STOCK PRICE PREDICTION USING SENTIMENT ANALYSIS

A PROJECT REPORT

Submitted by

AMIT YADAV
PRASHANT KUMAR VERMA
SAURABH SHUKLA

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

University Institute of Engineering and Technology
C. S. J. M. University, Kanpur

MAY 2020

University Institute of Engineering and Technology
C. S. J. M. University, Kanpur

BONAFIDE CERTIFICATE

Certified that this project report **“Stock Price Prediction Using Sentiment Analysis”** is the bonafide work of **“Amit Yadav (07), Prashant Kumar Verma (38), Saurabh Shukla (53)”** who carried out the project work under my supervision.

SIGNATURE
Mr. Shesh Mani Tiwari
SUPERVISOR
CSE Department, UIET
CSJM University, Kanpur

Abstract

Predicting stock market prices has been a topic of interest among both analysts and researchers for a long time. Stock prices are hard to predict because of their high volatile nature. Along with the known parameters such as previous day stock prices, price to earning ratio, stock prices largely depend on some of the unknown factors such as election results, rumors, financial news, public sentiment, etc. Existing studies in sentiment analysis have found that there is a strong correlation between the movement of stock prices and the publication of news articles. The aim of this project is to predict the stock prices of some IT companies listed in NSE, BSE using the previous year dataset and data extracted from various users tweets, social media etc. For this we will make use of Recurrent Neural Network and Long Short Term Memory (LSTM) Neural Network to get the best possible prediction.

TABLE OF CONTENTS

ACKNOWLEDGEMENT 5

1. Introduction 6

1.1 Problem Statement 6

1.2 Objectives 7

2. Literature Review 8

2.1 Papers 8

3. Methodology and Implementation 10

3.1 Basic Structure 10

3.2 Requirements 11

3.3 Datasets 11

3.4 Preprocessing of data 12

3.5 Algorithms and Techniques 13

3.6 Proposed Model 16

4. Results 17

5. Conclusions and Future Work 19

5.1 Conclusions 19

5.2 Future Work 19

6. Project Code 20

7. References 24

List of figures

Figure No.	Name of the figure	Page No.
1.1	Compares Bombay Stock Exchange with Standard & Poor's 500, NASDAQ and DOW	6
3.1	Data Flow Diagram	10
3.2	Table Showing the Attributes of Stock dataset	12
3.3	Table Showing the Attributes of Stock dataset with Sentiment	13
3.4	An unrolled Recurrent Neural Network	14
3.5	LSTM Network Architecture	15
3.6	Architecture of the Proposed Model	16
4.1	Prediction without Sentiment Analysis	17
4.2	Prediction with Sentiment Analysis	18

Acknowledgement

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I am extremely thankful to our supervisor **Mr. Shesh Mani Tiwari**, who took a keen interest in our project work and guided us all along, till the completion of our project by providing all the necessary information.

I would also like to express my gratitude towards my team members for their kind cooperation and encouragement which helped me in completing this project. I am thankful and fortunate enough to get constant encouragement, support and guidance from all my colleagues who helped me in successfully completing the project work.

Chapter 1

Introduction

Stock prices are very fluctuating in nature. They depend on various factors like the previous stock prices, current market trends, financial news, competitor's performance etc. The Efficient Market Hypothesis (EMH) states that all relevant information is fully and immediately reflected into stock prices, hence the price fluctuations are unpredictable. Many researchers have used Machine Learning techniques to better predict the stock prices, however most of the research is about the international market which is comparatively much stable as compared to the Indian stock market.

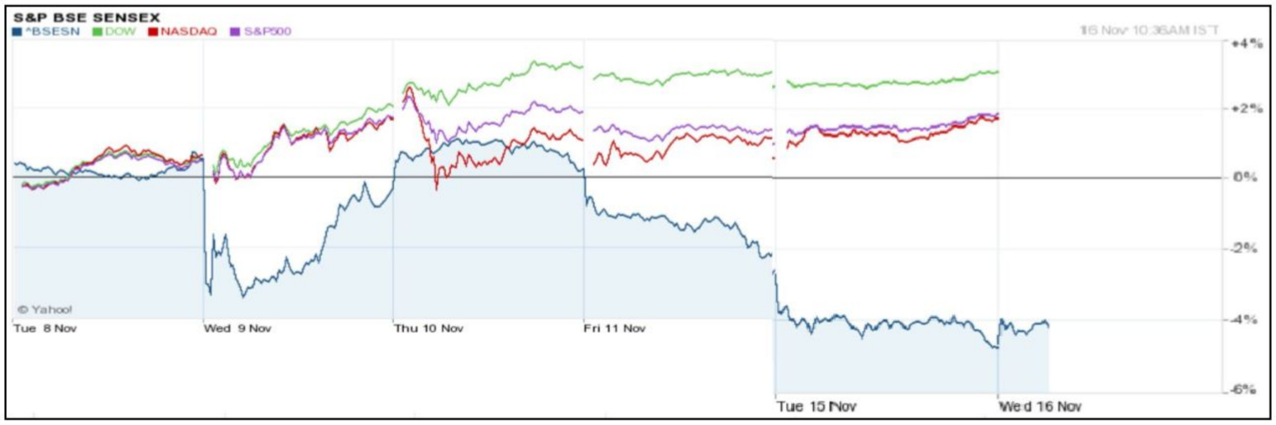


Fig 1.1 Compares Bombay Stock Exchange with Standard & Poor's 500, NASDAQ and DOW

This leads to the question: How can Machine Learning be used to better predict such fluctuating markets like the Indian stock market? Among the sectors: I.T., Financial, FMCG, Banking, Pharma and Media, this project focuses on I.T. sector.

1.1 Problem Statement

In this project, we proposed to predict the stock prices of some of the I.T. companies which are listed in National Stock Exchange (NSE), Bombay Stock Exchange (BSE) by correlating public sentiment from Twitter with historical stock market data using Machine Learning techniques.

1.2 Objectives

● To gather data about stock prices, financial events, and public sentiments from various data sources and build a predictive model.

● Use this model to predict the stock prices of some companies in the Indian I.T. domain.

Chapter 2

Literature Review

We studied various papers describing the research done in this domain and also the comparison of various models. This chapter summaries some of those research papers.

2.1 Papers

2.1.1 Sentiment Analysis of Twitter Data for Stock Market Prediction:

Rupawari Jadhav, M.S. Wakode discussed the prediction of future stock prices with the help of sentiment score. The paper presents two different textual representations, Word2vec and N-gram, for analyzing the public sentiments in tweets. The author applied sentiment analysis and supervised machine learning principles (such as logistic regression, random forest, Sequential Minimal Optimization) to tweets extracted from Twitter and analyzing the correlation between stock market movement of company and sentiments in tweets. It proposes a hybrid approach which combines unsupervised learning to cluster the tweets and then performing supervised learning methods for Classification. The author applied different machine learning techniques: Naive Bayes(NB), Maximum entropy, support vector machine (SVM) etc. and concluded that NB, SVM with 89.4% accuracy outperforms the other techniques in sentiment classification. They looked for a correlation between twitter sentiments with stock prices and determined which words in tweets correlate to change in stock price by doing a post analysis of price change and tweets.

2.1.2 Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation

Paul D. Yoo, Maria H. Kim and Tony Jan compared and evaluated some of the existing ML techniques used for stock market prediction. After comparing simple regression,multivariate regression, Neural Networks, Support Vector Machines and Case Based Reasoning models they concluded that Neural Networks offer the ability to predict market directions more accurately as compared to other techniques. Support Vector Machines and Case Based Reasoning are also popular for stock market prediction. In addition, they found that incorporating event information with a prediction model plays a very important role for more accurate prediction. The web provides the latest and latent event information about the stock market which is required to yield higher prediction accuracy and to make predictions in a short time frame.

2.1.3 Stock Price Prediction Using Financial News Articles

M.I. Yasef Kaya and M. Elef Karshgil analysed the correlation between the contents of financial news articles and the stock prices. News articles were labeled positive or negative depending on their effect on the stock market. Instead of using single word as features, they used word couples as features. A word couple consisted of a combination of a noun and a verb. SVM classifier was trained with labeled articles to predict the stock prices.

Chapter 3

Methodology and Implementation

In this chapter we present the basic architecture of our model and complete description about the dataset and the preprocessing steps that were undertaken.

3.1 Basic Structure

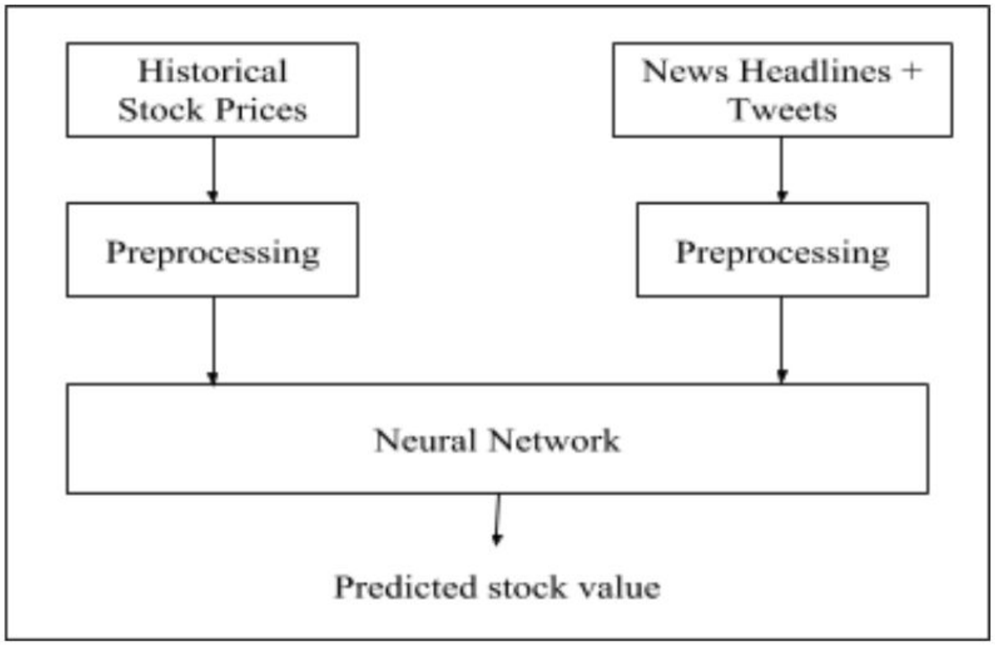


Fig 3.1: Data Flow Diagram

- Historical Stock Prices: This is the time series data which includes the stock prices of the companies. This data had to be pre-processed in order to handle missing values in the time series and remove all invalid data.
- Users Tweets: This includes all the tweets done by users. The tweets were pre processed and converted to multidimensional vectors suitable for input to the neural network.
- Neural Network: The LSTM Neural Network is trained to find the correlation between all the factors and predict the stock price based on the input.

3.2 Requirements

3.2.1 Software Requirements

- Python 3: Python is an interpreted high-level programming language for general-purpose programming.
- Keras: Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.
- Pandas: Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.
- NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- NLTK: The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

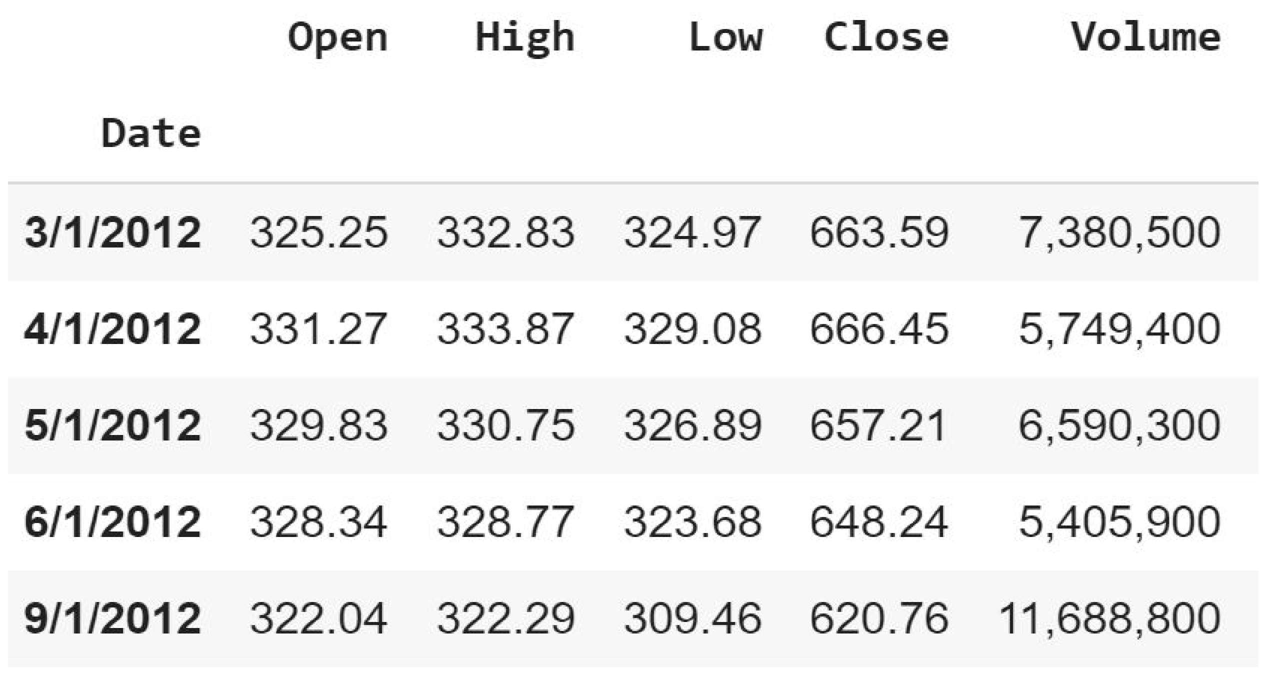
3.3 Dataset

Two different types of datasets have been used in this project for stock predictions

- Historical Stock Data: Daily price data.

  Attributes: date, opening price, closing price, maximum, minimum, shares traded
- Twitter Data: Company specific tweets were collected.

  Attributes: timestamp, username, tweet, retweet count



Date	Open	High	Low	Close	Volume
3/1/2012	325.25	332.83	324.97	663.59	7,380,500
4/1/2012	331.27	333.87	329.08	666.45	5,749,400
5/1/2012	329.83	330.75	326.89	657.21	6,590,300
6/1/2012	328.34	328.77	323.68	648.24	5,405,900
9/1/2012	322.04	322.29	309.46	620.76	11,688,800

fig 3.2: Table Showing the Attributes of Stock dataset

3.4 Preprocessing of Dataset

3.4.1 Preprocessing of Stock Data

Historical Stock Prices needed to be normalized before directly applying the values to the model since the parameters used were of different scales. Therefore Min-max scalar normalization technique was used.

Min-max scalar: This converts the prices in the range 0 to 1.

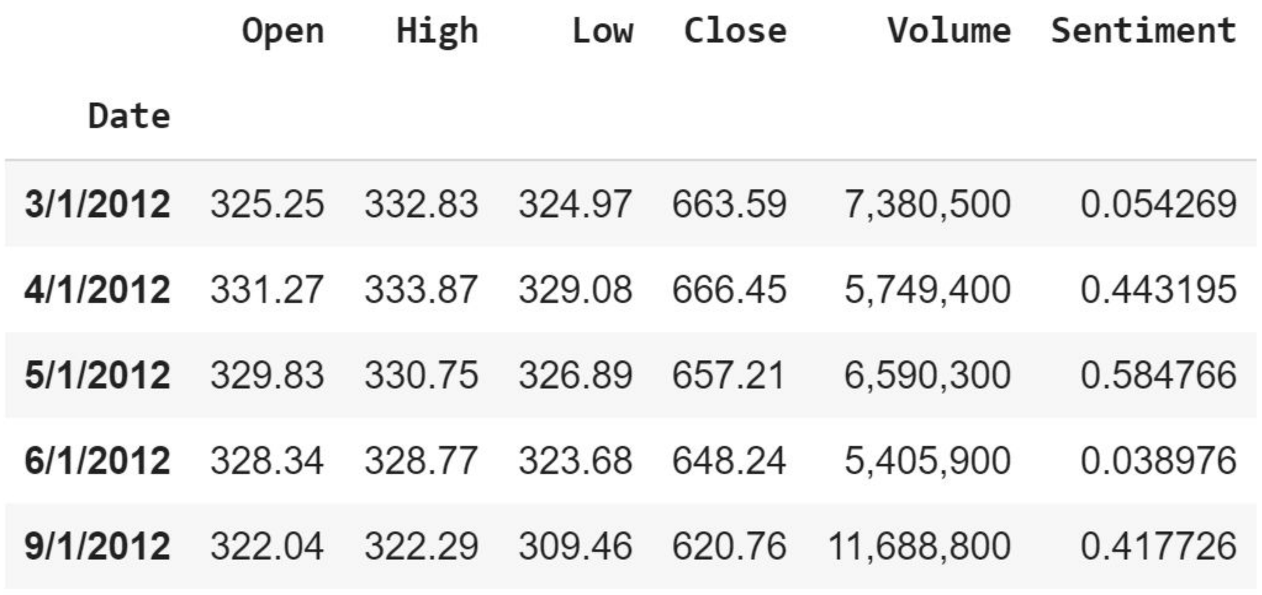
Normalized Value = (Current Value - Min_Value) / (Max_Value - Min_Value)

where Min_value, Max_value are the minimum and maximum value of the data to be normalized.

3.4.2 Preprocessing of Twitter Data

Preprocessing of twitter data was essential to remove the noises such as stopwords, redundant words and punctuations as they are not important for financial analysis.

- Punctuations and characters were removed, contractions were replaced by their original form by using a dictionary (eg. “ain’t” replaced by “am not”, “can’t” replaced by “cannot”).
- Tweets were tokenized and stopwords were removed.



	Open	High	Low	Close	Volume	Sentiment
Date						
3/1/2012	325.25	332.83	324.97	663.59	7,380,500	0.054269
4/1/2012	331.27	333.87	329.08	666.45	5,749,400	0.443195
5/1/2012	329.83	330.75	326.89	657.21	6,590,300	0.584766
6/1/2012	328.34	328.77	323.68	648.24	5,405,900	0.038976
9/1/2012	322.04	322.29	309.46	620.76	11,688,800	0.417726

fig 3.3: Table Showing the Attributes of Stock dataset with Sentiment

3.5 Algorithms and Techniques

Along with the historical prices, financial news, users tweets has some impact on the stock prices. In this model, we considered stock historical prices and users tweets to predict the stock price.

3.5.1 Recurrent Neural Network

Recurrent Neural Network is a generalization of a feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.
Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.

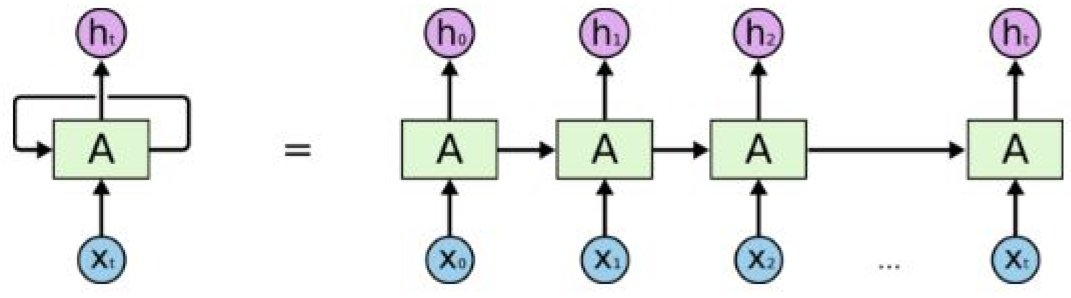


fig 3.4 : An unrolled Recurrent Neural Network

First, it takes the X(0) from the sequence of input and then it outputs h(0) which together with X(1) is the input for the next step. So, the h(0) and X(1) is the input for the next step. Similarly, h(1) from the next is the input with X(2) for the next step and so on. This way, it keeps remembering the context while training.

The formula for the current state is

$$h_t = f(h_{t-1}, x_t)$$

Applying Activation Function:

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

W is weight, h is the single hidden vector, Whh is the weight at previous hidden state, Whx is the weight at current input state, tanh is the Activation function, that implements a Non-linearity that squashes the activations to the range[-1.1]

Output:

$$y_t = W_{hy}h_t$$

Yt is the output state.

3.5.2 Long Short Term Memory Network

LSTM is a form of Recurrent Neural Network (RNN) which is capable of holding long term dependencies. LSTM can remember the information for a long period of time. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.

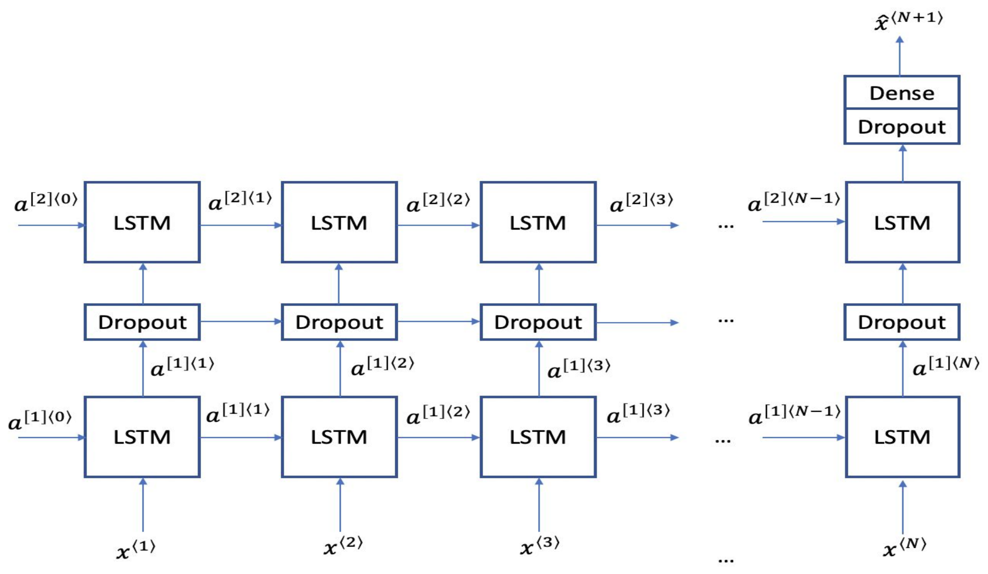

$\hat{x}^{\langle N+1 \rangle}$
Dense
Dropout
$a^{[2]\langle 0 \rangle}$
LSTM
$a^{[2]\langle 1 \rangle}$
LSTM
$a^{[2]\langle 2 \rangle}$
LSTM
$a^{[2]\langle 3 \rangle}$
...
$a^{[2]\langle N-1 \rangle}$
LSTM
Dropout
Dropout
Dropout
...
Dropout
$a^{[1]\langle 1 \rangle}$
$a^{[1]\langle 2 \rangle}$
$a^{[1]\langle 3 \rangle}$
$a^{[1]\langle N \rangle}$
$a^{[1]\langle 0 \rangle}$
LSTM
$a^{[1]\langle 1 \rangle}$
LSTM
$a^{[1]\langle 2 \rangle}$
LSTM
$a^{[1]\langle 3 \rangle}$
...
$a^{[1]\langle N-1 \rangle}$
LSTM
$x^{\langle 1 \rangle}$
$x^{\langle 2 \rangle}$
$x^{\langle 3 \rangle}$
...
$x^{\langle N \rangle}$

fig 3.5 : LSTM Network Architecture

- Cell: It is used to remember the values over arbitrary time intervals.
- Input Gate: It decides which information to keep in the cell.
- Output Gate: It is used to decide which part of cell state should be given as an output.
- Forget Gate: It is used to decide which information to throw away from the cell.

Since there is a trend in a stock market, Neural Network has to consider the previous day's stock prices. Simple Neural Network is not able to remember the previous values. Thus, Long Short Term Memory (LSTM) Network should be used in predicting the stock prices as it considers previous records as well.

3.5.3 Sentiment Analysis

Sentiment analysis is the automated process of identifying and extracting the subjective information that underlies a text. This can be either an opinion, a judgment, or a feeling about a particular topic or subject. The most common type of sentiment analysis is called 'polarity detection' and consists in classifying a statement as 'positive', 'negative' or 'neutral'. It is also known as Opinion Mining.
A sub-field of Natural Language Processing (NLP), sentiment analysis has been getting a lot of attention in recent years due to its many exciting applications in a variety of fields, ranging from

business to political studies. Sentiment analysis is particularly useful for social media monitoring because it goes beyond metrics that focus on the number of likes or retweets, and provides a qualitative point of view. Stock prices also depend on public sentiments, due to these reasons we are going to use Sentiment analysis in our project.

3.6 Proposed Model

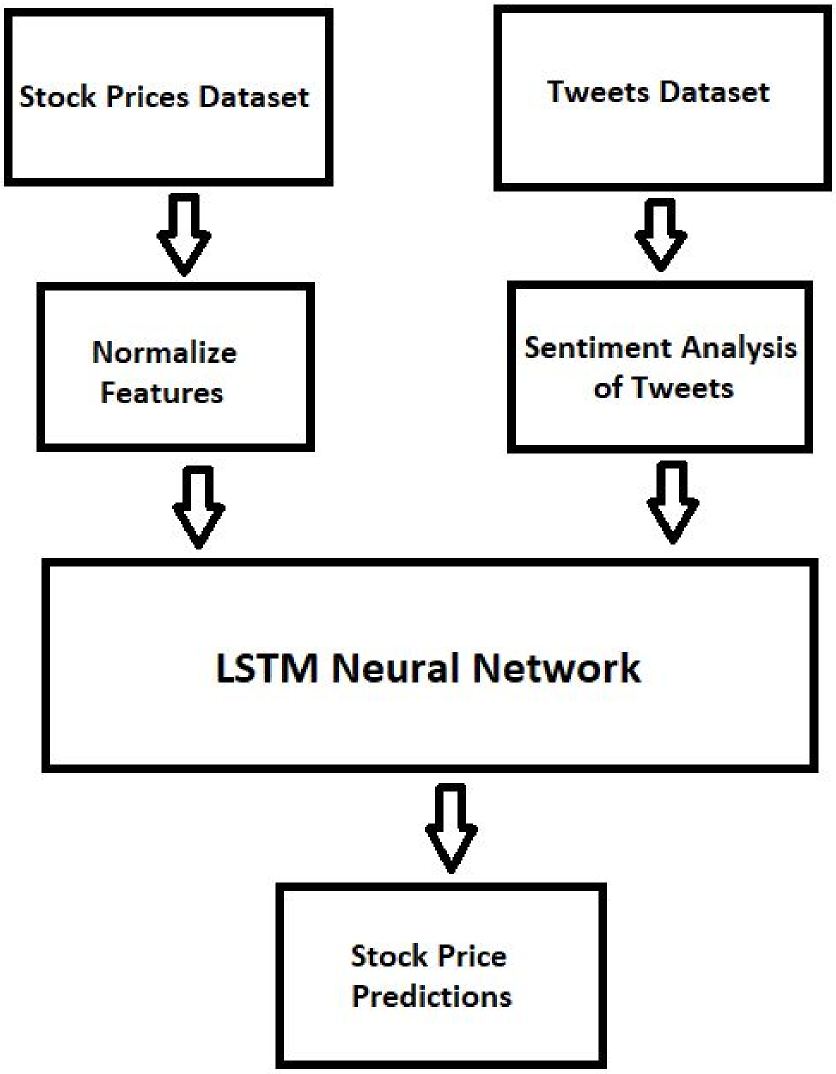


fig 3.6 : Architecture of the Proposed Model

First we will normalize the stock price dataset and also calculate the sentiment of tweets. After this we will give both of these data to LSTM Neural Network which will create our proposed Model. Now using this Model we can predict the required stock prices.

Chapter 4

Results

4.1 Prediction of Stock without Sentiment Analysis

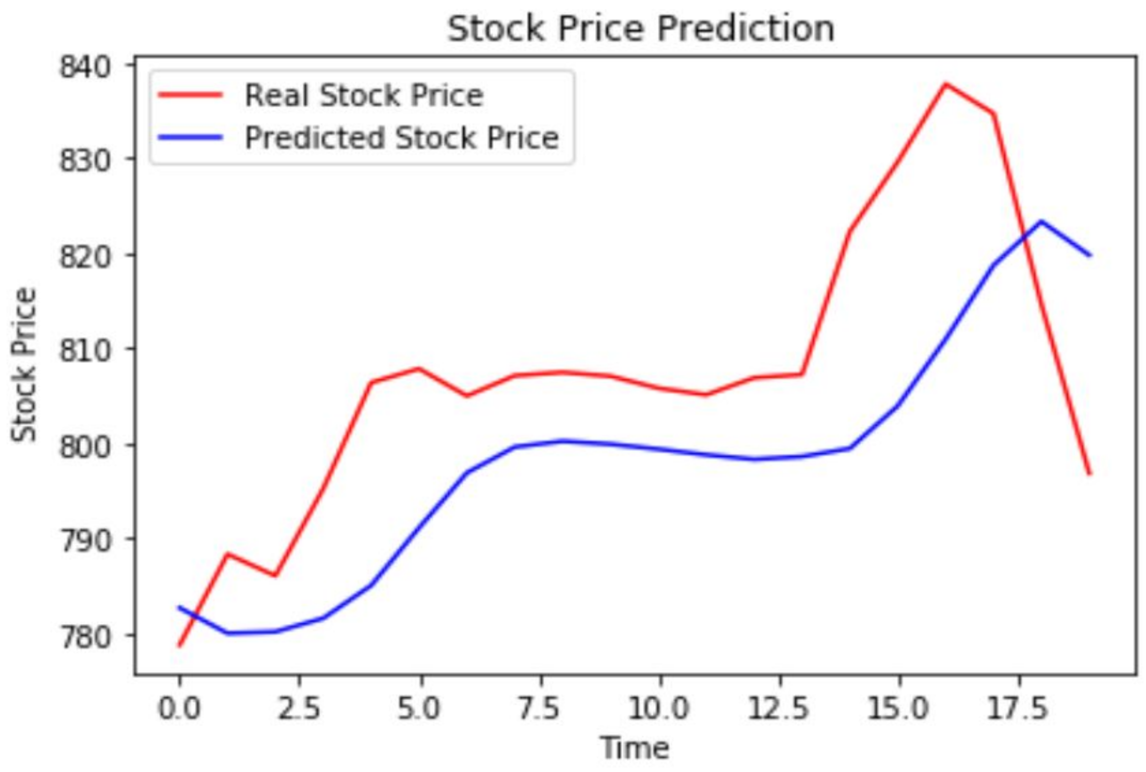
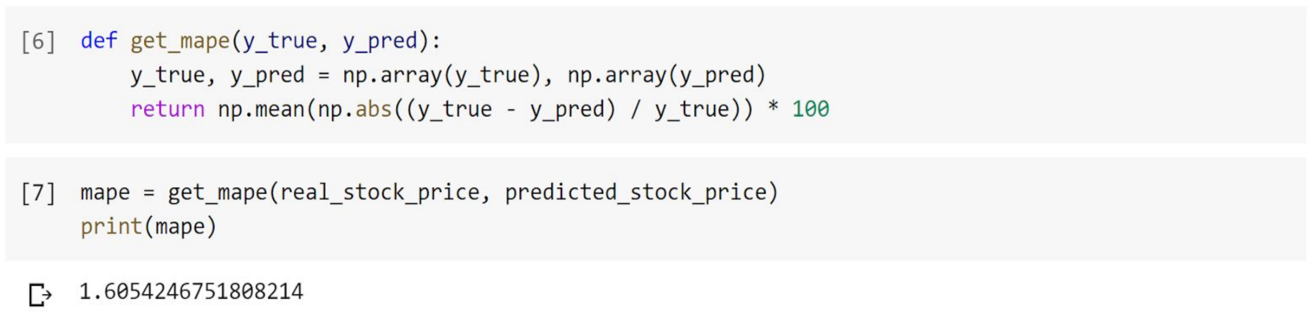


fig 4.1 : Prediction without Sentiment Analysis

4.1.1 Calculation of Mean Absolute Percentage Error (MAPE)



```
[6]  def get_mape(y_true, y_pred):
         y_true, y_pred = np.array(y_true), np.array(y_pred)
         return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

```
[7]  mape = get_mape(real_stock_price, predicted_stock_price)
     print(mape)
```

```
1.6054246751808214
```

4.2 Prediction of Stock with Sentiment Analysis

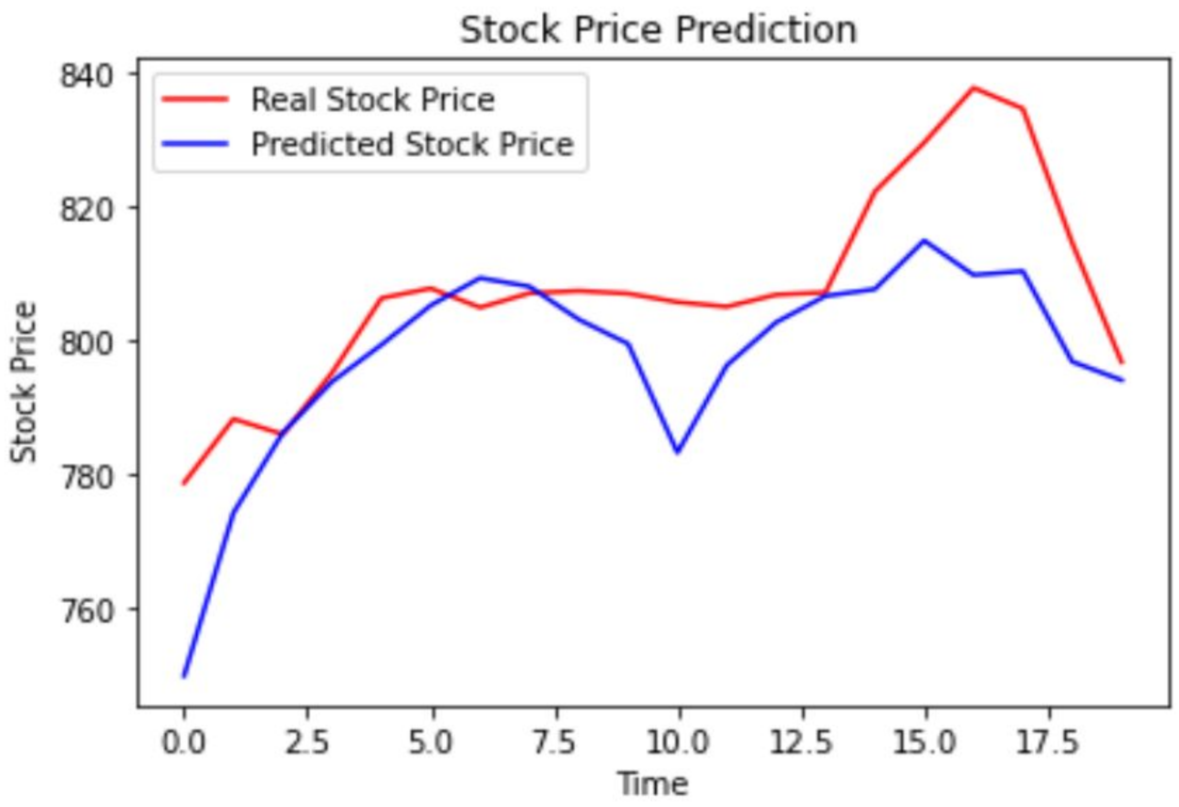
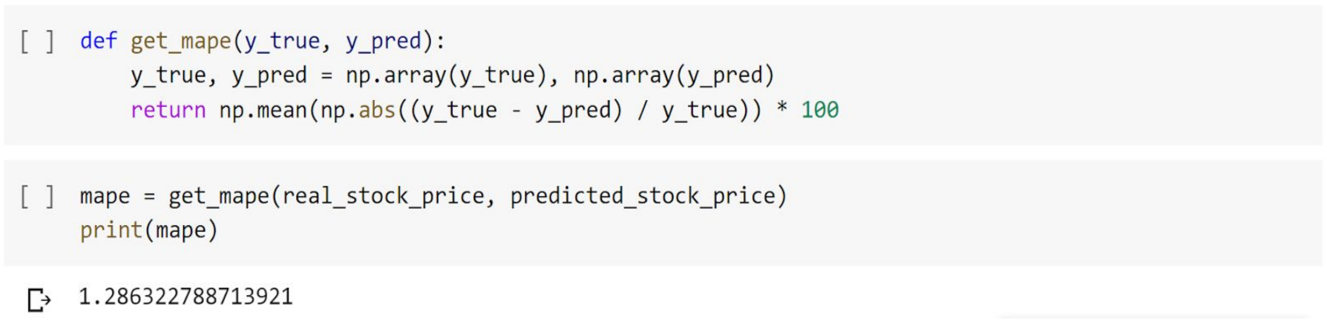


fig 4.2 : Prediction with Sentiment Analysis

4.2.1 Calculation of Mean Absolute Percentage Error (MAPE)



```
[ ]  def get_mape(y_true, y_pred):
         y_true, y_pred = np.array(y_true), np.array(y_pred)
         return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

```
[ ]  mape = get_mape(real_stock_price, predicted_stock_price)
     print(mape)
```

```
1.286322788713921
```

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

The project objective of adding Twitter data to increase the prediction accuracy along with historical stock prices did give us better results compared to only historical stock prices. Along with this, using LSTM helped in mapping the events better. However, addition of Twitter data to the model didn’t have a significant effect on the prediction value. This may be due to the smaller dataset. Collecting more streaming tweets with the paid APIs and working with them might help in achieving better accuracy.

5.2 Future Scope

The future scope of the project would involve incorporating more stocks from different domains and finding out the correlation among them to predict the trend for an entire sector. Also, how much does one sector depend on another and incorporating that into our model. Making use of Twitter data in a much better way with a large corpus. Looking out for some more features which can help us in getting more closer to the actual values. We can also modify the system to an online learning system that adapts in real time.

Project Code

#Code for LSTM

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the training set
dataset_train = pd.read_csv('Stock_Price_Train.csv')
training_set = dataset_train.iloc[:, 1:2].values

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

# Creating a data structure with 60 timesteps and 1 output
X_train = []
y_train = []
for i in range(60, 1258):
        X_train.append(training_set_scaled[i-60: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))

# Building the RNN
# 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

# Initialising the RNN
regressor = Sequential()

# Adding the LSTM and Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
# Adding a third LSTM layer and  Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
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')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)

# Making the predictions and visualising the results

# Getting the real stock price of 2017
dataset_test = pd.read_csv('Stock_Price_Test.csv')
real_stock_price = dataset_test.iloc[:, 1:2].values

# Getting the predicted stock price of 2017
dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(60, 80):
        X_test.append(inputs[i-60: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 = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)

# Visualising the results
plt.plot(real_stock_price, color = 'red', label = 'Real Stock Price')
plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted Stock Price')
plt.title(' Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel(' Stock Price')
plt.legend()
plt.show()
```

#Code for Sentiment Analysis

```
import re
import csv
import nltk
import pandas as pd

def processTweet(tweet):
        tweet = tweet.lower()
        tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
        tweet = re.sub('@[^\s]+','AT_USER',tweet)
        tweet = re.sub('[\s]+', ' ', tweet)
        tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
        tweet = tweet.strip('\'"')
        return tweet

def replaceTwoOrMore(s):
        pattern = re.compile(r"(.)\1{1,}", re.DOTALL)
        return pattern.sub(r"\1\1", s)

def getStopWordList(stopWordListFileName):
```

```
            stopWords = []
            stopWords.append('AT_USER')
            stopWords.append('URL')

            fp = open(stopWordListFileName, 'r')
            line = fp.readline()
            while line:
            word = line.strip()
            stopWords.append(word)
            line = fp.readline()
            fp.close()
            return stopWords

def getFeatureVector(tweet):
            featureVector = []
            words = tweet.split()
            for w in words:
            w = replaceTwoOrMore(w)
            w = w.strip('\'"?,.')
            val = re.search(r"^[a-zA-Z][a-zA-Z0-9]*$", w)
            if(w in stopWords or val is None):
            continue
            else:
            featureVector.append(w.lower())
            return featureVector

data=pd.read_csv('file:///C:/Users/Desktop/btp/neg.csv')
data1=pd.read_csv('file:///C:/Users/Desktop/btp/pos.csv')
data2=pd.read_csv('file:///C:/Users/Desktop/btp/test.csv')

headers=["A","B","C","D","E","F"]
headers1=["P","Q","R","S","T","U"]
headers2=["G","H","I","J","K","L"]
data.columns=headers
data1.columns=headers1
data2.columns=headers2
data.sample()

data1.sample()

data2.sample()

tweets = []
featureList = []
st = open('stopWords.txt', 'r')
stopWords = getStopWordList('stopWords.txt')
for index,row in data.iterrows():
            sentiment = row.A
            tweet = row.F
            processedTweet = processTweet(tweet)
            featureVector = getFeatureVector(processedTweet)
            featureList.extend(featureVector)
            tweets.append((featureVector, sentiment))

for index,row in data1.iterrows():
            sentiment = row.P
```

```
        tweet = row.U
        processedTweet = processTweet(tweet)
        featureVector = getFeatureVector(processedTweet)
        featureList.extend(featureVector)
        tweets.append((featureVector, sentiment))

test = []
for index,row in data2.iterrows():
        sentiment2 = row.G
        tweet2 = row.L
        processedTweet2 = processTweet(tweet2)
        featureVector2 = getFeatureVector(processedTweet2)
        test.append((featureVector2, sentiment2))

featureList = list(set(featureList))

def extract_features(tweet):
        tweet_words = set(tweet)
        features = {}
        for word in featureList:
        features['contains(%s)' % word] = (word in tweet_words)
        return features

training_set = nltk.classify.util.apply_features(extract_features, tweets)
test_set = nltk.classify.util.apply_features(extract_features, test)

NBClassifier = nltk.NaiveBayesClassifier.train(training_set)

print(nltk.classify.accuracy(NBClassifier, test_set))
print(NBClassifier.show_most_informative_features(8))
```

References

[1] Combination of Convolution and Recurrent Neural Networks for Sentiment Analysis of Short Texts https://www.aclweb.org/anthology/C/C16/C16-1229.pdf

[2] Forecasting of Stock Market Indices Using Artificial Neural Network http://irjcjournals.org/ijieasr/Dec2015/2.pdf

[3] Machine Learning Techniques and Use of Event Information for Stock Market Prediction : A Survey and Evaluation http://ieeexplore.ieee.org/abstract/document/1631572/

[4] Prediction Model of the Stock Market Index Using Twitter Sentiment Analysis http://www.mecs-press.org/ijitcs/ijitcs-v8-n10/IJITCS-V8-N10-2.pdf

[5] Sentiment analysis of Twitter data for predicting stock market and movements http://ieeexplore.ieee.org/document/7955659/

[6] Sentiment Analysis of Twitter Data for Stock Market Prediction https://www.ijarcce.com/upload/2017/march-17/IJARCCE%20129.pdf

[7] Stock market prediction using ARIMA model http://ijssst.info/Vol-15/No-4/data/4923a105.pdf