Forecasting the Stock Price using ARIMA and LSTM

Malavika Loka
SEAS
University at Buffalo
Buffalo, New York USA
mloka@buffalo.edu

Sai Anush Reddy Udyala
SEAS
University at Buffalo
Buffalo, New York USA
saianush@buffalo.edu

Koushik Kalluri
SEAS
University at Buffalo
Buffalo, New York USA
koushikk@buffalo.edu

Harish Kavipurapu
SEAS
University at Buffalo
Buffalo,New York USA
hkavipur@buffalo.edu

Jaya Venkata Mohana Harsha Matlapudi
SEAS
University at Buffalo
Buffalo,New York USA
jayavenk@buffalo.edu

***Abstract*— The stock market is a highly composite system and concealed with mystery, therefore it is very difficult to analyze all the impacting factors before making a decision. There are many stock market predictions available online which suggests users when to do an investment in stock. The Problem with the existing prediction system is that the accuracy of the prediction is so less and uses techniques dated back to the times when the term 'Stock' was defined. Technology has come a long way since many prediction techniques were engineered but not many of them were implemented. Many technologies were theoretically proved but not practically implemented. The scope and goal of the project is to dig out some of the theoretically proven prediction techniques and implement them from scratch. As stated above there are numerous prediction techniques but some offer high accuracy for different types of data. Choosing a classifier depends on the type of problem you are trying to solve. The performance of a classifier model also depends on the data you are working with. — the stock market has always been a center of attention for investors. Tools that help in stock trend forecasting are in high demand as they help in the direct accession of profits. The more precise the results, is the higher chances of acquiring more profit. Factors such as politics, economics, and society impact the trends of the stock market. The analysis of stock trends can be performed using fundamental or technical analysis. The fundamental analysis comprises data from financial records, company assets, market shares economic reports, etc. Here analysis of the company's financial factors including the strategic initiatives, microeconomic indicators, and consumer behavior is conducted. Technical analysis is the interpretation of past and present prices with the help of which the probable future prices will be predicted. To plot stock market forecast various deep learning and machine learning algorithms are used. Out of which LSTM and ARIMA have proven to provide fairly accurate results. The papers presented before focused on the individual models and their components to provide forecasts. Their aim was the provide the predictions with the best-suited parameter values. The purpose of the paper is to offer the investors models which can work well with the data with appropriate parameter values. The aim is to provide both the models LSTM and ARIMA because of their capability to provide appropriate results with the help of technical analysis of the data set. The objective is to compare both the models and use the best one suited when it comes to a particular company data set. The data set used are the historic stock prices of the companies which include the open, close, high, low values. On which pre-processing is done which involves sorting the data, feature scaling, autocorrelation check, splitting it into training, and testing data sets. The results obtained show that the individual models work well when the data provided suits the model and appropriate parameter values are set. The accuracy of the models for each attribute is over 90%. Both the models can prove to be an asset to stock traders. The LSTM model provides better results when the data set is large and has fewer Nan values. Whereas, despite providing better accuracy than LSTM, the ARIMA model requires more time in terms of processing and works well when all the attributes of the data set provide legitimate values. Thus, the paper presented provides models which can visualize the** future **trend of the stock as graphs and give an overview of stock trends.**

Keywords- LSTM, ARIMA, Stock trend forecasting, technical analysis, Machine Learning, Deep learning, Historic stock prices LSTM, Closing price, Auto Regression.

I. MOTIVATION

In the world of finance, stock trading is one of the most important activities. Professional traders have developed a variety of analysis methods such as fundamental analysis, technical analysis, quantitative analysis, and so on. In recent years, the increasing prominence of machine learning in various industries have enlightened many traders to apply machine learning techniques to the field, and some of them have produced quite promising results. Predicting whether the prices will increase or decrease in the next n days, using the stock prices and volumes in the past m days. For this classification problem, techniques that will be implemented are Logistic Regression, Bayesian Network, Simple Neural Network, and SVM and run them on the dataset extracted. After getting the preliminary results, the technical indicators should be included in the predictor and tried to predict the exact change in prices in the next n days. They reserve a particular stock for presenting the results, and use the data of other remaining stocks for training and testing. For every trial, the data is randomly shuffled and partitioned into a training set, and a test set in a ratio of 9:1. Then two metrics are used for evaluating different regression models: the mean squared error, and the prediction error rate (whether the price will increase or decrease). The first metric is primarily used for checking whether there is an issue of over-fitting since it's not as straightforward and cannot be generalised to different data with different variances. The second one is the primary criteria and is used for comparing the results to the preliminary findings.

There are numerous algorithms that predict the Stock prices but only a few offers high accuracy like Autoregressive Integrated Moving Average (ARIMA). ARIMA is a statistical model and is believed to be one of the best models which predict Stock prices. Statistical models like ARIMA are limited in the ways that they calculate the prediction values and cannot offer better predictions as it doesn't learn to predict, rather learn to calculate the statistical values and applies functions to them.

The purpose of this project is to build a Long Short-Term Memory (LSTM) model to predict the Stock prices and compare its performance to ARIMA's performance thereby gaining an insight into which model is better using the accuracy of the predictions as the parameter.

II. RELATED WORK

Umadevi.K.S and R. Jagadeesh Kannan (2018), developed a system that includes various operations ranging from a collection of stock scores of a predetermined time frame to analyzing them in the form of various plots. The methodology consists of the following processes:

1) Obtaining the live streaming data from reliable APIs.

2) Storing the data in the local database repository.

3) Fetching the data into the R environment using in-built library functions.

4) Visualizing the fetched data using candlelight plotting format.

5) Forecasting the stock scores using a time series model.

There are various APIs available today providing livestock scores precisely such as the Yahoo Finance, Google Stocks, etc. By using one of the apis, live data is streamed and stored in the storage repository. In the paper they decided to perform the analysis using the R tool. One of the packages in R named as forecast enables to perform the time series analysis ARIMA.

ARIMA is an acronym for Auto-Regressive Integrated Moving Average. It is a Univariate (single vector) prediction technique that predicts the future possible values based on its own momentum. For one prediction, ARIMA requires at least 40 historical data points so as to develop a training model. It provides best results only when the data under consideration possesses a steady form over time with as low outliers as possible. This is the base prediction technique on which the other models' prediction techniques are compared and contrasted to see if the efficiency can be improved.

Klaus Greff, Rupesh K. Srivastava, Jan Koutn´ık, Bas R. Steunebrink, Jurgen Schmidhuber, Recurrent neural networks with Long Short-Term Memory (referred to as LSTMs) have emerged as an effective and scalable model for several learning problems related to sequential data. Earlier methods for attacking these problems have either been tailored towards a specific problem or did not scale to long time dependencies

LSTMs on the other hand are both general and effective at capturing long- term temporal dependencies. They do not suffer from the optimization hurdles that plague simple recurrent networks and have been used to advance the state-of-the-art for many difficult problems.

(i) Forward Pass:

Let xt be the input vector at time t, N be the number of LSTM blocks and M the number of inputs. Then we get the following weights for an LSTM layer:

- Input_weights: W, W, W, W ∈RN×M
- Recurrent_weights: R, R, R, R ∈RN×N
- Peephole weights: pi, pf , po ∈ RN
- Bias_weights: bz, bi, bf, bo ∈RN

Then the vector formulas for a vanilla LSTM layer forward pass can be written as:

Where σ, g and h are pointwise non-linear activation functions.

(ii) Back-propagation Through Time:

$$
\begin{aligned}
\delta\mathbf{y}^t &= \Delta^t + \mathbf{R}_z^T\delta\mathbf{z}^{t+1} + \mathbf{R}_i^T\delta\mathbf{i}^{t+1} + \mathbf{R}_f^T\delta\mathbf{f}^{t+1} + \mathbf{R}_o^T\delta\mathbf{o}^{t+1} \\
\delta\bar{\mathbf{o}}^t &= \delta\mathbf{y}^t \odot h(\mathbf{c}^t) \odot \sigma'(\bar{\mathbf{o}}^t) \\
\delta\mathbf{c}^t &= \delta\mathbf{y}^t \odot \mathbf{o}^t \odot h'(\mathbf{c}^t) + \mathbf{p}_o \odot \delta\bar{\mathbf{o}}^t + \mathbf{p}_i \odot \delta\bar{\mathbf{i}}^{t+1} \\
&\quad + \mathbf{p}_f \odot \delta\bar{\mathbf{f}}^{t+1} + \delta\mathbf{c}^{t+1} \odot \mathbf{f}^{t+1} \\
\delta\bar{\mathbf{f}}^t &= \delta\mathbf{c}^t \odot \mathbf{c}^{t-1} \odot \sigma'(\bar{\mathbf{f}}^t) \\
\delta\bar{\mathbf{i}}^t &= \delta\mathbf{c}^t \odot \mathbf{z}^t \odot \sigma'(\bar{\mathbf{i}}^t) \\
\delta\bar{\mathbf{z}}^t &= \delta\mathbf{c}^t \odot \mathbf{i}^t \odot g'(\bar{\mathbf{z}}^t)
\end{aligned}
$$

Here Δt is the vector of deltas passed down from the layer above. If E is the loss function it formally corresponds to dE/dyt, but not including the recurrent dependencies. The deltas for the inputs are only needed if there is a layer below that needs training, and can be computed as follows:

$$
\delta\mathbf{x}^t = \mathbf{W}_z^T\delta\bar{\mathbf{z}}^t + \mathbf{W}_i^T\delta\bar{\mathbf{i}}^t + \mathbf{W}_f^T\delta\bar{\mathbf{f}}^t + \mathbf{W}_o^T\delta\bar{\mathbf{o}}^t
$$

Recurrent neural networks with Long Short-Term Memory (referred to as LSTMs) have emerged as an effective and scalable model for several learning problems related to sequential data. Earlier methods for attacking these problems have either been tailored towards a specific problem or did not scale to long time dependencies

LSTMs on the other hand are both general and effective at capturing long- term temporal dependencies. They do not suffer from the optimization hurdles that plague simple recurrent networks. The central idea behind the LSTM architecture is a memory cell which can maintain its state over time, and non-linear gating units which regulate the information flow into and out of the cell. We evaluate the most popular LSTM architecture (vanilla LSTM; Section II) and eight different

$$
\begin{aligned}
\bar{\mathbf{z}}^t &= \mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z \\
\mathbf{z}^t &= g(\bar{\mathbf{z}}^t) && \textit{block input} \\
\bar{\mathbf{i}}^t &= \mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i \\
\mathbf{i}^t &= \sigma(\bar{\mathbf{i}}^t) && \textit{input gate} \\
\bar{\mathbf{f}}^t &= \mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f \\
\mathbf{f}^t &= \sigma(\bar{\mathbf{f}}^t) && \textit{forget gate} \\
\mathbf{c}^t &= \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t && \textit{cell} \\
\bar{\mathbf{o}}^t &= \mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o \\
\mathbf{o}^t &= \sigma(\bar{\mathbf{o}}^t) && \textit{output gate} \\
\mathbf{y}^t &= h(\mathbf{c}^t) \odot \mathbf{o}^t && \textit{block output}
\end{aligned}
$$

variants thereof on three benchmark problems: acoustic modeling, handwriting recognition, and polyphonic music modeling.

Yue-Gang Song, Yu-Long Zhou and Ren-Jie Han

Equity price prediction is regarded as a challenging task in the financial time series prediction process since the stock market is essentially dynamic, nonlinear, complicated, nonparametric, and chaotic in nature. The paper tells how neural networks can be used to make stock market predictions. There are five main neural network models discussed in this paper namely back propagation (BP) neural network, radial basis function (RBF) neural network, general regression neural network (GRNN), support vector machine (SVMR), least squares support vector machine regression (LS-SVMR). The models are used to predict stocks of namely Bank of China, Vance A and Kweichou Moutai.

Back Propagation Neural Network (BPNN)

The back propagation neural network (BPNN) is successfully used in many fields, such as engineering, stock index forecasting, stock price variation prediction. The proposed hybrid forecasting model (Wavelet Denoising-based Back Propagation), consists firstly of decomposed original data into multiple layers by wavelet transform, then a BPNN model was established by the low-frequency signal of each layer for predicting the Shanghai Composite Index (SCI) closing price. They use the activation function in its matrix form. The cost function is calculated by its quadratic form. The learning process in back propagation is speeded up using stochastic gradient descent. Stochastic gradient descent works by randomly picking out a small number m of randomly chosen training inputs. They label those random training inputs X1, X2, Xm, and refer to them as a mini-batch. Provided the sample size m is large enough it is expected that the average value of the rCXj will be roughly equal to the average over all rCx, where C is the cost function.

Radial Basis Function (RBFN)

The radial basis function neural network is a simple feedforward neural network consisting of only one hidden layer. The RBFNN is applied as a tool for nonlinear pattern recognition to correct the estimation error of the prediction of linear models in predicting two stock series in Shanghai and Shenzhen stock exchanges. RBFNN is proposed to overcome the main drawback of BPNN of easily falling into local minima in the training process. RBFNN have also been used in various forecasting areas and achieve good forecasting performance, with demonstrated advantages over BPNN in some applications. The hidden layer in RBFNN has non-linear activation function.

General Regression Neural Network

The GRNN shows its effectiveness in pattern recognition, stock price prediction and groundwater level prediction showed the forecasting ability of GRNN in the prediction of closing stock price. However, their research is lack of comparison with other data mining models, which is also the limitation of other references in this paper. If the number of neurons in the hidden layer stayed at the sample size n of training data on each prediction, we call it a static GRNN. Instead, as new observations come, we may increase the number of neurons. We call such a pattern as a dynamic GRNN. Dynamic GRNN was chosen in the current paper since it has stronger prediction power. A dynamic GRNN has long memory and updates timely.

Support Vector Machines

SVM is based on statistical learning theory. Owing to its successful performance in classification tasks and regression tasks , especially in time series prediction and financial related applications , it has drawn significant attention and thus earned intensive study. By using the structural risk minimisation principle to turn the solving process into a convex quadratic programming problem, the support vector machine, obtains better generation performance and moreover the solution is unique and globally optimal.

Least Square Vector Machines

LS-SVMR is based on structural risk minimisation principle, is able to approximate any nonlinear system. As a reformulation of the SVM algorithm, LS-SVMR overcomes the drawbacks of local optima and overfitting in the traditional machine learning algorithm.

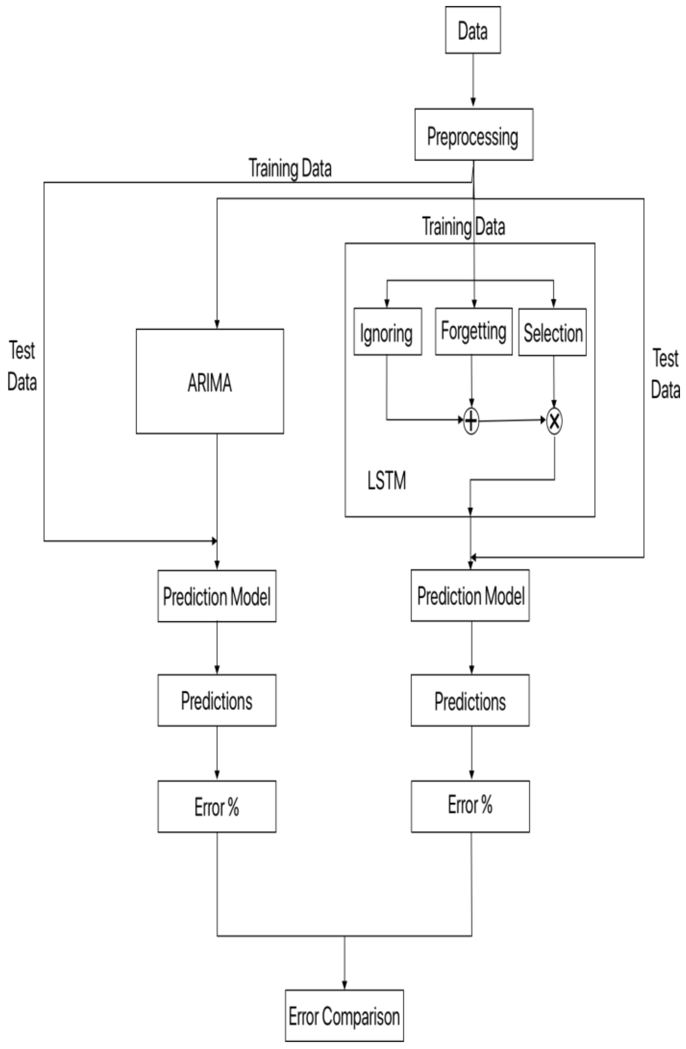
Data

The data used for comparison is the weekly stocks of Bank of China, Vance A and Kweichou Moutai. A three-layer neural network was used for prediction. The prediction of RBF and LS-SVMR is similar in case of Bank of China and Vance A. While on the prediction of Kweichou Moutai, LS-SVMR has a better performance. Overall, they share a similar accuracy level of prediction.

GRNN behaves worst consistently across the three stocks. In this work, they have successfully demonstrated that the five neural network models are all able to effectively extract meaningful information from past price. With evidence from forecast accuracy of three unrelated stocks, they find BP beats other our models consistently and robustly. Also, by implementing the algorithm many times and checking the standard deviation, the stability of BP is observed and confirmed. Based on our trial on different kernels. Thus, based on this paper we should not take the default kernel for granted and 'discretised' other kernels for BP algorithm, thus making all neural network models ineffective.

III. APPROACH

The dataset being utilized for analysis was downloaded from Yahoo Finance. The dataset consists of approximately 5 years of records of the required stock prices and other relevant values. The data reflected the stock prices at certain time intervals for each day of the year. It consisted of various sections namely date, symbol, open, close, low, high and volume. For the purpose of simulation and analysis, the data for only one company was considered. But we have used the Closing price for forecasting.



Data Block - This is the raw Data that will be given as input upon which preprocessing is done.

Preprocessing Block **-** This phase of the process is where the data will be operated with operations like slicing, normalization and various other operations so that only the required data is remaining. Stock has multiple attributes like Opening value, Highest value, Closing value, Date, etc. After preprocessing, the data will only contain the Closing value and the data.

Training Data - This is pre-processed data which will be used to train the model.

LSTM - The model will produce the predictions as output after going through series of processing nodes which will narrow down the predictions. It will take a stock's current data as input and will output the prediction that whether it will rise or fall.

Ignoring - This node is one of the three crucial nodes where the node eliminates/ignores all the unwanted data from the set of possible outcomes. Ignoring of data happens by overlapping possibilities and the previous predictions' data which will result in the possible outcome for the current data.

Forgetting - This is a node where the stored data is made forgotten so that there is a place for new pieces of information to remember. Forgetting of data happens depending on how long the data has been there. The longer the data is present, the higher the chances of it being forgotten.

Selection - This is the final node in the LSTM system, where the predictions are narrowed down to a single value depending on the latest value of the stock.

ARIMA – Stationarises the series, if necessary, by differencing (& logging, deflating, etc.), then studies the pattern of autocorrelations and partial autocorrelations to determine if lags of the stationarised series and/or lags of the forecast errors should be included in the forecasting equation. Fits the model that is suggested and checks its residual diagnostics, particularly the residual ACF and PACF plots, to see if all coefficients are significant and all of the patterns has been explained. Patterns that remain in the ACF and PACF may suggest the need for additional Auto-Regressive or Moving Average terms.

Prediction Model - This is the final model which will be able to output the predictions for any given data. The accuracy is different for different types of prediction models.

Predictions - Predictions of the data are collected here. All the predictions for the different stocks will be saved here.

Error% - Error is calculated here by comparing the true value and the predicted value.

Error Comparison - The error rate of the LSTM model is compared with the error rate of the ARIMA model.

A. Arima Model,

ARIMA uses a number of lagged observations of time series to forecast observations. A weight is applied to each of the past terms and the weights can vary based on how recent they are. ARIMA relies on Auto Regression. Autoregression is a process of regressing a variable on past values of itself. Autocorrelations gradually decay and estimate the degree to which white noise characterizes a series of data. It has a term called P-value. It can be found using Auto Correlation Function (ACF).

ACF is an (complete) auto-correlation function which gives us values of auto-correlation of any series with its lagged values. We plot these values along with the confidence band and tada! We have an ACF plot. In simple terms, it describes how well the present value of the series is related with its past values. A time series can have components like trend, seasonality, cyclic and residual. ACF considers all these components while finding correlations hence it's a 'complete auto-correlation plot'

PACF tells us the value of q. **PACF** is a partial auto-correlation function. Basically, instead of finding correlations of present with lags like ACF, it finds correlation of the residuals (which remains after removing the effects which are already explained by the earlier lag(s)) with the next lag value hence 'partial' and not 'complete' as we remove already found variations before we find the next correlation. So, if there is any hidden information in the residual which can be modeled by the next lag, we might get a good correlation and we will keep that next lag as a feature while modeling. Remember while modeling we don't want to keep too many features which are correlated as that can create multicollinearity issues. Hence, we need to retain only the relevant features.

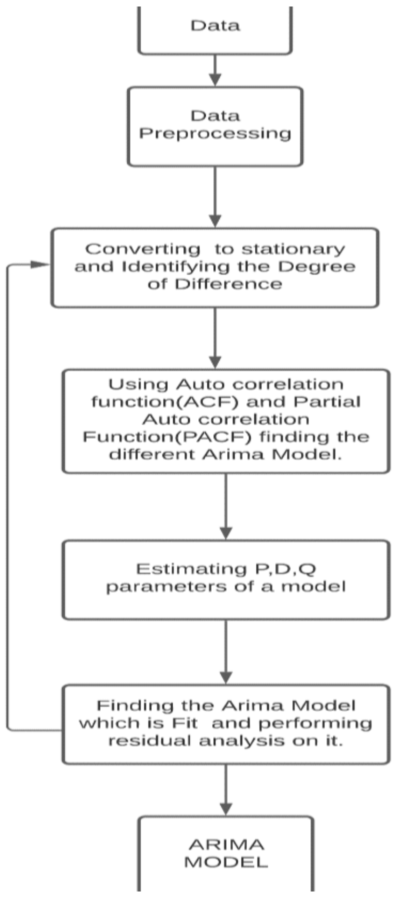

Data
Data Preprocessing
Converting to stationary and Identifying the Degree of Difference
Using Auto correlation function(ACF) and Partial Auto correlation Function(PACF) finding the different Arima Model.
Estimating P,D,Q parameters of a model
Finding the Arima Model which is Fit and performing residual analysis on it.
ARIMA MODEL

Fig. 1 Flow Chart for Arima Model

Integration: This is used for differencing of raw observations to allow for the time series to become stationery. From the graph of closing prices, we can see that cp(y+1) depends on cp(y) this implies that there is auto correlation present, but according to our assumption of regression there should not be any correlation. So, in order to remove this, we use this. It has a term called Q value and we take different values of q until we make the graph stationery like q= (1,2, 3...) so on.

Moving average (MA) removes non-determinism or random movements from a time series. This property represents Moving Average in ARIMA. It is expressed as MA(x) where x represents previous observations that are used to calculate current observation. Moving average models have a fixed window and weights are relative to the time. This implies that the MA models are more responsive to current event and are more volatile.

P (Auto Regressive), D(Integrated) and Q (Moving Average) are the three properties of the ARIMA model. Coefficients are calculated recursively. Model is chosen such that the estimated results calculated from the model are closer to the actual observed values. This process is iterative in nature.

B. LSTM

LSTM, short for 'Long Short-Term Memory, is an artificial recurrent neural network (RNN) architecture in which has feedback connections and is able to process entire sequences of data. LSTM networks are well-suited to classifying, processing, and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.

But LSTM [3], is a deep learning approach and where there are a huge array of parameters that you can tune in an effort to obtain optimal performance on most forecasting tasks. It uses feedback connections as it processes the whole data apart from treating each point independently, it keeps retaining useful information about previous data in the sequence to succor with the processing of new data points. Selective remembering of patterns for long durations of time makes LSTM a better option than other conventional feed-forward neural networks. LSTM model consists of three components or gates.
At first, the LSTM network output at distinct points in time depends on three things:
1. Cell state – It is the current long-term memory of the network.
2. Hidden state–The output of the LSTM network.
3. Input data – The current time step input.

The figure shows the structure of the LSTM model. Here at the particular point, the input is x(t), at the particular point, the output is h(t) and h(t-1) is the output of the previously hidden cell. The two inputs to the input gate are x(t) passed through the sigmoid function and tanh function which are multiplied and added to the resultant. The two inputs for the forget gates are x(t) and h(t-1) which are also passed through sigmoid and tanh functions which would help in discarding unnecessary information. The last gate, the output gate the outputs of the sigmoid function and tanh function are multiplied and the resultant is sent as output and also the hidden state of the next cell state.

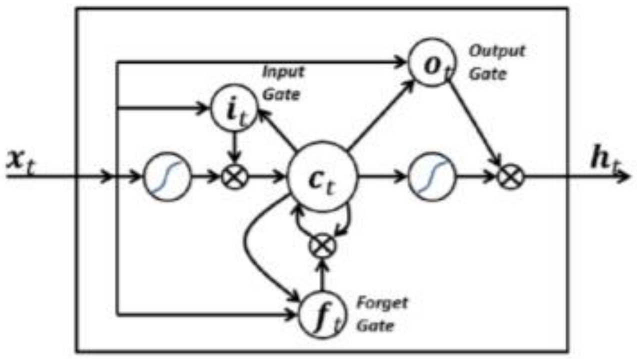


Figure no: 3 LSTM Architecture

Forward Pass in LSTM network:
Let x t be the input vector at time t, N be the number of LSTM blocks, and M the number of inputs. Then we get the following weights for an LSTM layer:
• Input weights: Wz, Wi, Wf , Wo ∈ R N×M
• Recurrent weights: Rz, Ri, Rf , Ro ∈ R N×N
• Peephole weights: pi, pf, po ∈ R N
• Bias weights: bz, bi , bf, bo ∈ R N
Then the vector formulas for a LSTM layer forward pass can be written as:

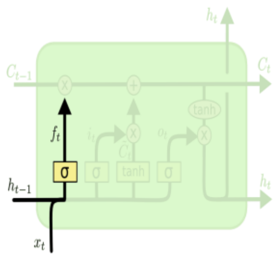
$$\begin{aligned}
\bar{\mathbf{z}}^t &= \mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z \\
\mathbf{z}^t &= g(\bar{\mathbf{z}}^t) && \textit{block input} \\
\bar{\mathbf{i}}^t &= \mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i \\
\mathbf{i}^t &= \sigma(\bar{\mathbf{i}}^t) && \textit{input gate} \\
\bar{\mathbf{f}}^t &= \mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f \\
\mathbf{f}^t &= \sigma(\bar{\mathbf{f}}^t) && \textit{forget gate} \\
\mathbf{c}^t &= \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t && \textit{cell} \\
\bar{\mathbf{o}}^t &= \mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o \\
\mathbf{o}^t &= \sigma(\bar{\mathbf{o}}^t) && \textit{output gate} \\
\mathbf{y}^t &= h(\mathbf{c}^t) \odot \mathbf{o}^t && \textit{block output}
\end{aligned}$$

Where σ, g, and h are point-wise non-linear activation functions.

Components of LSTM:

1. Forget gate: Forget gate is responsible for eliminating the information from the cell state. Two inputs enter the forget gate h(t-1) which is the hidden state from the previous cell and x(t) which is the input at the particular time step. The weight matrices and inputs

are multiplied and added by bias while applying the sigmoid function. The output of the sigmoid function ranges from 0 to 1, in correspondence to each number in the cell state. The sigmoid function determines which values to keep and which to discard. If the output is ‘0’ for a particular value in the cell state, i.e., forget gate wants the cell state to forget that piece of information. And if the output is ‘1’ then forget gate wants the cell state to retain the whole data. After which the multiplication of the cell state and vector output of the sigmoid function takes place.



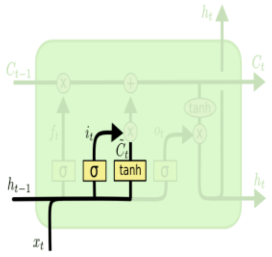
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

2. Input gate: This gate adds fresh data to the cells. This is done in three steps.

a) The values that are needed to be added are regulated using the sigmoid function. This phase is used to filter the data. information gleaned from the previous's hidden state output, h(t-1), and x(t) is the input that is similar to the functioning of the forget gate.

b) The values that must be added to the state of the cell are attained as a result of the vector that is produced using the tanh function.

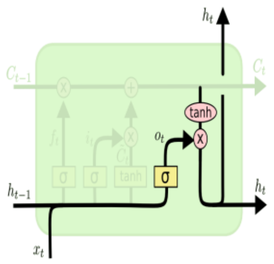
c)The sigmoid and tanh functions' outputs are multiplied, and the result is added to the cell state using the addition operation.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

3. Output gate: It extracts the appropriate data from the current state of the cell and displays it as the output. The functionality is accomplished in three steps:
a) To produce a vector to the right, the Tanh function utilizes a cell state with values ranging from -1 to 1
b) Using the filter to regulate the output values obtained from the vector formed in the first step.

c) The result of multiplication of the sigmoid function and tanh function is sent as the hidden state of the next cell state and also as the output.



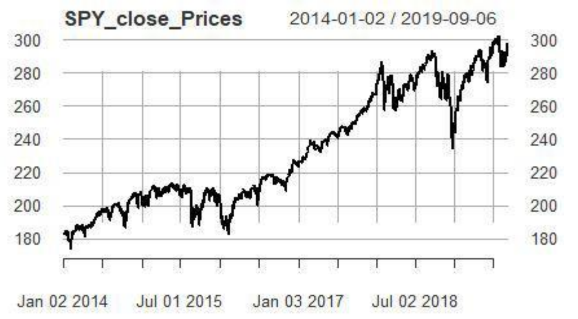
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

IV. RESULTS

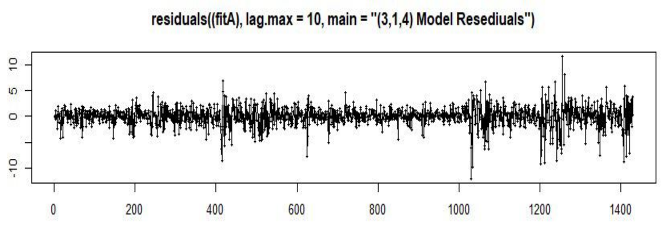
The proposed system is trained and tested over the dataset taken from Yahoo Finance. It is split into training and testing sets respectively and yields the following results upon passing through the different models:

A. *Results using ARIMA Model*

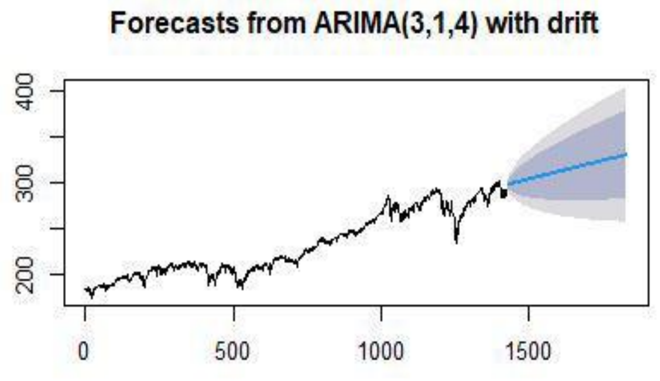
We have performed the ARIMA model in order to predict the stock closing price of the Yahoo finance data set.

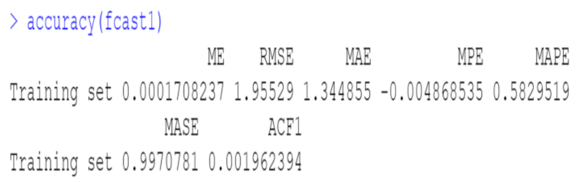


In order to remove auto Correlation and make it stationary we use this Integration and it has a value called ‘d’ term which can be changed to 1,2,3 and so on until it becomes stationary.



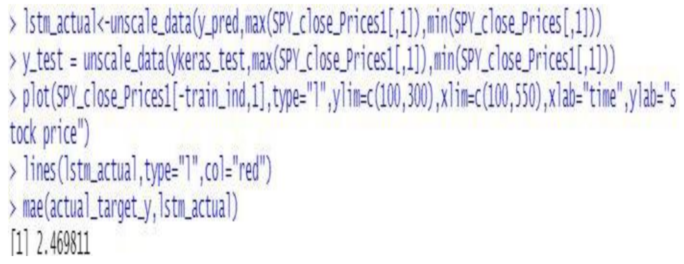
Finally, after trying different p, d, q values we got to know that 3,1,4 ARIMA model can better predict. The above graph is a forecasted result of closing price for 200 days.





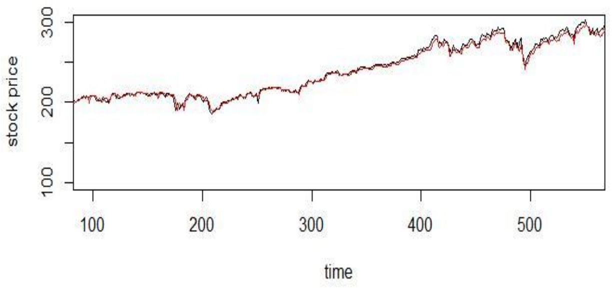
```
> accuracy(fcast1)
                      ME    RMSE      MAE          MPE      MAPE
Training set 0.0001708237 1.95529 1.344855 -0.004868535 0.5829519
                  MASE        ACF1
Training set 0.9970781 0.001962394
```

B. *Results using LSTM Model*
We have performed the LSTM model in order to predict the stock closing price of the Yahoo finance data set.

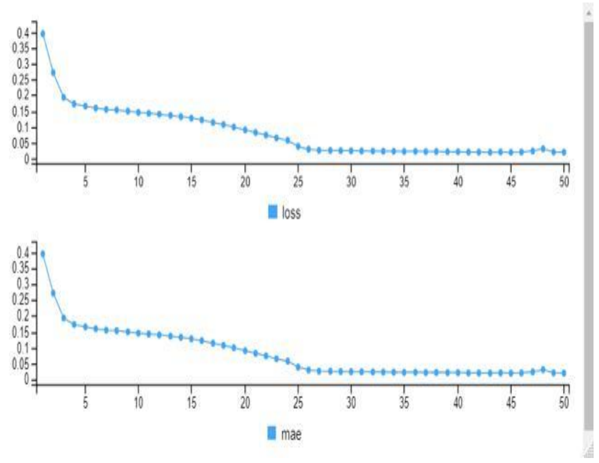


```
> lstm_actual<-unscale_data(y_pred,max(SPY_close_Prices1[,1]),min(SPY_close_Prices[,1]))
> y_test = unscale_data(ykeras_test,max(SPY_close_Prices1[,1]),min(SPY_close_Prices1[,1]))
> plot(SPY_close_Prices1[-train_ind,1],type="l",ylim=c(100,300),xlim=c(100,550),xlab="time",ylab="s
tock price")
> lines(lstm_actual,type="l",col="red")
> mae(actual_target_y,lstm_actual)
[1] 2.469811
```

Mean absolute Error for LSTM Model is 2.649811 and for ARIMA model is 1.344855. As the dataset is small ARIMA works better but, as the dataset size increases the LSTM tends to work better than ARIMA.



Here the red line indicates the stock price predicted by the LSTM Model.



The above graph represents a decline in Mean Absolute Error as the number of epochs increases, up to 5 epochs there is a steep decrease, then till 25 epochs there is steady decrease and remains almost stationary after 25 epochs.

V. Conclusion

Stock market prediction aims to determine the future movement of the stock value of a financial exchange. The accurate prediction of share price movement will lead to more profit investors can make. We have performed the ARIMA model and forecasted for 200 days.

ARIMA model can also be used for a wide variety of data sets as it first converts the data to a stationary form and has a better understanding of time series patterns. It is a powerful model as it is autoregressive i.e., it uses its past variables to make the forecast. Even though ARIMA model's accuracy is better than LSTM, the processing time required for ARIMA is much more. If devices with higher processing speed are used and data set with lesser empty values are used then ARIMA model can give better results.

REFERENCES

[1] K. S. Umadevi, A. Gaonka, R. Kulkarni and R. J. Kannan, "Analysis of Stock Market using Streaming data Framework," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 1388-1390, doi: 10.1109/ICACCI.2018.8554561.

[2] G. Bathla, "Stock Price prediction using LSTM and SVR," 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), 2020, pp. 211-214, doi: 10.1109/PDGC50313.2020.9315800.

[3] Neural Network For Stock Market Prediction by Yue-Gang Song, Yu-Long Zhou and Ren-Jie Han, 2018 arXiv

[4] A. Ashok and C. P. Prathibhamol, "Improved Analysis of Stock Market Prediction: (ARIMA-LSTM-SMP)," 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), 2021, pp. 1-5, doi: 10.1109/ICNTE51185.2021.9487745.

[5] S. A. Dwivedi, A. Attry, D. Parekh and K. Singla, "Analysis and forecasting of Time-Series data using S-ARIMA, CNN and LSTM," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2021, pp. 131-136, doi: 10.1109/ICCCIS51004.2021.93