

**A Mini Project Report (Internal Assessment for Machine Learning) on**

“STOCK PRICE PREDICTION USING MACHINE LEARNING”

Submitted

In partial fulfilment of the requirement for the ‘V’ Semester of B.Tech in Information Science Engineering during the academic year 2023-24

**Submitted By**

**Md Kaif Mustafa**

**(R21EJ020)**

**Under the guidance of**

Dr. Vishwanath R Hulipalled

Professor, School of C&IT

Deputy Director, R&D Cell

**REVA UNIVERSITY**

School of Computing and Information Technology

Bengaluru-560064

2023-24



**SCHOOL OF COMPUTING & INFORMATION TECHNOLOGY**

This is to certify that the mini project entitled “**Stock Price Prediction using Machine Learning**” is a Bonafide work carried out by **Md Kaif Mustafa (R21EJ020)** respectively in partial fulfilment of 5th semester of **CS&IT** program of Bachelor of Technology, REVA University during the academic year 2023-24. It is certified that all the corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the school library. The mini-project report has been approved as it satisfies the academic requirements.

Signature of the Faculty

**(Dr.Vishwanath R Hulipalled)**

**TABLE OF CONTENTS**

**Chapter No. Chapter Name**

**1 ABSTRACT**

**2 INTRODUCTION**

**3 METHODOLOGY**

**4 EXPERIMENTAL RESULTS**

**5 CONCLUSION**

**6 REFERENCES**

**ABSTRACT**

This study explores the application of machine learning models to predict stock prices. Using historical data with various features, including past prices, trading volumes, and technical indicators, we compare models such as linear regression, support vector machines, and deep neural networks. Feature engineering and hyperparameter tuning are employed for optimization. The dataset covers diverse companies, and models are evaluated on in-sample and out-of-sample data using metrics like MSE and RMSE. The research investigates the impact of time horizons and the significance of different features, offering insights for investors and financial professionals.

Keywords: Stock prediction, Machine learning, Regression, SVM, Neural networks, Feature engineering, Hyperparameter tuning, Time horizons.

**INTRODUCTION**

Stock price prediction typically involves the use of various functions and techniques from the field of machine learning and finance. Here are some commonly used functions and methods:

***Linear Regression:***

Function: y=mx+b

Linear regression is a simple and widely used technique that establishes a linear relationship between independent variables (features) and the target variable (stock price).

***Support Vector Machines (SVM):***

Function: Decision boundary defined by support vectors.

SVM is used for classification and regression tasks. It finds a hyperplane that best separates data points into different classes or predicts a continuous outcome.

***Moving Averages:***

Functions: Simple Moving Average (SMA), Exponential Moving Average (EMA).

Moving averages smooth out price data to create a single flowing line, making it easier to identify trends over time.

***Technical Indicators:***

Functions: Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Bollinger Bands, etc.

Technical indicators provide additional information about the market conditions and trends based on historical price and volume data.

***Neural Networks:***

Functions: Activation functions (e.g., Sigmoid, Tanh, ReLU)

Neural networks, especially deep learning models, use activation functions and complex architectures to capture non-linear relationships in the data.

***ARIMA (Auto Regressive Integrated Moving Average):***

Function: Combination of autoregressive, differencing, and moving average components.

ARIMA models are popular for time series analysis, capturing trends and seasonality in stock prices.

***Logistic Regression:***

Function:1/(1+e)^(-(mx+b))

Logistic regression is used for binary classification tasks, predicting whether a stock's price will go up or down.

***Decision Trees and Random Forests:***

Function: Tree-like model with decision nodes and leaf nodes.

Decision trees and random forests can capture complex relationships in the data and are used for both classification and regression tasks.

***GARCH (Generalized Autoregressive Conditional Heteroskedasticity):***

Function: Models volatility clustering in financial time series.

GARCH models are commonly used to predict volatility in financial markets.

***Ensemble Methods:***

Functions: Bagging, Boosting

Ensemble methods combine multiple models to improve overall predictive performance.

**METHODOLOGY**

***Stock Prediction Code:***

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))

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

df.head()

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')

from keras.models import Sequential

from keras.layers import LSTM,Dropout,Dense

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\_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))

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)

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))

closing\_price=model.predict(X\_test)

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

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

train\_data=new\_dataset[:987]

valid\_data=new\_dataset[987:]

valid\_data['Predictions']=prediction\_closing

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

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

***Stock App code:***

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("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("Stocks 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"})}

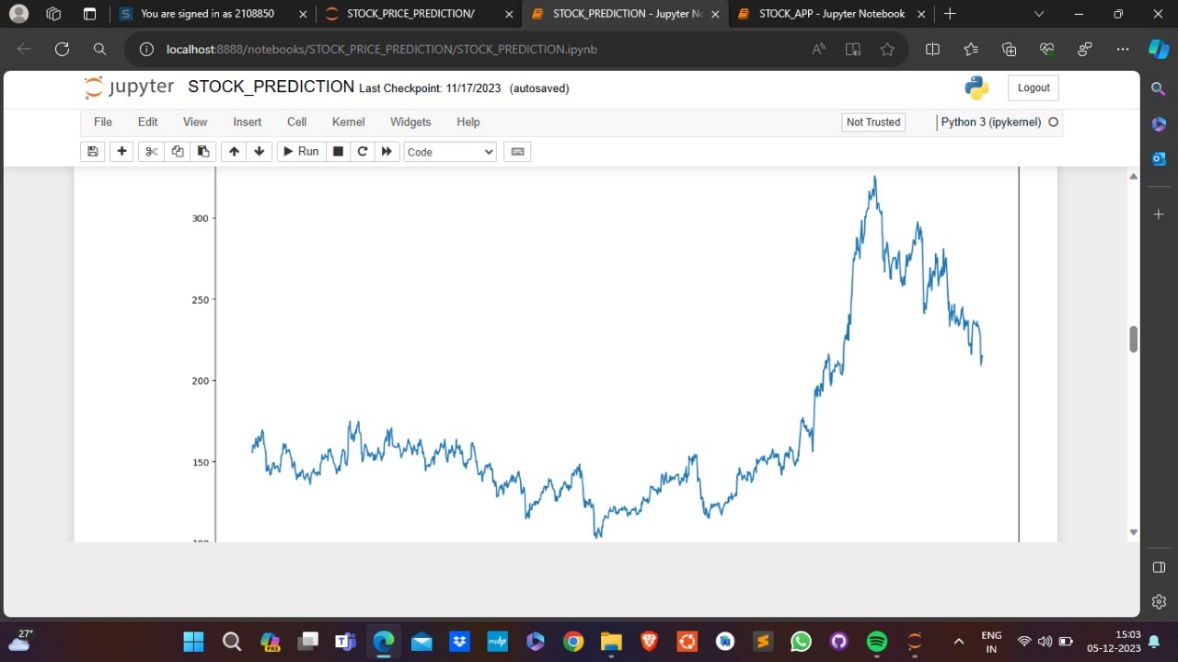
return figure

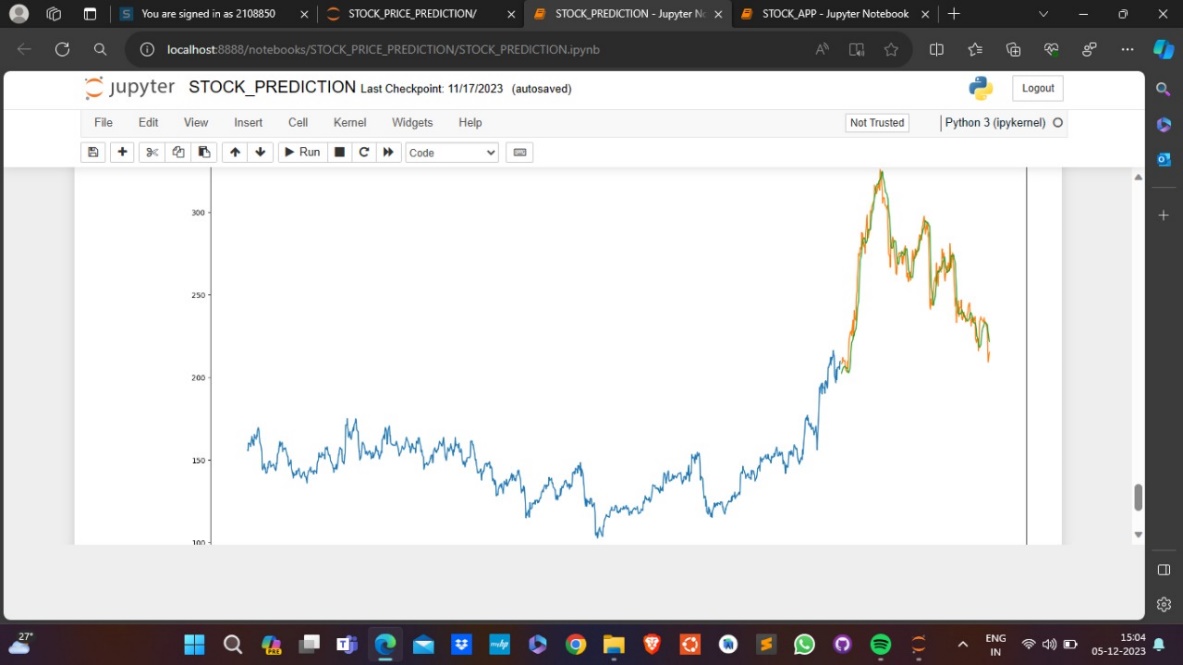
if \_name=='main\_':

app.run\_server(debug=True)

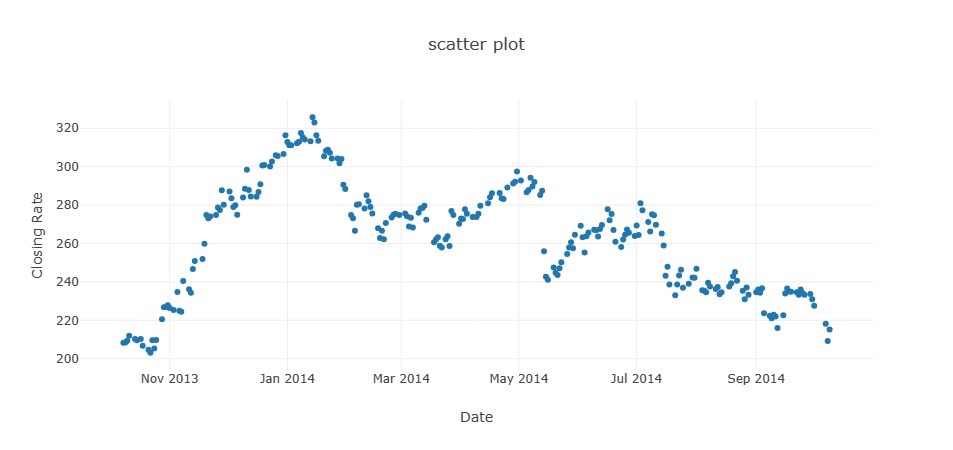
**EXPERIMENTAL RESULT**

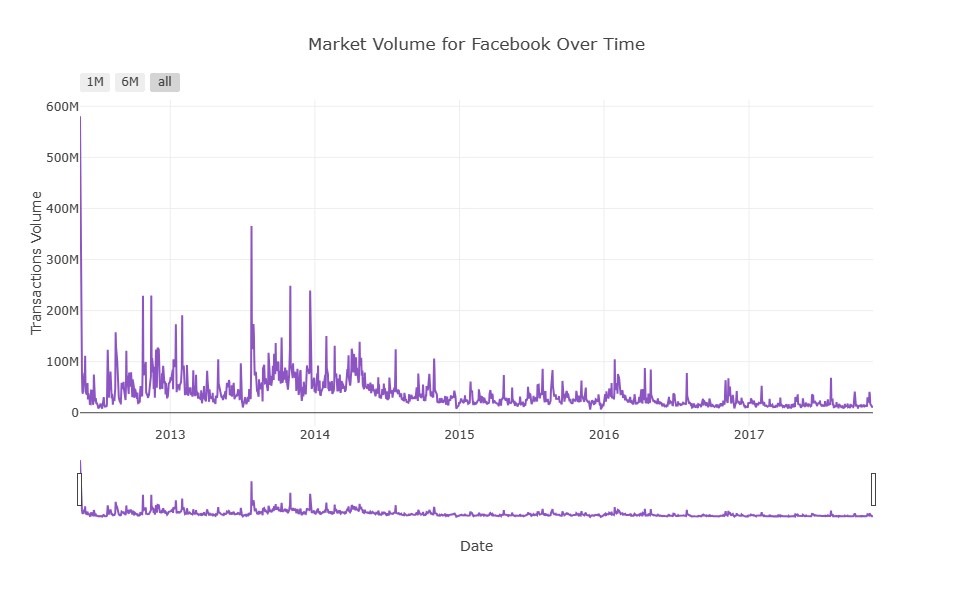
***Stock Prediction images:***

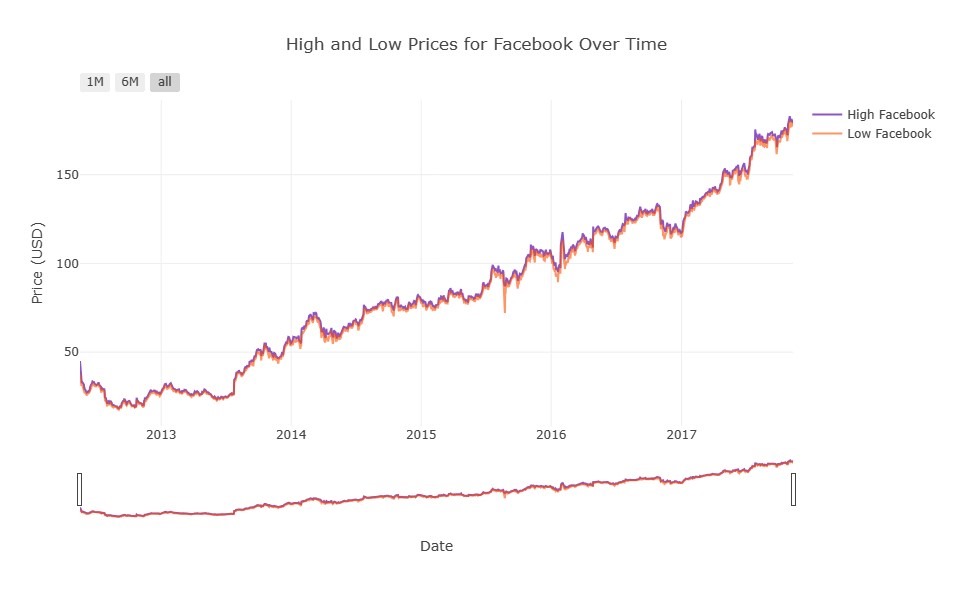


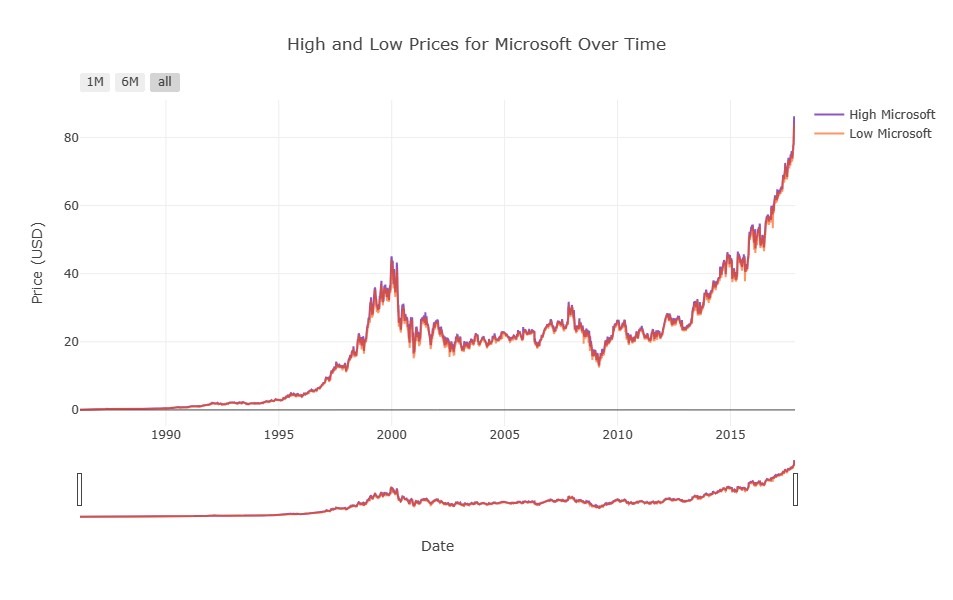


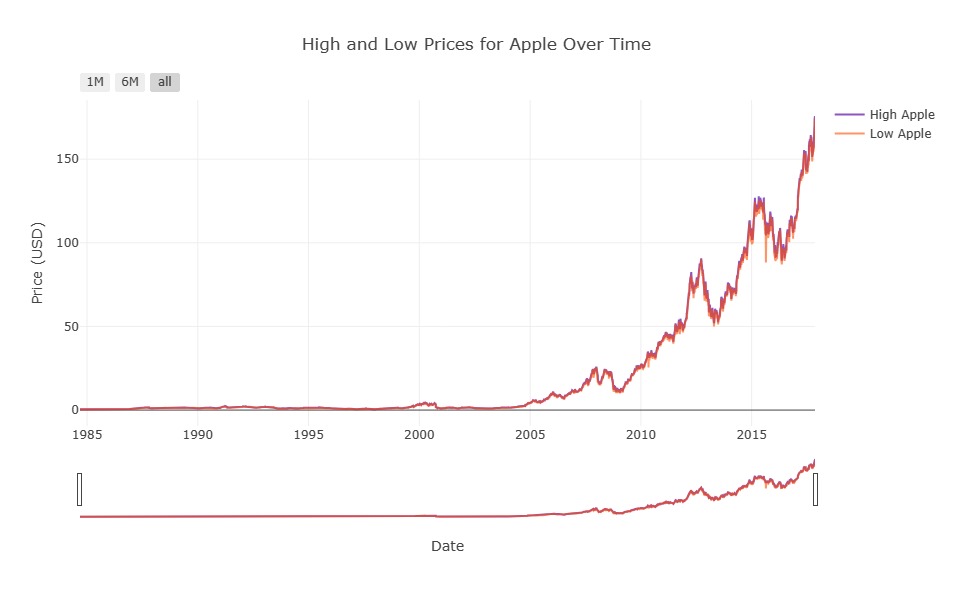
***Stock App images:***

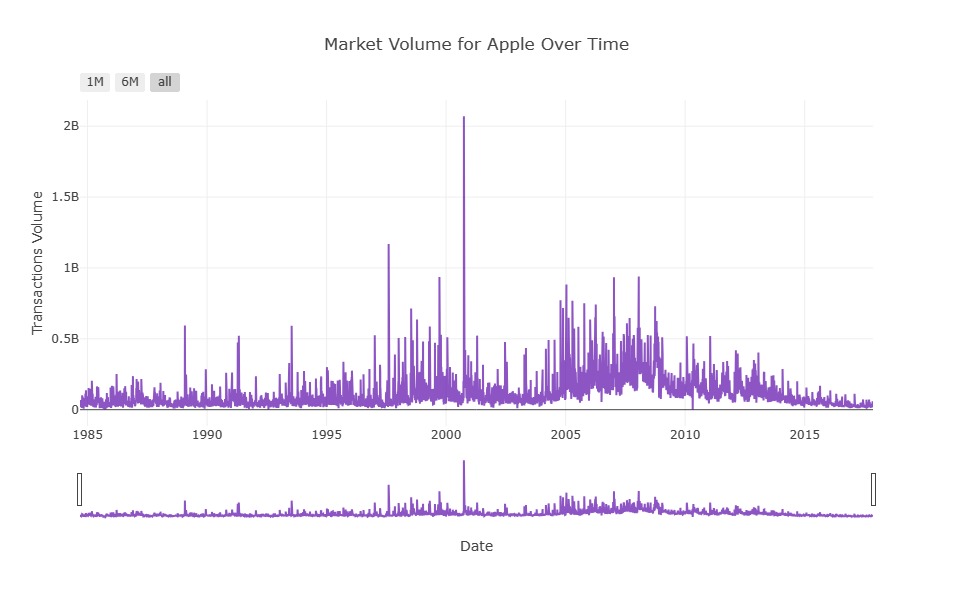


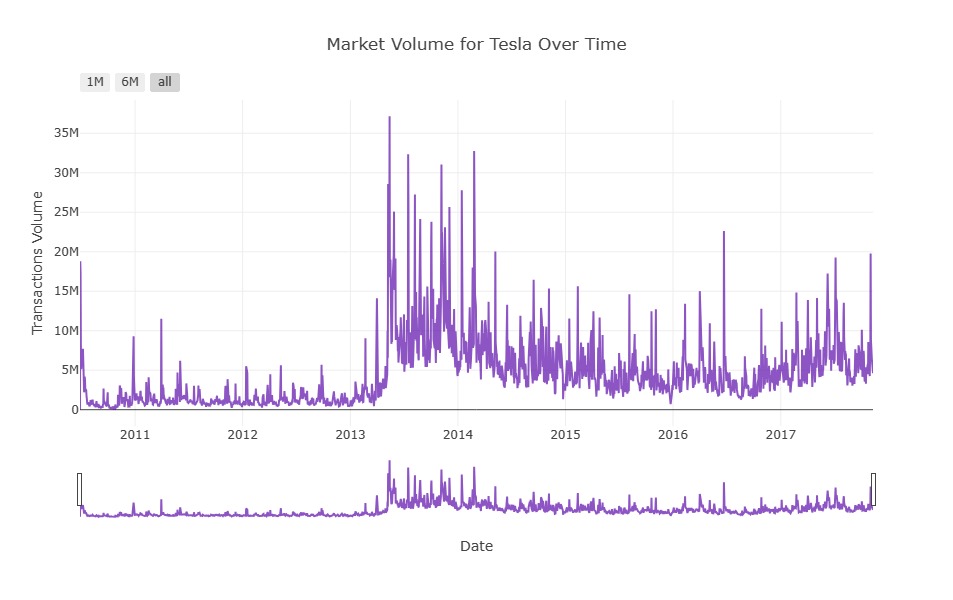


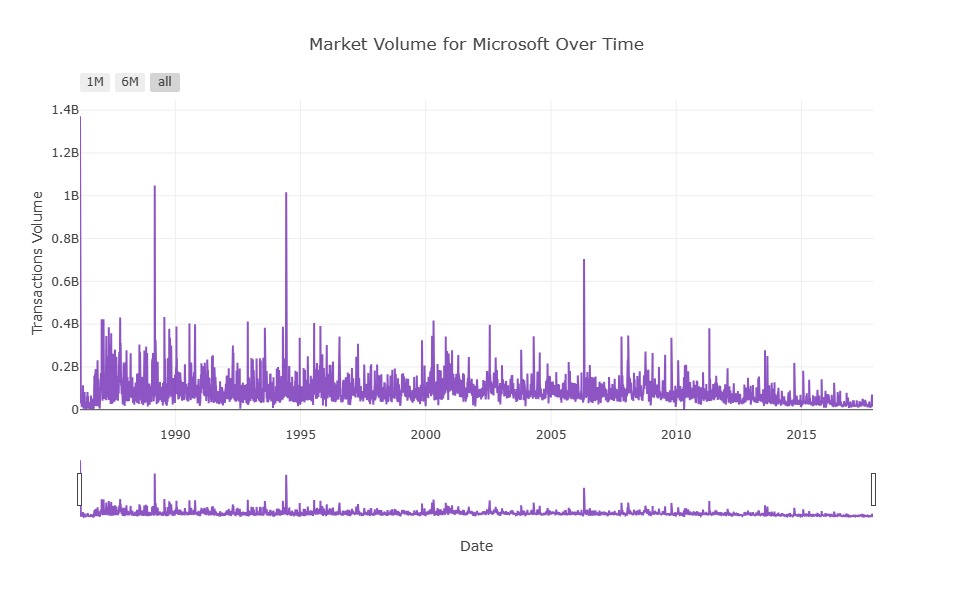














**CONCLUSION**

In summary, predicting the stock market is challenging due to the inherent complexity, unpredictable variables, and the impact of human behavior. While various models and analyses exist, they come with limitations, and the market's randomness and sensitivity to external events make accurate short-term predictions elusive. Investors should exercise caution, focus on long-term goals, and diversify their portfolios to manage risks effectively.

**REFERENCE**

# [Two Sigma: Using News to Predict Stock Movements | Kaggle](https://www.kaggle.com/c/two-sigma-financial-news/data)

[Stock Price Prediction - Machine Learning Project in Python - DataFlair (data-flair.training)](https://data-flair.training/blogs/stock-price-prediction-machine-learning-project-in-python/)