PROJECT REPORT ON STOCK MARKET ANALYSIS

Submitted by

ABHISHEK BHARDWAJ (230320525002)

DEEPAK POKHARIYA (230320525014)

SUMIT PANDEY (230320525046)

UTKARSH TEWARI (230320525048)

Under the Guidance of

MS. SIDHIDATRI NAYAK

CDAC-NOIDA

Candidate Declaration

I solely declare that the report titled “Stock Market Analysis” is a bonafied record of work carried by us. Submitted partial fulfillment of requirement for the award of Post graduation diploma in Big Data Analytics under the guidance of Ms. Sidhidhatri Nayak Mam

The matter embodied in this report has not been submitted for the award of any other degree.

Acknowledgement

I would like to thank Ms. Sidhidhatri Nayak Mam for her help through this project. Her suggestions are the keys to the successful completion of this project and to understand the basic of analysis and design of algorithms which is the most important factor behind implementing efficient code. I would also like to thank Mr. Shivam Pandey Sir and Ms. Preeti Bhardwaj Mam for giving us suggestions to improve our project.

Abstract

Stock market plays a pivotal role in financial aspect of the nation's growth, but stock market is highly volatile and complex in nature. It is affected by significant political issues, analyst calls, news articles, company's future plans of expansions and growth and many more.

On Every business day, millions of traders invest in stock market. Most of these investors lose money and others gain. However, considering any trading day, loss or gain is absolutely inconsistent. The demand to predict stock prices are extremely high hence is the need for stock market analysis.

Our problem statement is Analysis of Stock Market and Prediction using LSTM.

Our aim is to predict the opening price of a stock for up to 30 or 60 days.

TABLE OF CONTENTS

1. INTRODUCTION

The world is flooded with information and we are only concerned about the information relevant to us. The information available can be classified based on various fields like engineering, arts, science, history, sports, geography and economics. In such situations it becomes important for us to boil down to the information relevant to us. Especially when it comes to the field of finance, the information available is not well organized and they are just information of present state. The stock market can be viewed as a platform where almost all major economic transactions in the world occur at a dynamic rate called the stock value which is based on the market equilibrium. A correct prediction of this stock trend beforehand can earn huge profits. In this project, one approach is implemented to achieve the prediction. The approach used for prediction is LSTM (Long Short Term Memory) on Time-Series Analysis. It has the ability to handle new data robustly. In this model, based on the historical datasets that complement with present day's stock price behavior. This results in a forecast for 30 or 60 days stock trend of the open feature. Stock market can be analyzed based on fundamental analysis which is about financial results of the particular company which we are analyzing. Stock market runs on a well-known theory known as "Theory of Demand and Supply". It states demand is always inversely proportional to supply. Hence, as demand increases buyers take over the stock market and market turns bullish.

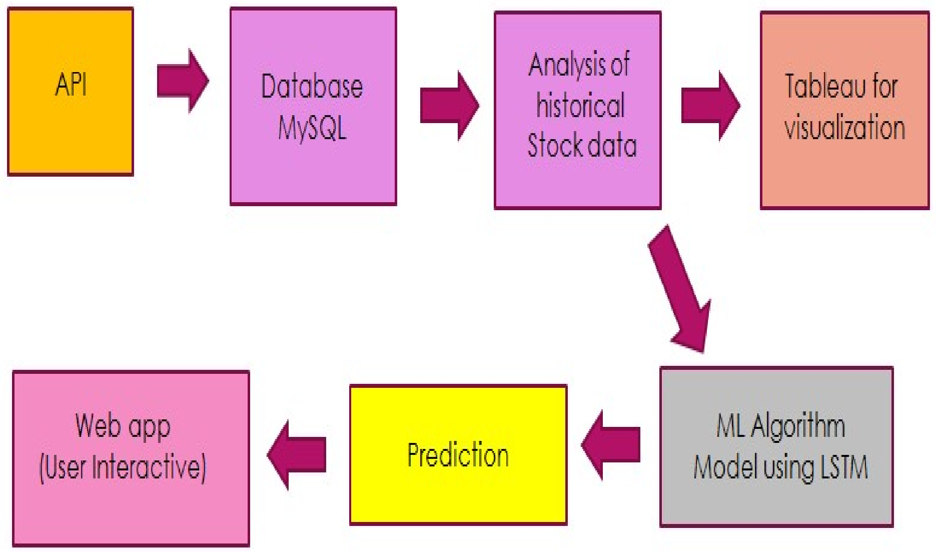
So, in our project we explore the past, make sense of the present, and peek into the future of stock markets using innovative techniques and tools.By the end of this report, you'll gain a deeper understanding of how data-driven analysis can empower investors and financial professionals.Let's embark on this exciting exploration of the Stock Market Analysis project.

2. Project Overview

Our Stock Market Analysis Project is a comprehensive effort to decode the complexities of stock markets and harness their predictive power. This project is driven by the goal of creating a prototype that can assist investors, traders, and financial analysts in making informed decisions. Key components of the project include data collection, pre-processing, data fetching, visualization, and predictive modelling.

- **Data Collection:** We have retrieved historical stock data from a company with an open-source APIs to ensure our analysis is based on accurate and up-to-date information.
- **Database Integration:** We have integrated MySQL database into our project to efficiently store and manage the data fetched from the API.
- **LSTM for Stock Prediction:** The LSTM (Long Short-Term Memory) model, a type of recurrent neural network (RNN), is employed to forecast stock prices.
- **Visualization with Tableau:** The power of visualization is harnessed through Tableau, enabling us to present data in an intuitive and insightful manner.
- **Web App using Streamlit:** We have made an interactive user-friendly web page to find out the stock market prediction for 30 or 60 days using Streamlit.

3. WORK FLOW OF THE PROJECT


API
Database MySQL
Analysis of historical Stock data
Tableau for visualization
ML Algorithm Model using LSTM
Prediction
Web app (User Interactive)

4. DATA COLLECTION/FETCH

Data collection is the foundational step of our Stock Market Analysis Project.

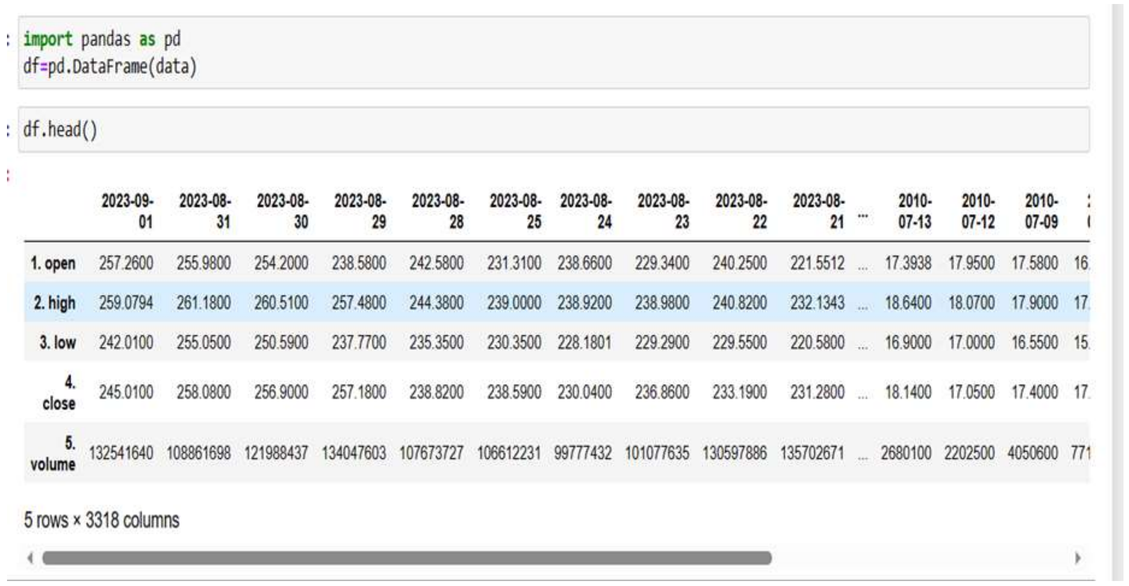
To ensure the accuracy and timeliness of our analysis, we obtained historical stock market data from online APIs.

APIs (Application Programming Interfaces) provided us with direct access to real-time and historical stock data from reputable sources.

The use of APIs allowed us to gather data efficiently and systematically, ensuring that our analysis is based on the most current information available.

Through these APIs, we retrieved data from a diverse range of companies and stock exchanges, offering a comprehensive view of the market.

This robust data collection process forms the backbone of our project, enabling us to perform in-depth analysis and provide valuable insights into the stock market's behaviour.



```
: import pandas as pd
  df=pd.DataFrame(data)
```

```
: df.head()
```

	2023-09-01	2023-08-31	2023-08-30	2023-08-29	2023-08-28	2023-08-25	2023-08-24	2023-08-23	2023-08-22	2023-08-21	...	2010-07-13	2010-07-12	2010-07-09	
1. open	257.2600	255.9800	254.2000	238.5800	242.5800	231.3100	238.6600	229.3400	240.2500	221.5512	...	17.3938	17.9500	17.5800	16.
2. high	259.0794	261.1800	260.5100	257.4800	244.3800	239.0000	238.9200	238.9800	240.8200	232.1343	...	18.6400	18.0700	17.9000	17.
3. low	242.0100	255.0500	250.5900	237.7700	235.3500	230.3500	228.1801	229.2900	229.5500	220.5800	...	16.9000	17.0000	16.5500	15.
4. close	245.0100	258.0800	256.9000	257.1800	238.8200	238.5900	230.0400	236.8600	233.1900	231.2800	...	18.1400	17.0500	17.4000	17.
5. volume	132541640	108861698	121988437	134047603	107673727	106612231	99777432	101077635	130597886	135702671	...	2680100	2202500	4050600	771

5 rows × 3318 columns

5. DATA PRE-PROCESSING AND DATABASE INTEGRATION

Data pre-processing and storage are essential steps in our Stock Market Analysis project to ensure that we work with clean and structured data.

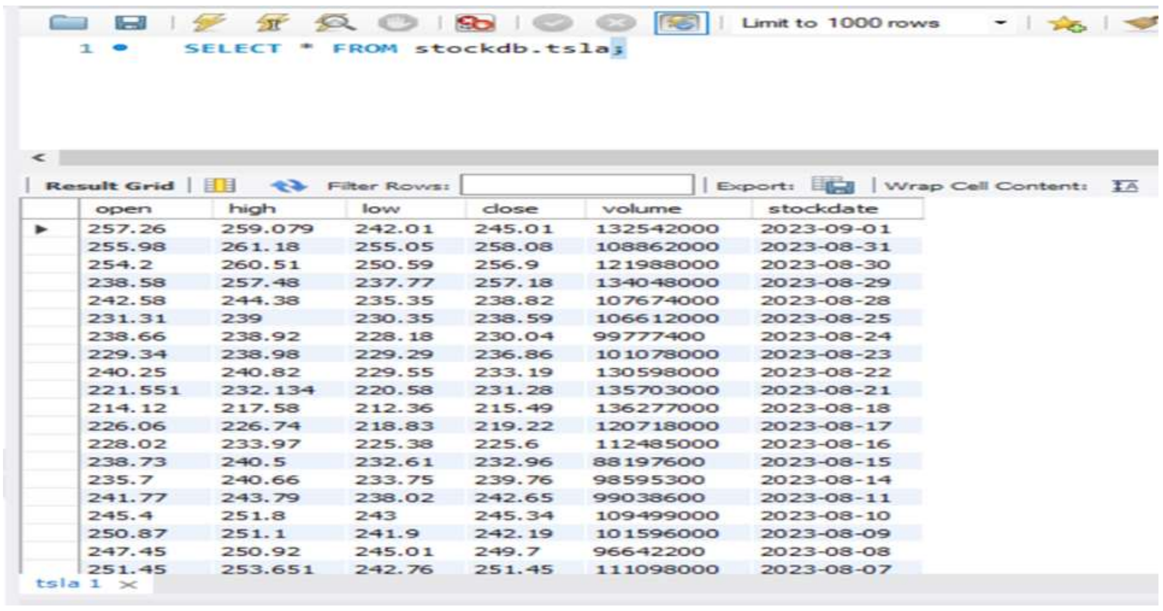
Data pre-processing includes cleaning and structuring data to ensure it's ready for analysis

To achieve this, we implemented a database integration using Python and MySQL through the pymysql library.

This code establishes a connection to the MySQL database, creates the 'stockdb' database if it doesn't exist, and inserts data into tables named after specific stocks.

The data_fetch and data_fetchAll methods in the class of our source code allow us to retrieve data based on specific criteria, such as date ranges or all available data.

This integration of Python and MySQL serves as the foundation for our subsequent data analysis and predictive phases.

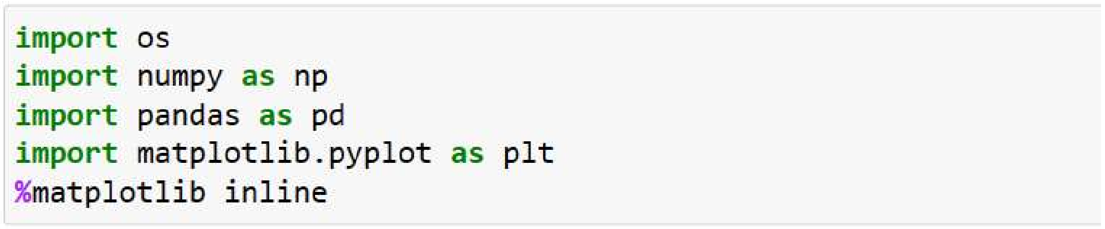


```
SELECT * FROM stockdb.tsla;
```

open	high	low	close	volume	stockdate
257.26	259.079	242.01	245.01	132542000	2023-09-01
255.98	261.18	255.05	258.08	108862000	2023-08-31
254.2	260.51	250.59	256.9	121988000	2023-08-30
238.58	257.48	237.77	257.18	134048000	2023-08-29
242.58	244.38	235.35	238.82	107674000	2023-08-28
231.31	239	230.35	238.59	106612000	2023-08-25
238.66	238.92	228.18	230.04	99777400	2023-08-24
229.34	238.98	229.29	236.86	101078000	2023-08-23
240.25	240.82	229.55	233.19	130598000	2023-08-22
221.551	232.134	220.58	231.28	135703000	2023-08-21
214.12	217.58	212.36	215.49	136277000	2023-08-18
226.06	226.74	218.83	219.22	120718000	2023-08-17
228.02	233.97	225.38	225.6	112485000	2023-08-16
238.73	240.5	232.61	232.96	88197600	2023-08-15
235.7	240.66	233.75	239.76	98595300	2023-08-14
241.77	243.79	238.02	242.65	99038600	2023-08-11
245.4	251.8	243	245.34	109499000	2023-08-10
250.87	251.1	241.9	242.19	101596000	2023-08-09
247.45	250.92	245.01	249.7	96642200	2023-08-08
251.45	253.651	242.76	251.45	111098000	2023-08-07

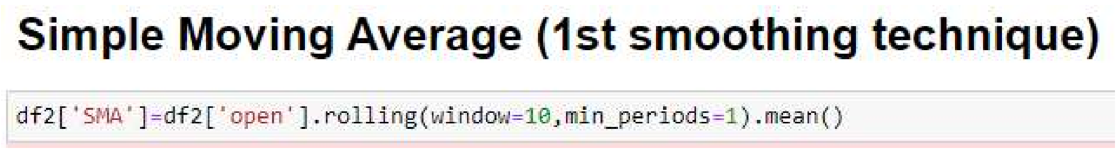
6. MODELING

In order to build our model, we used long short term memory (LSTM). Recurrent neural networks (RNNs) have been improved by LSTMs. RNNs are comparable to how people learn. We import pandas for loading and altering datasets, numpy for performing scientific computations, and matplotlib for creating graphs.



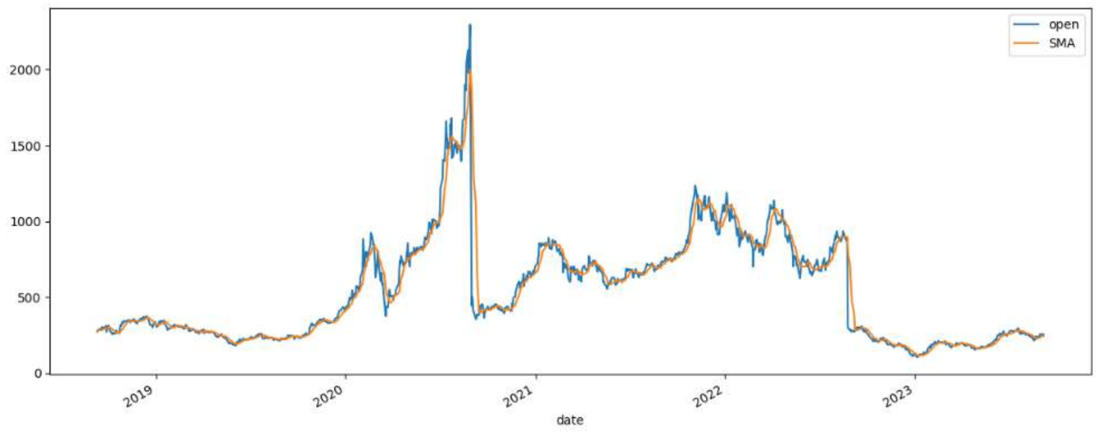
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

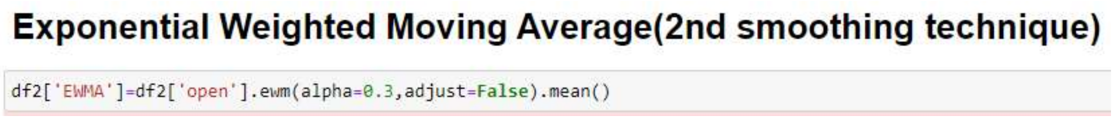
We have applied different technical indicators like simple moving average and exponential moving average for smoothing of the noisy data and curve which will be helpful in good and accurate prediction of the stock price.



Simple Moving Average (1st smoothing technique)

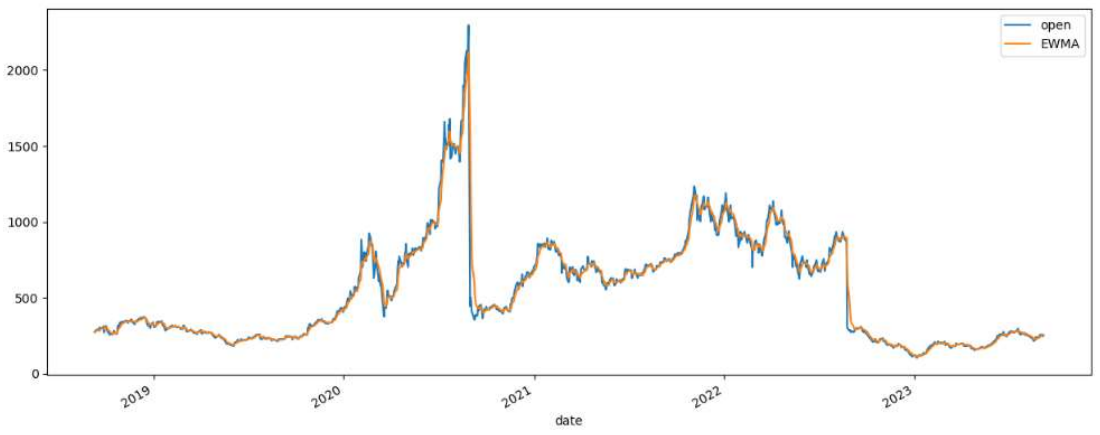
```
df2['SMA']=df2['open'].rolling(window=10,min_periods=1).mean()
```





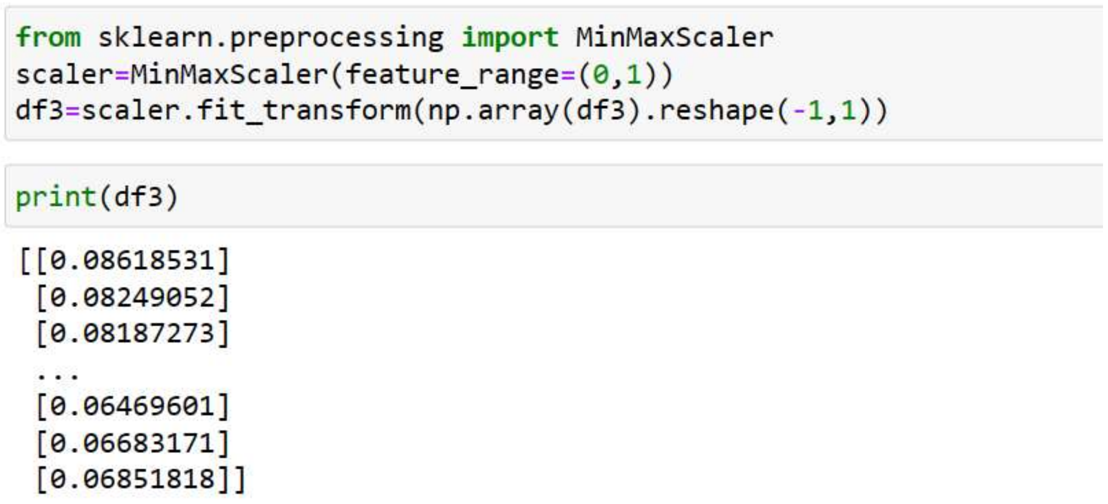
Exponential Weighted Moving Average(2nd smoothing technique)

```
df2['EWMA']=df2['open'].ewm(alpha=0.3,adjust=False).mean()
```



Then after comparing both the graphs we concluded that exponential moving average is giving a good smoothen curve so we have taken that data for further modelling.

The process of normalization, which improves the performance of our model, involves converting the values of the dataset's numeric columns to a standard scale. We employ Scikit-Learn's MinMaxScaler with values between zero and one to scale the training dataset.

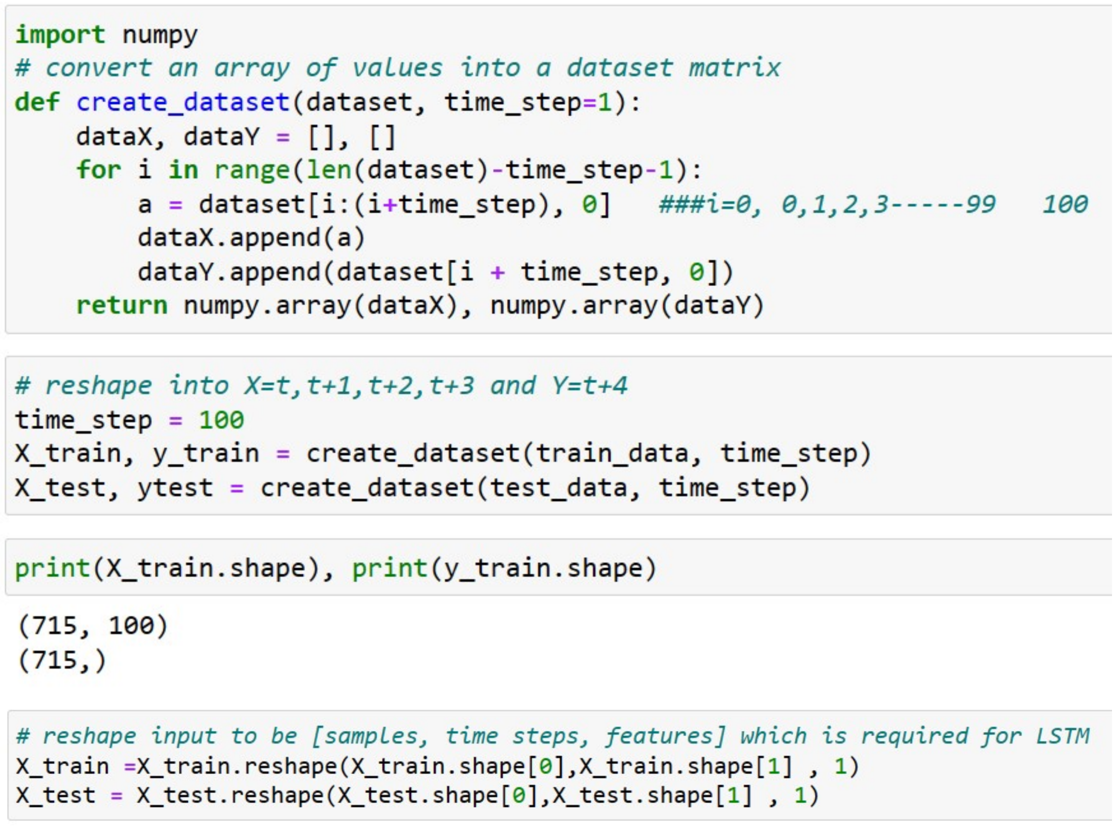


```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df3=scaler.fit_transform(np.array(df3).reshape(-1,1))
```

```
print(df3)
```

```
[[0.08618531]
 [0.08249052]
 [0.08187273]
 ...
 [0.06469601]
 [0.06683171]
 [0.06851818]]
```

The LSTM model should receive our data in the form of a 3D array. Before converting the data into an array with numpy, we first create the matrix or dataset which will be fed to our model considering 100 timesteps,(100 datapoints as input and 1 datapoint as a output). The data is then transformed into a 3D array.



```
import numpy
# convert an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

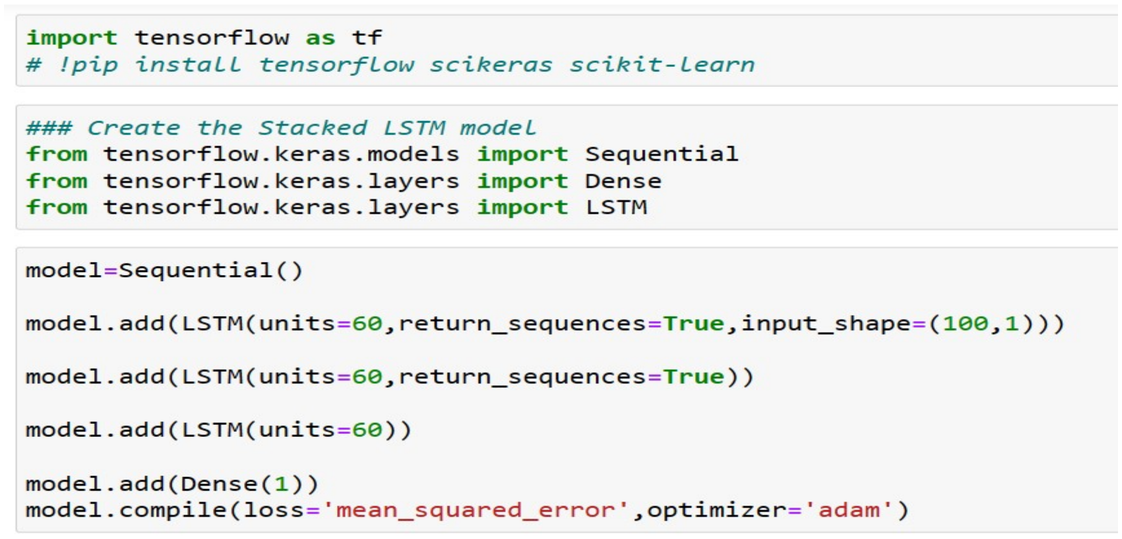
```
# reshape into X=t,t+1,t+2,t+3 and Y=t+4
time_step = 100
X_train, y_train = create_dataset(train_data, time_step)
X_test, ytest = create_dataset(test_data, time_step)
```

```
print(X_train.shape), print(y_train.shape)
```

```
(715, 100)
(715,)
```

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)
```

We must first import a few Keras modules: Sequential for initializing the neural network, LSTM for adding the LSTM layer, Dropout for preventing overfitting with dropout layers, and Dense for adding a densely connected neural network layer before we can start developing the LSTM.



```
import tensorflow as tf
# !pip install tensorflow scikeras scikit-learn
```

```
### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

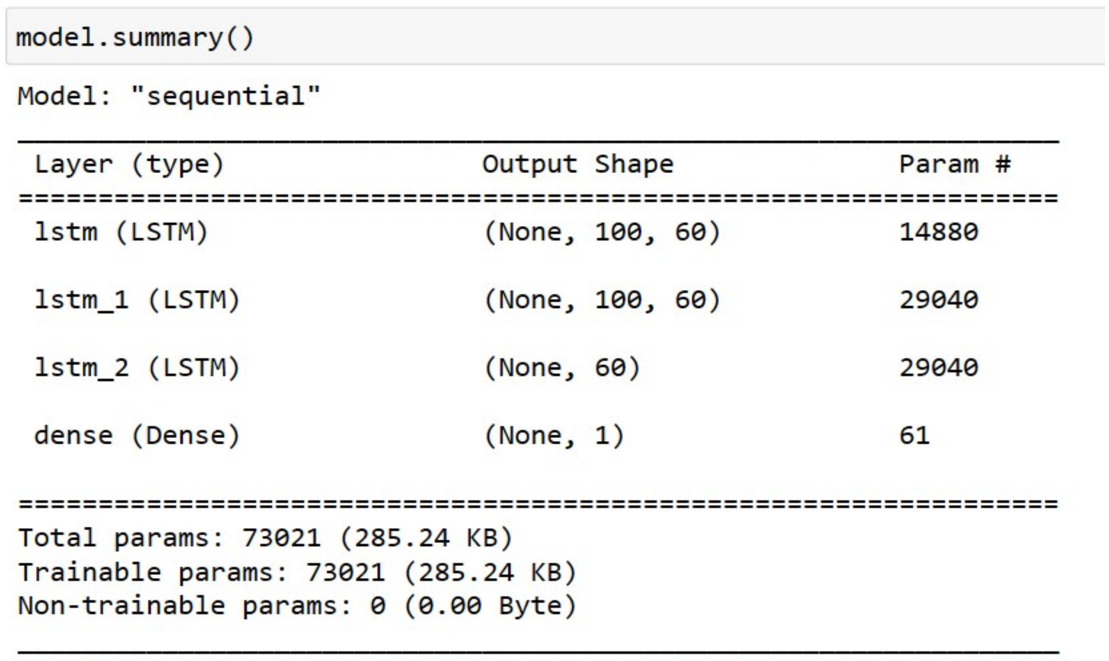
```
model=Sequential()

model.add(LSTM(units=60,return_sequences=True,input_shape=(100,1)))

model.add(LSTM(units=60,return_sequences=True))

model.add(LSTM(units=60))

model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```



```
model.summary()
```

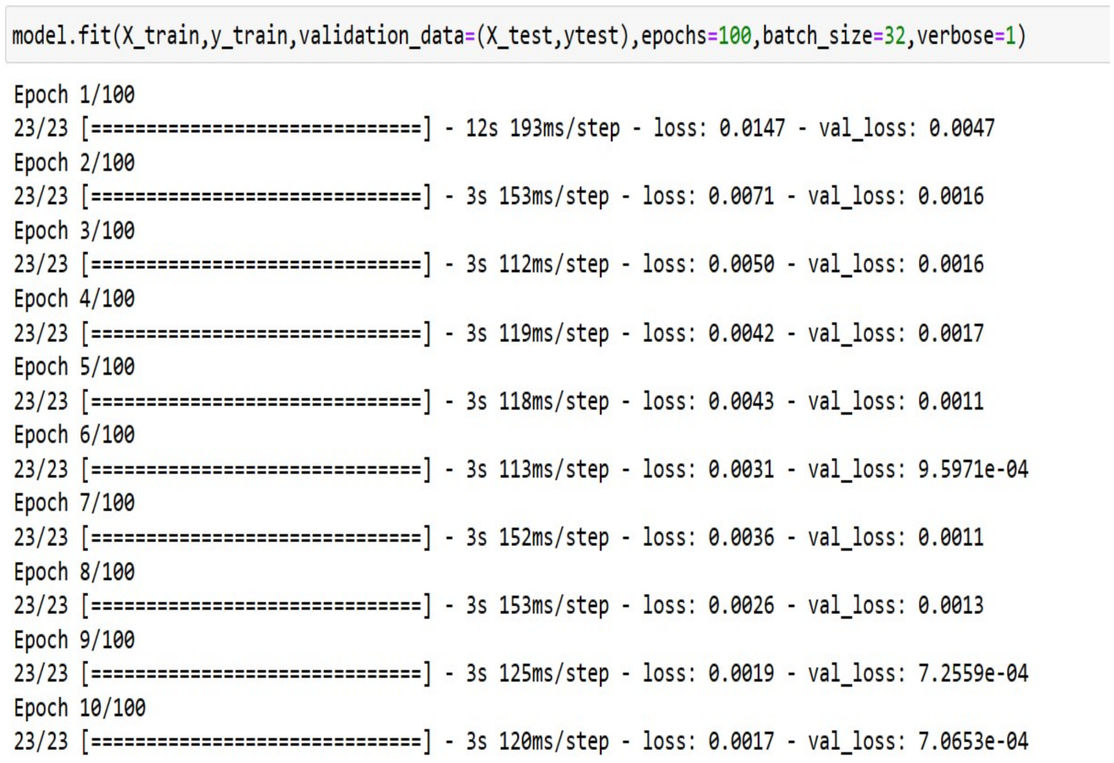
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 60)           14880

 lstm_1 (LSTM)               (None, 100, 60)           29040

 lstm_2 (LSTM)               (None, 60)                29040

 dense (Dense)               (None, 1)                 61

=================================================================
Total params: 73021 (285.24 KB)
Trainable params: 73021 (285.24 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



```
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=32,verbose=1)
```

```
Epoch 1/100
23/23 [==============================] - 12s 193ms/step - loss: 0.0147 - val_loss: 0.0047
Epoch 2/100
23/23 [==============================] - 3s 153ms/step - loss: 0.0071 - val_loss: 0.0016
Epoch 3/100
23/23 [==============================] - 3s 112ms/step - loss: 0.0050 - val_loss: 0.0016
Epoch 4/100
23/23 [==============================] - 3s 119ms/step - loss: 0.0042 - val_loss: 0.0017
Epoch 5/100
23/23 [==============================] - 3s 118ms/step - loss: 0.0043 - val_loss: 0.0011
Epoch 6/100
23/23 [==============================] - 3s 113ms/step - loss: 0.0031 - val_loss: 9.5971e-04
Epoch 7/100
23/23 [==============================] - 3s 152ms/step - loss: 0.0036 - val_loss: 0.0011
Epoch 8/100
23/23 [==============================] - 3s 153ms/step - loss: 0.0026 - val_loss: 0.0013
Epoch 9/100
23/23 [==============================] - 3s 125ms/step - loss: 0.0019 - val_loss: 7.2559e-04
Epoch 10/100
23/23 [==============================] - 3s 120ms/step - loss: 0.0017 - val_loss: 7.0653e-04
```

```
### Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

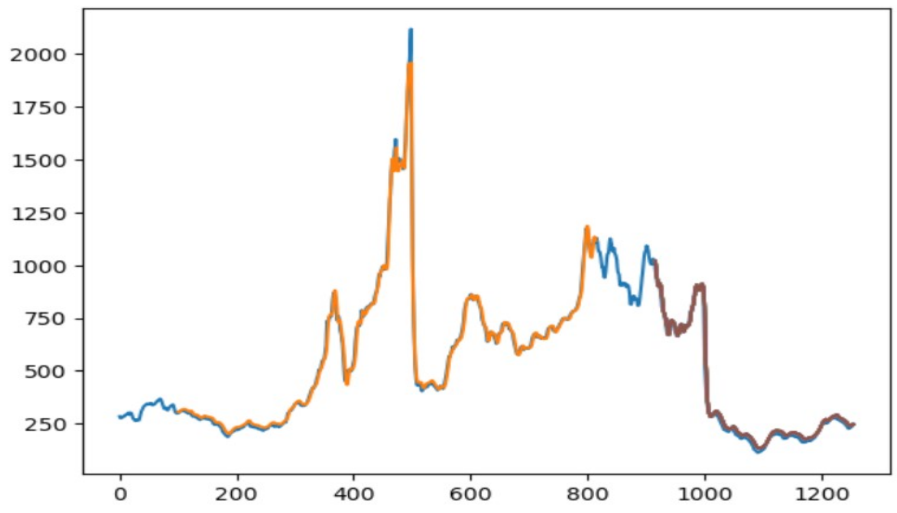
```
691.3372790171147
```

```
### Test Data RMSE
math.sqrt(mean_squared_error(ytest,test_predict))
```

```
446.40616643515847
```

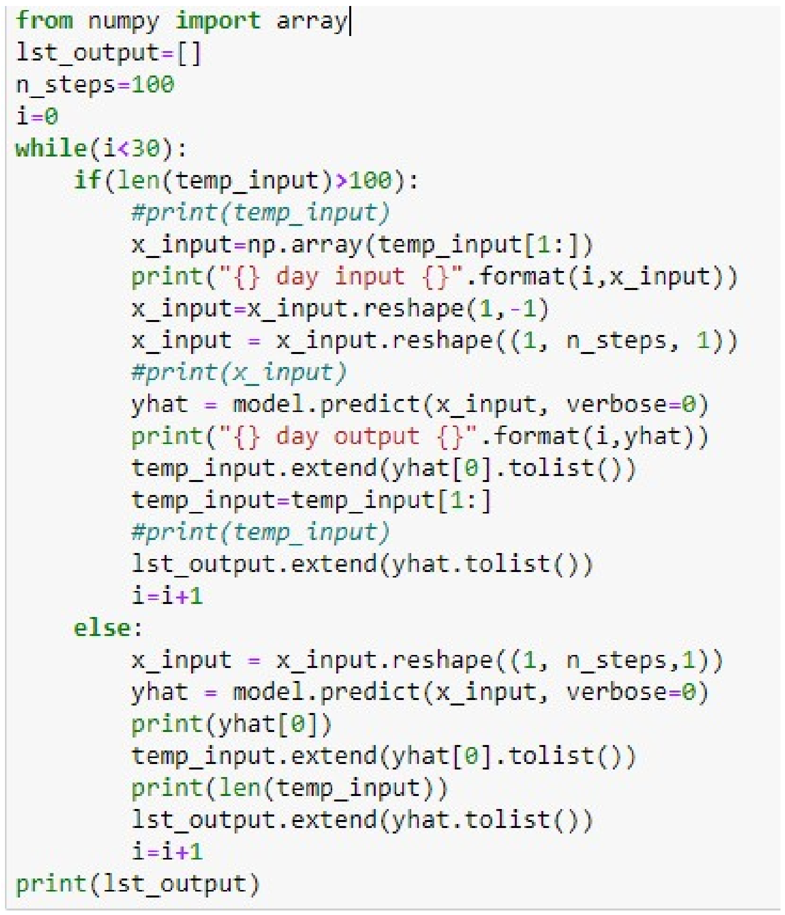
We can use matplotlib to visualize the outcome of our forecasted stock price and the real stock price after all these processes.

```
### Plotting
# shift train predictions for plotting
look_back=100
trainPredictPlot = numpy.empty_like(df3)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(df3))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```



The blue graph is indicating the true actual data, the orange line is the prediction made for the train data and the maroon line is the prediction made for the test data.

After that we have written the code to predict the stock prices for 30 days or 60 days.



```
from numpy import array
lst_output=[]
n_steps=100
i=0
while(i<30):
    if(len(temp_input)>100):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1
print(lst_output)
```

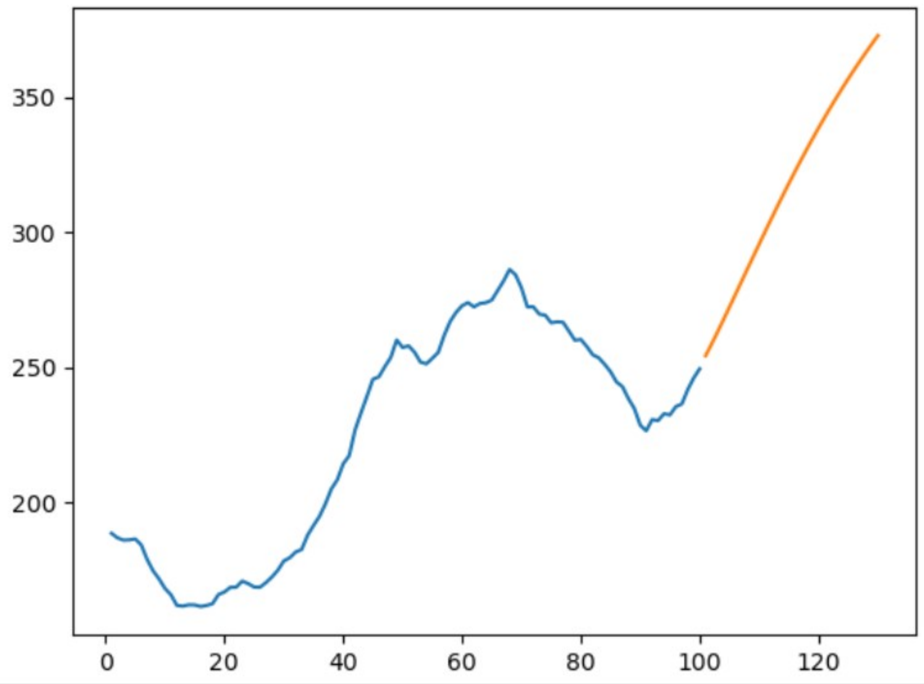
7. DATA PREDICTION

In this phase of our Stock Market Analysis Project, we employ advanced machine learning techniques to predict future stock price, the factors are taken into account for change in the open price of a particular company. We performed analysis on obtained data to establish relation between our predicted output and the input feature i.e. 'open' feature of the stock data. We utilized a custom-built predictive model to forecast stock price movements

After collection of predicted future data of 30 or 60 days as given by the user by the model, the value is then plotted on the graph in addition to the original data or actual data.

```
plt.plot(day_new,scaler.inverse_transform(df3[len(df3)-100:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
```

```
[<matplotlib.lines.Line2D at 0x24276344280>]
```



8. USER INTERACTIVE WEB APP

To provide an enhanced user experience and make our Stock Market Analysis Project more accessible, we've incorporated Streamlit, a Python library for creating interactive web applications.

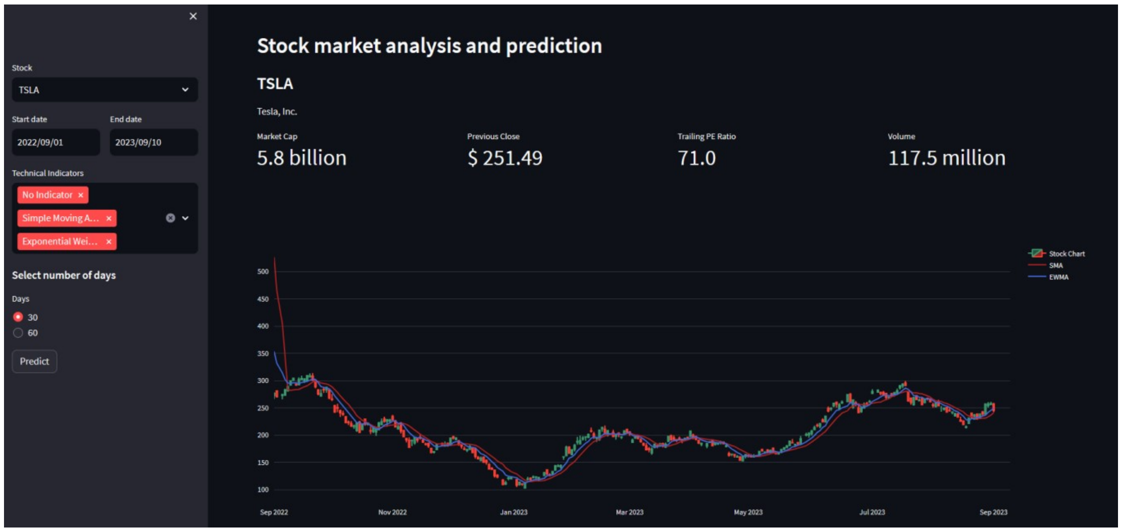
Stock Matrix Visualization: Our Streamlit-powered web application includes a dynamic stock matrix that allows users to select and visualize stocks.

Date Range Selection: Users can specify custom date ranges to focus on specific periods of interest. This flexibility enables detailed analysis of stock trends over specific time frames.

Technical Indicators: In this project, two technical indicators are implemented to achieve the prediction. These technical indicators are implemented to anticipate future changes in the stock trends and to do smoothing of the noisy data and curve. The technical indicators implemented in this stock market prediction tool will back the predictability strongly.

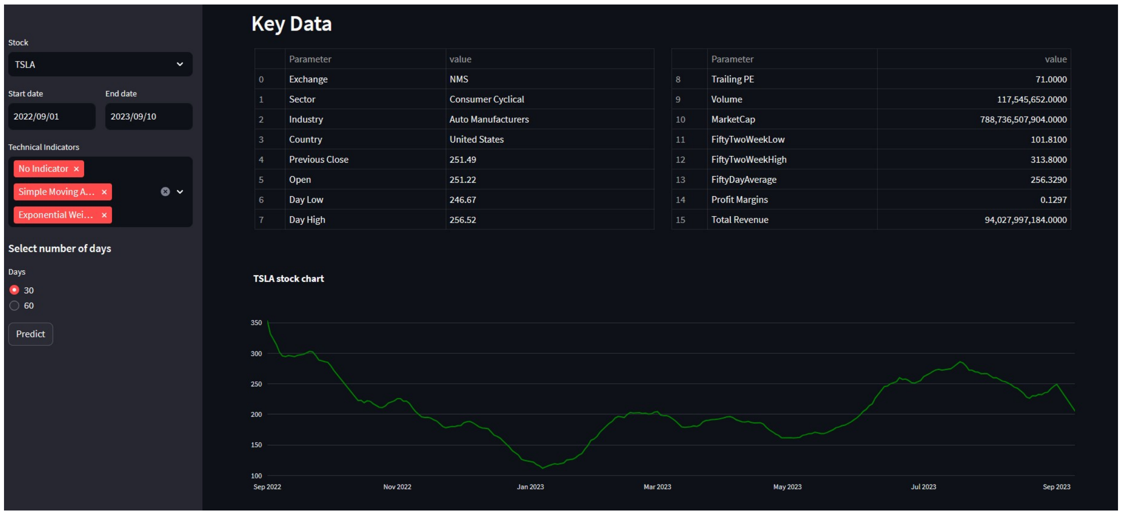
Two Prediction Options (30 and 60 Days): We've integrated predictive modelling directly into our web application, offering users the ability to forecast stock prices for two intervals.

Web Application


Stock: TSLA
Start date: 2022/09/01
End date: 2023/09/10
Technical Indicators: No Indicator, Simple Moving A…, Exponential Wei…
Select number of days
Days: [✓ 30] 60

Stock market analysis and prediction
TSLA
Tesla, Inc.
Market Cap: 5.8 billion
Previous Close: $ 251.49
Trailing PE Ratio: 71.0
Volume: 117.5 million

Key Data of Stock



Stock: TSLA

Start date: 2022/09/01

End date: 2023/09/10

Technical Indicators: No Indicator, Simple Moving A…, Exponential Wei…

Select number of days

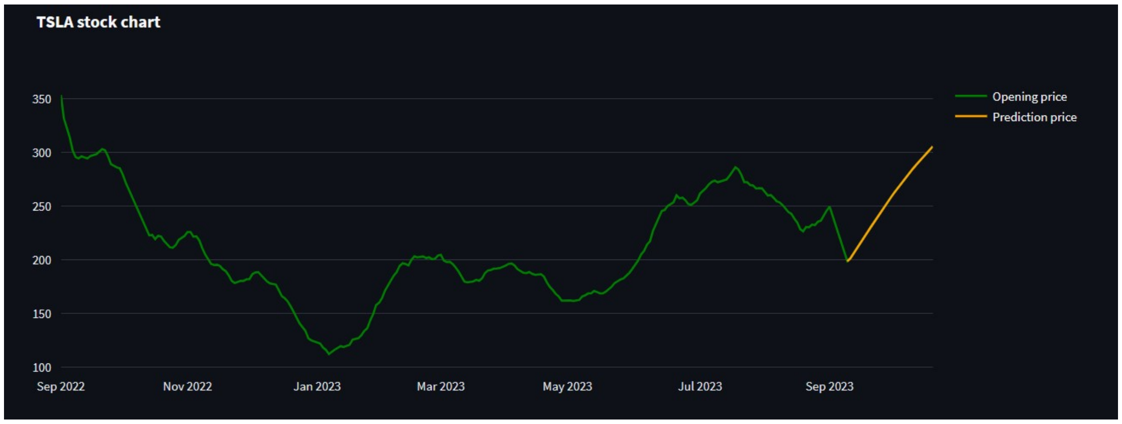
Days: [✓ 30] 60

Key Data

	Parameter	value
0	Exchange	NMS
1	Sector	Consumer Cyclical
2	Industry	Auto Manufacturers
3	Country	United States
4	Previous Close	251.49
5	Open	251.22
6	Day Low	246.67
7	Day High	256.52

	Parameter	value
8	Trailing PE	71.0000
9	Volume	117,545,652.0000
10	MarketCap	788,736,507,904.0000
11	FiftyTwoWeekLow	101.8100
12	FiftyTwoWeekHigh	313.8000
13	FiftyDayAverage	256.3290
14	Profit Margins	0.1297
15	Total Revenue	94,027,997,184.0000

Prediction



9. CONCLUSION AND FUTURE WORK

We considered LSTM model as our benchmark, our model, which we have discussed works optimally for all sectors of stocks. Based on the model results we have found a root mean squared error of 446 (Approx). Based on the model described we can also ensure we can detect wrong decisions and get the investor out of stock market by providing a stop loss at the time of triggering decision. The model used for educational purpose only which shows improved results by the prediction in this report. Prediction accuracies based on these models are relatively consistent across all sectors and stocks which we chose.

For future work we can also add more indicators like moving average convergence/divergence, relative strength index and also apply different models for the prediction of the stock market price.

10. REFERENCES AND BIBLIOGRAPHY

- https://www.datacamp.com/tutorial/lstm-python-stock-market
- https://www.geeksforgeeks.org/python-stock-data-visualisation/
- https://python.plainenglish.io/stock-analysis-dashboard-with-python-366d431c8721?gi=c2c7df96e9f6
- https://medium.com/analytics-vidhya/developing-an-interactive-dashboard-for-value-investment-with-python-dash-and-pandas-version-2-2c2616542b5c
- https://subscription.packtpub.com/book/data/9781789618518/2/ch02lvl1sec15/building-an-interactive-dashboard-for-ta
- https://machinelearningmastery.com/exponential-smoothing-for-time-series-forecasting-in-python/
- https://medium.com/@srv96/smoothing-techniques-for-time-series-data-91cccfd008a2#:~:text=Smoothing%20techniques%20are%20kinds%20of,economy%20compared%20to%20unsmoothed%20data.
- https://home.ubalt.edu/ntsbarsh/business-stat/otherapplets/ForecaSmo.htm
- https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning