Applied Soft Computing 134 (2023) 109830

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Novel optimization approach for stock price forecasting using multi-layered sequential LSTM

Abdul Quadir Md [a], Sanjit Kapoor [a], Chris Junni A.V. [a], Arun Kumar Sivaraman [b], Kong Fah Tee [c,*], Sabireen H. [a], Janakiraman N. [d]

[a] School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, 600127, India
[b] Digital Engineering, Solution Center-H, Photo Inc. DLF Cyber City, Chennai, 600089, India
[c] Faculty of Engineering and Quantity Surveying, INTI International University, 71800 Nilai, Malaysia
[d] Department of Electronics and Communication Engineering, K.L.N. College of Engineering, Madurai, Tamil Nadu, India

ARTICLE INFO

Article history:
Received 3 July 2022
Received in revised form 1 November 2022
Accepted 10 November 2022
Available online 15 November 2022

Keywords:
Recurrent neural network
Forecasting
Stock market
Adam optimizer
Long short-term memory

ABSTRACT

Stock markets can often be one of the most volatile places to invest. Statistical analysis of past stock performance and external factors play a major role in the decision to buy or sell stocks. These factors are all used to maximize profits. Stock price index forecasting has been a subject of great research for many years, and several machine learning and deep learning algorithms have been proposed to simplify this complex task, but little success has been found so far. In order to forecast stocks accurately, it is crucial to understand the context-specific dependence of stock prices on their past values. The use of Long Short Term Memory (LSTM), which is capable of understanding long-term data dependencies, can help overcome this obstacle. In this context, this paper proposes a novel optimization approach for stock price prediction that is based on a Multi-Layer Sequential Long Short Term Memory (MLS LSTM) model which makes use of the adam optimizer. Additionally, the MLS LSTM algorithm uses normalized time series data divided into time steps to determine the relationship between past values and future values in order to make accurate predictions. Furthermore, it eliminates the vanishing gradient problem associated with simple recurrent neural networks. The stock price index is forecasted by taking into account past performance information along with past trends and patterns. The results illustrate that a 95.9% prediction accuracy is achieved on the training data set and a 98.1% accuracy on the testing data set with the MLS LSTM algorithm, which dramatically exceeds the performance of other machine learning and deep learning algorithms. The mean absolute percentage error was observed to be 1.79% on the training set and 2.18% on the testing set, respectively. Moreover, the proposed model is able to estimate the stock price with a normalized root mean squared error of 0.019, thus giving an accurate forecast and making it a feasible real-world solution.

1. Introduction

The price of a stock or a share on the stock market is the amount of money it costs to buy one share in a publicly-traded company. This price is rarely constant and fluctuates on a very regular basis. It mainly depends on the company's past performances but also depends on factors like the company's perceived reputation among other things. The demand and supply of the stock also heavily dictate the price of the stock at any given moment. Higher the demand, the higher the value and vice versa. The buying and selling of the stocks are done in the stock market, which allows the buyers and sellers to meet, interact and transact among themselves. Numerous researchers and analysts have been attempting to accurately gauge and evaluate stocks to predict their future trends and value. The analysis that is performed is major of two types: Fundamental Analysis and Technical Analysis. The focus of fundamental analysis is on gathering and processing information from sources that include a company's financial records, economic reports, information about assets, market shares, etc. Focus is largely on the company's financial performance including profitability, liquidity, growth trajectory, etc. These factors are considered in determining how healthy the company is, which in turn gives the analysts an idea about whether or not the company will perform well in the future. A company with healthy finances will perform better in the stock market than a company with less-than-ideal finances. Technical analysis is mainly based on a study of past and present stock prices to predict the future movements of stock prices. Price, volume and supply–demand of the stock are taken into consideration.

* Corresponding author.
E-mail address: kongfah.tee@newinti.edu.my (K.F. Tee).

https://doi.org/10.1016/j.asoc.2022.109830
1568-4946/

The usage of machine learning paradigms in the financial sector, especially the stock market, has been on the rise for a long time. Several algorithms, including Linear Regression, Support Vector Machines (SVM), Auto-Regressive Integrated Moving Average (ARIMA), etc. have been used with varying degrees of accuracy. One of the major obstacles in using simple machine learning paradigms such as regression for stock price prediction is that the prices of stocks can be highly random or can be represented as complex time series models. This means that the past data of the stock prices often dictates the future prices of the stock, which regression and SVM models do not take into account. This problem can be solved by the use of a deep learning paradigm that incorporates the historical price of the stock into the prediction of the future value, such as the LSTM technique. LSTM allows for the learning of long-term dependencies in the data to perform highly accurate predictions. In this study, the usage of the Multi-Layered Sequential LSTM technique for the prediction of stock prices is proposed. The LSTM technique is a kind of Recurrent Neural Network (RNN) which has the capacity to learn from and use past data to predict future outcomes. Conventional RNN is not able to handle long-term dependencies in data. There is also no fine control over what context in data is moved ahead and what amount of the past context needs to be forgotten by the network. Another huge drawback of conventional RNN is the problem of vanishing gradient. The gradient loss function of a neural network approaches zero as more layers using certain activation functions are added to it. This is known as a vanishing gradient and it makes the network hard to train. All of these shortcomings are overcome by LSTM networks. LSTM networks have the ability to selectively remember patterns for long sequences. This allows it to learn the dependency of the data on its past values and perform the prediction more accurately. LSTM models also allow fine control over selective forgetting of the past context of the data and also completely eliminate the problem of vanishing gradient. These advantages provided by the LSTM networks make it a suitable candidate for the prediction of future stock prices.

The remainder of the paper is arranged as follows: Section 2 outlines the work in the area of stock price prediction. Section 3 introduces the concept of LSTM and describes its function. Section 4 presents the proposed architecture and the methodology for the estimation of stock prices. In Section 5, the performance metrics and the experimental results are discussed and compared with similar work done with different techniques. The paper concludes with future directions in Section 6.

2. Related work

Several works that use Machine Learning techniques to predict stock prices exist in the public domain for us to review. In [1] the author has made use of a simple linear regression to forecast the price of the S&P index of 500. The data used in the predictive analysis consisted of 7 columns that include low, high, date, volume, close, adjusted close and open in total. The authors normalized the data first and then based on the number of features selected, built two linear regression models [2]. Model 1 made use of all 7 features of the dataset, whereas, in Model 2, the author omitted 2 features, namely Volume and Adjusted Close. Both models performed very well, with the first model having an R^2 of 0.997 and the second model having an R^2 score of 0.992. Despite both models having very good performance metrics, it must be noted that according to the author, S&P 500 data consists of the value of 500 stocks and it was chosen because it is sensitive and predictable. Therefore the model may not perform when the dataset is changed to something less predictable. The authors in [3] performed a predictive analysis of stock prices using Linear Regression and Support Vector Machine (SVM) [4] models. The analysis was performed using R language with RStudio being the development environment. The authors used the Application Programming Interface (API) package provided by R to fetch the stock data of The Coca-Cola Company. The data was for exactly 1 year, ranging from January 2017 to 2018. The models were built using R's built-in lm() command for Linear Regression and the svm() command found in the liquid SVM package for SVM. The R^2 value of the Linear Regression model was observed to be 0.73 and that of the SVM model was 0.93. It shows that SVM performs significantly better than the LR model and has high accuracy.

In [5] the authors performed a comprehensive study of the usefulness of time series modelling (ARIMA) in stock forecasting. The data used in the study was that of fifty-six companies from seven different sectors and was obtained from the official website of the National Stock Exchange (NSE) India. Twenty-three months of training data from April 2012 to February 2014 was used for the prediction of the stock prices for the next month. ARIMA model was used in the predictive analysis. The results obtained showed that the accuracy of the ARIMA model in stock price prediction was above 85% for all the sectors. When looking at the sectors individually, it can be seen that the FMCG sector prediction had the highest accuracy but the Banking sector prediction accuracy was relatively lower. The work done in [6] helps in deciding which RNN (recurrent neural network) is best suited for the task. Three variants of RNN are tested. They are VRNN (Vanilla RNN), LSTM and Gated Recurrent Unit (GRU). GRUs, LSTM, RNN and CNN provide exceptional results in stock price forecasting as shown in [7–10]. A general rule of thumb is to use the simpler model as long as it does the job as simpler models perform well over time. As the name suggests VRNN is one of the most basic or naive RNNs. The major drawback of VRNN is that it suffers from the vanishing gradient problem, which in turn causes the gradient of the loss function to decay over time rendering the model ineffective. LSTM hopes to prevent loss function decay and is able to learn dependencies in long term. This makes LSTM an excellent option for time series analysis. GRU is LSTM but simpler and shows slightly better results when compared to LSTM but that is due to the fact that the dataset used is smaller and hence the potential of the long-term dependencies is shrouded. LSTM is chosen because of its ability to work well with large datasets.

In [11], the author uses Neural Network, LSTM to forecast the stock estimates. Stock prices of big corporations such as Apple, Google, etc. have been used for the prediction but the time frame of the stock price data is not mentioned explicitly. The author has reported an accuracy of 94% while using an Artificial Neural Network and an accuracy of 97% while using the LSTM algorithm. Although the reported accuracy for the models used is very high, the author has mentioned very little about the dataset used to perform the analysis and also the methodologies involved in building the model. The authors of [12] use LSTM to predict stock prices of two stocks on the NYSE with 15 years of stock prices taken from Yahoo Finance. The authors use different numbers of epochs for training to see at what number of epochs the LSTM successfully predicts the stock prices with decent accuracy. They have concluded that more epochs and less training data are the best way to go and also mention that the LSTM is a very capable model when it comes to predicting stock prices. In [13] the authors made use of four different techniques using deep learning for the forecast of prices of stocks from the New York and the National stock exchange. The authors trained LSTM, Multilayer Perceptron (MLP) [14], CNN and RNN using the stock prices of a company from the National Stock Exchange (NSE). The models were then used to predict the stock prices of five separate companies that were picked from the NSE and the NYSE. The authors compared their results with the ARIMA model

and showed that specialized neural network models outperform existing linear models like ARIMA. In [15], a comprehensive comparison between the performance of a LSTM and a MLP approach in the short-term prediction of stock prices is presented. The focus is on granular movements of the stock price and thus the data contains minute-by-minute stock prices for all trading days over the span of 1 year of 10 stocks that are listed on the NYSE.

In [16] the author performs an investigative study to analyse the effectiveness of networks of LSTMs for the purpose of predicting stock prices. The author tabulated the RMSE test error of LSTM networks with 50, 100, 250 and 500 hidden layers and proved that usage of LSTM networks is an effective method of predicting stock prices mainly because LSTM networks are resistant to overfitting and perform better when larger networks are joined together. The authors of [17] predict the prices of Chinese stocks with the help of LSTM. The authors claim that the reason behind choosing Chinese stocks [18] for prediction is that they are much more unpredictable. The data used for prediction contained 10 learning features and was transformed into 30-day-long sequences. According to the authors, the LSTM model used by them enhanced the accurate forecast of stock prices by 27.2% in contrast to the approach of random prediction. A comparison between grid and manual search techniques for initialization of hyper-parameters for neural networks was done in [19] and the authors concluded that the manual search technique is capable of finding as good or even better models than the grid search technique and in less time. The authors have suggested the use of a manual search technique for judging the hyper-parameters of neural network models. In [20] the authors perform experiments to find out the optimal number of hidden layers of a LSTM model that is used for stock price prediction. Stock price data from four companies — Maruti, ICICI, Reliance and TCS were used for training and prediction of the LSTM approach. The hidden layers of the LSTM models used in the experiment varied from one to seven. The authors observed that the LSTM model with $n = 1$ hidden layer performed the best, with a RMSE score of 5.85, while the RMSE score for $n = 2$ to $n = 7$ rose from 8.86 to 5.89, suggesting that one hidden layer must be used for stock price prediction as it yields higher accuracy.

Based on the literature survey, it can be concluded that deep learning paradigms perform better than simple machine learning paradigms, and thus are better suited for the task of stock price prediction. This can be attributed to the fact that deep learning systems can automatically learn complex mappings from inputs and outputs. The usage of deep learning algorithms such as RNN leads to the obstacle of vanishing gradient which also needs to be overcome. Thus, we can conclude that using LSTM is the best way to forecast stock prices, despite its drawbacks, which include high computational power and resource requirement along with huge training times. With time-series data, LSTM is very helpful in overcoming the problem of vanishing gradients using backpropagation. When combined with minmax normalization, LSTM is the best technique for accurately predicting stock prices while also avoiding any bias that may arise from outliers.

3. Long short-term memory

LSTM networks are considered to be the most effective solution to sequence prediction problems. The most advantageous property of LSTM is that it can remember patterns for a large number of sequences. This gives LSTM an advantage over standard feed-forward NN and RNN, which lack the ability to use past observations to predict future values. A typical LSTM network contains multiple memory blocks that are also called cells. Each cell transfers two states to its subsequent cell; the hidden and cell state as shown in Algorithm 1. The remembering of information

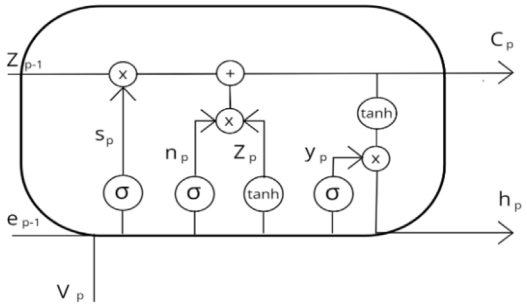

Fig. 1. LSTM cell.

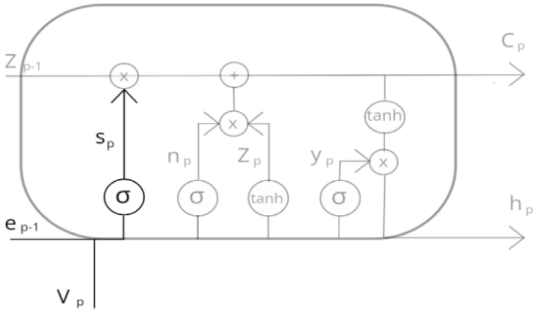

Fig. 2. Forget gate of LSTM cell.

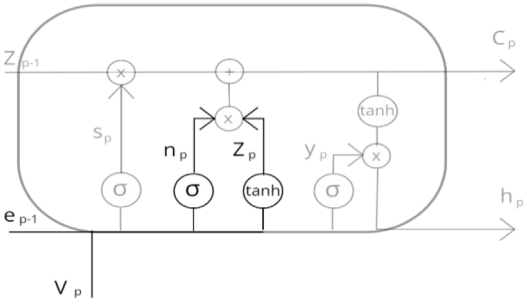

Fig. 3. Input gate of LSTM cell.

is done by the memory blocks. Manipulations to this memory are performed by three major structures that are known as gates. The illustrations of an LSTM cell along with every gate (namely, forget gate, input gate and output gate) are shown below in Figs. 1–4 and the roles played by them along with Eqs. (1)–(5) are discussed.

Forget Gate: This is the first gate in the LSTM cell and its function is to decide whether or not the information from the previous stamp is to be retained or not. The information from the present input V_p and hidden state e_{p-1} are taken and a sigmoid function is applied which makes its output between 0 and 1, after which it is multiplied by the cell state of the previous timestamp. If the final value is 0, then it forgets everything and if it is 1, then it forgets nothing. The weights are represented by D and bias is represented by g.

$$s_p = \sigma(D_s \times [e_{p-1}, V_p] + g_s) \tag{1}$$

Input Gate: In the input gate, another sigmoid function is applied to the current state V_p and hidden state e_{p-1} and a value between 0 and 1 is obtained, after which the tanh function is applied to it. It then takes the values of vector input n_p and carries out the addition in step-wise, after which, the cell state is updated

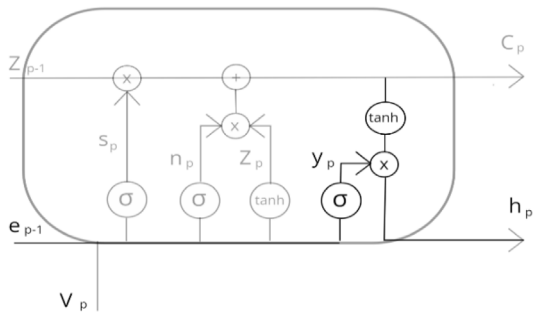

Fig. 4. Output gate of LSTM cell.

to a new cell state C.

$$n_p = \sigma(D_n \times [e_{p-1}, V_p] + g_n) \tag{2}$$

$$Z_p = tanh(D_Z \times [e_{p-1}, V_p] + g_Z) \tag{3}$$

Output Gate: A third sigmoid function is applied to the current state and previous hidden state and the new cell state generated in the Input Gate is passed through the tanh function, both of these outputs are multiplied point by point and decide what information the hidden state for the next cycle will carry. The hidden state, along with the new cell state, is carried over to the next step.

$$y_p = \sigma\left(D_y \times [e_{p-1}, V_p] + g_y\right) \tag{4}$$

$$e_p = y_p \times tanh(Z_p) \tag{5}$$

The abovementioned structure of the LSTM cell allows for backpropagation of weights and errors, which provides for fine-grained tuning of weights and biases, leading to greatly accurate predictions. It also helps in tackling the problems of vanishing and booming gradients.

Algorithm 1: LSTM cell computation algorithm

Input: V_p

Output: y_p

Initialize: $e_{p-1}, D_s, D_n, D_y, g_s, g_n, g_y$

while i less than time_steps **do**

Step 1: Forget gate of LSTM

$$s_p = \sigma(D_s \times [e_{p-1}, V_p] + g_s)$$

Step 2: Input gate of LSTM

$$n_p = \sigma(D_n \times [e_{p-1}, V_p] + g_n)$$

$$Z_p = tanh(D_Z \times [e_{p-1}, V_p] + g_Z)$$

Step 3: Output gate of LSTM

$$e_p = y_p \times tanh(Z_p)$$

$$y_p = \sigma\left(D_y \times [e_{p-1}, V_p] + g_y\right)$$

end while

4. Proposed methodology of multi-layer sequential long short-term memory

MLS LSTM consists of two major parts namely data preparation and sequential LSTM as shown in Fig. 5. The sequential LSTM consists of 4 layers out of which 3 layers are vanilla LSTM and the other layer is Dense. A high number of layers lead to better results but require greatly larger computational power, and thus a tradeoff between performance and resource use was reached with 4 layers. The close column is sliced from the main dataset and is used as the input for the MLS LSTM after being checked for any null values.

MLS LSTM much like any other LSTM, needs its data fed as time steps. Based on the empirical observations, a time step of 100 data points is utilized and a 65:35 train-to-test split is adopted. Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE) and R-squared measures are used for evaluating MLS LSTM's performance. Algorithm 2 represents the construction of the MLS LSTM model.

4.1. Importing data

The required dataset is downloaded from Yahoo Finances. The dataset in use contains the stock information of Samsung dating from 2016 to 2021. The dataset contains the low, high, date, volume, close, adjusted close and open.

- The Date column contains the stock price date
- The Open column contains information about the price at which the stock opened on the day
- The High column represents the highest price the stock touched on the day
- The Low column gives the lowest price of the stock on the day
- The Close column gives information about the price at which the stock was closed
- Adj Close value adjusts for the corporate activities
- The volume column gives the number of shares of the companies that were transacted on the given day.

Of the abovementioned fields, the columns of interest are the Close column and the Date column. This is because date-wise training and prediction of the closing stock price are to be performed using the proposed MLS LSTM model.

4.2. Cleaning data

The first major step is to handle missing data. Each row of the dataset is scanned to check for missing values. It is observed that the columns Open, High, Low, Close, Adj and Volume have 5 NA values each. The NA values are filled using the fillna function of Pandas library. The number of missing values in each column is then visualized using the library 'missingno'. In the bar plot that is plotted in Fig. 6, it can be seen that the Date column has 1227 rows with proper values whereas all the other rows only have 1222 rows with proper values (because of the 5 NA values in each row).

Since the stock prices of stocks do not differ massively from one day to the next, the values from the previous rows can be used to fill the NA values without adding inaccuracies to the data. This method is preferred over other using mean or median values to replace the null values because it requires far less computational power while also being relatively accurate. For this purpose, the pad method of the fillna function is used to fill the NA values with the values from the previous rows.

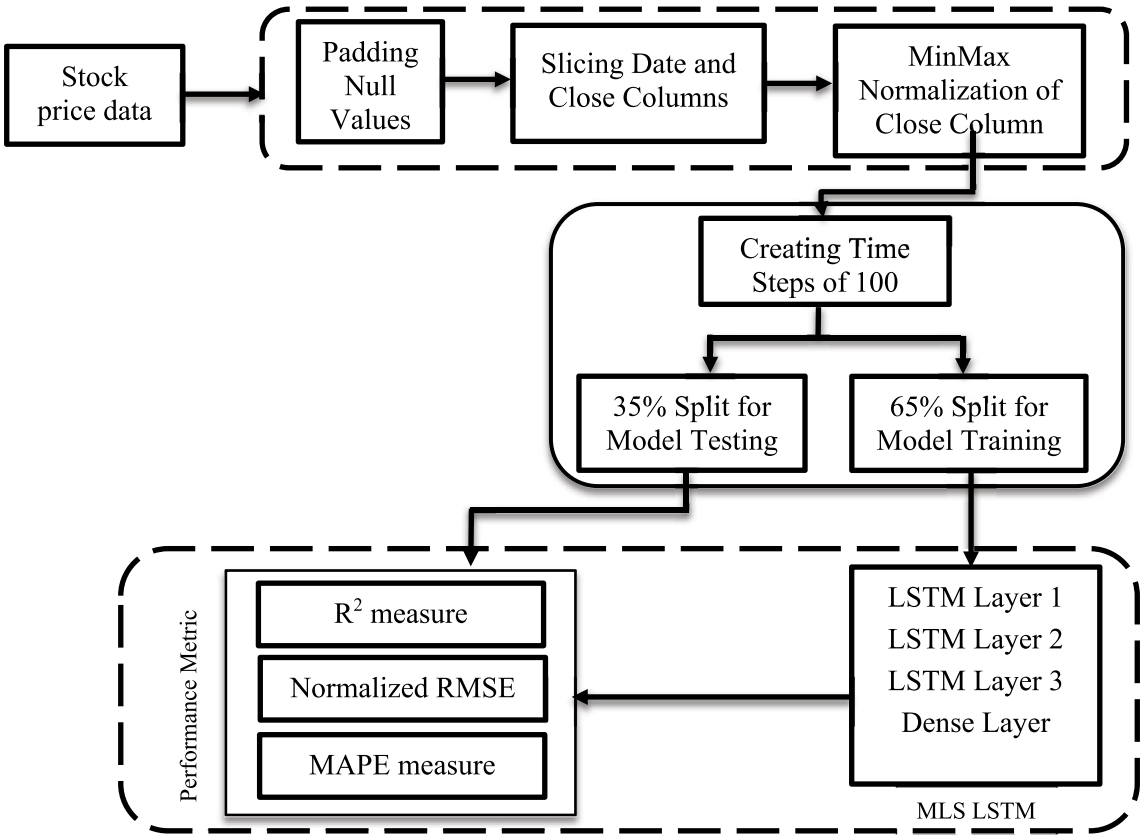

Fig. 5. Proposed architecture of multi-layer sequential long short-term memory.

Table 1
Stock closing price with date.

Date	Close
2017–11–24	55460
2018–03–19	50740
2019–01–17	41950
2019–07–04	46000
2020–12–04	71500

4.3. Visualizing data

The next step is the plotting of the closing price of the stock over the time period of the dataset. The date is plotted along the *x*-axis and the closing price of the stock along the *y*-axis. Fig. 7 shows the line plot which gives a fair idea about the progression of the value of the Samsung stock over time starting from November 2016 to November 2021. Next, the maximum and minimum closing prices of the stock along with the date of the valuation are extracted from the data and five random values are shown in Table 1.

It is observed that the lowest price of the stock was 32980 on 23-11-2016, which is the starting point of the chosen dataset. The maximum valuation of the stock was observed on 01-11-2021 with a closing price of 91,000, which is a little under three times the starting price (also the lowest valuation) of the Samsung stock.

4.4. Data normalization

After the visualization of the dataset, feature scaling is performed on it. Feature scaling helps in normalizing all the data in the dataset using Eq. (6) which may fall between different ranges and converts them into a new range of any given choice, which is usually 0 to 1. If the data is not normalized, it may lead to large gradient error values which will result in very large, and thus unstable, weight values for the LSTM model. To normalize the data, MinMaxScaler is used, which is available in the sklearn preprocessing library. It helps in scaling all the columns of the data between the value of 0 and 1. MinMax scaler is chosen for this purpose because it preserves the shape of the dataset, without any distortion. The transformation is given by the following equation:

$$Xscaled = \frac{(X - Xmin)}{(Xmax - Xmin)} \tag{6}$$

In the next step of the process, the data is split into training and testing sets. The data is split into the above-mentioned parts in the ratio of 65:35. It implies that 65% of the data is used for training the LSTM model and 35% is used for testing.

4.5. Reshaping data

To make the data suitable for the LSTM model, it is converted into time steps itself. The step value is taken to be 100. After that, the testing and training data are reshaped to fit the LSTM model that is going to be built. The reshape function takes three parameters, namely, the sample size, the time step and the number of features. For the proposed LSTM algorithm, the sample size is the number of rows in the training set, which is 329. The time step, as has already been mentioned, is 100. Finally, the feature size of the data is going to be 1, because only one feature, the closing price of the stock, is being predicted.

Algorithm 2: Building MLS (Multi-Layered Sequential) LSTM model

Step 1: *Data Collection*
Read stock_price

Step 2: *Data Preparation*
For each value in data:
If value==null:
Replace_null(method='pad')
Separate 'Date' and 'Close' columns
MinMaxScaler(data)
Split data into Training and Testing set
Reshape_data(timestep=100)

Step 3: *Build Prediction Model*
Add layers to LSTM
Layer 1: return_sequence = True, input_size = (100,1)
Layer 2: return_sequence = True
Layer 3: return_sequence = False
Layer 4: Dense layer (1)
Loss function = mean-squared error
Optimizer = adam

Step 4: *Train Model*
Fit Training data on model
epochs = 100
batch_size = 64

Step 5: *Check Accuracy*
R2_score()
root_mean_squared_error()
mean_absolute_percentage_error()

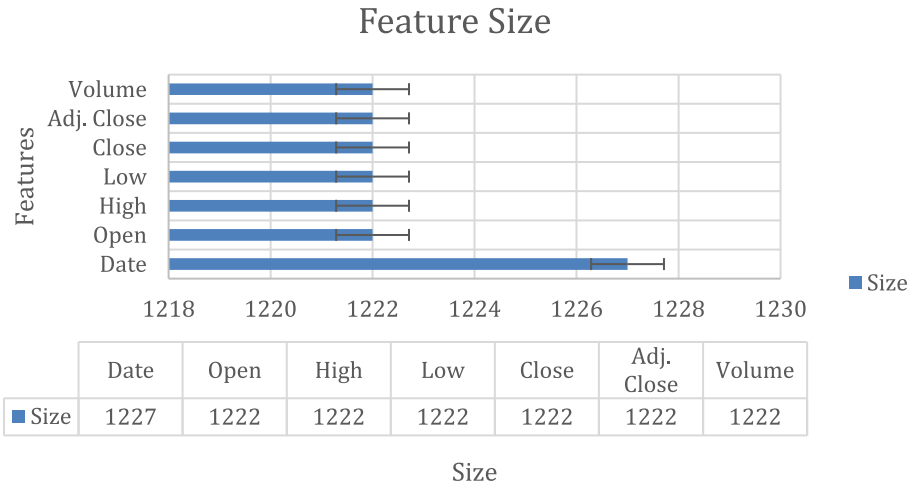

Fig. 6. Feature size of the data set.

4.6. Building MLS LSTM model

The next step is building the MLS LSTM model for the prediction of stock prices. The type of model is set to sequential, which means that the data is fed to all the layers of the LSTM model in a sequential way. The MLS LSTM consists of 4 layers, which was found to be the optimal number of layers for accurate predictions while avoiding overfitting. The first layer of the model is the LSTM model that is imported from the Keras library. The input shape of the layer is defined as (100,1) because of the chosen time-step of 100. It signifies that each input is of size 100 and there is 1 feature column that is being passed as the input. The number of LSTM cells is set as 50. The second layer of the model is also an LSTM layer of shape (50) and in the same way, the third layer is also an LSTM layer with shape (50). The final layer is the dense layer of shape (1). This indicates that all the output layers of the previous layer are densely connected (fully connected) to the output layer of the model. This results in a single output from all the cells of the LSTM cells. The optimizer used is adam, the working of which is illustrated in Algorithm 3.

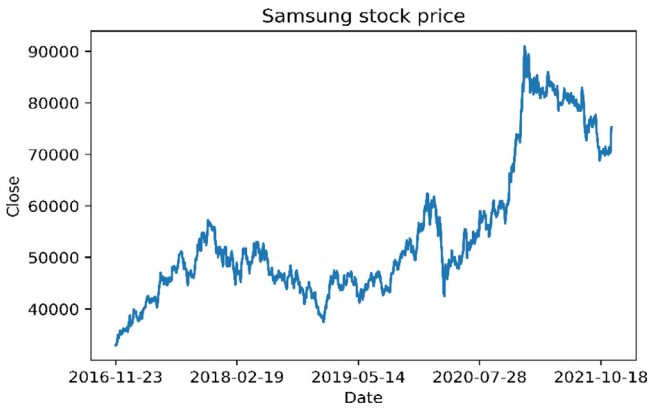

Fig. 7. Samsung stock price over a time period.

4.7. Model compilation and fitting

After the model is built, it is compiled with a batch size of 64 and an epoch size of 100. It signifies that the data is grouped together into batches of 64 and fed to the LSTM model. 100 epochs mean that the data is fed to the model 100 times. The validation split is 0.1, which means 10% of the training data is reserved for the validation of the model. After the training, predictions are made using both, training and testing data. The predicted data needs to be normalized back to the normal scale, which is done using the inverse_transform() function. After the data is inverse-scaled, it can be compared with the original data to obtain the accuracy measures. The accuracy measures used to gauge the performance of the model are the R^2 score, RMSE score and MAPE.

5. Experimental setup and results

The experiments were carried out on an i7-8750H Intel Core CPU with a 2.2 GHz machine with 8 GB ram and were implemented in Python 3.9.8 using Jupyter Notebook server 6.4.5. The dataset used in the experiment is that of Samsung's stock prices recorded between 23 November 2016 and 23 November 2021, ranging over five years. The dataset was obtained from the website of Yahoo Finances. Out of the seven columns in the dataset, only the Date column and the Closing price column are selected for further steps. The missing values are handled by replacement with the values from the previous row. For efficient training of the model, the data is normalized using the MinMax scaling technique. The training set is then split into 7 sets, while the testing set is divided into 10 sets. Both the training and testing sets are converted into time steps of 100. The input data is then reshaped as [samples, time steps, features]. Here, samples represent the number of rows in the data, the time step is 100 and features represent the number of features in every time step, which is 1.

For the prediction, a sequential LSTM model comprising three stacked LSTM layers is used. This is done with the help of Keras deep learning API that runs on top of TensorFlow. Finally, a dense layer is added to make the output a single value. During the compilation of the model, the loss function that is used is mean squared error and the optimizer is adam. The training of the model is performed for 100 epochs with 64 being the batch size and 0.1 being the validation split. The predictions are made on training and testing data using the compiled model. After the predictions are made, the original values and the predicted values are inverse-transformed back to their original scale for evaluation of their performance.

Table 2
Performance metrics.

Metric	Training data	Testing data
R^2	0.959	0.981
Adjusted R^2	0.958	0.979
Norm. RMSE	0.019	0.028
MAPE	1.79	2.18

5.1. Performance evaluation

The training as well as the testing data are used for the forecasting of the stock prices and are compared with the original values. The first metric used to measure the accuracy of the proposed algorithm is the R^2 score, which is defined as the proportion of the variance in the dependent variable that is predictable from the independent variables. R^2 is given by Eq. (7):

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2} \tag{7}$$

The R^2 score of the proposed model on training data was calculated to be 0.959 and that on testing data was calculated to be 0.981, which means that it performed with an accuracy of 95.9% on training data and with an accuracy of 98.1% on testing data. Adjusted R^2 takes into consideration the number of terms used in a model. It penalizes the addition of independent variables for prediction, which means that the Adjusted R^2 score decreases if useless variables are added. Adjusted R^2 can never be more than R^2, it is always less than or equal to R^2. The adjusted R^2 is calculated by the given Eq. (8):

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(n - 1)}{n - k - 1}\right] \tag{8}$$

The adjusted R^2 of the proposed MLS LSTM model is 0.958 on the training data and 0.979 on the testing data. As a second metric for measuring the performance of the model, the normalized root mean square error score (RMSE) is used, which is a value between 0 and 1 with 0 being the most favourable value. The formula to calculate the RMSE score is shown in Eq. (9). The Normalized RMSE score of the model on train data is 0.019 and on test data is 0.028.

$$Norm.RMSE = \frac{\sqrt{\frac{\sum(\hat{y}_i - y_i)^2}{n}}}{ymax - ymin} \tag{9}$$

The Mean Absolute Percentage Error (MAPE) gives information about how accurate a prediction model is. It is measured in percentage, and its value can be found by calculating the average modulus of the actual value of observation minus forecasted observation divided by the actual value of observation as shown in Eq. (10).

$$MAPE = \frac{1}{n}\sum_{1}^{n}\frac{A_t - F_t}{A_t} \tag{10}$$

The performance metrics, namely R^2, Adjusted R^2, Normalized RMSE and MAPE for the MLS LSTM are computed for training and testing data respectively, which are illustrated in Table 2. The values are then plotted in Fig. 8.

The actual stock price on a given date and the predicted stock price are tabulated in Table 3 and visualized in Fig. 9 to highlight the accuracy of forecasting by MLS LSTM. It can be observed that there is a very high overlap between the actual stock price and the stock prices predicted by the model, which means that the predictions are highly accurate.

Algorithm 3: Adam optimizer

Constrain: α: size of step

Constrain: $\eta_1, \eta_2 \in [0, 1)$: fractional decline proportions for the estimates moment

Constrain: $\boldsymbol{x}(\Theta)$: Stochastic function objective (parameter Θ)

Parameter Initialization : Θ_0:

$r_0 = 0$ (First moment)
$q_0 = 0$ (Second moment)
$j = 0$ (Time step)

while Θ_t not converged **do**

$j = j+1$
$u_j = \nabla_\Theta x_t (\Theta_{j-1})$ (Acquire gradients: at time step j for stochastic objective)
$r_j = \eta_t * r_{t-1} + (1- \eta_1) * u_j$ (Updating the biased - 1st moment approximation)
$q_j = \eta_2 * q_{j-1} + (1- \eta_2) * u_j^2$ (Updating the biased – 2nd raw moment approximation)
$\bar{r}_j = r_j / (1- \eta_1^j)$ (Calculate corrected bias - 1st moment approximation)
$\bar{q}_j = q_j / (1- \eta_2^j)$ (Calculate corrected bias – 2nd raw moment approximation)
$\Theta_j = \Theta_{j-1} - \alpha * \bar{r}_j / (\sqrt{\bar{q}_j} + \varepsilon)$ (parameters update)

end while
return Θ_j (parameter Outcome)

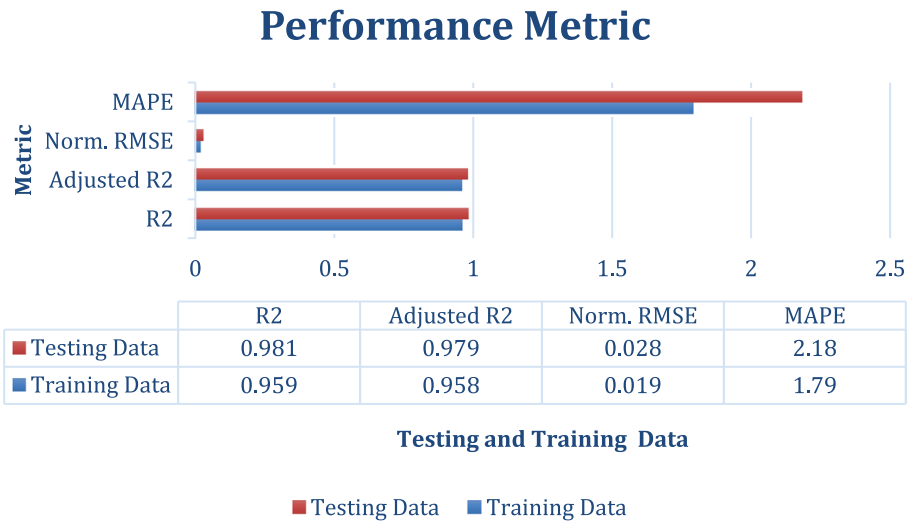

Fig. 8. Performance metrics.

Table 3
Comparison of actual price with the predicted price.

Actual value	Predicted value
48251	47580
48093	47420
50664	49900
44504	44000
47034	45100

Table 4
Training data R^2 comparison of different prediction techniques at multiple epochs.

Epoch count	Linear regression	SVM	MLS LSTM
5	0.39	0.39	0.42
30	0.60	0.75	0.79
45	0.64	0.80	0.83
75	0.69	0.88	0.90
100	0.73	0.93	0.95

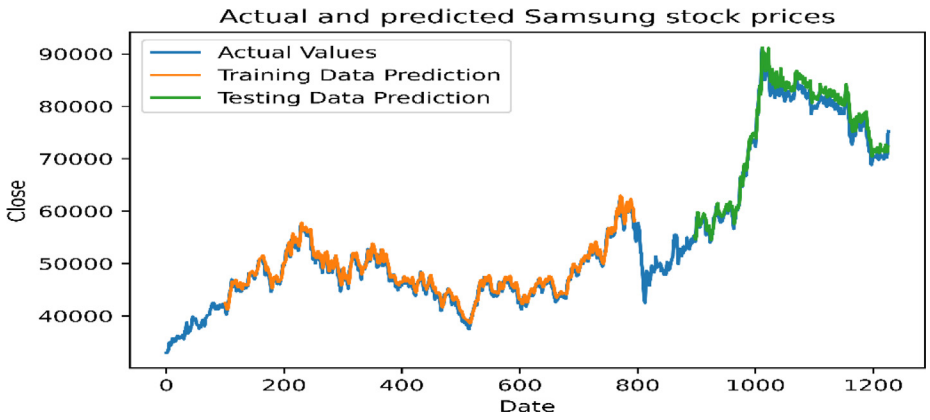

Fig. 9. Actual and predicted values of Samsung stock price.

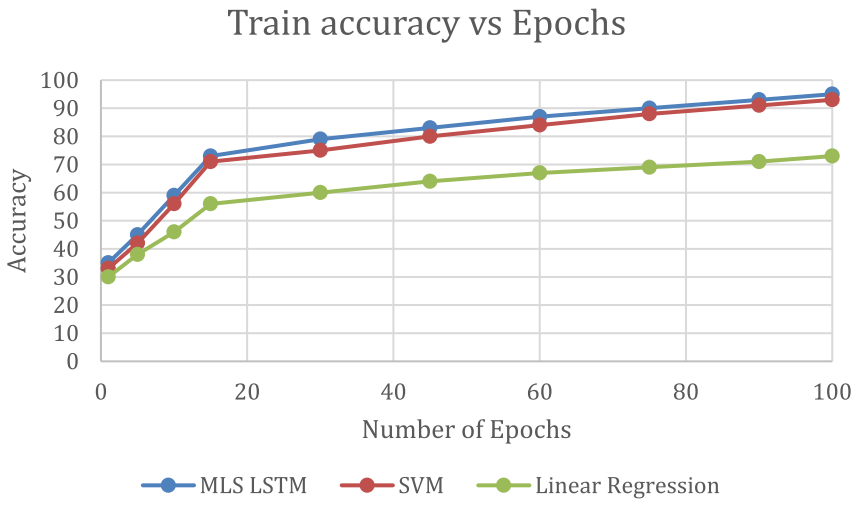

Fig. 10. Training data R^2 comparison.

Table 5
Testing data R^2 comparison of different prediction techniques at multiple epochs.

Epoch count	Linear regression	SVM	MLS LSTM
5	0.45	0.42	0.38
30	0.58	0.77	0.77
45	0.61	0.81	0.84
75	0.68	0.87	0.93
100	0.72	0.92	0.98

5.2. Performance comparison

In [2], the authors have used linear regression and support vector machines for the prediction of stock prices. The comparison of the R^2 score of the linear regression and the support vector machine model is performed with the proposed MLS LSTM model in Tables 4 and 5. The R^2 of the linear regression model trained by the authors has an R^2 score of 0.73.

The SVM model is better performing, with an R^2 score of 0.93. But the proposed MLS LSTM approach outperforms both, the regression model and the SVM model, with an R^2 score of 0.98. The epoch-wise values are plotted in Figs. 10 and 11. Based on this, it can be shown that MLS LSTM can produce more precise stock price predictions and predict values with higher accuracy when compared to simple machine learning approaches such as SVM and linear regression.

In [7], the authors made use of four different deep neural networks, namely LSTM, MLP, RNN and CNN to compare the performance of each of the techniques on three different stocks (Maruti, HCL and Axis Bank) for the duration of 400 days. To determine and compare the proposed MLS LSTM performance with theirs, the average MAPE value of the three stocks obtained by the authors is compared.

Table 6
MAPE comparison of different prediction techniques at multiple epochs.

Epoch count	RNN	LSTM	CNN	MLP	MLS LSTM
15	8.49	6.31	5.32	5.49	3.11
30	8.32	6.12	5.19	5.35	2.97
45	8.17	5.99	5.04	5.21	2.79
60	8.01	5.83	4.93	5.08	2.52
75	7.88	5.69	4.78	4.93	2.39
90	7.73	5.55	4.67	4.82	2.27
100	7.62	5.47	4.55	4.72	2.18

It can be observed that RNN has the largest MAPE value out of all the deep learning algorithms, which can be attributed to its inability to account for long-term dependencies in data. It is also observed that the MAPE value of the proposed MLS LSTM

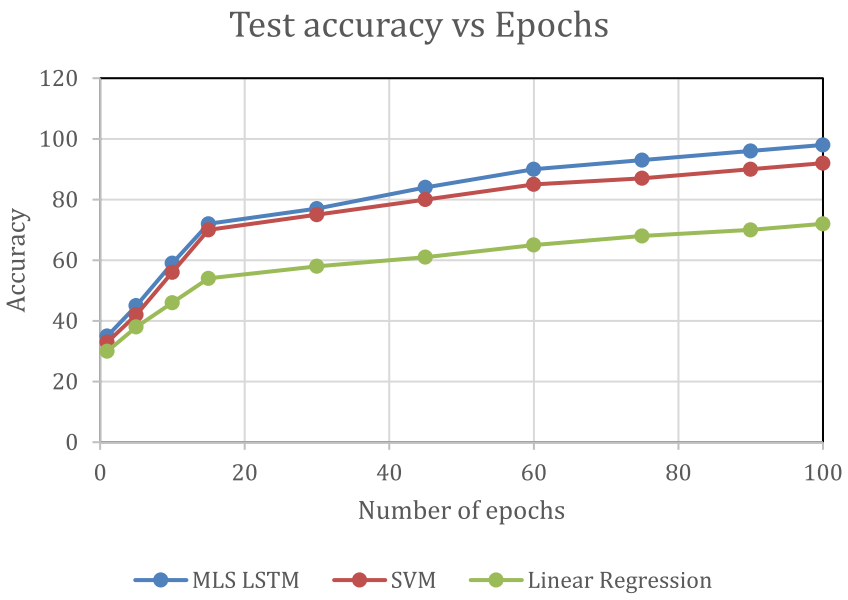

Fig. 11. Testing data R^2 comparison.

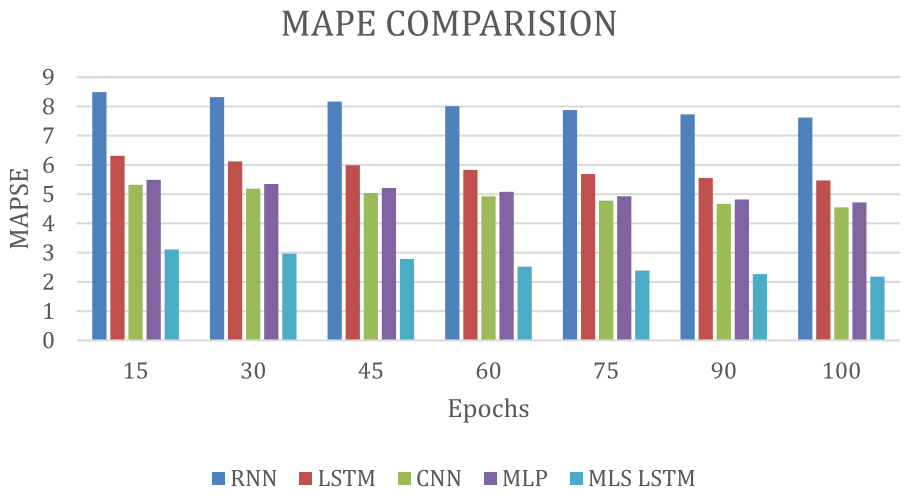

Fig. 12. MAPE comparison.

model is lower than any of the other deep learning architectures, with a mean error percentage of only 2.18% on the testing data. It can be noted that the MAPE of the proposed MLS LSTM is lower than a single-layer LSTM, thus showing that a multi-layered LSTM performs better than a single-layered LSTM. This can be attributed to the fact that each additional layer of the LSTM fine-tunes the weights to reach the optimally accurate solution. The MAPE values are illustrated in Table 6 and visualized in Fig. 12.

6. Conclusion and future work

The paper proposes a new approach for stock price prediction based on a Multi-Layer Sequential Long Short Term Memory (MLS LSTM) model that makes use of the Adam optimizer. As a result of the MLS LSTM algorithm, normalized time series data is divided into time steps to determine the relationship between past and future values for accurate prediction. In the testing data set, MLS LSTM achieved a prediction accuracy of 95.9% and 98.1% on the testing data set, outperforming every other deep learning and machine learning algorithm currently available. The average error percentage with which the proposed model is able to predict the values is 2.18%, which is considerably low, thus giving very accurate predictions. MLS LSTM is a better approach also because more layers in the model reduce the generality of the results.

In an effort to build a more efficient and reliable prediction system in the future, fundamental analysis and sentiment analysis will be included in the prediction process. By doing this, we will be able to gauge the general sentiment of the public regarding the company, which will allow us to tweak the prediction technique based on the sentiment and make even more accurate predictions in the future. Changing the way the missing data is handled, that is, replacing null values with running averages or median of a set range of data can also help increase training accuracy. An effort to reduce the training time as well as resource requirements of LSTM must be made. The problem of overfitting must also be taken care of, as LSTM can be sensitive to it.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] R. Seethalakshmi, Analysis of stock market predictor variables using linear regression, Int. J. Pure Appl. Math. 119 (15) (2018) 369–378.

[2] F.S. Gharehchopogh, T.H. Bonab, S.R. Khaze, A linear regression approach to prediction of stock market trading volume: a case study, Int. J. Manag. Value Supply Chains 4 (3) (2013) 25.

[3] V. Gururaj, V.R. Shriya, K. Ashwini, Stock market prediction using linear regression and support vector machines, Int. J. Appl. Eng. Res. 14 (8) (2019) 1931–1934.

[4] N.I. Sapankevych, R. Sankar, Time series prediction using support vector machines: a survey, IEEE Comput. Intell. Mag. 4 (2) (2009) 24–38.

[5] P. Mondal, L. Shit, S. Goswami, Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices, Int. J. Comput. Sci. Eng. Appl. 4 (2) (2014) 13.

[6] A.S. Saud, S. Shakya, Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE, Procedia Comput. Sci. 167 (2020) 788–798.

[7] M.O. Rahman, M.S. Hossain, T.S. Junaid, M.S.A. Forhad, M.K. Hossen, Predicting prices of stock market using gated recurrent units (GRUs) neural networks, Int. J. Comput. Sci. Netw. Secur. 19 (1) (2019) 213–222.

[8] M. Roondiwala, H. Patel, S. Varma, Predicting stock prices using LSTM, Int. J. Sci. Res. (IJSR) 6 (4) (2017) 1754–1756.

[9] K. Pawar, R.S. Jalem, V. Tiwari, Stock market price prediction using LSTM RNN, in: Emerging Trends in Expert Applications and Security, Springer, Singapore, 2019, pp. 493–503.

[10] E. Hoseinzade, S. Haratizadeh, CNNpred: CNN-based stock market prediction using a diverse set of variables, Expert Syst. Appl. 129 (2019) 273–285.

[11] K. Pothuganti, Long short-term memory (LSTM) algorithm based prediction of stock market exchange, Int. J. Res. Publ. Rev. 2 (1) (2021) 90–93.

[12] A. Moghar, M. Hamiche, Stock market prediction using LSTM recurrent neural network, Procedia Comput. Sci. 170 (2020) 1168–1173.

[13] M. Hiransha, E.A. Gopalakrishnan, V.K. Menon, K.P. Soman, NSE stock market prediction using deep-learning models, Procedia Comput. Sci. 132 (2018) 1351–1362.

[14] A.V. Devadoss, T.A.A. Ligori, Forecasting of stock prices using multi layer perceptron, Int. J. Comput. Algorithm 2 (1) (2013) 440–449.

[15] K. Khare, O. Darekar, P. Gupta, V.Z. Attar, Short term stock price prediction using deep learning, in: 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology, RTEICT, IEEE, 2017, pp. 482–486.

[16] H. Jia, Investigation into the effectiveness of long short term memory networks for stock price prediction, 2016, arXiv preprint arXiv:1603.07893.

[17] K. Chen, Y. Zhou, F. Dai, A LSTM-based method for stock returns prediction: A case study of China stock market, in: 2015 IEEE International Conference on Big Data, big data, IEEE, 2015, pp. 2823–2824.

[18] F. Jiang, G. Tang, G. Zhou, Firm characteristics and Chinese stocks, J. Manage. Sci. Eng. 3 (4) (2018) 259–283.

[19] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012).

[20] A. Yadav, C.K. Jha, A. Sharan, Optimizing LSTM for time series prediction in Indian stock market, Procedia Comput. Sci. 167 (2020) 2091–2100.