Zomato Data Analysis with Machine Learning

Overview:

In this article, we will be predicting average cost and price range for the restaurant data available in Zomato. We will be using data provided, contains two datasets - Zomato.csv and country_code.csv. “Average cost for two” prediction is through regression approach and “Price range” prediction is through classification approach.

Problem statement:

Zomato Data Analysis is one of the most useful analysis for foodies who want to taste the best cuisines of every part of the world which lies in their budget. This analysis is also for those who want to find the value for money restaurants in various parts of the country for the cuisines. Additionally, this analysis caters the needs of people who are striving to get the best cuisine of the country and which locality of that country serves that cuisines with maximum number of restaurants.

Following are the major steps in data science project life cycle:

1) Data analysis: Here we will get to know about how the present in provided data set
2) Exploratory data analysis: EDA is one of the most important steps in the data science project life cycle and here we make inferences from the visualizations and data analysis
3) Feature selection: Based on statistical support important features need to be selected
4) Model building: Here we will be using 5 ML models and then we will choose the best
5) Saving model: Saving the best model using pickle to make the prediction from real data.

There are many intermediate steps along with these above mentioned 5 steps, let us look at one-by-one along with the project flow.

1) Importing libraries to start with:

```
#importing required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

2) Loading the dataset:

```
#importing or loading the dataset
df1 = pd.read_excel('C:/Users/Shashanka S/Desktop/evaluation projects/Country-Code.xlsx')
df2 = pd.read_csv('zomato2.csv')
```

df1 – is of no use in prediction as it has only country code mapped to country name

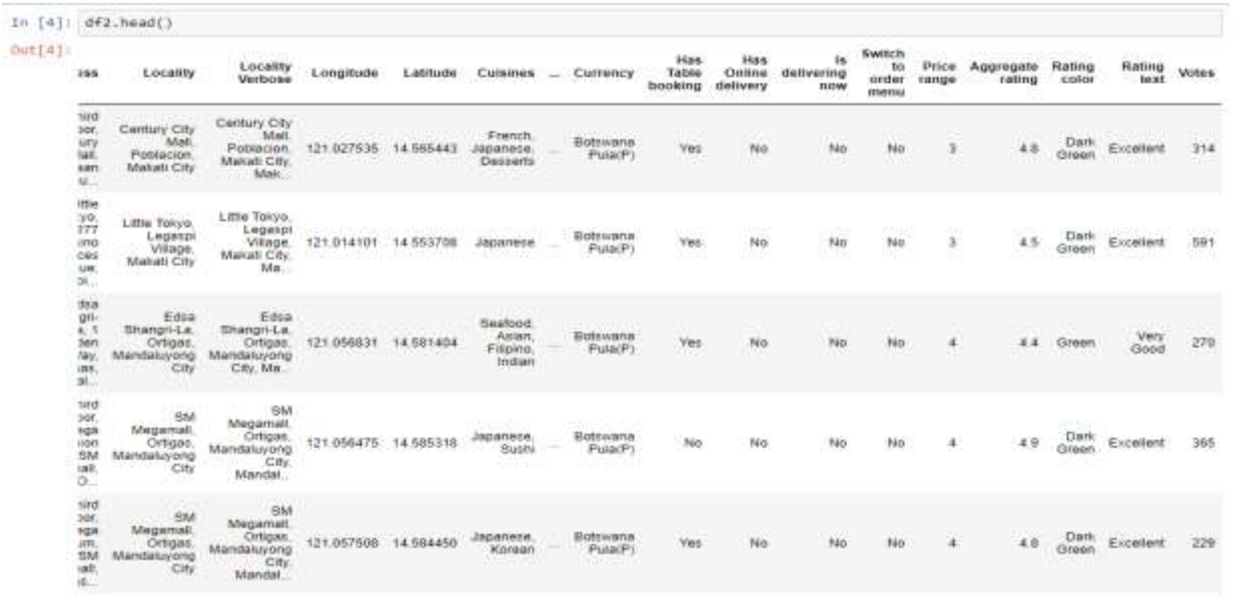
df2 – is the main data set which we are going to work on

df = df2

```
In [3]: df1.head()
```

Out[3]:

	Country Code	Country
0	1	India
1	14	Australia
2	30	Brazil
3	37	Canada
4	94	Indonesia



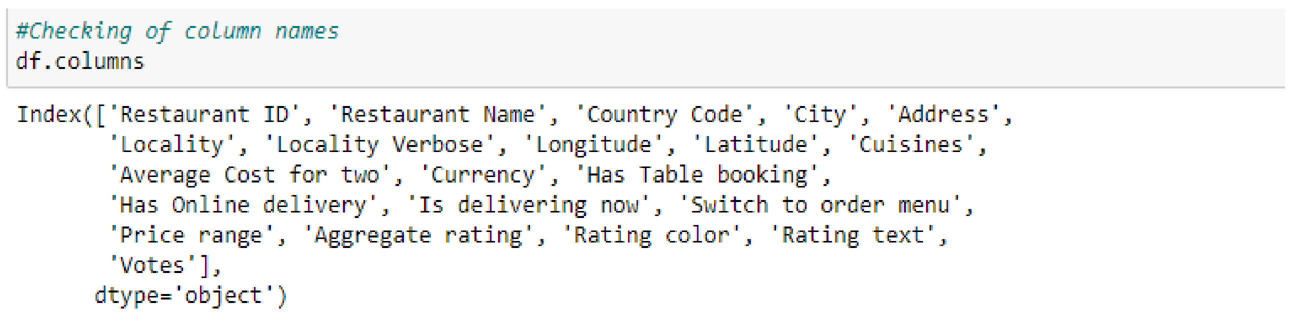
```
In [4]: df2.head()
```

Out[4]:

ss	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Price range	Aggregate rating	Rating color	Rating text	Votes
ird oor, ury lall, san u...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535	14.565443	French, Japanese, Desserts	...	Botswana Pula(P)	Yes	No	No	No	3	4.8	Dark Green	Excellent	314
ittle yo, 277 ino ces ue, oi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.014101	14.553708	Japanese	...	Botswana Pula(P)	Yes	No	No	No	3	4.5	Dark Green	Excellent	591
dsa gri- a, 1 3en Vay, ias, al...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	...	Botswana Pula(P)	Yes	No	No	No	4	4.4	Green	Very Good	270
iird oor, ega ion SM all, O...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475	14.585318	Japanese, Sushi	...	Botswana Pula(P)	No	No	No	No	4	4.9	Dark Green	Excellent	365
iird oor, ega im, SM all, d...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	...	Botswana Pula(P)	Yes	No	No	No	4	4.8	Dark Green	Excellent	229

3) **Exploratory data analysis**: Every Restaurant contains the following variables

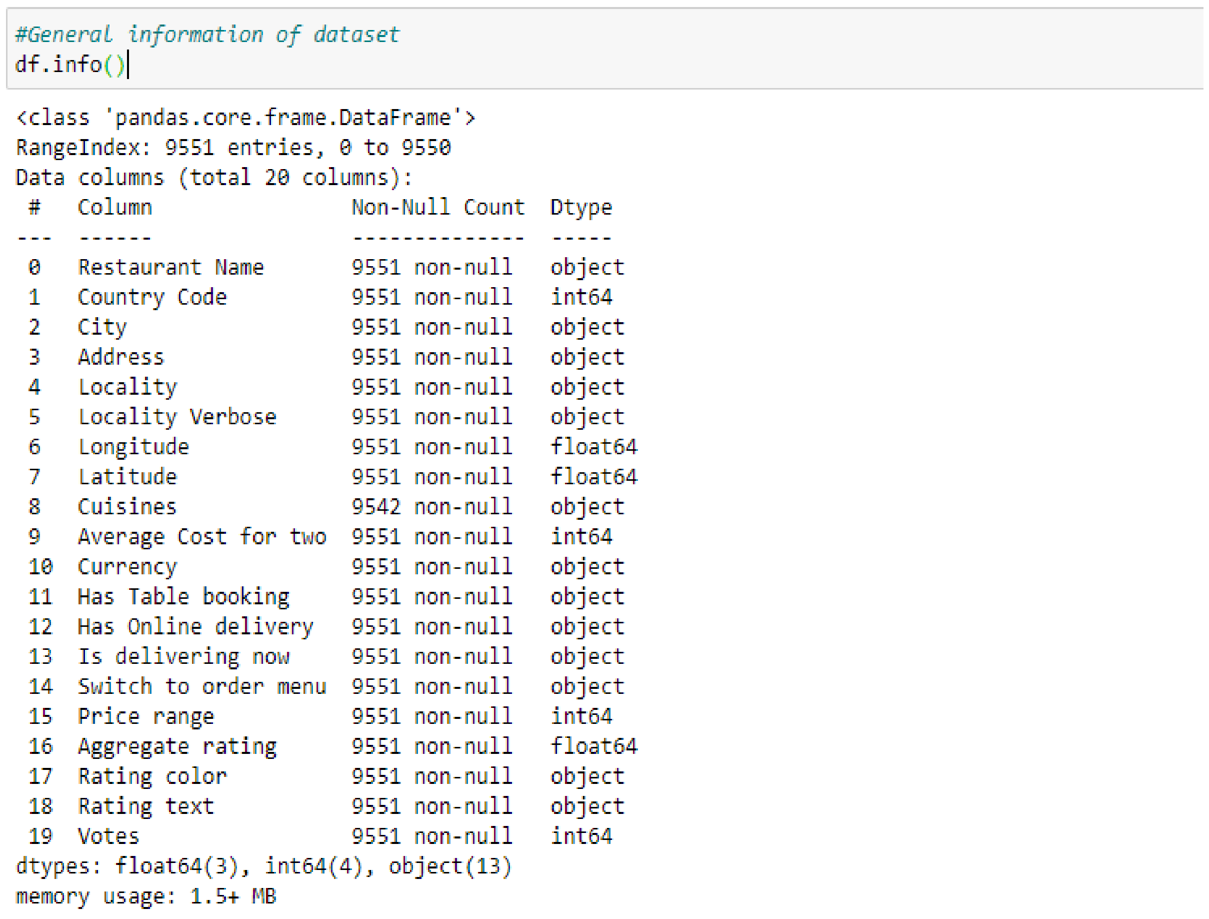
- Restaurant Id: Unique id of every restaurant across various cities of the world
- Restaurant Name: Name of the restaurant
- Country Code: Country in which restaurant is located
- City: City in which restaurant is located
- Address: Address of the restaurant
- Locality: Location in the city
- Locality Verbose: Detailed description of the locality
- Longitude: Longitude coordinate of the restaurant's location
- Latitude: Latitude coordinate of the restaurant's location
- Cuisines: Cuisines offered by the restaurant
- Average Cost for two: Cost for two people in different currencies
- Currency: Currency of the country
- Has Table booking: yes/no
- Has Online delivery: yes/ no
- Is delivering: yes/ no
- Switch to order menu: yes/no
- Price range: range of price of food
- Aggregate Rating: Average rating out of 5
- Rating color: depending upon the average rating color
- Rating text: text on the basis of rating of rating
- Votes: Number of ratings casted by people



```
#Checking of column names
df.columns
```

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes'],
      dtype='object')
```

4) General information of dataset:

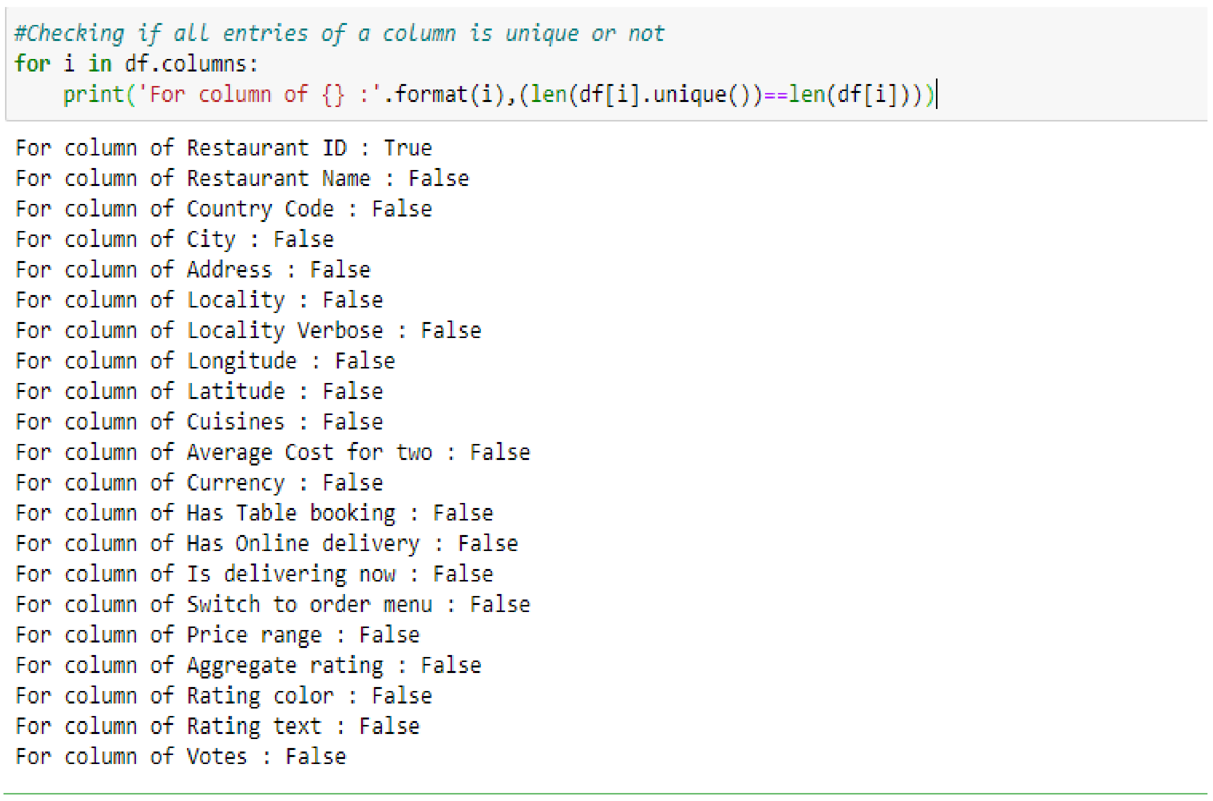


```
#General information of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9551 entries, 0 to 9550
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Restaurant Name       9551 non-null   object 
 1   Country Code          9551 non-null   int64  
 2   City                  9551 non-null   object 
 3   Address               9551 non-null   object 
 4   Locality              9551 non-null   object 
 5   Locality Verbose      9551 non-null   object 
 6   Longitude             9551 non-null   float64
 7   Latitude              9551 non-null   float64
 8   Cuisines              9542 non-null   object 
 9   Average Cost for two  9551 non-null   int64  
 10  Currency              9551 non-null   object 
 11  Has Table booking     9551 non-null   object 
 12  Has Online delivery   9551 non-null   object 
 13  Is delivering now     9551 non-null   object 
 14  Switch to order menu  9551 non-null   object 
 15  Price range           9551 non-null   int64  
 16  Aggregate rating      9551 non-null   float64
 17  Rating color          9551 non-null   object 
 18  Rating text           9551 non-null   object 
 19  Votes                 9551 non-null   int64  
dtypes: float64(3), int64(4), object(13)
memory usage: 1.5+ MB
```

We can observe that cuisines has 9 null (NaN) values and is of obect datatype. We can handle this missing value by mode imputation.

5) Checking if all entries of a column is unique or not:

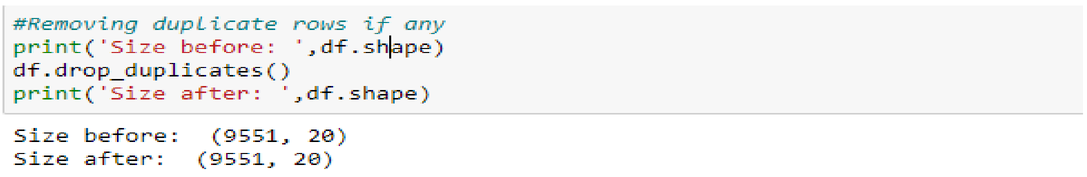


```
#Checking if all entries of a column is unique or not
for i in df.columns:
    print('For column of {} :'.format(i),(len(df[i].unique())==len(df[i])))
```

```
For column of Restaurant ID : True
For column of Restaurant Name : False
For column of Country Code : False
For column of City : False
For column of Address : False
For column of Locality : False
For column of Locality Verbose : False
For column of Longitude : False
For column of Latitude : False
For column of Cuisines : False
For column of Average Cost for two : False
For column of Currency : False
For column of Has Table booking : False
For column of Has Online delivery : False
For column of Is delivering now : False
For column of Switch to order menu : False
For column of Price range : False
For column of Aggregate rating : False
For column of Rating color : False
For column of Rating text : False
For column of Votes : False
```

We can observe that Restaurant ID is having all unique entries and act just like indexing column. So let drop this column.

6) Removing duplicate rows if any:

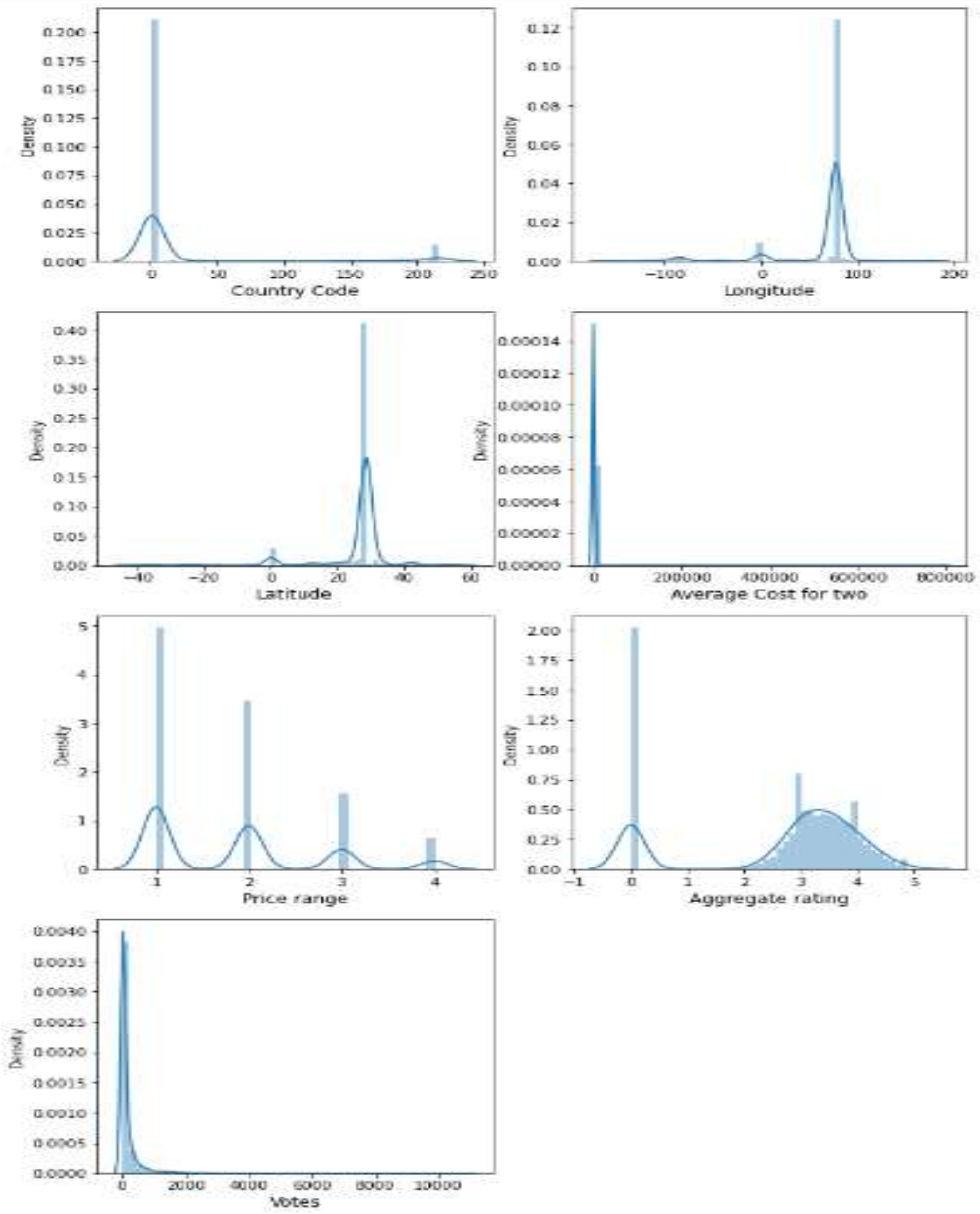


```
#Removing duplicate rows if any
print('Size before: ',df.shape)
df.drop_duplicates()
print('Size after: ',df.shape)
```

```
Size before:  (9551, 20)
Size after:  (9551, 20)
```

Therefore no any duplicate rows present in the dataset.

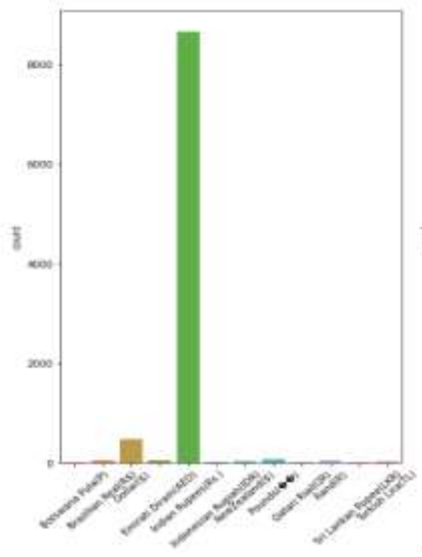
7) Checking of distribution plot of each columns having numerical data:

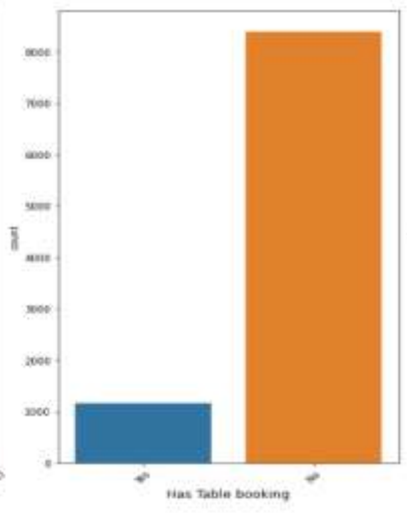


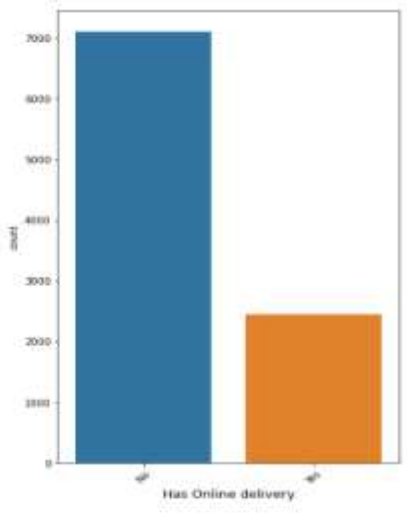
We can observe most of the country code is 1, which is India. Correspondingly Latitude and longitude locations points out Indian restaurants co-ordinates

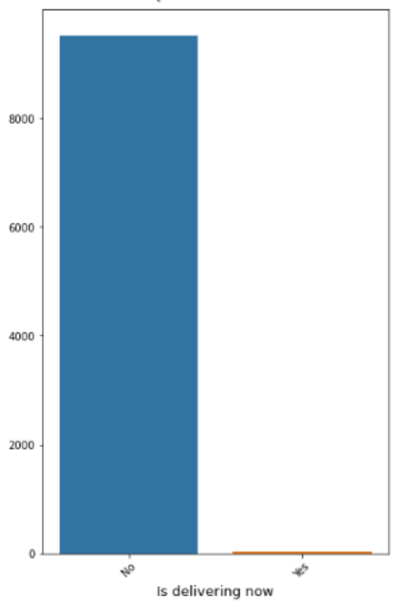
Majorly average cost for 2 lies within range of 1500 INR correspondingly within price range of 1 & 2 most of restaurants are there. The data are highly skewed, needs outliers handling and proper transformation before training the model

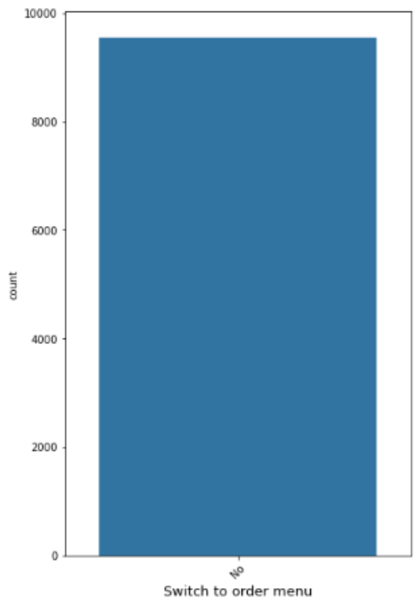
8) Checking of value counts in each columns having categorical data:

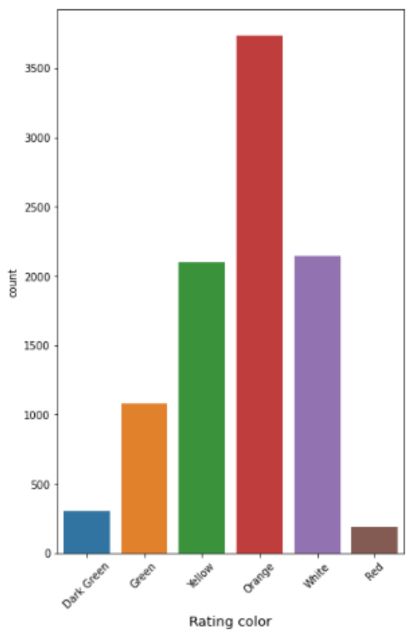


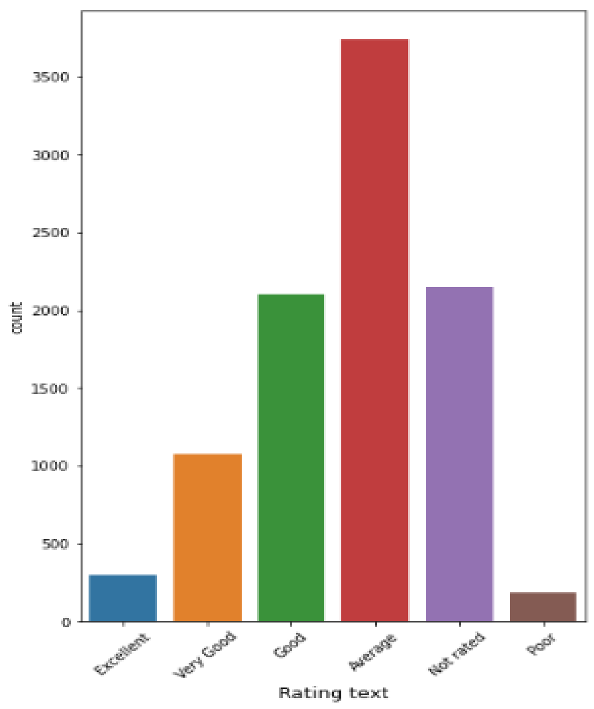












We can observe that majority of currency is INR.

Majority of restuarants has no table booking and has online delivery

'Switch to order menu' is having all single entries as 'No', this wonts help in predictions

'Is delivering now' is having highly imbalanced class having very less proportion of 'Yes'

So let us drop these columns

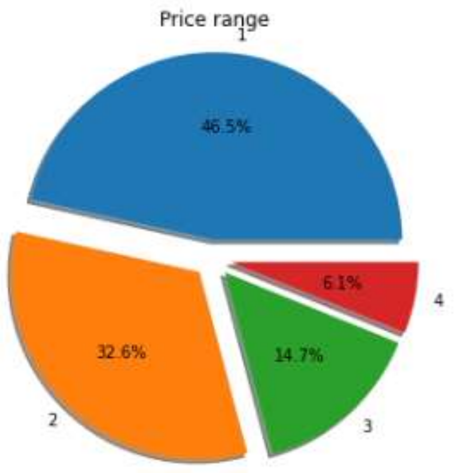
'Address' can be dropped as this carry almost all unique entries and location information can be obtained by co-ordinates also

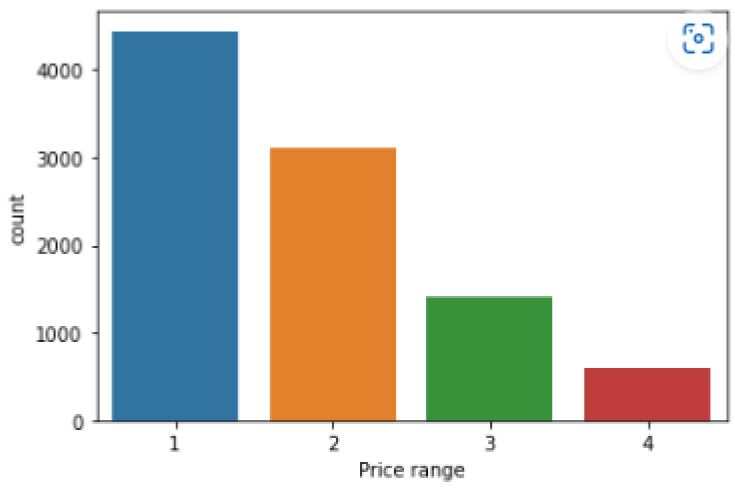
'Locality Verbose' convey almost similar information as that of 'Locality'

So let us drop these 2 columns

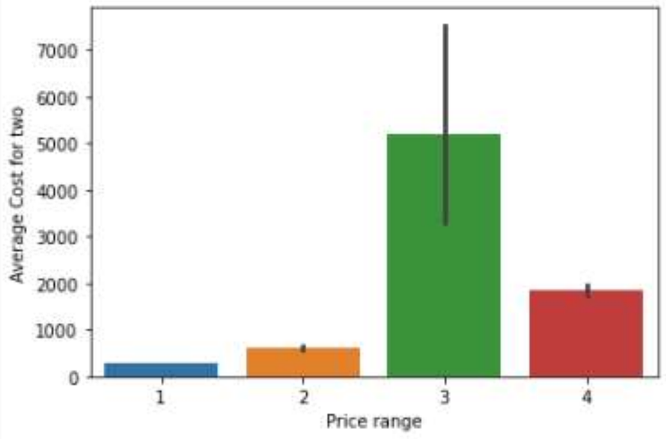
Rating color and text conveying same information, let us drop tone among the column.

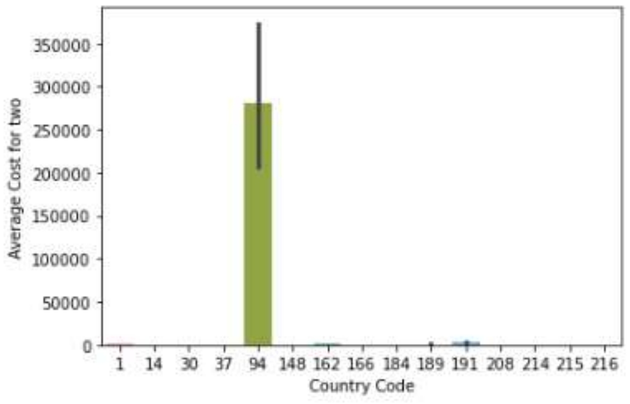
9) Checking of value counts in target variable:

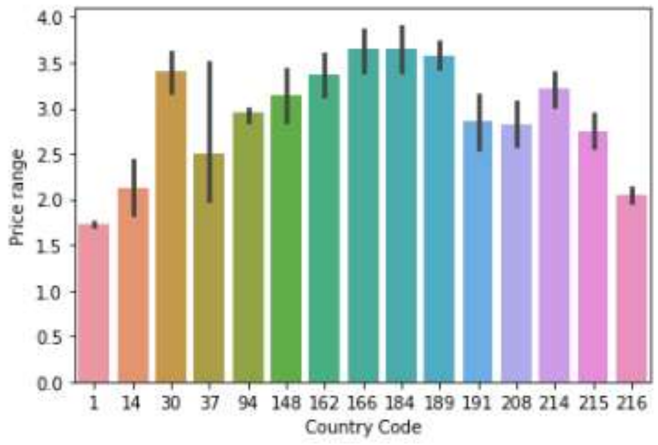




Let us plot target variables Price range vs Average cost and with different country code and check







We can observe that average cost is more in Price range 3 restuarants

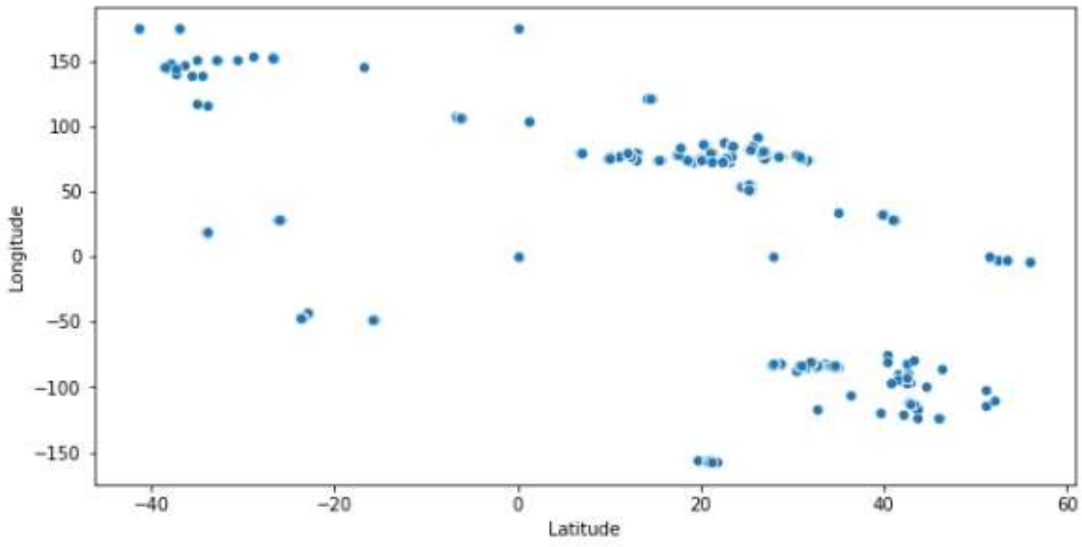
We can observe that average cost is more in country code 94 - Indonesia

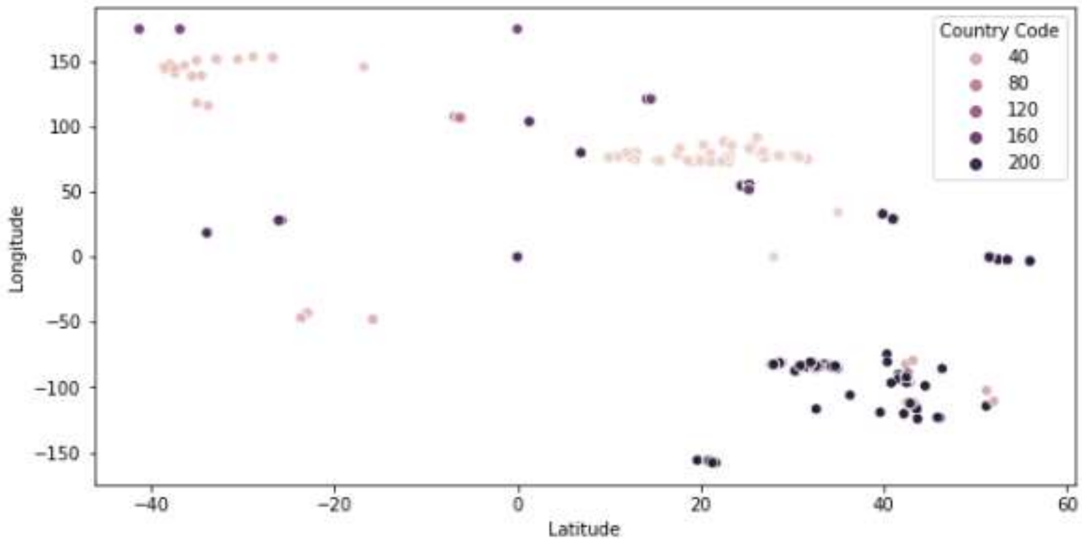
From garph 3 we can observe that Indonesia falls under price range 3

Least price range near 1.8 for 1 - India

Highest price range of 3.6 for 166 - Qatar and 184 – Singapore.

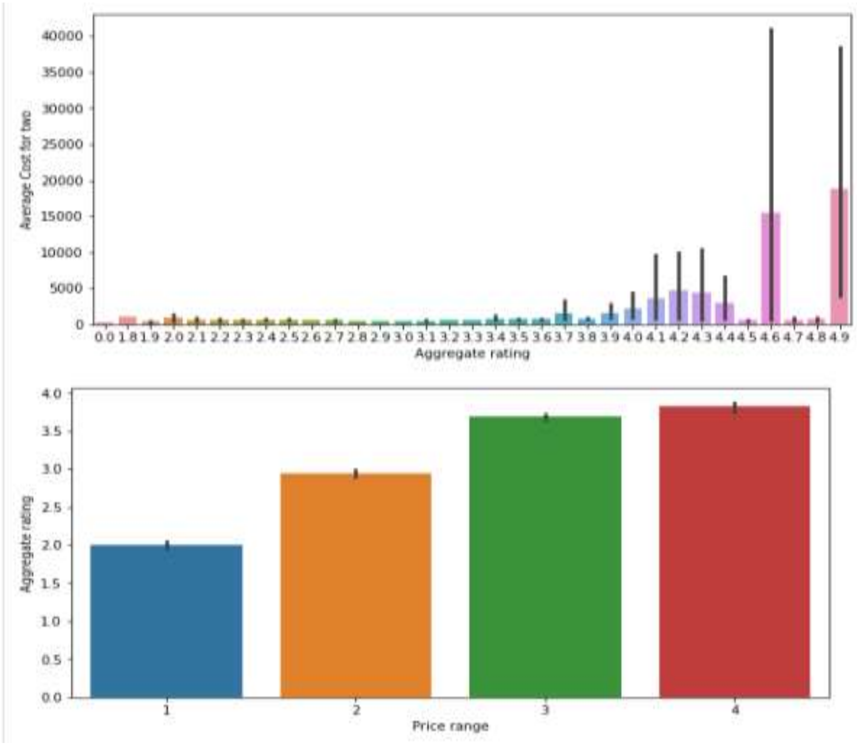
10) Latitude vs Longitude for getting co-ordinates:



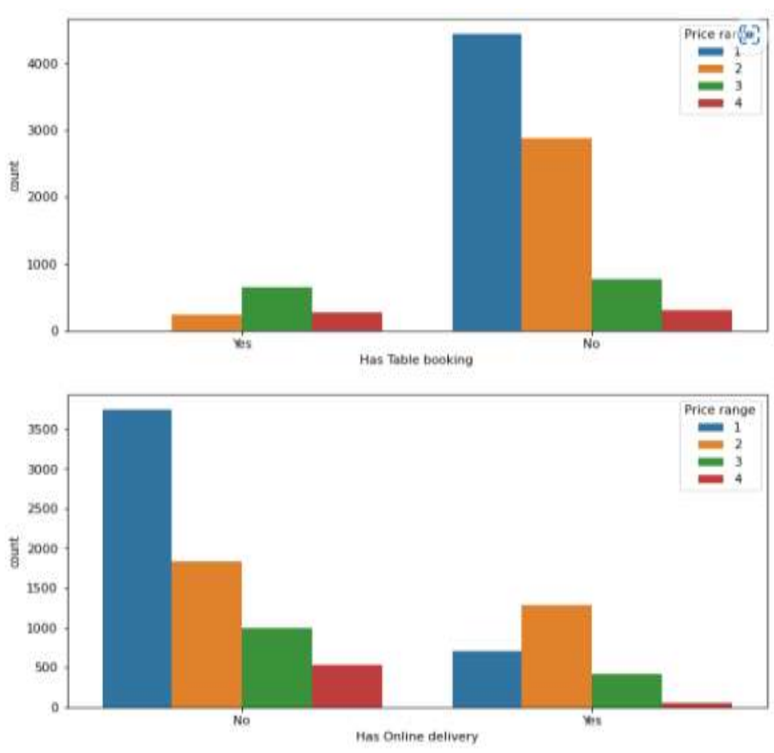


The above points shows geo co-ordinates of restaurants present around the world

11) Some more plots to visualize:



We can observe that as price range increases average rating increases and correspondingly there is more Average cost for 2



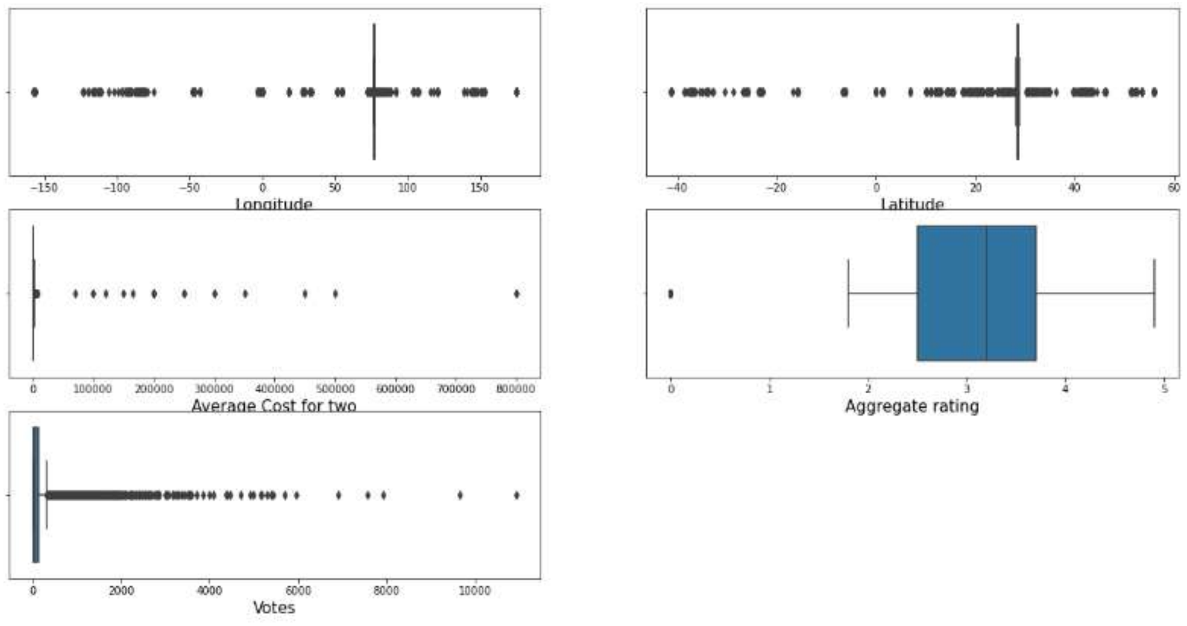
Among the resuarants having no table booking: Price range 1 hotels are most common

Among the resuarants having table booking: Price range 3 hotels are most common

Among the resuarants having no online delivery: Price range 1 hotels are most common

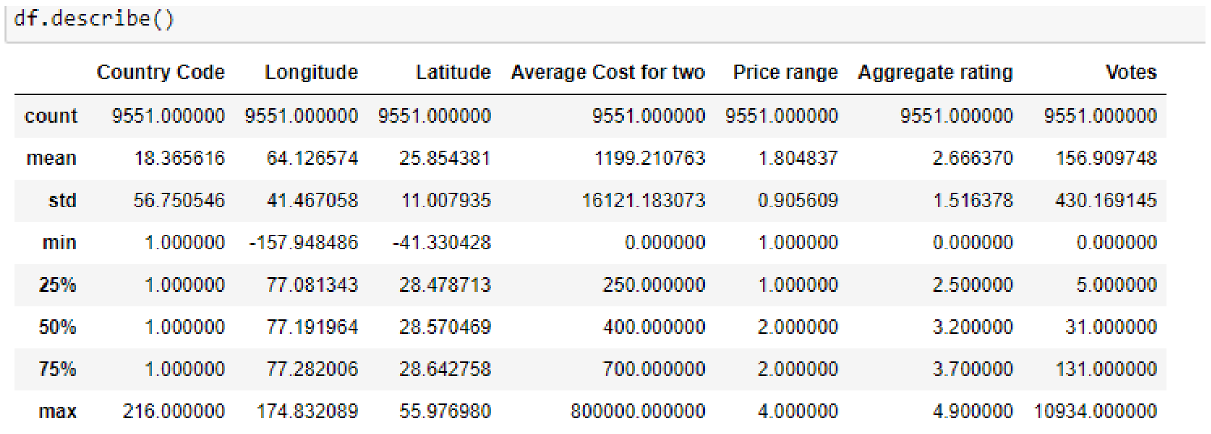
Among the resuarants having online delivery: Price range 2 hotels are most common

12) Outliers/noise checking for input numerical feature columns:



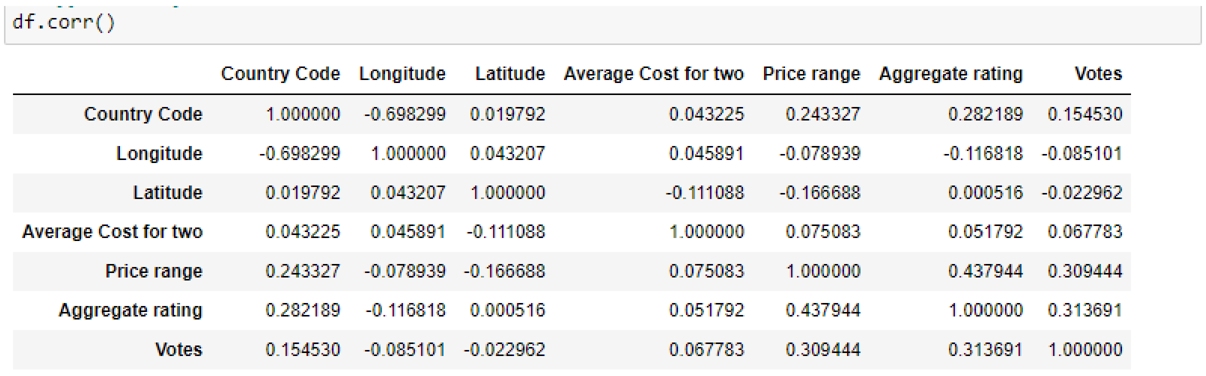
All data has outliers, need to be properly handled

13) Statistical summary of the data and correlation matrix:



```
df.describe()
```

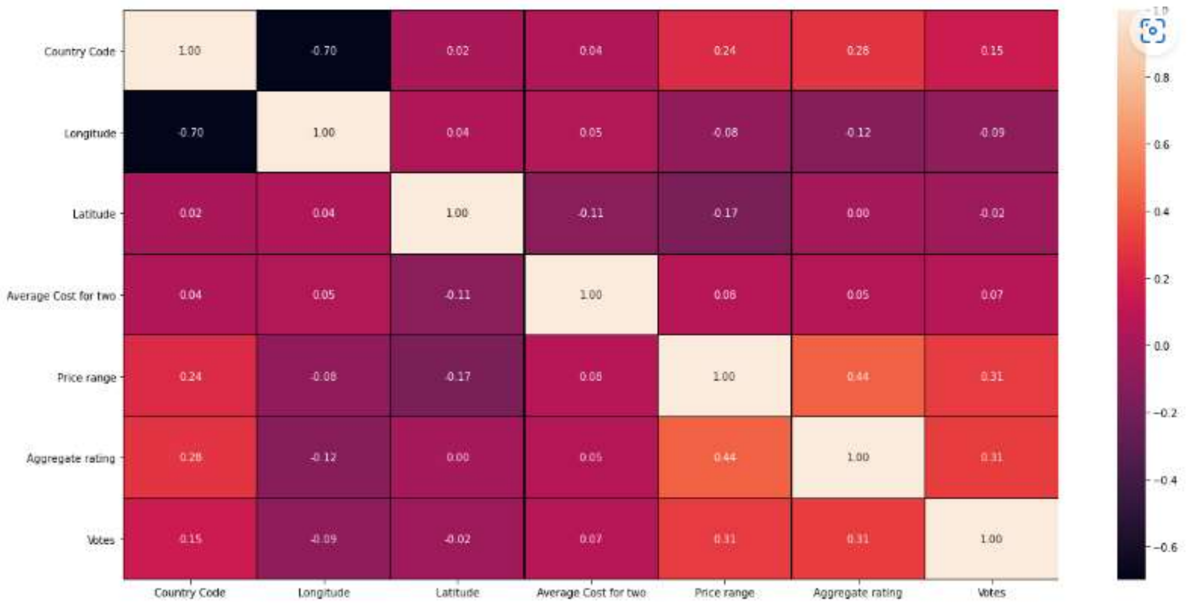
	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
count	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000
mean	18.365616	64.126574	25.854381	1199.210763	1.804837	2.666370	156.909748
std	56.750546	41.467058	11.007935	16121.183073	0.905609	1.516378	430.169145
min	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.000000	0.000000
25%	1.000000	77.081343	28.478713	250.000000	1.000000	2.500000	5.000000
50%	1.000000	77.191964	28.570469	400.000000	2.000000	3.200000	31.000000
75%	1.000000	77.282006	28.642758	700.000000	2.000000	3.700000	131.000000
max	216.000000	174.832089	55.976980	800000.000000	4.000000	4.900000	10934.000000



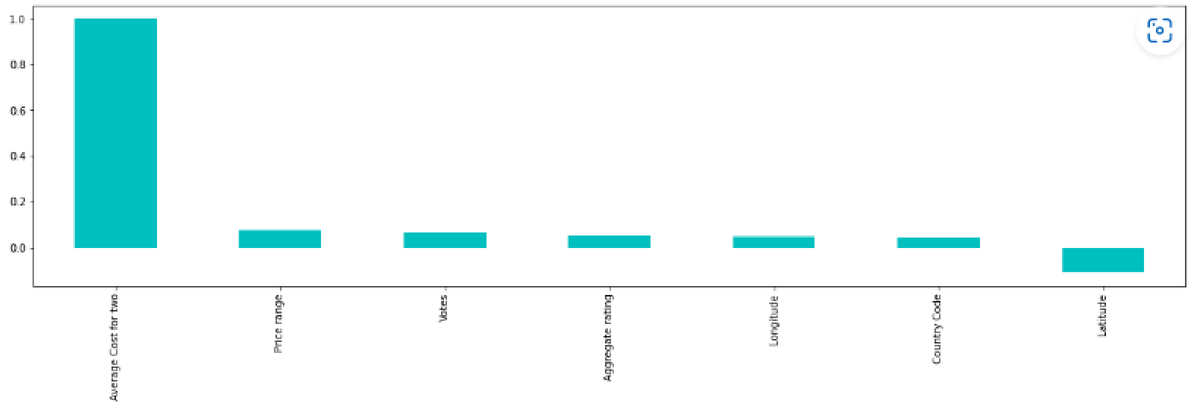
```
df.corr()
```

	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
Country Code	1.000000	-0.698299	0.019792	0.043225	0.243327	0.282189	0.154530
Longitude	-0.698299	1.000000	0.043207	0.045891	-0.078939	-0.116818	-0.085101
Latitude	0.019792	0.043207	1.000000	-0.111088	-0.166688	0.000516	-0.022962
Average Cost for two	0.043225	0.045891	-0.111088	1.000000	0.075083	0.051792	0.067783
Price range	0.243327	-0.078939	-0.166688	0.075083	1.000000	0.437944	0.309444
Aggregate rating	0.282189	-0.116818	0.000516	0.051792	0.437944	1.000000	0.313691
Votes	0.154530	-0.085101	-0.022962	0.067783	0.309444	0.313691	1.000000

14) Plotting heat map for proper visualization:



Sorting descending values of correlation data of target variable **Average Cost for two:**



```
Average Cost for two     1.000000
Price range              0.075083
Votes                    0.067783
Aggregate rating         0.051792
```

```
Longitude                0.045891
Country Code             0.043225
Latitude                -0.111088
```

15) Encoding the categorical/text type data into numerical form:

```
#Encoding the categorical/text type data into numerical form
from sklearn.preprocessing import LabelEncoder
enc = LabelEncoder()

for i in categorical:
    df[i] = enc.fit_transform(df[i].values.reshape(-1,1))
```

16) Removing of outliers by z-score method:

```
print('Shape before',df.shape)
from scipy.stats import zscore
z=np.abs(zscore(df[numerical]))
df_new = df[(z<3).all(axis=1)]
print('Shape after',df_new.shape)

#Percentage loss of data
loss = (df.shape[0]-df_new.shape[0])*100/(df.shape[0])
print(loss,'% loss of data')
```

```
Shape before (9551, 15)
Shape after (8756, 15)
8.323735734478065 % loss of data
```

Data loss is less than 10% after removal of outliers.

Model building:

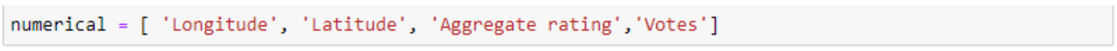
Average cost for two prediction (Regression approach)

Separating input and target feature columns:

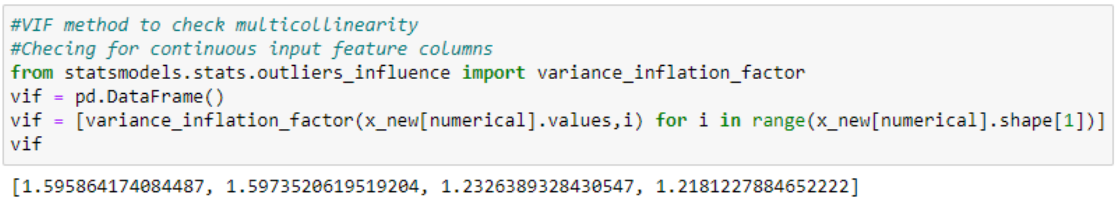
```
x = df_new.drop('Average Cost for two', axis=1)
y = df_new['Average Cost for two']
print('x shape',x.shape)
print('y shape',y.shape)
```

```
x shape (8756, 14)
y shape (8756,)
```

VIF values checking:



```
numerical = [ 'Longitude', 'Latitude', 'Aggregate rating','Votes']
```

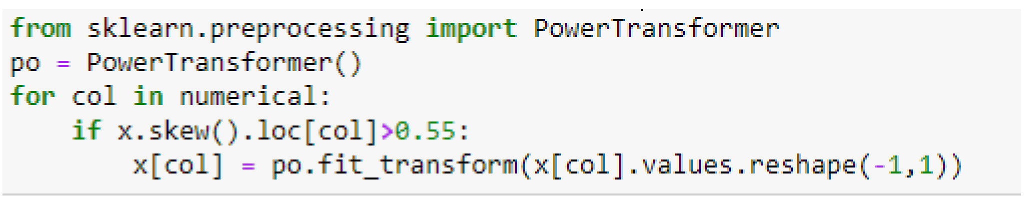


```
#VIF method to check multicollinearity
#Checing for continuous input feature columns
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif = [variance_inflation_factor(x_new[numerical].values,i) for i in range(x_new[numerical].shape[1])]
vif
```

```
[1.595864174084487, 1.5973520619519204, 1.2326389328430547, 1.2181227884652222]
```

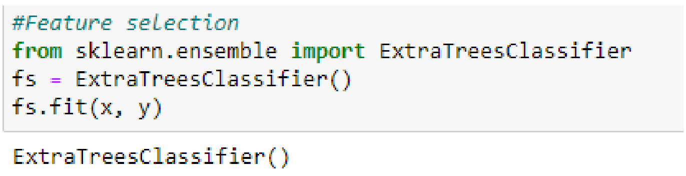
No issue of multicollinearity as the VIF value of columns having continuous data are within the limit of 5

Apply power transform to reduce skewness to less than 0.55:



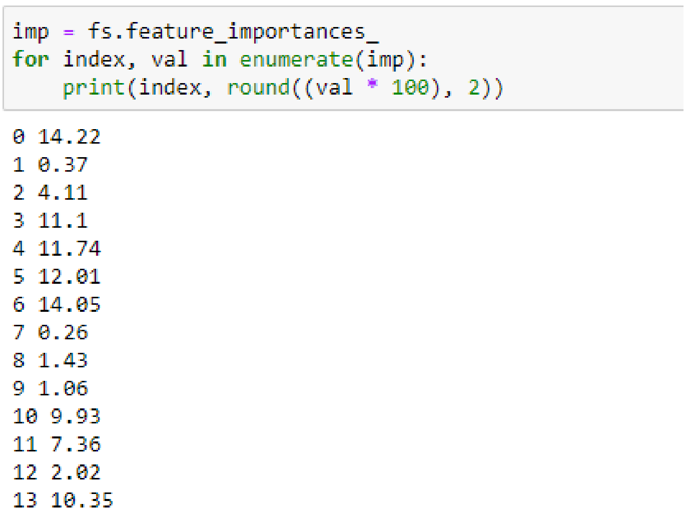
```
from sklearn.preprocessing import PowerTransformer
po = PowerTransformer()
for col in numerical:
    if x.skew().loc[col]>0.55:
        x[col] = po.fit_transform(x[col].values.reshape(-1,1))
```

Most important features selection:



```
#Feature selection
from sklearn.ensemble import ExtraTreesClassifier
fs = ExtraTreesClassifier()
fs.fit(x, y)
```

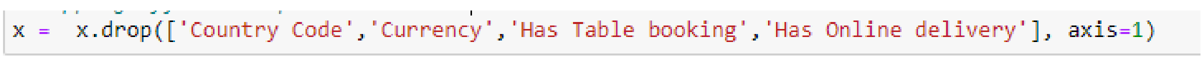
```
ExtraTreesClassifier()
```



```
imp = fs.feature_importances_
for index, val in enumerate(imp):
    print(index, round((val * 100), 2))
```

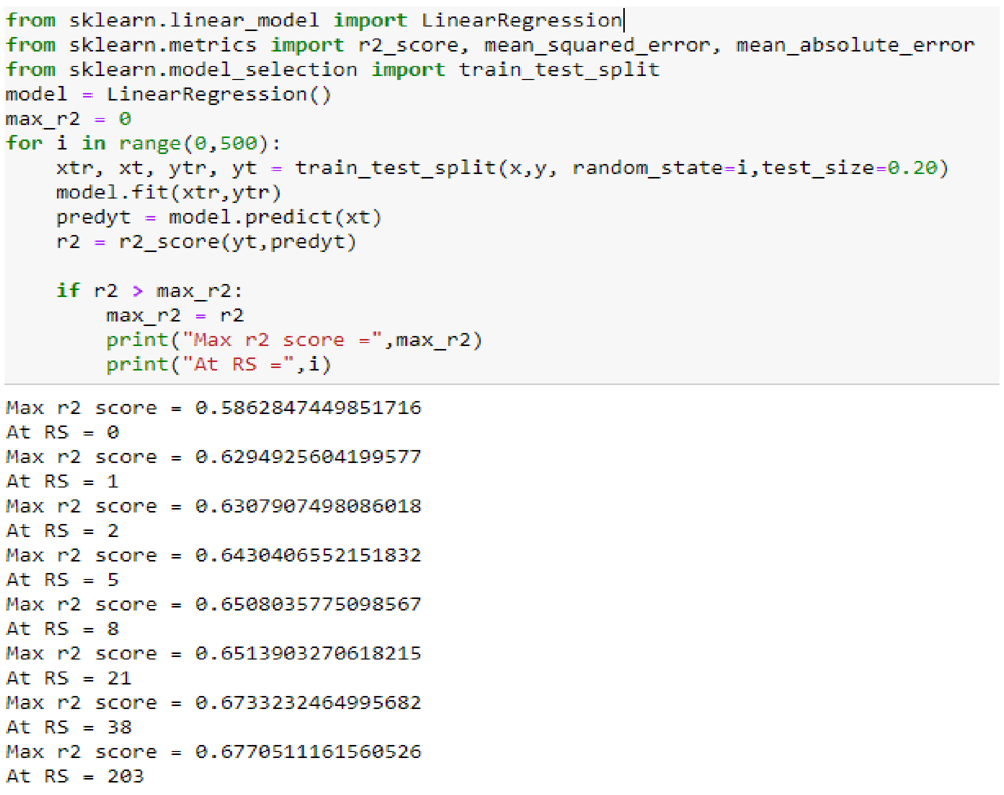
```
0 14.22
1 0.37
2 4.11
3 11.1
4 11.74
5 12.01
6 14.05
7 0.26
8 1.43
9 1.06
10 9.93
11 7.36
12 2.02
13 10.35
```

Let us select important features and drop rest columns. By considering top 70% are important. Dropping off less important columns.



```
x =  x.drop(['Country Code','Currency','Has Table booking','Has Online delivery'], axis=1)
```

Selecting of best random state for splitting train-test data:



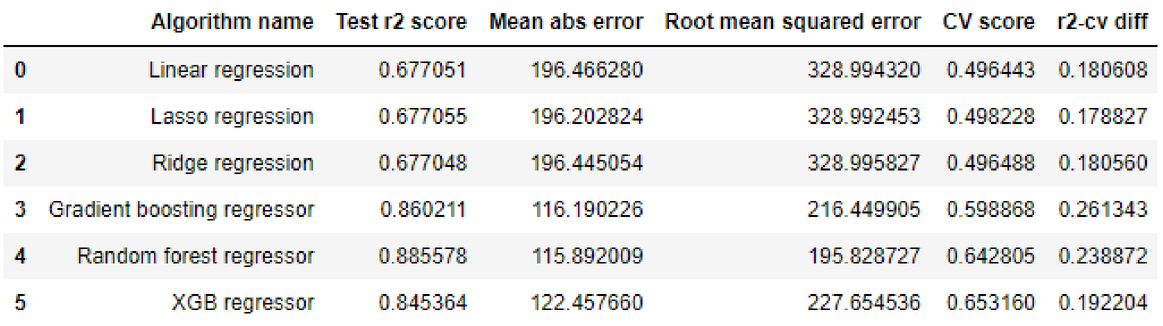
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
model = LinearRegression()
max_r2 = 0
for i in range(0,500):
    xtr, xt, ytr, yt = train_test_split(x,y, random_state=i,test_size=0.20)
    model.fit(xtr,ytr)
    predyt = model.predict(xt)
    r2 = r2_score(yt,predyt)

    if r2 > max_r2:
        max_r2 = r2
        print("Max r2 score =",max_r2)
        print("At RS =",i)
```

```
Max r2 score = 0.5862847449851716
At RS = 0
Max r2 score = 0.6294925604199577
At RS = 1
Max r2 score = 0.6307907498086018
At RS = 2
Max r2 score = 0.6430406552151832
At RS = 5
Max r2 score = 0.6508035775098567
At RS = 8
Max r2 score = 0.6513903270618215
At RS = 21
Max r2 score = 0.6733232464995682
At RS = 38
Max r2 score = 0.6770511161560526
At RS = 203
```

Steps in training ML algorithms:

1) Training with best random state
2) Cross validation to check over fitting
3) Hyper parameter tuning and selecting best parameters
4) Training with best parameters
5) Again cross validation to check over fitting
6) Saving the model result into a data frame

The above steps are repeated for following 6 regression algorithms and results compared:

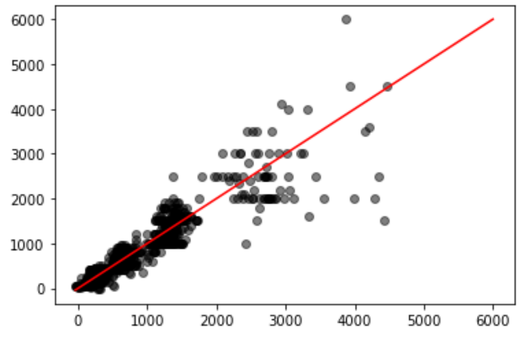


	Algorithm name	Test r2 score	Mean abs error	Root mean squared error	CV score	r2-cv diff
0	Linear regression	0.677051	196.466280	328.994320	0.496443	0.180608
1	Lasso regression	0.677055	196.202824	328.992453	0.498228	0.178827
2	Ridge regression	0.677048	196.445054	328.995827	0.496488	0.180560
3	Gradient boosting regressor	0.860211	116.190226	216.449905	0.598868	0.261343
4	Random forest regressor	0.885578	115.892009	195.828727	0.642805	0.238872
5	XGB regressor	0.845364	122.457660	227.654536	0.653160	0.192204

We can observe that XGB regressor giving the best results with max Test R2 score and least difference between CV score and test R2 score.

Prediction using best selected model and saving the model:

```
#Predicted data vs actual test data
model = xgb.XGBRegressor()
model.fit(xtr,ytr)
predyt = model.predict(xt)
actualyt = yt
plt.figure()
plt.scatter(predyt, actualyt, color = 'k', alpha=0.5)
plt.plot([min(predyt),max(actualyt)], [min(predyt),max(actualyt)], 'k-', color = 'r')
plt.show()
```



```
import pickle
filename='baseball_case.pkl'
pickle.dump(model,open(filename,'wb'))
```

Price Range prediction (Classification approach)

Separating input and target feature columns:

```
x = df_new.drop('Price range', axis=1)
y = df_new['Price range']
print('x shape',x.shape)
print('y shape',y.shape)
```

```
x shape (8756, 14)
y shape (8756,)
```

VIF values checking:

```
#We need to scale the data check VIF to handle multicollinearity
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_new = sc.fit_transform(x)
x_new = pd.DataFrame(x_new, columns = x.columns)

numerical = [ 'Longitude', 'Latitude', 'Aggregate rating','Votes']

#VIF method to check multicollinearity
#Checing for continuous input feature columns
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif = [variance_inflation_factor(x_new[numerical].values,i) for i in range(x_new[numerical].shape[1])]
vif
```

```
[1.595864174084487, 1.5973520619519204, 1.2326389328430547, 1.2181227884652222]
```

No issue of multicollinearity as the VIF value of columns having continuous data are within the limit of 5.

Apply power transform to reduce skewness to less than 0.55:

```
from sklearn.preprocessing import PowerTransformer
po = PowerTransformer()
for col in numerical:
    if x.skew().loc[col]>0.55:
        x[col] = po.fit_transform(x[col].values.reshape(-1,1))
```

Most important features selection:

```
x =x_new
```

```
#Feature selection
from sklearn.ensemble import ExtraTreesClassifier
fs = ExtraTreesClassifier()
fs.fit(x, y)
```

```
ExtraTreesClassifier()
```

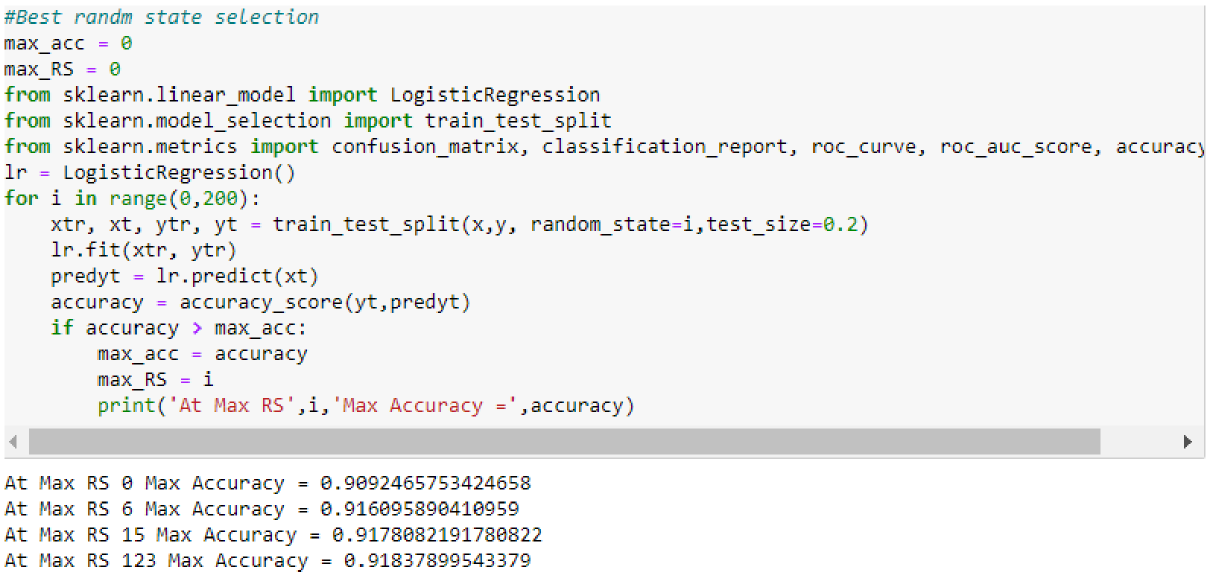
```
imp = fs.feature_importances_
for index, val in enumerate(imp):
    print(index, round((val * 100), 2))
```

```
0 4.87
1 1.0
2 2.97
3 4.55
4 4.57
5 4.95
6 6.12
7 40.54
8 0.52
9 8.24
10 3.61
11 5.98
12 2.39
13 9.7
```

Let us select important features and drop rest columns. By considering top 70% are important. Dropping off less important columns.

```
x =  x.drop(['Country Code','Has Table booking','City','Rating text'], axis=1)
```

Selecting of best random state for splitting train-test data:



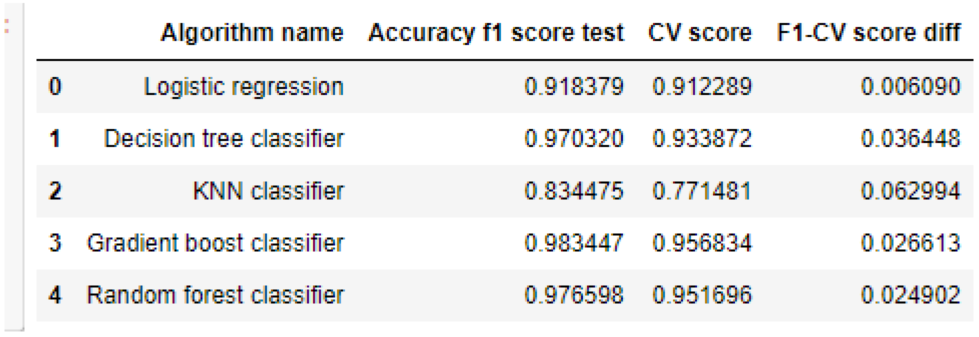
```
#Best randm state selection
max_acc = 0
max_RS = 0
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score, accuracy
lr = LogisticRegression()
for i in range(0,200):
    xtr, xt, ytr, yt = train_test_split(x,y, random_state=i,test_size=0.2)
    lr.fit(xtr, ytr)
    predyt = lr.predict(xt)
    accuracy = accuracy_score(yt,predyt)
    if accuracy > max_acc:
        max_acc = accuracy
        max_RS = i
        print('At Max RS',i,'Max Accuracy =',accuracy)
```

```
At Max RS 0 Max Accuracy = 0.9092465753424658
At Max RS 6 Max Accuracy = 0.916095890410959
At Max RS 15 Max Accuracy = 0.9178082191780822
At Max RS 123 Max Accuracy = 0.91837899543379
```

Steps in training ML algorithms:

1) Training with best random state
2) Cross validation to check over fitting
3) Hyper parameter tuning and selecting best parameters
4) Training with best parameters
5) Again cross validation to check over fitting
6) Saving the model result into a data frame

The above steps are repeated for following 5 classification algorithms and results compared:

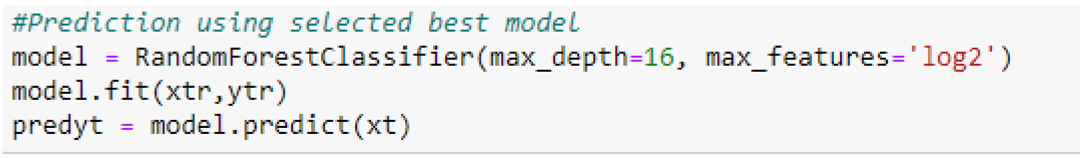


	Algorithm name	Accuracy f1 score test	CV score	F1-CV score diff
0	Logistic regression	0.918379	0.912289	0.006090
1	Decision tree classifier	0.970320	0.933872	0.036448
2	KNN classifier	0.834475	0.771481	0.062994
3	Gradient boost classifier	0.983447	0.956834	0.026613
4	Random forest classifier	0.976598	0.951696	0.024902

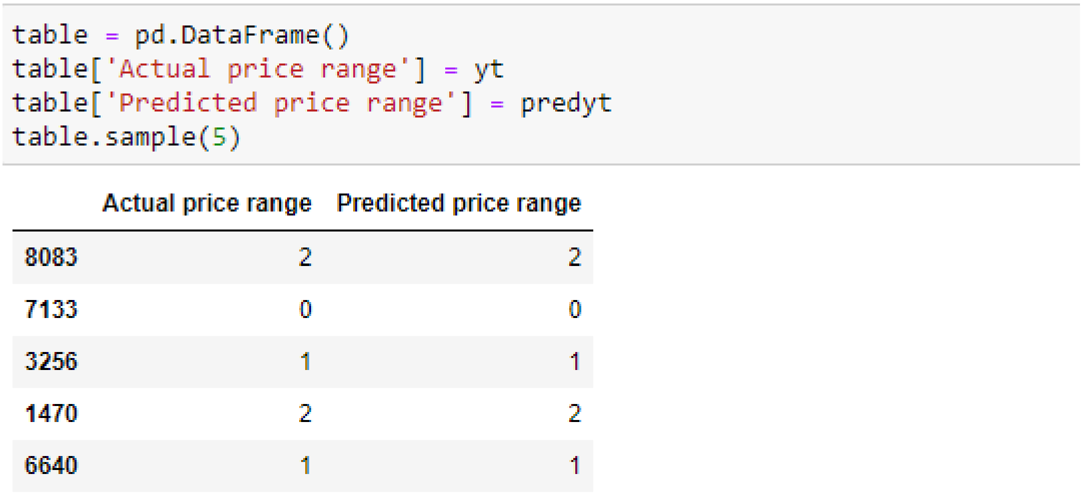
Both GB classifier and RF classifiers giving best results

Based on lowest CV-F1 test score differnce as minimum RFC can be selected as best model

Prediction using best selected model and saving the model:

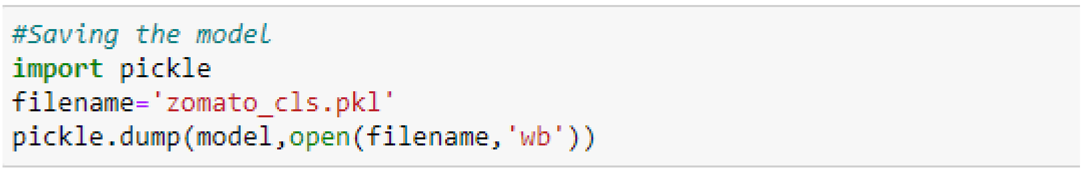


```
#Prediction using selected best model
model = RandomForestClassifier(max_depth=16, max_features='log2')
model.fit(xtr,ytr)
predyt = model.predict(xt)
```



```
table = pd.DataFrame()
table['Actual price range'] = yt
table['Predicted price range'] = predyt
table.sample(5)
```

	Actual price range	Predicted price range
8083	2	2
7133	0	0
3256	1	1
1470	2	2
6640	1	1



```
#Saving the model
import pickle
filename='zomato_cls.pkl'
pickle.dump(model,open(filename,'wb'))
```

17) Conclusion:

After using all these patient records, we are able to build the machine learning model

1) XGB regression – best one for average cost prediction
2) Random forest classifier – best one for price range prediction

Along with that we were able to draw some insights from the data via data analysis and visualization.