STOCK PRICE PREDICTION

A Project Work Synopsis

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE IN BIG DATA

Submitted by:
SUSHANT BISHT
20BCS6148

Under the Supervision of:

GURPREET SINGH PANESAR

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

2020-2024

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my project guide Gurpreet Singh Panesar, Department of Computer Science and Engineering, for his/her guidance with unsurpassed knowledge and immense encouragement. I am grateful to Mr. Aman Kaushik, the Head of the Department, Computer Science, and Engineering in Big Data, for providing me with the required facilities for the completion of the project work.

I express my thanks to Project Coordinator Gurpreet Singh Panesar, for his continuous support and encouragement. I thank all teaching faculty of the Department of CSE-Big Data, whose suggestions during reviews helped me in the accomplishment of my project.

I would like to thank my parents, friends, and classmates for their encouragement throughout my project period. Last but not the least, I thank everyone for supporting me directly or indirectly in completing this project successfully.

PROJECT STUDENT

SUSHANT BISHT - 20BCS6148

DECLARATION

I, Sushant Bisht of fourth semester B.E., in the department of Computer Science and Engineering-Big Data from CHANDIGARH UNIVERSITY, Gharuan, Mohali, hereby declare that the project work entitled STOCK PRICE PREDICTION is carried out by me and submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science Engineering-Big Data during the academic year 2020-2024 and has not been submitted to any other university for the award of any kind of degree.

PROJECT STUDENT	
SUSHANT BISHT -	20BCS6148

ABSTRACT

In this project, I attempt to implement a machine learning approach to predict stock prices. Machine learning is effectively implemented at forecasting stock prices. The objective is to predict the stock prices 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 to achieve better stock prediction accuracy and issue 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 my 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, ML, DL, Trade Open, Trade Close, Trade Low, Trade High

Table of Contents

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 a new business or the need for a high salary career. Stock markets are affected by many factors causing 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 judgments 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 analyzed.

Time-series prediction is a common technique widely used in many real-world applications such as weather forecasting and financial market prediction. It uses continuous data in a period to predict the result in the next time unit. Many time series prediction algorithms have shown their effectiveness in practice. The most common algorithms now are based on Recurrent Neural Networks (RNN), as well as its special type - Long-short Term Memory (LSTM) and Gated Recurrent Unit (GRU). The stock market is a typical area that presents time-series data and many researchers study it and proposed various models. In this project, the LSTM model is used to predict the stock price.

1.1 MOTIVATION FOR WORK

Businesses primarily run over customer 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 the trustworthiness of an organization. Hence there is a necessity for an unbiased 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 analyze 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.

1.2 PROBLEM STATEMENT

Time Series forecasting & modeling 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 and price depends on various factors. The main aim of this project is to predict stock prices using Long short-term memory (LSTM).

2 LITERATURE SURVEY

2.1 INTRODUCTION

"What other people think" has always been an important piece of information for most of us during the decision-making process. The Internet and the Web have now (among other things) made it possible to find out about the opinions and experiences of those in the vast pool of people that are neither our acquaintances nor well-known professional critics — that is, people we have never heard of. And conversely, more and more people are making their opinions available to strangers via the Internet. The interest that individual users show in online opinions about products and services, and the potential influence such opinions wield, is something that is the driving force for this area of interest. And there are many challenges involved in this process that needs to be walked all over to attain proper outcomes out of them. In this survey, I have analyzed the basic methodology that usually happens in this process and measures that are to be taken to overcome the challenges being faced.

2.2 STOCK MARKET PREDICTION USING MACHINE LEARNING

Stock market prediction is an act of trying to determine the future value of a stock other financial instrument traded on a financial exchange. This paper explains the prediction of a stock using Machine Learning. The technical and fundamental or the time series analysis is used by most of the stockbrokers while making the stock predictions. The programming language is used to predict the stock market using machine learning is Python. In this paper, I have proposed a Machine Learning (ML) approach that will be trained from the available stocks data and gain intelligence and then uses the acquired knowledge for an accurate prediction.

3 METHODOLOGIES

3.1 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, backpropagation network, single-layer LSTM, support vector machine, recurrent neural network, etc. They used a Long short-term memory network (LSTM).

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 that 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 fewer resources and has a faster convergence speed.

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

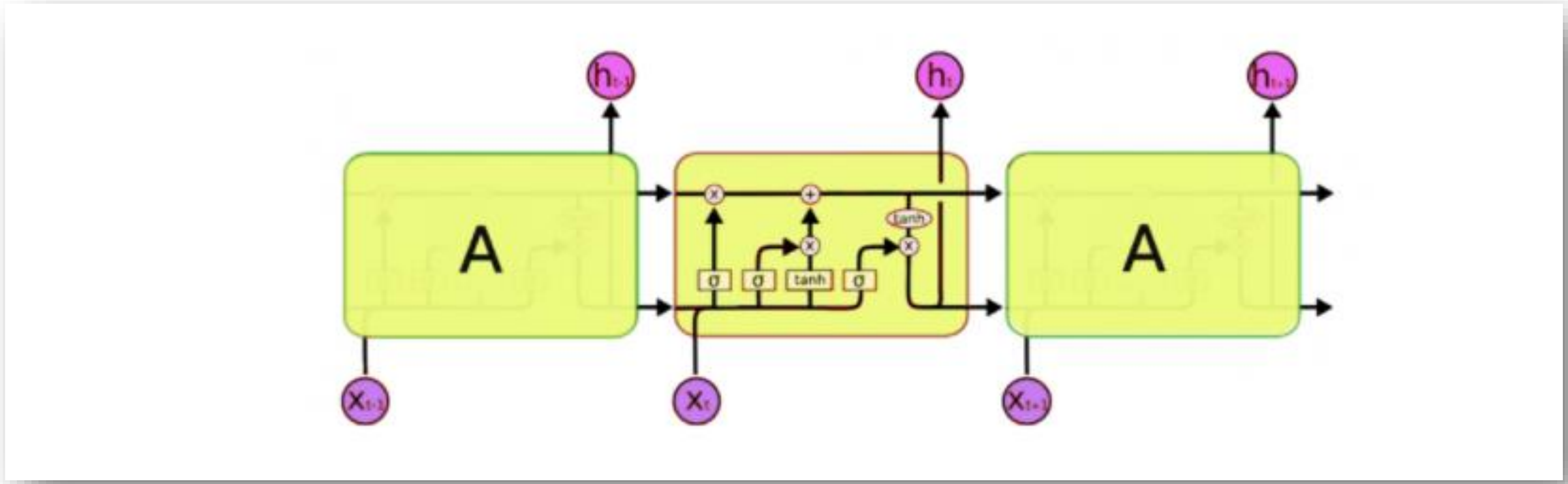
• LSTM is used instead of RNN to avoid exploding and vanishing gradients.

• 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.

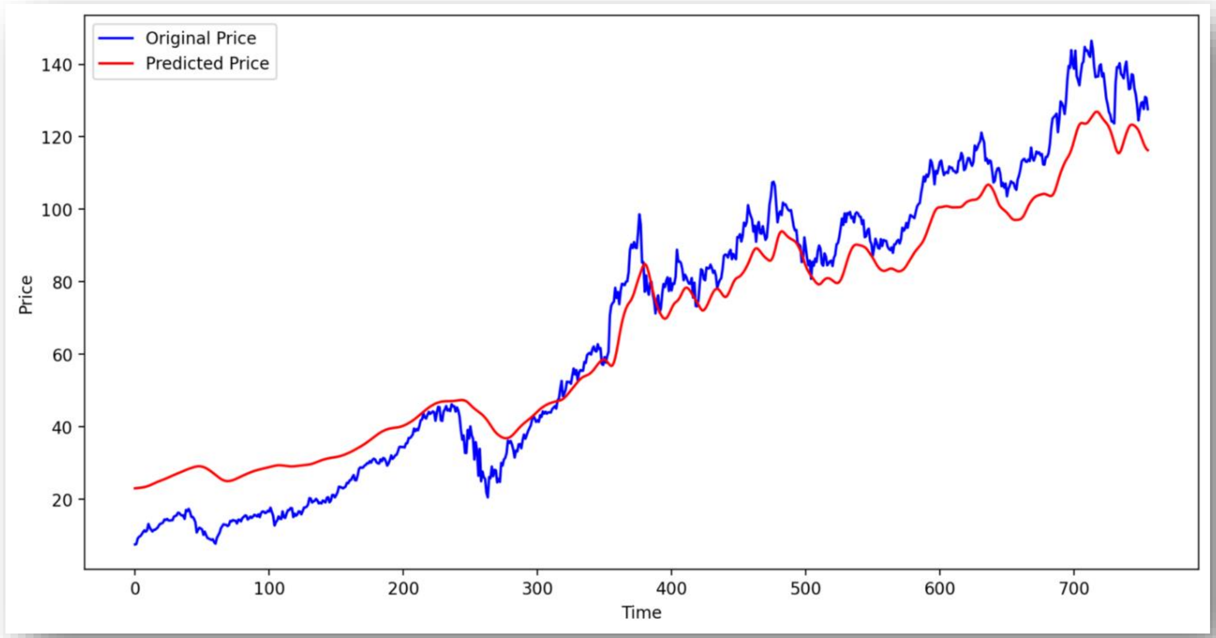
• The historical stock data table contains the information of opening price, the highest price, lowest price, closing price, transaction date, volume, and so on.

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

LSTM Architecture:



Result:



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 the 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 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.

Output Gate:

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

- Creating a vector after applying the 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 output and also to the hidden state of the next cell.

```
# LSTM
Inputs: dataset
Outputs: RMSE of the forecasted data

# Split the dataset into 75% training and 25% testing data
size = length(dataset) * 0.75
train = dataset [0 to size]
test = dataset [size to length(dataset)]

# Procedure to fit the LSTM model
Procedure LSTMAlgorithm (train, test, train_size, epochs)
    X = train
    y = test
    model = Sequential ()
    model.add(LSTM(50), stateful=True)
    model.compile(optimizer='adam', loss='mse')
    model.fit(X, y, epochs=epochs, validation_split=0.2)
    return model

# Procedure to make predictions
Procedure getPredictonsFromModel (model, X)
    predictions = model.predict(X)
    return predictions

epochs = 100
neurons = 50
predictions = empty
# Fit the LSTM model
model = LSTMAlgorithm (train, epoch, neurons)

# Make predictions
pred = model. predict(train)

# Validate the model
n = len(dataset)

error = 0
for i in range(n): error += (abs(real[i] - pred[i])/real[i]) * 100
accuracy = 100 - error/n
```

Hardware Requirements:

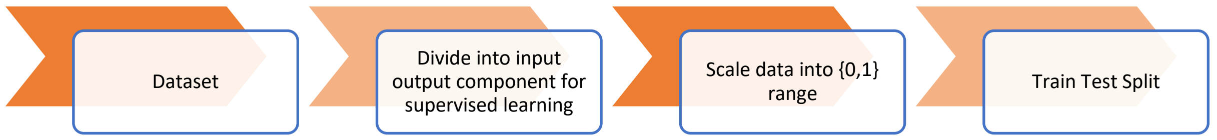
- RAM: 4 GB
- Storage: 500 GB
- CPU: 2 GHz or faster
- Architecture: 32-bit or 64-bit

Software Requirements:

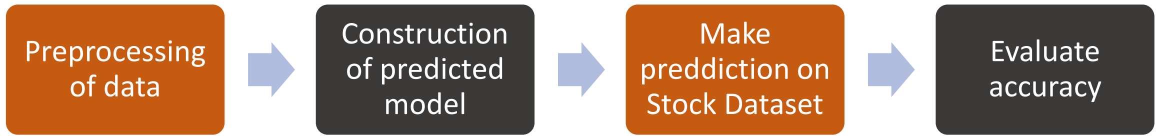
- Python 3.5 in Google Collab is used for data pre-processing, model training, and prediction.
- Operating System: Windows 10 and above or Linux based OS or MAC OS

System Architecture:

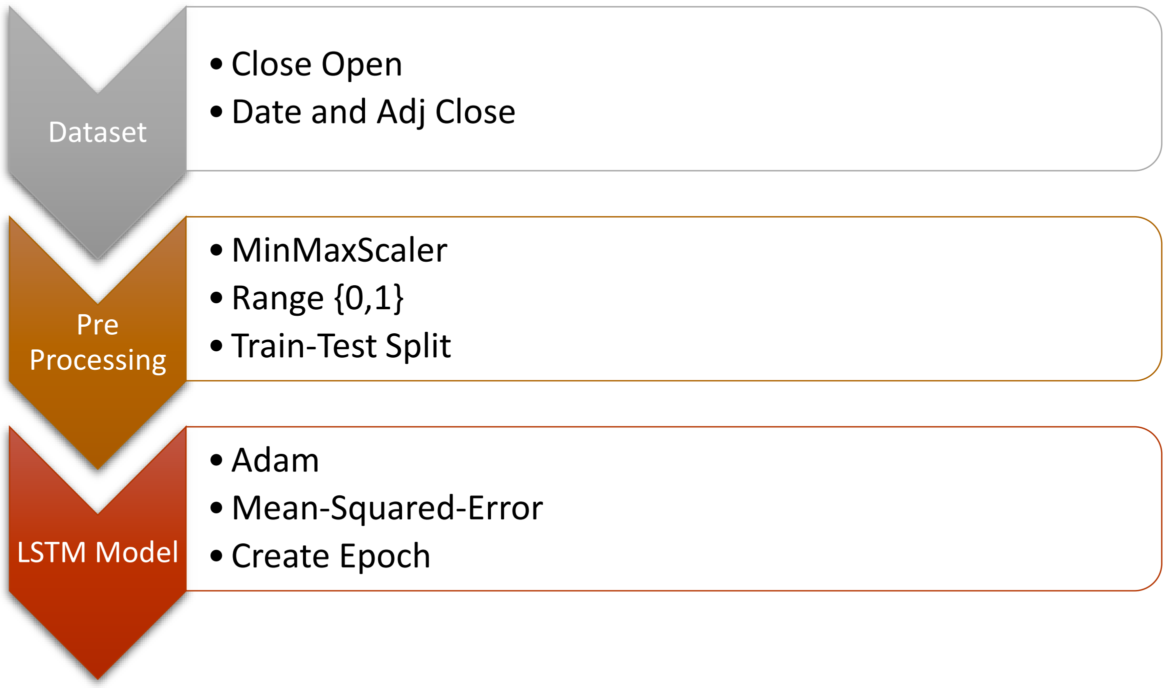
1) Preprocessing of data


Dataset
Divide into input output component for supervised learning
Scale data into {0,1} range
Train Test Split

2) Overall Architecture:

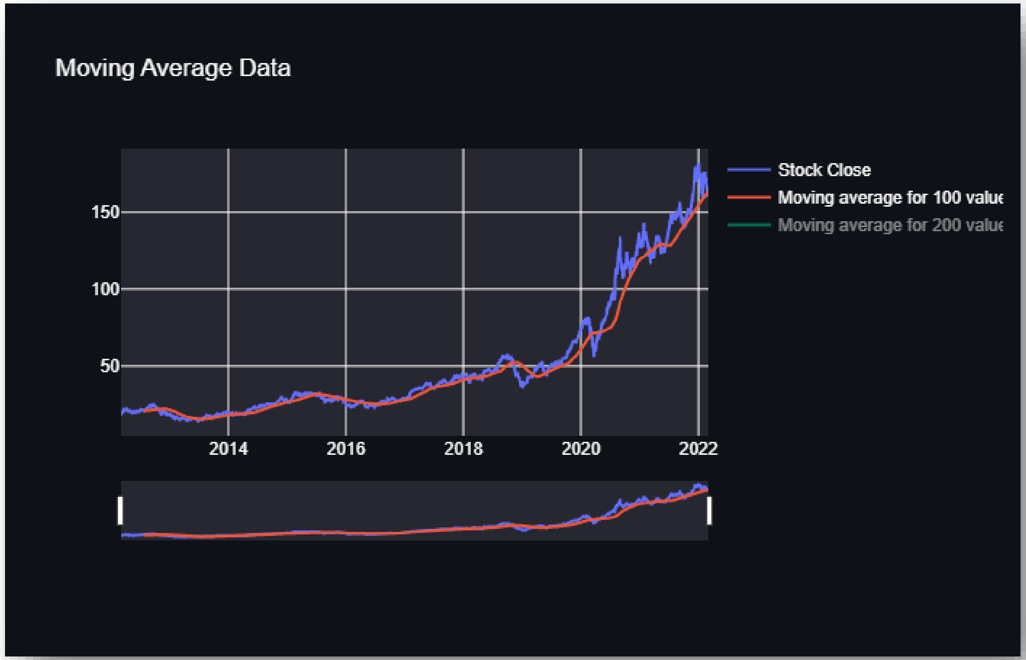

Preprocessing of data
Construction of predicted model
Make preddiction on Stock Dataset
Evaluate accuracy

3) Flowchart:

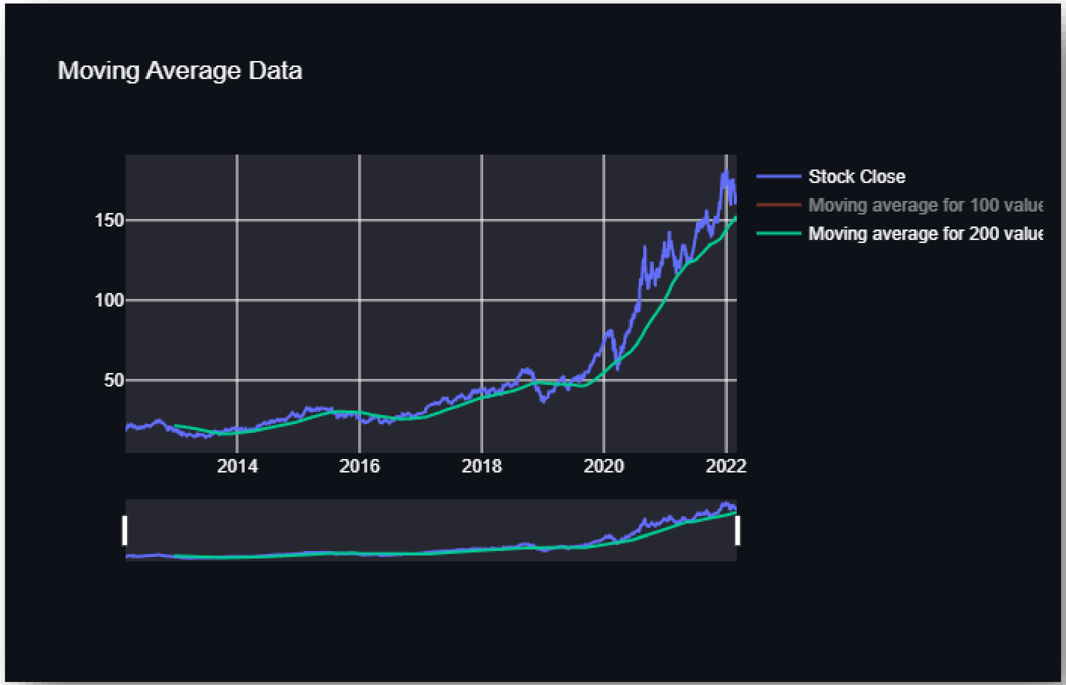


Sample Code:

```
import NumPy as np

import pandas as pd
import matplotlib.pyplot as plt

import pandas_datareader as data



#moving average for 100 values



ma100 = df.Close.rolling(100).mean()

ma100

# now n the below dataset you can see that the moving average for the first 100 terms is null but for the next, all value its have some
integer.



plt.figure(figsize=(12,6))

plt.plot(df.Close)

plt.plot(ma100,'r')
```



```
ma200 = df.Close.rolling(200).mean()

ma200

plt.figure(figsize=(12,6))

plt.plot(df.Close)

plt.plot(ma100,'r')

plt.plot(ma200,'y')
```

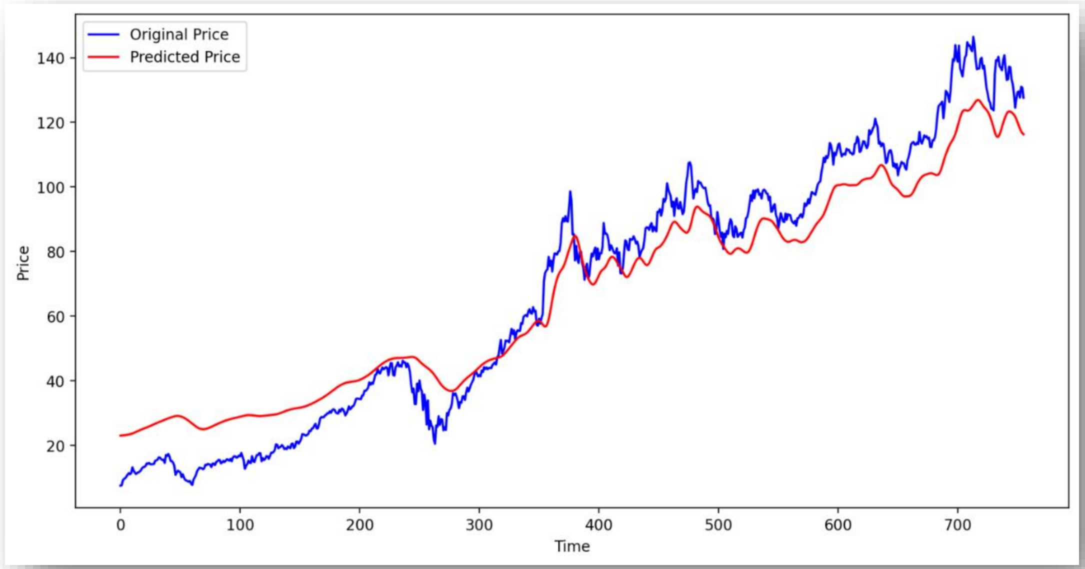


```
from sklearn.preprocessing import MinMaxScaler

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

data_training_array = scaler.fit_transform(data_training)

x_train = []
```

```
y_train = []

for i in range(100,data_training.shape[0]):
    x_train.append(data_training_array[i-100:i])
    y_train.append(data_training_array[i,0])

x_train ,y_train = np.array(x_train) , np.array(y_train)
# Machine learning MOdel

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

model = Sequential()
model.add(LSTM(units =50,activation='relu',return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(Dropout(0.2))

model.add(LSTM(units =60,activation='relu',return_sequences=True))
model.add(Dropout(0.3))


model.add(LSTM(units =80,activation='relu',return_sequences=True))

model.add(Dropout(0.4))


model.add(LSTM(units =120,activation='relu'))

model.add(Dropout(0.5))


model.add(Dense(units=1))


model.summary()
```

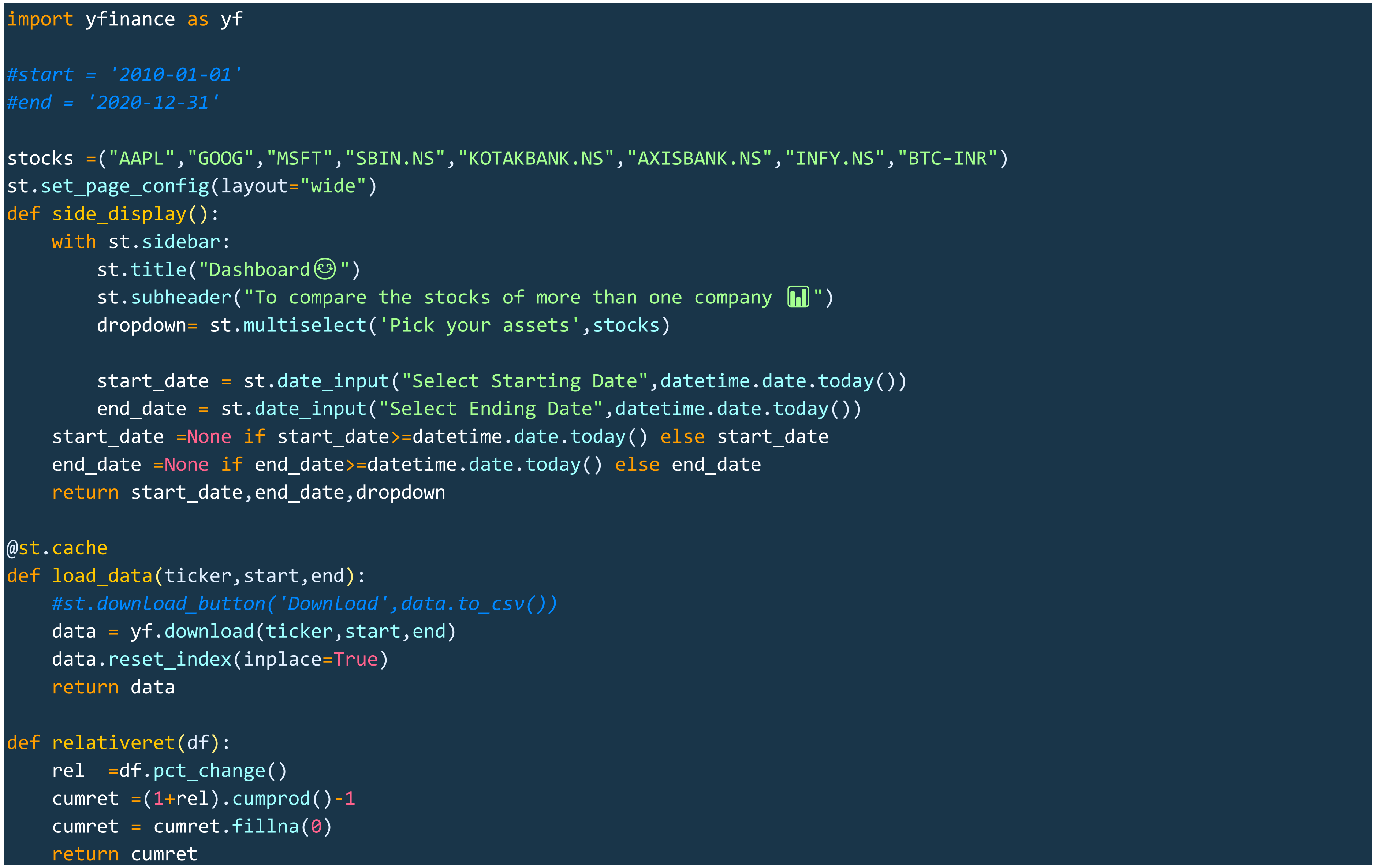
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 dropout (Dropout)           (None, 100, 50)           0

 lstm_1 (LSTM)               (None, 100, 60)           26640
 dropout_1 (Dropout)         (None, 100, 60)           0

 lstm_2 (LSTM)               (None, 100, 80)           45120

 dropout_2 (Dropout)         (None, 100, 80)           0

 lstm_3 (LSTM)               (None, 120)               96480

 dropout_3 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 178,761
Trainable params: 178,761
Non-trainable params: 0
```

```
model.compile(optimizer='adam',loss='mean_squared_error')

model.fit(x_train,y_train,epochs=50)

past_100_days = data_training.tail(100)

final_df = pd.concat([past_100_days,data_testing],ignore_index=True)

input_data = scaler.fit_transform(final_df)

x_test =[]

y_test =[]

for i in range(100,input_data.shape[0]):
    x_test.append(input_data[i-100:i])

    y_test.append(input_data[i,0])

y_predicted = model.predict(x_test)

scale_factor = 1/0.00988704

y_predicted = y_predicted * scale_factor

y_test = y_test*scale_factor

plt.figure(figsize=(12,6))

plt.plot(y_test,'b',label='Original Price')

plt.plot(y_predicted,'r',label='Predicted Price')

plt.xlabel('Time')

plt.ylabel('Price')

plt.legend()

plt.show()
```

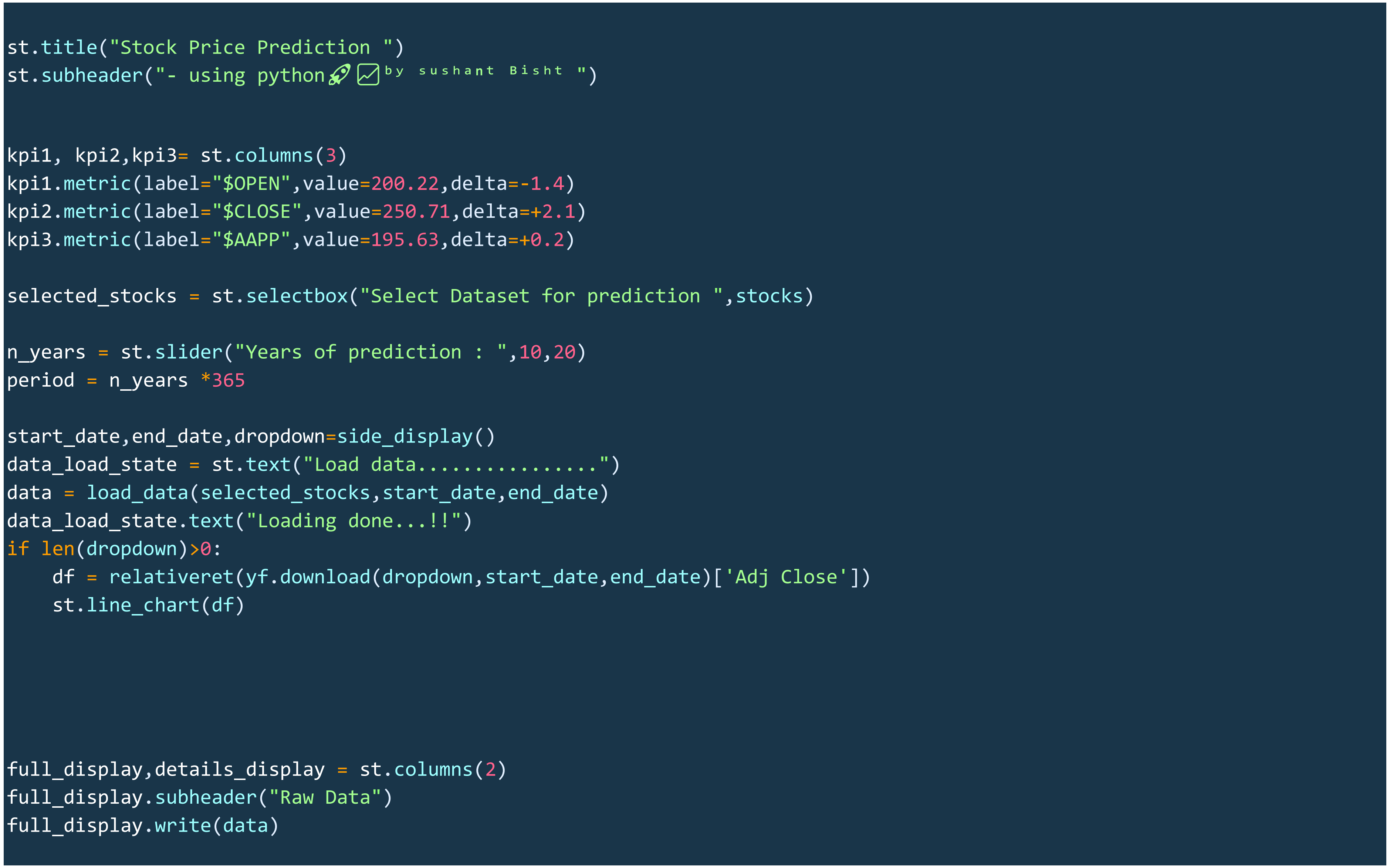


App.py(Front End):

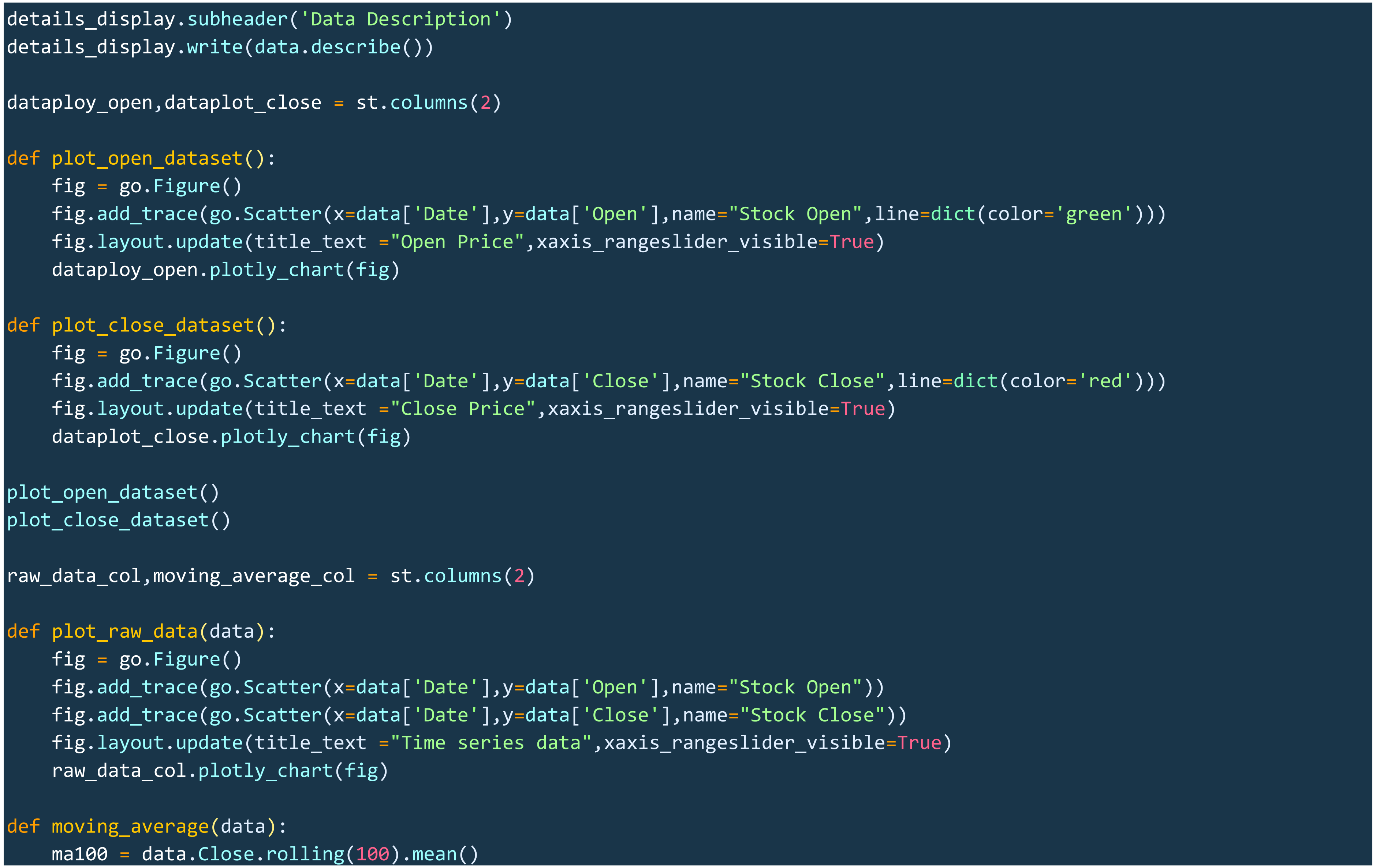
```
import datetime
from email.contentmanager import raw_data_manager
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pandas_datareader as data
from plotly import graph_objs as go
from keras.models import load_model
import streamlit as st
```



```python
import yfinance as yf

#start = '2010-01-01'
#end = '2020-12-31'

stocks =("AAPL","GOOG","MSFT","SBIN.NS","KOTAKBANK.NS","AXISBANK.NS","INFY.NS","BTC-INR")
st.set_page_config(layout="wide")
def side_display():
    with st.sidebar:
        st.title("Dashboard😊")
        st.subheader("To compare the stocks of more than one company 📊")
        dropdown= st.multiselect('Pick your assets',stocks)

        start_date = st.date_input("Select Starting Date",datetime.date.today())
        end_date = st.date_input("Select Ending Date",datetime.date.today())
    start_date =None if start_date>=datetime.date.today() else start_date
    end_date =None if end_date>=datetime.date.today() else end_date
    return start_date,end_date,dropdown

@st.cache
def load_data(ticker,start,end):
    #st.download_button('Download',data.to_csv())
    data = yf.download(ticker,start,end)
    data.reset_index(inplace=True)
    return data

def relativeret(df):
    rel  =df.pct_change()
    cumret =(1+rel).cumprod()-1
    cumret = cumret.fillna(0)
    return cumret
```



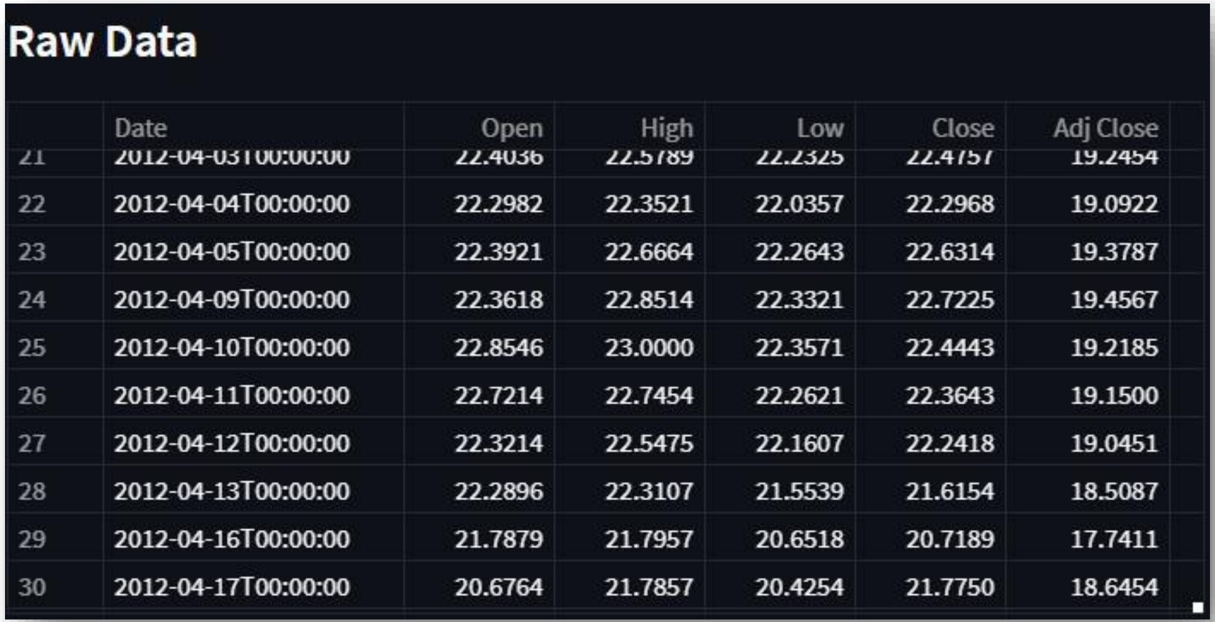
```python
st.title("Stock Price Prediction ")
st.subheader("- using python🚀📈by sushant Bisht ")


kpi1, kpi2,kpi3= st.columns(3)
kpi1.metric(label="$OPEN",value=200.22,delta=-1.4)
kpi2.metric(label="$CLOSE",value=250.71,delta=+2.1)
kpi3.metric(label="$AAPP",value=195.63,delta=+0.2)

selected_stocks = st.selectbox("Select Dataset for prediction ",stocks)

n_years = st.slider("Years of prediction : ",10,20)
period = n_years *365

start_date,end_date,dropdown=side_display()
data_load_state = st.text("Load data................")
data = load_data(selected_stocks,start_date,end_date)
data_load_state.text("Loading done...!!")
if len(dropdown)>0:
    df = relativeret(yf.download(dropdown,start_date,end_date)['Adj Close'])
    st.line_chart(df)





full_display,details_display = st.columns(2)
full_display.subheader("Raw Data")
full_display.write(data)
```



```python
details_display.subheader('Data Description')
details_display.write(data.describe())

dataploy_open,dataplot_close = st.columns(2)

def plot_open_dataset():
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=data['Date'],y=data['Open'],name="Stock Open",line=dict(color='green')))
    fig.layout.update(title_text ="Open Price",xaxis_rangeslider_visible=True)
    dataploy_open.plotly_chart(fig)

def plot_close_dataset():
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=data['Date'],y=data['Close'],name="Stock Close",line=dict(color='red')))
    fig.layout.update(title_text ="Close Price",xaxis_rangeslider_visible=True)
    dataplot_close.plotly_chart(fig)

plot_open_dataset()
plot_close_dataset()

raw_data_col,moving_average_col = st.columns(2)

def plot_raw_data(data):
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=data['Date'],y=data['Open'],name="Stock Open"))
    fig.add_trace(go.Scatter(x=data['Date'],y=data['Close'],name="Stock Close"))
    fig.layout.update(title_text ="Time series data",xaxis_rangeslider_visible=True)
    raw_data_col.plotly_chart(fig)

def moving_average(data):
    ma100 = data.Close.rolling(100).mean()
    ma200 = data.Close.rolling(200).mean()
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=data['Date'],y=data['Close'],name="Stock Close"))
    fig.add_trace(go.Scatter(x=data['Date'],y=ma100,name="Moving average for 100 values"))
    fig.add_trace(go.Scatter(x=data['Date'],y=ma200,name="Moving average for 200 values"))
    fig.layout.update(title_text ="Moving Average Data",xaxis_rangeslider_visible=True)
    moving_average_col.plotly_chart(fig)


plot_raw_data(data)
moving_average(data)
df = data.drop(['Date','Adj Close'], axis=1)

data_training = pd.DataFrame(df['Close'][0:int(len(df)*0.70)])
data_testing = pd.DataFrame(df['Close'][int(len(df)*0.70):int(len(df))])

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


model = load_model('keras_model.h5')

past_100_days = data_training.tail(100)
final_df = pd.concat([past_100_days,data_testing],ignore_index=True)
input_data = scaler.fit_transform(final_df)

x_test =[]
y_test =[]
for i in range(100,input_data.shape[0]):
    x_test.append(input_data[i-100:i])
```

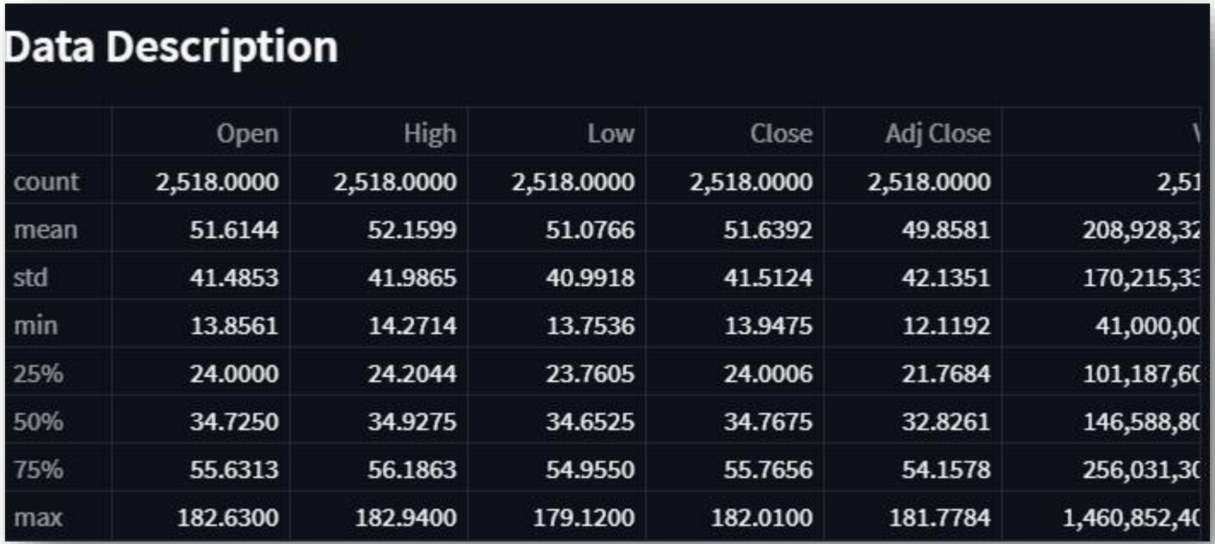
```python
    y_test.append(input_data[i,0])

x_test ,y_test = np.array(x_test) , np.array(y_test)
y_predicted = model.predict(x_test)
scale= scaler.scale_
scale_factor = 1/scale[0]
y_predicted = y_predicted * scale_factor
y_test = y_test*scale_factor

st.subheader("prediction")
fig2 =plt.figure(figsize=(12,6))
plt.plot(y_test,'b',label='Original Price')
plt.plot(y_predicted,'r',label='Predicted Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
st.pyplot(fig2)
```

Input and Output:



Raw Data

	Date	Open	High	Low	Close	Adj Close
21	2012-04-03T00:00:00	22.4036	22.5789	22.2325	22.4757	19.2454
22	2012-04-04T00:00:00	22.2982	22.3521	22.0357	22.2968	19.0922
23	2012-04-05T00:00:00	22.3921	22.6664	22.2643	22.6314	19.3787
24	2012-04-09T00:00:00	22.3618	22.8514	22.3321	22.7225	19.4567
25	2012-04-10T00:00:00	22.8546	23.0000	22.3571	22.4443	19.2185
26	2012-04-11T00:00:00	22.7214	22.7454	22.2621	22.3643	19.1500
27	2012-04-12T00:00:00	22.3214	22.5475	22.1607	22.2418	19.0451
28	2012-04-13T00:00:00	22.2896	22.3107	21.5539	21.6154	18.5087
29	2012-04-16T00:00:00	21.7879	21.7957	20.6518	20.7189	17.7411
30	2012-04-17T00:00:00	20.6764	21.7857	20.4254	21.7750	18.6454

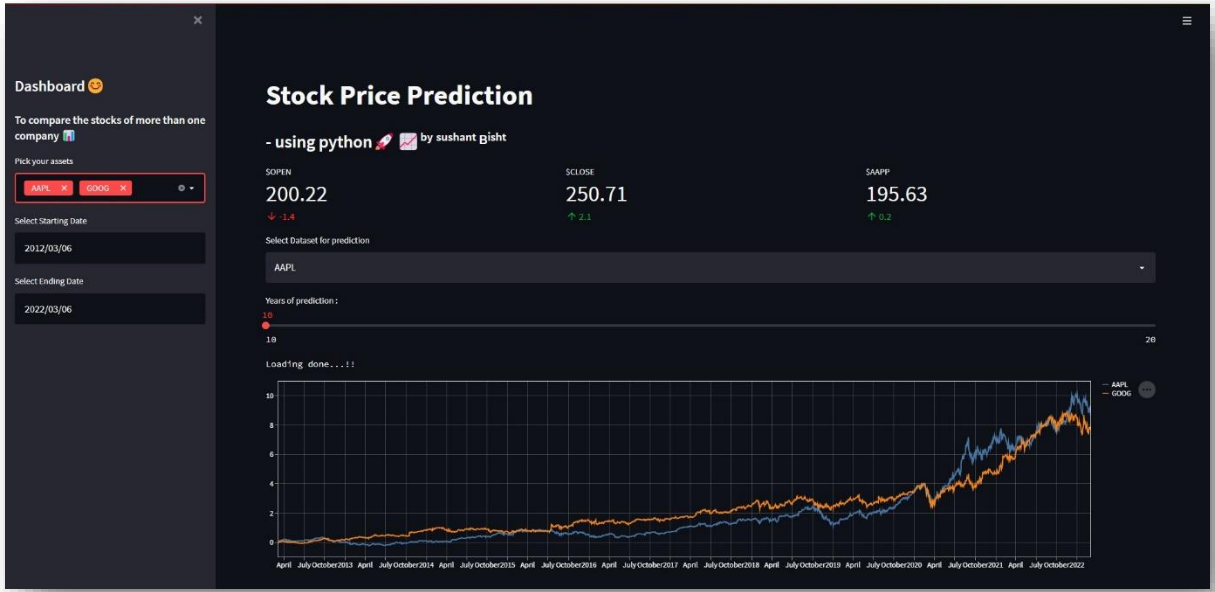


Data Description

	Open	High	Low	Close	Adj Close	V
count	2,518.0000	2,518.0000	2,518.0000	2,518.0000	2,518.0000	2,51
mean	51.6144	52.1599	51.0766	51.6392	49.8581	208,928,32
std	41.4853	41.9865	40.9918	41.5124	42.1351	170,215,33
min	13.8561	14.2714	13.7536	13.9475	12.1192	41,000,00
25%	24.0000	24.2044	23.7605	24.0006	21.7684	101,187,60
50%	34.7250	34.9275	34.6525	34.7675	32.8261	146,588,80
75%	55.6313	56.1863	54.9550	55.7656	54.1578	256,031,30
max	182.6300	182.9400	179.1200	182.0100	181.7784	1,460,852,40

App:

Home Page:



Dashboard 😊

To compare the stocks of more than one company 📊

Pick your assets: AAPL, GOOG

Select Starting Date: 2012/03/06

Select Ending Date: 2022/03/06

Stock Price Prediction

- using python 🚀 📈 by sushant gisht

$OPEN 200.22 ↓ -1.4

$CLOSE 250.71 ↑ 2.1

$AAPP 195.63 ↑ 0.2

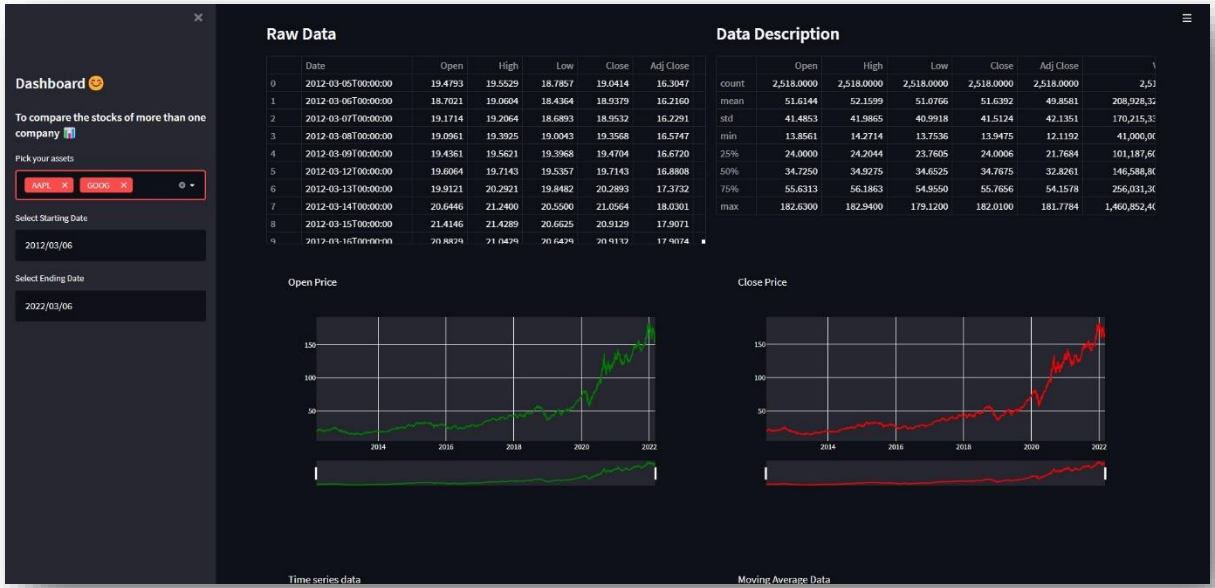
Select Dataset for prediction: AAPL

Years of prediction : 10

```
Loading done...!!
```

10
20
AAPL
GOOG
April July October2013 April July October2014 April July October2015 April July October2016 April July October2017 April July October2018 April July October2019 April July October2020 April July October2021 April July October2022

Page after selecting data:



Dashboard 😊

To compare the stocks of more than one company 📊

Pick your assets

AAPL × GOOG ×

Select Starting Date

2012/03/06

Select Ending Date

2022/03/06

Raw Data

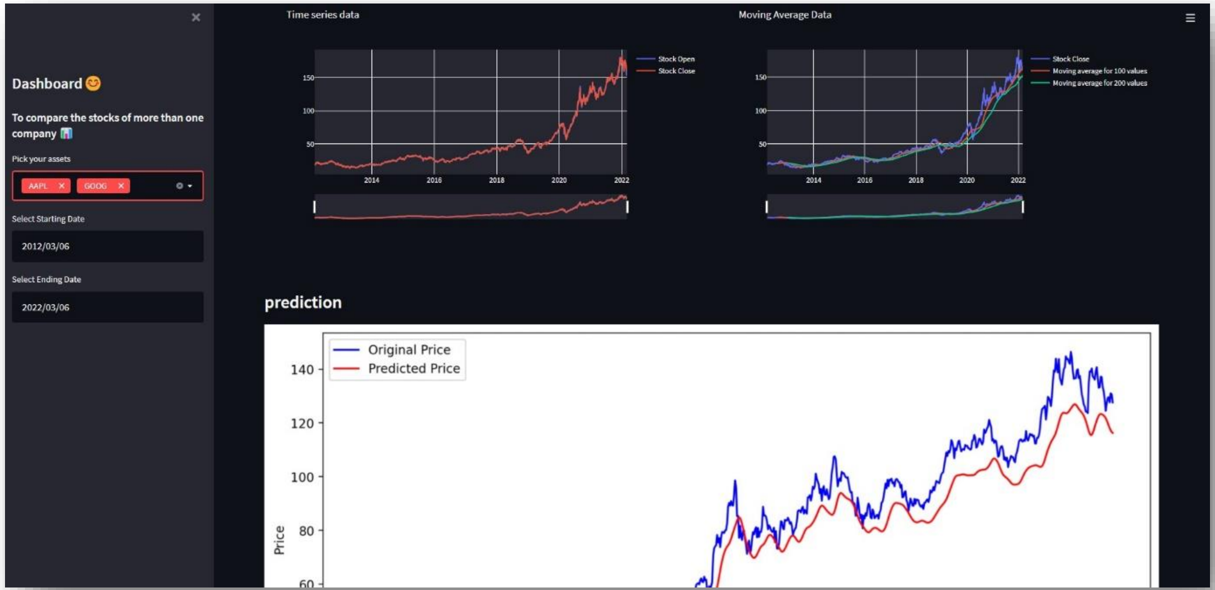
	Date	Open	High	Low	Close	Adj Close
0	2012-03-05T00:00:00	19.4793	19.5529	18.7857	19.0414	16.3047
1	2012-03-06T00:00:00	18.7021	19.0604	18.4364	18.9379	16.2160
2	2012-03-07T00:00:00	19.1714	19.2064	18.6893	18.9532	16.2291
3	2012-03-08T00:00:00	19.0961	19.3925	19.0043	19.3568	16.5747
4	2012-03-09T00:00:00	19.4361	19.5621	19.3968	19.4704	16.6720
5	2012-03-12T00:00:00	19.6064	19.7143	19.5357	19.7143	16.8808
6	2012-03-13T00:00:00	19.9121	20.2921	19.8482	20.2893	17.3732
7	2012-03-14T00:00:00	20.6446	21.2400	20.5500	21.0564	18.0301
8	2012-03-15T00:00:00	21.4146	21.4289	20.6625	20.9129	17.9071
9	2012-03-16T00:00:00	20.8829	21.0429	20.6429	20.9132	17.9074

Data Description

	Open	High	Low	Close	Adj Close	V
count	2,518.0000	2,518.0000	2,518.0000	2,518.0000	2,518.0000	2,51
mean	51.6144	52.1599	51.0766	51.6392	49.8581	208,928,32
std	41.4853	41.9865	40.9918	41.5124	42.1351	170,215,33
min	13.8561	14.2714	13.7536	13.9475	12.1192	41,000,00
25%	24.0000	24.2044	23.7605	24.0006	21.7684	101,187,60
50%	34.7250	34.9275	34.6525	34.7675	32.8261	146,588,80
75%	55.6313	56.1863	54.9550	55.7656	54.1578	256,031,30
max	182.6300	182.9400	179.1200	182.0100	181.7784	1,460,852,40

Time series data

Moving Average Data



Dashboard 😊

To compare the stocks of more than one company 📊

Pick your assets

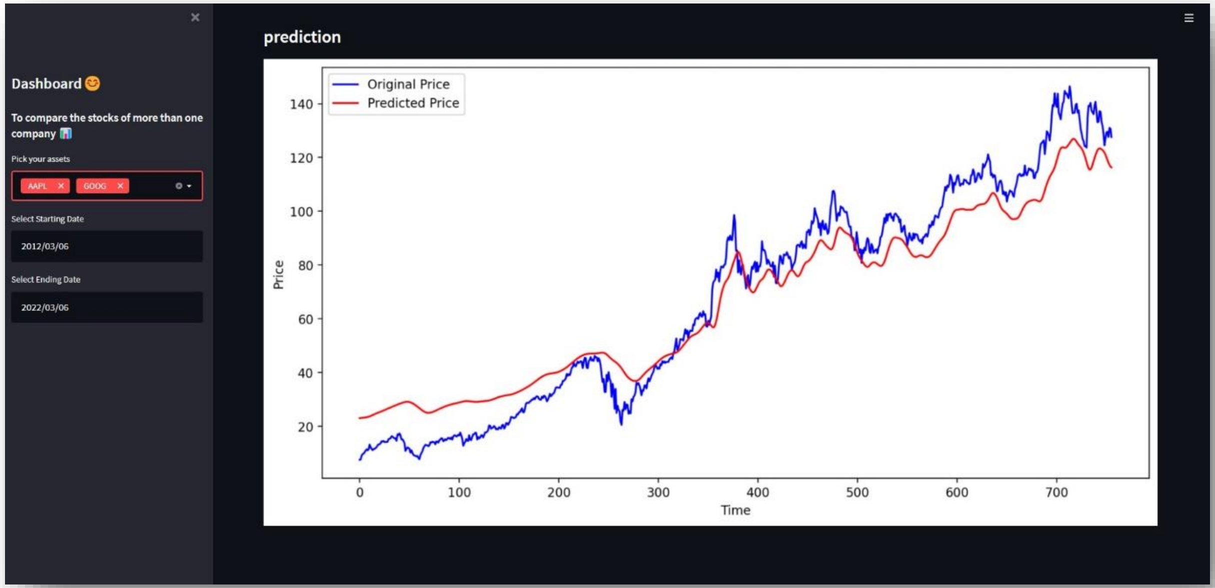
AAPL ✕ GOOG ✕

Select Starting Date

2012/03/06

Select Ending Date

2022/03/06


Dashboard 😊
To compare the stocks of more than one company 📊
Pick your assets
AAPL ×
GOOG ×
Select Starting Date
2012/03/06
Select Ending Date
2022/03/06
prediction
Original Price
Predicted Price
Price
Time
140
120
100
80
60
40
20
0
100
200
300
400
500
600
700

CONCLUSION AND FUTURE WORK

Conclusion:

In this project, we are predicting the closing stock price of any given organization, we developed a web application for predicting close stock price using LSTM algorithms for prediction. We have applied datasets belonging to Google, Apple, Microsoft, Infosys, and Reliance Stocks and achieved above 95% accuracy for these datasets.

Future work:

- We want to extend this application for predicting cryptocurrency trading.
- We want to add sentiment analysis for better analysis.

REFERENCES

[1] Yahoo Finance ww.yahoo.com

2.1 Literature Review Summary

Table 2.1: Literature review summary

Year and citation	Article Title	Purpose of the study	Tools/ Software used	Comparison of technique done	Source (Journal/ Conference)	Findings	Data set (if used)	Evaluation parameters
2010								