STOCK PRICE PREDICTION

A

PROJECT REPORT

By

Vishal Droch(2017509)

Manoj Soni(2017504)

Under the guidance of

Mrs. Mehak Gambhir

AP CSE Dept.

CEC CHANDIGARH ENGINEERING COLLEGE

Building Careers. Transforming Lives.

DEPARTMENT OF

COMPUTER SCIENCE ENGINEERING

CHANDIGARH ENGINEERING COLLEGE,

LANDRAN(MOHALI)

2022 - 2023

ACKNOWLEDGEMENT

I have immense pleasure to present this seminar on **STOCK PRICE PREDICTION**, a topic of my personal interest. Firstly, I thank ‘God’, the almighty for giving me such a great opportunity to present this seminar.

I sincerely express my thanks to **Mrs. Mehak Gambhir** for the approval and guidance given.

Lastly, I sincerely express my gratitude to other teachers and my dear friends for their valuable co-operation and help.

PROJECT STUDENTS

VISHAL DROCH	- 2017509
MANOJ SONI	-2017504

CHANDIGARH ENGINEERING COLLEGE, LANDRAN(MOHALI)

CERTIFICATE

*This is to certify that the project report entitled **“STOCK PRICE PREDICTION”** submitted by **Vishal Droch(2017509)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Science in Artificial Intelligence and Machine Learning of Chandigarh Engineering College, Landran Mohali is a record of bonafide work carried out under my guidance and supervision.*

Seminar Coordinator
Mrs. Mehak Gambhir
ASSISTANT PROFESSOR
Department of CSE

Head of the Department
Dr. Shukhpreet Kaur
Department of CSE
CEC

DECLARATION

We, **VISHAL DROCH, MANOJ SONI** of fifth semester BSc AI& ML., in the department of Computer Science and Engineering from CEC Landran, hereby declare that the project work entitled **STOCK PRICE PREDICTION** is carried out by us and submitted in partial fulfillment of the requirements for the award of **Bachelor of Technology in Computer Science Engineering** , under **Mrs. Mehak Gambhir** Department of Computer Science Engineering during the academic year 2020-2023 and has not been submitted to any other university for the award of any kind of degree.

Vishal Droch	-	2017509
Manoj Soni	-	2017504

ABSTRACT

In this project we attempt to implement machine learning approach
to predict stock prices. Machine learning is effectively implemented in forecasting stock prices. The objective is to predict the stock prices in order to make more informed and accurate investment decisions. We propose a stock price prediction system that integrates mathematical functions, machine learning, and other external factors for the purpose of achieving better stock prediction accuracy and issuing profitable trades.
There are two types of stocks. You may know of intraday trading by
the commonly used term "day trading." Interday traders hold securities positions from at least one day to the next and often for several days to weeks or months. LSTMs are very powerful in sequence prediction problems because they’re able to store past information. This is important in our case because the previous price of a stock is crucial in predicting its future price. While predicting the actual price of a stock is an uphill climb, we can build a model that will predict whether the price will go up or down.

Keywords: LSTM, CNN, ML, DL, Trade Open, Trade Close, Trade Low, Trade High

CONTENTS

CHAPTER 1 INTRODUCTION

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities and derivatives over virtual platforms supported by brokers. The stock market allows investors to own shares of public companies through trading either by exchange or over the counter markets. This market has given investors the chance of gaining money and having a prosperous life through investing small initial amounts of money, low risk compared to the risk of opening new business or the need of high salary career. Stock markets are affected by many factors causing the uncertainty and high volatility in the market. Although humans can take orders and submit them to the market, automated trading systems (ATS) that are operated by the implementation of computer programs can perform better and with higher momentum in submitting orders than any human. However, to evaluate and control the performance of ATSs, the implementation of risk strategies and safety measures applied based on human judgements are required. Many factors are incorporated and considered when developing an ATS, for instance, trading strategy to be adopted, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of the future stock value, and specific news related to the stock being analysed.

- Long-short Term Memory (LSTM) and **Gated Recurrent Unit (GRU).** Stock market is a typical area that presents time-series data and many researchers study on it and proposed various models. In this project, LSTM model is used to predict the stock price.

1.1MOTIVATION FOR WORK

Businesses primarily run over customer's satisfaction, customer reviews about their products. Shifts in sentiment on social media have been shown to correlate with shifts in stock markets. Identifying customer grievances thereby resolving them leads to customer satisfaction as well as trustworthiness of an organization. Hence there is a necessity of an un biased automated system to classify customer reviews regarding any problem. In today's environment where we're justifiably suffering from data overload (although this does not mean better or deeper insights), companies might have mountains of customer feedback collected; but for mere humans, it's still impossible to analyse it manually without any sort of error or bias. Oftentimes, companies with the best intentions find themselves in an insights vacuum. You know you need insights to inform your decision making and you know that you're lacking them, but don't know how best to get them. Sentiment analysis provides some answers into what the most important issues are, from the perspective of customers, at least. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition.

1.2PROBLEM STATEMENT

Time Series forecasting & modelling plays an important role in data analysis. Time series analysis is a specialized branch of statistics used extensively in fields such as Econometrics & Operation Research. Time Series is being widely used in analytics & data science. Stock prices are volatile in nature and price depends on various factors. The main aim of this project is to predict stock prices using Long short term memory (LSTM).

CHAPTER 2 METHODOLOGY

2.1 PROPOSED SYSTEMS

The prediction methods can be roughly divided into two categories, statistical methods and artificial intelligence methods. Statistical methods include logistic regression model, ARCH model, etc. Artificial intelligence methods include multi-layer perceptron, convolutional neural network, naive Bayes network, back propagation network, single-layer LSTM, support vector machine, recurrent neural network, etc. They used Long short-term memory network (LSTM).

Long short-term memory network:

Long short-term memory network (LSTM) is a particular form of recurrent neural network (RNN).

Working of LSTM:

LSTM is a special network structure with three "gate" structures. Three gates are placed in an LSTM unit, called input gate, forgetting gate and output gate. While information enters the LSTM's network, it can be selected by rules. Only the information conforms to the algorithm will be left, and the information that does not conform will be forgotten through the forgetting gate.

The experimental data in this paper are the actual historical data downloaded from the Internet. Three data sets were used in the experiments. It is needed to find an optimization algorithm that requires less resources and has faster convergence speed.

•Used Long Short-term Memory (LSTM) with embedded layer and the LSTM neural network with automatic encoder.

•LSTM is used instead of RNN to avoid exploding and vanishing gradients.

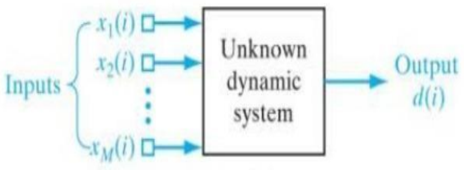
•In this project python is used to train the model, MATLAB is used to reduce dimensions of the input. MySQL is used as a dataset to store and retrieve data.

•The historical stock data table contains the information of opening price, the highest price, lowest price, closing price, transaction date, volume and so on.

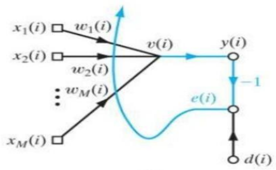
•The accuracy of this LSTM model used in this project is 57%.

LMS filter:

The LMS filter is a kind of adaptive filter that is used for solving linear problems. The idea of the filter is to minimize a system (finding the filter coefficients) by minimizing the least mean square of the error signal.



(a)



(b)

Fig. 1: LMS Inputs and Outputs

Fig 2: LMS updating weights

LSTM Architecture

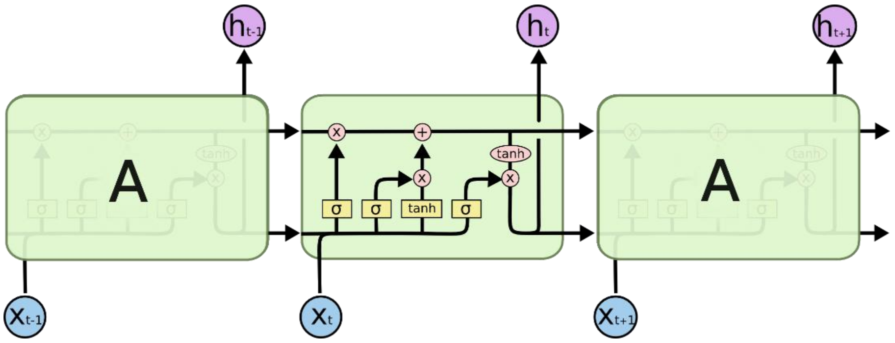


Fig. 3: LSTM Architecture

Forget Gate:

A forget gate is responsible for removing information from the cell state.

•The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter.

•This is required for optimizing the performance of the LSTM network.

•This gate takes in two inputs; h_t-1 and x_t. h_t-1 is the hidden state from the previous cell or the output of the previous cell and x_t is the input at that particular time step.

Input Gate:

1.Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from hi-1 and x_t.

2.Creating a vector containing all possible values that can be added (as perceived from h_t-1 and x_t) to the cell state. This is done using the tanh function, which outputs values from -1 to +1.

3.Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

The functioning of an output gate can again be broken down to three steps:

•Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.

•Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.

•Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

Hardware Requirements:

RAM: 4 GB

Storage: 500 GB

CPU: 2 GHz or faster

Architecture: 32-bit or 64-bit

Software Requirements:

- Python 3.5 in Google Colab is used for data pre-processing, model training and prediction.
- Operating System: windows 7 and above or Linux based OS or MAC OS

Functional requirements:

Functional requirements describe what the software should do (the functions). Think about the core operations.

Because the “functions” are established before development, functional requirements should be written in the future tense. In developing the software for Stock Price Prediction, some of the functional requirements could include:

The software shall accept the tw_spydata_raw.csv dataset as input.

The software should shall do pre-processing (like verifying for missing data values) on input for model training.

The software shall use LSTM ARCHITECTURE as main component of the software.

It processes the given input data by producing the most possible outcomes of a CLOSING STOCK PRICE.

Notice that each requirement is directly related to what we expect the software to do. They represent some of the core functions.

Non-Functional requirements

Product properties

Usability: It defines the user interface of the software in terms of simplicity of understanding the user interface of stock prediction software, for any kind of stock trader and other stakeholders in stock market.

Efficiency: maintaining the possible highest accuracy in the closing stock prices in shortest time with available data.

Performance: It is a quality attribute of the stock prediction software that describes the responsiveness to various user interactions with it.

2.1.1 SYSTEM ARCHITECTURE

1. Preprocessing of data

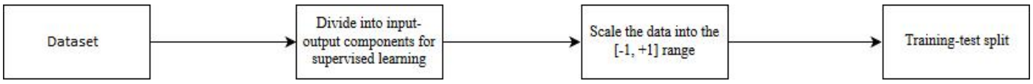

Dataset
Divide into input-output components for supervised learning
Scale the data into the [-1, +1] range
Training-test split

Fig. 4: Pre-processing of data

2. Overall Architecture

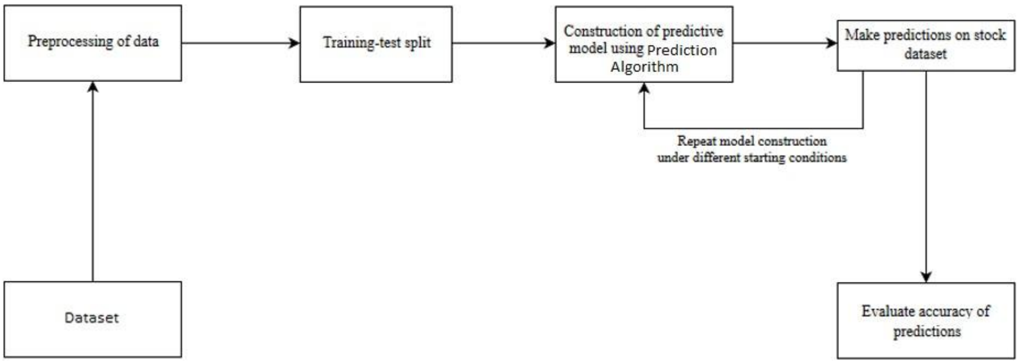

Preprocessing of data
Training-test split
Construction of predictive model using Prediction Algorithm
Make predictions on stock dataset
Repeat model construction under different starting conditions
Dataset
Evaluate accuracy of predictions

Fig. 5: Overall Architecture

CHAPTER 5 EXPERIMENT ANALYSIS

4.1system configuration: This project can run on commodity hardware. We ran entire project on an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, It also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

4.1.1Hardware Requirements:

RAM: 4 GB

Storage: 500 GB

CPU: 2 GHz or faster

Architecture: 32-bit or 64-bit

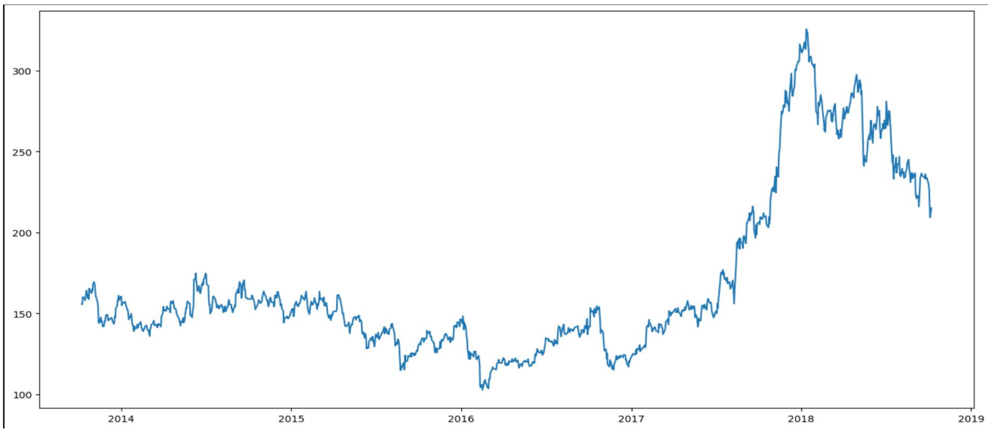
4.1.2 Software requirements: Python 3.5 in Google Colab is used for data pre-processing, model training and prediction.

Operating System: windows 7 and above or Linux based OS or MAC OS.

4.2 Sample Code:

```
#imports library
import pandas as pd
import numpy as np


import matplotlib.pyplot as plt
%matplotlib inline


from matplotlib.pylab import rcParams
rcParams['figure.figsize']=20,10


from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))


#read the dataset
df=pd.read_csv("/content/NSE_Tata.csv")
df.head()


#analyze the closing prices from dataframe
df["Date"]=pd.to_datetime(df.Date,format="%Y-%m-%d")
```

```
df.index=df['Date']

plt.figure(figsize=(16,8))
plt.plot(df["Close"],label='Close Price history')
```



MODEL CREATION:

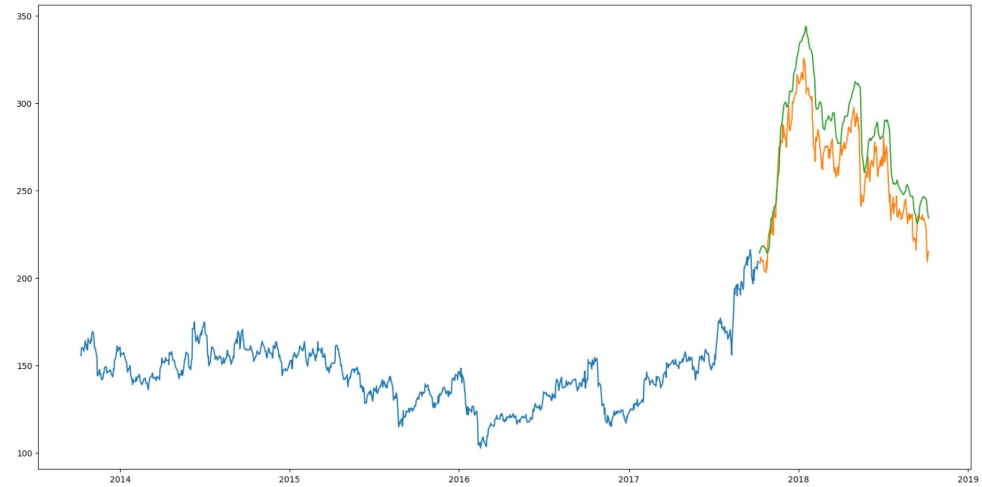
```
from keras.models import Sequential
from keras.layers import LSTM,Dropout,Dense

#Sort the dataset on date time and filter “Date” and “Close” columns
data=df.sort_index(ascending=True,axis=0)
new_dataset=pd.DataFrame(index=range(0,len(df)),columns=['Date','Close'])

for i in range(0,len(data)):
    new_dataset["Date"][i]=data['Date'][i]
    new_dataset["Close"][i]=data["Close"][i]



new_dataset.index=new_dataset.Date
new_dataset.drop("Date",axis=1,inplace=True)

final_dataset=new_dataset.values

train_data=final_dataset[0:987,:]
valid_data=final_dataset[987:,:]
```

```
scaler=MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(final_dataset)

x_train_data,y_train_data=[],[]

for i in range(60,len(train_data)):
    x_train_data.append(scaled_data[i-60:i,0])
    y_train_data.append(scaled_data[i,0])

x_train_data,y_train_data=np.array(x_train_data),np.array(y_train_data)

x_train_data=np.reshape(x_train_data,(x_train_data.shape[0],x_train_data.shape[1],1))
```

LSTM:

```
lstm_model=Sequential()
lstm_model.add(LSTM(units=50,return_sequences=True,input_shape=(x_train_data.sha
pe[1],1)))
lstm_model.add(LSTM(units=50))
lstm_model.add(Dense(1))
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
inputs_data=new_dataset[len(new_dataset)-len(valid_data)-60:].values
inputs_data=inputs_data.reshape(-1,1)
inputs_data=scaler.transform(inputs_data)
inputs_data=new_dataset[len(new_dataset)-len(valid_data)-60:].values
inputs_data=inputs_data.reshape(-1,1)
inputs_data=scaler.transform(inputs_data)
lstm_model.save("saved_lstm_model.h5")
train_data=new_dataset[:987]
valid_data=new_dataset[987:]
valid_data['Predictions']=closing_price
plt.plot(train_data["Close"])
plt.plot(valid_data[['Close',"Predictions"]])
```

OUTPUTS FROM THIS CODE:



Front End Pages:

```
import dash

import dash_core_components as dcc

import dash_html_components as html

import pandas as pd

import plotly.graph_objs as go

from dash.dependencies import Input, Output

from keras.models import load_model

from sklearn.preprocessing import MinMaxScaler

import numpy as np


app = dash.Dash()

server = app.server


scaler=MinMaxScaler(feature_range=(0,1))


df_nse = pd.read_csv("/content/NSE_Tata.csv")


df_nse["Date"]=pd.to_datetime(df_nse.Date,format="%Y-%m-%d")

df_nse.index=df_nse['Date']


data=df_nse.sort_index(ascending=True,axis=0)

new_data=pd.DataFrame(index=range(0,len(df_nse)),columns=['Date','Close'])
```

```
for i in range(0,len(data)):
    new_data["Date"][i]=data['Date'][i]
    new_data["Close"][i]=data["Close"][i]

new_data.index=new_data.Date
new_data.drop("Date",axis=1,inplace=True)

dataset=new_data.values

train=dataset[0:987,:]
valid=dataset[987:,:]

scaler=MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(dataset)

x_train,y_train=[],[]

for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])

x_train,y_train=np.array(x_train),np.array(y_train)

x_train=np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))

lstm_model=load_model("saved_lstm_model.h5")

inputs=new_data[len(new_data)-len(valid)-60:].values
inputs=inputs.reshape(-1,1)
inputs=scaler.transform(inputs)

X_test=[]
```

```
for i in range(60,inputs.shape[0]):
    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))
closing_price=lstm_model.predict(X_test)
closing_price=scaler.inverse_transform(closing_price)

train=new_data[:987]
valid=new_data[987:]
valid['Predictions']=closing_price
```

Building Dashboard:

```
df= pd.read_csv("/content/nse_sensex.csv")

app.layout = html.Div([

    html.H1("Stock Price Analysis Dashboard", style={"textAlign": "center"}),

    dcc.Tabs(id="tabs", children=[

        dcc.Tab(label='NSE-TATAGLOBAL Stock Data',children=[
                                     html.Div([
                                                 html.H2("Actual closing price",style={"textAlign": "center"}),
                                                 dcc.Graph(
                                                             id="Actual Data",
                                                             figure={
                                                                         "data":[
                                                                                     go.Scatter(
                                                                                                 x=train.index,
                                                                                                 y=valid["Close"],
                                                                                                 mode='markers'
                                                                                     )
```

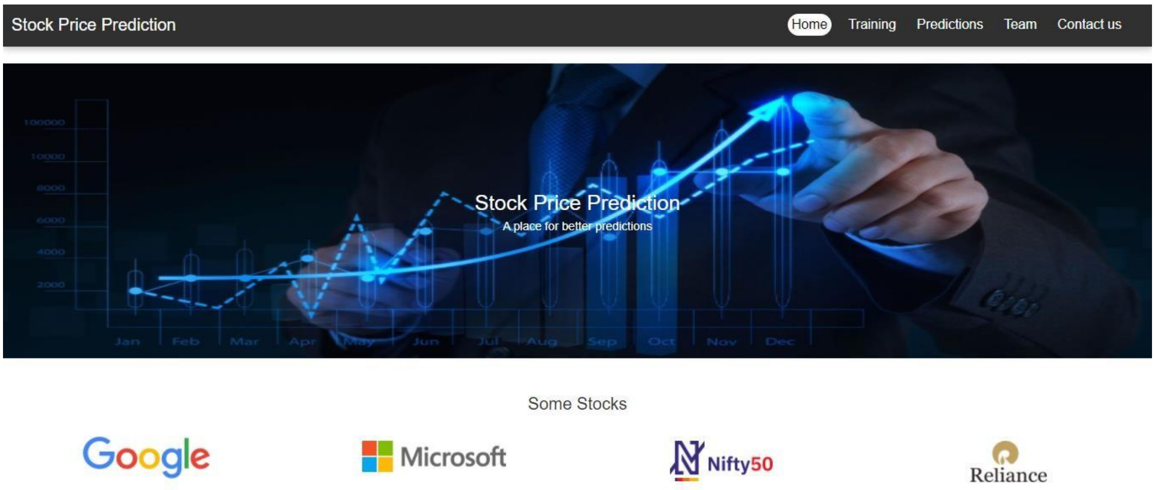
```
            ],
            "layout":go.Layout(
                title='scatter plot',
                xaxis={'title':'Date'},
                yaxis={'title':'Closing Rate'}
            )
        }

    ),
    html.H2("LSTM Predicted closing price",style={"textAlign": "center"}),
    dcc.Graph(
        id="Predicted Data",
        figure={
            "data":[
                go.Scatter(
                    x=valid.index,
                    y=valid["Predictions"],
                    mode='markers'
                )

            ],
            "layout":go.Layout(
                title='scatter plot',
                xaxis={'title':'Date'},
                yaxis={'title':'Closing Rate'}
            )
        }

    )
])
```

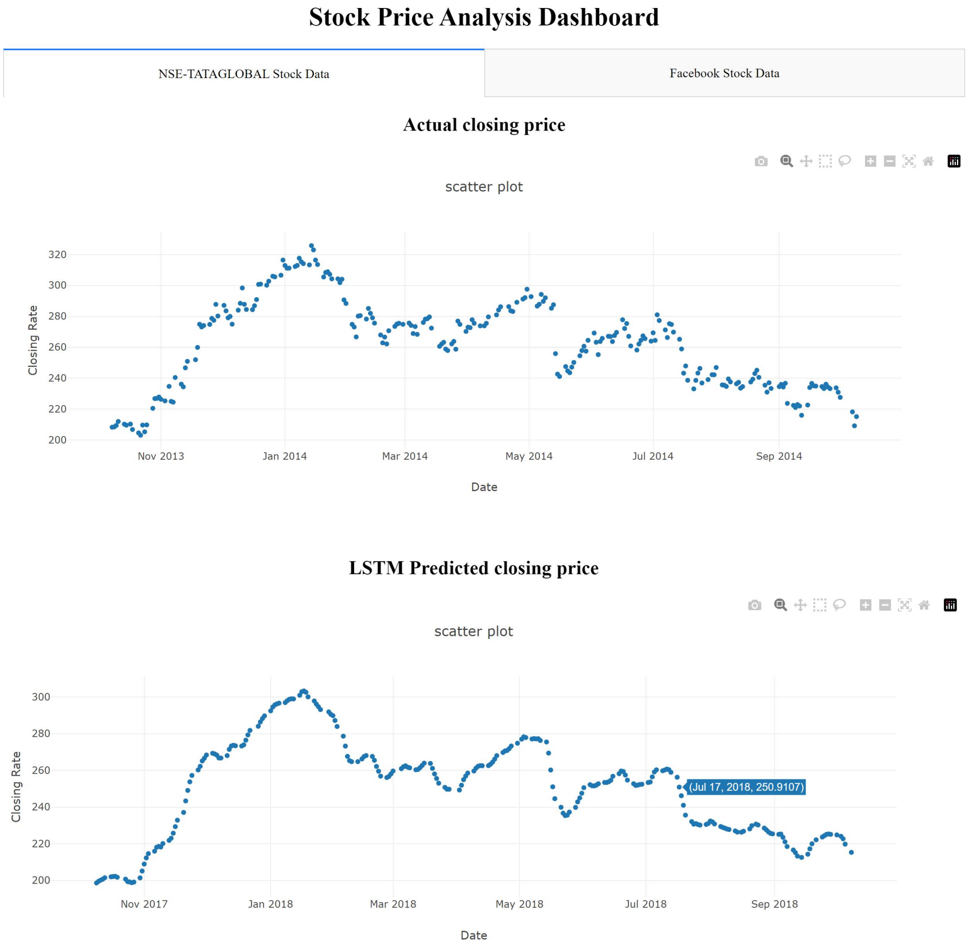
```
]),
dcc.Tab(label='Sensex Stock Data', children=[
   html.Div([
      html.H1("Stocks High vs Lows",
            style={'textAlign': 'center'}),

      dcc.Dropdown(id='my-dropdown',
               options=[{'label': 'ASP', 'value': 'ASIANPAINT'},
                     {'label': 'Axis','value': 'AXISBANK'},
                     {'label': 'Bajaj', 'value': 'BAJAJ-AUTO'},
                     {'label': 'BjF','value': 'BAJFINANCE'},
                     {'label': 'Airtel','value': 'BHARTIARTL'},
                     {'label': 'Hcl','value': 'HCLTECH'},
                     {'label': 'Hdfc','value': 'HDFC'},
                     {'label': 'Hfbank', 'value': 'HDFCBANK'},
                     {'label': 'Hero','value': 'HEROMOTOCO'},
                     {'label': 'unilevar','value': 'HINDUNILVR'},
                     {'label': 'icici','value': 'ICICIBANK'},
                     {'label': 'indusbank','value': 'INDUSINDBK'},
                     {'label': 'infosys','value': 'INFY'},
                     {'label': 'itc','value': 'ITC'},
                     {'label': 'kotak','value': 'KOTAKBANK'},
                     {'label': 'lt','value': 'LT'},
                     {'label': 'mahindra','value': 'M&M'},
                     {'label': 'maruti','value': 'MARUTI'},
                     {'label': 'nestle','value': 'NESTLEIND'},
                     {'label': 'ntpc','value': 'NTPC'},
                     {'label': 'ongc','value': 'ONGC'},
                     {'label': 'powergrid','value': 'POWERGRID'},
                     {'label': 'reliance','value': 'RELIANCE'},
                     {'label': 'sbi','value': 'SBIN'},
                     {'label': 'sunpharma','value': 'SUNPHARMA'},
              {'label': 'tatasteel','value': 'TATASTEEL'},
              {'label': 'tcs','value': 'TCS'},
              {'label': 'techm','value': 'TECHM'},
              {'label': 'titan','value': 'TITAN'},
              {'label': 'ultratech','value': 'ULTRACEMCO'}],
        multi=True,value=['AXISBANK'],
        style={"display": "block", "margin-left": "auto",
             "margin-right": "auto", "width": "60%"}),
dcc.Graph(id='highlow'),
html.H1("Stocks Market Volume", style={'textAlign': 'center'}),

dcc.Dropdown(id='my-dropdown2',
        options=[{'label': 'ASP', 'value': 'ASIANPAINT'},
              {'label': 'Axis','value': 'AXISBANK'},
              {'label': 'Bajaj', 'value': 'BAJAJ-AUTO'},
              {'label': 'BjF','value': 'BAJFINANCE'},
              {'label': 'Airtel','value': 'BHARTIARTL'},
              {'label': 'Hcl','value': 'HCLTECH'},
              {'label': 'Hdfc','value': 'HDFC'},
              {'label': 'Hfbank', 'value': 'HDFCBANK'},
              {'label': 'Hero','value': 'HEROMOTOCO'},
              {'label': 'unilevar','value': 'HINDUNILVR'},
              {'label': 'icici','value': 'ICICIBANK'},
              {'label': 'indusbank','value': 'INDUSINDBK'},
              {'label': 'infosys','value': 'INFY'},
              {'label': 'itc','value': 'ITC'},
              {'label': 'kotak','value': 'KOTAKBANK'},
              {'label': 'lt','value': 'LT'},
              {'label': 'mahindra','value': 'M&M'},
              {'label': 'maruti','value': 'MARUTI'},
              {'label': 'nestle','value': 'NESTLEIND'},
              {'label': 'ntpc','value': 'NTPC'},
              {'label': 'ongc','value': 'ONGC'},
                                        {'label': 'powergrid','value': 'POWERGRID'},
                                        {'label': 'reliance','value': 'RELIANCE'},
                                        {'label': 'sbi','value': 'SBIN'},
                                        {'label': 'sunpharma','value': 'SUNPHARMA'},
                                        {'label': 'tatasteel','value': 'TATASTEEL'},
                                        {'label': 'tcs','value': 'TCS'},
                                        {'label': 'techm','value': 'TECHM'},
                                        {'label': 'titan','value': 'TITAN'},
                                        {'label': 'ultratech','value': 'ULTRACEMCO'}],
                                  multi=True,value=['AXISBANK'],
                                  style={"display": "block", "margin-left": "auto",
                                       "margin-right": "auto", "width": "60%"}),
                         dcc.Graph(id='Volume')
                      ], className="container"),
                  ])



   ])
])
@app.callback(Output('highlow', 'figure'),
          [Input('my-dropdown', 'value')])
def update_graph(selected_dropdown):
   dropdown = {"AXISBANK": "Axis","BAJAJ-AUTO": "Bajaj","BAJFINANCE":
"Bjf","BHARTIARTL": "Airtel","Hcl": "HCLTECH","Hdfc": "HDFC","Hfbank":
"HDFCBANK","Hero": "HEROMOTOCO","unilevar": "HINDUNILVR","icici":
"ICICIBANK","indusbank": "INDUSINDBK","infosys": "INFY","itc": "ITC","kotak":
"KOTAKBANK","LT": "LT","mahindra": "M&M","maruti": "MARUTI","nestle":
"NESTLEIND","ntpc": "NTPC","ongc": "ONGC","powergrid":
"POWERGRID","reliance": "RELIANCE","sbi": "SBIN","sunpharma":
"SUNPHARMA","tatasteel": "TATASTEEL","tcs": "TCS","techm": "TECHM","titan":
"TITAN","ultratech": "ULTRACEMCO",}
   trace1 = []
   trace2 = []
   for stock in selected_dropdown:
      trace1.append(
        go.Scatter(x=df[df["Stock"] == stock]["Date"],
                      y=df[df["Stock"] == stock]["High"],
                      mode='lines', opacity=0.7,
                      name=f'High {dropdown[stock]}',textposition='bottom center'))
        trace2.append(
          go.Scatter(x=df[df["Stock"] == stock]["Date"],
                      y=df[df["Stock"] == stock]["Low"],
                      mode='lines', opacity=0.6,
                      name=f'Low {dropdown[stock]}',textposition='bottom center'))
    traces = [trace1, trace2]
    data = [val for sublist in traces for val in sublist]
    figure = {'data': data,
              'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',
                                              '#FF7400', '#FFF400', '#FF0056'],
            height=600,
            title=f"High and Low Prices for {', '.join(str(dropdown[i]) for i in
selected_dropdown)} Over Time",
            xaxis={"title":"Date",
                   'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',
                                                        'step': 'month',
                                                        'stepmode': 'backward'},
                                                       {'count': 6, 'label': '6M',
                                                        'step': 'month',
                                                        'stepmode': 'backward'},
                                                       {'step': 'all'}])},
                   'rangeslider': {'visible': True}, 'type': 'Date'},
             yaxis={"title":"Price (USD)"})}
    return figure



@app.callback(Output('Volume', 'figure'),
              [Input('my-dropdown2', 'value')])
def update_graph(selected_dropdown_value):
    dropdown = {"AXISBANK": "Axis","BAJAJ-AUTO": "Bajaj","BAJFINANCE":
```

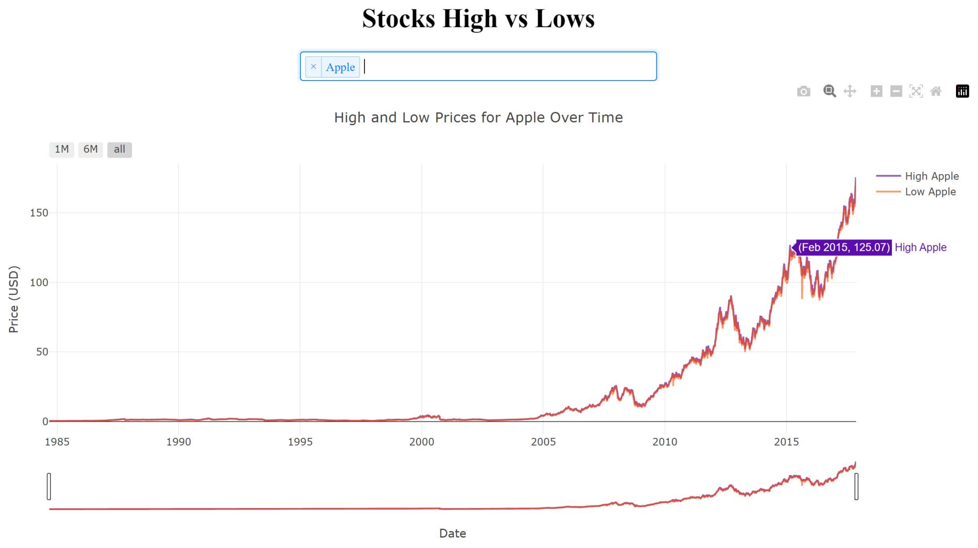
```
"Bjf","BHARTIARTL": "Airtel","Hcl": "HCLTECH","Hdfc": "HDFC","Hfbank":
"HDFCBANK","Hero": "HEROMOTOCO","unilevar": "HINDUNILVR","icici":
"ICICIBANK","indusbank": "INDUSINDBK","infosys": "INFY","itc": "ITC","kotak":
"KOTAKBANK","LT": "LT","mahindra": "M&M","maruti": "MARUTI","nestle":
"NESTLEIND","ntpc": "NTPC","ongc": "ONGC","powergrid":
"POWERGRID","reliance": "RELIANCE","sbi": "SBIN","sunpharma":
"SUNPHARMA","tatasteel": "TATASTEEL","tcs": "TCS","techm": "TECHM","titan":
"TITAN","ultratech": "ULTRACEMCO",}
  trace1 = []
  for stock in selected_dropdown_value:
    trace1.append(
     go.Scatter(x=df[df["Stock"] == stock]["Date"],
           y=df[df["Stock"] == stock]["Volume"],
           mode='lines', opacity=0.7,
           name=f'Volume {dropdown[stock]}', textposition='bottom center'))
  traces = [trace1]
  data = [val for sublist in traces for val in sublist]
  figure = {'data': data,
      'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',
                    '#FF7400', '#FFF400', '#FF0056'],
      height=600,
      title=f"Market Volume for {', '.join(str(dropdown[i]) for i in
selected_dropdown_value)} Over Time",
      xaxis={"title":"Date",
          'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',
                          'step': 'month',
                          'stepmode': 'backward'},
                         {'count': 6, 'label': '6M',
                          'step': 'month',
                          'stepmode': 'backward'},
                         {'step': 'all'}])},
          'rangeslider': {'visible': True}, 'type': 'Date'},
      yaxis={"title":"Transactions Volume"})}
  return figure
```

```
if __name__=="__main__":
        app.run(debug=False)
```

Wesbite Pages:

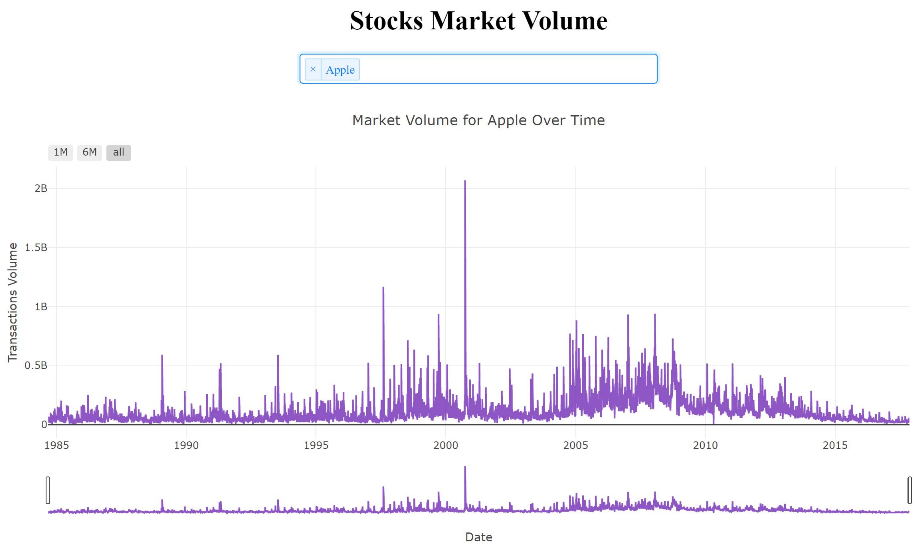






Stocks High vs Lows

× Apple



Stocks Market Volume

× Apple

CHAPTER 5 CONCLUSION

In this project, we are predicting closing stock price of any given organization, we developed a web application for predicting close stock price using LMS and LSTM algorithms for prediction. We have applied datasets belonging to NSE TATAGLOBAL, Apple, Facebook, Tesla , Microsoft and achieved above 90% accuracy for these datasets.

REFERENCES

DATSETS: TataGlobal, StockData.

[1] Stock Price Prediction Using LSTM on Indian Share Market by Achyut Ghosh, Soumik Bose1, Giridhar Maji, Narayan C. Debnath, Soumya Sen

[2] S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in International Conference on Advances in Computing, Communications and Informatics, 2017.

[3] Murtaza Roondiwala, Harshal Patel, Shraddha Varma, "Predicting Stock Prices Using LSTM" in Undergraduate Engineering Students, Department of Information Technology, Mumbai University, 2015.

[4] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, "An innovative neural network approach for stock market prediction", 2018

[5] Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikhin Gupta, Himanshu Dhiman, Lokesh Chouhan Department of Computer Science and Engineering National Institute of Technology, Hamirpur – 177005, INDIA - Stock Market Prediction Using Machine Learning.

[6] Pranav Bhat Electronics and Telecommunication Department, Maharashtra Institute of Technology, Pune. Savitribai Phule Pune University - A Machine Learning Model for Stock Market Prediction.

[7] Anurag Sinha Department of computer science, Student, Amity University Jharkhand Ranchi, Jharkhand (India), 834001 - Stock Market Prediction Using Machine Learning.