TOOLS AND TECHNIQUES LABORATORY MINI PROJECT

REPORT

ON THE TOPIC: ***Stock Market Price Prediction.***

Submitted to: ***Dr. Soumya Ranjan Mishra***

Made by: Aditi Sinha - 2006105
Himanshu Mishra - 2006125
Riya Ramuka - 2006137
S.P.Aliva - 2006138

CONTENTS

1. Introduction
2. Abstract
3. Literature Review
4. Analysis Codes
5. Results
6. Conclusion
7. Summary

ABSTRACT

This mini project aims to develop a model for predicting stock prices using machine learning techniques and we focused on linear regression. Historical stock price data are collected and pre processed, and a range of machine learning algorithms are trained and tested on this data. The performance of each algorithm is evaluated based on key metrics such as opening price, closing price, low, high and date. The results of the project are used for predicting stock prices, and to gain insights into the factors that influence stock prices. The potential applications of this project include assisting investors and financial analysts in making informed decisions about buying and selling stocks, and providing insights into the behavior of financial markets.

And then we have also made a website using

frontend development(React js) regarding this showing our various results of the dataset that we have performed and all the data visualization and the actual vs predicted value based graph,mean square error, root mean square error etc.

Stock market prediction is a crucial area of research and analysis that focuses on forecasting the future value of stocks or financial instruments traded on exchanges. It is a challenging task due to the dynamic and complex nature of the stock market, which is affected by various factors such as economic indicators, company performance, geopolitical events, and investor sentiments.
Researchers and practitioners have developed various techniques and approaches for stock market prediction, including traditional statistical methods, machine learning algorithms, and deep learning models. These techniques use historical data to train predictive models and identify patterns and trends in the market, which are then used to make predictions about future stock prices.
Despite the significant progress made in this area, stock market prediction remains a

challenging problem, with limited accuracy and reliability in many cases. There are also ethical and moral implications associated with the use of predictive models in the stock market, particularly in the context of insider trading and market manipulation.
Overall, stock market prediction is a critical area of research that has the potential to inform investment decisions and financial planning, but it requires ongoing development and evaluation of innovative techniques and careful consideration of the ethical implications.

Datasets here used and analyzed by:

Aditi Sinha did analysis on the ***Adaniports*** dataset.

Himanshu Mishra did analysis on the ***Britannia*** dataset.

Riya Ramuka did analysis on the ***SBIN*** dataset.

S.P.Aliva did analysis on the ***TCS*** dataset.

LITERATURE REVIEW

✓ What we learnt from stock market prediction

Stock market prediction has provided us with valuable insights into the dynamics of financial markets and the factors that affect the prices of financial instruments. Here are some of the key things we have learned from stock market prediction:

1. Historical data can be used to identify patterns and trends in the market: By analyzing historical stock market data, researchers and practitioners have identified various patterns and trends that can be used to make predictions about future market behavior. These patterns include seasonality, cycles, and trends, which can help investors make more informed decisions.
2. Market sentiment can affect stock prices: Investor sentiment, or the collective emotions and attitudes of investors, can have a significant impact on stock prices. Predictive models that

incorporate sentiment analysis can provide insights into investor sentiment and help investors make more informed decisions.

3. Economic indicators can be used to predict market behavior: Economic indicators such as GDP, inflation, and interest rates can provide valuable insights into the health of the economy and the likely direction of the market. Predictive models that incorporate these indicators can help investors make more informed decisions about their investments.
4. Predictive models have limited accuracy and reliability: Despite significant progress in this area, stock market prediction models have limited accuracy and reliability, and they should be used with caution. Predictive models should be continually evaluated and refined to improve their accuracy and reliability.
5. Predictive models can be used to inform investment decisions: While predictive models should not be relied upon exclusively to make investment decisions, they can provide valuable insights that can be used to inform investment decisions. Investors should use a combination of predictive models, fundamental analysis, and other factors to make informed decisions about their investments.

In summary, stock market prediction has provided us with valuable insights into the dynamics of financial markets and the factors that affect the prices of financial instruments. While predictive models have limitations, they can be a valuable tool for investors looking to make more informed investment decisions.

✓ Why did we do stock market prediction using linear regression?

Linear regression is one of the most widely used techniques in stock market prediction because it allows us to model the relationship between an independent variable (such as past stock prices, economic indicators, or company financial data) and a dependent variable (future stock prices).

Linear regression models assume that the relationship between the independent variable and the dependent variable is linear, meaning that a change in one variable corresponds to a proportional change in the other variable. This assumption can often hold true in the stock market, as past trends and economic indicators can provide insights into the likely direction of future stock prices.

Linear regression models are also relatively simple to implement and interpret, which makes them accessible to a wide range of analysts and investors. However, it's worth noting that stock market prediction is a

complex and challenging task, and no single technique or model can provide perfect predictions. As with any forecasting approach, it's important to consider the limitations and potential biases of the model being used, and to use a variety of tools and techniques to inform investment decisions.

Stock market price prediction is an important problem in finance, and in this report, we utilized machine learning techniques to predict future stock prices using historical data of four companies: Tata Consultancy Services (TCS), AdaniPorts, State Bank of India (SBIN), and Britannia. We employed linear regression as the algorithm for this project, along with data preprocessing and visualization techniques. After training the model, we evaluated its performance using Root Mean Squared Error (RMSE) on the testing set, and the results showed that the predicted prices closely followed the actual prices.

INTRODUCTION

Stock market price prediction is an important problem in finance and has been tackled with various approaches. One of the most common approaches is using machine learning techniques to predict future stock prices. In this report, we described a machine learning stock market price prediction that we did using CSV files of multiple companies, where we calculated the actual and predicted prices and generated a linear regression graph. We utilized linear regression as the algorithm for this project.
Along with linear regression we also did smaller analysis using different visualisation methods such as heatmap, bar graph of all the columns of the dataset, boxplot, histogram, scatter plot and lastly a graph stating the stock prices of the companies over years.

Stock market price prediction is a challenging task in finance, as it involves predicting the future prices of stocks based on historical data. Various approaches have been used, including

machine learning techniques, which have shown promising results in recent years. In this report, we aimed to predict the stock prices of four companies using linear regression as the algorithm.

Data Acquisition: We obtained historical stock price data of four companies, TCS, Adani Ports, SBIN, and Britannia, in the form of CSV files. The data included information such as date, opening price, highest price, lowest price, closing price, and volume. We loaded the data using Python and utilized pandas for data manipulation and visualization.

Data Preprocessing: Data preprocessing is an essential step in machine learning to clean and transform the data into a suitable format for model training. We performed data preprocessing by removing unnecessary columns and keeping only the date, open, high, low, last, and close columns. We then split the data into two sets, with 80% for training the model and 20% for testing its performance.

DATA

The first step in this project was to obtain the data. We downloaded four CSV files containing historical stock prices of four companies named as Tata Consultancy Services(TCS), AdaninPorts, State Bank of India(SBIN) and Britannia . The file contained each company's data, including the date, the opening price, the highest price, the lowest price, the closing price, and the volume. We used Python to load the file and used pandas,data visualization and linear regression to predict the company's stock market status.

PREPROCESSING

Next, we performed data preprocessing. We removed the unnecessary columns and kept only the date, open, high, low, last and close columns. Then we splitted the data into two sets, one for training the model, and another for testing its performance. We used the first 80% of the data for training and the remaining 20% for testing.

MODEL

The algorithm we chose for this project was linear regression. Linear regression is a simple but effective algorithm for predicting numeric values. It works by finding a line that best fits the data points. In this case, the line represents the relationship between the closing prices and the time.

For this project, we chose linear regression as the algorithm for predicting stock prices. Linear regression is a simple but effective algorithm that finds a line that best fits the data points, representing the relationship between the closing prices and time.

Results: After training the linear regression model, we used it to predict the stock prices for the testing set. We calculated the actual and predicted prices and plotted them on a graph. The graph showed that the predicted prices closely followed the actual prices, indicating that the model performed well. We evaluated the model's performance using Root Mean Squared Error (RMSE), which is a commonly used metric to measure the accuracy of regression models.

The RMSE values obtained for the testing set of each company were:

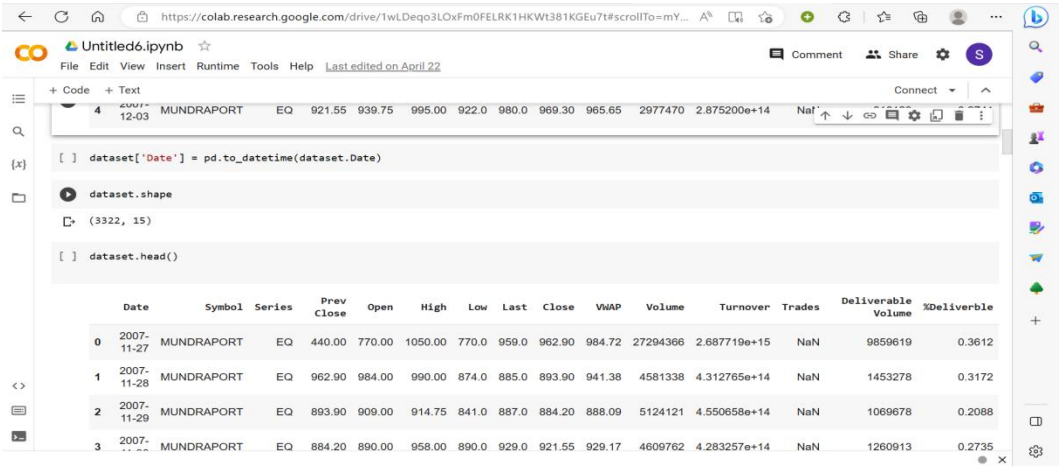
TCS: 479.59
Adani Ports: 862.07
SBIN: 862.07
Britannia: 16.92

Data Visualization

In addition to linear regression, we also utilized data visualization techniques to analyze the data. We created various visualizations, including heatmaps, bar graphs of all columns in the dataset, boxplots, histograms, scatter plots, and a graph depicting the stock prices of the companies over the years. These visualizations helped us gain insights into the data and better understand the trends and patterns.

STOCK PRICE PREDICTION ANALYSIS CODES

ADANIPORTS PRICE PREDICTION CODE:



Untitled6.ipynb

File Edit View Insert Runtime Tools Help Last edited on April 22

Comment Share S

+ Code + Text Connect

4	2007-12-03	MUNDRAPORT	EQ	921.55	939.75	995.00	922.0	980.0	969.30	965.65	2977470	2.875200e+14	Nal	

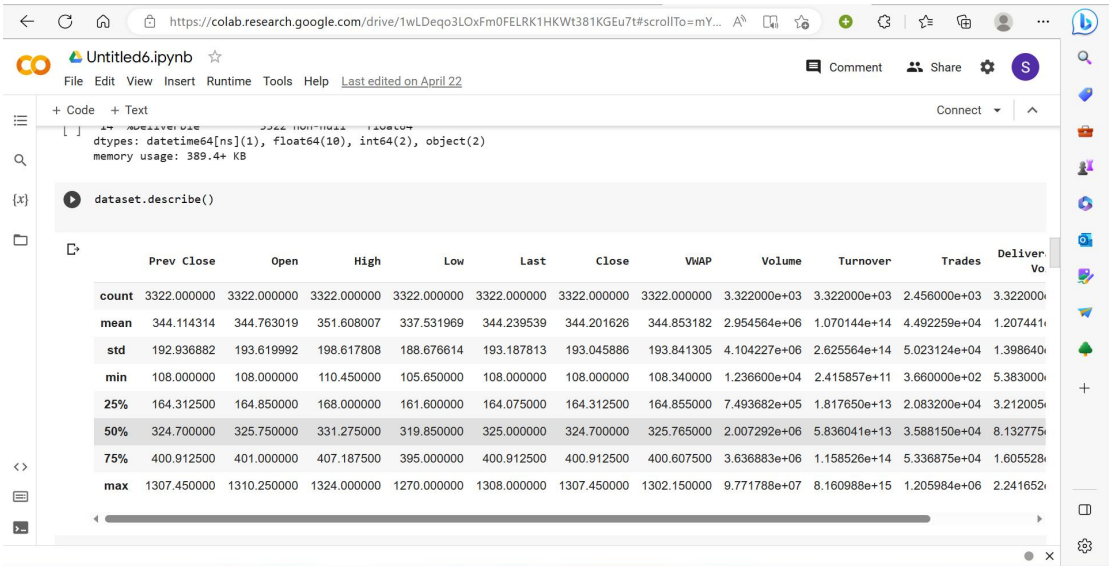
```
[ ] dataset['Date'] = pd.to_datetime(dataset.Date)
```

```
dataset.shape
```

```
(3322, 15)
```

```
[ ] dataset.head()
```

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2007-11-27	MUNDRAPORT	EQ	440.00	770.00	1050.00	770.0	959.0	962.90	984.72	27294366	2.687719e+15	NaN	9859619	0.3612
1	2007-11-28	MUNDRAPORT	EQ	962.90	984.00	990.00	874.0	885.0	893.90	941.38	4581338	4.312765e+14	NaN	1453278	0.3172
2	2007-11-29	MUNDRAPORT	EQ	893.90	909.00	914.75	841.0	887.0	884.20	888.09	5124121	4.550658e+14	NaN	1069678	0.2088
3	2007-	MUNDRAPORT	EQ	884.20	890.00	958.00	890.0	929.0	921.55	929.17	4609762	4.283257e+14	NaN	1260913	0.2735



Untitled6.ipynb

File Edit View Insert Runtime Tools Help Last edited on April 22

Comment Share S

+ Code + Text Connect

```
dtypes: datetime64[ns](1), float64(10), int64(2), object(2)
memory usage: 389.4+ KB
```

```
dataset.describe()
```

	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliver Vo
count	3322.000000	3322.000000	3322.000000	3322.000000	3322.000000	3322.000000	3322.000000	3.322000e+03	3.322000e+03	2.456000e+03	3.322000
mean	344.114314	344.763019	351.608007	337.531969	344.239539	344.201626	344.853182	2.954564e+06	1.070144e+14	4.492259e+04	1.207441
std	192.936882	193.619992	198.617808	188.676614	193.187813	193.045886	193.841305	4.104227e+06	2.625564e+14	5.023124e+04	1.398640
min	108.000000	108.000000	110.450000	105.650000	108.000000	108.000000	108.340000	1.236600e+04	2.415857e+11	3.660000e+02	5.383000
25%	164.312500	164.850000	168.000000	161.600000	164.075000	164.312500	164.855000	7.493682e+05	1.817650e+13	2.083200e+04	3.212005
50%	324.700000	325.750000	331.275000	319.850000	325.000000	324.700000	325.765000	2.007292e+06	5.836041e+13	3.588150e+04	8.132775
75%	400.912500	401.000000	407.187500	395.000000	400.912500	400.912500	400.607500	3.636883e+06	1.158526e+14	5.336875e+04	1.605528
max	1307.450000	1310.250000	1324.000000	1270.000000	1308.000000	1307.450000	1302.150000	9.771788e+07	8.160988e+15	1.205984e+06	2.241652

https://colab.research.google.com/drive/1wLDeqo3LOxFm0FELRK1HKWt381KGEu7t#scrollTo=S8...

Untitled6.ipynb

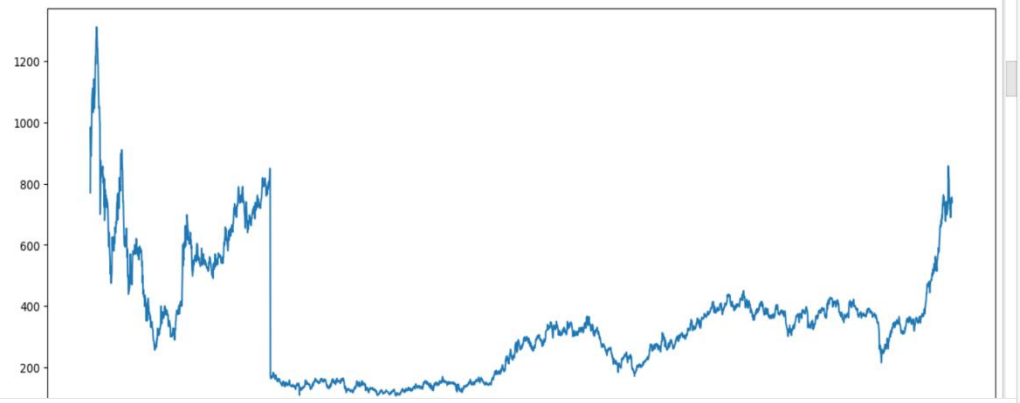
File Edit View Insert Runtime Tools Help Last edited on April 22

Comment Share S

+ Code + Text Connect

```
dataset['Open'].plot(figsize=(16,6))
```

```
<Axes: >
```



https://colab.research.google.com/drive/1wLDeqo3LOxFm0FELRK1HKWt381KGEu7t#scrollTo=S8...

Untitled6.ipynb

File Edit View Insert Runtime Tools Help Last edited on April 22

Comment Share S

+ Code + Text Connect


```
X  = dataset[['Open','High','Low','Volume']]
y = dataset['Close']
```

```
[ ] from sklearn.model_selection import train_test_split
    X_train , X_test , y_train , y_test = train_test_split(X ,y , random_state = 0)
```

```
[ ] X_train.shape
```

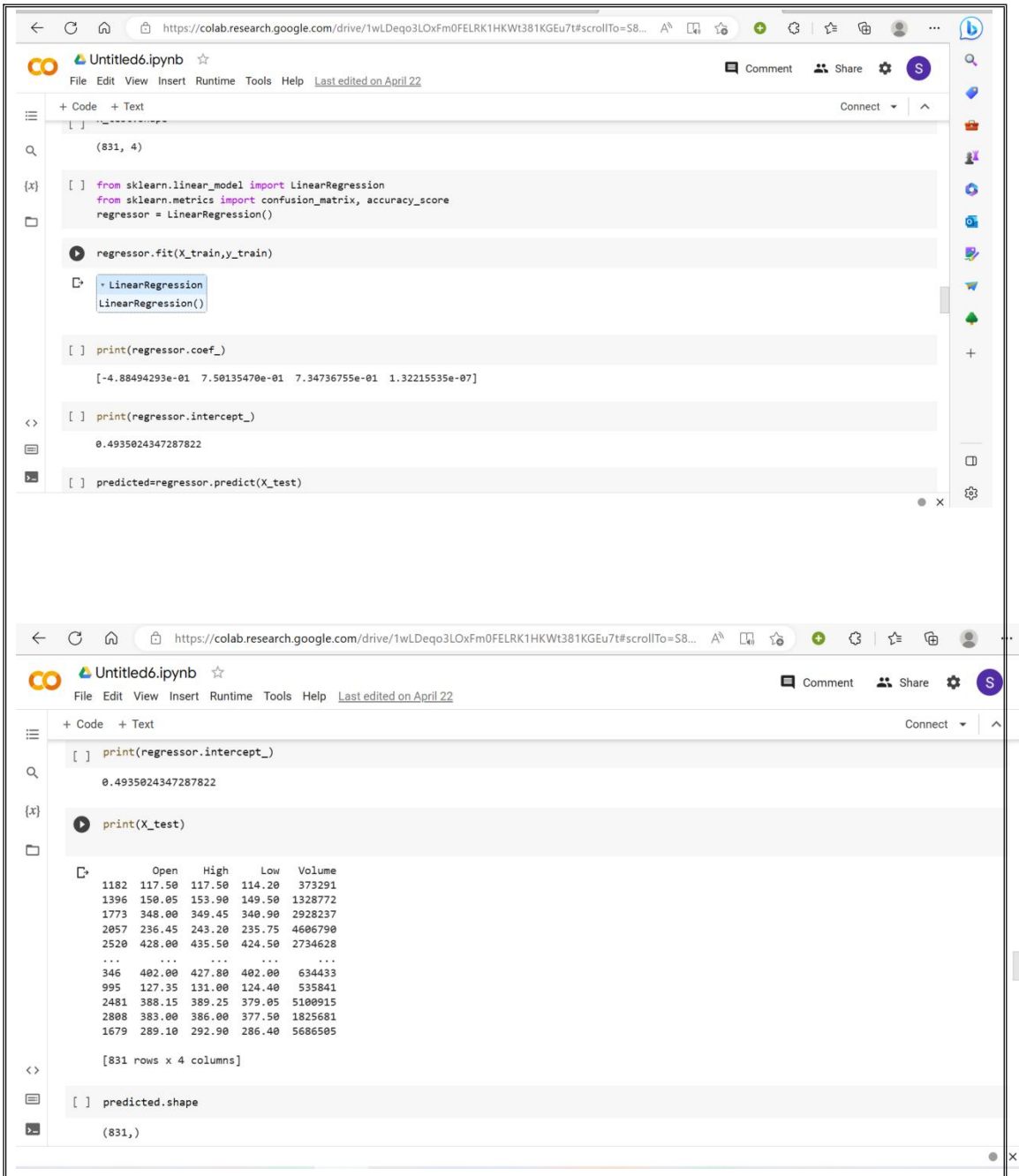
```
(2491, 4)
```

```
[ ] X_test.shape
```

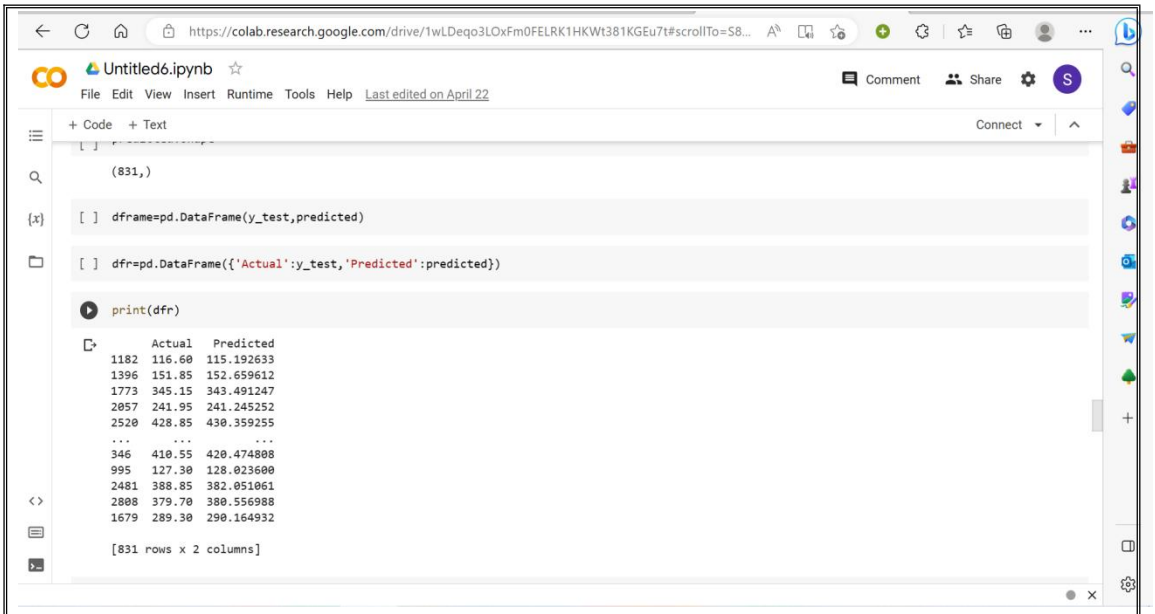
```
(831, 4)
```

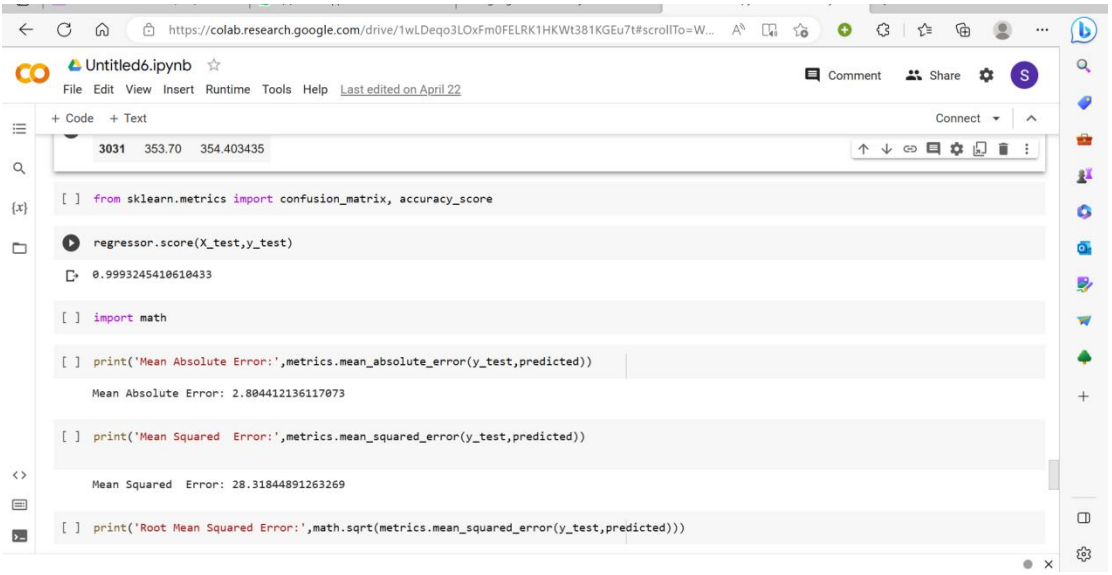
```
[ ] from sklearn.linear_model import LinearRegression
    from sklearn.metrics import confusion_matrix, accuracy_score
    regressor = LinearRegression()
```

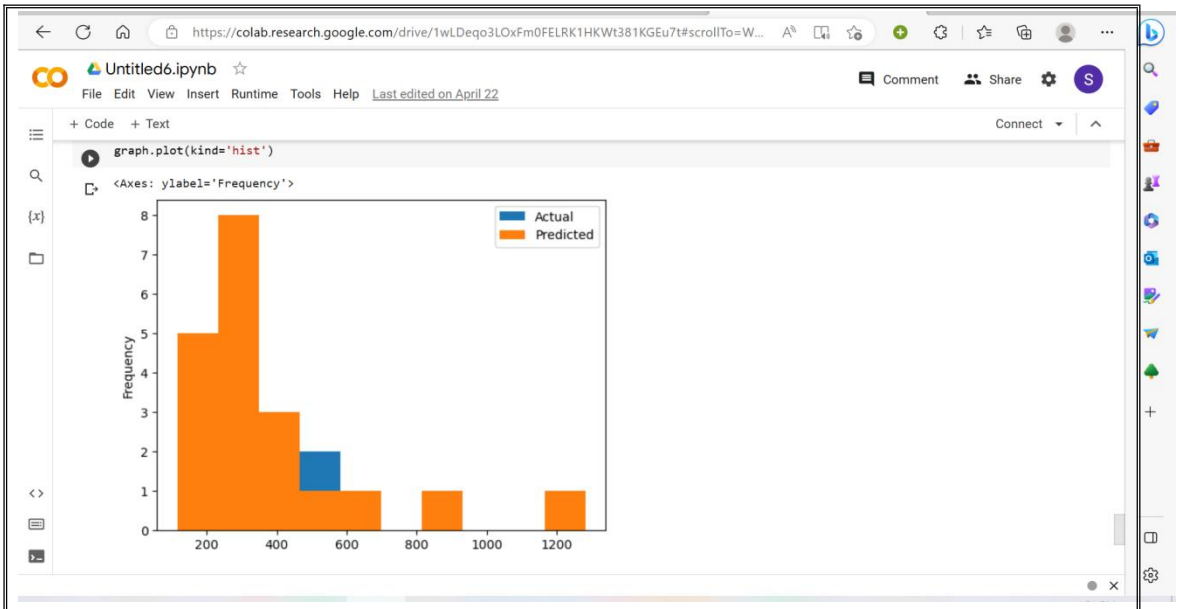
```
[ ] regressor.fit(X_train,y_train)
```

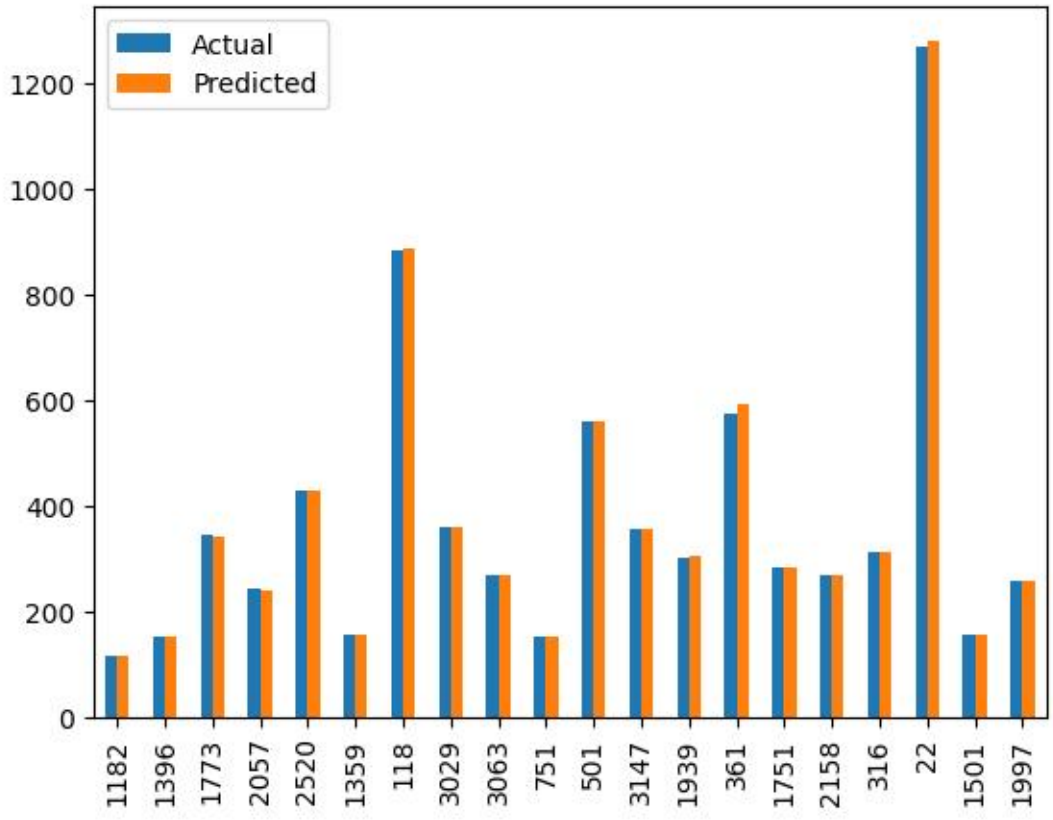

https://colab.research.google.com/drive/1wLDeqo3LOxFm0FELRK1HKWt381KGEu7t#scrollTo=S8...
Untitled6.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 22
Comment
Share
+ Code + Text
Connect
(831, 4)
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix, accuracy_score
regressor = LinearRegression()
regressor.fit(X_train,y_train)
LinearRegression
LinearRegression()
print(regressor.coef_)
[-4.88494293e-01 7.50135470e-01 7.34736755e-01 1.32215535e-07]
print(regressor.intercept_)
0.4935024347287822
predicted=regressor.predict(X_test)

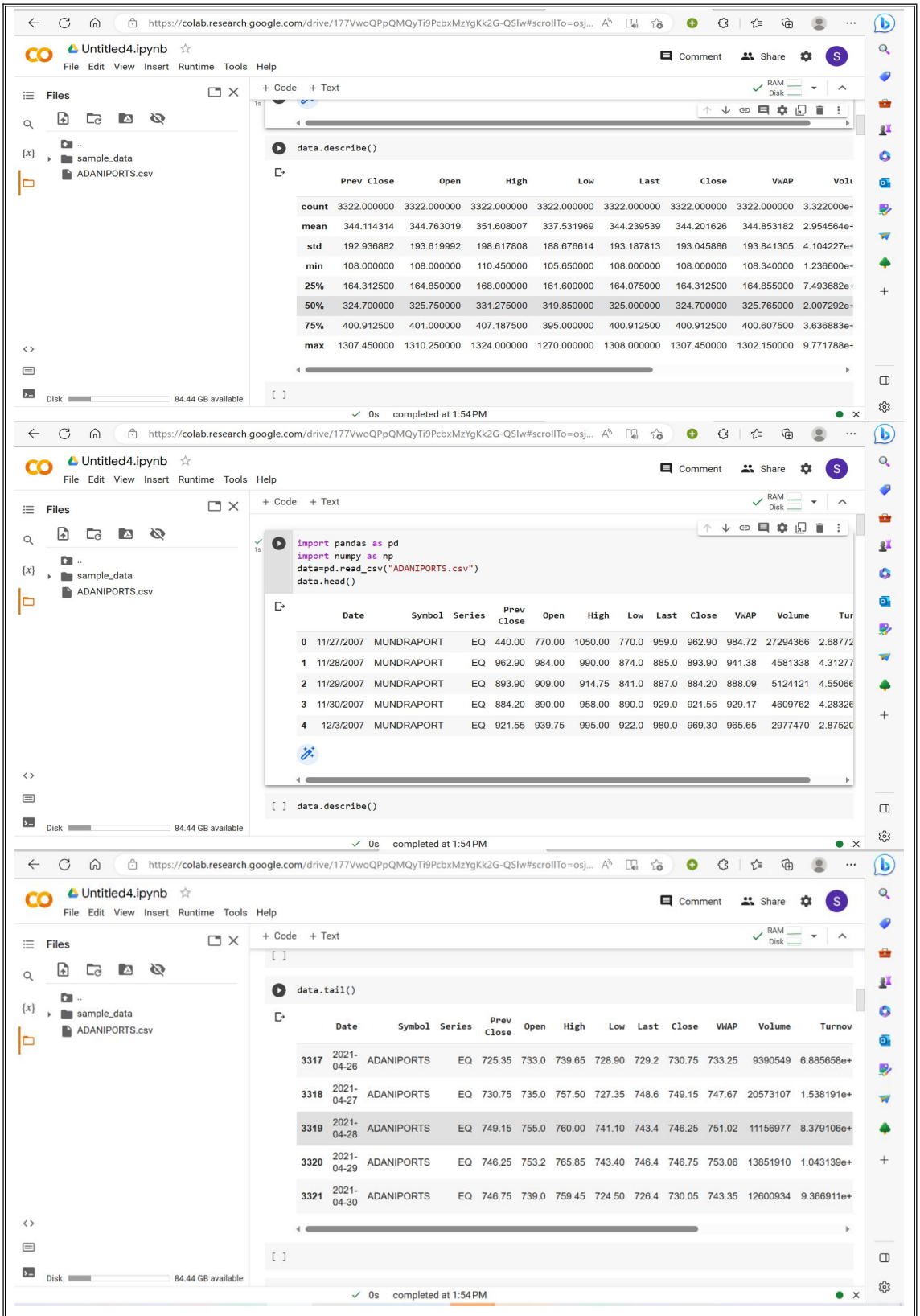
https://colab.research.google.com/drive/1wLDeqo3LOxFm0FELRK1HKWt381KGEu7t#scrollTo=S8...
Untitled6.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 22
Comment
Share
+ Code + Text
Connect
print(regressor.intercept_)
0.4935024347287822
print(X_test)
Open High Low Volume
1182 117.50 117.50 114.20 373291
1396 150.05 153.90 149.50 1328772
1773 348.00 349.45 340.90 2928237
2057 236.45 243.20 235.75 4606790
2520 428.00 435.50 424.50 2734628
... ...
346 402.00 427.80 402.00 634433
995 127.35 131.00 124.40 535841
2481 388.15 389.25 379.05 5100915
2808 383.00 386.00 377.50 1825681
1679 289.10 292.90 286.40 5686505
[831 rows x 4 columns]
predicted.shape
(831,)

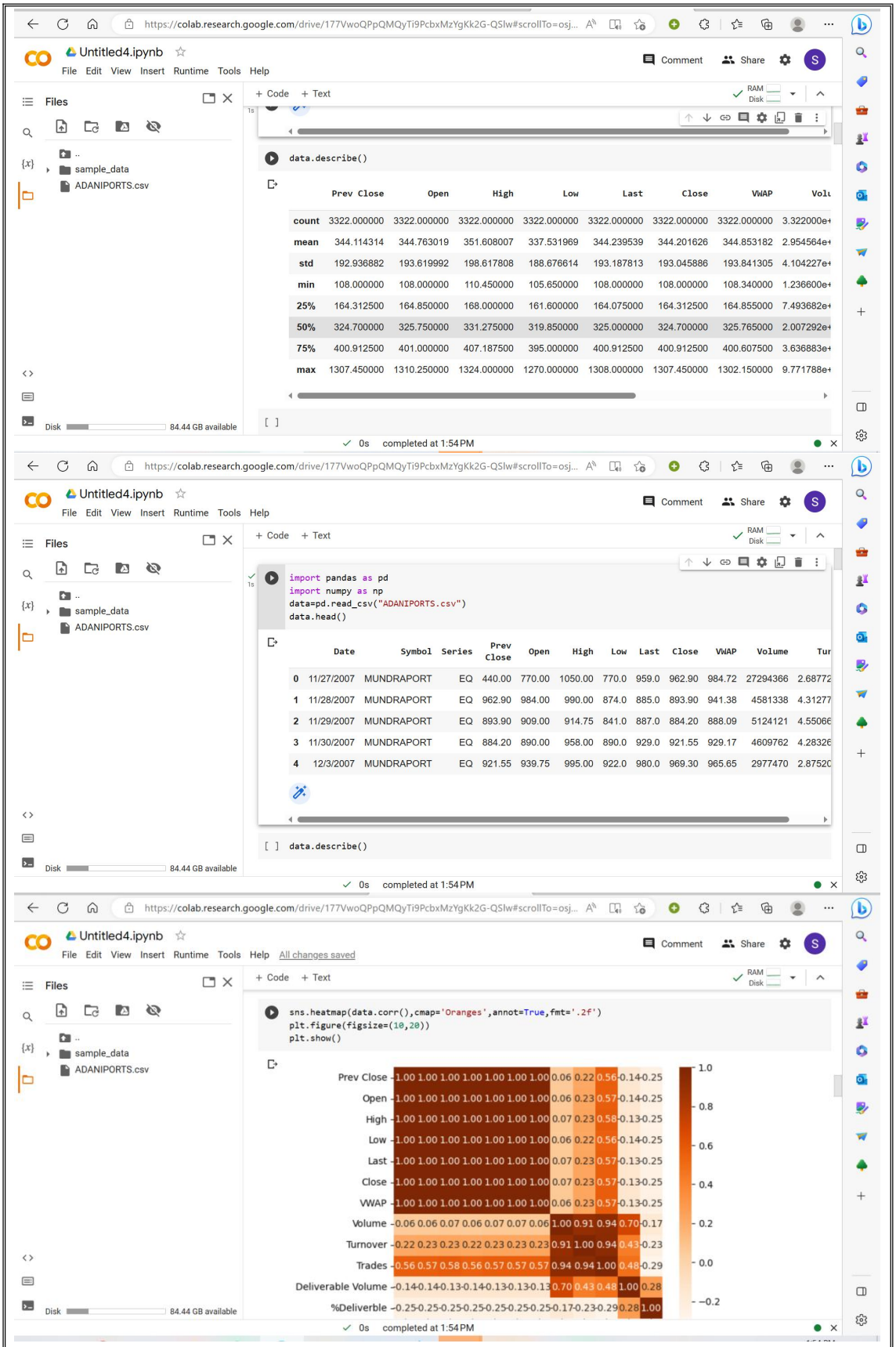

https://colab.research.google.com/drive/1wLDeqo3LOxFm0FELRK1HKWt381KGEu7t#scrollTo=S8...
Untitled6.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 22
Comment
Share
+ Code + Text
Connect
(831,)
[ ] dframe=pd.DataFrame(y_test,predicted)
[ ] dfr=pd.DataFrame({'Actual':y_test,'Predicted':predicted})
print(dfr)
Actual Predicted
1182 116.60 115.192633
1396 151.85 152.659612
1773 345.15 343.491247
2057 241.95 241.245252
2520 428.85 430.359255
... ...
346 410.55 420.474808
995 127.30 128.023600
2481 388.85 382.051061
2808 379.70 380.556988
1679 289.30 290.164932
[831 rows x 2 columns]

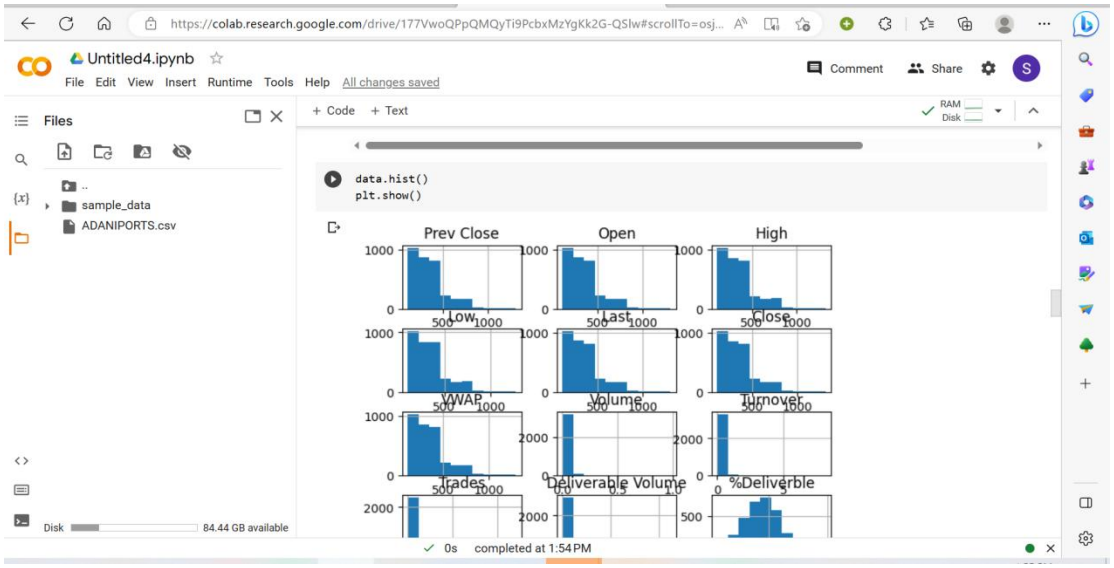

https://colab.research.google.com/drive/1wLDeqo3LOxFm0FELRK1HKWt381KGEu7t#scrollTo=W...
Untitled6.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 22
Comment
Share
+ Code + Text
Connect
3031 353.70 354.403435
[ ] from sklearn.metrics import confusion_matrix, accuracy_score
regressor.score(X_test,y_test)
0.9993245410610433
[ ] import math
[ ] print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,predicted))
Mean Absolute Error: 2.804412136117073
[ ] print('Mean Squared Error:',metrics.mean_squared_error(y_test,predicted))
Mean Squared Error: 28.31844891263269
[ ] print('Root Mean Squared Error:',math.sqrt(metrics.mean_squared_error(y_test,predicted)))

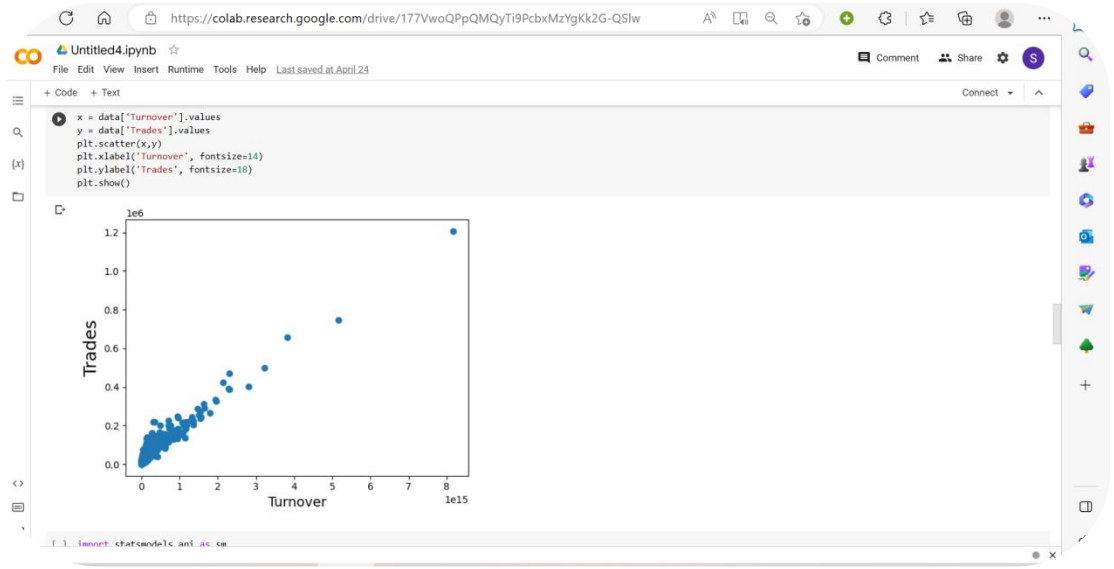

Untitled6.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 22
Comment
Share
+ Code + Text
Connect
graph.plot(kind='hist')
<Axes: ylabel='Frequency'>
Actual
Predicted
Frequency
0
1
2
3
4
5
6
7
8
200
400
600
800
1000
1200


Actual
Predicted
0
200
400
600
800
1000
1200
1182
1396
1773
2057
2520
1359
118
3029
3063
751
501
3147
1939
361
1751
2158
316
22
1501
1997


https://colab.research.google.com/drive/177VwoQPpQMQyTi9PcbxMzYgKk2G-QSIw#scrollTo=osj...
Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
Files
+ Code + Text
RAM
Disk
sample_data
ADANIPORTS.csv
data.describe()
Prev Close Open High Low Last Close VWAP Volu
count 3322.000000 3322.000000 3322.000000 3322.000000 3322.000000 3322.000000 3322.000000 3.322000e+
mean 344.114314 344.763019 351.608007 337.531969 344.239539 344.201626 344.853182 2.954564e+
std 192.936882 193.619992 198.617808 188.676614 193.187813 193.045886 193.841305 4.104227e+
min 108.000000 108.000000 110.450000 105.650000 108.000000 108.000000 108.340000 1.236600e+
25% 164.312500 164.850000 168.000000 161.600000 164.075000 164.312500 164.855000 7.493682e+
50% 324.700000 325.750000 331.275000 319.850000 325.000000 324.700000 325.765000 2.007292e+
75% 400.912500 401.000000 407.187500 395.000000 400.912500 400.912500 400.607500 3.636883e+
max 1307.450000 1310.250000 1324.000000 1270.000000 1308.000000 1307.450000 1302.150000 9.771788e+
Disk 84.44 GB available
0s completed at 1:54 PM
https://colab.research.google.com/drive/177VwoQPpQMQyTi9PcbxMzYgKk2G-QSIw#scrollTo=osj...
Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
Files
+ Code + Text
RAM
Disk
sample_data
ADANIPORTS.csv
import pandas as pd
import numpy as np
data=pd.read_csv("ADANIPORTS.csv")
data.head()
Date Symbol Series Prev Close Open High Low Last Close VWAP Volume Tur
0 11/27/2007 MUNDRAPORT EQ 440.00 770.00 1050.00 770.0 959.0 962.90 984.72 27294366 2.68772
1 11/28/2007 MUNDRAPORT EQ 962.90 984.00 990.00 874.0 885.0 893.90 941.38 4581338 4.31277
2 11/29/2007 MUNDRAPORT EQ 893.90 909.00 914.75 841.0 887.0 884.20 888.09 5124121 4.55066
3 11/30/2007 MUNDRAPORT EQ 884.20 890.00 958.00 890.0 929.0 921.55 929.17 4609762 4.28326
4 12/3/2007 MUNDRAPORT EQ 921.55 939.75 995.00 922.0 980.0 969.30 965.65 2977470 2.87520
[ ] data.describe()
Disk 84.44 GB available
0s completed at 1:54 PM
https://colab.research.google.com/drive/177VwoQPpQMQyTi9PcbxMzYgKk2G-QSIw#scrollTo=osj...
Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
Files
+ Code + Text
RAM
Disk
sample_data
ADANIPORTS.csv
[ ]
data.tail()
Date Symbol Series Prev Close Open High Low Last Close VWAP Volume Turnov
3317 2021-04-26 ADANIPORTS EQ 725.35 733.0 739.65 728.90 729.2 730.75 733.25 9390549 6.885658e+
3318 2021-04-27 ADANIPORTS EQ 730.75 735.0 757.50 727.35 748.6 749.15 747.67 20573107 1.538191e+
3319 2021-04-28 ADANIPORTS EQ 749.15 755.0 760.00 741.10 743.4 746.25 751.02 11156977 8.379106e+
3320 2021-04-29 ADANIPORTS EQ 746.25 753.2 765.85 743.40 746.4 746.75 753.06 13851910 1.043139e+
3321 2021-04-30 ADANIPORTS EQ 746.75 739.0 759.45 724.50 726.4 730.05 743.35 12600934 9.366911e+
[ ]
Disk 84.44 GB available
0s completed at 1:54 PM


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
Files
+ Code + Text
sample_data
ADANIPORTS.csv
Disk 84.44 GB available
data.describe()
Prev Close Open High Low Last Close VWAP Volu
count 3322.000000 3322.000000 3322.000000 3322.000000 3322.000000 3322.000000 3322.000000 3.322000e+
mean 344.114314 344.763019 351.608007 337.531969 344.239539 344.201626 344.853182 2.954564e+
std 192.936882 193.619992 198.617808 188.676614 193.187813 193.045886 193.841305 4.104227e+
min 108.000000 108.000000 110.450000 105.650000 108.000000 108.000000 108.340000 1.236600e+
25% 164.312500 164.850000 168.000000 161.600000 164.075000 164.312500 164.855000 7.493682e+
50% 324.700000 325.750000 331.275000 319.850000 325.000000 324.700000 325.765000 2.007292e+
75% 400.912500 401.000000 407.187500 395.000000 400.912500 400.912500 400.607500 3.636883e+
max 1307.450000 1310.250000 1324.000000 1270.000000 1308.000000 1307.450000 1302.150000 9.771788e+
0s completed at 1:54 PM
import pandas as pd
import numpy as np
data=pd.read_csv("ADANIPORTS.csv")
data.head()
Date Symbol Series Prev Close Open High Low Last Close VWAP Volume Tur
0 11/27/2007 MUNDRAPORT EQ 440.00 770.00 1050.00 770.0 959.0 962.90 984.72 27294366 2.68772
1 11/28/2007 MUNDRAPORT EQ 962.90 984.00 990.00 874.0 885.0 893.90 941.38 4581338 4.31277
2 11/29/2007 MUNDRAPORT EQ 893.90 909.00 914.75 841.0 887.0 884.20 888.09 5124121 4.55066
3 11/30/2007 MUNDRAPORT EQ 884.20 890.00 958.00 890.0 929.0 921.55 929.17 4609762 4.28326
4 12/3/2007 MUNDRAPORT EQ 921.55 939.75 995.00 922.0 980.0 969.30 965.65 2977470 2.87520
[ ] data.describe()
All changes saved
sns.heatmap(data.corr(),cmap='Oranges',annot=True,fmt='.2f')
plt.figure(figsize=(10,20))
plt.show()
Prev Close
Open
High
Low
Last
Close
VWAP
Volume
Turnover
Trades
Deliverable Volume
%Deliverble
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

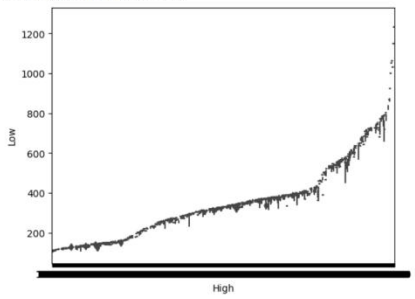

Untitled4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
sample_data
ADANIPORTS.csv
Disk 84.44 GB available
data.hist()
plt.show()
Prev Close
Open
High
Low
Last
Close
VWAP
Volume
Turnover
Trades
Deliverable Volume
%Deliverble
0s completed at 1:54 PM


Untitled4.ipynb
File Edit View Insert Runtime Tools Help Last saved at April 24
Comment
Share
+ Code + Text
Connect
x = data['Turnover'].values
y = data['Trades'].values
plt.scatter(x,y)
plt.xlabel('Turnover', fontsize=14)
plt.ylabel('Trades', fontsize=18)
plt.show()
1e6
Trades
Turnover
1e15

```
[ ]
```

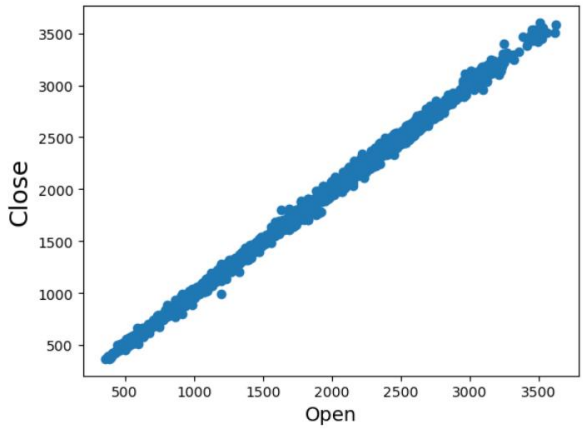
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.boxplot(x='High' ,y='Low' ,data=data)
```

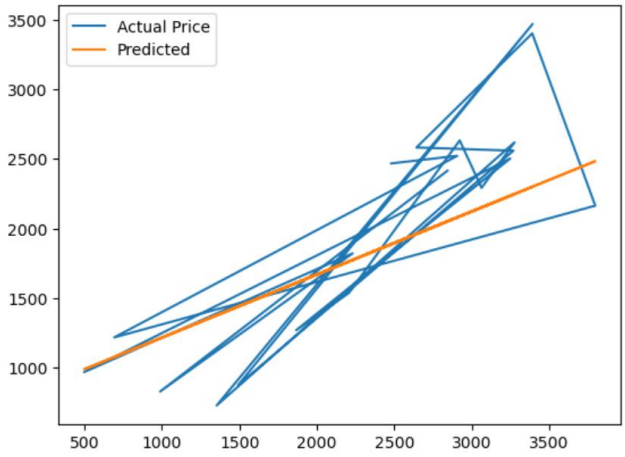
```
<Axes: xlabel='High', ylabel='Low'>
```



TCS PRICE PREDICTION CODE:

In [16]:
```
x = tcs['Open'].values
y = tcs['Close'].values
plt.scatter(x,y)
plt.xlabel('Open', fontsize=14)
plt.ylabel('Close', fontsize=18)
plt.show()
```





In [37]:
```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,predicted))
```

```
Mean Absolute Error: 427.7763398959711
```

In [38]:
```
print('Mean Squared Error:',metrics.mean_squared_error(y_test,predicted))
```

```
Mean Squared Error: 230006.8587044432
```

In [39]:
```
print('Root Mean Squared Error:',math.sqrt(metrics.mean_squared_error(y_test,predicted)))
```

```
Root Mean Squared Error: 479.59030297165435
```

In [37]:
```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,predicted))
```

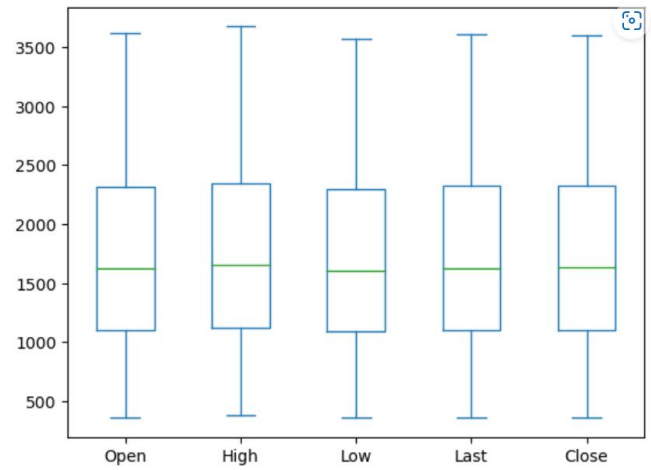
```
Mean Absolute Error: 427.7763398959711
```

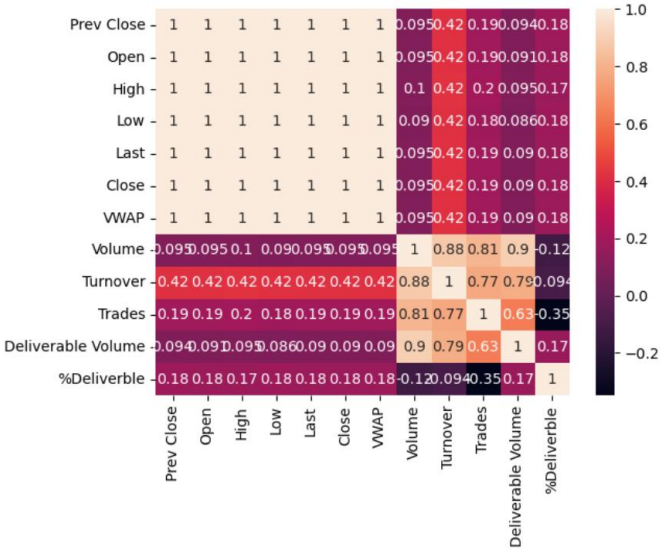
In [38]:
```
print('Mean Squared Error:',metrics.mean_squared_error(y_test,predicted))
```

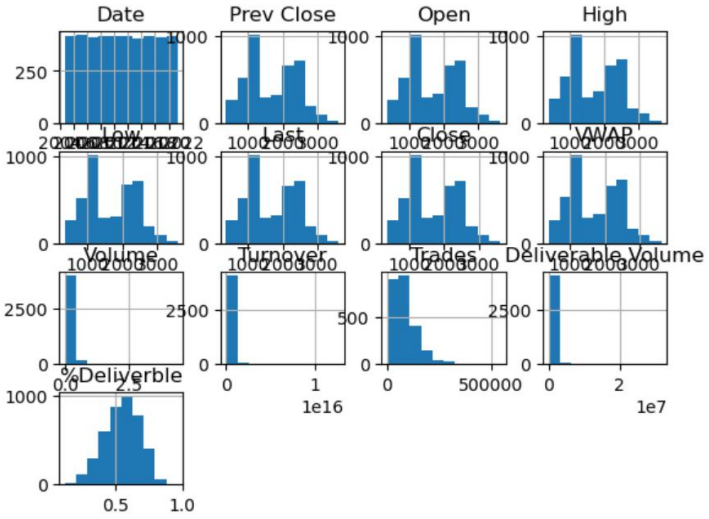
```
Mean Squared Error: 230006.8587044432
```

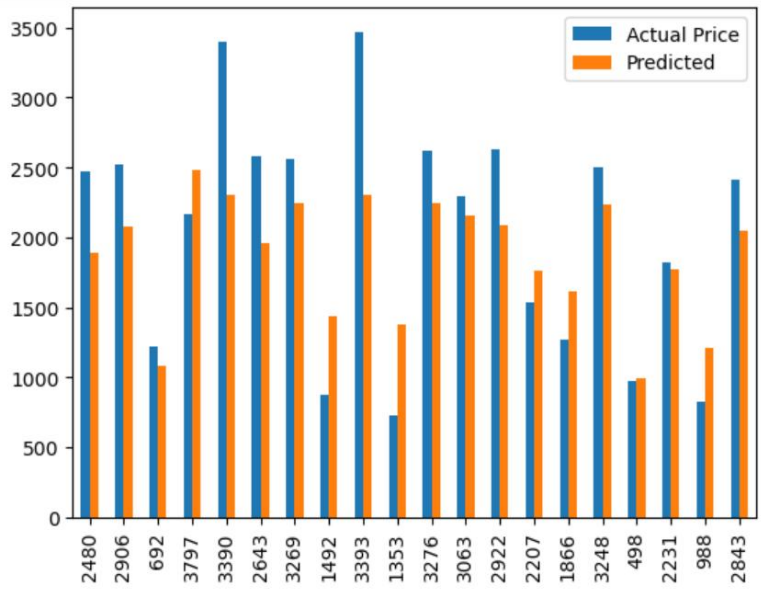
In [39]:
```
print('Root Mean Squared Error:',math.sqrt(metrics.mean_squared_error(y_test,predicted)))
```

```
Root Mean Squared Error: 479.59030297165435
```



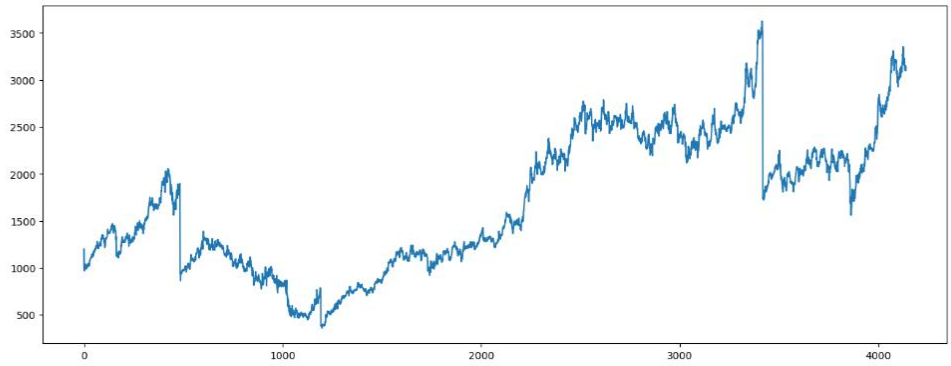






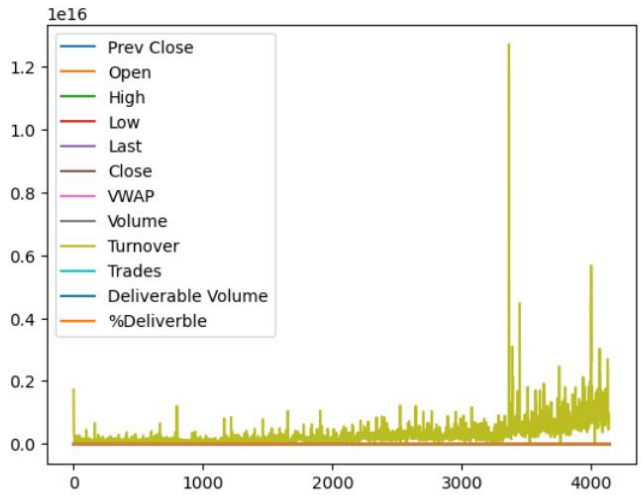
In [33]:
```
tcs['Open'].plot(figsize=(16,6))
```

Out[33]: <AxesSubplot:>



In [6]:
```
tcs.plot(legend=True)
```

Out[6]: <AxesSubplot:>



In [36]:
```
# building regression model
from sklearn.model_selection import train_test_split

#for preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

#for model evaluation
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
```

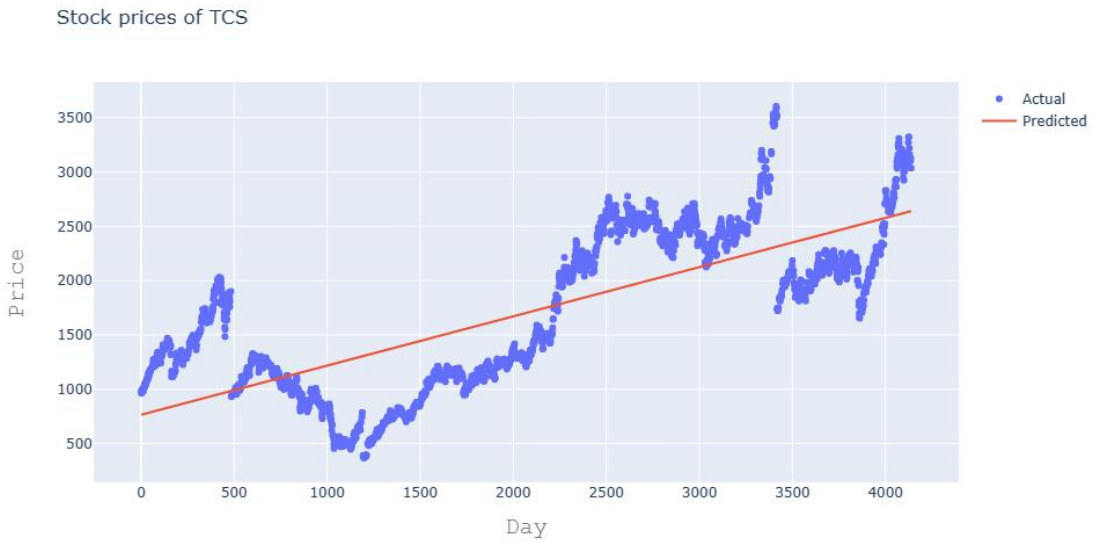
In [53]:
```
from sklearn.linear_model import LinearRegression
```

In [38]:
```
import plotly.graph_objs as go
from plotly.offline import plot

# for offline plotting
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

In [51]:
```
# plot actual and predicted values for train dataset
trace0=go.Scatter(
x=x_train.T[0],
y=y_train,
mode='markers',
name='Actual'
)
trace1=go.Scatter(
x=x_train.T[0],
y=lm.predict(x_train).T,
mode='lines',
name='Predicted'
)
tcs_data= [trace0,trace1]
layout.xaxis.title.text='Day'
plot2=go.Figure(data=tcs_data,layout=layout)
```

In [52]:
```
iplot(plot2)
```



SBIN PRICE PREDICTION CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import matplotlib as plib
```

```
[ ] sbin=pd.read_csv('SBIN.csv')
    sbin
```

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2000-01-03	SBIN	EQ	225.60	236.00	243.65	234.25	243.65	243.65	240.83	2373228	5.715338e+13	NaN	NaN	NaN
1	2000-01-04	SBIN	EQ	243.65	243.65	262.00	238.85	258.00	259.10	251.46	4495741	1.130506e+14	NaN	NaN	NaN
2	2000-01-05	SBIN	EQ	259.10	249.00	264.70	245.00	249.05	248.45	252.35	3434058	8.666008e+13	NaN	NaN	NaN
3	2000-01-06	SBIN	EQ	248.45	252.00	268.00	252.00	260.50	261.00	262.18	6658801	1.745817e+14	NaN	NaN	NaN
4	2000-01-07	SBIN	EQ	261.00	261.50	279.90	255.00	279.00	273.30	269.02	7873985	2.118287e+14	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5301	2021-04-26	SBIN	EQ	336.45	339.25	347.45	339.25	344.80	344.30	345.26	49234985	1.699874e+15	306217.0	11962001.0	0.2430
5302	2021-04-27	SBIN	EQ	344.30	344.00	354.95	342.40	354.20	353.05	349.42	46003023	1.607459e+15	293357.0	9882201.0	0.2148
5303	2021-04-28	SBIN	EQ	353.05	357.00	364.30	356.05	362.90	363.40	361.63	56696255	2.050313e+15	337002.0	13430042.0	0.2369
5304	2021-04-29	SBIN	EQ	363.40	365.00	369.95	355.50	357.45	359.40	361.26	63692926	2.300946e+15	348461.0	10008961.0	0.1571
5305	2021-04-30	SBIN	EQ	359.40	353.45	362.50	350.45	352.30	353.50	357.11	53832840	1.922408e+15	296959.0	13261927.0	0.2464

5306 rows × 15 columns

```
[ ] sbin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5306 entries, 0 to 5305
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                5306 non-null   object
 1   Symbol              5306 non-null   object
 2   Series              5306 non-null   object
 3   Prev Close          5306 non-null   float64
 4   Open                5306 non-null   float64
 5   High                5306 non-null   float64
 6   Low                 5306 non-null   float64
 7   Last                5306 non-null   float64
 8   Close               5306 non-null   float64
 9   VWAP                5306 non-null   float64
 10  Volume              5306 non-null   int64
 11  Turnover            5306 non-null   float64
 12  Trades              2456 non-null   float64
 13  Deliverable Volume  4792 non-null   float64
 14  %Deliverble         4792 non-null   float64
dtypes: float64(11), int64(1), object(3)
memory usage: 621.9+ KB
```

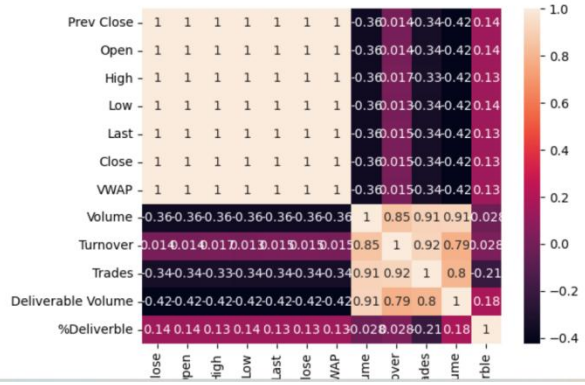
Double-click (or enter) to edit

```
[ ] sbin.describe()
```

	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%De
count	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5306.00000	5.306000e+03	5.306000e+03	2.456000e+03	4.792000e+03	47
mean	965.871438	967.182115	982.101583	950.898587	965.727026	965.895543	966.77033	1.003930e+07	4.136854e+14	1.507028e+05	3.012573e+06	
std	857.785547	858.925475	870.739801	845.761231	857.510230	857.766537	858.38307	1.783189e+07	4.578376e+14	1.082755e+05	5.073956e+06	
min	141.450000	142.850000	147.950000	140.050000	140.550000	141.450000	145.90000	3.633000e+03	7.680600e+10	4.016000e+03	1.548000e+04	

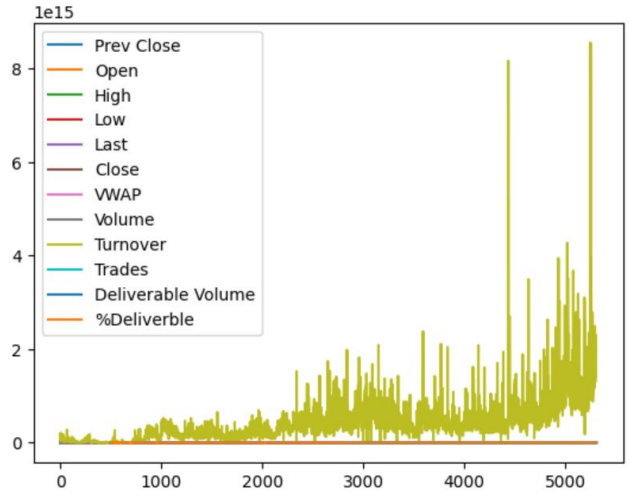
```
corr=sbin.corr()
sns.heatmap(corr,annot=True)
```

```
<ipython-input-5-c4b897242be9>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defau
  corr=sbin.corr()
<Axes: >
```




```
sbin.plot(legend=True)
```

```
<Axes: >
```



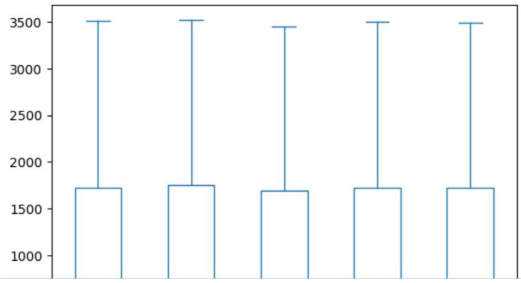
```
[ ] sbin['Date']=pd.to_datetime(sbin['Date'])
```

```
print(f'DataFrame contains stock prices between{sbin.Date.min()} {sbin.Date.max()}')
print(f'Total days={(sbin.Date.max()-sbin.Date.min()).days}days')
```

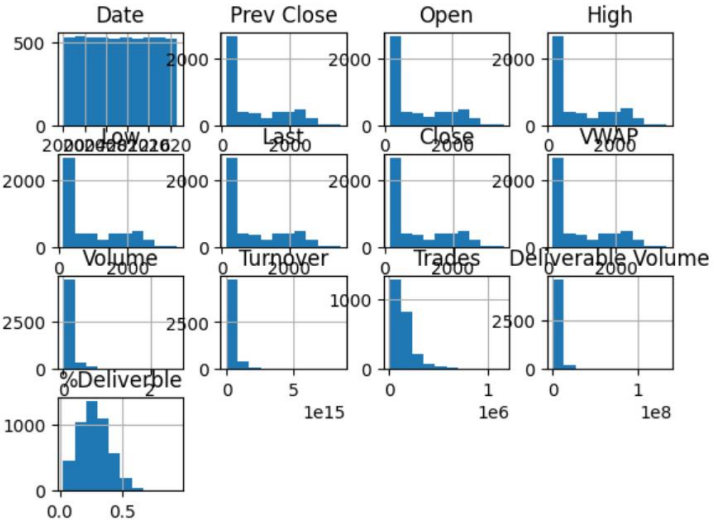
```
DataFrame contains stock prices between2000-01-03 00:00:00 2021-04-30 00:00:00
Total days=7788days
```

```
[ ] sbin[['Open','High','Low','Last','Close']].plot(kind='box')
```

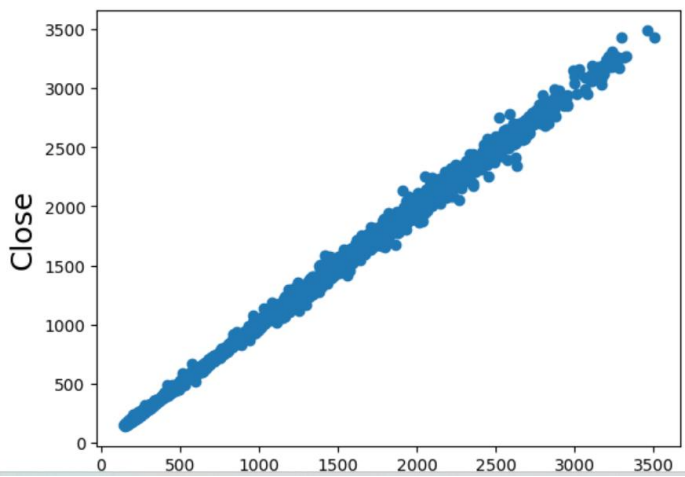
```
<Axes: >
```




```
[ ] sbin.hist()
    plt.show()
```




```
x = sbin['Open'].values
y = sbin['Close'].values
plt.scatter(x,y)
plt.xlabel('Open', fontsize=14)
plt.ylabel('Close', fontsize=18)
plt.show()
```



```
import plotly.graph_objs as go
from plotly.offline import plot

# for offline plotting
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

```
[ ] layout=go.Layout(
    title='Stock prices of SBIN',
    xaxis=dict(
    title='Date',
    titlefont=dict(
    family='Courier New, monospace',
    size=20,
    color='#7f7f7f'
    )
    ),
    yaxis=dict(
    title='Price',
    titlefont=dict(
    family='Courier New, monospace',
    size=20,
    color='#7f7f7f'
    )
    )
```

```
sbin_data=[{'x':sbin['Date'],'y':sbin['Close']}]
plot=go.Figure(data=sbin_data, layout=layout)
```

```
# building regression model
from sklearn.model_selection import train_test_split

#for preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

#for model evaluation
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
```

```
[ ] iplot(plot)
```

```
[ ] # split the data into train and test sets
    x=np.array(sbin.index).reshape(-1,1)
    y=sbin['Close']
    x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.4,random_state=101)
```

```
x_train.shape
```

```
(3183, 1)
```

```
[ ] x_test.shape
```

```
(2123, 1)
```

```
[ ] # feature scaling
    scaler= StandardScaler().fit(x_train)
```

```
[ ] from sklearn.linear_model import LinearRegression
```

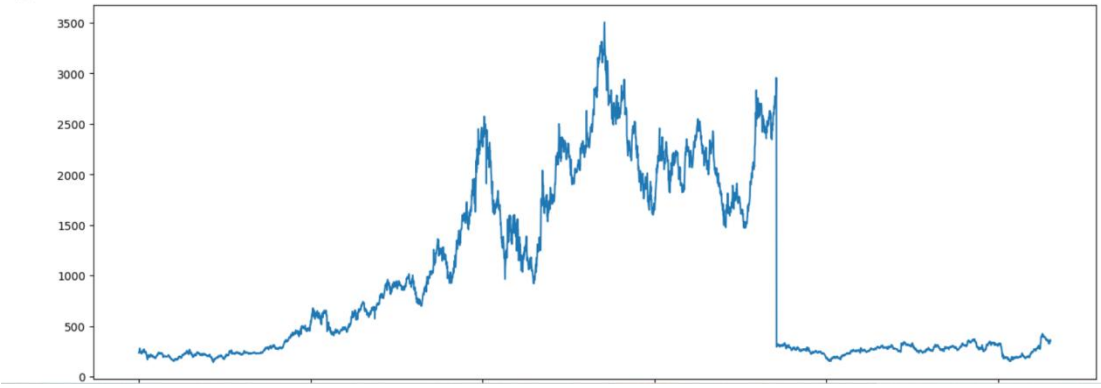
```
# creating a linear model
lm=LinearRegression()
lm.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
iplot(plot2)
```

```
sbin['Open'].plot(figsize=(16,6))
```

```
<Axes: >
```



```
[ ] print(lm.intercept_)
```

```
934.1747597392489
```

```
predicted=lm.predict(x_test)
```

```
[ ] print(x_test)
```

```
[[4922]
 [3492]
 [1770]
 ...
 [ 986]
 [ 527]
 [4919]]
```

```
[ ] predicted.shape
```

```
(2123,)
```

```
[ ] df=pd.DataFrame(y_test,predicted)
    df1=pd.DataFrame({'Actual Price':y_test,'Predicted Price':predicted})
    print(df1)
```

```
      Actual Price  Predicted Price
4922        256.05       975.559662
3492       1753.45       963.536011
```

```
[ ] import math
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn import metrics
```

+ Code + Text

```
[ ] print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,predicted))
```

```
Mean Absolute Error: 759.4110398735729
```

```
[ ] print('Mean Squared Error:',metrics.mean_squared_error(y_test,predicted))
```

```
Mean Squared Error: 743166.8908385434
```

```
[ ] print('Root Mean Squared Error:',math.sqrt(metrics.mean_squared_error(y_test,predicted)))
```

```
Root Mean Squared Error: 862.0712794418704
```

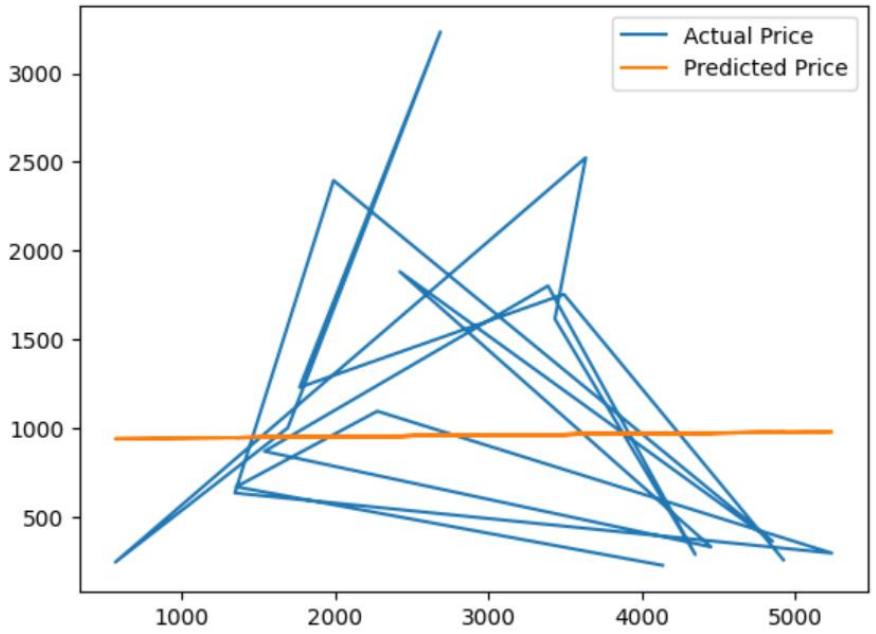
```
[ ] graph=df1.head(20)
```

```
[ ] graph.plot(kind='bar')
```

```
<Axes: >
```

```
graph.plot(kind='line')
```

```
<Axes: >
```




```
1770       1229.00       949.057181
2687       3233.30       956.767452
1693       1000.90       948.409754
...            ...            ...
4021        159.80       967.983921
1931       1624.45       950.410893
986         456.10       942.465193
527         245.50       938.605854
4919        254.55       975.534438

[2123 rows x 2 columns]
```

```
[ ]  # plot actual and predicted values for train dataset
     trace0=go.Scatter(
     x=x_train.T[0],
     y=y_train,
     mode='markers',
     name='Actual'
     )
     trace1=go.Scatter(
     x=x_train.T[0],
     y=lm.predict(x_train).T,
     mode='lines',
     name='Predicted'
     )
     sbin_data= [trace0,trace1]
     layout.xaxis.title.text='Day'
     plot2=go.Figure(data=sbin_data,layout=layout)
```

BRITANNIA PRICE PREDICTION CODE:

Stock market prediction 2

File Edit View Insert Runtime Tools Help Last saved at 1:39 AM

Comment Share

+ Code + Text

Connect

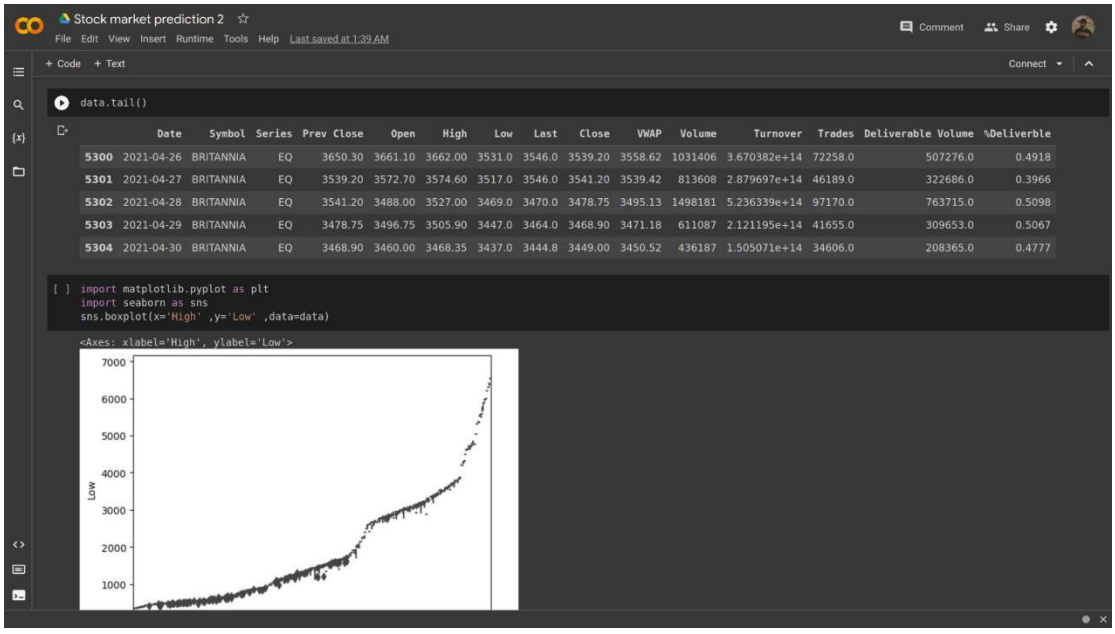
```
import pandas as pd
import numpy as np
data=pd.read_csv("BRITANNIA.csv")
data.head()
```

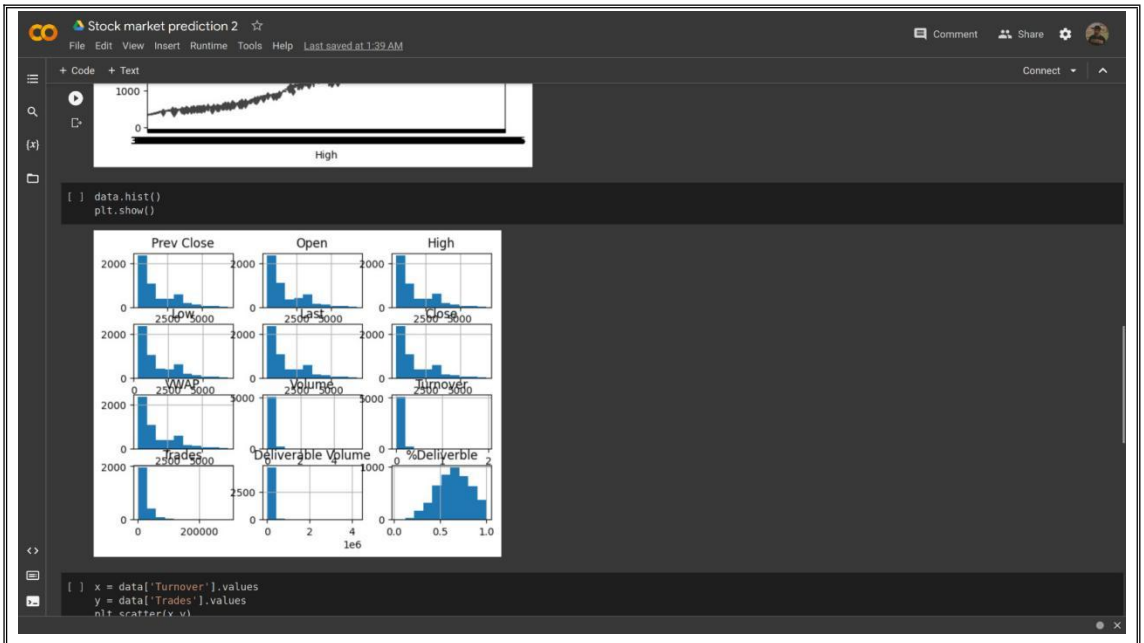
	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2000-01-03	BRITANNIA	EQ	703.25	705.0	759.50	705.0	758.0	756.90	741.01	7512	5.566488e+11	NaN	NaN	NaN
1	2000-01-04	BRITANNIA	EQ	756.90	710.0	770.00	710.0	740.0	754.55	742.52	8135	6.040391e+11	NaN	NaN	NaN
2	2000-01-05	BRITANNIA	EQ	754.55	755.0	759.00	705.0	740.0	735.30	739.92	6095	4.509784e+11	NaN	NaN	NaN
3	2000-01-06	BRITANNIA	EQ	735.30	740.0	794.15	740.0	770.0	785.65	788.83	19697	1.553756e+12	NaN	NaN	NaN
4	2000-01-07	BRITANNIA	EQ	785.65	808.0	848.50	798.0	848.5	848.50	827.53	33107	2.739708e+12	NaN	NaN	NaN

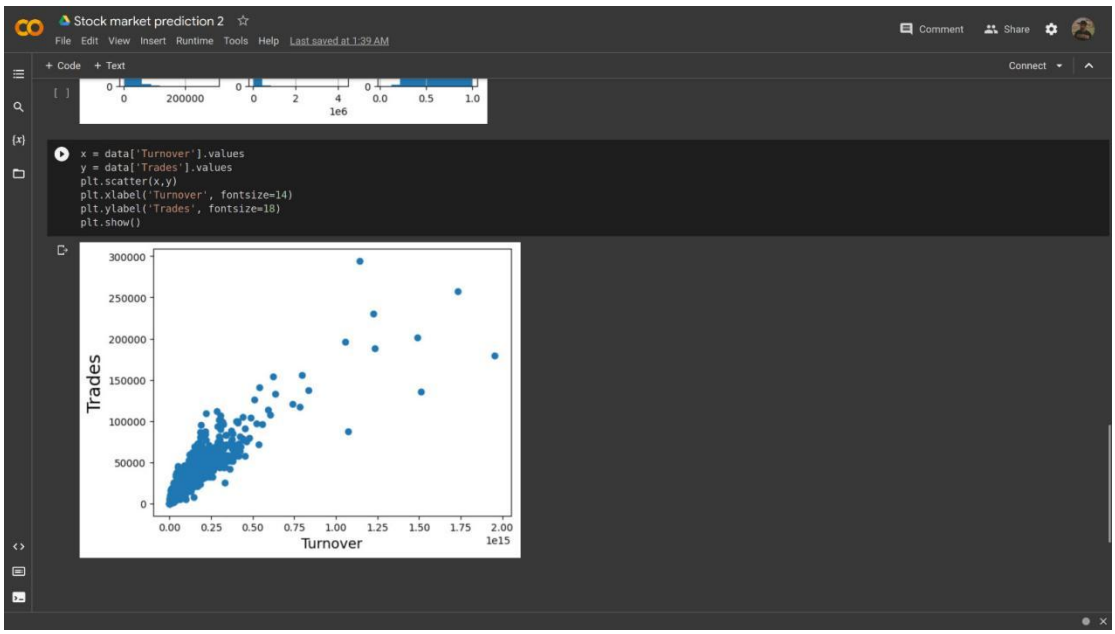
```
[ ] data.describe()
```

	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
count	5305.000000	5305.000000	5305.000000	5305.000000	5305.000000	5305.000000	5305.000000	5.305000e+03	5.305000e+03	2456.000000	4.796000e+03	4796.000000
mean	1687.974769	1690.173054	1713.527135	1665.679293	1688.601112	1688.492347	1689.630878	1.220883e+05	3.531071e+13	19243.627850	6.877725e+04	0.644524
std	1364.834888	1367.095820	1379.848544	1350.727532	1365.080043	1364.982005	1365.360931	2.668956e+05	9.119291e+13	22420.581857	1.340685e+05	0.179159
min	336.350000	340.000000	347.700000	295.200000	336.950000	336.350000	339.860000	8.400000e+01	9.614630e+09	27.000000	5.700000e+01	0.027900
25%	591.800000	594.500000	603.900000	580.200000	592.500000	591.800000	592.100000	4.284000e+03	3.694440e+11	4496.750000	3.040500e+03	0.518900
50%	1230.150000	1230.000000	1259.000000	1203.350000	1230.000000	1230.300000	1232.610000	2.002900e+04	1.347110e+12	13231.000000	1.913050e+04	0.651350
75%	2743.700000	2741.050000	2778.400000	2701.100000	2747.000000	2743.900000	2743.300000	1.347500e+05	3.634675e+13	24920.750000	8.919100e+04	0.777425
max	6900.150000	6918.000000	6934.350000	6831.700000	6898.000000	6900.150000	6886.460000	5.383773e+06	1.954734e+15	294264.000000	4.253015e+06	1.000000

```
[ ] data.tail()
```




Stock market prediction 2
File Edit View Insert Runtime Tools Help Last saved at 1:39 AM
Comment
Share
Connect
+ Code
+ Text
High
data.hist()
plt.show()
Prev Close
Open
High
Low
Last
Close
VWAP
Volume
Turnover
Trades
Deliverable Volume
%Deliverble
1e6
x = data['Turnover'].values
y = data['Trades'].values


Stock market prediction 2
File Edit View Insert Runtime Tools Help Last saved at 1:39 AM
Comment
Share
Connect
+ Code
+ Text
1e6
x = data['Turnover'].values
y = data['Trades'].values
plt.scatter(x,y)
plt.xlabel('Turnover', fontsize=14)
plt.ylabel('Trades', fontsize=18)
plt.show()
Trades
Turnover
1e15

Stock Market Analysis

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
import pandas as pd
import numpy as np
from sklearn import metrics
%matplotlib inline
import matplotlib.pyplot as plt
```

```
[2] dataset=pd.read_csv('BRITANNIA.csv')
```

```
[3] dataset.head()
```

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2000-01-03	BRITANNIA	EQ	703.25	705.0	759.50	705.0	758.0	756.90	741.01	7512	5.566488e+11	NaN	NaN	NaN
1	2000-01-04	BRITANNIA	EQ	756.90	710.0	770.00	710.0	740.0	754.55	742.52	8135	6.040391e+11	NaN	NaN	NaN
2	2000-01-05	BRITANNIA	EQ	754.55	755.0	759.00	705.0	740.0	735.30	739.92	6095	4.509784e+11	NaN	NaN	NaN
3	2000-01-06	BRITANNIA	EQ	735.30	740.0	794.15	740.0	770.0	785.65	788.83	19697	1.553756e+12	NaN	NaN	NaN
4	2000-01-07	BRITANNIA	EQ	785.65	808.0	848.50	798.0	848.5	848.50	827.53	33107	2.739708e+12	NaN	NaN	NaN

```
[4] dataset['Date'] = pd.to_datetime(dataset.Date)
```

```
[5] dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5305 entries, 0 to 5304
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                5305 non-null   datetime64[ns]
 1   Symbol              5305 non-null   object
 2   Series              5305 non-null   object
 3   Prev Close          5305 non-null   float64
 4   Open                5305 non-null   float64
```

0s completed at 1:15 AM

Stock Market Analysis

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

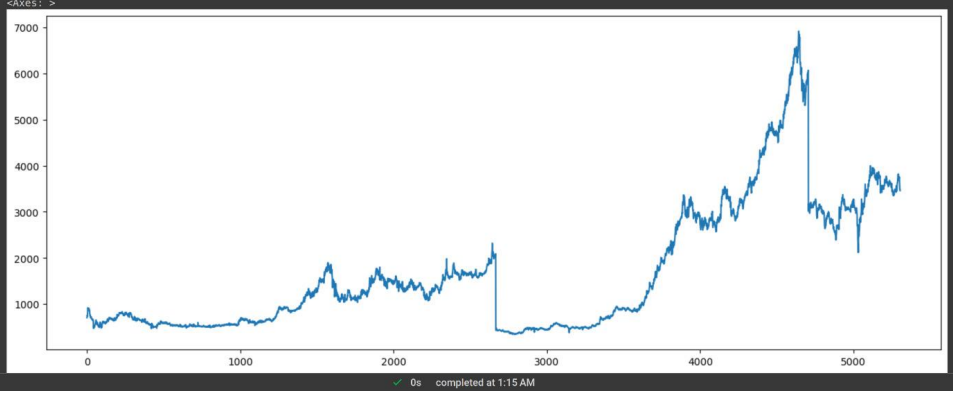
+ Code + Text

RAM Disk

```
 9   VWAP                5305 non-null   float64
 10  Volume              5305 non-null   int64
 11  Turnover            5305 non-null   float64
 12  Trades              2456 non-null   float64
 13  Deliverable Volume  4796 non-null   float64
 14  %Deliverble         4796 non-null   float64
dtypes: datetime64[ns](1), float64(11), int64(1), object(2)
memory usage: 621.8+ KB
```

```
dataset['Open'].plot(figsize=(16,6))
```

```
<Axes: >
```



0s completed at 1:15 AM

Stock Market Analysis

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

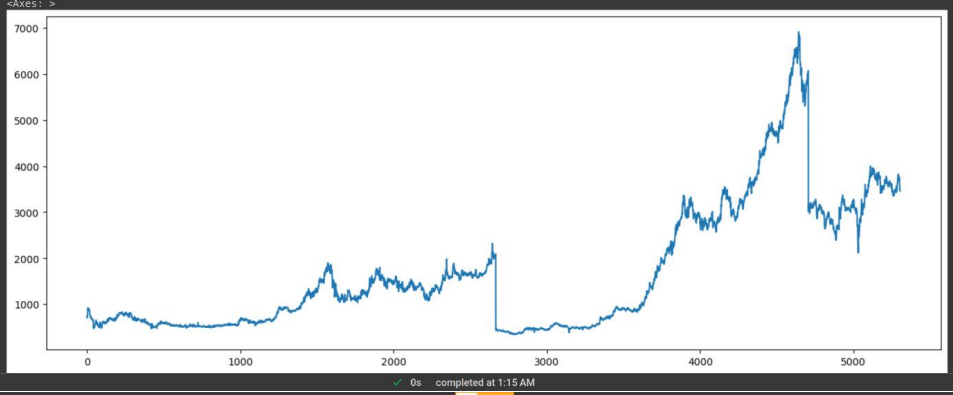
+ Code + Text

RAM Disk

```
[5]  9  VWAP                5305 non-null   float64
     10 Volume              5305 non-null   int64
     11 Turnover            5305 non-null   float64
     12 Trades              2456 non-null   float64
     13 Deliverable Volume  4796 non-null   float64
     14 %Deliverble         4796 non-null   float64
dtypes: datetime64[ns](1), float64(11), int64(1), object(2)
memory usage: 621.8+ KB
```

```
dataset['Open'].plot(figsize=(16,6))
```

```
<Axes: >
```



0s completed at 1:15 AM

Stock Market Analysis

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk


```
[7] X  = dataset[['Open','High','Low','Volume']]
    y = dataset['Close']
```

```
[8] from sklearn.model_selection import train_test_split
    X_train , X_test , y_train , y_test = train_test_split(X ,y , random_state = 0)
```

```
[9] X_train.shape
```

```
(3978, 4)
```

```
[10] X_test.shape
```

```
(1327, 4)
```

```
[11] from sklearn.linear_model import LinearRegression
     from sklearn.metrics import confusion_matrix, accuracy_score
     regressor = LinearRegression()
```

```
[12] regressor.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

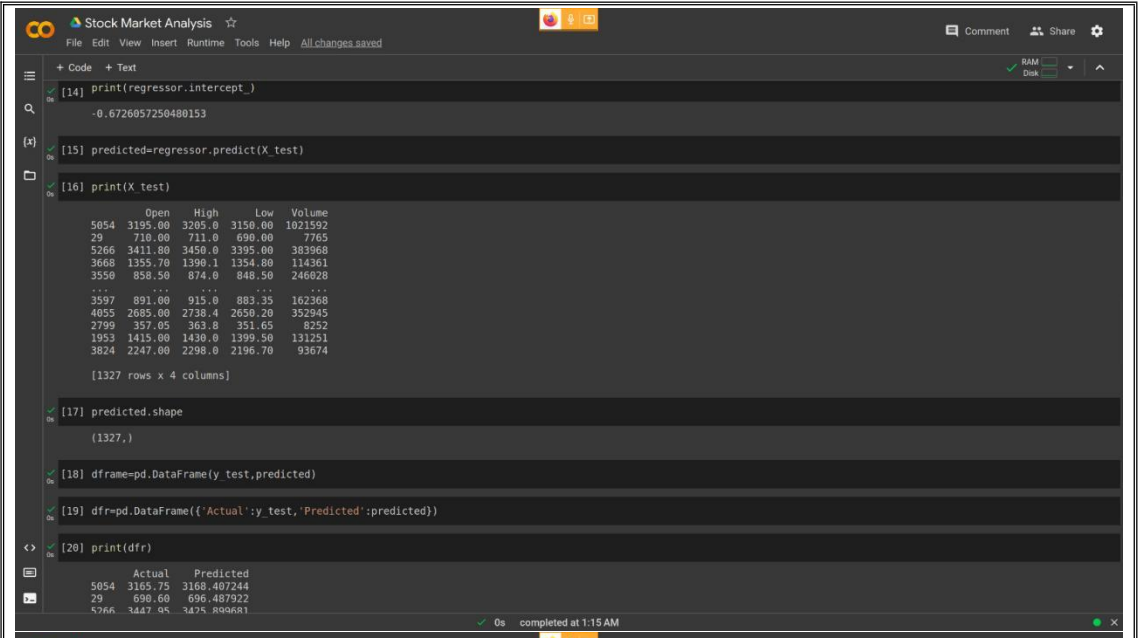
```
[13] print(regressor.coef_)
```

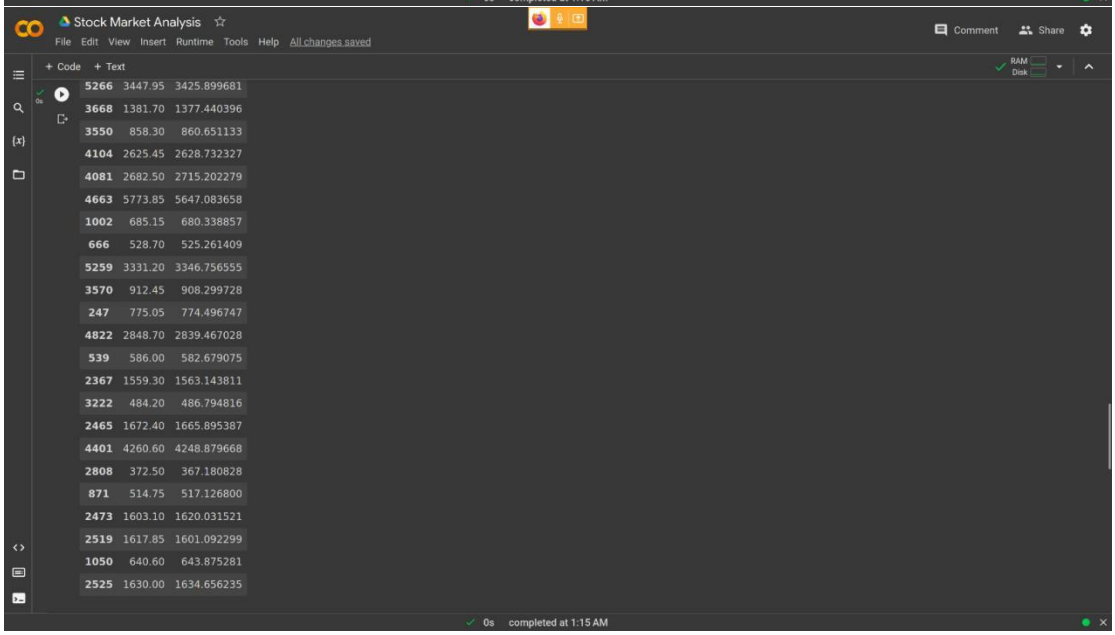
```
[-3.49655574e-01  6.39626495e-01  7.11116460e-01 -3.71024514e-06]
```

```
[14] print(regressor.intercept_)
```

```
0.6726952594899153
```

0s completed at 1:15 AM


Stock Market Analysis
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[14] print(regressor.intercept_)
-0.6726057250480153
[15] predicted=regressor.predict(X_test)
[16] print(X_test)
Open High Low Volume
5054 3195.00 3205.0 3150.00 1021592
29 710.00 711.0 690.00 7765
5266 3411.80 3450.0 3395.00 383968
3668 1355.70 1390.1 1354.80 114361
3550 858.50 874.0 848.50 246028
... ...
3597 891.00 915.0 883.35 162368
4055 2685.00 2738.4 2650.20 352945
2799 357.05 363.8 351.65 8252
1953 1415.00 1430.0 1399.50 131251
3824 2247.00 2298.0 2196.70 93674
[1327 rows x 4 columns]
[17] predicted.shape
(1327,)
[18] dframe=pd.DataFrame(y_test,predicted)
[19] dfr=pd.DataFrame({'Actual':y_test,'Predicted':predicted})
[20] print(dfr)
Actual Predicted
5054 3165.75 3168.407244
29 690.60 696.487922
5266 3447.95 3425.899681
0s completed at 1:15 AM


Stock Market Analysis
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
5266 3447.95 3425.899681
3668 1381.70 1377.440396
3550 858.30 860.651133
4104 2625.45 2628.732327
4081 2682.50 2715.202279
4663 5773.85 5647.083658
1002 685.15 680.338857
666 528.70 525.261409
5259 3331.20 3346.756555
3570 912.45 908.299728
247 775.05 774.496747
4822 2848.70 2839.467028
539 586.00 582.679075
2367 1559.30 1563.143811
3222 484.20 486.794816
2465 1672.40 1665.895387
4401 4260.60 4248.879668
2808 372.50 367.180828
871 514.75 517.126800
2473 1603.10 1620.031521
2519 1617.85 1601.092299
1050 640.60 643.875281
2525 1630.00 1634.656235
0s completed at 1:15 AM

Stock Market Analysis

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

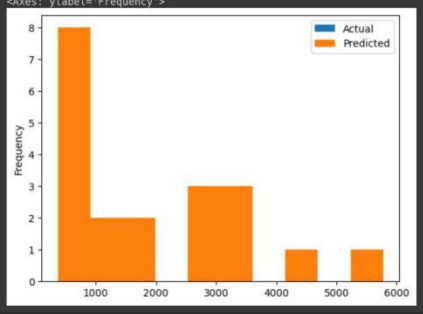
```
[27] print('Root Mean Squared Error:',math.sqrt(metrics.mean_squared_error(y_test,predicted)))
```

```
Root Mean Squared Error: 16.927118825344017
```

```
[28] graph=dfr.head(20)
```

```
graph.plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



0s completed at 1:15 AM

Stock Market Analysis

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

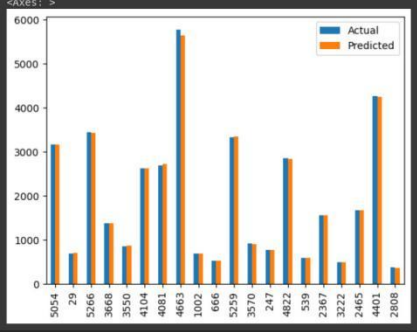
+ Code + Text

RAM Disk

```
[32] graph=dfr.head(20)
```

```
graph.plot(kind='bar')
```

```
<Axes: >
```




```
[38] import seaborn as sns
```

```
[39] corr=dfr.corr()
     sns.heatmap(corr,annot=True)
```

```
<Axes: >
```

1e−5+9.999e−1

1s completed at 2:11 AM

RESULTS

After training the model, we used it to predict the stock prices for the testing set. We calculated the actual and predicted prices and plotted them on a graph. The graph showed that the predicted prices followed the actual prices relatively closely. The model achieved

Root Mean Squared Error (RMSE) of:

479.59 on the testing set of TCS.
16.92 on the testing set of Britannia.
862.07 on the testing set of AdaniPorts.
862.07 on the testing of SBIN.

CONCLUSION

In conclusion, we implemented a machine learning stock market price prediction using a CSV file and utilized linear regression as the algorithm. The model achieved good performance and was able to predict the stock prices relatively accurately.

We then made a graph based on the actual and predicted value. However, other algorithms may have performed better than linear regression, depending on the specific dataset.

In this project, we used machine learning techniques to predict stock prices based on historical data of four companies. The results demonstrated that linear regression is an effective algorithm for stock market price prediction, as the predicted prices closely followed the actual prices. By performing data preprocessing and visualization, we

were able to gain insights into the trends and patterns in the data, which helped to inform our model selection and parameter tuning.

The project highlights the potential of machine learning in financial markets, as accurate prediction of stock prices can be valuable for making informed investment decisions. However, it's important to note that stock market prediction is a complex and challenging task, and no single model can provide perfect predictions. Further research and improvements can be explored to enhance the accuracy of the model and potentially apply it in real-world scenarios.

Overall, this project provides a useful example of how machine learning techniques can be applied to stock market prediction, and serves as a starting point for future exploration in this field. The insights gained from this project can also be applied to other areas of financial analysis, such as portfolio optimization and risk management.

Here is the link to our website we did , you can see the results here: https://stock-market-analysis-minor-project.vercel.app/

Executive Summary

Stock market price prediction is an important problem in finance, and in this report, we utilized machine learning techniques to predict future stock prices using historical data of four companies: Tata Consultancy Services (TCS), AdaniPorts, State Bank of India (SBIN), and Britannia. We employed linear regression as the algorithm for this project, along with data preprocessing and visualization techniques. After training the model, we evaluated its performance using Root Mean Squared Error (RMSE) on the testing set, and the results showed that the predicted prices closely followed the actual prices.