Stock Market Prediction Using LSTM

**Abstract:**

The trend of investing in stocks of different companies is increasing day by day. But it is very difficult to predict the stock price of any asset as a lot of factors affect it and some of the factors are natural and unknown. With every passing day interest in Machine Learning and Artificial Intelligence is increasing and they have also been useful for human beings as Artificial Intelligence has taken over a lot of work. This Paper tends to use LSTM stacked Recurrent Neural Networks to create a model for predicting stock market prices and analyzing the model and its accuracy for different values of considered error. The model in this paper gives promising results by giving an accuracy of 90.44% for a considered error of 5%.

**Keywords:** Stock market Prediction; RNN; LSTM; Forecasting; multivariate;

**Introduction:**

There has been a significant increase in people’s interest in financial activities like stock trading and cryptocurrency trading. We will hardly find any educated person who is unaware of the different financial activities and opportunities present in today’s world. These stock and crypto tradings can be a good source of income if done wisely. To buy or sell any share we must have a good idea about the current market and the previous value of the stock. The stock price depends on a lot of factors that must be analyzed carefully before investing the money. Studying the patterns and the various factors affecting stock price has caught the attention of a lot of machine learning enthusiasts and researchers. If we can have a model that can give us a good idea about the opening price of any stock then it can be really useful for us to invest accordingly because the financial data that we are having is too difficult to analyze and making the predictions on the basis of that data is a more tedious task as a lot of factors should be considered before we make prediction [1].

Earlier many economists have built and tried a lot of economic models to study the patterns and researchers have also tried a lot of machine learning algorithms to study the pattern and solve this time series forecasting problem. There have been a lot of algorithms like Artificial Neural Networks(ANN), Convolutional Neural Networks(CNN), Decision Support Systems (DSS), Naïve Bayes, Recurrent Neural Networks(RNN), Support Vector Machine(SVM), Hidden Markov Model(HMM) that have been used for making the predictions of the stock market but LSTM has shown much better results with low risks of losses[1][2]. There are also some univariate models like Autoregressive Integrated Moving Average (ARIMA) used for predicting stock markets but they get outperformed by the multivariate Deep Learning Models [3] which fit the dynamic nature of Stock Market more accurately. ANNs have been good approximators but rather than simple feed-forward networks RNNs perform much better than them [6] because future Stock Price depends on the previous stock prices and we must always consider the effect of past stock prices before predicting future prices. RNN carries out this work by using the back propagation algorithm. The performance of RNN is further improved by Long-Short-Term-Memory (LSTM).

This paper consists of majorly 4 sections:

Section 1 contains information about other related works that had been done in this area. It discusses the various algorithms that have been earlier used for this problem. Section 2 contains the theoretical overview of the algorithm i.e. LSTM stacked RNN. It discusses the need for LSTM and the working of LSTM. Section 3 contains information about the dataset used for training and testing our model. It discusses the work that has been done i.e. structure of the model, its training and testing, and the different features that have been used in this model. Section 4 contains the analysis of the work. It discusses the obtained results and analyzes the model by determining errors and the accuracy of the model by the concept of considered error. Final Section contains the discussions, the conclusion, and future scope.

**Literature Overview:**

A lot of people have tried to study the patterns in the stock market and tried to make algorithms to make correct predictions about the price of the stock market. In [5] they used adaptive boosting (Adaboost), gradient boosting, extreme gradient boosting (XGBoost), decision tree, bagging, random forest, artificial neural networks (ANN), and recurrent neural network (RNN), and long short-term memory (LSTM). They found that LSTM was showing better results and was able to fit the data more accurately. In [7] they applied Support Vector Machine (SVM) and Principal Component Analysis (PCA) to create a stock selection model, train the model and then test on the original data. They found that the accuracy on the test set was 61.7925% and its performance were superior to A-share index of Shanghai Stock Exchange. [Hyejung Chung](https://link.springer.com/article/10.1007/s00521-019-04236-3#auth-Hyejung-Chung) & [Kyung-shik Shin](https://link.springer.com/article/10.1007/s00521-019-04236-3#auth-Kyung_shik-Shin) used a hybrid model made up of CNN and Genetic Algorithm (GA) to predict stock prices and found that it outperformed the standard ANN and CNN models [8]. In [9], they used the logistic regression to classify any stock as good or poor depending on its rate of return and found that model was showing an accuracy of 74.6%. Osman Hegazy, Omar S. Soliman and Mustafa Abdul Salam propsed an algorithm made up of integration of Particle Swarm Optimisation (PSO) and Least Square Support Vector Machine (LS-SVM) and found that it has better accuracy in predicting daily stock prices than Artificial Neural Network (ANN) and Levenberg-Marquardt (LM) algorithm[10].

**Theoretical Overview:**

RNNs have been used for sequence prediction problems for a long time. RNN works on the principle of saving the output of a particular layer and feeding this back to the input so as to predict the output of the layer. They solved a lot of issues with feed-forward neural networks like:

1. RNN can handle sequential data while feed-forward neural networks can’t.

2. RNN can memorize previous inputs while feed-forward neural networks can’t.

But sometimes RNN also runs into the problem of Vanishing Gradient. Vanishing Gradient refers to the issue when the gradient becomes very small and the updates in parameters become insignificantly small. Due to this, learning long sequence data becomes very hard.

Wnew = Wold – learning\_rate \* gradient (1)

Where: Wnew : new weight after updation

Wold : old weight

Also, RNN cannot store long-term memory. It has a very short-term memory due to which it cannot carry forward previous data to further layers.

To solve these problems Long-Short-Term-Memory (LSTM) was introduced. It is a special kind of RNN. LSTM contains both cell state(C) and hidden state (h). A Cell State carries the previous long-term information while Hidden State carries information from immediately previous events. Output at any level depends on the current cell state, hidden state, and the input data at the current level. It consists of three gates namely: Forget Gate, Input Gate, and Output Gate.

1. Forget Gate: It is used to decide which information should be remembered and which should be forgotten from the previous cell state. It changes their weight depending on the importance of information. It contains a sigmoid activated layer that outputs a number ranging from 0 to 1 where 0 shows that information is important and should be remembered while 1 shows that information should be forgotten. It increases or decreases the weight of information by multiplying the data with numbers ranging from 0 to 1. If it is multiplied by a number closer to 1 then the information should be remembered (increased weight) else should be forgotten.

2. Input Gate: It is used to find the importance of the new information carried by the input. Firstly information is passed through the sigmoid activated layer (output between 0 and 1) and then passed through the tanh activated layer (output between -1 and 1). Negative values mean that this information is not going to be used in the cell state at the current timestamp and positive values will be used.

3. Output Gate: It decides the output and the new hidden state. It passes the current cell state through tanh activation and then passes the previously hidden layer and current input through a sigmoid activated layer. Then output the new hidden state.

The following image shows an LSTM network.

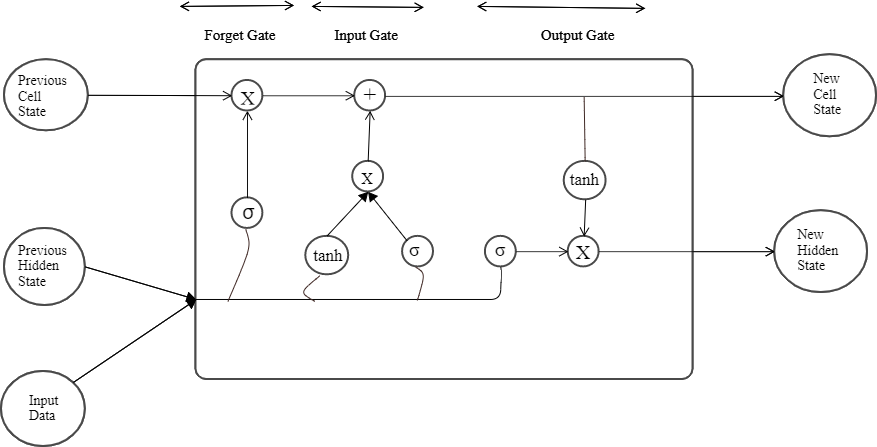
X: shows pointwise multiplication tanh in output gate: shows tanh activated neural network

+: shows pointwise addition

σ: sigmoid activated neural network

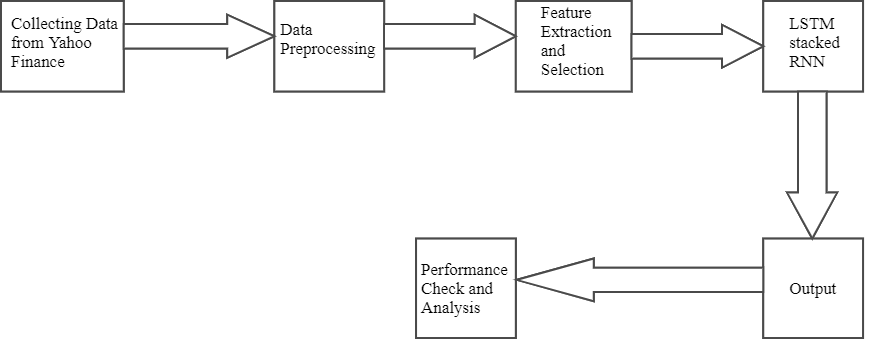
TANh in input gate: shows pointwise tanh

Fig 1: LSTM Network



**Our Proposed Work:**

Fig 2: Workflow of Stock Market Prediction System



The data has been taken from yahoo finance for Wipro Limited (WIPRO.NS). The work has been done for the time period 1/1/2005-17/4/2022. The data from 1/1/2005-31/12/2017 has been taken for training and testing has been done from 1/1/2018-17/4/2022. The dataset contained the following six features in it originally:

1. Open: Opening price on any particular day.

2. High: Highest price on that particular day.

3. Low: Lowest price on that particular day.

4. Close: Closing price on that particular day.

5. Adjusted Close: Adjusted closing price on that particular day.

6. Volume: Number of trades performed on that particular day.

Then, the Price of the Dollar was also added as the seventh feature in order to create our model as the Price of the dollar also affects [4] the Indian Stock Market. The price of the dollar affects the import and export between countries which eventually affects the price of assets of any company. Hence our dataset also contained the following seventh feature in addition to above mentioned six features.

7. Dollar: Price of US Dollar in INR.

After combining the original features in the dataset and the derived feature, the model was made using 4 LSTM layers which had corresponding dropouts associated which each of them. Dropouts were added for regularisation. Apart from these, the model had an output Dense Layer which provides us with the final output.

The following table shows the model summary.

|  |  |  |
| --- | --- | --- |
| Model: ”sequential” | | |
| Layer(Type) | Output Shape | Param # |
| Lstm (LSTM) | (None,30,64) | 18432 |
| dropout(Dropout) | (None,30,64) | 0 |
| Lstm\_1(LSTM) | (None,30,60) | 30000 |
| dropout\_1(Dropout) | (None,30,60) | 0 |
| Lstm\_2(LSTM) | (None,30,80) | 45120 |
| dropout\_2(Dropout) | (None,30,80) | 0 |
| Lstm\_3(LSTM) | (None,100) | 72400 |
| dropout\_3(Dropout) | (None,100) | 0 |
| Dense(Dense) | (None,1) | 101 |
| Total params: 166,053 | | |
| Trainable params: 166,053 | | |
| Non-trainable params: 0 | | |

Table 1 : Model Summary

In this table, Layer (Type) shows the type of layer that is present in our LSTM model. We have four LSTM layers in our system each having its corresponding dropout and finally, we have our output layer which is of dense type. Output Shape shows the shape of the output that is going out from each layer. None in the output shape refers to the variable batch size which is not fixed in Keras. Param shows the number of parameters that are being trained at each layer.

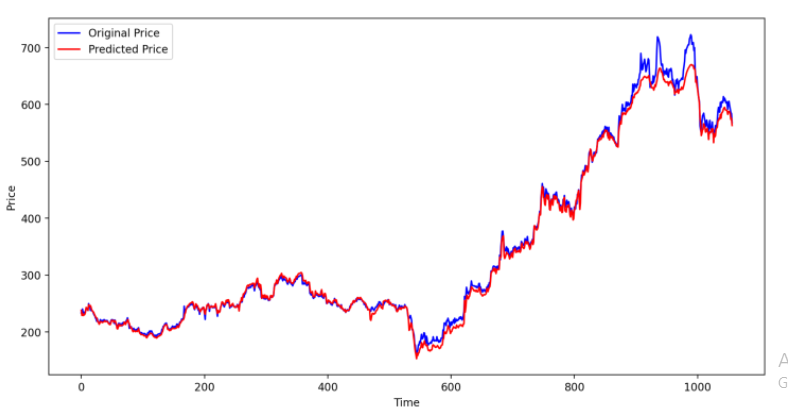
After the model was prepared, the model was trained. 100 epochs have been used to train the model. The model has been trained on 3207 data points (days) and Tested on 1057 data points (days).

**Result and Performance:**

After testing, this model showed the following results:

The below image shows a plot between the original and the predicted price.

Fig 3: Plot between Predicted and Original Price



In this figure, the blue line shows the real market price while the red line shows the predicted market price. Our Model was performing nicely continuously but around the 900th day of testing the stock price changed drastically due to which there was a big deviation between the predicted and original price but still our LSTM model is catching the patterns nicely.

Different Errors were calculated to analyze the model:

Error is the difference between the original price and its predicted value.

Mean Absolute Error (MAE): It is defined as the average of absolute values of the errors.

MAE: (Σ|(y\_test-y\_pred)|)/n (2)

Where: y\_test: original ground-truth value

y\_pred: predicted value

n: total number of values

Root Mean Square Error (RMSE): It is defined as the square root of the average of squares of the errors.

RMSE: ((Σ(y\_test-y\_pred))2/n)1/2 (3)

Where: y\_test: original ground-truth value

y\_pred: predicted value

n: total number of values

Normalised Root Mean Square Error: It is defined as the ratio of RMSE with the range, the difference between the maximum value and the minimum value. It helps in comparing the Root mean square error of two different entities of different scales.

NRMSE=RMSE/range (4)

Where: RMSE: root mean square error

Range: the difference between the maximum and the minimum value

The below table shows the different errors that are calculated to analyze the model.

Table 2: Error and their obtained values

|  |  |
| --- | --- |
| **Error** | **Value** |
| Mean Absolute Error | 6.833 |
| Root Mean Square Error | 10.790 |
| Normalized Root Mean Square Error | 0.019 |

Since it is difficult to define the accuracy in a regression problem, the accuracy, in this case, has been calculated by using the following method:

Let us say y\_pred is the predicted value and y\_test is the actual ground truth value on a particular day. Then Absolute error is calculated by using the following formula:

abs\_error = |y\_test – y\_pred| (5)

% abs\_error = (abs\_error/y\_test)\*100 (6)

Absolute % error has been calculated for each testing data. Then, we define a considered % error value (threshold). If that absolute % error is less than that considered % error then we consider that prediction as correct and if that %absolute error is more than %considered error then we classify that prediction as wrong. Now according to the number of correct predictions the accuracy of our model has been found.

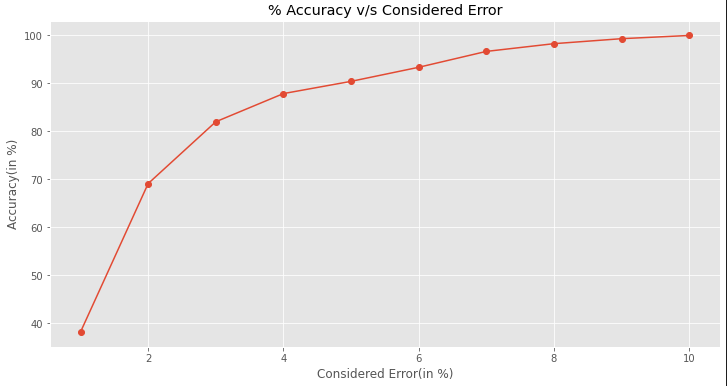
The results obtained for different considered error and corresponding accuracy has been tabulated below:

Table 3: Considered Error and its corresponding Accuracy

|  |  |
| --- | --- |
| Considered Error (in %) | Accuracy (in %) |
| 1 | 38.22 |
| 2 | 69.14 |
| 3 | 82.02 |
| 4 | 87.89 |
| 5 | 90.44 |
| 6 | 93.37 |
| 7 | 96.68 |
| 8 | 98.29 |
| 9 | 99.33 |
| 10 | 100 |

The following image shows a plot between Accuracy and Considered Error:

Fig 4: Plot between % accuracy and % considered error



It can be clearly seen that accuracy is increasing continuously if we increase the considered error.

**Discussion:**

As discussed above, LSTM has been very useful in solving time series forecasting problems efficiently. This paper shows the use of LSTM stacked RNN for predicting stock price for WIPRO limited. Apart from the features included in the original dataset we derived another feature i.e. Price of Dollar in INR as it also affects the stock market. The training was done over 100 epochs for the period 2005-2017. After that testing results were obtained and compared with the original values. The results have been closely analysed Different errors and accuracy has been clearly shown for this model. It is giving us good results and it is able to see the variations in the stock prices. Accuracy has been examined for different values of the considered error (deviation from the original price). The testing was done for 1057 days. Out of these 1057 days, predictions for 956 days had deviation, from the original value, less than 5% which gives us an accuracy for 90.44%.

**Conclusion:**

We can observe that the model is performing pretty nicely and able to read the patterns. It is giving 90.44% accuracy for a considered deviation of 5%. So, LSTM based models can be used to solve the time series problems efficiently. In future, we can constantly try to look for new ways to improve this model like adding some more derived features, integrating more than one models (create a hybrid model) etc. and thereby improving its predictions which makes its useful in real-time as a base for investing.

**References:**

[1]. Dattatray P. Gandhmal, K. Kumar, “Systematic analysis and review of stock market prediction techniques” School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India. (2019)

[2]. David M. Q. Nelson, Adriano C. M. Pereira, Renato A. de Oliveira, “Stock Market’s Price Movement Prediction with LSTM Neural Networks”

[3]. Hiransha Ma, Gopalakrishnan E.Ab, Vijay Krishna Menonab, Soman K.P\*, “NSE Stock Market Prediction Using Deep-Learning Models” International Conference on Computational Intelligence and Data Science (ICCIDS 2018)

## [4]. [Hsing Fang](https://www.sciencedirect.com/science/article/abs/pii/1059056094900345" \l "!), [Jean C. H. Loo](https://www.sciencedirect.com/science/article/abs/pii/1059056094900345#!), “Dollar value and stock returns” [International Review of Economics & Finance](https://www.sciencedirect.com/journal/international-review-of-economics-and-finance) [Volume 3, Issue 2](https://www.sciencedirect.com/journal/international-review-of-economics-and-finance/vol/3/issue/2), 1994, Pages 221-231

[5]   
settings

[5]. M. Nabipour, [P. Nayyeri](https://sciprofiles.com/profile/1109950), [H. Jabani](https://sciprofiles.com/profile/author/MWdPM3ZRZ3FmcHR0c2NlUWZZR1dWZGpzd2g3Mk0rVGJPVVlEMHE3KzVnRT0=), [A. Mosavi](https://sciprofiles.com/profile/414895), [E. Salwana](https://sciprofiles.com/profile/author/eWpwVExVOHdheEpYVGVqNDZmWGY2Znhack1PN3l3VHZHK0hvd2VHazR4OD0=), [Shahab S.](https://sciprofiles.com/profile/316438) , “Deep Learning for Stock Market Prediction”

# [6]. Rodolfo C. Cavalcante, Rodrigo C. Brasileiro, Victor L.F Souza, Jarley P. Nobrega, Adriano L.I. Oliveira, “Computational intelligence and financial markets: A survey and future directions.” Expert System with Applications 2016, Volume 55, Pages 194–211.

# [7] Huanhuan Yua , Rongda Chenb , Guoping Zhangc, “A SVM Stock Selection Model within PCA” 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014.

# [8]. [Hyejung Chung](https://link.springer.com/article/10.1007/s00521-019-04236-3#auth-Hyejung-Chung) & [Kyung-shik Shin](https://link.springer.com/article/10.1007/s00521-019-04236-3#auth-Kyung_shik-Shin), “Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction” 2019

[9]. Avijan Dutta, Gautam Bandopadhyay, Suchismita Sengupta, “Prediction of Stock Performance in the Indian Stock Market Using Logistic Regression” 2012.

[10]. Osman Hegazy, Omar S. Soliman and Mustafa Abdul Salam, “A Machine Learning Model for Stock Market Prediction” International Journal of Computer Science and Telecommunications [Volume 4, Issue 12, December 2013].