Stock price prediction with Machine Learning: Insights from Logistic Regression and LSTM Model

Vraj Patel[1], Shyam Sakhiya[2], Mayuri Popat[3]

[1,2,3]*U & P.U Patel Department of Computer Engineering, CSPIT, Charusat University of Science and Technology, Anand GJ 388421, INDIA*

22CE102@charusat.edu.in, 22CE113@charusat.edu.in, mayuripopat.ce@charusat.ac.in

Abstract

Predicting the stock market is one of the most challenging tasks in the field of computation. Numerous factors contribute to this complexity—physical versus psychological factors, rational and irrational behavior, investor sentiment, market rumors, and more. These elements combine to make stock prices highly volatile and difficult to predict accurately. We investigate data analysis as a potential game-changer in this domain. According to the Efficient Market Theory, when all information related to a company and stock market events is instantly available to all stakeholders and market investors, the effects of those events are already reflected in the stock price. Consequently, it is said that only the historical spot price carries the impact of all other market events and can be utilized to predict future movement. Therefore, by considering past stock prices as the cumulative outcome of all influencing factors, we employ Machine Learning (ML) techniques on historical stock price data to infer future trends. ML techniques have the potential to reveal patterns and insights previously unseen, which can be used to make highly accurate predictions. We propose a framework utilizing an LSTM (Long Short-Term Memory) model and a company net growth calculation algorithm to analyze and predict future growth trends for a company.

1. Introduction

Data analysis has been used across all businesses for data-driven decision-making. In the stock market, numerous factors drive share prices, and the pattern of price changes is not consistent. This makes it difficult to make robust decisions about future prices. Artificial Neural Networks (ANN) have the capability to learn from past data and make

decisions about the future. Deep learning networks such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are particularly effective with multivariate time series data. We train our model on historical stock data to predict future stock prices. This future price is then used to calculate a company's projected growth. Furthermore, we generate a future growth curve for various companies, allowing us to analyze and investigate similarities in growth trajectories between companies.

The stock price of a listed company on a stock exchange fluctuates every time an order is placed to sell or buy, and a transaction is completed. An exchange collects all sell bids with expected prices per stock (usually higher than the purchase price) and all buy bids with or without a price limit (investors typically expect the future price of the stock to exceed the current price). A buy-sell transaction occurs when both bids match, i.e., when the selling bid price equals the buying bid price.

Macro-economic conditions also play an important role in the growth or decline of an entire sector. Some intrinsic factors may include a company’s net profit, liabilities, demand stability, market competition, technological advancements, surplus cash for adverse situations, stakes in raw material suppliers, and relationships with distributors of finished products. Extrinsic factors, which are beyond the control of the company, include aspects such as crude oil prices, exchange rates, political stability, and government policy decisions.

Many researchers have attempted to use historical stock prices as the foundation for time series analysis to forecast future stock prices. Various statistical models have been applied, including moving average (MA), autoregression (AR), weighted moving average, ARIMA, and CARIMA. Deep neural networks like CNN and RNN are also used with different parameter settings and features. In this paper, we explore a specialized RNN known as LSTM to predict future company growth based on past stock prices.

2. Indian Stock Market Overview

Almost every country has one or more stock exchanges where shares of listed companies can be bought or sold. This forms a secondary marketplace. When a company first lists itself on a stock exchange to

become a public company, the promoter group sells a substantial amount of shares to the public, following government regulations. During the incorporation of a company, shares are bought by promoter groups or institutional investors in a primary market. Once the promoter offloads a major portion of the shares to public retail investors, these shares can then be traded in the secondary market, i.e., stock exchanges.

In India, the BSE (Bombay Stock Exchange) and the NSE (National Stock Exchange) are the two most active stock exchanges. The BSE has around 5,000 listed companies, whereas the NSE has around 1,600. Both exchanges have similar trading mechanisms, opening and closing times, and settlement processes. Stock exchanges enable individual investors to participate in the share market and allow them to buy even a single share of a listed company with the help of a trading account and a demat account.

These online markets have transformed the Indian investment landscape, supported by government initiatives like tax benefits on equity investments and the National Pension Scheme (NPS), which also invests in the stock market. Due to continuous reductions in bank interest rates and rising inflation, middle-class investors are increasingly shifting from the security of fixed deposits to the equity market. All these factors have contributed to the growth in capitalization of both exchanges.

3. Related Studies

There is extensive research in stock market prediction and LSTM. Almost every data mining and prediction technique has been applied to forecast stock prices, utilizing a variety of features and attributes. Stock market analysis and prediction can be divided into three main categories: (a) Fundamental analysis, (b) Technical analysis, and (c) Time series analysis. Most stock forecasting techniques with time series data typically use either linear models, such as AR, MA, ARIMA, ARMA, CARIMA, or non-linear models, such as ARCH, GARCH, ANN, RNN, and LSTM.

Some researchers have analyzed various macroeconomic factors that affect share price movement, such as crude oil prices, exchange rates, gold prices, bank interest rates, and political stability, by designing a data warehouse. Other researchers have employed frequent itemset mining techniques to find lagged correlations between price movements across

different sectoral indices in the Indian share market. Roondiwala and colleagues used an RNN-LSTM model on NIFTY-50 stocks with four features (high, close, open, and low prices of each day). They used a 21-day window to predict the next day's price movement, with a total of five years of data for prediction, and minimized error using RMSE with backpropagation.

Kim and colleagues proposed a model known as the "feature fusion long short-term memory-convolutional neural network (LSTM-CNN) model." They used CNN to learn features from stock chart images and found that candlestick charts are the best candidates for predicting future stock price movements. They then employed LSTM, fed with historical price data, and tested on minute-wise stock prices using a 30-minute sliding window to forecast the 35th minute price. They used S&P 500 ETF data with stock price and trade volume, employing CNN and LSTM individually on different representations of the same data. This is akin to examining the same object from different angles to gain new insights.

Their unique approach involved training models using data from a single company and applying those models to predict future prices of five different stocks from the NSE and NYSE (New York Stock Exchange). They argued that linear models attempt to fit the data to the model, but deep networks can uncover the underlying dynamics of stock prices. According to their results, CNN outperformed all other models, as well as classical linear models. The deep neural networks (DNN) could even forecast prices for NYSE-listed companies, although the model was trained on NSE data. This might be attributed to similar inner dynamics in both stock exchanges.

This simplicity is one reason why the GRU model has been gaining popularity. These are by no means an exhaustive list of modified LSTMs. Other variants include Depth Gated LSTMs by Yao and "Clockwork RNNs" by Koutnik, which address long-term dependencies in a completely different manner.

4. LSTM Architecture

4.1 An overview of Recurrent Neural Network (RNN)

In a classical neural network, final outputs seldom act as an output for the next step but if we pay attention to a real-world phenomenon, we observe that in many situations our final output depends not only the external

inputs but also on earlier output. For example, when humans read a book, understanding of each sentence depends not only on the current list of words but also on the understanding of the previous sentence or on the context that is created using past sentences. Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. This concept of 'context' or 'persistence' is not available with classical neural networks. Inability to use context-based reasoning becomes a major limitation of traditional neural network. Recurrent neural networks (RNN) are conceptualized to alleviate this limitation. RNN are networked with feedback loops within to allow persistence of information. The Figure 1Error! Reference source not found. shows a simple RNN with a feedback loop and its unrolled equivalent version side by side.

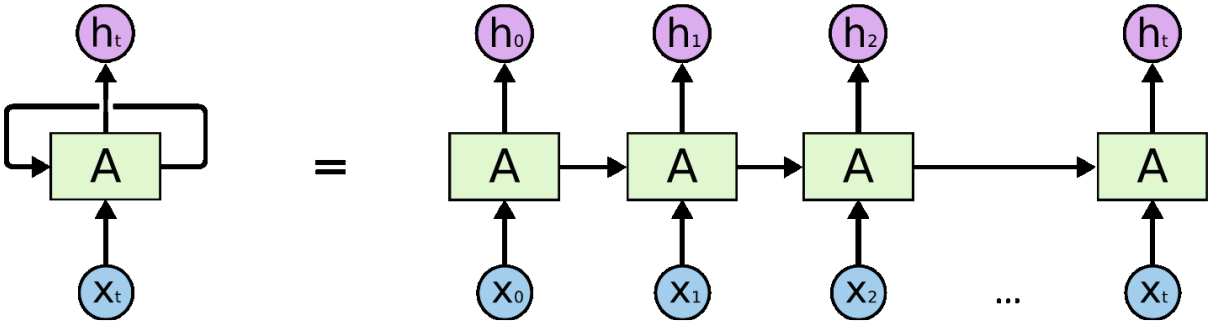

Figure 1: An unrolled recurrent neural network

Initially (at time step t) for some input Xt the RNN generates an output of ht. In the next time step (t+1) the RNN takes two input Xt+1 and ht to generate the output ht+1. A loop allows information to be passed from one step of the network to the next.

RNNs are not free from limitations though. When the 'context' is from near past it works great towards the correct output. But when an RNN has to depend on a distant 'context' (i.e. something learned long past) to produce correct output, it fails miserably. This limitation of the RNNs was discussed in great detail by Hochreiter [8] and Bengio, et al. [9]. They also traced back to the fundamental aspects to understand why RNNs may not work in long-term scenarios. The good news is that the LSTMs are designed to overcome the above problem.

4.2 LSTM Networks

Hochreiter and Schmidhuber introduced a special type of RNN that is capable of learning long-term dependencies. Many researchers later improved upon this pioneering work. LSTMs have been perfected over

time to address the long-term dependency issue. The evolution and development of LSTM from RNNs are explained in various studies.

Recurrent neural networks are structured as a chain of repeating modules within the neural network. In standard RNNs, this repeating module has a simple structure, often involving a single tanh layer, as shown in Figure 2.

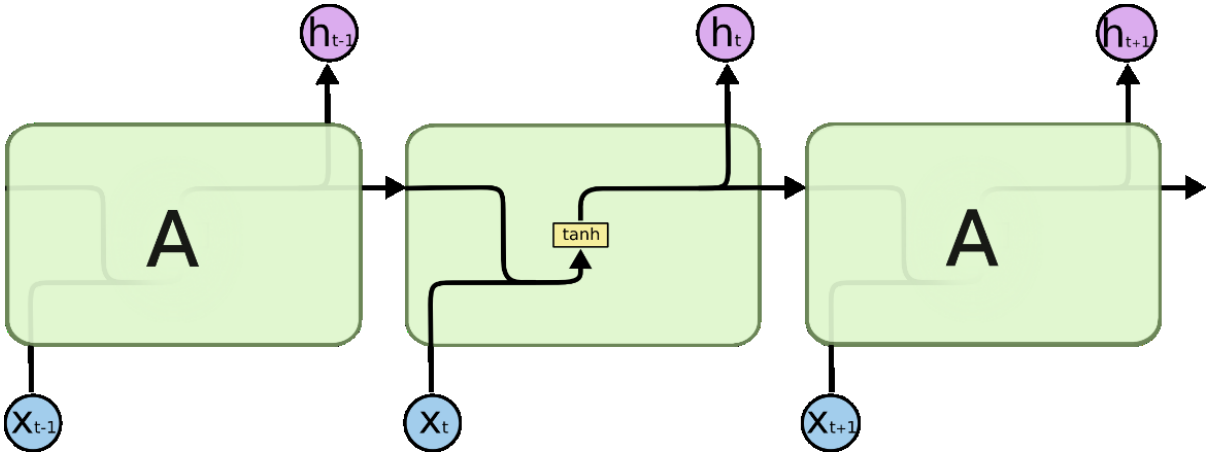

Figure 2:The repeating module in a standard RNN contains a single layer

LSTMs follow this chain-like structure, however the repeating module has a different structure. Instead of having a single neural network layer, there are four layers, interacting in a very special way as shown in Figure 3.

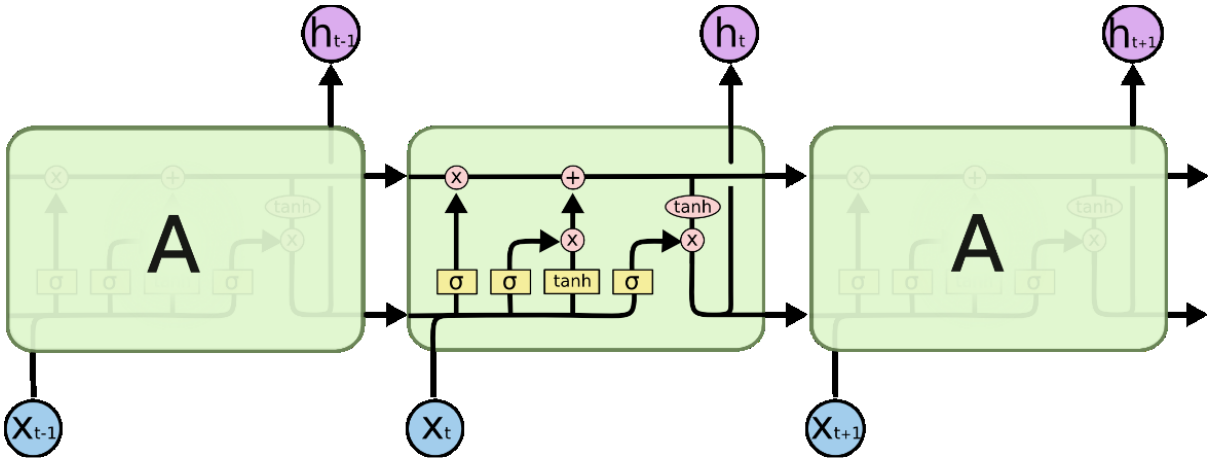

Figure 3:The repeating module in an LSTM contains four interacting layers

In Figure 3, every line represents an entire feature vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

4.3 The Working of LSTM

The key to LSTMs is the cell state, represented by a horizontal line running through the top of the diagram. The cell state is like a conveyor belt, running straight down the entire chain with some minor linear interactions. LSTMs have the ability to add or remove information to the cell state, controlled by structures called gates. Gates are used to selectively let information through, and they are composed of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between 0 and 1, indicating how much of each component should be allowed through. A value of 0 means “let nothing through,” while a value of 1 means “let everything through.” An LSTM has three of these gates to protect and control the cell state.

The first step of an LSTM is to decide what information should be discarded from the cell state, which is managed by a sigmoid layer called the “forget gate layer.” This layer looks at ht−1h_{t-1}ht−1 and xtx_txt and outputs a number between 0 and 1 for each value in the cell state Ct−1C_{t-1}Ct−1. A value of 1 means “completely keep this,” while a value of 0 means “completely remove this.”

The next step is to decide what new information will be stored in the cell state. This process has two parts. First, a sigmoid layer called the “input gate layer” decides which values will be updated. Then, a tanh layer creates a vector of new candidate values, C~t\tilde{C}_tC~t, which could be added to the state. These two outputs are combined to create an update to the cell state.

At this point, the old cell state, Ct−1C_{t-1}Ct−1, is updated into the new cell state, CtC_tCt. This update is achieved by multiplying the old state by ftf_tft, then adding it∗C~ti_t \ast \tilde{C}_tit∗C~t. These are the new candidate values, scaled by how much we decide to update each state value.

Finally, the output is determined, providing a filtered version of the cell state. First, a sigmoid layer determines which parts of the cell state will be output. Then, the cell state is passed through a tanh function (to constrain the values between -1 and 1) and multiplied by the output of the sigmoid gate, so only the parts we decided to output are included.

5. Proposed Framework to Forecast Share Price & Company Growth in Different Time Span

In this section, we shall first analyze some existing techniques and their merits to finally arrive at our methodology. Next, we shall discuss the algorithmic and implementation steps in detail. It is implemented in Python.

5.1 Analyzing Different Methods

Regression is one of the popular way to do the prediction of share prices. In figure 4 two figures on TCS share price using linear regression & polynomial regression of degree four are shown.

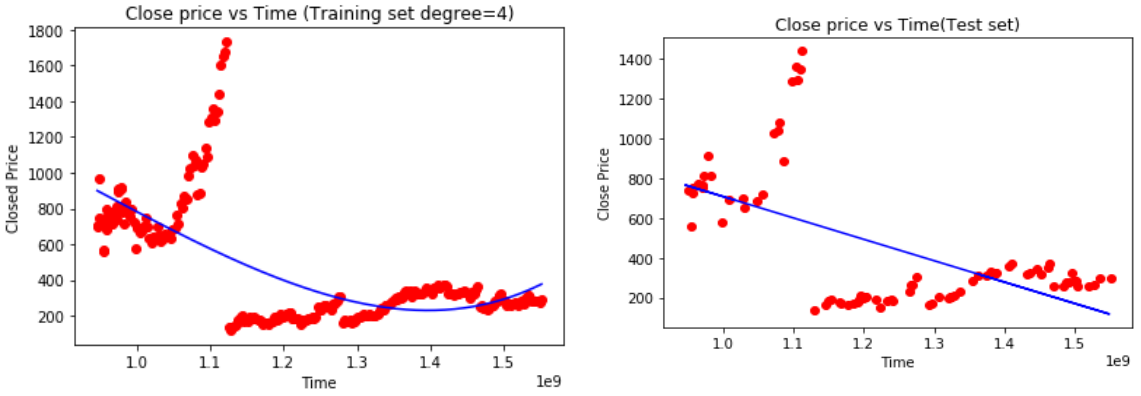

Figure 4: stock market closing prices of TCS over a time period and polynomial(degree 4) regression line

Regression is found not to be very much useful here to compute the error values. Also, we found a problem with curve fitting. The above graphs are showing a poor result in terms of curve fitting. This has a clear justification. For time series data, such as text, signals, stock prices, etc. LSTM is better suited to learn temporal patterns in deep neural networks. An LSTM solves the 'vanishing gradient' problem that exists in a RNN while learning long-term dependencies with time series dataset with the use of memory cell (states) and (input and forget) gates. So, LSTM may be a better option for future prediction of the company's share price as well as growth.

5.2 Methodology

The purpose of our framework is to analyze which is the best time span to predict the future share price of a company from a particular sector. Our objective is to predict the future price and calculate the future growth of the company in the different time span. Then we analyze the prediction

error for each company of different sector. Based on that we conclude which time span is best for future prediction of that particular sector.
We first predict the future closing price of 5 different companies from some pre-decided sectors with the help of LSTM. This prediction will be done on historical data & the future prediction will be done for 3-month, 6-month, 1 year & 3 years. In these four different time spans (3 & 6 months, 1 & 3 years), we calculate the growth of those companies. Then by analyzing the deviations of closing price for each time span, we took the resultant time span which has maximum growth, i.e. less error for the particular sector, e.g. companies A, B, C, D & E from a sector S1 has more growth in 3-months' time span of prediction then we draw an conclusion that for sector S1, our framework gives the best prediction for next 3-months for that particular sector. In our analysis, let's consider we are using the data for *M*onths. Then the weight of a company is defined as:
weight = 1/ (P ∗(P+1)/2)
In our case, month-wise weight (Yi) will be calculated using the following algorithm:
N:= M weight := 1/(M(M+1)/2) FOR i = 1 to M Begin Yi := weight * N ; /* Yi is the weight of previous ith month*/ Q = Q - 1; i := i + 1 End End FOR*
Suppose the growth rate between different time periods is *Gri* where i=1 to M, considering current year as 0th year. Therefore, *Gri* is the growth rate of (*i*-1)*th* time period w.r.t its immediate earlier year i.e. *ith* year. To maximize the impact of current growth over the growth of older year, we would develop a mathematical formula stated below. Suppose the growth rates of a company are *Gr*1*; Gr*2…*Grm* respectively from present to *M* years earlier.
Then the Company Net Growth Rate (CNGR) by the following formula.
*CNGRj= Y*1∗ *Gr*1 + *Y*2∗ *Gr*2 + · · · + *Yi*∗*Gri*+ · · · + *Yp*∗*Grm*
Where CNGRj is the Company Net Growth Rate of the *jth* company (where j=1 to m)

5.3 Implementation Steps

Step1: *Raw Stock Price Dataset:* Day-wise past stock prices of selected companies are collected from the BSE (Bombay Stock Exchange) official website.
Step2: *Pre-processing*: This step incorporates the following:

a) Data discretization: Part of data reduction but with particular importance, especially for numerical data
b) Data transformation: Normalization.
c) Data cleaning: Fill in missing values.
d) Data integration: Integration of data files. After the dataset is transformed into a clean dataset, the dataset is divided into training and testing sets so as to evaluate. Creating a data structure with 60 timesteps and 1 output.

Step3*: Feature Selection:* In this step, data attributes are chosen that are going to be fed to the neural network.In this study Date & Close Price are chosen as selected features.

Step4: *Train the NN model*: The NN model is trained by feeding the training dataset. The model is initiated using random weights and biases. Proposed LSTM model consists of a sequential input layer followed by 3 LSTM layers and then a dense layer with activation. The output layer again consists of a dense layer with a linear activation function.

Step5*: Output Generation:* The RNN generated output is compared with the target values and error difference is calculated. The Backpropagation algorithm is used to minimize the error difference by adjusting the biases and weights of the neural network.

Step6*: Test Dataset Update:* Step 2 is repeated for the test data set.

Step7: *Error and companies' net growth calculation:* By calculating deviation we check the percentage of error of our prediction with respect to actual price.

Step8: *Visualization*: Using Keras and their function APIs the prediction is visualized.

Step9: Investigate different time interval: We repeated this process to predict the price at different time intervals. For our case, we took 2-month dataset as training to predict 3-month, 6-month, 1 year & 3 years of close price of the share. In this different time span, we calculate the percentage of error in the future prediction. This would be different for different sectors. So, this will help to find a frame for the particular sector to predict future companies' net growth.

6 Results

The proposed LSTM based model is implemented using Python. In Table 1 the Error value for different companies belong to Banking Sector based on the historical data of 1 month, 3 month, 6 month, 1 Year, 3 Year span is shown.

Table 1: Error Value for Different Banks

Bank Names	1 month	3 month	6 month	1 year	3 year
SBI	93.30438	9.371283	19.5584	5.148866	0.830179
HDFC	532.8527	523.4962	162.8642	24.40721	0.987856
ICICI	71.80286	9.881709	10.76914	4.575525	0.863681
Avg Error	232.6533	180.9164	64.39726	11.3772	0.893905

Table 2: Error Value for Different Sectors

Sector	1 month	3 month	6 month	1 year	3 year
IT	39.56394	8.049353	1.48794	1.840666	0.782617
Pharma	250.7862	94.87654	29.48869	7.358529	0.903381
FMCG	426.7132	134.2102	60.45957	11.9643	0.874805
Aviation	291.025	35.08927	36.90103	30.97042	0.944595
Bank	232.6533	180.9164	64.39726	11.3772	0.893905

In the same way calculation is done for other sectors also based on the top level companies belong to that sector. The error values for the sector is shown in Table 2.

It has been observed from the result that for almost all the sectors the error level comes down drastically with the test data for longer periods. So we suggest to apply this LSTM based model to predict the share price on long time historical data.

7. Conclusions

In this paper, we analyze the growth of the companies from different sector and try to find out which is the best time span for predicting the future price of the share. So, this draws an important conclusion that companies from a certain sector have the same dependencies as well as

the same growth rate. The prediction can be more accurate if the model will train with a greater number of data set.
Moreover, in the case of prediction of various shares, there may be some scope of specific business analysis. We can study the different pattern of the share price of different sectors and can analyze a graph with more different time span to fine tune the accuracy. This framework broadly helps in market analysis and prediction of growth of different companies in different time spans. Incorporating other parameters (e.g. investor sentiment, election outcome, geopolitical stability) that are not directly correlated with the closing price may improve the prediction accuracy.

7. References

[1]F. a. o. Eugene, "Efficient capital markets: a review of theory and empirical work," Journal of finance, vol. 25, no. 2, pp. 383-417, 1970.

[2]Z. A. Farhath, B. Arputhamary and L. Arockiam, "A Survey on ARIMA Forecasting Using Time Series Model," Int. J. Comput. Sci. Mobile Comput, vol. 5, pp. 104-109, 2016.

[3]S. Wichaidit and S. Kittitornkun, "Predicting SET50 stock prices using CARIMA (cross correlation ARIMA)," in 2015 International Computer Science and Engineering Conference (ICSEC), IEEE, 2015, pp. 1-4.

[4]D. Mondal, G. Maji, T. Goto, N. C. Debnath and S. Sen, "A Data Warehouse Based Modelling Technique for Stock Market Analysis," International Journal of Engineering & Technology, vol. 3, no. 13, pp. 165-170, 2018.

[5]G. Maji, S. Sen and A. Sarkar, "Share Market Sectoral Indices Movement Forecast with Lagged Correlation and Association Rule Mining," in International Conference on Computer Information Systems and Industrial Management, Bialystok, Poland, Sprigner, 2017, pp. 327-340.

[6] M. Roondiwala, H. Patel and S. Varma, "Predicting stock prices using LSTM," International Journal of Science and Research (IJSR), vol. 6, no. 4, pp. 1754-1756, 2017.

[7]T. Kim and H. Y. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," PloS one, vol. 14, no. 2, p. e0212320, April 2019.

[8]S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in International Conference on Advances in Computing, Communications and Informatics, 2017.

[9]S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen," Diploma, Technische Universität München, vol. 91, no. 1, 1991.

[10]Y. Bengio, P. Simard, P. Frasconi and others, "Learning long-term dependencies with gradient descent is difficult," IEEE transactions on neural networks, vol. 5, no. 2, pp. 157-166, 1994.

[11]S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in Advances in neural information processing systems, NIPS, 1997, pp. 473--479.

[12]S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 6, no. 2, pp. 107-116, 1998.

[13]J. Schmidhuber, D. Wierstra, M. Gagliolo and F. Gomez, "Training recurrent networks by evolino," Neural computation, vol. 19, no. 3, pp. 757-779, 2007.

[14]L. Pasa and A. Sperduti, "Pre-training of recurrent neural networks via linear autoencoders," in Advances in Neural Information Processing Systems, NIPS, 2014, pp. 3572-3580.

[15]J. Chen and N. S. Chaudhari, "Segmented-memory recurrent neural networks," IEEE transactions on neural networks, vol. 20, no. 8, pp. 1267-1280, 2009.

[16]S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[17]R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction, MIT Press, 2018.