STOCK PRICE PREDICTION

PROJECT
BY
SK.HASEENA BEGUM & J.BHUVANASREE
S180720 &S180073
CSE-2E

PREFACE

About the project:

Stock price prediction using machine learning helps you discover the future value of company stock and other financial assets traded on an exchanges….

The entire idea of predicting stock prices is to gain significant profits…

INTRODUCTION Stock

Market:

Stock market is a place where buying and selling of shares happen for publicly listed companies.

Stock exchange is the mediator that allows buying and selling of shares.

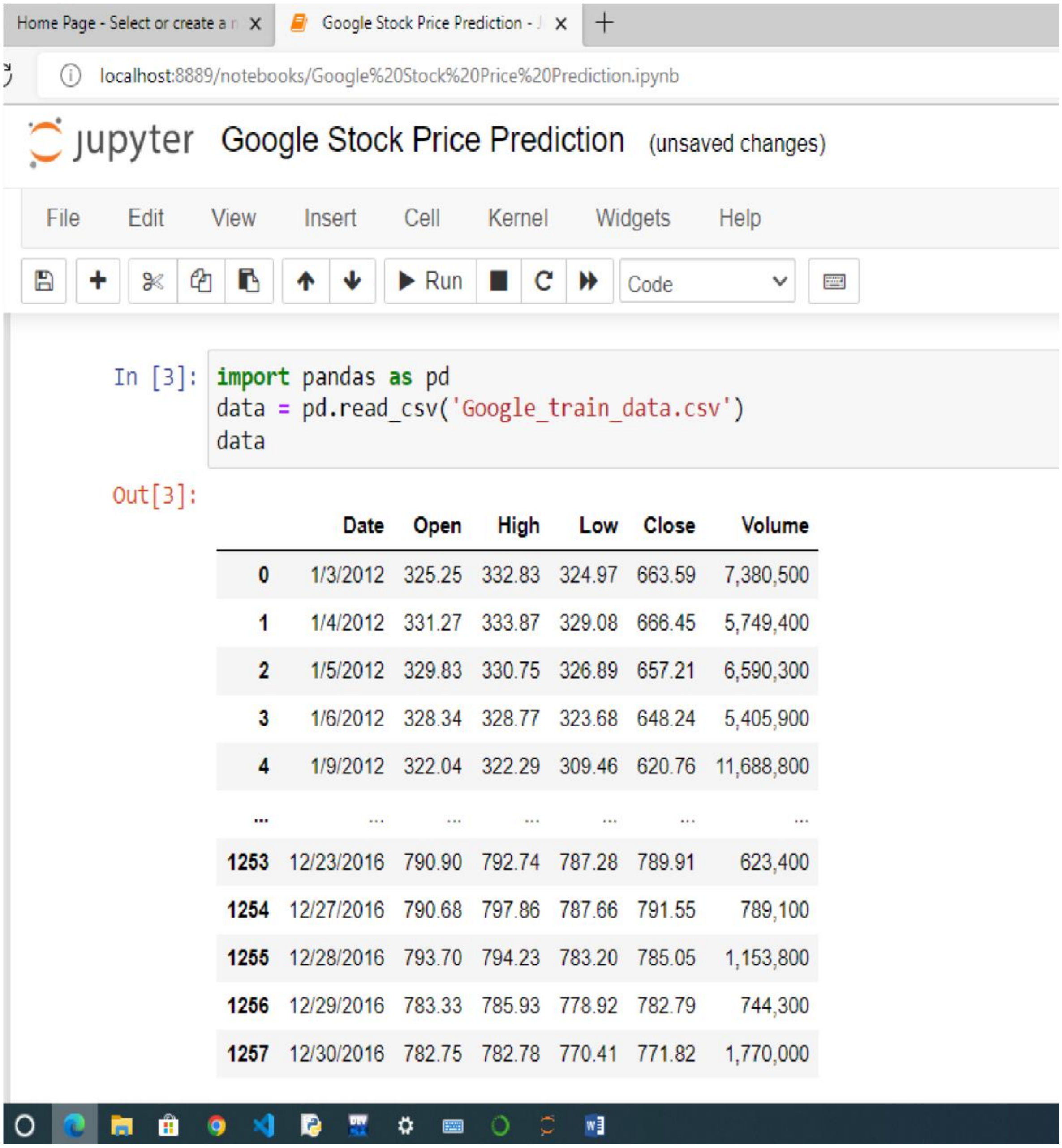
Importance of stock market:

- Helps companies to raise capital
- Helps create personal wealth
- Serves as an indicator of the state of the economy

- Helps to increase investment

Analysis

Taken Columns: High, Open, Close, Low, Volume..




```
import pandas as pd
data = pd.read_csv('Google_train_data.csv')
data
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800
...	...	...	...	...	...	...
1253	12/23/2016	790.90	792.74	787.28	789.91	623,400
1254	12/27/2016	790.68	797.86	787.66	791.55	789,100
1255	12/28/2016	793.70	794.23	783.20	785.05	1,153,800
1256	12/29/2016	783.33	785.93	778.92	782.79	744,300
1257	12/30/2016	782.75	782.78	770.41	771.82	1,770,000

High: High is the highest price at which the stock trading during the period.

Open: The price at which started trading when the market open on a particular date.

Close: Close refers to price of an individual stock and stock exchange closed market on the day. It represents last buy and sell order executed between two traders.

Low: Lowest price in the period.

Volume: Volume is the total amount of trading activity during the period of the time.

CODE:

Importing the libraries

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import MinMaxScaler
        from keras.models import Sequential
        from keras.layers import Dense,LSTM,Dropout
```

Load the training dataset:

```
In [3]: data = pd.read_csv('Google_train_data.csv')
        data.head()
```

Out[3]:

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

Exploring data analysis:

```
[3] data = pd.read_csv('Google_train_data.csv')
    data.head()
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

```
data.describe()
```

	Open	High	Low	Close
count	1149.000000	1149.000000	1149.000000	1149.000000
mean	531.604517	535.816449	526.879608	674.775527
std	158.412156	159.593385	157.008123	112.582696
min	279.120000	281.210000	277.220000	491.200000
25%	391.560000	394.700000	388.230000	571.580000
50%	536.350000	539.600000	531.540000	673.690000
75%	689.980000	698.200000	683.650000	761.680000
max	816.680000	816.680000	805.140000	922.160000

```
[4] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    1258 non-null   object
 1   Open    1258 non-null   float64
 2   High    1258 non-null   float64
 3   Low     1258 non-null   float64
 4   Close   1258 non-null   object
 5   Volume  1258 non-null   object
dtypes: float64(3), object(3)
memory usage: 59.1+ KB
```

Checking null values

```
[43] data.isnull()
```

	Date	Open	High	Low	Close	Volume
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
1253	False	False	False	False	False	False
1254	False	False	False	False	False	False
1255	False	False	False	False	False	False
1256	False	False	False	False	False	False
1257	False	False	False	False	False	False

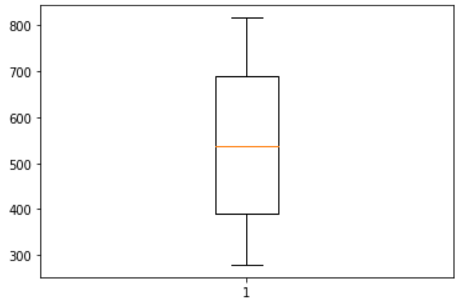
1149 rows × 6 columns

Data Visualization:

Box plot:

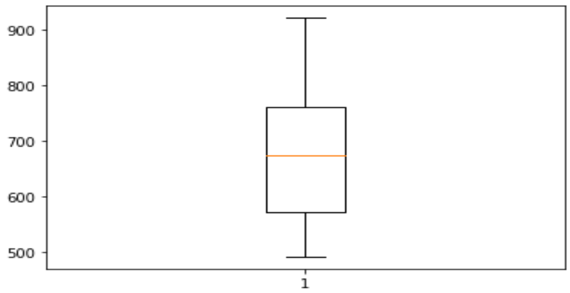
```
[21] plt.boxplot(data["Open"])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f67a8985890>,
  <matplotlib.lines.Line2D at 0x7f67a8831e10>],
 'caps': [<matplotlib.lines.Line2D at 0x7f67a8836390>,
  <matplotlib.lines.Line2D at 0x7f67a88368d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f67a8f94450>],
 'medians': [<matplotlib.lines.Line2D at 0x7f67a8836e50>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f67a883b3d0>],
 'means': []}
```



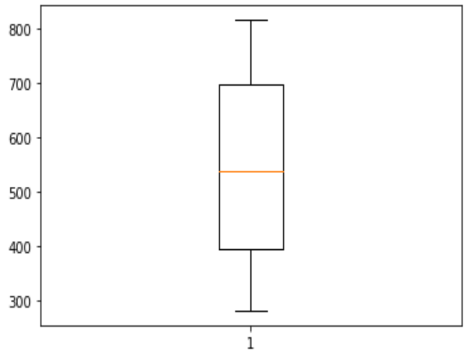

```
[23] plt.boxplot(data["Close"])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f67a8782d10>,
  <matplotlib.lines.Line2D at 0x7f67a87862d0>],
 'caps': [<matplotlib.lines.Line2D at 0x7f67a8786810>,
  <matplotlib.lines.Line2D at 0x7f67a8786d50>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f67a8782990>],
 'medians': [<matplotlib.lines.Line2D at 0x7f67a878e2d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f67a878e850>],
 'means': []}
```



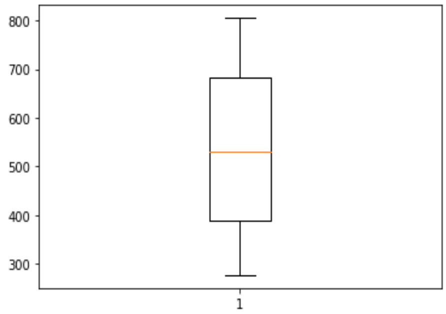
```python
[25] plt.boxplot(data["High"])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f67a863f4d0>,
  <matplotlib.lines.Line2D at 0x7f67a863fa50>],
 'caps': [<matplotlib.lines.Line2D at 0x7f67a863ff90>,
  <matplotlib.lines.Line2D at 0x7f67a8645510>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f67a863f150>],
 'medians': [<matplotlib.lines.Line2D at 0x7f67a8645a50>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f67a8645fd0>],
 'means': []}
```




```python
[26] plt.boxplot(data["Low"])
```

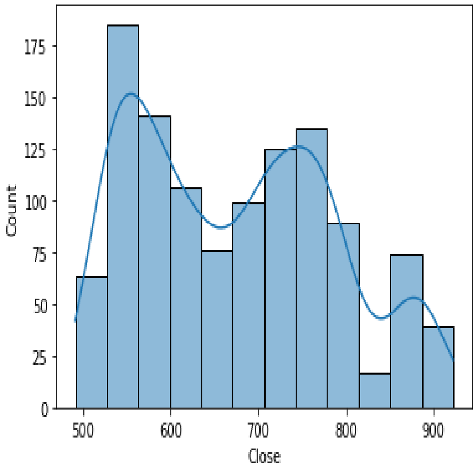
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f67a85aac50>,
  <matplotlib.lines.Line2D at 0x7f67a85b0210>],
 'caps': [<matplotlib.lines.Line2D at 0x7f67a85b0750>,
  <matplotlib.lines.Line2D at 0x7f67a85b0c90>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f67a85aa8d0>],
 'medians': [<matplotlib.lines.Line2D at 0x7f67a85b5210>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f67a85b5790>],
 'means': []}
```



Seaborn:

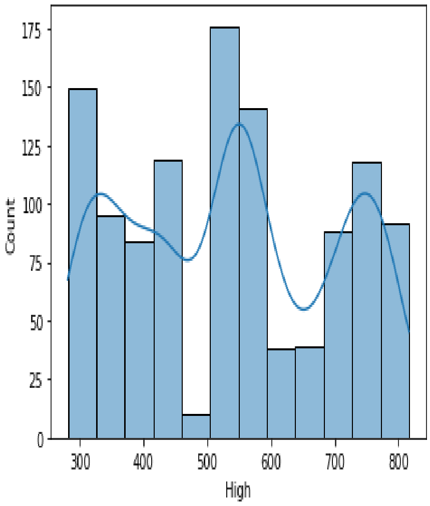
```
[40] sns.histplot(data.Close,kde=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f67a7e3af90>




```
sns.histplot(data.High,kde=True)
```

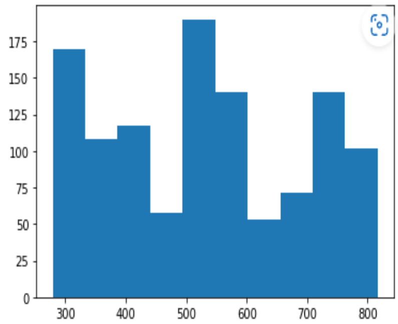
<matplotlib.axes._subplots.AxesSubplot at 0x7f67a7da2590>



Histograms:

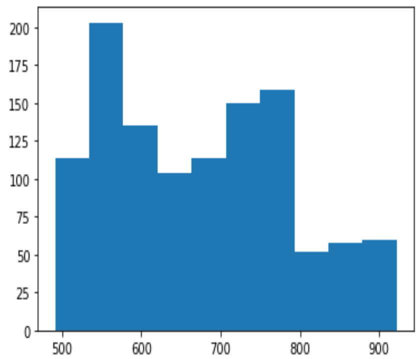
```
[28] plt.hist(data["Open"])
```

```
(array([170., 108., 117.,  58., 190., 140.,  53.,  71., 140., 102.]),
 array([279.12 , 332.876, 386.632, 440.388, 494.144, 547.9  , 601.656,
        655.412, 709.168, 762.924, 816.68 ]),
 <a list of 10 Patch objects>)
```



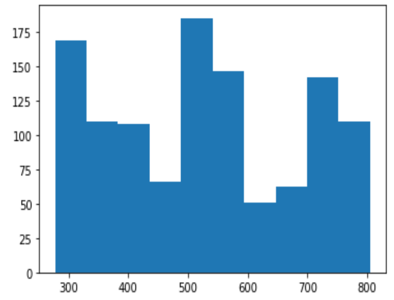

```
[30] plt.hist(data["Close"])
```

```
(array([114., 203., 135., 104., 114., 150., 159.,  52.,  58.,  60.]),
 array([491.2  , 534.296, 577.392, 620.488, 663.584, 706.68 , 749.776,
        792.872, 835.968, 879.064, 922.16 ]),
 <a list of 10 Patch objects>)
```



```
[31] plt.hist(data["Low"])

(array([169., 110., 108.,  66., 185., 146.,  51.,  62., 142., 110.]),
 array([277.22 , 330.012, 382.804, 435.596, 488.388, 541.18 , 593.972,
        646.764, 699.556, 752.348, 805.14 ]),
 <a list of 10 Patch objects>)
```



Creating x_train and y_trian data structures:

```
[8]: X_train = []
     y_train = []

     for i in range (60,1149): #60 : timestep // 1149 : length of the data
         X_train.append(trainData[i-60:i,0])
         y_train.append(trainData[i,0])

     X_train,y_train = np.array(X_train),np.array(y_train)
```

Reshape the dataset

```
In [9]: X_train = np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))
        X_train.shape

Out[9]: (1089, 60, 1)
```

Building the model by adding layers

```
In [10]: model = Sequential()

         model.add(LSTM(units=100, return_sequences = True, input_shape =(X_train.shape[1],1)))
         model.add(Dropout(0.2))

         model.add(LSTM(units=100, return_sequences = True))
         model.add(Dropout(0.2))

         model.add(LSTM(units=100, return_sequences = True))
         model.add(Dropout(0.2))

         model.add(LSTM(units=100, return_sequences = False))
         model.add(Dropout(0.2))

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

Fitting the model

```
In [11]: hist = model.fit(X_train, y_train, epochs = 20, batch_size = 32, verbose=2)

Epoch 1/20
 - 12s - loss: 0.0322
Epoch 2/20
```

Preparing the input for the model

```
In [13]: testData = pd.read_csv('Google_test_data.csv')
         testData["Close"]=pd.to_numeric(testData.Close,errors='coerce')
         testData = testData.dropna()
         testData = testData.iloc[:,4:5]
         y_test = testData.iloc[60:,0:].values
         #input array for the model
         inputClosing = testData.iloc[:,0:].values
         inputClosing_scaled = sc.transform(inputClosing)
         inputClosing_scaled.shape
         X_test = []
         length = len(testData)
         timestep = 60
         for i in range(timestep,length):
             X_test.append(inputClosing_scaled[i-timestep:i,0])
         X_test = np.array(X_test)
         X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
         X_test.shape

Out[13]: (192, 60, 1)
```

Predicting the values for stock prices

```
In [15]: y_pred = model.predict(X_test)
         y_pred

Out[15]: array([[1.1260811],
                [1.1293991],
                [1.1416496],
                [1.1594812],
                [1.1733905],
                [1.1727879],
```

To plot the data between actual and predicted stock price we use inverse_transform function over y_pred data.

```
[16]: predicted_price = sc.inverse_transform(y_pred)
```

Plotting the actual and predicted prices for stocks

```
In [17]: plt.plot(y_test, color = 'red', label = 'Actual Stock Price')
         plt.plot(predicted_price, color = 'green', label = 'Predicted Stock Price')
         plt.title('Google stock price prediction')
         plt.xlabel('Time')
         plt.ylabel('Stock Price')
         plt.legend()
         plt.show()
```

Output:

Conclusion:

As you can see above , the model can predict the trend of actual stock prices very closely the accuracy of the model can be enhanced by training with data and increasing the LSTM layers…

References:

https://www.simplelearn.com/tutorials/machinelearning-tutorial/stock-price-prediction-usingmachine-learning

Directed by

N. Sesha Kumar Sir

THE END