the classification models (*Case I*)..... 293
Table 6-18. Summary of the classification models (*Case II*) 293
Table 6-19. Summary of the classification models (*Case III*) 294
Table 6-20. Summary of the regression models (*Case I*) 295
Table 6-21. Summary of the regression models (*Case II*) 295
Table 6-22. Summary of the regression models (*Case III*)..... 295
Table 7-1. Indian banking sector time series (Jan 2010 – Dec 2020)..... 317
Table 7-2. Seasonality of different sectors (Jan 2010 – Dec 2020) 321
Table 7-3. Price movements of the auto sector stocks..... 322
Table 7-4. Price movements of the banking sector stocks..... 323
Table 7-5. Price movements of the capital goods sector stocks..... 324
Table 7-6. Price movements of the consumer durable sector stocks 325
Table 7-7. Price movements of the FMCG sector stocks 326
Table 7-8. Price movements of the healthcare sector stocks 327
Table 7-9. Price movements of the IT sector stocks..... 328
Table 7-10. Price movements of the large cap sector stocks 329

Table 7-11. Price movements of the metal sector stocks 330
Table 7-12. Price movements of the mid cap sector stocks 331
Table 7-13. Price movements of the oil and gas sector stocks 332
Table 7-14. Price movements of the power sector stocks 333
Table 7-15. Price movements of the realty sector stocks 334
Table 7-16. Price movements of the small cap sector stocks 335
Table 7-17. Price movements of the telecom sector stocks 336
Table 7-18. Percentage of cases matching sectoral seasonality 336
Table 7-19. Forecasted Indian IT sector index using *Method I* 340
Table 7-20. Forecasted Indian IT sector index using *Method II* 341
Table 7-21. The output of the *auto.arima* (Jan 2010 – Dec 2019) 343
Table 7-22. The six candidate ARIMA models 347
Table 7-23. Forecasted Indian IT sector index using *Method III* 348
Table 7-24. Iterations of building the model for *Method IV* 350
Table 7-25. Forecasted Indian IT sector index using *Method IV* 351
Table 7-26. Performance of the four methods of forecasting 352

PREFACE

The financial sector including financial services, banking, and insurance has witnessed the maximum applications and use cases of machine learning, deep learning, and artificial intelligence. While the financial organizations have only skimmed the surface of the rapidly evolving areas such as deep neural networks and reinforcement learning, the possibility of applying such techniques in many applications in finance and econometrics vastly remains unexplored yet.

The chapters in the volume present several techniques of financial time series analysis and forecasting financial series using statistical, econometric, machine learning, and deep learning approaches. The historical data of the daily and monthly index values of various sectors and important stocks listed in the National Stock Exchange (NSE) of India and the Bombay Stock Exchange (BSE) are used in building predictive models which are later on used to predict the future values of the index and their movement patterns. The time series decomposition results of the financial sectors provide the readers with several useful insights into the behavioral characteristics of different sectors which can prove important for understanding the sectors in a better way. A deeper understanding of the sectoral behavioral patterns will enable the investors to take more effective investment decisions and gain higher profits. The statistical and econometric modeling approaches discussed in the chapters include exponential smoothing methods, Holt and Winter trend and seasonality method, autoregressive (AR) and moving average (MA) method, autoregressive integrated moving average (ARIMA) method, Granger causality analysis, univariate linear regression, multivariate linear regression, and multivariate adaptive regression spline (MARS). The machine learning-based predictive models include several classification and regression approaches including logistic regression, *k*-nearest neighbors, decision trees, bagging, adaptive boosting, extreme gradient boosting, random forest, support vector machine, and artificial neural network. In the deep learning category, the applications of convolutional neural network (CNN) and long-and-short-term memory (LSTM) network architectures are demonstrated in designing predictive models for financial time series data. The use of text mining and natural language processing (NLP) in building precise financial models is also presented in one of the chapters in the volume.

In Chapter 1, *Stock Price Prediction using Deep Learning and Natural Language Processing*, Sen & Mehtab propose several machine learning and deep learning-based models for predicting NIFTY 50 index values and their movement patterns on the National Stock Exchange (NSE) of India. The models are trained on historical NIFTY index values from January 4, 2010, to December 31, 2018. The evaluation of the models is done on test data from January 1, 2019, to December 31, 2019. The authors present an additional text analysis module to improve the accuracy of the predictive models. This is done by incorporating a sentiment analysis module that analyses public sentiment on Twitter on the NIFTY 50 stocks. The output of the sentiment analysis module is used as the second input to the predictive model in addition to the past NIFTY 50 index values. To model the nonlinear relationship between Twitter sentiment and the NIFTY index values at different lags, the authors design a five-layer self-organizing fuzzy neural network (SOFNN) that uses the ellipsoidal function as the basis function. The results indicate that the use of the sentiment analysis module further increases the prediction accuracy of the deep learning-based predictive model.

In Chapter 2, *Machine Learning and Deep Learning in Stock Price Prediction*, Sen presents several machine learning-based models for predicting stock price movement in a short-term time frame using classification techniques. The stock prices are also predicted using a short forecast horizon using several regression models as well. A deep learning-based regression built on long-and-short-term memory (LSTM) network architecture is also presented. The classification and regression models are trained and tested on the historical prices of two important stocks listed on the NSE of India. Extensive results are provided on the performances of the models. The results clearly show that the LSTM is far superior to its machine learning counterparts in predicting future stock prices and their movement patterns for stock price records with rich features and high frequencies.

In Chapter 3, *Stock Price Prediction using Convolutional Neural Networks*, Sen & Mehtab use daily index values of NIFTY 50 from January 29, 2014, to July 31, 2020, for training and testing several machine learning-based classification and regression models. The models are trained on data from December 29, 2014, to December 28, 2018, while evaluation of the models was done on data from December 31, 2018 to July 31, 2020. The classification models are used for predicting the movement patterns of the daily NIFTY index during the test period. On the other hand, the daily NIFTY index values are predicted using the regression models. Four deep learning-based regression models built on the convolutional neural network (CNN) architecture are also proposed. The authors have presented detailed

results on the performances of the predictive models. The results have indicated that the CNN models are superior to the machine learning-based regression models in terms of their prediction accuracies.

In Chapter 4, *Robust Predictive Models for the Indian IT Sector using Machine Learning and Deep Learning*, Dutta & Sen present several machine learning models of classification and regression for accurately predicting the future daily index values and their movement patterns of the Indian information technology (IT) sector listed on the NSE. The Indian IT sector daily index values on the NSE from January 2008 to December 2018 are used for training the models. The models are tested on the data from January 2019 to December 2019. The authors present an extensive set of results that indicates the deep learning model built on LSTM architecture is more accurate than its machine learning counterparts in its future prediction of the daily IT index.

In Chapter 5, *A Causality Analysis between the Indian Information Technology Sector Index and the DJIA Index*, Paul & Sen investigate causality between the information technology sector index on the NSE of India and the Dow Jones Industrial Average (DJIA) index of the USA. In their work, the authors use the historical index values from January 5, 2009, to December 27, 2019. The historical index values from January 5, 2009, to December 28, 2018, are used for training the models while testing is done on the data from December 31, 2018, to December 27, 2019. Two regression models are built for predicting the index values of the Indian IT sector on the NSE. The first one is an autoregressive model that uses the lag values of the series as the predictors, while the other one is multivariate. The multivariate model uses the lag values of the DJIA index in addition to the lag values of the Indian IT sector index as the predictors. For both models, the target variable is the Indian IT sector index. It is found that the multivariate model is more accurate indicating a causal effect of the DJIA index series on the Indian IT sector index series.

In Chapter 6, *Stock Price Prediction using Machine Learning and Deep Learning Algorithms and Models*, Mehtab & Sen demonstrate how stock prices and stock returns and their movement patterns can be predicted on a short-term horizon with a fairly high level of accuracy. The authors propose a composite framework consisting of several statistical, econometric, machine learning, and deep learning architectures and models for predicting the future prices and the future price movements of an important stock listed on the NSE of India. The stock price data consist of records collected at five minutes intervals. The records are aggregated into three slots in a day, and the models are used to predict the price and its movement for successive slots. A deep learning regression model is also

designed on the LSTM architecture and its prediction accuracies are compared with those of the machine learning models. The results show that while all the models are reasonably accurate in their prediction, the LSTM model outperforms the machine learning model yielding a very high level of precision in forecasting.

In Chapter 7, *Analysis of Different Sectors of the Indian Economy for Robust Portfolio Construction*, Sen analyzes the behavioral patterns of the daily index values of fifteen sectors of the Indian stock market listed on the Bombay Stock Exchange (BSE). The author uses the time series decomposition approach to decompose the daily index series of the sectors from January 2010 to December 2020 into their trend, seasonality, and random components. Based on the decomposition results, the author illustrates how the sectors exhibit a difference in their behavior in trend, seasonality, and randomness, and the way these differences can be utilized by the investors to gain profit from the stock market. Furthermore, four methods of forecasting are proposed for predicting monthly sectoral index, and their prediction accuracies are compared. These forecasting models will help the analysts and investors in predicting the future index values accurately.

While the chapters in this volume do not primarily deal with the basic theories on the topics involved, all the relevant principles and fundamentals are discussed in brief in the chapters for the sake of completeness. Hence, even if some background knowledge of statistics, econometrics, and machine learning may be useful, the book does not presume it for the readers. We believe that the volume can be a valuable resource to anybody interested in gaining knowledge in financial time series analysis. However, the primary target audience for the book is the advanced postgraduate and doctoral students of finance, management, data science, computer science, information technology, and econometrics. In addition, faculty members of graduate schools and universities, practitioners in the industry working in the area of financial analytics, risk management, security analysis, and portfolio management, will also find the subject matters discussed in the book quite useful.

We express our sincere thanks to all the contributors to the chapters in the volume. Without their valuable contributions, it would have been impossible to make this project a success. We also express our sincere thanks to Cambridge Scholars Publishing for providing us with the opportunity to publish our work with their prestigious publishing house. Special thanks are due to Adam Rummens and Amanda Miller of Cambridge Scholars Publishing for their patience, cooperation, and support during different phases of the long publishing process. The members of our

respective families and our colleagues have been always the sources of our inspiration, and motivation during such a long, painstaking, and complex scholastic project. Without their support, the publication of this volume would not have been possible. Many thanks to all of them!

Jaydip Sen & Sidra Mehtab
Editors

CHAPTER 1

STOCK PRICE PREDICTION USING DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

JAYDIP SEN & SIDRA MEHTAB

Introduction

Prediction of the future movement of stock prices has been the subject of many studies. On the one hand, there are proponents of the *efficient market hypothesis* (EMH) who claim that stock prices cannot be predicted. Alternatively, some studies have shown that, if correctly modeled, stock prices can be predicted with a reasonable degree of accuracy. The latter focuses on the choice of variables, appropriate functional forms, and forecasting techniques. In this regard, Sen and Datta Chaudhuri proposed a novel approach to stock price forecasting based on a time series decomposition of the stock price time series (Sen & Datta Chaudhuri, 2016a; Sen & Datta Chaudhuri, 2016b; Sen & Datta Chaudhuri, 2016c; Sen & Datta Chaudhuri, 2016d; Sen & Datta Chaudhuri, 2017a; Sen & Datta Chaudhuri, 2017b; Sen & Datta Chaudhuri, 2018; Sen, 2017a; Sen, 2017b). Propositions also exist that use a granular approach to stock price prediction in a short-term time frame using machine learning- and deep learning-based models (Sen & Datta Chaudhuri, 2017c; Sen, 2018).

There are propositions in the literature on the technical analysis of stock prices where the objective is to identify patterns in stock movements and derive profits from them. Various indicators such as *Bollinger Bands*, *moving average convergence divergence* (MACD), *relative strength index* (RSI), *moving average* (MA), *momentum stochastics* (MS), and *meta sine wave* (MSW) have been devised towards this end. For gaining profits, traders extensively use patterns such as *head and shoulders*, *triangle*, *flag*, *Fibonacci fan*, and *Andrew's pitchfork*. These approaches provide the user

with visual manifestations of the indicators, which help ordinary investors understand how stock prices may move.

This chapter proposes several machine learning and deep learning-based predictive models for predicting the NIFTY 50 index movement in the National Stock Exchange (NSE) of India. We use the daily stock price values for the period January 4, 2010, to December 31, 2018, as the training dataset for building the models, and apply the models to predict the daily stock price movement and actual closing value of the stock for the period January 1, 2019, to December 31, 2019. We further augment the predictive model by incorporating a sentiment analysis module that analyses public sentiment on Twitter on the NIFTY 50 stocks. The sentiment analysis module's output is used as the second input to the model with the historical NIFTY 50 data to predict future stock price movements. Following the approach proposed by Mittal and Goel, we classify the public sentiment on Twitter into four classes and study the causal effect of these sentiment classes on the NIFTY 50 stock price movement using the Granger Causality Test (Mittal & Goel, 2012).

We organize the chapter as follows. In the section titled *Problem Statement*, we explicitly define the problem at hand. The section titled *Related Work* provides a brief review of related work on stock price movement prediction. In the section titled *Methodology*, we describe our research methodology. Extensive results on the performance of the predictive models are presented in the section titled *Performance Results*. This section also describes all the predictive models built in this study and the results they have produced. Finally, in the section titled *Conclusion*, we conclude the paper.

Problem Statement

Our proposed method is based on collecting the NIIFTY 50 index's historical values from India's NSE and developing a robust forecasting framework for future stock index movement. We contend that index values' daily movement patterns can be learned using powerful machine learning and deep learning-based approaches. The knowledge gained can be applied to predict the future price movements of stocks. In this study, we choose the prediction horizon as one week. We hypothesize that the learning-based approaches can be further augmented by sentiment analysis of social media data on Twitter so that stock price movements can be predicted with even higher accuracy. Here, we do not address the forecasting of the short-term movement of the stock price for intra-day traders. Instead, our analysis is more relevant to long-term investors.

At any point in time in the Indian economy, given the appetite of financial market players, including individuals, domestic institutions, and foreign financial institutions, there is a finite amount of funds deployed in the stock market. This fund is distributed among various stocks. Thus, if some stock prices rise, other stock prices should fall. Using our proposed approach, an investor will be able to predict the movement pattern of the NIFTY 50 index, which depicts the stock market sentiment in India.

Related Work

We can classify the existing propositions in the literature on stock price movements and stock price predictions into four broad categories based on the choice of variables, approaches, and techniques adopted in modeling. The first category includes approaches that use *simple regression techniques* on cross-sectional data (Asghar et al., 2019; Dutta et al., 2012; Roy et al., 2015; Zhong & Enke, 2017; Jaffe et al., 1989). These models do not yield accurate results because stock price movement is a highly nonlinear process. The propositions in the second category exploit time series models and techniques using *statistical and econometric methods* such as *exponential smoothing*, *autoregressive integrated moving average* (ARIMA), *Granger causality test*, *autoregressive distributed lag* (ARDL), and *generalized autoregressive conditional heteroscedasticity* (GARCH) to forecast stock prices (Du, 2012; Khandelwal et al., 2015; Ning et al., 2019; Pai & Lin, 2005; Wang et al., 2020). The third strand includes propositions using *machine learning*, *deep learning*, and *natural language processing* to predict stock returns (Mehtab & Sen, 2020a; Mehtab et al., 2020d; Mehtab & Sen, 2022; Sen & Mehtab, 2021a; Sen & Mehtab, 2021b; Mehtab & Sen, 2022; Sen et al., 2020; Yang et al., 2017; Wu et al., 2012; Ballings et al., 2015; Lv et al., 2019). The propositions of the fourth category are based on *hybrid models* built on machine learning and deep learning with inputs of historical stock prices and sentiments in news articles on the social web (Attigeri et al., 2015; Mittal & Goel, 2012; Medhat et al., 2014; Nam & Seong, 2019).

First, we briefly discuss some of the regression-based propositions existing in the literature.

Zhong and Enke propose methods for the future stock price and price movement prediction using *auto-regressive moving average* (ARMA), *auto-regressive integrated moving average* (ARIMA), *generalized autoregressive conditional heteroscedastic* (GARCH), and *smooth transition autoregressive* (STAR) models (Zhong & Enke, 2017). The authors further present another set of statistical methods involving multiple

input variables. The models of this category are *linear discriminant analysis* (LDA), *quadratic discriminant analysis* (QDA), and *multivariate regression analysis*. The authors forecast the daily direction of the S&P500 index return based on sixty predictor variables. Three methods of dimensionality reduction are first applied to the dataset. The methods used for dimensionality reduction are (i) *principal component analysis* (PCA), (ii) *fuzzy robust principal component analysis* (FRPCA), and (iii) *kernel-based principal component analysis* (KPCA). A neural network-based classification model is applied to the transformed data to forecast the direction of movement of future returns. It is observed that the PCA-ANN combination yields the highest level of classification accuracy among all the dimensionality reduction methods.

Dutta et al. propose a logistic regression model using financial ratios as the predictors to predict future stock returns (Dutta et al., 2012). This study attempts to classify the performance of companies as *good* or *bad* based on their one-year performance.

Roy et al. propose a penalty-based regression method – *least absolute shrinkage and selection operator* (LASSO) – to predict future stock prices (Roy et al., 2015). LASSO is an efficient linear regression model that avoids model overfitting and enables feature selection. The regression model is tested on the historical stock prices of the Goldman Sachs Group Inc. The performance of the model is found to be superior to that of another penalty-based regression method, the Ridge regression model.

In the following, we provide a brief description of some of the works in the literature based on econometric methods, such as ARIMA, Granger causality, and GARCH.

Du argues that because stock prices are complex nonlinear functions of many economic factors, it is necessary to design nonlinear models to achieve higher forecasting accuracy (Du, 2018). The author proposes a composite model integrating an ARIMA and a *backpropagation* (BP) *neural network* to forecast the Shanghai Securities Composition stock index. The integrated model's forecasting performance is then compared with those of the individual ARIMA and BP models. It is found that the prediction accuracy of the ARIMA-BP model is superior to that of the BP model, which in turn, is observed to be better than that of the linear ARIMA model.

Wang et al. propose a method of combining asymmetry and extreme volatility in stock prices to forecast volatility in future stock prices (Wang et al., 2020). The authors demonstrate that a shock has a highly significant effect on stock price volatility. It is also shown that volatility is affected by the asymmetry effect both in the long- and short-term. The model, when

tested on out-of-sample data, yields an improved accuracy compared to the GARCH-MIDAS model.

Khandelwal et al. present a scheme based on the *discrete wavelet transform* (DWT) for time series forecasting (Khandelwal et al., 2015). The proposed method separates the linear and nonlinear components in a given time series by applying the DWT. After the time series is decomposed, an ARIMA and an *artificial neural network* (ANN) model are separately used on the time series to reconstruct the original time series from its decomposed components. Once the reconstruction is achieved with a sufficiently high accuracy level, the model is tested on four real-world financial time series. The performance of the model on the out-of-sample data is found to be superior to those of the standalone ARIMA, ANN, and the hybrid ARIMA and ANN models proposed by Zhang (Zhang, 2003).

In the following, we discuss the salient features of some machine learning and deep learning-based propositions in the literature for stock price prediction.

Yang et al. present an ensemble of deep learning models for forecasting the Shanghai Composite Index and the SZSE Component Index of the Chinese stock market (Yang et al., 2017). Using the *backpropagation algorithm* to minimize the error and the *Adam* optimizer to optimize the *gradient descent algorithm*, a set of component networks is built. An ensemble of all these component networks is constructed using *bagging* to create a generalized model. The ensemble model is then tested on the out-of-sample data, and trend predictions are computed based on the predicted index values. The accuracy of the trend predictions of the daily *high* and *low* values for the Shanghai Composite Index is 74.15%. The corresponding values for the SZSE Component Index are 73.95% and 72.34%, respectively.

Wu et al. present a novel approach for predicting stock price trends using a combination of *sequential chart patterns*, *l-means*, and the *AprioriAll algorithm* (Wu et al., 2012). The time series of stock prices is divided into a set of subsequences. For each subsequence, appropriate charts are constructed using a *sliding window* method. The charts are then clustered using the *k*-means algorithm (Wu, 2012). After clustering, the charts form the chart pattern sequences, which are mined further to identify frequently occurring patterns using the *AprioriAll* algorithm (Yu & Li, 2018). The existence of such frequent patterns indicates that some market behaviors are exhibited in a correlated fashion. Detection of the correlations of such hidden market behaviors enables accurate prediction of the trend in stock price. The authors validate their propositions by experimental results.

Ballings et al. present a scheme that compares the performances of ensemble methods such as *random forest*, *AdaBoost,* and *kernel factory*,

with single classifier models such as *neural networks*, *logistic regression*, *support vector machines*, and *k-nearest neighbors* (Ballings et al., 2015). Using the time series of historical stock prices of 5767 public companies and the *area under the curve* (AUC) for the *receiver operating characteristic curve* (ROC) as the metric, the models' performances are compared over a forecast horizon of one year. The models, listed in the decreasing order of their performance, are *random forest*, *support vector machine*, *kernel factory*, *AdaBoost*, *neural network*, *k-nearest neighbors*, and *logistic regression*.

Mehtab and Sen present a series of works on the design of predictive models for predicting future stock prices and index values and movements using innovative machine learning and deep learning architectures (Mehtab & Sen, 2020a; Mehtab & Sen, 2020b; Mehtab & Sen, 2020c; Mehtab et al., 2020d; Mehtab & Sen, 2021a; Mehtab & Sen, 2021b; Mehtab & Sen, 2022). The authors use daily historical stock prices and stock index values at an interval of 5 minutes. Exploiting the power of *convolutional neural networks* (CNNs) and *long- and short-term memory* (LSTM) networks, the models are found to have achieved a high level of accuracy on the out-of-sample of data. The authors propose four CNN models and six LSTM models with different architectures and input data shapes (i.e., univariate or multivariate time series data). The models are compared on their *root mean square error* (RMSE) values. The results elicit two exciting observations: (i) the performances of the CNN models are superior to those of the LSTM models, and (ii) the models based on the univariate data yield more accurate results than the models using the multivariate data. In another set of works, Mehtab et al. propose further variants of CNN and LSTM-based models for predicting future stock price values and stock price movements (Mehtab et al., 2020d; Mehtab et al., 2021a; Sen & Mehtab, 2021b). The authors report extensive results of the performances of the models.

Finally, we discuss some of the hybrid models that use sentiments in the news and other relevant textual information for stock price prediction in the following.

Bollen et al. contend that emotions profoundly affect an individual's buy or sell decisions (Bollen et al., 2011). The authors propose a mechanism that computes the collective *mood* states of the public from many Twitter feeds and investigates whether the collective moods exhibit any correlation with the Dow Jones Industrial Average (DJIA) index. The public *moods* from the Twitter feed are computed using two mood tracking tools – (i) *OpinionFinder,* which classifies the *opinions* as either positive or negative, and (ii) the *Google Profile of Mood States* (GPOMS) which measures six dimensions of a *mood*. The six dimensions are – *calm*, *alert*, *sure*, *vital*,

kind, and *happy*. The results show that DJIA index values can be more accurately predicted by including specific public mood dimensions in the model. The model is further augmented using a *Granger causality* analysis and a *self-organizing fuzzy neural network* (SOFNN) so that the public moods can be more efficiently used as a predictor in forecasting the future DJIA index values. An accuracy level of 86% is found in predicting the daily up and down changes in the *DJIA index's close values* using the composite mode with the Granger causality and the SOFNN module.

Checkley et al. study the use of sentiment metrics extracted from microblogs in predicting stock market index values (Checkley et al., 2017). Using the bearish and bullish sentiments from micro-blogging sites at different time intervals, the authors model their forecasting ability in future stock price movements, price volatility, and trade volume. The study reveals a significant causal link between the sentiments on the micro-blogging sites to the price movements, volatility, and the volume traded at short-term time intervals. The study concludes that investors' behavior in stock markets is more like that of a hasty mob than a group of wise people.

Chen et al. study the impact of sentiments from the news sites on the stock market in Taiwan (Chen et al., 2019). After the news articles are collected from the web, they are preprocessed, and the text corpus is transformed into a word feature set using the *Word2Vec* approach (Jatnika et al., 2019). After the *Word2Vec* model is ready, an LSTM-based deep learning model predicts future stock prices using the news articles from the news sites.

Dang and Duong highlight the influence of online news articles on the movement patterns of stock prices (Dang & Duong, 2016). Based on their hypothesis, the authors propose using a time series analysis that integrates a gamut of text mining and text analysis mechanisms. The proposed model is tested on real-world time series of historical stock prices. The model is found to achieve an accuracy of around 73%.

Galvez and Gravano emphasize the impact of online sentiments on the news articles on the stock prices in Argentina (Galvez & Gravano, 2017). The authors investigate two critical issues. The first question that the authors pose is whether there is any useful information in the stock exchange boards for predicting stock returns. The second question is whether the information is novel or is carried in the time series of the historical stock prices. The authors train, validate, and test a series of predictive models using machine learning and topic modeling algorithms in text analysis to address these questions. The models are built using various combinations of features and predictor variables. The study reveals that the information extracted from the online exchange boards complements the information in the time series

of the historical stock prices. Hence, the use of information from the online exchange boards improves the performance of the predictive models.

Hu et al. argue that the quality, veracity, and comprehensiveness of online textual information related to the stock market and stock price movement vary widely, and in many cases, they are low-quality news (Hu et al., 2018). To make learning models robust against such chaotic information, the authors propose three principles of learning: (i) *sequential content dependency*, (ii) *diverse influence*, and (iii) *efficient and effective learning*. The proposition implements the first two characteristics of learning by including a *hybrid attention network* (HAN). The HAN attempts to predict the trend of the stock price time series using a sequence of recent and related online news. A controlled learning mechanism implements the third aspect of learning. Simulations show that trading strategies formulated based on the model yield a significantly improved annualized return on investment compared to the returns produced by the models without the text analytics module.

Jeong et al. propose a predictive model for forecasting stock prices and stock price movement that incorporates an *opinion mining* module (Jeong et al., 2018). The model is capable of performing three major tasks: (i) filtering fake information on the web, (ii) credit risk assessment, and (iii) detection of critical signals and using them in stock price prediction. These three operations are executed iteratively by the model on real-world time series of stock prices and news items available on the web. The results show that the proposed model helps an investor make investment decisions and increases the returns of investments.

Li et al. present a novel architecture of a predictive model for forecasting stock prices that uses an LSTM-based module and a text mining module that incorporates the sentiment of the investors and other market factors (Li et al., 2017). The model extracts the sentiment from the online news and opinions using a Naïve Bayes algorithm that filters out irrational and fake opinions. The proposed model is tested on the CSI300 index and produced an improved forecasting accuracy compared to the benchmark models that do not have any text processing components.

Li et al. discuss the limitation of the *bag-of-words* approach of online news processing in stock price prediction (Li et al., 2014). The authors argue that news sentiment should be considered a vital ring in the chain of sequence from the word patterns in the news to the final price movement of stocks. The authors use the Harvard psychological dictionary and the Loughran-McDonald financial sentiment dictionary to construct the vocabulary for the text processing module. After the vocabulary space is built, news articles are assigned quantitative measures and then placed in

the sentiment space. The model is evaluated for its prediction accuracy under different market classification levels. The results elicit some fascinating observations: (i) the models with the sentiment analysis module outperform the bag-of-words based models at the individual stock, sector, and index levels, (ii) the models that work only on the sentiment polarity and not on the sentiment analysis of the texts, perform poorly, (iii) models using two different dictionaries have a slight difference in their performance.

Nguyen et al. propose a future stock price prediction model that uses social media sentiment (Nguyen et al., 2015). The model uses specific topics of the companies for which the prices are predicted. However, it does not use the overall moods or sentiment in the entire social web. Instead, the scheme segregates the texts in a *message board* based on the topics and then extracts the sentiment from those specific topics. When applied to the time series of the historical prices of eighteen selected stocks over one year, the proposed model is found to yield a 2.07% improved performance than the models based on the historical stock prices only.

Schumaker and Chen use machine learning algorithms to analyze news articles related to finance using techniques such as *bag-of-words*, *noun phrases*, and *named entity recognition* (Schumaker & Chen, 2009). The authors examine 9,211 news articles and 10,259,042 stock quotes of S&P500 stocks over a week and attempt to estimate the stock price twenty minutes after a news article related to the stock is published. Using a *support vector machine* (SVM)-based model for regression, the authors show that, through a model using the information in the articles and the stock price at the time of the news article's release, the prediction error is minimized. The observed *mean-squared error* (MSE) is 0.04261. The *direction of movement* (i.e., increase or decrease in the current value relative to the previous value) of the predicted values matches the actual values in 57.1% of the cases. A simulated trading exercise yields a *return on investment* (ROI) of 2.06%. In the text analysis, it is found that a *proper noun-based scheme* performs better than the *bag-of-word method*.

Shah et al. present a text mining-augmented model incorporating opinion mining from the web in future stock price prediction (Shah et al., 2018). The model retrieves, extracts, and analyzes the effects of sentiment in online news articles. A new dictionary is created using the critical words mined from the web that apply to the financial market. Based on the dictionary, a sentiment analysis model is built. The model is finally tested on stocks from the pharmaceutical sector. Using the news sentiments only, the model yields an accuracy of 701% in predicting the short-term trend of the time series of stock prices.

Souza and Aste present a model to show that future market behavior can be predicted with a high level of accuracy by integrating information from social media and the historical data from financial time series (Souza & Aste, 2019). The authors contend that while the market structure is measured by a suitable numeric measure of the co-movement of the returns on the asset prices, the social structure is measured by the co-movement of the social media opinions of the same assets. Based on their hypothesis, the authors propose a model that predicts the stock prices using the persistence of the links and the formation of the connections by the triad's closure across both the financial and the social media layers. The experimental results show that the proposed model can predict the future market structure more accurately than a base model that assumes a stable financial correlation structure in the economic and social markets.

Wang et al. introduce the concept of *aspect-level sentiment classification* in their proposed model for stock price forecasting (Wang et al., 2016). The authors argue that the sentiment polarity is not solely dependent on the word contents in the sentence but also on the concerned aspect. To incorporate the *aspect-based opinion mining* from the social web, the authors propose an attention-based LSTM model. The attention-based model focuses on different parts of a sentence when it has various aspects from its contextual perspective. The model is tested on the SemEval 2014 dataset and is found to achieve state-of-the-art performance in *aspect-level classification* of opinions from the social web.

Weng et al. hypothesize that a holistic model that combines disparate online sentiments from the online news articles with traditional time series of stock prices for a stock will be more accurate in its prediction (Weng et al., 2017). The authors propose three machine learning models, decision trees, neural networks, and support vector machines, as three base models. The sentiment-integrated model is then tested on the real-world time series of stock prices of Apple Inc. listed on the NASDAQ. The sentiment-integrated model is found to outperform the base models.

Xie and Jiang propose a stock price prediction model that uses text mining and sentiment analysis of online financial news in China (Xie & Jiang, 2016). The model is built on a *support vector machine* (SVM). The authors collect 2302692 news items from the web from January 1, 2008, to January 1, 2015. Based on the news corpus, the authors create a dictionary of *stop words* for a specific lexicon and a precise sentiment dictionary based on those *stop words*. The SVM model is built upon the sentiment dictionary. The results show that both the quality of a news item and its audience size significantly impact future stock prices.

Zhang et al. present a new approach to stock price prediction using a model that can predict future stock price movement (i.e., growth or decline) in a specified time interval (Zhang et al., 2018). The model uses an unsupervised heuristic algorithm that chops the raw time series data of the stock price into multiple parts with a pre-defined fixed length. The chopped pieces of the original time series are categorized into four classes – *up*, *down*, *flat*, and *unknown*. The classification is done based on the behavior of the *close* prices. The prediction models learn from the behavior of the *close* prices of each of these categories. The models are constructed using a hybrid approach based on random forest, imbalanced learning, and feature selection. The testing of the models is done on the Shenzhen Growth Enterprise Market of China. The evaluation results indicate that the models are highly accurate and robust in predicting future stock prices and volatility.

The drawback of most of the existing propositions in the literature for stock price prediction is their inability to accurately predict highly dynamic and fast-changing patterns in stock price movements. The current work attempts to address this shortcoming by exploiting the power of machine learning with sentiment analysis of social media.

Methodology

In the section titled *Problem Statement*, we mentioned that the objective of this work is to develop a robust framework for forecasting the daily movement of the NIFTY 50 index. We collect the NIFTY 50 daily index data from January 4, 2010, to December 31, 2019, from the Yahoo Finance website (Yahoo Website). The raw data consist of the following variables: (i) *date*, (ii) the *open* value of the index, (iii) the *high* value of the index, (iv) the *low* value of the index, (v) the *close* value of the index, and (vi) the *volume* of the stock traded on a given date.

Using the six variables in the raw data, we derive the following variables for building the predictive models. We follow two approaches to forecasting: *regression* and *classification*. The two methods use the target variable in different ways, described later in this section.

The following nine variables are derived and used in our forecasting models:

i) *month*: it refers to the month to which a given record belongs. This variable is coded into numeric data, with "1" referring to January and "12" referring to December. The value of the variable *month* lies in the interval [1, 12].

ii) *day_month*: this variable refers to the day of the month to which a given record belongs. It is a numeric variable lying in the interval [1, 31]. For example, the date April 4, 2015, will have a value of 4 for the variable *day_month*.

iii) *day_week*: it is a numeric variable that refers to the day of the week corresponding to a given record. This variable lies in the interval [1, 5]. Monday is coded as 1, while Friday is coded as 5.

iv) *close_perc*: a numeric variable computed as the percentage change in the *close* prices on two successive days. The computation of the variable is done as follows. Suppose we have two consecutive days: D_1 and D_2. Let the *close* price of the stock for D_1 be X_1 and that for D_2 be X_2. Then, *close_perc* for D_2 is computed as $(X_2 - X_1)/X_1$ in percent. The variable *close_perc* is used as the target variable in our proposed models.

v) *low_perc*: it is calculated in the same way as *close_perc*. It represents the percentage change in the *low* values over two successive days.

vi) *high_perc*: it is also calculated in the same way as *close_perc*. It represents the percentage change in the *high* values over two consecutive days.

vii) *open_perc*: computed in the same ways as *close_perc*. It represents the percentage change in the *open* values over successive days.

viii) *vol_perc*: it is computed as the percentage change in the values of *volume* over two consecutive days.

ix) *range_perc*: it is calculated as a percentage change in the *range* values for two successive days. The *range* for a day is computed as the *difference* between the *high* and the *low* values. For two successive days, D_1 and D_2, suppose the *high* and *low* values are H_1, H_2, L_1, and L_2, respectively. Hence, the *range* value for D_1 is $R_1 = (H_1 - L_1)$ and for D_2 is $R_2 = (H_2 - L_2)$. The *range_perc* for the day D_2 is computed as $(R_2 - R_1)/ R_1$.

After we compute the values of the above nine variables for each day for the NIFTY 50 data from January 4, 2010, to December 31, 2019, we develop the machine learning and the deep learning models for classification and regression.

For training the models, we use the NIFTY index records for January 4, 2010, to December 31, 2018, while the models are tested using the data for the period January 1, 2019, to December 31, 2019. In the *regression approach*, based on the historical movement of the stock prices, we predict the daily NIFTY index for the test period, i.e., January 1, 2019, to December 31, 2019. The variable *close_perc* is used as the response variable (i.e., the target variable). According to the requirement of a regression model, the response variable *close_perc* is a continuous numeric variable. The

objective of the regression models is to predict the value of the *close_perc* for every day of 2019 after they are trained on the records of the training dataset. A positive value of the predicted *close_perc* indicates an expected increase in the NIFTY index that day compared to its value on the previous day. A negative *close_perc* value, on the other hand, suggests a fall in the NIFTY index.

In the *classification approach*, the response variable *close_perc* is a categorical variable with two possible labels: "0" or "1". The value "0" indicates a negative *close_perc* value, while "1" signifies a positive *close_perc* value. If the model predicts a rise in the *close_perc* value on the next day, then the label for the *close_perc* for the next day will be "1". A label of "0" will indicate a predicted negative value of the *close_perc* on the next day.

All classification and regression models are evaluated under two cases.

Case I: For training the models, we use the NIFTY index data from January 4, 2010, to December 31, 2018. The training dataset consists of 2347 records of the daily NIFTY index values. After the models are built, they are evaluated on the training data. In *Case I*, we compute the training accuracies of the models.

Case II: Here, we evaluate the models on the test dataset. The test data contain the daily NIFTY index values for January 1, 2019, to December 31, 2019. The test dataset consists of 261 records. The performance of the models in testing is evaluated based on their forecasting accuracies on the test data.

We design eight classification and eight regression models for developing the forecasting framework. The eight classification models are: (i) *logistic regression*, (ii) *k-nearest neighbor* (iii) *decision tree*, (iv) *bagging*, (v) *boosting*, (vi) *random forest*, (vii) *artificial neural network*, and (viii) *support vector machines*. The classification models are evaluated using the following metrics: (i) *sensitivity*, (ii) *specificity*, (iii) *positive predictive value*, (iv) *negative predictive value*, (v) *classification accuracy*, and (vi) *F1-score*.

We also build the following eight regression models: (i) *multivariate regression*, (ii) *decision tree*, (iii) *bagging*, (iv) *boosting*, (v) *random forest*, (vi) *artificial neural network*, (vii) *support vector machine*, and (viii) *long- and short-term memory* (LSTM) network. LSTM is a deep learning-based regression model, while the remaining seven models are based on machine learning. For evaluating the performance of the regression models, we use two metrics. These metrics are: (i) *mean absolute percentage error* (MAPE) and (ii) *correlation coefficient* between the actual and predicted values of the response variable (i.e., *close_perc*).

A further improved regression model is built using a sentiment analysis module that analyses NIFTY 50 stocks' tweets on the social web during the training and test periods. The tweets are first extracted from Twitter using the *Twitter Streaming APIs* in the Python language. The raw tweets are preprocessed using the *bag-of-words,* and the *lemmatization* approaches in the SpaCy toolkit in Python. The processed tweets are then fed into a sentiment analysis algorithm that categorizes the tweets into four *mood* classes. Following the approach proposed by Mittal and Goel, we use four categories of mood: *calm*, *happy*, *alert,* and *kind* (Mittal & Goel, 2012). The classified *mood* and the daily *close_perc* values of NIFTY 50 are passed as inputs to the SOFNN algorithm. The SOFNN algorithm produces the predicted daily values of the *close_perc* of NIFTY 50.

Motivated by Mittal and Goel's work, we use the *Granger Causality test* to validate our hypothesis. Our hypothesis assumes that the past and the current mood values returned by the sentiment analysis algorithm have a robust causal effect on the future *close_perc* values (Mittal & Goel, 2012). For this purpose, we compute the *p*-values of the Granger test for several lag periods. The *p*-value here determines the statistical significance of our hypothesis, and they show the likelihood of getting a piece of causality information by some random chance. Therefore, a low *p*-value implies a more robust causality, which indicates that the model has a higher predictive power.

The Granger causality approach is based on the assumption of a linear relationship. However, we suspect a high degree of nonlinearity between the sentiment status measured by the *mood* values and the actual *close_perc* values. Thus, we focus on building a nonlinear predictive model using the *mood* values at different lags with the current value of the *close_perc*. The maximum lag value that we choose is three days. We use a *self-organizing fuzzy neural network* (SOFNN) for designing the nonlinear model. The SOFNN algorithm is implemented using the NEFCLASS of Java. The SOFNN model consists of a five-layer fuzzy neural network that uses the *ellipsoidal function* as its basis function (Mittal & Goel, 2012).

For the training and the validation of the SOFNN-based sentiment analysis model, we do not use the *k*-fold *sequential cross-validation* (*k-SCV*). Instead, we train the model using the NIFTY 50 index records in the training dataset and test the model by predicting the daily *close_perc* values for each day of the following week. For evaluating the model performance, we compute the MAPE and the correlation coefficient between the actual and predicted values of *close_perc*.

Performance Results

This section provides a detailed discussion on the forecasting techniques that we have used and the results obtained using those techniques. We discuss the classification techniques, the regression techniques, and the sentiment analysis-based SOFNN model.

For evaluating the classification-based models, we use the following metrics:

Sensitivity: The ratio of the number of *true positive* cases (i.e., the correctly classified "1" cases) to the total number of *positive* cases in the dataset, expressed as a percentage. Sensitivity is also known as *recall*. Here, *positive* refers to the records belonging to the class "1". Sensitivity is expressed as a percentage.

Specificity: The ratio of the *true negative* cases (i.e., the correctly classified "0" cases) to the total number of *negative* cases in the dataset, expressed as a percentage. Here, *negative* refers to the records belonging to the class "0". Specificity is, most often, expressed as a percentage.

Positive Predictive Value (PPV): The ratio of the number of *true positive* cases to the sum of the *true positive* cases and the *false positive* cases, expressed as a percentage. PPV is also known as *precision*.

Negative Predictive Value (NPV): The ratio of the number of *true negative* cases to the sum of the *true negative* cases and the *false negative* cases, expressed as a percentage.

Classification Accuracy (CA): The ratio of the number of correctly classified cases to the total number of cases in the dataset, expressed as a percentage.

F1-score: The harmonic mean of the *sensitivity* (i.e., *recall*) and PPV (i.e., *precision*). *F1-score* is an important metric for a classification model, and it is computed using (1):

$$F1score = \frac{2*Sensitivity*PPV}{(Sensitivity+PPV)} \quad (1)$$

For evaluating the regression models, we use three metrics, which are as follows: (i) the ratio of RMSE to the *mean* of the absolute values of the target variable (i.e., the mean of the absolute values of *close_perc*), (ii) the *mean absolute percentage error* (MAPE), and (iii) the *product-moment correlation coefficient* between the actual and the predicted *close_perc* values of the NIFTY 50 index.

In Tables 1-1 to 1-8, we present the performance results of the machine learning-based classification models.

TABLE 1-1. LOGISTIC REGRESSION CLASSIFICATION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Sensitivity	42.86	Sensitivity	72.46
	Specificity	94.32	Specificity	81.08
	PPV	70.91	PPV	60.48
	NPV	83.62	NPV	88.05
	CA	81.74	CA	78.62
	F1-score	53.43	F1-score	65.93

TABLE 1-2. KNN CLASSIFICATION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Sensitivity	64.29	Sensitivity	10.63
	Specificity	94.67	Specificity	94.98
	PPV	79.59	PPV	45.83
	NPV	89.13	NPV	72.67
	CA	87.25	CA	70.90
	F1-score	71.13	F1-score	17.26

TABLE 1-3. DECISION TREE (CART) CLASSIFICATION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Sensitivity	40.66	Sensitivity	36.23
	Specificity	92.18	Specificity	89.77
	PPV	62.71	PPV	58.59
	NPV	82.78	NPV	77.89
	CA	79.60	CA	74.48
	F1-score	49.33	F1-score	44.77

TABLE 1-4. BAGGING CLASSIFICATION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Sensitivity	68.13	Sensitivity	41.06
	Specificity	96.80	Specificity	85.33
	PPV	87.32	PPV	52.80
	NPV	90.38	NPV	78.37
	CA	89.80	CA	72.69
	F1-score	76.54	F1-score	46.20

TABLE 1-5. BOOSTING (ADABOOST) CLASSIFICATION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Sensitivity	100.00	Sensitivity	54.60
	Specificity	100.00	Specificity	79.67
	PPV	100.00	PPV	45.89
	NPV	100.00	NPV	84.75
	CA	100.00	CA	73.66
	F1-score	100.00	F1-score	49.87

TABLE 1-6. RANDOM FOREST CLASSIFICATION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Sensitivity	37.36	Sensitivity	72.95
	Specificity	88.46	Specificity	63.51
	PPV	51.13	PPV	44.41
	NPV	81.37	NPV	85.45
	CA	75.97	CA	66.21
	F1-score	43.17	F1-score	55.21

TABLE 1-7. ANN CLASSIFICATION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Sensitivity	68.12	Sensitivity	7.73
	Specificity	91.31	Specificity	99.95
	PPV	75.81	PPV	99.98
	NPV	87.76	NPV	73.06
	CA	83.17	CA	73.66
	F1-score	71.76	F-score	14.35

TABLE 1-8. SVM CLASSIFICATION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Sensitivity	64.71	Sensitivity	71.67
	Specificity	78.53	Specificity	75.34
	PPV	18.13	PPV	20.77
	NPV	96.80	NPV	96.72
	CA	77.58	CA	75.03
	F1-score	28.32	F1-score	32.21

The performance results of the machine learning-based regression models are presented in Tables 1-9 to 1-15.

TABLE 1-9. MULTIVARIATE REGRESSION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Correlation	0.56	Correlation	0.53
	MAPE	35.02	MAPE	90.00
	RMSE/Mean	37.23	RMSE/Mean	84.31

TABLE 1-10. DECISION TREE REGRESSION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Correlation	0.98	Correlation	0.02
	MAPE	60.73	MAPE	78.37
	RMSE/Mean	62.26	RMSE/Mean	81.34

TABLE 1-11. BAGGING REGRESSION RESULTS

Index	Case I Training Data		Case II Test Data	
NIFTY 50	Correlation	0.70	Correlation	0.51
	MAPE	23.47	MAPE	29.32
	RMSE/Mean	24.31	RMSE/Mean	29.84

TABLE 1-12. BOOSTING REGRESSION RESULTS

Stock	Case I Training Data		Case II Test Data	
NIFTY 50	Correlation	0.69	Correlation	0.55
	MAPE	16.72	MAPE	21.34
	RMSE/Mean	17.23	RMSE/Mean	24.52

TABLE 1-13. RANDOM FOREST REGRESSION

Stock	Case I Training Data		Case II Test Data	
NIFTY 50	Correlation	0.95	Correlation	0.65
	MAPE	15.23	MAPE	19.35
	RMSE/Mean	17.23	RMSE/Mean	22.52

TABLE 1-14. ANN REGRESSION RESULTS

Stock	Case I Training Data		Case II Test Data	
NIFTY 50	Correlation	0.73	Correlation	0.42
	MAPE	12.32	MAPE	25.72
	RMSE/Mean	14.62	RMSE/Mean	28.87

TABLE 1-15. SVM REGRESSION RESULTS

Stock	Case I Training Data		Case II Test Data	
NIFTY 50	Correlation	0.71	Correlation	0.75
	MAPE	17.31	MAPE	13.59
	RMSE/Mean	17.83	RMSE/Mean	13.73

TABLE 1-16. LSTM REGRESSION RESULTS

Stock	Case I Training Data		Case II Test Data	
NIFTY 50	Correlation	0.99	Correlation	0.99
	MAPE	8.70	MAPE	10.75
	RMSE/Mean	8.81	RMSE/Mean	11.57
	Matched Cases	89%	Matched Cases	80%

After evaluating the machine learning models, we build a deep learning-based regression model using an LSTM network. LSTM is a variant of *recurrent neural networks* (RNNs) - neural networks with *feedback loops* (Geron, 2019). In such networks, the output at the current time slot depends on the current inputs and the previous state of the network. Moreover, LSTM networks overcome the problem of *vanishing and exploding gradients* of RNNs during the backpropagation phase in learning the weights of the network links (Geron, 2019). We use the Python programming language and the Tensorflow framework for implementing the LSTM regression model. The model is then used to predict the *close_perc* values of the NIFTY 50 index on a forecast horizon of one week. We use the *mean absolute error* (MAE) as the *loss function* and *Adam* as the optimizer. However, we train and validate the network with different values of epoch and batch size under the two cases for achieving the optimum performance of the network.

For the NIFTY 50 training data, i.e., in *Case I*, we use 80 epochs with a batch size of 60. On completion of epoch 72, the *model loss* is found to

converge to as low a value as 0.0102. On completion of epoch 80, the loss slowly increases to a value of 0.0678. For the test dataset, i.e., in *Case II*, we find 70 epochs with a batch size of 60 are sufficient. The loss converges to 0.0102 at the end of epoch 62. Upon completion of epoch 70, the loss is found to have increased very slowly to attain a value of 0.0251. Table 1-16 presents the performance results of the LSTM regression model.

TABLE 1-17. GRANGER TEST RESULTS AT DIFFERENT LAGS

Lag	**Sentiment Class**			
	Calm	**Happy**	**Alert**	**Kind**
1	0.0317	0.5867	0.0432	0.1278
2	0.0389	0.2374	0.2168	0.1325
3	0.0217	0.0874	0.3124	0.1653
4	0.0047	0.0542	0.3456	0.1732
5	0.0154	0.0673	0.1257	0.2345

Finally, we present the results of the sentiment analysis-based nonlinear regression model. First, we present the *p*-values of the Granger causality test between the *moods* and the actual values of *close_perc* in the training dataset for different values of lag. Table 1-17 shows the results. The *mood* states of *calmness* and *happiness* exhibit more substantial causal effects on the *close_perc* values. Hence, these *moods* are expected to yield more accurate predicted values of *close_perc*.

TABLE 1-18. PERFORMANCE OF SENTIMENT AUGMENTED MODEL

Model	**Case I Training Data**		**Case II Test Data**	
SOFNN	Correlation	1.00	Correlation	1.00
	MAPE	4.37	MAPE	5.37
	RMSE/Mean	5.14	RMSE/Mean	5.62
	Matched Cases	93%	Matched Cases	86%

The performance results of the sentiment analysis-based SOFNN model are presented in Table 1-18. The SOFNN model is built using the training dataset records of the historical NIFTY 50 index values. The sentiment values are provided as the second input to the model, and the *close_perc* values are predicted for the next five days (i.e., one week). The same steps are repeated for the test dataset with the identical forecast horizon. The metric "Matched Cases" in Table 1-18 represents the percentage of cases for which the model correctly predicts the up or down

movement patterns of the *close_perc* values. We observe that the SOFNN model with the sentiment analysis component has outperformed the deep learning-based LSTM model.

Among the classification models, in the training phase, the *boosting* model, quite expectedly, has yielded the best performance on all metrics. However, on the test dataset, the *random forest* model has produced the best results. Due to the NIFTY 50 dataset's imbalanced nature, the ANN classifier has performed the best on specificity and PPV while yielding a poor sensitivity value. As expected, the SVM model has performed well on all metrics except PPV.

Among the regression models, while the LSTM model has outperformed all machine learning models, the sentiment analysis-based SOFNN model has performed the best.

Conclusion

This chapter has presented several predictive models for forecasting the future movement patterns and future values of the stock market index on a weekly forecast horizon using eight regression and eight classification methods. These models are based on machine learning and deep learning approaches. We built, fine-tuned, and then tested these models using the daily data of the NIFTY 50 index from January 4, 2010, to December 31, 2019. The raw data are suitably preprocessed, and relevant variables are identified for building the predictive models. After designing and testing the machine learning and deep learning-based models, the predictive framework is further augmented using the public sentiment on NIFTY 50 stocks on Twitter. The public opinions in the tweets and the historical NIFTY 50 index values are used as the two inputs to a SOFNN model built on a fuzzy neural network. The sentiment analysis-based model is found to have yielded the most accurate results in forecasting. The study has proved that public sentiment in social media serves as a significant input for future stock price prediction.

References

Asghar, M. Z., Rahman, F., Kundi, F. M. and Ahmed, S. (2019) "Development of stock market trend prediction system using multiple regression", *Computational and Mathematical Organization Theory*, Vol. 25, pp. 271-301. DOI: 10.1007/s10588-019-09292-7.

Attigeri, G. V., Pai, M. M. M., Pai, R. M. and Nayak, A. (2015) "Stock market prediction: A big data approach", *Proceedings of TENCON*

2015 – 2015 IEEE Region 10 Conference, November 1-4, Macao, China, pp. 1-5. DOI: 10.1109/TENCON.2015.7373006.

Ballings, M., den Poel, D. V., Hespeels, N. and Gryp, R. (2015) "Evaluating multiple classifiers for stock price direction prediction", *Expert Systems with Applications*, Vol. 42, No. 20, pp.7046-7056. DOI: 10.1016/j.eswa.2015.05.013.

Bollen, J., Mao, H. and Zeng, X. (2011) "Twitter mood predicts the stock market", *Journal of Computational Science*, Vol. 2, No. 1, pp. 1-8. DOI: 10.1016/j.jocs.2010.12.007.

Checkley, M. S., Higon, D. A. and Alles, H. (2017) "The hasty wisdom of the mob: How market sentiment predicts stock market behavior", *Expert Systems with Applications*, Vol. 77, pp. 256-263. DOI: 10.1016/j.eswa.2017.01.029.

Chen, M-Y., Liao, C-H. and Hsieh, R-P. (2019) "Modeling public mood and emotion: Stock market trend prediction with anticipatory computing approach", *Computers in Human Behavior*, Vol. 101, pp. 402-408. DOI: 10.1016/j.chb.2019.03.021.

Dang, M. and Duong, D. (2016) "Improvement methods for stock market prediction using financial news articles", *Proceedings of the 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS'16)*, September 14-16, Danang, Vietnam, pp. 125-129. DOI: 10.1109/NICS.2016.7725636.

Du, Y. (2018) "Application and analysis of forecasting stock price index based on combination of ARIMA model and BP neural network", *Proceedings of the IEEE Chinese Control and Decision Conference (CCDC'18)*, June 9-1, 2018, Shenyang, China, pp. 2854-2857. DOI: 10.1109/CCDC.2018.8407611.

Dutta, A., Bandyopadhyay, G. and Sengupta, S. (2012) "Prediction of stock performance in Indian stock market using logistic regression", *International Journal of Business and Information*, Vol. 7, pp. 105-136.

Galvez, R. H. and Gravano, A. (2017) "Assessing the usefulness of online message board mining in automatic stock prediction systems", *Journal of Computational Science*, Vol. 19, pp. 43-56. DOI: 10.1016/j.jocs.2017.01.001.

Geron, A. (2019) *Hands-on Machine Learning with Scikit-Learn Keras, and Tensorflow*, 2nd Edition, ISBN: 978-1492032649, O'Reilly Media Inc.

Hu, Z., Liu, W., Bian, J., Liu, X. and Liu, T-Y. (2018) "Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction", *Proceedings of the 11th ACM International Conference on*

Web Search and Data Mining (WSDM'18), February 5-9, CA, USA, pp 261-269. DOI: 10.1145/3159652.3159690.

Jaffe, J., Keim, D. B. and Westerfield, R. (1989) "Earnings yields, market values and stock returns", *Journal of Finance*, Vol. 44, pp. 135 - 148. DOI: 10.1111/j.1540-6261.1989.tb02408.x.

Jatnika, D., Bijaksana, M. A. and Suryani, A. A. (2019) "Word2Vec model analysis for semantic similarities in English words", *Procedia Computer Science*, Vol. 157, pp. 160-167. DOI: 10.1016/j.procs.2019.08.153.

Jeong, Y., Kim, S. and Yoon, B. (2018) "An algorithm for supporting decision making in stock investment through opinion mining and machine learning", *Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET)*, August 19-23, Honolulu, HI, USA, pp. 1-10. DOI: 10.23919/PICMET.2018.8481802.

Khandelwal, I., Adhikari, R. and Verma, G. (2015) "Time series forecasting using hybrid ARIMA and ANN models based on DWT decomposition", *Procedia Computer Science*, Vol. 48, pp. 173-179. DOI: 10.1016/j.procs.2015.04.167.

Li. J., Bu, H. and Wu, J. (2017) "Sentiment-aware stock market prediction: A deep learning method", *Proceedings of the IEEE International Conference on Service Systems and Service Management (ICSSSM'17)*, June 16-18, 2017, Dalian, China, pp. 1-6. DOI: 10.1109/ICSSSM.2017.7996306.

Li, X., Xie, H., Chen, L., Wang, J. and Deng, X. (2014) "News impact on stock price return via sentiment analysis", *Knowledge-Based Systems*, Vol. 69, pp. 14-23. DOI: 10.1016/j.knosys.2014.04.022.

Lv, D., Yuan, S., Li, M. and Xiang, Y. (2019) "An empirical study of machine learning algorithms for stock daily trading strategy", *Mathematical Problems in Engineering*, Vol. 2019, Article ID: 7816154, pp. 1-30. DOI: 10.1155/2019/7816154.

Medhat, W., Hassan, A. and Korashy, H. (2014) "Sentiment analysis and applications: A survey", *Ain Shams Engineering Journal*, Vol. 5, No. 4, pp. 1093-1113. DOI: 10.1016/j.asej.2014.04.011.

Mehtab, S. and Sen, J. (2020a) "Stock price prediction using convolutional neural networks on a multivariate time series", *Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence (NCMLAI'20)*, February 1-2, 2020, New Delhi, India. DOI: 10.36227/techrxiv.15088734.v1.

Mehtab, S. and Sen, J. (2020b) *Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models*, Technical Report, No: NSHM_KOL_2020_SCA_DS_1. DOI: 10.13140/RG.2.2.14022.22085/2.

Mehtab, S. and Sen, J. (2020c) "Stock price prediction using CNN and LSTM-based deep learning models", *Proceedings of the IEEE International Conference on Decision Aid Sciences and Applications (DASA'20)*, November 8-9, 2020, Sakheer, Bahrain, Bahrain, pp. 447-453. DOI: 10.1109/DASA51403.2020.9317207.

Mehtab, S., Sen, J. and Dasgupta, S. (2020d) "Robust analysis of stock price time series using CNN and LSTM-based deep learning models", *Proceedings of the 4th IEEE International Conference on Electronics, Communication, and Aerospace Technology (ICECA'20)*, November 5-7, 2020, Coimbatore, India, pp. 1481-1486. DOI: 10.1109/ICECA49313.2020.9297652.

Mehtab, S., Sen, J. and Dutta, A. (2021a) "Stock price prediction using machine learning and LSTM-based deep learning models", In: Thampi, S. M., Piramuthu, S., Li, K. C., Beretti, S., Wozniak, M., and Singh, D. (eds), *Machine Learning and Metaheuristics Algorithms and Applications*. SoMMA 2020. Communications in Computer and Information Science, Vol 1366, pp. 88-106, Springer, Singapore. DOI: 10.1007/978-981-16-0419-5_8.

Mehtab, S. and Sen, J. (2021b) "A time series analysis-based stock price prediction using machine learning and deep learning models", *International Journal of Business Forecasting and Marketing Intelligence (IJBFMI)*, Vol 6, No 4, pp. 272-335, Inderscience Publishers. DOI: 10.1504/IJBFMI.2020.115691.

Mehtab, S. and Sen, J. (2022) "Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models", In: Sahoo, J. P., Tripathy, A. K., Mohanty, M., Li, K. C., Nayak, A. K. (eds) *Advances in Distributed Computing and Machine Learning*. Lecture Notes in Networks and Systems, Vol 202, pp. 405-423, Springer, Singapore. DOI: 10.1007/978-981-16-4807-6_39.

Mittal A. and Goel, A. (2012) *Stock Prediction Using Twitter sentiment Analysis*, Technical Report, Stanford University.

Nam, K. and Seong, N. (2019) "Financial news-based stock movement prediction using causality analysis of influence in the Korean stock market", *Decision Support Systems*, Vol. 117, pp. 100-112. DOI: 10.1016/j.dss.2018.11.004.

Nguyen, T. H., Shiri, K. and Velcin, J. (2015) "Sentiment analysis on social media for stock movement prediction", *Expert Systems with Applications*, Vol. 42, No. 24, pp, 9603-9611. DOI: 10.1016/j.eswa.2015.07.052.

Ning, Y., Wah, L. C. and Erdan, L. (2019) "Stock price prediction bases on error correction model and Granger causality test", *Cluster Computing*, Vol. 22, pp. 4849-4858. DOI: 10.1007/s10586-018-2406-6.

Pai, P-F. and Lin, C-S. (2005) "A hybrid ARIMA and support vector machines model in stock price forecasting", *Omega*, pp. 497-505. DOI: 10.1016/j.omega.2004.07.024.

Roy, S. S., Mittal, D., Basu, A. and Abraham, A. (2015) "Stock market-forecasting using LASSO linear regression model", In: Abraham, A., Kromar, P. and Snasel V. (eds), *Afro-European Conference for Industrial Advancement*, *Advances in Intelligent Systems and Computing*, Vol. 334, pp. 371-381, Springer, Cham. DOI: 10.1007/978-3-319-13572-4_31.

Schumaker, R. P. and Chen, H. (2009) "Textual analysis of stock market prediction using breaking financial news: The AZFin text system", *ACM Transactions on Information Systems*, Vol. 27, No. 2, pp. 1-19, Article No: 12. DOI: 10.1145/1462198.1462204.

Sen, J. (2017a) "A time series analysis-based forecasting approach for the Indian realty sector", *International Journal of Applied Economic Studies*, Vol. 5, No. 4, pp. 8 - 27. DOI: 10.36227/techrxiv.16640212.v1.

Sen, J. (2017b) "A robust analysis and forecasting framework for the Indian mid cap sector using time series decomposition", *Journal of Insurance and Financial Management*, Vol. 3, No. 4, pp. 1- 32. DOI: 10.36227/techrxiv.15128901.v1

Sen, J. (2018) "Stock price prediction using machine learning and deep learning frameworks", *Proceedings of the 6th International Conference on Business Analytics and Intelligence*, December 20-22, Bangalore, India.

Sen, J. and Datta Chaudhuri, T. (2016a) "An alternative framework for time series decomposition and forecasting and its relevance for portfolio choice - a comparative study of the Indian consumer durable and small cap sector", *Journal of Economics Library*, Vol 3, No 2, pp. 303 – 326. DOI: 10.1453/jel.v3i2.787.

Sen, J. and Datta Chaudhuri, T. (2016b) "Decomposition of time series data of stock markets and its implications for prediction - an application for the Indian auto sector", *Proceedings of the 2nd National Conference*

on Advances in Business Research and Management Practices (*ABRMP'16*), January 6 -7, Kolkata, India, pp. 15-28. DOI: 10.13140/RG.2.1.3232.0241.

Sen, J. and Datta Chaudhuri, T. (2016c) "A framework for predictive analysis of stock market indices – a study of the Indian auto sector", *Journal of Management Practices*, Vol 2, No 2, pp. 1-20, Calcutta Business School Publication, Kolkata, India. DOI: 10.13140/RG.2.1.2178.3448.

Sen, J. and Datta Chaudhuri, T. (2016d) "An investigation of the structural characteristics of the Indian IT sector and the capital goods sector - an application of the R programming language in time series decomposition and forecasting", *Journal of Insurance and Financial Management*, Vol. 1, No. 4, pp. 68 - 132, 2016. DOI: 10.36227/techrxiv.16640227.v1.

Sen, J. and Datta Chaudhuri, T. (2017a) "A time series analysis-based forecasting framework for the Indian healthcare sector", *Journal of Insurance and Financial Management*, Vol. 3, No. 1, pp. 66 - 94, 2017. DOI: 10.36227/techrxiv.16640221.v1.

Sen, J. and Datta Chaudhuri, T. (2017b) "A predictive analysis of the Indian FMCG sector using time series decomposition-based approach", *Journal of Economics Library*, Vol. 4, No. 2, pp. 206 - 226. DOI: 10.1453/jel.v4i2.1282.

Sen, J. and Datta Chaudhuri, T. (2017c) "A robust predictive model for stock price forecasting", *Proceedings of the 5th International Conference on Business Analytics and Intelligence*, December 11-13, Bangalore, India. DOI: 10.36227/techrxiv.16778611.v1.

Sen, J. and Datta Chaudhuri, T. (2018) "Understanding the sectors of Indian economy for portfolio choice", *International Journal of Business Forecasting and Marketing Intelligence*, Vol. 4, No. 2, pp. 178-222. DOI: 10.1504/IJBFMI.2018.090914.

Sen, J. and Mehtab, S. (2021a) "Accurate stock price forecasting using robust and optimized deep learning models", In *Proceedings of the IEEE International Conference on Intelligent Technologies (CONIT'21)*, June 25-27, Hubballi, India. DOI: 10.1109/CONIT51480.2021.9498565.

Sen, J. and Mehtab, S. (2021b) "Design and analysis of robust deep learning models for stock price prediction", In: Sen, J. (ed) *Machine Learning – Algorithms, Models and Applications*, IntechOpen, London UK, pp. 15 – 46. DOI: 10.5772/intechopen.99982.

Sen, J., Mehtab, S. and Nath, G. (2020) "Stock price prediction using deep learning models", *Lattice: The Machine Learning Journal*, Vol 1, No 3, pp. 34-40. DOI: 10.36227/techrxiv.16640197.v1.

Shah, D., Isah, H. and Zulkernine, F. (2018) "Predicting the effects of news sentiments on the stock market", *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*, December 10-13, Seattle, WA, USA, pp. 4705-4708. DOI: 10.1109/BigData.2018.8621884.

Souza, T. T. P. and Aste, T. (2019) "Predicting future stock market structure by combining social and financial network information", *Physica A: Statistical Mechanics and its Applications*, Vol. 535, 122343. DOI: 10.1016/j.physa.2019.122343.

Wang, L., Ma, F., Liu, J. and Yang, L. (2020) "Forecasting stock price volatility: New evidence from the GARCH-MIDAS model", *International Journal of Forecasting*, Vol. 36, No. 2, pp. 684-694. DOI: 10.1016/j.ijforecast.2019.08.005.

Wang, Y., Huang, M., Zhu, X. and Zhao, L. (2016) "Attention-based LSTM for aspect-level sentiment classification", *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, November 1-5, Austin, TX, USA, pp. 606-615, DOI: 10.18653/v1/D16-1058.

Weng, B., Ahmed, M. A. and Megahed, F. M. (2017) "Stock market one-day ahead movement prediction using disparate data sources", *Expert Systems with Applications*, Vol. 79, pp. 153-163. DOI: 10.1016/j.eswa.2017.02.041.

Wu, J. (2012) *Advances in K-means Clustering*, Springer-Verlag, Berlin Heidelberg, Germany. DOI: 10.1007/978-3-642-29807-3

Wu, Y-P., Wu, K-P. and Lee, H-M. (2012) "Stock trend prediction by sequential chart pattern via k-means and AprioriAll algorithm", *Proceedings of the IEEE Conference on Technologies and Applications of Artificial Intelligence (TAAI'12)*, Nov 16-18, Tainan, pp. 176-181. DOI: 10.1109/TAAI.2012.42.

Xie, Y. and Jiang, H. (2016) "Stock market forecasting based on text mining technology: A support vector machine method", *Journal of Computers*, Vol. 12, No. 6, pp. 500-510. DOI: 10.17706/jcp.12.6.500-510.

Yahoo Finance Website: http://in.finance.yahoo.com.

Yang, B., Gong, Z-J. and Yang, W. (2017) "Stock market index prediction using deep neural network ensemble", *Proceedings of the 336th IEEE Chinese Control Conference (CCC'17)*, July 26-28, Dalian, China, pp. 3882-3887. DOI: 10.23919/ChiCC.2017.8027964.

Yu, Z. and Li, D. (2018) "A AprioriAll sequence mining algorithm based on learner behavior", In Huang, D. S, Jo, K, H. and Zhang, X. L. (eds.) *Intelligent Computing Theories and Applications, ICIC 2018, Lecture Notes in Computer Science*, Vol. 10955, pp. 560-569, Springer, Cham. DOI: 10.1007/978-3-319-95933-7_65.

Zhang, G. P. (2003) "Time series forecasting using a hybrid ARIMA and neural network model", *Neurocomputing*, Vol. 50, pp. 159-175. DOI: 10.1016/S0925-2312(01)00702-0.

Zhang, J., Cui, S., Xu, Y., Li, Q. Lit, T. (2018) "A novel data-driven stock price trend prediction system", *Expert Systems with Applications*, Vol. 97, pp. 60-69. DOI: 10.1016/j.eswa.2017.12.026.

Zhong, X. and Enke, D. (2017) "Forecasting daily stock market return using dimensionality reduction", *Expert Systems with Applications*, Vol. 67, pp. 126-139. DOI: 10.1016/j.eswa.2016.09.027.

CHAPTER 2

MACHINE LEARNING AND DEEP LEARNING IN STOCK PRICE PREDICTION

JAYDIP SEN

Introduction

Prediction of future movement of stock prices has been the subject matter of many research works. On the one hand, we have proponents of the *Efficient Market Hypothesis* who claim that stock prices cannot be predicted. On the other hand, some works have shown that, if correctly modeled, stock prices can be predicted with a fairly reasonable degree of accuracy. The latter has focused on the choice of variables, appropriate functional forms, and techniques of forecasting. In this regard, Sen and Datta Chaudhuri propose a novel approach of stock price forecasting based on a decomposition approach to stock price time series (Sen, 2017a; Sen 2017b; Sen & Datta Chaudhuri, 2016a; Sen & Datta Chaudhuri, 2016b; Sen & Datta Chaudhuri, 2016c; Sen & Datta Chaudhuri, 2016d; Sen & Datta Chaudhuri, 2017a; Sen & Datta Chaudhuri, 2017b; Sen & Datta Chaudhuri, 2017c; Sen & Datta Chaudhuri, 2018).

There is also extensive literature on technical analysis of stock prices where the objective is to identify patterns in stock movements and derive profit from them. Various indicators have been proposed in the literature to guide investors in the stock market in making wise investment decisions. Some of the indicators are Bollinger Band, *moving average convergence divergence* (MACD), *relative strength index* (RSI), *moving average*, *momentum stochastics*, and *meta Sine wave*. There are also patterns like *head and shoulders*, *triangle*, *flag*, *Fibonacci fan*, and *Andrew's Pitchfork*, which are extensively used by traders for gain.

This chapter proposes a granular approach to stock price prediction by combining several statistical and machine learning methods of prediction on technical analysis of stock prices. We present various approaches for short-term stock price movement forecasting using multiple classification

approaches and regression techniques and compare their performance in predicting stock price movement. We believe this approach will provide several helpful information to the investors in the stock market who are particularly interested in short-term investments for profit. The initial version of the work was published earlier (Sen & Datta Chaudhuri, 2017c). In the present work, we have extended the predictive model by including an additional five classification and five regression models, including an advanced deep learning model.

The rest of the chapter is organized as follows. In the section titled *Problem Statement*, we present a clear statement of our problem at hand. The section titled *Related Work* provides a brief review of stock price movement modeling and prediction literature. In the section titled *Methodology*, we present a detailed discussion on the methodology we followed in this work. The section titled *Performance Results* describes the details of all the predictive models built in this work and the results they have produced. A comparative analysis is presented on the performance of the models in this section. Finally, the section titled *Conclusion* concludes the chapter.

Problem Statement

Our work aims to collect stock price data at five minutes intervals from the National Stock Exchange (NSE) of India and develop a robust forecasting framework for stock price movements. We contend that such a granular approach can model the inherent dynamics of stock prices, which can be fine-tuned for immediate forecasting of stock price or stock price movement. Here, we are not addressing the problem of predicting of long-term movement of stock price. Instead, our framework will be more relevant to a trade-oriented framework.

At any point in time in the Indian economy, given the appetite of financial market players, including individuals, domestic institutions, and foreign financial institutions, there is a finite amount of funds deployed in the stock market. This amount discounts the entire macroeconomics of that time. This fund would be distributed among various stocks. Thus, if some stock prices rise in the short run, some other stock prices should be falling. A potential investor in the stock market using our proposed method will be able to predict the price of a stock in the next slot of time, given its past movement pattern.

Related Work

The literature attempting to prove or disprove the *efficient market hypothesis* can be classified into three strands, according to the choice of variables and techniques of estimation and forecasting. The first strand consists of studies using simple regression techniques on cross-sectional data (Basu, 1993; Jaffe et al., 1989; Rosenberg et al., 1985; Fama & French, 1995; Chui & Wei, 1998). The second category of works is based on time series models and techniques to forecast stock returns. These propositions use econometric methods like *autoregressive integrated moving average* (ARIMA), *Granger causality test*, *autoregressive distributed lag* (ARDL), and *quantile regression* (QR) to forecast stock prices (Jarrett & Kyper, 2011; Adebiyi et al., 2014; Mondal et al., 2014; Mishra, 2016). The third strand includes approaches using machine learning tools for the prediction of stock returns (Mostafa, 2010; Dutta et al., 2006; Wu et al., 2008; Siddiqui & Abdullah, 2015; Jaruszewicz & Mandziuk, 2004; Mehtab & Sen, 2020a; Mehtab & Sen, 2020b; Mehtab et al., 2020c; Mehtab & Sen, 2020d; Mehtab & Sen, 2021a; Mehtab & Sen, 2021b; Mehtab & Sen, 2022; Sen, 2018; Sen & Mehtab, 2021a; Sen & Mehtab, 2021b).

Mehtab and Sen demonstrate how machine learning and *long- and short-term memory* (LSTM)-based deep learning networks can be used for accurately forecasting the future NIFTY 50 index values and their future movement patterns (Mehtab & Sen, 2019). The authors use the daily stock prices for three years, from January 2015 to December 2017, to build the predictive models. The accuracy of the models is then evaluated based on their ability to predict the movement patterns of the *close* values of the NIFTY index on a time horizon of one week. For testing, the authors use NIFTY 50 index values from January 2018 to June 2019. To further improve the models' predictive power, the authors introduce a *sentiment analysis* module for analyzing the public sentiments on Twitter on NIFTY 50 stocks. The output of the sentiment analysis module and the past NIFTY 50 index values are fed into the predictive model to build a robust and accurate forecasting framework. The sentiment analysis module uses a *self-organizing fuzzy neural network* (SOFNN) for designing a nonlinear multivariate framework (Qiao & Wang, 2008).

Mehtab and Sen propose another approach to stock price and movement prediction using *convolutional neural networks* (CNN) on a multivariate time series (Mehtab & Sen, 2020a). The predictive model proposed by the authors exploits the learning capability of a CNN in a *walk-forward validation* manner to realize a high level of accuracy in forecasting the future NIFTY index values and their movement patterns. The authors

propose three different CNN architectures that differ in the number of variables used in forecasting, the number of sub-models used in the overall system, and the size of the input data for training the models. The experimental results indicate that the CNN-based univariate forecasting model is highly accurate in predicting the movement of NIFTY index values with a weekly forecast horizon.

Enke et al. propose a stock price prediction model that works in three stages (Enke et al., 2011). In the first stage of the model execution, a multiple regression model is designed to establish a relationship between several predictors with the target variable, i.e., the predicted stock price. In the second phase, a *differential-evolution-based fuzzy clustering model* is designed to augment the predictive model. A fuzzy neural network is applied in the final stage to make the stock price prediction robust and accurate.

Ma and Liu analyze the past stock returns series for several stocks listed on the Shanghai Stock Exchange and design a predictive model to predict future stock returns using the concept of *nonlinear dynamical theory* (Ma & Liu; 2008). The authors find that the stock returns exhibit a nonlinear behavior with the chosen predictor variables in both univariate and multivariate regression models. It is also observed that the predictions made by the multivariate regression models are more accurate than those yielded by their corresponding univariate counterparts. The regression models are further augmented by introducing a *local polynomial prediction method* and a *backpropagation neural network*.

Ivanovski et al. describe a linear regression and a correlation analysis of the daily returns of several stocks and stock indexes at the Macedonian Stock Exchange (MSE). The objective of the work is to explore the statistical significance of the relationship between the ten most capitalized stocks and the Macedonian Stock Exchange Index over ten years (Ivanovski et al., 2016). The results of the study indicate that there is a significantly strong relationship between the stock prices and the stock exchange index values.

Adebiyi et al. present a study that deals with stock price prediction in a short-term time frame using an extensive set of ARIMA models (Adebiyi et al., 2014). The authors use stocks listed in the New York Stock Exchange (NYSE) and the Nigeria Stock Exchange for training and testing the performance of the models. The results show that the best-performing ARIMA model is highly accurate in predicting future stock prices.

Jammalamadaka et al. propose a framework that incorporates financial news and feeds from Twitter to build a *multivariate Bayesian structural time series* (MBSTS) model (Jammalamadaka et al., 2019). The time series

model can capture correlations among the response (i.e., the target) variables of multiple time series by studying the relationships between several contemporaneous predictor variables. The authors compare the performance of their proposed models with other approaches like ARIMA, *recurrent neural networks* (RNNs), and LSTMs. The sentiment-incorporated predictive model proposed by the authors is found to have outperformed the other models in its prediction accuracy.

Jarrett and Kyper present a scheme for analyzing and forecasting stock prices using the ARIMA-based approach (Jarrett & Kyper, 2011). The authors use data from the PACAP-CCER China database. The database is developed by the Pacific-Basin Capital Markets (PACAP) Research Center at the University of Rhode Island, USA, and the SINOFIN Information Service Inc. - an organization affiliated with the China Center for Economic Research (CCER) of Peking University, China. The study reveals two significant observations: (i) simpler models involving fewer parameters are more accurate than models containing a higher number of parameters, and (ii) the daily stock prices exhibit autoregressive property.

Moshiri and Cameron present a *back propagation-based neural network* and a set of econometric models to forecast inflation levels (Moshiri & Cameron, 2010). The collection of econometric models proposed by the authors includes the following components: (i) Box-Jenkins ARIMA model, (ii) *vector autoregression* (VAR) model, and (iii) *Bayesian vector autoregression* (BVAR) model. The forecasting accuracies of the three models are compared with a *hybrid back propagation network* (BPN) proposed by the authors. For testing the proposed models, three different forecasting horizons are used: (i) one month, (ii) two months, and (iii) twelve months. Using the *root mean square error* (RMSE) and the *mean absolute error* (MAE) as the two metrics for model evaluation, it is observed that the performance of the hybrid BPN is superior to the other econometric models.

Vantuch and Zelinka propose an ARIMA-based stock price forecasting model (Vantuch & Zelinka, 2014). The ARIMA model is further fine-tuned using evolutionary algorithms based on *genetic algorithms* (GA) and *particle swarm optimization* (PSO). The model is found to have a very high level of accuracy in forecasting future stock prices. However, its applicability on a very *highly granular time series data* with a *very short forecast horizon* is not addressed by the authors.

Mostafa presents a predictive framework for forecasting stock prices using historical stock price records from the Kuwait Stock Exchange (KSE) from 2001 to 2003 (Mostafa, 2010). The author presents two alternative architectures for predicting the *close* price of the stocks listed in the KSE.

The two possible architectures are (i) *multi-layer perceptron* (MLP) neural networks and (ii) *generalized regression neural networks*. The results elicit two significant observations: (a) the neural network models are instrumental and efficient in forecasting future stock prices, and (ii) the *quasi-Newton training algorithms* are most accurate in forecasting.

Tseng et al. build a robust forecasting framework for predicting future stock prices using several methods (Tseng et al., 2012). The proposition includes the following: (i) *traditional time series decomposition* (TSD) models, (ii) *HoltWinters* (H/W) exponential smoothing with trend and seasonality models, (iii) *Box-Jenkins* (B/J) models using autocorrelation and partial autocorrelation, and (iv) neural network-based models. The authors train the models on the stock price data of 50 randomly chosen stocks from September 1, 1998, to December 31, 2010. For training the models, the authors use 3105 observations based on the *close* prices of the stocks. The testing of the model is carried out on a forecast horizon of 60 trading days. The results show that the forecasting accuracies are higher for B/J, H/W, and the normalized neural network models. The errors associated with the time series decomposition-based models and the non-normalized neural network models are found to be higher.

Xiao et al. propose an ensemble of ARIMA and a feedforward neural network model to design a robust and accurate model for forecasting *financial market volatility* (Xiao et al., 2014). The raw time series of financial data is first decomposed into different components using *maximum overlap discrete wavelet transform* (MODWT). An approximate forecast is made using an ARIMA model, which is further refined at the next level by a feedforward neural network. While the ARIMA model explores the linear behavior in the data, the nonlinear patterns are learned by the neural network component. The performance of the model is evaluated using a forecast horizon of three weeks.

Wang et al. emphasize the importance of social media sentiments and news feed on stock prices and propose a model that analyzes the relationship between stock prices and the news sentiments on the web and social media (Wang et al., 2018). The learning-based model proposed by the authors scrapes through the web to fetch the news related to the stocks and utilizes the information contents in the news to provide the model with an enhanced forecasting capability. The results indicate that the use of news on the web for stock price prediction yields a lower value of the *mean square error* (MSE).

Porshnev et al. propose a stock price forecasting framework that uses data on the psychological states of Twitter users (Porshnev et al., 2013). Using Twitter users' psychological conditions, the authors use a *lexicon-*

based approach that exploits eight essential sentiments or emotions from more than 755 million tweets. The predictive models proposed by the authors are based on *support vector machines* (SVM) and *artificial neural networks* (ANNs). The models are built and tested on the historical DJIA and S&P 500 index values.

Tang and Chen present a hybrid model for stock price prediction that considers the historical stock prices and the news on the social web (Tang & Chen, 2018). The model combines the capabilities of two robust deep learning architectures – (i) RNN-LSTM for learning from sequential time series data and (ii) CNN for extracting features from high-dimensional data. The output of these two networks is used in making a robust prediction for future stock prices. The results show that the hybrid model is very accurate.

Shen et al. propose a scheme based on a *tapped delay neural network* (TDNN) with *adaptive learning* and *pruning* for forecasting a nonlinear time series of stock price (Shen et al., 2007). The TDNN model is trained using a *recursive least square* (RLS) technique that involves a tunable *learning-rate* parameter that enables faster convergence of the network's learning. The trained network is optimized using a *pruning algorithm* that reduces the possibility of a *model overfitting*. The experimental results in a simulated environment clearly show that the pruned model has less complexity, faster execution time, and improved prediction accuracy.

Wu et al. propose an *ensemble model* for stock price prediction using SVMs and *artificial neural networks* (ANNs) (Wu et al., 2008). The forecasting performance of the ensemble model is compared with those of the individual SVM and ANN models. It is observed that the ensemble approach is more accurate in its prediction results than the individual models constituting the ensemble model.

Chen et al. present a model that captures the patterns in a seemingly chaotic stock market behavior using a *local linear wavelet neural network* (LLWNN) technique (Chen et al., 2005). The LLWNN model is optimized using an *estimation of distribution algorithm* (EDA). The model's objective is to predict the stock price for the following trade day based on the *open*, *close*, and *high* values of a stock on a given day. The results show that even in seemingly random fluctuations in the stock price movements, there is an underlying deterministic feature directly enciphered in the stock price data that can be exploited for making an accurate prediction of future stock prices.

Bao et al. introduce a novel deep learning framework for stock price prediction that consists of three components: (i) *wavelet transform* (WT), (ii) *stacked autoencoders* (SAEs), and (iii) *long-and-short-term memory* (LSTM) gates (Bao et al., 2017). The proposed method first decomposes the

time series of the stock price using WT and carries out denoising of the data. The denoised data is then passed on to the SAEs to extract rich features from the denoised data. The extracted features are passed into the LSTM module that finally predicts the future stock prices. The predictive model is found to yield a very high level of accuracy. However, since the model is designed and evaluated on daily stock data, it is not suitable for aiding in any intra-day buy or sell decisions.

The major drawback of the existing propositions in literature for stock price prediction is their inability to predict stock price movement in a short-term interval. The current work attempts to address this shortcoming by exploiting the power of deep neural networks in stock price movement modeling and prediction.

Methodology

In the section titled *Problem Statement*, we mentioned that this work aims to develop a robust forecasting framework for the short-term movement of stock prices. We use the Metastock tool for collecting data on the short-term movement of stock prices (Metastock). We collect the stock price data for two companies listed on the NSE. The two stocks are *Tata Steel* and *Hero MotoCorp*. The stock data are collected every 5 minutes interval in a day. The data collection is done for a period of two years – from January 1, 2013, to December 31, 2014. The raw data for each stock consist of the following variables: (i) *date*, (ii) *time*, (iii) *open* value of the stock, (iv) *high* value of the stock, (v) *low* value of the stock, (vi) *close* value of the stock, and (viii) *volume* of the stock traded in a given time interval. The variable *time* refers to the time instance at which the stock values are noted. Hence, the time interval between two successive records in the raw data is 5 minutes. In addition to the six variables in the raw data, we also collect the NIFTY index at 5 minutes intervals for the same period. NIFTY index values reflect the sentiment in the overall stock market. Therefore, the raw data for both stocks now consist of seven variables. As an interval of 5 minutes is too granular, we make some aggregation of the raw data. We break the total time interval in a day into three slots as follows: (1) a morning slot that covers the time interval from 9:00 AM to 11:30 AM, (2) an afternoon slot that covers the time interval from 11:35 AM to 1:30 PM, and (3) an evening slot that covers the time interval from 1:35 PM to the time of closure of NSE on a given day. Hence, the daily stock information now consists of three records, each record containing stock price information for a time slot.

Using the eight variables in the raw data and incorporating the aggregation of data using the time slots, we derive the following variables that we use later on in our forecasting models. We follow two approaches in forecasting - *regression* and *classification*. The two methods differ in some of the variables they use, which we will describe later in this section.

The following eleven variables are derived and used in the proposed forecasting models:

a) *month*: it refers to the month to which a given record belongs. This variable is coded into numeric data, with "1" referring to January and "12" referring to December. The value of the variable *month* lies in the interval [1, 12].

b) *day_month*: this variable refers to the day of the month to which a given record belongs. It is a numeric variable lying in the interval [1, 31]. For example, the date February 14, 2013, will have a 14 for the variable *day_month*.

c) *day_week*: a numeric variable that refers to the day of the week corresponding to a given stock record. This variable lies in the interval [1, 5]. Monday is coded as 1, while Friday is coded as 5.

d) *time*: a numeric variable that refers to the time slot to which a given record belongs. The morning, afternoon, and evening slots are coded as 1, 2, and 3, respectively. Thus, if a stock record is noted at the time instance 2:45 PM, the value of the *time* variable corresponding to that record will be 3.

e) *open_perc*: a numeric variable computed as a percentage change in the value of the stock's *open* price over two successive time slots. The computation of the variable is done as follows. Suppose we have two consecutive slots: S_1 and S_2. Both of them consist of several records at five minutes intervals of time. Let the *open* price of the stock for the first record of S_1 be X_1 and that for S_2 be X_2. The *open_perc* for the slot S_2 is computed as $(X_2 - X_1)/X_1$ in terms of percentage.

f) *sensex_perc*: a numeric variable computed as a percentage change in the NIFTY index over two successive time slots. The computation of the variable is done as follows. We calculate the mean values of the NIFTY index for two consecutive time slots S_1 and S_2. Let the means values be M_1 and M_2, respectively. Then the *sensex_perc* for the slot S_2 is computed as $(M_2 - M_1)/M_1$ in terms of percentage.

g) *low_diff*: a numeric value computed as the difference between the *low* values of two successive slots. For two consecutive slots S_1 and S_2, first, we calculate the mean of all *low* values of the records in both slots. If L_1 and L_2 refer to the mean values of the *low* for S_1 and S_2, then *low_diff* for S_2 is computed as $(L_2 - L_1)$.

h) *high_diff*: a numeric value computed as the difference between the *high* values of two successive slots. The computation is identical to that of *low_diff* except that *high* values are used in this case.

i) *close_diff*: a numeric value computed as the difference between the *close* values of two successive slots. Its computation is similar to the *open_perc* variable, except for the fact that we use the *close* values in the slots, and we do not compute any percentage. Hence, if two successive slots S_1 and S_2 have *close* values C_1 and C_2, respectively, then *close_diff* for S_2 is calculated as $(C_2 - C_1)$.

j) *vol_diff*: a numeric value computed as the difference between the *volume* values of two successive slots. For two consecutive slots S_1 and S_2, we calculate the mean values of *volume* for both the slots, say V_1 and V_2, respectively. Now, the *vol_diff* for S_2 is computed as $(V_2 - V_1)$.

k) *range_diff*: a numeric value computed as the difference between the *range* values of two successive slots. For two successive slots S_1 and S_2, suppose the *high* and *low* values are H_1, H_2, L_1, and L_2, respectively. Hence, the *range* value for S_1 is $R_1 = (H_1 - L_1)$ and for S_2 is $R_2 = (H_2 - L_2)$. The *range_diff* for the slot S_2 is computed as $(R_2 - R_1)$.

After we compute the values of the above eleven variables for each slot for both the stocks for the time frame of two years (i.e., 2013 and 2014), we develop the forecasting framework. As mentioned earlier, we follow two broad approaches to forecasting stock movements - *regression* and *classification*.

In the regression approach, based on the historical movement of the stock prices, we predict the stock price for the next slot. We use *open_perc* as the response variable, which is a continuous numeric variable. The objective of the regression technique is to predict the *open_perc* value of the next slot given the stock movement pattern and the values of the predictors till the previous slot. In other words, if the current time slot is S_1, the regression techniques will attempt to predict *open_perc* for the next slot, S_2. If the predicted *open_perc* is positive, it will indicate an expected rise in the stock price in S_2, while a negative *open_perc* will indicate a fall in the stock price in the next slot. Based on the predicted values, a potential investor can make his or her investment strategy in stocks.

In the classification approach, the response variable *open_perc* is a discrete variable with two labels. For developing the classification-based forecasting approaches, we transform *open_perc* into a categorical variable with two possible labels, “0” or “1”. The value “0” indicates a negative or a zero *open_perc* value, while “1” indicates a non-zero positive *open_perc* value. Hence, if the current slot is S_1 and the model expects a rise in the *open_perc* value in the next slot, S_2, then the *open_perc* value for S_2 will be

"1". An expected negative value of the *open_perc* in the next slot will be indicated by a "0" for the *response* variable.

For both classification and regression approaches, we consider three cases, which are described below.

Case I: In this case, we use the data for the year 2013. The data consist of 19385 records at an interval of five minutes. These records are aggregated into 745 time-slot records for building the predictive model. We use the same dataset for testing the forecast accuracy of the models for both the stocks and make a comparative analysis of all the models.

Case II: In this case, we use the data for 2014, which consist of 18972 records at five minutes intervals. The granular data are aggregated into 725 time-slot records for building the predictive model. We use the same dataset to test the forecast accuracy of the model for both the stocks and analyze the performance of the predictive models.

Case III: In this case, we use the data of 2013 as the training dataset for building the models and test the models using the data of 2014. We, again, carry out an analysis of the performance of different models in this case.

We use eight approaches to classification and eight approaches to regression for building our forecasting framework. The classification techniques used are: (i) *logistic regression*, (ii) *k-nearest neighbor* (iii) *decision tree*, (iv) *bagging*, (v) *boosting*, (vi) *random forest*, (vii) *artificial neural network*, and (viii) *support vector machines*. We evaluate the performance of the classification models using several metrics such as *sensitivity*, *specificity*, *positive predictive value*, *negative predictive value*, *classification accuracy*, and *F-1 score*.

The nine regression methods that we built are: (i) *multivariate regression*, (ii) *multivariate adaptive regression spline*, (iii) *decision tree*, (iv) *bagging*, (v) *boosting*, (vi) *random forest*, (vii) *artificial neural network*, (viii) *support vector machine*, and (ix) *long- and short-term memory* Network. While all the classification techniques are machine learning-based approaches, the *long- and short-term memory* (LSTM) network-based regression is a deep learning method. For comparing the performance of the regression models, we use several metrics such as *root mean square error* (RMSE) and the *correlation coefficient* between the *actual* and *predicted* values of the response variable, e.g., *open_perc*. In the section titled *Performance Results*, we show that the deep learning regression model far outperforms the machine learning models.

Performance Results

This section provides a detailed discussion on the forecasting techniques we have used and the results obtained using those techniques. We first discuss the classification models and then the regression models.

For both the stocks and all three cases, we compute the prediction accuracy of the classification models using several metrics. We define the metrics below.

Sensitivity: The ratio of the number of *true positive* cases to the total number of *positive* cases in the dataset. It is also referred to as *recall*. Here, *positive* refers to the records belonging to the class "1". The term "*true positives"* refers to the number of positive cases that the model correctly classifies. Sensitivity is often expressed as a percentage.

Specificity: The ratio of the number of *true negative* cases to the total number of *negative* cases in the dataset. Here, *negative* refers to the records belonging to the class "0". The term "*true negatives"* refers to the number of negative cases that the model correctly classifies. Like *sensitivity*, *specificity* is often expressed as a percentage.

Positive Predictive Value (*PPV*): The ratio of the number of *true positive* cases to the total number of *true positive* cases and *false positive* cases. PPV is also sometimes called *precision* and is often expressed as a percentage.

Negative Predictive Value (*NPV*): The ratio of the number of *true negative* cases to the total number of *true negative* cases and the *false negative* cases. NPV is often expressed as a percentage.

Classification Accuracy (*CA*): The ratio of the number of correctly classified cases to the total number of cases in the dataset. It is often expressed as a percentage.

F1-Score: The harmonic mean of *precision* and *recall*, and it is given by (1):

$$F1Score = \frac{2*Precision*Recall}{(Precision+Recall)} \tag{1}$$

The eight classification models that we built are now discussed in detail. We discuss the classification techniques, the regression techniques, and the sentiment analysis-based SOFNN model.

Tables 2-1 to 2-8 depict the performance results of the machine learning-based classification models.

***Logistic Regression Classification*:** This is a classification technique. Therefore, we transform the response variable *open_perc* into a discrete domain from the continuous domain. In other words, we change the

response variable into a *categorical variable* with two possible labels, “0” or “1”. We convert all negative or zero values of *open_perc* to the class “0” and all non-zero positive values to the category “1”. We use the function *glm* in the R language for building the logistic regression model, passing three parameters into the function. The first parameter is the *formula*: *open_perc* ~. to include *open_perc* as the response variable and all the remaining variables as the predictors. The second parameter is "*family = binomial,*" indicating that the model is a *binary logistic regression* with two classes. The third parameter is the R *data object* containing the training dataset. We use the *predict* function in the R language to compute each test records' probability of belonging to the two classes, “0” and “1”. A threshold probability of 0.5 is used. In other words, when the probability of a record belonging to a class equals or exceeds 0.5, we assume that the record belongs to that class. Table 2-1 presents the performance results of the logistic regression models under the three cases.

TABLE 2-1. LOGISTIC REGRESSION CLASSIFICATION RESULTS

Stock	**Case I Training data 2013 Test data 2013**		**Case II Training data 2014 Test data 2014**		**Case III Training data 2013 Test data 2014**	
Tata Steel	Sensitivity	93.00	Sensitivity	94.20	Sensitivity	94.20
	Specificity	97.25	Specificity	98.65	Specificity	93.63
	PPV	92.54	PPV	96.53	PPV	85.52
	NPV	97.43	NPV	97.71	NPV	97.59
	CA	96.11	CA	97.38	CA	93.79
	F1-score	92.97	F1-score	95.35	F1-score	89.65
Hero Moto	Sensitivity	42.86	Sensitivity	55.56	Sensitivity	72.46
	Specificity	94.32	Specificity	91.70	Specificity	81.08
	PPV	70.91	PPV	72.78	PPV	60.48
	NPV	83.62	NPV	83.77	NPV	88.05
	CA	81.74	CA	81.38	CA	78.62
	F1-score	53.43	F1-score	63.01	F1-score	65.93

TABLE 2-2. KNN CLASSIFICATION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Sensitivity	81.50	Sensitivity	72.46	Sensitivity	95.17
	Specificity	97.80	Specificity	97.49	Specificity	57.72
	PPV	93.14	PPV	92.02	PPV	46.45
	NPV	93.51	NPV	89.86	NPV	94.62
	CA	93.42	CA	90.34	CA	67.44
	F1-score	86.93	F1-score	81.08	F1-score	62.43
Hero Moto	Sensitivity	64.29	Sensitivity	60.87	Sensitivity	10.63
	Specificity	94.67	Specificity	93.05	Specificity	94.98
	PPV	79.59	PPV	77.78	PPV	45.83
	NPV	89.13	NPV	85.61	NPV	72.67
	CA	87.25	CA	83.86	CA	70.90
	F1-score	71.13	F1-score	68.29	F1-score	17.26

TABLE 2-3. DECISION TREE (CART) CLASSIFICATION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Sensitivity	96.00	Sensitivity	93.72	Sensitivity	98.07
	Specificity	97.43	Specificity	99.42	Specificity	92.86
	PPV	93.30	PPV	98.48	PPV	84.58
	NPV	98.52	NPV	97.54	NPV	99.18
	CA	97.05	CA	97.79	CA	94.34
	F1-score	94.63	F1-score	96.04	F1-score	90.83
Hero Moto	Sensitivity	40.66	Sensitivity	53.62	Sensitivity	36.23
	Specificity	92.18	Specificity	92.47	Specificity	89.77
	PPV	62.71	PPV	74.00	PPV	58.59
	NPV	82.78	NPV	83.30	NPV	77.89
	CA	79.60	CA	81.38	CA	74.48
	F1-score	49.33	F1-score	62.18	F1-score	44.77

***K- Nearest Neighbor Classification*:** The *k*- nearest neighbor is an example of an *instance-based supervised learning* method used for classification in which the training data is stored. The class for a new unclassified record is found simply by comparing it to the most similar records in the training set. The value of *k* determines how many closest similar records in the training data set are considered for classifying a record of the test data set.

We use the R function *knn* defined in the library *class* to carry out the *k-nearest neighbor* (KNN) classification in the stock price data. The data is normalized using *min-max normalization* before applying the *knn* function so that all predictors are scaled down into the same range of values. Different values of *k* are tried out for building the models, and the value of *k* =3 is finally chosen. This value of *k* is found to produce the best performance of the model with the minimum probability of model overfitting. Table 2-2 provides the performance results of the KNN classification models.

***Decision Tree Classification*:** The *classification and regression tree* (CART) algorithm produces strictly binary decision trees containing exactly two branches at each decision node. CART recursively partitions the records in the training data set into subsets of records with similar values for the target attributes. The CART algorithm constructs the trees by carrying out an exhaustive search on each node for all available variables and all possible splitting values and selects the optimal split based on some *goodness of split* criteria.

We use the *tree* function defined in the *tree* library of the R language for the classification of the stock records. Table 2-3 presents the results produced by the decision tree classification models. For *Case I* and *Case IIII*, the classification trees consist of 9 nodes with *close_diff*, *low_diff*, *high_diff,* and *month* as the predictors. The tree for *Case II* having the same number of nodes finds *close_diff*, *month*, and *low_diff* as the discriminating predictors.

TABLE 2-4. BAGGING CLASSIFICATION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Sensitivity	96.00	Sensitivity	95.17	Sensitivity	98.07
	Specificity	97.43	Specificity	99.23	Specificity	92.86
	PPV	93.20	PPV	98.01	PPV	84.58
	NPV	98.52	NPV	98.09	NPV	99.18
	CA	97.05	CA	98.07	CA	94.34
	F1-score	94.58	F1-score	96.57	F1-score	90.83
Hero Moto	Sensitivity	68.13	Sensitivity	61.35	Sensitivity	41.06
	Specificity	96.80	Specificity	95.95	Specificity	85.33
	PPV	87.32	PPV	85.81	PPV	52.80
	NPV	90.38	NPV	86.14	NPV	78.37
	CA	89.80	CA	86.07	CA	72.69
	F1-score	76.54	F1-score	71.55	F1-score	46.20

TABLE 2-5. BOOSTING (ADABOOST) CLASSIFICATION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Sensitivity	100.00	Sensitivity	100.00	Sensitivity	84.62
	Specificity	100.00	Specificity	100.00	Specificity	98.17
	PPV	100.00	PPV	100.00	PPV	95.65
	NPV	100.00	NPV	100.00	NPV	93.05
	CA	100.00	CA	100.00	CA	93.79
	F1-score	100.00	F1-score	100.00	F1-score	89.80
Hero Moto	Sensitivity	100.00	Sensitivity	100.00	Sensitivity	54.60
	Specificity	100.00	Specificity	100.00	Specificity	79.67
	PPV	100.00	PPV	100.00	PPV	45.89
	NPV	100.00	NPV	100.00	NPV	84.75
	CA	100.00	CA	100.00	CA	73.66
	F1-score	100.00	F1-score	100.00	F1-score	49.87

Bagging Classification: *Bootstrap Aggregation* (Bagging) is an *ensemble technique*. It works as follows: Given a set D, of d tuples, for iteration i, a training set, D_i of d tuples is sampled *with replacement* from the original set of tuples D. Each training set is a bootstrap sample. Since the sampling is done *with replacement*, some tuples in D may not be

included in D_i, while some may be included in more than one sample. A classifier model M_i is learned for each training set, D_i. To classify an unknown tuple X, each classifier, M_i returns its class predictions. The prediction made by each classifier is considered as one vote. The *bagging classifier* counts the votes and assigns the class with the maximum number of votes to X.

We use the *bagging* function defined in the *ipred* library of the R language for carrying out the classification of the stock price data. The value of the parameter *nbag* that specifies the *number of samples* is taken as 25. The results of the performance of the *bagging* classification models are presented in Table 2-2 to 2-4.

***Boosting Classification*:** *Boosting* is an advanced version of the bagging ensemble method. Unlike *bagging*, *boosting* assigns weights to each training tuple. A series of k classifiers is iteratively learned. After a classifier M_i is learned, the weights are updated to build the subsequent classifier M_{i+1}, focusing more closely on the training tuples that M_i misclassified. The final boosted classifier combines the vote of each classifier. The weight of each classifier's vote is a function of its accuracy. *Adaptive Boosting* (AdaBoost) is a very popular variant of boosting for classification.

We use the *boosting* function of the *adabag* library in the R language for the classification of stock price data using the *Adaboost* algorithm. Table 2-5 presents the performance results of the *boosting* classification models.

***Random Forest Classification*:** Random forest is a powerful *ensemble technique* for building predictive models. Each of the classifiers in the ensemble is a *decision tree*. The collection of all the decision tree classifiers is a *forest*. The individual decision trees are created using a random selection of attributes at each node to determine the split. More formally, each tree is constructed based on a simple random sample from a vector of features. During classification, each tree votes, and the most popular class is returned. We use the *randomForest* function defined in the *randomForest* library in the R language for building the random forest classifier. Table 2-6 presents the performance results of the random forest classification model.

TABLE 2-6. RANDOM FOREST CLASSIFICATION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Sensitivity	95.00	Sensitivity	92.75	Sensitivity	96.14
	Specificity	97.06	Specificity	98.46	Specificity	92.86
	PPV	92.23	PPV	96.00	PPV	84.32
	NPV	98.14	NPV	97.14	NPV	98.36
	CA	96.51	CA	96.83	CA	93.79
	F1-score	93.65	F1-score	94.35	F1-score	89.84
Hero Moto	Sensitivity	37.36	Sensitivity	46.93	Sensitivity	72.95
	Specificity	88.45	Specificity	88.03	Specificity	63.51
	PPV	51.13	PPV	60.00	PPV	44.41
	NPV	81.37	NPV	80.00	NPV	85.45
	CA	75.97	CA	75.72	CA	66.21
	F1-score	43.18	F1-score	52.67	F1-score	55.21

TABLE 2-7. ANN CLASSIFICATION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Sensitivity	95.00	Sensitivity	94.69	Sensitivity	100.00
	Specificity	96.70	Specificity	98.84	Specificity	32.82
	PPV	91.35	PPV	97.03	PPV	37.30
	NPV	98.14	NPV	97.90	NPV	100.00
	CA	96.25	CA	97.66	CA	52.00
	F1-score	93.14	F1-score	95.85	F1-score	54.05
Hero Moto	Sensitivity	63.19	Sensitivity	68.12	Sensitivity	7.73
	Specificity	93.61	Specificity	91.31	Specificity	100
	PPV	76.16	PPV	75.81	PPV	100
	NPV	88.72	NPV	87.76	NPV	73.06
	CA	86.17	CA	83.17	CA	73.66
	F1-score	93.14	F1-score	95.85	F1-score	54.33

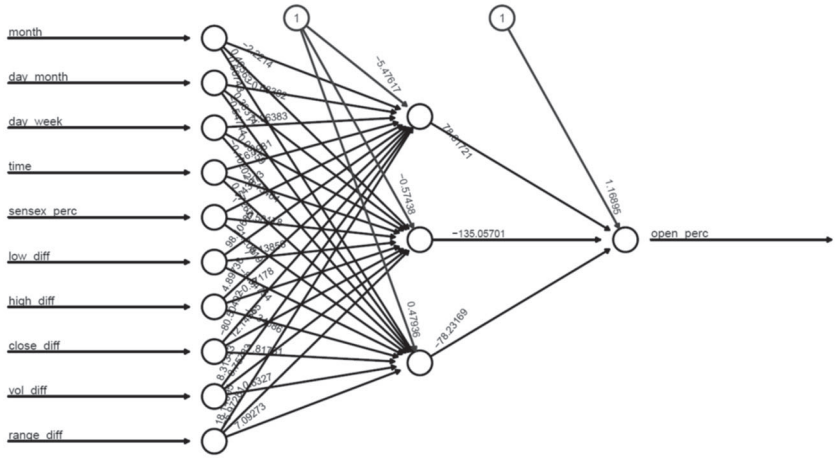

Figure 2-1. The ANN model for classification with three hidden nodes for the Hero Motocorp stock for Case II (Training and testing both on 2014 data)

***Artificial Neural Network Classification*:** An *artificial neural network* (ANN) consists of an input layer, one or more hidden layers, and an output layer. Each layer is made up of units, called *nodes*. The inputs to the network correspond to the attributes measured for each training tuple. The inputs are fed simultaneously into the nodes making up the input layer. These inputs pass through the input layers and are then weighted and fed simultaneously to another layer of neuron units, known as the *hidden layer*. The output of the hidden layer can be input to another hidden layer and so on. The weighted outputs of the last hidden layer are input to the units making up the output layer, which produces the network's prediction for the given tuples.

We use the *neuralnet function* defined in the *neuralnet library* in R for classifying the stock price data. The raw data is normalized using the *min-max normalization* method. Only the predictors are standardized, and the response variable: *open_perc* is kept unchanged. The parameter *hidden* in the function *neuralnet* is changed to alter the number of hidden layers in the network. The parameter *stepmax* is set to the maximum value of 10^6 so that the maximum number of iterations of the *neuralnet function* can be utilized. To carry out the classification task, we set the parameter "*linear.output"* to FALSE in the *neuralnet* function. For the Tata Steel stock and all three models, we need only one node in the hidden layer to obtain the results

presented in Table 2-7. However, for the Hero Motocorp data, we require three nodes in the hidden layer for all three models. Figure 2-1 depicts the architecture of the ANN model for classifying the Hero Motocorp stock data for *Case II*.

TABLE 2-8. SVM CLASSIFICATION RESULTS

Stock	**Case I Training data 2013 Test data 2013**		**Case II Training data 2014 Test data 2014**		**Case III Training data 2013 Test data 2014**	
Tata Steel	Sensitivity	91.26	Sensitivity	96.50	Sensitivity	85.22
	Specificity	97.77	Specificity	97.33	Specificity	97.78
	PPV	94.00	PPV	93.24	PPV	94.69
	NPV	96.70	NPV	98.65	NPV	93.44
	CA	95.97	CA	97.10	CA	93.79
	F1-score	92.61	F1-score	94.84	F1-score	89.71
Hero Moto	Sensitivity	64.71	Sensitivity	72.00	Sensitivity	71.67
	Specificity	78.53	Specificity	78.40	Specificity	75.34
	PPV	18.13	PPV	34.78	PPV	20.77
	NPV	96.80	NPV	94.59	NPV	96.72
	CA	77.58	CA	78.12	CA	75.03
	F1-score	76.65	F1-score	46.90	F1-score	32.21

***Support Vector Machine*:** A *support vector machine* (SVM) is a classification method for linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within the new higher dimension, the algorithm searches for the linear optimal hyperplane that separates the elements of different classes. SVM finds this hyperplane using *support vectors*, which are the essential and the discriminating training tuples to separate the two classes.

We use the *ksvm* function defined in the *kernlab* library in the R language for carrying out the classification of the stock price data. The function *ksvm* has an optional parameter called *kernel,* which is set to *vanilladot* in our implementation. Table 2-8 presents the performance results of the SVM classification models.

We now discuss the design details of the regression models and their performance results.

***Multivariate Regression*:** In this regression approach, we use *open_perc* as the response variable and the remaining ten variables as the predictors to build predictive models for the three cases. We use the R

programming language for data management, model construction, model testing, and visualization of results in all these cases.

Case I: We use 2013 data as the training dataset for building the model. The model is tested using the same dataset. For both the stocks, we use two methods of multivariate regression - (i) *backward deletion* and (ii) *forward addition* of variables. Both the approaches yield the same results.

For the Tata Steel dataset for the year 2013, we apply the *vif* function in the *farway* library to detect the collinear predictor variables to eliminate the *multicollinearity* problem. It is observed that the variables *low_diff*, *high_diff,* and *close_diff* exhibit multicollinearity. We retain the variable *close_diff* and leave out the other two variables. Using the *drop1 function* in the *backward deletion technique* and *add1* function in the *forward addition technique*, we identify the variables that are not significant in the model and do not contribute to its information content. For identifying the variables that contribute the least to the information contained in the model at each iteration, we use the *Akaike Information Criteria* (AIC). In the *backward deletion* approach, the variable with the least AIC value and a *p*-value that is not significant is removed from the model after each iteration. On the other hand, in the *forward addition* approach, the variable with the lowest AIC and a significant *p*-value is added at each iteration. It is found that *close_diff* and *sensex_perc* are the two predictors finally retained in the model.

For Hero MotoCorp for the year 2013, the variables *sensex_perc*, *close_diff,* and *vol_diff* are found to be significant.

Case II: In this case, the predictors *close_diff*, *sensex_perc*, and *day_week* are significant in both the backward deletion and the forward addition method for the Tata Steel stock. However, for the Hero MotoCorp stock, the significant variables are *sensex_perc*, *low_diff,* and *vol_diff*.

Case III: In this case, the model is identical to that in *Case I*. However, since the test dataset is different in this case, we find different values for the correlation coefficient, RMSE/Mean, and MAPE.

For testing the prediction accuracy of the models, we use the *predict* function for forecasting the values of the response variables. We compute the *correlation coefficient* between the *actual open_perc* values and the *predicted open_perc* values. We also calculate the RMSE between the *actual* and the *predicted* values. Using the RMSE values, we derive the percentage of the RMSE to the mean of the absolute values of the *actual open_perc* values. The MAPE values are also evaluated. Table 2-9 presents the results of the performance of the *multivariate regression* method.

TABLE 2-9. MULTIVARIATE REGRESSION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Correlation	0.97	Correlation	0.98	Correlation	0.98
	RMSE/Mean	32.40	RMSE/Mean	24.01	RMSE/Mean	24.08
	MAPE	18.6	MAPE	17.6	MAPE	27.6
Hero Moto	Correlation	0.56	Correlation	0.52	Correlation	0.53
	RMSE/Mean	116.23	RMSE/Mean	116.03	RMSE/Mean	116.03
	MAPE	139.65	MAPE	138.17	MAPE	145.37

***Multivariate Adaptive Regression Spline (MARS)*:** MARS is an adaptive statistical machine learning-based approach for building robust regression models. The algorithm works by splitting the input variables into multiple *basis functions* and then fitting a linear regression model to those basis functions (Lesmeister, 2017). We use the *earth* function defined in the *earth* library in R for building the MARS regression models. For the Tata Steel stock data, under *Case I*, only three predictors, viz., *close_diff*, *month*, and *low_diff*, are found to be used in the MARS model. Under *Case II*, the predictors that are used in the MARS model for the Tata Steel stock data are: *close_diff*, *low_diff*, and *sensex_perc*. For *Case III*, the predictors used in the MARS model for the Tata Steel stock are: *close_diff*, *low_diff*, and *sensex_perc*. For the Hero Motocorp stock, under *Case I*, the predictors that the MARS model uses are: *high_diff*, *close_diff*, *sensex_perc*, *range_diff*, *time*, and *vol_diff*. For *Case II*, the predictors used for the Hero Motocorp data in the MARS model are: *high_diff*, *close_diff*, *range_diff*, *sensex_perc*, and *low_diff*. The predictors used in the MARS model for *Case III* for the Hero Motocorp data are: *high_diff*, *close_diff*, *sensex_perc*, *range_diff*, *time*, and *vol_diff*. Table 2-10 presents the results of the performance results of the MARS regression models.

TABLE 2-10. MARS REGRESSION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Correlation	0.98	Correlation	0.99	Correlation	0.94
	RMSE/Mean	30.30	RMSE/Mean	22.75	RMSE/Mean	28.74
	MAPE	20.16	MAPE	12.15	MAPE	16.28
Hero Moto	Correlation	0.71	Correlation	0.72	Correlation	0.51
	RMSE/Mean	97.94	RMSE/Mean	95.00	RMSE/Mean	128.82
	MAPE	87.47	MAPE	41.08	MAPE	113.25

TABLE 2-11. DECISION TREE REGRESSION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Correlation	0.96	Correlation	0.97	Correlation	0.04
	RMSE/Mean	40.13	RMSE/Mean	33.65	RMSE/Mean	198.04
	MAPE	38.27	MAPE	32.92	MAPE	180.20
Hero Moto	Correlation	0.98	Correlation	0.97	Correlation	0.02
	RMSE/Mean	30.92	RMSE/Mean	34.88	RMSE/Mean	195.17
	MAPE	28.14	MAPE	32.16	MAPE	170.16

Decision Tree Regression: We use the *tree* function in the *tree* library in R for constructing the regression models as we did in building the decision tree classification models. However, in this case, the response variable is numeric and not categorical. The *predict* function is used to predict the values of the response variable. For both the stocks, *close_diff*, *high_diff* and *low_diff* are the predictors that are found to be significant for all three cases. The functions *cor*, *rmse,* and *mape* defined in the library *Metrics* are used to compute the *correlation coefficient*, the RMSE value, and the MAPE value for determining the prediction accuracy of the models. Table 2-11 presents the performance results of the decision tree regression models. Figure 2-2 depicts the decision tree model for the Tata Steel stock price dataset in *Case III*.

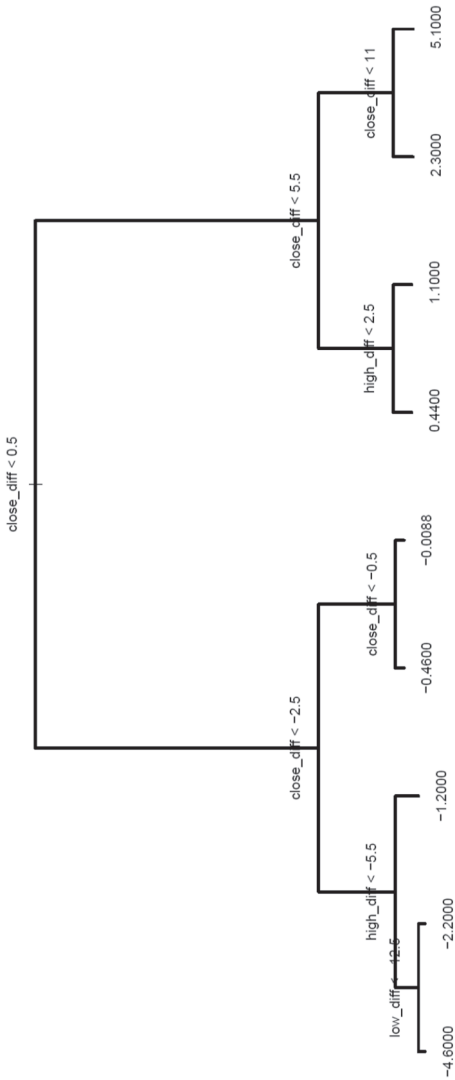

Figure 2-2. The Decision Tree regression model for the Tata Steel stock price time series for *Case III*
(Training using 2013 data and testing on 2014 data)

TABLE 2-12. BAGGING REGRESSION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Correlation	0.97	Correlation	0.98	Correlation	0.96
	RMSE/Mean	34.22	RMSE/Mean	28.54	RMSE/Mean	65.24
	MAPE	31.67	MAPE	26.23	MAPE	62.72
Hero Moto	Correlation	0.70	Correlation	0.69	Correlation	0.51
	RMSE/Mean	101.03	RMSE/Mean	100.85	RMSE/Mean	119.76
	MAPE	98.42	MAPE	97.33	MAPE	116.54

Bagging Regression: For building the bagging regression models on the stock price data, we use the *bagging* function defined in the *ipred* library of R. The value of the parameter *nbag* that specifies the number of samples is taken to be 100. We use the *predict* function in the *ipred* library to predict the response variable values and the *rmse and the mape* functions in the *Metric* library to compute the RMSE and the MAPE values of the models. The *cor* function in the R language is used to calculate the correlation coefficient between the actual and the predicted values of the response variable. Table 2-12 presents the performance results of the *bagging* regression model.

TABLE 2-13. BOOSTING REGRESSION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Correlation	0.98	Correlation	0.99	Correlation	0.77
	RMSE/Mean	34.65	RMSE/Mean	22.39	RMSE/Mean	106.23
	MAPE	31.67	MAPE	21.23	MAPE	103.72
Hero Moto	Correlation	0.69	Correlation	0.69	Correlation	0.51
	RMSE/Mean	102.12	RMSE/Mean	99.67	RMSE/Mean	119.88
	MAPE	97.58	MAPE	96.27	MAPE	111.63

Boosting Regression: We use the *blackboost* function defined in the *mboost* library in R for building the boosting regression models on the stock price data. As in other regression models, the *predict* function is used to compute the predicted values of the response variable. The functions *rmse* and *mape* defined in the *Metrics* library are used to calculate the RMSE and the MAPE values of the model. Table 2-13 presents the performance results of the boosting regression models.

TABLE 2-14. RANDOM FOREST REGRESSION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Correlation	0.99	Correlation	0.99	Correlation	0.97
	RMSE/Mean	15.23	RMSE/Mean	12.88	RMSE/Mean	59.77
	MAPE	12.32	MAPE	10.72	MAPE	45.17
Hero Moto	Correlation	0.95	Correlation	0.95	Correlation	0.51
	RMSE/Mean	53.09	RMSE/Mean	53.67	RMSE/Mean	119.60
	MAPE	51.43	MAPE	50.25	MAPE	108.53

Random Forest Regression: We use the *randomForest* function defined in the *randomForest* library in R for building the random forest regression models. The response variable *open_perc* is a numeric variable. We do not transform the response variable into a *factor* (i.e., a categorical) variable as it was done in the *random forest classification* model. The same *predict*, *rmse*, and *mape* functions are used for computing the predicted values, the RMSE values, and the MAPE values, respectively, for the random forest regression models. Table 2-14 presents the performance results of the random forest regression models.

TABLE 2-15. ANN REGRESSION RESULTS

Stock	Case I Training data 2013 Test data 2013		Case II Training data 2014 Test data 2014		Case III Training data 2013 Test data 2014	
Tata Steel	Correlation	0.97	Correlation	0.98	Correlation	0.98
	RMSE/Mean	31.70	RMSE/Mean	24.23	RMSE/Mean	72.91
	MAPE	28.75	MAPE	21.62	MAPE	65.36
Hero Moto	Correlation	0.73	Correlation	0.73	Correlation	0.42
	RMSE/Mean	95.38	RMSE/Mean	93.01	RMSE/Mean	359.91
	MAPE	87.32	MAPE	85.21	MAPE	320.17

Artificial Neural Network* (ANN) *Regression: We use the *neuralnet* function defined in the *neuralnet* library in the R programming language for building the ANN regression models. The predictors are *normalized* using *min-max normalization* before building the models. The *compute* function defined in the *neuralnet* library is used for computing the predicted values. The parameter *hidden* is used to change the number of nodes in the hidden layer. The value of the parameter *stepmax* is set to 10^6 to exploit the maximum number of iterations executed by the *neuralnet* function. The parameter *linear.output* is by default set to TRUE, and hence it is not altered. For the Tata Steel stock data, we need one node in the hidden layer for all three cases. However, for the Hero Motocorp data, two nodes are

required for the hidden layer for *Case I* and *Case III* and three nodes for *Case II*. Table 2-15 presents the results of ANN regression.

TABLE 2-16. SVM REGRESSION RESULTS

Stock	**Case I Training data 2013 Test data 2013**		**Case II Training data 2014 Test data 2014**		**Case III Training data 2013 Test data 2014**	
Tata Steel	Correlation	0.96	Correlation	0.97	Correlation	0.88
	RMSE/Mean	39.61	RMSE/Mean	31.51	RMSE/Mean	71.07
	MAPE	32.64	MAPE	28.15	MAPE	67.72
Hero Moto	Correlation	0.71	Correlation	0.70	Correlation	0.52
	RMSE/Mean	101.75	RMSE/Mean	98.95	RMSE/Mean	117.14
	MAPE	96.17	MAPE	95.23	MAPE	112.14

***Support Vector Machine* (SVM) *Regression*:** For building the regression models using SVM, we use the *svm* function defined in the *e1071* library in the R programming language. The *predict* function is used for predicting the response variable values using the regression model. The *rmse* and the *mape* functions are used to compute the RMSE and the MAPE values of the model. Table 2-16 presents the performance results of the SVM regression models.

Long-and-Short-Term Memory* (LSTM) *Network Regression: Finally, we apply a deep learning technique of regression using LSTM networks. First, we provide a brief discussion on the working principles of LSTM networks. LSTM is a variant of *Recurrent Neural Networks* (RNNs) - neural networks with feedback loops (Raschka & Mirjalili, 2019; Geron, 2019). In such networks, the output at the current time slot depends on the current inputs and the previous state of the network. However, RNNs suffer from a problem in which these networks cannot capture the long-term dependencies due to *vanishing* or *exploding gradients* during the backpropagation of the errors (Raschka & Mirjalili, 2019; Geron, 2019). LSTM networks overcome such problems, and hence such networks are pretty helpful in forecasting multivariate time series. LSTM networks consist of memory cells that can maintain their states over time using memory and gating units that regulate the information flow into and out of the memory. There are different variants of gates. The *forget gates* control what information to throw away from memory. The *input gates* are meant to control the new information added to the cell state from the current input. The *cell state vector* aggregates the two components - the old memory from the *forget gate* and the new memory from the *input gate*. The *output gates* conditionally decide what to output from the memory cells. The architecture of an LSTM network and the *backpropagation through time* (BPTT)

algorithm for learning provides such networks a compelling ability to learn and forecast in a multivariate time series framework.

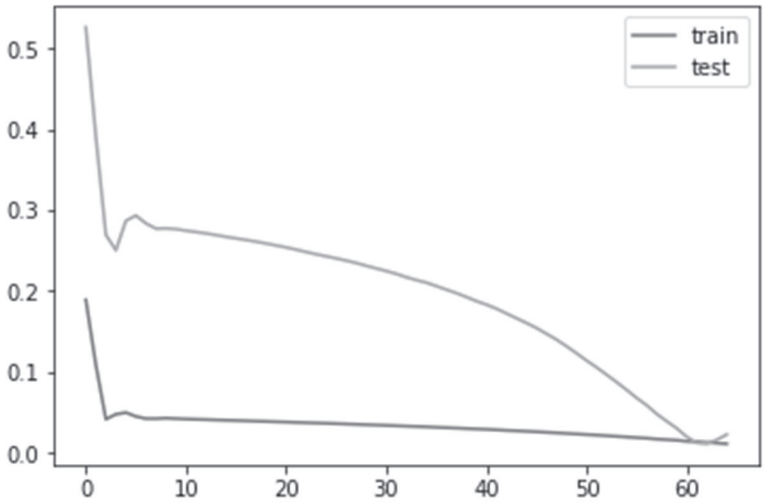

Figure 2-3. The loss convergence in the LSTM model for the Tata Steel stock for *Case III*
(*x*-axis: No of epoch, *y*-axis: Percentage of loss)

We use Python programming language and the Tensorflow deep learning framework for implementing the LSTM networks and utilize those networks to predict the prices of Tata Steel and Hero Motocorp in a multivariate time series framework. For this purpose, we use the *close* price of the stocks as the response variable, and the predictors chosen are - *open*, *high*, *low*, *volume,* and NIFTY index values. However, unlike machine learning techniques, we do not compute the differences between the successive slots. Rather, we forecast the *open* value of the next slot based on the predictor values in the previous slots. For both the stocks and all three cases, we use *Mean Absolute Error* (MAE) as the loss function and *Adam* as the optimizer. However, we train the network with different *epoch* values and *batch sizes* for different cases and different stocks to obtain the optimum performance of the network. The *Sequential* constructor in the Tensorflow framework is used to build the LSTM models. Figure 2-3 shows how the loss converges for both the training and test data sets with different epoch values in *Case III* for the Tata Steel stock.

For the Tata Steel stock, we use 70 epochs with a batch size of 72 for *Case I* and *Case II*. For *Case I*, after epoch 63, the loss goes down to a value low as 0.0106. However, after epoch 70, the loss increases to 0.0533. For *Case II*, the loss converges to 0.0114 after epoch 65 and then slowly increases to 0.0388 at the end of epoch 70. For *Case III*, 65 epochs with a batch size of 72 are used. The loss converges to 0.0114 at the end of epoch 63, which gradually increases to 0.0228 after epoch 70.

In the case of the Hero Motocorp stock data, we use 65 epochs with a batch size of 72 for *Case I* and *Case II*. After the final epoch, for *Case I* and *Case II*, the loss values are found to be 0.0286 and 0.0097, respectively. For *Case III*, 55 epochs with batch size 72 are used. The loss consistently decreases over the successive epochs, finally converging to a value of 0.0135 at the end. Table 2-17 presents the performance results of the LSTM regression models on the two stock price data sets.

TABLE 2-17. LSTM REGRESSION RESULTS

Stock	**Case I Training data 2013 Test data 2013**		**Case II Training data 2014 Test data 2014**		**Case III Training data 2013 Test data 2014**	
Tata Steel	Correlation	1.00	Correlation	1.00	Correlation	1.00
	RMSE/Mean	7.94	RMSE/Mean	4.04	RMSE/Mean	2.36
	MAPE	5.12	MAPE	2.12	MAPE	1.74
Hero Moto	Correlation	0.71	Correlation	0.70	Correlation	0.52
	RMSE/Mean	0.70	RMSE/Mean	0.20	RMSE/Mean	0.28
	MAPE	0.52	MAPE	0.18	MAPE	0.21

TABLE 2-18. THE BEST PERFORMING CLASSIFICATION MODEL UNDER DIFFERENT CASES

Stock	**Case I** **Training data 2013** **Test data 2013**		**Case II** **Training data 2014** **Test data 2014**		**Case III** **Training data 2013** **Test data 2014**	
Tata Steel	Sensitivity	AdaBoost	Sensitivity	AdaBoost	Sensitivity	ANN
	Specificity	AdaBoost	Specificity	AdaBoost	Specificity	AdaBoost
	PPV	AdaBoost	PPV	AdaBoost	PPV	AdaBoost
	NPV	AdaBoost	NPV	AdaBoost	NPV	ANN
	CA	AdaBoost	CA	AdaBoost	CA	BAG
	F1-score	AdaBoost	F1-score	AdaBoost	F1-score	BAG, DT
Hero Moto	Sensitivity	AdaBoost	Sensitivity	AdaBoost	Sensitivity	RF
	Specificity	AdaBoost	Specificity	AdaBoost	Specificity	ANN
	PPV	AdaBoost	PPV	AdaBoost	PPV	ANN
	NPV	AdaBoost	NPV	AdaBoost	NPV	SVM
	CA	AdaBoost	CA	AdaBoost	CA	LR
	F1-score	AdaBoost	F1-score	AdaBoost	F1-score	LR

TABLE 2-19. THE BEST PERFORMING REGRESSION MODEL UNDER DIFFERENT CASES

Stock	**Case I** **Training data 2013** **Test data 2013**		**Case II** **Training data 2014** **Test data 2014**		**Case III** **Training data 2013** **Test data 2014**	
Tata Steel	Correlation	LSTM RF	Correlation	LSTM RF, MARS, Boost	Correlation	LSTM ANN, MV
	RMSE/Mean	LSTM RF	RMSE/Mean	LSTM RF	RMSE/Mean	LSTM MV, MARS
	MAPE	LSTM MARS	MAPE	LSTM RF	MAPE	LSTM MARS
Hero Moto	Correlation	LSTM DT	Correlation	LSTM DT	Correlation	LSTM SVM
	RMSE/Mean	LSTM DT	RMSE/Mean	LSTM DT	RMSE/Mean	LSTM MV
	MAPE	LSTM DT	MAPE	LSTM DT	MAPE	LSTM Boost

Table 2-18 presents the summary of the performance results of the classification models. Among the classification techniques for *Case I* and *Case II*, boosting, quite expectedly, has performed the best on all metrics. These two cases exhibit the model's performance on the training data, and boosting has yielded the best training accuracies. However, *Case III* shows the performance of the models on the test dataset. Bagging and logistic regression have performed the best on the metric 'classification accuracy' for the Tata Steel and the Hero Motocorp stock data. ANN has performed the best on two metrics for both the stocks - sensitivity and NPV for the Tata Steel data and specificity and PPV for the Hero Motocorp data. Since the Hero Motocorp stock data are very imbalanced for both the years 2013 and 2014, the technique that has yielded the highest value of sensitivity for this data - random forest - can also be assumed to have performed very well. Table 2-18 identifies the best-performing classification model under the three cases.

Table 2-19 presents the summary of the performance results of the regression models. For the regression techniques, we find that the deep learning-based LSTM model outperforms all the machine learning models on all metrics for both stocks. In Table 2-19, we also list the best-performing machine learning model for the stocks based on the three metrics. For both *Case I* and *Case II*, the random forest regression model has performed the best for the Tata Steel stock. The MARS and the boosting models have also performed well for the Tata Steel stock for *Case I* and *Case II*. For the Tata Steel stock, in *Case III*, ANN, multivariate regression, and MARS have yielded the best results.

For the Hero Motocorp stock, the decision tree regression model has performed the best among all the machine learning models in *Case I* and *Case II*. On the other hand, in *Case III*, the boosting regression model has yielded the most accurate results.

Conclusion

This chapter proposed a framework for stock price movement prediction in the short-term period using eight classification and nine regression models. These models are based on machine learning and deep learning approaches. We built, fine-tuned, and tested these models using the data of two stocks listed on the National Stock Exchange (NSE) of India. The two stocks that had been studied were Tata Steel and Hero MotoCorp. The stock price data were collected for five minutes using the Metastock tool over two years - January 1, 2013, to December 31, 2014. The raw data were cleaned, the appropriate variable transformation was done, and the

predictors and the response variable were identified for building the classification and regression-based predictive models. In addition to building eight classification and eight regression models using the machine learning approach, we also constructed one deep learning-based regression model using *long- and short-term memory* (LSTM) network. Extensive results have been presented on the performance of these eight classifications and nine regression models. It has been observed that while among the classification techniques, *artificial neural networks* (ANNs) have produced the best results, on average, LSTM outperformed all other regression models by a large margin. The results have given us significant insight into stock price forecasting. Since regression using LSTM provided us with a very accurate prediction of stock price movement in a short interval of time, this approach can be used as a single solution to the problem of stock price prediction in a short-term interval on high-frequency data. Accordingly, machine learning-based classification and regression methods for predicting future stock price values and their movement patterns lose their relevance and applications in this use case scenario.

References

Adebiyi, A. A., Adewumi, A. O. and Ayo, C. K. (2014), "Stock price prediction using the ARIMA model", *Proceedings of the International Conference on Computer Modelling and Simulation (UKSim-AMSS)*, March 26-28, Cambridge, UK, pp. 105 -111. DOI: 10.1109/UKSim.2014.67.

Bao, W., Yue, J. and Rao, Y. (2017) "A deep learning framework for financial time series using stacked autoencoders and long-and-short-term memory", *PLOSE ONE*, Vol. 12, No. 7. DOI: 10.1371/journal.pone.0180944.

Basu, S. (1993) "The relationship between earnings yield, market value and return for NYSE common stocks: Further evidence", *Journal of Economics*, Vol. 12, No. 1, pp. 129-156. DOI: 10.1016/0304- 405X(83)90031-4.

Chen, Y., Dong, X. and Zhao, Y. (2005) "Stock index modeling using EDA based local linear wavelet neural network", *Proceedings of the IEE International Conference on Neural Networks and Brain*, October 13-15, Beijing, China. DOI: 10.1109/ICNNB.2005.1614946.

Chui, A. C. W. and Wei, K. C. J. (1998) "Book-to-market, firm size, and the turn of the year effect: Evidence from Pacific basin emerging markets", *Pacific Basin Finance Journal*, Vol. 6, No. 3- 4, pp. 275 - 293. DOI: 10.1016/50927-538X(98)00013-4.

Dutta, G., Jha, P., Laha, A. and Mohan, N. (2006) "Artificial neural network models for forecasting stock price index in the Bombay Stock Exchange", *Journal of Emerging Market Finance*, Vol. 5, No. 3, pp 283 - 295. DOI: 10.1177/097265270600500305.

Enke, D., Grauer, M. and Mehdiyev, N. (2011) "Stock market prediction with multiple regression, fuzzy type-2 clustering, and neural networks", *Procedia Computer Science*, Vol. 6, pp. 201-206. DOI: 10.1016/j.procs.2011.08.038.

Fama, E. F. and French, K. R. (1995) "Size and book-to-market factors in earnings and returns", *Journal of Finance*, Vol. 50, No. 1, pp. 131-155. DOI: 10.1111/j.1540-6261.1995.tb05169.x.

Geron, A. (2019) *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd Edition, ISBN: 978-1492032649, O'Reilly Media Inc.

Ivanovski, Z., Ivanovska, N. and Narasanov, Z. (2016) "The regression analysis of stock returns at MSE", *Journal of Modern Accounting and Auditing*, Vol. 12, No. 4, pp. 217-224. DOI: 10.17265/1548-6583/2016.04.003.

Jaffe, J., Keim, D. B. and Westerfield, R. (1989) "Earnings yields, market values and stock returns", *Journal of Finance*, Vol. 44, No. 1, pp. 135 - 148. DOI: 10.1111/j.1540-6261.1989.tb02408.x.

Jammalamadaka, S. R., Qui, J. and Ning, N. (2019) "Predicting a stock portfolio with multivariate Bayesian structural time series model: Do news or emotions matter?", *International Journal of Artificial Intelligence*, Vol. 17, No. 2, pp. 81-104.

Jarrett, J. E. and Kyper, E. (2011) "ARIMA modeling with intervention to forecast and analyze Chinese stock prices", *International Journal of Engineering Business Management*, Vol. 3, No. 3, pp. 53 - 58. DOI: 10.5772/50938.

Jaruszewicz, M. and Mandziuk, J. (2004) "One day prediction of NIKKEI index considering information from other stock markets", *Proceedings of the International Conference on Artificial Intelligence and Soft Computing(ICAISC'2004),* Japan, pp. 1130 - 1135. DOI: 10.1007/978-3-540-24844_6_177.

Lesmeister, C. (2017) *Mastering Machine Learning with R*, 2nd Edition, Packt Publication. ISBN: 9781787287471.

Ma, J. and Liu, I. (2008) "Multivariate nonlinear analysis and prediction of Shanghai stock market", *Discrete Dynamic in Nature and Society*, Vol. 2008, Article ID: 526734. DOI: 10.1155/2008/526734.

Mehtab, S. and Sen, J. (2022) "Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models", In: Sahoo,

J. P., Tripathy, A. K., Mohanty, M., Li, K. C., Nayak, A. K. (eds) *Advances in Distributed Computing and Machine Learning*. Lecture Notes in Networks and Systems, Vol 202, pp. 405-423, Springer, Singapore. DOI: 10.1007/978-981-16-4807-6_39.

Mehtab, S. and Sen, J. (2021a) "A time series analysis-based stock price prediction using machine learning and deep learning models", *International Journal of Business Forecasting and Marketing Intelligence (IJBFMI)*, Vol 6, No 2, pp. 272 – 335, Inderscience Publishers. DOI: 10.1504/IJBFMI.2020.115691.

Mehtab, S., Sen, J. and Dutta, A. (2021b) "Stock price prediction using machine learning and LSTM-based deep learning models", In: Thampi, S. M., Piramuthu, S., Li, K. C., Beretti, S., Wozniak, M., and Singh, D. (eds), *Machine Learning and Metaheuristics Algorithms and Applications*. SoMMA 2020. Communications in Computer and Information Science, Vol 1366, pp. 88-106, Springer, Singapore. DOI: 10.1007/978-981-16-0419-5_8.

Mehtab, S. and Sen, J. (2020a) "Stock price prediction using convolutional neural networks on a multivariate time series", *Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence (NCMLAI'20)*, February 1-2, 2020, New Delhi, India. DOI: 10.36227/techrxiv.15088734.v1.

Mehtab, S. and Sen, J. (2020b) *Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models*, Technical Report, No: NSHM_KOL_2020_SCA_DS_1. DOI: 10.13140/RG.2.2.14022.22085/2.

Mehtab, S. and Sen, J. (2020c) "Stock price prediction using CNN and LSTM-based deep learning models", *Proceedings of the IEEE International Conference on Decision Aid Sciences and Applications (DASA'20)*, November 8-9, Sakheer, Bahrain, pp. 447-453. DOI: 10.1109/DASA51403.2020.9317207.

Mehtab, S., Sen, J. and Dasgupta, S. (2020d) "Robust analysis of stock price time series using CNN and LSTM-based deep learning models", *Proceedings of the 4th IEEE International Conference on Electronics, Communication and Aerospace Technology (ICECA'20)*, November 5-7, 2020, Coimbatore, India, pp. 1481-1486. DOI: 10.1109/ICECA49313.2020.9297652.

Mehtab, S. and Sen, J. (2019) "A robust predictive model for stock price prediction using deep learning and natural language processing", *Proceedings of the 7th International Conference on Business Analytics and Intelligence (BAICONF'19)*, December 5-7, Bangalore, India. DOI: 10.2139/ssrn.3502624.

Metastock Website: http://www.metastock.com.

Mishra, S. (2016) "The quantile regression approach to analysis of dynamic interaction between exchange rate and stock returns in emerging markets: Case of BRIC nations", *IUP Journal of Financial Risk Management*, Vol. 13, No. 1, pp 7 -27.

Mondal, P., Shit, L. and Goswami, S. (2014) "Study of effectiveness of time series modeling (ARMA) in forecasting stock prices", *International Journal of Computer Science, Engineering and Applications*, Vol. 4, pp. 13 - 29. DOI: 10.5121/ijcsea.2014.4202.

Moshiri, S. and Cameron N. (2010) "Neural network versus econometric models in forecasting inflation", *Journal of Forecasting*, Vol. 19, No. 3, pp. 201-217. DOI: 10.1002/(SICI)1099-131X(200004)19:3 <201::AID-FOR753>3.0.CO;2-4.

Mostafa, M. (2010) "Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait", *Expert Systems with Applications*, Vol. 37, No. 9, pp. 6302-6309. DOI: 10.1016/j.eswa.2010.02.091.

Porshnev, A., Redkin, I. and Shevchenko, A. (2013) "Machine learning in prediction of stock market indicators based on historical data from Twitter sentiment analysis", *Proceedings of the IEEE 13th International Conference on Data Mining Workshops*, December 7-10, Dallas, TX, USA. DOI: 10.1109/ICDMW.2013.111.

Qiao. J. and Wang, H. (2008) "A self-organizing fuzzy neural network and its applications to function approximation and forecast modeling", *Neurocomputing*, Vol. 71, Nos. 4-6, pp. 564-569. DOI: 10.1016/j.neucom.2007.07.026

Raschka, S. and Mirjalili, V. (2019) *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-learn, and TensorFlow 2*, 3rd Edition, ISBN-13: 978-1789955750, Packt Publishing Ltd.

Rosenberg, B., Reid, K. and Lanstein, R. (1985) "Persuasive evidence of market inefficiency", *Journal of Portfolio Management*, Vol. 11, No. 3, pp. 9 -17. DOI: 10.3905/jpm.1985.409007.

Sen, J. (2017a) "A time series analysis-based forecasting approach for the Indian realty sector", *International Journal of Applied Economic Studies*, Vol. 5, No. 4, pp 8 - 27. DOI: 10.36227/techrxiv.16640212.v1.

Sen, J. (2017b) "A robust analysis and forecasting framework for the Indian mid cap sector using time series decomposition", *Journal of Insurance and Financial Management*, Vol. 3, No. 4, pp. 1- 32. DOI: 10.36227/techrxiv.15128901.v1.

Sen, J. (2018) "Stock price prediction using machine learning and deep learning frameworks", *Proceedings of the 6th International Conference on Business Analytics and Intelligence (ICBAI'18)*, December 20-22, Bangalore, India.

Sen, J. and Datta Chaudhuri, T. (2016a) "An alternative framework for time series decomposition and forecasting and its relevance for portfolio choice - A comparative study of the Indian consumer durable and small cap sector", *Journal Economics Library*, Vol. 3, No. 2, pp. 303 - 326. DOI: 10.1453/jel.v3i2.787.

Sen, J. and Datta Chaudhuri, T. (2016b) "Decomposition of time series data of stock markets and its implications for prediction: An application for the Indian auto sector", *Proceedings of the 2nd National Conference on Advances in Business Research and Management Practices* (*ABRMP'16*), January 8 – 9, Kolkata, India, pp. 15-28. DOI: 10.13140/RG.2.1.3232.0241.

Sen, J. and Datta Chaudhuri, T. (2016c) "A framework for predictive analysis of stock market indices – a study of the Indian auto sector", *Calcutta Business School (CBS) Journal of Management Practices*, Vol. 2, No, 2, pp. 1-20. DOI: 10.13140/RG.2.1.2178.3448.

Sen, J. and Datta Chaudhuri, T. (2016d) "An investigation of the structural characteristics of the Indian IT sector and the capital goods sector - An application of the R programming language in time series decomposition and forecasting", *Journal of Insurance and Financial Management*, Vol. 1, No. 4, pp. 68 - 132. DOI: 10.36227/techrxiv.16640227.v1.

Sen, J. and Datta Chaudhuri, T. (2017a) "A time series analysis-based forecasting framework for the Indian healthcare sector", *Journal of Insurance and Financial Management*, Vol. 3, No. 1, pp. 66 - 94. DOI: 10.36227/techrxiv.16640221.v1.

Sen, J. and Datta Chaudhuri, T. (2017b) "A predictive analysis of the Indian FMCG sector using time series decomposition-based approach", *Journal of Economics Library*, Vol. 4, No. 2, pp. 206-226. DOI: 10.1453/jel.v4i2.1282.

Sen, J. and Datta Chaudhuri, T. (2017c) "A robust predictive model for stock price forecasting", *Proceedings of the 5th International Conference on Business Analytics and Intelligence (BAICONF)*, December 11-13, Bangalore, India. DOI: 10.36227/techrxiv.16778611.v1.

Sen, J. Sen and Datta Chaudhuri, T. (2018) "Understanding the sectors of Indian economy for portfolio choice", *International Journal of*

Business Forecasting and Marketing Intelligence, Vol. 4, No. 2, pp. 178-222. DOI: 10.1504/IJBFMI.2018.090914.

Sen, J. and Mehtab, S. (2021a) "Accurate stock price forecasting using robust and optimized deep learning models", In: *Proceedings of the IEEE International Conference on Intelligent Technologies (CONIT'21)*, June 25-27, Hubballi, India. DOI: 10.1109/CONIT51480.2021.9498565.

Sen, J. and Mehtab, S. (2021b) "Design and analysis of robust deep learning models for stock price prediction", In: Sen, J. (ed) *Machine Learning – Algorithms, Models and Applications*, IntechOpen, London UK, pp. 15 – 46. DOI: 10.5772/intechopen.99982.

Shen, J., Fan, H. and Chang, S. (2007) "Stock index prediction based on adaptive training and pruning algorithm", In: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (eds.), *Advances in Neural Networks, Lecture Notes in Computer Science*, Springer-Verlag, Vol. 4492, pp. 457-464. DOI: 10.1007/978-3-540-72393-6_55.

Siddiqui, T. A. and Abdullah, Y. (2015) "Developing a nonlinear model to predict stock prices in India: An artificial neural networks approach", *IUP Journal of Applied Finance*, Vol. 21, No. 3, pp. 36-39.

Tang, J. and Chen, X. (2018) "Stock market prediction based on historic prices and news titles", *Proceedings of the International Conference on Machine Learning Technologies (ICMLT'18)*, May 26-28, Jinan, China, pp. 29-34. DOI: 10.1145/3231884.3231887.

Tseng, K-C., Kwon, O. and Tjung, L. C. (2012) "Time series and neural network forecast of daily stock prices", *Investment Management and Financial Innovations*, Vol. 9, No. 1, pp. 32-54.

Vantuch, T. and Zelinka, I. (2014) "Evolutionary based ARIMA models for stock price forecasting", *Proceedings of the Interdisciplinary Symposium on Complex Systems (ISCS'14)*, pp. 239-247. DOI: 10.1007/978-3-319-10759-2_25.

Wang, Z., Ho, S-B. and Lin, Z. (2018) "Stock market prediction analysis by incorporating social and news opinion and sentiment", *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW)*, November 17-20, Singapore, Singapore. DOI: 10.1109/ICDMW.2018.00195.

Wu, Q., Chen, Y. and Liu, Z. Liu. (2008) "Ensemble model of intelligent paradigms for stock market forecasting", *Proceedings of the 1st IEEE International Workshop on Knowledge Discovery and Data Mining (WKDD'08)*, January 23-24, Adelaide, SA, Australia, pp 205 - 208. DOI: 10.1109/WKDD.2008.54.

Xiao, Y., Xiao, J., Liu, J. and Wang, S. (2014) "A multiscale modeling approach incorporating ARIMA and ANNs for financial market volatility forecasting", *Journal of Systems Science and Complexity*, Vol. 27, No. 1, pp. 225-236. DOI: 10.1007/s11424-014-3305-4.

CHAPTER 3

STOCK PRICE PREDICTION USING CONVOLUTIONAL NEURAL NETWORKS

JAYDIP SEN & SIDRA MEHTAB

Introduction

Prediction of future movement of stock prices has been the subject matter of many research works. On the one hand, we have proponents of the *efficient market hypothesis* who claim that stock prices cannot be predicted. On the other hand, some works have shown that, if correctly modeled, stock prices can be predicted with a fairly reasonable degree of accuracy. The latter has focused on the choice of variables, appropriate functional forms, and techniques of forecasting. In this regard, Sen and Datta Chaudhuri propose a novel approach of stock price forecasting based on a time series decomposition approach (Sen & Datta Chaudhuri, 2016a; Sen & Datta Chaudhuri, 2016b; Sen & Datta Chaudhuri, 2016c; Sen & Datta Chaudhuri, 2016d; Sen & Datta Chaudhuri, 2017a; Sen & Datta Chaudhuri, 2017b; Sen & Datta Chaudhuri, 2018; Sen, 2017a; Sen, 2017b). Propositions also exist in the literature for using machine learning and deep learning models for future stock price prediction over a short-term time frame (Sen & Datta Chaudhuri, 2017c; Sen, 2018).

Mehtab and Sen propose a highly accurate forecasting framework that exploits the power of text mining and natural language prepossessing in analyzing the sentiments in the investment market and then utilizes that information in a nonlinear multivariate predictive framework based on *self-organizing fuzzy neural networks* (SOFNN) (Mehtab & Sen, 2019).

There are propositions in the literature on technical analysis of stock prices where the objective is to identify patterns in stock movements and derive profit from them. Several indicators have been proposed for understanding the behavior of stock price such as, Bollinger Band, *moving average convergence divergence* (MACD), *relative strength index* (RSI), *moving average, momentum stochastics*, *meta Sine wave,* etc. Traders

extensively use the patterns like *head and shoulders*, *triangle*, *flag*, *Fibonacci Fan*, and *Andrew's Pitchfork* to make wise investment decisions in the stock market. These approaches provide the user with visual manifestations of the indicators, which help ordinary investors to understand which way stock prices may move.

This chapter proposes several machine learning and deep learning-based predictive models for predicting NIFTY 50 stock price movements in the NSE of India. We use daily stock index values from December 29, 2014, to December 28, 2018, as the training dataset to build the models and apply them to predict stock index movement and the stock index's *close* values with a forecast horizon of one week using the test data. For testing, we use NIFTY 50 stock price data from December 31, 2018, to July 31, 2020. The prediction framework is further augmented by incorporating robust deep learning-based *convolutional neural network* (CNN) models to achieve a high level of accuracy in predicting NIFTY 50 index values for the test dataset. Three approaches have been followed in designing CNN models. In the first approach, we build a univariate *autoregressive* model that predicts the daily index values of the following week. The second approach follows a multivariate method with each of the variables *open*, *low*, *high*, and *close* of the NIFTY index for forecasting the *close* values for the next week. In the third approach, we build a set of multiple sub-models with each of the four variables used to construct a separate CNN for univariate forecasting and combine them into a final composite forecast for the daily *close* values of the next week.

The rest of the chapter is organized as follows. In the section titled *Problem Statement*, we explicitly define the problem at hand. The section titled *Related Work* provides a brief review of the related work on the prediction of stock price movements. In the section titled *Methodology*, we describe our research methodology. Extensive results on the performance of the predictive models are presented in the section titled *Performance Results*. This section also describes all the predictive models built in this work and the results they have produced. Finally, the section titled *Conclusion* concludes the chapter.

Problem Statement

Our work is based on collecting the stock price records of NIFTY 50 from the NSE of India over a reasonably long period of five years and developing a robust forecasting framework for the NIFTY index values. We believe that past movement patterns of the daily NIFTY index values can be learned by powerful machine learning and deep learning-based

approaches. The knowledge of the extracted features can be utilized for predicting the future movement of the NIFTY index values. We choose a prediction horizon of one week and hypothesize that the machine learning-based approaches can be further augmented by powerful deep learning-based models to predict NIFTY index movement with even higher accuracy. To validate our hypothesis, in our past work, we used sentiment analysis on the web to build highly accurate predictive models for forecasting the future NIFTY index values (Mehtab & Sen, 2019). In the current work, we follow four different approaches in building *convolutional neural network* (CNN) models to improve the forecasting power of the predictive framework. Here, we are not addressing forecasting of short-term movement of the stock price for intra-day traders. Instead, our analysis will be more relevant to a medium-term investor interested in the weekly forecast of the NIFTY index values.

At any point in time in the Indian economy, given the appetite of financial market players, including individuals, domestic institutions, and foreign financial institutions, there is a finite amount of funds deployed in the stock market. This amount discounts the entire macroeconomics of that time. This fund would be distributed among various stocks. Thus, if some stock prices are rising, some other stock prices should be falling. Using our proposition, it will be possible for an investor to predict the movement pattern of NIFTY 50 which generally depicts the stock market sentiment in India.

Related Work

The existing propositions in the literature for stock price movement and stock price predictions can be broadly classified into three broad categories based on the choice of variables and approaches and techniques adopted in modeling. The first category includes approaches that use simple regression techniques on cross-sectional data (Enke et al., 2011; Khan et al., 2018; Sharma et al., 2019; Ma & Liu, 2008; Ivanovski et al., 2016). These models don't yield very accurate results since stock price movement is a highly nonlinear process. The propositions in the second category exploit time series models and techniques using econometric tools like *autoregressive integrated moving average* (ARIMA), Granger causality test, *autoregressive distributed lag* (ARDL), and *quantile regression* (QR) to forecast stock prices (Adebiyi et al., 2014; Jarrett & Kyper, 2011; Xiao et al., 2014; Sen & Datta Chaudhuri, 2017b; Selvin et al., 2017). The third strand includes propositions using machine learning, deep learning, and natural language processing for prediction of stock returns (Mehtab & Sen,

2022; Mehtab & Sen, 2021a; Mehtab & Sen, 2021b; Mehtab & Sen, 2020a; Mehtab & Sen, 2020b; Mehtab et al., 2020c; Mehtab & Sen, 2019; Sen, 2018; Sen & Datta Chaudhuri, 2017c; Sen & Mehtab, 2021a; Sen & Mehtab, 2021b; Bao et al., 2017; Gocken et al., 2016; Moreno et al., 2011; Patel et al., 2015).

Adebiyi et al. present the performance of some ARIMA-based forecasting models and ANN models for predicting future stock prices (Adebiyi et al., 2014). It is observed that the ARIMA models and the ANN models yield a very high level of accuracy in stock price prediction. Moreover, the results also reveal that the pattern of the ARIMA forecasting models is directional. The performance of the ANN models is found to be more accurate than the ARIMA models.

Baek and Kim propose a framework named *ModAugNet* for forecasting the stock market index (Baek & Kim, 2015). The framework consists of two broad modules: (i) the first module is meant to prevent any *overfitting* of the core LSTM model of the framework, and (ii) the second module is a *predictive* LSTM model. The authors evaluate the performance of the predictive model using two different stock indexes and stock price data – S&P500 and Koreas Composite stock Price Index 200 (KOSPI200). The results show that *ModAugNet* yields a lower value of error on the test data set than the comparative model, *SingleNet,* which does not include the overfitting prevention LSTM module.

Chou and Nguyen argue that nonlinearity in the time series of stock prices poses the biggest challenge in accurately forecasting their future values (Chou & Nguyen, 2018). The authors propose a novel predictive system to handle nonlinearity. The proposition involves an intelligent predictive system that is based on a sliding-window metaheuristic optimization technique. The authors build the model using the historical stock prices of Taiwan's construction companies and deploy the model to make one-step ahead forecasting of the future values of the stock prices. The model yields highly accurate forecasting and provides a rich user experience with a powerful user interface.

Ding and Qin propose a deep learning-based network capable of producing the forecasted values of the opening price, the lowest price, and the highest price of a stock with a forecast horizon of one day (Ding & Qin, 2020). In other words, with the historical stock price data as the input, the model forecasts the *open*, *high*, and *low* values of the stock price for the next day. The proposition includes an LSTM-based deep RNN model that is called an *associated neural network* model. The performance of the associated neural network model is compared with those of a stand-alone LSTM model and a stand-alone RNN model. It is observed that the

performance of the associated network model is superior to those of the other two models.

Kim and Sayama present a novel method for predicting the future movement patterns of the SP500 index values by constructing a network connecting the SP500 companies with links (Kim & Sayama, 2017). The weights of the links are determined based on the mutual information of 60-min price movements for a pair of companies from the network on 5340 minutes price records. Using this network of companies, the authors show that any change in the links' strength provides a valuable insight into the future price movement patterns of the stocks. The authors design several useful metrics using the strength distributions in the network links and network properties like *centrality*. The results indicate that the predictors of the model exhibit a quadratic relationship with the changes in the SP500 index values.

Lin et al. demonstrate an approach based on *continuous wavelet analysis* to investigate the deep features in the pattern of variations of stock-bond return relationships (Lin et al., 2018). The authors attempt to establish a possible relationship between the *fundamental economic factors* and the *stock market uncertainty*. It is observed that both the short-term and long-term dependencies between the stocks and the bonds vary with time. Moreover, it is also found that while the stock and bond returns have a positive sign sensitivity to the short rate and the slope of the term structure, their sensitivity to the stock market volatility is negative. The fundamental economic factors driving the stock-bond relations are not found to vary across time frequencies.

Mehtab et al. present a gamut of machine learning and deep learning-based models for efficient and accurate stock price prediction (Mehtab & Sen, 2020a; Mehtab & Sen, 2020b, Mehtab et al, 2020c; Mehtab & Sen, 2021a; Mehtab & Sen, 2021b; Mehtab et al., 2021c; Sen & Mehtab, 2021). Using the NIFTY 50 index values of the National Stock Exchange (NSE) of India, from December 29, 2014, to July 31, 2020, the authors build eight regression models using machine learning algorithms. The framework is further augmented with four deep learning models using LSTM architectures. The LSTM models are optimized by tuning their hyperparameters. The results show that LSTM-based regression models with a *walk-forward validation* approach outperform the machine learning-based regression models in forecasting accuracy.

Mehtab and Sen describe a highly robust and reliable predictive framework for stock price prediction by combining the power of natural language processing in machine learning models like regression and classification (Mehtab & Sen, 2019). By analyzing the sentiments in the

social media platforms and utilizing the sentiment-related information in a nonlinear multivariate regression model based on *self-organizing fuzzy neural networks* (SOFNN), the model proposed by the authors achieves a high level of accuracy in the predicted values of the future NIFTY index. For training the predictive models, the authors use the historical NIFTY 50 index values listed in NSE from January 2015 to December 2017, while the models are tested on NIFTY 50 index values from January 2018 to June 2019.

Nam and Seong contend that for designing a robust model for stock price forecasting, it is crucial to consider the stock price movements of a set of relevant companies instead of just analyzing a target firm's historical stock prices (Nam & Seong, 2019). Based on their contention, the authors propose a method that explores the *causal relationship* between a set of *relevant* companies. For finding the causality, the proposed mechanism uses the *transfer entropy* method, and to combine the features of the target firm and the causal firms, *multiple kernel learning* is used. The model is trained and tested on a Korean market dataset. The results reveal that the proposed model outperforms the two traditional predictive algorithms in the prediction accuracy in the out-of-sample data.

Nava et al. present a forecasting framework for stock prices using a multistep-ahead forecasting approach that integrates the method of *empirical mode decomposition* (EMD) and a *support vector regressor* (SVR) (Nava et al., 2018). The authors argue that the accuracy of the SVR model is increased if it is provided with the decomposed input from the EMD method rather than the raw time series input. The performance of the proposed model is compared with an SVR model with the raw time series data as the input. The results reveal that the integrated model based on EMD and SVR is more accurate than the SVR model with the raw time series input.

Ning et al. investigate the relationship between critical macroeconomic variables like interest rate, money supply, exchange rate, and inflation rate with the market return in Hong Kong and Shanghai (Ning et al., 2019). The relationships are tested using various statistical and econometric tests like the *arbitrage pricing theory* (APT), the *vector error correction model* (VECM), and the Granger causality test. The results reveal two exciting observations: while in the Chinese stock market, returns are significant in the long-term, returns are substantial both in the long-term and short-term for the Hong Kong stock market.

Park et al. present a model for future stock price prediction based on a multiple regression method (Park et al., 2010). The model has the capability of estimating the risk associated with a stock price series. However, the

model is based on a too simple framework. The assumptions made in designing the model are not expected to be valid in a real-world time series associated with a high volatility level.

Pinheiro and Dras propose a novel scheme for future stock price prediction that uses a simple LSTM-based network with a character-level embedding of text using the web's financial news as the only predictor in the model (Pinheiro & Dras, 2017). The results reveal that the use of character-level embedding in the text is critical and accurate in predicting stock price compared to more complex models involving technical indicators of stocks and other event extraction methods in addition to the news articles on the web.

Selvin et al. present an approach that is not based on building a model on the time series of a specific stock (Selvin et al., 2017). The authors call their approach a *model-independent* method. Instead of fitting a stock price time series onto a specific model, the approach attempts to extract the data features using deep learning architectures like CNN, RNN, and LSTM. A sliding window approach predicts future stock prices on a short time horizon based on latent feature extraction.

Sharma et al. propose a collection of models to predict future stock prices (Sharma et al., 2019). The proposition includes models built using machine learning algorithms, content examination in news articles, and fundamental stocks analysis. A multivariate regression model is also presented that provides sentiment analysis of the news articles on the web and decomposition of time series of historical stock prices.

Vargas et al. propose a sophisticated deep learning model to detect and analyze intricate patterns and interactions among stock price data (Vargas et al., 2017). The deep learning framework consists of a CNN and an RNN model. The results show that the CNN model is more effective in capturing the semantic meaning from the text inputs to the model. In contrast, the RNN model is superior in understanding the context information and modeling the temporal characteristics for stock price prediction.

Wang et al. argue that *backpropagation neural networks* (BPNNs) tend to operate at a suboptimal level since the gradient descent algorithm optimizing the parameters of the networks can get stuck in a local minimum point (Wang et al., 2015). To overcome this problem, the authors propose a hybrid framework that combines an *adaptive differential evolution* (ADE) algorithm with a BPNN. ADE first searches for the global initial weights for all the links and the thresholds. Then, the BPNN model searches for the optimal weights of the links and the thresholds for the nodes. The model is built and tested on two real-world time series data of historical stock prices.

It is observed that the ADE-BPNN model has a superior forecasting accuracy when compared to a basic BPNN model and an ARIMA model.

Yan et al. propose a hybrid predictive model consisting of a multiple linear regression model and a *backpropagation* (BP) neural network model for predicting future movements of stock prices (Yan et al., 2019). Both the models are built and tested on the historical stock index values of "Gree Electric Appliances" over 220 trading days. The results show that the BP neural network is more accurate in forecasting future stock prices than the multiple regression model.

Zhang discusses various aspects of neural networks in data mining tasks such as design, architectures, and their applications (Zhang, 2008). The author focuses on the prediction of stock prices and their movement patterns. The author also presents a detailed discussion of the evolution of neural network design and its architectural complexities. Three types of networks are discussed elaboratively. These networks are (i) multilayer perceptrons, (ii) Hopfield networks, and (iii) Korhonen's self-organizing maps.

Zhu et al. hypothesize a significant bidirectional nonlinear causality between the stock returns and the trading volumes (Zhu et al., 2008). The authors propose the use of a neural network-based scheme to forecast stock index movement. The model is further augmented by the inclusion of different combinations of indices and trading volumes of the component stock as the inputs. NASDAQ, DJIA, and STI data of stock prices and the volume of transactions are used in training the neural network. The experimental results show that the augmented neural networks with trading volumes lead to an improved forecasting performance under different sizes of the forecast horizons.

The drawback of most of the existing propositions in the literature for stock price prediction is their inability to accurately predict highly dynamic and fast-changing patterns in stock price movement. The current work attempts to address this shortcoming by exploiting the power of CNNs in learning the past behavior of the stock price movements and making a highly accurate forecast for the stock market's future behavior.

Methodology

In the section titled *Problem Statement*, we mentioned that this work aims to develop a predictive framework for forecasting the daily price movement of NIFTY 50. We collect the NIFTY 50 daily data from December 29, 2014, to July 31, 2020, from the Yahoo Finance website (Yahoo Finance). The raw data consists of the following variables: (i) *date*,

(ii) *open* value of the index, (iii) *high* value of the index, (iv) *low* value of the index, (v) *close* value of the index, and (vi) *volume* of the stock traded on a given date.

Using the six variables in the raw data, we derive the following variables that we use later on for building our predictive models. We use two approaches in forecasting - *regression* and *classification*. The two methods involved a difference in using some of the variables, which we will describe later in this section. The following nine variables are derived and used in our forecasting models:

a) *month*: it refers to the month to which a given record belongs. This variable is coded into numeric data, with "1" referring to January and "12" referring to December. The value of the variable *month* lies in the closed interval [1, 12].

b) *day_month*: this variable refers to the day of the month to which a given record belongs. It is a numeric variable lying within the interval [1, 31]. For example, the date February 14, 2015, will have a value of 14 against the variable *day_month*.

c) *day_week*: a numeric variable that refers to the day of the week corresponding to a given record. This variable lies in the interval [1, 5]. Monday is coded as 1, while Friday is coded as 5.

d) *close_perc*: a numeric variable computed as a standardized value of the percentage change in the *close* prices on two successive days. The computation of the variable is done as follows. Suppose we have two consecutive days: D_1 and D_2. Let the *close* price of the stock for D_1 be X_1 and that for D_2 be X_2. The *close_perc* for D_2 is computed as $(X_2 - X_1)/X_1$ in a percent value.

e) *low_perc*: calculated similarly as *close_perc* as the percentage change in the *low* values over two successive days.

f) *high_perc*: it is also computed in the same way as the *close_perc* as the percent change in the *high* values over two consecutive days.

g) *open_perc*: this variable is also calculated in the same way as *close_perc* as the percent change in the *open* values over successive days.

h) *vol_perc*: computed as the percent change in *volume* over two consecutive days.

k) *range_diff*: it is calculated as the change in *range* values for two successive days – the *range* value for a day is computed as the difference between the *high* and the *low* values.

We compute the values of the above nine variables for each day from January 5, 2015, to December 27, 2019. Using the NIFTY 50 index values, we develop machine learning and deep learning models for classification and regression.

We use the data for the first four years (i.e., December 29, 2014, to December 28, 2018) for training the models. The models are tested on the remaining data, i.e., the data for the period December 31, 2018, to July 31, 2020. In the regression approach, based on the historical movement of the stock index, we predict the stock index values over a horizon of one week. A week consists of five working days - denoted as *Mon*, *Tue*, *Wed*, *Thu*, and *Fri*. The training dataset consists of the records for 209 weeks, and the test dataset contains the stock index records for 83 weeks. We use *close_perc* as the response variable, which is a continuous numeric variable. The objective of the regression methods is to predict the *close_perc* value for each day in the next week, based on the historical data of the stock index movement till the current week. If the predicted *close_perc* is positive, it will indicate an expected rise in the stock index on that day compared to the previous day, while a negative *close_perc* will indicate a fall in the stock index on the day.

In the classification approach, the response variable *close_perc* is a categorical variable having two labels. We convert *close_perc* into a categorical variable with two labels for developing the classification models: "0" and "1". The value "0" indicates a negative *close_perc* value, while "1" signifies a positive *close_perc* value. Hence, if the model expects a rise in the *close_perc* value on the next day, then the predicted *close_perc* for the next day will be "1". A predicted negative value of the *close_perc* on the following day will be indicated by a "0".

For both classification and regression approaches, we consider the following two cases.

Case I: We use the data from December 29, 2014, to December 28, 2018, consisting of 1045 daily records, for training the model. In other words, the training data include stock index records of 209 weeks, where each week consists of five days. The predictive models are tested on the training data to compute the accuracy of the models on the training data. As mentioned earlier, we predict the daily stock index values on a one-week forecast horizon.

Case II: We apply the predictive models to the test dataset to predict the daily *close_perc* for the test period. The test dataset comprises the stock index records from December 31, 2018, to July 31, 2020. Hence, the test data consist of records of 83 weeks. The predictive accuracy of the models is evaluated on the test data on a prediction horizon of one week.

We build eight approaches of classification and eight approaches of regression for constructing our forecasting framework. The following classification models are built: (i) *logistic regression*, (ii) *k-nearest neighbor*, (iii) *decision tree*, (iv) *bagging*, (v) *boosting*, (vi) *random forest*, (vii) *artificial neural network*, and (viii) *support vector machines*. The

models are tested using the following metrics: (i) *recall*, (ii) *specificity*, (iii) *precision*, (iv) *negative predictive value*, (v) *classification accuracy*, and (vi) *F1-score*.

In a similar line, we design eight regression models using the following methods: (i) *multivariate regression*, (ii) *decision tree*, (iii) *bagging*, (iv) *boosting*, (v) *random forest*, (vi) *artificial neural network*, (vii) *support vector machine*, and (viii) *CNN-based multi-step ahead forecasting*. Among these approaches, CNN is a deep learning model, while the remaining others are based on machine learning. For evaluating the regression models, we use two metrics: (i) the *root mean square error* (RMSE) and (ii) the *correlation coefficient* between the *actual* and the *predicted* values of the *target variable* (i.e., *close_perc*).

To make forecasting robust and more accurate, we train a CNN model. CNNs are known for their effectiveness in modeling complex computer vision and image processing problems (Lecun & Bengio, 1995; Geron, 2019). However, in this work, we utilize the power of CNN in forecasting a volatile multivariate time series of NIFTY 50 index values. CNNs have two types of significant processing layers – *convolutional layers* and *pooling* or *subsampling layers*. The convolutional layers read an input such as a two-dimensional image or a one-dimensional signal using a *kernel* by reading the data in small segments and scanning across the entire input field (Binkowski et al., 2018). Each reading results in an interpretation of the input projected onto a *filter map*. The pooling or the subsampling layers take the feature map projections and distill them to the most essential elements, such as using a signal averaging (*average pool*) or signal maximizing process (*max pool*). The convolution and pooling layers are repeated at depth, providing multiple layers of abstraction of the input signals. The output of the final pooling layer is fed into one or more fully-connected layers that interpret what has been read. Finally, the result is mapped to a class value.

We exploit the power of CNN in multi-step time series forecasting in the following way. The *convolutional layers* are used to read sequences of input data and automatically extract features. On the other hand, the pooling layers distill the extracted features and focus attention on the most salient elements. The *fully connected layers* are deployed to interpret the internal representation and output a vector representing multiple time steps. The benefits that CNN provides in our forecasting job are automatic feature learning and the ability to output a multi-step vector directly.

We utilize the power of CNN in forecasting stock prices in two different ways. In the *recursive* or *direct forecast* strategy, the model makes *one-step* predictions, and the outputs are fed as the inputs for subsequent

predictions. In the other approach, we used CNNs to predict the entire output sequence as a *one-step* prediction of the whole vector. Using these two approaches, we build four different types of CNN models for multi-step time series forecasting. In the following, the models are described in detail.

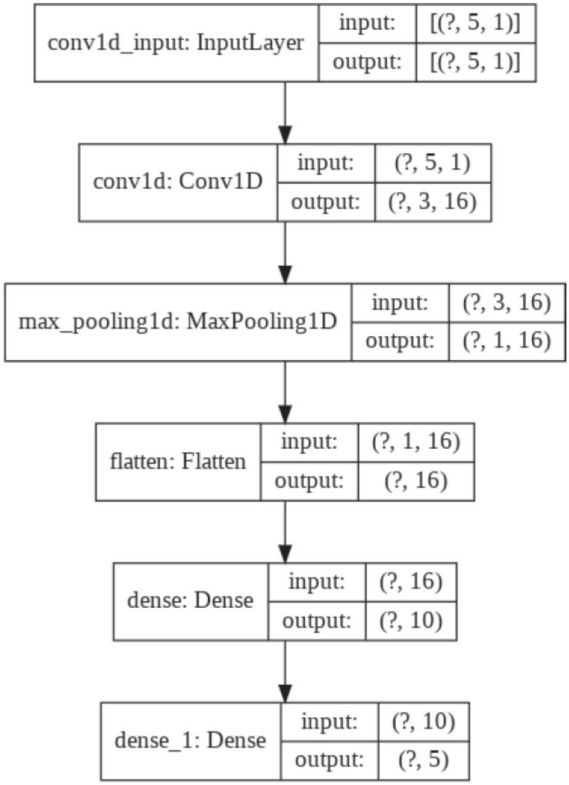

Figure 3-1. The architecture of the *CNN#1* model

(i) *Univariate CNN model with last week's close values as the input*. With the input of last week's (i.e., the last five days') *close* values as the input, the model predicts the *close* values for the days (Monday to Friday) of the next week. In other words, the model predicts the following five *close* values in sequence. We call this model the *CNN#1* model.

(ii) *Univariate CNN model with last two weeks' close values as the input*. With the last two weeks (i.e., last ten days) *close* values as the input, the model predicts the *close* values for the five days in the next week. We call this model the *CNN#2* model.

(iii) *Multivariate CNN model with last two weeks' multivariate data as the input*. This CNN model has five channels. Each channel receives one variable as the input – *open*, *high*, *low*, *close*, *volume*. With the five-variable inputs for 10 days, the model predicts the *close* values for the five days of the next week. We call this the *CNN#3* model.

(iv) *Multivariate, multi-headed CNN model*, wherein each of the five variables – *open*, *high*, *low*, *close*, and *volume* – is handled by a separate CNN. Finally, the outputs of the five CNNs are combined to produce the output of the hybrid model. The input to the model is multivariate data with all the five variables over the last two weeks (i.e., ten days), and the output is the *close* values for the next five days. We call this model the *CNN#4* model.

The architecture of the *CNN#1* model is presented in Figure 3-1. The shape (5, 1) of the input data to the model indicates that only one attribute, i.e., the *close* values of the five days (the last week's daily *close* values), is used as the model's input. The CNN#1 model consists of only one *convolutional* layer that extracts 16 *feature maps* from the data with a *kernel* size (i.e., filter size) of 3. The convolutional layer reads each input three times, and for each reading, it extracts 16 features from the input. The data shape after the execution of the convolution operation is (3,16). The subsampling (also called the *max-pooling* layer) following the convolutional layer reduces the dimensions of the data transforming the data shape into (1, 16). The output of the *max-pooling* layer is converted into a *one-dimensional vector*. The one-dimensional vector is provided as the input into a *fully-connected layer* before the output layer generates the predicted *close* values for the next five days. The *rectified linear unit* (ReLU) activation function is used in the convolutional and the *fully-connected layer*. The *Adam* optimizer is used for optimizing the performance of the *gradient descent* algorithm. The model is trained using 20 epochs using a batch size of 4.

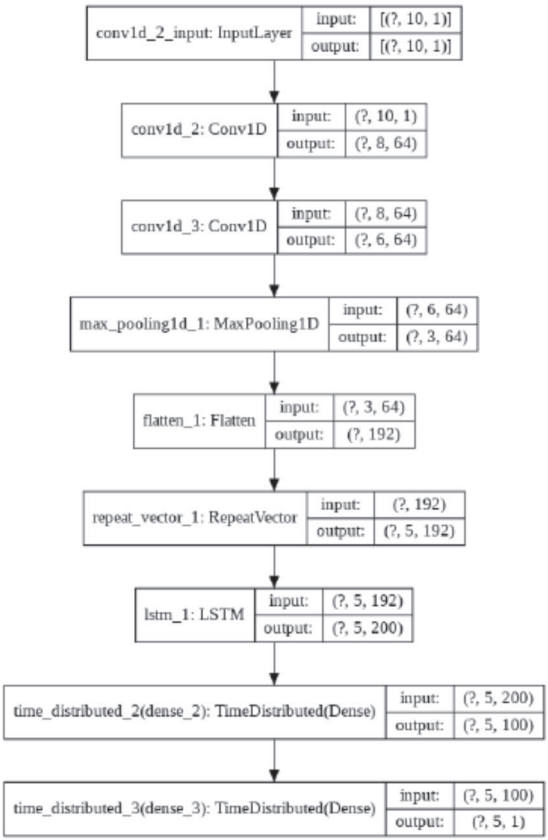

Figure 3-2. The architecture of the CNN#2 model

The architecture of the *CNN#2* model is presented in Figure 3-2. The model uses the *close* values of the last two weeks as the input and produces the predicted values of the *close* prices for the next five days of the coming week. The architecture of the *CNN#2* model is identical to that of the *CNN#1* model. The only difference is in the size of the input. While the *CNN#1* model uses the last week's *close* values as the input, the input to the model *CNN#2* is the previous two weeks' data. This model is also trained using 20 epochs using a batch size of 4.

The third CNN model – *CNN#3* is a multivariate CNN model that uses the multivariate input of the last two weeks. Each of the five variables, i.e., *open*, *high*, *low*, *close*, and *volume,* is used as a separate channel in a single

CNN. Each channel uses its kernel and reads the sequence of input containing its designated variable. The model first uses two *convolutional layers,* each with 32 filter maps (i.e., feature space size) and a kernel size of 3. A *max-pooling layer* follows each convolutional layer. Another convolutional layer receives the output from the second max-pooling layer. This convolutional layer has a feature space size of 16 and a kernel size of 3. A final max-pooling layer follows. One *fully connected layer* with 100 nodes processes the output from the last max-pooling layer before the output layer produces the forecast of the *close* values of the NIFTY index for the five days in the next week. Figure 3-3 presents the architecture of the *CNN#3* model. This model is trained using 70 epochs with a batch size of 16 samples of records. The activation function for all the layers except the output layer is chosen to be ReLU, and the *Adam* optimizer is used for optimizing the performance of the gradient descent algorithm. In the final output layer, the *sigmoid* activation function is used.

The fourth CNN model uses a separate sub-CNN model for each of the five input variables. We refer to this model as a *multi-headed* CNN model and call it the *CNN#4* model. Two convolutional layers with 32 feature maps and a kernel size of 3 are used in this model, followed by a max-pooling layer of size 2. Two dense layers are used with 200 and 100 nodes, respectively before the output layer receives the data using 100 nodes at its input. The model finally produces five output values using the five nodes at its output layer. As in all three previously discussed CNN models, ReLU and ADM are used for activation and optimization. The number of epochs used is 25 with a batch size of 16 records. The multi-headed model is specified using a flexible, functional API defined in the Keras framework (Brownlee, 2019). The model loops over each input variable and creates a sub-model that takes a one-dimensional sequence of 10 days (i.e., two weeks) of data and outputs a flat vector containing a summary of the learned features from the sequence. Each of these vectors is merged by concatenation to make one very long vector that is then interpreted by two successive fully-connected layers before the forecast for the next week is made. Figure 3-4 presents the architecture of the *CNN#4* model.

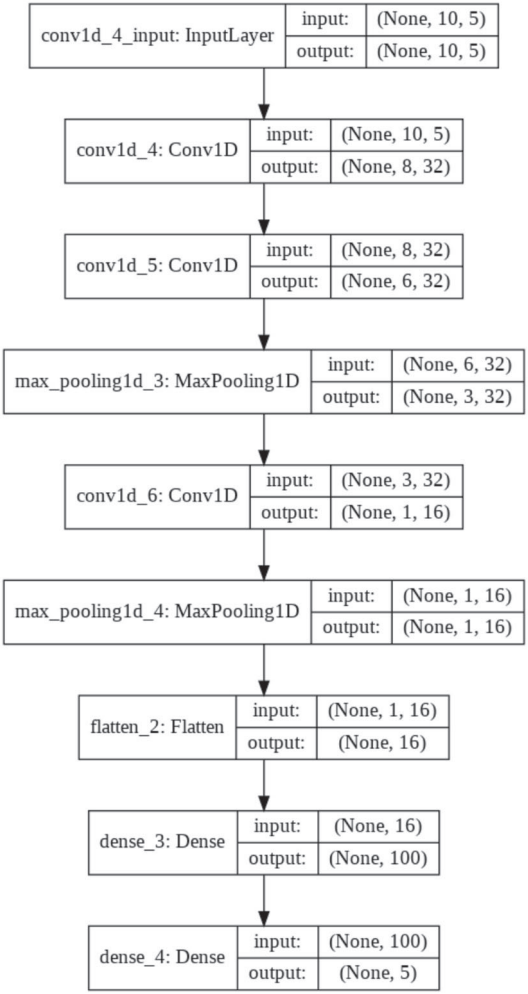

Figure 3-3. The architecture of the *CNN#3* model

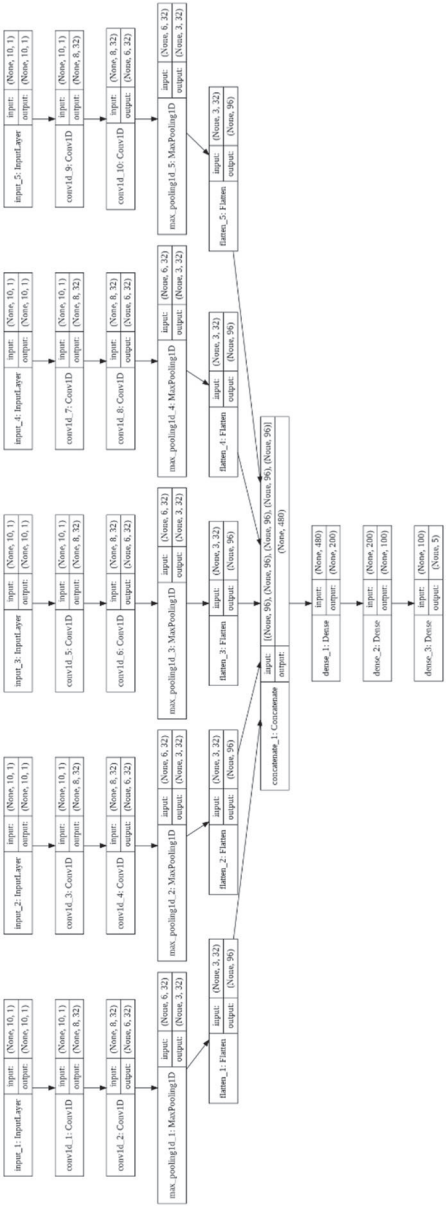

Figure 3-4. The architecture of the *CNN#4* model

Performance Results

This section provides a detailed discussion of the forecasting models that we have used and the results obtained using those models.

For evaluating the classification-based models, we use the following metrics:

Recall: The ratio of the true positives (correctly identified "1" s) to the total number of positives in the dataset. It is often expressed as a percentage.

Specificity: The ratio of the true negatives (correctly identified "0" s) to the total number of negatives in the dataset. Specificity is often expressed as a percentage.

Precision: The ratio of number of true positives to the sum of the true positive cases and the false-positive cases. Precision is often expressed as a percentage.

Negative Predictive Value (NPV): The ratio of the number of true negative cases to the sum of the true negative cases and the false-negative cases. NPV is often expressed as a percentage.

Classification Accuracy (CA): The ratio of the number of correctly classified cases to the total number of cases. CA is often expressed as a percentage.

F1-score: The harmonic mean of the *recall* and the *precision* and is given by (1). The *F1 score* is often expressed as a percentage.

$$F1score = \frac{2*Precision*Recall}{(Precision+Recall)} \tag{1}$$

For comparing the performance of the regression models, we use the *root mean square error* (RMSE) values and the *product-moment correlation* values between the actual and the predicted *close_perc* values of NIFTY 50. Tables 3-1 to 3-8 depict the performance results of the machine learning-based classification models.

TABLE 3-1. LOGISTIC REGRESSION CLASSIFICATION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Sensitivity	65.66	Sensitivity	21.22
	Specificity	96.65	Specificity	94.98
	PPV	81.54	PPV	45.83
	NPV	92.12	NPV	72.67
	CA	87.25	CA	70.90
	F1-score	72.74	F1-score	29.01

TABLE 3-2. KNN CLASSIFICATION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Sensitivity	43.34	Sensitivity	73.21
	Specificity	95.38	Specificity	82.02
	PPV	71.22	PPV	61.57
	NPV	85.21	NPV	89.02
	CA	82.78	CA	81.62
	F1-score	53.89	F1-score	66.89

TABLE 3-3. DECISION TREE (CART) CLASSIFICATION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Sensitivity	42.88	Sensitivity	39.24
	Specificity	93.21	Specificity	89.23
	PPV	64.82	PPV	61.23
	NPV	84.51	NPV	79.21
	CA	81.23	CA	79.49
	F1-score	51.62	F1-score	47.83

TABLE 3-4. BAGGING CLASSIFICATION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Sensitivity	69.21	Sensitivity	42.61
	Specificity	97.36	Specificity	87.31
	PPV	88.51	PPV	53.74
	NPV	91.62	NPV	79.42
	CA	91.24	CA	73.72
	F1-score	77.68	F1-score	47.53

TABLE 3-5. BOOSTING CLASSIFICATION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Sensitivity	100.00	Sensitivity	55.21
	Specificity	100.00	Specificity	81.57
	PPV	100.00	PPV	48.62
	NPV	100.00	NPV	85.76
	CA	100.00	CA	74.64
	F1-score	100.00	F1-score	51.71

TABLE 3-6. RANDOM FOREST CLASSIFICATION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Sensitivity	39.46	Sensitivity	74.91
	Specificity	89.15	Specificity	64.72
	PPV	53.42	PPV	47.82
	NPV	83.21	NPV	87.51
	CA	78.42	CA	68.34
	F1-score	45.39	F1-score	58.38

TABLE 3-7. ANN CLASSIFICATION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Sensitivity	70.12	Sensitivity	21.22
	Specificity	93.34	Specificity	99.72
	PPV	76.71	PPV	99.98
	NPV	88.23	NPV	75.03
	CA	85.41	CA	75.41
	F1-score	73.27	F1-score	35.01

TABLE 3-8. SVM CLASSIFICATION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Sensitivity	64.71	Sensitivity	71.67
	Specificity	78.53	Specificity	75.34
	PPV	18.13	PPV	20.77
	NPV	96.80	NPV	96.72
	CA	77.58	CA	75.03
	F1-score	28.32	F1-score	32.21

The performance results of the machine learning-based regression models are presented in Tables 3-9 to 3-15.

TABLE 3-9. MULTIVARIATE REGRESSION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Correlation RMSE	0.61 33.02	Correlation RMSE	0.57 92.00

TABLE 3-10. DECISION TREE REGRESSION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Correlation RMSE	0.98 61.73	Correlation RMSE	0.52 79.84

TABLE 3-11. BAGGING REGRESSION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Correlation RMSE	0.72 24.45	Correlation RMSE	0.55 29.02

TABLE 3-12. BOOSTING REGRESSION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Correlation RMSE	0.71 18.72	Correlation RMSE	0.59 24.36

TABLE 3-13. RANDOM FOREST REGRESSION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Correlation RMSE	0.95 14.72	Correlation RMSE	0.68 19.26

TABLE 3-14. ANN REGRESSION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Correlation RMSE	0.71 16.32	Correlation RMSE	0.44 28.72

TABLE 3-15. SVM REGRESSION RESULTS

Index	Case I Training Performance		Case II Test Performance	
NIFTY 50	Correlation RMSE	0.73 17.31	Correlation RMSE	0.78 15.59

Next, we present the performance of the four CNN models. For evaluating the performance of the CNN regression models, we use the raw values of the input variables, *open*, *high*, *low*, *close,* and *volume*. All the models are trained and tested on a system consisting of an Intel i5-8250U processor with a clock speed of 1.60GHz-1.80GHz, 8 GB RAM, running a 64-bit Windows 10 operating system. The models are implemented using Python 3.7.4 on TensorFlow 2.30 and Keras 2.4.5 frameworks. The training and testing of the models are done over ten rounds for each model. For every round, we note the performance of the models on four metrics: (i) the overall RMSE, (ii) the RMSE values for the individual days of a week (i.e., Monday – Friday), (iii) the execution time of the model, and (iv) the ratio of the RMSE of the model to the mean of the *close* values in the test dataset. The training and the test datasets consist of 1045 and 415 records, respectively, and the mean *close* value in the test dataset is 11071. Table 3-16 presents the performance of the *CNN#1* model. It is evident from Table 3-16 that Tuesday has the highest RMSE values, and Wednesday to Friday exhibited the lowest RMSE values. The mean RMSE to the mean of the *close* values in the test dataset for the CNN#1 model is 0.8859, while the mean execution time for the model over ten rounds is 15.55 sec. The correlation coefficient between the predicted and the actual close values is 1.00.

TABLE 3-16. THE PERFORMANCE RESULTS OF *CNN#1* MODEL

Round	Overall RMSE	Mon	Tue	Wed	Thu	Fri	Time	Correlation
1	0.876	0.9	1.2	0.7	0.7	0.7	15.43	
2	0.876	0.9	1.3	0.7	0.7	0.7	16.32	
3	0.882	0.9	1.2	0.7	0.7	0.7	15.82	
4	0.897	0.9	1.3	0.7	0.7	0.7	14.27	
5	0.882	0.9	1.3	0.7	0.7	0.7	15.80	
6	0.898	0.9	1.3	0.7	0.7	0.7	15.60	1.00
7	0.883	0.9	1.3	0.7	0.7	0.7	15.72	
8	0.889	0.9	1.3	0.7	0.7	0.7	14.97	
9	0.890	0.9	1.3	0.7	0.7	0.7	15.33	
10	0.886	0.9	1.3	0.7	0.7	0.7	16.21	
Mean	0.8859	0.9	1.3	0.7	0.7	0.7	15.55	

Figure 3-5 presents the plot of the RMSE for different days in a week for the *CNN#1* model as per round 1 listed in Table 3-16.

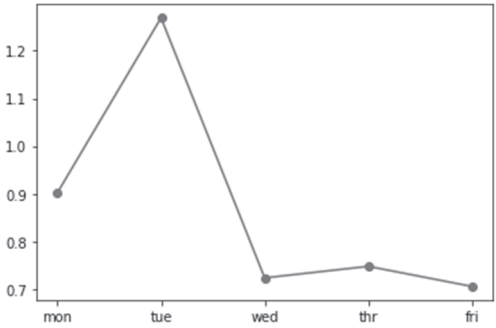

Figure 3-5. The variation of RMSE with different days in a week for the *CNN#1* model (Round #1 in Table 3-16)

TABLE 3-17. THE PERFORMANCE RESULTS OF *CNN#2* MODEL

Round	Overall RMSE	Mon	Tue	Wed	Thu	Fri	Time	Correlation
1	0.894	0.9	1.3	0.7	0.8	0.7	15.73	
2	0.912	0.9	1.3	0.8	0.8	0.7	17.42	
3	0.897	1.0	1.3	0.7	0.8	0.7	15.48	
4	0.887	0.9	1.3	0.7	0.7	0.7	15.29	
5	0.898	0.9	1.3	0.7	0.8	0.7	16.40	
6	0.912	0.9	1.3	0.7	0.7	0.7	16.63	1.00
7	0.886	0.9	1.3	0.7	0.7	0.7	15.93	
8	0.894	0.9	1.3	0.7	0.7	0.7	15.99	
9	0.912	0.9	1.3	0.7	0.9	0.7	16.02	
10	0.869	0.9	1.2	0.6	0.7	0.7	16.29	
Mean	0.8961	0.91	1.29	0.70	0.76	0.70	16.12	

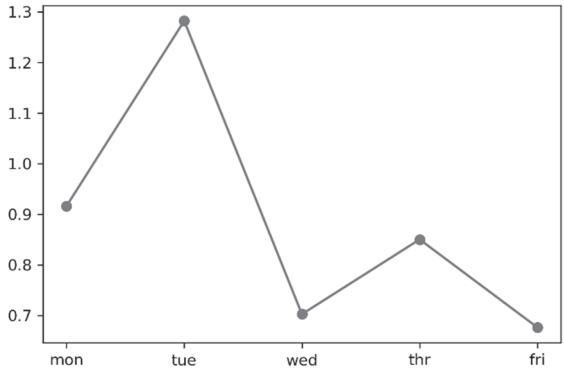

Figure 3-6. The variation of RMSE with different days in a week for the *CNN#2* model (Round #9 in Table 3-17)

Table 3-17 presents the performance results of the CNN#2 model. It is seen from Table 3-17 that, as in the CNN#1 model, the CNN#2 model also exhibits the highest RMSE to mean of the *close* values on Tuesday while the lowest values are observed on Wednesday to Friday. The mean RMSE to the mean of the *close* values over ten rounds is 0.8961, which is also marginally higher than the corresponding value yielded by the CNN#1 model. The mean time of execution of one round is 16.12, which is marginally higher than that of the CNN#1 model. The correlation coefficient for this case is also found to be 1.00, which is identical to the CNN#1 model. Figure 3-6 presents the plot of the RMSE for different days in a week for the CNN#2 model as per round 9 listed in Table 3-17.

TABLE 3-18. THE PERFORMANCE RESULTS OF *CNN#3* MODEL

Round	Overall RMSE	Mon	Tue	Wed	Thu	Fri	Time	Correlation
1	1.156	1.3	1.4	1.0	0.9	1.2	18.35	
2	1.082	1.1	1.4	0.9	1.0	0.9	19.32	
3	1.156	1.2	1.5	0.9	1.0	1.1	18.36	
4	1.105	1.1	1.4	0.9	1.0	1.1	19.57	
5	1.136	1.1	1.6	0.8	1.1	1.0	19.21	
6	1.102	1.1	1.4	0.9	0.9	1.1	18.88	1.00
7	1.042	1.1	1.3	0.9	0.9	0.9	17.23	
8	1.157	1.1	1.4	1.1	1.1	1.0	20.31	
9	1.240	1.4	1.5	0.9	1.0	1.2	19.83	
10	1.205	1.1	1.6	0.9	1.3	1.0	18.93	
Mean	1.1381	1.16	1.45	0.92	1.02	1.05	18.99	

The performance of the CNN#3 model is presented in Table 3-18. It is seen from Table 3-18 that similar to the CNN#1 and the CNN#2 models, the model CNN#3 also exhibits the highest value of the ratio of RMSE to the mean of *close* values on Tuesday. The lowest RMSE to the mean of the *close* values is found to be exhibited on Wednesday. The mean value of the ratio of RMSE to the mean of the *close* values over ten rounds is found to be 1.1381. This model exhibits a higher value for the ratio of the RMSE to the mean of the *close* values, compared to the models CNN#1 and CNN#2. The mean time of execution for one round of the model is 18.99 sec, which is higher than the corresponding values of the CNN#1 and the CNN#2 models. The correlation coefficient for this case is found to be 1.00, which is identical to the CNN#1 and the CNN#2 models. Figure 3-7 presents the plot of the RMSE for different days in a week for the CNN#3 model as per round 3 listed in Table 3-18.

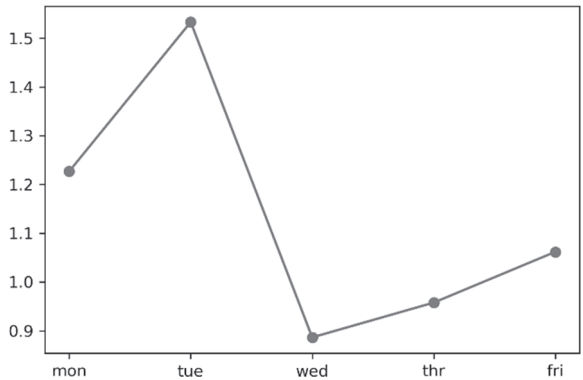

Figure 3-7. The variation of RMSE with different days in a week for the *CNN#3* model (Round #3 in Table 3-18)

TABLE 3-19. THE PERFORMANCE RESULTS OF *CNN#4* MODEL

Round	Overall RMSE	Mon	Tue	Wed	Thu	Fri	Time	Correlation
1	1.082	1.2	1.4	1.0	1.0	0.8	19.20	
2	1.122	1.1	1.3	1.2	1.0	0.9	19.67	
3	1.164	1.2	1.4	1.2	1.1	0.9	20.17	
4	1.167	1.2	1.4	1.3	1.0	0.9	18.21	
5	1.149	1.2	1.4	1.2	1.0	0.9	19.34	
6	1.056	1.1	1.3	0.9	0.9	1.0	17.34	1.00
7	1.166	1.2	1.3	1.2	1.1	0.9	16.38	
8	1.112	1.2	1.5	1.0	0.9	0.8	19.42	
9	1.133	1.2	1.3	1.1	1.0	1.0	20.48	
10	1.066	1.0	1.4	0.9	0.9	1.0	21.03	
Mean	1.1217	1.16	1.37	1.10	0.99	0.91	19.12	

The performance of the *CNN#4* model is presented in Table 3-19. Like the other three models, the *CNN#4* model also exhibits the highest RMSE to the mean *close* value on Tuesday. The lowest RMSE to the mean *close* values is found for Friday. The mean value of the ratio of RMSE to the mean of the *close* values is found to be 1.1217 over ten rounds. The RMSE to the mean of the *close* values for this model is lower than the corresponding value yielded by the model *CNN#3*. However, the same is greater than those of the models *CNN#1* and *CNN#2*. The mean time for execution of one round of the model is 19.12 sec, which is the highest among all the four CNN regression models proposed in this work. However, like the other three models, the correlation coefficient for this model is also found to be 1.00. Figure 3-8 presents the plot of the RMSE for different days in a week for the *CNN#4* model as per round 2 listed in Table 3-19.

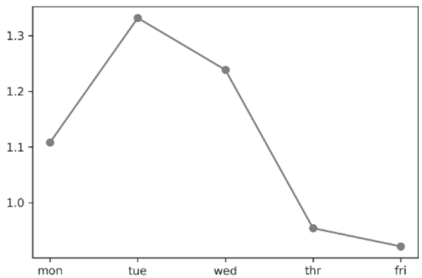

Figure 3-8. The variation of RMSE with different days in a week for the *CNN#4* model (Round #2 in Table 3-19)

It is observed that the *CNN#1* model exhibits the best performance on both the metrics: (i) the accuracy in forecasting and (ii) the execution time for each round. In general, all CNN regression models are found to be more accurate than the machine learning-based regression models.

In the following, we summarize the results of the performance of the classification and regression models.

Table 3-20 lists the best-performing classification models for the training and the testing phases. In Table 3-20, for each metric, the machine learning model that exhibited the best performance in classification for predicting the future stock price movement is listed. It is observed that the KNN model performs the best on the test dataset on two critical metrics – CA and *F1-score*. The boosting model performs the best on the training dataset on all metrics.

TABLE 3-20. THE BEST PERFORMING MACHINE LEARNING CLASSIFICATION MODELS

Index	**Case I Training Performance**		**Case II Test Performance**	
NIFTY 50	Sensitivity	AdaBoost	Sensitivity	RF
	Specificity	AdaBoost	Specificity	ANN
	PPV	AdaBoost	PPV	ANN
	NPV	AdaBoost	NPV	SVM
	CA	AdaBoost	CA	KNN
	F1-score	AdaBoost	F1-score	KNN

TABLE 3-21. THE BEST PERFORMING MACHINE LEARNING REGRESSION MODELS

Index	**Case I Training Performance**		**Case II Test Performance**	
NIFTY 50	Correlation	DT	Correlation	SVM
	RMSE/Mean	RF	RMSE/Mean	SVM

Table 3-21 lists the summary of the best-performing machine learning-based regression models. The SVM model is found to be the best performing regression model on both the metrics – correlation and RMSE on the test dataset. On the training dataset, the *decision tree* yields the highest value of the correlation coefficient, while the *random forest* model produced the lowest value of the RMSE to the mean of the *close* values.

All the CNN-based regression models outperformed the machine learning-based regression models. However, the *CNN#1* model based on the previous week's *close* values as the univariate input is the fastest in execution. This model is also the most accurate one as it has yielded the lowest value of the RMSE to the mean *close* values.

Conclusion

This chapter presented several approaches to the prediction of stock price value and stock price movement on a weekly forecast horizon using eight regression and eight classification methods. The models are based on machine learning and deep learning approaches. Using the daily historical data of NIFTY 50 from December 29, 2014, to July 31, 2020, the models are built, fine-tuned, and tested. The raw data are pre-processed, and relevant variables are identified before the models are designed and optimized. While the NIFTY 50 index values from December 29, 2014, to December 28, 2018, are used for training, the models are tested on the data from December 31, 2018, to July 31, 2020. The regression models are further improved upon by developing four CNN regression models. The CNN models are based on univariate or multivariate input data and they differ in their network architecture and the input data shape. It is observed that the CNN models outperform all machine learning-based regression models in their forecast accuracies. The study has conclusively shown that the deep learning-based models have a higher capability in extracting and learning the features of time series data than their corresponding machine learning counterparts. It also reveals that univariate CNN models are faster in execution and more accurate in their forecasting capability than their corresponding multivariate counterparts. Exploring the applicability and effectiveness of *generative adversarial networks* (GAN) in forecasting stock price movement is a future direction of work.

References

Adebiyi, A. A., Adewumi, A. O. and Ayo, C. K. (2014), "Comparison of ARIMA and artificial neural networks models for stock price prediction", *Journal of Applied Mathematics*, Vol. 2014, Art ID: 614342. DOI: 10.1155/2014/614342.

Baek, Y. and Kim, H. Y. (2015) "ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSRM module", *Expert Systems with Applications*, Vol. 113, pp. 457-480. DOI: 10.1016/j.eswa.2018.07.019

Bao, W., Yue, J. and Rao, Y. (2017) "A deep learning framework for financial time series using stacked autoencoders and long-and-short-term memory", *PLOS ONE*, Vol. 12, No. 7. DOI: 10.1371/journal.pone.0180944.

Binkowski, M., Marti, G. and Domnnat, P. (2018) "Autoregressive convolutional neural networks for asynchronous time series", *Proceedings of the 35th International Conference on Machine Learning*, July 10-15, Stockholm, Sweden, Vol. 80, pp. 580-589.

Brownlee, J. (2017) *Introduction to Time Series Forecasting with Python*, Jason Brownlee Publications. Available Online at: https://machinelearningmastery.com/introduction-to-time-series-forecasting-with-python (Accessed on February 25, 2021)

Chou, J. and Nguyen, T. (2018) "Forward forecast of stock price using sliding-window metaheuristic-optimized machine-learning regression", *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 7, pp. 3132-3142. DOI: 10.1109/TII.2018.2794389.

Ding, G. and Qin, Liangxi (2020) "Study on prediction of stock price based on the associated network model of LSTM", *International Journal of Machine Learning and Cybernetics*, Vol. 11, pp. 1307-1317. DOI: 10.1007/s13042-019-01041-1.

Enke, D., Grauer, M. and Mehdiyev, N. (2011) "Stock market prediction with multiple regression, fuzzy type-2 clustering, and neural networks", *Procedia Computer Science*, Vol. 6, pp. 201-206. DOI: 10.1016/j.procs.2011.08.038.

Geron, A. (2019) *Hands-on Machine Learning with Scikit-Learn Keras & Tensorflow*, O'Reilly Publications, USA. ISBN: 9781492032649.

Gocken, M., Ozcalici, M., Boru, A. and Dosdogru, A. T. (2016) "Integrating metaheuristics and artificial neural networks for improved stock price prediction", *Expert Systems with Applications*, Vol. 44, pp. 320-331. DOI: 10.1016/j.eswa.2015.09.029

Jarrett, J. E. and Kyper, E. (2011) "ARIMA modeling with intervention to forecast and analyze Chinese stock prices", *International Journal of Engineering Business Management*, Vol. 3, No. 3, pp. 53 – 58.

Ivanovski, Z., Ivanovska, N. and Narasanov, Z. (2016) "The regression analysis of stock returns at MSE", *Journal of Modern Accounting and Auditing*, Vol. 12, No. 4, pp. 217-224. DOI: 10.17265/1548-6583/2016.04.003.

Khan, U., Aadil, F., Ghaznfar, M., Khan, S., Metawa, N., Muhammad, K., Mehmood, I. and Nam, Y. (2018) "A robust-regression-based stock exchange forecasting and determination of correlation between stock markets", *Sustainability*, Vol. 10, No. 3702.

DOI: 10.3390/su10103702.

Kim, M. and Sayama, H. (2017) "Predicting stock market movements using network science: An information theoretic approach", *Applied Network Science*, Vol. 2, Art No. 35.DOI: 10.1007/s41109-017-0055-y.

Lecun, Y. & Bengio, Y. (1995) "Convolutional networks for images, speech, and time-series", In: M. A. Arbib (Eds), *The Handbook of Brain Theory and Neural Networks*, MIT Press, Cambridge, MA.

Lin, F-L., Yang, S-Y., Marsh, T. and Chen, Y-F. (2018) "Stock and bond return relations and stock market uncertainty: Evidence from wavelet analysis", *International Review of Economics & Finance*, Vol. 55, pp. 285-294. DOI: 10.1016/j.iref.2017.07.013.

Ma, J. and Liu, I. (2008) "Multivariate nonlinear analysis and prediction of Shanghai stock market", *Discrete Dynamics in Nature and Society*, Vol. 2008, Article ID: 526734. DOI: 10.1155/2008/526734.

Mehtab, S. and Sen, J. (2022) "Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models", In: Sahoo, J. P., Tripathy, A. K., Mohanty, M., Li, K. C., Nayak, A. K. (eds), *Advances in Distributed Computing and Machine Learning*. Lecture Notes in Networks and Systems, Vol 202, pp. 405-423, Springer, Singapore. DOI: 10.1007/978-981-16-4807-6_39.

Mehtab, S. and Sen, J. (2021a) "A time series analysis-based stock price prediction using machine learning and deep learning models", *International Journal of Business Forecasting and Marketing Intelligence (IJBFMI)*, Vol 6, No 4, 272 – 335, Inderscience Publishers. DOI: 10.1504/IJBFMI.2020.115691.

Mehtab, S., Sen, J. and Dutta, A. (2021b) "Stock price prediction using machine learning and LSTM-based deep learning models", In: Thampi, S. M., Piramuthu, S., Li, K. C., Beretti, S., Wozniak, M., and Singh, D. (eds), *Machine Learning and Metaheuristics Algorithms and Applications*. *SoMMA 2020*. Communications in Computer and Information Science, Vol 1366, pp. 88-106, Springer, Singapore. DOI: 10.1007/978-981-16-0419-5_8.

Mehtab, S. and Sen, J. (2020a) *Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models*, Technical Report, No: NSHM_KOL_2020_SCA_DS_1. DOI: 10.13140/RG.2.2.14022.22085/2.

Mehtab, S. and Sen, J. (2020b) "Stock price prediction using CNN and LSTM-based deep learning models", *Proceedings of the IEEE International Conference on Decision Aid Sciences and Applications (DASA'20)*, November 8-9, Sakheer, Bahrain, pp. 447-453. DOI: 10.1109/DASA51403.2020.9317207.

Mehtab, S., Sen, J. and Dasgupta, S. (2020c) "Robust analysis of stock price time series using CNN and LSTM-based deep learning models", *Proceedings of the 4th IEEE International Conference on Electronics, Communication and Aerospace Technology (ICECA'20)*, November 5-7, 2020, Coimbatore, India, pp. 1481-1486. DOI: 10.1109/ICECA49313.2020.9297652.

Mehtab, S. and Sen, J. (2019) "A robust predictive model for stock price prediction using deep learning and natural language processing", *Proceedings of the 7th International Conference on Business Analytics and Intelligence*, December 5-7, Bangalore, India. DOI: 10.2139/ssrn.3502624.

Moreno, J. J. M., Pol, A. P. and Gracia, P. M. (2011) "Artificial neural networks applied to forecasting time series", *Psicothema*, Vol. 23, No. 2, pp. 322-329.

Nam, K. and Seong, N. (2019) "Financial news-based stock movement prediction using causality analysis of influence in the Korean stock market", *Decision Support Systems*, Vol. 117, pp. 100-112. DOI: 10.1016/j.dss.2018.11.004.

Nava, N., Di Matteo, T. and Aste, T. (2018) "Financial time series forecasting using empirical mode decomposition and support vector regression", *Risks*, Vol. 7, No. 1. DOI: 10.3390/risks6010007.

Ning, Y., Wah, L. C. and Erdan, L. (2019) "Stock price prediction based on error correction model and Granger causality test", *Cluster Computing*, Vol. 22, pp. 4849-4858. DOI: 10.1007/s10586-018-2406-6.

Park, N. J., George, K. M. and Park, N. (2010) "A multiple regression model for trend change prediction", *Proceedings of the IEEE International Conference on Financial Theory and Engineering*, June 18-20, Dubai, UAE, pp. 22-26. DOI: 10.1109/ICFTE.2010.5499430.

Patel, J., Shah, S., Thakkar, P. and Kotecha, K. (2015) "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques", *Expert Systems with Applications*, Vol. 42, No. 1, pp. 259-268. DOI: 10.1016/j.eswa.2014.07.040.

Pinheiro, L. D. S. and Dras, M. (2017) "Stock market prediction with deep learning: A character-based neural language model for event-based trading", *Proceedings of the Australasian Language Technology Association Workshop (ALTA'17)*, Brisbane, Australia, pp. 6-15.

Selvin, S., Vinaykumar, R., Gopalakrishnan, E. A., Menon, V. K. and Soman, K. P. (2017) "Stock price prediction using LSTM, RNN, and CNN-sliding window model", *Proceedings of 2017 International Conference on Advances in Computing, Communications and*

Informatics (ICACCI'17), September 13-16, Udupi, India, pp. 1643-1647. DOI: 10.1109/ICACCI.2017.8126078.

Sen, J. (2018) "Stock price prediction using machine learning and deep learning frameworks", *Proceedings of the 6th International Conference on Business Analytics and Intelligence (ICBAI'18)*, December 20-22, Bangalore, India.

Sen, J. (2017a) "A time series analysis-based forecasting approach for the Indian realty sector", *International Journal of Applied Economic Studies*, Vol. 5, No. 4, pp. 8 - 27. DOI: 10.36227/techrxiv.16640212.v1.

Sen, J. (2017b) "A robust analysis and forecasting framework for the Indian mid cap sector using time series decomposition", *Journal of Insurance and Financial Management*, Vol. 3, No. 4, pp. 1- 32. DOI: 10.36227/techrxiv.15128901.v1.

Sen, J. and Datta Chaudhuri, T. (2018) "Understanding the sectors of Indian economy for portfolio choice", *International Journal of Business Forecasting and Marketing Intelligence*, Vol. 4, No. 2, pp. 178-222. DOI: 10.1504/IJBFMI.2018.090914.

Sen, J. and Datta Chaudhuri, T. (2017a) "A time series analysis-based forecasting framework for the Indian healthcare sector", *Journal of Insurance and Financial Management*, Vol. 3, No. 1, pp. 66 - 94. DOI: 10.36227/techrxiv.16640221.v1.

Sen, J. and Datta Chaudhuri, T. (2017b) "A predictive analysis of the Indian FMCG sector using time series decomposition-based approach", *Journal of Economics Library*, Vol. 4, No. 2, pp. 206 - 226. DOI: 10.1453/jel.v4i2.1282.

Sen, J. and Datta Chaudhuri, T. (2017c) "A robust predictive model for stock price forecasting", *Proceedings of the 5th International Conference on Business Analytics and Intelligence BAICONF'17)*, December 11-13, Bangalore, India. DOI: 10.36227/techrxiv.16778611.v1.

Sen, J. and Datta Chaudhuri, T. (2016a) "An alternative framework for time series decomposition and forecasting and its relevance for portfolio choice - a comparative study of the Indian consumer durable and small cap sector", *Journal of Economics Library*, Vol. 3, No. 2, pp. 303 - 326. DOI: 10.1453/jel.v3i2.787.

Sen, J. Sen and Datta Chaudhuri, T. (2016b) "Decomposition of time series data of stock markets and its implications for prediction - an application for the Indian auto sector", *Proceedings of the 2nd National Conference on Advances in Business Research and Management Practices*(*ABRMP'16*), Kolkata, India, pp. 15-28.

DOI: 10.13140/RG.2.1.3232.0241.

Sen, J. and Datta Chaudhuri, T. (2016b) "A framework for predictive analysis of stock market indices – a study of the Indian auto sector", *Journal of Management Practices*, Vol. 2, No, 2, pp. 1-20, Calcutta Business School Publication, Kolkata, India. DOI: 10.13140/RG.2.1.2178.3448.

Sen, J. and Datta Chaudhuri, T. (2016d) "An investigation of the structural characteristics of the Indian IT sector and the capital goods sector - an application of the R programming language in time series decomposition and forecasting", *Journal of Insurance and Financial Management*, Vol. 1, No. 4, pp. 68 - 132. DOI: 10.36227/techrxiv.16640227.v1.

Sen, J. and Mehtab, S. (2021a) "Accurate stock price forecasting using robust and optimized deep learning models", *Proceedings of the IEEE International Conference on Intelligent Technologies (CONIT'21)*, June 25-27, Hubballi, India. DOI: 10.1109/CONIT51480.2021.9498565.

Sen, J. and Mehtab, S. (2021b) "Design and analysis of robust deep learning models for stock price prediction", In: Sen, J. (ed.) *Machine Learning – Algorithms, Models and Applications*, IntechOpen, London UK, pp. 15 – 46. DOI: 10.5772/intechopen.99982.

Sharma, V., Khemnar, R., Kumari, R. and Mohan, B. R. (2019) "Time series with sentiment analysis for stock price prediction", *Proceedings of the 2nd IEEE International Conference on IntelligentCommunication and Computational Techniques (ICCT'19)*, September 28-29, Jaipur, India, pp. 178-181. DOI: 10.1109/ICCT46177.2019.8969060.

Vargas, M. R., de Lima, B. S. L. P. and Evsukoff, A. G. (2017) "Deep learning for stock market prediction from financial news articles", *Proceedings of the IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, June 26-28, Annecy, France, pp. 60-65. DOI: 10.1109/CIVEMSA.2017.7995302.

Wang, L., Zeng, Y. and Chen, T. (2015) "Back propagation neural network with adaptive differential evolution algorithm for time series", *Expert Systems with Applications*, Vol. 42, No. 2, pp. 855-863. DOI: 10.1016/j.eswa.2014.08.018.

Yahoo Finance Website: http://in.finance.yahoo.com.

Yan, Z., Huang, Z. and Liang, M. (2019) "Stock prediction via linear regression and BP regression network", *Communications in Mathematical Finance*, Vol. 8, No. 1, pp. 1-20.

Zhang, G. P. (2008) "Neural networks for data mining", In: Maimon, O. Rokach, L. (eds.) *Soft Computing for Knowledge Discovery and Data Mining*, pp. 17-44, Springer, Boston, MA, DOI: 10.1007/978-0-387-69935-6_2.

Zhu, X., Wang, H., Xu, L. and Li, H. (2008) "Predicting stock index increments by neural networks: The role of trading volume under different horizons", *Expert Systems with Applications*, Vol. 34, No. 4, pp. 3043-304. DOI: 10.1016/j.eswa.2007.06.023.

CHAPTER 4

ROBUST PREDICTIVE MODELS FOR THE INDIAN IT SECTOR USING MACHINE LEARNING AND DEEP LEARNING

ABHISHEK DUTTA & JAYDIP SEN

Introduction

A stock market index helps us scout the performance of the movements of the values, which account for an individual company's indexes for an entire sector like metal, FMCG, IT, oil and natural gas, etc. The stock market can be referred to as a platform where people can make a significant amount of money by understanding the more delicate elements in the behavior and movement pattern of the stock prices. As a reference, we can say that the Indian stock market comprises various sectors and independent indexes of multiple companies. For example, the NIFTY 50 index is a significant indicator of the Indian stock market. It is computed using the stock prices of companies belonging to seventeen industries of the country, including the top fifty companies that have a significant impact on the Indian economy. Similarly, the USA has the Dow Jones Industrial Average (DJIA) as its primary stock market index. Germany has the DAX performance index, which comprises thirty significant companies of the country currently enlisted in the Frankfurt Stock Exchange. Forecasting of these index values can lead to planned investments in the market yielding profits to the investors. However, this requires an extensive amount of knowledge about the intrinsic details of the market. The prediction of the future performance of the stock market and its behavior has been the subject of intensive research for many years now. Researchers in the fields of statistics, economics, and mathematics have done extensive research in this particular area.

Various sectors exhibit their intrinsic characteristics and behave differently. The movement patterns of the index values of the sectors can be

visualized and compared effectively. A solid understanding of these patterns can then be utilized for making wise investment strategies leading to high returns investments. Experts have extensively studied market behavior. However, due to frequent fluctuations of the market index because of various concomitant factors like uncertain demand and supply, political climate, investment environment, and natural disasters, accurate prediction of stock prices always remains a challenging task. The technical indicators like *moving average convergence divergence*, *moving average*, *relative strength index*, and other stochastic approaches are used in identifying various market trends like a *Fibonacci fan*, *head and shoulders*, *triangle*, *inverse head*, etc. In this work, we demonstrate how machine learning and deep learning models can be designed for making accurate predictions of stock prices and stock price movement patterns. It has been established that with proper design and training of models it is possible to make highly accurate predictions of future values of stock prices. Since the data that have been used here is quite challenging, we observe that machine learning models sometimes fail to yield satisfactory results. Hence, we introduce deep learning models based on a *long-and-short-term memory* (LSTM) architecture. Our study shows that the LSTM model is capable of learning from the hidden patterns of the data and complex behavior at a very granular level. The prediction accuracy is also higher for such models.

This chapter studies the behavior of India's *information technology* (IT) sector as it is one of the important sectors of the country's economy. A gamut of statistical, machine learning, and deep learning models is proposed to analyze the *close* values of the IT sector index. Both regression and classification methods are used for predicting the future stock index and its future movement patterns. We use the daily stock price index values of the Indian IT sector from the *NSEIndia* website and use these values to predict the movement of the prices on a forecast horizon of one week. An LSTM regression model is proposed using the univariate stock index data of close index values. The LSTM model is found to outperform all machine learning models. For building the models, we use the historical daily index values from the NSEIndia website from December 29, December 2008 to December 27, 2019. The performance results of the LSTM models show that it is possible to very accurately forecast the daily stock index *close* values for the next week using the last week's stock price records.

Related Work

Stock price prediction and its analysis have been a long-prevailing topic of research. The main idea is to understand the market thoroughly and

determine the best possible way to make a profit. The *efficient market hypothesis (EMH)* believes that it is still impossible to predict the uncertainties in the market. Therefore, the hypothesis states that it is fundamental to understand that to obtain higher returns, an investor must invest his or her money in riskier purchases. Predicting the stock market is not an easy job due to its resemblance to a *random walk* pattern. We can often expect a strong correlation between the current values and their previous ones.

The first breakthrough in this field was the design and construction of regression models **(**Rosenberg et al., 1985; Basu, 1983; Jaffe et al., 1989; Fama & French, 1995; Chui & Wei, 1998; Sen & Datta Chaudhuri, 2016c; Sen & Datta Chaudhuri, 2016d). However, the accuracy levels which we can achieve are not satisfactory. The second category of work for stock price prediction is based on the application of econometric principles and concepts such as *autoregressive moving average (ARMA)*, *Granger causality test, autoregressive integrated moving average (ARIMA), quantile regression,* and *autoregressive distributed lag (ARDL)*(Adebiyi et al., 2014; Jarrett & Kyper, 2011; Mishra, 2016; Mondal et al., 2014; Hanias, 2012; Sen, 2017a; Sen, 2017b; Sen & Datta Chaudhuri, 2018; Sen & Datta Chaudhuri, 2017b; Sen & Datta Chaudhuri, 2017c; Sen & Datta Chaudhuri, 2016a; Sen & Datta Chaudhuri, 2016b; Sen & Datta Chaudhuri, 2016e). The propositions of the third category advocate the use of machine learning and deep learning models in stock price prediction (Mostafa, 2010; Wu et al., 2008; Dutta et al., 2006; Jaruszewicz & Mandziuk, 2004; Mehtab & Sen, 2021a; Mehtab & Sen, 2021b; Mehtab & Sen, 2021c; Mehtab & Sen, 2020a; Mehtab & Sen, 2020b; Mehtab et al., 2020c; Mehtab & Sen, 2020d; Mehtab & Sen, 2019; Siddiqui & Abdullah, 2015; Sen and Dutta Chaudhuri, 2017a; Sen, 2018; Sen et al., 2021a; Sen & Mehtab, 2021b).

Among the well-known stock price prediction methods, an important one is the work of Zhang et al., who propose the use of multilayer backpropagation in data mining (Zhang et al., 2004). The proposal is for an adaptive version of a neural network model for forecasting. The method proposed by the authors is well-suited for forecasting the selling and buying of stocks by following the trends of the current market. The simulation results using the Shanghai composite index for seven years show that the return of assets yielded by the proposed model is approximately three times that is achieved by the *buy-and-hold strategy*.

Mehtab and Sen present machine learning and deep learning techniques using *long-and-short- term memory (LSTM)* and text mining on the social web for accurately predicting the NIFTY 50 index movements in the *National Stock Exchange (NSE)* of India (Mehtab & Sen, 2019). The

authors collect data for three years from January 2015 to December 2017 for building the models. The models are evaluated based on how precisely they can forecast the movements of the *close values* of the index using classification methods and predict the index values using regression techniques. The predictions are made for one week. For the test case, the authors use the stock data from January 2018 to June 2019. The proposed model includes a sentiment analysis module to enhance the predictive accuracy further. The sentiment analysis module analyzes the public sentiment of those NIFTY 50 stocks on Twitter. The output of the sentiment analysis module is fed to the predictive model with the historical NIFTY 50 index values. A non-linear regression model is built based on a s*elf-organizing fuzzy neural network* (SOFNN).

Mondal et al. study the performances of three stocks from three critical sectors of the Indian stock market (Mondal et al., 2020). The three sectors considered by the authors are *information technology*, *healthcare*, and *banking*. The three stocks chosen from these sectors are *Tata Consultancy Services, Cipla,* and *HDFC bank.* Regression and classification models are built using historical stock prices with efficient machine learning techniques and *long-and-short-term memory (LSTM)* to predict the *close* values of each of the stocks and predict the movements of those values. The models are trained using the daily stock data for six years, from January 2013 to December 2018. The models are tested on the stock price data for one year, from January 2019 to December 2019. A comparative analysis of the performance of the models is done. The LSTM regression model is found to have produced the best results.

Chen et al. demonstrate a model for predicting the return on the index of the Taiwan Stock Exchange (Chen et al., 2003). The authors have shown that the investment guided through their models yields higher returns. The proposition includes a *probabilistic neural network (PNN)* for prediction using the historical prices of the stocks. The authors have also used a *Kalman filter* and a *generalized method of movements (GMM)*. The PNN models are found to generate the highest return values.

Analysis of the companies enlisted in the Dow Jones Industrial Average (DJIA) has been done by Basalto et al. (Basalto et al., 2005). The authors propose an approach to identify the temporal behavior of the stock price. A chaotic map is implemented (a pairwise version of clustering), which works on the correlation coefficients between the temporal patterns and the time series. The observation helps in identifying the companies which belong to a particular industry segment. The proposition also includes a framework for designing an efficient portfolio construction and management.

de Faria et al. propose a model for forecasting the stock index movements of the Brazilian stock market using adaptive exponential smoothening and neural nets (de Faria et al., 2009). The authors carry out a comparative study of the performances of the neural network models and the exponential smoothening models based on their return values. The simulated results show that the models yield almost identical results in predicting the return values. However, the accuracy of the neural network models is found to be superior to the exponential smoothing models.

Support vector machines (SVM) and *artificial neural networks (ANN)* are used by Wu et al. in their work, where they propose to make a collaborative effort using these two approaches (Wu et al., 2008). The ensemble models proposed by the authors are compared to some existing methods in the literature. The study reveals that the ensemble approach yields better results than the standalone ANN and SVM models.

Hegazy et al. propose how the gain of an investor can be maximized using the forecasted future stock prices (Hegazy et al., 2013). The authors present a machine learning approach that incorporates *particle swarm optimization (PSO)* with the *least square support vector machine (LS-SVM).* The SVM algorithm is optimized by the PSO method for improving the forecast accuracy of the model on the daily index values. The authors showcased the advantage of using the PSO algorithm to choose the best-fitted parameters of the LS-SVM algorithm. Thirteen financial datasets are analyzed, and the performance of the proposed model is compared with the ANN and the Levenberg-Marquardt algorithm. The results show that the prediction accuracy is improved using the PSO and the LS_SVM models.

Dutta et al. show how the SENSEX values of the Bombay Stock Exchange (BSE) can be forecasted using ANN (Dutta et al., 2006). The authors use the SENSEX records from January 2002 to December 2003 to train and test their proposed models. The weekly *close* values in the test dataset are predicted using two neural networks each having three hidden layers with different input values. The two ANN models are compared using the *mean absolute error (MSE),* and the *root mean squared (RMSE)* values.

Enke et al. propose a model to predict the values of the future stock prices or the movements of the stock prices using a three-stage stock market prediction analysis (Enke et al., 2011). The authors use multiple regression analysis to identify the relationships that the financial variables have with the response variable. The proposition also includes a differential evolution-based fuzzy clustering for setting up the model, and finally, a type 2 fuzzy neural network. It is observed that the model produces superior results when compared to some traditional methods of stock price prediction.

Ma and Liu study and characterize the stock market values of the

Shanghai Stock Exchange (Ma and Liu, 2008). The authors use the concept of non-linear dynamic theory to demonstrate that time series of historical stock prices exhibit nonlinearity. It is observed that for a multivariate time series, the multivariate non-linear prediction model outperforms all other prediction models for polynomials and backpropagation neural networks.

Khan et al. propose regression models to evaluate the robustness of four important stock exchanges - New York, London, NASDAQ, and Karachi (Khan et al., 2018). The proposed model is then applied to the stock data of three top technology companies - Apple, Google, and Microsoft. The authors acquired historical stock data spanning over 20 years and used the same to train and test the models. The performance of the model is examined over a 1-step, 6-steps, and 12-steps horizon of prediction. The performance results of the model based on the *mean absolute error* (MAE) and the *root mean squared error* (RMSE) values show that the model is quite robust in its prediction accuracy.

Ivanovski et al. aim at proving the presence of statistical significance of the daily stock values of the Macedonian Stock Exchange (MSE) (Ivanovski et al., 2016). The analysis of statistics aims to determine the characteristic relationship between the top ten most liquid stocks listed in the MSE. The data comprises ten years of the close price of the Macedonian Stock Exchange Index (MSI-10). The analysis of the stock prices is based on the R^2 values and the correlation between the dependent and independent variables in producing accurate and robust predictions.

Unayik and Guler propose that statistical techniques are very well-suited for predicting time series values and estimating the relationship between various variables (Unayik & Guler, 2013). The main objective of a univariate regression is to analyze the relationship between the independent and the dependent variable. Similarly, multiple independent variables are used in a multivariate regression model. The authors use data from Sakarya University Education Faculty. The data consist of figures of measurement and evaluation, education psychology, instructions, counseling scores, and 2012-KPSS score. Examination of normality and linearity is done, imputation of the missing values is carried out, the output values are verified, and the KPSS values are predicted.

Cakra et al. propose how stock price prediction can be improved by using additional relevant information (Cakra et al., 2015). For example, one of the prime sources of information is *public opinion* on various platforms like social media. It has been observed that the opinions about the products of a company determine the overall performance of its stocks and its reputation. Hence, opinion mining on the social web using sentiment analysis is used to predict stock prices. The proposition illustrates how the

Indonesian stock market can be analyzed using a sentiment analysis based on a Naïve Bayes and a Random Forest classifier. For building the sentiment analysis module, the public sentiments on Twitter are used. A linear regression model is also built to further improve the performance of the sentiment analysis module.

Saadaoui and Messaoud propose a new multi-scaled *feed-forward neural network* (FNN) to predict the values of the response variable in a multivariate regression framework (Saadaoui and Messaoud, 2020). The authors call their new model *empirical mode decomposition* (EMD). The model is inspired by neural *autoregressive distributed lag* (ARDL) and can identify non-linear patterns present in Econophysics time series like seasonality, dependency over long-range periods, non-linear trends, etc. The performance of the model is compared with several existing models, and an extensive set of results is presented based on the comparison.

Wang et al. show that numerous factors are responsible for the characteristic behavior of a stock market (Wang et al., 2018). The authors argue that to handle such dynamic, nonlinear, and volatile data, one must implement more sophisticated learning-based models with enhanced strategies. The authors propose a hybrid model that combines the stock prices with the news sentiments. The hybrid model is found to yield a more precise prediction of future stock prices.

Selvin et al. show that stock markets are profoundly affected by the domestic and the international economy (Selvin et al., 2017). The dynamic movement of the price determines the gain or loss of the investors in the stock market The authors focus on predicting the future movement of stock index values. In addition to this, they also aim at forecasting the close index of a company using a forecast horizon of one day. Instead of fitting a dataset into a specific model, the approach proposed by the authors explores the hidden dynamics existing in the data using three different deep learning architectures. A comparative study is made based on the results and the errors yielded by the models.

Brownlee proposes several deep learning-based models for building a robust predictive framework (Brownlee, 2017). The author argues that several efficient and robust models can be integrated to build an even more accurate model. While building the models, the authors consider both univariate and multivariate stock price data.

Jammalamadaka et al. show that it is possible to incorporate information from Twitter feeds and financial news in stock price prediction (Jammalamadaka et al., 2019). The authors use a *multivariate Bayesian structural time series* (MBSTS), a machine learning model capable of capturing correlations between various time series target variables. The

model is augmented using an *auto-regressive moving average* (ARIMA), a *long-and-short-term memory* (LSTM) network, and a *recurrent neural network* (RNN). The testing of the model is done using a *one-step-ahead* forecasting method. This threefold approach is found to be very accurate in predicting future stock prices.

Porshnev et al. show how the accuracy of stock price prediction can be improved using the stock market indicators based on the psychological states of Twitter users (Porshnev et al., 2013). For a robust analysis of the sentiments on Twitter, the authors propose a *lexicon-based* approach and evaluate the eight basic emotions in more than 755 million tweets.

Mehtab et al. propose a robust approach to predicting stock prices using appropriate modeling and variable selection methods (Mehtab et al., 2020a). The authors use the daily NIFTY 50 index of the Indian stock market from December 2014 to July 2020. Eight regression models are built to predict the *open* values of the stocks using machine learning algorithms. The proposition also includes four LSTM-based deep learning regression models with a *walk-forward validation* approach. Extensive results are presented on the performance of the proposed machine learning and deep learning models. The study reveals that the LSTM model with the univariate input data of last week's *open* values is the most accurate as it yields the lowest RMSE.

Zhou and Fan propose a novel approach for recognizing the modulation of the bit rates (Zhou and Fan, 2019). The authors show how pol-mux (polarisation multiplex) signal can be split by a beam splitter to produce the *asynchronous delay tap plots* (ADTPs). These patterns are then learned further by using the method of *principal component analysis* (PCA). The weights are further converted to their eigenvector values and then fed to an artificial neural network (ANN). The output of the ANN yields the predicted values of the bit rate modulation. A similar approach is also proposed by Khan et al. (Khan et al., 2015).

Adebiyi et al. demonstrate the use of autoregressive integrated moving average (ARIMA) models to predict future stock prices. An extensive set of models with varying parameters is built for predicting the future values of stocks listed on the New York Stock Exchange (NYSE) and the Nigerian Stock Exchange (NSE) (Adebiyi et al., 2014). The results show that the ARIMA models are robust and accurate for short-term predictions of stock prices.

Sharma et al. argue that selling and purchasing of stocks can be done more wisely by investors to maximize the profit by accurate predictions of future stock prices (Sharma et al., 2019). The authors propose a scheme for predicting future stock prices and price movement patterns using machine

learning techniques and content analysis. The proposition includes a tool that helps the investors successfully handle the ups and downs of the ever-changing market. A hybrid model integrating a sentiment analysis module, a multivariate linear regression model, and a decomposable time series framework is found to achieve a high level of accuracy in prediction.

Data Cleaning and Methodology

The first step of the work is to collect the historical index records for the Information Technology (IT) sector of the Indian economy as listed on the National Stock Exchange (NSE) of India. The downloaded data are in Microsoft Excel's *comma-separated values* (.csv) format. The data consist of the following attributes (i.e., variables): (i) *date*, (ii) *open*, (iii) *high*, (iv) *low*, and (v) *close* for the daily stock price records from December 29, 2008, to December 27, 2019. The stock price data from December 29, 2008, to December 28, 2018, are used for training the models. The models are tested using the data from December 31, 2018, to December 27, 2019. The raw data are cleaned, normalized, and imputed before being used in the model building. The following derived variables are first computed:

day: it is an attribute that stores the day of the month for a record. For example, if the date of a record is December 12, 2011, then the value for the *day* attribute is 12. The value of the *day* attribute is an integer in the interval [1, 31].

month: it is an attribute that stores the value of the month for a record. The *month* attribute for a stock record of January 16, 2013, will be 1. The value of the *month* attribute is an integer in the interval [1, 12].

day_week: it is an attribute that stores the day of the week corresponding to a stock price record. For example, Monday's stock index record will have a value of 1 for the *day_week* attribute. The value of the day_week attribute is an integer in the interval [1, 5].

year: it is an attribute that stores the numeric value of the *year* corresponding to a stock index record. In this work, the IT sector index values are considered from 2008 to 2019. Hence, the attribute *year* is an integer in the interval [2008, 2019].

open_norm: it is a normalized variable computed as the fractional change in the *open* value of the index for two consecutive days. For example, the value of the *open_norm* variable for the current day is computed as *open_norm* = (current *open* value – previous *open* value)/previous *open* value.

high_norm: it is a normalized variable calculated as a fractional change in the *high* value of the index for two successive days. For example,

the value of the *high_norm* variable for the current day is computed as *high_norm* = (current *high* value – previous *high* value)/previous *high* value.

low_norm: it is a normalized variable calculated as the fractional change in the *low* value of the index for two consecutive days. For example, the value of the *low_norm* variable for the current day is computed as *low_norm* = (current *low* value – previous *low* value)/previous *low* value.

close_norm: it is a normalized variable calculated as the fractional change in the *close* values of the index for two consecutive days. For example, the value of the *close_norm* variable for the current day is computed as *close_norm* = (current *close* value – previous *close* value)/previous *close* value. We use the *close_norm* as the response variable. All other variables are used as predictors in our proposed models. The positive values in the normalized variables indicate that there has been a rise in the value from the previous day's record. That is, the percentage change in values has been in an upward direction. On the other hand, negative values in the normalized variables indicate a downward movement of that particular value from its previous record.

For regression analysis, the task is to predict the expected values of the *close_norm* variable using the other variables as the predictors. In the training phase, the machine learning models learn from the data and then predict the *close_norm* values for the records in the training data. The trained model is then used to predict the response variable values for the records of the test data set. The performance metrics used are the *root mean squared error* (RMSE), the ratio of the RMSE to the mean of the absolute values of the actual *close_norm* values in the test dataset, and the percentage of cases for which the actual and the predicted *close_norm* values mismatch in their signs. The predictive models are fed with ten years of data from January 2008 to December 2018. The trained models are used to forecast the stock index values for the days in 2019.

The response variable has been transformed into a categorical type for the classification methods. The categorical variable has two labels – 0 and 1. The label 0 refers to all the cases which have negative values for the *close_norm* variable. On the other hand, a case is assigned with label 1 if the *close_norm* value for the case is either zero or positive.

In this work, we have considered two cases – one for training the models and the other for testing. In the following, we explain these two cases:

i) *Case I*: In this case, the models are built using the training dataset consisting of 2609 records. The records are daily data of the IT sector spanned over ten years. We use the stock price records of the training dataset

to train the classification- and regression-based machine learning models. The predicted *close_norm* values are analyzed to determine the accuracy of the models. A comparative study has been done using various metrics for evaluating the performances of the models.

ii) *Case II*: In this case, the trained models are used in predicting the *close_norm* values for the records of the test dataset. The test data set consists of 259 records of the daily stock price for the year 2019. The same accuracy metrics are used to evaluate the robustness and precision of the performance of the predictive models.

We present eight regression models using machine learning approaches and one deep learning-based LSTM model. The eight machine learning regression models are bagging, boosting, decision trees, random forest, multivariate regression, *multivariate adaptive regression spline* (MARS), artificial neural networks, and support vector machines. In addition to these, we also design several classification models. The models are described in detail in the following section.

Machine Learning Models

Machine learning has been prevailing for years now in the domain of modern computing. The concept of machine learning is not just bound within the shackles of the traditional idea of it being imagined in the form of a robot. The spam filter, just like the one in our emails, learns which information is to be retained and which one to be tagged as irrelevant. This is an example of supervised learning to provide prior information to the algorithm to learn and behave accordingly. Machine learning can be implemented to learn from the huge amount of data to gain newer insights and find patterns and behavioral traits. This part of the literature is also known as data mining. The following segment contains brief descriptions of the working principles of the models and their results. First, we explain the regression models and then the classification models.

Regression Models: In the following, we present the regression models proposed in this work. We discuss the design and the performance of the models in detail.

Bagging*:* *Bagging* stands for Bootstrap Aggregation. It is an example of an *ensemble learning* approach in machine learning. In this process, the algorithm creates multiple subsets of the training dataset for analysis. The subsets are created randomly without replacement; if the subsets are chosen *with replacement*, then the process is called *pasting*. In bagging, a particular predictor can be sampled multiple times during the training phase. After the

completion of the training, the algorithm aggregates all the predictions and produces the final output. A simple average function performs the aggregation of the predicted output of the individual models.

We use the library *ipred* in the R language for building the bagging models. Since the bagging approach creates random subsets of the predictors, it is necessary to use the *set.seed* function to produce identical results in each round of execution. In *Case I*, the *bagging* function is implemented to construct the model named *mybag* on the training data. As mentioned earlier, the training dataset consists of 2609 records. The *nbag* parameter is set to a value of 100 so that 100 decision trees are created in the bagging model. The actual and predicted values of the *close_norm* are plotted as shown in Figure 4-1(a). The relationship between the actual *close_norm* values and the predicted *close_norm* values is shown in Figure 4-1(b). In *Case I*, the RMSE is found to be 0.006593, and the mean of the absolute values of the actual *close_norm* is 0.009158. The ratio of the RMSE to the mean of the absolute values of the actual *close_norm* is 71.99. The model misclassified a total of 441 cases out of 2609 records. For these cases, the actual and the predicted *close_norm* values exhibit a mismatch in their signs. Therefore, the overall percentage of mismatch cases is 16.9. The correlation between the actual and the predicted values of *close_norm* is found to be 0.8765.

In *Case II*, the *bagging* function is used to construct the bagging model using the same training data consisting of 2609 records. This model is used to predict the *close_norm* values for the records of the test dataset containing 259 records. The actual and the predicted values of the *close_norm* for *Case II* are plotted as shown in Figure 4-2. The RMSE for the model, in this case, is found to be 0.005669, and the mean of the absolute values of the actual *close_norm* values is 0.00709. The ratio of the RMSE to the mean for *Case II* yields a value of 79.89. The model misclassifies 1247 cases out of 2609 records. For these cases, the actual and the predicted *close_norm* differ in their signs. The overall percentage of mismatched cases is 47.79. The correlation between the actual values of *close_norm* and their predicted values is found to be 0.80506. The model residuals are plotted for both the training and test cases in Figure 4-3(a) and Figure 4-3(b), respectively.

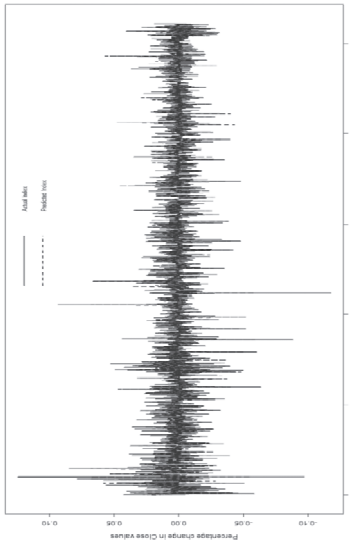

Figure 4-1(a). Bagging regression - actual and predicted *close_norm* values (*Case I*)

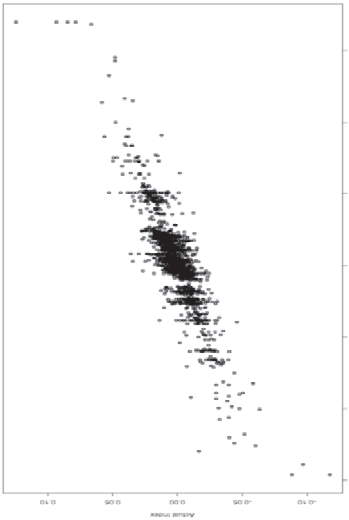

Figure 4-1(b). Bagging regression - relationship between actual and predicted *close_norm* values (*Case I*)

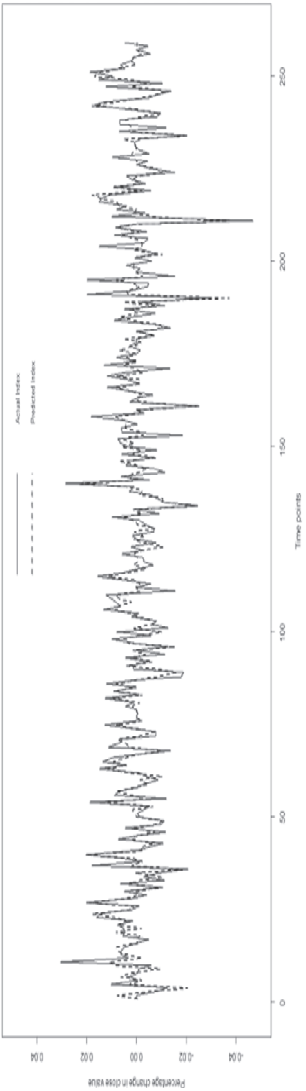

Figure 4-2. Bagging regression - actual and predicted *close_norm* values (*Case II*)

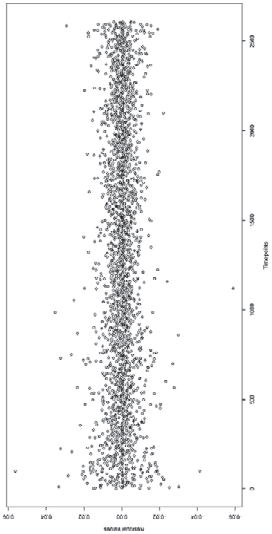

Figure 4-3(a). Bagging regression - the residual plot (*Case I*)

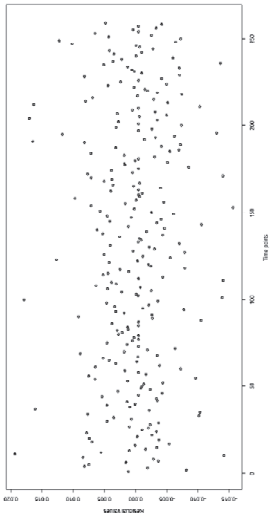

Figure 4-3(b). Bagging regression – the residual plot (*Case II*)

Table 4-1 shows the bagging regression results for both the training and the test cases.

TABLE 4-1. BAGGING REGRESSION RESULTS

Accuracy Metrics	Case I Training Set	Case II Test Set
Correlation between actual and predicted values	0.88	0.81
Root Mean Squared Error (RMSE)	0.00659	0.00566
RMSE/mean of absolute values of the close values	71.99	79.90
Percentage of mismatch cases	16.90	47.79

Boosting: The boosting technique is another example of an ensemble machine learning method. It combines several weak learners (independent variables having a weak correlation with the response variable) and forms a strong learner. These weak learners of the model are singly split decision trees. Boosting has an iterative approach to learning from the data, and thus it continuously tries to rectify the error it commits in the previous steps. Hence, the algorithm assigns more weight to the wrongly classified observations, and thus the concept of adaptation is associated with it. The *blackboost* function defined in the *mboost* library in the R language builds the *boosting* model on the training dataset. Since boosting uses random subsets of the data, *set.seed* function is used so that the same output is produced after every round of execution of the model on the test data. In *Case I*, the RMSE is found to be 0.00531. The ratio of the RMSE to the mean of the actual *close_norm* values is 58.04427, while the correlation coefficient between the actual and the predicted *close_norm* values is found to be 0.92087. 377 cases out of 2609 records exhibit a mismatch in sign between the actual and the predicted *close_norm* values. The actual and the predicted values of *close_norm* values are plotted as shown in Figure 4-4(a). The relationship between the actual and the predicted *close_norm* values are exhibited in Figure 4-4(b).

In *Case II*, the model trained in *Case I* is used to predict the *close_norm* values of the records in the test dataset. The RMSE of the model, in this case, is found to be 0.004605. The ratio of the RMSE to the mean of the absolute values of *close_norm* is 64.90054. The correlation between the actual and the predicted values of *close_norm* is 0.87628, and 42 cases out of 259 instances mismatch in the sign of the actual and predicted

close_norm values. The relationship between the actual and the predicted *close_norm* values for *Case II* is shown in Figure 4-5.

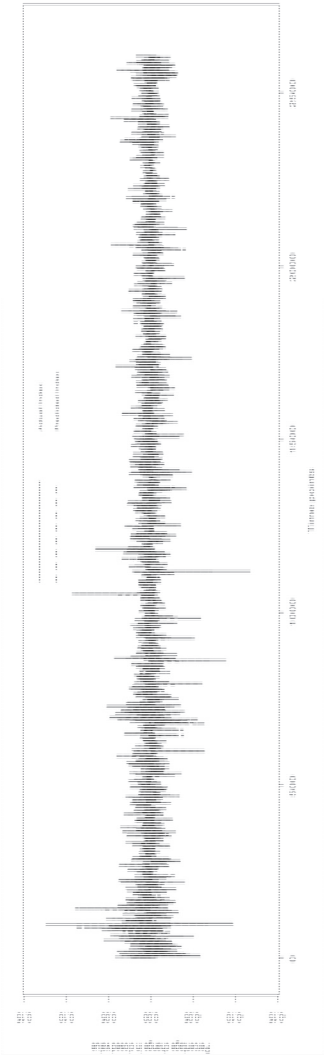

Figure 4(a). Boosting regression - actual and predicted *close_norm* values (*Case I*)

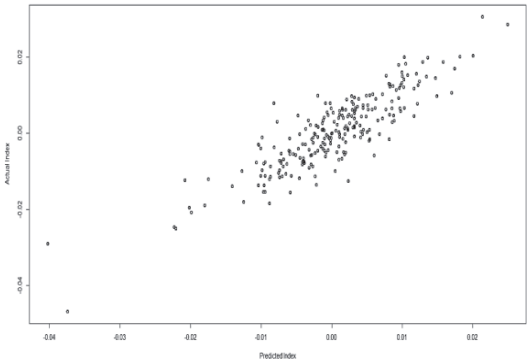

Figure 4-5. Boosting regression - relationship between actual and predicted *close_norm* values (*Case II*)

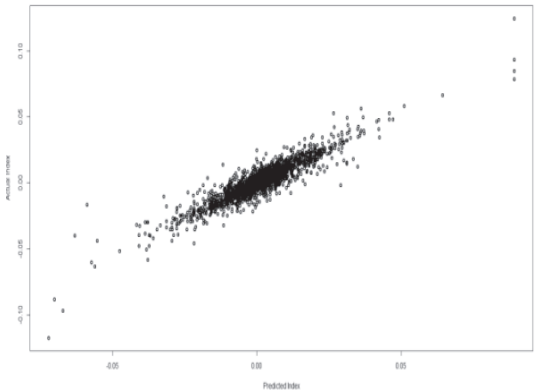

Figure 4-4(b). Boosting regression – relationship between actual and predicted *close_norm* values (*Case I*)

The residual values of the boosting model are plotted for both the training and test datasets in Figure 4-6(a) for *Case I* and Figure 4-6(b) for *Case II*. Table 4-2 presents the values of the metrics for the boosting models. The models have performed quite satisfactorily in both cases.

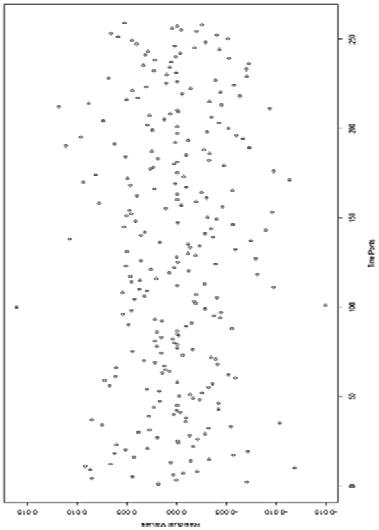

Figure 4-6(b). Boosting regression – the residual plot (*Case II*)

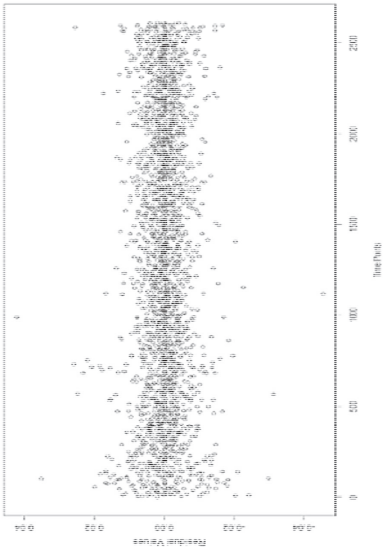

Figure 4-6(a). Boosting regression – the residual plot (*Case I*)

TABLE 4-2. BOOSTING REGRESSION RESULTS

Accuracy Metrics	Case I	Case II
	Training Case	Test Case
Correlation between actual and predicted values	0.92	0.88
Root Mean Squared Error (RMSE)	0.00531	0.00460
RMSE/mean of absolute values of the close values	58.04	64.90
Percentage of mismatch cases	14.45	16.22

Table 4-2 shows the boosting regression results of both the training and test cases.

Decision tree*:* We use the *classification and regression tree* (CART) algorithm for building the decision tree-based regression models. A decision tree is a collection of many nodes connected via branches that protrude from a node, called the *root* node. By convention, the root node is placed at the top of the decision tree. At each splitting of nodes, the attributes of the model are analyzed, and the outcomes are represented by the branches. The process is repeated as the algorithm produces subsets of the data having the same characteristic features or target attributes. The splitting at the nodes is done after considering all the possibilities based on "goodness" criteria.

We use the *tree* function defined in the *tree* library in the R language to build the decision tree-based regression models. In *Case I*, the model is built using the *close_norm* variable as the response variable with *mincut* set to 1. Figure 4-7(a) depicts the model. We observe that the splitting is performed using the predictors *high_norm* and *low_norm* at different levels of the tree. The correlation coefficient between the actual and the predicted values of *close_norm* is 0.84303, while the RMSE is 0.00729. The high correlation coefficient indicates a robust linear relationship between the actual and the predicted *close_norm* values. The ratio of the RMSE to the mean of the absolute values of the *close_norm* is found to be 79.63. Out of the 2609 cases in the training dataset, 548 cases yield a mismatch between the actual and the predicted *close_norm* values resulting in a mismatch percentage of 21. Figure 4-7(b) depicts the actual, and the predicted values of the *close_norm* values for *Case I*. Figure 4-7(c) illustrates the relationship between the actual and the predicted values of *close_norm*.

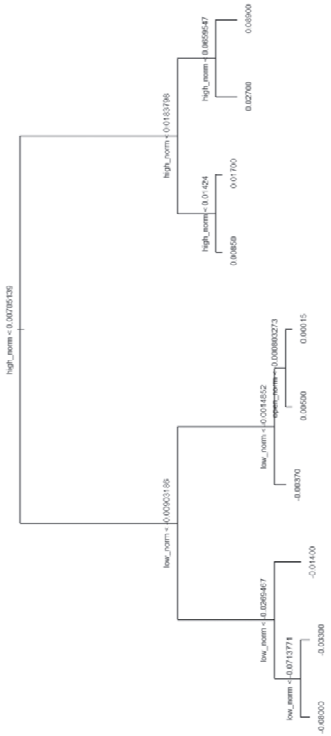

Figure 4-7(a). The decision tree regression model (*Case I*)

In *Case II*, the decision tree model is built using the training dataset, and the model is used to predict the values of the *close_norm* for the records of the test dataset. The correlation coefficient between the actual and the predicted values of *close_norm* is 0.0406. The RMSE of the model is 0.019556, and the ratio of the RMSE to the mean of the absolute values of the *close_norm* is found to be 275.6033. Out of 259 cases of the test dataset,

the model yields 115 cases that mismatch the actual and predicted *close_norm* values resulting in a mismatch percentage of 44.40. The plot of the actual and the predicted values of the *close_norm* is shown in Figure 4-8(a). The relationship between the actual and the predicted *close_norm* values is shown in Figure 4-8(b).

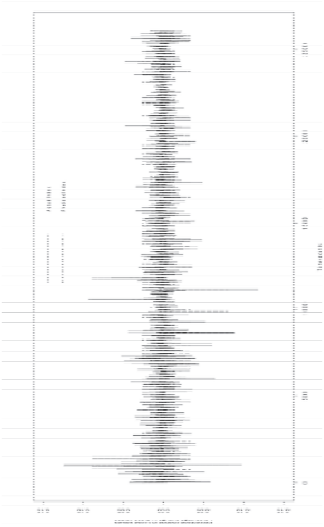

Figure 4-7(b). Decision tree regression - actual and predicted *close_norm* values (*Case I*)

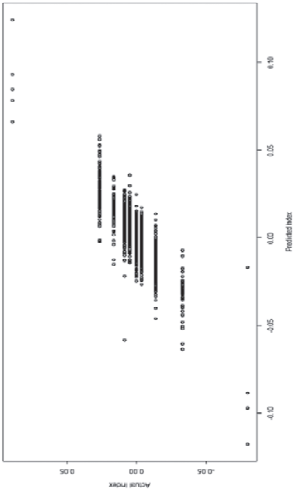

Figure 4-7(c). Decision tree regression – relationship between actual and predicted *close_norm* values (*Case I*)

The results of *Case II* for the decision tree regression model have been unsatisfactory. While the prediction accuracy is very low, the percentage of cases having mismatched signs between the actual and predicted *close_norm* values is very high. The real reason for the failure of the model is its attempt to map a large number of actual *close_norm* values of the test dataset onto a very few *close_norm* values as determined by the trained model. Specifically, the model attempts to fit 259 values of the *close_norm* in the test dataset into the ten leaf nodes in the training model. The residual plots of the training and the test data sets are depicted in Figure 4-9(a) and Figure 4-9(b), respectively. Table 4-3 presents the performance results of the decision tree regression model.

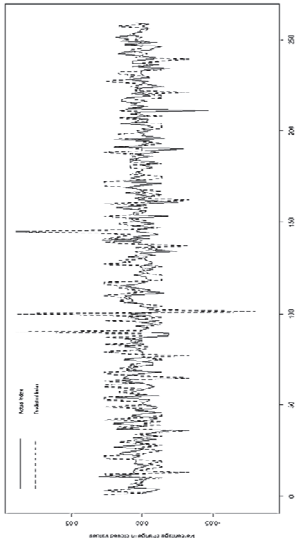

Figure 4-8(a). Decision tree regression-actual and predicted *close_norm* values (*Case II*)

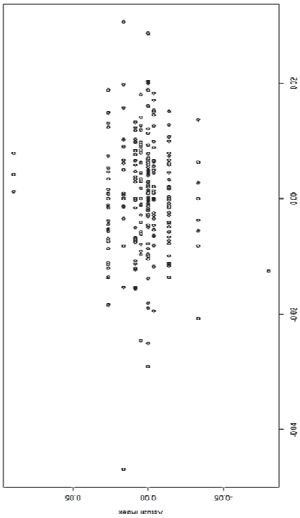

Figure 4-8(b). Decision tree regression – relationship between actual and predicted *close_norm* values (*Case II*)

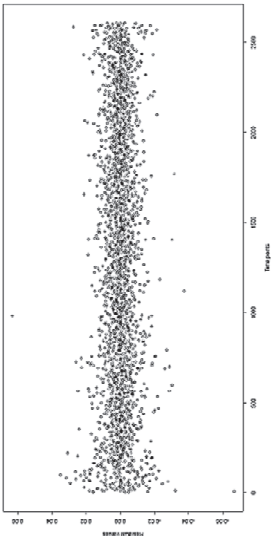

Figure 4-9 (a). Decision tree regression - the residual plot (*Case I*)

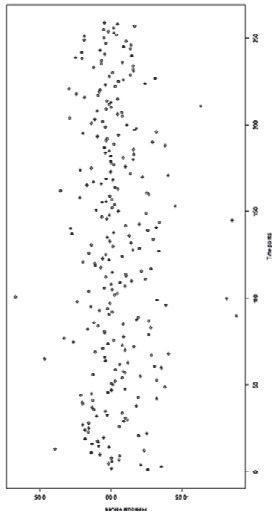

Figure 4-9(b). Decision tree regression – the residual plot (*Case II*)

TABLE 4-3. DECISION TREE REGRESSION RESULTS

Accuracy Metrics	Case I Training Case	Case II Test Case
Correlation between actual and predicted values	0.84	0.0406
Root Mean Squared Error (RMSE)	0.00729	0.01955
RMSE/mean of absolute values of the close values	79.63	275.603
Percentage of mismatch cases	21.004	44.40

Random Forest*:* Decision trees consist of nodes and branches. Splitting at a node for further branching is done based on the identification of the predictor that contributes the most in recognizing the behavior of the data. However, an ensemble of such trees makes the decision-making more robust and accurate. This technique helps in better regularisation of the model so that its performance on the test data does not degrade. Random forest introduces randomness in the model, and unlike a single decision tree, it doesn't split a node based on the identification of the best feature. Instead, it looks for the best features among a selected subset of the features from all the decision trees built during the training. Such randomness makes the model more regularised and robust to work effectively on the *out-of-sample* test data.

We build a random forest regression model using the *randomForest* function defined in the *randomForest* library in the R language. In *Case I*, the model is built using the training data, including all the predictor variables. The random forest model is constructed using 500 trees with two candidate variables being considered at each split. The percentage of variance explained by the model is 76.4, with the mean of the squared residual values of 0.04339. The *root mean squared error* (RMSE) value is 0.00307, while the ratio of the RMSE to the mean of the absolute values of the *close_norm* is 33.603. The correlation coefficient between the actual and the predicted *close_norm* values is found to be 0.979. Out of 2609 cases, 144 cases exhibit mismatched signs between the actual and predicted *close_norm* values, resulting in a mismatch percentage of 5.519. The plot of the actual versus the predicted values of *close_norm* is depicted in Figure 4-10(a). The relationship between the actual and the predicted *close_norm* is exhibited in Figure 4-10(b).

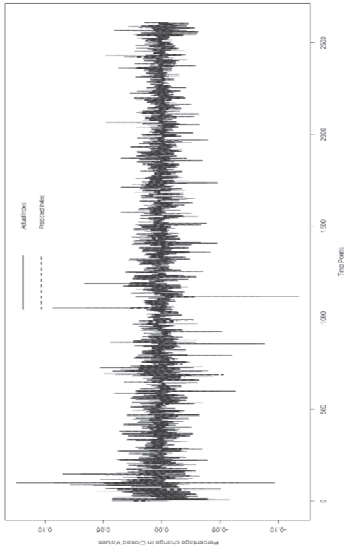

Figure 4-10 (a). Random forest regression - actual and predicted *close_norm* values (*Case I*)

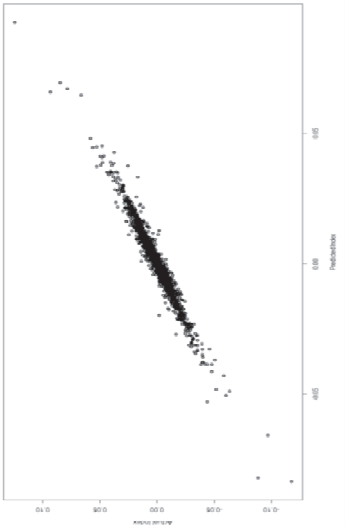

Figure 4-10 (b). Random forest regression-relationship between actual and predicted *close_norm* values (*Case I*)

For *Case II*, the random forest model constructed in *Case I* is used in predicting the *close_norm* values for the test dataset consisting of 259 records. The model is built using 500 trees. Two randomly chosen variables are considered for the splitting at each node. The mean of the squared residuals for the model is 0.04339, and the explained variance is 76.4. The RMSE for the model is 0.00476, while the ratio of the RMSE to the mean of the absolute values of *close_norm* is found to be 67.17. The correlation coefficient between the actual and the predicted values of the *close_norm* is 0.8697. The model yields 47 cases out of 258 that exhibit a sign mismatch, resulting in a mismatch percentage of 18.14. The plot of the actual and the predicted *close_norm* is shown in Figure 4-11.

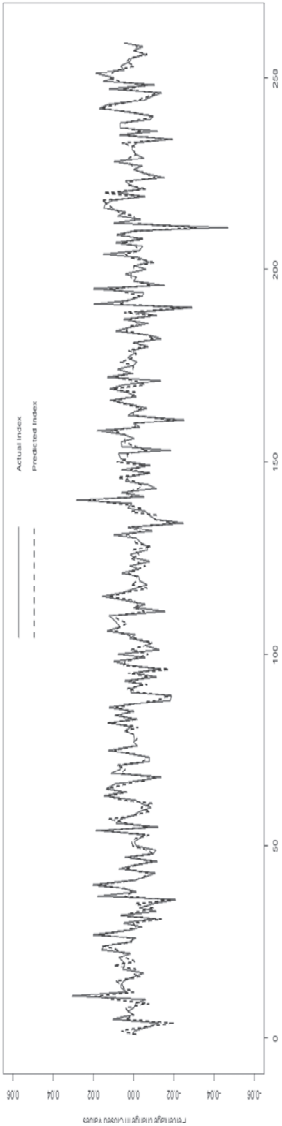

Figure 4-11. Random forest regression - actual and predicted *close_norm* values (*Case II*)

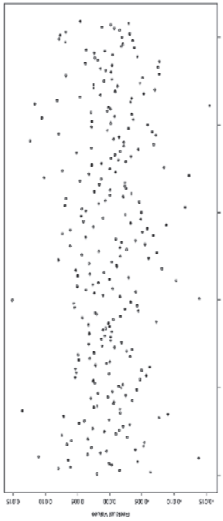

Figure 4-12(b). Random forest regression–the residual plot (*Case II*)

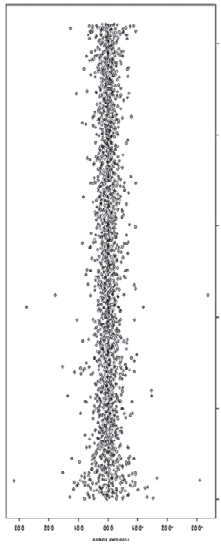

Figure 4-12 (a). Random forest regression - the residual plot (*Case I*)

TABLE 4-4. RANDOM FOREST REGRESSION RESULTS

Accuracy Metrics	Case I	Case II
	Training Case	Test Case
Correlation between actual and predicted values	0.98	0.87
Root Mean Squared Error (RMSE)	0.0031	0.0047
RMSE/mean of absolute values of the close values	33.60	67.17
Percentage of mismatch cases	5.52	18.15

The residuals plots of the random forest models for *Case I* and *Case II* are shown in Figure 4-12(a) and Figure 4-12(b), respectively. Table 4-4 presents the values for all the accuracy metrics of the random forest models. The random forest model for *Case I* has produced the best results among all machine learning-based regression models. However, the results of the random forest regression in *Case II* are not that impressive.

Artificial Neural Networks: A typical neural network consists of three layers: the input layer, the hidden layer, and the output layer. These networks can extensively learn from the input data by using an arbitrary number of hidden layers. It is also observed that the increase in the number of hidden layers also increases the complexity of the model. Artificial neural networks or simply ANNs imitate the working of a brain and inherit its characteristics, such as the weighted sums, which are nothing but the synapses of a brain. The nodes of the network act as neurons. The input layer receives the data values through the nodes, and then weighted values are fed into the first hidden layer which, in turn, are fed into the next layer and so on. The output layer receives the weighted sum from the last hidden layer, and then it applies an *activation* function (for example, the *sigmoid* function) to produce the final output.

We build two ANN models using the *neuralnet* function defined in the *nueralnet* library of the R language. In *Case I*, the ANN model is built using the 2609 records of the training dataset to predict the *close_norm* values. The values of the predictors are normalized using the *min-max normalization* method. The number of hidden layers used in the model is two. The RMSE of the model is found to be 0.00548, and the ratio of the RMSE to the mean of the absolute values of actual *close_norm* values is 59.92. There are 385 cases out of 2609 whose signs of the actual and predicted *close_norm* values mismatch, yielding a mismatch percentage of

14.75. The correlation coefficient between the actual and the predicted *close_norm* values is 0.9144. The neural network model for *Case I* is shown in Figure. 4-13. A total of 39649 iterations are required to build the model.

Figure 4-14(a) shows the actual and the predicted values of the *close_norm* plotted across time in *Case I*, while Figure 4-14(b) shows the linear relationship between them. For *Case II*, the model of *Case I* is used in predicting the values of the *close_norm* for the records in the test dataset. It may be noted here that the test dataset contains 259 records. *Case II* has an RMSE value of 0.0052, and the ratio of the RMSE to the mean of the absolute values of *close_norm* is 73.41. Out of the 259 test records, 50 cases exhibit mismatch in the signs of the actual and predicted *close_norm* values leading to a 19.3% mismatch. The correlation between the actuals and the predicted values is 0.841. The performance of the ANN model for *Case II* is not as accurate as of the model in *Case I*. This is evident from a higher mismatch percentage and a lower value of the correlation coefficient between the actual and the predicted *close_norm* values in *Case II*. The architecture of the ANN model for *Case II* is shown in Figure 4-15. The actual and the predicted values of *close_norm* for the ANN regression model under *Case II* are depicted in Figure 4-16. The residuals plots for the ANN regression models under *Case I* and *Case II* are shown in Figure 4-17(a) and Figure 4-17(b), respectively.

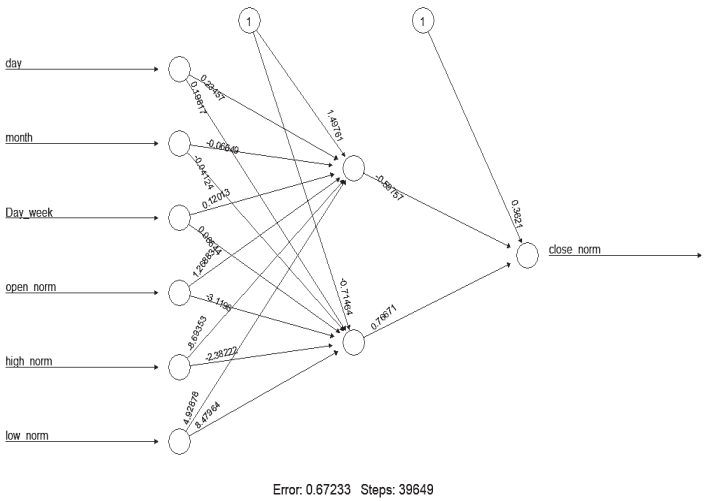

Figure 4-13. The architecture of the ANN regression model (*Case I*)

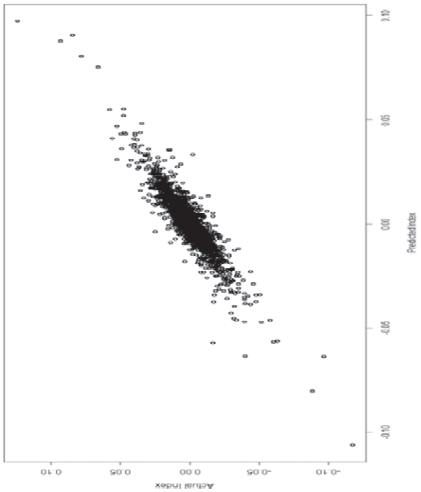

Figure 4-14(a). ANN regression- actual and predicted *close_norm* (*Case I*)

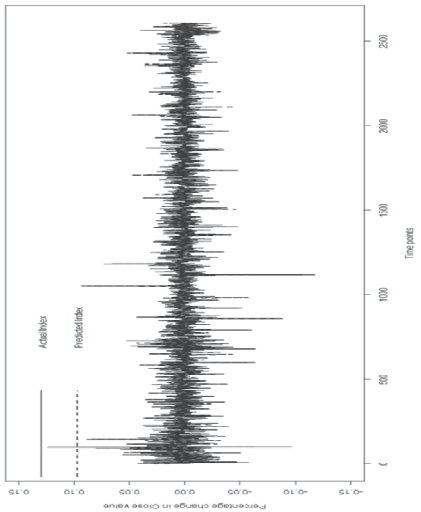

Figure 4-14(b). ANN regression - relationship between actual and predicted *close_norm* (*Case I*)

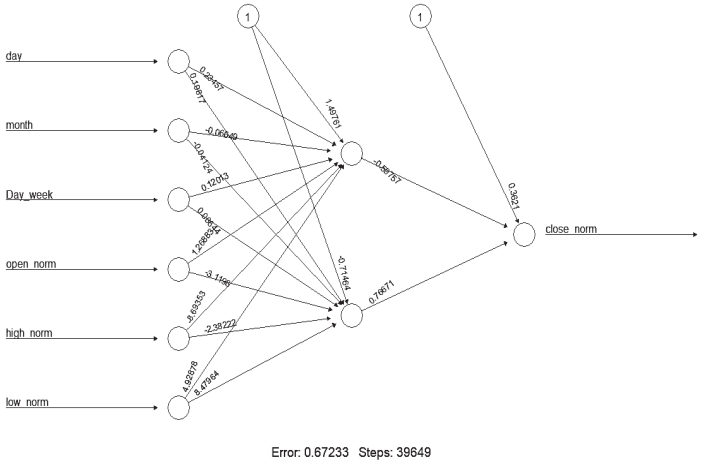

Figure 4-15. The architecture of the ANN regression model (*Case II*)

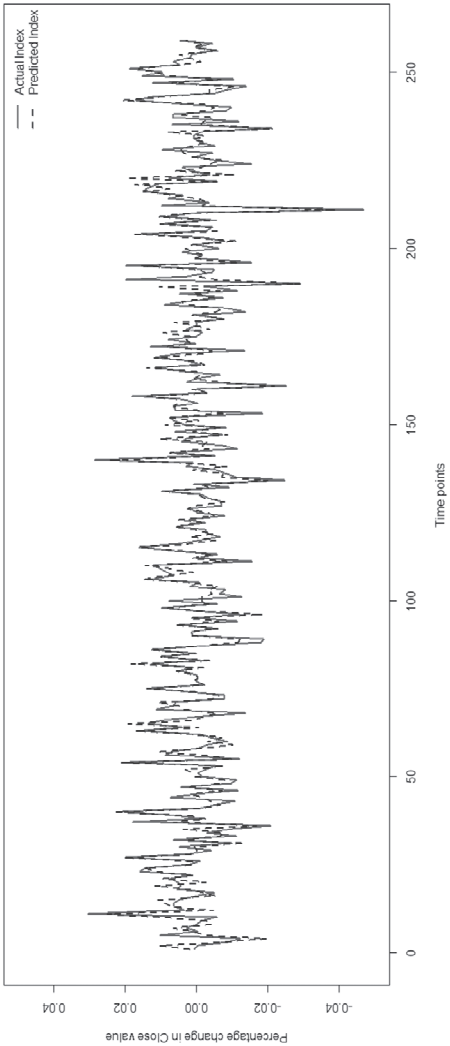

Figure 4-16. ANN regression - actual and predicted *close_norm* values (*Case II*).

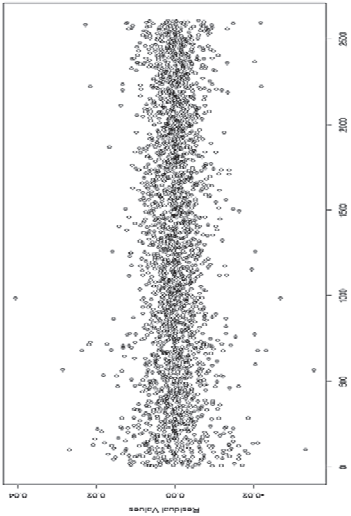

Figure 4-17(a). ANN regression - the residual plot (*Case I*)

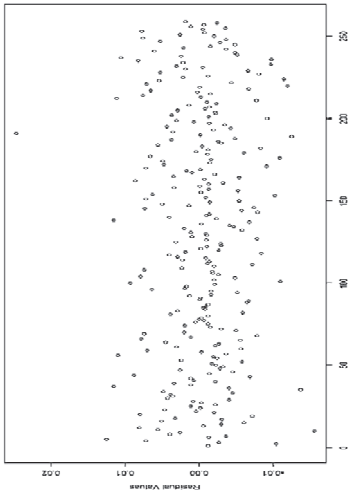

Figure 4-17(b). ANN regression – the residual plot (*Case II*)

Table 4-5 presents the performance results of the ANN regression models for *Case I* and *Case II*. The models are evaluated based on the three metrics: (i) correlation between the actual and the predicted values of *close_norm*, (ii) the ratio of the RMSE to the mean of the *close_norm* values, and (iii) the percentage of the mismatched cases.

TABLE 4-5. ANN REGRESSION RESULTS

Accuracy Metrics	**Case I**	**Case II**
	Training Case	**Test Case**
Correlation between actual and predicted values	0.91	0.84
Root Mean Squared Error (RMSE)	0.0054	0.0052
RMSE/mean of absolute values of the close values	59.92	73.41
Percentage of mismatch cases	14.75	19.3

Multivariate regression: Multivariate regression models in the R programming language can be built in two different ways: (i)) the *forward addition* method and (ii) the *backward deletion* method. However, both methods when applied to the same dataset lead to identical results. We check the existence of multicollinearity among the predictors by applying the *variance inflation factor (VIF)* function to the regression model. The *Akaike Information Criteria (AIC)* is used as the information criterion to determine the predictors to be included in the regression model.

For *Case I*, the regression model is trained using the *lm* function on the training dataset records. The *vif* test is carried out to identify the multicollinear variables using the *vif* function defined in the *faraway* library in the R language. The VIF values are found to be: 1.000654 for *day*, 1.000442 for *month*, 1.005116 for *day_week*, 1.002519 for *year*, 1.970780 for *open_norm*, 2.111402 for *high_norm*, and 2.151080 for *low_norm*. Since none of the predictors exhibit multicollinearity, we apply the *drop1* function to identify the predictor that should be removed from the model based on its AIC and *p*-values. It is observed that among the predictors having non-significant *p*-values, the predictor *day* has the least AIC value of 26712.0. It implies that the predictor *day* provides the least information in predicting the *close_norm* values. We remove the preditor *day* from the model and rebuild the regression model using the *lm* function. We again apply the *drop1* function on the model to identify the predictor that should be dropped in the next round. Following the same method, we remove the predictors, *year,* and *month*. The remaining predictors cannot be removed

as they are found to have significant *p*-values.

The RMSE of the model is computed using the function *rmse*. This function is defined in the *Metrics* library in the R language. The RMSE value in *Case I* is found to be 0.00596, while the ratio of the RMSE to the mean of the absolute values of the response variable *close_norm* is 65.14. The correlation coefficient between the actual and the predicted values of *close_norm* is 0.8980, and the 394 cases out of 2609 records mismatch in sign between the actual and the predicted *close_norm* values. The percentage of mismatched cases is 15.01. The forward addition method of regression yields identical results. In the *forward addition* approach, we use the *add1* function to include the predictor with the lowest AIC and a significant *p*-value in the model. Figure 4-18(a) shows the plot of the actual and the predicted *close_norm* values for *Case I*. The relationship between the actual and the predicted *close_norm* values is illustrated in Figure 4-18(b).

In *Case II*, the multivariate model of *Case I* is used in predicting the *close_norm* values for the records of the test dataset. *Case II* yields an RMSE value of 0.0047, and the ratio of the RMSE to the mean of the absolute values of the *close_norm* is found to be 66.27. There are 44 sign-mismatched cases out of 259 cases, and hence there is a 16.98% mismatch. The correlation coefficient between the actual and the predicted *close_norm* values is 0.87. Figure 4-19 shows the plot of the actual and the predicted *close_norm* values for *Case II*. Figure 4-20(a) and Figure 4-20(b) depict the residual plots for *Case I* and *Case II,* respectively. It is evident that the residuals are random and do not exhibit any significant autocorrelation. Table 4-6 presents the performance of the multivariate regression models for both *Case I* and *Case II*.

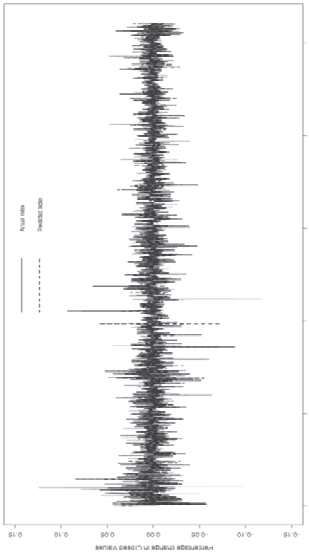

Figure 4-18(a). Multivariate regression - actual and predicted *close_norm* (*Case I*)

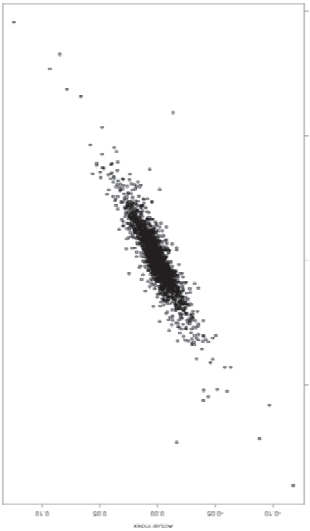

Figure 4-18(b). Multivariate regression - relationship between actual and predicted *close_norm* (*Case I*)

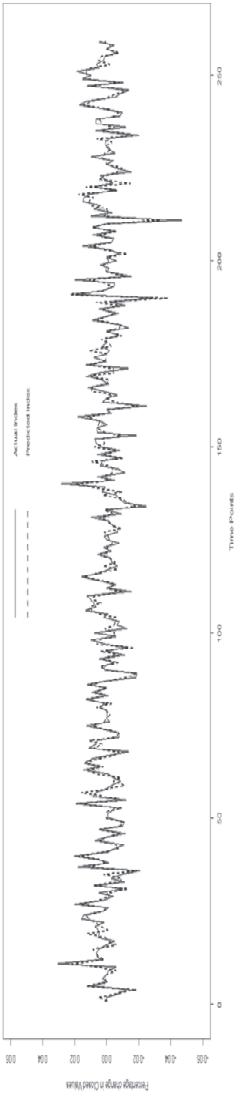

Figure 4-19. Multivariate regression - actual and predicted *close_norm* values (*Case II*)

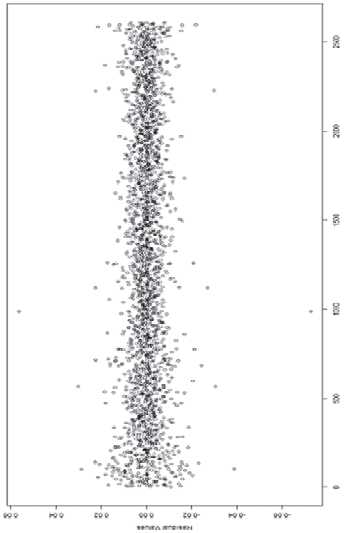

Figure 4-20(a). Multivariate regression - the residual plot (*Case I*)

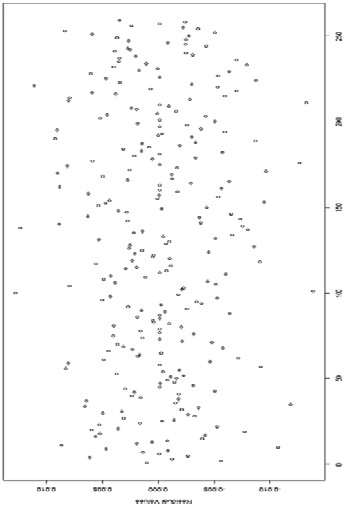

Figure 4-20(b). Multivariate regression – the residual plot (*Case II*)

Table 4-6 presents the performance results of the multivariate regression model.

TABLE 4-6. MULTIVARIATE REGRESSION RESULTS

Accuracy Metrics	Case I Training Case	Case II Test Case
Correlation between actual and predicted values	0.90	0.87
Root Mean Squared Error (RMSE)	0.0059	0.0047
RMSE/mean of absolute values of the close values	65.14	66.27
Percentage of mismatch cases	15.10	16.98

Multivariate Adaptive Regression Spline (MARS): MARS is an efficient statistical modeling technique based on regression whose working principle involves splitting the input variables into several step functions. These functions are known as the *basis functions*. The basis functions are assessed from the cutpoints of the data, known as the *knots.* The algorithm searches through a range of values at each knot or cutpoint and identifies the step function that minimizes the error. Subsequently, a *hinge function* is applied at each of the knots, and this procedure is continued for building a robust non-linear predictive model. An increase in the number of knots will lead to a better fitting of the model to the records in the training dataset. However, it will also cause the overfitting of the model. An overfitted model will produce less accurate results when applied to the test data because of its lack of generalization. Hence, pruning by cross-validation of the model will be required to obtain the optimal number of cutpoints or *knots.* The working principle of MARS involves designing a complex model by using the step functions in pairs at each knot. The identification of the knots is made during the *forward pass* phase. In the *backward pass* of the execution of the algorithm, the terms which are not significant contributors to the model are pruned off to avoid *model overfitting*.

In *Case I*, the MARS model is built using the *earth* function defined in the *earth* library in the R language. The parameters of the *earth* function are determined using a *grid search* using the *expand.grid* function under the *dplyr* library in the R language. After the optimization process, it is found that the model has chosen the *nprune* value of 23. In other words, 23 terms are retained in the model with a degree of freedom of 2. We have already mentioned that MARS prunes off some of the terms in its backward pass of execution. We observe the reduction of terms done by the model using

gridExtra::grid.arrange function and the values of the *generalized cross-validation* (GCV) and the *residual sum of squares* (RSS) values. From Figure 4-21, we find that the features, *day*, *day_week*, and *year*, have zero significance in the model. Hence, these three features are excluded from the primary model. The MARS model is then built using the *earth* function, and the values of the parameters *nprune* and degree are set to 23 and 2, respectively.

The model in *Case I* selects all the fifteen terms and four out of the seven predictors. The predictors *day*, *day_week*, and *year* are excluded. The GCV value of the model is 2.915836e-05, and the value of RSS is 0.07398924. The *R*-square and the generalized *R*-square values of the model are 0.8415275 and 0.8457524. It implies that the predictors explain 84.57% of the variance of the response variable. The RMSE value in *Case I* is found to be 0.00532, the ratio of the RMSE to the mean of the absolute values of *close_norm* is 58.1466. The correlation coefficient between the actual and the predicted values of *close_norm* is 0.9196. It is observed that 390 cases out of 2609 exhibit sign mismatch, yielding a mismatch percentage of 14.940. Figure 4-22(a) shows the actual and predicted values of *close_norm* in *Case I*, while Figure 4-22(b) shows their linear relationship.

In *Case II*, the model of *Case I* is used in predicting the *close_norm* values for the test dataset records. The *R*-square and the generalized *R*-square values of the model are 0.8415275 and 0.8457524, respectively. The model in *Case II* yields an RMSE value of 0.00461, while the ratio of the RMSE to the mean of the absolute values of *close_norm* is 64.97. Out of 259 cases, 44 cases exhibit sign mismatch in the actual and the predicted values of *close_norm*, leading to a mismatch percentage of 16.98. The correlation coefficient between the actual and the predicted *close_norm* values is found to be 0.8758.

Figure 4-23 shows the plot of the actual and the predicted values of the *close_norm* in *Case II* of the MARS regression model. Figure 4-24(a) and Figure 4-24(b) show the residual plots for *Case I* and *Case II* for the MARS regression model. The performance results of the MARS regression models are presented in Table 4-7.

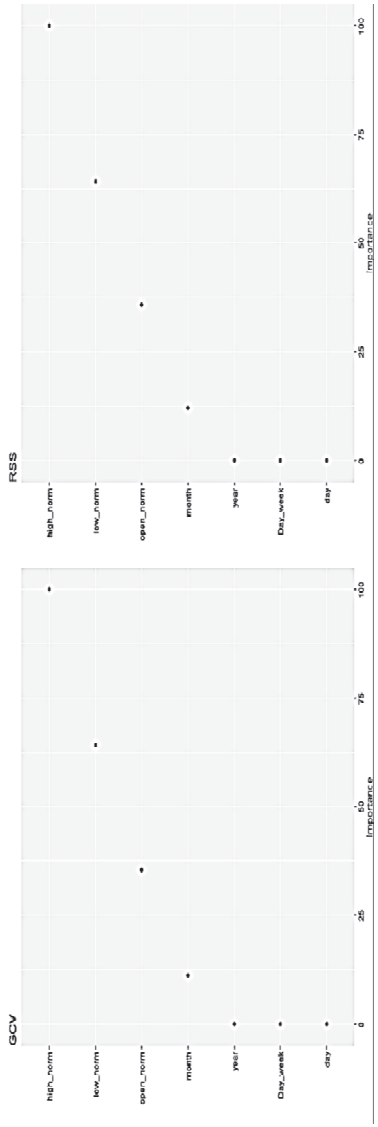

Figure 4-21. MARS regression- post model tuning with insignificant features exclude excluded (*Case I*)

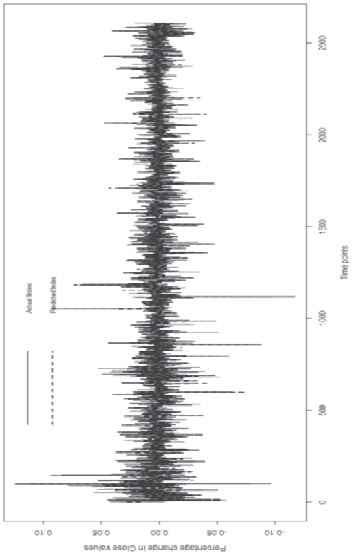

Figure 4-22(a). MARS regression - actual and predicted *close_norm* values (*Case I*)

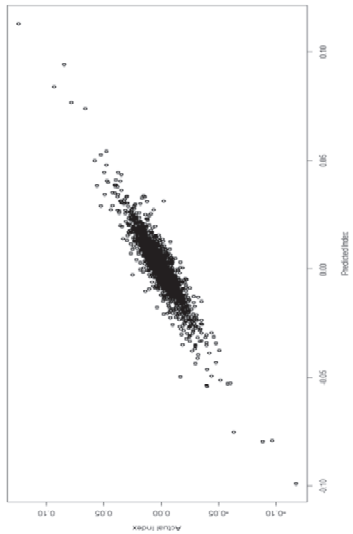

Figure 4-22(b). MARS regression – relationship between actual and predicted *close_norm* values (*Case I*)

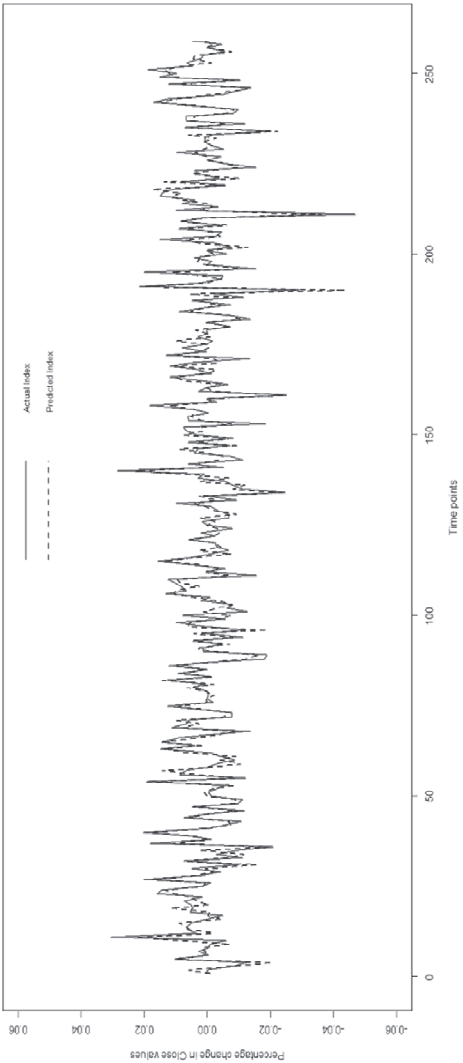

Figure 4-23. MARS regression - actual and predicted *close_norm* value (*Case II*)

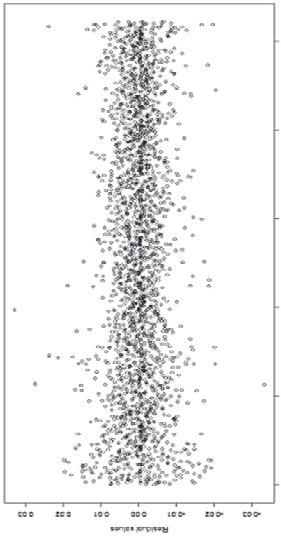

Figure 4-24(a). MARS regression – the residual plot (*Case I*)

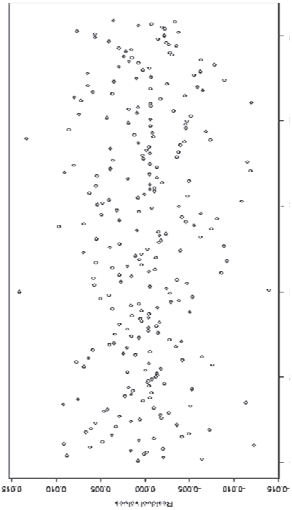

Figure 4-24(b). MARS regression – the residual plot (*Case II*)

TABLE 4-7. MARS REGRESSION RESULTS

Accuracy Metrics	Case I	Case II
	Training Case	Test Case
Correlation between actual and predicted values	0.92	0.88
Root Mean Squared Error (RMSE)	0.00532	0.0046
RMSE/mean of absolute values of the close values	58.15	64.97
Percentage of mismatch cases	14.94	16.98

Support Vector Machines: The Support Vector Machine is a very efficient machine learning technique invented by Vladimir Naumovich Vapnik. SVM regression can be used for both linear data and non-linear data. SVM searches for an optimal separating hyperplane or decision boundary that separates tuples of different classes. This hyperplane is determined so that it has the maximum marginal distance from its nearest data points (also called the *support vectors*), and hence is known as the *maximum marginal hyperplane* (MMH). Figure 4-25 illustrates the working principles of an SVM model.

For building the SVM regression models, we use the *svm* function defined in the *e1071* library of the *R* programming language. The model is built on a training dataset containing 2609 records. In training the model, 1886 support vectors are found. The model is used to predict the *close_norm* values for the training dataset in *Case I*. Figure 4-26 (a) depicts the plot of the actual and the predicted values of the *close_norm* for the training case. The relationship between the actual and the predicted *close_norm* values is shown in Figure 4-26(b). The RMSE of the model for *Case I* is 0.00635, and the ratio of the RMSE to the mean of the absolute values of the *close_norm* is 69.36. The correlation coefficient between the actual and the predicted *close_norm* is found to be 0.8857. Among 2609 records, 339 cases exhibit a mismatch in sign between the actual and the predicted *close_norm* values, yielding a mismatch percentage of 12.99.

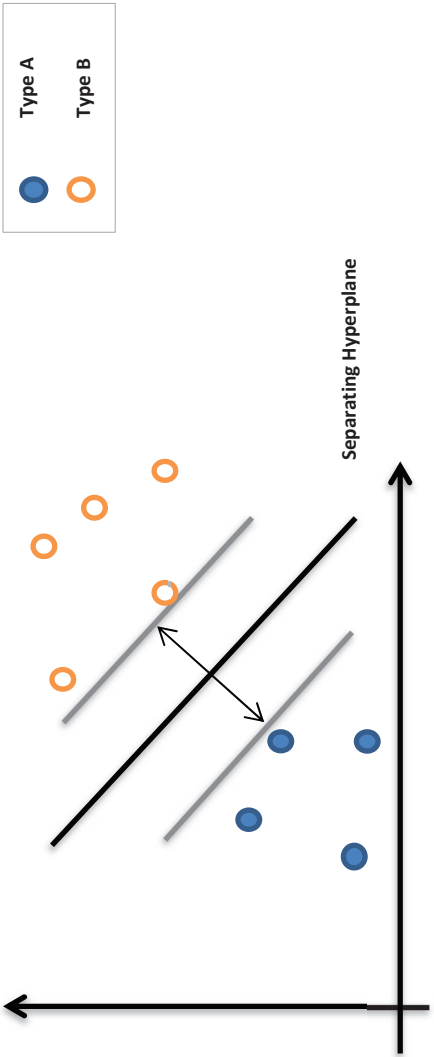

Figure 4-25. SVM model. A hyper-plane has been used for separating two classes of datapoints

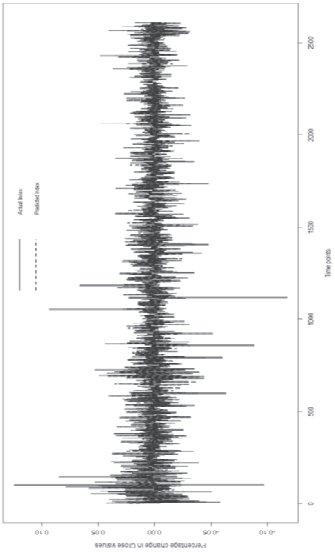

Figure 4-26(a). SVM regression- actual and predicted *close_norm* values (*Case I*)

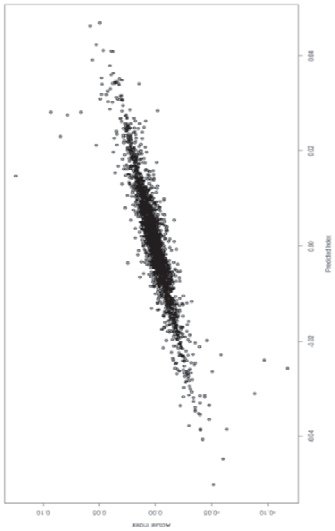

Figure 4-26(b). SVM regression - relationship between actual and predicted *close_norm* values (*Case I*)

In *Case II*, the SVM model is used to predict the *close_norm* values of the records in the test dataset using the model built in the *Case I*. test dataset. The model yields an RMSE value of 0.0050, and the ratio of the RMSE to the mean of the absolute values of *close_norm* is 70.84. Out of 259 cases, 49 cases exhibit a mismatch in sign between the actual and the predicted values of *close_norm*. Hence, 18.91 percent of the cases mismatch in sign. The correlation coefficient between the actual and the predicted values of the *close_norm* is 0.8567. Figure 4-27 depicts the actual and the predicted values of *close_norm* under *Case II* of the SVM regression. The residual plots for the SVM regression models in *Case I* and *Case II* are shown in Figure 4-28(a) and Figure 4-28(b), respectively. The residual values are random and do not exhibit any significant autocorrelation. Table 4-8 presents the performance results of the SVM regression models under *Case I* and *Case II*.

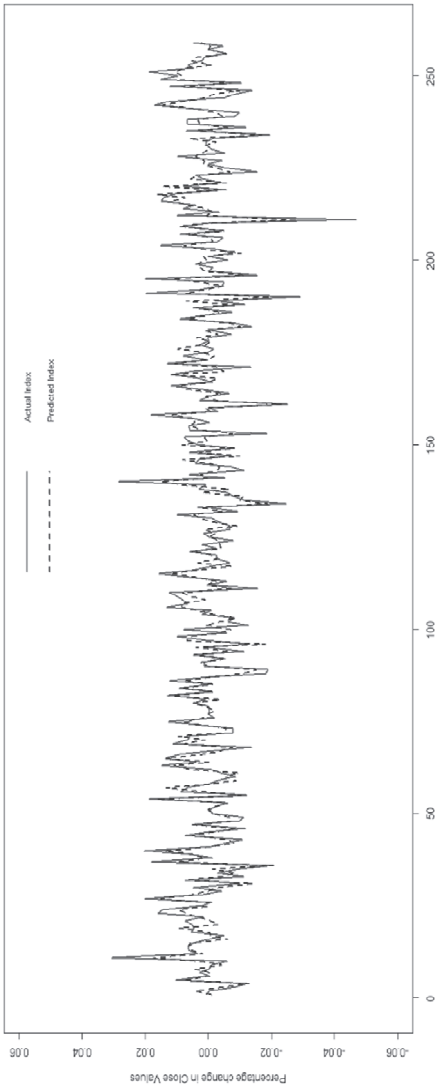

Figure 4-27. SVM regression - actual and predicted *close_norm* values (Case II).

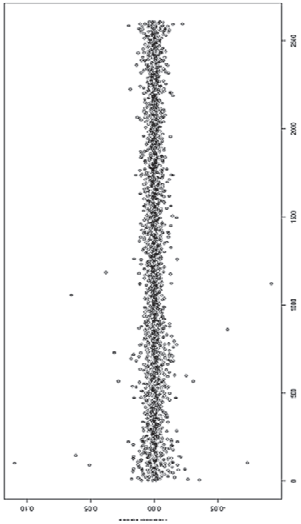

Figure 4-28 (a). SVM regression – the residual plot (*Case I*)

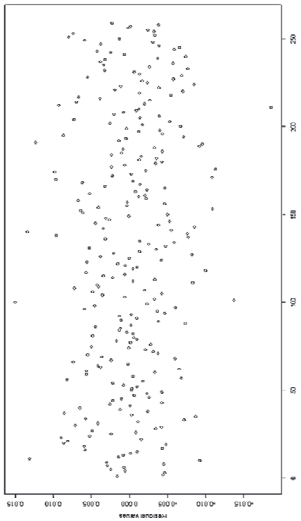

Figure 4-28 (b). SVM regression – the residual plot (*Case II*)

TABLE 4-8. SVM REGRESSION RESULTS

Accuracy Metrics	Case I Training Case	Case II Test Case
Correlation between actual and predicted values	0.89	0.86
Root Mean Squared Error (RMSE)	0.00635	0.00502
RMSE/mean of absolute values of the close values	69.36	70.84
Percentage of mismatch cases	12.99	18.91

We conclude that for the Indian IT sector, the random forest model has performed the best in the model training phase. In contrast, the boosting model has performed the best in the testing phase.

Classification Models: We now present the classification models proposed in this work. In the following, the design and performance of each model are discussed in detail.

Before we present the classification models, let us define some important metrics which are used for the evaluation and analysis of these models.

True negatives are those cases that belong to class 0 and have been correctly classified by a classifier model. *False positives*, on the other hand, are the cases of actual class 0 but misclassified by a classifier model as cases of class 1. False negatives are those instances that actually belong to class 1 but have been misclassified by a classifier model. Finally, the *true positives* are those cases that are of class 1 and have been correctly classified by a classifier model.

While there are numerous metrics for evaluating a classification model, the following six metrics are critical: (i) *sensitivity*, (ii) *specificity*, (iii) *positive predictive value*, (iv) *negative predictive value*, (v) *classification accuracy*, and (vi) *F1-score*. In the following, we define these metrics.

Sensitivity: The ratio of the number of true positives to the sum of the number of *true positives* (TP) and the *false negatives* (FN) is called the sensitivity of a model. It refers to the number of correctly classified positive cases. Sensitivity is also known as *recall* and is computed using (1).

$$Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives} = \frac{TP}{TP+FN} \quad (1)$$

Specificity: The ratio of the number of *true negatives* (TN) to the sum of the number of *true negatives* (TN) and *false positives* (FP) is known as the specificity of a model. It refers to the number of correctly classified negative cases. Specificity is computed using (2)

$$Specificity = \frac{\text{True Negatives}}{\text{True Negatives + False Positives}} = \frac{\text{TN}}{\text{TN+FP}} \quad (2)$$

Positive Predictive Values (PPV): The ratio of the number of *true positives* (TP) to the sum of *true positives* (TP) and *false positives* (FP) is called the PPV of a model. PPV returns the number of correctly classified positives cases out of the total number of positive cases. PPV is also known as *precision* and computed using (3).

$$PPV = \frac{\text{True Positives}}{\text{True Positives + False Positives}} = \frac{\text{TP}}{\text{TP+FP}} \quad (3)$$

Negative Predictive Values (NPV): The ratio of the number of *true negatives* (TN) to the sum of *true negatives* (TN) and the *false negatives* (FN) is known as the NPV of a model. NPV returns the number of correctly classified negative cases out of the total number of negative instances. NPV is computed using (4).

$$NPV = \frac{\text{True Negatives}}{\text{True Negatives + False Negatives}} = \frac{\text{TN}}{\text{TN+FN}} \quad (4)$$

Classification Accuracy: The classification accuracy of a model is defined as the ratio of the number of cases correctly classified by the model to the total number of cases in the dataset. It is computed using (5).

$$CA = \frac{\text{True Negatives + True Positives}}{\text{True Negatives + False Negatives + True Positives+False Positives}} = \frac{\text{TN+TP}}{\text{TN+TP+FN+FP}} \quad (5)$$

F1-score: The F1-score is of a model is computed as the *harmonic mean* of the *sensitivity* (i.e., the *recall*) and the PPV (i.e., the *precision*). It

is considered to be the most robust metric for a classification model. *F1-score* is computed using (6).

$$F1\text{-}score = \frac{2 * \text{Sensiitivity} * \text{PPV}}{\text{Sensitivity} + \text{PPV}} \tag{6}$$

The *lift curve* and the *ROC curve* are visualization-based metrics for the evaluation of a logistic regression classifier. The lift curve assigns probability values to the predicted class for each record. A steep lift curve signifies that the model effectively distinguishes between the records of the two different classes. The ROC curve shows how much specificity needs to be sacrificed for one unit gain of sensitivity in the model. A steep ROC implies that a minor specificity sacrifice is needed for a unit gain of sensitivity. We convert the response variable *close_norm* into a *categorical* type before building the classification models. This is achieved by the use of the function *as.factor* in R.

Logistic regression: It is a classification model that uses a *logit* function that converts the odds of an event by applying the natural logarithm to it. Each of the predicted outcomes is assigned with a probability value between 0 and 1, which signifies the credence of the model's classification ability. Since the odds of an event is, *odds* (event *A*) $=\frac{\text{probability (event } A \text{ occurring)}}{\text{probability (event } A \text{ not occurring)}}$; the *logit* function can be stated as *ln*(*odds*) (natural logarithm of odds). The values of the response variable are transformed into a discrete domain from a continuous domain. We convert the response variable *close_norm* into a categorical type of two labels – 0 or 1. The *close_norm* values which are less than zero are assigned with a class label of 0, and the other cases are assigned with a label of 1. The *glm* function in the R language is used for building the logistic regression model. The parameter *family* is set to "*binomial*", to indicate that the model is for a binary classification task involving two possible outcomes (classes). The *predict* function is used to predict the probability values of the cases. A threshold probability value of 0.5 is assigned. For a given record, the class that gets a probability value equal to or greater than the threshold becomes the class for that record. The *lift* curve and the *ROC* curve plot of the model are also analyzed to evaluate the performance of the model.

In *Case I*, the model is built on the training dataset, and its performance is evaluated on the training data. It is found that the model misclassifies 381 cases out of a total of 2609 records in the training dataset.

Figure 4-29(a) shows the lift curve for the logistic regression model in *Case I*. The steeper a lift curve is from its baseline, the better classification it indicates. Here we can see the model did a reasonably good job. Figure 4-29(b) represents the *receiver operating characteristics* (ROC) curve of the model for *Case I*. The red portion of the curve represents the actual 0's which are correctly classified, and the blue portion of the curve represents the actual 1's that are correctly classified. The misclassified cases are depicted by the yellow and green portions of the curve. The value of the *area under the curve* (AUC) for *Case I* is 0.9211.

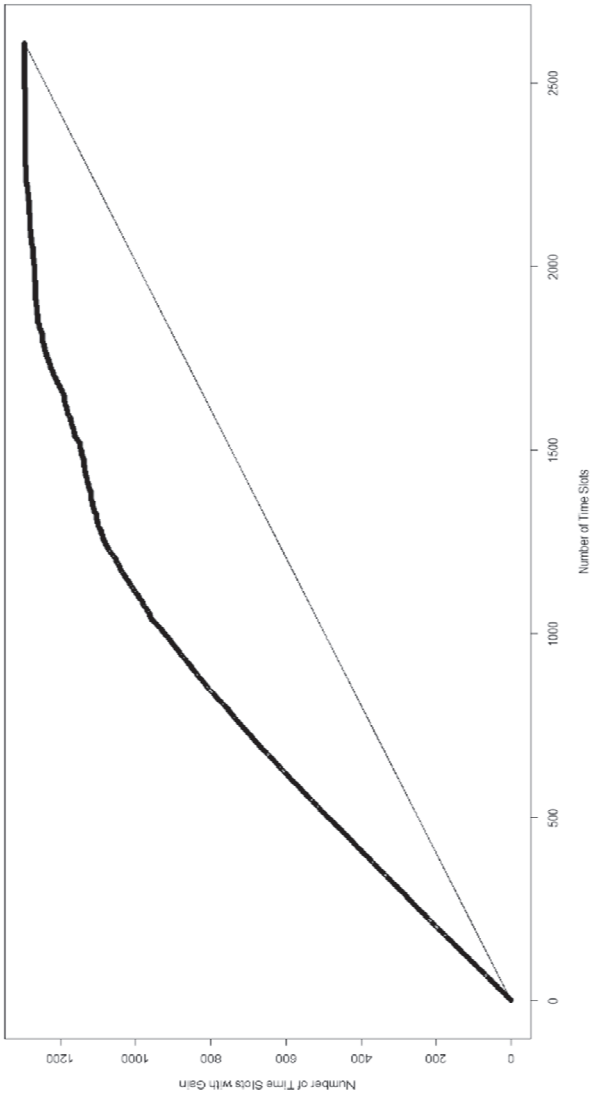

Figure 4-29(a). Logistic regression - lift curve (Case I)

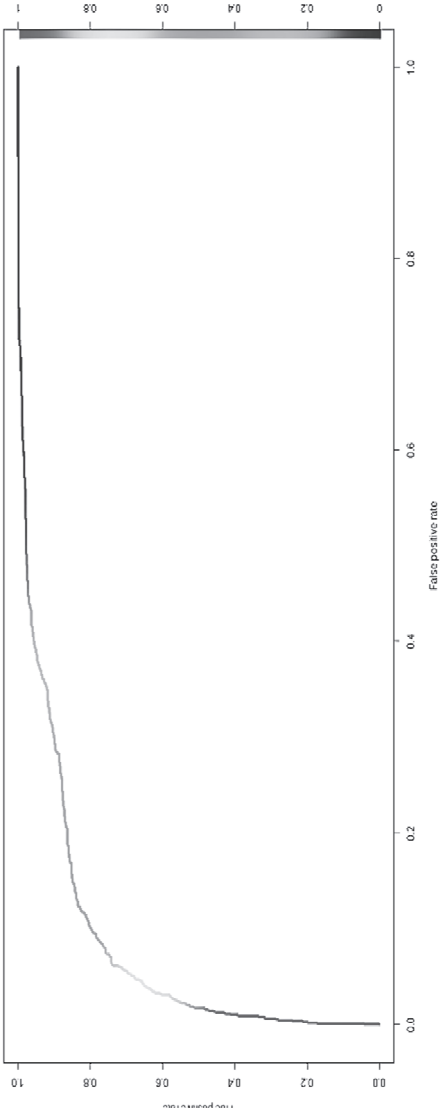

Figure 4-29 (b). Logistic regression - ROC curve (Case I)

In *Case II*, the logistic regression model misclassifies 36 cases out of 134 cases that belong to class 0. On the other hand, 23 cases belonging to class 1 are also misclassified by the model. Thus, the model misclassifies 59 cases out of 259 total records in the test dataset, leading to a classification error of 22%. Figure 4-30(a) shows the performance of the model in classifying the cases of the test dataset. The points which are on level 0 along the *y*-axis and situated to the right-hand side of the threshold probability value of 0.5 along the *x*-axis are the misclassified cases. Similarly, the points lying on level 1 along the *y*-axis and situated on the left-hand side of the threshold value of 0.5 along the *x*-axis are also misclassified cases. The lift curve of the model is shown in Figure 4-30(b). The area under the curve is computed using the *auc* function. The value of the AUC function for the model is found to be 0.8926. The ROC plot of the model is depicted in Figure 4-30(c).

Table 4-9 presents the performance results of the logistic regression model for *Case I* and *Case II*. It is evident that the performance of the model in *Case II* is inferior to that in *Case I* as the model predicted on the *out-of-sample* cases (i.e., the records of the test dataset) in *Case II*.

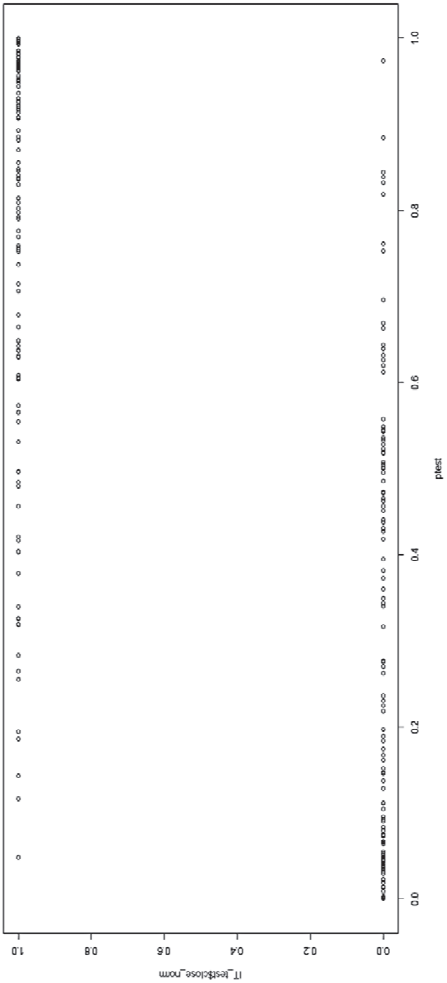

Figure 4-30(a). Logistic regression - actual and predicted probabilities (*Case II*).

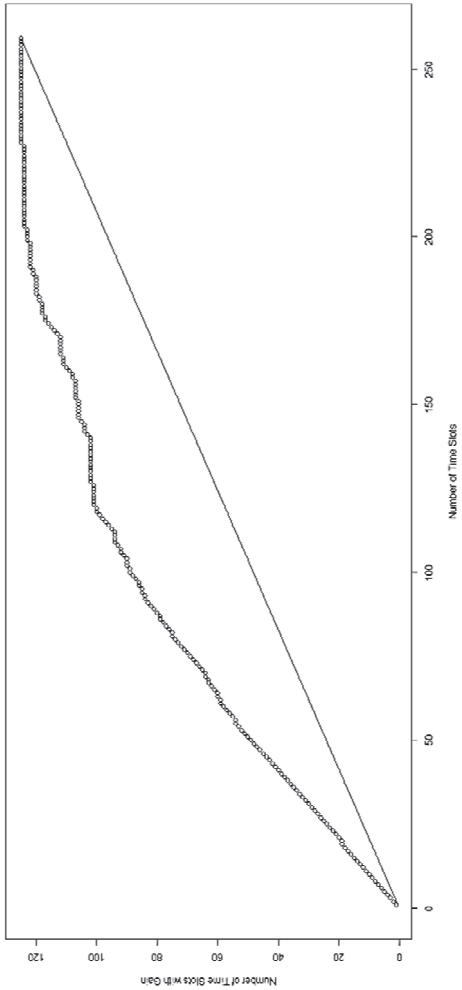

Figure 4-30(b). Logistic regression - lift curve (*Case II*)

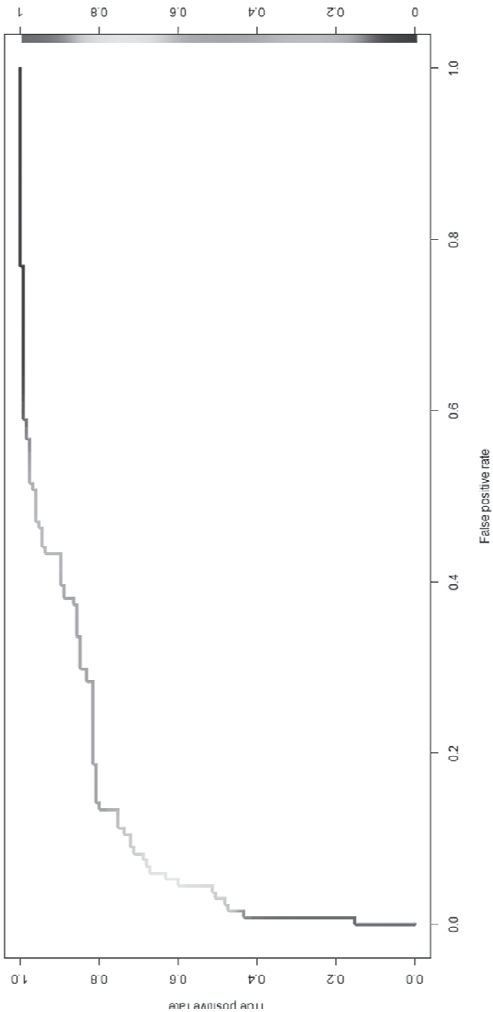

Figure 4-30(c). Logistic regression - ROC curve (*Case II*)

TABLE 4-9. LOGISTIC REGRESSION CLASSIFICATION RESULTS

Accuracy Metrics	Case I	Case II
Sensitivity	83.71	81.6
Specificity	87.00	73.13
Precision	86.45	73.91
NPV	84.41	80.99
Classification Accuracy	85.39	77.20
F1-score	85.00	77.50

Bagging classification*:* The word *bagging* is an acronym for *bootstrap aggregation*. It is an ensemble learning technique in machine learning. In the bagging method, the algorithm creates multiple subsets of the records in the training dataset, and the classification task is done based on the majority voting of the classifiers. We use the *bagging* function defined in the *ipred* library of the R language for building the bagging classification model. In *Case I*, the model is built using the 2609 records of the training dataset with the *nbag* parameter set to 25.

The bagging classification results show that out of 2609 cases, 386 cases are misclassified. Among 1313 cases that are actually of class 0, 187 cases are misclassified. On the other hand, 209 cases out of 1296 cases that are actually of class 1, are misclassified. The error of the model has turned out to be 0.1479. Figure 4-31(a) shows the plot of the actual and the predicted classes in *Case I* of the bagging classification model.

In *Case II*, the model is built using the training data, and the classification task is performed on the test dataset containing 259 records. The model misclassifies 46 cases out of 259 records. Out of 134 cases that are actually of 0 class, 20 cases are misclassified. On the other hand, 29 cases out of 125 actual 1s are also misclassified. The error value of the model has turned out to be 0.1776. Figure 4-31(b) shows the plot of the actual and the predicted classes for *Case II* of the bagging classification model.

The performance results of the bagging classification model are presented in Table 4-10.

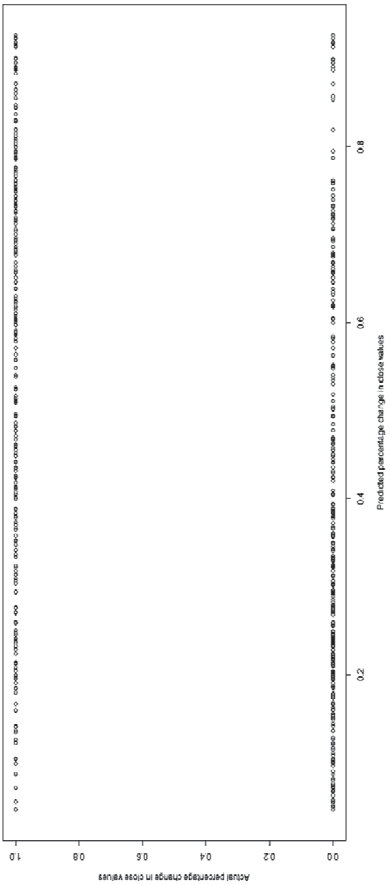

Figure 4-31(a). Bagging classification - actual and predicted probabilities (*Case I*)

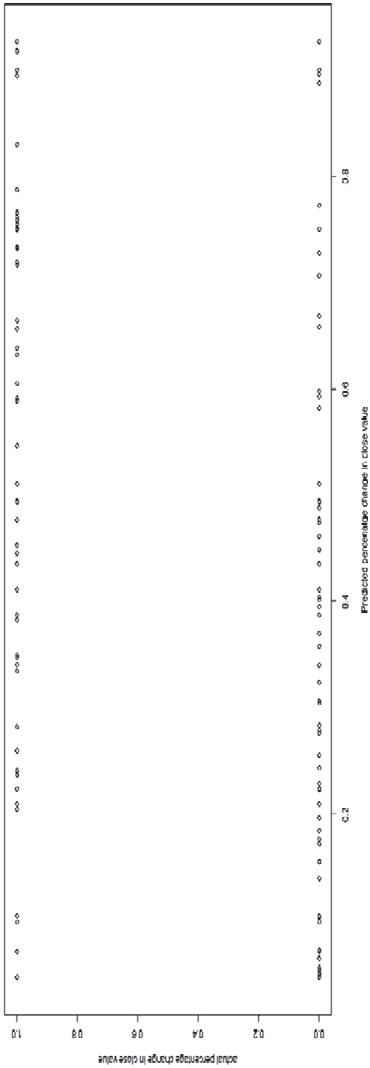

Figure 4-31(b). Bagging classification - actual and predicted probabilities (*Case II*).

TABLE 4-10. BAGGING CLASSIFICATION RESULTS

Accuracy Metrics	Case I	Case II
Sensitivity	85.26	77.6
Specificity	85.14	86.56
Precision	85.00	84.34
NPV	85.40	80.55
Classification Accuracy	85.20	82.23
F1-score	85.11	80.82

K-Nearest Neighbours classification: KNN classification method is an instance-based learning method. The predictions are made based on some similarity measures of the records. The *k*-value of the algorithm determines the number of neighboring records whose similarity with a given case is computed before the classification task is done. A small *k*-value may lead to overfitting of the model, while a large value will fail to exploit the local features effectively leading to poor performance in the classification task. An optimum value of k is determined based on the behavior of the given data. In *Case I*, the KNN model is built using the *knn* function defined in the *class* library of the R language. The predictor variables in the data are first normalized using the *min-max normalization* function. Different values of *k* are tried out and the performance of the model on various metrics is observed. Finally, the value of k = 3 is found to be yielding the best results. In the training dataset, there are 1313 records of class 0, while 1296 records are of class 1. In *Case I*, the model misclassifies 232 cases of class 0 and 241 cases of class 1. The classification accuracy of the model is found to be 81.87. The classification accuracy values (in percent) are 77.76, 75.62, and 72.74, respectively, for *k* values 5, 6, and 7. Figure 4-32 shows the plot of the actual and predicted cases for *Case I* of the KNN classification model.

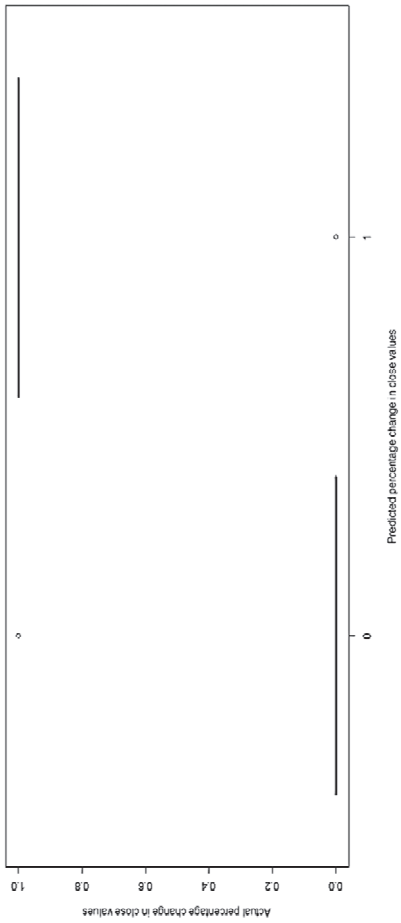

Figure 4-32. KNN - actual and predicted probabilities (*Case I*)

In *Case II*, the KNN model is used to predict the classes of the records of the test dataset. The test dataset has 259 records, out of which 134 cases are of class 0, and 125 cases are of class 1. The model misclassifies 70 cases of class 0 and 53 cases of class 1. The classification accuracy of the model in *Case II* is found to be 53.93 with k=3. Table 4-11 presents the performance results of the KNN classification model under *Class I* and *Class II*.

TABLE 4-11. KNN CLASSIFICATION RESULTS

Accuracy Metrics	Case I	Case II
Sensitivity	81.48	57.60
Specificity	82.33	47.76
Precision	81.98	50.70
NPV	81.83	54.70
Classification Accuracy	81.9	52.50
F1-score	81.59	53.93

Decision Trees*:* Decision tree classification works on the *classification and regression tree* (CART) algorithm. The algorithm creates a binary decision tree, in which the splitting at each node is carried out identifying the best predictor and the optimum value of that predictor so that the *information gain* in the model is maximized. In other words, the most appropriate split is considered based on the *goodness of fit* criteria after considering all the candidate predictors.

In *Case I*, the decision tree model is built using the *tree* function defined in the *tree* library of the R language. The response variable *close_norm* is converted into a categorical type with two levels, '0' and '1'. The model is built using the 2609 records of the training dataset with the value of the parameter *mincut* set as 1. It is observed that the predictor variables *high_norm* and *low_norm* are the two most significant predictors in the model. The classification model produces reasonably good results, yielding an *F1*-score of 83.72 and a CA of 84.09. Figure 4-33 shows the decision tree model in *Case I*.

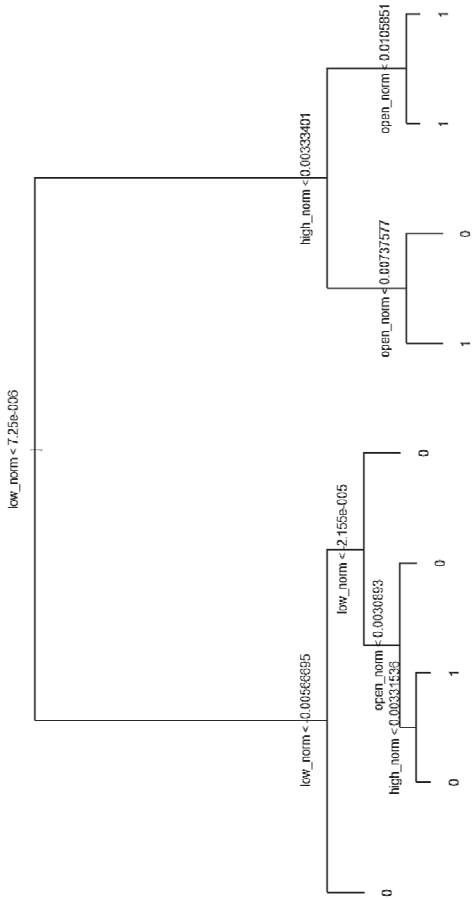

Figure 4-33. The Decision tree classification model (*Case I*).

In *Case II*, the model is built using the 2609 records of the training dataset with a *mincut* set as 1. Again, it is observed that the predictors *high_norm* and *low_norm* are the two most significant variables contributing to the model. The *F1*-score and the *CA* value of the model in *Case II* are found to be 78.94 and 81.08, respectively. Table 4-12 shows the performance results of the decision tree classifier model in *Case II*.

TABLE 4-12. DECISION TREE CLASSIFICATION RESULTS

Accuracy Metrics	Case I	Case II
Sensitivity	82.4	78.4
Specificity	85.75	83.58
Precision	85.09	81.66
NPV	83.16	80.57
Classification Accuracy	84.09	81.08
F1-score	83.72	78.94

Artificial neural network: We explained the working principle of an artificial neural network for the regression model. Here, we describe how we build the neural network classification model for predicting the stock index movement pattern. As mentioned earlier, a neural network consists of three types of layers; the input layer, one or more hidden layers, and the output layer. The number of input nodes is the same as the number of predictor variables in the data. The data pass through different layers over the links having different weights associated with them. These weighted inputs are adjusted with biases and are continued to be fed through a certain number of hidden layer nodes. The number of hidden layers used depends on the complexity of the data. Increasing the number of hidden layers increases the complexity of the model. Finally, the nodes at the output layer produce the output of the network. For building the ANN classification model, first, the predictors are normalized using the *min-max normalization* method. The normalization needs to be done to scale down all the predictors to the same range of values [0, 1]. The normalized predictors are used for building the model. However, the response variable, *close_norm,* is not normalized.

We use the *neuralnet* function defined in the *neuralnet* library in the R language for building the ANN classification model. In *Case I*, the ANN model is built with two hidden layers using the value of the parameter *hidden* as 2. The value of the parameter *stepmax* is set to 10^6 and the argument *linear.output* is set to *false*. The model misclassifies 149 cases of class 0 and 222 cases of class 1. The classification error of the model is 0.4979. The ANN model architecture for *Case I* is shown in Figure. 4-34.

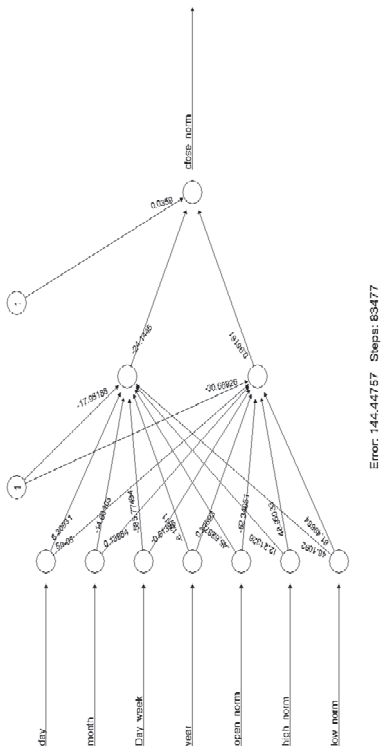

Figure 4-34. The architecture of the ANN classification model (*Case I*).

In *Case II*, the model built under *Case I* is used for predicting the class labels of the records of the test dataset. The test dataset consists of 259 records. The performance results of the ANN regression model are presented in Table 4-13.

Figure 4-35(a) depicts the plot of the actual versus the predicted cases for the ANN classification model in *Case I*. The same plot for *Case II* of the ANN classifier is shown in Figure 4-35(b).

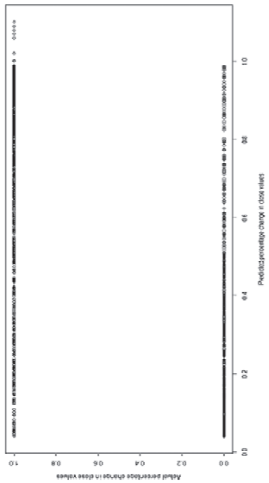

Figure 4-35 (a). ANN classification – actual and predicted probabilities (*Case I*)

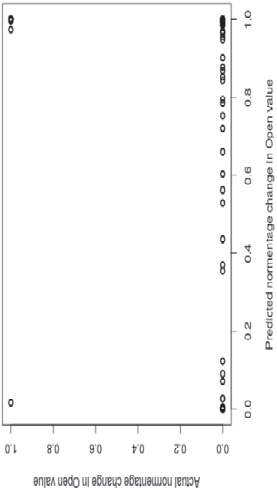

Figure 4-35 (b). ANN classification - actual and predicted probabilities (*Case II*)

TABLE 4-13. ANN CLASSIFICATION RESULTS

Accuracy Metrics	Case I	Case II
Sensitivity	82.87	99.2
Specificity	87.71	17.16
Precision	88.03	37.01
NPV	83.98	95.83
Classification Accuracy	81.94	56.75
F1-score	84.49	53.90

From Table 4-13, it is observed that in *Case II*, the sensitivity is very high. It implies that the model correctly classified a large proportion of the class 1 cases. On the other hand, a very low value of specificity indicates that the model failed to classify the majority of the class 0 cases. Since the dataset is balanced, we can consider CA as a useful metric for the ANN regression models.

Boosting: Boosting classification is an ensemble method of classification with the iterative learning approach from the data. It iterates through the records in the training dataset and attempts to correctly classify the records in the current round, which were wrongly classified in the previous iteration. The algorithm assigns more weight to the incorrectly classified observations, and the classifiers which can classify them correctly in the next round are assigned higher weights. A weighted majority voting mechanism is applied at the time of final classification. The boosting classifiers often produce a hundred percent classification accuracy during the training phase.

The boosting classifiers are built using the *boosting* function defined in the *adabag* library in the R programming language. Since boosting works on a random selection of subsets of the records in the training dataset, we use the *set.seed* function so that identical results are produced in each round of invocation of the model. In our case, the model is built using 2609 records in the training dataset.

In *Case I*, the model produces 100% accuracy on all metrics. Boosting models achieve 100% percent accuracy during training if a sufficient number of iterations are allowed in the model building process. The algorithm learns from the error in the previous round and correctly classifies some of the cases in the next round, which were wrongly classified in the

previous round. The improvement continues till all records are correctly classified.

In Case II, the boosting model which was built in *Case I* is used in predicting the class labels for the test dataset records. The test dataset has 259 total records. The model misclassifies 20 cases out of 121 actual 0s. On the other hand, 33 cases out of 138 actual 1s are also misclassified. The CA of the boosting model for *Case II* is 79.53%, and the *F1*-score is 79.84%. Figure 4-36 shows the actual versus the predicted classes plot for *Case 1* of the boosting regression model. The performance of the boosting classification model is presented in Table 4-14.

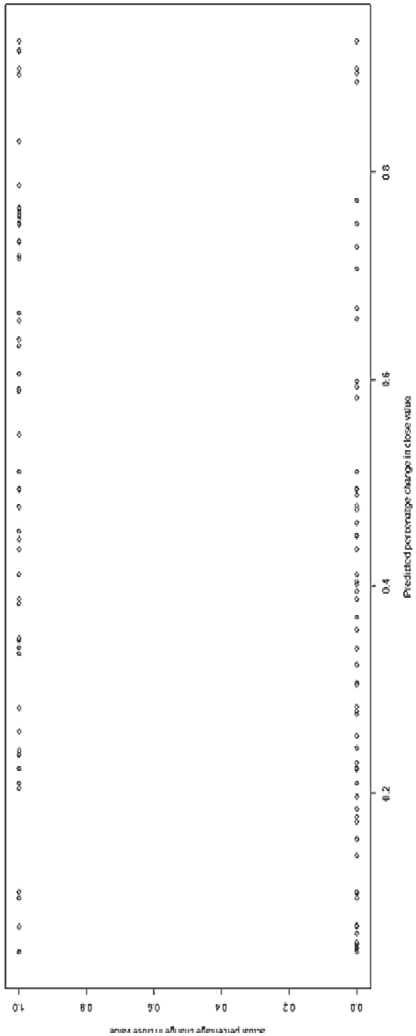

Figure 4-36. Boosting classification - actual and predicted probabilities (*Case I*)

TABLE 4-14. BOOSTING CLASSIFICATION RESULTS

Accuracy Metrics	Case I	Case II
Sensitivity	100	76.08
Specificity	100	83.47
Precision	100	84.0
NPV	100	75.37
Classification Accuracy	100	79.53
F1-score	100	79.84

Random Forest: Random Forest classification introduces a high level of randomness in creating an ensemble of decision trees, and randomly splitting the nodes in the trees. The algorithm searches for the best feature among a randomly selected subset of features from all the decision trees built during the training. Simple majority-based voting is used for the final classification of the records. The random forest classification models are built using the *randomForest* function defined in the *randomForest* library in the R language. In *Case I*, the model is built using the 2609 records of the training dataset. The response variable *close_norm* is converted into a categorical type using the *as.factor* function. The model produces an *out-of-bag* (OOB) error of 15.6%. It uses 2 splits for each node and produces 500 decision trees. In *Case I*, the random forest model misclassifies 3 cases out of 1313 actual 0 cases. On the other hand, out of 1296 actual 1 cases, only one case is misclassified.

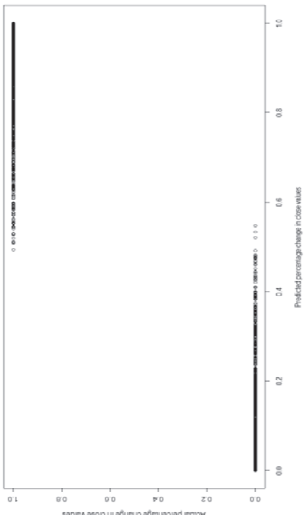

Figure 4-37(a). Random forest classification - actual and predicted probabilities (*Case I*)

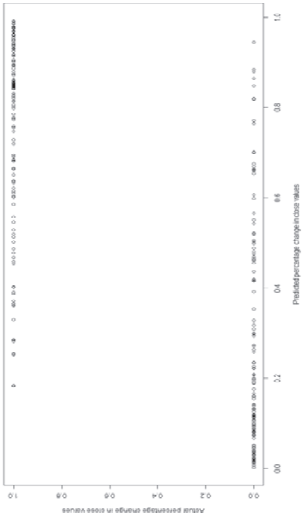

Figure 4-37(b). Random forest classification - actual and predicted probabilities (*Case II*)

In *Case II*, the same model built under *Case I* is used in predicting the class for the records in the test dataset of 259 cases. Out of 129 actual 0 cases in the test dataset, 21 cases are misclassified. On the other hand, out of 130 actual 1s, 26 cases are misclassified. Figure 37(a) and Figure 37(b) show the actual and the predicted probabilities for the *close_norm* values for the random forest regression under *Case I* and *Case II*. Table 4-15 shows the performance results of the random forest classification models.

TABLE 4-15. RANDOM FOREST CLASSIFICATION RESULTS

Accuracy Metrics	Case I	Case II
Sensitivity	99.95	80.00
Specificity	99.77	83.72
Precision	99.76	83.20
NPV	99.92	80.59
Classification Accuracy	99.84	81.85
F1-score	99.85	81.56

Support Vector Machines*:* We build the SVM classification models using the *ksvm* function defined in the *kernlab* library in the R language. For *Case I*, the SVM model is built on the training data using the parameter *kernel* set to *vanilladot*. The model produces an overall classification accuracy of 81 percent. The model misclassifies 171 cases out of 1313 actual 0s. Again, 250 records out of 1296 cases are misclassified which are actually of class 1. In *Case II*, the built model in *Case I* is used in predicting the classes for the 259 records in the test dataset. It is observed that the model misclassifies 24 cases out of 134 actual 0's. On the other hand, out of 125 actual 1s, 22 cases are misclassified. Table 4-16 presents the performance results for the SVM classification models. Figure 38(a) and Figure 38(b) exhibit the actual and the predicted probabilities for the *close_norm* values by the SVM classification model under *Case I* and *Case II*, respectively.

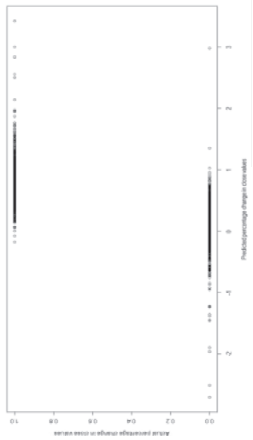

Figure 4-38(a). SVM classification – actual and predicted probabilities (*Case I*)

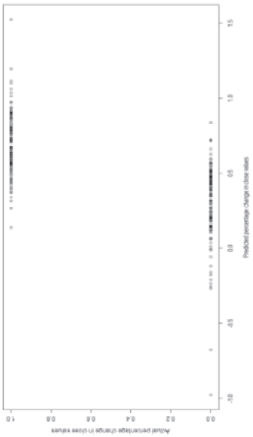

Figure 4-38(b). SVM classification - actual and predicted probabilities (*Case II*)

TABLE 4-16: SVM CLASSIFICATION RESULTS

Accuracy Metrics	Case I	Case II
Sensitivity	83.87	75.36
Specificity	85.75	82.64
Precision	85.32	83.2
NPV	84.34	74.62
Classification Accuracy	84.82	78.76
F1-score	84.58	79.08

Deep Learning Model

Long-and-Short-Term Memory (LSTM) Network: The LSTM architecture is a modified version of the architecture of the *recurrent neural networks* (RNN). In RNNs, the output at the current state depends on the input at the current state and the state information of the previous state. Traditionally. RNNs are considered to be very efficient for managing sequential data. However, RNNs suffer from a drawback, known as the *vanishing and exploding gradients problem* (Geron, 2019). RNNs cannot retain the long-term dependencies during the backpropagation process for adjusting the weights through previous timestamps. Thus, when the gradients become extremely high due to the multiplicative nature of weights in an RNN network, the convergence of the weights does not take place, and the optimal values of the weights are never achieved. This phenomenon is known as the *exploding gradient problem*. On the other hand, if the magnitudes of the link weights are less than 1, successive multiplication at different layers leads to a decrease in the gradient, resulting in a pre-mature stopping of the learning in the network. This situation is known as the *vanishing gradient problem*, which leads to sub-optimal weights of the links in the network (Hochreiter & Schmidhuber, 1997). While it is possible to eliminate the exploding gradient problem using the *truncated backpropagation through time (TBTT)* algorithm, the architecture and working principles of LSTM networks can prevent both vanishing and exploding gradient problems from occurring in a deep neural network. Therefore, LSTM is more effective in handling sequential data and temporal data.

We build the LSTM model using the daily index records of the Indian IT sector listed in the NSE, India, from December 29, 2008, to December 28, 2018. The trained model is then tested using the model to predict the daily *close* values of the IT sector index from December 31, 2018, to December 27, 2019. The raw sector index dataset consists of five attributes: (i) *date*, (ii) *open*, (iii) *low*, (iv) *high*, and (v) *close*. Figure 4-39 depicts the plot of the four attributes of the stock index data over the entire period from December 28, 2008, to December 27, 2019. We build the LSTM model using the univariate data of the *close* values of the index. In other words, the model predicts the future values of the *close* index based on the past *close* values. The model uses the *close* values of the last week (i.e., the most recent five observations) as the input and predicts the *close* values for the five days of the next week. The LSTM model is built using the Python programming language on the Tensorflow and the Keras frameworks. The *mean absolute error* (MAE) is used as the accuracy metric for the model, and the Adam optimizer is used for optimizing the performance of the *gradient descent algorithm*. For understanding the amount of error committed by the model, we compute a new metric based on the ratio of the RMSE to the mean value of the *close* index in the test dataset. This metric shows the magnitude of the RMSE's contribution (as an error) to the mean value of the response variable, the *close* price of the stock.

We use the *Adam* optimizer because of its ability to use adaptive learning rates for the parameters of the gradient descent. Like momentum optimization, *Adam* remembers the exponentially decaying average of its past gradients and requires much less effort to tune the parameters. We train the model using a different number of epochs with varying batch sizes. The optimum number of epochs is found to be 80 with a batch size of 16. Using the Tensorflow and the Keras frameworks, the training and the validation of the model are performed. After completing 80 epochs, the RMSE and the correlation coefficient between the actual and the predicted *close* values are found to be 1.65 and 0.878, respectively. Figure 4-40 and Figure 4-41 exhibit the convergence of the training and the validation loss and accuracy, respectively, for different epoch values, batch size remaining constant at 16.

Initially, 50 epochs are used to train the model with a batch size of 16, which yields a correlation coefficient (between the actual and the predicted values *close* values) value of 0.80. The value of the RMSE is found to be 2.87. In this case, the number of nodes used in the LSTM layer and the dense layer in the model are 200 and 50, respectively. In the second configuration of the model too, we use the same number of nodes in the LSTM model. In the second configuration, the model yields the correlation coefficient and the RMSE values as 0.847 and 1.89, respectively. The choice of 80 epochs,

batch size of 16, 200 nodes in the LSTM layer, and 100 nodes in the dense layer have produced the best results. With these optimal settings, the model yields a correlation coefficient of 0.875 and an RMSE of 1.562.

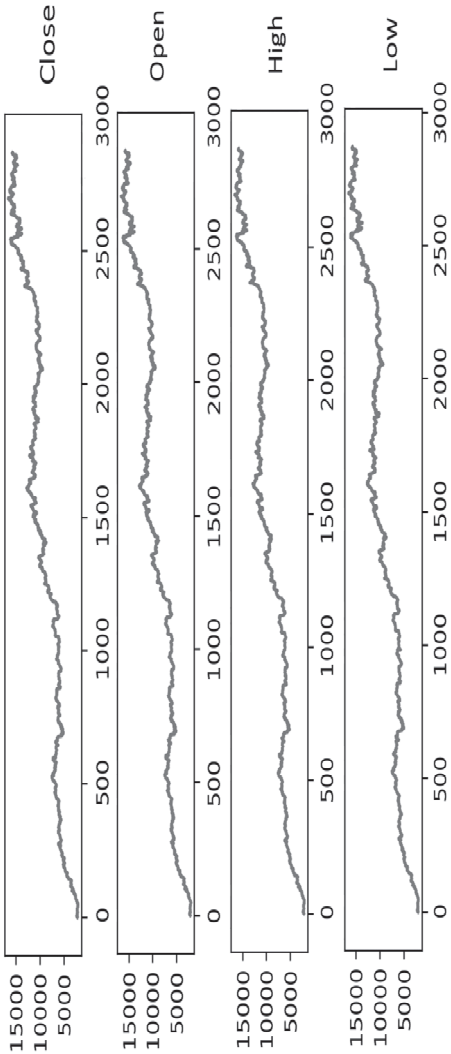

Figure 4-39. The Indian IT sector daily index values for the training dataset for the LSTM

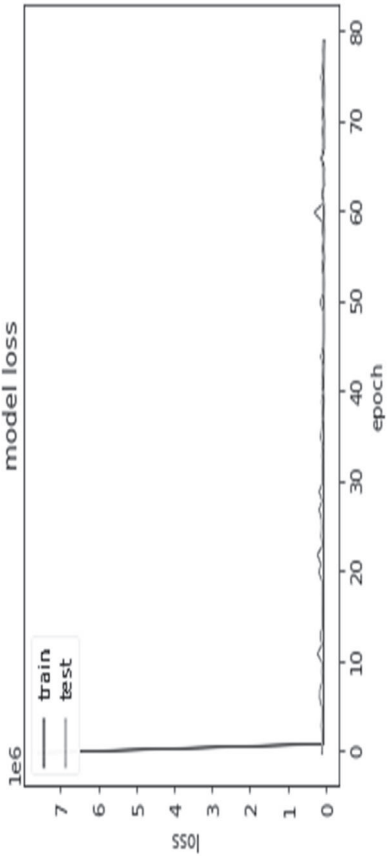

Figure 4-40. The convergence of the training and validation loss for the LSTM model

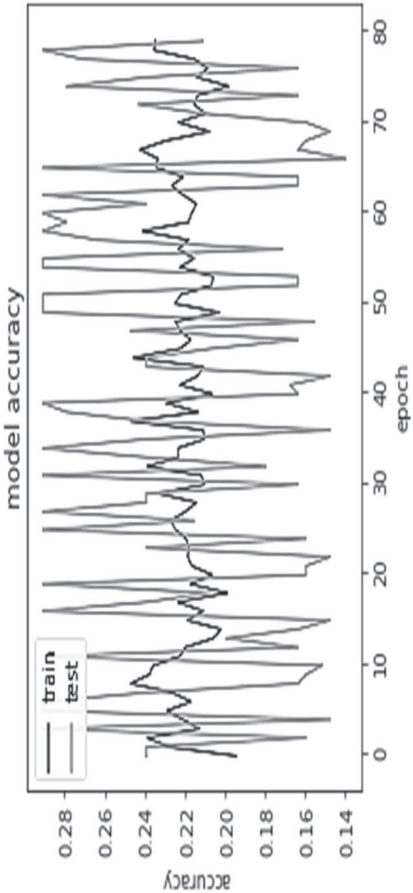

Figure 4-41. Training and validation accuracies of the LSTM model

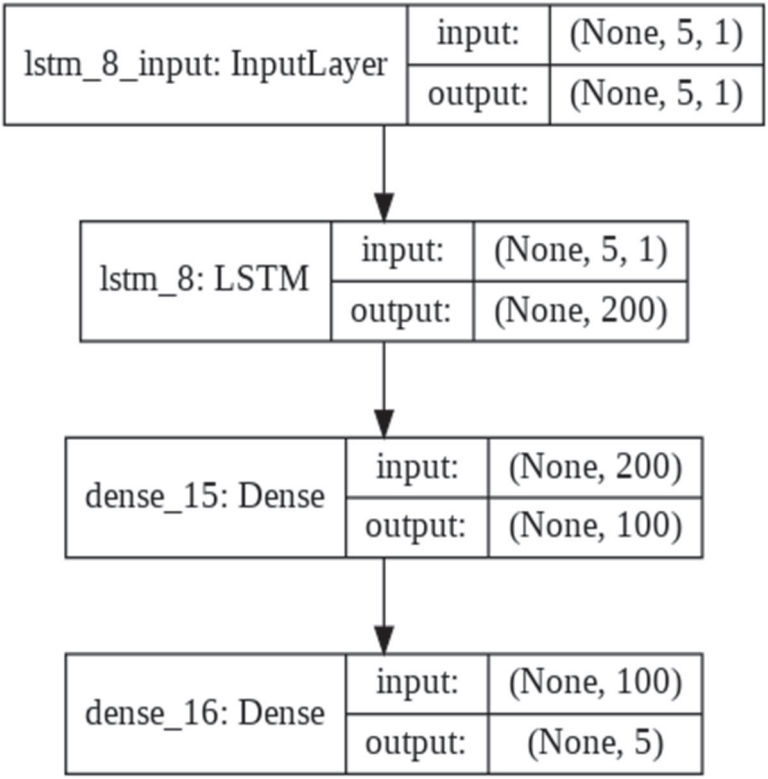

Figure 4-42. The architecture of the LSTM regression model

Figure 4-42 presents the architecture of the LSTM model. The schematic is obtained using the *plot_model* function defined in the Keras framework. The schematic shows the data shape and the layer architectures of the LSTM model. The input layer has a data shape of (5, 1), indicating that 5 values of the last week's *close* index values are passed as the input to the model. The second value, 1, represents the number of attributes (i.e., *close*) used in the input. The output of the input layer is passed on to the LSTM layer that consists of 200 nodes, which extracts 200 features from the 5 input values. The output of the LSTM layer is further passed on to a dense layer of 100 nodes. The dense layer has 5 nodes at the output layer, producing the predicted 5 values for the next week.

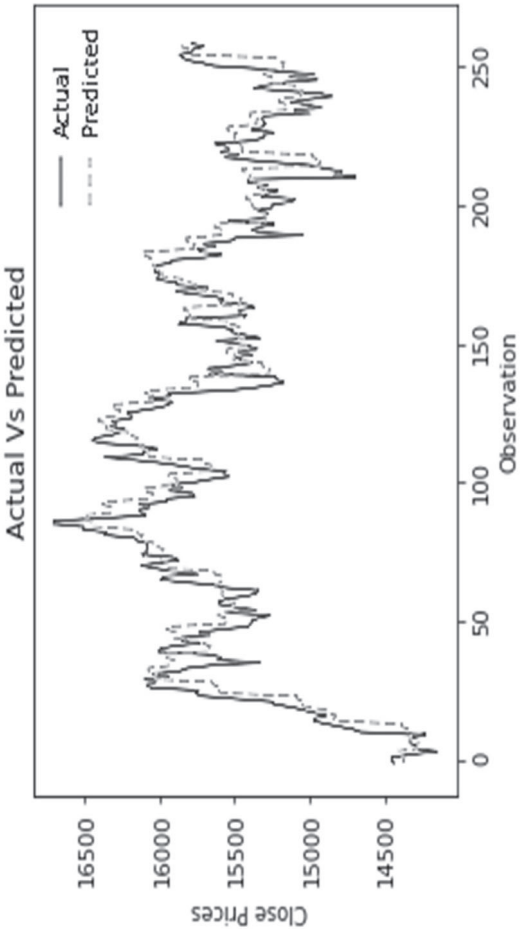

Figure 4-43. The actual vs the predicted values of the *close* index values for the LSTM model

Figure 4-43 depicts the plot of the actual and the predicted *close* index values for the records in the test dataset. It is evident that the predicted values very closely follow the pattern of the actual *close* index values. Table 4-17 presents the performance results of the LSTM regression model. The

model is executed over 10 rounds. For each round, the following observations are recorded: (i) the RMSE values for each day in a week, (ii) the *correlation coefficient* between the actual and the predicted *close* values, and (iii) the time required for the execution of the round (in second). The RMSE values are found to increase from Monday to Friday. The RMSE values for the five days of the week are found to be 172.0, 235.4, 249.1, 285.7, and 318.5. The execution of the model is done on the *Google Collaboratory* (Google Colab) platform. The Google Colab is a computing platform for machine learning and artificial intelligence-related research. In Table 4-17, we present the values of the ratio of the RMSE to the mean *close* values for the records in the test dataset. It is observed that the average value of the ratio of the RMSE to the mean *close* value increases from Monday to Friday. The average correlation coefficient between the actual and the predicted *close* values is 0.875.

TABLE 4-17. THE PERFORMANCE RESULTS OF THE LSTM REGRESSION MODEL

Round	Mon	Tue	Wed	Thu	Fri	Correlation	Time
1	0.0102	0.0148	0.0161	0.0187	0.0202	0.871	146.38
2	0.0091	0.0145	0.0148	0.0177	0.0213	0.877	139.52
3	0.0097	0.0143	0.0144	0.0169	0.0194	0.879	136.62
4	0.0093	0.0148	0.0161	0.0171	0.0191	0.880	138.45
5	0.0109	0.0144	0.0135	0.0167	0.0197	0.877	146.94
6	0.0108	0.0146	0.0133	0.0173	0.0211	0.879	144.56
7	0.0102	0.0144	0.0142	0.0187	0.0201	0.869	147.79
8	0.0096	0.0142	0.0163	0.0179	0.0199	0.879	146.64
9	0.0111	0.0156	0.0155	0.0189	0.0202	0.859	146.27
10	0.0107	0.0163	0.0182	0.0184	0.0203	0.879	144.68
Mean	**0.0102**	**0.0148**	**0.0152**	**0.0178**	**0.0201**	**0.875**	**143.79**

References

Adebiyi, A. A., Adewumi, A. O. and Ayo, C. K. (2014) "Stock price prediction using the ARIMA model", *Proceedings of the International Conference on Computer Modelling and Simulation (UKSim-AMSS)*, March 26-28, Cambridge, UK, pp. 105 -111. DOI: 10.1109/UKSim.2014.67.

Basalto, N., Bellotti, R., De Carlo, F., Facchi, P. and Pascazio, S. (2005) "Clustering stock market companies via chaotic map synchronization", *Physica A: Statistical Mechanics and its Applications*, Vol. 345, No. 1-2, pp. 196-206. DOI:10.1016/j.physa.2004.07.034.

Basu, S. (1983) "The relationship between earnings yield, market value and return for NYSE common stocks: Further evidence", *Journal of Financial Economics*, Vol.12, No.1, pp. 129-156. DOI: 10.1016/0304-405X(83)90031-4.

Brownlee, J. (2017) *Introduction to Time Series Forecasting with Python*, Jason Brownlee Publications. Available Online at: https://machinelearningmastery.com/introduction-to-time-series-forecasting-with-python (Accessed on February 25, 2021)

Cakra, Y. E. and Trisedya, B. D. (2015) "Stock price prediction using linear regression based on sentiment analysis", *Proceedings of International Conference on Advanced Computer Science and Information Systems* (*ICACSIS*), October 10-11, Depok, Indonesia, pp. 147-154. DOI: 10.1109/ICACSIS.2015.7415179.

Chen, A.-S., Leung, M. T. and Daouk, H. (2003) "Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index", *Operations Research in Emerging Economics*, Vol. 30, No. 6, pp. 901– 923. DOI:10.1016/S0305-0548(02)00037-0.

Chui, A. and Wei, K. (1998) "Book-to-market firm size, and the turn of the year effect: Evidence from Pacific basin emerging markets", *Pacific Basin Finance Journal*, Vol. 6, No. 3-4, pp. 275-293. DOI:10.1016/S0927-538X(98)00013-4.

de Faria, E., Albuquerque, M. P., Gonzalez, J., Cavalcante, J. and Albuquerque, M. P. (2009) "Predicting the Brazilian stock market through neural networks and adaptive exponential smoothing methods", *Expert Systems with Applications*, Vol. 36, No. 10, pp. 12506-12509. DOI. 10.1016/j.eswa.2009.04.032.

Dutta, G., Jha, P., Laha, A. & Mohan, N. (2006) "Artificial neural network models for forecasting stock price index in the Bombay Stock

Exchange", *Journal of Emerging Market Finance*, Vol. 5, No. 3, pp. 283-295. DOI: 10.1177/097265270600500305.

Enke, D., Grauer, M. and Mehdiyev, N. (2011) "Stock market prediction with multiple regression, fuzzy type-2 clustering, and neural networks", *Procedia Computer Science*, Vol. 6, pp. 201-206. DOI: 10.1016/j.procs.2011.08.038.

Fama, E. F. and French, K. R. (1995) "Size and book-to-market factors in earnings and returns", *Journal of Finance*, Vol. 50, No. 1, pp. 131-155. DOI: 10.1111/j.1540-6261.1995.tb05169.x.

Geron, A. (2019) *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd Edition, ISBN: 978-1492032649, O'Reilly Media Inc, USA.

Hanias, M., Curtis, P. and Thalassinos, J. (2012) "Time series prediction with neural networks for the Athens Stock Exchange indicator", *European Research Studies Journal*, Vol. 15, No. 2, pp. 23-32. DOI: 10.35808/ersj/351.

Hegazy, O., Soliman, O. S., Salam, M. A. (2013). "A machine learning model for stock market prediction", *International Journal of Computer Science and Telecommunications*, Vol. 4, No. 12, pp. 17-23.

Ivanovski, Z., Ivanovska, N. and Narasanov, Z. (2016) "The regression analysis of stock returns at MSE", *Journal of Modern Accounting and Auditing*, Vol. 12, No. 4, pp. 217-224. DOI: 10.17265/1548-6583/2016.04.003.

Jaffe, J., Keim, D. B. and Westerfield, R. (1989) "Earnings yields, market values, and stock returns", *The Journal of Finance*, Vol. 44, No. 1, pp. 135 - 148. DOI:10.1111/j.1540-6261.1989.tb02408.x.

Jammalamadaka, S. R., Qui, J. and Ning, N. (2019) "Predicting a stock portfolio with multivariate Bayesian structural time series model: Do news or emotions matter?", *International Journal of Artificial Intelligence*, Vol. 17, No. 2, pp. 81-104.

Jarrett, J. E. and Kyper, E. (2011) "ARIMA modeling with intervention to forecast and analyze Chinese stock prices", *International Journal of Engineering Business Management*, Vol. 3, No. 3, pp. 53 - 58. DOI: 10.5772/50938.

Khan, U., Aadil, F., Ghaznfar, M., Khan, S., Metawa, N., Muhammad, K., Mehmood, I. and Nam, Y. (2018) "A robust-regression-based stock exchange forecasting and determination of correlation between stock markets", *Sustainability*, Vol. 10, No. 3702. DOI: 10.3390/su10103702.

Ma, J. and Liu, I. (2008) "Multivariate nonlinear analysis and prediction of Shanghai stock market", *Discrete Dynamics in Nature and Society*, Vol. 2008, Article ID: 526734. DOI: 10.1155/2008/526734.

Mehtab, S. and Sen, J. (2021a) "A time series analysis-based stock price prediction using machine learning and deep learning models", *International Journal of Business Forecasting and Marketing Intelligence (IJBFMI)*, Vol 6, No 4, pp. 272 – 335. DOI: 10.1504/IJBFMI.2020.115691.

Mehtab, S. and Sen, J. (2021b) "Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models", In: Sahoo, J. P., Tripathy, A. K., Mohanty, M., Li, K. C., and Nayak, A. K. (eds), Advances in Distributed Computing and Machine Learning. Lecture Notes in Networks and Systems, Vol 202, pp. 405-423, Springer, Singapore. DOI: 10.1007/978-981-16-4807-6_39.

Mehtab, S., Sen, J. and Dutta, A. (2021c) "Stock price prediction using machine learning and LSTM-based deep learning models", In: Thampi, S. M., Piramuthu, S., Li, K. C., Beretti, S., Wozniak, M., and Singh, D. (eds), *Machine Learning and Metaheuristics Algorithms and Applications*. *SoMMA 2020*. Communications in Computer and Information Science, Vol 1366, pp. 88-106, Springer, Singapore. DOI: 10.1007/978-981-16-0419-5_8.

Mehtab, S., Sen, J. and Dasgupta, S. (2020a) "Robust analysis of stock price time series using CNN and LSTM-based deep learning models", *Proceedings of the 4th IEEE International Conference on Electronics, Communication and Aerospace Technology (ICECA'20)*, November 5-7, 2020, Coimbatore, India, pp. 1481-1486. DOI: 10.1109/ICECA49313.2020.9297652.

Mehtab, S. and Sen, J. (2020b) "Stock price prediction using CNN and LSTM-based deep learning models", *Proceedings of the IEEE International Conference on Decision Aid Sciences and Applications (DASA'20)*, November 8-9, 2020, Sakheer, Bahrain, pp. 447-453. DOI: 10.1109/DASA51403.2020.9317207.

Mehtab, S. and Sen, J. (2020c) *Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models*, Technical Report, No: NSHM_KOL_2020_SCA_DS_1. DOI: 10.13140/RG.2.2.14022.22085/2.

Mehtab, S. and Sen, J. (2020d) "Stock price prediction using convolutional neural networks on a multivariate time series", *Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence (NCMLAI'20)*, February 1-2, 2020, New Delhi, India. DOI: 10.36227/techrxiv.15088734.v1.

Mehtab, S. and Sen, J. (2019) "A robust predictive model for stock price prediction using deep learning and natural language processing", *Proceedings of the 7th International Conference on Business Analytics and Intelligence (ICBAI'19)*, December 5-7, Bangalore, India. DOI: 10.2139/ssrn.3502624.

Mishra, S. (2016) "The quantile regression approach to analysis of dynamic interaction between exchange rate and stock returns in emerging markets: Case of BRIC nations", *IUP Journal of Financial Risk Management*, Vol. 13, No. 1, pp 7 -27.

Mondal, P., Shit, L. and Goswami, S. (2014) "Study of effectiveness of time series modeling (ARMA) in forecasting stock prices", *International Journal of Computer Science, Engineering and Applications*, Vol. 4, pp. 13 - 29. DOI: 10.5121/ijcsea.2014.4202.

Mondal, S., Dutta, A. and Chatterjee, P. (2020) "Application of deep learning techniques for precise stock market prediction", *Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence,* February 1, 2000, New Delhi, India. DOI: 10.13140/RG.2.2.19138.12489.

Mostafa, M. (2010) "Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait", *Expert Systems with Applications*, Vol. 37, No. 9, pp. 6302-6309. DOI:10.1016/j.eswa.2010.02.091.

Porshnev, A., Redkin, I. and Shevchenko, A. (2013) "Machine learning in prediction of stock market indicators based on historical data from Twitter sentiment analysis", *Proceedings of the 13th IEEE International Conference on Data Mining Workshops*, December 7-10, 2013, Dallas, TX, USA. DOI: 10.1109/ICDMW.2013.111.

Rosenberg, B., Reid, K. and Lanstein, R. (1985) "Persuasive evidence of market inefficiency", *Journal of Portfolio Management*, Vol. 11, No. 3, pp. 9 -17. DOI: 10.3905/jpm.1985.409007.

Saadaoui, F. and Messaoud, O. B. (2020) "Multiscaled neural autoregressive distributed lag: A new empirical mode decomposition model for nonlinear time series forecasting", *International Journal of Neural Systems*, Vol. 30, No. 8. DOI: 10.1142/S0129065720500392.

Selvin, S., Vinaykumar, R., Gopalakrishnan, E. A., Menon, V. K. and Soman, K. P. (2017) "Stock price prediction using LSTM, RNN, and CNN-sliding window model", *Proceedings of 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI'17)*, September 13-16, 2017, Udupi, India, pp. 1643-1647. DOI: 10.1109/ICACCI.2017.8126078.

Sen, J. (2017a) "A time series analysis-based forecasting approach for the Indian realty sector", *International Journal of Applied Economic Studies*, Vol. 5, No. 4, pp. 8 - 27. DOI: 10.36227/techrxiv.16640212.v1.

Sen, J. (2017b) "A study of the Indian metal sector using time series decomposition-based approach", Book Chapter No 8 in: *Analysis and Forecasting of Financial Time Series Using R: Models and Applications*, Sen, J. & Datta Chaudhuri, T., pp. 223- 255, August 2017, Scholars' Press, Germany. ISBN: 978-3-330-65386-3.

Sen, J. (2018) "Stock price prediction using machine learning and deep learning frameworks", *Proceedings of the 6th International Conference on Business Analytics and Intelligence (ICBAI'18)*, December 20-22, 2018, Bangalore, India.

Sen, J. and Datta Chaudhuri, T. (2018) "Understanding the sectors of Indian economy for portfolio choice", *International Journal of Business Forecasting and Marketing Intelligence*, Vol. 4, No. 2, pp. 178-222. DOI: 10.1504/IJBFMI.2018.090914.

Sen, J. and Datta Chaudhuri, T. (2017a) "A robust predictive model for stock price forecasting", *Proceedings of the 5th International Conference on Business Analytics and Intelligence (BAICONF'17)*, December 11-13, 2017, Bangalore, India. DOI: 10.36227/techrxiv.16778611.v1.

Sen, J. and Datta Chaudhuri, T. (2017b) "A time series analysis-based forecasting framework for the Indian healthcare sector", *Journal of Insurance and Financial Management*, Vol. 3, No. 1, pp. 66 – 94. DOI: 10.36227/techrxiv.16640221.v1.

Sen, J. and Datta Chaudhuri, T. (2017c) "A predictive analysis of the Indian FMCG sector using time series decomposition-based approach", *Journal of Economic Library*, Vol. 4, No. 2, pp. 206 - 226. DOI: 10.1453/jel.v4i2.1282.

Sen J. and Datta Chaudhuri, T. (2016a) "Decomposition of time series data of stock markets and its implications for prediction – an application for the Indian auto sector", *Proceedings of the 2nd National Conference on Advances in Business Research and Management Practices (ABRMP'16)*, pp. 15 – 28, Kolkata, India, January 2016. DOI: 10.13140/RG.2.1.3232.0241.

Sen, J. and Datta Chaudhuri, T. (2016b) "A framework for predictive analysis of stock market indices – a study of the Indian auto sector", *Calcutta Business School (CBS) Journal of Management Practices*, Vol. 2, No, 2, pp. 1-20. DOI: 10.13140/RG.2.1.2178.3448.

Sen, J. and Datta Chaudhuri, T. (2016c) "An alternative framework for time series decomposition and forecasting and its relevance for portfolio choice: A comparative study of the Indian consumer durable and small cap sectors", *Journal of Economics Library*, Vol. 3, No. 2, pp. 303-326. DOI: 10.1453/jel.v3i2.787.

Sen, J. and Datta Chaudhuri, T. (2016d) "An investigation of the structural characteristics of the Indian IT sector and the capital goods sector – An application of the R programming in time series decomposition and forecasting", *Journal of Insurance and Financial Management*, Vol. 1, No. 4, pp. 68-132. DOI: 10.36227/techrxiv.16640227.v1

Sen, J. and Datta Chaudhuri, T. (2016e) "Decomposition of time series data to check consistency between fund style and actual fund composition of mutual funds", *Proceedings of the 4th International Conference on Business Analytics and Intelligence (ICBAI'16)*, Bangalore, India, December 19-21, 2016. DOI: 10.13140/RG.2.2.14152.93443.

Sen, J., Dutta, A. and Mehtab, S. (2021a) "Accurate stock price forecasting using robust and optimized deep learning models", *Proceedings of the IEEE International Conference on Intelligent Technologies (CONIT'21)*, pp. 1-9, June 25-27, 2021, Hubballi, India. DOI: 10.1109/CONIT51480.2021.9498565.

Sen, J. and Mehtab, S. (2021b) "Design and analysis of robust deep learning models for stock price prediction", In: Sen, J. (ed) *Machine Learning – Algorithms, Models and Applications*, IntechOpen, London UK, pp. 15 – 46. DOI: 10.5772/intechopen.99982.

Sharma, V., Khemnar, R., Kumari, R. and Mohan, B. R. (2019) "Time series with sentiment analysis for stock price prediction", *Proceedings of the 2nd IEEE International Conference on Intelligent Communication and Computational Techniques (ICCT'19)*, September 28-29, 2019, Jaipur, India, pp. 178-181. DOI: 10.1109/ICCT46177.2019.8969060.

Siddiqui, T.A. and Abdullah, Y. (2015) "Developing a nonlinear model to predict stock prices in India: An artificial neural networks approach", *IUP Journal of Applied Finance*, Vol. 21, No. 3, pp. 36-49.

Unayik, G. K., Guler, N.(2013) "A Study on Multiple Linear Regression Analysis", *Procedia - Social and Behavioral Sciences*, Vol. 106, pp. 234–240. DOI: 10.1016/j.sbspro.2013.12.027.

Wang, Z., Ho, S-B. and Lin, Z. (2018) "Stock market prediction analysis by incorporating social and news opinion and sentiment", *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW)*, November 17-20, 2018, Singapore, Singapore. DOI: 10.1109/ICDMW.2018.00195.

Wu, Q., Chen, Y. and Liu, Z. (2008) "Ensemble model of intelligent paradigms for stock market forecasting", *Proceedings of the 1st IEEE International Workshop on Knowledge Discovery and Data Mining (WKDD'08)*, January 23-24, 2008, Adelaide, SA, Australia, pp 205 - 208. DOI: 10.1109/WKDD.2008.54.

Zhou, J. and Fan, P. (2019) "Modulation format/bit rate recognition based on principal component analysis (PCA) and artificial neural networks (ANNs)", *OSA Continuum*, Vol. 2, No. 3, pp. 923–937. DOI: 10.1364/OSAC.2.000923.

CHAPTER 5

A CAUSALITY ANALYSIS BETWEEN THE INDIAN INFORMATION TECHNOLOGY SECTOR INDEX AND THE DJIA INDEX

ASHMITA PAUL & JAYDIP SEN

Introduction

Stock price prediction is a broad area of interest in the modern research community. The stocks of different sectors like IT, Healthcare, Oil and Gas, Banking, etc., portray the country's economy. Each of the sectors has several independent stocks of companies. For example, NIFTY 50 stock index data comprise the top 50 companies from all 17 sectors of India. Like India, the USA has DJIA (Dow Jones Industrial Average), and Germany has the DAX performance index, which comprehends the top 30 companies in the country. The future movements of stock price prediction are very challenging to predict. However, an adequately planned and modeled framework can accurately forecast the stock price and negate the proposition of the *efficient market hypothesis*. The forecasting of stock price movement using time series decomposition has already been introduced in several works (Sen & Datta Chaudhuri, 2016a; 2016b; 2016c; 2016d; 2016e). A study is also conducted to show a definite association between the Indian IT sector index and the DJIA index of the USA (Sen & Datta Chaudhuri, 2016d). It is well-known that a good correlation between two time series does not necessarily imply any causal relation between them. Among many causality tests existing in the literature, the Granger causality is the most robust one (Granger, 1969). The central concept of Granger Causality is to analyze the information flow between two or more time series.

The primary objective of this work is to design a predictive model for forecasting the future movement patterns of the Indian IT sector index using its historical index values and the historical index values of the DJIA time series of the USA. It investigates a possible causality effect between the pair

of time series, the Indian IT sector index and the DJIA index of the USA. The statement "*x* Granger causes *y*" implies that the current value of y can be explained by the historical values of *y* and the lagged values of *x*. If we say that *x* causes *y*, the predictive model of the *y* time series has a statistically significant coefficient of the lagged values of *x*. Hence, the Granger causality is suited for an empirical investigation of a cause-effect relationship, and it is based on the principles of an OLS regression model. The null hypothesis associated with the Granger test assumes that the coefficients attached with lagged values of *x* are all zero. The existence of a causal relationship between two time series mandatorily requires the coefficients of both lagged time series to have significant values. For this purpose, the *F*-test is used to determine whether the inclusion of the significant coefficient values of the lagged time series in a model creates any subsequent changes in the predicted values of *y*. The *vector autoregression* (VAR) model is used to check the effectiveness of the Granger Causality test (Sims, 1980). In the VAR model, the optimal number of lags is determined using the Akaike and Schwarz-Bayesian information criteria.

In this chapter, we study the causal relationship between the Indian IT sector index and the DJIA index of the USA using their historical records. We apply several statistical, machine learning, and deep learning models for examining their forecasting accuracy on the future values of the IT and the DJIA index. The learning-based models proposed in this chapter are based on regression.

The section titled *Methodology* presents an outline of the methods followed in building various predictive models in this work. The next section, titled *Theoretical Background*, provides the readers with a brief discussion on some fundamental theoretical details on various statistical, machine learning, and deep learning models, along with a quick primer on the Granger causality. In the section titled *Performance Results and Analysis*, we present details on the forecasting performance of all the models, including a critical analysis of the performance results. The section titled *Related Work* briefly discusses some of the existing works in the literature on stock price prediction. Finally, the section titled *Conclusion* concludes the chapter.

Methodology

In the "Introduction" section, we have mentioned that our work's primary goal is to design a predictive model for two stock index time series and use the model for forecasting and a causality analysis between them. For developing the predictive models, we use the daily data of the Indian IT

sector index and the DJIA index from January 5, 2009, to December 27, 2019. The stock price data have been extracted from the Yahoo Finance website in the comma-separated values format (Yahoo Finance Website). First, we have checked the causal relation between the Indian IT sector and the DJIA indices using the Granger causality test and the VAR model. These approaches of causality analysis have been followed in our work on multivariate stock price data. For this purpose, we use the *var* library in the R programming language and utilize the log returns of the *close* values of the time series of the Indian IT sector index and the DJIA index. The raw time series are converted into two time series objects using the *ts* function in the R programming language. The Adjusted Dickey-Fuller (ADF) test is carried out to check the stationarity property of the series. Next, VAR models are built using the time series objects. The optimum value of the lag for the Granger causality test on the DJIA series and the Indian IT time series is found to be 2.

The following variables in the stock price data are considered in our analysis: (i) *d*ate, (ii) *open*, (iii) *high*, (iv) *low*, and (v) *close*. The variable *close* is chosen as the response variable, while the other variables are selected as the predictors. The dataset is split into two parts- the *training* and the *test* dataset. The training dataset consists of the records from January 5, 2009, to December 28, 2018. The remaining records, i.e., the data from December 31, 2018, to December 27, 2019, are used in testing the models. The raw data are cleaned, missing values imputed, and some critical derived variables computed before building the predictive models. The response variable in a regression model is a continuous numeric variable. The nine derived variables used in modeling are discussed in the following.

month: this integer variable is used to refer to the month of a given record. The twelve months are coded using the integers 1 to 12. January is coded as 1, while 12 is the code for December.

day_month: this is an integer variable that denotes the day of a month for a given record. This variable can assume integral values in the range [1, 31]. For example, a record with a date of January 31, 2013, will have the *day_month* variable value of 31.

day_week: an integer variable that denotes the day of the week for a record. The variable can assume any integral value from the interval [1, 5]. Monday is coded as 1, while the code for Friday is 5.

open_perc_IT: this derived variable is computed as the fractional change in the successive *open* values of the Indian IT sector index. For example, for two successive days, D_1 and D_2, if the *open* values are for two consecutive days are O_1 and O_2 respectively, then the *open_perc* for D_2 is computed as $(O_2 - O_1)/O_2$ as a percent value.

high_perc_IT: this variable is computed as the fractional change in the high values of the Indian IT sector index for two consecutive days. For example, for two consecutive days, D_1 and D_2, if the *high* values are H_1 and H_2 respectively, then the *high_perc* for D_2 is computed as $(H_2 - H_1)/H_2$ as a percent value.

low_perc_IT: this variable is computed as the fractional change in the *Indian* IT sector index's *low* values over two successive days. For example, for two consecutive days, D_1 and D_2, if the *low* values are L_1 and L_2 respectively, then the *low_perc* for D_2 is computed as $(L_2 - L_1)/L_2$ as a percent value.

close_perc_IT: this variable is computed as the fractional change in the *close* values of the Indian IT sector index series over two successive days. For two consecutive days, D_1 and D_2, if the *close* values of the Indian IT sector index are CIT_1 and CIT_2 respectively, the *close_perc_IT* for D_2 is computed as $(CIT_2 - CIT_1)/CIT_2$ as a percent value.

close_perc_DJIA: this variable is computed as the fractional change in the *close* values of the DJIA index over two successive days. For two consecutive days, D_1 and D_2, if the DJIA index's *close values* are $DJIA_1$ and $DJIA_2$ respectively, the *close_perc_DJIA* for D_2 is computed as $(DJIA_2 - DJIA_1)/DJIA_2$ as a percent value.

range_diff: It is a numeric variable computed by calculating the difference of the range values for two successive days. First, the range value for each record is computed by calculating the difference between the *high* and the *low* values of each record. Then, the difference between two successive range values is obtained to arrive at the value of the *range_diff* variable.

We develop two regression approaches. In the first approach, we predict the future values of the response variable *close_perc_IT* using the predictors *month*, *day_month*, *day_week*, *open_perc_IT*, *high_perc_IT*, *low_perc_IT*, and *range_diff*. In the second approach, we add the variable *close_perc_DJIA,* to the predictor set while predicting the future values of *close_perc_IT*. We compare the forecast accuracies of these two approaches to determine whether the predictor *close_perc_DJIA* has any significant contribution to the forecasted values of *close_perc_IT*.

Using the regression models, we forecast the future stock price for the next day using the historical stock index data. As in all regression models, the response variable *close_perc_IT* is a continuous numeric variable whose values are predicted using its past values and the past values of the predictors.

We consider two cases in which the regression models are used for prediction. These two cases are discussed in the following.

Case I: in this case, the data from January 5, 2009, to December 28, 2018, are used for training the models. The models are built and then evaluated for their performance on the same training data sets. In other words, *Case I* shows the training accuracies of the models.

Case II: In this case, the models built in *Case I* are used in predicting the values of the response variable for the cases in the test dataset. The test dataset consists of stock index time-series records from December 31, 2018, to December 27, 2019. The test dataset consists of 260 records. Since the test results reflect the models' real performance, the models performing better in *Case II* are considered to be the superior ones.

We build eight regression models for developing the proposed forecasting framework. The eight regression approaches are: (i) *multivariate regression*, (ii) *multivariate adaptive regression spline* (MARS), (iii) *bagging*, (iv) *boosting*, (v) *random forest*, (vi) *artificial neural network* (ANN), (vii) *support vector machine* (SVM), and (viii) *long- and short-term memory* (LSTM) network.

The models are evaluated using three metrics: (i) the *root mean square error* (RMSE) of the model, (ii) the correlation coefficient between the actual and the predicted values of the response variable, i.e., c*lose_perc_IT*, *and* (iii) the percentage of the test cases for which the predicted and the actual values of the response variable *close_perc_IT* exhibits a sign mismatch**.** We will discuss these three metrics more in the section *Performance Results and Analysis*.

Theoretical Background

Granger Causality: Granger causality is a statistical model that checks for any possible existence of a causal relationship between two variables. In the context of time series variables, the Granger causality test checks whether one time series can be used for forecasting the observations in the other (Granger, 1969). The basic principle of Granger causality (GC) is as follows: "If the prediction of one time series is improved by putting the knowledge together of another time series, the second time series is said to have a causal effect on the first". In this work, we use the VAR model to investigate the relationship between two time series. A time series X granger causes Y if the different lagged values of X and the lagged values of Y provide important statistical information about the future values of Y. To analyze this relationship, the VAR provides a very robust multivariate framework.

In the following, we briefly discuss the working principles of machine learning and deep learning models used in this work.

Multivariate Regression: In this model, *month, day_month, day_week, open_perc_IT*, *high_prec_IT, low_perc_IT, close_perc_DJIA, and range_diff,* all are used as predictors, and *close_perc_IT* is used as the response variable. This regression model attempts to establish a linear relationship between the predictors with the response variable. In building the multivariate regression models, we follow two methods: (i) *backward deletion* and (ii) *forward addition*. In the backward deletion and the forward addition method, we use the *drop1* and *add1* functions, respectively. In the backward deletion method, the variable with the lowest AIC and a non-significant *p*-value is removed from each iteration model. However, in each iteration, the variable with the lowest AIC and a significant *p*-value is added to the model for the forward addition method. For a given training dataset, the models built by the forward addition and the backward deletion method are identical.

MARS: *Multivariate adaptive regression spline* (MARS) is a robust machine learning-based regression method. It explores the linear and non-linear relationship of the predictors with the response variable in a dataset. The training data are split into multiple *basis functions* by the algorithm. The basis of MARS are: $f(x) = \{x - t$, if $x > t$, 0 otherwise$\}$ and $g(x) = \{t - x$, if $x < t$, 0 otherwise$\}$. The algorithm splits the non-linear data by adding the cutting points called a *knot*. Here, in the basis function, t denotes the knot. This phase of execution of the algorithm is called the *forward pass*. The main aim of the MARS model is to find out the optimal number of *knot*s. The algorithm identifies those hinges that do not significantly contribute to the model's accuracy and trims them off in the *backward pass*. Thus, at the end of the execution of the *backward pass*, only the significant predictors remain in the model with the hinges that effectively identify the linear sub-section in the entire dataset. In the *backward pass*, the algorithm computes the value of *generalized cross-validation (GCV)* to determine the model's accuracy while avoiding any possibility of model *overfitting*. We use the *earth* function defined in the *earth* library in the R programming language.

Bagging: It is a very robust ensemble technique based on several decision trees. Bagging is a short name for *Bootstrap aggregation*, where bootstrapping is a sampling method, and it creates several subsets of observation of the original data with replacement. Before we describe the bagging approach of modeling, the concept of the decision tree is described briefly. Decision tree regression is a supervised learning method. The method splits the whole data into multiple homogeneous sets depending on the most significant splitter in input variables. The algorithm searches each node for all the variables and splits them to build up the trees and obtain the

optimal split. The optimal separation has been created using the goodness of fit criteria. In Bagging (Bootstrap aggregation), a base model is built on each of the models, and all the models run in parallel and independently. The final predictions are determined by the aggregated prediction of all the models. Aggregation is a generalization procedure to reduce the variance for those algorithms that have high variance. We build the bagging regression models using the *bagging* function defined in the *ipred* library in the R programming language. The parameter *nbag* is used in the bagging function to specify the number of decision trees used in the ensemble method. The *predict* function is used to compute the predicted value. Finally, from the *Metrics* library, the *cor* and the *rmse* functions are used to calculate the correlation coefficient between the actual and the predicted values of the response variable and the model's RMSE value.

Boosting: It is also an ensemble technique, and it is one of the most effective machine learning models for robust prediction. It is a succeeding process where each subsequent model tries to correct the errors of the preceding model. In this technique also, the original data has been sampled into multiple subsets of classifiers. At first, all the classifiers got equal weights, and a base model has been applied to them. After the first prediction, the errors are calculated using actual and predicted values. The wrongly predicted observations get higher weights, and the next model that correctly predicts them is assigned with higher weights in the weighted majority voting method. Thus boosting combines several weak learning models to build up a strong one. The individual performance of the models may not be well, but it boosts the performance of the ensemble. The boosting regression model is constructed using the *blackboost* function defined in the *mboost* library of the R programming language. The predicted values of the response variable are computed using the *predict* function. The correlation coefficient between the actual and the predicted values of the response variable and the RMSE of the model are computed using the *cor* and the *rmse* functions defined in the *Metrics* library of the R programming language.

***Random Forest*:** Random Forest is a supervised learning algorithm that uses an *ensemble learning* technique that follows the bagging for regression. In this technique, bagging is the base estimator, and unlike bagging, random forest randomly selects a set of features to obtain the best split at each node of the tree. For this model, first random subsets are created, and at each node of the decision tree, the model uses random features to determine the best split, which would be beneficial for the model.

In this way, the decision tree model is fitted to each of the subsets of the dataset. The final prediction is made by averaging the overall forecast of decision trees. This model introduces high randomness because its main objective is to avoid overfitting during training. For regression, the *randomForest* function from the *randomForest* library in R is exerted. Here the response variable *close_perc_IT* was a numeric variable, so it is not required to convert it into categorical variables. The *predict* function is used to predict the response variable. The *rmse* and the *cor* functions defined in the *Metrics* library in the R programming language are used to compute the model RMSE and the correlation between the actual and the predicted values of the response for evaluation of the model performance.

Artificial Neural Network (ANN): It is a robust predictive model with several nodes and interconnecting links. The nodes are set in several layers - an input layer, one or more hidden layers, and one output layer. The network learns from the data adaptively by adjusting the weights of the links in the networks and the bias values at each node. The ANN model has quite a good resemblance to the human brain; like dendrites of the biological neuron, the model can collect predictor variables of training data as its input variables. Then the input values are passed on to the nodes of the hidden layers. Multiple hidden nodes process the information by adjusting weights and biases, and finally, the last hidden layer fed the information to the output nodes to produce the result.

We build ANN models using the *neuralnet* function defined in the *neuralnet* library of the R programming language. The predictor variables need to be converted into their normalized values before being processed by an ANN. Otherwise, predictors with higher values tend to dominate the model resulting in an inaccurate prediction model. Hence, we normalize all the predictor variable values using the *min-max normalization* method. The parameters, *hidden,* and *stepmax* for the *neuralnet* function are set to appropriate values for the model. For building a regression model, the parameter *linear.output* is set to TRUE. The predictions are made using the regression model using the *compute* function.

SVM (Support Vector Machine): It is a very efficient supervised learning model invented by Vladimir Vapnik. An SVM model can be adapted to carry out both classification and regression tasks. Both linear and non-linear SVM models can be built to handle data exhibiting linear and non-linear characteristics. For handling non-linear data, the non-linear SVM models transform the data into a higher dimension, in which the data behave linearly. At this higher dimension, the SVM models carry out linear classification or regression to perform their tasks. The critical task for an

SVM model is to find out the *maximum marginal hyperplane* (MMH) that optimally separates the records of two different classes. We build the SVM regression model using the *svm* function defined in the *e1071* library in the R programming language. The *predict* function is used to compute the predicted values by the model. The RMSE and the correlation coefficient between the actual and the predicted values are computed using the functions *rmse* and *cor* defined in the *Metrics* library.

Performance Results and Analysis

***Granger causality*:** As mentioned earlier, we use the *var* library in the R language to create time series objects using the log returns of the *close* values. The time series objects are created using the *ts* function. The *Granger causality test* based on the *var* model is used to check the causal relationship between the Indian IT index and the DJIA index time series. We use the *var* function defined in the *var* library of R with the following three parameters: (i) variable type (ii) *lag.max,* which denotes the maximum lag for causal relationship (iii) *ic*, the information criteria. The optimal lag value is determined based on the minimum value of the AIC. The stationarity characteristic of the series is checked using the Augmented Dickey-Fuller (ADF) test. Finally, R data frames are created based on the *log returns* of the *close* values of both the Indian IT index and the DJIA series. It is found that lag 2 is the most optimum lag yielding the minimum value of the AIC. Therefore, it is clear that the DJIA series has a causal effect on the Indian IT sector index with a lag value of 2. Hence, we create a predictor variable *close_perc_DJIA* and use this new predictor in the multivariate framework to investigate the regression model's prediction performance of the Indian IT sector index. Table 5-1 presents the *F*-statistic value and its associated *p*-value for the Granger causality test of DJIA on the Indian IT sector index.

TABLE 5-1. GRANGER CAUSALITY TEST WITH DJIA INDEX AS A PREDICTOR VARIABLE

Parameter	Value
F-statistics	27.212
Significance *p*-value	<2.2e-16

We consider two regression models to analyze any possible causal effect the DJIA index series has on the Indian IT sector index. These two regression models are as follows:

(a) ***IT_DJIA regression model***: In this regression model, we include *close_perc_DJIA* in the predictor set along with the other predictors for predicting the future values of the *close_perc_IT.* It may be noted that the response variable *close_perc_IT* denotes the percentage change in the *close* index of the Indian IT sector. The predictor set of the model consists of the following variables: (i) *month*, (ii) *day_month*, (iii) *day_week*, (iv) *open_perc_IT*, (v) *high_perc_IT*, (vi) *low_perc_IT*, (vii) *range_diff*, and (viii) *close_perc_DJIA*. We call this model the *IT_DJIA* model.

(b) ***IT regression model***: In this model, we do not include *close_perc_DJIA* in the predictor set. The remaining predictors remain unchanged. The set of predictors is used in predicting the response variable *close_perc_IT*. We refer to this model as the *IT* model.

In the following, we present the performance results of the regression models.

Multivariate regression: We consider two models here, *IT_DJIA* and *IT*. For both models, two cases are considered. In *Case I*, the model is trained on the training dataset and evaluated on the same dataset to assess the training performance of the model. In *Case II*, however, the model is trained on the training data and evaluated on the out-of-sample test dataset.

***IT_DJIA* model:** In *Case I*, the model is trained using the 2604 records of the training dataset. The multicollinear variables are identified using the *vif* function defined in the *faraway* library in the R programming language. We find that the variables *high_perc_IT, low_perc_IT,* and *range_diff* exhibit multicollinearity as their *variance inflation factor* (VIF) values are high. The variable *range_diff* is retained in the model while the other two are removed. Two methods we follow for the model building are *backward deletion* and *forward addition*. In the *backward deletion* method, we use the *drop1* function to remove the variable that exhibits the lowest AIC and a non-significant *p*-value in each iteration. On the other hand, in the *forward addition method*, we start with a blank model, and in each subsequent iteration, we use the *add1* function in R to include the variable that exhibits the lowest AIC and a significant *p*-value. The final model is consist of two predictors, *open_perc_IT* and *close_perc*_DJIA. We evaluate *Case I* of model *IT_DJIA* using three metrics: (i) correlation coefficient between the actual and the predicted values of the *close_perc_IT*, (ii) RMSE to the mean of the absolute values of the *close_perc_IT* in the test dataset, and (iii) the percentage of cases which exhibit sign mismatch in their actual and predicted *close_perc_IT* values. In *Case I*, the ratio of the RMSE to the mean is 34.08. The correlation test yields a correlation coefficient of 0.85. 623 cases out of 2604 exhibit a sign mismatch between the predicted and

actual values of the *close_perc_IT*. We test the residuals of the model to check whether the residuals exhibit heteroscedasticity. The Breusch-Pagan test yields a test statistic value of 1.1986 with an associated *p*-value of 0.5492. The *p*-value indicates substantial support for the null hypothesis indicating no significant heteroscedasticity in the residuals. The Durbin-Watson test yields a test statistic value of 2.3668 with an associated *p*-value of 1. The *p*-value indicates the highest support for the null hypothesis indicating no significant heteroscedasticity in the residuals of the model.

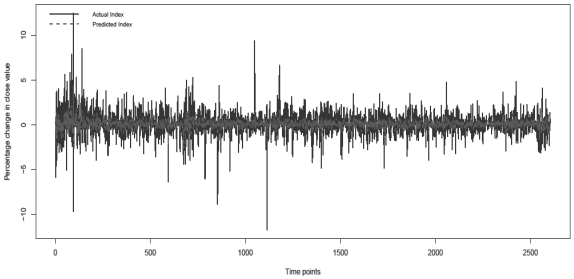

Figure 5-1. Multivariate Regression (*IT_DJIA* model) - the actual and the predicted values of *close_perc_IT* (*Case I*)

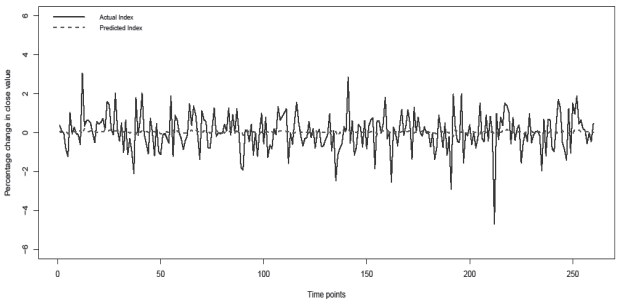

Figure 5-2. Multivariate Regression (*IT_DJIA* model) – the actual and the predicted values of *close_perc_IT* (*Case II*)

In *Case-II*, model *IT_DJIA* built under *Case I* is used in predicting the values of the response variable *close_perc_IT* for the records in the test dataset. The performance of the model is evaluated on the same three metrics. The values of the three metrics in *Case II* are found to be as follows. The correlation coefficient between the predicted and the actual values is 0.81, the RMSE to the mean ratio is 38.23, and the percentage of cases that

mismatch in sign is 26.80. These values indicate that the *IT_DJIA* regression model for *Case II* is quite precise. Figure 5-1 shows the plot of the daily actual and the predicted *close_perc_IT* values for the records in the test dataset for the model *IT_DJIA* for *Case I*. The same plot for *Case II* is shown in Figure 5-2.

***IT* model:** For this model, in *Case I*, we find that the variable *open_perc_IT* is the only significant predictor for predicting the response variable *close_perc_IT*. Both the *backward deletion* and *forward addition* methods of regression yield the same results. The ratio of the RMSE to the mean of the absolute values of the *close_perc_IT* values is found to be 36.28. Among the total 2604 cases, 710 cases exhibit a sign mismatch between the actual and predicted values of *close_perc_IT*. The correlation coefficient between the actual and the predicted values of *close_perc_IT* is 0.77. The percentage of cases that exhibit a mismatch in sign between the actual and the predicted values are found to be 27.34. The Breusch-Pagan (BP) test does not indicate any presence of heteroscedasticity in the residuals. No significant autocorrelations are also observed among the errors by the Durbin-Watson test.

In *Case II*, the ratio of RMSE to the mean of its absolute values of *close_perc_IT* is found to be 42.92. Out of 260 records, 82 records exhibit a sign mismatch between the predicted and the actual values of *close_perc_IT*. The correlation coefficient is 0.72. The BP test does not show any significant heteroscedasticity in the residuals of the model. The error values do not reflect any significant autocorrelations as verified by the results of the Durbin-Watson test.

TABLE 5-2. MULTIVARIATE REGRESSION RESULTS

Model	**IT_DJIA**		**IT**	
Metrics	**Case-I**	**Case-II**	**Case-I**	**Case-II**
	Training 2009-2018	**Test 2019**	**Training 2009-2018**	**Test 2019**
Correlation Coefficient	0.85	0.81	0.77	0.72
RMSE/Mean	34.08	38.23	36.28	42.92
Percentage of Mismatch Cases	23.92	26.80	27.27	31.54

The results presented in Table 5-2 indicate that the model *IT_DJIA* is more accurate than the model *IT*. The multivariate regression results validate our hypothesis of the causality effect of *DJIA* on the *IT* index series. The inclusion of the variable *DJIA* in the predictor list of the *IT_DJIA* model makes the model more precise than the *IT* model that does not use this predictor.

MARS: We build the MARS regression models using the *earth* function defined in the *earth* library of the R programming language.

***IT_DJIA model*:** For this model under *Case I*, the forward pass creates 12 basis functions and yields the following values of two important metrics: (i) *generalized R square*: 0.842 and (ii) *R-square: 0.844*. In the backward pass, one basis function is trimmed off to eliminate any possibility of model overfitting. The *generalized R-square* and the *R-square* values after the backward pass are 0.841 and 0.844, respectively. Out of the 2604 cases, 374 cases exhibit a mismatch in sign between their actual and the predicted values of *close_perc_IT*. The ratio of the RMSE to the mean of the absolute values of *close_perc_IT* values is 28.23, and the correlation coefficient between the actual and predicted values of *close_perc_IT* is 0.92. In *Case-II,* the MARS algorithm identifies 16 basis functions during the forward pass using three predictors out of the eight predictors in the original dataset. The *generalized R-square* and the *R-square* are found to be 0.846 and 0.879, respectively, on completion of the forward pass. In the backward pass, the eight terms are pruned by the model, and only three predictors, *high_perc_IT*, *low_perc_IT, and open_perc_IT,* are selected. The backward pass yields the values for the *generalized R square* and *R-square* of 0.8553 and 0.87. Out of the 260 cases, 48 cases mismatch in the sign between the actual and the predicted values of *close_perc_IT*. The ratio of the RMSE to the mean is 32.73, while the correlation coefficient between the actual and the predicted *close_perc_IT* is 0.87. Figure 5-3 depicts the actual and the predicted values of the *close_perc_IT* for each record in the test dataset for the *IT_DJIA* model under *Case I*. The same plot for *Case II* of the *IT_DJIA* model is shown in Figure 5-4.

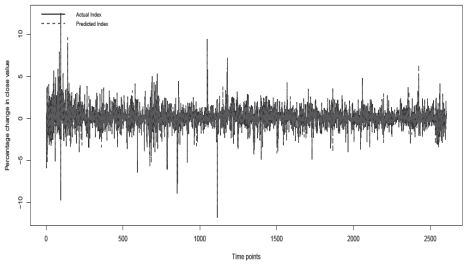

Figure 5-3. MARS Regression (*IT_DJIA* model) - actual vs. predicted values of *close_perc_IT* (*Case I*)

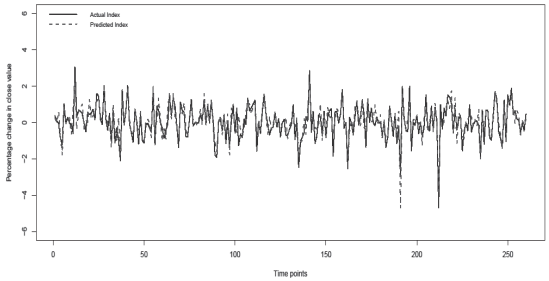

Figure 5-4. MARS Regression (*IT_DJIA* model) - actual vs. predicted values of *close_perc_IT* (*Case II*)

The *IT* model in *Case I* yields a correlation coefficient of 0.90. The ratio of the RMSE to the mean of the absolute values of the actual *close_perc_IT* is 30.74. Out of the 2604 cases, 545 cases exhibit a sign mismatch between the actual and the predicted *close_perc_IT* values. The percentage of cases that exhibit a mismatch in the sign of among the actual and the predicted *close_perc_IT* values is 20.92. In *Case II*, the *IT* model yields a correlation coefficient of 0.85. The ratio of the RMSE to the mean is 35.74. Out of the 260 cases, 61 cases exhibit a mismatch in the actual and the predicted *close_perc_IT* values.

The performance results of the MARS regression models are presented in Table 5-3. It is observed that in both cases, the model *IT_DJIA* performs superior to the model *IT*. The results support our hypothesis of the DJIA's causality on the IT sector's index values.

TABLE 5-3. MARS REGRESSION RESULTS

Model	**IT_DJIA**		**IT**	
Metrics	**Case-I**	**Case-II**	**Case-I**	**Case-II**
	Training 2009-2018	**Test 2019**	**Training 2009-2018**	**Test 2019**
Correlation Co-efficient	0.92	0.87	0.90	0.85
RMSE/Mean	28.23	32.73	30.74	35.74
Percentage of Mismatch Cases	14.36	18.46	20.92	23.46

***Bagging regression**:* We use the *bagging* function defined in the *ipred* library of the R language to build the bagging models.

***IT_DJIA model*:** The *IT_DJIA* model in *Case I* yields the value of the ratio of RMSE to the mean as 42.63. Out of the total 2604 cases, 737 cases exhibit a sign mismatch in the predicted and the actual *close_perc_IT*

values, Thus, 28.30 percent of cases exhibit a sign mismatch between their actual and predicted *close_perc_IT* values. The correlation coefficient between the actual and the predicted values of *close_perc_IT* is 0.82. In *Case II* of the *IT_DJIA* model, the ratio of the RMSE to the mean is 45.37, and the correlation coefficient between the actual and the predicted values is 0.80. Out of the total 260 cases, 79 cases exhibit a sign mismatch resulting in a percentage of mismatch of 30.38. Figure 5-5 exhibits the plot of the actual and the predicted values of *close_perc_IT* values for the *IT_DJIA* model under *Case I*. The same plot for *Case II* is depicted in Figure 5-6.

***IT model*:** *Case I* of the IT model yields a correlation coefficient of 0.82. The ratio of the RMSE to the mean is 43.82. Out of 2604 cases, 793 cases exhibit a sign mismatch between the actual and the predicted values of the *close_perc_IT*. Hence, 30.41 percent of the cases exhibit a sign mismatch between the actual and the predicted *close_perc_IT* values. In *Case II*, the *IT* model yields a correlation coefficient of 0.76. The ratio of the RMSE to the mean is 47.23. Out of 260 cases, 86 cases exhibit a sign mismatch between the actual and the predicted *close_perc_IT* values.

The performance results of both the *IT_DJIA* and the *IT* bagging models are presented in Table 5-4. The performance of the *IT_DJIA* model in both cases is found to be superior to that of the *IT* model. The results, therefore, support our hypothesis of the DJIA's causality on the IT sector index.

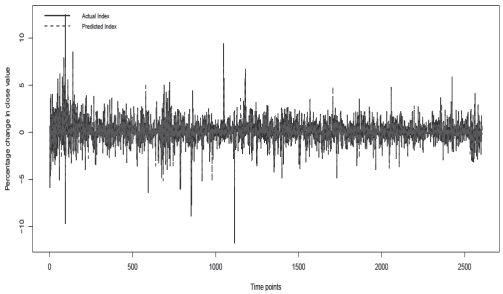

Figure 5-5. Bagging regression (*IT_DJIA* model) - actual and predicted values of *close_perc_IT* (*Case I*)

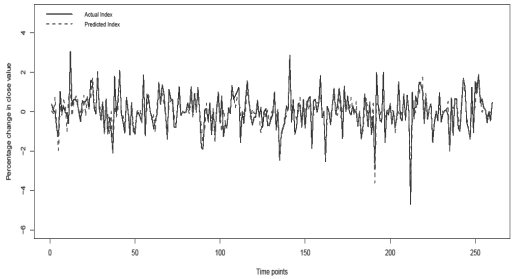

Figure 5-6. Bagging regression (*IT_DJIA* model) - actual and predicted values of *close_perc_IT* (*Case II*)

TABLE 5-4. BAGGING REGRESSION RESULTS

Framework	IT_DJIA		IT	
Metrics	Case-I	Case-II	Case-I	Case-II
	Training 2009-2018	Test 2019	Training 2009-2018	Test 2019
Correlation Co-efficient	0.82	0.80	0.82	0.76
RMSE/Mean	42.63	45.37	43.82	47.23
Percentage of Mismatch Cases	28.30	30.38	30.41	33.08

Boosting regression: We use the *blackboost* function defined in the *mboost* library of the R programming language to build the boosting models. The *IT_DJIA* model in *Case I* yields the value of 18.31 for the ratio of the RMSE to the mean of the response variable *close_perc_IT*. The correlation coefficient between the actual and the predicted values of *close_perc_IT* is 0.94. Out of the total 2604 cases, 292 cases exhibit a mismatch in the actual and the predicted values of the *close_perc_IT*. Hence, the percentage of cases that exhibit a sign mismatch is 11.21. Figure 5-7 shows the relationship between the actual and the predicted *close_perc_IT* values for the *IT* model in *Case I*. The linear relationship between the actual and the predicted *close_perc_IT* values yielded by the *IT_DJIA* model in *Case I* is depicted in Figure 5-8. The residuals of the model *IT_DJIA* in *Case I* are plotted depicted in Figure 5-9.

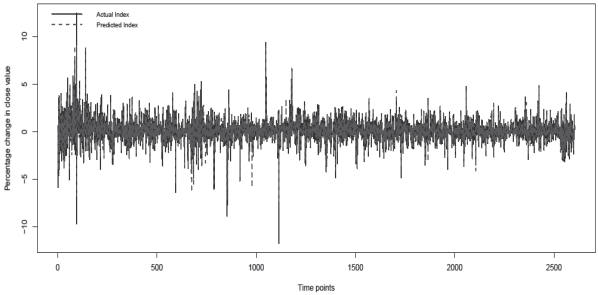

Figure 5-7. Boosting regression (*IT_DJIA* model) – the actual and the predicted values of *close_perc_IT* (*Case I*)

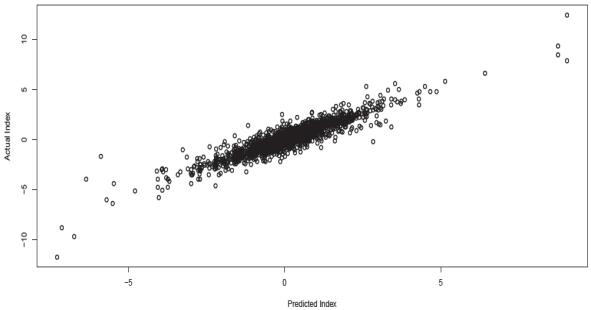

Figure 5-8. Boosting regression (*IT_DJIA* model) - the relationship between the actual and the predicted *close_perc_IT* values (*Case I*)

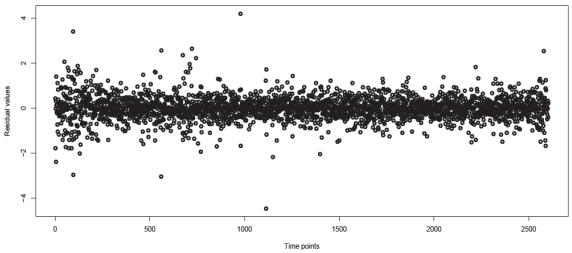

Figure 5-9. Boosting regression (*IT_DJIA* model) – the residual plot (*Case I*)

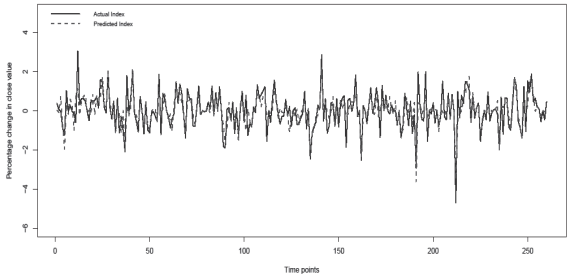

Figure 5-10. Boosting regression (*IT_DJIA* model) – the actual and the predicted *close_perc_IT values* (*Case II*)

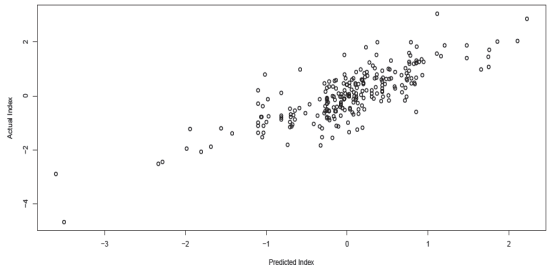

Figure 5-11. Boosting regression (IT_DJIA model) - the relationship between actual and predicted *close_perc_IT* values (*Case II*)

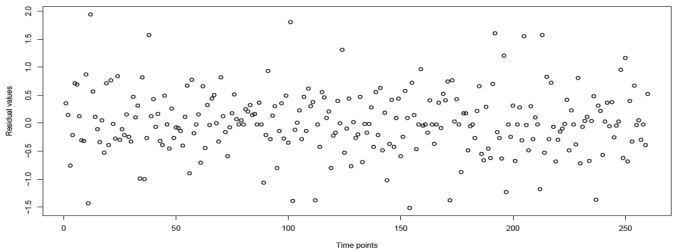

Figure 5-12. Boosting regression (*IT_DJIA* model) – the residual plot (*Case II*)

In *Case II*, the model *IT_DJIA* produces the following values of the metric. The ratio of the RMSE to mean is 21.34. Out of the total 260 cases, 37 cases exhibit a mismatch in sign between the actual and the predicted

close_perc_IT values, resulting in 21.34 percent sign-mismatched cases. The correlation coefficient between the actual and the predicted *close_perc_IT* is 0.89. Figure 5-10 depicts the actual and predicted *close_perc_IT* values for the *IT_DJIA* model in *Case II*. The relationships between the actual and the predicted *close_perc_IT* values are exhibited in Figure 5-11. The residual plot of the *IT_DJIA* model in *Case II* is presented in Figure 5-12.

The *IT* model in *Case I* yields a correlation coefficient of 0.92. The ratio of the RMSE to the mean is 20.32. Out of the total 2604 cases, 321 cases exhibit a mismatch in sign between the actual and the predicted *close_perc_IT* values. In *Case II*, the *IT* model produces a correlation coefficient value of 0.85. The ratio of the RMSE to the mean of the absolute values of the *close_perc_IT* is 23.34. Out of the total 206 cases, 41 cases exhibit a mismatch in sign between the predicted and the actual *close_perc_IT* values, leading to a percentage of mismatched cases of 15.79.

The performance results of the *IT_DJIA* and the *IT* boosting models are presented in Table 5-5. Since, in both cases, the performance of the *IT_DJIA* is superior to that of the *IT* model, the results support our hypothesis of the DJIA's causal effect on the IT sector index values.

TABLE 5-5. BOOSTING REGRESSION RESULTS

Framework	**IT_DJIA**		**IT**	
Metrics	**Case-I**	**Case-II**	**Case-I**	**Case-II**
	Training 2009-2018	**Test 2019**	**Training 2009-2018**	**Test 2019**
Correlation Co-efficient	0.94	0.89	0.92	0.85
RMSE/Mean	18.31	21.34	20.32	23.34
Percentage of Mismatch Cases	11.21	14.23	12.33	15.77

Random Forest: The *randomForest* function defined in the *randomForest* library of the R language R is used for building the *random forest* models. The *IT_DJIA* model in *Case I* yields a value of 17.32 for the ratio of the RMSE to the mean of the response variable *close_perc_IT*. Out of the total 2604 cases, 142 cases exhibit a sign mismatch between the actual and the predicted values of *close_perc_IT*. The correlation coefficient between the actual and the predicted values of *close_perc_IT* values is 0.98. In *Case II,* the *IT_DJIA* model produces 33 cases out of the total 260 cases that exhibit a sign mismatch between the actual and the predicted values of the *close_perc_IT* values. The ratio of RMSE to the mean is 21.57, while the correlation coefficient between the actual and the predicted *close_perc_IT* values is 0.92. The high value of the correlation coefficient

indicates a robust linear relationship between the actual and the predicted *close_perc_IT* values.

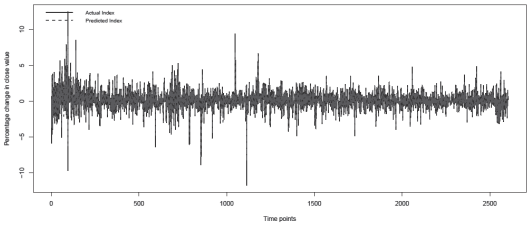

Figure 5-13. Random Forest regression (*IT_DJIA* model) - the actual and the predicted values of *close_perc_IT* (*Case I*)

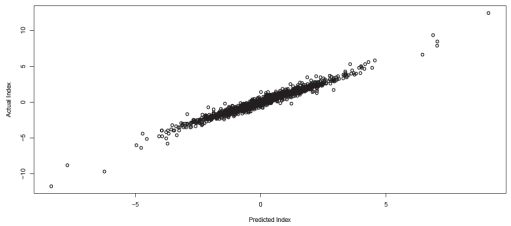

Figure 5-14. Random Forest (*IT_DJIA* model) – the relationship between the actual and the predicted values of *close_perc_IT* (*Case I*)

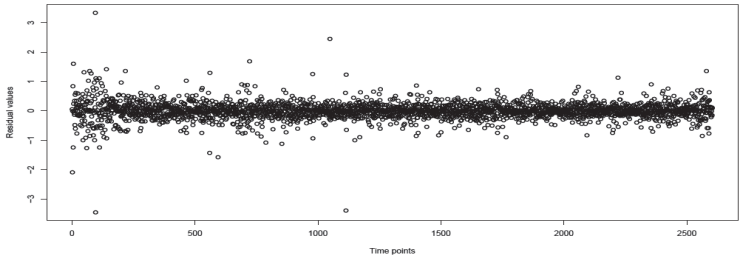

Figure 5-15. Random Forest regression (*IT_DJIA* model) – the residual plot (*Case I*)

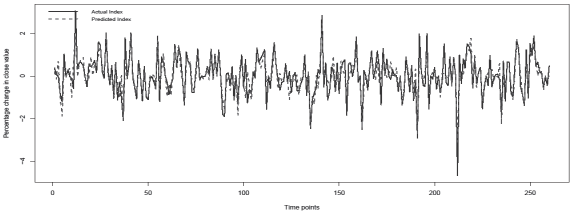

Figure 5-16. Random Forest (*IT_DJIA* model) – the actual and the predicted values of *close_perc_IT* (*Case II*)

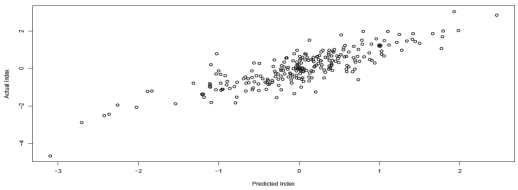

Figure 5-17. Random Forest (*IT_DJIA* model) - the relationship between the actual and the predicted values of c*lose_perc_IT* (*Case II*)

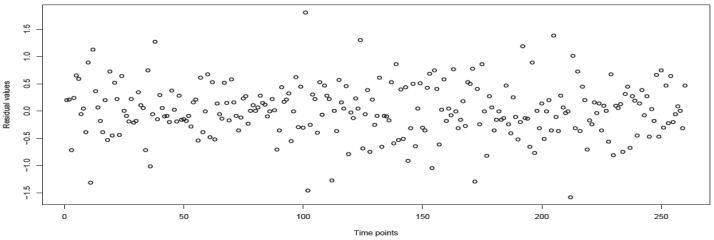

Figure 5-18. Random Forest regression (*IT_DJIA* model) – the residual plot (*Case II*)

The *IT* model of Random Forest regression in *Case I* yields a value of 20.72 for the ratio of RMSE to the mean of the absolute values of *close_perc_IT*. Out of the total 2604 cases, 269 cases exhibit a mismatch in sign between the actual and the predicted *close_perc_IT* values. The correlation coefficient between the actual and the predicted values of *close_perc_IT* is 0.93. In *Case II,* the *IT* model yields a value of 26.36 for

the ratio of the RMSE to the mean of the absolute values of the *close_perc_IT*. The correlation coefficient between the actual and the predicted values of *close_perc_IT* is found to be 0.88. Out of the total 260 cases, 41 cases exhibit a mismatch in sign between the actual and the predicted values of *close_perc_IT*. Table 5-6 presents the performance results of the *IT_DJIA* and the *IT* models for *Case I* and *Case II*.

Figure 5-13 and Figure 5-16 present the pattern of variation of the actual and the predicted values of *close_perc_IT* for the *IT_DJIA* model in *Case I* and C*ase II*, respectively. The relationships between the actual and the predicted values are depicted in Figure 5-14 and Figure 5-17 for the *IT_DJIA* model under *Case I* and *Case II*, respectively. The residual plots for the *IT_DJIA* model in *Case I* and *Case II* are exhibited in Figure 5-15 and Figure 5-18, respectively.

Table 5-6 presents the performance results of the random forest regression results. The performance of the model *IT_DJIA* is superior to that of the *IT* model in *Case I* and *Case II*. The results support our hypothesis of DJIA's causal effect on the IT index values.

TABLE 5-6. RANDOM FOREST REGRESSION RESULTS

Framework	**IT_DJIA**		**IT**	
Metrics	**Case-I**	**Case-II**	**Case-I**	**Case-II**
	Training 2009-2018	**Test 2019**	**Training 2009-2018**	**Test 2019**
Correlation Co-efficient	0.98	0.92	0.93	0.88
RMSE/Mean	17.32	21.57	20.72	26.36
Percentage of Mismatch Cases	5.45	12.69	10.33	15.77

ANN Regression: We use the *neuralnet* function defined in the *neuralnet* library in the R programming language for building the ANN models. The *IT_DJIA* model in *Case I* yields a value of 0.94 for the correlation coefficient between the actual and the predicted *close_perc_IT* values. The ratio of the RMSE to the mean of the absolute values of the *close_perc_IT* is 25.32. Out of the total 2604 cases, 475 cases exhibit a sign mismatch between the predicted and the actual *close_perc_IT* values. Figure 5-19 presents the architecture of the ANN regression model for the *IT_DJIA* for *Case I*. The plot of the actual and the predicted *close_perc_IT* values for the *IT_DJIA* model for *Case I* is shown in Figure 5-20.

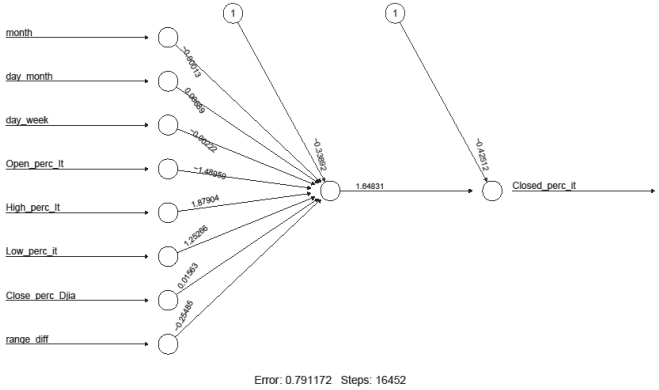

Figure 5-19. The architecture of the ANN regression model for *IT_DJIA* (*Case I)*

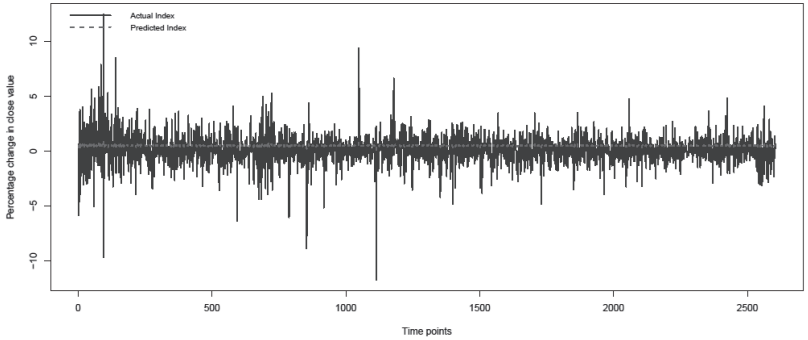

Figure 5-20. ANN regression *(IT_DJIA* model) - the actual and the predicted c*lose_perc_IT* (*Case I*)

In *Case II,* the *IT_DJIA* model yields a value of 0.86 for the correlation coefficient between the actual and the predicted *close_perc_IT* values. The ratio of the RMSE to the mean of the absolute values of the *close_perc_IT* values is 31.28. Out of the total 260 cases, 59 cases exhibit a sign mismatch between the actual and the predicted *close_perc_IT* values. Figure 5-21 depicts the plot of the actual and the predicted values of *close_perc_IT* for the *IT_DJIA* model in *Case II*.

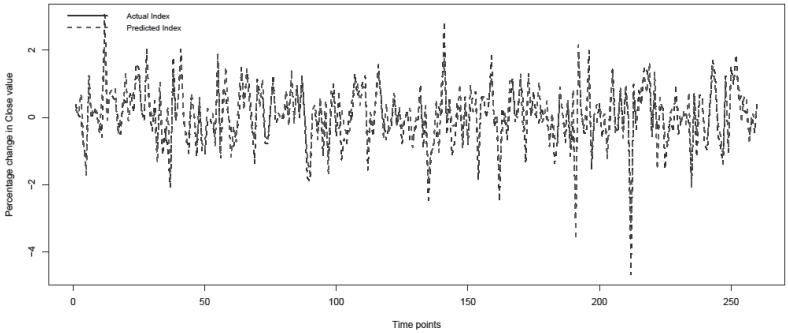

Figure 5-21. ANN regression (*IT_DJIA* model) – the actual and the predicted c*lose_perc_IT* (*Case II*)

The *IT* model in *Case I* yields a correlation coefficient value of 0.92 between the actual and the predicted *close_perc_IT*. The ratio of the RMSE to the mean of the absolute values of *close_perc_IT* is 29.35. Out of the total 2604 cases, 647 cases exhibit a sign mismatch between the predicted and the actual *close_perc_IT* values. The correlation coefficient value yielded by the *IT* model in *Case II* is 0.84. The ratio of the RMSE to the mean is 34.86. Out of the total 260 cases, 74 cases exhibit a sign mismatch between the actual and the predicted *close_perc_IT* values. Table 5-7 presents the performance results of the ANN regression models, *IT_DJIA* and *IT* under two cases. Figure 5-22 shows the architecture of the ANN regression model *IT* for *Case I*.

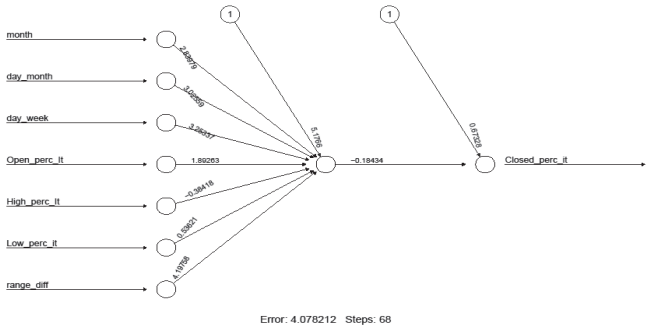

Figure 5-22. The architecture of the ANN regression model for *IT* (*Case I)*

TABLE 5-7. ANN REGRESSION RESULTS

Framework	IT_DJIA		IT	
Metrics	Case-I	Case-II	Case-I	Case-II
	Training 2009-2018	Test 2019	Training 2009-2018	Test 2019
Correlation Co-efficient	0.94	0.86	0.92	0.84
RMSE/Mean	25.32	31.28	29.35	34.86
Percentage of Mismatch Cases	18.24	22.69	24.85	28.46

It is evident from Table 5-7 that the performance of the model *IT_DJIA* is more accurate than that of the model *IT*. The observations on the performance of the ANN regression models support our hypothesis of the DJIA's causal effect on the IT index.

SVM Regression: We use the *svm* function defined in the *e1071* library in the R programming language for building the SVM models. The *IT_DJIA* model in *Case I* uses 1891 support vectors for building the regression model. The correlation coefficient between the actual and the predicted *close_perc_IT* values for the model is 0.89. The ratio of RMSE to the mean of the absolute values of the *close_perc_IT* is 28.34. Out of the total 2604 cases, 687 cases exhibit a sign mismatch between the actual and the predicted *close_perc_IT* values. In *Case II*, the *IT_DJIA* model yields a value of 0.83 for the correlation coefficient between the actual and the predicted *close_perc_IT* values. The ratio of the RMSE to the mean of the absolute values of the actual *close_perc_IT* values is 32.73. Out of the total 260 cases, 77 cases exhibit a sign mismatch between the actual and the predicted *close_perc_IT* values. Figure 5-23 depicts the plots of the actual and the predicted *close_perc_IT* values for the *IT_DJIA* model in *Case I*. The same plot for *Case II* is depicted in Figure 5-24.

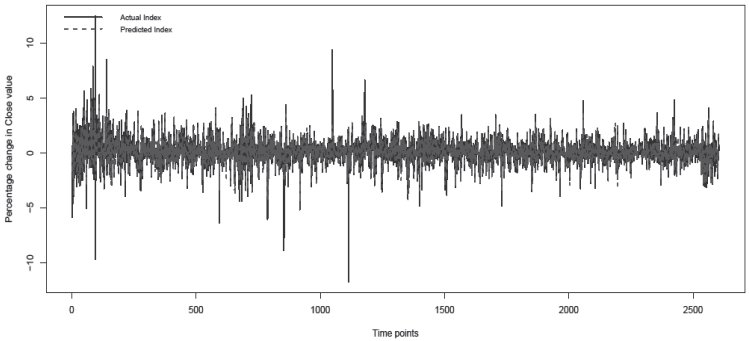

Figure 5-23. SVM regression (*IT_DJIA* model) - the actual and the predicted values of c*lose_perc_IT* (*Case I*)

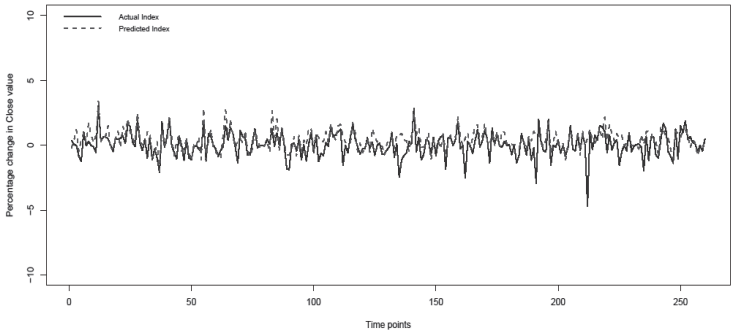

Figure 5-24. SVM regression (*IT_DJIA* model) – the actual and the predicted values of c*lose_perc_IT* (*Case II*)

The *IT* model in *Case I* yields a correlation coefficient value of 0.86 between the actual and the predicted values of the *close_perc_IT*. The ratio of the RMSE to the mean of the absolute values of the *close_perc_IT* values is 31.42. Out of the total 2604 cases, 738 cases exhibit a sign mismatch between the actual and the predicted values of *close_perc_IT.* In *Case II*, the *IT* model yields a correlation coefficient value of 0.81 between the actual and the predicted *close_perc_IT* values. The ratio of the RMSE to the mean of the absolute values of the *close_perc_IT* is 36.87. Out of the total 260 cases, 87 cases exhibit a sign mismatch between the actual and the predicted

close_perc_IT values. The performance results of the SVM regression models *IT_DJIA* and *IT* are presented in Table 5-8.

TABLE 5-8. SVM REGRESSION RESULTS

Framework	**IT_DJIA**		**IT**	
Metrics	**Case-I**	**Case-II**	**Case-I**	**Case-II**
	Training 2009-2018	**Test 2019**	**Training 2009-2018**	**Test 2019**
Correlation Co-efficient	0.89	0.83	0.86	0.81
RMSE/Mean	28.34	32.73	31.42	36.87
Percentage of Mismatch Cases	26.38	29.62	28.34	33.46

The results presented in Table 5-8 support our hypothesis of the DJIA's causal effect on the IT sector index as the performances of the *IT_DJIA* models for both cases are found to be superior to those of the *IT* model.

LSTM Regression: Finally, we discuss the deep learning-based regression models based on LSTM architecture. In building the LSTM models, we use the following steps: (i) reading the raw stock index data of the Indian IT sector and the DJIA index, (ii) carrying out data cleaning and imputation operation, (iii) normalizing all the predictor variables using *min-max normalization*, (iv) converting the normalized data into a time series, and then defining it into the problem of supervised learning, (v) building a deep learning model using the Tensorflow and the Keras framework, (vi) training and validating the model, (vii) visualizing the training and the validation performance, and (viii) evaluating the prediction accuracy of the model on the test data.

For the *IT_DJIA* model, the data have nine attributes: (i) *year*, (ii) *month*, (iii) *day*, (v) *open_IT*, (vi) *high_IT*, (vii) *low_IT*, (viii) *close_IT*, and (ix) *close_DJIA*. In the *IT* model, the data do not include the predictor *close_DJIA*. For both models, during the training phase (i.e., in *Case I*), a total of 2604 records are used. Out of which, 2344 records are used for training the models, while the remaining 260 records are used for validation. The models are tested on the remaining 250 records in the dataset. The records are normalized using the *MinMaxScala*r function defined in the *sklearn.processing* module of the Python language. The libraries in the Tensorflow and the Keras frameworks are utilized in training and testing the models. The *Adam* optimizer and the *mean absolute error* (MAE) loss function are used in compiling the models.

Figure 5-25 depicts the training data (i.e., *Case I*) for the model *IT_DJIA*. While training the model, it is found that the training and the validation losses are minimized with a batch size of 288 for 140 epochs. Figure 5-26 shows the plot of the training and validation versus the number of epochs. The *IT_DJIA* model in *Case I* (i.e., in the training phase) yields an RMSE value of 0.072, a correlation coefficient of 0.995 between the actual and the predicted values of *close_IT*. After the final epoch, the training and validation loss values are 0.0073 and 0.0059, respectively. Figure 5-27 depicts the training and the test loss for the *IT_DJIA LSTM* model in *Case II* (i.e., the test phase). Using the same batch size and epoch size as in the training case (i.e., in *Case I*), the model yields an RMSE value of 0.096 and a correlation coefficient of 0.971. The training and test loss values are found to be 0.0050 and 0.0077, respectively.

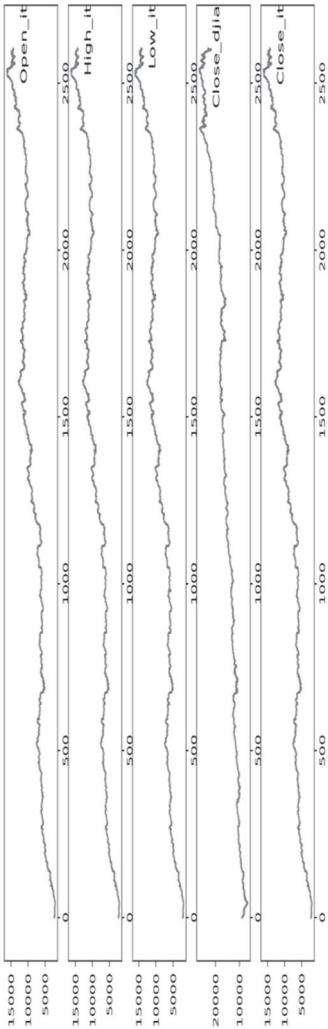

Figure 5-25. The Indian IT sector index values for training the *IT_DJIA* model (*Case I*)

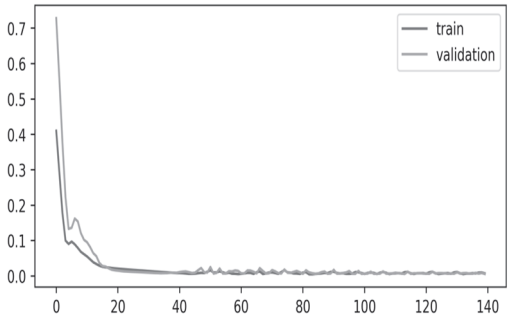

Figure 5-26. The *IT_DJIA LSTM* model – the training and the validation loss for different epochs (*Case I*)

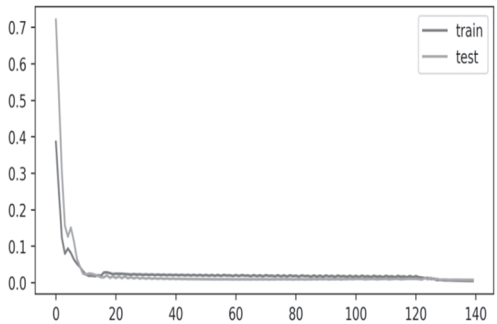

Figure 5-27. The *IT_DJIA LSTM* model – the training and the validation loss for different epochs (*Case II*)

The training of the *IT LSTM* model is done using the same dataset as the *IT_DJIA* training dataset, except that the variable *close_DJIA* is not included in the predictor set. The model, *IT LSTM*, is trained using the same batch size and the same epoch number as the *IT_DJIA* model in *Case I*. Figure 5-28 depicts the convergence of training and validation loss as the number of epochs increases in the training phase (i.e., in *Case I*). The same plot for the test phase (i.e., *Case II*)of the model is shown in Figure 5-29. After the final epoch, the RMSE of the *IT LSTM* model in the training phase is 0.101, and the correlation coefficient is 0.992. The training and the validation loss values after the training phase are 0.0075 and 0.0054, respectively. In *Case II* (i.e., the test phase), the *IT LSTM* model yields an RMSE value of 0.111 and the correlation coefficient value of 0.972. At the

end of the final epoch, the training and testing loss values are 0.134 and 0.0092, respectively.

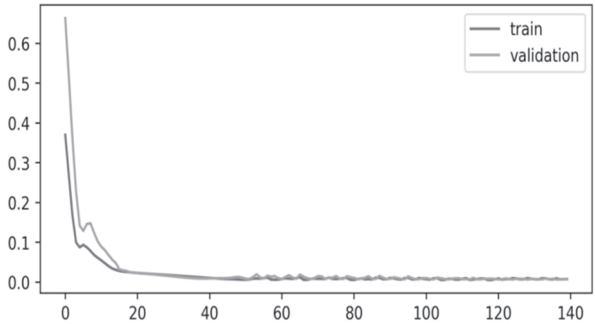

Figure 5-28. The *IT LSTM* model – the training and the validation loss for different epochs (*Case I*)

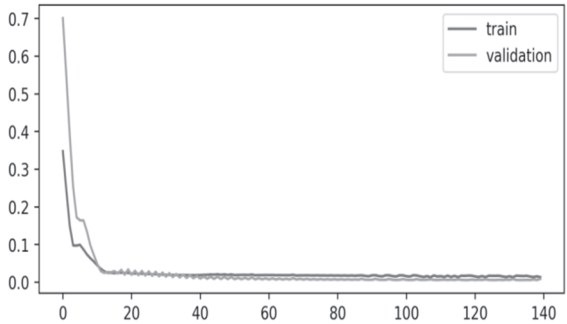

Figure 5-29. The *IT LSTM* model – the training and the validation loss for different epochs (*Case II*)

Overall Performance of Regression: We summarize the observations. Since the performance of the models in the test phase is the most critical indicator of a model's effectiveness, we consider the test performance of the two models, *IT_DJIA* and *IT*. In Table 5-9, we present the summary results of all *IT_DJIA* models. Similarly, the performance results of all *IT* models are shown in Table 5-10.

In Table 5-9 and Table 5-10, the following abbreviations are used in the column names: LR – Logistic Regression, BAG – Bagging, BOOST – Boosting, RF – Random Forest, ANN – Artificial Neural Networks, SVM – Support Vector Machines, LSTM – Long-and-Short-Term Memory Network.

TABLE 5-9. SUMMARY OF THE PERFORMANCE OF THE REGRESSION MODELS IN *CASE II OF IT_DJIA*

	MV	**MARS**	**BAG**	**BOOST**	**RF**	**ANN**	**SVM**	**LSTM**
Correlation	0.81	0.87	0.80	0.89	**0.92**	0.86	0.83	**0.99**
RMSE/Mean	38.23	32.73	45.37	**21.34**	21.57	31.28	32.73	**8.270**
Mismatched cases (%)	26.80	18.46	30.38	14.23	**12.69**	22.69	29.62	**0**

Table 5-10. Summary of the performance of the regression models in Case II of IT

	MV	**MARS**	**BAG**	**BOOST**	**RF**	**ANN**	**SVM**	**LSTM**
Correlation	0.72	0.85	0.76	0.85	**0.88**	0.84	0.81	**0.99**
RMSE/Mean	42.92	35.74	47.23	**23.34**	26.36	34.86	36.37	**1.160**
Mismatched cases (%)	31.54	23.46	33.08	**15.77**	**15.77**	28.46	33.46	**0**

It is observed that the deep learning-based LSTM regression models for *IT_DJIA* and *IT* have outperformed all machine learning models. However, among the machine learning models for *IT_DJIA*, the random forest model has performed the best on two metrics, correlation coefficient and the percentage of sign-mismatched cases. The boosting regression model has performed best on the ratio of the RMSE to the mean of the actual *close_perc_IT* values. For the *IT* model, the random forest regression has performed the best on two metrics, the correlation coefficient and the percentage of sign-mismatched cases. The boosting regression also has exhibited the best performance on two metrics, the ratio of the RMSE to the mean and the percentage of sign-mismatched cases.

We also observe that the overall performance of the *IT_DJIA* models is superior to the corresponding *IT* models. This observation is critical as it supports our central hypothesis of DJIA's causal effect on the Indian *IT* sector index values.

Related Work

The design and development of models for forecasting stock prices and their movement patterns have been a very active area of research. While extensive work has been done in these areas, most of the existing propositions in the literature can be categorized into three broad types. The frameworks belonging to the first category are essentially built on *multivariate ordinary least square regression* (Chatterjee et al., 2021; Ivanovski et al., 2016; Cakra & Trisedya, 2015; Roy et al., 2015; Sen & Datta Chaudhuri, 2016c; Sen & Datta Chaudhuri, 2016d). However, these models fail to perform well on real-world data as the stringent requirements that these models impose on the data are usually not satisfied.

The propositions in the second category are time series and econometric models like *autoregressive moving average* (ARIMA), Granger causality, quantile regression etc. (Babu & Reddy, 2014; Chatterjee et al., 2021; Saadaoui & Messaoud, 2020; Kim & Sayama, 2017; Sen & Datta Chaudhuri, 2018; Sen, 2017a; Sen, 2017b; Sen & Datta Chaudhuri, 2017b; Sen & Datta Chaudhuri, 2017c; Sen & Datta Chaudhuri, 2016a; Sen & Datta Chaudhuri, 2016b, Sen & Datta Chaudhuri, 2016e). These models yield high accuracy in forecasting if the financial time series data is dominated by a trend and a seasonal component. However, their accuracy level falls drastically in the presence of any strong random component in the time series.

The predictive models of the third category are based on *machine learning*, *deep learning*, and *natural language processing algorithms*

(Chatterjee et al., 2021; Mehtab & Sen, 2022; Mehtab & Sen, 2021a; Mehtab et al., 2021b; Mehtab et al. 2020a; Mehtab & Sen, 2020b; Mehtab & Sen, 2020c; Mehtab & Sen, 2020d; Mehtab & Sen, 2019; Sen, 2018; Sen & Datta Chaudhuri, 2017a; Sen et al., 2021a; Sen & Mehtab, 2021b; Sen et al., 2020; Rundo, 2019; Wang et al., 2015). These models learn from the past data patterns and the textual information on the web and social media and exploit it to forecast future stock prices. The performance of the models is superior on financial time series data compared to the models of the first two categories.

Conclusion and Future Work

This chapter investigated a possible causal relationship between the Indian IT sector index and the DJIA index series of the USA. The study conclusively revealed that the DJIA index series has a causal effect on the Indian IT sector index series. We have also found that DJIA index values influence the Indian IT sector's index values with a lag value of 2. We developed predictive frameworks: *IT_DJIA* and *IT*. While both the predictive frameworks are used to predict the future return of *close* values of the IT sector index, the *IT_DJIA* model includes the return values of the *close* index of DJIA as one of its predictors. However, the predictive framework IT does include the return values of the *close* index of the DJIA in its set of predictors. It is observed that the accuracy of the *DJIA_IT* model is higher compared to the *IT* model, clearly indicating that the return values of the close index of DJIA contribute significantly to forecasting the future close values of the Indian IT sector index.

As a future scope of work, we explore several LSTM and CNN-based deep learning models for forecasting stock price movements and the stock price values of the Indian IT sector using the *IT_DJIA* framework.

References

Babu, C. N. and Reddy, B. E. (2014) "A moving-average filter based hybrid ARIMA-ANN model for forecasting time series data", *Applied Soft Computing*, Vol. 23, pp. 27-38, 2014. DOI: 10.1016/j.asoc.2014.05.028.

Cakra, Y. E. and Trisedya, B. B. (2015) "Stock price prediction using linear regression based on sentiment analysis", *Proceedings of the International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, October 10-11, Depok, Indonesia, pp.147-154. DOI: 10.1109/ICACSIS.2015.7415179.

Chatterjee, A., Bhowmick, H. and Sen, J. (2021) "Stock price prediction using time series, econometric, machine learning, and deep learning models", *Proceedings of IEEE Mysore Sub Section International Conference (MysuruCon'21)*, pp. 289-296, October 24-25, 2021, Hassan, Karnataka, India.
DOI: 10.1109/MysuruCon52639.2021.9641610.

Granger, C. W. J. (1969) "Investigating causal relations by econometric models and cross-spectral methods", *Econometrica*, Vol 37, No 3, pp. 424-438. DOI: 10.2307/1912791.

Ivanovski, Z., Ivanovska, N. and Narasanov, Z. (2016) "The regression analysis of stock returns at MSE", *Journal of Modern Accounting and Auditing*, Vol. 12, No. 4, pp. 217-224. DOI: 10.17265/1548-6583/2016.04.003.

Kim, M. and Sayama, H. (2017) "Predicting stock market movements using network science: An information-theoretic approach", *Applied Network Science*, Vol. 35, 2017. DOI: DOI 10.1007/s41109-017-0055-y.

Mehtab, S. and Sen, J. (2022) "Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models". In: Sahoo J. P., Tripathy, A. K., Mohanty, M., Li, K. C., Nayak, A. K. (eds) *Advances in Distributed Computing and Machine Learning*, Lecture Notes in Networks and Systems, Vol 202, pp. 405-423, Springer, Singapore. DOI: 10.1007/978-981-16-4807-6-39.

Mehtab, S. and Sen, J. (2021a) "A time series analysis-based stock price prediction using machine learning and deep learning models", *International Journal of Business Forecasting and Marketing Intelligence (IJBFMI)*, Vol 7, No 2, pp. 143-193, Inderscience Publishers. DOI: 10.1504/IJBFMI.2021.10043037.

Mehtab, S., Sen, J. and Dutta, A. (2021b) "Stock price prediction using machine learning and LSTM-based deep learning models". In: Thampi, S. M., Piramuthu, S., Li, K. C., Berretti, S., Wozniak, M., Singh, D. (eds) *Machine Learning and Metaheuristics Algorithms, and Applications, SoMMA'2020*. Communications in Computer and Information Science, Vol 1366, Springer, Singapore.
DOI: 10.1007/978-981-16-0419-5_8.

Mehtab, S., Sen, J. and Dasgupta, S. (2020a) "Robust analysis of stock price time series using CNN and LSTM-based deep learning models", *Proceedings of the 4th IEEE International Conference on Electronics, Communication and Aerospace Technology (ICECA'20)*, November 5-7, 2020, Coimbatore, India, pp. 1481-1486.
DOI: 10.1109/ICECA49313.2020.9297652.

Mehtab, S. and Sen, J. (2020b) "Stock price prediction using CNN and LSTM-based deep learning models", *Proceedings of the IEEE International Conference on Decision Aid Sciences and Applications (DASA'20)*, November 8-9, Sakheer, Bahrain, pp. 447-453. DOI: 10.1109/DASA51403.2020.9317207.

Mehtab, S. and Sen, J. (2020c) *Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models*, Technical Report, No: NSHM_KOL_2020_SCA_DS_1. DOI: 10.13140/RG.2.2.14022.22085/2.

Mehtab, S. and Sen, J. (2020d) "Stock price prediction using convolutional neural networks on a multivariate time series", *Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence (NCMLAI'20)*, February 1-2, 2020, New Delhi, India. DOI: 10.36227/techrxiv.15088734.v1.

Mehtab, S. and Sen, J. (2019) "A robust predictive model for stock price prediction using deep learning and natural language processing", *Proceedings of the 7th International Conference on Business Analytics and Intelligence (BAICONF'19)*, December 5-7, Bangalore, India. DOI: 10.36227/techrxiv.15023361.v1.

Roy, S. S., Mittal, D., Basu, A. and Abraham, A. (2015) "Stock market forecasting using LASSO linear regression model", In: Abraham, A., Kromer, P., Snasel, V. (eds.) *Afro-European Conference for Industrial Advancement, Advances in Intelligent Systems and Computing*, Vol. 334, pp. 371-381, Springer, Cham. DOI: 10.1007/978-3-319-13572-4_31.

Rundo, F., Trenta, F., di Stallo, A. L. and Battlato, S. (2019) "Machine learning for quantitative finance applications: A survey", *Applied Sciences*, Vol 9, No 24, Article Id: 5574. DOI: 10.3390/app9245574.

Saadaoui, F. and Messaoud, O. B. (2020) "Multiscaled neural autoregressive distributed lag: A new empirical mode decomposition model for non-linear time series forecasting", *International Journal of Neural Systems*, Vol. 30, No. 8. DOI: 10.1142/S0129065720500392.

Sen, J. (2017a) "A time series analysis-based forecasting approach for the Indian realty sector", *International Journal of Applied Economic Studies*, Vol. 5, No. 4, pp. 8 - 27. DOI: 10.36227/techrxiv.16640212.v1.

Sen, J. (2017b) "A study of the Indian metal sector using time series decomposition-based approach", Book Chapter No 8 in: *Analysis and Forecasting of Financial Time Series Using R: Models and Applications*, Sen, J. & Datta Chaudhuri, T., pp. 223- 255, August 2017, Scholars' Press, Germany. ISBN: 978-3-330-65386-3.

Sen, J. (2018) "Stock price prediction using machine learning and deep learning frameworks", *Proceedings of the 6th International Conference on Business Analytics and Intelligence (ICBAI'18)*, December 20-22, Bangalore, India.

Sen, J. and Datta Chaudhuri, T. (2018) "Understanding the sectors of Indian economy for portfolio choice", *International Journal of Business Forecasting and Marketing Intelligence*, Vol. 4, No. 2, pp. 178-222. DOI: 10.1504/IJBFMI.2018.090914.

Sen, J. and Datta Chaudhuri, T. (2017a) "A robust predictive model for stock price forecasting", *Proceedings of the 5th International Conference on Business Analytics and Intelligence (BAICONF'17)*, December 11-13, Bangalore, India. DOI: 10.36227/techrxiv.16778611.v1.

Sen, J. and Datta Chaudhuri, T. (2017b) "A time series analysis-based forecasting framework for the Indian healthcare sector", *Journal of Insurance and Financial Management*, Vol. 3, No. 1, pp. 66 - 94. DOI: 10.36227/techrxiv.16640221.v1.

Sen, J. and Datta Chaudhuri, T. (2017c) "A predictive analysis of the Indian FMCG sector using time series decomposition-based approach", *Journal of Economic Library*, Vol. 4, No. 2, pp. 206 - 226. DOI: 10.1453/jel.v4i2.1282.

Sen J. and Datta Chaudhuri, T. (2016a) "Decomposition of time series data of stock markets and its implications for prediction – an application for the Indian auto sector", *Proceedings of the 2nd National Conference on Advances in Business Research and Management Practices (ABRMP'16)*, pp. 15 – 28, Kolkata, India, January 2016. DOI: 10.13140/RG.2.1.3232.0241.

Sen, J. and Datta Chaudhuri, T. (2016b) "A framework for predictive analysis of stock market indices – a study of the Indian auto sector", *Journal of Management Practices*, Vol. 2, No, 2, pp. 1-20, Calcutta Business School Publication, Kolkata, India. DOI: 10.13140/RG.2.1.2178.3448.

Sen, J. and Datta Chaudhuri, T. (2016c) "An alternative framework for time series decomposition and forecasting and its relevance for portfolio choice: A comparative study of the Indian consumer durable and small cap sectors", *Journal of Economics Library*, Vol. 3, No. 2, pp. 303-326. DOI: 10.1453/jel.v3i2.787.

Sen, J. and Datta Chaudhuri, T. (2016d) "An investigation of the structural characteristics of the Indian IT sector and the capital goods sector – An application of the R programming in time series decomposition and

forecasting", *Journal of Insurance and Financial Management*, Vol. 1, No. 4, pp. 68-132. DOI: 10.36227/techrxiv.16640227.v1.

Sen, J. and Datta Chaudhuri, T. (2016e) "Decomposition of time series data to check consistency between fund style and actual fund composition of mutual funds", *Proceedings of the 4th International Conference on Business Analytics and Intelligence (ICBAI'16)*, Bangalore, India, December 19-21. DOI: 10.13140/RG.2.2.14152.93443.

Sen, J., Dutta, A. and Mehtab, S. (2021a) "Accurate stock price forecasting using robust and optimized deep learning models", *Proceedings of the IEEE International Conference on Intelligent Technologies (CONIT'21)*, pp. 1-9, June 25-27, 2021, Hubballi, India. DOI: 10.1109/CONIT51480.2021.9498565.

Sen, J. and Mehtab, S. (2021b) "Design and analysis of robust deep learning models for stock price prediction", In: Sen, J. (ed) *Machine Learning – Algorithms, Models and Applications*, IntechOpen, London UK, pp. 15 – 46. DOI: 10.5772/intechopen.99982.

Sen, J., Mehtab, S. and Nath, G.(2020) "Stock price prediction using deep learning models", *Lattice: The Machine Learning Journal*, Vol 1, No 3, pp. 34-40, December 2020. DOI: 10.36227/techrxiv.16640197.v1.

Sims, C. A. (1980) "Macroeconomics and Reality", *Econometrica*, Vol 48, No 1, pp. 1-48. DOI: 10.2307/1912017.

Wang, L., Zeng, Y. and Chen, T. (2015) "Back propagation neural network with adaptive differential evolution algorithm for time series", *Expert Systems with Applications*, Vol. 42, No. 2, pp. 855-863. DOI: 10.1016/j.eswa.2014.08.018.

Yahoo Finance Website: https://in.finance.yahoo.com.

CHAPTER 6

STOCK PRICE PREDICTION USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS AND MODELS

SIDRA MEHTAB & JAYDIP SEN

Introduction

Prediction of future movement patterns of stock prices has been a widely researched area in the literature. While there are proponents of the efficient market hypothesis who believe that it is impossible to predict stock prices, there are also propositions that demonstrate that if correctly formulated and modeled, the prediction of stock prices can be made with a reasonably high level of accuracy. The latter school of thought focuses on building robust statistical, econometric, and machine learning models based on the careful choice of variables and appropriate functional forms or forecasting models. In the literature, propositions exist for future stock price prediction using time series analysis and decomposition. In this regard, Sen and Datta Chaudhuri propose several frameworks (Sen & Datta Chaudhuri, 2018; Sen & Datta Chaudhuri, 2017a; Sen & Datta Chaudhuri 2017b; Sen & Data Chaudhuri, 2017c; Sen & Datta Chaudhuri, 2016a; Sen & Datta Chaudhuri, 2016b; Sen & Datta Chaudhuri, 2016c; Sen & Datta Chaudhuri, 2015).

There is also extensive literature that deals with various technical analyses of stock price movements. Propositions exist for mining sock price patterns using various important indicators like *Bollinger Bands*, *moving average convergence divergence* (MACD), *relative strength index* (RSI), *moving average* (MA), and *stochastic momentum index* (SMI). There are also well-known patterns like *head and shoulders pattern*, *inverse head and shoulders pattern*, *triangle, flag*, *Fibonacci fan*, and *Andrew's Pitchfork*, which traders exploit for investing intelligently. These approaches provide the user with visual manifestations of the indicators, which help ordinary

investors to understand which way stock prices are more likely to move soon.

This chapter proposes a granular approach to forecasting stock price and the price movement pattern by combining several statistical, machine learning, and deep learning methods. We present several methods for predicting the short-term stock price movement patterns using classification and regression techniques and compare their performance. We believe this approach will provide several useful information to the investors in the stock market who are particularly interested in making a profit from short-term investments. This work is a modified and extended version of our previous work (Mehtab & Sen, 2019). We have presented a predictive framework that aggregates eight classification and eight regression models in the present work, including a *long-and-short-term memory* (LSTM)-based advanced deep learning model.

Our proposed models are built on stock price data at five minutes intervals from the National Stock Exchange (NSE) of India. We contend that models built on such high-frequency data can capture the inherent features of and patterns hidden in the data that can be utilized to make robust and accurate forecasting of the stock price in a short time horizon. Here, we are not addressing the problem of predicting the long-term movement of the stock price. Instead, our framework will be more relevant to a trade-oriented framework.

The rest of the chapter is organized as follows. The section titled *Related Work* presents a comprehensive review of the literature on stock price modeling and prediction. In the section titled *Methodology*, we present a detailed discussion on the methodology we followed in this work. The section titled *Machine Learning Models* provides a brief discussion on the working principles of the classification and the regression models proposed in this work. In the section titled *Deep Learning Models*, we present a brief discussion on the LSTM-based deep learning model for regression used in our proposed framework. The section titled *Performance Results and Analysis* presents a detailed discussion on the performance of the predictive models. A comparative analysis of the performances of the models is also presented in this section. Finally, the section titled *Conclusion* concludes the chapter.

Related Work

According to the choice of variables and techniques of estimation and forecasting, the propositions in the literature on stock price prediction can be classified into three strands. The first strand consists of studies using

simple regression techniques on cross-sectional data (Basu, 1983; Chui & Wei, 1998; Fama & French, 1995; Jaffe et al., 1989; Rosenberg et al., 1985). The second strand of the literature uses time series models and techniques to forecast stock returns. The propositions under this category are based on econometric models such as *autoregressive integrated moving average* (ARIMA), *Granger causality test*, *autoregressive distributed lag* (ARDL), and *quantile regression* (QR). (Adebiyi et al., 2014; Jarrett & Kyper, 2011; Mishra, 2016; Mondal et al., 2014). The third strand includes works using machine learning tools for the prediction of stock returns (Dutta et al., 2006; Jaruszewicz & Mandziuk, 2004; Mostafa, 2010; Siddiqui & Abdullah, 2015; Wu et al., 2008; Sen, 2018; Mehtab & Sen, 2020a, Mehtab & Sen, 2020b; Mehtab et al., 2020c; Mehtab & Sen, 2020d; Mehtab & Sen, 2021a; Mehtab et al., 2021b; Mehtab & Sen, 2022).

Among some of the recent propositions, Mehtab and Sen demonstrate how machine learning and *long- and short-term memory* (LSTM)-based deep learning networks can be used for accurately forecasting NIFTY 50 stock price movements (Mehtab & Sen, 2019). The authors use the daily stock prices for three years, from January 2015 to December 2017, for building the predictive models. The forecast accuracies of the models are evaluated based on their ability to predict the movement patterns of the *close* value of the NIFTY index on a time horizon of one week. For testing, the authors use the NIFTY 50 index values from January 2018 to June 2019. To further improve the predictive power of the model, the authors incorporate *a sentiment analysis module* for analyzing the public sentiments on Twitter on NIFTY 50 stocks. The past NIFTY 50 index values and the output of the sentiment analysis module are fed into the predictive model. The sentiment analysis module uses a *self-organizing fuzzy neural network (SOFNN)* for handling nonlinearity in a *multivariate predictive environment*.

Mehtab and Sen propose another approach to stock price and movement prediction using *convolutional neural networks* (CNN) on a multivariate time series (Mehtab & Sen, 2020a). The predictive model proposed by the authors exploits the learning ability of a CNN with a *walk-forward validation* to attain a high level of accuracy in forecasting the future NIFTY index values and their movement patterns. Three different architectures of CNN are proposed by the authors that differ in: (i) the number of variables used in forecasting, (ii) the number of sub-models used in the overall system, and (iii) the size of the input data for training the models. The experimental results indicate that the CNN-based forecasting models are highly efficient and accurate in forecasting the movement of NIFTY index values.

Designing efficient predictive models and algorithms for accurately forecasting the movement patterns of stock prices and stock returns has attracted considerable attention and effort from the research community. Many of such propositions involve the application of various types of neural networks. The neural networks can model nonlinearity in data, which is extremely useful in mining the intricate patterns in stock price movements. Moreover, the ability to model nonlinearity can be controlled adaptively by choosing a suitable number of hidden layers and the number of nodes in such hidden layers (Hornik et al., 1989).

Mostafa shows how accurately neural network-based models can predict the movement of Kuwait's stock market index (Mostafa, 2010).

Kimoto et al. demonstrate how neural-network-based predictive models can be applied to historical accounting data (Kimoto et al., 1990). In building the neural network model, the authors utilize various macroeconomic variables. The model is used in forecasting the patterns of variations in the stock return movements.

Zhang et al. propose applying a *multi-layer backpropagation* (BP) neural network in financial data mining (Zhang et al., 2007). The proposed scheme is a modified neural network-based forecasting model that carries out intelligent mining tasks. The system makes robust forecasting on the buying and selling signs according to the prediction of future trends. Simulations are carried out on the Shanghai composite index values. The results show that the return yielded by the system is three times that is achieved using the *buy-and-hold strategy*.

Basalto et al. propose an approach of pair-wise clustering to analyze the Dow Jones Index companies so that similar temporal behavior of the traded stock prices can be identified (Basalto et al., 2005). The main goal of the authors is to investigate and understand the dynamics that govern companies' stock prices. The proposed scheme deploys a pair-wise version of the *chaotic map algorithm*. The algorithm uses the correlation coefficient values between the financial time series and finds the *similarity measures* for clustering the temporal patterns. The resultant dynamics of such systems form the clusters of companies that belong to different industrial branches. The authors argue that these clusters of companies can be gainfully exploited to optimize portfolio construction.

Chen et al. propose a model for predicting the direction of return on the Taiwan Stock Exchange index (Chen et al., 2003). The authors contend that stock trading guided by robust forecasting models is more effective and usually leads to a higher return on investment. To build a robust forecasting model, the authors train a *probabilistic neural network* (PNN) using historical stock market data. The forecasted output of the model is used to

form various index trading strategies. The effectiveness of the strategies is compared with those generated by the *buy and hold* strategy, the investment strategy using a *random walk model*, and the parametric *generalized method of moments* (GMM) with a Kalman filter. The results show that the investment strategies using the output of the PPN yield the highest return of investment in the long run.

de Faria et al. propose a predictive model using a neural network and an *adaptive exponential smoothing method* to forecast the Brazilian stock market (de Faria et al., 2009). Focusing on the sign of the market returns, the authors compare the forecasting accuracy of the neural network with an exponential smoothing model. The simulation results show that both methods are equally efficient in predicting the index returns. However, the neural network model is found to be more accurate in predicting the market movement than the *adaptive exponential smoothing model*.

Leigh et al. build *linear regression* and *simple neural network* models for forecasting the stock market indices of the New York Stock Exchange for the period 1981-1999 (Leigh et al., 2005). The proposed scheme by the authors uses a *template matching mechanism* based on statistical pattern recognition that efficiently and accurately identifies the spikes in the trading volumes. A threshold limit for the spike in volume is determined, and the days on which the traded volume exhibits significant spikes are identified. A linear regression model is applied to forecast the changes in the future price, the traded volume, and the prime interest rate.

Shen et al. propose a novel scheme based on a *tapped delay neural network* (TDNN) with *adaptive learning* and *pruning* for forecasting a nonlinear time series of stock price (Shen et al., 2007). The TDDN model is trained by a *recursive least square* (RLS) technique that involves a *tunable learning rate,* ensuring a faster network convergence. The trained neural network model is optimized using a pruning algorithm that reduces the possibility of model overfitting. The experimental results in a simulated environment clearly show that the pruned model has reduced complexity, faster execution, and improved prediction accuracy.

Ning et al. propose a stock index prediction scheme based on a *chaotic neural network* model (Ning et al., 2009). Data from a Chinese stock market and a Shenzhen stock market are used for building the model. The chaotic neural network learns the nonlinear, stochastic, and chaotic patterns in the stock market indices. The learnings of the chaotic neural network are applied in forecasting the future index values of the stock markets.

Hanias et al. conduct a study to predict the daily index values of the Athens Stock Exchange (ASE) using a neural network with *backpropagation* (Hanias et al., 2012). The neural network is used to make *multi-step*

forecasting for nine days and yields a very low *mean square error* (MSE) value of 0.0024.

Wu et al. propose an ensemble model using *support vector machines* (SVM) and *artificial neural networks* (ANN) for predicting future stock prices (Wu et al., 2008). The forecasting performance of the ensemble model is compared with those of the SVM model and the ANN model. It is observed from the experimental results that the ensemble model produces the most accurate results among all three models.

Liao et al. carry out a detailed study on the stock market investment issues in the Taiwan stock market (Liao et al., 2008). The authors propose a predictive scheme consisting of two phases. In the first phase, the *apriori algorithm* identifies the *association rules* and the *knowledge patterns* about the stock category association and the possible stock category investment collections. After the association rules are successfully mined, the *k*-means clustering algorithm identifies the various clusters of stocks in the second phase. The authors also propose several stock market portfolio alternatives.

Zhu et al. hypothesize a significant *bidirectional nonlinear causality* between the stock returns and the trading volumes (Zhu et al., 2008). The authors use a neural network-based scheme for forecasting the movement patterns of the stock index. The model is further enriched by incorporating different combinations of indices and the volumes of the component stocks as additional inputs. The National Association of Securities Dealers Automated Quotations (NASDAQ), Dow Jones Industrial Average (DJIA), and Straits Times Index (STI) data of stock prices and volume of transactions are used in training the neural network. The experimental results demonstrate that the augmented neural networks with trading volumes yield a distinct improvement in forecasting performance under different forecasting horizons.

Bentes et al. present a study on the long memory and volatility clustering for the Standard and Poor's 500 (SP 500), NASDAQ 100, and EURO STOXX 50 indexes to compare the US and the European markets (Bentes et al., 2008). The authors compare the performance of two different approaches. The first approach is based on traditional methods using *generalized autoregressive conditional heteroscedasticity* GARCH(1, 1), IGARCH(1, 1), and FIGARCH (1, d, 1). On the other hand, the second approach utilizes the concept of *entropy* in *Econophysics*. In the second approach, three different measures are considered. The three measures are (i) Shannon, (ii) Renyi, and (iii) Tsallis measure. The results indicate nonlinearity and volatility of SP 500, NASDAQ 100, and EURO STOXX 50 indexes.

Chen et al. demonstrate how the random and chaotic behavior of stock price movement can be effectively modeled using a *local linear wavelet neural network* (LLWNN) method (Chen et al., 2005). The proposed wavelet-based model is optimized using a novel algorithm known as the *estimation of distribution algorithm* (EDA). The purpose of the model is to predict the stock price for the following trade day accurately, given the *open*, *close,* and *high* values of the stock price on a given day. The study proves that even for a time series that exhibits a high degree of random fluctuations in its values, it is possible to extract important features from the *open, close,* and *high* values of the stock index, enabling an accurate prediction of its future behavior.

Hutchinson et al. propose a non-parametric method for estimating a derivative's pricing formula using the *principles of learning networks* (Hutchinson et al., 1994). The input variables used in the model are as follows: (i) the current fundamental asset price, (ii) the strike price, and (iii) the time to maturity. These variables have a direct influence on the derivative price. The learning network maps the input values to their output values. For training the model, the authors use a dataset consisting of the daily *closing prices* of the SP 500 futures and the *options prices* for the five years from January 1987 to December 1991. To evaluate the efficacy and the efficiency of various models, the authors compare the performance of four models: (i) *ordinary least squares*, (ii) *radial basis function networks*, (iii) *multi-layer feed-forward neural networks*, and (iv) the *projection pursuit*. The simulation results indicate that the *non-parametric model* is the most accurate one.

Dutta et al. illustrate how ANN models can be applied in forecasting the Bombay Stock Exchange's SENSEX *weekly closing values* from January 2002 to December 2003 (Dutta et al., 2006). The proposed approach involves building two neural networks consisting of three hidden layers and the input and the output layers. The inputs to the first neural network are: (i) the *weekly closing values*, (ii) the *52-week moving average of the weekly closing* SENSEX values, (iii) the *5-week moving average of the closing values*, and (iv) the *10-week oscillator values for the past 200 weeks*. On the other hand, the second network is provided with the following inputs: (i) the *weekly closing value of* SENSEX, (ii) the *moving average of the weekly closing values computed on the 52-week historical data*, (iii) the *moving average of the closing values computed on the 5-week historical data*, and (iv) the *volatility of the* SENSEX *records computed on 5-week basis over the past 200 weeks*. The forecasting performances of the two neural networks are compared using their *root mean square error* (RMSE) and *mean absolute error* (MSE) values on the test data. The performances

of the networks are tested on the weekly *closing* SENSEX values from January 2002 to December 2003.

Hammad et al. demonstrate that an *artificial neural network* (ANN) model can be trained to converge to an optimal solution while maintaining a very high level of precision in the forecasting of stock prices (Hammad et al., 2007). The proposed scheme is based on a *multi-layer feed-forward neural network* model that uses the backpropagation algorithm. The model is used for forecasting the Jordanian stock prices. Simulations are carried out on seven Jordanian companies from the service and manufacturing sectors using the MATLAB tool. The accuracy of the model in forecasting stock price movement is found to be quite high.

Tsai and Wang show how Bayesian Network-based approaches can produce better forecasting results than the traditional regression and neural network-based approaches (Tsai & Wang, 2009). The authors propose a hybrid predictive model for stock price forecasting that combines a neural network-based model with a decision tree. The experimental results indicate that the hybrid model is more accurate in predicting future stock prices than the single ANN and the single decision tree-based approach.

Tseng et al. propose various predictive models for stock price prediction (Tseng et al., 2012). The proposition includes the following models: (i) a traditional *time series decomposition* (TSD) model, (ii) a *HoltWinters* (H/W) *exponential smoothing with trend and seasonality models*, (iii) *Box-Jenkins* (B/J) *models* using *autocorrelation* and *partial autocorrelation*, and (iv) *neural network-based models*. The authors train the models on the stock price data of 50 randomly chosen stocks from September 1, 1998, to December 31, 2010. The models are trained on 3105 observations of the *close* prices of the stocks. The testing of the model is carried out on data for 60 trading days. The study shows that the forecasting accuracies are higher for the B/J, the H/W, and the normalized neural network models. The errors associated with the time series decomposition-based model and the non-normalized neural network models are found to be higher.

Senol and Ozturan show how a well-trained ANN model can be used to predict stock prices and their direction of change (Senol & Ozturan, 2008). The experimental results yield an accuracy of 81%.

In the literature, a substantial number of contributions exist based on the application of time series and fuzzy time series approaches for forecasting stock price movements. Thenmozhi investigates the applicability of chaos theory in modeling the Bombay Stock Exchange (BSE) time series (Thenmozhi, 2006). The author uses the return values of the BSE SENSEX time series data from August 1980 to September 1997 and shows that the

time series of the daily and the weekly return values exhibit nonlinearity and chaotic properties.

Fu et al. propose an approach to representing the data points in a financial time series according to their importance (Fu et al., 2008). Using the ranked data points, the authors construct a tree that enables incremental updating of data in the time series. The scheme facilitates representing a large-sized time series in different levels of detail and allows multi-resolution dimensionality reduction. The authors present several evaluation methods of the importance of data points and a novel process of updating a time series. Extensive experimental results are presented, demonstrating the effectiveness of all the propositions.

Phua et al. present a predictive model using neural networks with genetic algorithms for forecasting stock price movements in the Singapore Stock Exchange (Phua et al., 2001). The predictive model's forecasting accuracy on the test dataset is 81%, indicating the model's effectiveness.

Moshiri and Cameron describe a backpropagation-based neural network and a set of econometric models to forecast inflation levels (Moshiri, & Cameron, 2010). The set of econometric models proposed by the authors includes the following: (i) Box-Jenkins *autoregressive integrated moving average* (ARIMA) model, (ii) *vector autoregression* (VAR) model, and (iii) Bayesian vector autoregression (BVAR) model. The forecasting accuracies of the three models are compared with the hybrid *back propagation network* (BPN) model proposed by the authors. The models are tested on three different forecasting horizons: one month, two months, and twelve months. With the *root mean square error* (RMSE) and the *mean absolute error* (MAE) as the two metrics, the authors observe that the hybrid BPN is superior to the other econometric models.

The major drawback of the existing propositions in literature for stock price prediction is their inability to predict stock price movement in a short-term interval. The current work attempts to address this shortcoming by exploiting the learning ability of a gamut of machine learning and a deep neural network in stock price movement modeling and prediction.

Methodology

In the *Introduction* section, we mentioned that this work aims to develop a robust forecasting framework for the short-term price movement of stocks. The Metastock tool is used for collecting data on the short-term price movement of stocks (Metastock). We collect the stock data for the company – Godrej Consumer Products Ltd. The stock price data are collected at every five-minute interval in a day for all the days when the

National Stock Exchange (NSE) was open during 2013 and 2014. The raw data of the stock price consist of the following variables: (i) *date*, (ii) *time*, (iii) *open* value of the stock, (iv) *high* value of the stock, (v) *low* value of the stock, (vi) *close* value of the stock, and (viii) the *volume* of the stock traded in a given interval. The variable *time* refers to the time at which a particular record is collected. The time interval between two successive records in the raw data is, therefore, 5 minutes. The raw data in this format is collected for the stock Godrej Consumer Products. for two years. In addition to the six variables in the raw data that we have mentioned, we also collect the NIFTY index at 5 minutes intervals for the same period of two years to capture the overall market sentiment at each time instant, so that more accurate and robust forecasting can be made using the combined information of historical stock prices and the market sentiment index. As an interval of 5 minutes is too granular, we carry out an *aggregation* operation on the raw stock price data. We break the total time interval in a day into three slots as follows: (1) the morning slot covering the time interval from 9:00 AM to 11:30 AM, (2) the afternoon slot starting at 11:35 AM and continuing till 1:30 PM, and (3) the evening slot from 1:35 PM till the time of closure of the stock exchange on a given day. Hence, after the aggregation operation, the stock price data consist of three records for a given day.

Using the eight variables in the raw data, we design eleven derived variables. These derived variables are used as the inputs for building the predictive models. We follow two approaches to stock price forecasting - *regression* and *classification*. The difference between these two approaches is how the response variable *open_perc* is used in model building. We will describe this point in more detail later in this section. We compute the following eleven derived variables.

month: it is a numeric variable that refers to the month for a given stock price record. The twelve months are assigned numeric codes of 1 through 12, January is coded as 1, and the month of December is assigned with 12.

day_month: this numeric variable denotes the particular day of a given month to which a stock price record corresponds. The value of this variable lies in the interval [1, 31]. For instance, if the date for a stock price record is May 22, 2013, then the *day_month* variable for that record will be assigned a value of 22.

day_week: it is a numeric variable that corresponds to the day of the week for a given stock price record. The five days in a week on which the stock market remains open are assigned numeric codes of 1 through 5. Monday is assigned a code of 1, while Friday is coded as 5.

time: this numeric variable refers to the time slot to which a stock price record belongs. There are three slots in a day - *morning*, *afternoon*, and

evening. The slots are assigned codes 1, 2, and 3, respectively. For example, if a stock price record refers to the time 3:45 PM, the variable *time* will be assigned a value of 3 for the stock price record.

open_perc: it is a numeric variable whose value is computed as a percentage change in the stock's *open* price over two successive time slots. The computation of the variable is done as follows. Suppose we have two consecutive time slots S_1 and S_2. Both of them consist of several records at five minutes intervals. Let the *open* price of the stock for the first record of S_1 be O_1 and that for S_2 be O_2. The *open_perc* for slot S_2 is computed using $(O_2 - O_1)/O_2$.

high_perc: It is a numeric variable whose value is computed as a percentage change in the *high* values of a stock over two consecutive time slots. The computation of the variable is done as follows. Suppose we have two successive slots, S_1 and S_2. Both of them consist of several records at five minutes intervals. We compute the mean of all *high* values of the records in both slots. If H_1 and H_2 are the mean *high* values for S_1 and S_2, then *high_perc* for slot S_2 is computed using $(H_2 - H_1)/H_2$.

low_perc: It is a numeric variable whose value is computed as a percentage change in the *low* values of a stock over two successive time slots. For two consecutive slots S_1 and S_2, first, we calculate the mean of all *low* values of the records in both slots. If L_1 and L_2 refer to the mean of the *low* values for S_1 and S_2, then *low_perc* for slot S_2 is computed using $(L_2 - L_1)/L_2$.

close_perc: It is a numeric value computed as a percentage change in the *close* values of the stock over two successive time slots. The computation of the variable is done as follows. Suppose we have two consecutive slots S_1 and S_2. Both of them consist of several records at five minutes intervals. Let the *open* price of the stock for the first record of S_1 be C_1, and that for S_2 be C_2. The *close_perc* for slot S_2 is computed using $(C_2 - C_1)/C_2$.

vol_perc: It is a numeric variable whose value is computed as a percentage change in the *volume* values of a stock over two consecutive time slots. The computation of the variable is done as follows. Suppose we have two successive slots, S_1 and S_2. Both of them consist of several records at five minutes intervals. We compute the mean of all *volume* values of the records in both slots. If V_1 and V_2 are the *volume* values for S_1 and S_2, then *vol_perc* for slot S_2 is computed using $(V_2 - V_1)/V_2$.

nifty_perc: It is a numeric variable whose value is computed as a percentage change in the NIFTY index over two successive time slots. The computation of the variable is done as follows. Suppose we have two consecutive slots S_1 and S_2. Both of them consist of several records at five

minutes intervals. We compute the mean of all NIFTY index values of the records in both slots. If N_1 and N_2 are the mean NIFTY index values for S_1 and S_2, then *nifty_perc* for slot S_2 is computed using $(N_2 - N_1)/N_2$.

range_diff: It is a numeric variable whose value is computed as the difference in the *range* values of two consecutive time slots. The *range* value for a given slot is the difference between its *high* and *low* values. If S_1 and S_2 denote two successive slots, and if H_1, H_2, L_1, and L_2 respectively represent the *high* and the *low* values of slots S_1 and S_2, then the *range* value for S_1 is $R_1 = (H_1 - L_1)$ and for S_2 is $R_2 = (H_2 - L_2)$. The *range_diff* for slot S_2 is computed using $(R_2 - R_1)$.

After we compute the above eleven variables for each time slot, we develop the forecasting framework. As mentioned earlier, we follow two broad approaches to forecasting future stock price movements and stock price values. The two approaches are *regression* and *classification*.

In the regression approach, based on the historical movement of the stock prices, we predict the stock price for the next slot. We use *open_perc* as the response variable, which is a continuous numeric variable. The objective of the regression technique is to predict the *open_perc* value for the next slot given the past stock movement pattern and the values of the predictors. In other words, if the current time slot is S_1, the regression methods attempt to predict *open_perc* for the next slot, S_2. If the predicted *open_perc* is positive, it indicates an expected rise in the stock price in S_2. A negative *open_perc* will mean a fall in the stock price in the next slot. Based on the predicted values of *open_perc*, a potential investor can make her investment strategy.

In the classification approach, the response variable *open_perc* is a discrete variable having two class labels. For developing the classification-based forecasting approaches, we transform *open_perc* into a categorical variable with two labels, "0" and "1". The value "0" indicates a negative *open_perc* value, while "1" denotes a positive *open_perc* value. Hence, if the current slot is S_1 and the model expects a rise in the *open_perc* value in the next slot, S_2, then the *open_perc* for S_2 is assigned a label "1". A "0" label for the *open_perc* indicates an expected negative value of the *open_perc* in the next slot.

For both classification and regression approaches, we consider three cases. The three cases are discussed below.

***Case I*:** For *Case I*, we use the dataset of the year 2013. The dataset consists of 19385 records at five minutes intervals. These records are aggregated into 745 records so that each aggregate record corresponds to one of three slots in a day. The aggregated records are used for building the

predictive models. We use the same dataset for evaluating the training accuracies of the models.

***Case II*:** We use the dataset of the year 2014 dataset in this case. The dataset consists of 18972 records at five minutes intervals. These granular data are aggregated into 725 records, where each record corresponds to one of the three slots in a day. Using these 725 records, we build the predictive models. We use the same dataset for evaluating the training performance of the models.

***Case III*:** We use the stock price dataset of 2013 as the training dataset for building the models. The models are tested using the dataset of the year 2014. We, again, carry out an analysis of the performance of different models.

We build eight classification and nine regression models for developing our forecasting framework. The classification models are: (i) *logistic regression*, (ii) *k-nearest neighbor* (iii) *decision tree*, (iv) *bagging*, (v) *boosting*, (vi) *random forest*, (vii) *artificial neural network*, and (viii) *support vector machines*. We use several metrics for evaluating the performance of the classification models. These metrics are *sensitivity*, *specificity*, *positive predictive value*, *negative predictive value*, *classification accuracy*, and *F1-score*. *Sensitivity* and *positive predictive value* are also known as *recall* and *precision,* respectively.

The nine regression models are the following: (i) *multivariate regression*, (ii) *multivariate adaptive regression spline*, (iii) *decision tree*, (iv) *bagging*, (v) *boosting*, (vi) *random forest*, (vii) *artificial neural network*, (viii) *support vector machine*, and (ix) *long- and short-term memory network.*

While all the classification models are built on machine learning algorithms, one of the regression models, the *long-and short-term memory* (LSTM) network, is a deep learning model. For comparing the performance of the regression methods, we use several metrics such as *root mean square error* (RMSE) and the *correlation coefficient* between the actual and predicted values of the response variable, e.g., *open_perc*.

Machine Learning Models

First, we discuss the design details of the eight classification models.

***Logistic Regression*:** This is a classification method, and therefore, we transform the response variable *open_perc* into a discrete domain from a continuous domain. In other words, we change the response variable into a categorical type with two labels, “0” or “1”. We assign all records with negative or zero values of *open_perc* to class “0”, and all other records are

set to class "1". We use the function *glm* in R for building the logistic regression models. The three parameters passed to the *glm* function are: (i) the first parameter is the formula, *open_perc* ~., which specifies *open_perc* as the response variable and the remaining variables as the predictors, (ii) the second parameter is "*family = binomial*" indicating that the model is a binary logistic regression with two classes, and (iii) the third parameter is the R data object containing the records of the training dataset. We use the *predict* function in R to compute the probability of the test records belonging to the two classes. We assume a threshold value of 0.5 as the probability. In other words, when the probability of a record belonging to a class exceeds 0.5, we assume that the record belongs to that class.

***K-Nearest Neighbor*:** The *K- nearest neighbor* (KNN) is an example of *instance-based learning*. Based on the training, the classification of a new unclassified record is done by comparing it to the most similar records in the training dataset. The value of *k* determines how many closest similar records in the training data set are considered for classifying a record of the test data. We use the R function *knn* defined in the library *class* to build the KNN classification models on the stock price data. The stock price data are normalized using the *min-max* normalization method before applying the *knn* function so that all predictors are scaled down into the same range of values. Different values of *k* are tried out for building the models, and the value of $k = 3$ is finally chosen. This value of *k* is found to produce the best performance of the model with the minimum probability of model overfitting.

***Decision Tree*:** The *classification and regression tree* (CART) algorithm produces decision trees that are strictly binary so that there are precisely two branches at each node. The algorithm recursively partitions the records in the training dataset into subsets of records with similar values for the target attributes. The trees are constructed by carrying out an exhaustive search on each node for all available variables and all possible splitting values. The optimal split is selected based on some *goodness of split* criteria. We use the *tree* function defined in the *tree* library of R for building the decision tree models.

***Bagging*:** *Bootstrap Aggregation* (Bagging) is an ensemble technique. It works as follows: Given a set *D*, of *d* tuples, for iteration *i*, a training set, D_i of *d* tuples is sampled with replacement from the original set of tuples *D*. Each training set represents a bootstrap sample. Since the samples are simple random samples *with replacement*, it is possible that some records (i.e., tuples) in *D* may not get a chance to be included in D_i, while some tuples may get included in more than one sample. A classifier model M_i is trained on the information contained in each training set, D_i. For classifying

an unknown tuple X in the *out-of-sample* set (i.e., in the test dataset), each classifier, M_i, is asked to return its class predictions. The classification result of each of the trained classifiers is considered as one vote. The bagging classifier counts the votes and finally assigns the class with the maximum number of votes to the tuple X. We use the *bagging* function defined in the *ipred* library of the R programming language for building the bagging classification models. The value of the parameter *nbag*, which specifies the number of samples, is chosen to be 25.

***Boosting*:** Unlike bagging, *boosting* assigns weights to each tuple in a training dataset. Based on the training dataset, k classification models are built iteratively. However, all the classifiers are not given equal importance in the final classification decision. While bagging uses a *simple majority voting* among the classifiers, boosting employs a weighted majority voting technique. After a classifier M_i is constructed, the weights assigned to the classifiers are updated before building the subsequent classifier M_{i+1}. After the current iteration is over, the classifiers that could correctly classify the tuples misclassified in the previous round are assigned higher weights before the next iteration starts. As the final iteration is finished, the boosted classifier model combines the weighted votes of each classifier. The weights are computed based on some functions of the classification accuracies of the results reported by the individual classifier. *Adaptive Boosting* (AdaBoost) is a very popular variant of boosting for classification tasks. We use the *boosting* function of the *adabag* library in R for building the *Adaboost* models for classification.

***Random Forest*:** Random forest is an ensemble machine learning approach. The algorithm first builds many decision tree classifiers separately so that the collection of the classifiers is a *forest*. The decision tree classifier models are created based on a random selection of the attributes at each node. The splitting at each node is done by selecting one feature randomly and then arbitrarily choosing a value for the feature. This process introduces a high level of randomness in building the individual classifier in the ensemble model. In other words, each decision tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The objective of introducing so much randomness in building the decision tree models is to avoid model overfitting during the training phase. During the classification phase, each tree votes, and the most popular class is returned. We use the *randomForest* function defined in the *randomForest* library in R to build the random forest classification models.

***Artificial Neural Network*:** An *artificial neural network* (ANN) is a connectionist network that consists of nodes and their interconnecting links

where the nodes are arranged in several layers - an *input* layer, one or more *hidden* layers, and an *output* layer. The input layer nodes correspond to the predictor variables (i.e., attributes) in the training dataset. The inputs are fed simultaneously into the units making up the input layer. The input values pass through the respective nodes in the input layer. They are then weighted using the weights associated with the links connecting the nodes and fed simultaneously to the second layer of nodes, known as the hidden layer nodes. The outputs of the nodes in the first hidden layer are weighted again using the corresponding link weights. The resultant values are provided as the inputs to a possible second hidden layer and so on. The weighted outputs of the last hidden layer are input to units making up the output layer, which produces the network's prediction for given tuples. We use the *neuralnet* function defined in the *neuralnet* library in R for building the ANN classification models for the stock price data. The raw data is normalized using the *min-max* normalization approach. Only the predictors are normalized; the response variable: *open_perc* is kept unchanged. The parameter *hidden* in the function *neuralnet* is set to different values for changing the number of nodes in the hidden layer in the network. The parameter *stepmax* is set to the maximum value of 10^6 so that the maximum number of iteration capability of the *neuralnet* function may be utilized. For building a classification model, we set the parameter *linear.output* to FALSE in the *neuralnet* function.

***Support Vector Machine*:** A *support vector machin*e (SVM) is a machine learning model for classification and regression. When applied to classification tasks, it can classify both linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new higher dimension, it searches for the linear optimal hyperplane that separates the two classes. SVM finds this hyperplane using *support vectors,* which are the essential and the discriminating training tuples to separate the two classes. We use the *ksvm* function defined in the *kernlab* library in the R language for building the SVM models. The function *ksvm* has an optional parameter called *kernel,* which is set to *vanilladot* in our implementation.

We now briefly discuss the regression models.

***Multivariate Regression*:** As in all regression models, we use *open_perc* as the response variable and the remaining ten variables as the predictors to build the predictive models for three cases mentioned earlier in the section titled *Methodology*.

Case I: We use the stock price data of 2013 to train the regression model and test the model using the same dataset. Two approaches are followed in building the multivariate regression models: (i) *backward*

deletion and (ii) *forward addition* of variables. However, both methods yield the same results.

To eliminate the multicollinearity problem, we apply the *vif* function defined in the *faraway* library and identify the collinear predictors. The *variance inflation factor* (VIF) values of the predictors are found to be as follows: *month* = 1.003, *day_month* = 1.008, *day_week* = 1.002, *time* = 1.095, *high_perc* = 4372. 547, *low_perc* = 4369.694, *close_perc* = 165.436, *vol_perc* = 1.072, *nifty_perc* = 1.046, *range_diff* = 156.198. Hence, it is clear that *high_perc*, *low_perc*, *close_perc,* and *range_diff* exhibit multicollinearity. We retain *low_perc* and *range_diff* and remove the other two variables since their VIF values are smaller than the other two. Using the *drop1* function in the *backward deletion* method and the *add1* function in the forward addition technique, we identify the variables that are not significant and do not contribute to the information content of the model. We use the *Akaike Information Criteria* (AIC) for identifying the variables that contribute the least to the information content of the model. In the *backward deletion* method, the variable yielding the lowest AIC and a non-significant *p*-value is removed from the model at the end of every iteration. On the other hand, the *forward addition method* adds the variable with the lowest AIC and a significant *p*-value at each iteration. The two predictors that finally remain in the regression model are *low_perc* and *range_diff*.

***Case II*:** Using the stock price data of the year 2014, we compute the VIF values for the predictors. The VIF values are found to as follows: *month* = 1.007, *day_month* = 1.004, *day_week* = 1.007, *time* = 1.057, *high_perc* = 1161.446, *low_perc* = 1331.035, *close_perc* = 115.161, *vol_perc* = 1.022, *range_diff* = 92.092, *nifty_perc* = 1.073. The variables exhibiting multicollinearity are *high_perc*, *low_perc*, *close_perc*, and *range_diff*. As in *Case I*, we retain *low_perc* and *range_diff* as their VIF values are smaller than the other two. The use of *backward deletion* and *forward addition* methods both yield a model identical to the model of *Case I*. The model includes *low_perc* and *range_diff* as the predictors and *open_perc* as the response variable.

***Case III*:** In this case, the model is identical to that in *Case I*. However, the model is tested on the 2014 dataset. Therefore, the performance results of the model are expected to be different. We present the performance results and their critical analysis in the section titled *Performance Results and Analysis*.

***Multivariate Adaptive Regression Spline*:** *Multivariate Adaptive Regression Spline* (MARS) is a machine learning approach for building robust regression models. MARS works by splitting the input variables into multiple *basis functions* and then fitting a linear regression model to those

basis functions. The *basis functions* used by MARS are designed in pairs: $f(x) = \{x - t, if\ x > t, 0\ otherwise\}$ and $g(x) = \{t - x, if\ x < t, 0\ otherwise\}$. The main property of the *basis functions* is that these functions are *piecewise linear functions*. The value t for which the two functions meet is called a *knot*. The working principles of MARS are very similar to that of CART. Like CART, MARS first builds a complex model involving many basis functions separated from each other by a large number of knots. This phase of the algorithm execution is called the *forward pass* of the model building. In the subsequent phase, known as the *backward pass*, the algorithm prunes back unimportant terms (i.e., the basis functions), which could not significantly contribute to the model's generalized R^2 values. This phase essentially enables MARS to avoid *overfitting* during the training phase. During the backward pass, the algorithm computes the *generalized cross-validation* (GCV) values to determine how well the model fits into the data while avoiding possible overfitting. Finally, the algorithm returns the model with the optimum *cost/benefit* ratio. To fit a MARS model using the R language, we use the function *earth* defined in the library *earth*.

Decision Tree: For building the decision tree regression models, we use the same *tree* function in the *tree* library in R as we did in decision tree classification. However, in this case, the response variable is of a numeric type. The *predict* function is used to predict the values of the response variable. The functions *cor* and *rmse* defined in the library *Metrics* are used to compute the correlation coefficient between the actual and the predicted values of the response variable and the RMSE of the model.

***Bagging*:** We use the *bagging* function defined in the *ipred* library of R for building the bagging regression models. The value of the parameter *nbag*, which specifies the number of samples, is set to 100. The functions *predict,* and *rmse* are used to compute the predicted values of the response and the RMSE of the model, respectively. The *cor* function defined in the *Metrics* library in the R programming language is used to calculate the correlation coefficient between the actual and the predicted values of the response variable.

***Boosting*:** We use the *blackboost* function defined in the *mboost* library in R for building the *gradient-boosting* regression models. It may be noted that we used the *boosting* function of the *adabag* library in R for building the *Adaboost* model for classification. As in other regression models, the *predict* and *rmse* functions are used to compute the predicted values and the RMSE of the model. The *cor* function is used to calculate the correlation coefficient between the actual and the predicted values of the response variable, *open_perc*.

***Random Forest*:** We use the *randomForest* function defined in the *randomForest* library in R for building the random forest regression model. The response variable *open_perc* is a numeric variable here, unlike the random forest classification model in which the response variable was categorical. The same *predict* and *rmse* functions are used for computing the predicted values and the RMSE of the model. The correlation coefficient between the actual and the predicted values of the response variable is calculated using the *cor* function in R.

***Artificial Neural Network*:** As in the case of classification, for building the ANN regression model on the stock data, we use the *neuralnet* function defined in the *neuralnet* library in R. The predictors are normalized using *min-max* normalization before building the model. The *compute* function defined in the *neuralnet* library is used for computing the predicted values. The parameter *hidden* is used to change the number of nodes in the hidden layer. The value of the parameter *stepmax* is set to 10^6 to exploit the maximum number of iterations executed by the *neuralnet* function. The parameter *linear.output* is by default set to TRUE, and hence it is not altered. It is observed that we need only one node in the hidden layer in all three cases for building ANN regression models.

***Support Vector Machine*:** Using the *svm* function defined in the *e1071* library in the R language, we build the SVM regression models. The *predict* function is used for predicting the response variable values, and the *rmse* function is used to compute the RMSE of the model. The correlation coefficient between the actual and the predicted values of the response variable is computed using the *cor* function.

Deep Learning Model

This section discusses the principles of the *long-and short-term memory* (LSTM) network model of regression.

***Long- and Short-Term Memory Network*:** LSTM is a variant of *recurrent neural networks* (RNNs) - neural networks with feedback loops (Geron, 2019). In such networks, output at the current time slot depends on the current inputs and the previous state of the network. However, RNNs suffer from the problem that these networks cannot capture long-term dependencies due to *vanishing* or *exploding gradients* during the backpropagation in learning the weights of the links (Geron, 2019). LSTM networks overcome such problems, and hence such networks are pretty helpful for forecasting multivariate time series. LSTM networks consist of memory cells that can maintain their states over time using memory and gating units that regulate the information flow into and out of the memory.

There are different variants of *gates* used. The *forget gates* control what information to throw away from memory. The *input gates* are meant to control the new information added to the cell state from the current input. The cell state vector aggregates the two components - the old memory from the forget gate and the new memory from the input gate. In the end, the *output gates* conditionally decide what to output from the memory cells. The architecture of an LSTM network and the *backpropagation through time* (BPTT) algorithm for learning provides such networks with the ability to learn and forecast in a multivariate time series framework. We use Python programming language and the Tensorflow deep learning framework for implementing the LSTM networks and utilize those networks to predict the future values and the movement patterns of the stock price. For this purpose, we use the *open* price of the stocks as the response variable. The chosen predictors are – *high*, *low*, *close*, *volume,* and the NIFTY index values. However, in this case, we do not compute the differences between successive slots. Instead, we forecast the *next slot's open value* based on the predictor values in the previous slots.

We use the *mean absolute error* (MAE) for computing the error and the *adaptive moment estimation* (Adam) as the optimizer for evaluating the model performance in all three cases. *Adam* computes adaptive learning rates for each parameter in the gradient descent algorithm. In addition to storing an exponentially decaying average of the past squared gradients, *Adam* also keeps track of the exponentially decaying average of the past gradients. This strategy of *Adam* helps in building momentum in the learning process. Instead of behaving like a ball running down a steep slope like momentum, *Adam* manifests itself like a heavy ball with a rough outer surface. This high level of friction results in *Adam*'s preference for a flat minimum in the error surface. Due to its ability to integrate adaptive learning with momentum, *Adam* is found to perform very efficiently in optimizing the performance of large-scale networks. For this reason, we use the *Adam* optimizer in our proposed LSTM model. However, we train LSTM models using different epoch values and batch sizes for the three cases and determine the network's optimum performance under those parameter values. The *sequential* constructor in the Tensorflow framework is used to build the LSTM model. The performance results of the LSTM regression models are presented in the section titled *Performance Results and Analysis*.

Performance Results and Analysis

This section provides a detailed discussion on the forecasting techniques that we have used and the results obtained using those techniques. We first discuss the classification techniques and then the regression techniques. For all three cases, we compute the accuracy of the classification models using several metrics. We define the metrics below.

Sensitivity: The ratio of the number of *true positives* to the total number of *positives* in a dataset is called *sensitivity*. It is often expressed as a percentage. Here, *positive* refers to the cases that belong to the target group (i.e., the class "1"). The term *true positives* refers to the number of *positive* instances the model correctly identified. *Sensitivity* is also sometimes referred to as *recall*. *Sensitivity* is computed using (1).

$$Sensitivity = \frac{Number\ of\ true\ positives}{Total\ number\ of\ positives} = \frac{Number\ of\ true\ positives}{Number\ of\ true\ positives + Number\ of\ flase\ negatives} \quad (1)$$

***Specificity*:** The ratio of the number of true negatives to the total number of negatives in the dataset is called *specificity*. Often, it is expressed as a percentage. Here, *negative* refers to the cases that belong to the non-target group (i.e., the class "0"). The term *true negative* refers to the number of negative cases the model correctly identified. *Specificity* is computed using (2).

$$Specificity = \frac{Number\ of\ true\ negatives}{Total\ number\ of\ negatives} = \frac{Number\ of\ true\ negatives}{Number\ teue\ negatives + Number\ of\ false\ positives} \quad (2)$$

***Positive Predictive Value*:** *Positive predictive ratio* (PPV), sometimes referred to as *precision*, indicates the model's accuracy in classifying the target group cases among the total number of target group cases identified by the model. PPV is computed as the ratio of the number of correctly classified target group cases to the total number of target group cases as determined by the model. Since the total number of target group cases identified by the model is the sum of the number of *true positive* cases and the number of *false-positiv*e cases, PPV is the ratio of the total number of true positive cases to the sum of the number of true positive cases and the number of false-positive cases. Most often, PPV is expressed as a percentage. The complement of PPV is also called the *false discovery rate* (FDR). PPV and FDR are computed using (3) and (4), respectively.

$$PPV = \frac{Numebr\ of\ true\ positives}{Number\ of\ true\ positives + Number\ of\ false\ positives} = \frac{Number\ of\ true\ positives}{Total\ Number\ of\ positive\ calls} \quad (3)$$

$$FDR = 1 - PPV = \frac{No\ of\ fasle\ positives}{No\ of\ true\ positives + No\ of\ fasle\ positives} = \frac{No\ of\ false\ positives}{Total\ Number\ of\ positive\ calls} \quad (4)$$

***Negative Predictive Value*:** *Negative predictive value* (NPV) refers to the accuracy of the model in classifying the non-target group cases among the total number of non-target elements identified by it. It is computed as the ratio of the number of correctly classified non-target group cases to the total number of non-target group cases identified by the model. Since the total number of non-target group cases identified by the models is the sum of the number of true negative cases and the number of false-negative cases, NPV is the ratio of the total number of true negative cases to the sum of the number of true negative cases and the number of false-negative cases, expressed as a percentage. The complement of NPV is also called *false omission rate* (FOR). NPV and FOR are computed using (5) and (6), respectively.

$$NPV = \frac{Number\ of\ true\ negatives}{Number\ of\ true\ negatives + Number\ of\ false\ negatives} = \frac{Number\ of\ true\ negatives}{Total\ Number\ of\ negative\ calls} \quad (5)$$

$$FOR = 1 - NPV = \frac{No\ of\ false\ negatives}{No\ of\ true\ negatives + No\ of\ flase\ negatives} = \frac{No\ of\ false\ negatives}{Total\ No\ of\ negative\ calls} \quad (6)$$

***Classification Accuracy* (CA):** The ratio of the total number of cases correctly classified by the model to the total number of cases in the dataset is called *classification accuracy*. It is often expressed as a percentage. CA is computed using (7).

$$CA = \frac{Number\ of\ true\ positives +\ Number\ of\ true\ negatives}{Total\ number\ of\ postives + Total\ number\ of\ negatives} \quad (7)$$

F1 Score: If the test data set is highly imbalanced, with the cases belonging to the non-target group far outnumbering the target group cases, the sensitivity value is usually very low even if the classification accuracy may be high. Hence, classification accuracy is not considered a very robust and reliable metric. *F1-score*, which is computed as the harmonic mean of the sensitivity and PPV, is a robust metric. *F1-score* is calculated using (8).

$$F1 = \frac{2\ x\ PPV\ x\ Sensitivity}{(PPV + Sensitivity)} = \frac{2\ x\ Recall*Precision}{(Recall+Precision)} \quad (8)$$

***Classification Methods*:**

***Logistic Regression*:** We use the *glm* function in the R programming language to build logistic regression-based classification models for all three cases. The response variable is transformed into a categorical type using the function *as.factor* before constructing the models. The parameter *family* is set to *binomial* to build a *binary* logistic regression model. The *predict* function is used to predict the label (i.e., the class) of the response variable. We also plot the *lift* curve and the *receiver operating characteristic* (ROC) curve of each case's model. The output of the *performance* function defined in the ROCR library is plotted to illustrate the model's ROC curve. The *area under the curve* (AUC) for each ROC curve is computed using the *auc* function defined in the *pROC* library in the *R* programming language.

Table 6-1 presents the performance results of the logistic regression classification method. For *Case I*, out of 419 actual "0" cases, only 10 cases were misclassified, while among 316 actual "1" cases, 17 cases are found to be misclassified. The value of AUC for the ROC curve for *Case I* is 0.9934. For *Case II*, 16 cases out of 396 actual "0" cases are misclassified. On the other hand, out of 329 cases that are actually "1," are misclassified by the model. The AUC value for this case is found to be 0.9891. *Case III* misclassifies 42 instances which are actually "0" out of a total of 396 cases. On the other hand, among 329 cases that are actually "1", 26 cases are misclassified. The AUC value for *Case III* is found to be 0.9587.

TABLE 6-1. THE PERFORMANCE RESULTS OF THE LOGISTIC REGRESSION CLASSIFICATION MODELS

Metrics	Case I	Case II	Case III
	Training Results 2013	**Training Results 2014**	**Test Results 2014**
Sensitivity	94.79	94.83	92.10
Specificity	97.61	95.96	89.39
PPV	96.87	95.12	87.83
NPV	96.01	95.72	93.16
CA	96.38	95.45	90.62
F1 Score	95.82	94.97	89.91

For the sake of conciseness and avoiding repetitions, we present the performance of each model only for *Case III* since this case represents the performance of each model on the test data. The performance of the model on the test data is considered to be the most critical.

Figure 6-1 depicts the classification performance of the logistic regression model for *Case III*. In Figure 6-1, the *y*-axis represents the *actual classes* of the records (either "0" or "1"), and the *x*-axis denotes the probability that a case will belong to the category "1". The threshold value along the *x*-axis is, by convention, taken to be 0.5. Hence, all the cases found on the level "0" along the *y*-axis and situated to the right of the threshold value of 0.5 along the *x*-axis are misclassified instances.

Similarly, all the points on the level "1" along the *y*-axis and are situated to the left of the threshold value of 0.5 along the *x*-axis are also misclassified. Figure 6-2 presents the lift curve for *Case III*. The *lift curve* is pulled up from the baseline, indicating that the model is very accurate in discriminating between the two classes. Figure 6-3 depicts the ROC curve for the logistic regression model for *Case III*. The curve's steepness makes it evident that the model has effectively optimized the values of the *true positive rate* (TPR) and the *false positive rate* (FPR).In Figure 6-3, the line segment with *red* color presents the class "1" cases that are correctly classified, while the *blue* line segment denotes the correctly classified instances of the class "0". The portion of the ROC curve colored with *yellow* represents the class "0" cases, which the model misclassifies. The *green*-

colored part of the ROC curve depicts the class "1" cases misclassified by the model.

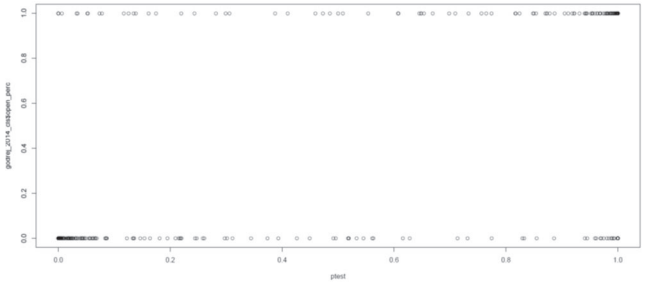

Figure 6-1. The logistic regression model – actual vs. predicted probabilities of *open_perc* (*Case III*)

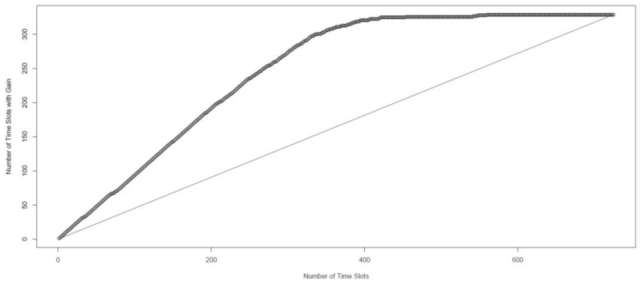

Figure 6-2. The logistic regression model for classification – the *lift curve* (*Case III*)

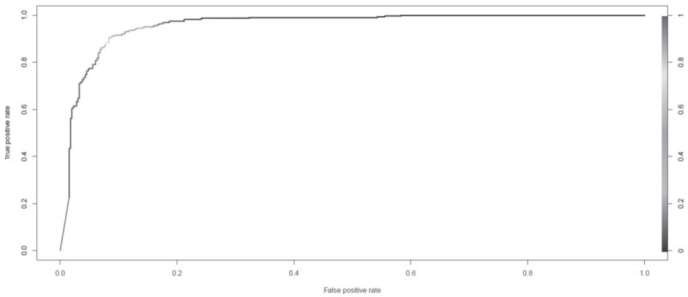

Figure 6-3. The logistic regression model for classification – the *ROC curve* (*Case III*)

KNN Classification: Table 6-2 presents the performance results of the KNN classification method. For *Case I*, with the values of $k = 1, 3, 5, 7$, and 9, the classification accuracy values are found to be 100, 93.42, 91.68, 92.35, and 92.08, respectively. We choose $k = 3$ to avoid the overfitted model with $k = 1$. In this case, 419 instances are of class "0", and 326 cases are of class "1". Among class "0" cases, 51 cases are misclassified, while 34 cases of class "1" are also misclassified. In *Case II*, for $k = 1, 3, 5, 7$, and 9, the classification accuracy values are 100, 90.21, 85.10, 83.22, and 84.16 respectively. Again $k = 3$ is chosen to avoid model overfitting. Twenty-eight actual "0" cases are misclassified, while 43 actual "1" cases are also misclassified. For *Case III*, the classification accuracy values are found to be 65.24, 65.10, 67.17, 68.69, and 67.44 for $k = 1, 3, 5, 7$, and 9, respectively. We choose $k = 3$, for which 202 cases that are actual "0" are misclassified. Among the class "1" cases, 51 cases are misclassified.

TABLE 6-2. THE PERFORMANCE RESULTS OF THE KNN CLASSIFICATION MODELS

Metrics	**Case I**	**Case II**	**Case III**
	Training Results 2013	**Training Results 2014**	**Test Results 2014**
Sensitivity	89.57	86.93	84.50
Specificity	96.42	92.93	48.99
PPV	95.11	91.08	57.92
NPV	92.24	89.54	79.18
CA	93.42	90.21	65.10
F1 Score	92.26	88.96	68.73

Decision Tree Classification: We use the *tree* function defined in the *tree* library in the R programming language for building the decision tree-based classification models in all three cases. The response variable *open_perc* is converted into a categorical type using the *as.factor* function for classification. The *predict* function in the *tree* library is used for predicting the classes of the response variable *open_perc* for the records in the test dataset. For *Case I* and *Case III,* the models are identical as they are trained on the 2013 data. While the model in *Case I* is tested on the 2103 data, the model in *Case III* uses the data of the year 2014 for testing. For all

three cases, we find *high_perc*, *low_perc*, and *close_perc* are the three predictor variables used by the algorithm to construct the decision trees. However, in *Case I*, the predictor used for splitting at the root node is *close_prec*, indicating that *close_perc* is the most important predictor for classification in the 2013 dataset. For the 2014 dataset, however, *high_perc* is found to be the most discriminating one as the model uses it for splitting at the root node. In *Case I*, the decision tree classifier misclassifies 8 cases out of a total of 419 cases of the class "0". On the other hand, 16 cases are wrongly classified out of 326 records of the class "1". In *Case II*, the model fails to correctly classify 17 cases out of 396 "0" class records. On the other hand, 25 cases are misclassified out of 329 instances of the class "1". In *Case III*, the model has a more difficult task at hand. We find that the model misclassifies 30 cases out of 396 class "0" records. On the other hand, 33 cases are misclassified out of 329 records of the class of "1". Table 6-3 presents the performance results of the decision tree classification models under three different cases. Figure 6-4 depicts the decision tree classifier for *Case III*.

TABLE 6-3. THE PERFORMANCE RESULTS OF THE DECISION TREE CLASSIFICATION MODELS

Metrics	**Case I**	**Case II**	**Case III**
	Training Results 2013	**Training Results 2014**	**Test Results 2014**
Sensitivity	95.09	92.40	89.97
Specificity	98.09	95.71	92.42
PPV	97.48	94.70	90.80
NPV	96.25	93.81	91.73
CA	96.78	94.21	91.31
F1 Score	96.07	93.54	90.38

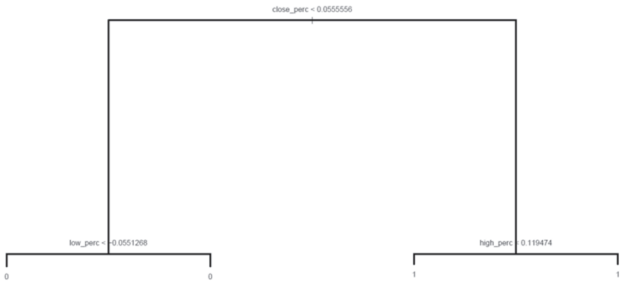

Figure 6-4. The decision tree model for classification (*Case III*)

Bagging Classification: We use the *bagging* function defined in the *ipred* library in the R programming language for building the bagging classification models. We set the value of the parameter *nbag* to 25 so that 25 decision trees are created randomly, and a simple majority voting mechanism is applied in constructing the classifier. In *Case I*, we find that the model fails to correctly classify 8 cases out of 419 that belong to the class "0". On the other hand, 16 instances out of 326 that belong to class "1" are also misclassified.

In *Case II*, the model fails to correctly classify 14 cases out of 396 that are of actual class "0". On the other hand, 15 cases out of 329 are misclassified, which belong to the class "1".

In *Case III*, 30 cases out of 396 actual "0" class cases are incorrectly classified by the model, while 33 cases out of a total of 329 of the class "1" are also misclassified. The performance results of the bagging classification model for all three cases are presented in Table 6-4.

Figure 6-5 depicts the classification accuracy of the model for *Case III*. In Figure 6-5, the *y*-axis represents the *actual* class labels, while the values along the *x*-axis show the probabilities of the predicted class for the records. The cases on the label "0" on the *y*-axis and having their probability values greater than 0.5 along the *x*-axis are the misclassified cases. Similarly, those cases lying on the label "1" along the *y*-axis and having their probability values less than 0.5 along the *x*-axis are also misclassified.

TABLE 6-4. THE PERFORMANCE RESULTS OF THE BAGGING CLASSIFICATION MODELS

Metrics	Case I	Case II	Case III
	Training Results 2013	Training Results 2014	Test Results 2014
Sensitivity	95.09	95.44	89.97
Specificity	98.09	96.46	92.42
PPV	97.48	95.73	90.78
NPV	96.25	96.22	91.73
CA	96.78	96.00	91.31
F1 Score	96.07	95.58	90.37

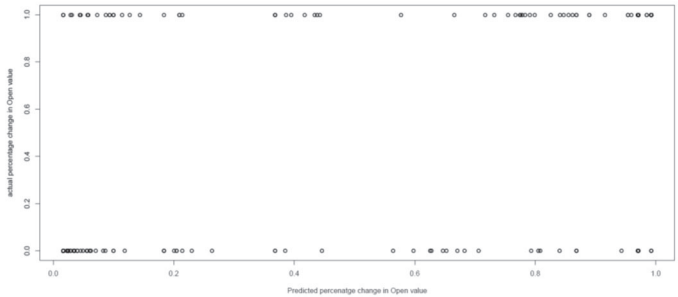

Figure 6-5. The bagging classification model – *actual* vs. *predicted* probabilities of *open_perc* (*Case III*)

TABLE 6-5. THE PERFORMANCE RESULTS OF THE BOOSTING CLASSIFICATION MODELS

Metrics	Case I	Case II	Case III
	Training Results 2013	**Training Results 2014**	**Test Results 2014**
Sensitivity	100	100	92.10
Specificity	100	100	93.43
PPV	100	100	92.10
NPV	100	100	93.43
CA	100	100	92.83
F1 Score	100	100	92.10

Boosting Classification: We use the *boosting* function defined in the *adabag* library in the R programming language for building the boosting models for classification. The response variable *open_perc* is transformed into a categorical type using the *as.factor* function to satisfy the requirement of a classification model. The *predict* function is used for predicting the label (i.e., the class) of the response variable for a record. For both *Case I* and *Case II*, the boosting classification models yield 100% accuracy on all the classification metrics, as presented in Table 6-5. These results are not surprising, as in both cases, the models are built and tested using the same dataset. Therefore, the learning of the models is very accurate using the ensemble of the weighted majority voting on a large number of random decision tree classifiers. However, the model faces a more difficult challenge in *Case* III. In this case, the ensemble model is built using the 2013 data and tested using the 2014 data.

In *Case III*, the model misclassifies 26 cases out of a total of 396 cases that belong to the class "0". On the other hand, among 329 instances that are actually of the type "1", 26 cases are incorrectly classified. Table 6-5 presents the performance results of the boosting classification models for all three cases. Figure 6-6 depicts the performance of the boosting classifier for *Case III*. In Figure 6-6, along the *y*-axis, the *actual* classes are plotted – there are two *actual* class levels, "0" and "1". The *x*-axis presents the *predicted probability* that a case will belong to the class "1". The data points lying on the left side of the threshold value of 0.5 along the *x*-axis and the level "1" along the *y*-axis are the misclassified cases. Similarly, the point on

the right side of the threshold value of 0.5 and the level "0" along the *y*-axis are misclassified cases. It is evident from Figure 6-6 that the boosting classifier yields quite a high level of accuracy in classification for *Case III*.

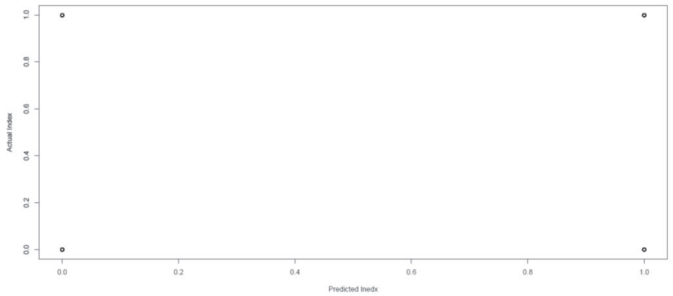

Figure 6-6. The boosting classification model – *actual* vs. *predicted* probabilities of *open_perc* (*Case III*)

Random Forest Classification: We use the *randomForest* function defined in the *randomForest* library in the R programming language for building the random forest-based classification models. In all three cases, the random forest algorithm creates 500 decision trees using three predictors at each node in the decision trees for carrying out the splitting task. In *Case I*, the model misclassifies 10 cases out of 419 cases that belong to the class "0". On the other hand, 18 cases are misclassified out of 326 cases of class "1". The *out-of-bag* (OOB) estimate of the error rate of the model, in this case, is 3.76%. In *Case II*, the model misclassified 23 cases out of 396 cases that belong to the actual class of "0". On the other hand, 23 cases out of 329 cases that belong to class "1" are also misclassified. The OOB estimate of the error rate of the classification model, in this case, is 6.34%. In *Case III*, the random forest classification model is identical to that in *Case I*. However, the model is tested on 2014 data, unlike the *Case I* model tested on the 2013 data. In *Case III*, we find that the model misclassifies 28 cases out of a total number of 396 instances of the class "0". On the other hand, 29 cases are misclassified out of 329 actual "1" cases. The performance results of the random forest classification model for all three cases are presented in Table 6-6.

TABLE 6-6. THE PERFORMANCE RESULTS OF THE RANDOM FOREST CLASSIFICATION MODELS

Metrics	Case I	Case II	Case III
	Training Results 2013	**Training Results 2014**	**Test Results 2014**
Sensitivity	94.48	93.01	91.19
Specificity	97.61	94.19	92.93
PPV	96.86	93.01	91.46
NPV	98.08	94.19	92.70
CA	96.24	93.66	92.14
F1 Score	95.66	93.01	91.32

ANN Classification: We use the *neuralnet* function defined in the *neuralnet* library in the R programming language to build ANN classification models. The parameter *linear.output* is set to FALSE. The response variable *open_perc* is converted into a categorical variable type using the function *as.factor* before the classification models are built. We find that only one node at the hidden layer is sufficient to build the model. Hence, we pass the value of the parameter *hidden* as 1 in the *neuralnet* function. To avoid any possible scenario in which the backpropagation algorithm fails to converge, we set the parameter *stepmax* to its maximum value of 10^6. In *Case I*, the ANN classification model misclassifies 10 cases out of a total of 419 instances of the class "1". On the other hand, 15 cases of class "1" are misclassified out of a total of 326 cases.

In *Case II*, the ANN classification model misclassifies 18 cases out of 396 cases of the class "1". On the other hand, 24 cases are misclassified out of 329 instances of the class "0".

TABLE 6-7. THE PERFORMANCE RESULTS OF THE ANN CLASSIFICATION MODELS

Metrics	Case I	Case II	Case III
	Training Results 2013	**Training Results 2014**	**Test Results 2014**
Sensitivity	95.40	92.71	99.70
Specificity	97.61	95.45	34.85
PPV	96.88	94.43	55.97
NPV	96.24	94.03	99.28
CA	96.64	94.21	64.28
F1 Score	96.13	93.56	71.69

In *Case III*, the model is built using the 2013 data, and it is identical to the model of *Case I*. However, since the model of *Case III* is tested on the 2014 data, the performance results of the model of *Case III* are different from those of *Case I*. We find that in *Case III*, the model misclassifies 258 cases out of 396 cases of the class "0". On the other hand, only 1 case out of 329 cases of the class "1" is misclassified. It is evident that the model fails, to a large extent, in classifying the class "0" cases, which results in a low value of *specificity*. The specificity in *Case III* is found to be only 34.85%, while for *Case I* and *Case II*, the specificity values are 97.61% and 95.45%, respectively. The results indicate that the ANN classification model does not generalize well in learning during the training phase, which leads to *model overfitting*. This overfitted model fails to correctly classify most of the "0" cases in the test data of 2014, resulting in a low value of specificity. Table 6-7 presents the performance of the ANN classification models for all three cases.

The ANN model for classification in *Case III* is presented in Figure 6-7, and its performance in the classification task is shown in Figure 6-8. Figure 6-7 plots along the *y*-axis the actual classes and along the *x*-axis the predicted classes. The points lying on the actual class label "0" along the *y*-axis while having their predicted class probabilities greater than 0.5 (i.e., those points on the "0" label lying on the right-hand side of the threshold value of 0.5 along the *x*-axis) represent the misclassified cases. Similarly, the points on the label "1" along the *y*-axis and having their probability values less than 0.5 (i.e., those points on the "1" label lying on the left-hand side of the threshold value of 0.5 along the *x*-axis) are also misclassified.

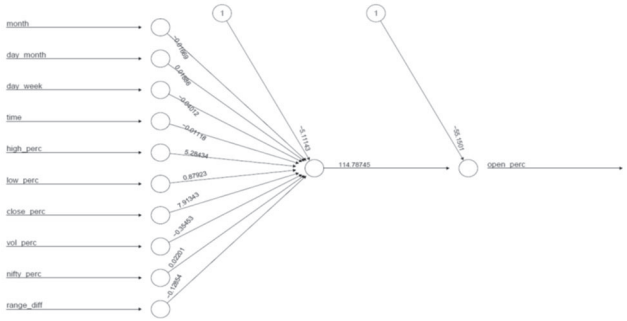

Figure 6-7. The architecture of the ANN classification model (*Case III*)

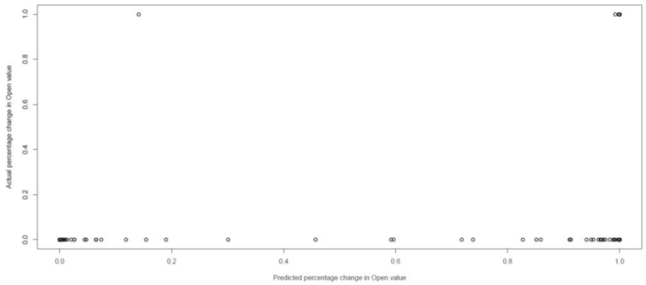

Figure 6-8. The ANN classification model – *actual* vs. *predicted* probabilities of *open_perc* (*Case III*)

TABLE 6-8. THE PERFORMANCE RESULTS OF THE SVM CLASSIFICATION MODELS

Metrics	Case I	Case II	Case III
	Training Results 2013	Training Results 2014	Test Results 2014
Sensitivity	94.46	94.67	97.46
Specificity	95.58	93.35	95.58
PPV	94.17	91.79	94.17
NPV	98.09	95.71	98.09
CA	96.38	93.93	96.37
F1 Score	94.31	93.21	95.79

SVM Classification: We use the *ksvm* function defined in the *kernlab* library in the *R* programming language for building the SVM-based classification models. The function *ksvm* is used with the parameter *kernel* set to *vanilladot*. It implies that a linear kernel is used for constructing the SVM classification models. For *Case I*, the model finds 120 support vectors. We find that the model misclassifies 19 cases out of 430 cases of the class "0". On the other hand, 8 cases are misclassified out of 315 cases of class "1". The training error for *Case I* is found to be 3.62%. For *Case II*, the model finds 156 support vectors to classify all the 725 records. Among 406 cases of the class "0", the model misclassifies 27 cases. On the other hand, 8 cases are misclassified out of 315 cases of class "1". The training error for *Case II* is found to be 6.07%. The SVM classification model finds 120 support vectors in *Case III*. The model misclassifies 19 cases out of 430 instances of the class "0". On the other hand, out of 315 cases of class "1", 8 cases are misclassified. Table 6-8 presents the results of the SVM classification models for all three cases.

***Regression Methods*:** We now present the performance results of the regression models.

Multivariate Regression: We mentioned in the section titled "Machine Learning Models" that the predictors that are finally included in the multivariate regression models for all three cases are: *low_perc* and *range_diff*.

In *Case I*, the regression model yields a value of 0.9919 for the adjusted R^2 and an F statistic value of $4.58*10^4$ with an associated p-value of $2.2*10^{-16}$. These values indicate that the regression model is able to establish a linear relationship between the response variable *open_perc* and the predictor variables *low_perc* and *range_diff*. The RMSE value yielded by the regression model for this case is found to be 0.0853, and the mean of the absolute values of the actual *open_perc* is 0.6402. The ratio of the RMSE to the mean of the absolute values of the actual *open_perc* is found to be 13.317. It is observed that 14 cases out of 745 instances exhibit a sign mismatch between the predicted and the actual values of *open_perc*. The correlation test produces a correlation coefficient value of 0.99 with the p-value of the t-statistic as $2.2*10^{-16}$. These values indicate a robust linear relationship between the predicted and the actual values of *open_perc*. The Breusch-Pagan test yields a test statistic value of 10.239 with a p-value of 0.005978. Hence, it is evident that the residuals are not *homoscedastic*. However, the Durbin-Watson test of autocorrelation produces a test statistic value of 3.023 with an associated p-value of 1. Hence, the null hypothesis

of no autocorrelation among the residuals has got the highest support. We conclude that the residuals do not exhibit any significant *autocorrelation*.

For *Case II*, the regression model yields an adjusted R^2 value of 0.9827 with an F-statistic value of $2.052*10^4$. The p-value of the F statistics is found to be less than $2.2*10^{-16}$, indicating a very highly significant F statistic and a good model fit. The RMSE value for *Case II* is found to be 0.1749, with the mean of the absolute values of actual *open_perc* as 0.9286. The ratio of the RMSE to the mean of the absolute values of the actual *open_perc* is 18.84. It is found that 39 out of 725 cases have a sign mismatch between the predicted and the actual *open_perc* values. The correlation test yields a correlation coefficient value of 0.99 with a t-statistic value of 202.74. The p-value of the t-statistic is $2.2*10^{-16}$. The very low p-value indicates a robust linear relationship between the predicted and the actual *open_perc* values. The Breusch-Pagan test yielded a test statistic value of 3.1877 with an associated p-value of 0.203. It is, therefore, evident that the residuals do not exhibit any significant *heteroscedasticity*. The Durbin-Watson test of autocorrelation produces a test statistic value of 2.9005. The p-value of the Durbin-Watson test is found to be 1. The maximum p-value indicates full support for the null hypothesis of no significant autocorrelation among the residuals. Hence, we conclude that the residuals in the regression model in *Case II* do not exhibit any significant autocorrelation.

The model in *Case III* is identical to the model in *Case I*. However, its performance results are different as it is tested on the data of the year 2014. The RMSE for *Case III* is found to be 0.1753. Since the mean of the absolute values of the actual *open_perc* is 0.9286, the ratio of the RMSE to the mean of the absolute values of the actual *open_perc* is found to be 18.88. We find that 39 out of 725 cases exhibit a sign mismatch between the predicted and the actual values of *open_perc*. The correlation test yields a correlation coefficient of 0.99 with the t-statistics value of 202.53 and the associated p-value of $2.2*10^{-16}$. These results indicate that the predicted and the actual values of *open_perc* exhibit a robust linear relationship. The Breusch-Pagan test yields a test statistic of 3.1877 with an associated p-value of 0.2031, indicating that the residuals are not heteroscedastic. The test statistic value produced by the Durbin-Watson test is found to be 2.9005 with an associated p-value of 1. The null hypothesis of no autocorrelation among the residuals finds full support. Hence, we conclude that the residuals do not exhibit any significant autocorrelation. Table 6-9 presents the results of the multivariate regression models for all three cases.

TABLE 6-9. THE PERFORMANCE RESULTS OF THE MULTIVARIATE REGRESSION MODELS

Metrics	Case I	Case II	Case III
	Training 2013	Training 2014	Test 2014
Correlation Coefficient	0.99	0.99	0.99
RMSE/Mean of Absolute Values of Actuals	13.32	18.84	18.88
Percentage of Mismatched Cases	18.67	5.38	5.24

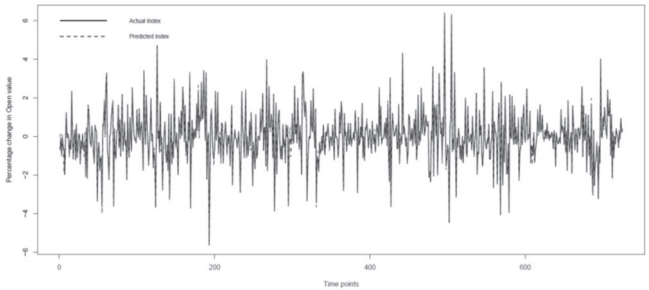

Figure 6-9. The multivariate regression model – *actual* and *predicted open_perc* (*Case III*)

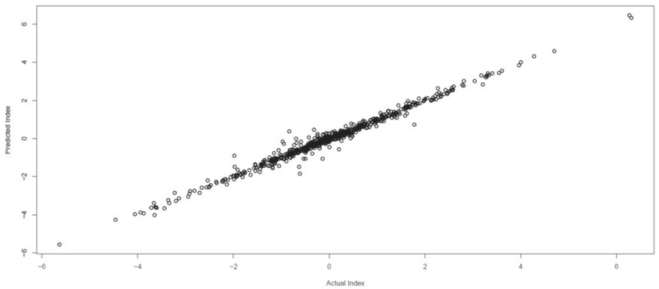

Figure 6-10. The multivariate regression model – the relationship between actual and predicted *open_perc* (*Case III*)

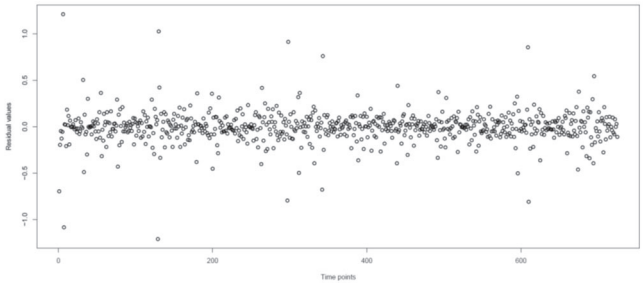

Figure 6-11. The multivariate regression model – the residual plot (*Case III*)

Figures 6-9 to 6-11 present some performance results of the multivariate regression model for *Case III*. Figure 6-9 shows that the predicted values very closely follow the pattern of the actual *open_perc* values. Figure 6-10 shows a robust linear relationship between the *predicted* and the *actual* values of *open_perc*. The residuals of the model are scattered and random and exhibit no significant autocorrelation, as depicted in Figure 6-11.

MARS: We use the *earth* function defined in the *earth* library in the R programming language for building MARS regression models.

In *Case I*, during the *forward pass* of the execution of the algorithm, seven terms are used in the model building. After the inclusion of the 8th term, the change in the value of R^2 is found to be only $5*10^{-5,}$ which is less than the threshold value of 0.001. As the forward pass completes, both the *generalized R-square* (GRSq) and the R^2 converge to a common value of 0.993. During the *backward pass*, the algorithm cannot prune any term, and all the 7 terms used in the forward pass are finally retained in the model. In *Case 1*, the model retains 3 predictors out of a total of 10 predictors. The selected predictors in the decreasing order of their importance in the model are *close_perc*, *high_perc*, and *low_perc*. The predictors, which are not used by the algorithm are: *month*, *day_month*, *day_week*, *time*, *vol_perc*, *nifty_perc*, and *range_diff*. After the execution of the algorithm, the values of some of the important metrics are as follows: (i) *generalized cross validation* (GCV): 0.0065, *residual sum of square* (RSS): 4.7006, GRSq: 0.9928, and R^2: 0.9930. The 7 terms that the MARS algorithm uses in *Case I* are found to be as follows: (i) the *intercept* of the model, (ii) *h*(-0.83682 – *high_perc*), (iii) *h*(*high_perc* – 0.83682), (iv) *h*(-0.692841 – *low_perc*), (v) *h*(*low_perc* – 0.692841), (vi) *h*(-2.11268 – *close_perc*), and (vii) *h*(*close_perc* – 2.11268). In *Case I*, the MARS regression model yields 9

cases out of a total of 745 that exhibit a mismatch in sign between the predicted and the actual values of *open_perc*. The RMSE value for this case is 0.0794, while the mean of the absolute values of the actual *open_perc* is 0.6402. Hence, the ratio of the RMSE to the mean of the absolute values of the actual *open_perc* is 2.4065. The correlation test yields a correlation coefficient value of 0.99. The value of the *t*-statistic is 325.41 with an associated *p*-value of $2.2*10^{-16}$. These statistics indicate a robust linear relationship between the predicted and the actual values of *open_perc* in *Case I*.

In *Case II*, the algorithm used 9 terms during its forward execution. The change in the R^2 value at the end of the 9th term is only 0.0002, while the threshold value is 0.001. After completing the forward pass, the values of GRSq and R^2 are found to be 0.985 and 0.986, respectively. During the backward pass of its execution, the algorithm prunes one term. Hence, the algorithm uses 8 terms in building the regression model. We also observe that the algorithm retains 4 predictors out 10 variables that were initially available. The 4 predictors included in the model in the decreasing order of their importance are *low_perc*, *close_perc*, *range_diff,* and *high_perc*.

At the end of the execution of the backward pass of the algorithm, the values of the following important metrics are found: GCV: 0.0262, RSS: 18.2512, GRSq: 0.9852, and R^2: 0.9858. The algorithm retains the following eight terms in the final regression model in *Case II*: (i) the *intercept* of the model, (ii) *h*(0.3675 – *high_perc*), (iii) *h*(*high_perc* – 0.3675), (iv) *h*(-2.6685 – *low_perc*), (v) *h*(*low_perc* – 2.6685), (vi) *h*(0.3996 – *close_perc*), (vii) *h*(-1.8 – *range_diff*), and (viii) h(*range_diff* - -1.8). In *Case II*, we find that 31 cases out of a total of 725 cases exhibit mismatched signs between the predicted and the actual values of *open_perc*. The RMSE for this case is 0.1587. Since the mean of the absolute values of the actual *open_perc* is 0.9286, the ratio of the RMSE to the mean is found to be 17.09. The correlation test yields a correlation coefficient value of 0.99. The value of the *t*-statistic is 223.87 with an associated *p*-value of $2.2*10^{-16}$. The high value of the correlation coefficient and the negligible support for the null hypothesis in the form of a very low *p*-value indicates a very robust linear relationship between the predicted and the actual values of *open_perc* in *Case II*.

TABLE 6-10. THE PERFORMANCE RESULTS OF THE MARS REGRESSION MODELS

Metrics	Case I	Case II	Case III
	Training 2013	Training 2014	Test 2014
Correlation Coefficient	0.99	0.99	0.99
RMSE/Mean of Absolute Values of Actuals	12.41	17.09	20.40
Percentage of Mismatched Cases	1.21	4.28	6.34

For *Case III*, the MARS model of regression is identical to that of *Case I*. The model is, however, tested on the 2014 data. We observe that 46 cases out of 725 instances depict a sign mismatch between the predicted and the actual *open_perc* values. The RMSE for this case is 0.1894, while the mean of the absolute values of the actual *open_perc* is 0.9286. The ratio of the RMSE to the mean value is found to be 20.40. The correlation test on the predicted and the actual values of *open_perc* yields the correlation coefficient value of 0.99. The value of the *t*-statistic is 187.13 with an associated *p*-value of $2.2*10^{-16}$. These statistics indicate a robust linear relationship between the actual and the predicted values of *open_perc*.

Table 6-10 presents the results of the MARS regression models for all three cases. Figures 6-12 to 6-14 present the performance results of the MARS regression model for *Case III*.

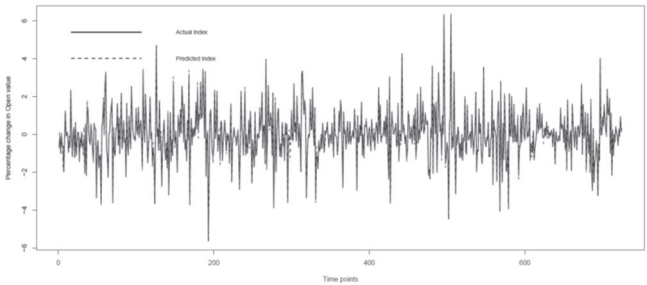

Figure 6-12. The MARS regression model – *actual* and *predicted open_perc* (*Case III*)

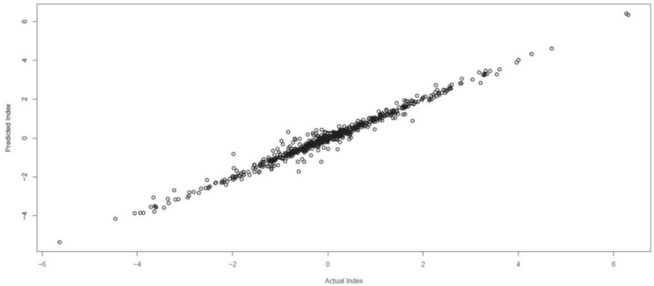

Figure 6-13. The MARS regression model – the relationship between actual and predicted *open_perc* (*Case III*)

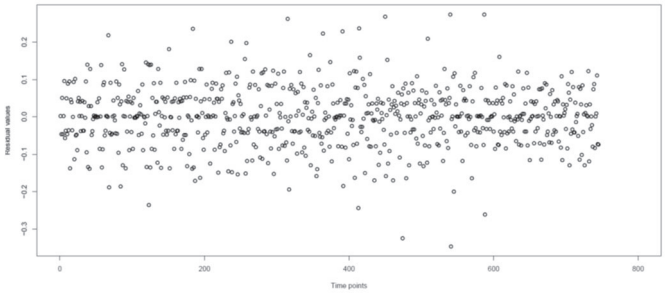

Figure 6-14. The MARS regression model – the residual plot (*Case III*)

Decision Tree Regression: We use the *tree* function defined in the *tree* library in the R programming language to build decision tree-based regression models.

For *Case I*, *close_perc* turns out to be the splitting variable at the root node. Other important variables that lead to the splitting at different nodes are *high_perc* and *low_perc*. The RMSE for this case is 0.2263, and the mean of the absolute values of the actual *open_perc* is found to be 0.6402. Out of 745 cases, 100 cases exhibit a sign mismatch between the predicted and the actual values of *open_perc*. The correlation coefficient between the *predicted* and the *actual open_perc* values is 0.97. The *t*-statistic for the correlation test yields a value of 111.35 with a *p*-value of $2.2*10^{-16}$. The nominal *p*-value implies no support for the null hypothesis and indicates a robust linear relationship between the *predicted* and the *actual open_perc* values.

For *Case II*, the variable *close_perc* is split at the root node of the tree. The other variables that are split at subsequent nodes are *high_perc* and

low_perc. This case yields an RMSE value of 0.3440, and the mean of the absolute values of the *actual open_perc* values is 0.9286. It is found that 126 cases out of 725 cases exhibit a sign mismatch between their *predicted* and *actual open_perc* values. The correlation coefficient between the *actual* and the *predicted* values of *open_perc* is found to be 0.96. The *t*-statistic value of the correlation test is 100.47, and its associated *p*-value is $2.2*10^{-16}$. The correlation test indicates that the *predicted* and the *actual open_perc* values are highly correlated.

The decision tree regression model for *Case III* is identical to that of *Case I*. The decision tree model is depicted in Figure 6-15. However, the performance of the model yields different results as it is tested on the 2014 data. The correlation coefficient between the predicted and the actual values of *open_perc* is found to be 0.96. The *t*-statistic value of the correlation test is 100.47, and its associated *p*-value is $2.2*10^{-16}$. However, as expected, the RMSE for this case is higher than in the previous two cases. The RMSE is found to be 0.5232. Since the mean of the absolute values of the actual *open_perc* is 0.9286, the ratio of the RMSE to the mean is 56.34. The value of the ratio of the RMSE to the mean is higher compared to the other two cases. It is observed that 126 cases out of 725 cases exhibit a mismatch in sign between the *predicted* and the *actual* values of *open_perc*. Figures 6-16 to 6-18 present the graphical representation of the performance of the model in *Case III*. While the behavior of the model is almost identical to that in the other two cases, from Figure 6-17, it is evident that there is more deviation between the patterns exhibited by the *actual* values and the *predicted* values of *open_perc*. This deviation leads to a significantly high RMSE for *Case III*.

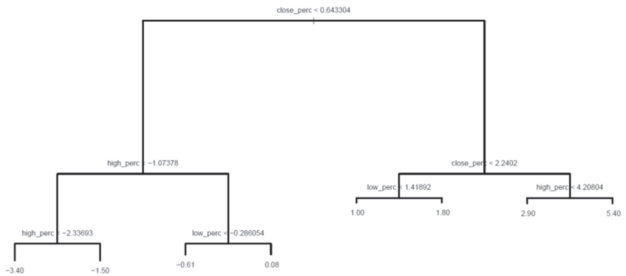

Figure 6-15. The decision tree regression model (*Case III*)

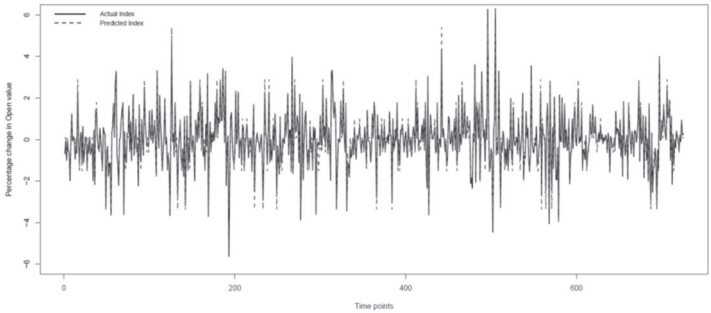

Figure 6-16. The decision tree regression – *actual* and *predicted open_perc* (*Case III*)

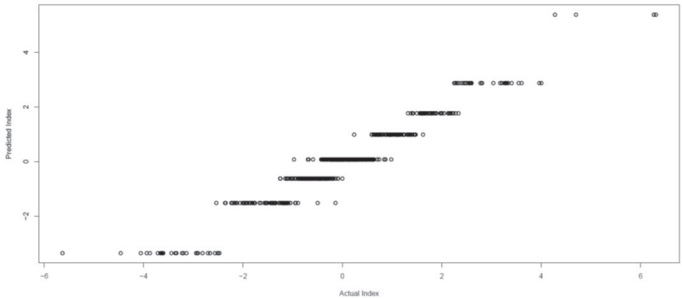

Figure 6-17. The decision tree regression – the relationship between actual and predicted *open_perc* (*Case III*)

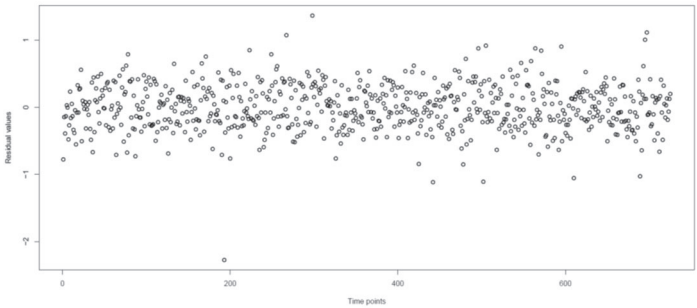

Figure 6-18. The decision tree regression – the residual plot (*Case III*)

TABLE 6-11. THE PERFORMANCE RESULTS OF THE DECISION TREE REGRESSION MODELS

Metrics	Case I	Case II	Case III
	Training 2013	Training 2014	Test 2014
Correlation Coefficient	0.97	0.97	0.96
RMSE/Mean of Absolute Values of Actuals	35.34	37.04	56.34
Percentage of Mismatched Cases	13.42	17.38	17.38

Table 6-11 presents the performance results of the decision tree regression models for all three cases.

***Bagging Regression*:** The *bagging* function defined in the *ipred* library of R programming language is used in building the bagging regression model. In *Case I*, the RMSE value is found to be 0.2141, with the mean of the absolute values of *open_perc* being 0.6402. Among 745 total cases, 22 cases exhibit a mismatch in their *predicted* and the corresponding *actual* values of *open_perc*. *Case II* yields an RMSE value of 0.2386, and the mean of the absolute values of the *actual open_perc* is 0.9286. A mismatch in the predicted and the actual values of *open_perc* is found in 37 cases out of 725 cases. The RMSE value for *Case III* is found to be 0.3242. The mean of the absolute values of the actual *open_perc* is 0.9286. Out of 725 cases, 67 cases show a mismatch in sign between the predicted and the actual values of *open_perc*. Figures 6-19 to 6-21 depict some graphical representations of the performance of the bagging regression model for *Case III*. Table 6-12 presents the performance results of the bagging regression models.

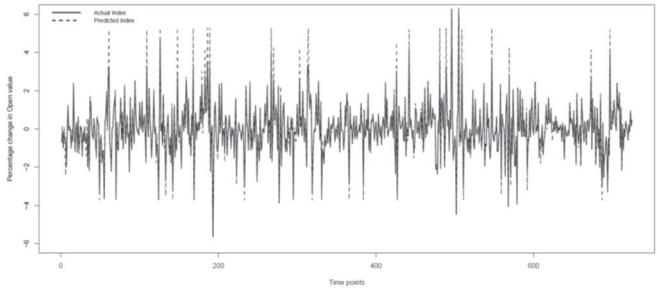

Figure 6-19. The bagging regression – *actual* and *predicted open_perc* (*Case III*)

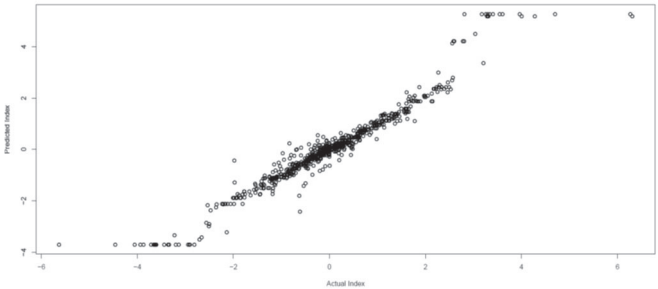

Figure 6-20. The bagging regression – the relationship between actual and predicted *open_perc* (*Case III*)

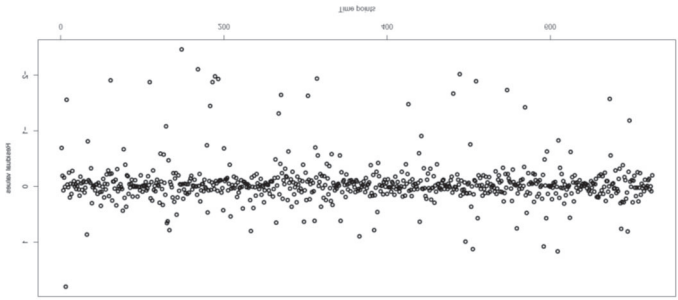

Figure 6-21. The bagging regression model – the residual plot (*Case III*)

TABLE 6-12. THE PERFORMANCE RESULTS OF THE BAGGING REGRESSION MODELS

Metrics	Case I	Case II	Case III
	Training 2013	Training 2014	Test 2014
Correlation Coefficient	0.97	0.98	0.97
RMSE/Mean of Absolute Values of Actuals	33.43	25.70	34.91
Percentage of Mismatched Cases	2.95	5.10	9.24

Boosting Regression: We use the *blackboost* function defined in the *mboost* library in the R programming language for building the boosting regression models. In *Case I*, we find that the model yields 6 cases out of 745 cases that exhibit mismatched signs between the predicted and the actual *open_perc* values. The RMSE for this case is found to be 0.1498, while the mean of the absolute values of the actual *open_perc* is 0.6402. For *Case II*, out of 725 total cases, 34 cases yield mismatched signs between the actual and their corresponding predicted values of *open_perc*. The RMSE for this case is 0.1596, and the mean of the absolute values of the actual *open_perc* is 0.9286. *Case III* yields an RMSE value of 0.3855, and the mean of the absolute values of the actual *open_perc* is 0.9286. A sign mismatch between the actual and the predicted values of *open_perc* is observed in 50 cases out of 725 cases. Table 6-13 presents the performance of the boosting regression models for all three cases. Figures 6-22 to 6-24 depict some graphical representations of the performance results of the boosting regression model in *Case III*.

TABLE 6-13. THE PERFORMANCE OF THE BOOSTING REGRESSION MODELS

Metrics	Case I	Case II	Case III
	Training Results 2013	Training Results 2014	Test Results 2014
Sensitivity	100	100	92.10
Specificity	100	100	93.43
PPV	100	100	92.10
NPV	100	100	93.43
CA	100	100	92.83
F1 Score	100	100	92.10

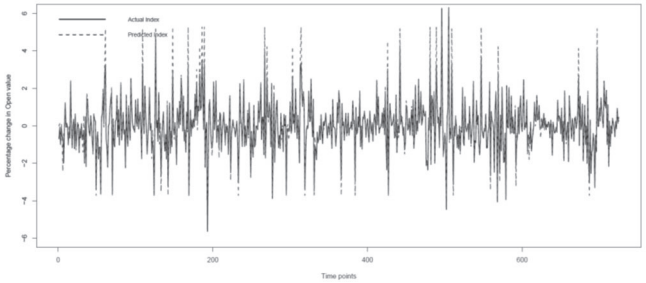

Figure 6-22. The boosting regression – *actual* and *predicted open_perc* (*Case III*)

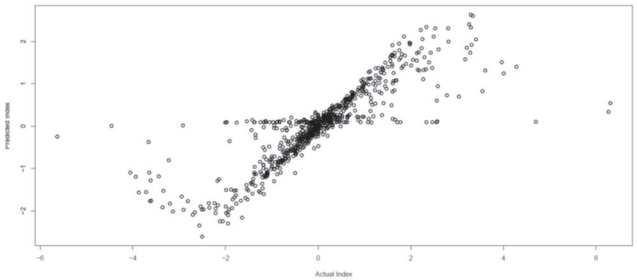

Figure 6-23. The boosting regression – the relationship between actual and predicted *open_perc* (*Case III*)

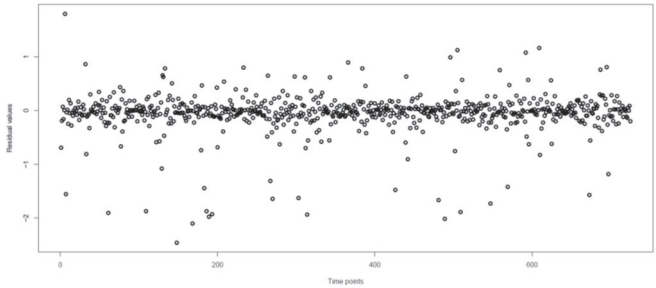

Figure 6-24. The boosting regression – the residual plot (*Case III*)

Random Forest Regression: We use the *randomForest* function defined in the *randomForest* library in the R programming language for building the random forest regression models. For all three cases, the algorithm tries three variables at each split of the associated decision tree. The number of regression decision trees constructed in each case is 500. The mean squared residual values are 0.0441, 0.0512, and 0.0441for *Case I*, *Case II,* and *Case III,* respectively. In *Case I*, the percentage of variance explained by the model is 95.13. None of the 745 cases exhibit any mismatch between the predicted and the actual values of *open_perc*. While the RMSE for this case is 0.1005, the mean of the absolute values of the actual *open_perc* is 0.9286051. For *Case II*, the model explains 97.11% of the variance, and 19 cases out of a total of 725 cases exhibit mismatched signs between the predicted and the actual values of *open_perc*. The RMSE for this case is 0.1005, while the mean of the absolute values of the actual *open_perc* is 0.9286. In *Case III*, the model explains 95.13% of the variance of the response variable. Out of 725 cases, 47 cases exhibit mismatched signs for the predicted and the actual *open_perc* values. The RMSE value for the model is 0.2973, and the mean of the absolute values of the actual *open_perc* values is 0.9286. Table 6-14 presents the results of the random forest regression model. Figures 6-25 to 6-27 depict some graphical representation of the performance results of the random forest regression model for *Case III*.

TABLE 6-14. THE PERFORMANCE OF THE RANDOM FOREST REGRESSION MODELS

Metrics	Case I	Case II	Case III
	Training 2013	Training 2014	Test 2014
Correlation Coefficient	0.99	0.99	0.97
RMSE/Mean of Absolute Values of Actuals	16.26	10.82	32.02
Percentage of Mismatched Cases	0.00	2.62	6.48

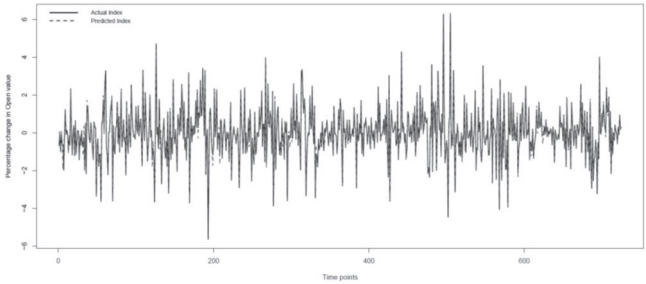

Figure 6-25. The random forest regression – actual and predicted *open_perc* (*Case III*)

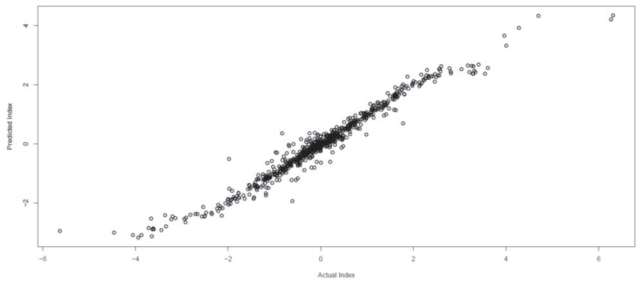

Figure 6-26. The random forest regression - the relationship between actual and predicted *open_perc* (*Case III*)

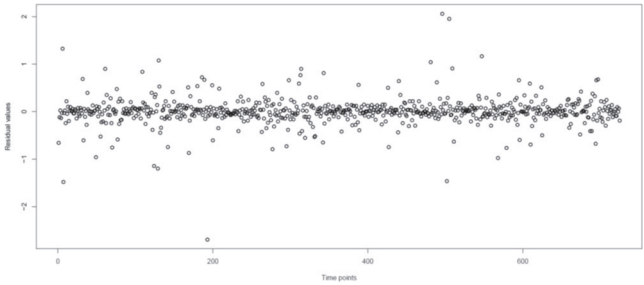

Figure 6-27. The random forest regression – the residual plot (*Case III*)

ANN Regression: We use the *neuralnet* function defined in the *neuralnet* library in the R programming language for building the ANN regression models. For *Case I*, 8 cases out of 745 records exhibit mismatched signs between the actual and the predicted *open_perc* values. The RMSE is found to be 0.0960, while the mean of the absolute values of the actual *open_perc* values is 0.6402. In *Case II*, out of 725 cases, 66 cases show a mismatch in sign between the actual and the predicted values of *open_perc*. The RMSE of the model for *Case II* is found to be 0.3389. In *Case III*, we find that 75 cases out of 725 instances exhibit mismatched signs in their actual and predicted *open_perc* values. The RMSE for this case is found to be 0.3278. The results for the ANN regression model are presented in Table 6-15.

TABLE 6-15. THE PERFORMANCE OF THE ANN REGRESSION MODELS

Metrics	**Case I**	**Case II**	**Case III**
	Training 2013	**Training 2014**	**Test 2014**
Correlation Coefficient	0.99	0.97	0.98
RMSE/Mean of Absolute Values of Actuals	16.96	36.49	35.29
Percentage of Mismatched Cases	1.07	15.31	9.10

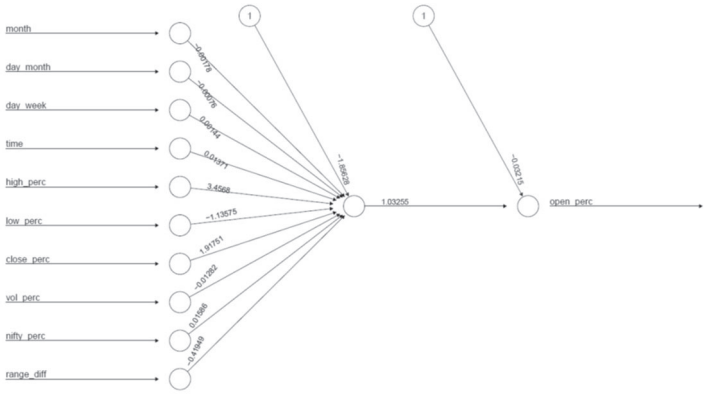

Figure 6-28. The architecture of the ANN regression model (*Case III*)

Figure 6-28 depicts the architecture of the ANN model for *Case III*. Figures 6-29 to 6-31 show several performance results of the ANN regression model for *Case III*. The plots in Figures 6-29 to 6-31 and the values of the numerical metrics like correlation coefficient, the ratio of RMSE and the mean of the absolute values of the actual *open_perc*, and the number of cases in which the predicted and the actual values have different signs as evident from Table 6-15, show that the ANN regression model is highly accurate in its test performance (i.e., under *Case III*).

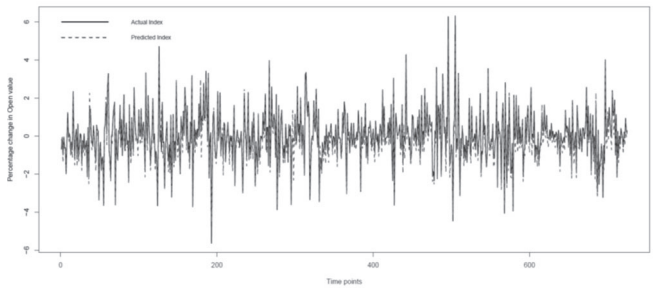

Figure 6-29. The ANN regression – *actual* and *predicted open_perc* (*Case III*)

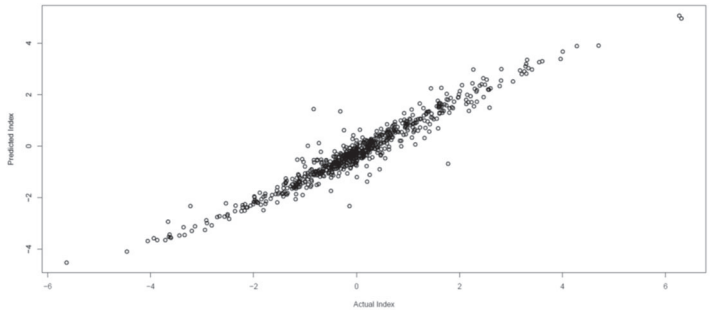

Figure 6-30. The ANN regression – the relationship between actual and predicted *open_perc* (*Case III*)

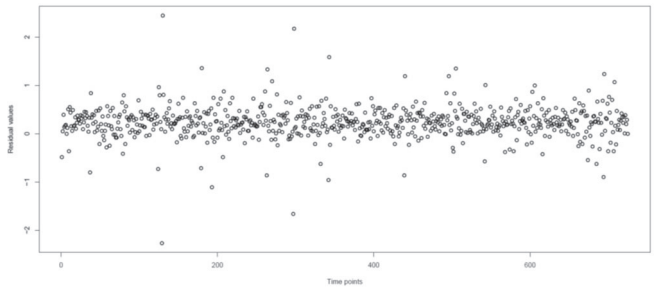

Figure 6-31. The ANN regression – the residual plot (*Case III*)

TABLE 6-16. THE PERFORMANCE OF THE SVM REGRESSION MODELS

Metrics	Case I	Case II	Case III
	Training 2013	Training 2014	Test 2014
Correlation Coefficient	0.93	0.98	0.83
RMSE/Mean of Absolute Values of Actuals	53.88	27.92	82.96
Percentage of Mismatched Cases	0.27	4.41	13.10

SVM Regression: In SVM regression, we use the *svm* function defined in the *e1071* library of the R programming language. We use the *eps-regression* type and the *radial kernel* for the SVM regression models for all

three cases. The values of the parameters *gamma* and *epsilon* are both found to be 0.1. The algorithm finds the number of support vectors as 248, 265, and 246 for *Case I*, *Case II,* and *Case III*. The RMSE values for the three cases are found to be 0.3450, 0.2593, and 0.7703, respectively. For *Case III*, the mean of the absolute values of the *open_perc* is 0.6402. We compute the ratio of the RMSE values to the mean of the absolute values of *open_perc* for all three cases to have an idea about the magnitude of the RMSE compared with the mean of the actual *open_perc* values. We also identify the cases exhibiting a mismatch in sign in the *actual* and the *predicted* values of *open_perc*. For these cases, the regression model fails to predict the direction of the movement of the *open_perc* values. For *Case I*, 2 cases out of 745 cases exhibit sign mismatch in the *actual* and the *predicted* values of *open_perc*. For Case *II*, out of 725 cases, 32 cases show a sign mismatch. In *Case III*, the model finds a more difficult challenge. For *Case III*, 95 cases out of 725 instances mismatch in sign in their actual and predicted values of *open_perc*. The SVM regression results are presented in Table 6-16. For all three cases, SVM regression models are found to produce accurate predictions.

Figures 6-32 to 6-34 depict the performance of the SVM model under *Case III*. The SVM model under *Case III* performs worse compared to *Case I* and *Case II*. For *Case I* and *Case II*, the performances of the models are measured on the in-sample data. For *Case III,* the results are of the performance on the out-of-sample data.

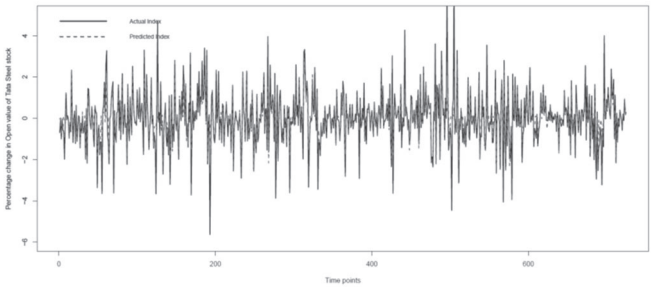

Figure 6-32. The SVM regression – *actual* and *predicted open_perc* (*Case III*)

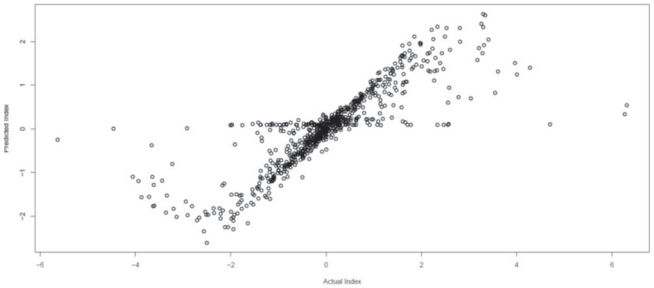

Figure 6-33. The SVM regression – the relationship between actual and predicted *open_perc* (*Case III*)

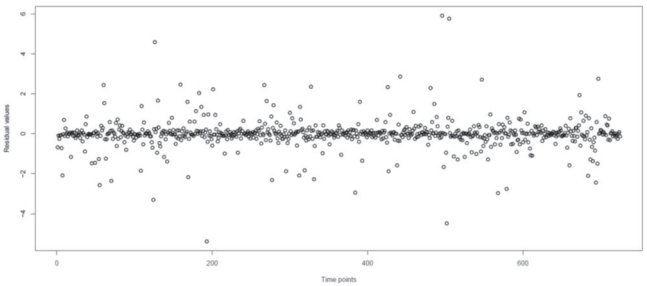

Figure 6-34. The SVM regression – the residual plot (*Case III*)

LSTM Regression: In the section titled "Deep Learning Models", we briefly discussed some essential characteristics of LSTM networks in deep learning. In the following, we present, in detail, the results related to the forecasting performance of the LSTM-based regression models used in the three cases. For all three cases, we use the following sequence of operations for building the LSTM models: (i) *reading the raw data*, (ii) *normalizing the predictors*, (iii) *converting the normalized data into a time series and then into a supervised learning problem*, (iv) *creating a deep learning model using the Tensorflow and the Keras frameworks*, (vi) *training and validating the model*, (v) *visualization of the training and validation performance*, and (vi) *evaluating the accuracy of the model on the test data*. For all three cases, the raw data consist of the following attributes: (i) *year*, (ii) *month*, (iii) *day*, (iv) *hour* (i.e., the time slot), (v) *open*, (vi) *high*, (vii) *low*, (viii) *close*, (ix) *volume*, and (x) the NIFTY index. Using Python programming, we combine the attributes (i) through (iv) into a single attribute so that the resultant dataset consists of seven attributes. We provide the details of the three cases in the following.

For *Case I*, we first plot the *open*, *high*, *low*, *close, volume,* and the NIFTY time series. In this case, there are 746 records in total. Figure 6-35 depicts the time series for each of the attributes for *Case I*.

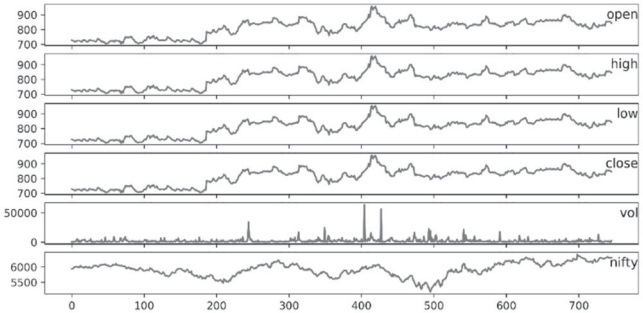

Figure 6-35. The stock data representation of Godrej Consumers for the year 2013 (*Case I*)

The architecture of the model is presented in Figure 6-36. The input layer receives six time series variable data. The output of the input layer is passed on to the LSTM layer that extracts 50 features from the input variables. The output of the LSTM layer is passed on to a *fully connected layer* (also known as a *dense layer*) consisting of 50 nodes. The dense layer is connected to the output layer containing a single node that produces the next predicted value.

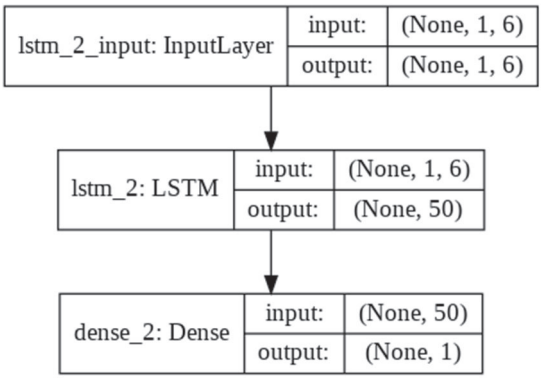

Figure 6-36. The architecture of the LSTM model (*Case I*, *Case II,* and *Case III*)

All the six attributes (leaving out the *time* attribute) are first normalized using the *MinMaxScala*r function defined in the *sklearn.processing* module in Python. Out of the 746 records, the first 500 records are used for training and the remaining 246 for validation. The *Sequential* function defined in Keras is used for building the LSTM, and the model is compiled using *mae* (mean absolute error) as the loss function and *Adam* as the optimizer. The behavior of the training and the validation loss values is studied for different values of epochs and batch sizes. With a batch size of 72 and an epoch value of 100, the training and validation losses converge to a very low value.

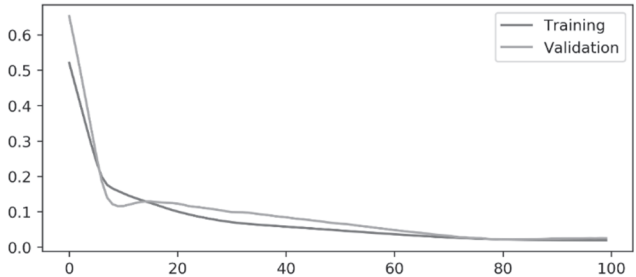

Figure 6-37. The training and the validation loss of the LSTM regression model for different epochs (*Case I*)
(*x*-axis: no. of epochs, *y*-axis: % loss in prediction)

Figure 6-37 presents the patterns of the training and the validation losses in *Case I*. After the final epoch, the RMSE value is 8.812, and the product-moment correlation coefficient between the *actual* and the *predicted open* values is 0.983. The training and validation loss values are 0.0194 and 0.0252, respectively.

Case II uses the stock price data for the year 2014. The dataset consists of 725 tuples. Figure 6-38 depicts the plots for the attributes – *open*, *high*, *low*, *close*, *volume,* and the NIFTY index. The raw values of these six attributes are normalized using the *MinMaxScalar* function. The first 500 records are used for the model construction, and the remaining 225 records are utilized in validating the model.

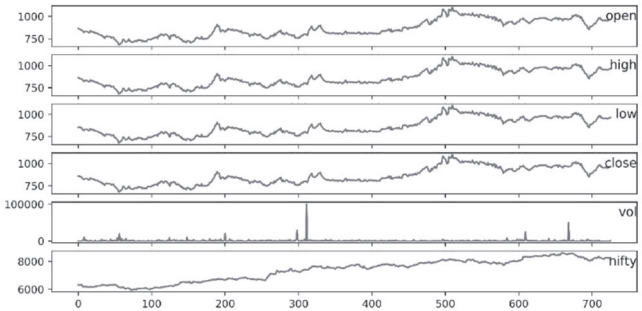

Figure 6-38. The stock data representation of Godrej Consumers for the year 2014 (*Case II*)

The validation loss converges with the training loss at an epoch value of 40. However, it starts increasing again with the increase in the epoch value. The validation loss finally converges with the training loss at an epoch value of 100 and a batch size of 72. The RMSE of the model is 15.002, with a correlation coefficient of 0.982 between the *actual* and the *predicted open* values. On completion of the last epoch, the training and the validation loss are found to be 0.0134 and 0.0301, respectively. Figure 6-39 depicts the variation of the training and the validation loss with different epoch values in *Case II*.

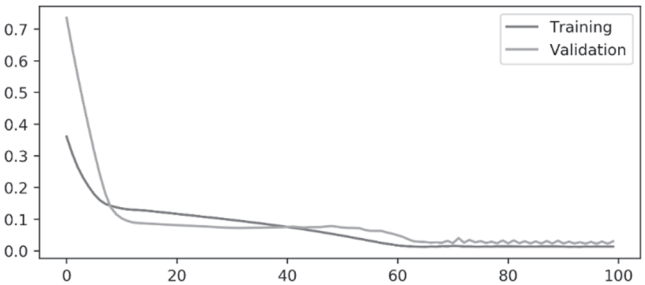

Figure 6-39. The training and the validation loss of the LSTM regression model for different epochs (*Case II*)
(*x*-axis: no. of epochs, *y*-axis: % loss in prediction)

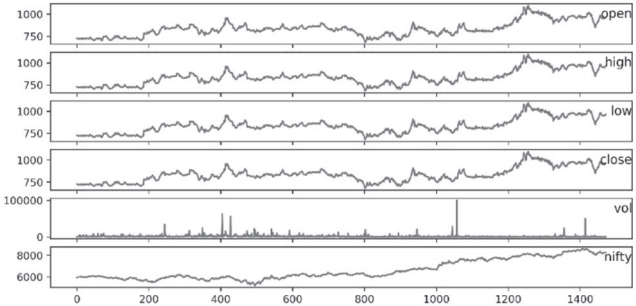

Figure 6-40. The stock data representation of Godrej Consumers for the period 2013 - 2014 (*Case III*)

For *Case III*, the LSTM regression model is built using the stock price records of the year 2013. The model is tested on the records of the year 2014. In this case, the raw dataset consists of 1471 records, of which 746 records (i.e., the records of 2013) are used in building the model, and the remaining 725 records (i.e., the records of 2014) are used for testing. Figure 6-40 depicts the *open*, *high*, *low*, *close*, *volume*, and the NIFTY time series for this case with 1471 records.

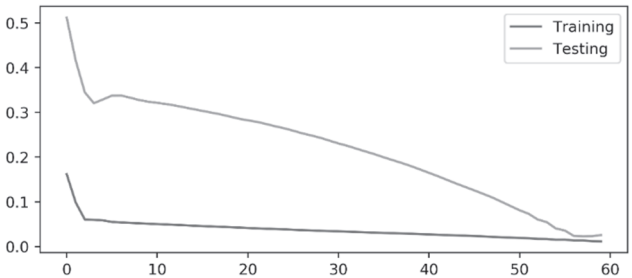

Figure 6-41. The training and the validation loss of the LSTM regression for different epochs (*Case III*)
(*x*-axis: no. of epochs, *y*-axis: % loss in prediction)

The training loss and the test loss converge after epoch 60 with a batch size of 72. The model RMSE and the correlation coefficient between the actual and the predicted values are 13.477 and 0.996, respectively. After completing the final epoch, the training and the test loss are 0.0116 and

0.0258, respectively. Figure 6-41 depicts the patterns exhibited by the training and the test losses for different values of the epoch.

Overall Performance: Finally, we summarize the performance of different predictive models that we have built, validated, and tested on the stock price data of Godrej Consumer Products from January 2013 to December 2014. Tables 6-17 to 6-19 present the performance of the classification models under *Case I*, *Case II*, and *Case III,* respectively. For each case and metric, the model that yields the best performance is marked with a bold font. We observe that for *Case I* and *Case II* and all the metrics, boosting performs the best among all the classification models. However, because *Case I* and *Case II* exhibit only the training accuracy, the performance of *Case III* should be considered the most critical as it depicts the test accuracy. From Table 6-19, we find that ANN performs the best on sensitivity and NPV while boosting outperforms all other models on specificity, PPV, and classification accuracy. However, SVM is found to have performed best on the *F1* score, which is usually considered the most critical metric for classification.

TABLE 6-17. SUMMARY OF THE PERFORMANCE OF THE CLASSIFICATION MODELS FOR *CASE I*

	LR	**KNN**	**DT**	**BAG**	**BOOST**	**RF**	**ANN**	**SVM**
Sensitivity	94.79	89.57	95.09	95.09	**100.00**	94.48	95.40	94.46
Specificity	97.61	96.42	98.09	98.09	**100.00**	97.61	97.61	95.58
PPV	96.87	95.11	97.48	97.48	**100.00**	96.86	96.88	94.17
NPV	96.01	92.24	96.25	96.25	**100.00**	98.08	96.24	98.09
CA	96.38	93.42	96.78	96.78	**100.00**	96.24	96.64	96.38
F1 Score	95.82	92.26	96.07	96.07	**100.00**	95.66	96.13	94.31

TABLE 6-18. SUMMARY OF THE PERFORMANCE OF THE CLASSIFICATION MODELS FOR *CASE II*

	LR	**KNN**	**DT**	**BAG**	**BOOST**	**RF**	**ANN**	**SVM**
Sensitivity	94.83	86.93	92.40	95.44	**100.00**	93.01	92.71	94.67
Specificity	95.96	92.93	95.71	96.46	**100.00**	94.19	95.45	93.35
PPV	95.12	91.08	94.70	95.73	**100.00**	93.01	94.43	91.79
NPV	95.72	89.54	93.81	96.22	**100.00**	94.19	94.03	95.71
CA	95.45	90.21	94.21	96.00	**100.00**	93.66	94.21	93.93
F1 Score	94.97	88.96	93.54	95.58	**100.00**	93.01	93.56	93.21

TABLE 6-19. SUMMARY OF THE PERFORMANCE OF THE CLASSIFICATION MODELS FOR *CASE III*

	LR	**KNN**	**DT**	**BAG**	**BOOST**	**RF**	**ANN**	**SVM**
Sensitivity	92.10	84.50	89.97	89.97	92.10	91.19	**99.70**	93.81
Specificity	89.39	48.99	92.42	92.42	**93.43**	92.93	34.85	90.19
PPV	87.83	57.92	90.80	90.78	**92.10**	91.46	55.97	87.54
NPV	93.16	79.18	91.73	91.73	93.43	92.70	**99.28**	95.02
CA	90.62	65.10	91.31	91.31	**92.83**	92.14	64.28	91.72
F1 Score	89.91	68.73	90.38	90.37	92.10	91.32	71.69	**95.79**

Tables 6-20 to 6-22 present the performance of the regression models, including the LSTM-based deep learning model. Since the LSTM model has outperformed the machine learning models on all metrics and for all three cases, we have also identified the best-performing machine learning model on each metric. In *Case I*, multivariate regression, MARS, boosting, random forest, and ANN all yield the highest correlation coefficient value of 0.99. However, the correlation coefficient is found to be 1.00 for LSTM. For the ratio of the RMSE to the mean of the absolute values of the *open_perc* values, MARS yields the lowest value of 12.41 among the machine learning models, while the corresponding value for LSTM is 7.94. Both random forest and LSTM produce no sign mismatch between the predicted and the actual values of *open_perc*. In *Case II*, the highest value of the correlation coefficient is achieved by multivariate regression, MARS, boosting, and random forest. LSTM outperforms all machine learning models on this metric by attaining a value of 1.00. The RMSE to the mean ratio value of 10.82 is the least for random forest among the machine learning models. However, the corresponding value yielded by LSTM is 4.04. The random forest model produces only 2.62 percent of cases mismatching in the sign between the actual and the predicted open_perc values. However, the LSTM model yields no sign mismatch. For *Case III*, while LSTM exhibits the best performance on all metrics, multivariate regression and MARS yield the same value for the correlation coefficient. For the metric RMSE to the mean ratio and the percentage of the mismatched cases, multivariate regression and MARS produced the best results among the machine learning models.

In Tables 6-17 to 6-22, the following abbreviations are used in the column names: LR – Logistic Regression, KNN – K-Nearest Neighbor, DT-Decision Tree, BAG – Bagging, BOOST – Boosting, RF – Random Forest, ANN – Artificial Neural Networks, SVM – Support Vector Machines, LSTM – Long and Short Term Memory.

TABLE 6-20. SUMMARY OF THE PERFORMANCE OF THE REGRESSION MODELS FOR *CASE I*

	MV	**MARS**	**DT**	**BAG**	**BOOST**	**RF**	**ANN**	**SVM**	**LSTM**
Correlation	**0.99**	**0.99**	0.97	0.97	**0.99**	**0.99**	**0.99**	0.93	**1.00**
RMSE/Mean	13.32	**12.41**	35.34	33.43	23.40	16.26	14.99	53.88	**7.94**
Mismatched Cases	18.67	1.21	13.42	2.95	0.81	**0.00**	1.07	0.27	**0.00**

TABLE 6-21. SUMMARY OF THE PERFORMANCE OF THE REGRESSION MODELS FOR *CASE II*

	MV	**MARS**	**DT**	**BAG**	**BOOST**	**RF**	**ANN**	**SVM**	**LSTM**
Correlation	**0.99**	**0.99**	0.97	0.98	**0.99**	**0.99**	0.97	0.98	**1.00**
RMSE/Mean	18.84	17.09	37.04	25.70	17.19	**10.82**	36.49	27.92	**4.04**
Mismatched Cases	51.31	4.28	17.38	5.10	4.69	**2.62**	15.31	4.41	**0.00**

TABLE 6-22. SUMMARY OF THE PERFORMANCE OF THE REGRESSION MODELS FOR *CASE III*

	MV	**MARS**	**DT**	**BAG**	**BOOST**	**RF**	**ANN**	**SVM**	**LSTM**
Correlation	**0.99**	**0.99**	0.96	0.97	0.97	0.97	0.98	0.83	**0.99**
RMSE/Mean	**18.88**	20.40	56.34	34.91	41.51	32.02	35.29	82.96	**2.36**
Mismatched Cases	51.31	**6.34**	17.38	9.24	6.90	6.48	10.34	13.10	**2.40**

Conclusion

This chapter has proposed a robust forecasting framework for predicting the future stock price and the price movement. The predictive model consists of eight classification and eight regression models based on several machine learning approaches. Additionally, the framework also includes a deep learning model of regression using an LSTM network. All these models work on a short-term time horizon. We construct the models, train, validate, and finally test them using the historical stock prices of a company – Godrej Consumer Products Ltd. The stock data are acquired from January 2013 to December 2014 at five minutes intervals from historical stock price records in the National Stock Exchange (NSE), India. The Metastock tool is used for extracting the data from the source. The raw data are pre-processed, appropriate transformation methods, e.g., normalization, standardization, NA removal, are applied, and several derived predictor variables are created using the rich features of the stock data. While several newly derived predictors are used in building the models, we use the percentage change in the *open* values of the stock, called *open_perc*, as the response variables.

The high-frequency stock price records at five minutes intervals are aggregated into three slots on a given day. The predictive models are used to forecast the value of *open_perc* in the next slot, given the current slot's stock price data. While the classification-based models are used to predict the movement pattern of *open_perc* values, the objective of the regression models is to predict the value of the *open_perc* accurately. In addition to exploiting the machine learning algorithms for building the classification and regression models, we also leverage the powerful capabilities of the Tensorflow and Keras frameworks in building a robust deep learning-based regression model using an LSTM network. We use the R programming language to construct the machine learning models, while the Python language is used for building the LSTM model. The models are trained, validated, and tested on the stock data, and extensive results are critically analyzed. The results elicit a very interesting observation. While no single machine learning model performs the best on all the classification and regression metrics, the LSTM model outperforms all the regression models on every metric considered. In a previously published paper, we demonstrated the efficacy and accuracy of a CNN-based deep learning regression model in time series forecasting (Mehtab & Sen, 2020a). It is a very well-known fact that deep learning models have a much higher capability to extract and learn the features from time series data than their machine learning counterparts. However, to exploit the power of deep

learning models, the data volume should be substantial. As a future scope of work, we would explore *generalized adversarial networks* (GAN) in forecasting future stock prices. We believe that an integrated approach to building deep learning models that combines the power of LSTM, CNN, and GAN will be a fascinating area of work in this direction.

References

Adebiyi, A. A., Adewumi, A. O. and Ayo, C. K. (2014), "Stock price prediction using the ARIMA model", *Proceedings of the International Conference on Computer Modelling and Simulation (UKSim-AMSS)*, March 26-28, Cambridge, UK, pp. 105 -111. DOI:10.1109/UKSim.2014.67.

Basalto, N., Bellotti, R., De Carlo, F., Facchi, P. and Pascazio, S. (2005) "Clustering stock market companies via chaotic map synchronization", *Physica A: Statistical Mechanics and its Applications*, Vol. 345, No. 1-2, pp. 196-206. DOI:10.1016/j.physa.2004.07.034.

Basu, S. (1983) "The relationship between earnings yield, market value and return for NYSE common stocks: Further evidence", *Journal of Financial Economics*, Vol.12, No.1, pp. 129-156. DOI: 10.1016/0304-405X(83)90031-4.

Bentes, S. R., Menezes, R. and Mendes, D. A. (2008) "Long memory and volatility clustering: Is the empirical evidence consistent across stock markets?", *Physica A: Statistical Mechanics and its Applications*, Vol. 387, No. 15, pp. 3826-3830. DOI:10.1016/j.physa.2008.01.046.

Chen, Y., Dong, X. and Zhao, Y. (2005) "Stock index modeling using EDA based local linear wavelet neural network", *Proceedings of the IEEE International Conference on Neural Networks and Brain*, October 13-15, Beijing, China, pp. 1646–1650. DOI: 10.1109/ICNNB.2005.1614946.

Chen, A.-S., Leung, M. T. and Daouk, H. (2003) "Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index", *Operations Research in Emerging Economics*, Vol. 30, No. 6, pp. 901– 923. DOI:10.1016/S0305-0548(02)00037-0.

Chui, A. and Wei, K. (1998) "Book-to-market firm size, and the turn of the year effect: Evidence from Pacific basin emerging markets", *Pacific Basin Finance Journal*, Vol. 6, No. 3-4, pp. 275-293. DOI:10.1016/S0927-538X(98)00013-4.

de Faria, E., Albuquerque, M. P., Gonzalez, J., Cavalcante, J. and Albuquerque, M. P. (2009) "Predicting the Brazilian stock market

through neural networks and adaptive exponential smoothing methods", *Expert Systems with Applications*, Vol. 36, No. 10, pp. 12506-12509. DOI. 10.1016/j.eswa.2009.04.032.

Dutta, G., Jha, P., Laha, A. & Mohan, N. (2006) "Artificial neural network models for forecasting stock price index in the Bombay Stock Exchange", *Journal of Emerging Market Finance*, Vol. 5, No. 3, pp. 283-295. DOI: 10.1177/097265270600500305.

Fama, E. F. and French, K.R. (1995) "Size and book-to-market factors in earning and returns", *Journal of Finance*, Vol. 50, No. 1, pp. 131-155. DOI: 10.1111/j.1540-6261.1995.tb05169.x.

Fu, T-C, Chung, F-L., Luk, R. and Ng, C-M, (2008) "Representing financial time series based on data point importance", *Engineering Applications of Artificial Intelligence*, Vol. 21, No. 2, pp. 277-300. DOI: 10.1016/j.engappai.2007.04.009.

Geron, A. (2019). *Hands-on Machine Learning with Scikit-Learn Keras & Tensorflow*. O'Reilly Publications, USA. ISBN: 9781492032649.

Hammad, A. A. A., Ali, S. M. A. and Hall, E. L. (2007) "Forecasting the Jordanian stock prices using artificial neural network", *Intelligent Engineering Systems through Artificial Neural Networks*, Vol. 17. DOI: 10.1115/1.802655.paper42.

Hanias, M., Curtis, P. and Thalassinos, J. (2012) "Time series prediction with neural networks for the Athens Stock Exchange indicator", *European Research Studies Journal*, Vol. 15, No. 2, pp. 23-32. DOI: 10.35808/ersj/351.

Hornik, K., Stinchcombe, M. and White, H. (1989) "Multilayer feedforward networks are universal approximators", *Neural Networks*, Vol. 2, No. 5, pp. 359-366. DOI: 10.1016/0893-6080(89)90020-8.

Hutchinson, J. M., Lo, A. W. and Poggio, T. (1994) "A nonparametric approach to pricing and hedging derivative securities via learning networks", *The Journal of Finance*, Vol. 49, No. 3, pp. 851-889. DOI: 10.1111/j.1540-6261.1994.tb00081.x.

Jaffe, J., Keim, D. B. and Westerfield, R. (1989) "Earnings, yields, market values and stock returns", *The Journal of Finance*, Vol. 44, No. 1, pp. 135-148. DOI: 10.1111/j.1540-6261.1989.tb02408.x.

Jarrett, J.E. and Kyper E. (2011) "ARIMA modeling with intervention to forecast and analyze Chinese stock prices", *International Journal of Engineering Business Management*, Vol. 3, No. 3, pp. 53-58. DOI:10.5772/50938.

Jaruszewicz, M. and Mandziuk, J. (2004) "One day prediction of Nikkei index considering information from other stock markets", In: Rutkowski, L., Siekmann, J. H., Tadeusiewicz, R. and Zaseh, L. A.

(eds.), *Artificial Intelligence and Soft Computing, ICAISC'04*, *Lecture Notes in Computer Science*, Vol. 3070, pp. 1130-1135, Springer, Berlin, Heidelberg, Germany. DOI: 10.1007/978-3-540-24844-6_177.

Kimoto, T., Asakawa, K., Yoda, M. and Takeoka, M. (1990) "Stock market prediction system with modular neural networks", *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, June 17-21, San Diego, CA, USA, pp. 1-6. DOI: 10.1109/IJCNN.1990.137535.

Leigh, W., Hightower, R. and Modani, N. (2005) "Forecasting the New York Stock Exchange composite index with past price and interest rate on condition of volume spike", *Expert Systems with Applications*, Vol. 28, No. 1, pp. 1-8. DOI: 10.1016/j.eswa.2004.08.001.

Liao, S-H., Ho, H-H. and Lin, H-W. (2008) "Mining stock category association and cluster on Taiwan stock market", *Expert System with Applications*, Vol. 35, No. 1-2, pp.19-29. DOI: 10.1016/j.eswa.2007.06.001.

Mehtab, S. and Sen, J. (2022) "Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models", In: Sahoo, J. P., Tripathy, A. K., Mohanty, M., Li, K, C., Nayak, A. K. (eds), *Advances in Distributed Computing and Machine Learning*. Lecture Notes in Networks and Systems, Vol 202, pp. 405-423, Springer, Singapore. DOI: 10.1007/978-981-16-4807-6_39.

Mehtab, S. and Sen, J. (2021a) "A time series analysis-based stock price prediction using machine learning and deep learning models", *International Journal of Business Forecasting and Marketing Intelligence (IJBFMI)*, Vol 6, No 4, pp. 272 – 335, Inderscience Publishers. DOI: 10.1504/IJBFMI.2020.115691.

Mehtab, S., Sen, J. and Dutta, A. (2021b) "Stock price prediction using machine learning and LSTM-based deep learning models", In: Thampi, S. M., Piramuthu, S., Li, K. C., Beretti, S., Wozniak, M., and Singh, D. (eds.), *Machine Learning and Metaheuristics Algorithms and Applications*. *SoMMA 2020*. Communications in Computer and Information Science, Vol 1366, pp. 88-106, Springer, Singapore. DOI: 10.1007/978-981-16-0419-5_8.

Mehtab, S. and Sen, J. (2020a) "Stock price prediction using convolutional neural networks on a multivariate time series", *Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence* (*NCMLAI'20*), February 1, 2020, New Delhi, India. DOI: 10.36227/techrxiv.15088734.v1.

Mehtab, S. and Sen, J. (2020b) "Stock price prediction using CNN and LSTM-based deep learning models", *Proceedings of the IEEE*

International Conference on Decision Aid Sciences and Applications (DASA'20), November 8-9, 2020, Sakheer, Bahrain, Bahrain, pp. 447-453. DOI: 10.1109/DASA51403.2020.9317207.

Mehtab, S., Sen, J. and Dasgupta, S. (2020c) "Robust analysis of stock price time series using CNN and LSTM-based deep learning models", *Proceedings of the 4th IEEE International Conference on Electronics, Communication and Aerospace Technology (ICECA'20)*, November 5-7, 2020, Coimbatore, India, pp. 1481-1486. DOI: 10.1109/ICECA49313.2020.9297652.

Mehtab, S. and Sen, J. (2020d) *Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models*, Technical Report, No: NSHM_KOL_2020_SCA_DS_1. DOI: 10.13140/RG.2.2.14022.22085/2.

Mehtab, S. and Sen, J. (2019) "A robust predictive model for stock price prediction using deep learning and natural language processing", *Proceedings of the 7th International Conference on Business Analytics and Intelligence* (*BAICONF'19*), December 5-7, 2019, Bangalore, India. DOI:10.2139/ssrn.3502624.

Metastock Website: http://www.metastock.com

Mishra, S. (2016) "The quantile regression approach to analysis of dynamic interaction between exchange rate and stock returns in emerging markets: Case of BRIC nations", *IUP Journal of Financial Risk Management*, Vol. 13, No. 1, pp.7-27.

Mondal, P., Shit, L. and Goswami, S. (2014) "Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices", *International Journal of Computer Science, Engineering and Applications*, Vol. 4, No. 3, pp. 13-29. DOI: 10.5121/ijcsea.2014.4202.

Moshiri, S. and Cameron, N. (2010) "Neural network versus econometric models in forecasting inflation", *Journal of Forecasting*, Vol. 19, No. 3, pp. 201-217. DOI: 10.1002/(SICI)1099-131X(200004)19:3< 201::AID-FOR753>3.0.CO;2-4.

Mostafa, M. (2010) "Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait", *Expert Systems with Application*, Vol. 37, No. 9, pp. 6302-6309. DOI: 10.1016/j.eswa.2010.02.091.

Ning, B., Wu, J., Peng, H. and Zhao, J. (2009) "Using chaotic neural network to forecast stock index", In: Yu, W., He, H., Zhang, N. (eds.) *Advances in Neural Networks, Lecture Notes in Computer Science*, Vol 5551, pp. 870-876. DOI: 10.1007/978-3-642-01507-6_98.

Phua, P. K. H., Ming, D., and Lin, W. (2001) "Neural network with genetically evolved algorithms for stocks prediction", *Asia-Pacific Journal of Operational Research*, Vol. 18, No. 1, pp. 103-107.

Rosenberg, B., Reid, K. and Lanstein, R. (1985) "Persuasive evidence of market inefficiency", *Journal of Portfolio Management*, Vol. 11, No. 3, pp. 9-16. DOI: 10.3905/jpm.1985.409007.

Sen, J. (2018) "Stock price prediction using machine learning and deep learning frameworks", *Proceedings of the 6th International Conference on Business Analytics and Intelligence (ICBAI'18)*, Bangalore, India, December 20-22, 2018.

Sen, J. and Datta Chaudhuri, T. (2018) "Understanding the sectors of Indian economy for portfolio choice", *International Journal of Business Forecasting and Marketing Intelligence*, Vol. 4, No. 2, pp. 178-222. DOI: 10.1504/IJBFMI.2018.090914.

Sen, J. and Datta Chaudhuri, T. (2017a) "A time series analysis-based forecasting framework for the Indian healthcare sector", *Journal of Insurance and Financial Management*, Vol. 3, No. 1, pp. 66-94.

Sen, J. and Datta Chaudhuri, T. (2017b) "A predictive analysis of the Indian FMCG sector using time series decomposition-based approach", *Journal of Economics Library*, Vol. 4, No. 2, pp. 206-226. DOI: 10.1453/jel.v4i2.1282.

Sen, J. and Datta Chaudhuri, T. (2017c) "A time series analysis-based forecasting approach for the Indian realty sector", *International Journal of Applied Economic Studies*, Vol. 5, No. 4, pp. 8 – 27.

Sen, J. and Datta Chaudhuri, T. (2017d) "A robust predictive model for stock price forecasting", *Proceedings of the 5th International Conference on Business Analytics and Intelligence*, Bangalore, India, December 11-13, 2017.

Sen, J. and Datta Chaudhuri, T. (2016a) "An investigation of the structural characteristics of the Indian IT sector and the capital goods sector – An application of the R programming language in time series decomposition and forecasting", *Journal of Insurance and Financial Management*, Vol. 1, No. 4, pp. 68-132.

Sen, J. and Datta Chaudhuri, T. (2016b) "An alternative framework for time series decomposition and forecasting and its relevance for portfolio choice – A comparative study of the Indian consumer durable and small-cap sector", *Journal of Economic Library*, Vol. 3, No. 2, pp. 303-326. DOI: 10.1453/jel.v3i2.787.

Sen, J. and Datta Chaudhuri, T. (2016c) "Decomposition of time series data of stock markets and its implications for prediction: An application for the Indian auto sector", *Proceedings of the 2nd National Conference*

on Advances in Business Research and Management Practices (*ABRMP'16*), January 8 – 9, Kolkata, India, pp. 15-28. DOI: 10.13140/RG.2.1.3232.0241.

Sen, J. and Datta Chaudhuri, T. (2015) "A framework for predictive analysis of stock market indices – A study of the Indian auto sector", *CBS Journal of Management Practices*, Vol. 2, No. 1, pp. 1-20. DOI: 10.13140/RG.2.1.2178.3448.

Sen, J. and Mehtab, S. (2021a) "Accurate stock price forecasting using robust and optimized deep learning models", *Proceedings of the IEEE International Conference on Intelligent Technologies (CONIT'21)*, June 25-27, Hubballi, India. DOI: 10.1109/CONIT51480.2021.9498565.

Sen, J. and Mehtab, S. (2021b) "Design and analysis of robust deep learning models for stock price prediction", In: Sen, J. (ed.) *Machine Learning – Algorithms, Models and Applications*, IntechOpen, London UK, pp. 15 – 46. DOI: 10.5772/intechopen.99982.

Senol, D. and Ozturan, M. (2008) "Stock price direction prediction using artificial neural network approach: The case of Turkey", *Journal of Artificial Intelligence*, Vol. 1, No. 2, pp. 70-77. DOI: 10.3923/jai.2008.70.77.

Shen, J., Fan, H. and Chang, S. (2007) "Stock index prediction based on adaptive training and pruning algorithm", In: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (eds.), *Advances in Neural Networks, Lecture Notes in Computer Science*, Springer-Verlag, Vol. 4492, pp. 457-464. DOI: 10.1007/978-3-540-72393-6_55.

Siddiqui, T.A. and Abdullah, Y. (2015) "Developing a nonlinear model to predict stock prices in India: An artificial neural networks approach", *IUP Journal of Applied Finance*, Vol. 21, No. 3, pp. 36-49.

Thenmozhi, M. (2006) "Forecasting stock index numbers using neural networks", *Delhi Business Review*, Vol. 7, No. 2, pp. 59-69.

Tsai, C.-F. and Wang, S.-P. (2009) "Stock price forecasting by hybrid machine learning techniques", *Proceedings of International MultiConference of Engineers and Computer Scientists* (*IMECS*), Vol. 1, March 18-20, Hong Kong.

Tseng, K-C., Kwon, O. and Tjung, L. C. (2012) "Time series and neural network forecast of daily stock prices", *Investment Management and Financial Innovations*, Vol. 9, No. 1, pp. 32-54.

Wu, Q., Chen, Y. and Liu, Z. (2008) "Ensemble model of intelligent paradigms for stock market forecasting", *Proceedings of the 1st International Workshop on Knowledge Discovery and Data Mining*, Washington DC, USA, pp. 205 – 208. DOI: 10.1109/WKDD.2008.54.

Zhang, D., Jiang, Q. and Li, X. (2007) "Application of neural networks in financial data mining", *International Journal of Computer, Electrical, Automation, and Information Engineering*, Vol. 1, No. 1, pp. 225-228. DOI: 10.5281/zenodo.1333234.

Zhu, X., Wang, H., Xu, L. and Li, H. (2008) "Predicting stock index increments by neural networks: The role of trading volume under different horizons", *Expert Systems Applications*, Vol. 34, No. 4, pp. 3043-3054. DOI: 10.1016/j.eswa.2007.06.023.

CHAPTER 7

ANALYSIS OF DIFFERENT SECTORS OF THE INDIAN ECONOMY FOR ROBUST PORTFOLIO CONSTRUCTION

JAYDIP SEN

Introduction

Different sectors of an economy have varying characteristics. For a country like India, the *information technology* (IT) sector's revenue and profitability are derived mainly from the rest of the world, particularly from North America, Europe, and the Asia-Pacific region. In contrast, the revenue of the capital goods sector is derived from the domestic economy. Sales of automobiles and consumer durables vary with the nature of the monsoon and the agricultural income and festivals; the profitability of the metal sector depends on the growth of the domestic manufacturing sector. While this is true for most economies, understanding different sectors and their characteristics has not been explicitly incorporated into the theories of portfolio choice. The reasons could be that profitability indicators, dividend history, P/E multiples, price/book value per share include the sectoral properties. If the IT sector is not performing well due to the world recession, it would be reflected in the profitability of the sector. Similarly, if the manufacturing sector is not growing domestically, then the metal sector prospects do not look good, and hence stocks from this sector trade at a discount.

Researchers have looked at specific characteristics of some sectors of the Indian economy through time series decomposition and analysis with the R programming language (Sen, 2018; Sen & Datta Chaudhuri, 2018; 2017; 2016a; 2016b; 2016c; 2016d; 2016e). The authors have tried to ascertain the sectoral characteristics through the trend, seasonal (periodic), and random patterns of the time series of different stock market indices of India. It is postulated that various sectors in an economy do not behave

uniformly, and the sectors differ in their trend patterns, seasonal characteristics, and randomness. The decomposition approach enables the study of trend, seasonal and random components separately and validates specific hypotheses. The decomposition approach helps us understand the seasonal components, during which months which sectors are strong/weak so that buy/sell decisions about companies' stocks in those sectors can be made. However, the sectors with dominant random components in their time series can be used for pure speculative gains. This decomposition also indicates which of the three components is more robust and can shed light on the *efficient market hypothesis*.

The literature on portfolio design is quite rich. Numerous contributions by several researchers in the past provide us with factors that can influence the returns that can be expected of stocks (Basu, 1977; Basu, 1983; Jaffe et al., 1989; Banz, 1981; Rosenberg et al., 1985; Chan et al., 1991; Fama & French, 1988; Fama & French, 1992; Fama & French, 1995; Chui & Wei, 1988; Bhandari, 1988; Kothari & Shanken, 1997; Ibbotson & Idzorek, 1998). Some of the notable factors identified by the researchers are: (i) *price-earnings* (P/E) ratio, (ii) *price to book* value (P/B) ratio, (iii) *debt-equity ratio* (DER), (iv) *interest coverage ratio* (ICR), (v) *gross profit margin* (GPM), (vi) *dividend pay-out ratio* (DPR), (vii) *extent of promoter holding* (EPH) of shares, (ix) *sectoral returns* (SR) and *volume* (V).

Jegadeesh and Titman show that the stock returns tend to exhibit short-term momentum; stocks that have done well over the previous few months continue to have high returns over the next month (Jegadeesh & Titman, 1993). In contrast, stocks with low returns in recent months tend to continue their poor performance for another month. Again, the pattern here is opposite to that found in work dealing with long-term overreaction, where the long-term losers outperform the long-term winners. Some researchers highlighted the importance of fundamental factors like *earnings per share* (EPS), DER, ICR, DPR, stability of earnings, and earnings growth, which investors must look at before buying shares of a company (Graham & Dodd, 1934; Dorsey, 2004).

While pointing out the key indicators for fundamental analysis of a company, most of the existing works in the literature have not considered the characteristics of different sectors to which the companies belong. Dorsey introduces the concept of *economic moats* and presents examples of such moats for various sectors of an economy (Dorsey, 2004). Lynch and Rothchild define several categories of stocks: *stalwarts*, *slow growers*, *cyclical* and *fast growers* (Lynch and Rothchild, 2000). However, their analysis does not reflect the true nature of the sectors and only refines the fundamental parameters mentioned above.

Of late, there are quite a few interesting works on portfolio design and robust forecasting methods of stock prices that have been proposed in the literature. We mention some of them below.

Chow et al. examine different *optimized minimum-variance* portfolios of stock. The authors propose an efficient heuristic approach for portfolio construction that assigns suitable weights which are inversely proportional to the volatility of the stocks and their *beta* values for realizing low-volatility investment that yields higher returns at lower risk (Chow et al., 2014). The authors also studied the long-term performance of the strategies using the following characteristics: (i) sector, (ii) country, (iii) stock concentration, (iv) turnover, (v) liquidity, and (vi) capacity. The authors find that the optimized and the heuristic-based methodologies have almost identical risk profiles. To justify this observation, the authors argue that the volatility of a portfolio depends on the *beta* of the equity market. An approach that shifts portfolio allocations from *high-beta* stocks to *low-beta* stocks has the volatility that matches the minimum-variance portfolio.

Carriero et al. present a new statistical model based on Bayesian Vector Autoregression (BVAR) for the entire term structure of interest rates (Carriero et al., 2012). The BVAR has a time variation that approaches that of a set of univariate *autoregression* (AR) models. The authors contend that even if the univariate AR and the *random walk* models usually produce superior results, there are time intervals with enhanced interaction between the yields at different maturities. If one can effectively capture such interactions, it will increase the robustness and accuracy of the forecasting method. An extensive set of results has been presented on the forecasting performance of the proposed approach. Some significant observations found in this study are: (i) the proposed model produces higher forecasting accuracy than the random walk model, (ii) for some selected maturities and forecast horizons, BVAR performs poorly in comparison to some models, and (iii) in the remaining period, BVAR's performance is far superior to those of the other methods.

Bao et al. propose a stock price forecasting model that uses a deep learning-based approach combining the techniques of *wavelet transforms* (WT), *stack autoencoders* (SAEs), and *long-and-short-term memory* (LSTM) (Bao et al., 2016). The time series of the stock price is first decomposed using the WT to eliminate noise. In the next step, numerous high-level features of the stock price movement are generated by applying the SAEs. Finally, the high-level denoised features of the stock price movement are given as inputs to the LSTM to predict the *close* prices of the stocks for the next day. The authors select six market indices to test the performance of their proposed predictive models. These six indices are: (i)

CSI 300 index in mainland China, (ii) NIFTY 50 index in India, (iii) Hang Seng index in Hong Kong, (iv) Nikkei 225 index in Tokyo, (v) S&P 500 index, and (vi) DJIA index in New York. The performance of the combined system, which the authors call WSAE-LSTM, is compared with those of three other methods: (i) WLSTM - a combination of WT and LSTM, (ii) LSTM, and (iii) *recursive neural network* (RNN). WSAE-LSTM is found to have produced more accurate forecasting of stock prices than the other three methods.

Berger presents a method of forecasting the financial return by transforming a return series into its frequency and time domain using the wavelet decomposition technique (Berger, 2015). The transformation enables one to separate the short-term noise from the long-term trends exhibited by the return series. In this way, it is possible to assess the relevance of each frequency to the *value at risk* (VaR) forecast. The authors also analyze the price movements of stocks at different times, stable and volatile situations, and observe that daily 95% VaR forecasts are mainly due to the volatility that could be captured by the short-term information. However, for predicting 99% VaR, the author finds that larger timescales are needed.

Zhao et al. propose a Bayesian approach to forecasting large-dimensional time series data, which is applicable in the financial industry (Zhao et al., 2016). The proposition is based on building multiple time series to model multivariate volatility while making the model scalable and efficient. The authors leverage the concept of dynamic dependence network models wherein the individual time series can be decoupled for a more in-depth analysis. Upon completion of the analysis, the time series can be recoupled for making robust forecasting and further decision making. The Bayesian model that averages the discounted historical information is effectively used in the predictive framework. The authors have also demonstrated how these models can be used for efficiently designing financial portfolios.

Caldeira et al. argue that model uncertainties in finance and business are usually taken care of by using a combination of forecasting strategies that alleviates the effects of structural breaks and model misspecification (Caldeira et al., 2017). On the other hand, model weights are estimated using statistical measures of forecast accuracy. The estimation of the weights usually has no contextual connotations with the decision-making process in which the forecasts are used. A novel *forecast combination strategy* solves the problem, wherein the model weights are computed based on the return yielded by a forecast model. To validate their proposition, the authors compute the average net excess returns, the standard deviation, and the Sharpe ratio of the portfolios formed with nine different yield curve

specifications with 12 different forecast combination strategies, and using two different datasets. The results indicate that the return-based approach proposed by the authors performs superior to the methods that use statistical measures of forecast accuracy. Moreover, the return-based approach is found to be more robust to monetary policy changes.

We argue that understanding companies by some specific set of fundamental parameters leads to standardization and robs their individuality. Each company is distinct, and one of the sources of this distinctiveness stems from the sector to which it belongs. While a company is a microeconomic entity, an industry is macroeconomic. It has its dynamics arising from the socio-economic and demographic characteristics, the income distribution pattern, the extent of global integration, the domestic endowment of resources, the state of the technology, and the market size of the nation. Most of the propositions in the literature we mentioned have not explicitly captured these aspects, and hence sectoral distinctions have not been modeled adequately. Thus, there remains this one aspect of portfolio choice that requires deliberation.

Further, financial market players are not merely interested in understanding past patterns. They are more interested in getting hold of a framework that can predict the future prices of stocks. Our approach provides such a framework.

The rest of the chapter is organized as follows. In the section titled *Related Work*, some existing works on portfolio selection and management are briefly discussed. In the section titled *Method of Time Series Decomposition*, we discuss how the time series index of a sector or stock can be decomposed into its three components – *trend*, *seasonal* and *random*. The section titled *Decomposition of Time Series Data of the Sectoral Indices* provides a detailed analysis of several stocks from some chosen fifteen sectors of the Indian economy. We illustrate how a sector's overall behavior is also reflected in the stock prices and how investors can exploit that information in their portfolio formation. In the section titled *Forecasting of Sectoral Index*, four forecasting methods of sectoral indices are presented. These methods are generic, and hence, they can also be applied to predict the prices of individual stocks. The forecasting performance of each method is shown for all the sectors, and a comparative analysis is made. Finally, the section titled *Conclusion* concludes the chapter.

Related Work

Akcay and Yalcin present a novel approach for optimal portfolio selection that allocates assets based on the maximization of the expected return while considering a possible shortfall probability constraint (Akcay & Yalcin, 2010). In this way, the approach models a typical risk-averse investor whose primary objective is to minimize the maximum likely loss. The model produces superior results compared to the mean-variance approach based on *cumulative cash values*, *geometric mean returns*, and *average risk-adjusted returns*.

Anagnostopoulos and Mamanis propose a solution to portfolio optimization problems using *multi-objective evolutionary algorithms* (MOEAs) (Anagnostopoulos & Mamanis, 2011). The authors use *non-dominated sorting genetic algorithm II* (NSGA-II), *Pareto envelop-based selection algorithm* (PESA), and *strength Pareto evolutionary algorithm 2* (SPEA2) for building a portfolio optimization model to maximize the return and minimize the risk in a portfolio. The results indicate that MOEAs are effective and efficient in portfolio optimization, and their performance is usually independent of the risk function used in the portfolio model.

Armananzas and Lozano present another *multi-objective optimization* approach to the portfolio selection problem (Armananzas & Lozano, 2005). The authors use three optimization techniques: *greedy search*, *simulated annealing*, and *ant colony optimization*, for building the multi-objective optimization framework.

Cesarone et al. present a novel approach for solving portfolio management problems using a variant of the *limited asset Markowitz model* (LAM), in which the assets are limited by several quantity and cardinality constraints (Cesarone et al., 2013). The authors reformulate the LAM model problem as a standard quadratic program and solve portfolio problems of smaller size. The authors also design heuristics for extensive portfolios that find the optimal solutions for some standard financial data.

Cui et al. propose a *nonlinear portfolio selection model* using approximate parametric *value-at-risk* (VaR) (Cui et al., 2013). The authors demonstrate how to construct portfolios using the first-order and second-order approximations of VaR in polynomial time using *interior-point methods*.

Deng and Li propose a portfolio management model with borrowing constraints (Deng & Li, 2012). The model is built using a possibilistic mean, variance, and covariance under the assumption of fuzzy asset returns. The authors also present a quadratic programming model that incorporates an inequality constraint for fuzzy trapezoid numbers for asset returns.

Duan and Stahlecker present a portfolio selection model in which the future stock returns are given as fuzzy sets (Duan & Stahlecker, 2011). The authors argue that investment decisions are not usually based on statistical expectation values. Instead, they are guided by the maximal and minimal potential returns. After determining the maximal and minimal returns and aggerating them, the proposed method computes the weights for the best and the worst possible cases. Based on these weights, the authors design a new objective function for the optimal portfolio.

Huang and Qiao present a *multi-period portfolio selection problem* with a built-in evaluation mechanism (Huang & Qiao, 2012). The authors propose an uncertain risk adjustment model for the security returns and build an optimal portfolio model that maximizes the total incremental wealth.

Kumar and Bhattacharya present an agent-based model for portfolio selection with a constraint on the number of stocks in the portfolio (Kumar & Bhattacharya, 2012). The proposed scheme uses a collection of autonomous agents. Each agent starts with a pseudorandom portfolio and follows individual investment strategies as it scans through the historical stock price data. Periodically, the agents share information among themselves, and they can switch from one stock to another.

Li et al. propose a fuzzy portfolio selection model based on a *possibilistic return* and a *possibilistic risk* (Li et al., 2015). For returns of assets, the authors propose a background risk-based portfolio selection model. Using historical stock prices, the authors develop an efficient portfolio with background risk and then compare it with a portfolio return that does not consider any background risk. The results indicate that model that considers the background risk is more efficient in portfolio management.

Mehlawat and Gupta present a portfolio selection model from the perspective of *chance-constrained multi-objective optimization* (Mehlawat & Gupta, 2014). While maintaining the liquidity in the portfolio, the model attempts to maximize both the short-term and long-term returns. To capture the uncertainty in the market, the authors use the fuzzy parameters in generalized functional forms. The optimization problem is solved using a hybrid algorithm that combines fuzzy simulation with a real-coded genetic algorithm.

Liu and Zhang present a multi-period fuzzy portfolio optimization problem that involves a minimum number of transaction lots (Liu & Zhang, 2015). Using the *possibility theory*, the authors design a mean-semivariance portfolio selection model that maximizes the terminal wealth and minimizes the cumulative risk over the entire investment horizon.

Rios and Sahinidis use an indefinite *quadratic utility function* for optimizing a portfolio (Rios & Sahinidis, 2010). The convex and the concave segments of the utility function reflect the attitude of the investor towards risk. The segments change based on the deviations from a fixed goal, and accordingly, the associated risk also varies. The authors propose an optimization technique for solving the non-convex optimization problem associated with portfolio selection. The results show that the proposed approach consistently selects portfolios with higher skewness values than the classical expectation-variance system.

Sadjadi et al. propose a portfolio optimization model using a genetic algorithm (Sadjadi et al., 2015). The scheme considers the risks associated with a portfolio and makes an optimization to minimize the risk while maximizing the maximum return of assets. The proposal is validated using a set of well-known and widely used benchmark data sets.

Vaclavik and Jablonsky argue that fluctuations in market sentiments are responsible for causing changes in financial instrument prices (Vaclavik & Jablonsky, 2011). The change in the prices of the financial instruments reduces the efficiency of the traditional method of risk diversification. Traditionally, risk diversification depends on the Markowitz Optimal Portfolio Selection Model (MOPSM). MOPSM is based on a *convex quadratic programming problem*, and any attempt to reformulate the model leads to a suboptimal solution. The authors propose a way to transform the original MOPSM into a convex optimization problem. The method of transformation is robust enough to work in the presence of correlations among the financial instruments that tend to destabilize the convexity of the original problem.

Yang et al. formulate the multi-account investment decision problem as a *Nash Equilibrium Problem* (NEP) (Yang et al., 2013). The authors propose a generalized NEP for multi-account investment problems. Global constraints are imposed on all the accounts, and a desirable outcome is defined for each of them. The scheme utilizes several well-known signal processing techniques.

Zhang et al. present a portfolio optimization model that uses a V-shaped transaction cost associated with a shift from the current portfolio to an adjusted one (Zhang et al., 2011). The proposition includes an optimization algorithm known as the *sequential minimal optimization* (SMO) algorithm that guides the adjustment in the portfolio. The experimental results show that the SMO algorithm is highly efficient in constructing an optimal portfolio of stocks.

Zhu et al. propose a novel portfolio selection model that optimizes the return distribution model and the portfolio return (Zhu et al., 2014). The

proposed mixed distribution strategy harmonizes information from several channels such as the historical data, market information, and the subjective views of the investors. The simulation results validate the hypothesis of the model construction.

Mehtab and Sen present a series of machine learning, deep learning, and text mining-based predictive models for stock price forecasting with a very high level of accuracy (Mehtab & Sen, 2022; Mehtab & Sen, 2021a; Mehtab et al., 2021b; Mehtab & Sen, 2020a; Mehtab & Sen, 2020b; Mehtab et al., 2020c; Mehtab & Sen, 2020d; Mehtab & Sen, 2019; Sen & Mehtab, 2021a; Sen & Mehtab, 2021b; Sen et al., 2020). These models can be used as the core components in a portfolio selection and management system.

Method of Time Series Decomposition

We use the *R programming language* for data management, data analysis, and presentation of results (de Micheaux et al., 2013). The language R is an open-source language with very rich libraries, and it is ideally suited for data analysis work. We use daily data from the Bombay Stock Exchange (BSE) on fifteen important sectors of the Indian economy from January 2010 to December 2020. These fifteen sectors are: (i) *auto*, (ii) *banking*, (iii) *capital goods*, (iv) *consumer durable*, (v) *fast moving consumer goods* (FMCG), (vi) *healthcare*, (vii) *information technology* (IT), (viii) *large-cap*, (ix) *metal*, (x) *mid cap*, (xi) *oil and gas*, (xii) *power*, (xiii) *realty*, (xiv) *small cap*, and (xv) *telecommunication*. The monthly index values (*close* values for each month) for each sector are first stored in fifteen plain text (*.txt*) files – each sector data in one file. There are eleven years under study, and each year consists of twelve records, the text file for each sector contains 132 records. The records in the text file for each sector are read into an R variable using the *scan* function. The resultant R object is converted into a monthly time series object using the *ts* function defined in the *R* programming language. For each of the fifteen sectors, the monthly time series variable in R is now an aggregate of its three constituent components: (i) *trend*, (ii) *seasonal* (periodic), and (iii) *random*. We then decompose the time series into its three components. For this purpose, we use the *decompose* function. Once the time series variables are constructed, the *plot* function is used to produce the time series plot and its three components.

For example, the time series of the banking sector index of India and its decomposition into three components from January 2010 to December 2020 are shown in Figure 7-1. The figure has four boxes arranged in a stack.

The boxes depict the *overall time series*, the *trend*, the *seasonal*, and the *random* component, respectively, set from top to bottom.

The time series of each sector depicts how the sector performed over the last eleven years and how its three components behaved. For example, a consistently positive trend for a time series indicates a sustained growth of the sector over the period. A time series with a significant seasonal component implies a strong seasonality in the sector. A high degree of randomness is exhibited by a strong random component in the time series.

It is evident from Figure 7-1 that the banking sector index passed through several different phases during the period of our study. In 2010, the time series exhibited an increasing tendency before it started falling in its value in the last couple of months of the year. The downward fall of the index continued throughout 2011 before it began to increase at a very sluggish pace in 2012. However, the index experienced a continual fall in 2013 before it started increasing again in November 2013. The year 2014 witnessed a consistent rise in the banking sector index. A downward fall in the index started at the beginning of 2015 which continued until the end of the year. However, in March 2016, the index started picking up its value rapidly till November 2016, before it started falling again. The year 2017 was a year of consistent growth in the banking sector index. The sluggish growth of the sector continued through 2018 and 2019. However, in March 2020, the global pandemic COVID-19 caused a sharp fall in the banking sector index. From October 2020, the index started increasing again. The trend of the time series was very sluggish during June 2010-December 2013, before it began to rise consistently till June 2015. The trend value decreased consistently from then till the end of the year 2016. The trend increased till August 2019. After August 2019, the banking sector trend experienced either a fall or exhibited sluggish behavior. The seasonal component of the time series was found to be at its highest value during October, while the lowest value of the seasonal component occurred during March. A close look at the graph reveals that the random component of the time series was also quite significant at some specific points.

Similarly, we decompose the time series of the index values of the remaining fourteen sectors and analyze their components to understand each sector's characteristics.

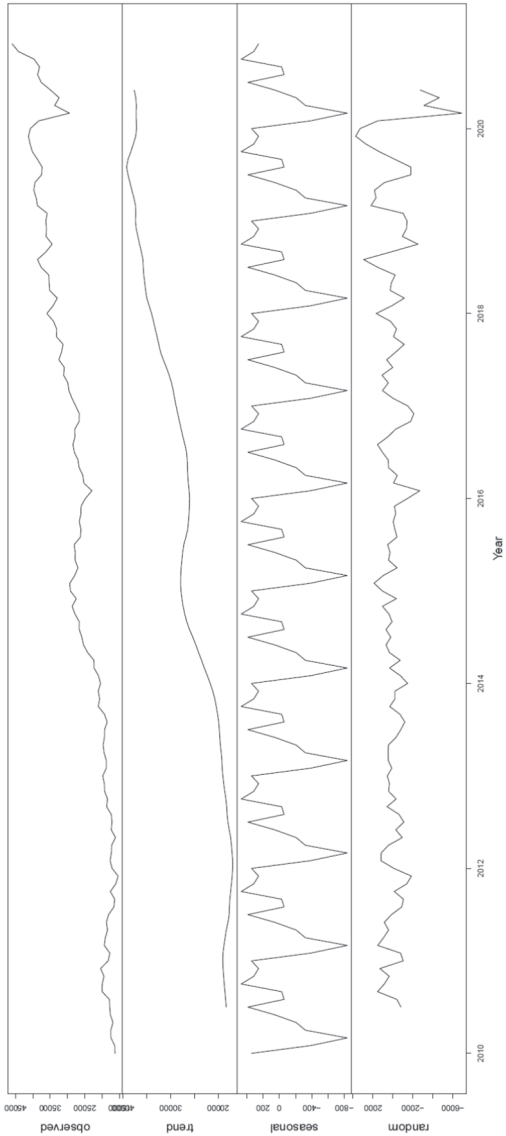

Figure 7-1. Decomposition of Indian banking sector time series into its *trend*, *seasonal* and *random* components
(Period: January 2010 – December 2020)

Decomposition of Time Series Data of the Sectoral Indices

In this section, we discuss the decomposition results of the banking sector time series of the Indian economy. Table 7-1 presents the numerical values of the time series data and its three components. It is interesting to observe that the trend and random components are not available from January 2010 to June 2010 and from July 2020 to December 2020. Let us understand why it is so. The *decompose* function in the R language uses a *12-month moving average* method for computing the trend component. As an example, we first illustrate the computation of the trend value corresponding to July 2011 as follows:

$$X = \frac{Sum\ of\ the\ time\ series\ values\ for\ the\ period: Jan\ 2011 - Dec\ 2011}{12} = \frac{212692.6}{12} = 17724.38 \quad (1)$$

$$Y = \frac{Sum\ of\ the\ time\ series\ values\ for\ the\ period: Feb\ 2011 - Jan\ 2011}{12} = \frac{211558.4}{12} = 17629.87 \quad (2)$$

$$Trend\ value\ for\ July\ 2010 = \frac{X+Y}{2} = \frac{17724.38 + 17629.87}{2} = 17677.12 \quad (3)$$

In (1), the variable X stores the value of the 12-month moving average of the time series for January 2011 - December 2011. In (2), the variable Y stores the same from February 2011 to January 2012. The values of X and Y are computed based on the time series records listed in Table 7-1 are used. The average of X and Y is computed in (3), which gives the trend value for July 2011. In this way, the trend values for all other months are calculated.

Following the same approach, to compute the trend value for January 2010, we need time series data from July 2009 to June 2010. However, since in this work, we use the time series data from January 2010 to December 2020, the first trend value the *decompose* function could compute was for July 2010, and the last was for June 2020. Thus, the trend values from January 2010 to July 2010 and July 2020 to December 2020 could not be computed.

For computing the seasonal component, the *decompose* function first *detrends* (i.e., subtracts the trend component from the overall time series) and arranges the *detrended* time series values in a 12 column format. The seasonal value for each month is derived by computing the average of each column. The value of the seasonal component for a given month remains the same for the entire period. As an illustration, we compute the seasonal value for August. First, we compute the mean seasonal value for August as follows:

$$Mean\ seasonality\ for\ Aug = \frac{Sum\ of\ detrended\ values\ for\ Aug}{10} = \frac{-1049.29}{10} = -104.93 \quad (4)$$

For computing the detrended values in (4), we use the time series and the trend values listed in Table 7-1. The detrended values for August are computed by subtracting the trend values from the corresponding time series values for the same month. The detrended values for the month of August are: -458.67, -950.17, -676.61, -1253.03, 631.3, -475.50, 1475.15, -351.47, 2866.67, -1856.96 for the years 2010-2019 respectively. The mean seasonal value for August is computed as the mean of the detrended values as in (4). Next, we compute the grand mean of the detrended series. For computing the grand mean of the detrended series, we pass the variable storing the detrended values in the *mean* function and a second parameter *na.rm* = TRUE. In other words, if *detrend* is the variable that stores the detrended series, then we use the code: *mean(detrend, na.rm=TRUE)* for computing the grand mean. The grand mean for the detrended series is -46.8. Finally, the seasonal component for a month is found by computing the difference between the *mean* of the month and the *grand mean*. The mean seasonal values for the months: January-December are found to be: 295.70, -429.79, -882.04, -366.50, -252.74, 10.52, 340.96, -104.93, -74.20, 423.67, 268.05, and 209.55, respectively. Hence, the seasonal value for August, for example, is computed as follows:

$$Seasonality\ for\ Aug = -104.93 - (-46.81) = -58.12 \quad (5)$$

Similarly, the seasonal values for all other months are computed. The seasonality for the months January to December are computed as: 295.70 + 46.81= 342.51, -429.79 + 46.81= -382.98, -882.04 + 46.81= -835.23, -366.50 + 46.81= -319.69, -252.74 + 46.81 = -205.93, 10.52 + 46.81 = 57.33, 340.96 + 46.81 = 387.77, -104.93 + 46.81 = -58.12, -74.20 + 46.81 = -27.39, 423.67 + 46.81 = 470.48. 268.05 + 46.81 = 314.86, 209.55 + 46.81 = 256.36, respectively. Finally, the random components are obtained by subtracting the sum of the corresponding trend and seasonal values from the overall time series values. Since the trend values from January 2010 to June 2010 and from July 2020 to December 2020 are missing, the random components for those periods could not be computed as well.

TABLE 7-1. INDIAN BANKING SECTOR TIME SERIES AND ITS COMPONENTS (PERIOD: JANUARY 2010 – DECEMBER 2020)

Year	Month	Aggregate	Trend	Seasonal	Random
2010	Jan	16357.96		342.51	NA
	Feb	16429.55		-382.98	NA
	Mar	17527.77		-835.23	NA
	Apr	17558.71		-319.69	NA
	May	16944.63		-205.93	NA
	Jun	17700.90		57.33	NA
	Jul	17868.29	18289.64	387.77	-809.12
	Aug	17971.12	18429.79	-58.12	-400.55
	Sep	20069.12	18567.76	-27.39	1528.75
	Oct	20032.34	18713.37	470.48	848.49
	Nov	19521.25	18844.03	314.86	362.35
	Dec	20509.09	18956.68	256.36	1296.04
2011	Jan	18327.76	19018.10	342.511	-1032.85
	Feb	17823.40	18977.87	-382.98	-771.50
	Mar	19445.22	18773.30	-835.23	1507.15
	Apr	19135.96	18525.68	-319.69	929.97
	May	18503.28	18287.14	-205.93	422.07
	Jun	18845.87	17934.97	57.33	853.57
	Jul	18197.20	17677.12	387.77	132.30
	Aug	16676.75	17626.92	-58.12	-892.05
	Sep	16453.76	17538.93	-27.39	-1057.78
	Oct	17705.01	17378.17	470.48	-143.64
	Nov	16123.46	17207.26	314.86	-1398.66
	Dec	15454.92	17053.07	256.36	-1854.51
2012	Jan	17193.55	16954.03	342.511	-102.99
	Feb	17752.68	16945.35	-382.98	1190.29
	Mar	17404.20	17072.93	-835.23	1166.50
	Apr	17318.81	17202.48	-319.69	436.02
	May	16218.53	17369.85	-205.93	-945.39
	Jun	17429.98	17669.36	57.33	-296.71
	Jul	17236.18	17947.41	387.77	-1099.00
	Aug	17429.56	18106.17	-58.12	-618.49
	Sep	18762.74	18212.02	-27.39	578.10
	Oct	18505.38	18362.73	470.48	-327.83
	Nov	19339.90	18601.36	314.86	423.68
	Dec	19426.71	18830.84	256.36	339.50
2013	Jan	19894.98	19000.65	342.511	551.82
	Feb	18861.54	19138.14	-382.98	106.37
	Mar	18835.77	19213.44	-835.23	457.56
	Apr	19504.18	19349.94	-319.69	473.93
	May	19760.30	19521.24	-205.93	444.98
	Jun	19395.81	19654.41	57.33	-315.93

	Jul	19345.70	19752.86	387.77	-794.93
	Aug	18619.72	19872.75	-58.12	-1194.92
	Sep	19379.77	20114.80	-27.39	-707.64
	Oct	21164.52	20384.14	470.48	309.90
	Nov	20791.93	20691.25	314.86	-214.18
	Dec	21170.68	21127.71	256.36	-213.39
2014	Jan	20513.85	21651.34	342.511	-1480.00
	Feb	21120.12	22258.33	-382.98	-755.24
	Mar	22386.27	22894.54	-835.23	326.95
	Apr	22417.80	23475.88	-319.69	-738.39
	May	24217.34	24084.35	-205.93	338.92
	Jun	25413.78	24677.30	57.33	679.15
	Jul	25894.97	25302.21	387.77	204.99
	Aug	26638.11	26006.81	-58.12	689.41
	Sep	26630.51	26582.34	-27.39	75.56
	Oct	27865.83	27005.87	470.48	389.48
	Nov	28693.99	27347.73	314.86	1031.40
	Dec	27499.42	27596.82	256.36	-353.76
2015	Jan	29182.95	27787.93	342.511	1052.51
	Feb	29361.50	27865.62	-382.98	1878.85
	Mar	27957.49	27831.01	-835.23	961.71
	Apr	27011.31	27760.81	-319.69	-429.81
	May	27828.44	27604.26	-205.93	430.11
	Jun	27780.83	27440.50	57.33	283.00
	Jul	28114.56	27203.24	387.77	523.55
	Aug	26283.09	26758.59	-58.12	-417.38
	Sep	26154.83	26384.62	-27.39	-202.40
	Oct	26656.83	26217.11	470.48	-30.76
	Nov	26145.67	26110.23	314.86	-279.42
	Dec	26117.54	26029.33	256.36	-168.15

2016	Jan	24870.69	25994.17	342.511	-1465.99
	Feb	23002.00	26081.93	-382.98	-2696.97
	Mar	25341.86	26243.61	-835.23	-66.52
	Apr	25606.62	26367.96	-319.69	-441.65
	May	26667.96	26442.15	-205.93	431.73
	Jun	26999.72	26484.49	57.33	457.90
	Jul	28051.86	26621.75	387.77	1042.34
	Aug	28452.17	26977.02	-58.12	1533.27
	Sep	27865.96	27394.52	-27.39	498.83
	Oct	27930.21	27752.45	470.48	-292.72
	Nov	26652.81	28118.69	314.86	-1780.74
	Dec	26626.46	28468.68	256.36	-2098.58
2017	Jan	27655.96	28818.05	342.511	-1504.60
	Feb	28743.32	29140.61	-382.98	-14.32
	Mar	29620.50	29419.61	-835.23	1036.11
	Apr	29918.40	29782.14	-319.69	455.95
	May	31145.80	30272.95	-205.93	1078.78
	Jun	30921.61	30853.24	57.33	11.04
	Jul	32514.94	31509.05	387.77	618.12
	Aug	31730.49	32081.96	-58.12	-293.35
	Sep	31283.72	32448.16	-27.39	-1137.05
	Oct	33213.13	32806.08	470.48	-63.43
	Nov	33149.35	33198.52	314.86	-364.04
	Dec	34056.83	33560.12	256.36	240.34
2018	Jan	35965.02	33959.85	342.511	1662.66
	Feb	34184.04	34460.11	-382.98	106.89
	Mar	32968.68	34954.20	-835.23	-1150.29
	Apr	35160.36	35211.38	-319.69	268.67
	May	35322.38	35389.45	-205.93	138.85
	Jun	35423.48	35600.14	57.33	-233.99
	Jul	37606.58	35696.11	387.77	1522.70
	Aug	38645.07	35778.40	-58.12	2924.79
	Sep	36227.14	36086.22	-27.39	168.31
	Oct	34442.05	36485.19	470.48	-2513.62
	Nov	36194.30	36829.49	314.86	-950.05
	Dec	36068.33	37177.94	256.36	-1365.98

2019	Jan	36256.69	37338.18	342.511	-1424.00
	Feb	35867.44	37278.28	-382.98	-1027.87
	Mar	38672.91	37325.27	-835.23	2182.86
	Apr	39031.55	37663.90	-319.69	1687.34
	May	39714.20	38092.51	-205.93	1827.62
	Jun	39394.64	38500.21	57.33	837.10
	Jul	37481.12	38902.39	387.77	-1809.04
	Aug	37332.79	39189.75	-58.12	-1798.84
	Sep	38667.33	38907.48	-27.39	-212.76
	Oct	40129.05	38302.54	470.48	1356.03
	Nov	40793.81	37777.38	314.86	2701.57
	Dec	41253.74	37287.00	256.36	3710.37
2020	Jan	40723.49	37105.63	342.511	3275.35
	Feb	38297.29	37164.85	-382.98	1515.41
	Mar	29468.49	37193.85	-835.23	-6890.13
	Apr	33717.62	37147.42	-319.69	-3110.11
	May	32424.10	37265.79	-205.93	-4635.76
	Jun	34915.80	37601.71	57.33	-2743.24
	Jul	37606.89		387.77	NA
	Aug	38628.29		-58.12	NA
	Sep	38067.93		-27.39	NA
	Oct	39614.07		470.48	NA
	Nov	44149.72		314.86	NA
	Dec	45959.88		256.36	NA

From Table 7-1, we identify the month of the *highest* and the *lowest seasonality*, the behavior of the *trend* component, and the contribution of the *random* components in the time series. To understand the degree of randomness in the time series, we compute the ratio of the random component to the aggregate value of the time series. The *highest* and the *lowest* percentage of the random component for the aggregate index of the banking sector time series are found to be 8.99 and -23.38, respectively. The *highest ratio* of the random component is found in December 2019, while the *lowest* ratio is observed in March 2020. The mean of the absolute values of the random component percentage for the banking sector time series is found to be 3.69, while the standard deviation turns out to be equal to 3.32. These values indicate that the time series of the banking sector from January 2010 to December 2020 contains a moderately strong random component.

TABLE 7-2. SEASONALITY CHARACTERISTICS OF DIFFERENT SECTORS OF THE INDIAN ECONOMY (PERIOD: JANUARY 2010 - DECEMBER 2020)

Sl No	Sector	Month of Highest Seasonality	Month of Lowest Seasonality	Max Value of Seasonal Comp.	Min Value of Seasonal Comp.
1	Auto	October	March	550	-694
2	Banking	October	February	470	-835
3	Capital Goods	June	February	555	-515
4	Consumer Durable	January	May	334	-263
5	FMCG	July	February	158	-275
6	Healthcare	October	March	415	-342
7	IT	January	June	320	-300
8	Large Cap	October	March	56	-92
9	Metal	December	August	346	-340
10	Mid Cap	December	March	266	-374
11	Oil & Gas	October	March	485	-334
12	Power	June	Feb	45	-47
13	Realty	July	March	40	-58
14	Small Cap	October	March	294	-497
15	Telecommunication	July	March	53	-37

In summary, we can conclude that the banking sector of the Indian economy did not exhibit a consistently good performance from January 2010 to December 2020. Interestingly, the index fluctuated alternately, showing a rise during 2010, 2012, 2014, 2016, and 2018, and a fall during 2011, 2013, 2015, 2017, and 2019. When the trend experienced a consistent fall in its value, the random component exhibited its dominance, making the time series volatile.

Following the same approach, we decompose the sectoral index values of the other fourteen sectors and compute the three components. We identify the months of the highest and the lowest seasonality exhibited by each sector based on its decomposition results. Table 7-2 lists the months of the highest seasonality and the months of the lowest seasonality for each sector.

After identifying the months of the highest and the lowest seasonality for each sector, we investigate whether the same seasonality patterns are also exhibited by the stock prices of the companies in those sectors. For this purpose, we select five stocks from each of the fifteen sectors from the list of top ten companies in terms of their market capitalization and study their price movements from January 2010 to December 2020. In the following, we present the results of our study.

TABLE 7-3. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE AUTO SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Maruti Suzuki		Mahindra		Bajaj Auto		Hero MotoCorp		Tata Motors	
	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct
2020	5359	7036	367	722	2628	3173	2167	3105	93	180
2019	6666	7246	531	645	2993	3176	2432	2513	161	214
2018	**8815**	**7662**	791	873	**2948**	**2737**	**3774**	**3058**	171	340
2017	6525	8599	668	703	2872	3312	3330	3642	405	457
2016	3795	5266	592	665	2485	2684	2901	3169	408	459
2015	3726	4555	573	683	1949	2482	2329	2703	**512**	**423**
2014	1923	3346	537	661	1913	2640	2194	3142	410	527
2013	1670	1677	462	472	1882	1975	1645	2051	296	394
2012	1372	1474	355	474	1623	1930	1828	2174	271	313
2011	972	1318	363	378	1463	1673	1710	2003	170	243
2010	1280	1422	263	382	1047	1575	1905	1973	172	245

Auto Sector: The ten most significant companies in this sector in terms of their market capitalization in December 2020 (i.e., at the time of carrying out our study) are: (i) Maruti Suzuki India Ltd. (MSIL), (ii) Tata Motors Ltd. (TML), (iii) Mahindra & Mahindra Ltd. (MML), (iv) Bajaj Auto Ltd. (BAL), (v) Eicher Motors Ltd. (EML), (vi) MRF, (vii) Hero MotoCorp Ltd. (HML), (viii) Motherson Sumi Systems Ltd. (MSSL), (ix) Ashok Leyland (AL), and (x) Balkrishna Industries Ltd. (BIL). We note the price movements of the five stocks in this sector. The results are listed in Table 7-3. It is observed from Table 7-3 that out of the 55 records (there are 11 years and five stocks), only four records do not follow the seasonality pattern of the auto sector. The auto sector exhibits the highest positive seasonality in October. The highest negative seasonality is shown in February. The records that do not follow this pattern of seasonality are Maruti Suzuki in 2018, Bajaj Auto in 2018, Hero MotoCorp in 2018, and Tata Motors in 2015.

TABLE 7-4. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE BANKING SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	HDFC Bank		ICICI Bank		Kotak Bank		Axis Bank		State Bank	
	Feb	Oct	Feb	Oct	Feb	Oct	Feb	Oct	Feb	Oct
2020	862	1441	325	473	1296	1902	379	602	197	244
2019	1158	1274	399	512	1336	1614	740	776	321	342
2018	946	1059	278	355	1049	1232	509	626	250	284
2017	721	926	252	308	872	1000	491	542	293	306
2016	536	600	215	241	681	756	445	470	194	258
2015	511	1076	249	287	657	692	**546**	**469**	250	267
2014	374	479	226	319	389	601	292	481	174	321
2013	312	331	190	194	326	378	231	260	**207**	**182**
2012	260	647	161	200	271	334	229	263	214	217
2011	234	221	**202**	**130**	228	232	**281**	**190**	**277**	**178**
2010	193	229	173	208	187	238	234	274	208	349

Banking Sector: The top ten stocks in this sector in terms of their market capitalization values in December 2020 are: (i) HDFC Bank, (ii) ICICI Bank, (iii) Kotak Mahindra Bank, (iv) Axis Bank, (v) State Bank of India, (vi) IndusInd Bank, (vii) Bandhan Bank, (viii) Federal Bank, (ix) RBL Bank, and (x) IDFC First Bank. Table 7-4 presents data on the historical stock prices of five banks of India from the top ten list. We observed in Table 7-2 that the banking sector exhibits the highest positive and negative seasonality during October and February, respectively. Table 7-4 shows that except for five records, the stock prices in the banking sector follow the seasonality pattern of the overall sector. The observations that do not follow the sector's seasonality pattern are ICICI Bank in 2011, Axis Bank in 2015 and 2011, and State Bank of India in 2013 and 2011.

TABLE 7-5. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE CAPITAL GOODS SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Larsen & Toubro		**Siemens**		**Havells**		**BEL**		**ABB**	
	Feb	Jun	Feb	Jun	Feb	Jun	Feb	Jun	Feb	Jun
2020	807	913	1113	1162	480	583	74	96	**935**	**895**
2019	1384	1387	1129	1145	**771**	**646**	93	103	1316	1363
2018	**1312**	**1303**	**1073**	**989**	488	634	**142**	**115**	**1295**	**1218**
2017	1052	1192	1255	1455	468	472	142	162	1162	1421
2016	811	1039	1100	1319	322	389	111	112	1148	1164
2015	1146	1193	1397	1455	**305**	**279**	101	121	1144	1259
2014	848	1003	772	877	186	239	35	53	775	944
2013	607	846	**549**	**512**	122	130	33	35	444	454
2012	581	609	681	760	108	114	35	46	723	738
2011	734	767	881	924	73	74	51	53	723	789
2010	723	799	699	740	60	66	**66**	**55**	737	754

Capital Goods Sector: The top ten companies in this sector in terms of their market capitalization in December 2020 are: (i) Larsen & Toubro, (ii) Siemens, (iii) Havells India, (iv) Hindustan Aeronautics Ltd (HAL), (v) Bharat Electronics Ltd (BEL), (vi) Honeywell Automation, (vii) ABB India, (viii) AIA Engineering, (ix) Bharat Heavy Electricals Ltd. (BHEL), and (x) Thermax. Table 7-5 presents the price movements of five stocks in the capital goods sector. Table 7-5 shows that, except for nine cases, the stock prices of the capital goods sector exhibit higher values in June (i.e., the month of the highest seasonality of the capital goods sector). Lower prices are shown in February (i.e., the month of the lowest seasonality of the capital goods sector). The records that do not follow the seasonality pattern of the sector are Larsen & Toubro in 2018, Siemens in 2018 and 2013, Havells in 2019 and 2015, BEL in 2018 and 2010, and ABB in 2020 and 2018.

TABLE 7-6. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE CONSUMER DURABLE SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Titan Company		Havells		Voltas		Whirlpool India		Bata India	
	May	Jan	May	Jan	May	Jan	May	Jan	May	Jan
2019 -2020	1056	1255	**646**	**614**	644	679	1590	2176	1445	1644
2018-2019	879	1027	634	715	568	630	**1735**	**1378**	908	1289
2017-2018	524	816	460	507	458	609	1137	1427	588	732
2016-2017	405	437	389	405	321	372	831	1038	509	604
2015-2016	**353**	**316**	260	272	**318**	**223**	**746**	**596**	457	598
2014-2015	326	435	239	273	192	257	334	678	611	622
2013-2014	224	242	122	154	78	134	170	195	482	529
2012-2013	222	260	108	127	79	106	199	219	**444**	**374**
2011-2012	215	229	77	108	110	137	189	257	329	351
2010-2011	118	166	66	74	165	212	229	263	143	171
2009-2010	59	87	26	52	125	155	49	142	73	117

Consumer Durable Sector: The top ten stocks of this sector in terms of their market capitalization in December 2020 are: (i) Titan Company, (ii) Havells India, (iii) Voltas, (iv) Crompton Greaves Consumer Electricals, (v) Bata India, (vi) Dixon Technologies India, (vii) Whirlpool of India, (viii) Rajesh Exports, (ix) Relaxo Footwears, and (x) Blue Star. Table 7-6 presents the historical prices of the five stocks from the consumer durable sector. It is observed that except for six records, the stocks in the consumer durable sector exhibit higher prices in May (i.e., the month of the highest seasonality of the capital goods sector). Lower prices are observed in January (i.e., the month of the lowest seasonality of the capital goods sector). The records that do not match with the seasonality pattern of the capital goods sector are Titan Company in 2015-2016, Havells in 2019-2020, Voltas in 2015-2016, Whirlpool India in 2018-2019, and 2015-2016, and Bata India in 2012-2013.

TABLE 7-7. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE FMCG SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	HUL		ITC		Nestle India		Britannia Ind.		Dabur	
	Feb	Jul	Feb	Jul	Feb	Jul	Feb	Jul	Feb	Jul
2020	**2298**	**2117**	172	191	**16302**	**15946**	2690	3724	450	475
2019	1708	1880	297	246	10994	12850	**3082**	**2701**	409	449
2018	1336	1781	256	319	8502	11577	2486	3367	327	479
2017	910	1217	280	282	6681	7101	1691	2106	277	315
2016	870	917	218	260	5769	6560	1338	1727	249	291
2015	862	873	217	217	**6942**	**5976**	1079	3142	265	284
2014	604	741	235	237	5008	5798	422	627	180	232
2013	466	629	205	206	4589	4901	260	351	137	163
2012	410	518	151	178	4603	4657	251	296	106	124
2011	285	321	121	133	3669	4388	185	243	96	111
2010	239	264	88	108	2676	3060	160	205	79	105

FMCG Sector: Based on their market capitalization values, the top ten stocks of this sector in December 2020 are: (i) Hindustan Unilever (HUL), (ii) ITC, (iii) Nestle India, (iv) Britannia Industries, (v) Tata Consumer Products, (vi) Dabur India, (vii) Godrej Consumer Products, (viii) Colgate Palmolive India, (ix) GlaxoSmithKline Consumer, and (x) Marico. In Table 7-7, we present the historical prices of five stocks from the FMCG sector. The observations in Table 7-7 show that, except for four records, the stocks exhibit higher prices in July (i.e., the month of the highest seasonality of the FMCG sector). In general, the price is in February (i.e., the month of the lowest seasonality of the FMCG sector). The records that do not follow the FMCG sector's seasonality pattern are Hindustan Unilever (HUL) in 2020, Nestle India in 2020 and 2015, and Britannia Industries in 2019.

Healthcare Sector: The stocks, in this sector, with the largest market capitalization values in December 2020 are: (i) Dr. Reddy's Labs, (ii) Sun Pharmaceuticals, (iii) Divi's Labs, (iv) Cipla, (v) Aurobindo Pharma, (vi) Apollo Hospitals Enterprise, (vii) Lupin, (viii) Biocon, (ix) Ipca Labs, and (x) Torrent Pharmaceuticals. Table 7-8 exhibits the historical prices of five stocks in the healthcare sector. It is observed that except for six cases, the FMCG stocks show higher prices in October (i.e., the month of the highest seasonality for the healthcare sector). Lower prices are observed in March (i.e., the month of the lowest seasonality for the healthcare sector). The six records that do not follow the healthcare sector's seasonality pattern are Dr.

Reddy's Labs in 2017, Sun Pharmaceuticals in 2018, Divi's Labs in 2010, Cipla in 2019, and Aurobindo Pharma in 2019 and 2011.

TABLE 7-8. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE HEALTHCARE SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Dr. Reddy's Labs		Sun Pharma		Divi's Labs		Cipla		Aurobindo Pharma	
	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct
2020	3937	4830	465	512	2333	3608	590	745	626	868
2019	2677	2913	450	457	1743	1787	**564**	**467**	**819**	**450**
2018	2111	2713	**528**	**492**	1197	1437	540	607	641	810
2017	**2605**	**2286**	540	643	628	1015	557	600	606	695
2016	3094	3199	710	811	1050	1173	537	567	740	761
2015	3308	4108	731	939	862	1147	629	644	642	811
2014	2705	3602	632	840	690	865	398	635	287	546
2013	2024	2489	476	572	537	576	406	413	95	147
2012	1760	1823	301	355	415	587	312	414	65	94
2011	1575	1672	233	262	353	374	309	328	**97**	**45**
2010	1262	1785	157	224	**337**	**309**	342	344	95	125

IT Sector: In terms of their market capitalization, the top ten stocks in this sector in December 2020 are: (i) Tata Consultancy Services (TCS), (ii) Infosys, (iii) Wipro, (iv) Tech Mahindra, (v) Info Edge India, (vi) HCL, (vii) Larsen & Toubro Infotech, (viii) Mphasis, (x) Mindtree, and (x) Coforge. Table 7-9 depicts the historical prices of five stocks from the IT sector. Table 7-2 showed that the months of the highest and the lowest seasonality for the IT sector are January and June, respectively. Hence, we compare the prices in June in one year and January in the next year for the IT sector stocks. For example, we compare a stock price in June 2015 with its price in January 2016. It is observed from Table 7-9 that, except for three cases, the stocks in the IT sector follow the seasonality pattern of the overall sector. The records that do not follow the sector's seasonality pattern are TCS in 2019-2020 and 2015-2016 and Wipro in 2019-2020.

TABLE 7-9. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE IT SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	TCS		Infosys		Wipro		Tech Mahindra		Info Edge	
	Jun	Jan	Jun	Jan	Jun	Jan	Jun	Jan	Jun	Jan
2019-2020	**2205**	**2001**	732	732	**265**	**221**	637	744	2246	2599
2018-2019	1941	1985	683	734	208	278	680	832	1369	1785
2017-2018	1247	3038	505	587	217	293	385	613	991	1276
2016-2017	1309	1358	506	537	183	204	487	500	819	852
2015-2016	**1255**	**1091**	539	542	194	213	530	415	832	705
2014-2015	1289	1332	421	567	203	245	538	709	689	816
2013-2014	908	1136	371	478	164	224	312	468	317	632
2012-2013	620	757	278	363	127	156	178	262	322	348
2011-2012	567	611	347	360	146	162	150	194	372	323
2010-2011	421	557	348	375	154	164	175	161	228	288
2009-2010	263	381	258	325	110	152	212	223	151	208

Large Cap: The stocks in this sector which had the most significant market capitalization values in December 2020 are: (i) Reliance Industries, (ii) Tata Consultancy Services (TCS), (iii) HDFC Bank, (iv) Hindustan Unilever (HUL), (v) Infosys, (vi) HDFC, (vii) Kotak Mahindra Bank, (viii) ICICI Bank, (ix) ITC, and (x) Asian Paints. Table 7-10 presents the historical prices of five stocks in the large-cap sector. We have already observed in Table 7-2 that the large-cap exhibits the highest and the lowest seasonality during October and March, respectively. Hence, we compare the stock prices in March and October for a given year to understand the seasonal behavior of the stock prices. It is observed from Table 7-10 that five cases do not follow the seasonal pattern of the large-cap sector. The cases that do not follow the sector's seasonal pattern are Reliance Industries in 2011, TCS in 2016 and 2015, and ICICI Bank in 2015 and 2011.

TABLE 7-10. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE LARGE CAP SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Reliance Ind.		TCS		HDFC Bank		Kotak Bank		ICICI Bank	
	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct
2020	1453	1930	2015	2679	1002	1441	1357	1902	380	473
2019	1392	1551	1255	2053	1157	1274	1387	1614	408	512
2018	963	1168	1766	1971	972	1059	1210	1232	284	355
2017	697	882	1136	2706	771	926	901	1000	253	308
2016	491	496	**1265**	**1138**	567	600	716	756	215	241
2015	874	937	**1233**	**1183**	495	1076	667	692	**302**	**249**
2014	468	473	1095	1322	359	478	389	601	226	319
2013	394	427	688	1002	312	330	326	378	190	194
2012	371	397	603	656	270	647	292	334	157	200
2011	**490**	**389**	557	582	229	244	215	232	**202**	**169**
2010	537	547	383	461	199	229	184	238	173	208

Metal Sector: The stocks in this sector that had the largest market capitalization values in December 2020 are: (i) Tata Steel, (ii) JSW Steel, (iii) Hindalco Industries, (iv) Coal India, (v) Jindal Steel and Power, (vi) National Mineral Development Corporation, (vii) Hindustan Zinc, (viii) Steel Authority of India, (ix) APL Apollo Tubes, and (x) National Aluminum Company. Table 7-11 presents the historical prices of five stocks in the metal sector. Table 7-2 showed that the metal sector exhibits its highest and lowest seasonality during August and December, respectively. Hence, we compare the stock prices in August and December. It is observed from Table 7-11 that eight cases do not follow the seasonal pattern of the metal sector. The stock records that do not follow the seasonal pattern of the sector are Tata Steel in 2018 and 2014, JSW Steel in 2018 and 2014, Hindalco Industries in 2018, Coal India in 2018 and 2016, and Jindal Steel in 2018.

TABLE 7-11. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE METAL SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Tata Steel		**JSW Steel**		**Hindalco Ind.**		**Coal India**		**Jindal Steel**	
	Aug	Dec	Aug	Dec	Aug	Dec	Aug	Dec	Aug	Dec
2020	413	635	278	366	175	245	116	143	187	273
2019	360	439	230	251	191	216	200	211	104	176
2018	**580**	**476**	**380**	**275**	**229**	**226**	**267**	**240**	**196**	**157**
2017	620	705	248	290	241	274	271	298	135	266
2016	357	441	173	198	153	190	**323**	**310**	76	80
2015	202	238	89	106	71	85	327	329	61	64
2014	**438**	**396**	**116**	**98**	157	159	341	361	173	194
2013	259	339	73	92	112	123	250	290	235	250
2012	382	386	76	88	121	131	352	355	428	448
2011	396	430	59	70	131	147	327	333	505	542
2010	498	608	118	134	197	229	333	334	707	713

Mid-Cap Sector: The top ten stocks in terms of their market capitalization under this sector in December 2020 are: (i) Berger Paints, (ii) Info Edge, (iii) Larsen & Toubro Infotech, (iv) Biocon, (v) Tata Consumer Products, (vi) Adani Enterprises, (vii) Muthoot Finance, (viii) Adani Transmission, and (x) Torrent Pharmaceutical. Table 7-12 presents the historical prices of five stocks from the top ten list of mid-cap sector stocks. It is observed that except for four cases, the stock prices are higher during December (i.e., the month of the highest seasonality for the mid-cap sector). On the other hand, lower prices are observed during March (i.e., the month of the lowest seasonality for the mid-cap sector). The records that do not follow the seasonality pattern of the mid-cap sector are Info Edge in 2011, Torrent Pharmaceuticals in 2016, Biocon in 2011, and Adani Enterprise in 2015.

TABLE 7-12. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE MID CAP SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Berger Paints		Info Edge		Torrent Pharma		Biocon		Adani Enterprise	
	Mar	Dec	Mar	Dec	Mar	Dec	Mar	Dec	Mar	Dec
2020	508	684	2553	4636	2345	2758	353	455	141	462
2019	318	560	1938	2848	1792	1847	270	294	127	230
2018	286	330	1240	1746	1415	1771	297	324	91	137
2017	261	271	829	1374	1410	1418	184	307	61	111
2016	184	211	751	824	**1425**	**1315**	98	168	48	71
2015	149	190	769	793	1168	1454	75	86	**107**	**41**
2014	86	164	582	845	577	1131	70	71	66	100
2013	69	72	352	494	346	474	48	77	34	42
2012	43	62	344	357	330	363	40	48	39	43
2011	33	35	**357**	**313**	265	277	**61**	**44**	66	99
2010	22	37	238	272	267	282	50	58	89	103

Oil & Gas Sector: The top ten stocks in terms of their market capitalization under this sector in December 2020 are: (i) Reliance Industries, (ii) Bharat Petroleum Corporation, (iii) Oil and Natural Gas Corporation (ONGC), (iv) Indian Oil Corporation, (v) GAIL, (vi) Petronet LNG, (vii) Indraprastha Gas, (viii) Hindustan Petroleum Corporation, (ix) Adani Gas, and (x) Gujarat State Petronet. Table 7-13 presents the historical prices of five stocks in the Oil and Gas sector. It is observed that except for eight cases, the stock prices are higher during October (i.e., the month of the highest seasonality for the oil and gas sector). In general, the price is low in March (i.e., the month of the lowest seasonality for the oil and gas sector). The records that do not follow the seasonality pattern of the oil and gas sector are Reliance Industries in 2011, Bharat Petroleum in 2018, ONGC in 2019, 2018 and 2011, Indian Oil in 2019 and 2018, and 2018, and GAIL in 2019.

TABLE 7-13. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE OIL AND GAS SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Reliance Ind.		Bharat Petroleum		ONGC		Indian Oil		GAIL	
	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct
2020	1453	1930	370	399	78	100	84	95	95	125
2019	1392	1551	379	511	**169**	**132**	**157**	**131**	**178**	**126**
2018	963	1168	**388**	**363**	**181**	**153**	**162**	**133**	163	172
2017	697	882	480	503	186	194	193	197	159	176
2016	491	496	326	429	145	192	108	153	101	119
2015	874	937	255	301	203	251	90	105	101	103
2014	468	473	151	249	217	253	66	91	104	137
2013	394	427	115	138	199	208	50	75	99	99
2012	371	397	111	116	178	179	66	66	93	99
2011	**490**	**389**	104	104	**206**	**177**	66	85	110	134
2010	537	547	86	113	176	208	74	87	121	138

Power Sector: The top ten stocks in terms of their market capitalization under this sector in December 2020 are: (i) Power Grid Corporation, (ii) National Thermal Power Corporation (NTPC), (iii) Adani Transmission, (iv) Tata Power, (v) NHPC, (vi) Adani Power, (vii) Torrent Power, (viii) JSW Energy, (ix) SJVN, (x) Calcutta Electric Supply Corporation (CESC). Table 7-14 presents the historical prices of five stocks of the realty sector from the top ten list of this sector. It is observed that except for eleven cases, the stock prices are higher during June (i.e., the month of the highest seasonality for the power sector). In general, the price of the stocks is low in February (i.e., the month of the lowest seasonality for the power sector). The eleven cases that do not follow the seasonality pattern of the realty sector are NTPC in 2019, 2015, and 2011, Torrent Power in 2013 and 2012, JSW Energy in 2018, 2015, and 2013, SJVN in 2018, 2015, and 2012.

TABLE 7-14. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE POWER SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	Power Grid		NTPC		Torrent Power		JSW Energy		SJVN	
	Feb	Jun	Feb	Jun	Feb	Jun	Feb	Jun	Feb	Jun
2020	159	179	84	87	279	325	43	46	21	23
2019	198	211	**135**	**126**	257	305	68	73	24	25
2018	182	194	129	141	229	240	**73**	**66**	**33**	**27**
2017	197	223	138	138	232	182	63	71	32	34
2016	139	175	107	132	170	230	70	83	27	29
2015	141	145	**122**	**112**	146	163	**118**	**84**	**26**	**24**
2014	105	133	100	121	94	140	60	75	21	25
2013	100	106	118	123	**140**	**75**	**55**	**42**	19	19
2012	108	119	136	130	**202**	**159**	50	61	**22**	**20**
2011	102	105	**161**	**147**	240	250	68	72	22	23
2010	107	108	163	173	346	289	112	127	24	24

Realty Sector: The top ten stocks in terms of their market capitalization under this sector in December 2020 are: (i) DLF, (ii) National Building Construction Corporation (NBCC), (iii) Oberoi Realty, (iv) Godrej Properties, (v) Prestige Estate Projects, (vi) Phoenix Mills, (vii) Indiabulls Real Estate, (viii) Housing Development Infrastructure, (ix) Sobha, (x) Prestige Estate, and (xii) Unitech. Table 7-15 presents the historical prices of five stocks of the realty sector from the top ten list of this sector. From Table 7-15, it is observed that except for ten cases, the stock prices are higher during July (i.e., the month of the highest seasonality for the realty sector). In general, the price is low in March (i.e., the month of the lowest seasonality for the realty sector). The ten cases that do not follow the seasonality pattern of the realty sector are DLF in 2013, Shobha in 2018, 2015, and 2013, Phoenix Mills in 2013, Godrej Properties in 2013, Prestige Estate in 2019, 2018, 2013, and 2011.

TABLE 7-15. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE REALTY SECTOR ON THE BSE DURING 2010 - 2020

Year	Stocks									
	DLF		Sobha		Phoenix Mills		Godrej Properties		Prestige Estate	
	Mar	Jul	Mar	Jul	Mar	Jul	Mar	Jul	Mar	Jul
2020	128	138	201	262	549	645	677	913	174	247
2019	173	178	482	537	601	693	830	901	**259**	**195**
2018	220	223	**549**	**460**	626	629	805	687	**237**	**304**
2017	186	193	373	394	412	520	464	579	236	274
2016	129	155	292	296	319	400	341	362	169	186
2015	114	167	**391**	**295**	338	363	242	308	202	264
2014	140	178	371	453	248	317	213	244	168	224
2013	**238**	**130**	**349**	**248**	**271**	**189**	**288**	**196**	**168**	**115**
2012	181	196	3303	333	175	208	241	276	110	135
2011	222	242	253	286	202	212	319	321	**146**	**99**
2010	311	377	315	328	212	237	233	340	92	96

Small Cap Sector: The top ten stocks in terms of their market capitalization under this sector in December 2020 are: (i) Adani Gas, (ii) SRF, (iii) Tata Communications, (iv) Ipca Labs, (v) Varun Beverages, (vi) Trent, (vii) Gujarat Gas, (viii) Syngene International, (ix) Mindtree, and (x) Pfizer. Table 7-16 presents the historical prices of five stocks of the small-cap sector listed in BSE. It is observed that 12 cases do not follow the seasonality pattern of the small-cap sector. In Table 7-2, we observed that the highest and the lowest seasonality exhibited by the small-cap sector are during October and March, respectively. In Table 7-16, the stock price records that do not follow the seasonality pattern of the small-cap sector are SRF in 2018, 2012, and 2011, Tata Communications in 2019, 2018, and 2017, Ipca Labs in 2014 and 2011, Mindtree in 2019, 2018, and 2015, and Trent in 2018.

TABLE 7-16. PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE SMALL CAP SECTOR LISTED ON THE BSE DURING 2010 - 2020

Year	Stocks									
	SRF		Tata Comm.		Ipca Labs		Mindtree		Trent	
	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct	Mar	Oct
2020	3714	5135	432	1059	1619	2197	915	1417	505	723
2019	2567	3201	**562**	**442**	954	1145	**981**	**700**	357	520
2018	**2393**	**2161**	**622**	**539**	751	796	**1085**	**876**	**352**	**333**
2017	1780	1829	**723**	**687**	595	597	486	539	251	342
2016	1362	1577	413	658	496	578	477	678	169	180
2015	961	1197	424	432	657	789	**1191**	**719**	114	160
2014	387	890	283	458	**839**	**684**	355	599	98	146
2013	170	206	228	287	509	659	200	348	108	126
2012	**235**	**208**	237	239	360	457	145	180	92	121
2011	**334**	**295**	202	212	**304**	**258**	92	102	100	101
2010	217	326	271	306	263	316	125	144	68	90

Telecommunication Sector: The top ten stocks in terms of their market capitalization in this sector in December 2020 are: (i) Bharti Airtel, (ii) Bharti Infratel, (iii) Tata Communications, (iv) Vodafone Idea, (v) Honeywell Automation, (vi) ITI, (vii) Sterlite Technologies, (viii) Finolex Cables, (ix) HFCL, and (x) Tata Teleservices. Table 7-17 presents the historical prices of five stocks in the telecommunication sector. In Table 7-2, we observed that the telecommunication sector exhibits the highest and the lowest seasonality in July and March, respectively. However, it is observed that eleven records in Table 7-17 do not follow this seasonality pattern of the telecommunication sector. The cases that do not follow the seasonality pattern of the sector are Bharti Airtel in 2015 and 2012, Finolex Cables in 2019 and 2018, Tata Communications in 2019, 2018 and 2017, Vodafone Idea in 2019, 2015, and 2012, and Honeywell Automation in 2012.

TABLE 7-17. THE PRICE MOVEMENTS OF THE MAJOR STOCKS OF THE TELECOMMUNICATION SECTOR LISTED ON THE BSE

Year	Stocks									
	Bharti Airtel		Finolex Cables		Tata Comm.		Vodafone Idea		Honeywell Auto	
	Mar	Jul	Mar	Jul	Mar	Jul	Mar	Jul	Mar	Jul
2020	440	514	255	278	432	871	4	10	28644	32577
2019	320	347	**441**	**367**	**562**	**430**	**16**	**5**	23975	24860
2018	343	351	**698**	**547**	**622**	**539**	30	42	18912	22534
2017	325	393	524	538	**723**	**653**	52	55	10961	13136
2016	332	334	284	444	413	526	56	71	9235	9793
2015	**399**	**325**	244	270	424	438	**182**	**94**	7367	8848
2014	301	339	136	218	283	376	81	98	3118	5163
2013	293	317	52	56	228	265	80	97	2441	2492
2012	**282**	**226**	32	37	237	243	**49**	**45**	**2736**	**2462**
2011	347	371	40	53	202	236	41	60	2483	2554
2010	274	300	58	58	271	332	37	43	2366	2532

TABLE 7-18. PERCENTAGE OF CASES THAT MATCHED WITH THE SEASONALITY PATTERNS OF THE RESPECTIVE SECTORS

Sector	No. of Matched Cases	% of Matched Cases
Auto	51	92.73
Banking	50	90.91
Capital Goods	46	83.64
Consumer Durable	49	89.09
FMCG	51	92.73
Healthcare	49	89.09
IT	52	94.55
Large Cap	50	90.91
Metal	47	85.45
Mid Cap	51	92.73
Oil & Gas	47	85.45
Power	44	80.00
Realty	45	81.82
Small Cap	43	78.18
Telecommunication	44	80.00
Overall	719	87.15

Table 7-18 presents the number of cases under various sectors that match the respective sectors' seasonality patterns. The corresponding percentage figures are also mentioned in a separate column. The sector with the maximum number of cases matching the sector's seasonality pattern is the IT sector. For the IT sector, 52 cases match, resulting in a matching percentage of 94.55. For both the power and the telecommunication sectors, 44 cases match, resulting in a matching percentage of 80. The smallest matching percentage of 78.18 is found for the small-cap sector. For all the sectors together, 719 cases match out of 855 cases, resulting in an overall match percentage of 87.15. The percentage of cases that match the respective sectors' seasonality patterns is quite satisfactory for all the sectors. The results demonstrate the validity of the hypothesis proposed in this work.

Forecasting of Sectoral Index

In this section, we present four different approaches for forecasting the sectoral indices of the Indian economy. Although we applied our techniques to all the fifteen sectors we have studied, we present the results for the Indian IT sector in detail as an illustration. For the remaining fourteen sectors, we have given the summary results later in this section.

For each forecasting method, we use the IT sector index time series from January 2010 to December 2019 for building the forecasting model. After the model is built, it is used for forecasting the monthly index values of the IT sector for the year 2020. For every month of 2020, we compare the *predicted index* with its *actual value* and compute the error in forecasting. We use two metrics to evaluate the performance of the forecasting methods: (i) the ratio of the *root mean square error* (RMSE) to the mean of the target variable and (ii) the *mean absolute percentage error* (MAPE).

The RMSE measure is computed using (6), where y_i's are the actual values of the time series index, $\hat{y}_i$'s are the corresponding forecasted values, and N is the total number of observations. Since different time series will have different values of observations, we compute the ratio of the RMSE to the mean of the actual values of the time series so that the error is expressed as a fraction (or percentage) of the mean value.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{6}$$

Hence, the first metric that we use for comparing our forecasting models is given by (7):

$$\frac{RMSE}{Mean} = \frac{\sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}}{\sum_{1=}^{n} y_i / N} \tag{7}$$

The second metric for evaluating the forecasting models that we have used is given by (8):

$$MAPE = \frac{1}{n}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{8}$$

The function *rmse* and *mape* defined in the R package *Metrics* have been used for computing the RMSE and the MAPE measures, respectively. Both the functions require two parameters for their execution – the first parameter is the R object representing the *actual* time series index values, and the second parameter is the R object representing their corresponding *forecasted* values.

We forecast the monthly index values of the fifteen sectors of the Indian economy based on the historical monthly index values of the sectors from January 2010 to December 2019. For this purpose, we propose four methods of forecasting. The methods are as follows:

(i) **Method I:** it is an *exponential smoothing* method on the *level*, *trend*, and *seasonality* of the time series using the *Holt-Winters method* of forecasting with a *forecast horizon* of twelve months.
(ii) **Method II:** it is an *exponential smoothing* method on the *level*, *trend*, and *seasonality* of the time series using the *Holt-Winters* method of forecasting with a *forecast horizon* of one month.
(iii) **Method III:** it is based on an *auto-regressive integrated moving average* (ARIMA) method with a *forecast horizon* of twelve months.
(iv) **Method IV:** it is based on an *autoregressive integrated moving average* (ARIMA) method with a *forecast horizon* of one month.

In the following, we describe the four methods of forecasting in detail.

Method 1: In this approach, we forecast for every month of the year 2020 based on the time series index values from January 2010 to December 2019. The forecasting is done at the end of December 2019 for all the months of the year 2020. We use the *HoltWinters* function in the R language, which is defined in the *forecast* library. The *HoltWinters* function uses an *exponential smoothing* technique. We need to specify two parameters – *beta* and *gamma* - to ensure that the forecasting algorithm considers the trend and the seasonality of the series in computing the forecasted values (Coghlan, 2015). Since the IT sector time series from January 2010 to December 2019 exhibits both trend and seasonality, we set the parameters *beta* and *gamma* as TRUE in the *HoltWinters* model. While the parameter *beta* set as TRUE ensures that the forecasting algorithm takes into account the trend, the parameter *gamma* allows the algorithm to consider the seasonality behavior of the time series. The *forecast horizon parameter* (*h*) of the *HoltWinters* function is assigned a value of 12 since we are interested in computing the forecasted values for all months of the year 2020. The results obtained in the forecasting method are presented in Table 7-19. As mentioned earlier in this section, the functions *rmse* and *mape* in the R language are used for computing the RMSE and the MAPE for the forecasting model. Figure 7-2 depicts the plot of the *IT sector's actual index values* and the corresponding forecasted values using *Method I* for all the months of the year 2020. The RMSE value for this method of forecasting is 9948. Since the mean of the *actual* index values for the IT sector in the year 2020 is 17378, the ratio of the RMSE to the mean of the target variable for this method is found to be 57.25%. The MAPE for this method is computed to be 58.14. The high values of the ratio of the RMSE to the mean and the MAPE indicate that *Method I* of forecasting lacks accuracy. Table 7-19 shows the errors for March, April. June and July are high.

TABLE 7-19. FORECASTED INDEX OF THE INDIAN IT SECTOR FOR THE YEAR 2020 USING *METHOD 1* OF FORECASTING

Month	Actual Value	Forecasted Value	Error (%)	RMSE/Mean	MAPE
Jan	15871	20330	28.10	57.25	58.14
Feb	14987	22299	48.79		
Mar	12843	23653	84.17		
Apr	14235	25647	80.17		
May	14067	26272	86.76		
Jun	14887	27576	85.24		
Jul	18251	29221	60.11		
Aug	18055	30793	70.55		
Sep	19980	30636	53.33		
Oct	21059	29657	40.83		
Nov	21635	28447	31.49		
Dec	22667	29047	28.14		

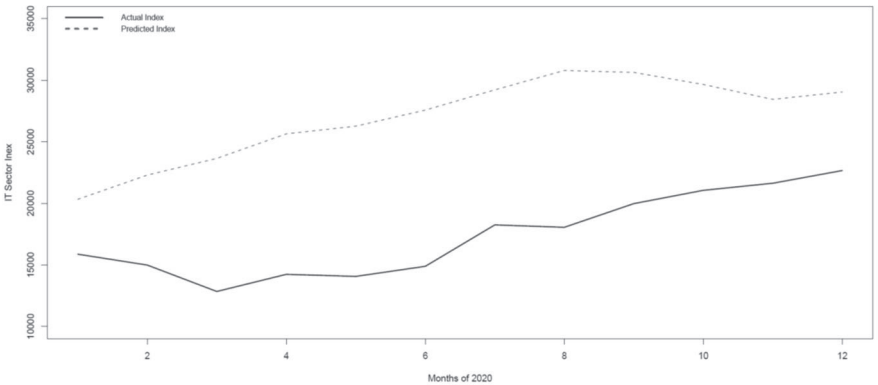

Figure 7-2. The actual and the predicted Indian IT sector index using *Method I* (Period: January 2020 – December 2020)

TABLE 7-20. FORECASTED INDEX OF THE INDIAN IT SECTOR FOR THE YEAR 2020 USING *METHOD II* OF FORECASTING

Month	Actual Value	Forecasted Value	Error (%)	RMSE/Mean	MAPE
Jan	15871	20330	28.10	12.79	10.22
Feb	14987	15413	2.84		
Mar	12843	14486	12.79		
Apr	14235	11105	21.99		
May	14067	14812	-5.30		
Jun	14887	14554	2.24		
Jul	18251	14596	20.03		
Aug	18055	20629	14.26		
Sep	19980	19841	0.70		
Oct	21059	20186	4.15		
Nov	21635	21614	0.10		
Dec	22667	24962	10.12		

Method II: In this method, we forecast the index for each month of the year 2020, based on the time series data from January 2010 till the end of the previous month for which the forecast is being made. For example, to forecast the index for April 2020, the time series values from January 2010 to March 2020 are considered in building the forecast model. We use the same *HoltWinters* function in the R language that was used in *Method I* of forecasting. The parameters *beta* and *gamma* are set as TRUE so that the forecasting algorithm takes into account both the *trend* and the *seasonality* components in the time series. However, unlike in *Method I*, in this method, the *forecast horizon* (*h*) parameter value is set equal to 1 so that the index for the immediately following month is forecasted. The forecasting model is rebuilt every time before a forecast is made since the training data set changes before each iteration of model building. Table 7-20 presents the results obtained using this method. Figure 7-3 depicts the plot of the actual and the forecasted index values for each month of the year 2020 using *Method II* of forecasting. The RMSE value of this model is found to be 2222, and the ratio of the RMSE to the mean is 12.79%. The MAPE value of the model is 10.22%. Hence, *Method II* of forecasting exhibits a significant improvement over *Method I*.

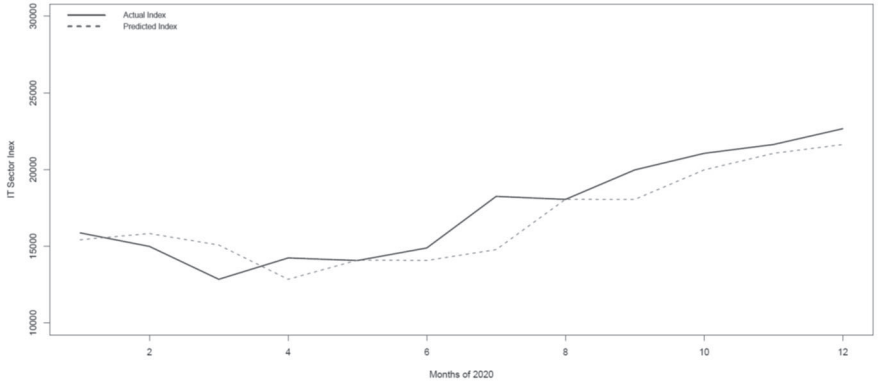

Figure 7-3. The actual and the predicted Indian IT sector index using *Method II* (Period: January 2020 – December 2020)

Method III: In this method, we use an *autoregressive integrated moving average* (ARIMA) - based approach to forecasting (Coghlan, 2015; Tsay, 2014; Hyndman & Athanasopoulos, 2018). For building the ARIMA model, we use the Indian IT sector S&P BSE index from January 2010 to December 2019. Based on this time series data, we compute the three parameters of the *autoregressive moving average* (ARMA) model. The three parameters of an ARMA model are (i) the *autoregression parameter* (p), (ii) the *difference parameter* (d), and (iii) the *moving average parameter* (q). We first use the *auto.arima* function defined in the *forecast* library in the R language to determine the values of the parameters p, d, and q. The returned values of the three parameters by the *auto.arima* function are cross-verified by plotting the *partial autocorrelation function* (PACF), the unit root tests, *augmented dickey fuller* (ADF) test, and the *autocorrelation function* (ACF), respectively (Coghlan, 2015; Hyndman & Athanasopoulos, 2018; Tsay, 2014).

The output of the *auto.arima* function on the IT index time series from January 2010 to December 2019 is presented in Table 7-21.

TABLE 7-21. THE OUTPUT OF THE *AUTO.ARIMA* FUNCTION FOR THE INDIAN IT SECTOR INDEX SERIES (JANUARY 2010 – DECEMBER 2019)

Series: IT		
ARIMA (0, 1, 1) with drift		
Coefficients	**ma1**	**drift**
	-0.1415	87.5981
Standard Error	0.0934	40.0331
Sigma-square estimate: 262444 **Log likelihood:** -910.28		
AIC = 1826.57 **AICc** = 1826.78 **BIC** = 1834.91		

From Table 7-21, it is evident that the *auto.arima* function computes the values for the three parameters p, d, and q as 0, 1, and 1, respectively. We will first explain the model in detail. Then we verify the values of the parameters using the ACF and the PACF plots and the unit root tests. The model ARIMA (0, 1, 1) with drift is expressed in (9).

$$y_t = c - 0.1415\varepsilon_{t-1} + \varepsilon_t \tag{9}$$

In (9), ε_{t-1} denotes the error in forecasting at lag-1, c is the constant given by the value of the parameter *drift*, and ε_t is the error at the current instant, which is a *white noise* with a standard deviation $\sigma = \sqrt{262444} = 512.29$. While estimating an ARIMA model's parameters, R uses the *maximum likelihood estimation* (MLE) (Hyndman & Athanasopoulos, 2018). The method computes the parameters' values that maximize the probability of obtaining the observations in the given series. In practice, R reports the logarithm of the maximum likelihood of the data, that is, the logarithm of the probability of the observed data fitting into the built model. In other words, R computes the values of p, d, and q that maximize the values of the logarithm of the likelihood of those estimates.

The metrics AIC, AICc, and BIC in Table 7-21 are information criteria of the ARIMA model. AIC is computed using (10).

$$AIC = -2\log(L) + 2(p + q + k + 1) \tag{10}$$

In (10), L denotes the likelihood of the data fitting into the model, $k = 1$, if $c \neq 0$, and $k = 0$, if $c = 0$. Here, c refers to the constant term in the ARIMA model equation. The second term in the RHS of (10) refers to the total number of parameters in the model that includes the variance of the residuals (σ^2).

AIC is corrected in ARIMA models, and the metric corrected AIC (AICc) is computed using (11). In (11), T denotes the sample size, i.e., the number of observations in the series.

$$AICc = AIC + \frac{2(p+q+k+1)(p+q+k+2)}{T-p-q-k-2} \tag{11}$$

The metric *Bayesian Information Criterion* (BIC) is given by (12).

$$BIC = AIC + [\log(T) - 2](p + q + k + 1) \tag{12}$$

Among a set of candidate models, the model that yields the lowest value of the information criteria is chosen as the best-fitting model. For ARIMA applications, the model that minimizes AICc is determined to be the optimum one.

To verify the values of the parameters of the ARIMA model, we plot the ACF and the PACF plots and carry out unit root tests.

First, we verify the value of the parameter d by carrying out a couple of *unit-root tests* – (i) the *augmented Dickey Fuller* (ADF) test and (ii) the *Kwiatkowski-Phillips-Schmidt-Shin* (KPSS) test (Dickey & Fuller, 1979; Kwiatkowski et al., 1992). The null hypothesis of the ADF test assumes that a unit root is present in a time series sample. If the p-value of the null hypothesis is more than 0.05 at the 5% level of significance, then the null hypothesis cannot be rejected. In other words, a p-value greater than 0.05 indicates the existence of a *unit root*, and hence a *non-stationary* time series. On the other hand, the null hypothesis of the KPSS test assumes that the series is stationary. If the p-value of the null hypothesis is greater than 0.05, then the null hypothesis about the stationarity of the series cannot be rejected.

We use the *adf.test* function defined in the *tseries* library in the R language. The *adf.test* applied on the Indian IT sector index series from January 2010 to December 2019 yields a *Dickey-Fuller test statistic* value of -1.9821 with an associated p-value of 0.584. The test indicates that the series is a *non-stationary* series. We compute the *first-order difference* of the series using the *diff* function in the *tseries* library. The *first-order difference* series applied with the *adf* function yields the *Dickey-Fuller test statistic* value of -4.9392, with an associated p-value of 0.01. Hence, the *first-order difference* of the IT index time series is *stationary*. Hence, the value of the parameter $d = 1$ is verified.

We also carry out the KPSS test on the IT index time series data from January 2010 to December 2019. The *kpss.test* function defined in the *tseries* library defined in the R language is used to carry out the KPSS test.

When the *kpss.test* function is called with the IT index series from January 2010 to December 2019 for checking the *level stationarity*, the *KPSS Level statistics* value is found to be 2.22 with an associated *p*-value of 0.01. Hence, the null hypothesis that assumes stationarity of the level in the series is rejected. However, when the test is applied to the *first-order difference* of the series, the *KPSS Level statistic* value turns out to be 0.08 with an associated *p*-value of 0.1. The null hypothesis is not rejected in this case, and the *first-order difference* series is assumed to be *stationary*. Hence, the KPSS test also yields the value of the parameter $d = 1$.

It is observed that the computed value of the parameter d by the *auto.arima* function is also supported by both the ADF test and the KPSS test. Figure 7-4 depicts the *first-order difference* series of the IT sector index time series from January 2010 to December 2019. The *first-order difference* time series looks like a stationary time series with the mean and the variance of the series not exhibiting any change over different time lags.

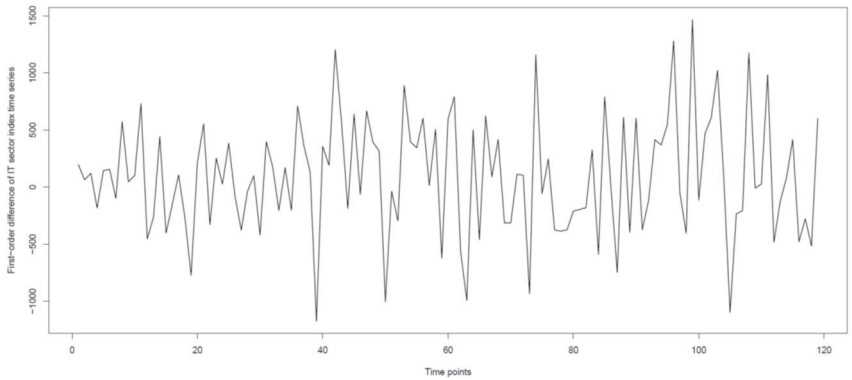

Figure 7-4. The first-order difference of the Indian IT sector index time series (Period: January 2020 – December 2020)

Next, we plot the ACF of the *stationary* time series, i.e., the *first-order difference* series of the IT index series from January 2010 to December 2019. We use the *acf* function in the *forecast* library in the R language for plotting the ACF. We investigate the *autocorrelation* values of the difference for 24 lags (i.e., 2 years) by setting the value of the parameter *lag.max* in the *acf* function to 24. Figure 7-5 shows the ACF plot. It is evident from Figure 7-5 that none of the lags exhibit significant autocorrelation. The little autocorrelation displayed at lag 11 is ignored as we can ignore one exception in 24 observations and remain within 95% confidence. Hence, the ACF plot indicates $q = 0$.

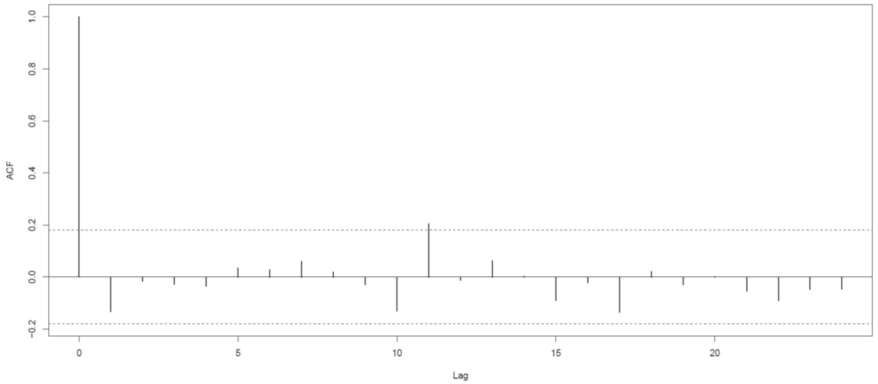

Figure 7-5. The ACF plot of the first-order difference of the Indian IT sector index time series (Period: Jan 2020 – Dec 2020)

To explore the behavior of the *partial autocorrelations* among the values in the *stationary* time series, we plot the PACF for 24 lags. We use the *pacf* function defined in the *forecast* library in the R programming language for studying the partial autocorrelation values. Figure 7-6 depicts the PACF plot, which indicates no significant partial autocorrelation at any of the lags. The PACF plot indicates $p = 0$.

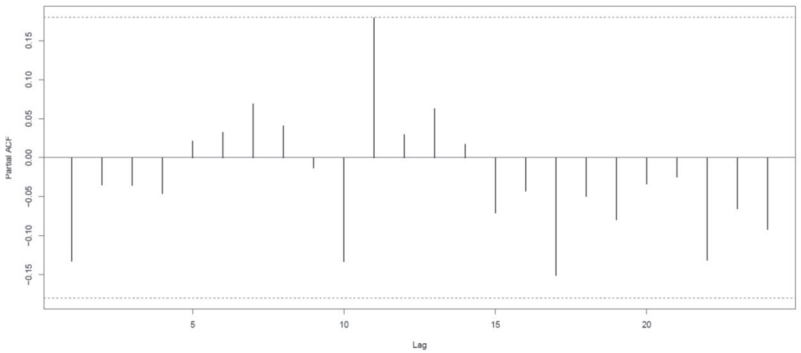

Figure 7-6. The PACF plot of the first-order difference of the Indian IT sector index time series (Period: Jan 2020 – Dec 2020)

The output of the *auto.arima* function and the ACF and PACF plots have yielded conflicting results. While the values of the parameters p and q are 0 and 1, respectively, as computed by the *auto.arima* function, the ACF and PACF plots indicate that values of both the parameters should be 0. We explain this conflict now. The *auto.arima* function returns that ARIMA

model, which yields the lowest *corrected value* of the *Akaike Information Criteria* (AIC) (Hyndman & Athanasopoulos, 2018). The *auto.arima* function first carries out the KPSS test on the given series to determine the parameter *d*. For the IT index series from January 2010 to December 2019, the values of $d = 1$. There is no conflict among the results produced by the *auto.arima* function, the KPSS test, and the ADF test. However, for identifying the optimum values of the parameters *p* and *q*, the *auto.arima* function will check all the candidate models (Hyndman & Athanasopoulos, 2018). These candidate models are: (i) ARIMA(0, 1, 0), (ii) ARIMA (0, 1, 0) with drift, (iii) ARIMA (0, 1, 1), (iv) ARIMA (0, 1, 1) with drift, (v) ARIMA(1, 1, 0), and (vi) ARIMA (1, 1, 0) with drift. We build all these models using the *Arima* function defined in the *forecast* library in the R language. For each model, we note the values of AIC, the AICc, and the *Bayesian Information Criteria* (BIC). While building ARIMA models with drift, we use the parameter *include.drift* = TRUE, in the Arima function. Table 7-22 presents the values of AIC, AICc, and BIC for the six candidate models. It is observed from Table 7-22 that the model that has the lowest value for the metric AICc, ARIMA(0, 1, 1) with drift, is the one that is reported by the *auto.arima* function.

TABLE 7-22. THE SIX CANDIDATE ARIMA MODELS

Models	**Metrics**		
	AIC	**AICc**	**BIC**
ARIMA (0, 1, 0)	1828.28	1828.32	1831.06
ARIMA(0, 1, 0) with drift	1826.81	1826.92	1832.37
ARIMA (0, 1, 1)	1829.05	1829.16	1834.61
ARIMA(0, 1, 1) with drift	1825.57	1826.78	1834.91
ARIMA(1, 1, 0)	1829.03	1829.14	1834.59
ARIMA (1, 1, 0) with drift	1826.70	1826.91	1835.04

Having identified the best-fitted ARIMA model, we study the behavior of the residuals of the in-sample observations and check for the existence of any possible autocorrelations among the residuals. The ACF plot of the residuals for the ARIMA (0, 1, 1) with drift model on the IT index time series from January 2010 to December 2019 is depicted in Figure 7-7. The residuals do not exhibit any significant autocorrelation for any lag, where 24 lags have been plotted. The Ljung Box test is carried out on the residuals using the *Box.test* function defined in the forecast library in the R language with three parameters to the function as (i) the variable storing the IT series index values for the period January 2010 to December 2019, (ii) *lag* = 24,

(iii) *type* = 'Ljung'. The *chi-squared* statistic of the test is found to be 17.22 with an associated *p*-value of 0.82. Hence, the null hypothesis that assumes that there is no significant autocorrelation among the residuals at any lag cannot be rejected.

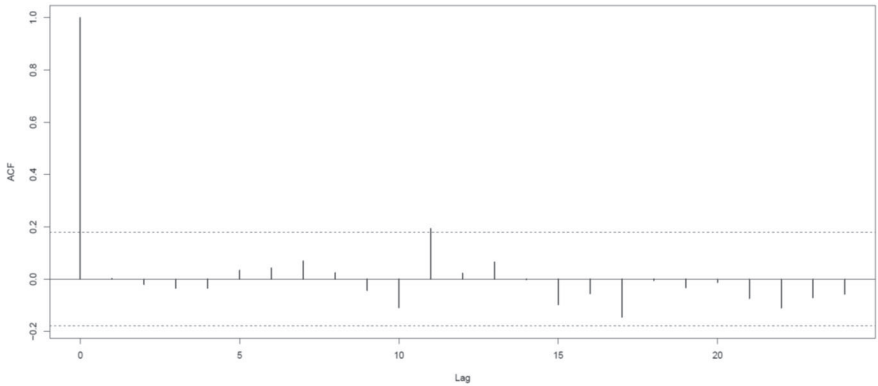

Figure 7-7. The ACF of the residuals of the best-fit ARIMA model on the IT index series (January 2010 – December 2019)

TABLE 7-23. FORECASTED INDEX OF THE INDIAN IT SECTOR FOR THE YEAR 2020 USING *METHOD III* OF FORECASTING

Month	Actual Value	Forecasted Value	Error (%)	RMSE/Mean	MAPE
Jan	15871	15504	2.31	18.71	14.59
Feb	14987	15592	4.04		
Mar	12843	15679	22.08		
Apr	14235	15767	10.76		
May	14067	15855	12.71		
Jun	14887	15942	7.09		
Jul	18251	16030	12.17		
Aug	18055	16117	10.73		
Sep	19980	16205	18.89		
Oct	21059	16293	22.63		
Nov	21635	16380	24.29		
Dec	22667	16468	27.35		

Since the residuals of the model ARIMA (0, 1, 1) with drift do not exhibit any significant autocorrelation at any lag, we now use the model to

make an *out-of-sample forecast* for each of the months in 2020. We use the *forecast.Arima* function defined in the *forecast* library in the R language with a *forecast horizon* (*h*) parameter value of 12 to compute the IT sector index's monthly forecast values in 2020. The results of forecasting using this method are presented in Table 7-23. The functions *rmse* and *mape* in the R language are used for computing the RMSE and the MAPE values. Figure 7-8 exhibits the plot of the *IT sector's actual index values* and the corresponding forecasted values using this model for all the months of the year 2020. The RMSE value for this method of forecasting is 3252. Since the mean of the *actual* index values for the IT sector in the year 2020 is 17378, the ratio of the RMSE to the mean of the target variable for this method is found to be 18.71%. The MAPE for this method is computed to be 14.59.

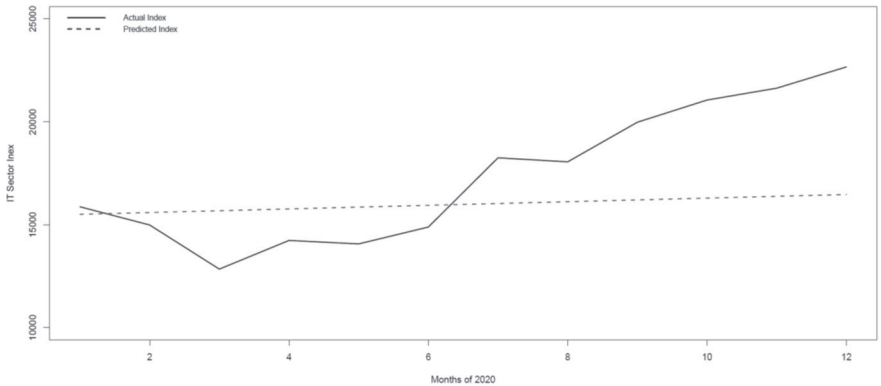

Figure 7-8. The actual and the predicted Indian IT sector index using *Method III* (Period: Jan 2020 – Dec 2020)

Method IV: This approach uses an ARIMA forecasting model with a prediction horizon of 1 month. Similar to *Method II*, in which we used the *HoltWinters* method of forecasting with a prediction horizon of 1 month, in this approach, we use the time series of the IT sector index from January 2010 to one month of 2020 and make the forecast of the immediately following month. The ARIMA model's training data changes with the inclusion of a new observation in each iteration. Hence, after each prediction, we re-evaluate the three parameters (*p*, *d*, and *q*) using the *auto.arima* function. Table 7-24 presents the details of each iteration of model building. In Table 7-24, for each iteration, the training dataset used for building the ARIMA model, the order of the model, the *AICc* value of

the model, and the month for which the forecasting is done using the model are reported.

TABLE 7-24. DESCRIPTION OF EACH ITERATION OF MODEL BUILDING FOR *METHOD IV* OF FORECASTING

Training Dataset	**Model**	**AICc**	**Forecasted Month**
Jan 2010 – Dec 2019	ARIMA (0, 1, 1) with drift	1827	Jan 2020
Jan 2010 – Jan 2020	ARIMA (0, 1, 1) with drift	1842	Feb 2020
Jan 2010 – Feb 2020	ARIMA (0, 1, 1) with drift	1859	Mar 2020
Jan 2010 – Mar 2020	ARIMA (0, 1, 0)	1891	Apr 2020
Jan 2010 – Apr 2020	ARIMA (0, 1, 1) with drift	1911	May 2020
Jan 2010 – May 2019	ARIMA (0, 1, 1) with drift	1925	Jun 2020
Jan 2010 – Jun 2019	ARIMA (0, 1, 1) with drift	1941	Jul 2020
Jan 2010 – Jul 2019	ARIMA (0, 1, 0) with drift	1987	Aug 2020
Jan 2010 – Aug 2019	ARIMA (0, 1, 0) with drift	2002	Sep 2020
Jan 2010 – Sep 2019	ARIMA (0, 1, 0) with drift	2024	Oct 2020
Jan 2010 – Oct 2019	ARIMA (0, 1, 0) with drift	2042	Nov 2020
Jan 2010 – Nov 2019	ARIMA (0, 1, 0) with drift	2057	Dec 2020

Table 7-25 presents the forecasting performance of *Method IV*. The RMSE of the forecasted results for *Method IV* is found to be 1472, while the mean of the target variable for the out-of-sample data (i.e., the mean of the IT sector monthly index values) is 17378. Hence, the ratio of the RMSE to the mean of the target variable for this Method is found to be 8.47%, and the MAPE is 6.86%. Figure 7-9 presents the plot of the *IT sector's actual index values* and the corresponding forecasted values using *Method IV* for all the months of the year 2020.

TABLE 7-25. FORECASTED INDEX OF THE INDIAN IT SECTOR FOR THE YEAR 2020 USING *METHOD IV* OF FORECASTING

Month	Actual Value	Forecasted Value	Error (%)	RMSE/Mean	MAPE
Jan	15871	15504	2.31	8.47	6.86
Feb	14987	15912	6.17		
Mar	12843	15208	18.41		
Apr	14235	12843	9.78		
May	14067	14157	0.64		
Jun	14887	14155	4.92		
Jul	18251	14846	18.66		
Aug	18055	18356	1.67		
Sep	19980	18158	9.12		
Oct	21059	20097	4.57		
Nov	21635	21183	2.09		
Dec	22667	21764	3.98		

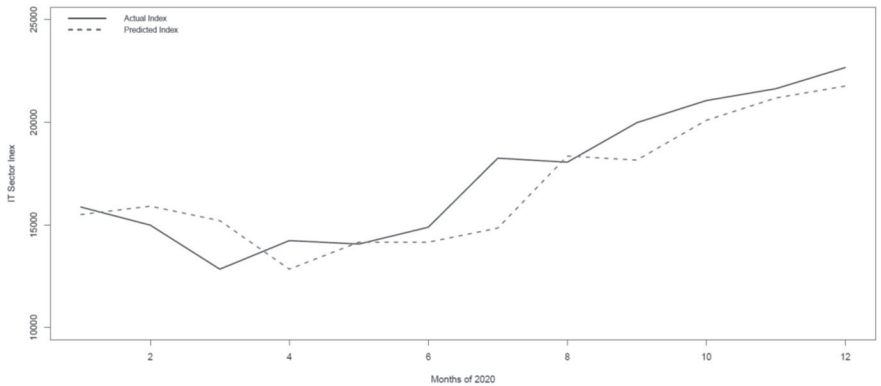

Figure 7-9. The actual and the predicted Indian IT sector index using *Method IV* (Period: Jan 2020 – Dec 2020)

TABLE 7-26. PERFORMANCE OF THE FOUR METHODS OF FORECASTING ON FIFTEEN SECTORS OF THE INDIAN ECONOMY

Sector	Method I		Method II		Method III		Method IV	
	RMSE/ Mean	MAPE	RMSE/ Mean	MAPE	RMSE/ Mean	MAPE	RMSE/ Mean	MAPE
Auto	30.99	25.01	17.22	17.16	20.61	18.99	**12.28**	**10.76**
Banking	11.87	9.41	10.96	9.49	14.56	13.02	**9.86**	**8.41**
Capital Goods	37.82	33.35	23.27	20.13	25.11	23.83	**12.47**	**11.32**
Cons Durable	21.90	21.16	16.15	12.60	14.98	13.87	**10.25**	**7.75**
FMCG	28.14	24.11	5.82	4.93	5.76	4.78	**4.43**	**3.65**
Healthcare	57.56	40.99	10.09	8.75	26.66	20.38	**7.87**	**6.37**
IT	57.25	58.14	12.79	10.22	21.61	15.64	**8.60**	**6.86**
Large Cap	11.66	9.67	10.79	9.35	14.71	13.19	**9.42**	**7.62**
Metal	75.35	77.55	25.94	25.03	32.20	31.99	**14.33**	**12.17**
Mid Cap	14.87	14.19	15.65	12.12	14.95	12.68	**10.85**	**8.54**
Oil & Gas	13.42	10.75	13.37	10.46	18.88	17.76	**9.62**	**8.06**
Power	15.46	12.23	16.98	13.53	19.39	18.34	**9.77**	**8.55**
Realty	334.69	304.15	31.11	26.26	33.22	33.18	**16.34**	**12.68**
Small Cap	24.69	24.55	17.69	15.02	14.23	12.05	**11.74**	**10.34**
Telecom	59.33	46.25	16.00	13.20	8.50	7.63	**9.23**	**7.55**

The performance results of the four methods of forecasting based on their ratios of RMSE to the mean value of the target variable and the MAPE for all the 15 sectors are presented in Table 7-26. Two important observations are noted in Table 7-26. First, it is interesting to note that the MAPE values are less than the RMSE to mean ratio values except for two cases. The two cases for which MAPE values are higher than the RMSE to mean values are the IT and the metal sectors for *Method I*. The second observation is that *Method IV* produces the most accurate results based on the MAPE values for all sectors. However, for the RMSE to the mean value, the more precise result is produced by *Method III* for the telecommunication sector. For all other sectors, *Method IV* is found to be more accurate based on the RMSE to the mean value.

Conclusion

This chapter emphasizes that understanding the sectoral characteristics is an important task before making a practical portfolio choice. The work described in the chapter also demonstrates that the sectors are different in terms of their trend, seasonal and random components of their respective time series of historical index values. The R programming language is used to decompose the time series of the historical index values of the fifteen critical sectors of the Indian economy, and the decomposition results are analyzed to understand their characteristics further. Several stocks from each sector are also analyzed in a similar line. Using the seasonal component only for illustrative purposes, we present extensive company-wise results to show that the individual companies, most of the time, mimic the corresponding sectoral seasonal patterns. It is found that from January 2010 to December 2020, 87.15 percent of the cases for the stocks from fifteen critical sectors of the Indian economy exhibit the same seasonality patterns as their respective sectors. For the stocks belonging to the capital goods, IT, realty, and the small-cap sector, the similarities between the seasonal behavior of the sectors and the stocks are found in 88.57 percent of the cases. The maximum similarity is found for the IT sector stocks with a matching percentage of 94.55, while the stocks in the small-cap exhibit the lowest similarity percentage of 78.18. We contend that this information can be effectively used by individual investors or asset management companies in designing efficient portfolios. Sen and Datta Chaudhuri pursued this line of thought to check for the consistency between the fund composition and the fund style for some pure equity funds in India and found some interesting observations (Sen, 2018; Sen and Datta Chaudhuri, 2016e).

Four different forecasting models using exponential smoothing techniques with the trend and seasonal components and ARIMA approaches are also presented for robust and accurate prediction of stock prices. The four forecasting methods differ in their algorithms and the length of the forecast horizon used. We applied these four methods to predict the monthly index of the IT sector for 2020 using the models built on the time series data from January 2010 to December 2019. It is found that the method based on the ARIMA model with a forecast horizon of one month produces the lowest value of the ratio of RMSE to the mean index value and the lowest MAPE value. Accordingly, this method is found to be the most accurate among the four methods. On the other hand, the method that uses exponential smoothing for the level, the trend, and the seasonality (i.e., the *HoltWinters* method) with a forecast horizon of twelve months for predicting the monthly index values of the IT sector for the year 2020 is found to be the least accurate one. It is interesting to note that the *beta* and the *gamma* parameters in the *HoltWinters* function for exponential smoothing in the R programming language identify the trend and the seasonal components, respectively. Hence, they can be gainfully exploited for building useful and accurate forecasting models.

References

Akcay, Y. and Yalcin, A. (2010) "Optimal portfolio selection with a shortfall probability constraint: Evidence from alternative distribution functions", *Journal of Financial Research*, Vol. 33, No. 1, pp. 77-102. DOI: 10.1111/j.1475-6803.2009.01263.x.

Anagnostopoulos, K. P. and Mamanis, G. (2011) "Multiobjective evolutionary algorithms for complex portfolio optimization problems", *Computational Management Science*, Vol. 8, pp. 259-279. DOI: 10.1007/s10287-009-0113-8.

Armananzas, R., Lozano, J. A. (2005) "A multiobjective approach to the portfolio optimization problem", *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, September 2-5, Edinburgh, Scotland, UK, Vol. 2, pp. 1388-1395. DOI: 10.1109/CEC.2005.1554852.

Banz, R. W. (1981) "The relationship between return and market value of common stocks", *Journal of Financial Economics*, Vol. 9, No. 1, pp. 3-18. DOI: 10.1016/0304-405X(81)90018-0.

Bao, W., Yue, J. and Rao, Y. (2017) "A deep learning framework for financial time series using stacked autoencoders and long-short term memory", *PLoS ONE*, Vol. 12, No. 7, e0180944.

DOI: 10.1371/journal.pone.0180944.

Basu, S. (1977) "Investment performance of common stocks in relation to their price earnings ratios: A test of the efficient market hypothesis", *The Journal of Finance*, Vol. 32, No. 3, pp. 663-682. DOI:10.1111/j.1540-6261.1977.tb01979.x.

Basu, S. (1983) "The relationship between earnings' yield, market value and return for NYSE common stocks: Further evidence", *Journal of Financial Economics*, Vol. 12, No. 1, pp. 129-156. DOI:10.1016/0304-405X(83)90031-4.

Berger, T. (2015) "Forecasting based on decomposed financial return series: A wavelet analysis", *Journal of Forecasting*, Vol. 35, No. 5, pp. 419-433. DOI:10.1002/for.2384.

Bhandari, L. C. (1988) "Debt/Equity ratio and expected common stock returns: Empirical evidence", *The Journal of Finance*, Vol. 43, No. 2, pp. 507-528. DOI:10.1111/j.1540-6261.1988.tb03952.x.

Caldeira, J.F, Moura, G.V. and Santos, A.A. (2017) "Yield curve forecast combinations based on bond portfolio performance", *Journal of Forecasting, Special Issue Article*. DOI:101.1002/for.2476.

Carriero, A., Kapetanios, G. and Marcellino, M. (2012) "Forecasting government bond yields with large Bayesian vector autoregressions", *Journal of Banking & Finance*, Vol. 36, No. 7, pp. 2026-2047. DOI: 10.1016/j.jbankfin.2012.03.008.

Cesarone, F. Scozzari, A. and Tardella, F. (2013) "A new method for mean-variance portfolio optimization with cardinality constraints", *Annals of Operations Research*, Vol. 205, pp. 213-234. DOI: 10.1007/s10479-012-1165-7.

Chan, L., Hamao, Y. and Lakonishok, J. (1991) "Fundamentals and stock returns in Japan"', *The Journal of Finance*, Vol. 46, No. 5, pp. 1739-1764. DOI:10.1111/j.1540-6261.1991.tb04642.x.

Chow, T-M., Hsu, J. C., Kuo, L-L. and Li, F. (2014) "A study of low-volatility portfolio construction methods", *The Journal of Portfolio Management*, Vol. 40, No. 4, pp. 89-105. DOI: 10.3905/jpm.2014.40.4.089.

Chui, A. C. W. and Wei, K. C. J. (1998) "Book-to-market, firm size, and the turn of the year effect: Evidence from pacific basin emerging markets", *Pacific Basin Finance Journal*, Vol. 6, No. 3, pp. 275-293. DOI:10.1016/S0927-538X(98)00013-4.

Coghlan, A. (2015) *A Little Book of R for Time Series*, Release 02. Available at: https://media.readthedocs.org/pdf/a-little-book-of-r-for-time-series/latest/a-little-book-of-r-for-time-series.pdf (Accessed on: August 14, 2017)

Cui, X., Zhu, S., Sun, X. and Li, D. (2013) "Nonlinear portfolio selection using approximate parametric value-at-risk", *Journal of Banking & Finance*, Vol. 37, No. 6, pp. 2124-2139. DOI: 10.1016/j.jbankfin.2013.01.036.

de Micheaux, P. L., Drouilhet, R. and Liquet, B. (2013) *The R Software: Fundamentals of Programming and Statistical Analysis*, Springer, New York, NY. DOI: 10.1007/978-1-4614-9020-3.

Deng, X. and Li, R. (2012) "A portfolio selection model with borrowing constraint based on possibility theory", *Applied Soft Computing*, Vol. 12, No. 2, pp. 754-758. DOI: 10.1016/j.asoc.2011.10.017.

Dickey, D. A. and Fuller, W. A. (1979) " Distribution of the estimators for autoregressive time series with a unit root", *Journal of the Americal Statistical Association*, Vol. 74, No. 366, pp. 427-431. DOI: 10.2307/2286348.

Dorsey, P. (2004) *The Five Rules for Successful Stock Investing: Morningstar's Guide to Building Wealth and Winning in the Market*, Wiley International. ISBN: 978-0-471-68617-0.

Duan, L. and Stahlecker, P. (2011) "A portfolio selection model using fuzzy returns", *Fuzzy Optimization and Decision Making*, Vol. 10, pp. 167-191. DOI: 10.1007/s10700-011-9101-x.

Fama, E. F. and French, K. R. (1988) "Dividend yields and expected stock returns", *Journal of Financial Economics*, Vol. 22, No. 1, pp. 3-25. DOI:10.1016/0304-405X(88)90020-7.

Fama, E. F. and French, K. R. (1992) "The cross-section of expected stock returns", *The Journal of Finance*, Vol. 47, No. 2, pp. 427-465. DOI:10.1111/j.1540-6261.1992.tb04398.x.

Fama, E. F. and French, K. R. (1995) "Size and book-to-market factors in earnings and returns", *The Journal of Finance*, Vol. 50, No. 1, pp. 131-155. DOI:10.1111/j.1540-6261.1995.tb05169.x.

Graham, B. and Dodd, D. (2008) *Security Analysis*, The McGraw-Hill Inc, ISBN: 978-0-070-14065-3.

Huang, X. and Qiao, L. (2012) "A risk index model for multi-period uncertain portfolio selection", *Information Sciences: An International Journal*, Vol. 217, pp.108-116. DOI: 10.1016/j.ins.2012.06.017.

Hyndman, R. J. and Athanasopoulos, G. (2018) *Forecasting: Principles and Practice*, 2nd Edition, OTexts: Melbourne, Australia. ISBN-13: 978-0987507112.

Ibbotson, R. G. and Idzorek, T. M. (2014) "Dimensions of popularity", *The Journal of Portfolio Management*, Vol. 40, No. 5, pp. 68 - 74. DOI:10.3905/jpm.2014.40.5.068.

Jaffe, J., Keim, D. B. and Westerfield, R. (1989) "Earnings yields, market values, and stock returns", *The Journal of Finance*, Vol. 44, No. 1, pp. 135 - 148. DOI:10.1111/j.1540-6261.1989.tb02408.x.

Jegadeesh, N. and Titman, S. (1993) "Returns to buying winners and selling losers: Implications for stock market efficiency", *The Journal of Finance*, Vol. 48, No. 1, pp. 65 - 91. DOI: 10.1111/j.1540-6261.1993.tb04702.x.

Kothari, S. P. and Shanken, J. (1997) "Book-to-market, dividend yield, and expected market returns: A time series analysis", *Journal of Financial Economics*, Vol. 44, No. 2, pp. 169-203. DOI:10.1016/S0304-405X(97)00002-0.

Kumar, R. and Bhattacharya, S. (2012) "Cooperative search using agents for cardinality constrained portfolio selection problem", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 42, No. 6, pp. 1510-1518. DOI: 10.1109/TSMCC.2012.2197388.

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P. and Shin, Y. (1992) "Testing the null hypothesis of stationary against the alternative of a unit root: How sure are we that economic time series have a unit root?", *Journal of Econometrics*, Vol. 54, No. 1- 3, pp. 159-178. DOI: 10.1016/0304-4076(92)90104-Y.

Li, T., Zhang, W. and Xu, W. (2015) "A fuzzy portfolio selection model with background risk", *Applied Mathematics and Computation*, Vol. 256, pp. 505-513. DOI: 10.1016/j.amc.2015.01.007.

Liu, Y. J. and Zhang, W. G. (2015) "A multi-period fuzzy portfolio optimization model with minimum transaction lots", *European Journal of Operational Research*, Vol. 242, No. 3, pp. 933-941. DOI: 10.1016/j.ejor.2014.10.061.

Lynch, P. and Rothchild, J. (2000) *One Up On Wall Street: How to Use What You Already Know to Make Money in the Market*, Simon & Schuster, ISBN 978-0743200400.

Mehlawat, M. K. and Gupta, P. (2014) "Fuzzy chance-constrained multiobjective portfolio selection model", *IEEE Transactions on Fuzzy Systems*, Vol. 22, Noo. 3, pp. 653-671. DOI: 10.1109/TFUZZ.2013.2272479.

Mehtab, S. and Sen, J. (2022) "Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models", In: Sahoo, J. P., Tripathy, A, K., Mohanty, M., Li, K. C., Nayak, A. K. (eds.), *Advances in Distributed Computing and Machine Learning*. Lecture Notes in Networks and Systems, Vol 202, pp. 405-423, Springer, Singapore. DOI: 10.1007/978-981-16-4807-6_39.

Mehtab, S. and Sen, J. (2021a) "A time series analysis-based stock price prediction using machine learning and deep learning models", *International Journal of Business Forecasting and Marketing Intelligence (IJBFMI)*, Vol 6, No 4, pp. 272-335. Inderscience Publishers. DOI: 10.1504/IJBFMI.2020.115691.

Mehtab, S., Sen, J. and Dutta, A. (2021b) "Stock price prediction using machine learning and LSTM-based deep learning models", In: Thampi, S. M., Piramuthu, S., Li, K. C., Beretti, S., Wozniak, M., and Singh, D. (eds.), *Machine Learning and Metaheuristics Algorithms and Applications*, *SoMMA 2020*. Communications in Computer and Information Science, Vol. 1366, pp. 88-106, Springer, Singapore. DOI: 10.1007/978-981-16-0419-5_8.

Mehtab, S. and Sen, J. (2020a) "Stock price prediction using convolutional neural networks on a multivariate time series", *Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence* (*NCMLAI'20*), February 1, 2020, New Delhi, India. DOI: 10.36227/techrxiv.15088734.v1.

Mehtab, S. and Sen, J. (2020b) "Stock price prediction using CNN and LSTM-based deep learning models", *Proceedings of the IEEE International Conference on Decision Aid Sciences and Applications (DASA'20)*, November 8-9, 2020, Sakheer, Bahrain, Bahrain, pp. 447-453. DOI: 10.1109/DASA51403.2020.9317207.

Mehtab, S., Sen, J. and Dasgupta, S. (2020c) "Robust analysis of stock price time series using CNN and LSTM-based deep learning models", *Proceedings of the 4th IEEE International Conference on Electronics, Communication and Aerospace Technology (ICECA'20)*, November 5-7, 2020, Coimbatore, India, pp. 1481-1486. DOI: 10.1109/ICECA49313.2020.9297652.

Mehtab, S. and Sen, J. (2020d) *Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models*, Technical Report, No: NSHM_KOL_2020_SCA_DS_1. DOI: 10.13140/RG.2.2.14022.22085/2.

Mehtab, S. and Sen, J. (2019) "A robust predictive model for stock price prediction using deep learning and natural language processing", *Proceedings of the 7th International Conference on Business Analytics and Intelligence* (*BAICONF'19*), December 5-7, 2019, Bangalore, India. DOI:10.2139/ssrn.3502624.

Rios, L. M. and Sahinidis, N. V. (2010) "Portfolio optimization for wealth-dependent risk preference", *Annals of Operations Research*, Vol. 177, pp. 63-90. DOI: 10.1007/s10479-009-0592-6.

Rosenberg, B., Reid, K. and Lanstein, R. (1985) "Persuasive evidence of market inefficiency", *The Journal of Portfolio Management*, Vol. 11, No. 3, pp. 9 -17. DOI:10.3905/jpm.1985.409007.

Sadjadi, S. J., Gharakhani, M. and Safari, E. (2015) "Robust optimization framework for cardinality constrained portfolio problem", *Applied Soft Computing*, Vol. 12, No. 1, pp. 91-99. DOI: 10.1016/j.asoc.2011.09.006.

Sen, J. (2018) "Stock composition of mutual funds and fund style: A time series decomposition approach towards testing for consistency", *International Journal of Business Forecasting and Marketing Intelligence (IJBFMI)*, Vol. 4, No. 3, pp. 235-292. DOI: 10.1504/IJBFMI.2018.092781.

Sen, J. and Datta Chaudhuri, T. (2018) "Understanding the sectors of Indian economy for portfolio choice", *International Journal of Business Forecasting and Marketing Intelligence*, Vol. 4, No. 2, pp. 178-222. DOI: 10.1504/IJBFMI.2018.090914.

Sen, J. and Datta Chaudhuri, T. (2017) "A predictive analysis of the Indian FMCG sector using time series decomposition-based approach", *Journal of Economics Library*, Vol. 4, No. 2, pp. 206-226. DOI: 10.1453/jel.v4i2.1282.

Sen J. and Datta Chaudhuri, T. (2016a) "Decomposition of time series data of stock markets and its implications for prediction - An application for the Indian auto sector", *Proceedings of the 2nd National Conference on Advances in Business Research and Management Practices* (*ABRMP'16*), January 8 – 9, pp. 15-28, Kolkata, India. DOI:10.13140/RG.2.1.3232.0241.

Sen, J. and Datta Chaudhuri, T. (2016b) "A framework for predictive analysis of stock market indices - A study of the Indian auto sector", *Calcutta Business School (CBS) Journal of Management Practices*, Vol. 2, No. 2, pp. 1-20. DOI:10.13140/RG.2.1.2178.3448.

Sen, J. and Datta Chaudhuri, T. (2016c) "An alternative framework for time series decomposition and forecasting and its relevance for portfolio choice: A comparative study of the Indian consumer durable and small cap sectors", *Journal of Economics Library*, Vol. 3, No. 2, pp. 303-326. DOI:10.1453/jel.v3i2.787.

Sen, J. and Datta Chaudhuri, T. (2016d) "An investigation of the structural characteristics of the Indian IT sector and the capital goods sector - An application of the R programming in time series decomposition and forecasting", *Journal of Insurance and Financial Management*, Vol. 1, No. 4, pp. 68 - 132. DOI: 10.36227/techrxiv.16640227.v1.

Sen, J. and Datta Chaudhuri, T. (2016e) "Decomposition of time series data to check consistency between fund style and actual fund composition of mutual funds", *Proceedings of the 4th International Conference on Business Analytics and Intelligence* (*ICBAI'16*), Paper ID: 82, December 19-21, Bangalore, India. DOI:10.13140/RG.2.2.33048.19206.

Sen, J. and Mehtab, S. (2021a) "Accurate stock price forecasting using robust and optimized deep learning models", *Proceedings of the IEEE International Conference on Intelligent Technologies (CONIT'21)*, June 25-27, Hubballi, India. DOI: 10.1109/CONIT51480.2021.9498565.

Sen, J. and Mehtab, S. (2021b) "Design and analysis of robust deep learning models for stock price prediction", In: Sen, J. (ed.) *Machine Learning – Algorithms, Models and Applications*, IntechOpen, London UK, pp. 15 – 46. DOI: 10.5772/intechopen.99982.

Sen, J., Mehtab, S. and Nath, G. (2020) "Stock price prediction using deep learning models", *Lattice: The Machine Learning Journal*, Vol 1, No 3, pp. 34-40, December 2020. DOI: 10.36227/techrxiv.16640197.v1.

Tsay, R. S. (2014) *Analysis of Financial Time Series*, 3rd Edition, Wiley, NJ, USA. ISBN: 978-8126548934.

Vaclavik, M., Jablonsky, J. (2011) "Revisions of moderns portfolio theory optimization model", *Central European Journal of Operations Research*, Vol. 20, No. 3, pp. 473-483. DOI: 10.1007/s10100-011-0227-2.

Yang, Y., Rubio, F., Scutari, G. and Palomar, D. P. (2013) "Multi-portfolio optimization: A potential game approach", *IEEE Transaction of Signal Processing*, Vol. 61, No. 22, pp. 5590-5602. DOI: 10.1109/TSP.2013.2277839.

Zhao, Z. Y., Xie, M. and West, M. (2016) "Dynamic dependence networks: Financial time series forecasting and portfolio decisions", *Bayesian Statistics and Machine Learning in Business*, *Special Issue*, Vol. 32, No. 3, pp. 311-332.

Zhang, X., Zhang, W. G., Xu, W. J. (2011) "An optimization model of the portfolio adjusting problem with fuzzy return and a SMO algorithm", *Expert Systems with Applications*, Vol. 38, No. 4, pp. 3069-3074. DOI: 10.1016/j.eswa.2010.08.097.

Zhu, S., Fan, M. and Li, D. (2014) "Portfolio management with robustness in both prediction and decision: A mixture model based learning approach", *Journal of Economic Dynamics and Control*, Vol. 48, pp. 1-25. DOI: 10.1016/j.jedc.2014.08.015.

CONTRIBUTORS

Jaydip Sen has experience in research, teaching and industry over a span of 27 years. He had worked in reputed organizations like Oil and Natural Gas Corporation Ltd, India, Oracle India Pvt Ltd, and Akamai Technology Pvt Ltd, Tata Consultancy Services Ltd and National Institute of Science and Technology, India, and Calcutta Business School, India. Currently, he is associated with Praxis Business School, Kolkata, India, as a professor in the department of Data Science and Artificial Intelligence. His research areas include security in wired and wireless networks, intrusion detection systems, secure routing protocols in wireless ad hoc and sensor networks, trust, and reputation-based systems, and privacy issues in ubiquitous and pervasive communication and the Internet of Things, machine learning, deep learning and artificial intelligence in the financial domain. He has more than 200 publications in reputed international journals and referred conference proceedings and 20 book chapters in books published by internationally renowned publishing houses like Springer, CRC press, IGI-Global, etc. He has also authored two books which are published by two internationally reputed publishing houses. He has been listed among the top 2% of scientists in the world for the last three consecutive year 2019-2021, as per studies conducted by Stanford University, USA. Prof. Sen serves on the editorial board of three prestigious international journals and in the technical program committee in a number of international conferences of repute.

Sidra Mehtab completed her BS with honors in Physics from Calcutta University, India in 2018. She has done MS in Data Science and Analytics from Maulana Abul Kalam Azad University of Technology (MAKAUT), Kolkata, India in 2020. Her research areas include econometrics, time series analysis, machine learning, deep learning, and artificial intelligence with a focus on the financial sector. Ms. Mehtab has published seventeen papers which are listed in the Scopus. She has won the best paper awards in two prestigious international conferences – BAICONF 2019, and ICADCML 2021, organized in the Indian Institute of Management, Bangalore, India in December 2019, and SOA University, Bhubaneswar, India in January 2021. Besides, Ms. Mehtab has also published four book chapters. Currently, she is a practicing data scientist with a reputed bank in India.

Abhishek Dutta did his BS in Computer Science from Calcutta University, India in 2018. He did his MS in Data Science and Analytics from Maulana Abul Kalam Azad University of Technology (MAKAUT), Kolkata, India in 2020. Mr. Dutta is currently working in the private banking sector as a data analyst in financial crime-risk monitoring and anti-money laundering. The majority of his works are aligned with stock price prediction using statistical methods, machine learning, and deep learning techniques. He has published eight papers which are indexed in Scopus. He is an avid sports enthusiast.

Ashmita Paul completed her BS in Mathematics from West Bengal State University, India in 2018. She completed her MS in Data Science from Maulana Abul Kalam Azad University of Technology (MAKAUT), Kolkata, India in 2020. Her areas of interest include statistical, econometric and machine learning approaches to time series analysis.