Chapter 1

ABSTRACT

In a monetarily unstable market, as the *Stock Market*, it is essential to have an exceptionally exact forecast of a future pattern. Due to the budgetary emergency and scoring benefits, it is required to have a safe prediction of the prices of the stocks. As the *Stock Prices* are not restricted to change linearly, it is very much difficult to make a prediction depending over some traditional linear function, equation. Now, *Neural Network* comes into action.

As We can't predict this type of nonlinear, statistical dependency by seeing one instance (*Input Vector*), *Recurrent Neural Network (RNN)* is being introduced in our work. As *RNN* have a *Memory* which can remember the behavior of input and output values not only for one instance but also it considers many previous inputs-outputs statistics to predict the output of the present input instance, moreover We are also introduced here *LSTM*, which is a specific kind of RNN having a bigger memory than RNN.

We applied the RNN and LSTM to the inputs and outputs, which are recorded after applying different *Time Steps* on the *Closing Price*, feature from *Datasets* downloaded from *Yahoo Finance* to create the *Model*. The model worked very good on different datasets, moreover increasing the *Epoch Value* will show good prediction results, When the Model can't predict the stock price properly it will generate a *pattern* how the stocks will behave, a new property implemented this model.

An additional work has been done on this model, predicting how ten days stocks will behave depending on the outputs we get after applying the model for the test data. Although the future prediction is not appropriate, at least it can tell us the stock price will be increasing or decreasing, which is also very helpful for stock marketing.

Chapter 2

INTRODUCTION

The specialty of anticipating stock costs has been a troublesome errand for huge numbers of the scientists and examiners. Truth be told, speculators are profoundly intrigued by the exploration territory of stock value forecast. For a decent and effective speculation, numerous financial specialists are excited about knowing the future circumstance of the securities exchange. Great and viable expectation frameworks for financial exchange help merchants, speculators, and examiner by giving steady data like the future course of the securities exchange. In this work, we present Recurrent-neural-network (RNN) and Long-short-term-memory (LSTM) way to deal with foresee financial exchange lists.

2.1 Terminologies and Definitions:

Stock is used to depict the possession authentications of any organization.

A share alludes to the stock testament of a specific organization. A share hold by a person makes him the investor in the company.

Share Market is a market where protections are purchased and sold, a stock trade.

Share Price is the cost of a saleable supplies of an organization, subsidiary or other. Generally, the share price is the amount somebody is willing to pay for the stock, or the least amount that it very well may be purchased for.

2.2 Problem Definition:

The problem focuses on predicting the closing prices of a stock whether it will go up or go down up to 10 days.

2.3 Tools and Platforms:

• NumPy is used for performing various logic with Python.

• matplotlib.pyplot is an accumulation of order style works that make matplotlib work like MATLAB.

• Pandas is a library for the programming language in Python and to check the information control.

• MinMaxScaler from sklearn.preprocessing is used to downsize the dataset.

• Keras is an open-source neural-organize library written in Python. It is used for running operation on TensorFlow, Theano or PlaidML, and is designed for empowering experiment quickly with profound neural networks.

2.4 Developer End Requirements:

Stock Market Price prediction is executed utilizing Python (Spyderv3.7 API). The proposed Deep Learning approach for highlight extraction, is executed utilizing Python's Keras Toolbox (TensorFlow in the back-end).

2.5 User End Requirements:

The Predictive Model needs Scikit-Learn on a machine with 8GB RAM. The classifier preparing stage multifaceted nature is needy upon the size of the database. Henceforth, to encourage preparing for database of impressive size, the framework must have the above details.

Chapter 3

LITERATURE REVIEW

Machine learning plays an important role in predicting stock market prices. The different techniques in Machine learning such as RNN, LSTM, etc helps to find the patterns of the features on which prediction can be done. In [3], the author used LSTM to convert the GE stock prices in time-series problem and used supervised learning tactics to train and test the dataset.

In [4], the author tried to predict the stock prices for a certain company using different techniques and models such as Moving Average, Linear Regression, k-nearest Neighbors, Auto Arima model, but failed to give satisfactory prediction due to poor designing. The author finally tried LSTM on which the author got satisfactory results. Still, the author declares that stock prices are affected due to different intangible factors such as news of the company, demonetization or merging/demerging of the company.

In [5], the author(s) tried to predict the stock prices using regression techniques over numerical and textual analysis of a specific stock unit. They consider the source through newspaper from where the textual content as the stock name and its prices are figured out.

In [6], the author(s) tried to used LSTM network to find the stock price movement and proved that lstm network proves very promising while analysing various stocks and if very beneficial when inputs are given in a large amount.

In [12], the author(s) tried to propose a hybrid machine learning system using Genetic Algorithm and Support Vector Machines for stock market predictions. But the correlation factor between the variables didn't suit the perfect results.

In [13], the author(s) tried to predict the prices of stocks using Artificial Neural Networks(ANN), Random Forest(RF), Support Vector Regressions(SVR). Ten years of historical data are given as inputs to get desired outputs.

In [15], the author(s) proposed a model using Support Vector Machines and Reinforcement learning. The results produced using SVM techniques guarantees high accuracy for specific stocks such as NASADAQ, S&P500, DJIA.

In [16], the author(s) tried to predict the stock prices considering formidable patterns where the results could be satisfactory. Various Machine learning classifiers such as Artificial Neural Networks, Decision Tree, k nearest neighbour techniques were used to predict the outputs where the results are satisfactory up to 65% accuracy.

Chapter 4

CONCEPT AND PROBLEM ANALYSIS

4.1 Machine Learning (ML) :

Machine Learning is the modern approach to logical investigation of calculations and measurable models that gives computer frameworks the capacity to consequently take input and improve as a matter of fact without being expressly customized.

4.1.1 Types of Machine Learning (ML) :

Machine Learning Algorithm are often categorized four types. They are defined :

- **Supervised Learning** :

  Supervised Learning is used for defining a function from labelled training data. The training data contains a set of training instance. In this technique, each example is simply a pair an input object (containing different combination of features) and a desired output value (also called the supervisory signal). There different types of lupervised Learning Algorithm exists, They are as follows :

    1. K Nearest Neighbour (KNN) Algorithm
    2. Naive Bayes Algorithm
    3. Decision Tree Algorithm
    4. Random Forest Algorithm
    5. Linear Regression Algorithm
    6. Support Vector Machine(SVM) Algorithm
    7. Neural Networks Algorithm

- **Unsupervised Learning :**

  Unsupervised learning can be characterized as a part of an *artificial intelligence (AI)* method which fundamentally use information that is neither classified nor marked and enable the calculation to make reasonable move on that given unlabelled data with no direction. Unsupervised Learning Algorithms are as follows :

    1. K-means clustering, Association Rules

- **Semi-supervised Learning :**

  Semi-Supervised Learning can be characterized as a mix of supervised and unsupervised techniques.

- **Reinforcement Learning :**

Reinforcement is going to act suitably to expand result accuracy in a specific circumstance. It is connected by different software and machines to decide the most ideal way or way the algorithm should take in a specific case.

1. Q-Learning
2. Temporal Difference
3. Deep Adversarial Network

4.2 Deep Learning (DL):

Deep Learning is a machine learning technique that comprehends to errand the sys to do the task which can be done automatically. Deep learning is connected as the key innovation on programmed vehicles. They can perceive a stop sign or a person on foot from a lamppost.

4.2.1 Types of Deep Learning (DL)

There are different deep learning techniques exists. They are as follows :

- Feedforward Neural Network (FNN)
- Radial Basis Function Neural Network (RBFNN)
- Kohonen Self Organizing Neural Network (KSONN)
- Recurrent Neural Network (RNN)
- Convolutional Neural Network (CNN)
- Modular Neural Network (MNN)

4.3 Recurrent Neural Network (RNN):

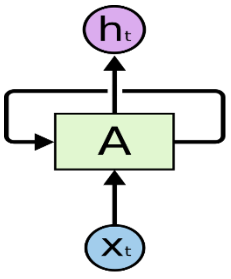


Figure 4.3.1 : RNN have

Recurrent-Neural-Network (RNN) is a sort of neural-network where the past advance's yields are sustained as the contribution to the present advance. In traditional ***neural networks***, none of the data sources and yields are needy of one another, yet now and again past experience are required. At the point when the machines need to predict the following expression of a sentence, it is especially required to recall the past words, same as when we have to predict the stock price we have to remember the pattern of previous days stocks, how they are behaving, the pattern or statistical analysis of the previous stock price will help us to have an idea about future stock price. In this way RNN came into existence. This type of issues are handled by a *Hidden Layer*, which is its most important feature which is used to remember a sequence having some information.

In Figure 4.3.2, a bunch of neural-network, A takes *input* x_t and give *outputs* a value h_t. A loop enables data to be passed starting with one stage then onto the next in a network. These loops characterize recurrent neural networks to us in an alternate way. Any way they aren't completely not quite the same as a typical neural network. A recurrent neural network is characterized as where every networks are passing a message to its *successor*. If we open the loop it will look like this :

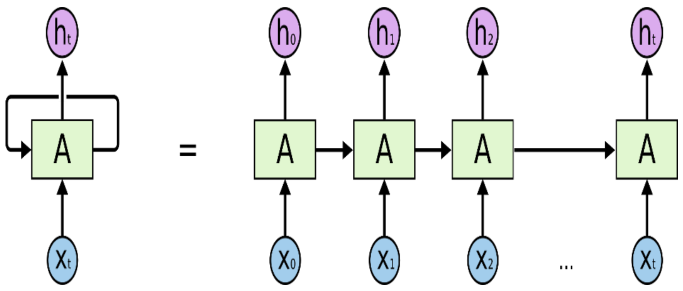


Figure 4.3.2 : Unrolled

We find recurrent neural networks as very much dependent to lists and sequences after examining the open structure.

There is incredible achievement applying RNN to various issues in couple of years: language modelling, recognition of speech and image, captioning of image, translation, etc. In our case **Stock Price prediction (Time Step Analysis)**.

The reality behind these triumphs is the utilization of **LSTM**s, an exceptional sort of recurrent neural network. Practically all exciting outcomes dependent on recurrent neural networks are accomplished with them. In our case, time step analysis is being analysed by LSTM. We are going to explore LSTM later in detail.

4.4 LSTM Networks :

Long short term memory (LSTM) can learn and work with long-term dependencies**.** They are categorized as a special kind of RNN, designed for handling the long-term dependency problem as they have memory to remember long period of information, it is their default nature.

The form of a chain of repeating modules of neural network have to be maintained by all Recurrent neural network not like the standard RNN where the repeating module will have a very simple structure like a tanh layer.

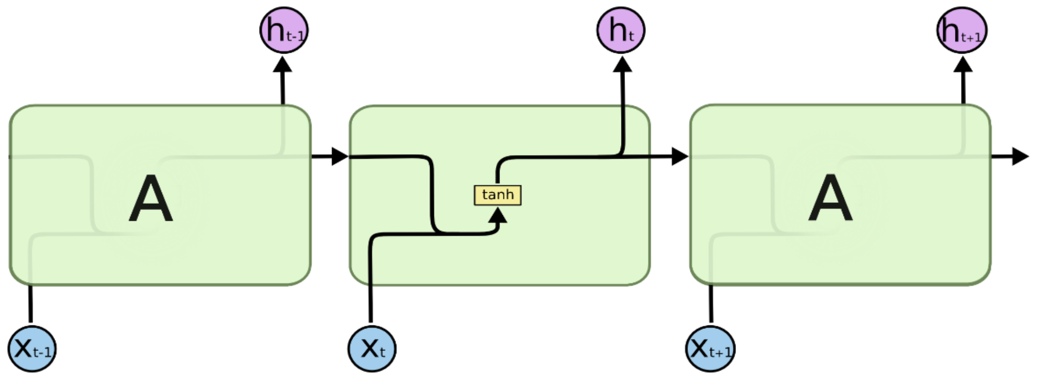


Figure 4.1.1 : Single layer in Repeating module of standard RNN

LSTM is also like the same but the repeating module of LSTM has a bit difference in its structure. There are four interacting layer present in the case of LSTM instead of having a single one.

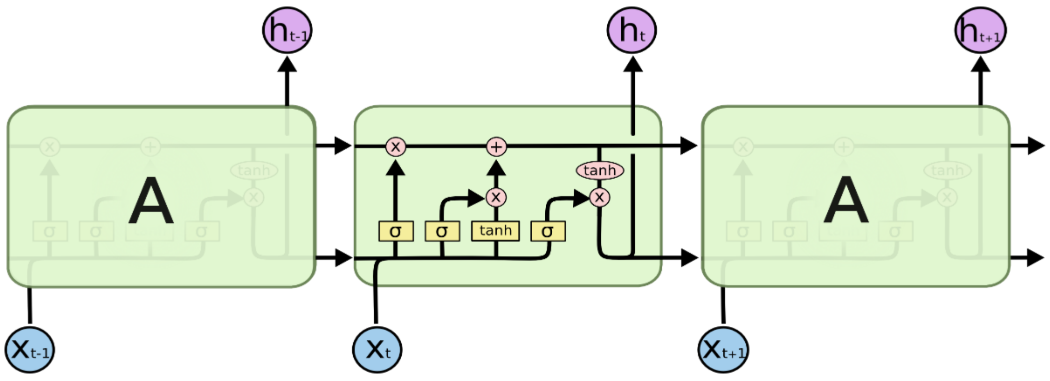


Figure 4.5.2 : Four interacting layer in Repeating module of LSTM

In Figure 4.5.3, each line which are drawn from the output of one node to the inputs of others is an entire vector. The *circles*, *rectangular boxes* in the diagram represents *pointwise Operation*, *vector addition* and the *Learned Neural Network* respectively. Where lines are *appending*, denotes concatenation, while a line is forking denotes that after being copied the contents of the vector is being stored in some other location.

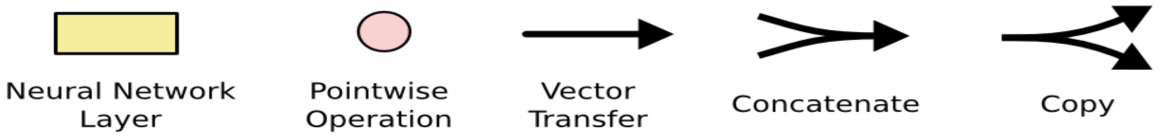


Figure 4.5.3 : Notation which will be used on LSTMs

4.5.1 The Core Idea Behind LSTMs

Cell state is the principle part of LSTM, which resembles a *conveyor belt*, straight down the whole chain of the LSTM. The horizontal line, having not many straight association of the Figure 4.5.1.2 is the cell state here through which data can pass being unaltered. Using Gates new data can be included excess data can likewise be deleted from cell state in LSTM systems. Gates are normally a mix of *Sigmoid neural network layer* and *pointwise multiplication operation* where the output of sigmoid layer vary from 0 (let no information through the Network) to 1 (let all information through the system). By

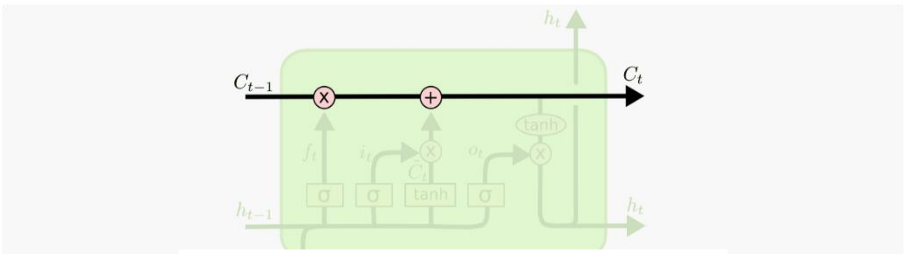


Figure 4.5.1.2 : Cell State representation in LSTM

containing three of this gates LSTM can control the progression of cell state.

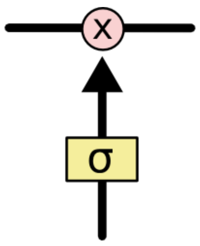
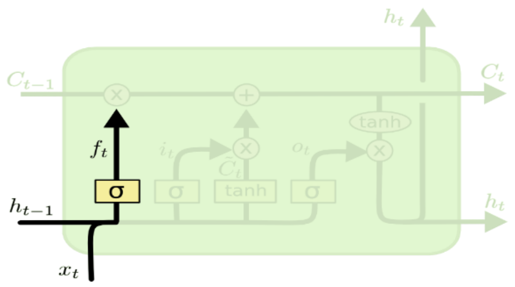


Figure 4.5.1.1 : SNNL & PMO

4.5.2 Working Principle of LSTM

FIRST STEP :

In LSTM the main choice must be made by a sigmoid layer is what are the excess data to discard from Cell State. The Sigmoid Layer which is named Forget Gate Layer in this progression yields a number between 0 (totally dispose of this) and 1(completely keep this) in the Cell State c_{t-1} taking a gander at h_{t-1} and x_t.

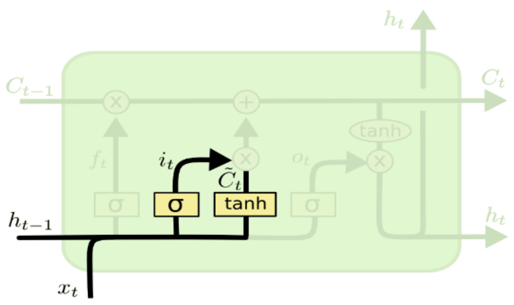


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Figure 4.5.2.1 : First Step of LSTM

SECOND STEP :

In the subsequent step which data will be put away in the cell state are being chosen by a sigmoid layer is additionally called as *Input gate layer* which chooses which esteems must be refreshed and a tanh Layer, makes a vector of new candidate value esteems $\check{C}_t$, going to be added to the cell state.

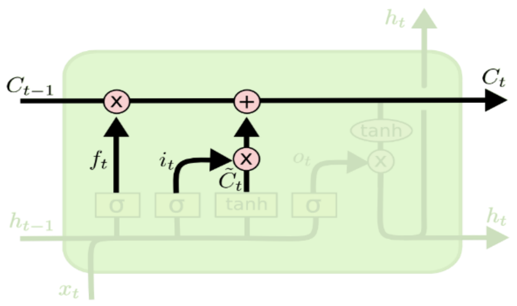


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

Figure 4.5.2.2 : Second Step of LSTM

THIRD STEP :

Now LSTM is going to refresh the old cell state C_{t-1} into C_t, multiplying old cell state by f_t (overlooking the things we chose to overlook before) and including i_t*$\check{C}_t$. Contingent upon t this we are going to refresh the following cell state esteem.

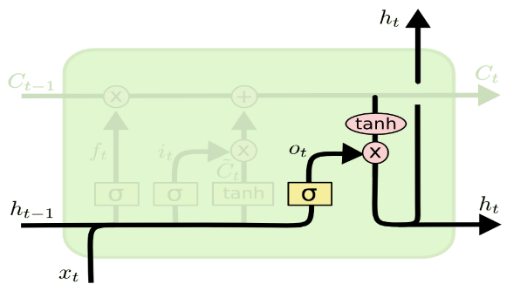


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 4.5.2.3 : Third Step of LSTM

FOURTH STEP :

Now LSTM will be giving us the output contingent upon the sifted variant of the cell state. LSTM will apply a sigmoid function first, at that point multiply with cell state after it is experiencing the tanh layer to give just the data as output that we have chosen to produce.



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

Figure 4.5.2.4 : Fourth or Last Step of LSTM

IMPORTANT SYMBOLS :

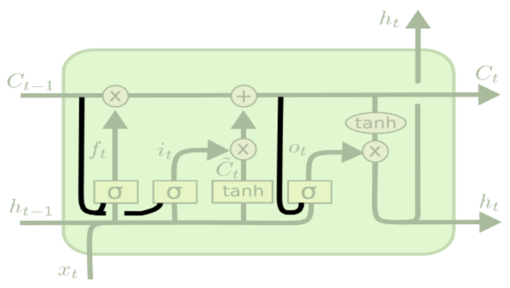
C_t = Cell State , $\check{C}_t$ = new candidate value which we are going to add with Cell State , f_t = Output of sigmoid layer , i_t = Output of input gate layer , h_{t-1} = Output of the previous step , x_t = Input of present step , h_t = Output of the present step , o_t = Output of another sigmoid layer.

4.5.3 Different Types of LSTM :

What we have discussed till now is actually normal LSTM. But there some more LSTMs exists with a bit of addition of new concept in the hidden layers.

TYPE 1:

Adding Peephole connections to all the Gate of the Cell State. This was discovered by *Gers & Schmidhuber (2000).*

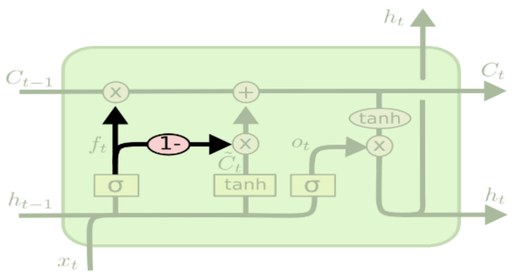


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \;+\; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \;+\; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \;+\; b_o\right)$$

Figure 4.5.3.1 : Addition of Peephole Connection to the LSTM

TYPE 2:

Addition of coupled forget and input gate, in this LSTM we will choose what to add as new data to the cell state and what are the excess or redundant data of the cell state.

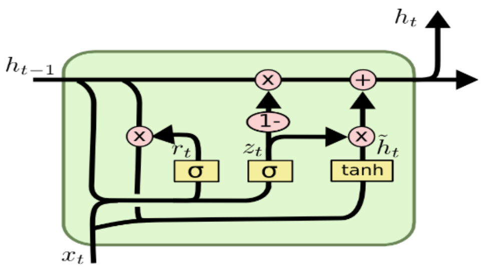


$$C_t = f_t * C_{t-1} + \boldsymbol{(1 - f_t)} * \tilde{C}_t$$

Figure 4.5.3.2 : Coupled Forget and Output Gate

TYPE 3:

GRU (gated_recurrent_unit) introduced by Cho, et al. (2014), a different type of RNN. In spite of the fact that forget and input gates are joined in it, however another show of blending the cell state and hidden state was additionally included, making this LSTM arrange much straightforward.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 4.5.3.3 : GRU applied LSTM

Chapter 5

DESIGN AND METHODOLOGY

5.1 System Phases:

Our proposed system consists of RNN & LSTM model.

Feature extraction is fundamental for framing a feature or pattern. This vector contains data. Accurate features results to better classification and grouping.

Each of the phases are described in the following sections-

5.2 Data Pre-Processing:

The dataset used in the model were collected from Yahoo Finance. The link for the dataset is provided with the dataset. The model was executed on three dataset-

(1) NSEI – National Stock Exchange of India aka Nifty 50.
https://in.finance.yahoo.com/quote/%5ENSEI/history?p=%5ENSEI
(2) BSE – Bombay Stock Exchange aka BSE SENSEX
https://in.finance.yahoo.com/quote/BTC-USD/history?p=BTC-USD
(3) BTC-USD – Bitcoin in US Dollars value.
https://in.finance.yahoo.com/quote/BTC-USD/history?p=BTC-USD

5.2.1 Dataset Reading-

The datasets are in *csv (comma separated values)* file. The data are read first and then scaled down in (0, 1) scale.

5.2.2 Dataset Filtration-

The dataset collected, had *null* values however. In command language, it has *Nan (not a number)* values. Those nan values were filtered out to minimise the complexities and error probabilities.

5.2.3 Dataset Dividing-

In Machine Learning, the dataset are divided in two types-

1. Training Data.
2. Testing Data.

Training Data consists of 70-80% of the whole dataset on which the model is being trained. To test the model the rest 30-20% data is taken to check the prediction is correct or not.

In our model, the dataset is being divided in continuous format.

1. For *Nifty 50* - The training data is of 5 years(1st May, 2014 - 31st May, 2019) while training data is of the very next month(1st June, 2019 – 30th June, 2019).
2. For *BSE* - The training data is of 5 years(1st April, 2014 – 30th April, 2019) while training data is of the continuous 2 next months(1st May, 2019 – 30th June, 2019).
3. For *BTC-USD* - The training data is of 5 years(1st March, 2014 – 31st March, 2019) while training data is of the continuous 2 next months(1st April, 2019 – 30th June, 2019).

In Tabular Format-

Stock Names	Training Data (in Years)	Testing Data (in Months)
Nifty 50	5	1
BSE	5	2
BTC-USD	5	3

5.3 Feature Selection:

The model is designed to predict the prices of stocks in advance to get an idea if the stock prices would rise or fall. So, the feature selection for the model must be prices. The csv file of the dataset had following features-

a. Dates
b. Opening Prices.
c. Closing Prices.
d. Lowest Price of the day
e. Highest Price of the day
f. Average Price of the day
g. Volume of the Stock.

Stock market opens at 10.00 AM and closes at 3.00 PM.

Opening prices are the price at which the Stock market opens. After that, prices fluctuate as the importing and exporting of the stocks gets started. To get benefit out of Stock market, one must try to find out the Closing Price as to know if the particular stock is to be bought or to be sold.So, Closing Price would be the optimum feature to be selected to design the model.

5.4 Predictive Model Development:

5.4.1 Training & Testing Data-To design the model, we first needed the data to work on. In the proposed model we require training data to train the model and testing data to test the model.

The data is divided in following format into Training data and Testing data.

1. For Nifty 50 - The training data is of 5 years(1st May, 2014 - 31st May, 2019) while training data is of the very next month(1st June, 2019 – 30th June, 2019).
2. For BSE - The training data is of 5 years(1st April, 2014 – 30th April, 2019) while training data is of the continuous 2 next months(1st May, 2019 – 30th June, 2019).
3. For BTC-USD - The training data is of 5 years(1st March, 2014 – 31st March, 2019) while training data is of the continuous 2 next months(1st April, 2019 – 30th June, 2019).

In Tabular Format-

Stock Names	Training Data(in Years)	Testing Data(in Months)
Nifty 50	5	1
BSE	5	2
BTC-USD	5	3

5.4.2 Proposed Model-

The Model works on RNN and LSTM module in a repetitive layer. The training data is being trained over different timesteps**.** The timesteps taken into considerations are 40,50,60 and 70. It may vary according to the dataset.

Dataflow Diagram-

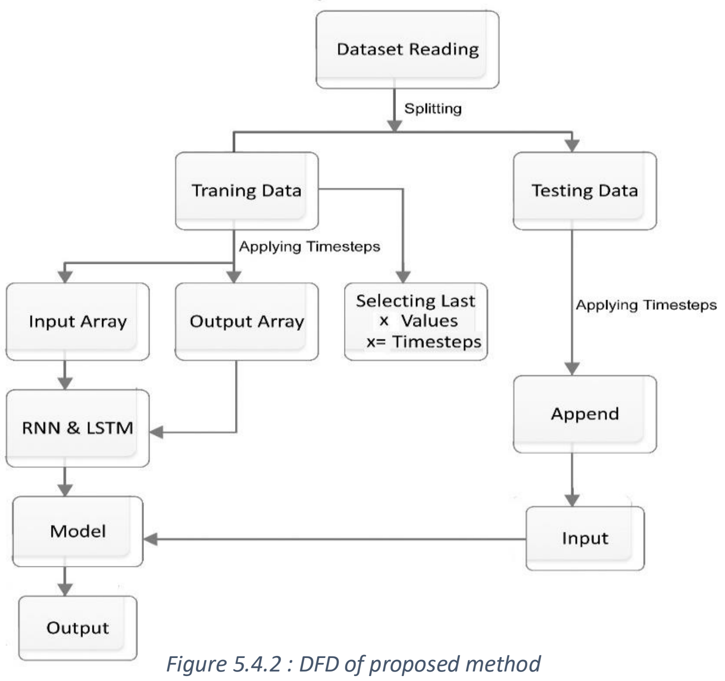


Figure 5.4.2 : DFD of proposed method

Explanation-

The model runs over RNN and LSTM module. First, we read the dataset and split it into training data and testing data. Now, over training data, we use timesteps to build input and output array. Let, ‘x’ be the timesteps. Then, from (0-x) elements would be in the input array and (x+1)th element would be in the output array. This process is continued till the last element of the training data is inserted into output array. Now, we run RNN and LSTM over our input array and output array to find relation between them. To find better relation we apply different epochs and adjust our timesteps and find the optimum model.

After training the model, we test the accuracy and the loss percentage of the model providing the testing data to the model and checking the prediction is how much correct.

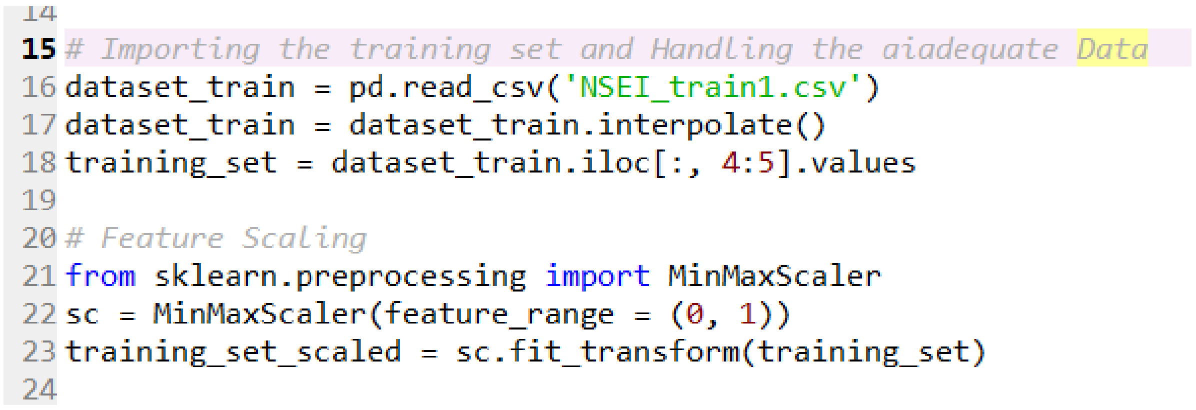
After getting the satisfying results, we re-run the model appending the predicted output with the dataset for 10 times to get the prediction of 10 days ahead.

Chapter 6

CODE SAMPLE

Here We are supplying some instances of our code and defining how they works step by step.

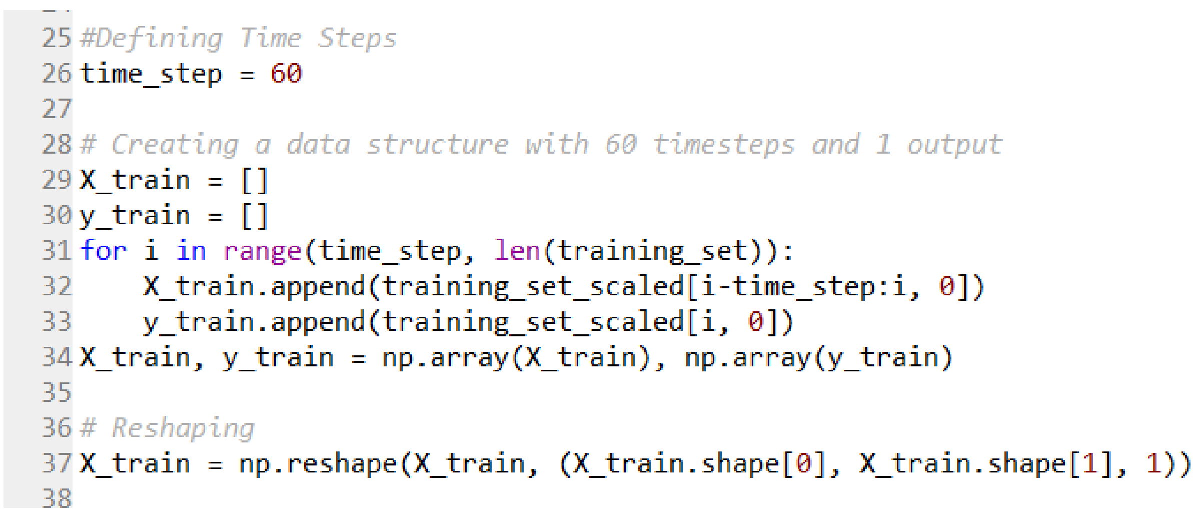
STEP 1 : Importing Training Data Set and Feature Scaling



```
# Importing the training set and Handling the aiadequate Data
dataset_train = pd.read_csv('NSEI_train1.csv')
dataset_train = dataset_train.interpolate()
training_set = dataset_train.iloc[:, 4:5].values

# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
```

Getting data using *pandas.read()* function We are handling the inadequate data (data having null, NaN, 0 values). We are selecting the 'close price' as our feature. It is not possible to work with a high range of data, So, We are Scaling the from 0 to 1 (fixing the range 0 to 1) data using *MinMaxScaler.fit_transform()* .

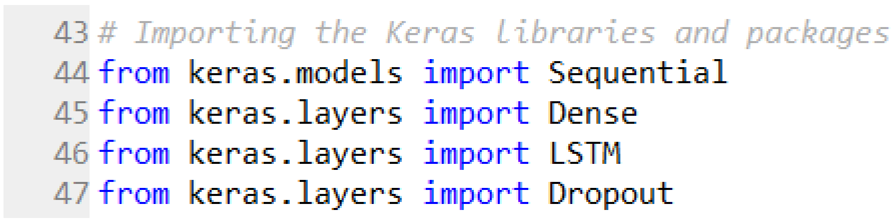
STEP 2 : Applying timesteps and reshaping the Data



```
#Defining Time Steps
time_step = 60

# Creating a data structure with 60 timesteps and 1 output
X_train = []
y_train = []
for i in range(time_step, len(training_set)):
    X_train.append(training_set_scaled[i-time_step:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)

# Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

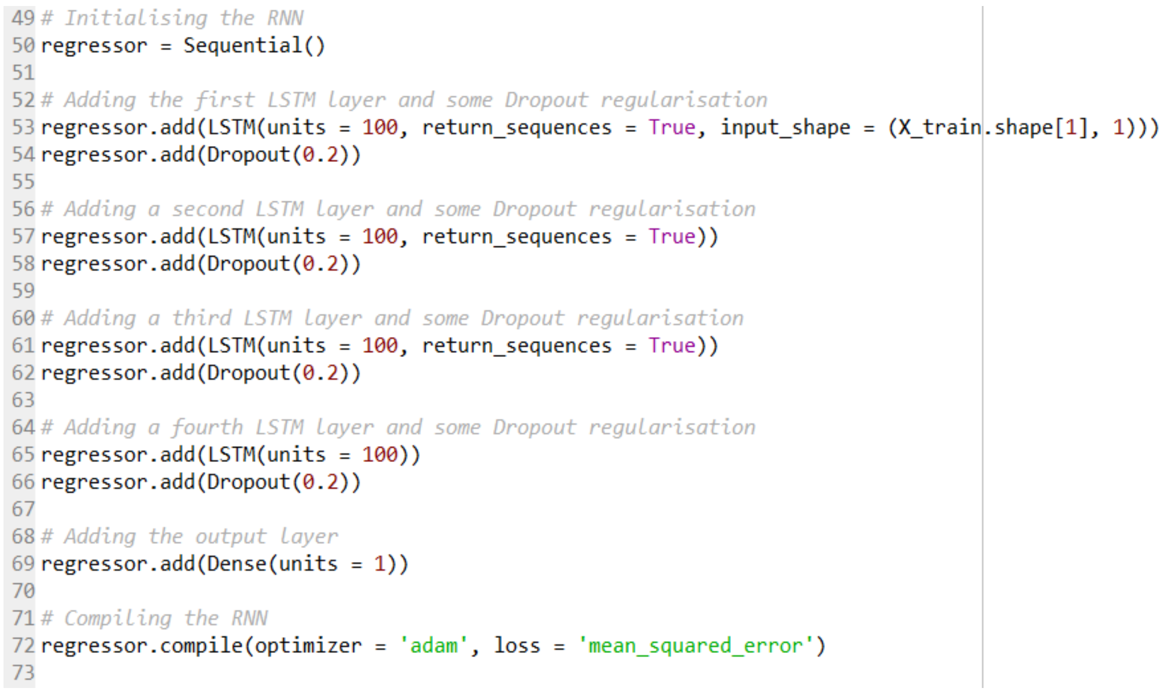
Here the timestep is set to 60. But **timestep** varies from **dataset to dataset**. We have check and execute the code until we get a suitable timestep which will give us the better results. Here we are going to check for the timestep values 40, 50, 60, 70 etc.

STEP 3 : Importing the important packages for creating the model



```python
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

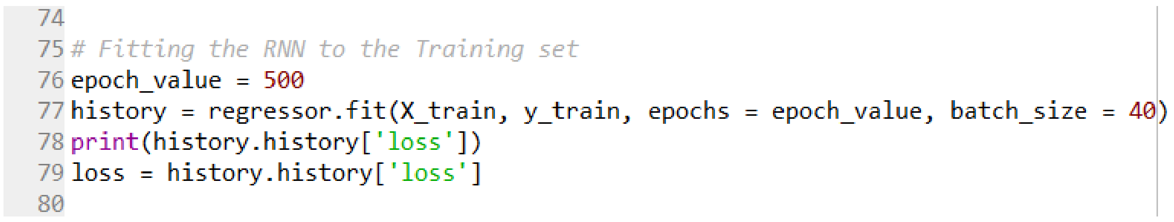
STEP 4 : Creating a model on RNN and LSTM



```python
# Initialising the RNN
regressor = Sequential()

# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 100, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 100, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 100, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 100))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

```

Here first We have initialized the RNN using *Sequential()* function. Then We have added different LSTM layers, output layer to the RNN. Then We have compiled our model using *adam* optimizer and *mean_squred_error loss*.

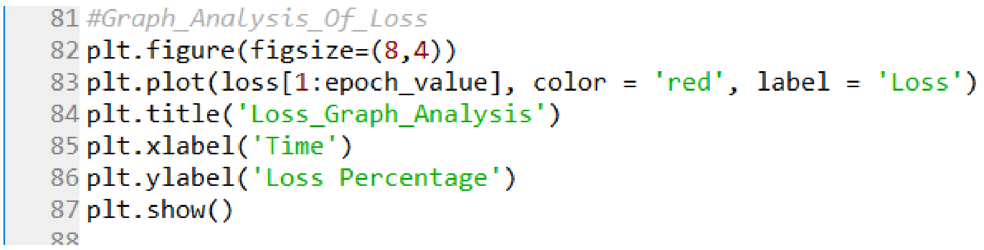
STEP 5 : Fit the training set into the model



```python

# Fitting the RNN to the Training set
epoch_value = 500
history = regressor.fit(X_train, y_train, epochs = epoch_value, batch_size = 40)
print(history.history['loss'])
loss = history.history['loss']

```

Here the epoch value is a very important factor. We have to set the epoch value depending on the analysis of the Data. Very *small epoch* as well as very *large epoch* value is not right sometimes. Sometimes it can cause *underfitting* and *overfitting*. We will see in our result. We have executed the code for epch value 100, 250, 500.

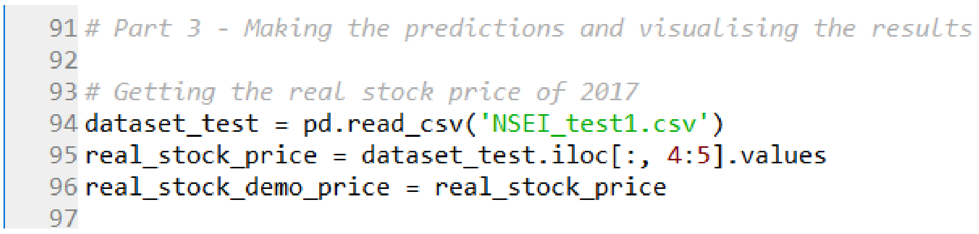
STEP 6 : Visualizing the loss graph



```
#Graph_Analysis_Of_Loss
plt.figure(figsize=(8,4))
plt.plot(loss[1:epoch_value], color = 'red', label = 'Loss')
plt.title('Loss_Graph_Analysis')
plt.xlabel('Time')
plt.ylabel('Loss Percentage')
plt.show()
```

Here we are visualizing the loss/error Graph.

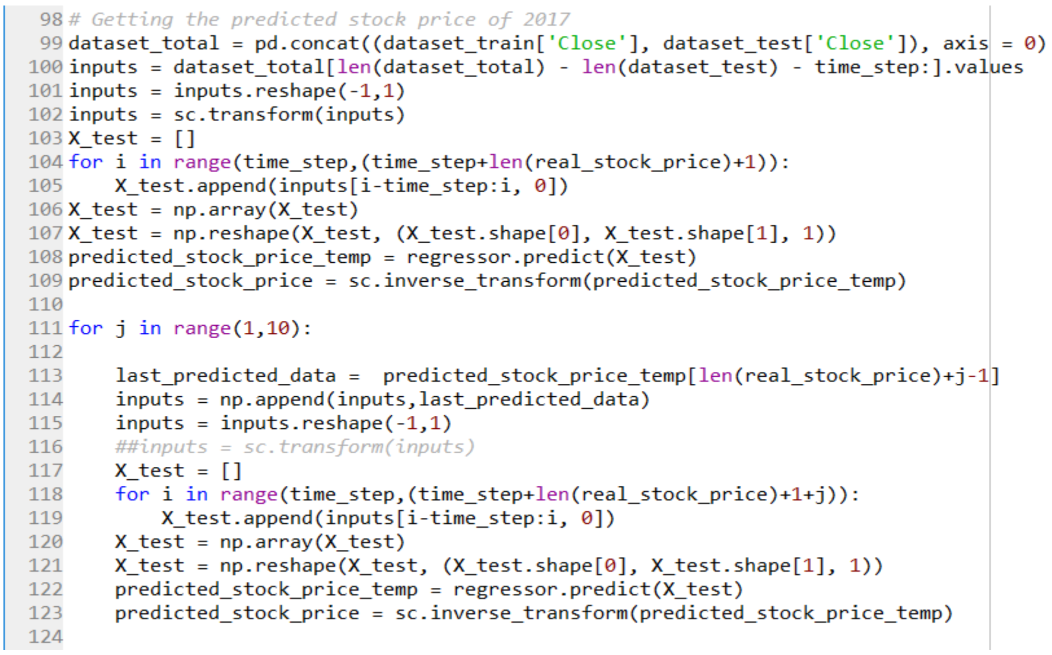
STEP 7 : Reading the testing data



```
# Part 3 - Making the predictions and visualising the results

# Getting the real stock price of 2017
dataset_test = pd.read_csv('NSEI_test1.csv')
real_stock_price = dataset_test.iloc[:, 4:5].values
real_stock_demo_price = real_stock_price

```

Here We are getting the testing data which We have downloaded from Yahoo Finance**.**

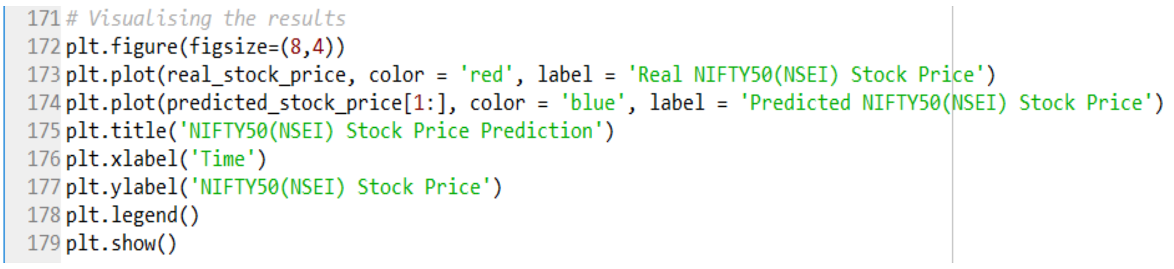
STEP 8 : Predicting the Stock Prices



```
# Getting the predicted stock price of 2017
dataset_total = pd.concat((dataset_train['Close'], dataset_test['Close']), axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - time_step:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(time_step,(time_step+len(real_stock_price)+1)):
    X_test.append(inputs[i-time_step:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_stock_price_temp = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price_temp)

for j in range(1,10):

    last_predicted_data =  predicted_stock_price_temp[len(real_stock_price)+j-1]
    inputs = np.append(inputs,last_predicted_data)
    inputs = inputs.reshape(-1,1)
    ##inputs = sc.transform(inputs)
    X_test = []
    for i in range(time_step,(time_step+len(real_stock_price)+1+j)):
        X_test.append(inputs[i-time_step:i, 0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
    predicted_stock_price_temp = regressor.predict(X_test)
    predicted_stock_price = sc.inverse_transform(predicted_stock_price_temp)

```

Here We are supplying the testing data to the model we have prepared. It will give us the predicted price of some day. We are predicting Stock Price of last 1, 2, 3 months on NSEI, BSESN, BTC-USD dataset which We get from Yahoo Finance. We are Predicting 10 days stock price ahead.

STEP 9 : Visualizing the real vs predicted stock price



```python
# Visualising the results
plt.figure(figsize=(8,4))
plt.plot(real_stock_price, color = 'red', label = 'Real NIFTY50(NSEI) Stock Price')
plt.plot(predicted_stock_price[1:], color = 'blue', label = 'Predicted NIFTY50(NSEI) Stock Price')
plt.title('NIFTY50(NSEI) Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('NIFTY50(NSEI) Stock Price')
plt.legend()
plt.show()
```

Here We are visualizing and comparing the real stock price vs predicted stock price through Graph and showing the next 10 days prediction. As we are taking the previous day predicted stock price as one of the feature in input vector so it will not be appropriate, will lead to going up or down, but We can have an idea about how the stock price will behave, We can predict if it will up or down for next few days.

Chapter 7

EXPERIMENTAL RESULTS AND ANALYSIS

We have executed our program for three different stock dataset named *NSEI (National Stock Exchange of India)*, *BSESN (Bombay Stock Exchange aka BSE SENSEX*), BTC-USD (BIT Cash USD), for different timesteps, different epoch values. Then after analyzing all the real vs predicted stock price graph we are going to select a optimum timestep and epochvalue associated with it for which the algorithm works perfectly, in every single dataset on which we executed the program.

7.1 NSEI (National Stock Exchange of India) :

Time Step = 40 and Epoch Value = 100 :

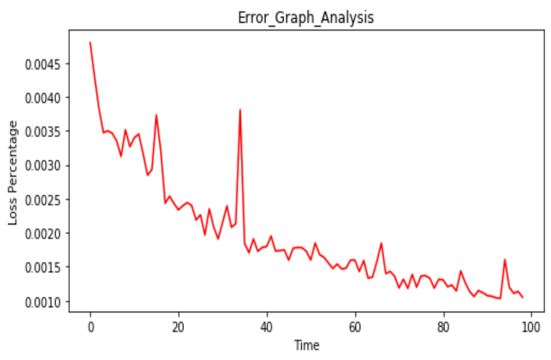


Figure 7.1.1 : Loss/Error Graph

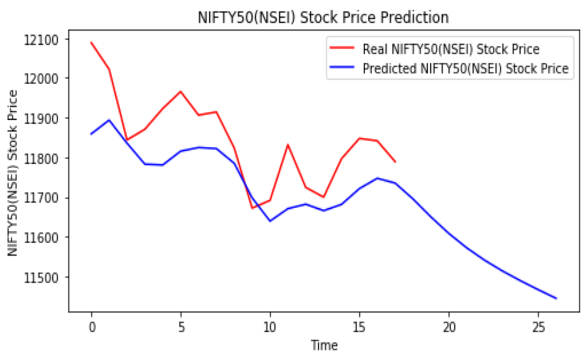


Figure 7.1.2 : Real vs Predicted Stock Price Graph

Time Step = 40 and Epoch Value = 250 :

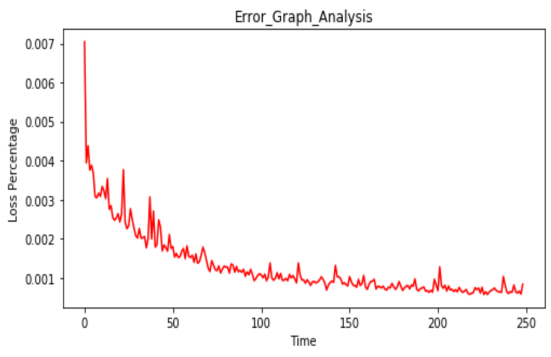


Figure 7.1.3 : Loss/Error Graph

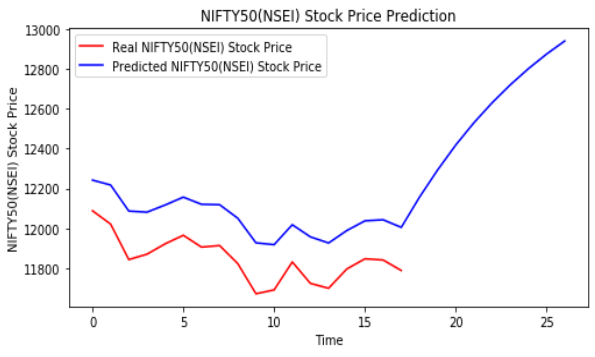


Figure 7.1.4 : Real vs Predicted Stock Price Graph

Time Step = 40 and Epoch Value = 500 :

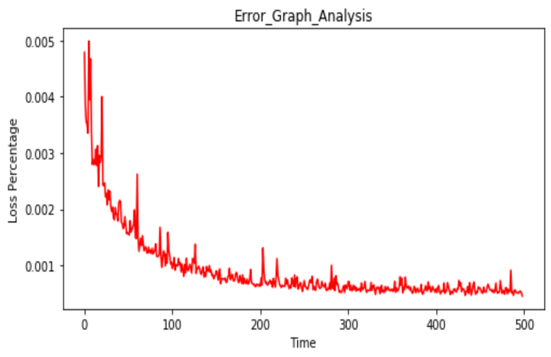


Figure 7.1.5 : Loss/Error Graph

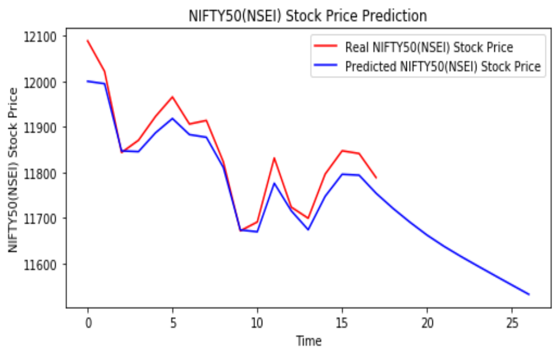


Figure 7.1.6 : Real vs Predicted Stock Price Graph

After analyzing the graph which We get from the execution taking **Time Step** = 40 we can say that the behavior of **NSEI Stock Price** is good in **Epoch Value** = 500, however in **Epoch Value** = 250 the **patterns** of two graph are same.

Time Step = 50 and Epoch Value = 100 :

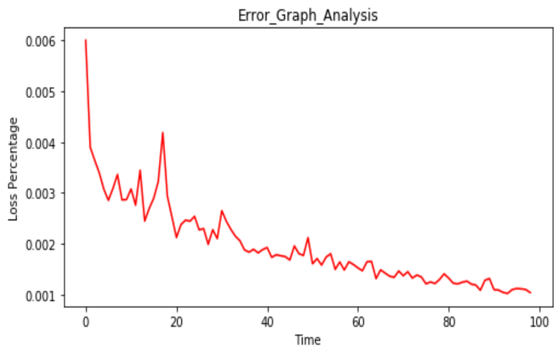


Figure 7.1.7 : Loss/Error Graph

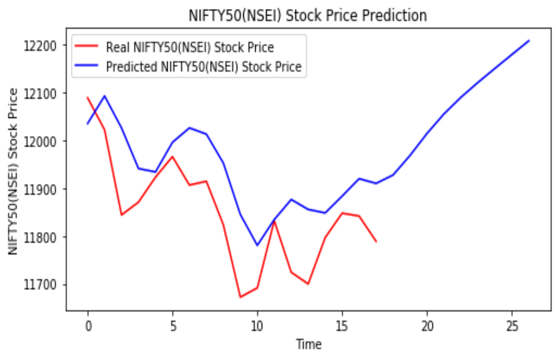


Figure 7.1.8 : Real vs Predicted Stock Price Graph

Time Step = 50 and Epoch Value = 250 :

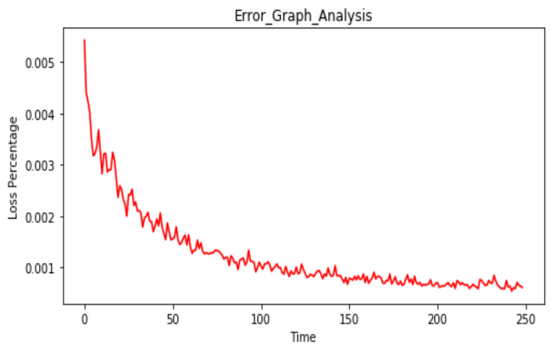


Figure 7.1.9 : Loss/Error Graph

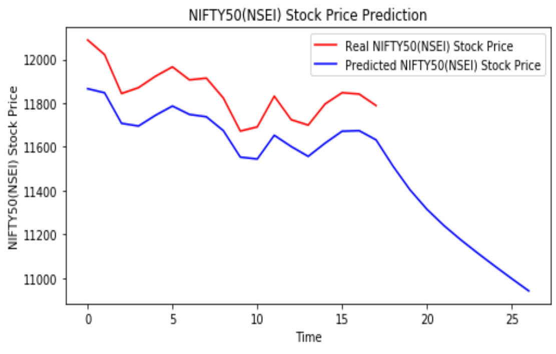


Figure 7.1.10 : Real vs Predicted Stock Price

Time Step = 50 and Epoch Value = 500 :

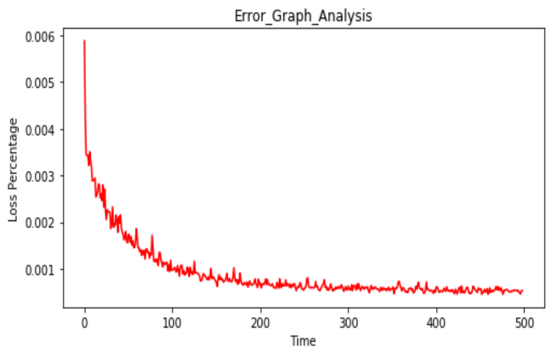


Figure 7.1.11 : Loss/Error Graph

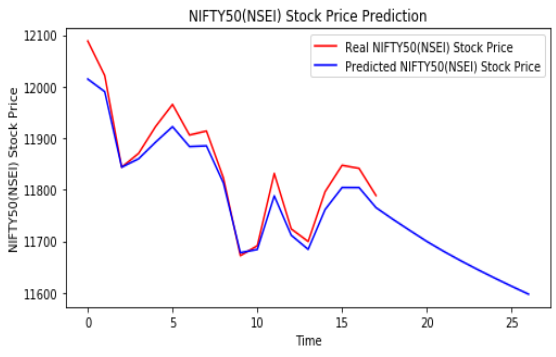


Figure 7.1.12 : Real vs Predicted Stock Price Graph

In this case when **Time Step** = 50, the graph patterns are same like **Time Step** = 40 in case of **Epoch Value** = 250, 500 but it is better than the case where Epoch Value = 100

Time Step = 60 and Epoch Value = 100 :

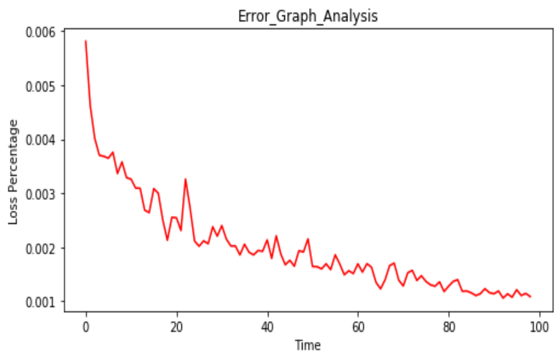


Figure 7.1.13 : Loss/Error Graph

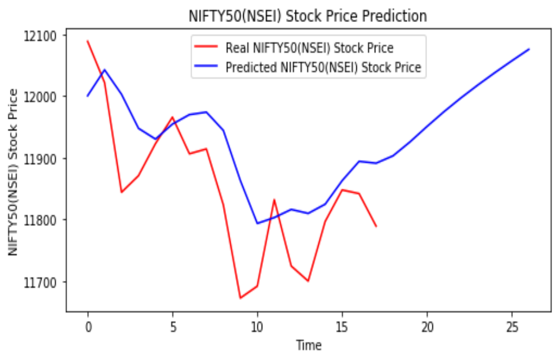


Figure 7.1.14 : Real vs Predicted Stock Price Graph

Time Step = 60 and Epoch Value = 250 :

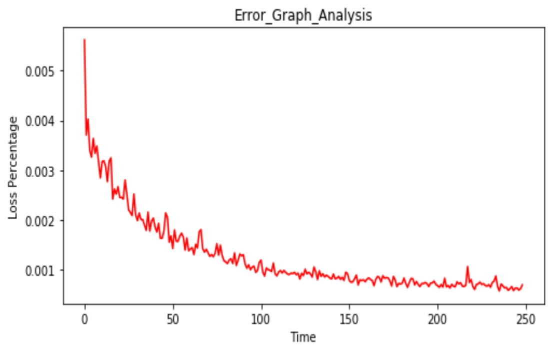


Figure 7.1.15 : Loss/Error Graph

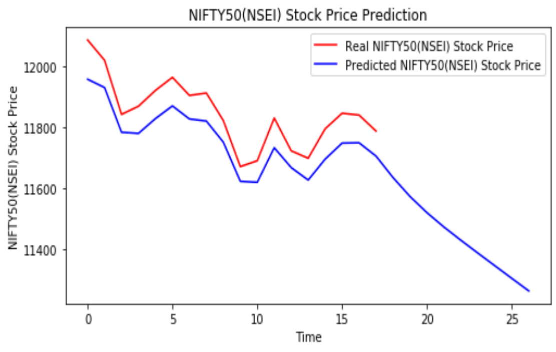


Figure 7.1.16 : Real vs Predicted Stock Price

Time Step = 60 and Epoch Value = 500 :

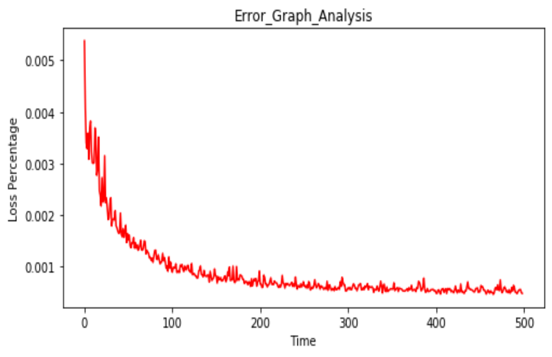


Figure 7.1.17 : Loss/Error Graph

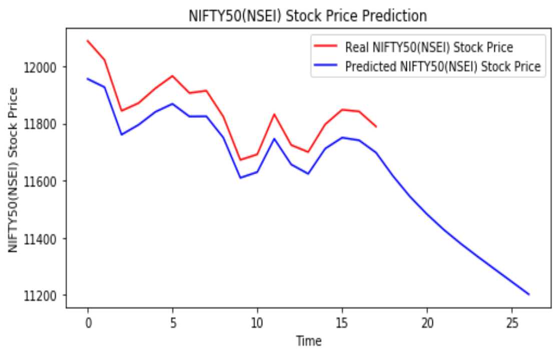


Figure 7.1.18 : Real vs Predicted Stock Price Graph

When We executed our model with **Time Step** = 60, noticed that The pattern of the Graphs are same in case of Epoch Value = 250, 500. It is also a good result, whenever it is not possible to accurately predict the **Stock Price** the **Pattern** will help much.

Time Step = 70 and Epoch Value = 100 :

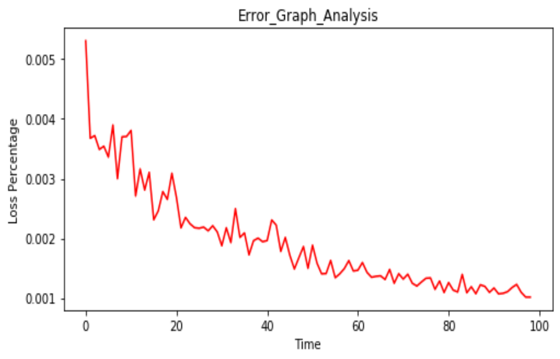


Figure 7.1.19 : Loss/Error Graph

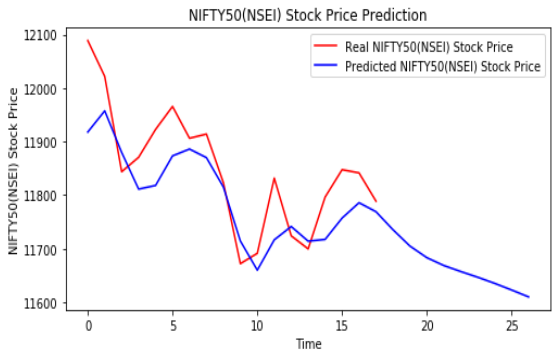


Figure 7.1.20 : Real vs Predicted Stock Price Graph

Time Step = 70 and Epoch Value = 250 :

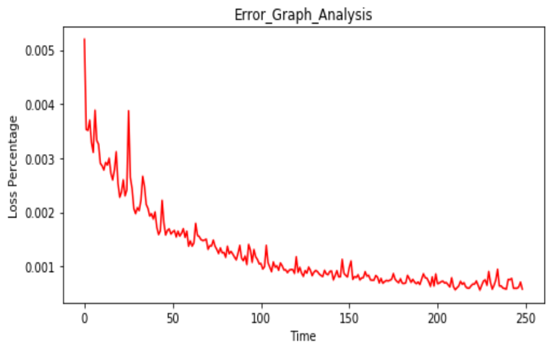


Figure 7.1.21 : Loss/Error Graph

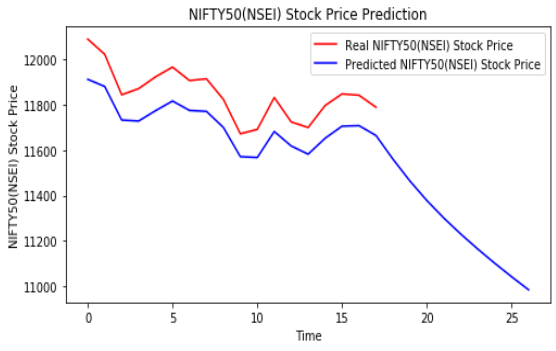


Figure 7.1.22 : Real vs Predicted Stock Price

Time Step = 70 and Epoch Value = 500 :

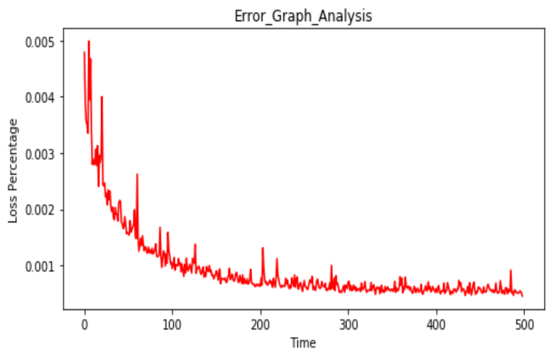


Figure 7.1.23 : Loss/Error Graph

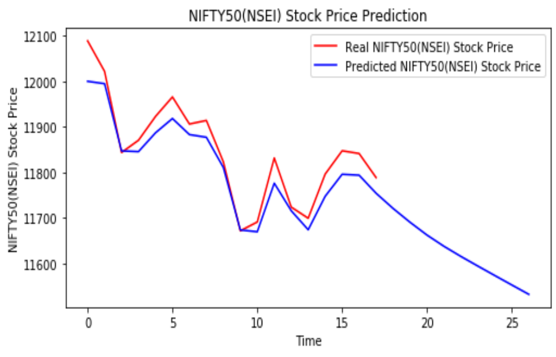


Figure 7.1.24 : Real vs Predicted Stock Price Graph

7.2 BSESN (Bombay Stock Exchange aka BSE SENSEX)

Time Step = 40 and Epoch Value = 100 :

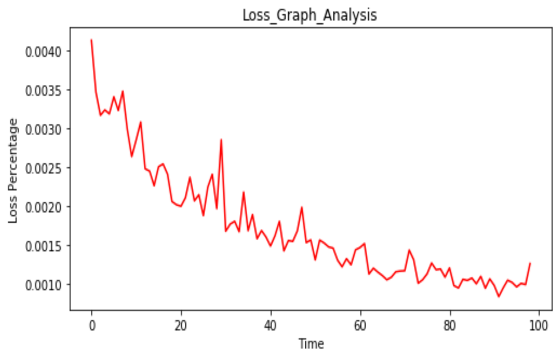


Figure 7.2.1 : Loss/Error Graph

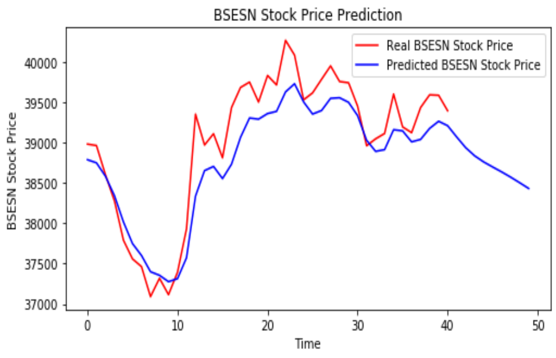


Figure 7.2.2 : Real vs Predicted Stock Price Graph

Time Step = 40 and Epoch Value = 250

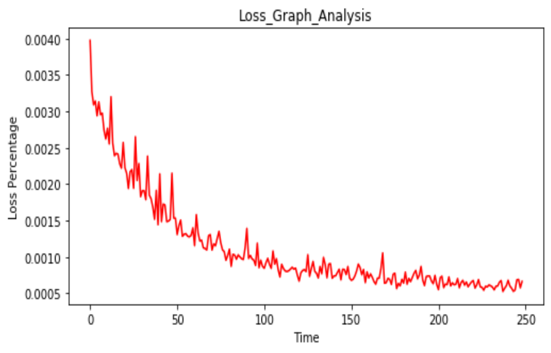


Figure 7.2.3 : Loss/Error Graph

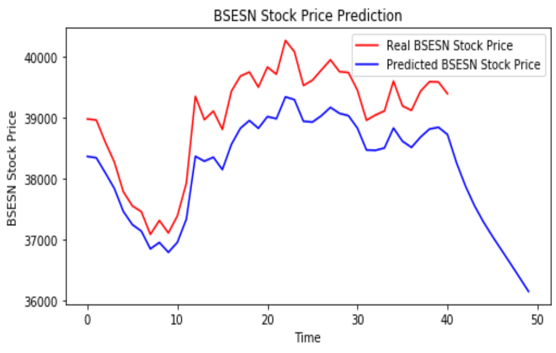


Figure 7.2.4 : Real vs Predicted Stock Price

Time Step = 40 and Epoch Value = 500 :

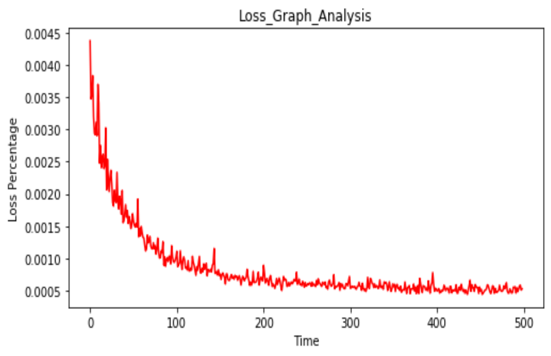


Figure 7.2.5 : Loss/Error Graph

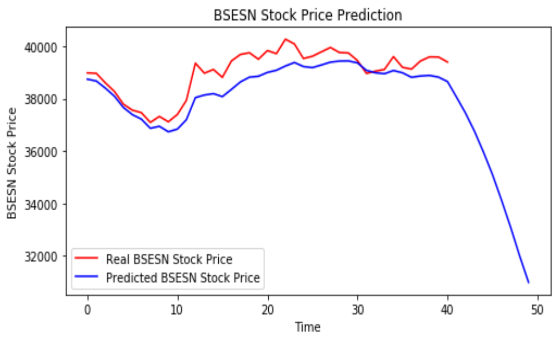


Figure 7.2.6 : Real vs Predicted Stock Price Graph

After analyzing the Real vs Predicted Price graph we can conclude that the prediction is being accurate while we increasing the Epoch Value, also in case of Epoch Value = 100, 250 the patterns of the graph are very much same.

Time Step = 50 and Epoch Value = 100 :

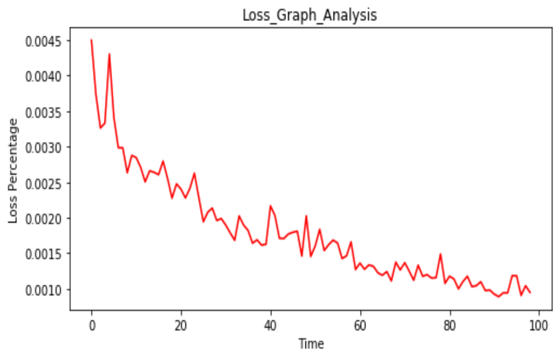


Figure 7.2.7 : Loss/Error Graph

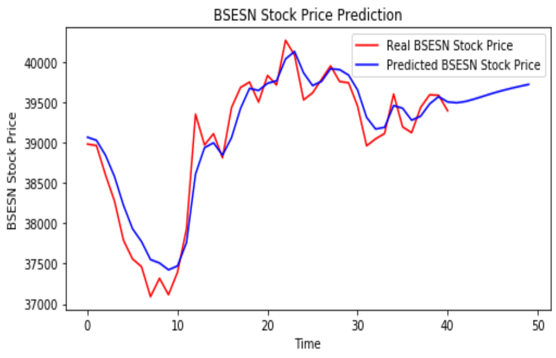


Figure 7.2.8 : Real vs Predicted Stock Price Graph

Time Step = 50 and Epoch Value = 250 :

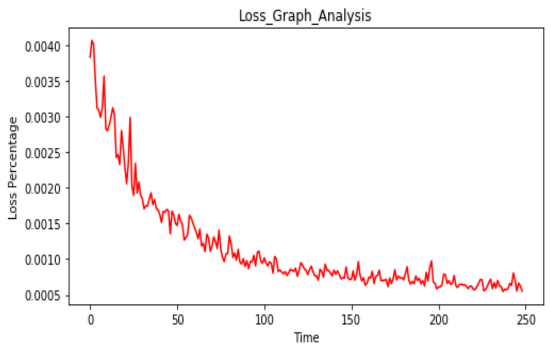


Figure 7.2.9 : Loss/Error Graph

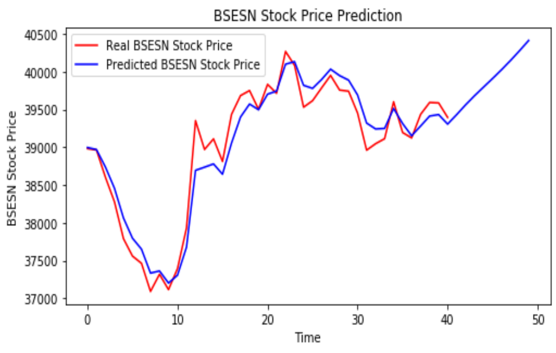


Figure 7.2.10 : Real vs Predicted Stock Price

Time Step = 50 and Epoch Value = 500 :

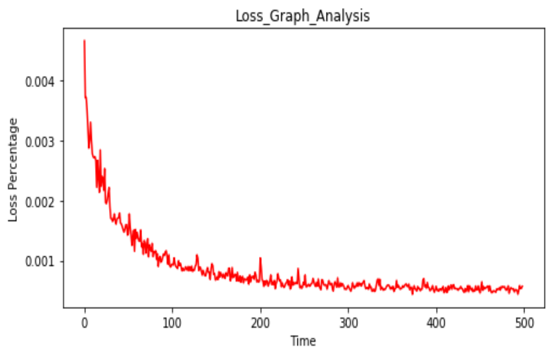


Figure 7.2.11 : Loss/Error Graph

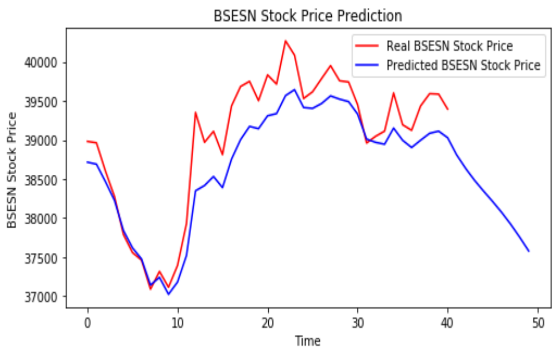


Figure 7.2.12 : Real vs Predicted Stock Price Graph

In this case Time Step = 50 the model has been introduced with some overfitting, but not in a great amount. It is considerable.

Time Step = 60 and Epoch Value = 100 :

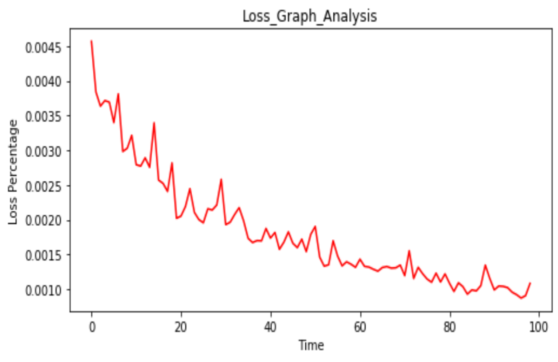


Figure 7.2.13 : Loss/Error Graph

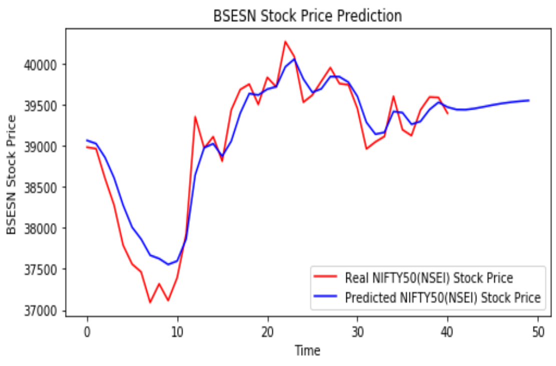


Figure 7.2.14 : Real vs Predicted Stock Price Graph

Time Step = 60 and Epoch Value = 250 :

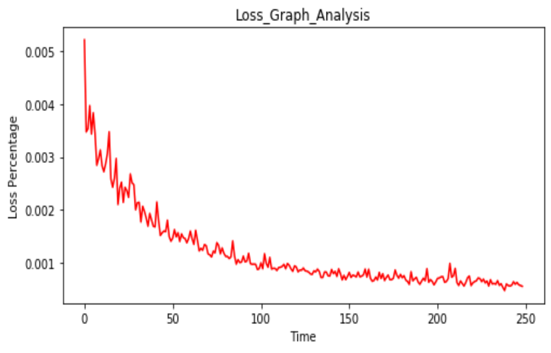


Figure 7.2.15 : Loss/Error Graph

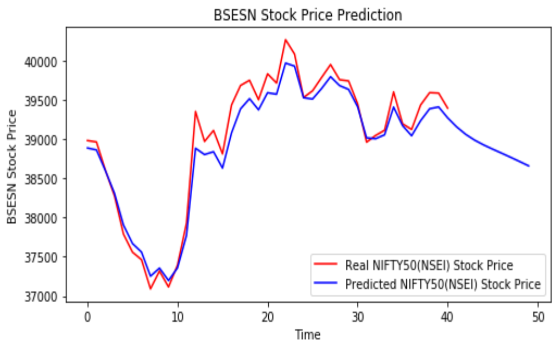


Figure 7.2.16 : Real vs Predicted Stock Price

Time Step = 60 and Epoch Value = 500 :

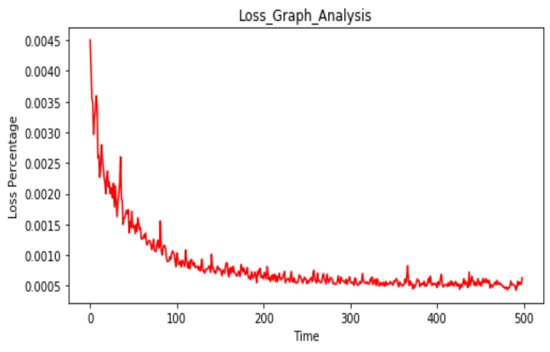


Figure 7.2.17 : Loss/Error Graph

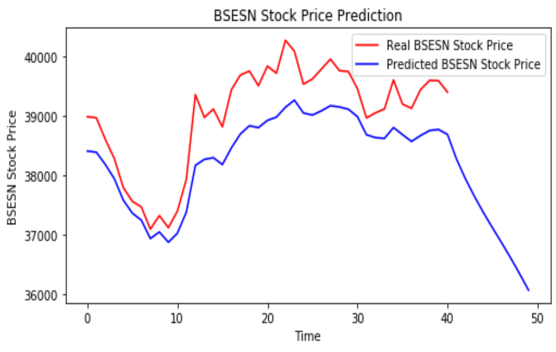


Figure 7.2.18 : Real vs Predicted Stock Price Graph

Same things happen in the case of timestep = 60, but in this case also epoch value = 100, 250 the predicted results are very good. So, we can conclude that in case of BSESN when we will increase the Epoch Value the results will be going towards low accuracy.

Time Step = 70 and Epoch Value = 100 :

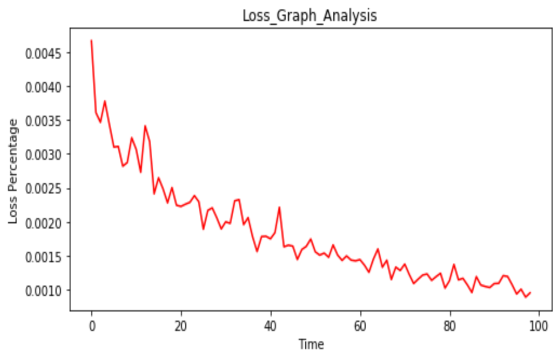


Figure 7.2.19 : Loss/Error Graph

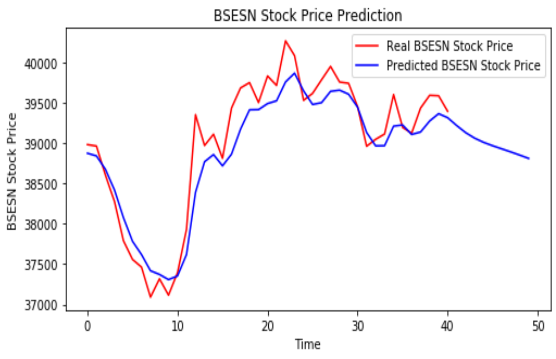


Figure 7.2.20 : Real vs Predicted Stock Price Graph

Time Step = 70 and Epoch Value = 250 :

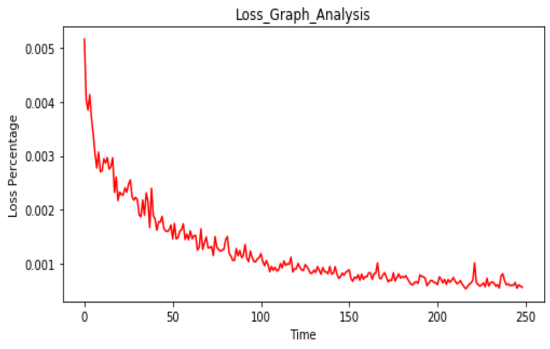


Figure 7.2.21 : Loss/Error Graph

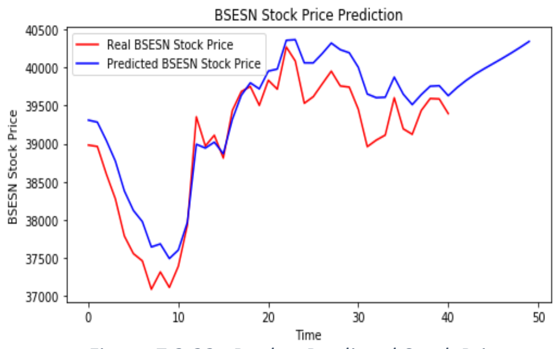


Figure 7.2.22 : Real vs Predicted Stock Price

Time Step = 70 and Epoch Value = 500 :

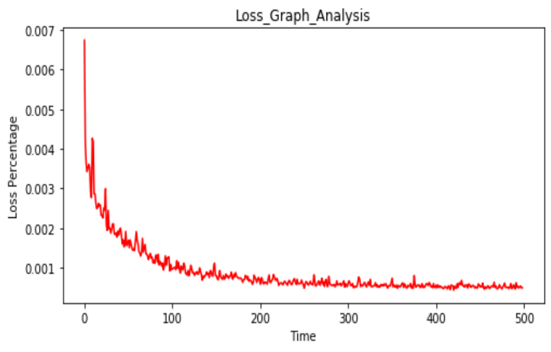


Figure 7.2.23 : Loss/Error Graph

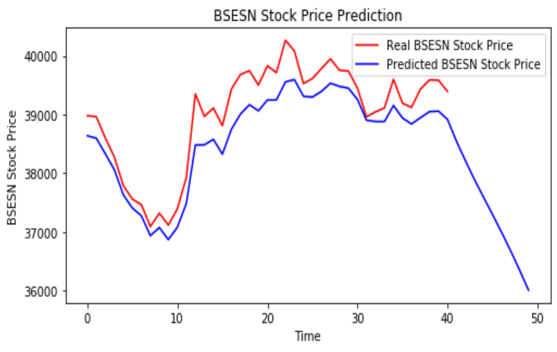


Figure 7.2.24 : Real vs Predicted Stock Price Graph

7.3 BTC-USD (Bitcoin in US Dollars value.)

Time Step = 40 and Epoch Value = 100 :

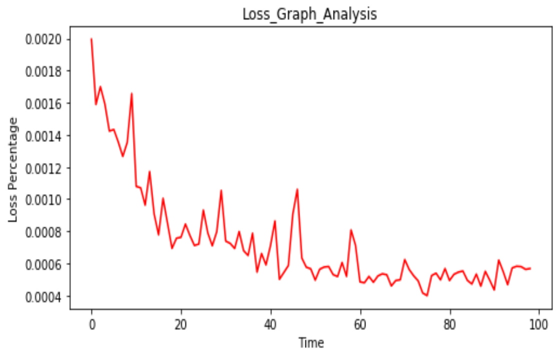


Figure 7.3.1 : Loss/Error Graph

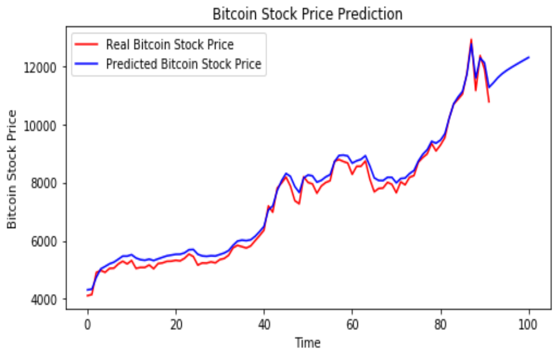


Figure 7.3.2 : Real vs Predicted Stock Price Graph

Time Step = 40 and Epoch Value = 250 :

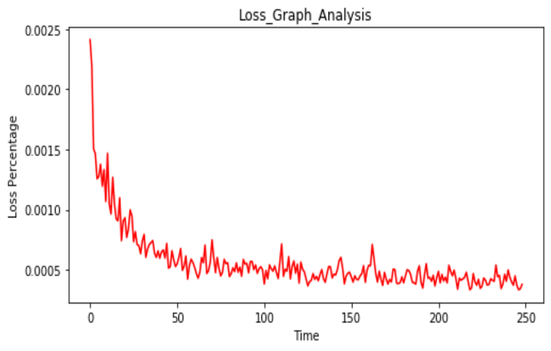


Figure 7.3.3 : Loss/Error Graph

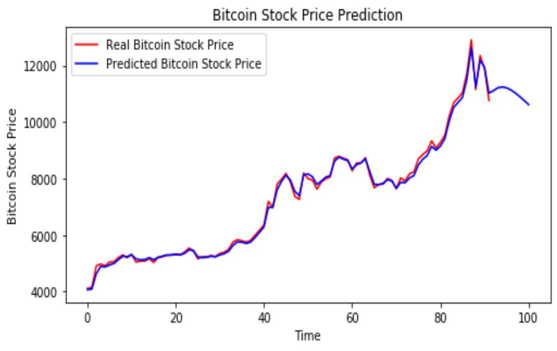


Figure 7.3.4 : Real vs Predicted Stock Price

Time Step = 40 and Epoch Value = 500 :

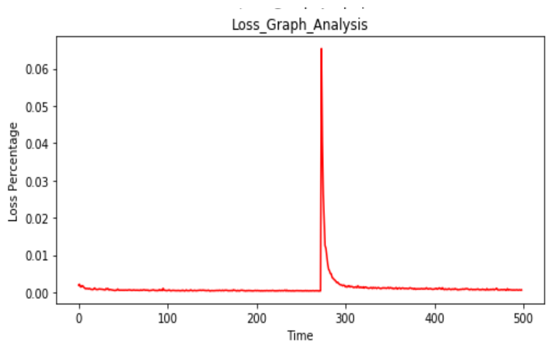


Figure 7.3.5 : Loss/Error Graph

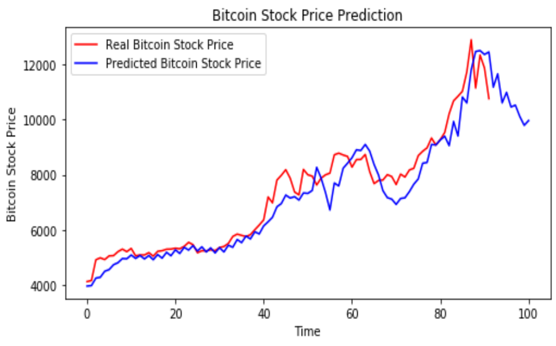


Figure 7.3.6 : Real vs Predicted Stock Price Graph

When We are working on the BITC-USD dataset we can see the model is giving us a wonderful result, but introduced with overfitting when We are increasing the epoch value. But the result for epoch value = 500 is not bad at all and we can notice here that the predicted value for next 10 days is not continuously increasing or decreasing. So, we can say that the model is worked very good.

Time Step = 50 and Epoch Value = 100 :

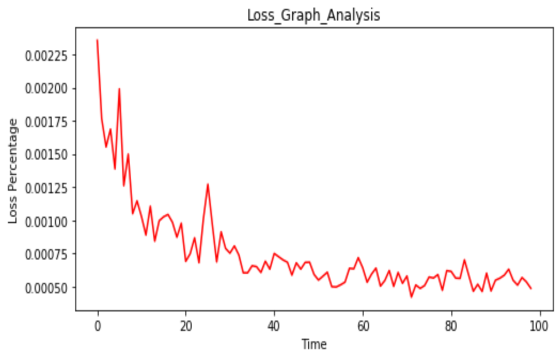


Figure 7.3.7 : Loss/Error Graph

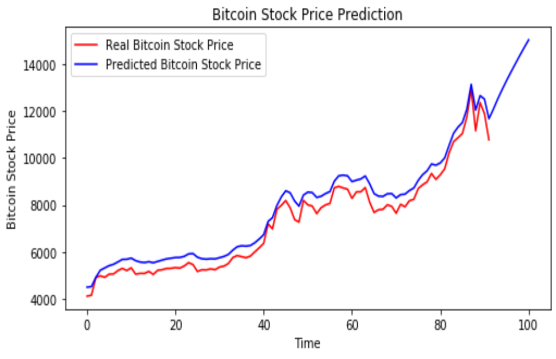


Figure 7.3.8 : Real vs Predicted Stock Price Graph

Time Step = 50 and Epoch Value = 250

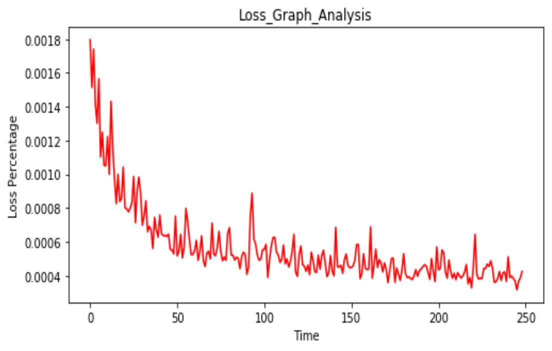


Figure 7.3.9 : Loss/Error Graph

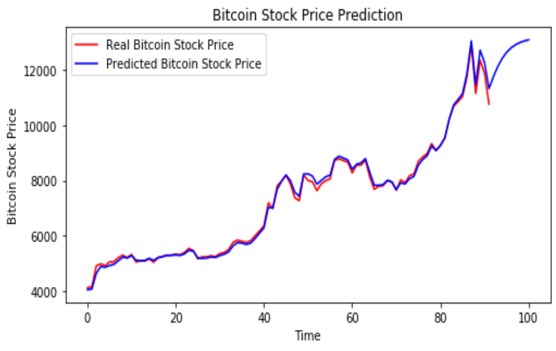


Figure 7.3.10 : Real vs Predicted Stock Price

Time Step = 50 and Epoch Value = 500 :

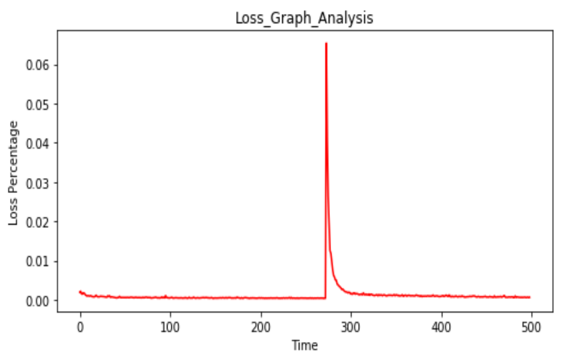


Figure 7.3.11 : Loss/Error Graph

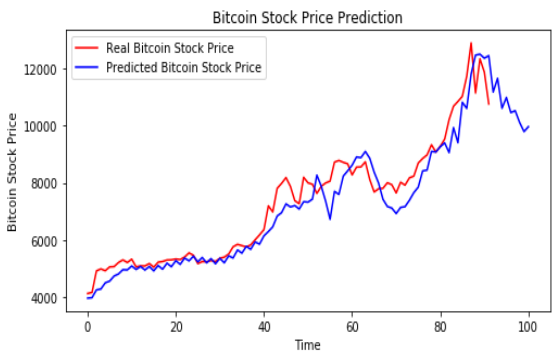


Figure 7.3.12 : Real vs Predicted Stock Price Graph

Time Step = 60 and Epoch Value = 100 :

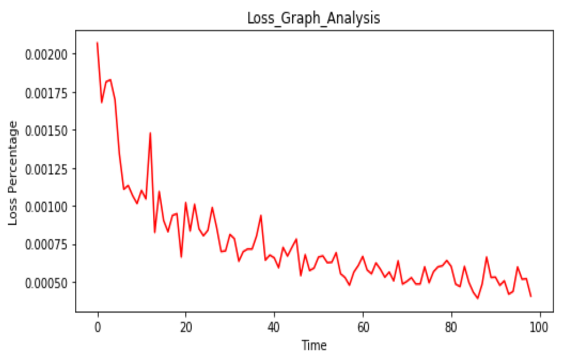


Figure 7.3.13 : Loss/Error Graph

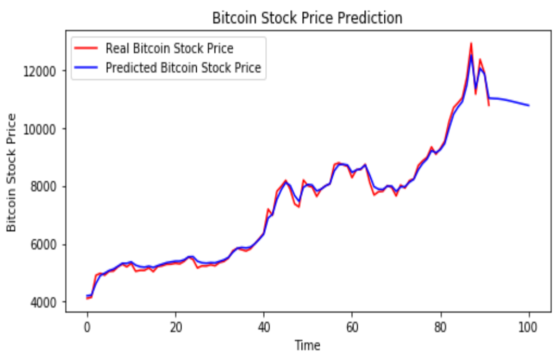
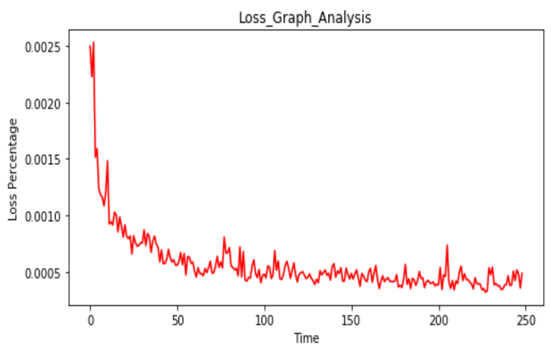
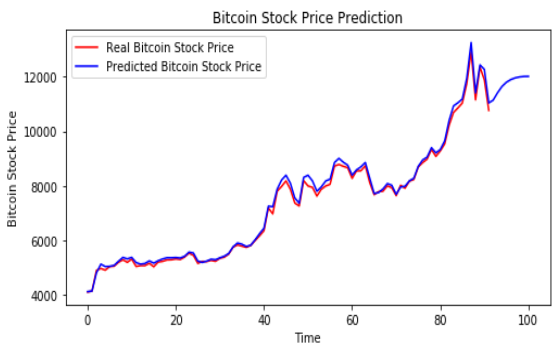


Figure 7.3.14 : Real vs Predicted Stock Price Graph

Time Step = 60 and Epoch Value = 250 :





Time Step = 60 and Epoch Value = 500 :

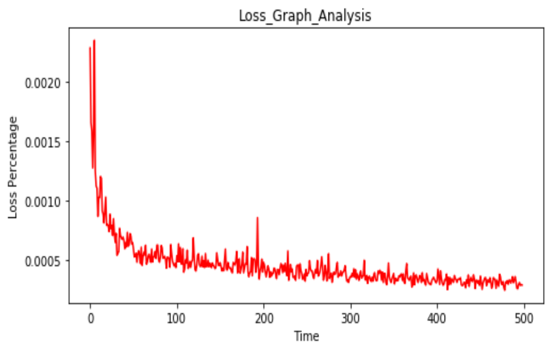


Figure 7.3.17 : Loss/Error Graph

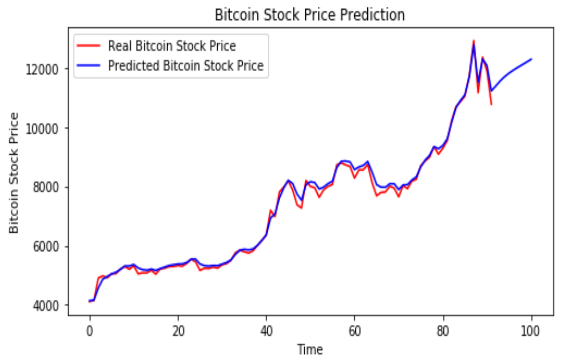


Figure 7.3.18 : Real vs Predicted Stock Price Graph

Time Step = 70 and Epoch Value = 100 :

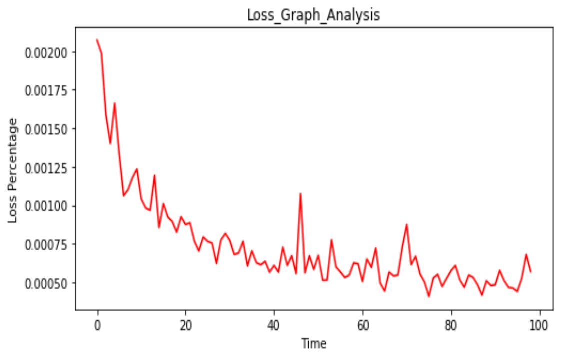


Figure 7.3.19 : Loss/Error Graph

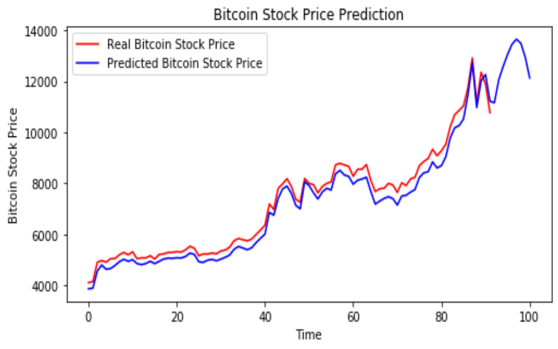


Figure 7.3.20 : Real vs Predicted Stock Price Graph

Time Step = 70 and Epoch Value = 250 :

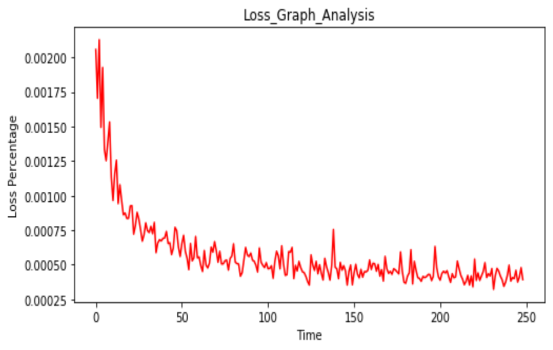


Figure 7.3.21 : Loss/Error Graph

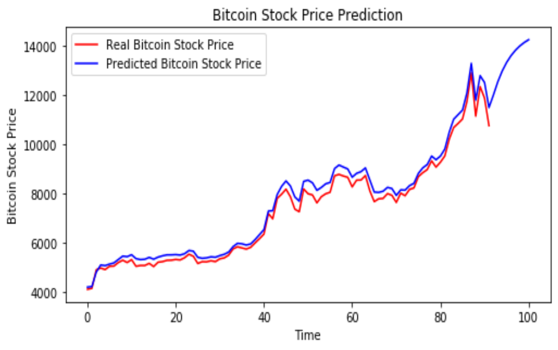


Figure 7.3.22 : Real vs Predicted Stock Price

Time Step = 70 and Epoch Value = 500 :

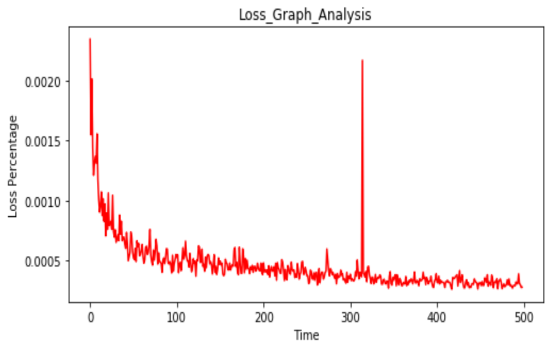


Figure 7.3.23 : Loss/Error Graph

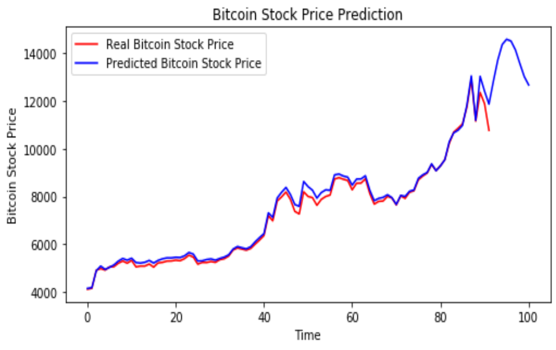


Figure 7.3.24. : Real vs Predicted Stock Price Graph

In all the cases we are getting the same type of results when we work with BTC-USD dataset. Now we can conclude that if We are going to test the model with large amount (not very large) of data this will work perfectly as the range of Stock prices are increasing. When there is big difference in previous day's closing price and next day's closing price then it will be very easy for the model to predict. For this in case of BTC-USD the model is working very appropriate.

We have executed our model for three different dataset which are mentioned as NSEI, BSESN, BTC-USD, for three different epoch values 100, 250, 500 considering four different timesteps 40, 50, 60, and 70 respectively. We showed the loss or error graph for all the cases we have executed the model.

In some cases where epoch value is set to 100 the predicted stock price after executing the model are giving us an idea about the behavior of stock, how it is changing, but the results are not so appropriate in some cases. In case of NSEI the prediction is good at timestep 40, 50, 70 but not in timestep 70. We can analyze the result from Figure:7.1.2 , Figure:7.1.8 , Figure:7.1.20 and Figure:7.1.14 respectively. In Figure:7.1.14 the prediction graph and real stock graph are not of same pattern some places. Figure:7.1.2 and Figure:7.1.20 shows that the future stocks will going down and Figure:7.1.8 and Figure:7.1.14 shows that the future stocks will going to increase. Although they are showing some pattern of predicted stock price matched with the real stock price we can't consider this epoch value (100) for this dataset for prediction. We have to further investigate for different epoch values. In case of epoch value 250 on NSEI stocks giving us the prediction more accurately. In Figure:7.1.4 , Figure:7.1.10 , Figure:7.1.16 and Figure:7.1.22 the model is able to generate a pattern of changing of predicted stock price which is exactly same as the real stock price change. Here also the decision of predicting the stock price will be increasing or decreasing can't be determined appropriately as Figure:7.1.4 is showing the stock price is going to be increasing but Figure:7.1.10 , Figure:7.1.16 , Figure:7.1.22 are showing the stock market will down in future. So, we can have an idea that the stock price will decrease in the coming few days as three graph Figure:7.1.10 , Figure:7.1.16 , Figure:7.1.22 are leading us to that conclusion. Now, considering epoch value 500 and analyzing the graphs for different timestep, Figure:7.1.6 , Figure:7.1.12 , Figure:7.1.18 and Figure:7.1.24 we can conclude that they not only give us a suitable pattern for real vs predicted stock values but also the difference between real and predicted stock prices are also very less, which we can conclude as a satisfactory observation. In epoch value 500, the graphs are also indicating towards one possibility regarding the behavior of future stock, they all are saying that the stock market (closing price) will be going to down for next few days. We can determine after investigating Figure:7.1.6 , Figure:7.1.12 , Figure:7.1.18 and Figure:7.1.24 although timestep 40, 50, 70 are giving us the satisfactory results but for timestep 50 the real vs predicted stock price graph is giving us the best result. So we can conclude that if we consider timestep as 50 and epoch value as 500 for NSEI stock the result will be perfect.

In case of BSESN stock where the epoch value 100, they are giving us the result pretty good in some cases. Figure:7.2.2 , Figure:7.2.8 , Figure:7.2.14 and Figure:7.2.20 are showing us a very good result even in epoch value 100. They have clearly given a pattern in all the cases how stocks are behaving, considerably Figure:7.2.8 , Figure:7.2.14 , Figure:7.2.20 the results are good compared to Figure:7.2.2. Here also the behavior of predicted future stocks are not same for all the graphs, In Figure:7.2.2 and Figure:7.2.20 future stock price decreasing and Figure:7.2.8 and Figure:7.2.14 the future stock price increasing. So, we can't conclude it as a satisfactory answer as we are focusing on the future stock price for some day although the difference of real and predicted stock prices were very much small and negligible. Now considering the epoch value 250 for different timesteps, we analyze that for timestep 50 and 60 the model give us satisfactory prediction Figure:7.2.10 and Figure 7.2.16 respectively, but in case of Figure:7.2.4 and Figure:7.2.22 which means for timestep 40 and 70 the model is not working perfectly, some of the predicted values are perfectly matching with the actual stock values but some of them are not satisfactorily matching. So, we can conclude timestep 40 and 70 can't be consider as the optimum timestep for BSESN stock. Further investigation done on the graphs which we get executing the model for epoch value 500. We can now conclude that for timestep 40 and timestep 60, Figure:7.2.6 and Figure:7.2.18 respectively the behavior of the predicted vs real stock price is not good enough although Figure:7.2.12 and Figure:7.2.24 can be consider as a good prediction. Now after analyzing the behavior of all the graphs, Figure:7.2.6 , Figure:7.2.12 , Figure:7.2.18 , Figure:7.2.24, we can say that the stock price will decrease in for few days as they all are indicating that the stock price is going to down. Now considering all the graphs it is very obvious that time step 50 and 60 are optimum timestep for BSESN stock, and the preferable epoch value will be 250. But as we have examined that the stock will going to decreases so we are selecting timestep 50 as optimum one as it can show us the stocks will be down for some days.

Now, analyze the model how it worked for BTC-USD stock. For epoch value 100 timestep 60 and 70, Figure:7.3.14, Figure:7.3.20 respectively are giving us incredibly good prediction. The line for predicted and actual stock price are overlapped everywhere and for timestep 40 and 50, Figure:7.3.2, Figure:7.3.8, also the prediction are not so bad, considerable. After this we can come into an conclusion that if the difference of stock price (closing price) of previous day and next day are large and the closing prices pattern of increasing or decreasing then the prediction can be done perfectly even in 100 epoch values, as in the case of BTC-USD stock. In BTC-USD stock the closing prices are increasing rapidly compared to the NSEI stocks and BSESN stocks. Now examining for epoch value 250, we discovered that the prediction are very good at timestep 40, 50, 60, Figure:7.3.4, Figure:7.3.10 and Figure:7.3.16 respectively and considerably good at timestep 70, Figure:7.3.22. Analyzing the graphs we can conclude that Figure:7.3.10, Figure:7.3.16, Figure:7.3.22 are giving us the prediction that stock price will rise for next few days but Figure:7.3.4 is not leading us to the same conclusion. So, we will not be considering 40 as a optimum timestep. A new characteristic of the graphs is being observed after analyzing for epoch value 500, Figure:7.3.6, Figure:7.3.12, Figure:7.3.18 and Figure:7.3.24. Here, we observe that the predicted price are also fluctuating which could never be observed before, predicted price were either continuously decreasing or continuously increasing before. In Figure:7.3.12 and Figure:7.3.6 the model is being introduced with overfitting, For predicted stock price are not behaving properly in this to cases. So, after all that we can conclude 50 as an optimum timestep and if we have to make analysis on actual vs predicted price we can take epoch value as 250 or 500 or if we have to predict how stocks will behave in future then we have to take 500 as epoch value.

Now we have evaluated our model for different dataset, for different epoch values taking different timesteps in consideration, previously no one evaluated their model after executing the model for different dataset. We are fixing the timestep and epoch values manually, in the future scope we can do one thing that not choosing the epoch values and timestep manually we can do that after analyzing all the evaluation the model will self recognize the moderate epoch value and optimum time step. We are taking 40, 50, 60, 70, large amount of input for predicting the next day value (use of LSTM), no other model is considering that much timesteps to predict the stocks. In our model we can predict the stock prices for next 10 days, however the prediction for future is not very appropriate but from this prediction also we can have an idea how the stocks will behave for few days. All the existing model is just analyze how predicted price behaving with actual stock price, but our model can determine 10 days prediction for stock price (not so appropriate) additional with analyzing the actual vs predicted stock behavior.

Chapter 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion:

This model presents the prediction of stock prices using Machine Learning / Deep Learning approach particularly using RNN and LSTM. Through this model, we can predict the prices of stocks if they would go high or low. The model predicts prices upto 10 days ahead at hand.

The related work done over this area had analysis of their model about how much their prediction is correct or not. But our model can predict the prices of future or coming days.

However, the model doesn’t perform well in small changes as the variation of those small changes are infinitesimally small in compare to the stock prices.

8.2 Future Scope:

This model sets a new benchmark in the IT industry. Yet, this model can be improved in various ways.

a) We can include rate of difference of stock prices of every day to find better results.

b) More accuracy can be attained if variation of highest and lowest prices range can be set into any relation.

c) We set timesteps and epochs manually in this model, to train the model in such a way to select the optimum timesteps and epochs such that overfitting and underfitting doesn’t happen and model decide the timesteps and epochs by its own can be a level up task for the model.

However, prices predicted in the model are based on the previous prices so, if the prices change due to any other factor or emergent conditions, the model couldn’t predict it.

Chapter 9

REFERENCES

1. Weng L., https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html, May-July,(2019).

2. Malm R., https://www.kaggle.com/raoulma/ny-stock-price-prediction-rnn-lstm-gru., May-July,(2019).

3. Nayak A., https://towardsdatascience.com/predicting-stock-price-with-lstm-13af86a74944, May-July,(2019).

4. Singh A., https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/, May-July,(2019).

5. Akita R., Yoshihara A., Matsubara T., Uehra K., "Deep Learning for Stock Prediction using Numerical and Textual Information", copyright: IEEE ICIS 2016, Available: Google Scholar, (2016).

6. Nelson David M. Q., Pereira Adriano C. M., de Oliveira Renato A., "Stock Market's Price Movement Prediction With LSTM Neural Networks", copyright IEEE, Available: Google Scholar, (2017).

7. Selvin S., Vinayakumar R, Gopalakrishnan E.A, Menon V. K., Soman K.P, "STOCK PRICE PREDICTION USING LSTM,RNN AND CNN-SLIDING WINDOW MODEL", copyright IEEE, Available: Google Scholar, (2017).

8. Roondiwala M., Patel H., Varma S., "Predicting Stock Prices Using LSTM", Paper ID: ART20172755, licensed under Creative Commons Attributions CC BY, (2017).

9. Nayak A., Pai M. M. M., Pai R. M., "Prediction Models for Indian Stock Market", Paper Id and Licensed by: Procedia Computer Science 89(2016)441–449, (2016).

10. Nandakumar R., Uttamraj K. R., Vishal R., Lokeswari Y. V., "Stock Price Prediction Using Long Short Term Memory", Volume:05 Issue:03,Mar-2018, e-ISSN: 2395-0056 p-ISSN: 2395-0072, (2018).

11. Kazem A., Sharifi E., Hussain F. K., Saberi M., Hussain O. K., "Support vector regression with chaos-based firefly algorithm for stock market price forecasting", copyright by Elsevier B.V., Volume 13, Issue 2, Pages 947-958, (2013).

12. Choudhry R., Garg K., "A Hybrid Machine Learning System for Stock Market Forecasting", World Academy of Science, Engineering and Technology 39, (2008).

13. Patel J., Shah S., Thakkar P., Kotecha K., "Predicting stock market index using fusion of machine learning techniques", Copyright by Elsevier Ltd., Volume 42, Issue 4, Pages 2162-2172, (2015).

14. Lee J. W., "Stock price prediction using reinforcement learning", Published in: ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570), INSPEC Accession Number: 7091841, (2001).

15. Shen S., Jiang H., Zhang T., "Stock Market Forecasting Using Machine Learning Algorithms", Available: Google Scholar, (2012)

16. Qian B., Rasheed K., "Stock market prediction with multiple classifiers", copyright by Springer Volume 26, Issue 1, pp 25–33, (2007).