STOCK MARKET PREDICTION (DATA MINING TECHNIQUES)

Abstract

In general way if we say that we want to predict the stock which means that we want to know whether the price of a stock will go high or low. The price will rise if tons of individuals want to shop for a stock. The value will drop if there are a much bigger number of sellers than buyers. Therefore, people generally will take the assistance of a broker which can assist the client in order that they will observe return. Therefore, we'll attempt to take history of the stock and analyse it by using data processing techniques and check out to predict the worth of the stock using linear regression and decision tree regressor but it's very difficult to predict share prices because it depends upon such huge numbers of things such organization financial status and national policy then on. Forecasting stock return involves an assumption that fundamental information publicly available within the past has some predictive relationships to the longer-term stock returns. This study tries to assist the investors within the stock exchange to make a decision the higher timing for purchasing or selling stocks supported the knowledge extracted from the historical prices of such stocks. The stock exchange is actually hard to model with any reasonable accuracy. to realize those objectives, people use the techniques of fundamental analysis, where trading rules are developed supported, the knowledge related to macroeconomics, industry, and company. The authors said that fundamental analysis assumes that the worth of a stock depends on its intrinsic value and expected return on investment. Analysing the corporate 's operations and therefore the market during which the company is working can do that. Consequently, the stock prices are often predicted reasonably well. However, for short- and medium-term speculations, fundamental analysis is usually not suitable.

Keywords

Linear Regression

Decision Tree Regressor

Dataset Link

https://www.kaggle.com/bibinvargheset/tatasteel

Introduction

The dataset here is a TATA STEEL Stock data. We want to know better about the future of the stock price by training the stock data of the previous days. Specifically, here the problem is a regression problem where we are trying to predict the dependent variable with the help of the information contained in the other variables. Data mining technique have been effectively revealed to produce high forecasting accurateness of movement of stock price. Now a days, as an alternative of a particular method, traders have to use various predicting methods to increase several signals and more information about the market's future. Data mining methods have been introduced for forecasting of movement indication of stock market index. Data mining techniques have a more successful act in predicting various fields such as policy, economy and engineering compared to usual statistical techniques by discovering unknown information of data.

Problem Statement

Everyone want to be rich in his life with low efforts and great advantages. Similarly, we want to look in our future with inner most desire as we do not want to take risks or we want to decrease risk factor. Stock market is a place where, selling and purchasing can provide future aims of life. So, the question is how to predict the market? How machine learning algorithms can help in prediction of stock market?

Software Used

We have used python for our project. Python can be used for wide range of applications. We are using various libraries in python.

- Numpy
- Matplotlib
- Pandas
- Sci-Kit Learn

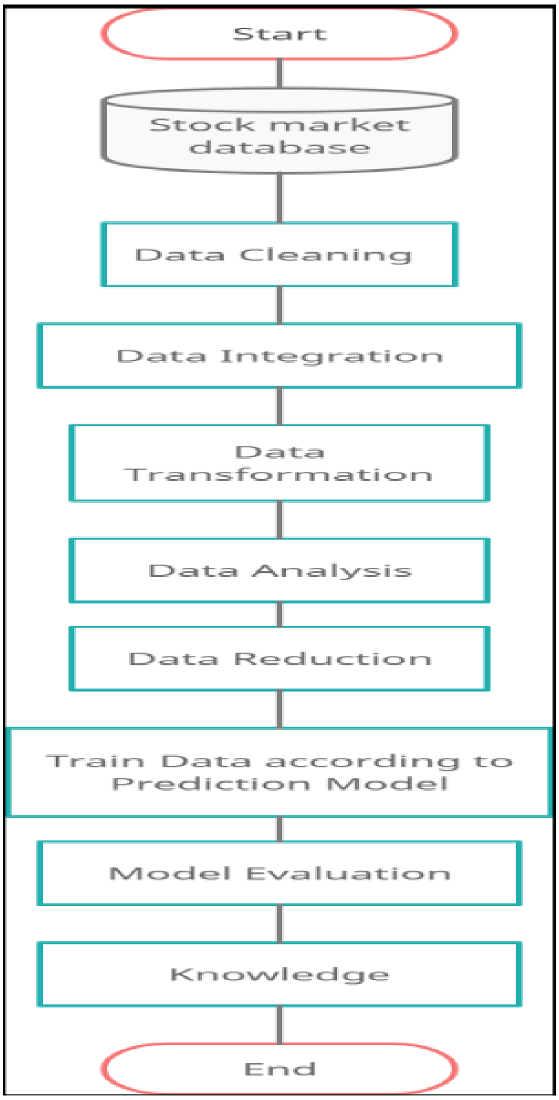
Platform Used

Google Colab

ALGORITHMIC STEPS

- Importing necessary modules
- Reading dataset
- Dropping Unknown 'Unnamed row'
- Visualizing with columns - open low high close
- Splitting dataset into test and train
- Calculating X_Future (Prediction)
- Predicting using decision tree and linear regressor
- Calculating RMSE score and accuracy
- Evaluating and visualizing the predictions

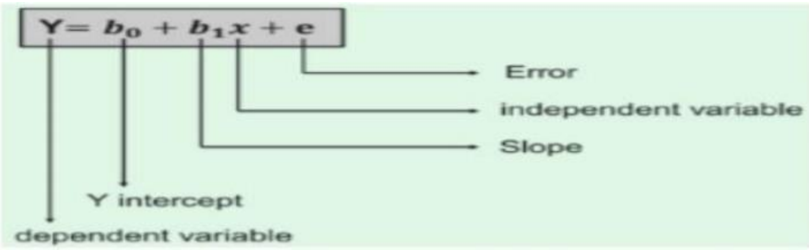
WORKFLOW DIAGRAM



MODULES AND DESCRIPTION

Linear Regression

The process of finding a straight line (as by least squares) that best approximates a set of points on a graph is linear regression. Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.



The cost function provides the best possible values for b0 and b1 to make the best fit line for the data points. We do it by converting this problem into a minimization problem to get the best values for b0 and b1. The error is minimized in this problem between the actual value and the predicted value.

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

We choose the function above to minimize the error. We square the error difference and sum the error over all data points, the division between the total number of data points. Then, the produced value provides the averaged square error over all data points (MSE). we change the values of b0 and b1 so that the MSE value is settled at the minimum.

Decision Tree Regressor

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs and utility.

Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

The branches/edges represent the result of the node and the nodes have either:

- Conditions [Decision Nodes]
- Result [End Nodes]

The branches/edges represent the truth/falsity of the statement and takes makes a decision based on that. Decision Tree Regression:

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

Discrete output example: A weather prediction model that predicts whether or not there'll be rain in a particular day.

Continuous output example: A profit prediction model that states the probable profit that can be generated from the sale of a product.

$$I\left(P(v_1), \ldots, P(v_i)\right) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i)$$

$$Gain(Attribute) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

RMSE (Evaluation Metrix)

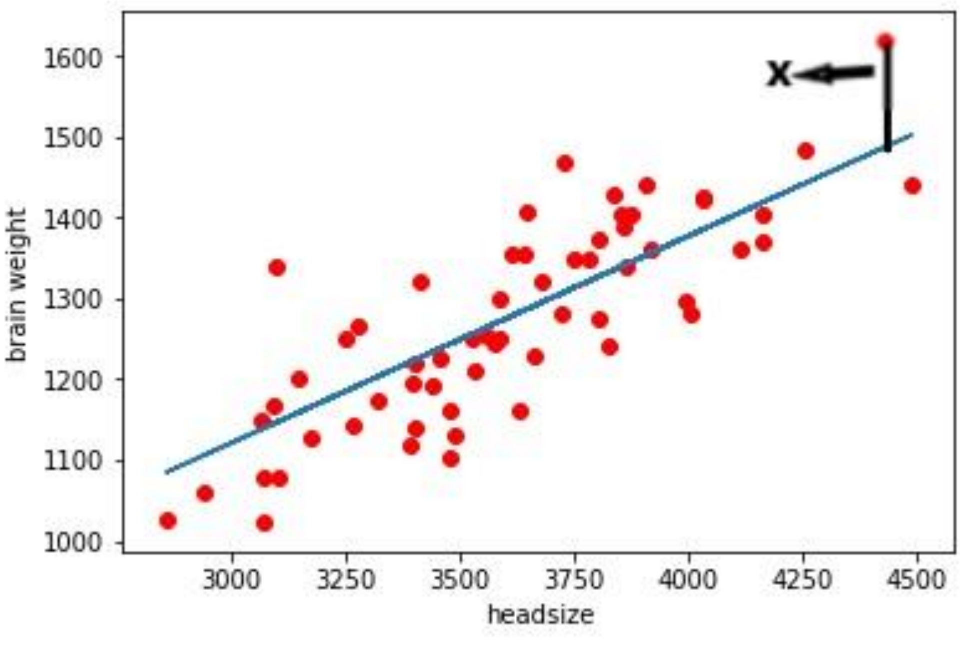
It follows an assumption that error is unbiased and follow a normal distribution.

Let a= (predicted value- actual value) ^2

Let b= mean of a = a (for single value)

Then RMSE= square root of b

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$



As you can see in this scattered graph the red dots are the actual values and the blue line is the set of predicted values drawn by our model. Here X represents the distance between the actual value and the predicted line this line represents the error, similarly, we can draw straight lines from each red dot to the blue line. Taking mean of all those distances and squaring them and finally taking the root will give us RMSE of our model.

IMPLEMENTATION(S) ⬇

```
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
[ ] import numpy as np
    import pandas as pd
```

1. Data Collection ⬇

+ Code + Text

```
id = '1F9PTxteeWx1iZR1QcKD0irhJL9srU8NY'

downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile('tata.csv')
df = pd.read_csv('tata.csv')
df.head(10)
```

	scrip	date	time	Open	High	Low	Close	Volume	Unnamed: 8
0	TATASTEEL	24-06-2013	09:16	270.20	270.70	269.10	269.35	69119	NaN
1	TATASTEEL	24-06-2013	09:17	269.30	270.10	269.25	270.00	30508	NaN
2	TATASTEEL	24-06-2013	09:18	270.00	271.10	269.80	271.00	16785	NaN
3	TATASTEEL	24-06-2013	09:19	271.00	271.00	270.30	270.40	22661	NaN
4	TATASTEEL	24-06-2013	09:20	270.30	270.40	270.10	270.20	30794	NaN
5	TATASTEEL	24-06-2013	09:21	270.10	270.30	270.05	270.25	31049	NaN
6	TATASTEEL	24-06-2013	09:22	270.25	270.85	270.25	270.65	15813	NaN
7	TATASTEEL	24-06-2013	09:23	270.80	271.60	270.25	270.40	32092	NaN
8	TATASTEEL	24-06-2013	09:24	270.15	271.10	270.15	271.05	15286	NaN
9	TATASTEEL	24-06-2013	09:25	271.10	271.10	270.70	270.75	8634	NaN

2. Data Preprocessing ⬇

```
df[['Close']].fillna(value=df[['Close']].mean(),inplace=True)
```

```
[ ] df.drop('Unnamed: 8' , axis = 'columns' , inplace = True)
```

```
df.drop('time' , axis = 'columns' , inplace = True)
df.head(10)
```

	Open	High	Low	Close	Volume	Prediction
0	270.20	270.70	269.10	269.35	69119	272.65
1	269.30	270.10	269.25	270.00	30508	273.60
2	270.00	271.10	269.80	271.00	16785	274.15
3	271.00	271.00	270.30	270.40	22661	273.75
4	270.30	270.40	270.10	270.20	30794	273.70
5	270.10	270.30	270.05	270.25	31049	273.60
6	270.25	270.85	270.25	270.65	15813	273.90
7	270.80	271.60	270.25	270.40	32092	274.50
8	270.15	271.10	270.15	271.05	15286	274.35
9	271.10	271.10	270.70	270.75	8634	273.90

4. Data Analysis ⬇

\+ Code + Text

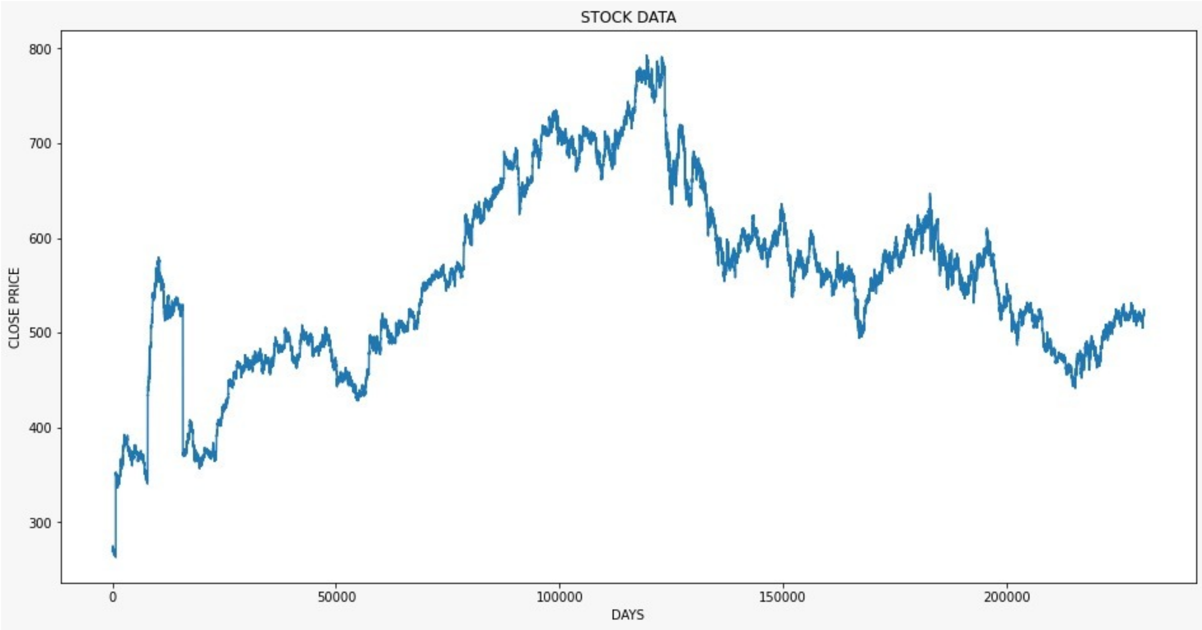
```
[ ] df.describe()
```

	Open	High	Low	Close	Volume	Prediction
count	230815.000000	230815.000000	230815.000000	230815.000000	2.308150e+05	230790.000000
mean	556.270728	556.586028	555.947558	556.266577	1.781764e+04	556.297438
std	97.322036	97.360675	97.283910	97.321571	2.624750e+04	97.281660
min	263.400000	263.400000	262.600000	263.100000	4.000000e+00	263.100000
25%	488.700000	489.000000	488.400000	488.700000	5.108000e+03	488.700000
50%	555.850000	556.200000	555.500000	555.850000	1.019700e+04	555.900000
75%	614.900000	615.250000	614.500000	614.850000	2.059900e+04	614.850000
max	792.750000	793.000000	792.050000	792.750000	2.368212e+06	792.750000

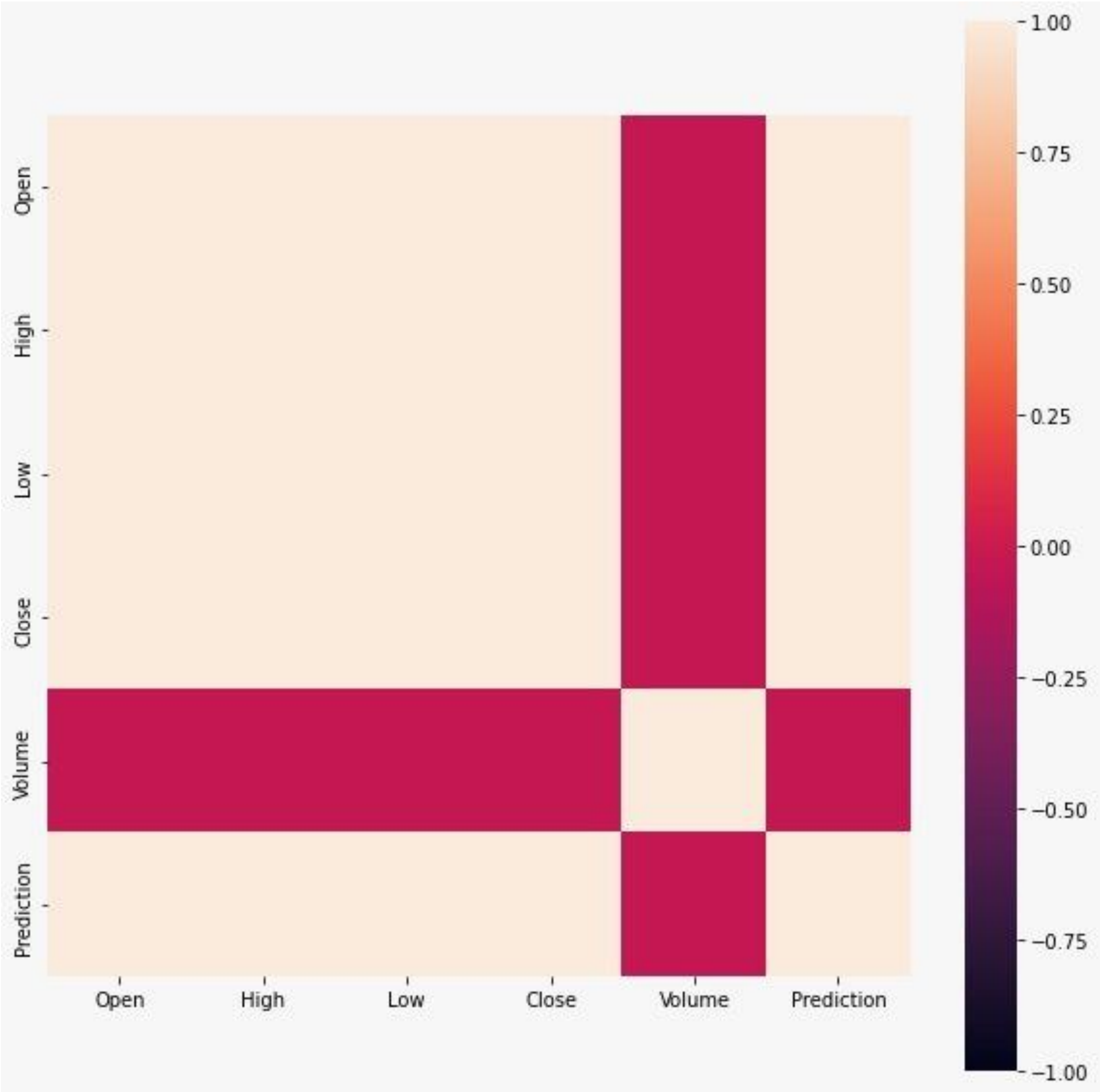
3. Data Transformation ⬇

```
[ ] pd.options.mode.chained_assignment = None

    future_days = 25

    df[['Prediction']] = df[['Close']].shift(-future_days)
    df.tail(10)
```

	scrip	date	time	Open	High	Low	Close	Volume	Unnamed: 8	Prediction
230805	TATASTEEL	29-03-2019	15:21	520.00	520.95	519.95	520.95	69733	NaN	NaN
230806	TATASTEEL	29-03-2019	15:22	520.85	521.00	520.75	520.80	54403	NaN	NaN
230807	TATASTEEL	29-03-2019	15:23	521.00	521.70	520.85	521.45	55252	NaN	NaN
230808	TATASTEEL	29-03-2019	15:24	521.45	521.55	520.60	520.60	111182	NaN	NaN
230809	TATASTEEL	29-03-2019	15:25	520.90	520.90	520.45	520.60	41907	NaN	NaN
230810	TATASTEEL	29-03-2019	15:26	520.55	520.85	520.35	520.35	37074	NaN	NaN
230811	TATASTEEL	29-03-2019	15:27	520.40	520.60	520.20	520.50	26332	NaN	NaN
230812	TATASTEEL	29-03-2019	15:28	520.35	520.35	519.65	519.80	51001	NaN	NaN
230813	TATASTEEL	29-03-2019	15:29	519.25	519.95	519.25	519.50	17073	NaN	NaN
230814	TATASTEEL	29-03-2019	15:30	519.70	520.50	518.50	520.50	47406	NaN	NaN



STOCK DATA
CLOSE PRICE
DAYS
800
700
600
500
400
300
0
50000
100000
150000
200000



Open
High
Low
Close
Volume
Prediction
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

4(A). Relevant Attribute Selection ⬇

```
# Lets create the feature dataset (Lets say X) & transform it into a numpy array
# & removing the last 'Z' rows per day

X = np.array(df.drop(['Prediction'],1)[:-future_days])
print(X)
```

```
[[  270.2     270.7     269.1     269.35 69119.  ]
 [  269.3     270.1     269.25    270.    30508.  ]
 [  270.      271.1     269.8     271.    16785.  ]
 ...
 [  522.4     522.45    521.75    522.1   36608.  ]
 [  522.15    522.4     522.      522.1   33187.  ]
 [  522.2     522.5     522.      522.15 33555.  ]]
```

```
# Lets create a target data set (Y) and convert it into a numpy array and all of
# the target values except 'Z' rows

Y = np.array(df['Prediction'])[:-future_days]
print(Y)
```

```
[272.65 273.6  274.15 ... 519.8  519.5  520.5 ]
```

4(B). Dataset Split (Training & Testing) ⬇

```
# As the dataset is big enough , splitting will be easy into Training dataset
# and Testing dataset. For any model , the size of training dataset should
# surpass the size of testing dataset , because training a model is a laborious
# procedure

# Thus we shall split the dataset into 75% (training) and 25% (testing) portions

# For the aforementioned step , we shall be using scikit-learn library

from sklearn import model_selection as mls

x_train , x_test , y_train , y_test = mls.train_test_split(X,Y,test_size = 0.25)
```

4(C). Model Creation ⬇

```
# Creation of models begin

# Starting with Decision Tree Regressor Model
# Why Regressor ? Because each value under an attribute is unique

from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor().fit(x_train , y_train)


# Lets create the Linear Regression model
# Why Regressor ? Because each value under an attribute is unique

from sklearn.linear_model import LinearRegression
linreg = LinearRegression().fit(x_train , y_train)
```

4(D). Model Prediction ⬇

```
# Model Tree Prediction
tree_pred = tree.predict(x_test)
print(tree_pred)
```

```
[633.15 727.5   664.1   ... 583.5   750.4   483.2 ]
```

```
# Model Linear Regression Prediction
linreg_pred = linreg.predict(x_test)
print(linreg_pred)
```

```
[635.63935401 724.50980358 664.49474302 ... 584.26331195 751.28817138
 482.41689671]
```

4(E). Model Evaluation ⬇

```
[ ] # Let us compare the results with respect to various
    # evaluation metrics

    # For this , we have to import the metrics library of Scikit-learn

    # let us first check for Decision Tree
    from sklearn import metrics
    import math

    # NOTE : don't get surprised if you see the range of values taken from 173117th row
    # onwards. Remember, we had mentioned earlier that we will be predicting stocks
    # for the upcoming days.

    print('Mean Absolute Error (MAE) : ' , metrics.mean_absolute_error(df['Close'][173117:],tree_pred))
    print('Mean Squared Error (MSE) : ' , metrics.mean_squared_error(df['Close'][173117:],tree_pred))
    print('Root Mean Squared Error (RMSE) : ' , math.sqrt(metrics.mean_absolute_error(df['Close'][173117:],tree_pred)))
```

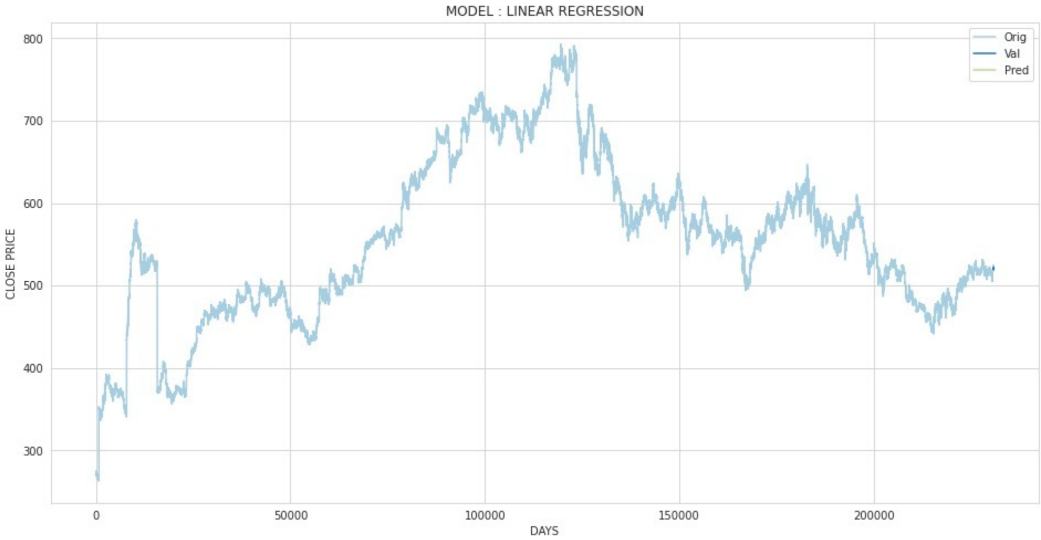
```
Mean Absolute Error (MAE) :  87.31900932441332
Mean Squared Error (MSE) :  12078.814653887483
Root Mean Squared Error (RMSE) :  9.344464100440074
```

```
# Now let us check for Linear Regression

print('Mean Absolute Error (MAE) : ' , metrics.mean_absolute_error(df['Close'][173117:],linreg_pred))
print('Mean Squared Error (MSE) : ' , metrics.mean_squared_error(df['Close'][173117:],linreg_pred))
print('Root Mean Squared Error (RMSE) : ' , math.sqrt(metrics.mean_absolute_error(df['Close'][173117:],linreg_pred)))
```

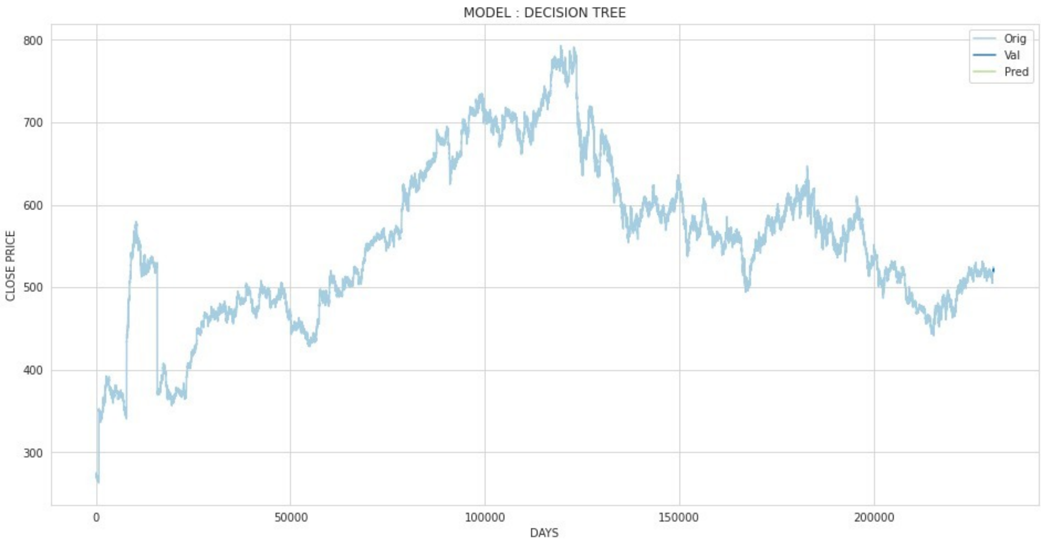
```
Mean Absolute Error (MAE) :  87.2485129045042
Mean Squared Error (MSE) :  12061.976989297467
Root Mean Squared Error (RMSE) :  9.340691243398648
```

5. Data Visualization ⬇

```
# Here , we will again use Matplotlibs' PyPlot library

preds = linreg_pred

# The result of the Linear regression is in the form of numpy array
# But , We cannot concatenate dataframe column values with a numpy array

# Thus , we either have to convert the numpy array into a Pandas Series OR
# into a dataframe column. Converting to Pandas series is relatively easy.
preds = pd.Series(preds)

# NOTE : Pandas Series is a one-dimensional labeled array capable of
# holding data of any type (integer, string, float, python objects, etc.).

# Now lets visualize the result(s) of Linear Regression
valid = df[X.shape[0]:]
valid['Prediction'].append(preds)
plt.figure(figsize = (16,8))
plt.title('MODEL : LINEAR REGRESSION')
plt.xlabel('DAYS')
plt.ylabel('CLOSE PRICE')
plt.plot(df['Close'])
plt.plot(valid[['Close' , 'Prediction']])
plt.legend(['Orig' , 'Val' , "Pred"])
plt.show()
```




```python
# Now we shall visualize the results of Decision Tree

preds = tree_pred

preds = pd.Series(preds)

valid = df[X.shape[0]:]
valid['Prediction'].append(preds)
plt.figure(figsize = (16,8))
plt.title('MODEL : DECISION TREE')
plt.xlabel('DAYS')
plt.ylabel('CLOSE PRICE')
plt.plot(df['Close'])
plt.plot(valid[['Close' , 'Prediction']])
plt.legend(['Orig' , 'Val' , "Pred"])
plt.show()

# NOTE : IN the x-axis (DAYS) , there is a decimal point on each scale value
# The decimal point should be after 3rd digit from the right
```



6. COMPARITIVE STUDY / RESULTS AND DISCUSSION ⬇

[Linear Regression] : **RMSE SCORE =** `9.340691243398648`

[Decision Tree] : **RMSE SCORE =** `9.344464100440074`

RESULTS AND ANALYSIS

Root Mean Squared Error (RMSE)

- Linear Regression: – 9.3406
- Decision Tree: – 9.3444

Mean Absolute Error

- Linear Regression: 87.2485
- Decision Tree Regressor: 87.3190

Mean Squared Error

- Linear Regression: 12061.9769
- Decision Tree Regressor: 12078.8146

CONCLUSION

It is known that lower RMSE implies better result. Thus, after analysing the above result we conclude that Linear Regression model was about 0.0038% more efficient than Decision Tree model on the given data set.

EXPOSURE GAINED

Data mining technique have been effectively revealed to produce high forecasting accurateness of movement of stock price.

Now a days, as an alternative of a particular method, traders have to use various predicting methods to increase several signals and more information about the market's future. Data mining methods have been introduced for forecasting of movement indication of stock market index.

After completing the project, we gained knowledge about how linear regression and decision tree algorithm works.