Restaurant Recommender

In this project we are going to assist a restaurant consolidator to build a restaurant recommender which will help a customer find the best restaurants in an area and find out which cuisines are served best so on and so forth. Factors such as delivery options, ratings and price options will be explored and the insights will be presented. Also we will find out what makes a restaurant more appealing to the customer and which factors make the ratings go up or down.

Step 1: Setting up the work environment

We are going to download the necessary packages for our work. We are going to view the dataset and check the datatypes.

In [1]:
```
#installing packages
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sb
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
import os
from collections import Counter
from numpy import mean
from numpy import std
from pandas import read_csv


from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import auc
from sklearn.metrics import make_scorer

from pandas import Series;  from numpy.random import randn
from statsmodels.stats.weightstats import ttest_ind
import scipy.stats as stats
from scipy.stats import ttest_ind

%matplotlib inline
from matplotlib import pyplot as plt
from scipy.stats import shapiro
from scipy import stats


from sklearn.metrics import classification_report





from datetime import datetime
from datetime import date
import statsmodels.api as sm
from statsmodels.formula.api import ols



from sklearn import preprocessing as preproc
from sklearn import metrics

from sklearn.utils import shuffle
from sklearn.pipeline import make_pipeline

from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
```

In [2]:
```
pip install lazypredict.supervised

from lazypredict.supervised import LazyRegressor
```

```
  Cell In [2], line 1
    pip install lazypredict.supervised
        ^
SyntaxError: invalid syntax
```

In [ ]:
```
#downloading the file
zom= pd.read_excel("C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Datasets for ML\\Regression\data.xlsx")
zom.head()
```

In [ ]:
```
#getting the secondary data
cn= pd.read_excel("C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Datasets for DV\\Country-Code.xlsx")
cn.head()
```

In [ ]:
```
#merging secondary into primary
rr=pd.merge(zom, cn, on='Country Code', how='left')
rr.head()
```

In [ ]:
```
#examining the dataset
rr.info()
```

In [ ]:
```
#examining the dataset
rr.shape
```

Step 2: Data cleaning

In this step, we are going to clean our dataset. We are going to look for null values and replace them with mean and mode. We are going to modify some variables if it is necessary and change datatypes for better analysis. We will also remove outliers from the dataset. Outliers hamper the machine learning algorithms and hence they have to be removed.

In [ ]:
```
# column name cleaning

rr.columns = rr.columns.str.replace(' ', '')
rr.columns
```

In [ ]:
```
#checking null values
rr.isna().sum(axis=0)
```

In [ ]:
```
#drop null
rr=rr.dropna()
rr
```

In [ ]:
```
#dropping useless columns
rr=rr.drop(['Address','Locality','LocalityVerbose','Longitude','Latitude','Ratingcolor','RestaurantID'], axis=1
rr.head()
```

Univariate analysis

We are using boxplot,histogram and plot to find the outliers,extreme values and distribution of the numerical variables.

In [ ]:
```
#checking distribution for Average Cost for two
rr['AverageCostfortwo'].plot.density()
```

In [ ]:
```
#checking distribution for Votes
rr['Votes'].plot.density()
```

In [ ]:
```
#checking distribution for Aggregate rating
rr['Aggregaterating'].plot.density()
```

In [ ]:
```
#checking outliers using boxplot Average Cost for two
sb.boxplot(rr['AverageCostfortwo'])
```

In [ ]:
```
#outlier treatment for variable Average Cost for two
rr['AverageCostfortwo'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

In [ ]:
```
#seeing limits
AverageCosttwo_HE=rr[rr['AverageCostfortwo'] > 600].copy()
```

In [ ]:
```
#putting them at one place
print(len(AverageCosttwo_HE))
```

In [ ]:
```
#removing outliers
for x in rr['AverageCostfortwo']:
    if x > 600.0:
        rr['AverageCostfortwo'].replace(x,np.nan,inplace=True)
```

In [ ]:
```
#checking outliers gone or not
sb.boxplot(rr['AverageCostfortwo'])
```

In [ ]:
```
#replacing with median values

medacft=rr['AverageCostfortwo'].median()
print(medacft)
rr['AverageCostfortwo'].replace(np.nan, medacft, inplace=True)
```

In [ ]:
```
#checking outliers using boxplot Votes
sb.boxplot(rr['Votes'])
```

In [ ]:
```
#outlier treatment for variable votes
rr['Votes'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

In [ ]:
```
#seeing limits
vote_HE=rr[rr['Votes'] > 31].copy()
```

In [ ]:
```
#putting them at one place
print(len(vote_HE))
```

In [ ]:
```
#removing outliers
for x in rr['Votes']:
    if x > 31:
        rr['Votes'].replace(x,np.nan,inplace=True)
```

In [ ]:
```
#checking if outliers gone
sb.boxplot(rr['Votes'])
```

In [ ]:
```
#replacing with median values

medvote=rr['Votes'].median()
print(medvote)
rr['Votes'].replace(np.nan, medvote, inplace=True)
```

In [ ]:
```
#checking outliers using boxplot Aggregate rating
sb.boxplot(rr['Aggregaterating'])
```

In [ ]:
```
#outlier treatment for variable Aggregate rating
rr['Aggregaterating'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

In [ ]:
```
#seeing limits
ar_LE=rr[rr['Aggregaterating'] <2.5].copy()
ar_HE=rr[rr['Aggregaterating'] >4.7].copy()
```

In [ ]:
```
#removing outliers
for x in rr['Aggregaterating']:
    if x < 2.5:
        rr['Aggregaterating'].replace(x,np.nan,inplace=True)

for x in rr['Aggregaterating']:
    if x > 4.7:
        rr['Aggregaterating'].replace(x,np.nan,inplace=True)
```

In [ ]:
```
#checking if outliers gone
sb.boxplot(rr['Aggregaterating'])
```

In [ ]:
```
#replacing with median values

medar=rr['Aggregaterating'].median()
print(medar)
rr['Aggregaterating'].replace(np.nan, medar, inplace=True)
```

Step 3:Exploratory data analysis

In this step, we are going to explore the dataset. Perform hypothesis tests,bivariate analysis and check for correlation between variables.

In [ ]:
```
#summary stats
rr.describe().round(2)
```

In [ ]:
```
#seeing correlation of numerical variables
num_vars=['AverageCostfortwo','Aggregaterating','Votes']
cordata =rr[num_vars].corr()
cordata.round(2)
```

In [ ]:
```
#heatmap of correlation
sb.heatmap(cordata, annot=True, cmap="Greens")
```

In [ ]:
```
#seeing the top franchise
francnt=rr['RestaurantName'].value_counts().head(5)
francnt.plot(kind = "bar")
```

In [ ]:
```
#top 3 cities with most restaurants

n_by_city= rr.groupby("City")["RestaurantName"].count().nlargest(3)
n_by_city.plot(kind = "bar")
```

In [ ]:
```
#Which country makes up the most for zomato restaurants
rest=rr['Country'].unique()
numbers=rr['Country'].value_counts()


plt.pie(numbers, labels = rest, startangle = 75,autopct='%.0f%%')
plt.show()
```

In [ ]:
```
#Are delivery restaurants more or not?
rest=rr['HasOnlinedelivery'].unique()
numbers=rr['HasOnlinedelivery'].value_counts()


plt.pie(numbers, labels = rest, startangle = 75,autopct='%.2f%%')
plt.show()
```

In [ ]:
```
#Are Table booking restaurants more or not?
bookyes= len(rr[rr['HasTablebooking']=='Yes'])
bookno=len(rr[rr['HasTablebooking']=='No'])

ratio=bookyes/bookno

print("The ratio to restaurants that allow bookings vs dont allow",ratio)
```

In [ ]:
```
#difference of votes btw delivery non delivery restaurants
bookyes= rr[rr['HasOnlinedelivery']=='Yes']
bookno=rr[rr['HasOnlinedelivery']=='No']
bookyesvote= bookyes["Votes"].sum()
booknovote= bookno["Votes"].sum()
diff=bookyesvote-booknovote
print("The difference of votes between delivery and non delivery restaurants is",diff)
```

In [ ]:
```
#cuisine count
print("The most number of cuisines offered by a restaurant is",rr['cuisinecount'].max(),
      "and the least is",rr['cuisinecount'].min())
```

In [ ]:
```
#Average cost of meals across top restaurants

n_by_city= rr.groupby("RestaurantName")["AverageCostfortwo"].mean().nlargest(7)
n_by_city.plot(kind = "bar")
```

In [ ]:
```
#box plot to see distribution of ratings across price range
plt.figure(figsize=[12,6])
rr.boxplot(by="Pricerange",column="Aggregaterating", grid = False)
```

In [ ]:
```
#checking if Rating differs according to the Price range of the restaurant
mod = ols("Aggregaterating ~ Pricerange", data = rr).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

In [ ]:
```
#box plot to see distribution of ratings across delivering/non delivering restaurants
plt.figure(figsize=[12,6])
rr.boxplot(by="HasOnlinedelivery",column="Aggregaterating", grid = False)
```

In [ ]:
```
#checking if Rating differs across delivering/non delivering restaurants
mod = ols("Aggregaterating ~ HasOnlinedelivery", data = rr).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

In [ ]:
```
#box plot to see distribution of ratings across online/non online booking restaurants
plt.figure(figsize=[12,6])
rr.boxplot(by="HasTablebooking",column="Aggregaterating", grid = False)
```

In [ ]:
```
#checking if Rating differs across online/non online booking restaurants
mod = ols("Aggregaterating ~ HasTablebooking", data = rr).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

In [ ]:
```
#number of distinct cuisines
list_all = rr['Cuisines'].str.split(r'(?:,|;)\s*').dropna().to_numpy()

list_unique = np.unique(sum(list_all, []))
list_unique=list(list_unique)
list_unique
```

In [ ]:
```
# Seeing which cuisine is the most popular
all_cuisines = []

# iterate through each cell in the column and add the words to the list
for cell in rr['Cuisines']:
    cuisines = cell.split(', ') # split the cell into a list of words
    all_cuisines.extend(cuisines) # add the words to the list

# count the occurrences of each word using the Counter() function
cuisine_counts = Counter(all_cuisines)


#seeing top 10 most popular cuisines
top_cuisines=cuisine_counts.most_common(10)
top_cuisines
```

In [ ]:
```
# create a bar plot of the top 10 cuisines
fig, ax = plt.subplots(figsize=(12, 8))
ax.bar([cuisine[0] for cuisine in top_cuisines], [cuisine[1] for cuisine in top_cuisines])
ax.set_xlabel('cuisine')
ax.set_ylabel('Count')
ax.set_title('Top 10 cuisines')
plt.show()
```

Insights

1.The most popular dishes are chinese and North indian.

2.The most number of restaurants are in India (around 91%).

3.The average cost of meals is mostly same everywhere.

4.New delhi has the most restaurants.

5.CCD, MCD and Subway are some top franchises.

6.The ratings go down when the votes are more and the cost is more.

7.Non delivery restaurants have higher number of votes than delivery ones.

8.Online booking restaurants are 13% to 14%.

9.Around 3/4th of restaurants have home delivery.

10.There are around 143 different cuisines.

11.The ratings doesnt depend on home delivery options but on table booking options and price level.

12.Restaurants can offer upto at most 8 types of cuisines.

Step 4: Model Building

After the data has been cleaned and formatted. Now its time to analyse and get insights. We will uuse Linear Regression to get the factors affecting ratings.

In [ ]:
```
rr.head()
```

In [ ]:
```
#dropping not needed variables
rr1=rr.drop(['RestaurantName','City','Ratingtext'],axis=1)
rr1.head(0)
```

In [ ]:
```
#creating target variable
target=rr['Aggregaterating']
```

In [ ]:
```
#removing target variable from main dataframe
rr1= rr.drop(['Aggregaterating'], axis=1)
```

In [ ]:
```
#defining category variables
category_variables=['CountryCode', 'cuisinecount','Cuisines',
        'HasTablebooking', 'HasOnlinedelivery', 'Pricerange', 'Country']
category_variables
```

In [ ]:
```
#initalizing encoder
label_encoder=preproc.LabelEncoder()
```

In [ ]:
```
#encoding categorical variables
cusenc=label_encoder.fit_transform(rr1['Cuisines'])
odenc=label_encoder.fit_transform(rr1['HasOnlinedelivery'])
tbenc=label_encoder.fit_transform(rr1['HasTablebooking'])
cntenc=label_encoder.fit_transform(rr1['Country'])
```

In [ ]:
```
# Create linear regression object
regressor = LinearRegression()
```

In [ ]:
```
#setting values of S and y
X=pd.DataFrame([cusenc,
                 odenc,
                 tbenc,
                 cntenc,
                 rr1['CountryCode'],
                 rr1['cuisinecount'],
                 rr1['Pricerange']]).T
y= target.values
```

In [ ]:
```
# Train, test, split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = .30, random_state= 1000)
```

In [ ]:
```
# Fit model to training data
regressor.fit(X_train,y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=-2, normalize=False)
```

In [ ]:
```
# Predict
# Predicting test set results
y_pred = regressor.predict(X_test)
y_pred
```

In [ ]:
```
# Calculated R Squared
print('R^2 =',metrics.explained_variance_score(y_test,y_pred))
```

In [ ]:
```
# Actual v predictions scatter
plt.scatter(y_test, y_pred)
```

In [ ]:
```
# Histogram of the distribution of residuals
sb.distplot((y_test - y_pred))
```

In [ ]:
```
#coefficients
cdf = pd.DataFrame(data = regressor.coef_, index = X.columns, columns = ['Coefficients'])
cdf
```

Conclusion

The model is not very good as it can explain only 25% of the change in the dependent variable. The fit is not achieved. The variables are not enough and maybe another methodology needs to be used to develop this model.