A COMPARATIVE ANALYSIS OF STOCK VALUE PREDICTION USING DEEP LEARNING AND LSTM

Submitted in partial fulfillment for j component for the course of

Artificial Intelligence

by

Ramchander (17BCE1038)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

July,2021

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

DECLARATION

I hereby declare that the thesis entitled **A Comparative Analysis of Stock Value Prediction using Deep Learning and LSTM** submitted by me, for the project component of course, Artificial Intelligence is a record of Bonafide work carried out by me under the supervision of Dr. Richa Singh.

I further declare that the work reported in this thesis has not been submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai **Ram Chander**

Date: 16-07-2021 **Signature of the candidate**

ABSTRACT

Deep Learning is the future of computer technology, it will be used to perform various task across different fields. One of main sectors that can benefit the most from deep learning is the financial industry, especially the stock market. The stock market trades billions of dollars daily, based on anticipation and prediction, that may or may not turn profitable. Deep Learning and other new technologies can be applied to better help in the understanding of the trends of the stock market, and thereby make better predictions. In this project, various deep learning models were developed, analyzed and compared. The best model was chosen and was developed to predict the various stock trends, it was developed using the public data available from the National Stock Exchange on the NIFTY50 companies. Twitter was also to extract tweets about the required company from various new magazines and stock traders. Sentiment Analysis was done on the tweets to provide further information and support the predictions of the deep learning model.

Contents

Chapter 1

INTRODUCTION

1.1 STOCK MARKET

Stock market is one of the most profitable, yet volatile market in the financial sector. It deals with millions of transactions every day that involve stocks amounting to billions of United Stated Dollars (USD $). Stock market, in recent times, due to advancement of news delivery and communication technology, has proved very volatile and capable of drastic change within minutes of a particular news.

Stock exchanges are the main institutes of the global stock market, which are often regulated by government of that country. These stock exchanges are where the stocks are listed for the public trading. Examples of stock exchanges include the New York Stock Exchange (NYSE), National Stock Exchange (NSE), Bombay Stock Exchange (BSE).

Stock traders usually employ a variety of trading techniques to try and maximize their profit. One of the major ways of trading stock, was investing in stock and companies for the long term. Investors tend to buy stock at a low price, very early in the development of a company and wait for years, for the stock to grow in value, thus giving huge returns.

This kind of stock trading tends to work best when invested in a diversified portfolio, that is being invested in variety of stocks, to minimize the risk of complete loss. The disadvantage of this method is long waiting period, and diversification reduces the profits. That is why, many traders and hedge fund companies try to focus on day-to-day trading.

Day to day stock trading is a form of stock trading that focuses on holding stocks (positions) for a short term and maximize profit within the day. Day to day stock trading, mainly involves the method of pattern recognition and predictions of sudden rise or fall of stock value.

The traders do not diversify their portfolio, they tend to invest in one particular stock that they expect to increase in value within the short time, mostly within a day. The traders use various technology, data, graphs, news and social media to try and predict the future value of a particular stock and thereby maximize the profit.

1.2 TWITTER AND ONLINE COMMUNICATION

The growth of the computer technology and internet has fundamentally changed the workings of almost all the fields. One such field that it had massive effect on is the media and news industry. Generally, news takes a day or two to reach the public but with online news websites and online public forums such as Twitter, Facebook and Reddit news from halfway around the world reaches the public within seconds of publication.

Social Media has become so main steam now, that people get news directly from personal involved by following them on the different social media platforms such as Twitter, Facebook, etc. Twitter, mainly, used by various prominent personals and companies to convey and publish important information about various important topics.

1.3 EFFECT ON STOCK MARKET

As like other fields, stock market has also been drastically changed in its way of working by the recent yet huge development of online communication. Social media in particular has changed how traders get vital information for their job.

Twitter has changed the stock trading in 2 major ways, particularly for day-to-day stock traders. One being that, the traders must now be alert 24/7 for sudden and important tweet from a company, prominent personal, or the government that can have a drastic positive or negative impact on a particular stock and thereby must react immediately to it.

The other major effect, is the possibility for various traders across the world to form groups and communicate with each other and work together. They tend to discuss the various aspects and information available about a stock take a general stand on the movement of the stock, so all holding, buying or selling particular stock. This tends to move the stock in the desired path.

Therefore, the job description of a day-to-day stock trader has changed, from being able to predicts the future trend, to now include various other aspects of being able to use and obtain useful information from social media on the stocks. It also requires a constant 24/7 awareness of the news from around of the world, and how it will affect the stock.

1.4 DEEP LEARNING

Deep Learning is a subfield of Artificial Intelligence, that involves the usage of various algorithms for the development of a model, that is able to recognize a pattern in the given data and therefore be able to classify or predict for new data. Deep learning is used in various fields, from bank security to medical diagnosis, to help improve the accuracy and speed of decision making.

Deep Learning models have been proven to perform very well in recognizing a time series pattern that predict future value of the required variable. They tend to find the pattern that are difficult to analyze for the human brain, and therefore perform better in certain circumstances.

1.5 NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) is another subfield of Artificial Intelligence, that involves the processing of human language by the computers to interpret, analyze, understand and extract meaning details from the text.

Sentimental analysis is the process of classifying a particular text or statement, as positive or negative towards an object, by analyzing the words of the statement. Sentiment analysis is used in many places to automate and derive a conclusion about a product or company, based on the analysis of thousands of reviews/feedbacks that are available in human understandable languages.

1.6 PROJECT STATEMENT

As the expertise and work done by stock traders requires involves major tasks, which can be helped with the use of the currently available technologies. The project aims to develop different deep learning models for the prediction of stock value, in the short term. It also aims to provide the traders with the current tweets about various companies from news organizations and stock trend suggestion by other stock traders on twitter, and perform sentiment analysis of the tweets to give an overall report on the stock, with the prediction from the developed deep learning model and sentimental analysis of the current set of tweets.

1.7 OBJECTIVES

1. Develop and analyze various deep learning models for stock value prediction
2. Implement the best model for prediction
3. Track and extract twitter for tweets and information about the required stock.
4. Perform sentimental analysis on the tweets.

1.8 SCOPE OF THE PROJECT

The project aims to give information to traders about the stocks, their predicted trends and analysis of the news currently circulating about the stock or company, to help them make better decisions in managing their portfolio or to manage the risk in day-trading of the stocks

Chapter 2

Literature Review

2.1 Literature Survey

Title: Stock Market Prediction using Machine Learning

Author: Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikin Gupta, Himanshu Dhinam, Lokesh Chouhan [1]

This paper has implemented and compared two different machine learning algorithms for the prediction of stock values, based on the trend of the stock. The authors have obtained the dataset required for the training and testing of models, from the historical data available in Yahoo Finance Website. They have retrieved data in the CSV format, with the open, close, high, low, volume features given priority, and kept 20% data available for testing the models. They have implemented two models, namely Regression and Long Short-Term Memory (LSTM), trained and tested these models. The Regression model tried to form a linear equation, using the features as independent variables and the prediction as the dependent variable. The LSTM is an advancement of Recurring Neural Networks (RNN), which try for form a connection between the past values of the prediction value and the current value, this follows a hypothesis that the current value of the stock is dependent on the historical performance of the stock. The R-square confidence metric was used to score the regression model, while Root Mean Square Error was calculated for the LSTM model. After testing the models on the test data, separated earlier, the results are plotted for the purpose of comparing the models. Finally, they have concluded that the proposed LSTM architecture is more accurate and efficient than the regression model.

Title: Applying Long Short-Term Memory Neural Networks for predicting Stock Closing Price

Author: Tingwei Gao, Yueting Chai, Yi Lui [2]

In this paper, the authors have tried the hypothesis of predicting the next day's closing stock value using Recurring Neural Networks and informative input features. They have proposed and implemented a Long Short Term Memory model, which is an advanced RNN model for the prediction. They have used the data of the Standard and Poor (S&P500) and NASDAQ historical values for the training and testing of the models. The features of the dataset of include various stock market variable, such as the High Price which is the highest price of a stock during a given trading day, the Low Price which is the lowest price of a stock in the given trading day, the Open and Close price was also considered which represent the price of the stock in the opening and closing time of the exchange respectively. They also included the Adjusted Price (AD) which depicts the effect of the company actions on the stock price. Using this data, they prepared a simple LSTM model, which the architecture of an input layer, Single LSTM hidden layer, and an output layer. They have used the ReLU activation function for the activation of the neuron with the threshold at zero, they have also used the Adam optimizer, because its easy to implement, efficient in computations, low memory requirement and is invariant to diagonal rescaling of the gradients. Therefore, it is optimal for problems such as the stock trend which deal with huge amount of data and parameters. This model was trained and tested on the data gathered, metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE) and Average Mean Absolute Percentage Error (AMAPE) was calculated and analyzed. The authors have reached the conclusion that general trend of the stock market can be predicted using LSTM model within acceptable error percentage.

Title: Stock Market Forecasting using Machine Learning: Today and Tomorrow

Author: Sukhman Singh, Tarun Kumar Madan, Jitendra Kumar, Ashutosh Kumar Singh [3]

In this paper, the author has compared and analyzed various different proposed methods for the prediction on stock market trend. They have also proposed new features and variables that they will be important in increasing the performance and accuracy of the available models. They have proposed the inclusion of analysis of the public sentiments, government initiatives, or other important events such as disasters or elections. These external factors, are used by stock traders regularly while evaluating their positions on the market, so they must also be considered by the model in predicting the trend. They have analyzed the results of multiple different techniques and compared them across different metrics and features used in it. They have compared Classification techniques, regression models, Ensemble techniques, Evolutionary models, Deep Learning methods and Hybrid Methods. In Classification techniques, they have compared Support Vector Machine (SVM) and Multi-Source Multiple Instance Learning (MMI). In Regression models, they have compared Linear Regression with an Artificial Neural Network, the Multi-Layer Perceptron (MLP). In Ensemble techniques, they have compared a Random Forest Model with Boosted Decision Tree model, which gave a better performance than SVM and Linear regression. The Evolutionary techniques of, Generic algorithm and Particle Swarm Optimization (PSO) was compared, the PSO proved better than other models under certain circumstances of the market. Some Hybrid models, like Support Vector Regression and Artificial Neural Network, were also compared, along with Deep Learning Models such as Convolutional Neural Networks and LSTM. It was concluded the Hybrid Models provide more accurate results but are computationally intensive process.

Chapter 3

PROPOSED METHODOLOGY

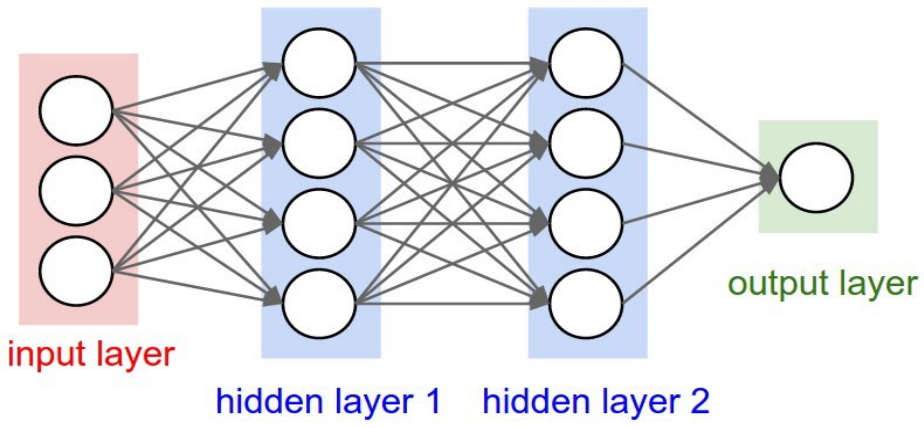
3.1 Proposed Methodology

In this project, three different models are planned to be implemented, analyzed and compared. Those are, a basic Deep Neural Network (DNN), Recurrent Neural Network (RNN), and a Long Short-Term Memory Model (LSTM). These models will used for the prediction of the trend of the stock market. Then tweets will be imported from Twitter for the sentiment analysis using python.

3.2. Deep Neural Network

All Artificial Neural Networks are made up 3 types of layers, the input layer, hidden layers and the output layer. A deep neural network is any artificial neural network that has more than one hidden layer.

DNN are generally feedforward networks, that is data usually flows in one direction, from input layer to output layer. The DNN calculates the output by performing matrix vector multiplication of the input sequence and the edge weights added with the bias of each neuron.



3.2.1 Input Layer

The input layer, is common to all neural networks and is the first layer in the network. It is designed in such a way to match the number of features in the input sequence. It takes the initial data for calculation from the user or external source and passes it on as a vector to next layer, the first hidden layer, for calculation

3.2.2 Hidden Layers:

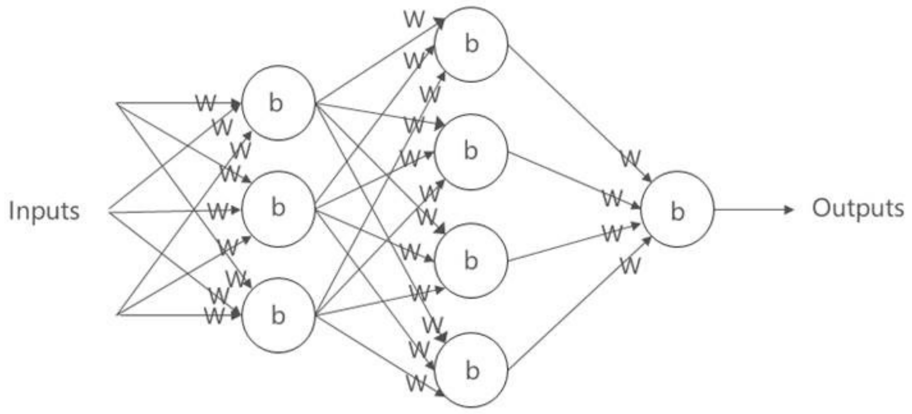
The layers in between the input layer and the output layer are known as the hidden layers. The neurons in this layer contain values known as bias, the edge weights are multiplied with the input values and added with the buyers of the neuron and the value calculated, is forwarded to the activation function if the value satisfies the activation function the neuron is activated and the output value is pause to the next layer. The number of layers in the hidden layer part of the neural network can vary from 1 to any number and every layer can be of any type example Dense, SimpleRNN, LSTM.

3.2.3 Activation Function

The activation function is the main function of a neuron responsible for calculating the output based on the given inputs. There are various types of activation function that are useful in different situations. The activation functions decide whether the neurons should be activated, that is whether the output of the neuron should be pause down to the next layer. Some of the common activation functions are Rectified Linear Activation (RELU), Hyperbolic Tangent (Tanh) and Logistic activation.

3.2.4 Neuron Weights

Every neuron contains a trainable parameter known as weight, the weight value, with the bias, is used in calculation of the output by the activation function. On initialization of the neural network, the neuron weights are assigned random values. During the training, the weight values are changed to match the output with the actual value. The value of the weight affects the scale in which the input affects the output, smaller the weight value smaller the influence of the input on the output value. As seen in the figure (Fig 3.4) below, the weights are represented by W and the bias by B.



3.2.5 Output Layer

The output layer is the final layer of an artificial neural network. The value calculated by the output neurons is expected to match the actual values during training. The number of neurons in the output layer should match the number of expected outputs per input sequence.

Deep neural networks are robust, they can usually overcome the errors in the data. For example, small errors in the training data do not affect the outcome as seen in machine learning models. deep neural networks are usually fast in the calculation of the target variable.

Deep neural networks require a lot of computational power and usually carry out processes in parallel therefore require a lot of parallel processors to speed up the calculations.

3.3 RECURRENT NEURAL NETWORK

Recurrent neural network is a special type of artificial neural network that considers the output of the previous step in calculation of the output of the present step. RNN is mainly used in predicting statements where to predict the next word the previous words in the statement need to be considered. RNNs usually have a parameter called memory which remembers the information about the previously calculated outputs which is used with the new inputs to perform the same task.

A decision taken by the recurrent network at time T - 1 affects the decision that will be taken at time T. So recurrent networks can be thought to have two sets of input, The present input sequence and recent past output. Recurrent networks defer from feedforward networks by the presence of a lookback edge that makes the output off the previous sequence considered for the current sequence. This is sometimes referred to as a memory, that is recurrent networks are thought to have stored the output in their memory for further calculations.

The information of the sequences is stored in in the recurrent networks hidden state, where it exists for a long time affecting the calculations of the next sequences. The recurrent network is trying to find a correlation between the different inputs and their corresponding outputs like a series of events, this is known as long term dependency events.

3.3.1 Backpropagation through time

Backpropagation through time (BPTT) is the technique used in the recurrent neural networks to update the weights of the neurons during the training phase. The goal of the backpropagation through time is to minimize the error of the output.

Backpropagation through time is usually applied to time series data, which have usually shown the dependence of the present value on the previous historical values.

Some of the disadvantages of add propagation through time are, it requires her huge amount of computational power it also requires a lot of iterations to update the values of the rates this also tends to make the model too noisy or overfit.

Truncated backpropagation through time is an advancement of the oh backpropagation through time algorithm to reduce the impact of oh the memory parameter but still keep it significant. In this method the input is process wondered that time and the parameters are updated only after a fixed number of time steps.

Truncated backpropagation through time uses 2 parameters to determine the influence of memory parameters, namely K1 and K2. K1 specifies the number of forward pass time steps before an update to the parameters of the networks such as weights and bias. This influences the speed of the training process and the number of times the weights are updated. care to specify the total number of time steps for them backpropagation through time algorithm, it must be large enough to allow the network to learn properly but yet be small enough to avoid overfitting and reduce time and computational requirements

3.3.2 Problems in RNN

One of the most common problems in RNN is the vanishing gradient problem. As during the backpropagation algorithm, the weights are updated proportional to the partial derivative of the error and the current weight during every iteration of the backpropagation algorithm.

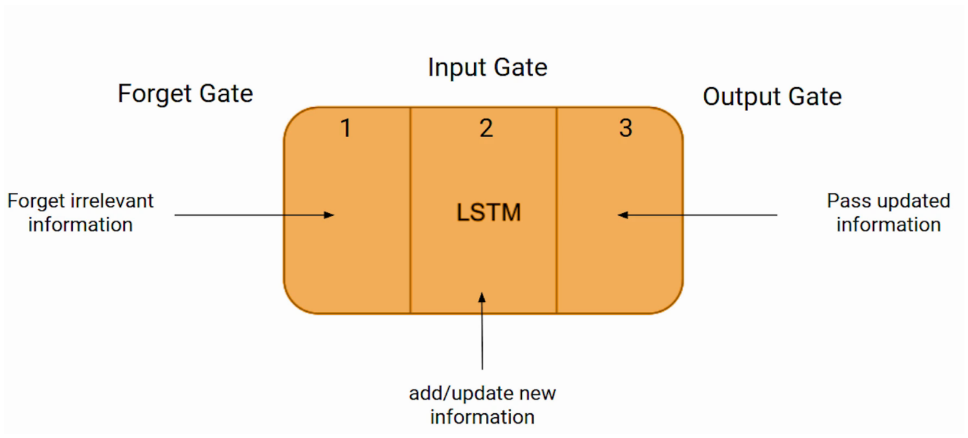
When the gradient becomes too small it affects the change to the weight therefore preventing further training of the network. In the worst-case scenario, the gradient becomes too small to have any meaningful effect on the change of the weights and this scenario is known as vanishing gradient problem.

The reverse scenario of the vanishing gradient problem, the situation where the gradient of error becomes too high and thus affecting the weight too much. This scenario causes the model to become unstable and overfit for a particular training data set, this scenario is known as gradient explosion problem.

The gradient explosion problem can be overcome by the use of activation functions such as sigmoid function, ReLU function that prevent the gradient value from reaching a value more than a set limit.

3.4 Long Short-term memory models

Long short-term memory networks or an advancement to the recurrent neural networks, that is still capable of handling sequential time series data but can overcome the banishing gradient problems experienced in recurrent neural networks. LSTM neurons also work similar to RNN neuron but have three gates that make it more efficient than a RNN neuron



As seen in the image above and LSTM neuron is made of three gates, the forget gate the input gate and the output gate. Like any RNN neuron, the LSTM neuron gets input from the sequence as well as the previous output for its calculations but the forget gate now determines whether the output sequence is required for this calculation and thus discards unwanted historical sequences.

The input sequence is fed to the input gate after being filtered by the forget gate and calculations performed by the activation function the calculated value is passed down to the output gate and touch forward to the other layers. This method helps overcome the vanishing gradient problem.

3.4.1 Advantages and Disadvantages of LSTM

Advantages:

1. The constant error backpropagation in LSTM allows for them to understand complex sequences and time series data efficiently.
2. LSTM networks usually handle noise in the data very well they tend not to overcompensate to fit the data.
3. LSTM has a huge number of parameters that can be edited, such as input bias, output bias and learning rate, that helps the model to be altered easily for the given task.

Disadvantages

1. Even though LSTM's can handle gradient vanishing problems they do not completely eliminate the problem they are also prone to the problem.
2. LSTM models require huge computational resources and time to be trained for real world applications like speech recognition time series predictions.
3. LSTM models do usually overfit the training data.

3.5 Activation functions

An activation function is the most important function of a neuron it basically calculates the output of the neuron based on the input given, the weights and the bias. Depending on the output value, the activation function determines whether the neuron is to be activated or not. there are various types of activation functions that pass the output after modification to the next layer.

Rectified Linear Unit function (ReLU):

The rectified linear unit function is an activation function that helps in reducing the vanishing gradient problem in many networks. The ReLU function works like gives a 0 if the value calculated is negative and gives the value itself if the calculated value is positive. It is one of the most common and useful activation functions available.

$$f(x)= \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x => 0 \end{cases}$$

3.6 Sentiment Analysis

Sentiment analysis is one of the major subfields of natural language processing. It builds with the ability to get meaningful information and contextual sentiments from a statement regarding an entity, company or product. Sentiment analysis use is natural language processing, linguistics and analysis to classify various statements on their meanings.

Sentiment Analysis is majorly used in understanding and creating a conclusion from various customer reviews in various businesses and movies. It aims to categorize the various reviews in two fields of positive comments negative comments and neutral comments. It tries to give a score, averaging the values of across various reviews, which is usually a time taking toss for humans.

In any sentiment analysis the first step carried out is part of speech tagging categorized into various grammatical parts such as nouns verbs and adjectives. the model then understands which part of the statement is being reviewed the noun specifically or sometimes the verb. The adjective indicates is the sentiment of the review or statement, this is done by the use of certain words being already categorized to be either positive or negative in the word bank.

Automatic Sentiment Analysis

An automated sentiment analysis is the best way to get the most out of your messages. The automatic approach involves the use of supervised machine learning algorithms. In other words, it works by taking advantage of the various advantages of machine learning. Due to this, the operation becomes more precise and accurate. It allows you to process more information without getting too complex.

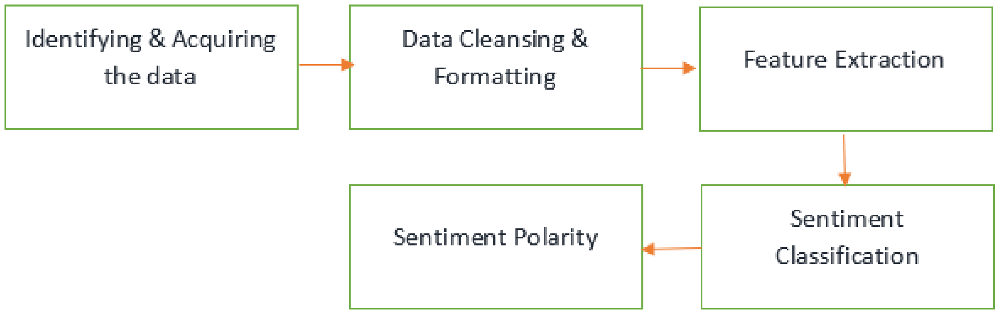
Automatic sentiment analysis usually incorporates a classification-based machine learning algorithm such as:

- Logistic Regression
- Support Vector Machines
- LSTM and GRU (Deep Learning)
- Naïve Bayes

Common Challenges in Sentiment Analysis

Some of the common challenges in sentiment analysis are

- Understanding the context of the statement
- Subjectivity of the statement
- Sarcasm
- Classification of Neutral statements



3.7 Twitter

Twitter is a social media platform where users can post and interact with each other. It allows for anyone in the world, from anywhere in the world with internet connection to register and share their ideas, interact with people, in the form of small texts known as tweets. Its platform allows unregistered users to only read tweets.

In 2012, more than 100 million people used Twitter each day. By 2013, it was the most-visited website in the world. As of Q1 2019, it had more than 330 million active users.

Twitter is recognized as one of the most powerful and open developer APIs in the world. Its public API was released in September 2006, and it became a reference implementation for other developers. From 2006 to 2010, Twitter's developer platform grew at a fast pace, creating the first mobile phone clients and the first URL shortener. In 2010, Twitter required all developers to adopt OAuth authentication within 9 weeks of notice.

Twitter allows access to various features such tweet search, accessing tweet contents, retweeting and many more, to developers using their open APIs. Since twitter has a lot influential personals, companies tweeting various important information, it has proved to be a data mine for all things and news.

Some of the feature available as developer APIs is:

- Tweets
- Users
- Direct Messages
- Lists
- Trends
- Media
- Places

Tweets from various accounts have been extracted and fed to Natural Language processing models to extract and live track important information in the social media about various topics.

Various programming language have many different libraries that help to access twitter APIs and integrate twitter into required project. One such popular library for python programming language is the tweep library.

3.7.1 Tweepy

Tweepy is an open sources python package, that is developed specifically for the integration of twitter into python applications. Tweepy is defined with various classes and methods than can access and make use of almost all twitter functionalities. Some of the twitter functionalities that can be used in tweepy are:

- Data encoding and decoding
- HTTP requests
- Results pagination
- OAuth authentication
- Rate limits

Chapter 4

System Requirements

4.1 Hardware Requirements

As for any machine learning or deep learning project, this project also requires a lot of memory for data storage. A minimum set of hardware requirements are

- 8GB RAM
- 7th gen or later desktop processors
- A good internet connection

A Graphical Processing Unit can help in speeding up the mathematical calculations

4.2 Software Requirements

- Operation System: Windows 10 or Ubuntu or MacOS
- Python 3.7 or later
- A python IDE would help with the development of the project. Google Colab was used in this project for development. It is a free to use to web IDE for python, developed by Google. It also provides for memory and GPU for running the project. It can be connected to google drive for access of memory space, for storing files and datasets.
- Python Libraries used are: Pandas, Numpy, MatplotLib, Go, Keras, Sklearn,etc

Chapter 5

Implementation Modules

5.1 Data Collection

This project mainly used the data about the NIFTY50 stock prices and various other features corresponding to the stocks for training of the neural networks. First data about the stocks, including features like Open price, close price, High Price, Low Price, Volume traded, were collected from the National Stock Exchange for the time period 2007 – 2020. This dataset was used for training the data. This dataset contains 50 '.csv' files each containing about 3200+ rows of data.

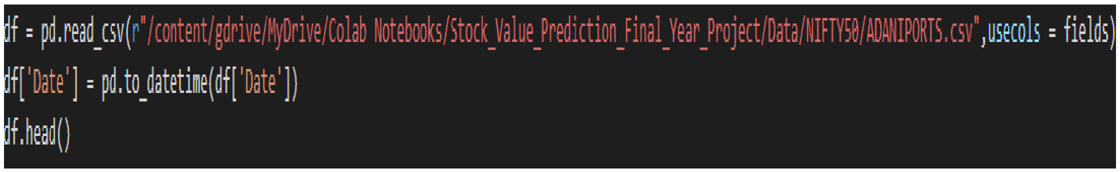
Current stock prices of the these were also downloaded from the Yahoo Finance website for testing and prediction purposes.

5.2 Feature Selection

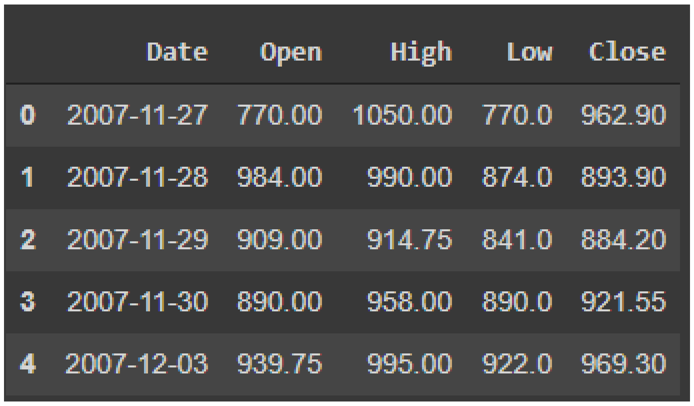
Based on the research and analysis done, only certain features from the dataset were considered for the training and development of the model. The chosen features are

- High Price
- Low Price
- Open Price
- Close Price

These features were used against the date of the trade to create a timeline series values of each feature.



```
df = pd.read_csv(r"/content/gdrive/MyDrive/Colab Notebooks/Stock_Value_Prediction_Final_Year_Project/Data/NIFTY50/ADANIPORTS.csv",usecols = fields)
df['Date'] = pd.to_datetime(df['Date'])
df.head()
```



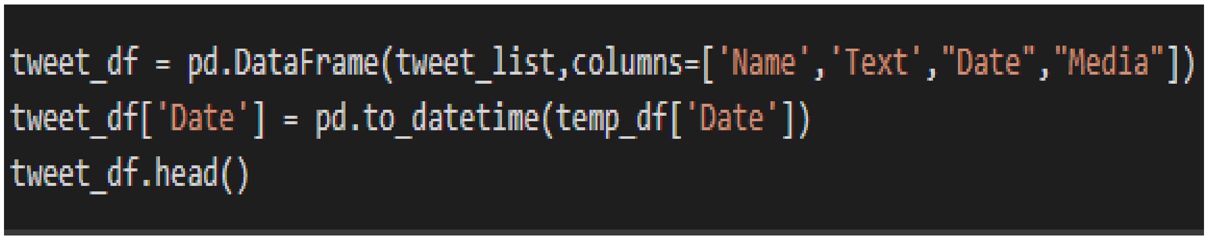
	Date	Open	High	Low	Close
0	2007-11-27	770.00	1050.00	770.0	962.90
1	2007-11-28	984.00	990.00	874.0	893.90
2	2007-11-29	909.00	914.75	841.0	884.20
3	2007-11-30	890.00	958.00	890.0	921.55
4	2007-12-03	939.75	995.00	922.0	969.30

In obtaining tweets using the twitter API, the tweet from some major news magazines like Economics Times, Forbes India were given priority. If they have not tweeted anything about the required company, then tweet from stock traders were extracted.

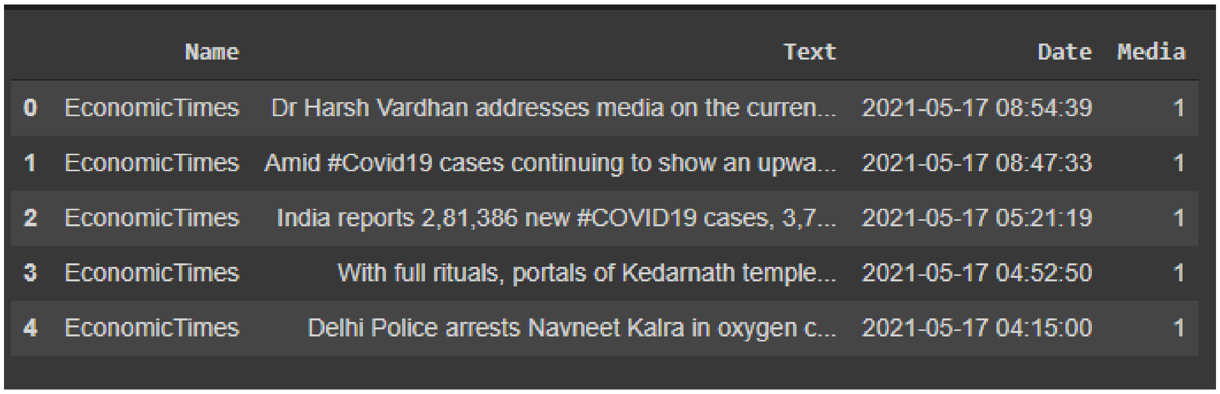
Only certain details about the tweets were extracted from the search query and written to a .csv file for the preprocessing. The extracted features are:

- Username / ScreenName
- Text of the tweet
- Date of the tweet creation

A new column media was added, to categorize the source of the tweet.



```
tweet_df = pd.DataFrame(tweet_list,columns=['Name','Text',"Date","Media"])
tweet_df['Date'] = pd.to_datetime(temp_df['Date'])
tweet_df.head()
```

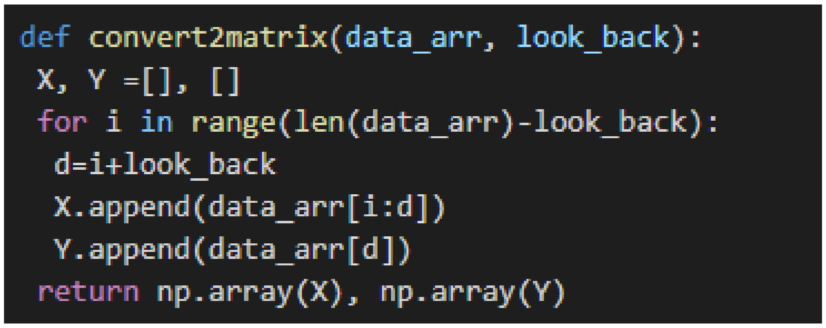


```
            Name                                               Text                 Date  Media
0  EconomicTimes  Dr Harsh Vardhan addresses media on the curren...  2021-05-17 08:54:39      1
1  EconomicTimes  Amid #Covid19 cases continuing to show an upwa...  2021-05-17 08:47:33      1
2  EconomicTimes  India reports 2,81,386 new #COVID19 cases, 3,7...  2021-05-17 05:21:19      1
3  EconomicTimes      With full rituals, portals of Kedarnath temple...  2021-05-17 04:52:50      1
4  EconomicTimes     Delhi Police arrests Navneet Kalra in oxygen c...  2021-05-17 04:15:00      1
```

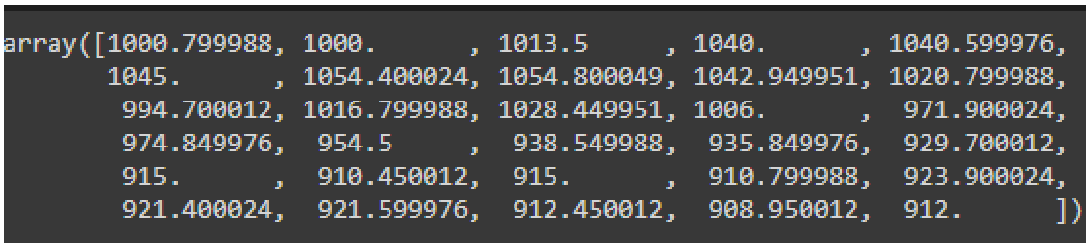
5.3 Data Preprocessing

The stock data was all in the same scale, with all price in Indian Rupees and also did not need much of cleaning as the data was obtained from government data source. One of the issues with the dataset is the fact somedays the stock is closed, and the stock prices are not available for those days.

The data was converted, into a matrix with values of the previous 30 days making the features and the current day value is made the target value.

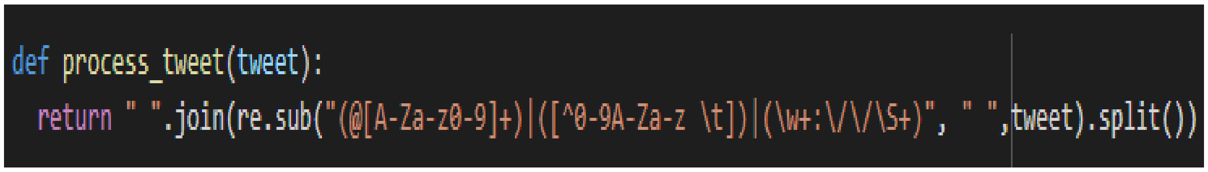


```
def convert2matrix(data_arr, look_back):
 X, Y =[], []
 for i in range(len(data_arr)-look_back):
  d=i+look_back
  X.append(data_arr[i:d])
  Y.append(data_arr[d])
 return np.array(X), np.array(Y)
```

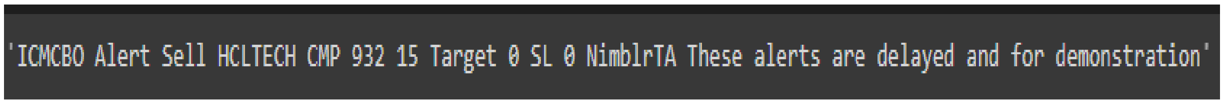


```
array([1000.799988, 1000.      , 1013.5     , 1040.      , 1040.599976,
       1045.      , 1054.400024, 1054.800049, 1042.949951, 1020.799988,
        994.700012, 1016.799988, 1028.449951, 1006.      ,  971.900024,
        974.849976,  954.5     ,  938.549988,  935.849976,  929.700012,
        915.      ,  910.450012,  915.      ,  910.799988,  923.900024,
        921.400024,  921.599976,  912.450012,  908.950012,  912.      ])
```

In the twitter data extracted and stored in the csv file, the text needed to be cleaned. Unwanted special characters, hashtags and emojis needed to be removed. This was done using the regular expressions in python. After cleaning, the text was added as a new column to the dataframe, so that the original text could still be displayed if required later.



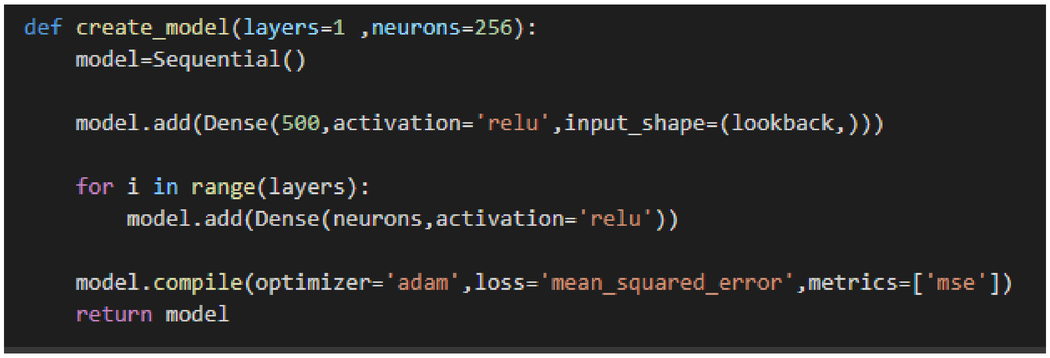
```
def process_tweet(tweet):
  return " ".join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ",tweet).split())
```



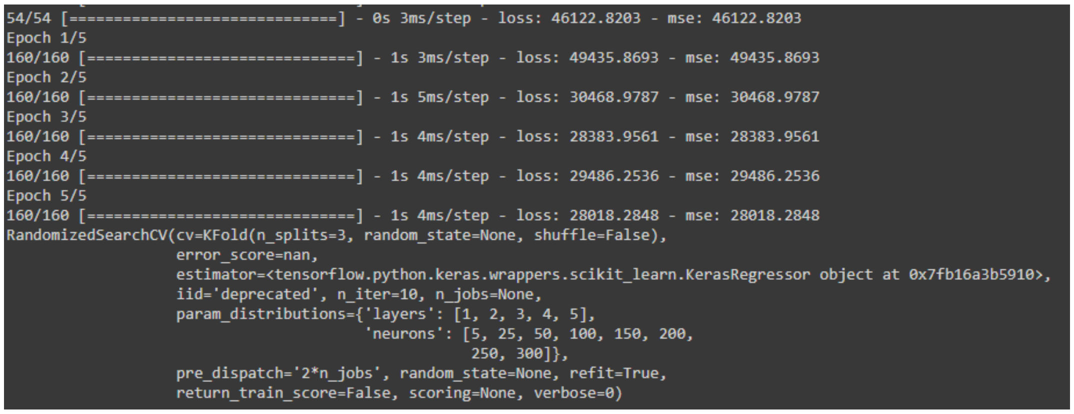
```
'ICMCBO Alert Sell HCLTECH CMP 932 15 Target 0 SL 0 NimblrTA These alerts are delayed and for demonstration'
```

5.4 Models

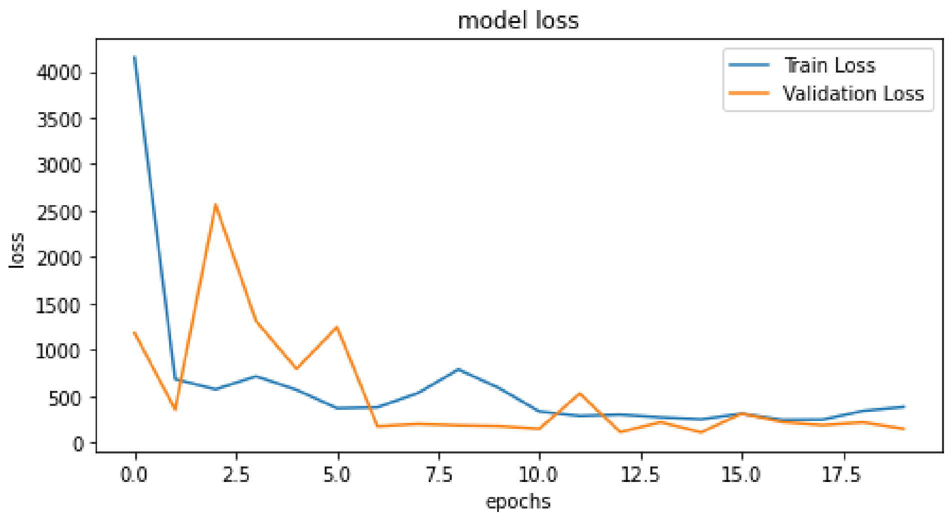
DNN Model

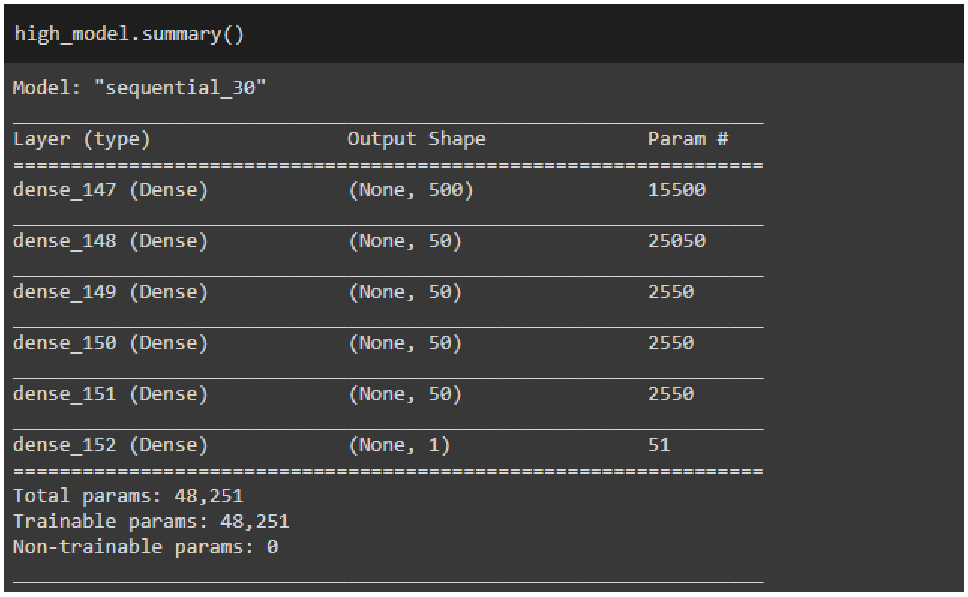


```
def create_model(layers=1 ,neurons=256):
    model=Sequential()

    model.add(Dense(500,activation='relu',input_shape=(lookback,)))

    for i in range(layers):
        model.add(Dense(neurons,activation='relu'))

    model.compile(optimizer='adam',loss='mean_squared_error',metrics=['mse'])
    return model
```

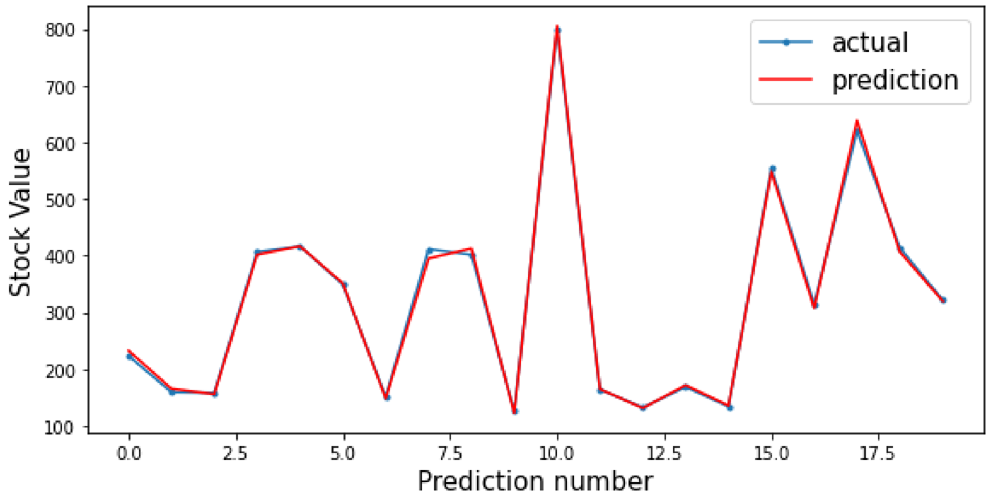


```
54/54 [==============================] - 0s 3ms/step - loss: 46122.8203 - mse: 46122.8203
Epoch 1/5
160/160 [==============================] - 1s 3ms/step - loss: 49435.8693 - mse: 49435.8693
Epoch 2/5
160/160 [==============================] - 1s 5ms/step - loss: 30468.9787 - mse: 30468.9787
Epoch 3/5
160/160 [==============================] - 1s 4ms/step - loss: 28383.9561 - mse: 28383.9561
Epoch 4/5
160/160 [==============================] - 1s 4ms/step - loss: 29486.2536 - mse: 29486.2536
Epoch 5/5
160/160 [==============================] - 1s 4ms/step - loss: 28018.2848 - mse: 28018.2848
RandomizedSearchCV(cv=KFold(n_splits=3, random_state=None, shuffle=False),
                   error_score=nan,
                   estimator=<tensorflow.python.keras.wrappers.scikit_learn.KerasRegressor object at 0x7fb16a3b5910>,
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'layers': [1, 2, 3, 4, 5],
                                        'neurons': [5, 25, 50, 100, 150, 200,
                                                    250, 300]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=0)
```

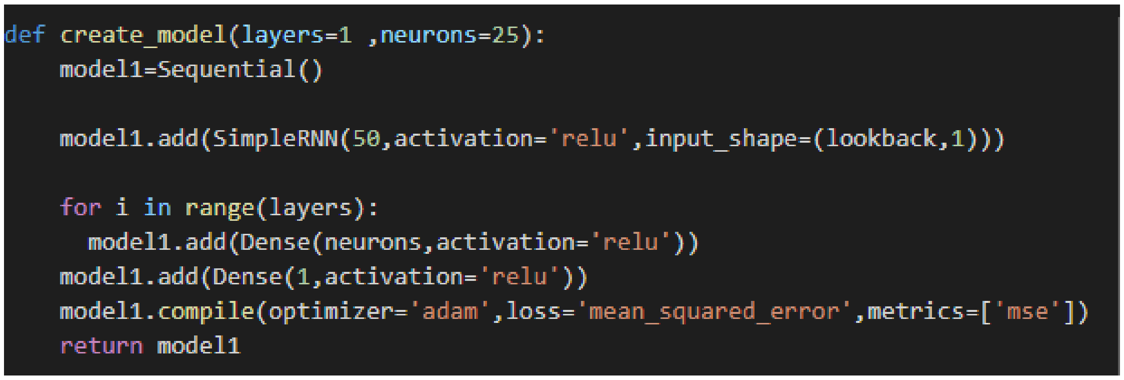




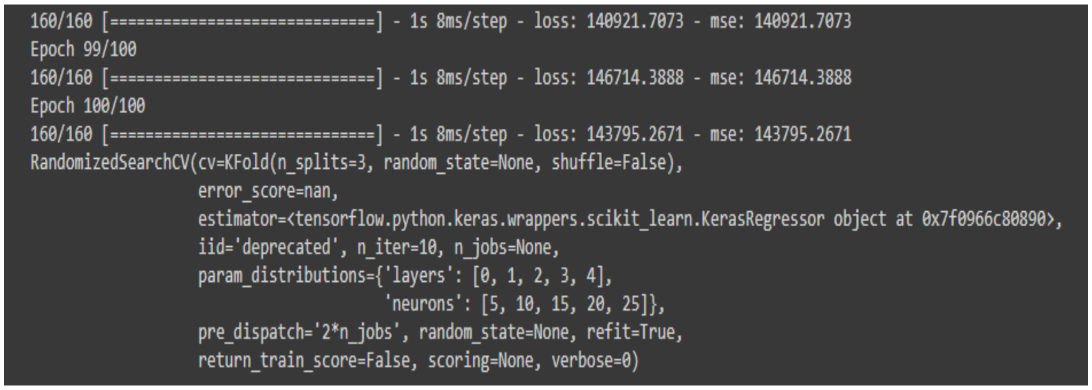
```
high_model.summary()

Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_147 (Dense)            (None, 500)               15500     
_________________________________________________________________
dense_148 (Dense)            (None, 50)                25050     
_________________________________________________________________
dense_149 (Dense)            (None, 50)                2550      
_________________________________________________________________
dense_150 (Dense)            (None, 50)                2550      
_________________________________________________________________
dense_151 (Dense)            (None, 50)                2550      
_________________________________________________________________
dense_152 (Dense)            (None, 1)                 51        
=================================================================
Total params: 48,251
Trainable params: 48,251
Non-trainable params: 0
_________________________________________________________________
```



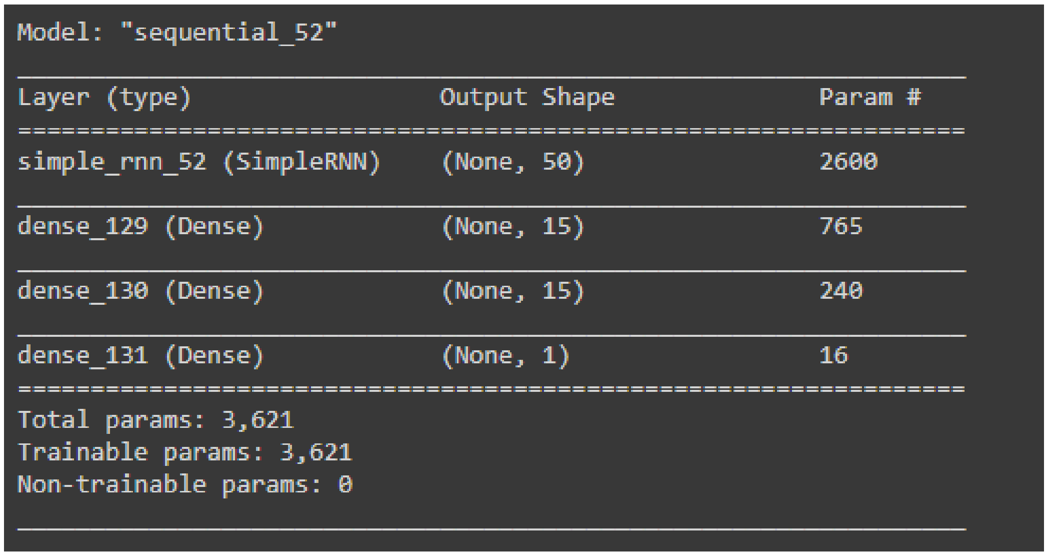
RNN Model



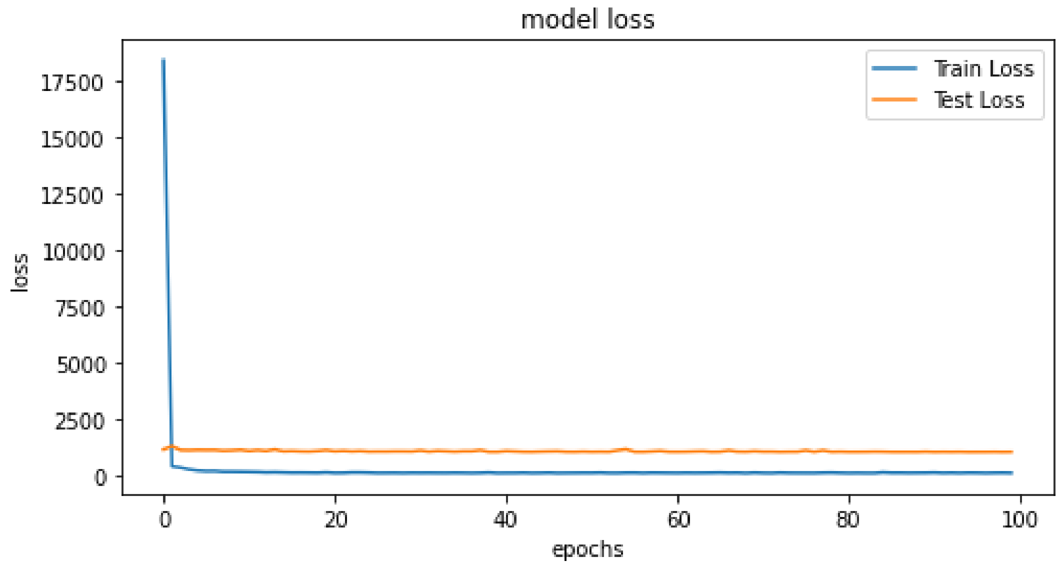
```
def create_model(layers=1 ,neurons=25):
    model1=Sequential()

    model1.add(SimpleRNN(50,activation='relu',input_shape=(lookback,1)))

    for i in range(layers):
      model1.add(Dense(neurons,activation='relu'))
    model1.add(Dense(1,activation='relu'))
    model1.compile(optimizer='adam',loss='mean_squared_error',metrics=['mse'])
    return model1
```



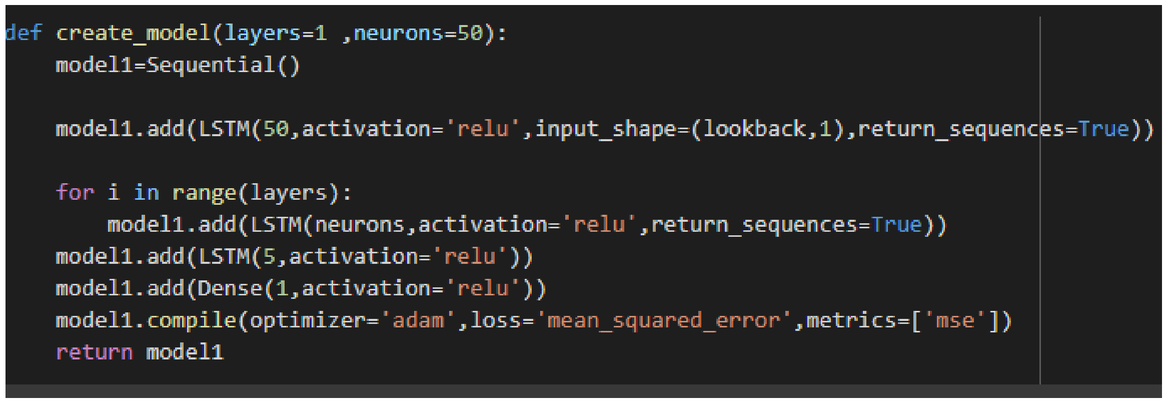
```
160/160 [==============================] - 1s 8ms/step - loss: 140921.7073 - mse: 140921.7073
Epoch 99/100
160/160 [==============================] - 1s 8ms/step - loss: 146714.3888 - mse: 146714.3888
Epoch 100/100
160/160 [==============================] - 1s 8ms/step - loss: 143795.2671 - mse: 143795.2671
RandomizedSearchCV(cv=KFold(n_splits=3, random_state=None, shuffle=False),
                   error_score=nan,
                   estimator=<tensorflow.python.keras.wrappers.scikit_learn.KerasRegressor object at 0x7f0966c80890>,
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'layers': [0, 1, 2, 3, 4],
                                        'neurons': [5, 10, 15, 20, 25]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=0)
```



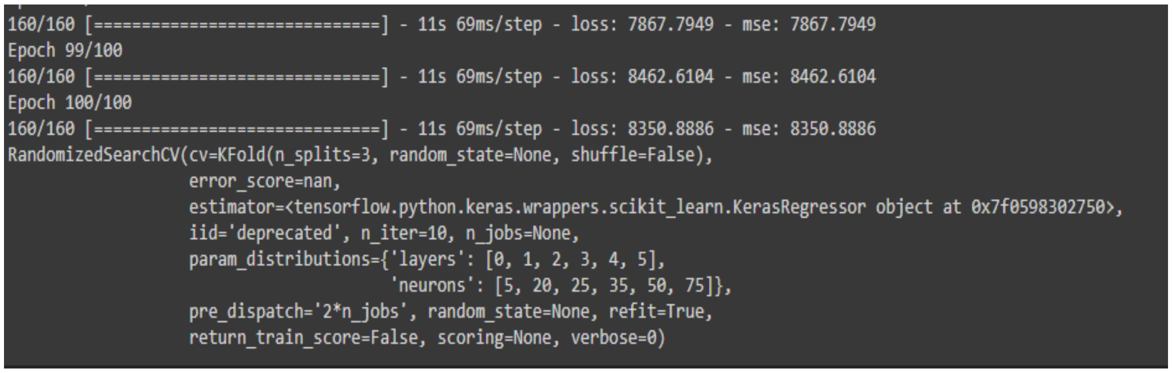
```
Model: "sequential_52"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_52 (SimpleRNN)    (None, 50)                2600
_________________________________________________________________
dense_129 (Dense)            (None, 15)                765
_________________________________________________________________
dense_130 (Dense)            (None, 15)                240
_________________________________________________________________
dense_131 (Dense)            (None, 1)                 16
=================================================================
Total params: 3,621
Trainable params: 3,621
Non-trainable params: 0
_________________________________________________________________
```

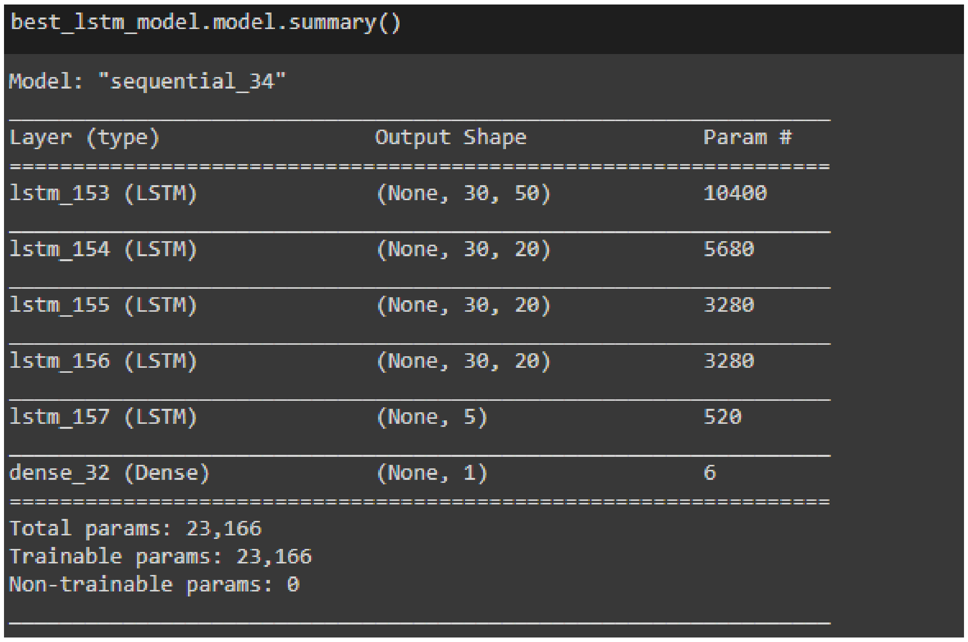


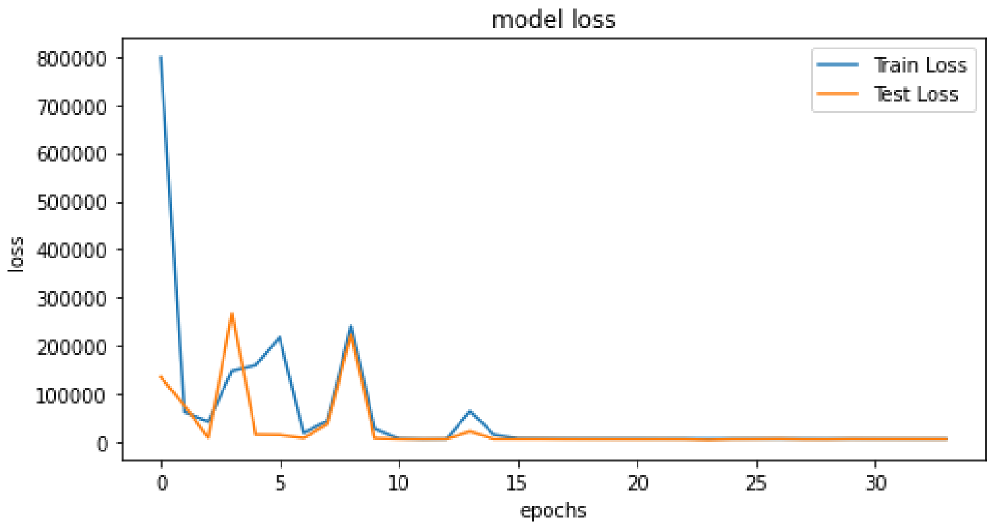
LSTM Model



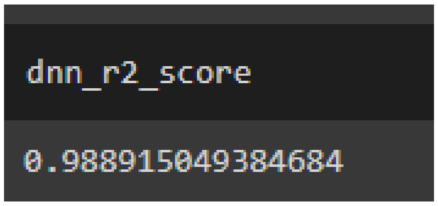
```
def create_model(layers=1 ,neurons=50):
    model1=Sequential()

    model1.add(LSTM(50,activation='relu',input_shape=(lookback,1),return_sequences=True))

    for i in range(layers):
        model1.add(LSTM(neurons,activation='relu',return_sequences=True))
    model1.add(LSTM(5,activation='relu'))
    model1.add(Dense(1,activation='relu'))
    model1.compile(optimizer='adam',loss='mean_squared_error',metrics=['mse'])
    return model1
```


```
160/160 [==============================] - 11s 69ms/step - loss: 7867.7949 - mse: 7867.7949
Epoch 99/100
160/160 [==============================] - 11s 69ms/step - loss: 8462.6104 - mse: 8462.6104
Epoch 100/100
160/160 [==============================] - 11s 69ms/step - loss: 8350.8886 - mse: 8350.8886
RandomizedSearchCV(cv=KFold(n_splits=3, random_state=None, shuffle=False),
                   error_score=nan,
                   estimator=<tensorflow.python.keras.wrappers.scikit_learn.KerasRegressor object at 0x7f0598302750>,
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'layers': [0, 1, 2, 3, 4, 5],
                                        'neurons': [5, 20, 25, 35, 50, 75]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=0)
```

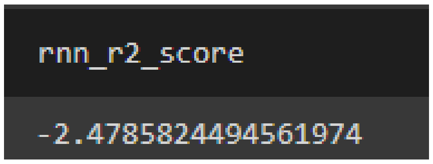

```
best_lstm_model.model.summary()

Model: "sequential_34"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_153 (LSTM)              (None, 30, 50)            10400
_________________________________________________________________
lstm_154 (LSTM)              (None, 30, 20)            5680
_________________________________________________________________
lstm_155 (LSTM)              (None, 30, 20)            3280
_________________________________________________________________
lstm_156 (LSTM)              (None, 30, 20)            3280
_________________________________________________________________
lstm_157 (LSTM)              (None, 5)                 520
_________________________________________________________________
dense_32 (Dense)             (None, 1)                 6
=================================================================
Total params: 23,166
Trainable params: 23,166
Non-trainable params: 0
_________________________________________________________________
```



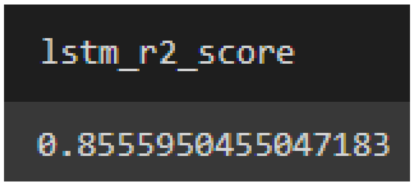
On Comparing the various models, the various metrics of R-Square score, loss values and graphs.



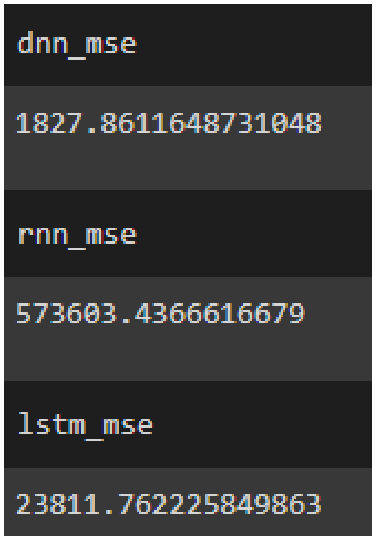
```
dnn_r2_score
0.988915049384684
```



```
rnn_r2_score
-2.4785824494561974
```



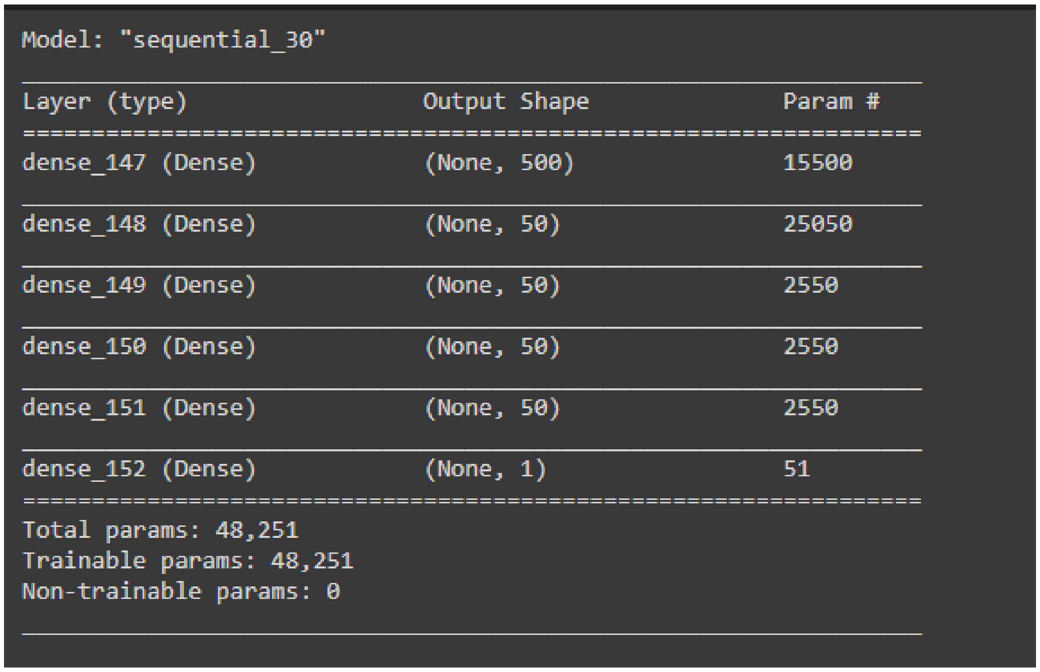
```
lstm_r2_score
0.8555950455047183
```



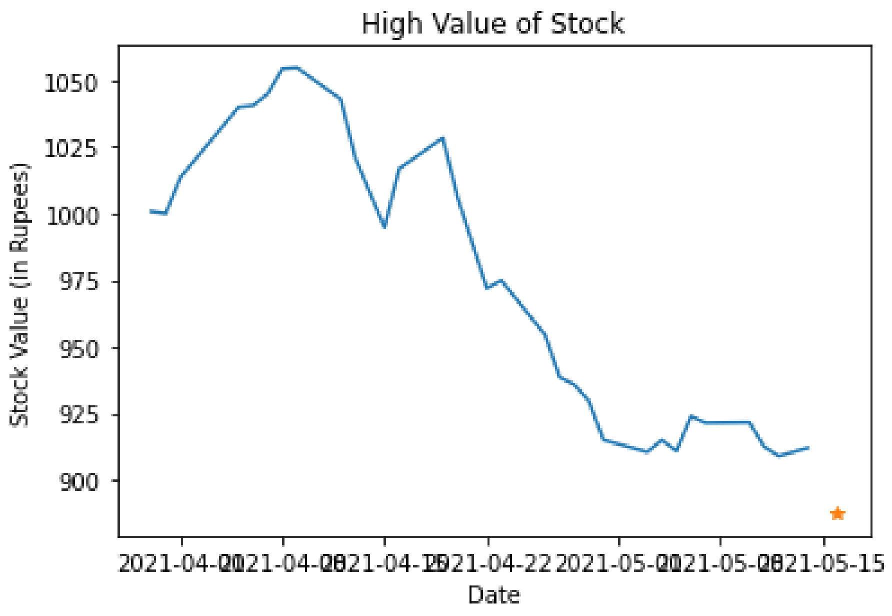
```
dnn_mse
1827.8611648731048
rnn_mse
573603.4366616679
lstm_mse
23811.762225849863
```

This clearly shows the DNN model performs best for the given dataset. Thus 4 models were prepared, hyperparameter tuning the models for the different stock values needed.

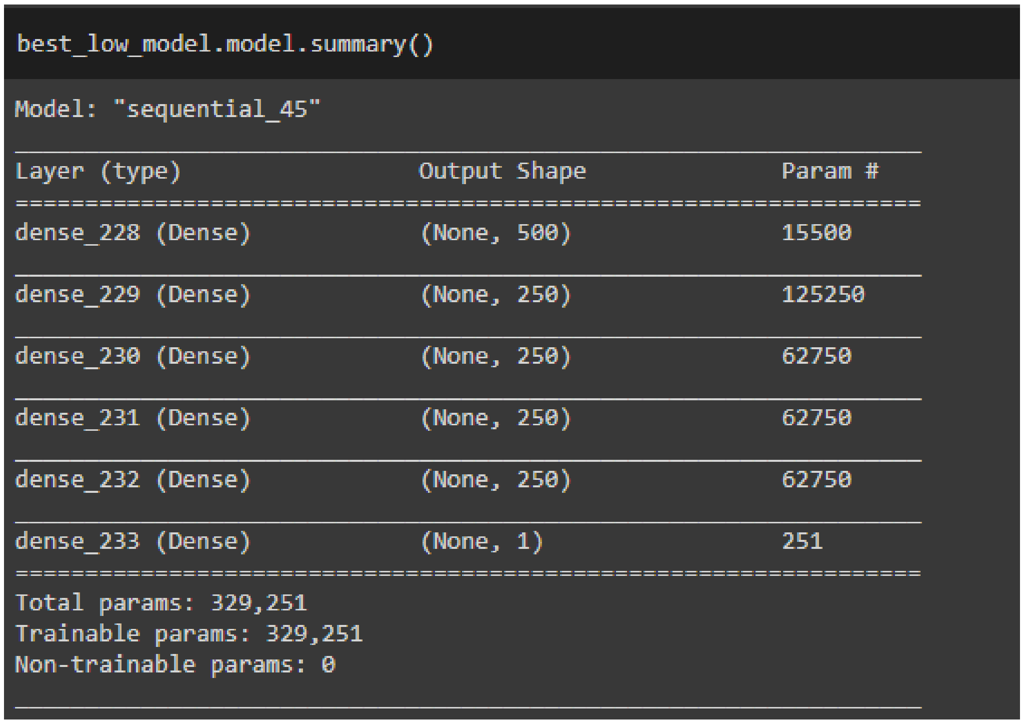
High Stock Price Model



```
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_147 (Dense)            (None, 500)               15500
_________________________________________________________________
dense_148 (Dense)            (None, 50)                25050
_________________________________________________________________
dense_149 (Dense)            (None, 50)                2550
_________________________________________________________________
dense_150 (Dense)            (None, 50)                2550
_________________________________________________________________
dense_151 (Dense)            (None, 50)                2550
_________________________________________________________________
dense_152 (Dense)            (None, 1)                 51
=================================================================
Total params: 48,251
Trainable params: 48,251
Non-trainable params: 0
_________________________________________________________________
```

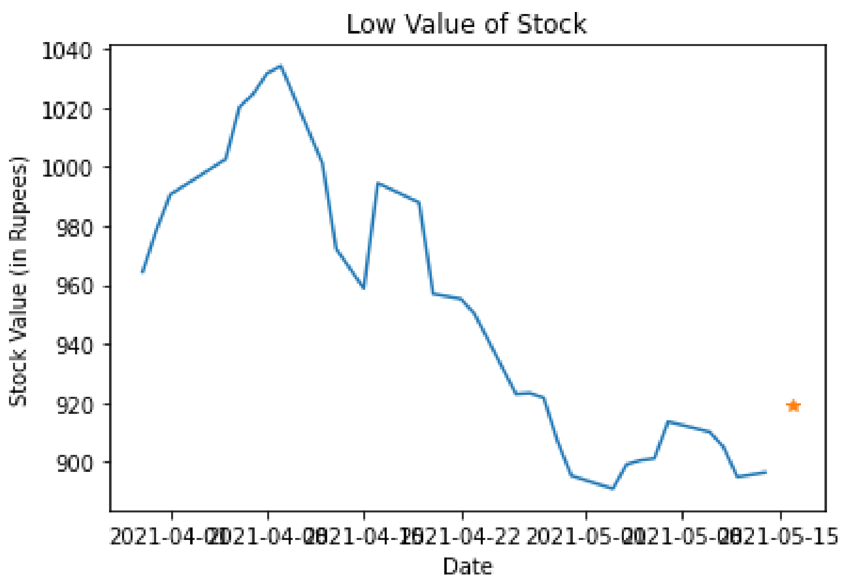


Low Stock Price Model

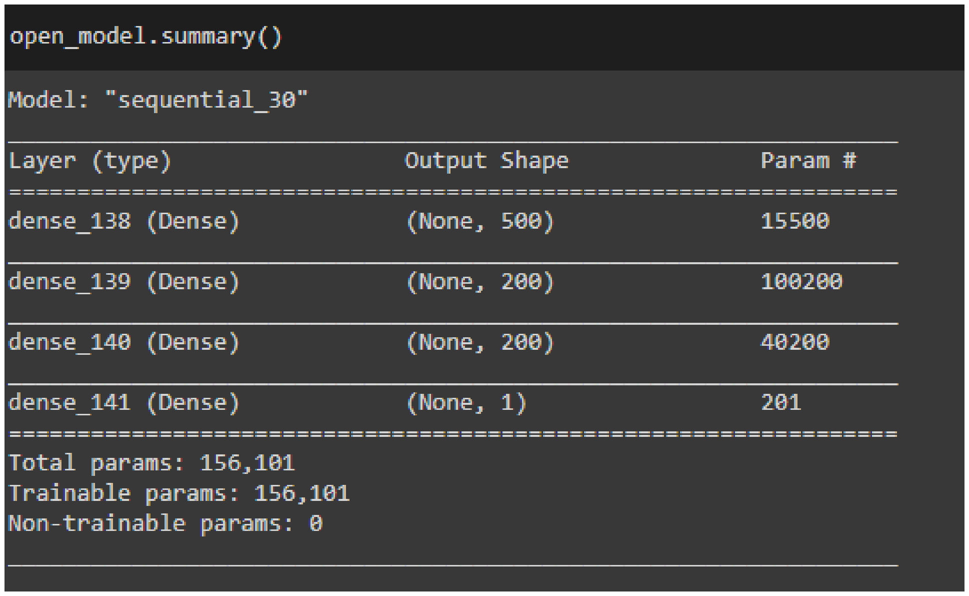


```
best_low_model.model.summary()
```

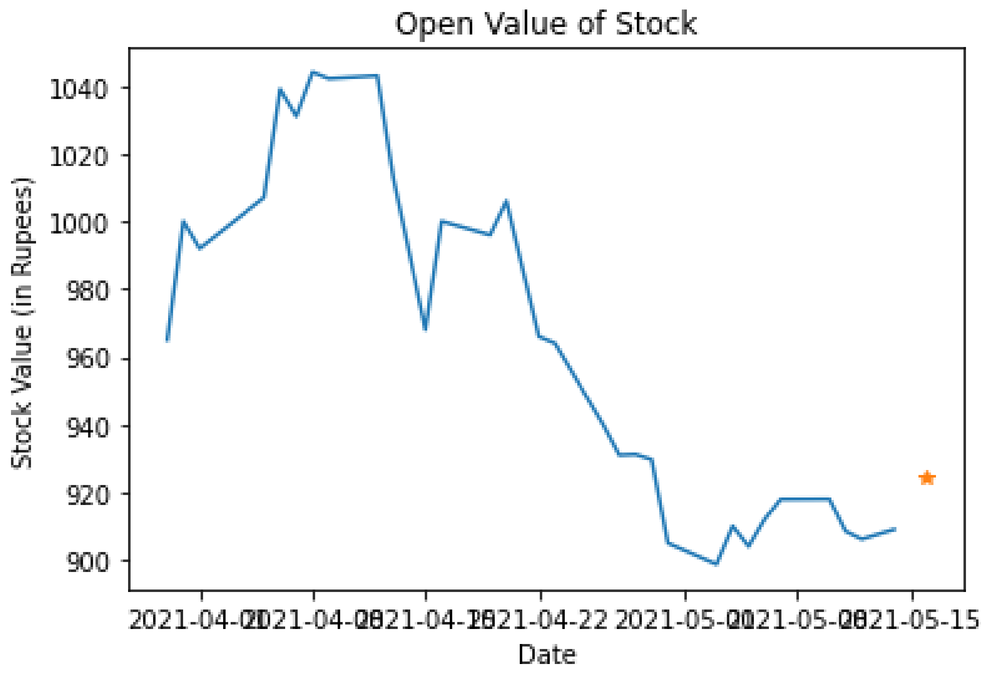
```
Model: "sequential_45"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_228 (Dense)            (None, 500)               15500
_________________________________________________________________
dense_229 (Dense)            (None, 250)               125250
_________________________________________________________________
dense_230 (Dense)            (None, 250)               62750
_________________________________________________________________
dense_231 (Dense)            (None, 250)               62750
_________________________________________________________________
dense_232 (Dense)            (None, 250)               62750
_________________________________________________________________
dense_233 (Dense)            (None, 1)                 251
=================================================================
Total params: 329,251
Trainable params: 329,251
Non-trainable params: 0
_________________________________________________________________
```



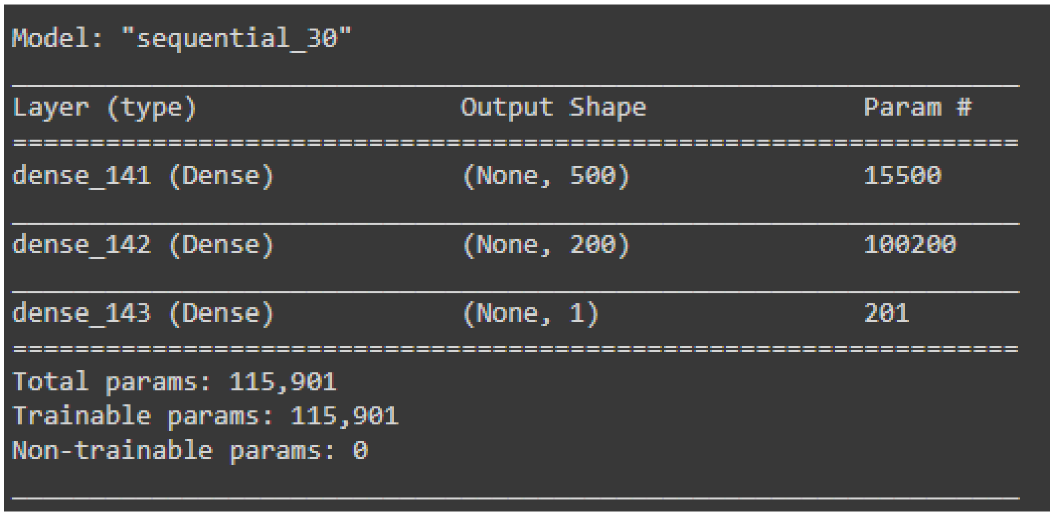
Open Stock Price Model



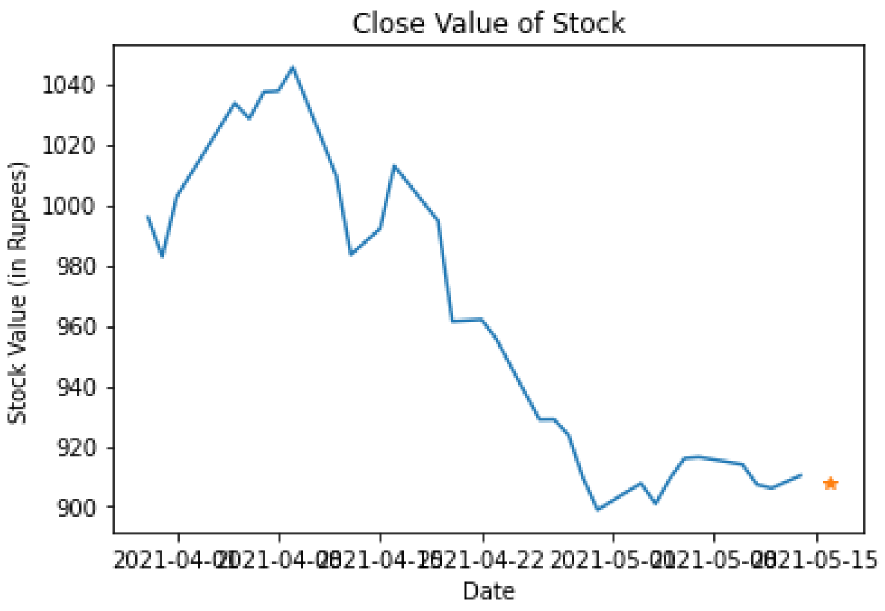
```
open_model.summary()

Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_138 (Dense)            (None, 500)               15500
_________________________________________________________________
dense_139 (Dense)            (None, 200)               100200
_________________________________________________________________
dense_140 (Dense)            (None, 200)               40200
_________________________________________________________________
dense_141 (Dense)            (None, 1)                 201
=================================================================
Total params: 156,101
Trainable params: 156,101
Non-trainable params: 0
_________________________________________________________________
```



Close Stock Price Model



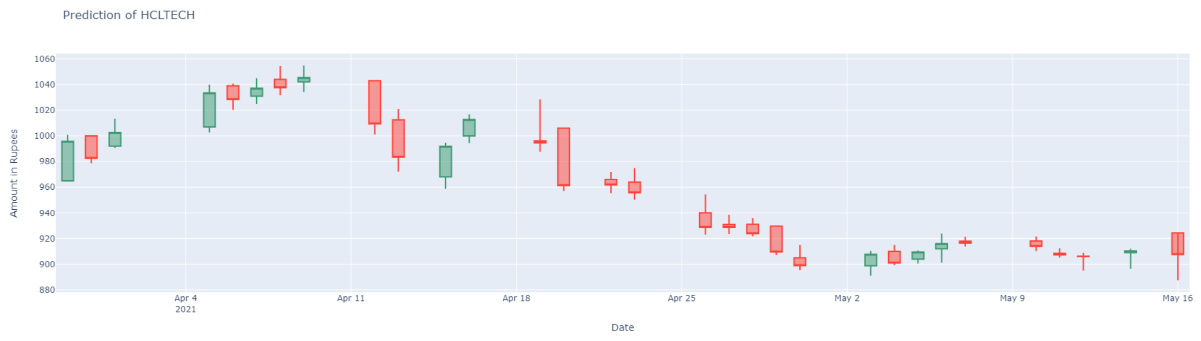
```
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_141 (Dense)            (None, 500)               15500
_________________________________________________________________
dense_142 (Dense)            (None, 200)               100200
_________________________________________________________________
dense_143 (Dense)            (None, 1)                 201
=================================================================
Total params: 115,901
Trainable params: 115,901
Non-trainable params: 0
_________________________________________________________________
```


Close Value of Stock
Stock Value (in Rupees)
1040
1020
1000
980
960
940
920
900
2021-04-01
2021-04-08
2021-04-15
2021-04-22
2021-05-01
2021-05-08
2021-05-15
Date

Chapter 6

Results and Output

Candlestick Graph of the predicted values



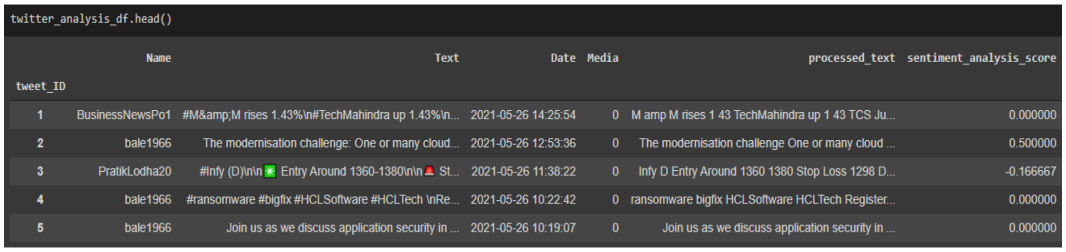
The above graph is known as candlestick graph, it is one of the most commonly used graphs in stock market industry, by stock traders to gain insight into the stock trend. The main reason for the amount of data that can be represented in the simple graph.

Each bar represents a particular stock trading day. A green color candle indicates that the day was profitable to the company, as the closing price of the stock was greater than opening price of the stock, the bigger the candle the bigger the profit for the day.

A red color candle indicates that the day was a loss for the company, as the closing price of the stock was lesser than opening price of the stock, the bigger the candle the bigger loss on that day.

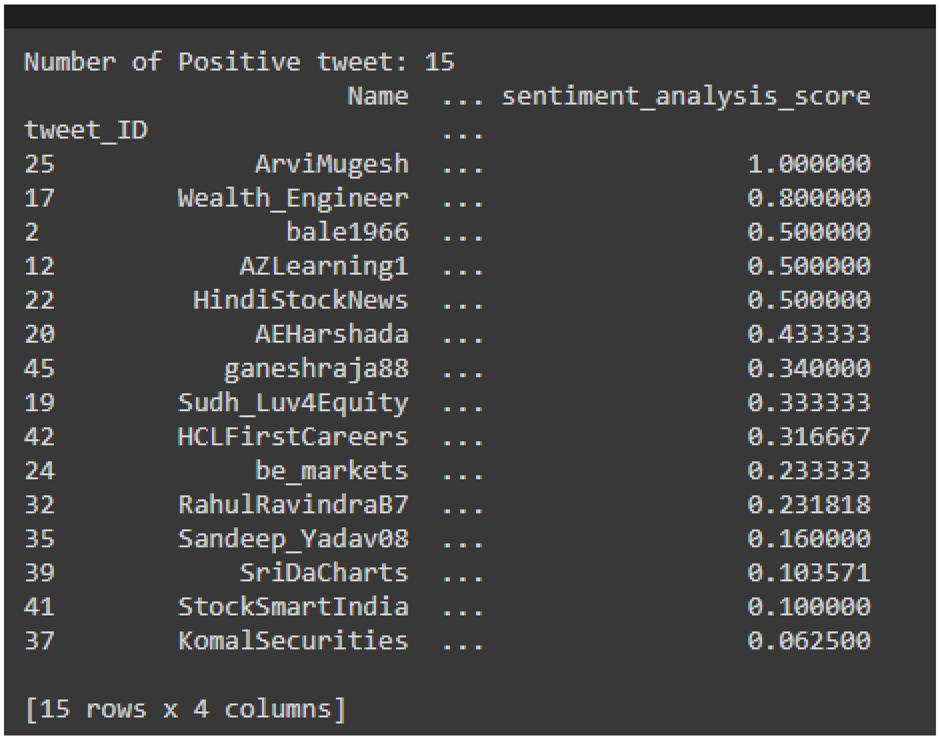
The stick on either side of the candle, depicts the highest and lowest price of the stock on that day. The longer the stick, it depicts the an abnormally high or low price.

Twitter Output

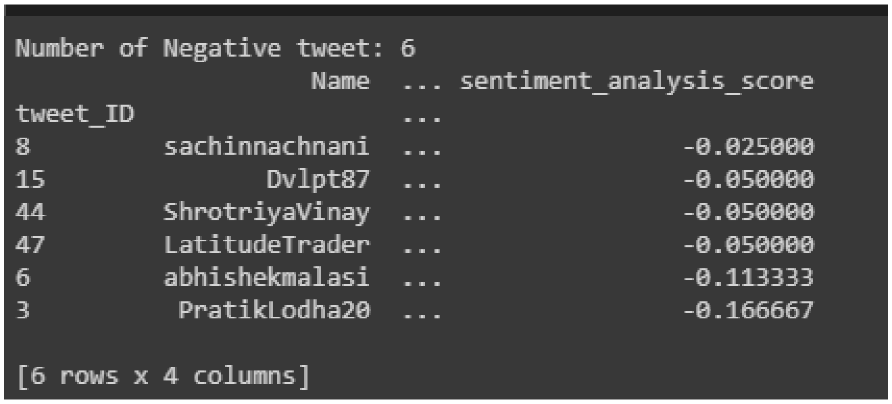


```
twitter_analysis_df.head()
```

tweet_ID	Name	Text	Date	Media	processed_text	sentiment_analysis_score
1	BusinessNewsPo1	#M&amp;M rises 1.43%\n#TechMahindra up 1.43%\n...	2021-05-26 14:25:54	0	M amp M rises 1 43 TechMahindra up 1 43 TCS Ju...	0.000000
2	bale1966	The modernisation challenge: One or many cloud...	2021-05-26 12:53:36	0	The modernisation challenge One or many cloud ...	0.500000
3	PratikLodha20	#Infy (D)\n\n✳ Entry Around 1360-1380\n\n🚨 St...	2021-05-26 11:38:22	0	Infy D Entry Around 1360 1380 Stop Loss 1298 D...	-0.166667
4	bale1966	#ransomware #bigfix #HCLSoftware #HCLTech \nRe...	2021-05-26 10:22:42	0	ransomware bigfix HCLSoftware HCLTech Register...	0.000000
5	bale1966	Join us as we discuss application security in ...	2021-05-26 10:19:07	0	Join us as we discuss application security in ...	0.000000



```
Number of Positive tweet: 15
                         Name  ... sentiment_analysis_score
tweet_ID                       ...
25                 ArviMugesh  ...                 1.000000
17            Wealth_Engineer  ...                 0.800000
2                    bale1966  ...                 0.500000
12                AZLearning1  ...                 0.500000
22              HindiStockNews ...                 0.500000
20                 AEHarshada  ...                 0.433333
45               ganeshraja88  ...                 0.340000
19             Sudh_Luv4Equity ...                 0.333333
42             HCLFirstCareers ...                 0.316667
24                 be_markets  ...                 0.233333
32             RahulRavindraB7 ...                 0.231818
35             Sandeep_Yadav08 ...                 0.160000
39                SriDaCharts  ...                 0.103571
41             StockSmartIndia ...                 0.100000
37             KomalSecurities ...                 0.062500

[15 rows x 4 columns]
```



```
Number of Negative tweet: 6
                        Name  ... sentiment_analysis_score
tweet_ID                      ...
8             sachinnachnani  ...                -0.025000
15                   Dvlpt87  ...                -0.050000
44            ShrotriyaVinay  ...                -0.050000
47            LatitudeTrader  ...                -0.050000
6             abhishekmalasi  ...                -0.113333
3              PratikLodha20  ...                -0.166667

[6 rows x 4 columns]
```

Chapter 7

Conclusion and Future Scope

At the end of the project, we can conclude that the DNN model with only dense layers performed better than the LSTM and RNN models tested, for the given dataset. This can be attributed to the vanishing gradient problem usually experience in RNN networks.

As seen in the train and test loss graph of the RNN and LSTM model, did not train well after the first few epochs, thus making the rest of the epochs training useless. The R-Square of the RNN model proved it was one that most suffered from the vanishing gradient problem with a final R-Square score on the test data, being –2.47. As expected, LSTM model did overcome the vanishing gradient problem better than the RNN model, but still did suffer a litter bit. The LSTM model had a R-Square of 0.855 and mean square error value of 23811.76 on the test data set, while DNN model had a R-Square score of 0.988 and mean square error of 1827.86 across a test data set of more than 3000 rows in the dataset.

A report was also displayed, showing the top 50 tweets on that day about the required company, from the major financial news magazine twitter handles and stock traders. The tweets were also classified into three categories, positive, negative and neutral, and sorted on the sentiment analysis score and displayed in descending order on the score, to show the most influential tweet first in either category.

In the future, more various algorithms can be tried to figure out the best model and more architecture with different optimizations and activation functions. For sentiment analysis, different algorithms could be tried and a front end could be designed, and make it a complete product.

Bibliography

References

[1] I. Parmar et al., "Stock Market Prediction Using Machine Learning," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), 2018, pp. 574-576, doi: 10.1109/ICSCCC.2018.8703332.

[2] T. Gao, Y. Chai and Y. Liu, "Applying long short term momory neural networks for predicting stock closing price," 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 575-578, doi: 10.1109/ICSESS.2017.8342981.

[3] S. Singh, T. K. Madan, J. Kumar and A. K. Singh, "Stock Market Forecasting using Machine Learning: Today and Tomorrow," 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), 2019, pp. 738-745, doi: 10.1109/ICICICT46008.2019.8993160.

[4] towardsdatascience.com

[5] machinelearningmastery.com

[6] ieeexplore.ieee.org

[7] developer.twitter.com/en/docs

[8] keras.io/api

[9] https://webkid.io/blog/neural-networks-in-javascript/

[12]
https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network

[13] https://cs231n.github.io/neural-networks-1/

[14]
https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

[15] https://en.wikipedia.org/wiki/File:Long_Short-Term_Memory.svg

[16]
https://medium.com/analytics-vidhya/various-aspects-of-sentiment-analysis-681d2ab969d0