**ADANIPORTS**

# ABSTRACTION:

**Adani Ports and Special Economic Zone Limited** is an Indian [multinational](https://en.wikipedia.org/wiki/Multinational_corporation) [port operator](https://en.wikipedia.org/wiki/Port_operator) and [logistics company](https://en.wikipedia.org/wiki/Logistics), based in [Ahmedabad](https://en.wikipedia.org/wiki/Ahmedabad), [India](https://en.wikipedia.org/wiki/India).[[8]](https://en.wikipedia.org/wiki/Adani_Ports_%26_SEZ#cite_note-8)[[9]](https://en.wikipedia.org/wiki/Adani_Ports_%26_SEZ#cite_note-9) APSEZ is India's largest private port operator with a network of 12 ports and [terminals](https://en.wikipedia.org/wiki/Container_port), including India's first deep water Transshipment Port [Vizhinjam International Seaport Thiruvananthapuram](https://en.wikipedia.org/wiki/Vizhinjam_International_Seaport_Thiruvananthapuram) (Trivandrum Seaport) and India's first port-based [SEZ](https://en.wikipedia.org/wiki/Special_economic_zone) at [Mundra](https://en.wikipedia.org/wiki/Mundra_Port).[[10]](https://en.wikipedia.org/wiki/Adani_Ports_%26_SEZ#cite_note-businesstoday.in-10)[[11]](https://en.wikipedia.org/wiki/Adani_Ports_%26_SEZ#cite_note-11)

indicators

# METHODOLOGY:

# Multiple linear regression:

Multiple linear regression (MLR), also known simply as multipleregression, is a **statistical technique that uses several explanatory variables to predict the outcome of a response variable**. Multiple regression is an extension of linear (OLS) regression that uses just one explanatory variable.

# RandomForestRegression:

RandomForestRegression is a supervised learning algorithm and bagging technique that uses an ensemble learning method for regression in machine learning

# ANN model:

**An Artificial Neural Network (ANN) is a computational model inspired by the structure and functioning of the human brain. It is a type of machine learning algorithm that is designed to recognize patterns and make predictions or decisions based on input data. ANNs can be used for various tasks, including classification, regression, and pattern recognition. They have been successfully applied in areas such as image and speech recognition, natural language processing, financial forecasting, and many other domains.**

## # IMPORTING LIBRARIES

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import confusion\_matrix,accuracy\_score

import math

from sklearn import metrics

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import confusion\_matrix,accuracy\_score

from sklearn.metrics import mean\_squared\_error,mean\_absolute\_error

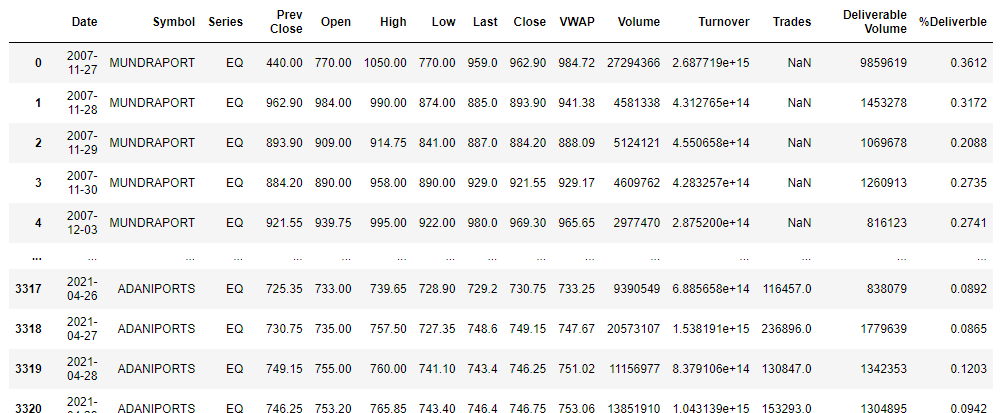
from sklearn.ensemble import RandomForestRegressorv

# # READING DATASET

df=pd.read\_csv(‘**ADANIPORTS**.csv')

df

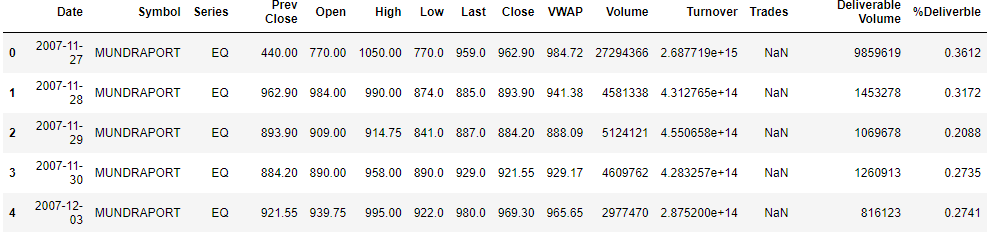
## output:



# # TO CHECK THE FIRST FIVE COLUMNS

df.head()

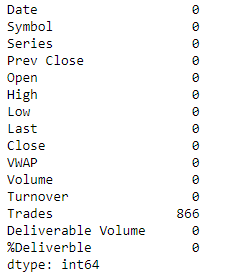
# output:



# # CHECKING MISSING VALUES

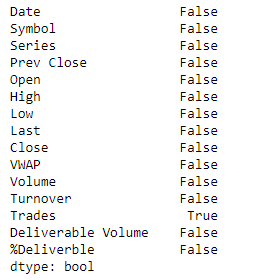
df.isnull().sum()

# output:



# df.isnull( ).any( )

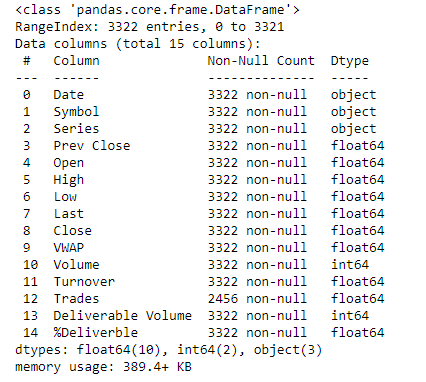
# output:



# # CHECKING INFORMATION ABOUT DATASET

df.info()

# output:



# #TO CHECK STATISTICS OF DATASET

df.describe()

# output:



print('lenghth of dataset:',len(df))

# output:



# TRAINING THE MODEL FOR MultipleLinearRegression

# splitting the dataset

x=df.drop('open','Trdes'**,** 'Symbol','Series',axis=1)

y=df[''open']

|  |  |
| --- | --- |

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,random\_state=0)

print(x\_train.shape)

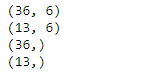
|  |  |
| --- | --- |

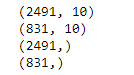
print(x\_test.shape)

print(y\_train.shape)

print(y\_test.shape

# output:





|  |  |
| --- | --- |

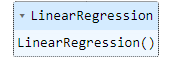
|  |  |
| --- | --- |

# # MODEL TRAINING

regressor=LinearRegression()

regressor.fit(x\_train,y\_train)

# output:



# #TO CHECK CO-EFFICIENT AND INTERCEPT

print(regressor.coef\_)

print(regressor.intercept\_)

# output:

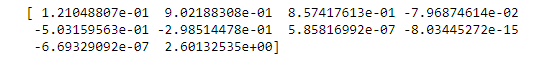


# # PREDICTION

predicted=regressor.predict(x\_test)

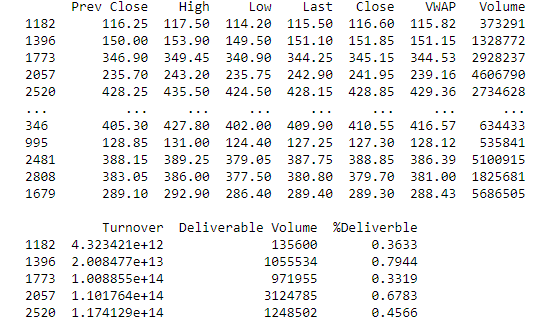
predicted

# output:



print(x\_test)

# output:



predicted.shape

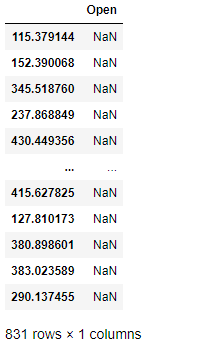
# output:



dframe=pd.DataFrame(y\_test,predicted)

dframe

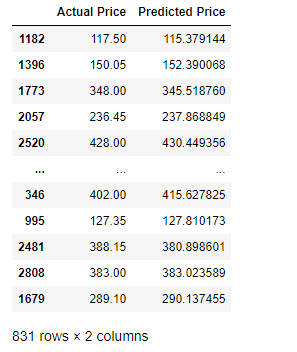
# output:



# # TO CHECK ACTUAL PRICE AND PREDICTED PRICE

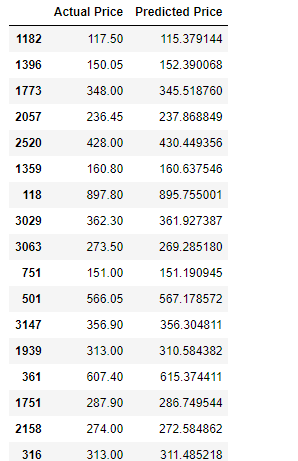
dfr=pd.DataFrame({'Actual Price':y\_test,'PredictedPrice':predicted})

dfr

output:

dfr.head(40)

# output:



# # EVALUATING THE MODEL

from sklearn.metrics import confusion\_matrix,accuracy\_score

train\_accuracy=regressor.score(x\_train,y\_train)

print('train\_accuracy:',train\_accuracy)

test\_accuracy=regressor.score(x\_test,y\_test)

print('test\_accuracy:',test\_accuracy)

# output:



import math

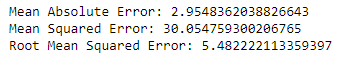
from sklearn import metrics

print('Mean Absolute Error:',metrics.mean\_absolute\_error(y\_test,predicted))

print('Mean Squared Error:',metrics.mean\_squared\_error(y\_test,predicted))

print('Root Mean Squared Error:',math.sqrt(metrics.mean\_squared\_error(y\_test,predicted)))

# output:



graph=dfr.head(20)

graph

# output:



**from**sklearn.model\_selection**import**train\_test\_split

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.90,random\_state**=**100)

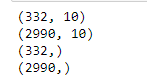
print(x\_train**.**shape)

print(x\_test**.**shape)

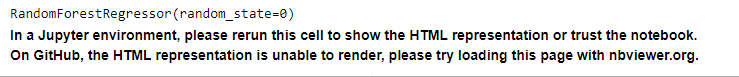
print(y\_train**.**shape)

print(y\_test**.**shape)

# output:



# output:



predicted=regressor.predict(x\_test)

predicted

# output:



from sklearn.metrics import confusion\_matrix,accuracy\_score

train\_accuracy=regressor.score(x\_train,y\_train)

print('train\_accuracy:',train\_accuracy)

R\_test\_accuracy=regressor.score(x\_test,y\_test)

print('test\_accuracy:',R\_test\_accuracy)

# output:



# # TO PLOT THE BAR GRAPH AND TO CHECK THE ACTUAL PRICE AND PREDICTED PRICE

graph.plot(kind='bar')

plt.title('BarGraph')

plt.xlabel('Actual&prediceted price')

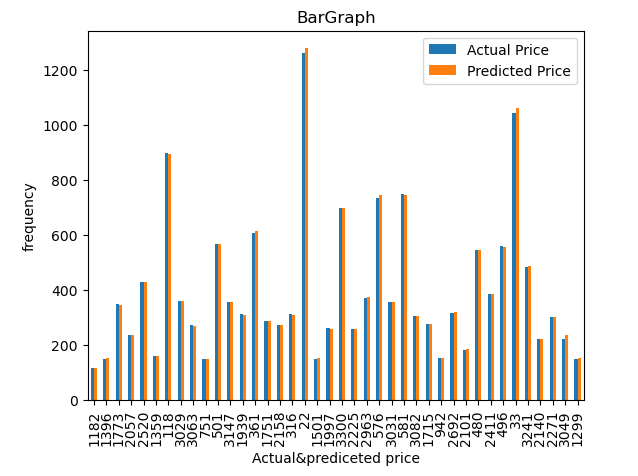
plt.ylabel('frequency')

plt.show()

# output:

# Result:

**In the above graph x-axis represents the actual and** prediceted price and y-axis represent the frequency. It comparision between**the actual and** predicetedprice,in this graph prediction price is almost same as actual price .



# # TRAINING THE MODEL FOR RandomForestRegressor

# #MODEL TRAINING

**from**sklearn.ensemble**import**RandomForestRegressor

regressorRandomForestRegressor(n\_estimators**=**100,random\_state**=**0)

regressor**.**fit(x\_train,y\_train)

# output:



# Prediction

predicted=regressor.predict(x\_test)

predicted

# evaluating the model

train\_accuracy=regressor.score(x\_train,y\_train)

print('train\_accuracy(R\_Squered):',train\_accuracy)

R\_test\_accuracy=regressor.score(x\_test,y\_test)

print('test\_accuracy(R\_Squered):',test\_accuracy)

# output:

train\_accuracy: 0.999155403718174

test\_accuracy: 0.9992925939370841

# Preprocessing the x\_train and x\_test

scaler=StandardScaler()

x\_train=scaler.fit\_transform(x\_train)

x\_test=scaler.transform(x\_test)

# ­­­ANN Model

# training the model using ANN model

ann\_model=MLPRegressor(hidden\_layer\_sizes=(128,64,32),activation='relu',solver='lbfgs')

ann\_model.fit(x\_train,y\_train)

# output:



# Prediction

predicted=ann\_model.predict(x\_test)

predicted

# output:

array([ 561.0733676 , 301.93313859, 183.73176285, ..., 1053.90075514,

398.29812686, 115.6456688 ])

# evaluating the model

train\_accuracy=ann\_model.score(x\_train,y\_train)

print('train\_accuracy(R\_Squered):',train\_accuracy)

R\_test\_accuracy=ann\_model.score(x\_test,y\_test)

print('test\_accuracy(R\_Squered):',test\_accuracy)

print('Mean Absolute Error:',metrics.mean\_absolute\_error(y\_test,y\_pred))

print('Mean Squared Error:',metrics.mean\_squared\_error(y\_test,y\_pred))

print('Root Mean Squared Error:',math.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

# output:

­­­