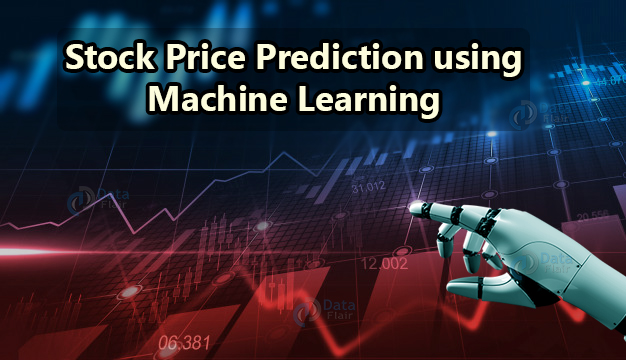
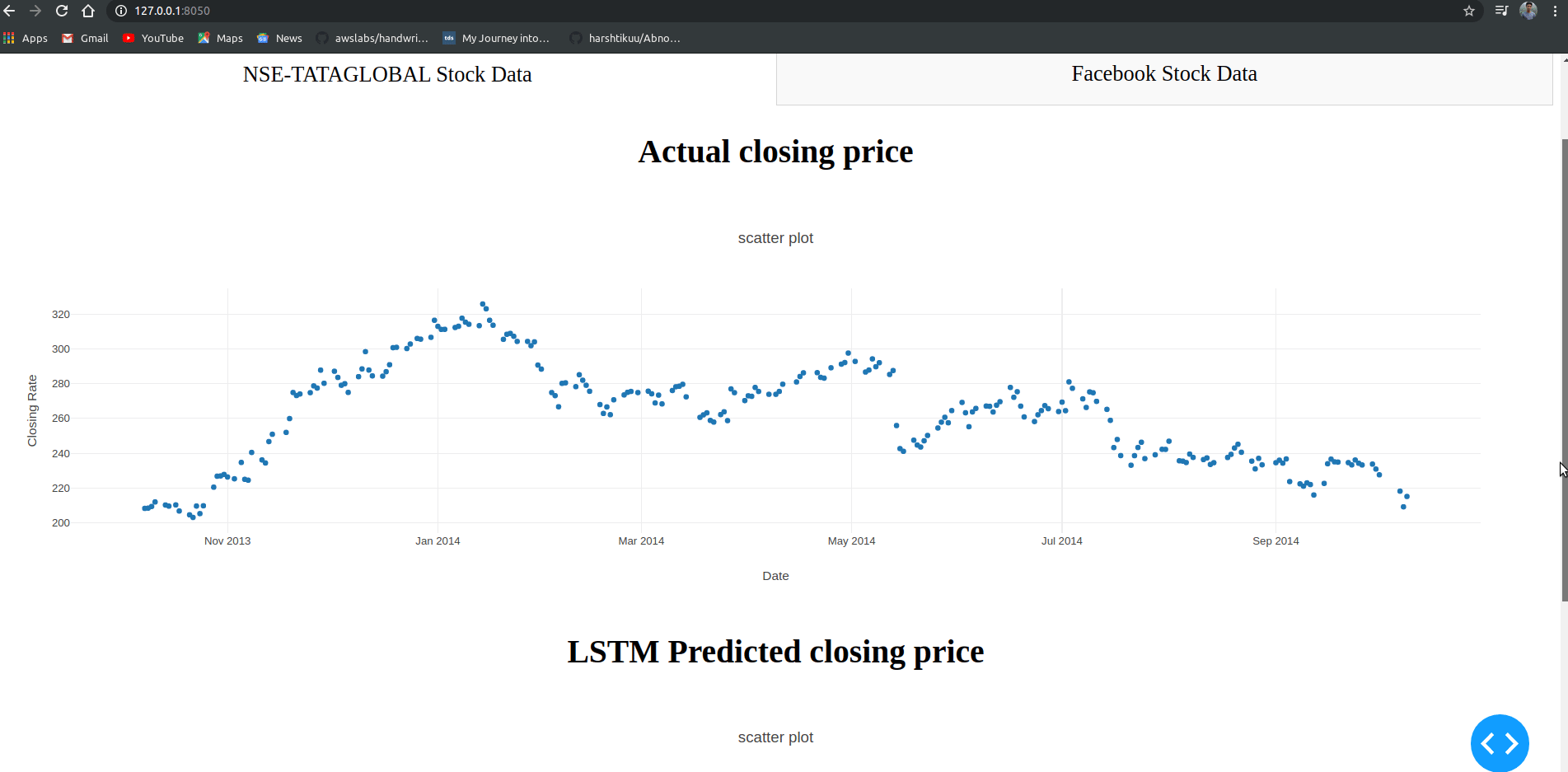
**Stock Price Prediction using Machine Learning**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2020/01/stock-price-prediction-using-ML.jpg)

**Project idea –** There are many datasets available for the stock market prices. This machine learning beginner’s project aims to predict the future price of the stock market based on the previous year’s data.

Machine learning has significant applications in the stock price prediction. In this machine learning project, we will be talking about predicting the returns on stocks. This is a very complex task and has uncertainties. We will develop this project into two parts:

1. First, we will learn how to predict stock price using the LSTM neural network.
2. Then we will build a dashboard using Plotly dash for stock analysis.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2020/05/Stock-Price-Prediction-project-dashboard.gif)

## Stock Price Prediction Project

#### Datasets

1. To build the stock price prediction model, we will use the NSE TATA GLOBAL dataset. This is a dataset of Tata Beverages from Tata Global Beverages Limited, National Stock Exchange of India: [Tata Global Dataset](https://data-flair.training/blogs/download-tata-global-beverages-stocks-data/)
2. To develop the dashboard for stock analysis we will use another stock dataset with multiple stocks like Apple, Microsoft, Facebook: [Stocks Dataset](https://data-flair.training/blogs/download-stocks-price-data/)

### Stock price prediction using LSTM

1. Imports:

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 keras.models import Sequential

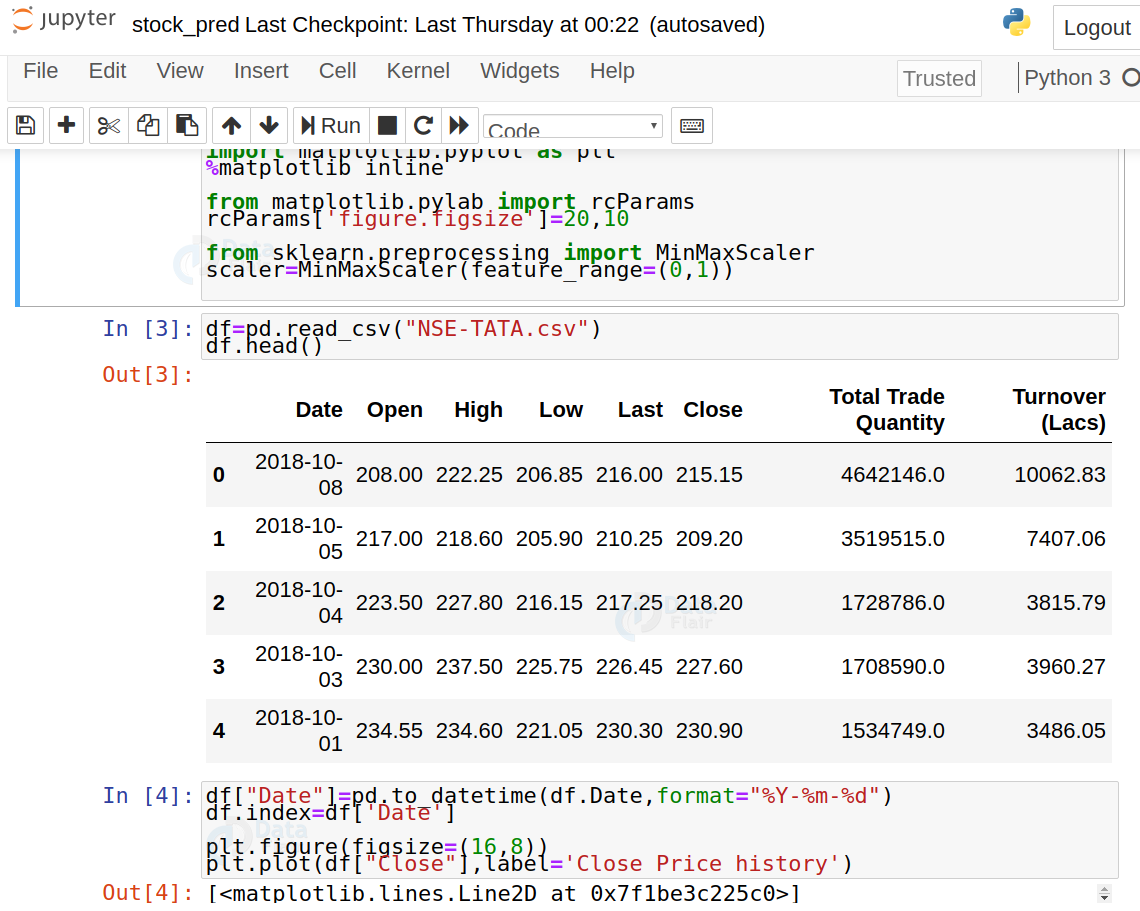
from keras.layers import LSTM,Dropout,Dense

from sklearn.preprocessing import MinMaxScaler

**2. Read the dataset:**

df=pd.read\_csv("NSE-TATA.csv")

df.head()

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2020/05/read-stock-data.png)

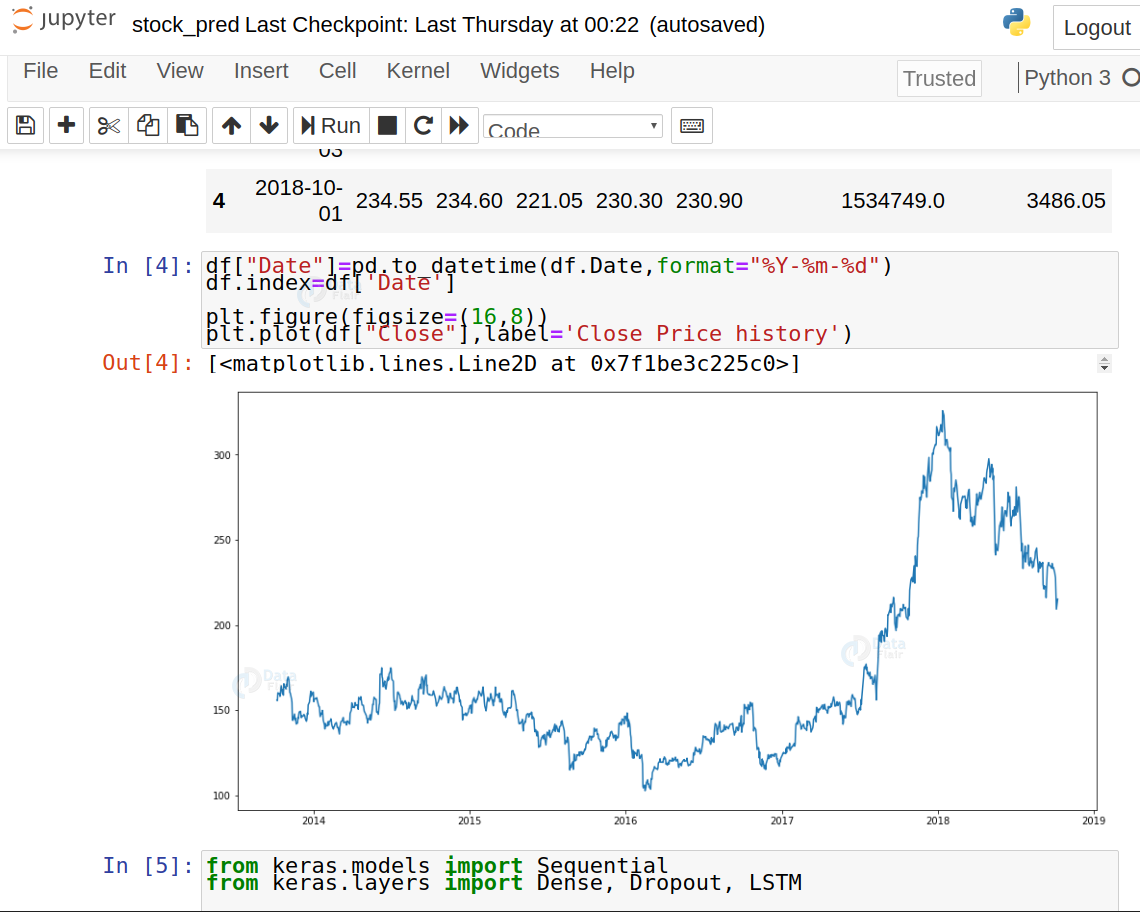
**3. Analyze the closing prices from dataframe:**

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')

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2020/05/analyse-stock-price.png)

**4. 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]

**5. Normalize the new filtered dataset:**

scaler=MinMaxScaler(feature\_range=(0,1))

final\_dataset=new\_dataset.values

train\_data=final\_dataset[0:987,:]

valid\_data=final\_dataset[987:,:]

new\_dataset.index=new\_dataset.Date

new\_dataset.drop("Date",axis=1,inplace=**True**)

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))

**6. Build and train the LSTM model:**

lstm\_model=Sequential()

lstm\_model.add(LSTM(units=50,return\_sequences=**True**,input\_shape=(x\_train\_data.shape[1],1)))

lstm\_model.add(LSTM(units=50))

lstm\_model.add(Dense(1))

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.compile(loss='mean\_squared\_error',optimizer='adam')

lstm\_model.fit(x\_train\_data,y\_train\_data,epochs=1,batch\_size=1,verbose=2)

7. **Take a sample of a dataset to make stock price predictions using the LSTM model:**

X\_test=[]

**for** i **in** range(60,inputs\_data.shape[0]):

X\_test.append(inputs\_data[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\_closing\_price=lstm\_model.predict(X\_test)

predicted\_closing\_price=scaler.inverse\_transform(predicted\_closing\_price)

8.**Save the LSTM model:**

lstm\_model.save("saved\_model.h5")

9. **Visualize the predicted stock costs with actual stock costs:**

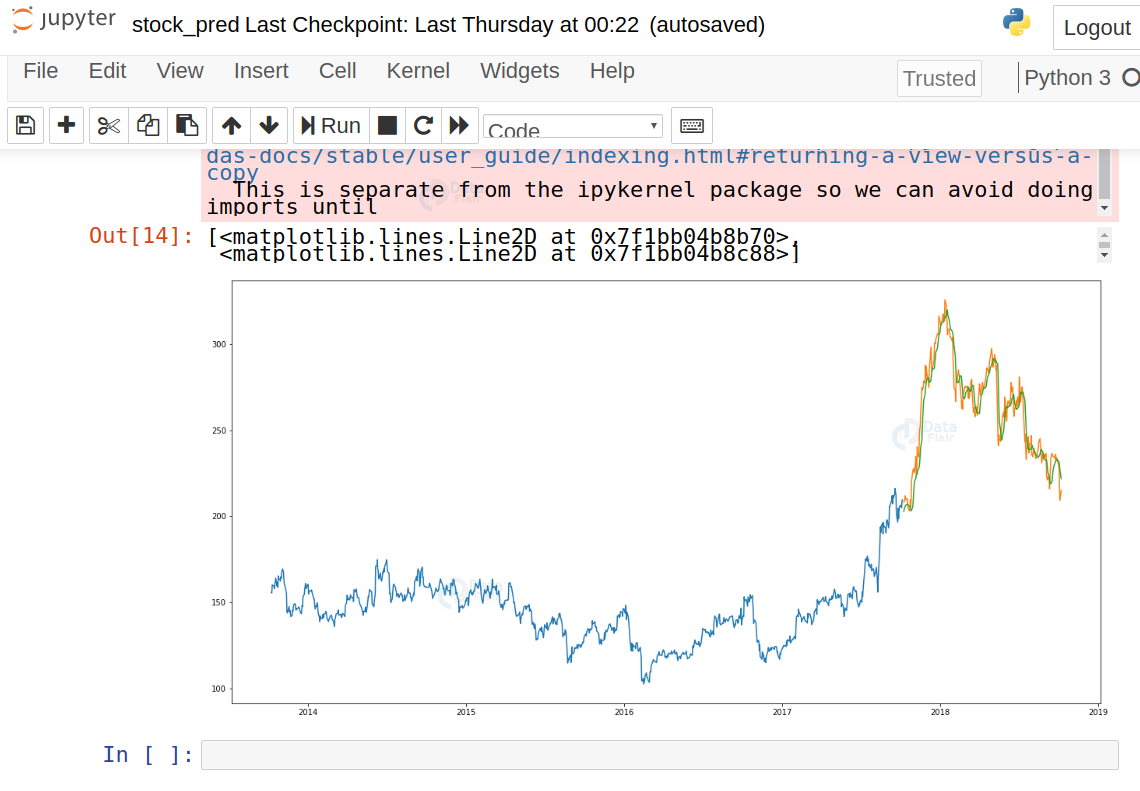
train\_data=new\_dataset[:987]

valid\_data=new\_dataset[987:]

valid\_data['Predictions']=predicted\_closing\_price

plt.plot(train\_data["Close"])

plt.plot(valid\_data[['Close',"Predictions"]])

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2020/05/predict-stock-cost.png)

## Build the dashboard using Plotly dash

In this section, we will build a dashboard to analyze stocks. [Dash](https://dash.plotly.com/introduction) is a python framework that provides an abstraction over flask and react.js to build analytical web applications.  
Before moving ahead, you need to install dash. Run the below command in the terminal.

pip3 install dash

pip3 install dash-html-components

pip3 install dash-core-components

Now make a new python file stock\_app.py and paste the below script:

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("./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))

model=load\_model("saved\_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=model.predict(X\_test)

closing\_price=scaler.inverse\_transform(closing\_price)

train=new\_data[:987]

valid=new\_data[987:]

valid['Predictions']=closing\_price

df= pd.read\_csv("./stock\_data.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='Facebook Stock Data', children=[

html.Div([

html.H1("Facebook Stocks High vs Lows",

style={'textAlign': 'center'}),

dcc.Dropdown(id='my-dropdown',

options=[{'label': 'Tesla', 'value': 'TSLA'},

{'label': 'Apple','value': 'AAPL'},

{'label': 'Facebook', 'value': 'FB'},

{'label': 'Microsoft','value': 'MSFT'}],

multi=**True**,value=['FB'],

style={"display": "block", "margin-left": "auto",

"margin-right": "auto", "width": "60%"}),

dcc.Graph(id='highlow'),

html.H1("Facebook Market Volume", style={'textAlign': 'center'}),

dcc.Dropdown(id='my-dropdown2',

options=[{'label': 'Tesla', 'value': 'TSLA'},

{'label': 'Apple','value': 'AAPL'},

{'label': 'Facebook', 'value': 'FB'},

{'label': 'Microsoft','value': 'MSFT'}],

multi=**True**,value=['FB'],

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 = {"TSLA": "Tesla","AAPL": "Apple","FB": "Facebook","MSFT": "Microsoft",}

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 = {"TSLA": "Tesla","AAPL": "Apple","FB": "Facebook","MSFT": "Microsoft",}

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"})}

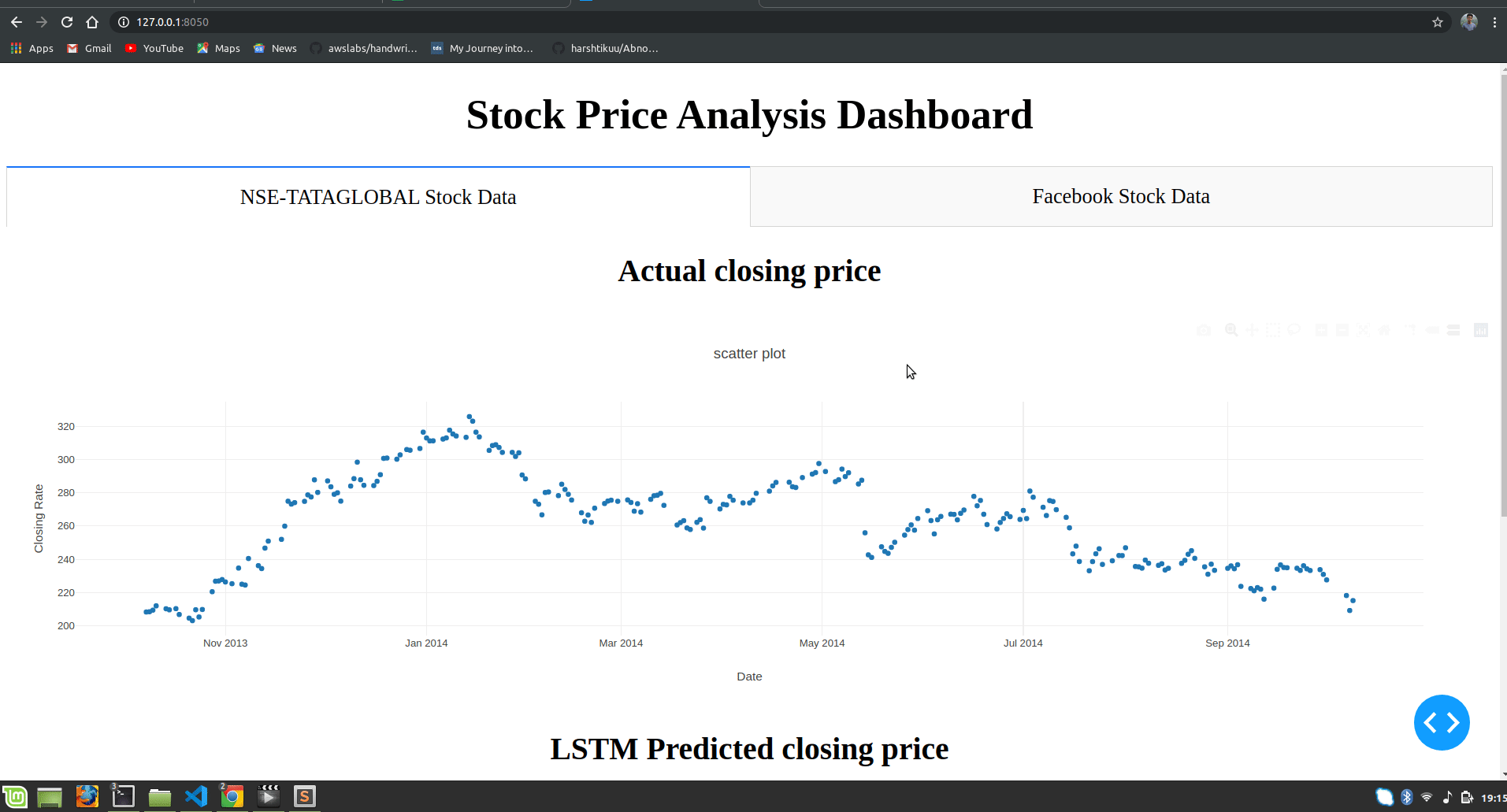
**return** figure

**if** \_\_name\_\_=='\_\_main\_\_':

app.run\_server(debug=**True**)

Now run this file and open the app in the browser:

python3 stock\_app.py

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2020/05/Stock-Price-Prediction-project-dashboard-2.gif)

### Summary

Stock price prediction is a machine learning project for beginners; in this tutorial we learned how to develop a stock cost prediction model and how to build an interactive dashboard for stock analysis. We implemented stock market prediction using the LSTM model. OTOH, Plotly dash python framework for building dashboards.