Analysis and Forecasting of Financial Time Series Using CNN and LSTM-Based Deep Learning Models

Sidra Mehtab and Jaydip Sen(✉)

Department of Data Science and Artificial Intelligence, Praxis Business School, Bakrahat Road, Kolkata 700104, India
smehtab@acm.org, jaydip@praxis.ac.in

Abstract. Designing predictive models for forecasting future stock price has always been a very popular area of research. On the one hand, the proponents of the famous *efficient market hypothesis* believe that it is impossible to accurately predict stock prices, on the other hand, propositions exist in the literature that demonstrate that it is possible to very precisely predict stock prices by accurate modeling of the predictive systems. We propose a robust framework consisting of a suite of deep learning-based regression models that yields a very high level of accuracy in forecasting of stock prices. The models are built using the historical stock price data of a well-known company listed in the National Stock Exchange (NSE) of India. The stock prices are recorded at five minutes interval during the period December 31, 2012 to January 9, 2015. Exploiting the features in these extremely granular stock price data, we build four convolutional neural network (CNN) and five long- and short-term memory (LSTM)-based deep learning models, validate them and finally test them on their performance. Extensive results are presented on the performance of the models on two metrics- execution time and root mean square error (RMSE) values.

Keywords: Stock price prediction · Regression · Long and short-term memory network · Walk-forward validation · Multivariate time series

1 Introduction

Analysis of financial time series and prediction of future stock prices and future stick price movement patterns have been an active area of research over a considerable period of time. While there are people who believe in the well-known *efficient market hypothesis* and claim that it is impossible to forecast stock prices accurately, propositions exist in the literature that demonstrate that it is possible to predict the values of stock prices with a very high level of accuracy using optimized designed and fine-tuned models. The latter have focused on the choice of variables, appropriate functional forms, and techniques of forecasting. Time series decomposition of stock price data is also a popular approach for stock price forecasting [1, 2]. The use of machine learning and deep learning-based

J. P. Sahoo et al. (eds.), *Advances in Distributed Computing and Machine Learning*,
Lecture Notes in Networks and Systems 302,
https://doi.org/10.1007/978-981-16-4807-6_39

approaches are also adopted in some propositions [3]. An approach based on text mining and natural language processing in analyzing the sentiments in the social media, and utilizing that information in building a non-linear predictive model for accurate stock price prediction has been proposed in the literature [4]. Deployment of a suite of *convolutional neural networks* (CNN) for achieving a very high level of accuracy in the forecasting of the stock price has also been proposed been also proposed [5].

In this paper, we propose a suite of deep learning-based predictive models for forecasting of future stock prices of a well-known company listed in the National Stock Exchange (NSE) of India. Our proposition includes three regression models that are built on *convolutional neural networks* (CNNs) and four predictive models based on *long-and-short-term memory* (LSTM) networks. The contributions of the paper are three-fold. First, unlike most of the existing propositions in the literature, our models are designed to handle extremely granular stock price data collected at 5 min interval of time. Second, our propositions include deep learning models that provide a very high level of accuracy in stock price forecasting. The most accurate model in our proposition yielded a value of 0.0057 as a ratio of *root mean square error* (RMSE) to the mean value of the target variable in the test data set. Third, the proposed models are very fast in execution. The fastest model took to 81.98 s to execute over a training dataset consisting of 19,500 records and a test dataset with 20,500 records on our target hardware platform.

The rest of the paper is organized as follows. In Sect. 2, we very briefly discuss some related work on forecasting of stock prices. Section 3 presents a detailed discussion on the stock price data that we use, and the research methodology we follow in this work. Detailed experimental results on the performance of the proposed models are presented in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 Related Work

Forecasting of future stock prices and stock price movement patterns have been an active area of research over a long period of time. The propositions are quite diverse in their methodology and algorithms used by them. However, most the approaches can broadly categorized into three groups. The propositions belonging to the first category follow *ordinary least square* (OLS) regression or some form of its variants on cross-sectional data [6–8]. Unfortunately, models using this approach fail to achieve high accuracy in forecasting. This is because of the fact that the assumptions made by the models are, most often, not satisfied by the real-world data. The propositions of the second category use time series models using econometric approaches like *autoregressive integrated moving average* (ARIMA), *vector autoregression* (VAR), and *autoregressive distributed lag* (ARDL) [7, 9, 10]. Although, these models perform well on financial time series data with dominant trend and seasonality components, their performance on volatile data exhibiting high degree of randomness has been suboptimal. The models in the third category are leaning-based systems that use various algorithm of machine learning, deep learning and natural language processing on structured and unstructured data including textual information in the social web [11–13]. The predictive models belonging to the third category are found to be most effective in handling volatile and granular financial time series data.

The most common drawback of the majority of the existing approaches for stock price prediction is their inability to effectively handle randomness in financial time series data. Our current work attempts to address this problem by exploiting the high learning abilities of *convolutional neural networks* (CNNs) and *long- and short-term memory* (LSTM) networks.

3 Methodology and Model Designs

In this work, our main goal is to build a framework of predictive models for accurately forecasting the *open* value of the stock price of the company *Tata Steel* listed in the NSE, India. We used stock price data for the company *Tata Steel* for the period December 31, 2012 (which was a Monday) to January 9, 2015 (which was a Friday). During this period of time, the stock price data have been captured at an interval of 5 min by the Metastock tool [14]. For building the models we used the stock price data for the period December 31, 2012 to December 30, 2013. The models were tested on the historical stock price data from December 31, 2013 to January 9, 2015. The entire dataset has also been organized in a form of a weekly sequence of data from Monday to Friday. The training and the test datasets consisted of 19,500 and 20,500 records respectively. Each record consists of the following attributes: (i) *date*, (ii) *time* slot, (iii) *open*, (iv) *high*, (v) *low*, (vi) *close*, and (vii) *volume*.

We build seven deep learning-based regression models for forecasting stock prices. In the univariate models, we use *open* as the target variable and predict the future *open* values using its past values. In the multivariate models, while *open* is still used as the response variable, other variables, e.g., *high*, *low*, *close*, and *volume* are used as the predictors. For testing our proposed models, we followed the *multi-step forecasting with walk-forward validation* approach [8]. Following this method, we build our models using records in the training dataset and then forecast the *open* values of the *Tata Steel* stock for each day in a week. At the completion of one week, we include the actual *open* values of the stock records in the training dataset, and then make the forecast for the *open* values for the next week. At each round, we make forecasting of the *open* values for the five days in the upcoming week.

We have demonstrated the efficacy and efficiency of *convolutional neural networks* (CNNs) in time series analysis and forecasting in one of our recently published work [5]. In current work, in addition to exploiting the power of CNN, we propose the application of *long-and-short-term memory* (LSTM) networks in analysis and forecasting on complex multivariate time series.

CNNs have two processing layers that are responsible for carrying out the major computations [5]. While the convolutional layers are responsible for identifying important features from the input data, the pooling or sub-sampling layers summarize those features and extract the most prominent among them in a locality. The final pooling layer feeds its output into one or more dense layers that finally carryout the classification or regression task. LSTM is a variant of deep neural network that has the capability to read and interpret sequential data like text or time series [13]. LSTM networks have the ability to maintain their state information using memory cells and gates. The gates enable these networks to reject irrelevant information of the past, remember important information

in the current state, and capture the input to the system at the current instant of time in order to produce the output as the forecast for the next time instance. The state vector in the LSTM memory cell carries out aggregation of the old information received from the *forget gates*, and the most recent information received from the *input gates*. Finally, the output gates produce the output from the network at the current slot. This output can be considered as the forecasted value computed by the model for the current slot [13].

We propose seven different deep learning-based predictive models in this work. The models vary in their design, architecture, and input data shapes. Among these models, there are based on CNN architecture and the remaining four are of LSTM networks. The models are: (i) CNN model with univariate input data of the previous one week, (ii) CNN model with univariate input data of the previous two weeks, (iii) CNN model with multivariate input data of the previous two weeks, (iv) LSTM model with univariate input data of previous one week, (v) LSTM model with univariate input data of previous two weeks, (vi) Encoder-decoder LSTM with univariate data of previous two weeks, (vii) Encoder-decoder LSTM model with multivariate input data of previous two weeks.

The first model is a CNN that takes the previous one week's data as the input and makes forecasting of the next week's *open* values in a *multi-step walk-forward manner*. The input data shape is (5, 1) indicating that only one attribute (i.e., *open* values) of the stock price time series is used for the five days in the previous week. The model deploys one *convolutional* layer and one *max pooling* layer. The output of the subsampling layer is reshaped into a flat one-dimensional vector, and then the flattened vector is passed into a *fully-connected* layer. Finally, the output layer predicts the *open* values for the next five days. The model is trained using 20 epochs and a batch size of 4 using a *rectified linear unit* (ReLU) as the activation function, and ADAM as the optimizer. The architecture of the model is presented in Fig. 1. We refer to this model as CNN#1 model.

The second model that we propose is also a CNN that uses the previous two weeks' *open* values and forecasts the next week's *open* values in a univariate forecasting method. The architecture of the model is identical to that of CNN#1 model, except for the input data shape, which is (10, 1) for this model. We refer to this model as the CNN#2 model.

The third model is a multivariate CNN model that uses the previous two week's data as its input. Each of the five variables, i.e., *open*, *high*, *low*, *close*, and *volume* is used as a separate channel in a CNN. Each channel uses its own kernel and reads the sequence of input corresponding to its variable. Two *convolutional* layers with 32 filter maps with a kernel size of 3 are deployed in this model. A *max-pooling* layer follows each of the *convolution* layers. One fully connected layer with 100 nodes processes the data after the second *max-pooling* layer before the output layer produces the forecast of the *open* values of the stock for the five days in the next week. Figure 2 presents the architecture of the model, which we refer to as the CNN#3 model.

The fourth model and the first of the LSTM suite that we propose in this work is a univariate LSTM model that uses the previous one week's *open* values to forecast the *open* values for the five days in the next week. The shape of the input data to the model is (5, 1) as in the case of the CNN#1 model. The input data is passed through an LSTM layer consisting of 200 nodes. The output of the LSTM layer propagates through a dense layer that has 200 nodes at its input and 100 nodes at the output. Finally, the forecasted values are produced by the output layer that is connected to the fully-connected layer.

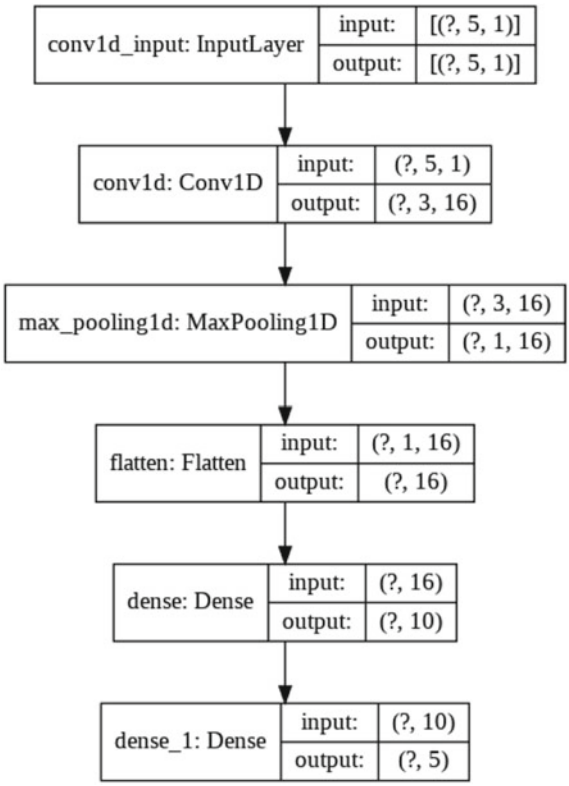

Fig. 1. The univariate CNN model (CNN#1) with one week's data as the input

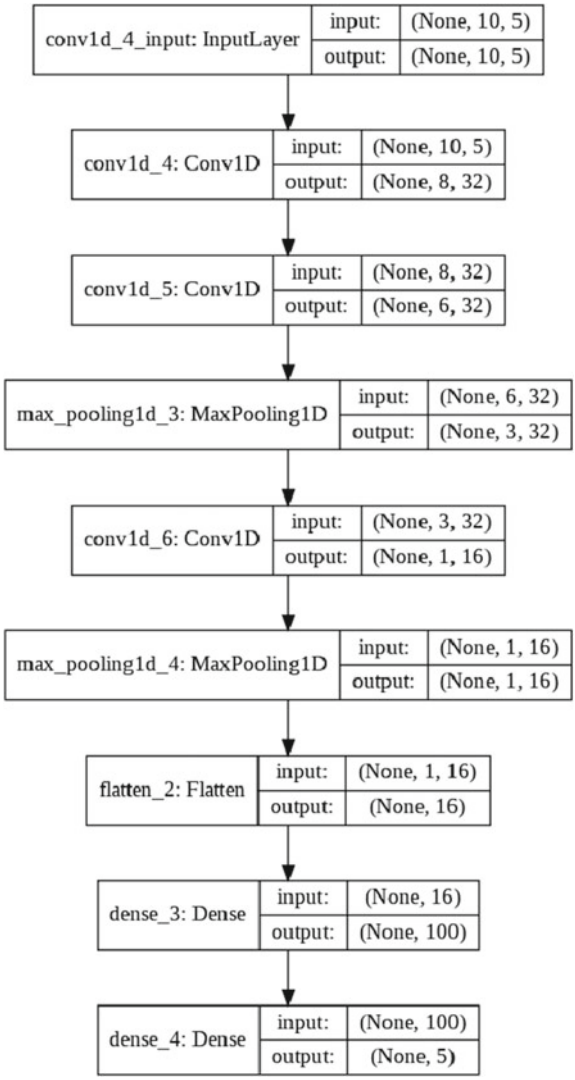

Fig. 2. The multivariate CNN model (CNN#3) with two weeks' data as the input

The output layer has 100 nodes at its input and 5 nodes at the output. Figure 3 depicts the architecture of the model, which we refer to as LSTM#1.

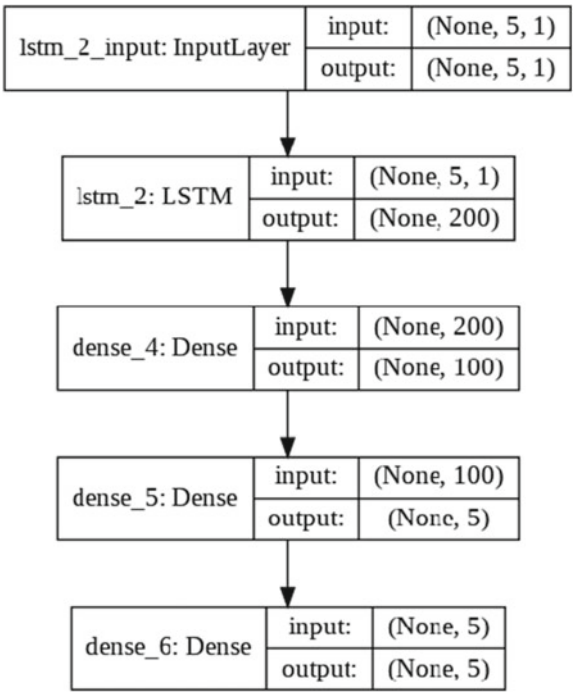

Fig. 3. The univariate LSTM model (LSTM#1) with one week's data as the input

The fifth model and the second model in the LSTM suite is a univariate model with the previous two weeks' *open* values as the input. The design of the model is identical to that of the LSTM#1 model, except for the input data shape which is (10, 1) as the input is the previous two weeks' *open* values. We call this model as LSTM#2.

The sixth model and the third one in the LSTM suite is an encoder-decoder univariate model that takes the previous two weeks' data as the input and predicts the *open* values for the five days of the next week [13]. We call this model as LSTM#3, which is shown in Fig. 4.

The seventh and the final model in our proposition and the fourth model in the LSTM suite is an encoder-decoder LSTM model that uses a multivariate time series data as its input. The model receives the previous two weeks' stock price data with all the variables. i.e., *open*, *high*, *low*, *close*, and *volume*, as the inputs. Based on this multivariate input, the model forecasts the *open* values of the stock for the five days in the next week. We call this model as LSTM #4 which is shown in Fig. 5.

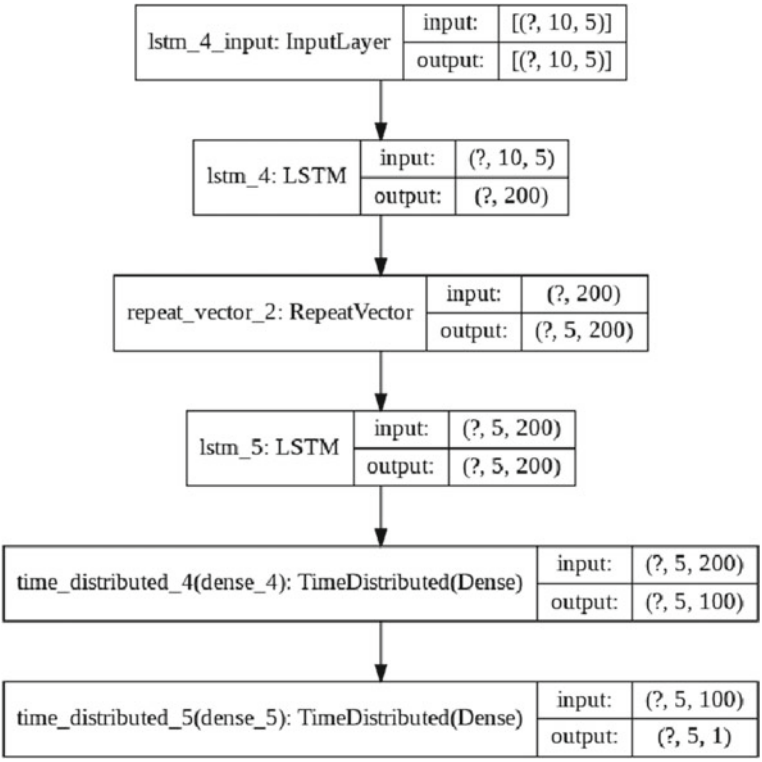

Fig. 4. The univariate encoder-decoder model (LSTM#3) with two weeks' data as the input

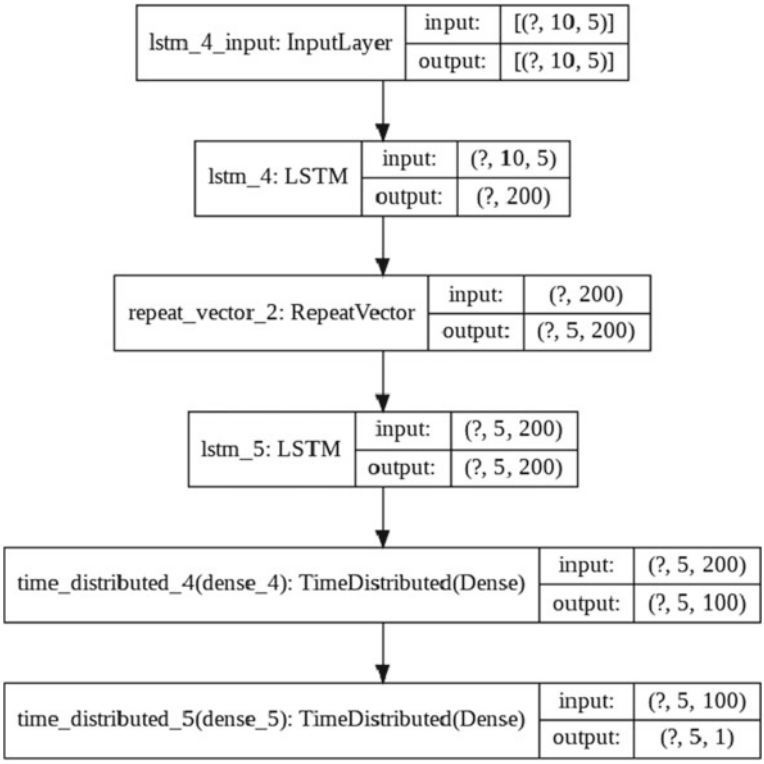

Fig. 5. The multivariate encoder-decoder LSTM#4 model with two weeks' data as the input

4 Methodology and Model Designs

In this section, we present the performance results of the deep learning models. The details of the design of all the five models were presented in Sect. 3. Each model was tested over 10 rounds, and the performance of the model was noted in terms of the overall RMSE, the RMSE values for the individual days of a week (i.e., Monday–Friday), the time needed to execute one round of the model, and the ratio of the RMSE to the mean of the actual open value in the test dataset. It may be recollected that the number of records in the training and the test dataset were19,500 and 20,250 respectively. The mean open value in the test dataset was 451.44. The model has been executed on a laptop computer consisting of an Intel i7 CPU with a clock speed of 2.60–2.569 GHz, and 16 GB RAM, and running on 64-bit Windows 10 operating system. The execution time needed for each round of each model is noted in seconds.

Table 1 presents the performance results of the CNN#1 model. It is observed that the CNN#1 model took, on an average, 81.98 s for the execution of one round. It yielded a mean value of the ratio of RMSE to the mean of the *open* values in the test dataset as 0.0057. Figure 6 presents the performance results of the CNN#1 model for round #5 presented in Table 1. The ratio of the mean RMSE to the mean of the actual *open* values for the model for Monday to Friday have been found to be 0.004386, 0.004984, 0.005826, 0.006357, and 0.006690 respectively. Figure 6 shows the variation of the mean RMSE for different days in a week as per the round 5 in Table 1.

Table 1. The performance results of the CNN#1 model

No.	RMSE	Mon	Tue	Wed	Thu	Fri	Time
1	2.501	1.80	2.10	2.40	2.90	3.00	83.00
2	2.636	2.2	2.20	2.40	3.00	3.20	84.57
3	2.415	2.0	2.20	2.40	2.60	2.80	83.63
4	2.356	1.9	2.10	2.40	2.50	2.70	83.06
5	2.405	2.1	2.20	2.30	2.60	2.80	81.92
6	2.320	1.8	2.10	2.30	2.60	2.70	80.60
7	3.475	2.0	2.60	4.00	4.10	4.00	80.48
8	2.538	1.9	2.20	2.60	2.90	3.00	80.98
9	2.906	2.3	2.70	3.20	3.00	3.30	79.85
10	2.289	1.8	2.10	2.30	2.50	2.70	81.68
Mean	**2.5841**	1.98	2.25	2.63	2.87	3.02	**81.98**
Min	2.289	1.80	2.10	2.30	2.50	2.70	79.85
Max	3.475	2.30	2.70	4.00	4.10	4.00	84.57
SD	0.3619	0.18	0.22	0.55	0.48	0.40	1.54
RMSE/Mean	**0.0057**	0.004	0.005	0.006	0.006	0.007	

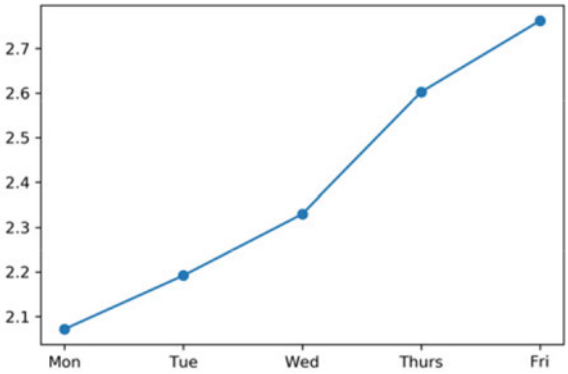

Fig. 6. RMSE of CNN#1 (Round#5 of Table 1)

The performance results of the CNN#2 are presented in Table 2. The model exhibited a mean execution time of 84.71 s for one round, which is marginally higher than that of the model CNN#1 model. The average value the ratio of the RMSE to the mean of the actual *open* values yielded by the model was 0.0067, which is also larger than the corresponding value of the model CNN#1. The ratio of the mean RMSE to the mean of the actual *open* values for the model for Monday to Friday have been found to be 0.005604, 0.006668, 0.006690, 0.006668, and 0.007642 respectively. Figure 7 depicts the variations of RMSE with different days in a week exhibited by CNN#2 for round 3 in Table 2.

The performance results of the CNN#3 model are presented in Table 3. While the mean execution time for one round of the model is 126.23 s, the RMSE to the mean of the actual *open* values in the test dataset for the model is found to be 0.0075. The ratio of the mean RMSE to the mean of the actual open values for Monday to Friday are found to be 0.006512, 0.007310, 0.007531, 0.007930, and 0.008285 respectively. Figure 8 shows the variations of RMSE with respect to different days in a week as per the round 2 in Table 3.

Table 4 presents the performance of the LSTM#1 model. It is observed that the mean time of execution of one round of the model is 316.74 s, while the value of the RMSE to the mean *open* value in the test dataset is 0.0059. The ratio of the mean RMSE to the mean of the actual open values for Monday to Friday are found to be 0.005028, 0.005471, 0.005560, 0.006490, and 0.006800 respectively. Figure 9 shows how the mean RMSE of the model varies for different days in a week as per the round 6 in Table 4.

Table 5 shows that the mean time for execution of one round of the model LSTM#2 is 525.47 s, while the value of the ratio of RMSE to the mean of the actual *open* values in the test dataset is 0.0067. The ratio of the mean RMSE to the mean of the actual open values for Monday to Friday are found to be 0.006136, 0.006114, 0.006867, 0.006845, and 0.007310 respectively. Figure 10 depicts the way the mean RMSE of the model varies with different days in a week as per the round 9 in Table 5.

The performance results of the model LSTM#3 are presented in Table 6. The model took a mean time of 314.87 s for each round, and it yielded a value of 0.0063 as the ratio of the RMSE to the mean *open* value. The ratio of the mean RMSE to the mean of the

Table 2. The performance results of the CNN#2 model

No.	RMSE	Mon	Tue	Wed	Thu	Fri	Time
1	5.566	4.60	5.90	5.20	5.20	6.8	83.64
2	2.502	2.20	2.10	2.90	2.60	2.70	82.74
3	2.886	2.60	2.60	2.80	3.10	3.20	85.45
4	3.195	3.10	3.60	3.50	2.50	3.20	83.07
5	2.641	2.60	2.40	2.50	2.80	2.90	82.37
6	3.141	1.90	3.60	3.40	2.60	3.80	87.63
7	3.175	2.60	3.00	3.10	3.60	3.50	82.94
8	2.441	2.00	2.50	2.30	2.50	2.80	86.71
9	2.317	1.70	2.20	2.20	2.60	2.70	87.07
10	2.397	2.00	2.20	2.30	2.60	2.90	85.53
Mean	**3.0261**	2.53	3.01	3.02	3.01	3.45	**84.71**
Min	2.317	1.70	2.10	2.20	2.50	2.70	82.37
Max	5.566	4.60	5.90	5.20	5.20	6.80	87.63
SD	0.9544	0.84	1.15	0.89	0.84	1.23	1.99
RMSE/Mean	**0.0067**	0.006	0.007	0.007	0.007	0.008	

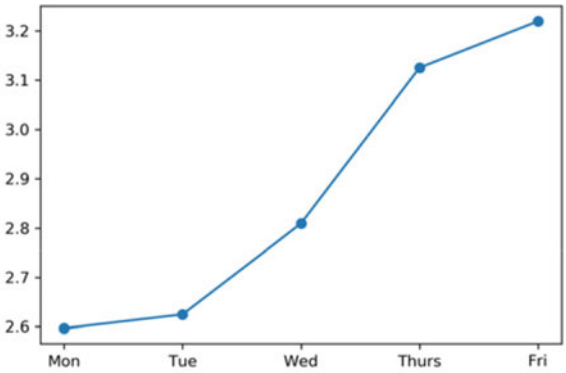

Fig. 7. RMSE of CNN#2 (Round#3 of Table 2)

Table 3. The performance results of the CNN#3 model

No.	RMSE	Mon	Tue	Wed	Thu	Fri	Time
1	4.001	3.70	4.10	4.00	4.10	4.10	115.46
2	5.207	4.90	5.00	5.40	5.30	5.50	124.68
3	3.496	3.10	3.30	3.30	3.70	3.90	119.17
4	2.552	2.20	2.30	2.50	2.70	2.90	127.67
5	4.072	3.30	4.00	4.20	4.20	4.50	133.50
6	2.814	2.40	2.70	2.80	3.00	3.20	115.34
7	2.847	2.30	2.70	2.80	3.10	3.20	128.92
8	2.880	2.40	2.80	2.90	3.10	3.10	135.88
9	3.234	2.60	3.30	3.20	3.40	3.60	131.03
10	2.967	2.50	2.80	2.90	3.20	3.40	130.62
Mean	**3.4070**	2.94	3.30	3.40	3.58	3.74	**126.23**
Min	2.552	2.20	2.30	2.50	2.70	2.90	115.34
Max	5.207	4.90	5.00	5.40	5.30	5.50	135.88
SD	0.8124	0.84	0.83	0.89	0.77	0.79	7.34
RMSE/Mean	**0.0075**	0.007	0.007	0.008	0.008	0.008	

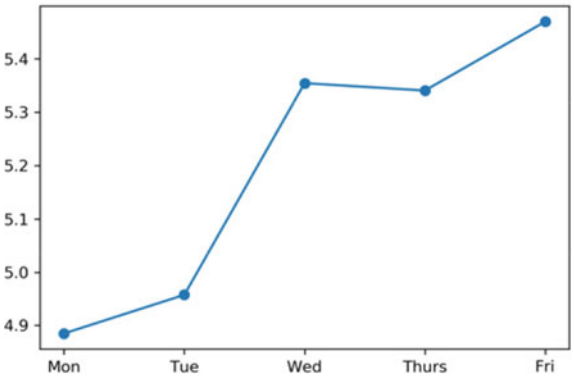

Fig. 8. RMSE of CNN#3 (Round#2 of Table 3)

Table 4. The performance results of the LSTM#1 model

No.	RMSE	Mon	Tue	Wed	Thu	Fri	Time
1	3.245	1.90	3.00	2.90	3.80	4.10	320.96
2	2.374	2.00	2.10	2.20	2.50	3.10	314.90
3	2.459	1.90	2.20	2.20	2.60	3.20	313.82
4	2.518	2.90	2.30	2.30	2.40	2.60	318.80
5	3.533	3.80	3.20	3.10	4.00	3.50	315.70
6	2.423	2.00	2.30	2.40	2.70	2.70	320.46
7	2.869	2.70	2.90	3.00	2.80	2.90	320.59
8	2.066	1.60	1.80	2.00	2.30	2.50	321.66
9	2.759	1.60	2.60	2.80	3.60	2.80	307.76
10	2.582	2.30	2.30	2.20	2.60	3.30	312.77
Mean	**2.6828**	2.27	2.47	2.51	2.93	3.07	**316.74**
Min	2.066	1.60	1.80	2.00	2.30	2.50	307.76
Max	3.533	3.80	3.20	3.10	4.00	4.10	321.66
SD	0.4359	0.69	0.44	0.40	0.62	0.48	4.53
RMSE/Mean	**0.0059**	0.005	0.005	0.006	0.006	0.007	

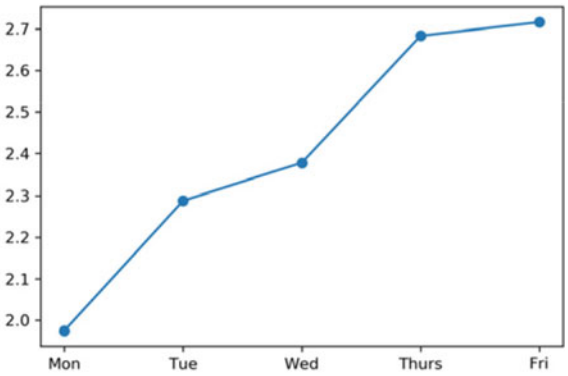

Fig. 9. RMSE of LSTM#1 (Round#6 of Table 4)

Table 5. The performance results of the LSTM#2 model

No.	RMSE	Mon	Tue	Wed	Thu	Fri	Time
1	2.519	2.60	2.30	2.40	2.60	2.80	526.02
2	2.274	1.70	1.80	2.40	2.20	3.00	526.18
3	2.369	1.80	2.20	2.40	2.60	2.70	514.97
4	3.760	3.70	4.20	3.80	3.40	3.50	527.84
5	2.748	2.50	2.90	2.70	2.70	2.90	527.20
6	2.705	1.90	2.30	2.50	2.90	3.70	521.21
7	2.863	2.40	2.50	3.30	2.90	3.10	535.06
8	3.820	4.30	3.10	3.60	4.00	4.10	534.75
9	2.370	1.80	2.20	2.40	2.60	2.80	510.91
10	4.821	5.00	4.10	5.50	5.00	4.40	530.55
Mean	**3.0249**	2.77	2.76	3.10	3.09	3.30	**525.47**
Min	2.274	1.70	1.80	2.40	2.20	2.70	510.91
Max	4.821	5.00	4.20	5.50	5.00	4.40	535.06
SD	0.8355	1.16	0.82	1.00	0.84	0.60	7.84
RMSE/Mean	**0.0067**	0.006	0.006	0.007	0.007	0.007	

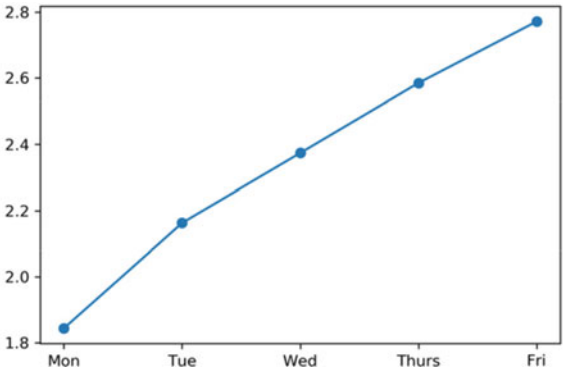

Fig. 10. RMSE of LSTM#2 (Round#9 of Table 5)

actual *open* values for Monday to Friday are found to be 0.005117, 0.006003, 0.006335, 0.006823, and 0.007044, respectively. Figure 11 shows the variations of RMSE of the model as per the round 1 in Table 6.

Table 6. The performance results of the LSTM#3 model

No.	RMSE	Mon	Tue	Wed	Thu	Fri	Time
1	3.914	3.4	3.8	3.9	4.1	4.4	328.42
2	2.819	2.3	2.6	2.8	3.0	3.3	318.36
3	2.210	1.6	2.0	2.2	2.5	2.6	307.65
4	2.584	2.2	2.4	2.6	2.8	2.9	296.16
5	4.542	4.4	4.7	4.7	4.6	4.1	321.79
6	2.857	2.3	2.7	2.9	3.1	3.2	335.01
7	2.591	1.8	2.2	2.6	2.9	3.2	316.05
8	2.448	1.8	2.3	2.5	2.8	2.9	315.37
9	2.283	1.6	2.4	2.2	2.5	2.6	294.91
10	2.215	1.7	2.0	2.2	2.5	2.6	314.96
Mean	**2.8463**	2.31	2.71	2.86	3.08	3.18	**314.87**
Min	2.215	1.60	2.00	2.20	2.50	2.60	294.91
Max	4.542	4.40	4.70	4.70	4.60	4.40	335.01
SD	0.7767	0.91	0.87	0.82	0.71	0.63	12.70
RMSE/Mean	**0.0063**	0.005	0.006	0.006	0.007	0.007	

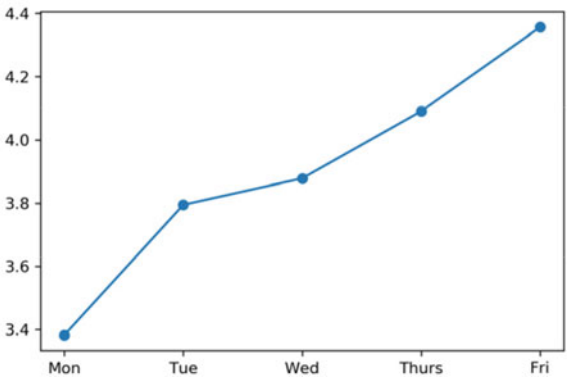

Fig. 11. RMSE of LSTM#3 (Round#1 of Table 6)

Table 7 presents the performance of LSTM#4. The model needs 855.68 s, on an average, for each round. The ratio of the RMSE to the mean *open* value in the test dataset yielded by the model was 0.0114. The ratio of the mean RMSE to the mean

of the actual *open* values for Monday to Friday are found to be 0.010655, 0.011253, 0.011474, 0.011762, and 0.011895, respectively. Figure 12 shows how the RMSE varied over different days in a week for LSTM#4 as per the round 10 in Table 7.

Table 7. The performance results of the LSTM#4 model

No.	RMSE	Mon	Tue	Wed	Thu	Fri	Time
1	2.972	2.60	2.80	3.00	3.20	3.30	854.05
2	5.662	5.50	5.60	5.70	5.80	5.80	843.56
3	5.246	4.80	5.30	5.30	5.40	5.40	836.14
4	3.794	3.50	3.70	3.80	3.90	4.00	838.25
5	5.633	5.30	5.70	5.60	5.80	5.80	831.45
6	4.553	4.30	4.50	4.60	4.70	4.80	812.12
7	7.174	7.00	6.90	7.30	7.30	7.30	855.05
8	8.720	8.10	8.80	8.80	8.90	8.90	912.19
9	4.613	4.40	4.50	4.60	4.70	4.80	909.97
10	3.168	2.60	3.00	3.10	3.40	3.60	864.05
Mean	**5.1535**	4.81	5.08	5.18	5.31	5.37	**855.68**
Min	2.972	2.60	2.80	3.00	3.20	3.30	812.12
Max	8.720	8.10	8.80	8.80	8.90	8.90	912.19
SD	1.7796	1.77	1.82	1.82	1.77	1.72	32.54
RMSE/Mean	**0.0114**	0.011	0.011	0.011	0.012	0.012	

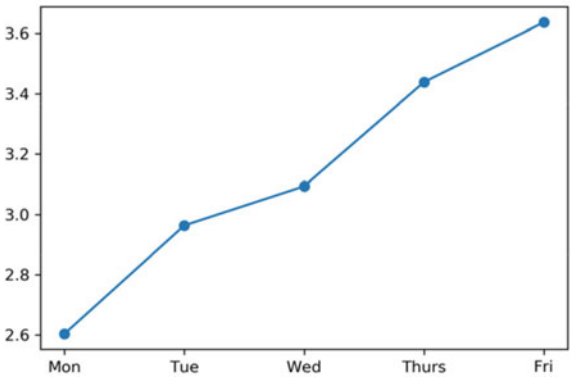

Fig. 12. RMSE of LSTM#4 (Round #10 of Table 7)

Table 8 provides a comparative analysis of all the seven models. The models are ranked based on two metrics: execution time, and the ratio of RMSE to the mean *open* value in the test dataset. It is observed that the CNN #1 model is found to be the fastest

in execution, and the most accurate in its forecasting accuracy. Overall, the CNN models are found to be faster than their corresponding LSTM counterparts.

Table 8. Comparison of different models based on execution time and RMSE

Execution time			RMSE/Mean		
Rank	Model	Value (s)	Rank	Model	Value
1	CNN #1	81.98	1	CNN #1	0.0057
2	CNN #2	84.71	2	LSTM #1	0.0059
3	CNN #3	126.23	3	LSTM #3	0.0063
4	LSTM #3	314,87	4	CNN #2	0.0067
5	LSTM #1	316,74	4	LSTM #2	0.0067
6	LSTM #2	525.47	6	CNN #3	0.0075
7	LSTM #4	855.68	7	LSTM #4	0.0114

5 Conclusion

In this paper, we have presented a suite of the deep learning-based regression model for forecasting stock price on a daily basis for a forecast horizon of one week. While three models are built on CNN architecture, four regression models are designed following LSTM networks. The models were constructed using optimized hyperparameters and then tested on a highly granular stock price data collected at an interval of five minutes. Experimental results showed that while the models exhibited wide divergence in their accuracy and execution speeds, all of them yielded a very high level of accuracy in their forecasting results.

References

1. Sen, J., Datta Chaudhuri, T.: 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. J. Econ. Library **3**(2), 303–326 (2016)
2. Sen, J.: Stock composition of mutual funds and fund style: a time series decomposition approach towards testing for consistency. Int. J. Bus. Forecast. Mark. Intell. **4**(3), 235–292 (2018)
3. Sen, J.: Stock price prediction using machine learning and deep learning frameworks. In: Proceedings of the 6th International Conference on Business Analytics and Intelligence, Bangalore, India, December 20–22 (2018)
4. Mehtab, S., Sen, J.: A robust predictive model for stock price prediction using deep learning and natural language processing. In: Proceedings of the 7th International Conference on Business Analytics and Intelligence, Bangalore, India, December 5–7 (2019)

5. Mehtab, S., Sen, J.: Stock price prediction using convolutional neural network on a multivariate time series. In: Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence (NCMLAI), New Delhi, India (2020)
6. Yan, Z., Huang, Z., Liang, M.: Stock prediction via linear regression and BP regression network. In: Proceedings of Interdisciplinary Symposium on Complex Systems (ISCS'14), pp. 239–247 (2014)
7. Ning, Y., Wah, L.C., Erdan, L.: Stock price prediction based on error correction model and granger causality test. Clust. Comput. **22**, 4849–4858 (2019)
8. Khan, U., Aadil, F., Ghazanfar, M., Khan, S., Metawa, N., Muhammad, K., Mehmood, I., Nam, Y.: A robust regression-based stock exchange forecasting and determination of correlation between stock markets. Sustainability **10**(3702) (2018)
9. Xiao, Y., Xiao, J., Liu, J., Wang, S.: A multiscale modeling approach incorporating ARIMA and ANNs for financial market volatility forecasting. J. Syst. Sci. Complex. **27**(1), 225–236 (2014)
10. Jammalamadaka, S.R., Qui, J., Ning, N.: Predicting a stock portfolio with multivariate Bayesian structural time series model: do news or emotions matter? Int. J. Artif. Intell. **17**(2), 81–104 (2019)
11. Porshnev, A., Redkin, I., Shevchenko, A.: Machine learning in prediction of stock market indicators based on historical data and data from Twitter sentiment analysis. In: Proceedings of the IEEE International Conference on Data Mining Workshops, Dallas, TX, USA (2013)
12. Bao, W., Yue, J., Rao, Y.: A deep learning framework for financial time series using stacked autoencoders and long-and-short-term memory. PLOSE ONE **12**(7) (2017)
13. Mehtab, S., Sen, J., Dutta, A.: Stock price prediction using machine learning and LSTM-based deep learning models. In: Proceedings of the 2nd Symposium on Machine Learning and Metaheuristics Algorithms and Applications, Chennai, India (2020) (Accepted for Publication)
14. Metastock tool: http://www.metastock.com