Restaurant food cost

'PG PROGRAM IN DATA SCIENCE, MACHINE LEARNING AND NEURAL NETWORKS'

**PREPARED BY:**

**SHEFALI MEVADA**

**BATCH NO: 1826**

# ****Predicting the Food Cost of Restaurant****

##### In this Blog, I will go through the whole process of creating a machine learning model on the Restaurant food cost dataset, which is used by many people all over the world. In which It provides Cuisines, Locality , City, Ratings of the food and Restaurant kind of information about the Restaurants. And some factors that really affect the cost, and we can gain some very interesting insights that might help us to choose what to eat and from where.

This dataset contains information about various aspects of Restaurant. All of us must have craving for at least a few favourite food items, we may also have a few places where we like to get them, a restaurant which serves our favourite food the way we want it to be. But there is one factor that will make us reconsider having our favourite food from our favourite restaurant, the **cost**. Here in this hackathon, we will be predicting the cost of the food served by the restaurants across different cities in India. We Use Data Science skills to investigate the factors that really affect the cost, and gain some very interesting insights that might help us choose what to eat and from where.

**Importing the Libraries**

# linear algebra  
**import** **numpy** **as** **np**   
  
# data processing  
**import** **pandas** **as** **pd**

# data visualization

**import seaborn as sns  
from matplotlib.pyplot import pyplot as plt**

# Algorithms

##### from scipy.stats import zscore

##### from sklearn.preprocessing import LabelEncoder

##### from sklearn.preprocessing import StandardScaler

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

##### from sklearn.linear\_model import LinearRegression, Lasso,Ridge,ElasticNet

##### from sklearn.svm import SVR

##### from sklearn.tree import DecisionTreeRegressor

##### from sklearn.neighbors import KNeighborsRegressor

##### from sklearn.ensemble import RandomForestRegressor

##### from sklearn.ensemble import AdaBoostRegressor

##### from sklearn.ensemble import GradientBoostingRegressor

##### from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error,r2\_score

##### from sklearn.model\_selection import GridSearchCV,cross\_val\_score

##### from sklearn.model\_selection import GridSearchCV

##### import warnings

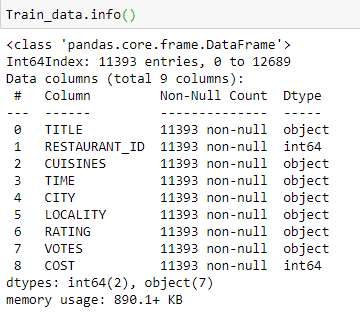
##### warnings.filterwarnings('ignore')

# Getting the Data

Train\_data=pd.read\_csv("food\_train.csv")

Test\_data=pd.read\_csv("food\_test.csv")

# Data Exploration/Analysis



**The training-set has 11393 examples and 8 features + the target variable (COST)**. 2 of the features are integers and 7 are objects.. Below I have listed the features with a short description:

**TITLE:**The feature of the restaurant which can help identify what and for whom it is suitable for.

**RESTAURANT\_ID:**A unique ID for each restaurant.

**CUISINES:**The variety of cuisines that the restaurant offers.

**TIME:**The open hours of the restaurant.

**CITY:**The city in which the restaurant is located.

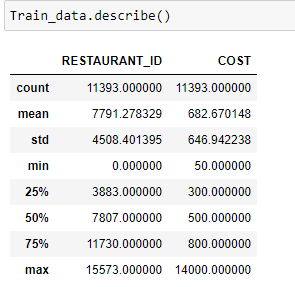
**LOCALITY:**The locality of the restaurant.

**RATING:** The average rating of the restaurant by customers.

**VOTES:**The overall votes received by the restaurant.

**COST:** The average cost of a two-person meal.

**Summary statistics**

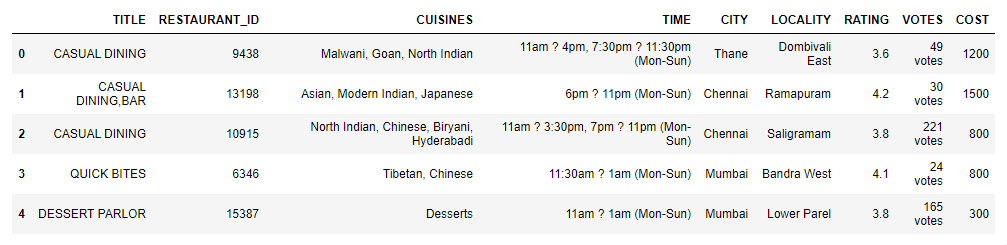
****

Here, In above Image We Describing the Datasets. we are determining mean, standard deviation, minimum and maximum value of each column, It helps Us further in Data Cleaning.

From Above observation we can see that The minimum Cost of Restaurant is 50 and maximum is 14000.

Mean is greater than median that’s why target variable is right skewed



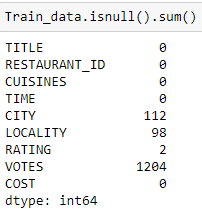


From the above table, We can see the first five row of data.

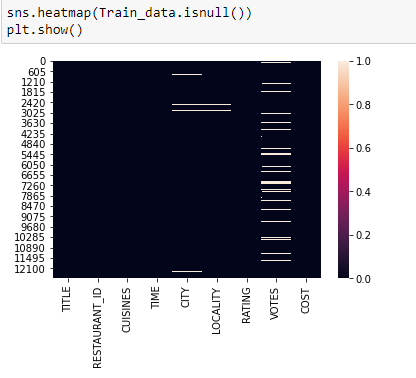
Now let's check the null values for the dataset



there is null values in this data, so take the sum of null values present in the data.



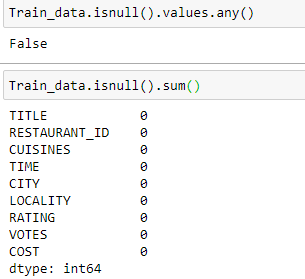
Using heatmap to showing null values in dataset



now dropping the null values from the dataset.

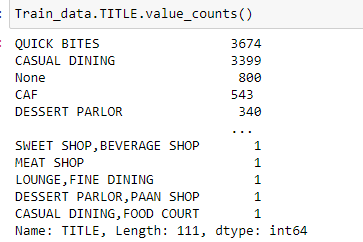
Train\_data.dropna(inplace=True)

After Dropping,now checking again if there is any null values

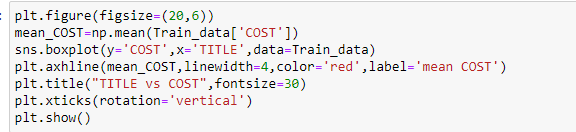


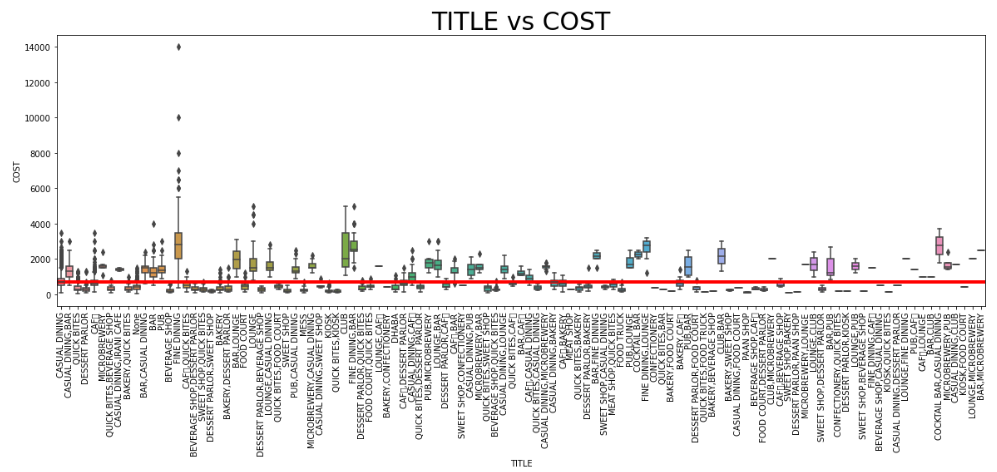
Now as per showing in the above image, there is no null values in dataset.

How the TITLE of the Restaurant effects the COST of Restaurant Food Cost. we will check the title and cost varient of the dataset using boxplot method. First we find the total Value counts of the Title.



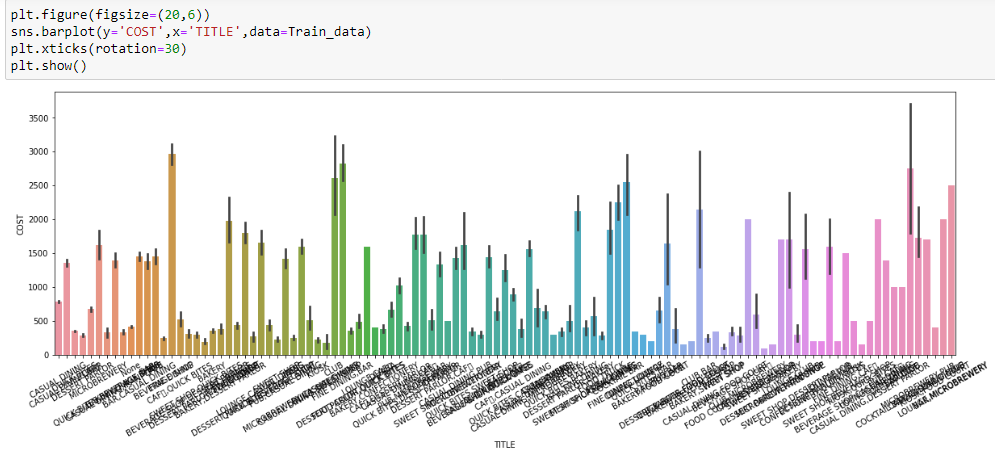
the length of the value counts of the TITLE columns is 111, so we can make it possible to using boxplot method.



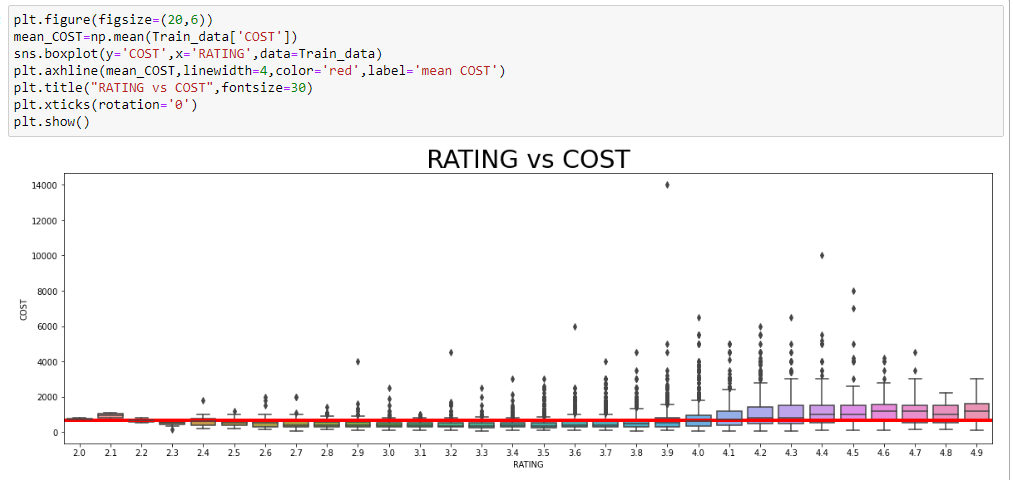


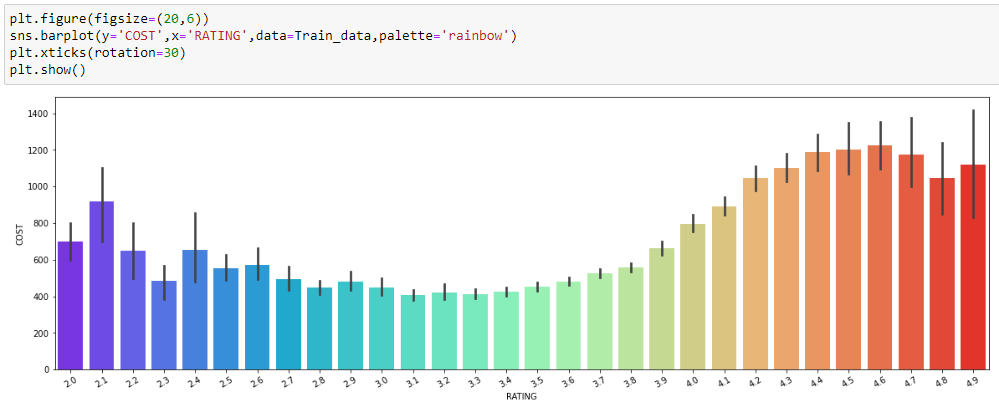
As per shown in the image we take here figure size of 20 and 6.in boxplot here we take the x is TITLE as independent variable and y is COST the target variable as Dependent Variable. and the data is from train dataset.

Now we are checking it with barplot for better visualization of Outliers.



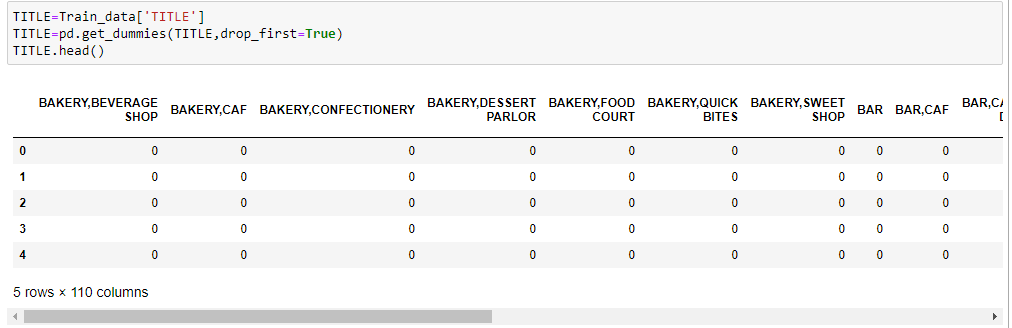
'RATING' column can affect the 'COST' of Restaurant Food. So we are checking for its visualization of Boxplot and Barplot method.



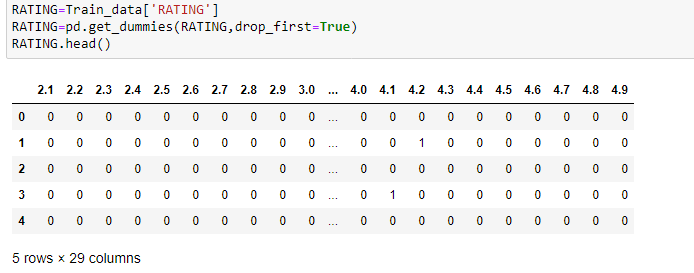


The other features of train data of Restaurant food Cost dataset, like 'RESTAURANT\_ID', 'CUISINES', 'TIME', 'CITY', 'LOCALITY','VOTES' are having huge number of value counts, so we can't make in a boxplot or barplot method visualization.

Now getting the dummies of dataset for TITLE and RATING columns.



For 'TITLE' column, as we can see in the above image now there are 5 rows and 110 columns for dummies.



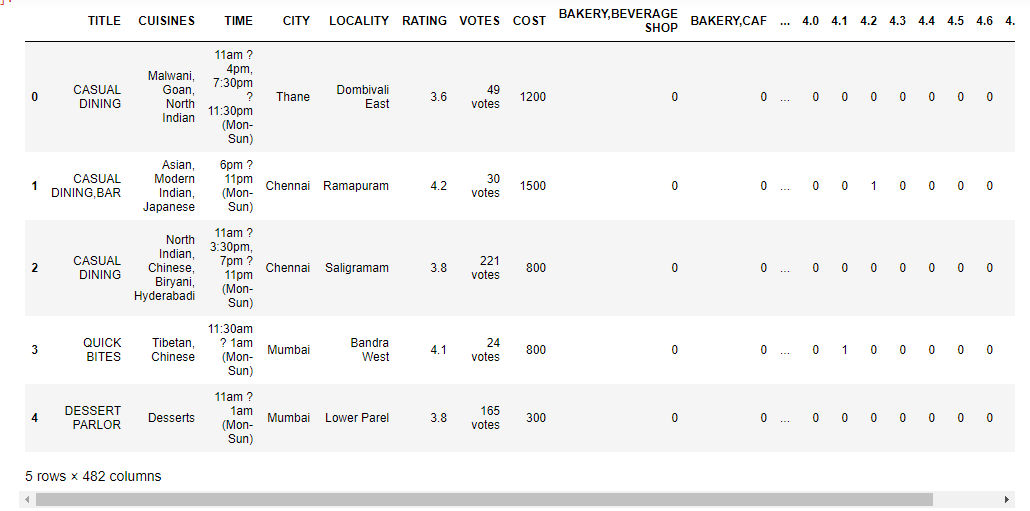
for 'RATING' column as seen in this image for dummies of 'RATING' column there are 5 rows and 29 columns.

**Data Preprocessing**

**Data Cleaning**

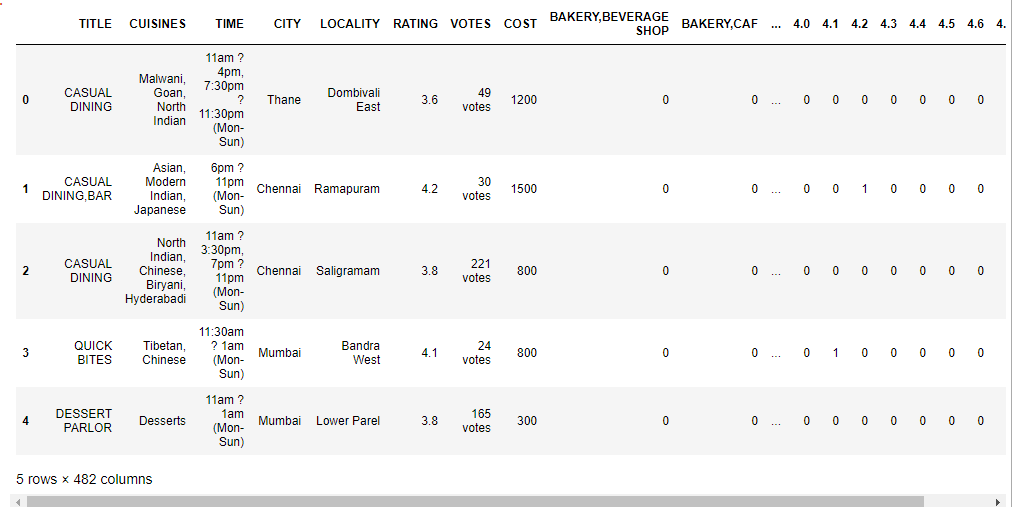
I will drop 'RESTAURANT\_ID' from the train set, because it does not contribute to a Food Cost of Restaurant. **Train\_data.drop(['RESTAURANT\_ID'],axis=1,inplace=True)**

Now concating the columns of Train data **train=pd.concat([Train\_data,TITLE,CITY,RATING],axis=1) train.head()**

****

Here we can see our concat data of train dataset.reduced at 5 rows and 482 coulmns. After this chart we are dropping the 'TITLE','CITY','RATING'

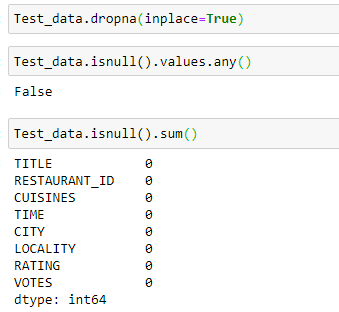
**train.drop(['TITLE','CITY','RATING'],axis=1,inplace=True) train.head()**

****

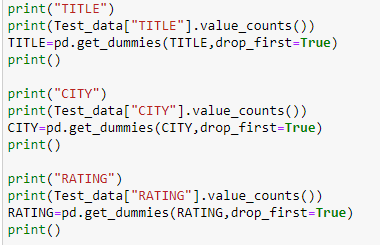
Now we are checking the shape of dataset after this preocess.



for TEST DATA we are doing the same process as Train data,



Now getting the dummies for Test dataset columns.



Output for the above image's input is as following:

TITLE

QUICK BITES 1225

CASUAL DINING 1157

None 240

CAF 205

DESSERT PARLOR 109

...

BAR,LOUNGE 1

PUB,LOUNGE 1

CLUB,BAR 1

CASUAL DINING,FOOD COURT 1

CASUAL DINING,IRANI CAFE 1

Name: TITLE, Length: 83, dtype: int64

CITY

Chennai 671

Hyderabad 637

Bangalore 633

Mumbai 547

New Delhi 399

...

opposite to western side of ITPL Service gate 1

Balavinayagar Nagar Chennai 1

Thane West Thane West 1

Palarivattom 1

Chennai Injambakkam 1

Name: CITY, Length: 142, dtype: int64

RATING

3.8 411

3.9 403

4.0 397

3.7 348

3.6 309

4.1 280

4.2 244

3.5 209

3.4 201

4.3 183

4.4 122

3.3 119

3.2 83

3.1 77

4.5 72

3.0 59

2.9 56

2.8 44

4.6 41

2.7 40

2.6 24

4.7 21

2.5 16

4.8 15

2.4 9

4.9 6

2.3 5

2.2 2

2.1 1

Name: RATING, dtype: int64

Now concating the Test data and dropping some columns that doesn’t affect the Cost of Restaurant Food.

**test=pd.concat([Test\_data,TITLE,CITY,RATING],axis=1) test.head()**

**test.drop(['TITLE','CITY','RATING'],axis=1,inplace=True) test.head()**

Checking the Shape of test data after dropping some columns from the dataset.

**test.shape**

**(11781, 479)**

Now the reduced Shape of test dataset is 11781 and 479.

Using fit\_transform method form labelEncoder to covert the datatype into object.

**for col in train.columns:**

**if train[col].dtypes=='object':**

**le=LabelEncoder()**

**train[col]=le.fit\_transform(train[col])**

Removing Outliers With Using z-score method,

**from scipy.stats import zscore z=np.abs(zscore(train)) z**

**threshold=3 print(np.where(z>3))**

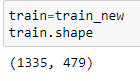
****

Checking the shape of Original train data and New Train data

**train\_new=train[(z<3).all(axis=1)]**

**print(train.shape,'\t\t', train\_new.shape)**

(11393, 479) (1335, 479)



**x=train.drop(columns=['COST'],axis=1)**

**x.head()**

**y=train['COST']**

**y.head()**

checking the shape of input and target variables

**print(x.shape,'\t\t',y.shape)**

# Building Machine Learning Models

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

scaling the input variables

**from sklearn.preprocessing import StandardScaler**

**sc=StandardScaler()**

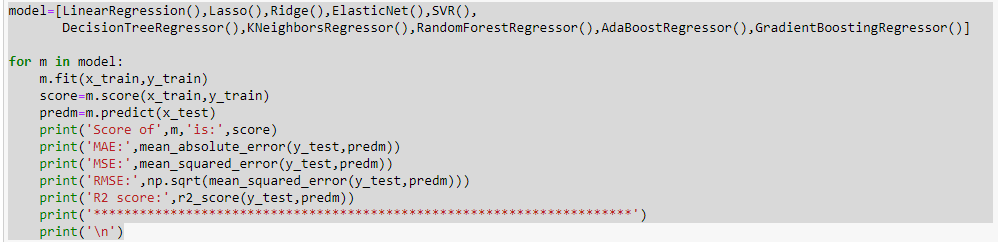
**x=sc.fit\_transform(x)**

splitting the data into training and testing data

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

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=20,random\_state=42)**

**Applying Linearmodel method from sklearn library**

****

Showing Output of the above image code

Score of LinearRegression() is: 0.41716771762081617

MAE: 184.07646855691542

MSE: 55678.4323526168

RMSE: 235.96277747267004

R2 score: 0.44529581715948396

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of Lasso() is: 0.419148905644145

MAE: 184.7034959505986

MSE: 55550.74446297132

RMSE: 235.69205430597637

R2 score: 0.44656792564910264

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of Ridge() is: 0.41922979200770316

MAE: 184.99113579648932

MSE: 55688.5083307959

RMSE: 235.98412728570517

R2 score: 0.44519543381523385

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of ElasticNet() is: 0.3921343503261431

MAE: 191.5586386724612

MSE: 57168.55355709545

RMSE: 239.09946373234604

R2 score: 0.4304502758944413

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of SVR() is: 0.10136529168440322

MAE: 218.4327704948505

MSE: 84232.85299196746

RMSE: 290.2289664936418

R2 score: 0.1608184010762893

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of DecisionTreeRegressor() is: 1.0

MAE: 190.0

MSE: 58250.0

RMSE: 241.3503677229434

R2 score: 0.41967621419676215

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of KNeighborsRegressor() is: 0.5255405095936139

MAE: 202.0

MSE: 58620.0

RMSE: 242.11567483333252

R2 score: 0.41599003735990037

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of RandomForestRegressor() is: 0.9189848914401401

MAE: 210.51999999999998

MSE: 62047.458999999995

RMSE: 249.09327369481497

R2 score: 0.38184349688667496

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of AdaBoostRegressor() is: 0.3893868933395578

MAE: 213.98398449119583

MSE: 67809.00123171357

RMSE: 260.4016152632575

R2 score: 0.3244433252133144

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Score of GradientBoostingRegressor() is: 0.6691899764603606

MAE: 206.356652467638

MSE: 60892.23815247137

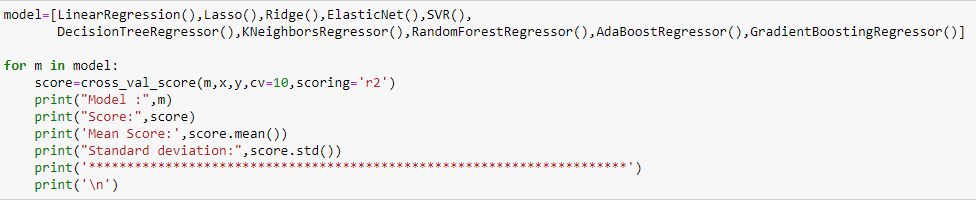
RMSE: 246.76352678722878

R2 score: 0.3933525464261881

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Which is the best Model ?

# As we can see, the Random Forest classifier goes on the first place. But first, let us check, how random-forest performs, when we use cross validation.



Model : LinearRegression()

Score: [0.40637808 0.39190597 0.43753376 0.43353477 0.34010717 0.34681648

0.28174836 0.46937656 0.43895629 0.5387606 ]

Mean Score: 0.4085118050090582

Standard deviation: 0.06920935128906724

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : Lasso()

Score: [0.40666285 0.3918883 0.45264073 0.42642033 0.34791865 0.34360403

0.28495392 0.46885988 0.45751264 0.55039463]

Mean Score: 0.413085596718816

Standard deviation: 0.07187729159730782

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : Ridge()

Score: [0.40578144 0.39098811 0.45251963 0.42395012 0.34816639 0.34401484

0.28276558 0.47018503 0.45734357 0.54876393]

Mean Score: 0.41244786352381346

Standard deviation: 0.07197705547860904

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : ElasticNet()

Score: [0.39052561 0.37431001 0.42506405 0.46054314 0.30580617 0.29693853

0.28949645 0.43438356 0.44885856 0.4921606 ]

Mean Score: 0.39180866747149523

Standard deviation: 0.06938341465382941

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : SVR()

Score: [ 0.08474061 0.05865019 0.10832788 0.19117553 -0.01841624 -0.01777854

0.05722658 0.08157637 0.1522158 0.16083272]

Mean Score: 0.08585508980484544

Standard deviation: 0.06694363206551045

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : DecisionTreeRegressor()

Score: [-0.09051467 -0.19507091 -0.39511488 -0.26568155 0.0835831 0.07959498

-0.00720669 -0.3125707 -0.14198583 0.03013274]

Mean Score: -0.12148343963290882

Standard deviation: 0.16047043090748817

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : KNeighborsRegressor()

Score: [0.29121944 0.25748013 0.37408942 0.1303465 0.27796185 0.29674873

0.22077089 0.22263379 0.52861351 0.42205059]

Mean Score: 0.3021914859015479

Standard deviation: 0.10771632936324624

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : RandomForestRegressor()

Score: [0.57487903 0.35116619 0.28531478 0.29909441 0.49712106 0.37964861

0.3407247 0.37111611 0.46896257 0.60269019]

Mean Score: 0.41707176459988216

Standard deviation: 0.10649942377487703

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : AdaBoostRegressor()

Score: [-0.00572614 0.21146626 0.26364032 0.15781906 0.21866034 0.30054083

0.27763954 0.31881624 0.26708291 0.23866848]

Mean Score: 0.22486078440497853

Standard deviation: 0.08871340717219967

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model : GradientBoostingRegressor()

Score: [0.59375596 0.44428891 0.40358708 0.38949691 0.46656131 0.418013

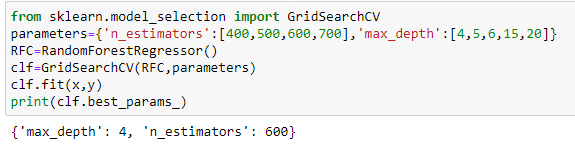
0.35749633 0.48649641 0.52160361 0.57904914]

Mean Score: 0.4660348663608704

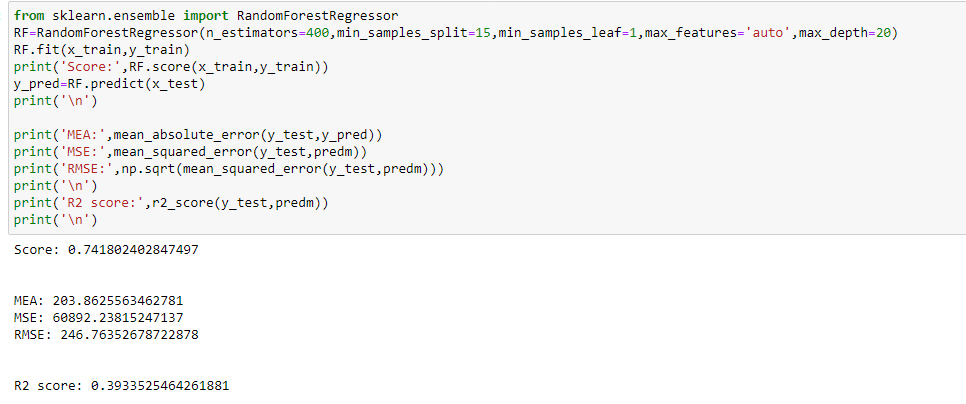
Standard deviation: 0.07543813606513125

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

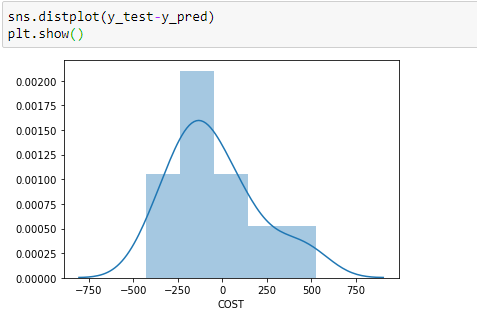
Now applying GridsearchCV algorithm from model selection library



Now from sklearn.ensemble technique importing RandomForestRegressor



showing output using distplot for test data and predicted data



Using Scatterplot for test data and predicted data

