International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-05, Issue-05, Aug 2019

NIFTY50 Stock Price Forecasting Using LSTM

N.L.Anbarivan, Student, Master of Computer Science, VIT, Vellore, India,
anbarivan007@gmail.com
V. Geethanjali, Student, Bachelor of Technology, Peri Institute of technology, Chennai, India,
anjali.mala.2110@gmail.com

Abstract-The stock market has turned into a critical aspect for development of the nation. They have also grown as a hub for exponential investments and growth. But due to its highly unpredictable nature, they are always risk oriented. Traditional methods used before the digital age indulged in forecasting based on intuition of the people. Later many tools and methods were developed so people could draw better conclusions and jeopardies can be reduced. These methods are highly linear in nature. Since stock market are unpredictable, linear models cannot be used. Moreover, bygone models depend on historical data, but it is difficult to feed more data using these models and predict the underlying trend. Our proposed project predicts next day's closing price with the help of historical data collected from various data sources and analyze the various crucial factors which influence it. This is implemented using deep learning algorithms and data mining techniques. Recurrent neural networks are chosen as the base model as they work well with sequential and time series data. The efficiency of the model is improved using Long-short term memory (LSTM). This research will help in choosing the best methods for predicting the stock prices and serves as an advantage to the public. This design is an elaborated justification of foretelling stock prices and help in better investments.

Keywords — ADAM, LSTM, RELU, RMSE, RNN, Stock markets.

I. INTRODUCTION

The stock market is considered as one of the most important aspects for the development of the nation. It affects our day to day life in many ways. There are fluctuations occurring daily and there is a need to predict the stock movement so we can take better decisions about out investments.

Some of the traditional methods include technical, statistical, fundamental, etc. But these methods are linear in nature. Since stock market are inconsistent, linear models cannot be used. Another problem is that there are various constraints that influence the stock movements. Some of the parameters include opening price, volume, day to day news, global financial crisis and world stock market movement. It is vital to choose these factors correctly so reasonable judgements can be made and public can benefit from the investments. Due to these drawbacks, they provide a platform for researchers and interested people to analyze and provide better insights.

Predictive analysis can be based on behavior as well as the characteristics of the previously obtained historical data. For this machine learning and deep learning models are used and have been popular in the recent years. Deep learning highly deals with nonlinear data and functions and provide a better way of anticipating the stock prices. Mainly neural networks are designed to deal better with nonlinear time series analysis because they have nonlinear activation function. They examine the characteristic of the share market well and help in ease of extrapolation.

Neural networks can be classified as artificial neural networks (ANN), convolutional neural networks (CNN) , recurrent neural networks, etc. Recurrent neural networks (RNN), also known as Backpropagation through Time (BPTT) is chosen as the base model since sequential data works well with it. This is taken into consideration by analyzing the input, internal memory constraints. They are strong and influential than the other types of neural networks. Also due to their deep-rooted nature, more analysis has been done in the past and many perceptions are drawn that serve as an advantage to us.

Another advantage is that the rapid growth of technology and Graphical Processing Units (GPU), handling massive amount of data is easy. Hence, they are preferred more since noise is also reduced. One of the drawbacks is that they do not work well with long term data. Hence long short-term memory (LSTM) are combined with RNN for better performance. Customary neurons are replaced by a memory cell-that does some complex computations in the hidden layers. They have memory associated with them and are able to understand the inherent structure as well as behavior of the neural networks.

 DOI : 10.35291/2454-9150.2019.0382

II. LITERATURE SURVEY

A new generalized score function of interval-valued intuitionistic fuzzy sets and applications in expert systems: In this paper, a new generalized improved score function has been presented for finding the best alternatives among the feasible alternatives where decision matrix related to different criteria about the selection of attributes is represented in interval-valued fuzzy sets [1].

In this paper, the concept of an of intuitionistic fuzzy sets (IFS), it starts with the definition of the degree of similarity between IFS's is introduced. Then, several new similarity measures between IFS are proposed and corresponding proofs are given. Finally, the similarity of IFS is applied to pattern recognitions [2]. The next paper covers decision-making for financial trading algorithm, A fusion approach of machine learning and portfolio selection on financial datasets [3].

In the next paper, a novel technique for evaluating and selecting logistics service providers the logistics resource views the increasing importance of logistics outsourcing and availability of logistics service providers (LSPs) highlights the significance and complexity of the LSP evaluation and selection process [4].

This paper investigates price and trade size clustering in individual trades executed in the NSE's fully computerized order-driven trading system and explores size and price clustering in pure order driven market. This exchange provides an ideal setting to examine trade size and price clustering where investors can trade without an intermediary [5].

This paper details about using deep learning to forecast long as well as short term stock market. Vanishing gradient problem is taken care. Bidirectional and Stacked LSTM are used for investigational purpose. Training is done using publically available dataset [6].

A collaborative prediction of stock market using numerical and textual parameters is considered in this paper. Traditional methods like word embedding for textual features are discarded and event embedding's vectors are used. They are modeled using LSTM and CNN (Convolutional neural networks). [7].

This paper uses attention based techniques to find and gain insights between the financial news and numerical features. Stock market trend insights gained from news is converted to numbers for better predictive strategy. Noise is also taken care well in this paper [8].

This paper elaborates on recognizing stock movement pattern and trading strategies using long short term memory in combination with attention model [9].

This paper introduces usage of artificial neural networks to predict the market behavior. It is regularized by Bayes network. Their objective is to predict the next day's stock price using historical data and few vital parameters. They have built a probability based weight assignment model to forecast stock prices. [10].

Ensemble modeling using support vector machines and ARIMA is implemented in this paper. ARIMA falls short in predicting nonlinear patterns which is taken care using support vector machines. ARIMA is best suited for time series analysis.

III. METHODOLOGY

The problem proceeds in the following way. First, stock data is gathered from the web. It consists of preliminary factors that influence the stock market. They include parameters like the open, close, etc. Then in the next step, preprocessing is done. It is crucial because outliers, noise can be reduced or removed. Then normalization of data is done so good quality data can be fed into the model. Feature engineering selects the most apt feature, does feature reduction and finds relation between the features, etc. After preprocessing the data, it is fed into the model. Here test train split, scaling, x and y selections, hyper parameters and parameters are decided. After fitting the data into the model, performance is evaluated. Mean squared error (MSE) and accuracy is computed.

A. Data Preprocessing

Data pre-processing is a unavoidable step when you want to get some insights from data sets to help you make the prediction. As the initial data may have a lot of noise, it is necessary to reduce them so that they will not interfere with you result. Besides, some features of data may make no sense, in order to improve efficiency, one should neglect them when you train the data.

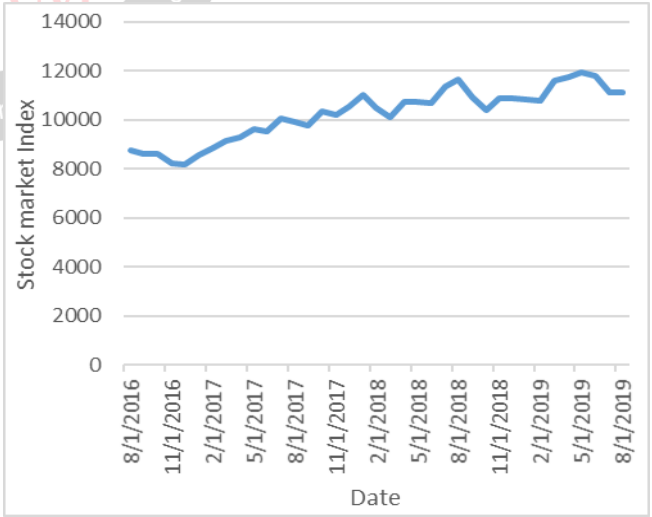

Fig 1. National Stock Exchange of India's benchmark stock market index for Indian equity market VS year

The data initially contains not only several features, including opening price, closing price, high price, low price, volume, turnover, change rate, etc., but also the name and the code of the stock. The part of closing price chart is shown in Figure 1. Most of them will change once every five minutes. To accelerate the speed of running a program

 DOI : 10.35291/2454-9150.2019.0382

and improve its efficiency, the first thing we should do is to clean the data.

B. Data virtualization

Each of the features are visualized to know the range, distribution, etc

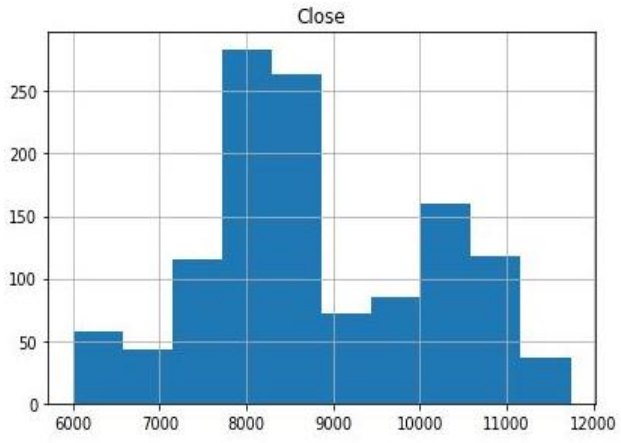

Fig 2. Histogram of low

C. Outlier analysis on features

In this problem there are various outlier, so box plot outlier analysis is used to identify and remove the outliers from the Data frame. The box plot consists of various quartiles Q1, Q2, Q3. Q1 and Q3 are the first and third quartile and Q2 is the median in the Data frame. After the Q1 region the points are called smallest non outlier and Q3 region has largest outlier.

The data points which are away from the Q1 and Q3 regions are classified as outlier from the Data frame. These outliers should be removed so the data will be cleaned, and various machine learning algorithms can be applied.

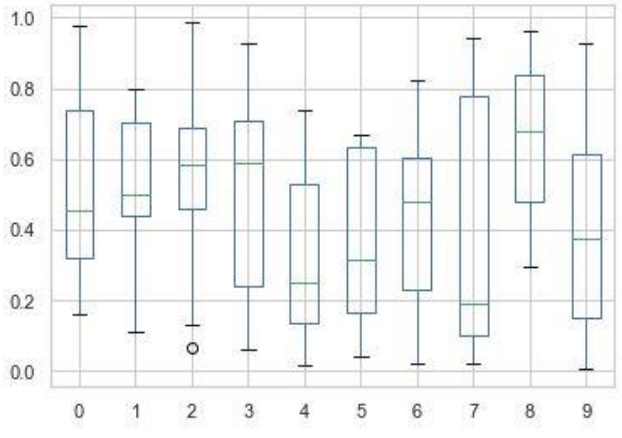

Fig 3. Box plot outlier for the features

D. Correlation matrix of all features

Correlation matrix is generated for the Data frame to identify the dependency or relationship between the features. Various features like Open, Close, Volume Etc. Data values are taken and correlation matrix is generated. This also helps in feature selection of the chosen Data frame. Here output is the next days closing price that can be obtained by the shift operation. Here in the column next_close the final row will have null value due to the shift operation .hence this is removed.

	OPEN	HIGH	LOW	CLOSE	VOLUME	NEXT_CLOSE
OPEN	1	0.999573	0.999394	0.998855	0.621808	0.99715
HIGH	0.999573	1	0.999359	0.999487	0.626667	0.997874
LOW	0.999394	0.999359	1	0.99956	0.61474	0.997977
CLOSE	0.998855	0.999487	0.99956	1	0.620311	0.998471
VOLUME	0.621808	0.626667	0.61474	0.620311	1	0.620158
NEXT_CLOSE	0.99715	0.997874	0.997977	0.998471	0.620158	1

Fig 4. Spearmans correlation metrix for the dataframe

E. Recurrent neural networks

We will start by getting perceptions about sequential data. They are a collective set of relative data in an ordered manner. Examples include time series data, DNA, etc. This data is considered as a time series data as the data is presented as a series of time stamps and their values. The difference between the feed forward neural networks and recurrent neural networks are that in RNN'S there are loops of information cycles. Whenever computation is done, it is fed back into the neural network.

This helps in memory whereas in feed forward neural network, data moves in one direction hence there is no memory retention of the previous computations and inputs. Hence time series data does not perform well here but RNN's work well with short term memory as they have their internal memory but not useful in keeping long term memory.

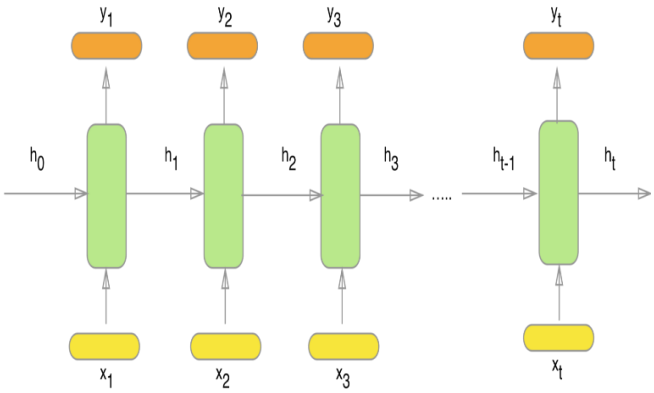

Fig 5. RNN representation

In the below representation, x1, x2, x3, etc. represents the inputs that are fed into the model. Y1, Y2, Y3, etc. represents the output obtained from the model. In our case it refers to the next day's closing price. h0, h1, h2, etc. represents the information gathered from the previous step.

 DOI : 10.35291/2454-9150.2019.0382

Here output is produced by applying activation functions and applying weights to both current and previous state. They are considered as a complex gradient descent problem. They are useful in handling one to one, many to many, one to many, many to one problem. Here backpropagation is done where error from each step is subtracted from the weights and then these derivations are appended to gradient descent problem which decreases the cost function. Hence weights are updated frequently until they fit into the model correctly. It helps us to understand what is going and understand the inherent structure of the model. The information is passed from the first-time stamp to the next. Error in this step dominantly depends upon the previous step.

One example includes the language model where we try to predict the next word based on the previous list of words. How well the model predicts depends entirely on the distance between the relevant data and area where we are trying to predict. If it's small, then the model predicts well. But as the distance between them increases, then the model's capacity to predict accurately decreases over time. They are unable to learn and this serves as a problem for our stock prediction. So, they are optimized and combined with Long short-term memory. Theoretically it is assumed that RNN's work well with long term memory but practically they do not perform well.

Another two problems in the regular RNN are –gradients that is used to estimate how change in input affects the output. They are considered as complex linear regression problems where we estimate the slope. Higher the steep, the model learns quickly but when it reaches zero the model stops learning and introduces to two issues that are exploding gradient and vanishing gradient.

F. Long short term memory

Long short-term memory provides special memory units that extend the RNN's memory. Important events from a sequence of data is extracted. they form the building blocks of RNN. They are more capable than RNN's in handling long term data because they are specialized in reading, writing, deleting memory similar to a computer's memory. They can be considered as a series of gated cells that opens or closes based on the decision to keep or throw away the memory. They can be imagined like a gate that opens when information is kept or closed when the information is of no use and thrown. The importance of these information is decided using the weights assigned to them. Hence it trains over time. In LSTM's there are three gates –input gate, forget and output gate. Input gate decides if the new input fed is relevant or not, forget gate deletes the input that is not important, Output gate impacts the output at current time stamp.

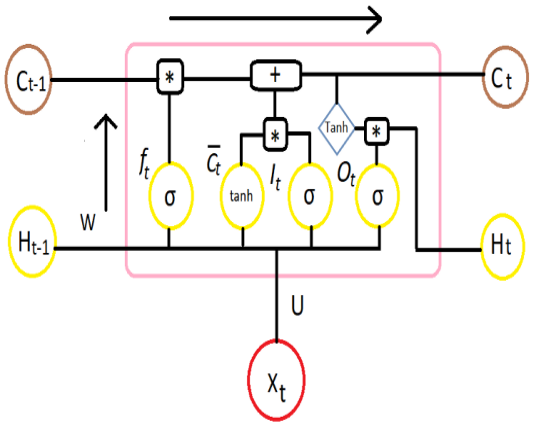

Fig 6. LSTM representation

LSTM does series of addition and multiplications and has mechanism called the cell state which is important for flow of information in LSTM. Here the input is a combination of previous memory's, previous hidden state represented as 'h' and the current input. Output includes the current input state and memory state. The gates discussed above takes the input, output, etc. and applies dot product and concatenates them by applying an activation function. Finally, four vectors are produced at each time stamp.

Next important attribute is memory state c where input of the memory is stored. Here is where information can be added or removed and is modifiable. Element wise multiplication with forget gate is done with previous state of memory also known as c(t-1).

*Ct=previous state memory(Ct-1) * forget gate (ft)*

Here the forget gates value plays an important role. If its 0, then the memory is thrown out else if its 1 it its kept and passed on to the next state. Hence forget gates value is between 0 and 1. The new memory state is calculated using the previously obtained memory state, input state and c layer.

C_t=C_t *(previous memory state) + (It (input state) * C'*$_t$)

Next, we will calculate the final output from tanh of the cell state. Multiplication with each element is done and hidden state is obtained.

Ht=tanh (Ct (CELL STATE))

This process is repeated and the values of Ct and Ht are passed at the end of each layer

G. Hyper parameter optimization

Network structure is greatly influenced by some variable also known as hyper parameters that determine how well the network learns through the new data and fits into the model. They must be set before the training and includes updating the weights and optimization

 DOI : 10.35291/2454-9150.2019.0382

Table 1. Hypermarameters for LSTM nets

Hyperparameter	Values considered
Scaling of predefined features	{standard-deviation, tanh, sqrt}
Number of Hidden Units	{1024, 2048, 4096, 8192, 16,384}
Number of Layers	{1, 2, 3, 4}
Backpropagation Learning Rate	{0.01, 0.05, 0.1}
Dropout usage/rate	{no, yes (50% Hidden Dropout, 20% Input Dropout)}
L2 Weight Decay	{0, 10^{-6}, 10^{-5}, 10^{-4}}

H. Fine-tuning hidden layer and their units

Hidden layers consist neuron units that are connected to input of other layers. Complex computations are done on the inputs they receive and activation functions are applied to get the actual output. The activation function as well as the calculations depend on the application and the kind of neural network that is implemented. Hidden layers convert the input they are fed into usable form. They give a nonlinearity to the input and help in forming a complex function that presents the entire problem statement. So hidden layers should be chosen in such a way that there is a good fit. But it is difficult to be implemented in practical life.

I. LSTM Network weight optimization

This is the most important step in the entire neural network. There are two cases to be considered when deciding this - one is assignment of initial weights to zero. Here partial derivation is same for all the values in the epochs with respect to the loss function. Hence all the hidden layers become symmetric and results in an outcome that is a nonlinear function. Thus, setting a model to zero is generally of no use. But bias can be set to 0 and model learns through it.

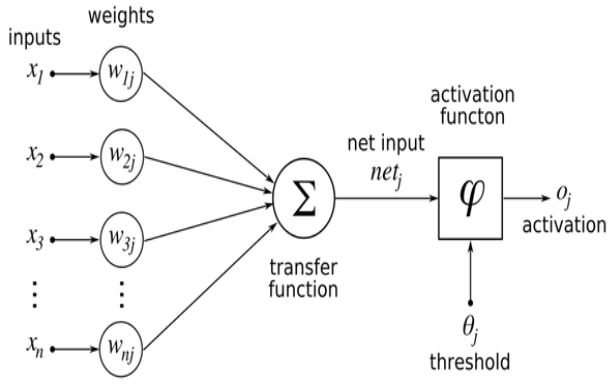

Fig 7. Network weight initialization

In the second case we initialize the weights randomly and sometimes this leads to the issues of vanishing and exploding gradients. In vanishing gradient, the weight is updated to very small values that results in slower conjunction. Sometimes entire neural network may be disrupted.

This happens mostly in the case of sigmoid and tanh. So mostly relu function is preferred. This is because gradient becomes zero when the input is negative and becomes 1 when the inputs are positive. The next problem is exploding gradient where exact opposite of vanishing gradient takes place. Here there are large weights that are multiplied and cause loss function to change abruptly and the momentum is so large that it fails to get the pattern or the inherent structure. One of the best practices is to use their leaky relu or relu especially in cases where they are not too dense. We can opt for a heuristic to initialize the weights.

For RELU activation function:

$$\sqrt{\frac{2}{size^{[l-1]}}}$$

$$W^{[l]} = np.random.randn(size\_l, size\_l\text{-}1) * np.sqrt(2/size\_l\text{-}1)$$

Here random values are multiplied by the above function. They serve as good places for initialization as well as avoiding the exploding and vanishing gradient. They stabilize the momentum as well as avoids converging slowly. RELU Activation function plays an important part in weight initialization.

J. LSTM Network weight optimization

It is an optimization algorithm that minimizes the cost function by choosing the values of parameters correctly.

input : $f : \mathbb{R}^n \to \mathbb{R}$ a differentiable function
$\mathbf{x}^{(0)}$ an initial solution
output: $\mathbf{x}^\star$, a local minimum of the cost function f.
1 begin
2 $k \leftarrow 0$;
3 while STOP-CRIT and $(k < k_{max})$ do
4 $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \alpha^{(k)} \nabla f(\mathbf{x})$;
5 with $\alpha^{(k)} = \arg\min_{\alpha \in \mathbb{R}_+} f\left(\mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x})\right)$;
6 $k \leftarrow k + 1$;
7 return $\mathbf{x}^{(k)}$
8 end

Fig 8. derivation for minimizing the cost function

It is used when we cannot calculate parameters manually or when it is computationally expensive. Values for coefficients are picked, cost function is computed then new coefficients are estimated. Repeating the method for a few times will lead to choosing the best values for parameters and reducing the cost function. Initial values for the parameters are chosen as 0 or very small values. They are then put into a function and cost of coefficients is

 DOI : 10.35291/2454-9150.2019.0382

computed. Here we use batch gradient decent with batch size of 21.

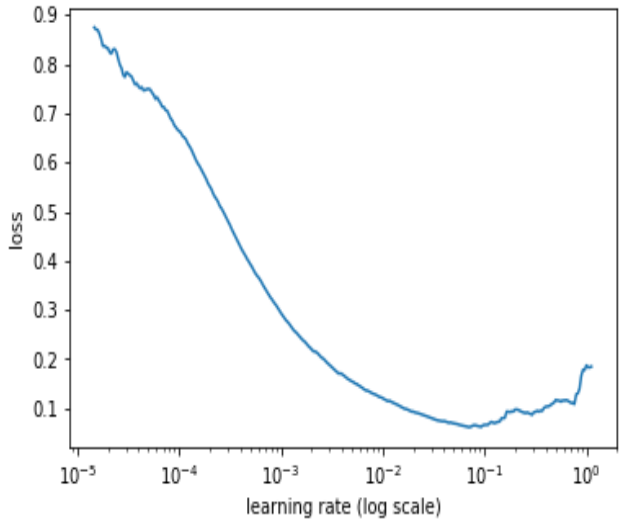

Fig 9. Loss vs leaning rate

The calculus derivation is also computed for a function slope at a particular point. We need to estimate the direction in which the slope moves. It should move downhill. This is done until the problem is optimized.

K. One hot encoding

Different parameters are in different ranges.Hence they are condensed to a range between -1 to 1.Min-max scalar is chosen in this problem. So it transforms each parameter into a particular fixed range.

```
array([[ 0.94243851,  0.9440055 ,  0.94701525,  0.93828118, -0.93556183],
       [ 1.        ,  0.9685675 ,  0.92191721,  0.90104301, -0.87214377],
       [ 0.98883656,  0.99038131,  1.        ,  1.        , -0.93610923],
       ...,
       [-0.95499738, -0.95705943, -0.94422658, -0.95465908, -0.7226035 ],
       [-0.96790511, -0.95929234, -0.95276688, -0.95414188, -0.77154922],
       [-0.96075353, -0.96375816, -0.95607843, -0.96069304, -0.71288023]])
```

fig 10. Encoded array of features

Our data is split into two parts –training set and testing set. Our model is made to fit the training data and we make predicitions on the testing data. But it is subjective to overfitting or underfitting problem which can affect our performance tremendously. Overfitting occurs when it learn training set in detail fits closely to the training data .This is the case when number of parameters are larger than the observations. This models fails in predicting new data as it has not generalized and it is unable to conclude or make inferences. There is noise in training data and that should be taken care of. In the next problem,the model ignores or does not learn the inherent structure. This happens in simple model and happens when we fit a non linear model to a linear data. Initally in train dataset ,output is present that the model learns and gets a generalized inference that is useful in predicting new data.

The calculus derivation is also computed for a functions slope at a particular point. We need to estimate the direction in which the slope moves. It should move downhill. This is done until the problem is optimized.

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 2000)              16016000
_________________________________________________________________
dropout_1 (Dropout)          (None, 2000)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2001
=================================================================
Total params: 16,018,001
Trainable params: 16,018,001
Non-trainable params: 0
```

Fig 11. LSTM model Architecture for NIFTY 50 stock data

IV. RESULTS

Traditional models like ARIMA produce more noise due to their simplicity and linear updations. Better results are produced when dealing with short term data whereas they perform poorly in comparison to LSTM RNN for long term predictions.Traditional models generally focus on linear univariate data. Models like LSTM have proof of reducing the loss rate by 84-87 % compared to the older models. LSTM also handles overfitting problem well. Also LSTM handles non linear noisy data and are capable of storing long term relationships in a sequence data successfully.

The mean squared error for train and test is evaluated and they turn out to be fairly small which means our model has a good fit. On training the model with larger epochs the model starts to overfit, hence we choose an epoch of 20.

 DOI : 10.35291/2454-9150.2019.0382

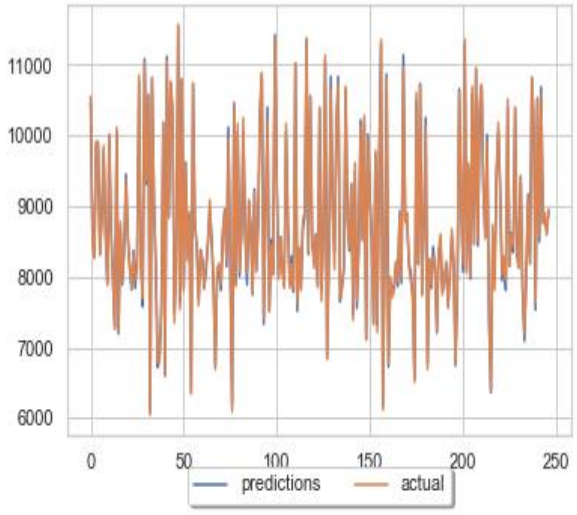

Fig 12. Actual vs predicted data points

Training data is fed and model is trained. With inverse transform the values predicted are in normalized form. They are obtained in the orginial form. So that next day closing price can be extracted. It turns out to be 83.559135 which is the next days closing price.using this sell or buy decsions can be made. The denormalized data is plotted on a graph with actual and predicted closing prices.

Table 2: Actual vs predicted data -Predicted data points using LSTM neural networks

index	Actual	predicted
0	10548.7	10485.275
1	8448.1	8641.677
2	8267	8296.681
3	9915.25	9892.565
4	9914.9	9868.44
5	8319	8318.919
6	8756.75	8922.32
7	9852.5	9786.144
8	8638	8597.539
9	7880.7	7914.994
10	10020.55	10001.597
11	8632.6	8644.901
12	7954.35	7938.6
13	7275.5	7275.18

For comparing the model performance, same data points are predicted using Navie Logistic regression algorithim. Calculated the logistic regression coefficients for every features.

Table 3: Coefficients for Logistic Regression

	Coefficient
OPEN	-0.138836
HIGH	0.042311
LOW	0.054475
CLOSE	1.039566

Since the data points have huge variation and seasonality logistic regression predected the data points with more error rate.

Table 4: Actual vs predicted data

	Actual	Predicted
18	6126.25	6142.777915
342	8319.00	8342.831998
467	7942.70	7895.684170
851	9588.05	9576.870904
950	10308.95	10299.114702

The actual and predicted results show that the model predicts the next days price fairly well. The model achieves an high accuracy of 98% for next days closing price. On backtracking the model,we find that actual and predicted test data are almost similar and this is largely contibuted by hyperparameter optimiztaion, preprocessing.

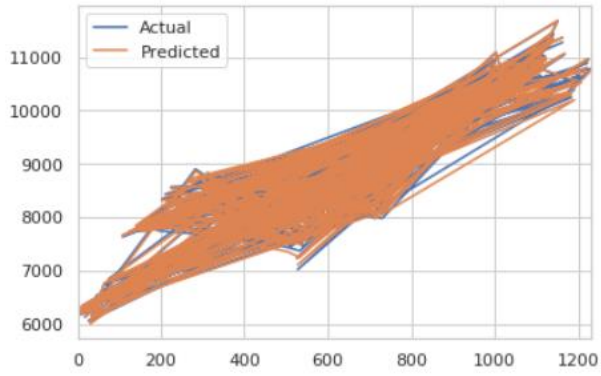

Fig 13. Actual vs predicted results.

V. CONCLUSION

Prediction of stock market is cumbersome because of evaluation of plenty of marketed data. In present study, we have created a LSTM neural network model to predict the NIFTY 50 data using various architectures of Neural networks, RNN, LSTM etc. Our developed model is capable of forecasting the next day's closing price of a particular stock along with understanding of the underlying dynamics of the time series. The results shows that our model provides more accurate , precise and reliable prediction data when compared to other baseline models like Logistic regression, ARIMA, etc.

There is a possibility of increasing the accuracy and precision by adding a large number of data which helps in creating and extracting more patterns. In future, the developed model can be linked with real time data feed and fine tuning with parallel GPU'S. In addition, we can also link it with news feed, twitter data to produce more accurate and precise model to predict stock prices.

 DOI : 10.35291/2454-9150.2019.0382

REFERNCES

[1] Garg,Harish.” A new generalized improved score function of interval-valued intuitionistic fuzzy sets and applications in expert systems.” Applied Soft Computing 38(2016):988-999.

[2] Dengfeng, L., & Chuntian, C. (2002). New similarity measures of intuitionistic fuzzy sets and application to pattern recognitions. Pattern Recognition Letters, 23(1-3), 221-225.

[3] Paiva, Felipe Dias, Rodrigo Tomás Nogueira Cardoso, Gustavo Peixoto Hanaoka, and Wendel Moreira Duarte. "Decision-making for financial trading: A fusion approach of machine learning and portfolio selection." Expert Systems with Applications 115 (2019): 635-655.

[4] Alkhatib, S. F., Darlington, R., Yang, Z., & Nguyen, T. T. (2015). A novel technique for evaluating and selecting logistics service providers based on the logistics resource view. Expert systems with applications, 42(20), 6976-6989.

[5] Mishra, A.K. and Tripathy, T., 2018. Price and trade size clustering: Evidence from the national stock exchange of India. The Quarterly Review of Economics and Finance, 68, pp.63-72.

[6]Khaled A. Althelaya, El-Sayed M. El-Alfy, Salahadin Mohammed, "Evaluation of bidirectional LSTM for short- and long-term stock market prediction", Information and Communication Systems (ICICS) 2018 9th International Conference on, pp. 151-156, 2018.

[7]Pisut Oncharoen, Peerapon Vateekul, "Deep Learning for Stock Market Prediction Using Event Embedding and Technical Indicators", Advanced Informatics: Concept Theory and Applications (ICAICTA) 2018 5th International Conference on, pp. 19-24, 2018.

[8]Guang Liu, Xiaojie Wang, "A Numerical-Based Attention Method for Stock Market Prediction With Dual Information", Access IEEE, vol. 7, pp. 7357-7367, 2019.

[9]Li-Chen Cheng, Yu-Hsiang Huang, Mu-En Wu, "Applied attention-based LSTM neural networks in stock prediction", Big Data (Big Data) 2018 IEEE International Conference on, pp. 4716-4718, 2018.

[10] Ticknor, J. L. (2013). A Bayesian regularized artificial neural network for stock market forecasting. Expert Systems with Applications, 40(14), 5501-5506.

[11] Pai, P. F., & Lin, C. S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. Omega, 33(6), 497-505.

 DOI : 10.35291/2454-9150.2019.0382