Project Report

Batch details	PGPDSE-FT Online Jul22
Team members	Chaturima M Aishwarya Mishra Mohit Changlani Sejal Baweja
Proposed project title	SLR, SLC and USL Mini Project
Group Number	Group 11

Date: 14-05-2023

Table of Contents

Project Details

OVERVIEW:

Zomato is an Indian multinational restaurant aggregator and food delivery company founded by Deepinder Goyal and Pankaj Chaddah in 2008. Zomato provides information, menus, and user reviews of restaurants as well as food delivery options from partner restaurants in select cities.

Attributes:

1. URL - Website of the Zomato for each restaurant. - Object datatype
2. Address - Address of the Restaurant. - Object datatype
3. Name - Name of the restaurant. - Object datatype
4. Online Order - The customer ordered the menu online or not. - Object datatype
5. Book table - The customer has booked the table or not. - Object datatype
6. Rate - Rating of the restaurant that was given by the customer. - Numerical datatype
7. Votes - The votes have been given by the customer to the restaurant. - Numerical Datatype
8. Phone - Contact number of the Restaurant. - Object datatype
9. Location - The city name where the restaurant is located. - Object datatype
10. Rest Type - The type of restaurant. - Object datatype
11. Dish liked - Dishes liked by the customer from the restaurant. - Object datatype
12. Cuisines - The cuisines that have been prepared by the restaurant. - Object datatype
13. Approx Cost for two people - The approximate cost of the customer for 2 people. - Number datatype
14. Reviews list - The reviews made by the customers in the restaurant. - Object Datatype
15. Menu Item - The menu items that are usually available at the restaurant. - Object Datatype

16. Listed in (type) - Contains the type of the meal. - Object datatype

17. Listed in (city) - This contains the neighborhood in which the restaurant is listed. -

Object datatype

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from warnings import filterwarnings
filterwarnings('ignore')
from scipy import stats
import statistics
import statsmodels.api as sm
from scipy import stats
import statistics
from scipy.stats import shapiro
from statsmodels.stats import weightstats as stests
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
from scipy.stats import chi2_contingency
from scipy.stats import chi2
from scipy.stats import chisquare
import statsmodels.stats.multicomp as mc
from sklearn import datasets, linear_model, metrics
from sklearn.model_selection import train_test_split
from statsmodels.api import add_constant
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
print('imported all libraries')
```

```
imported all libraries
```

Data

```
# read the data
df_zomato = pd.read_csv('/content/drive/MyDrive/ zomato.csv')

# display the first five observations
df_zomato.head()
```

	url	address	name	online_order	book_table	rate	votes	phone	location
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashankari
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashankari
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banashankari
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302	Banashankari
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	+91 8026612447\r\n+91 9901210005	Basavanagudi

```
# checking shape of data
df_zomato.shape
```

```
(51717, 17)
```

```
[ ] # checking datatypes of data
    df_zomato.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   url                          51717 non-null  object
 1   address                      51717 non-null  object
 2   name                         51717 non-null  object
 3   online_order                 51717 non-null  object
 4   book_table                   51717 non-null  object
 5   rate                         43942 non-null  object
 6   votes                        51717 non-null  int64
 7   phone                        50509 non-null  object
 8   location                     51696 non-null  object
 9   rest_type                    51490 non-null  object
 10  dish_liked                   23639 non-null  object
 11  cuisines                     51672 non-null  object
 12  approx_cost(for two people)  51371 non-null  object
 13  reviews_list                 51717 non-null  object
 14  menu_item                    51717 non-null  object
 15  listed_in(type)              51717 non-null  object
 16  listed_in(city)              51717 non-null  object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

Cleaning the data

a. Removing unwanted columns

```
# dropping unwanted columns
df.drop(['name','url', 'address', 'phone','dish_liked','reviews_list','menu_item'], axis=1, inplace=True)
```

b. Checking for duplicate entries & treating them

```
# checking for duplicate entries
df.duplicated().sum()
```

```
372
```

```
#calculating ratio of duplicated values to entire dataset
df.duplicated().sum()/df.shape[0]
```

```
0.007192992632983352
```

```
# dropping duplicate rows
df = df.drop_duplicates()

# checking shape of data after dropping duplicates
df.shape
```

```
(51345, 10)
```

c. Renaming column names for better understanding

```
# checking column names
df.columns
```

```
Index(['online_order', 'book_table', 'rate', 'votes', 'location', 'rest_type',
       'cuisines', 'approx_cost(for two people)', 'listed_in(type)',
       'listed_in(city)'],
      dtype='object')
```

```
#renaming some columns for better understanding
df.rename(columns = {'approx_cost(for two people)':'cost', 'listed_in(type)':'service', 'listed_in(city)':'city'}, inplace=True)
```

```
df.columns
```

```
Index(['online_order', 'book_table', 'rate', 'votes', 'location', 'rest_type',
       'cuisines', 'cost', 'service', 'city'],
      dtype='object')
```

d. Checking for missing values

```
# checking for missing values
df.isnull().sum()
```

```
online_order       0
book_table         0
rate            7520
votes              0
location          19
rest_type        225
cuisines          43
cost             341
service            0
city               0
dtype: int64
```

e. Treating missing values

```
[ ] # # getting index of all rows having missing value in 'location' column
    indexLocation = df[df.location.isnull()].index
```

```
[ ] # # dropping rows with missing value in 'location' column
    df = df.drop(indexLocation)
```

```
[ ] # # checking for missing values in 'location' column
    df['location'].isnull().sum()

    0
```

```
[ ] df.shape

    (51326, 10)
```

```
[ ] # converting datatype of 'rate' column to string
    df['rate'] = df['rate'].astype(str)
```

```
[ ] #dropping rows having 'rate' as '-' or 'NEW'
    df = df[~((df['rate'] == '-') | (df['rate'] == 'NEW'))]
    df.shape

    (49086, 10)
```

```
[ ] df['rate'] = df['rate'].apply(lambda x: x.replace('/5',''))
```

```
[ ] # converting datatype of 'rate' column to float
    df['rate'] = df['rate'].astype(float)
```

```
    df['rate'].isnull().sum()

    7501
```

```
[ ] df['rate'].mean()

    3.70016352050018
```

```
[ ] df['rate'].median()

    3.7
```

```
[ ] df['rate'] = df['rate'].fillna(df['rate'].median())
```

```
[ ] df['rate'].isnull().sum()

    0
```

```
[ ] df['cost'].isnull().sum()/df['cost'].shape[0]

    0.00645805321272868
```

As number of missing values in 'cost' column has very small ratio as compared to entire datset,we can remove these rows.

```
    indexLocation = df[df['cost'].isnull()].index
    df = df.drop(indexLocation)
```

```
[ ] df['cuisines'].isnull().sum()/df['cuisines'].shape[0]

    0.0003895917488568558
```

As number of missing values in 'cuisines' column has very small ratio as compared to entire datset,we can remove these rows.

```
[ ] indexLocation = df[df['cuisines'].isnull()].index
    df = df.drop(indexLocation)
```

```
[ ] df['rest_type'].isnull().sum()/df['rest_type'].shape[0]

    0.00414358974358974
```

As number of missing values in 'rest_type' column has very small ratio as compared to entire datset,we can remove these rows.

```
[ ] indexLocation = df[df['rest_type'].isnull()].index
    df = df.drop(indexLocation)
```

```
df['cost'] = df['cost'].str.replace(',','').astype('int')
df['cost']
```

```
0         800
1         800
2         800
3         300
4         600
        ...
51712    1500
51713     600
51714    2000
51715    2500
51716    1500
Name: cost, Length: 48548, dtype: int64
```

```
[ ] df.head()
```

	online_order	book_table	rate	votes	location	rest_type	cuisines	cost	service	city
0	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	Buffet	Banashankari
1	Yes	No	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	Buffet	Banashankari
2	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	Buffet	Banashankari
3	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300	Buffet	Banashankari
4	No	No	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	Buffet	Banashankari

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48548 entries, 0 to 51716
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   online_order  48548 non-null  object
 1   book_table    48548 non-null  object
 2   rate          48548 non-null  float64
 3   votes         48548 non-null  int64
 4   location      48548 non-null  object
 5   rest_type     48548 non-null  object
 6   cuisines      48548 non-null  object
 7   cost          48548 non-null  int64
 8   service       48548 non-null  object
 9   city          48548 non-null  object
dtypes: float64(1), int64(2), object(7)
memory usage: 4.1+ MB
```

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

	count	mean	std	min	25%	50%	75%	max
rate	48548.0	3.701487	0.405196	1.8	3.5	3.7	4.0	4.9
votes	48548.0	299.108820	823.721029	0.0	9.0	48.0	214.0	16832.0
cost	48548.0	562.973037	444.516649	40.0	300.0	400.0	700.0	6000.0

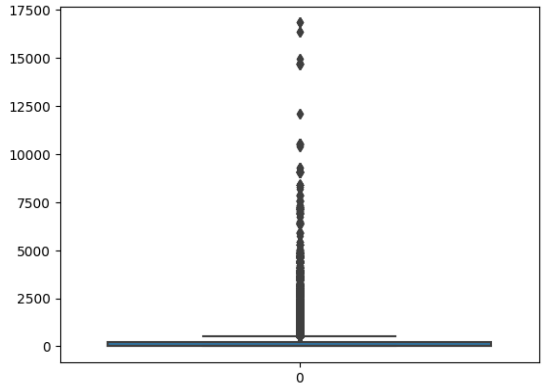
Interpretation:

1. For 'votes' column, mean>>median. Hence, we can say that this column is highly skewed and also there is huge difference between 75% of the data and the maximum value. So, we can say that this column contains outliers.
2. Similarly, for 'cost' column we can say that this column is also skewed and contains outliers.

f. Checking for outliers

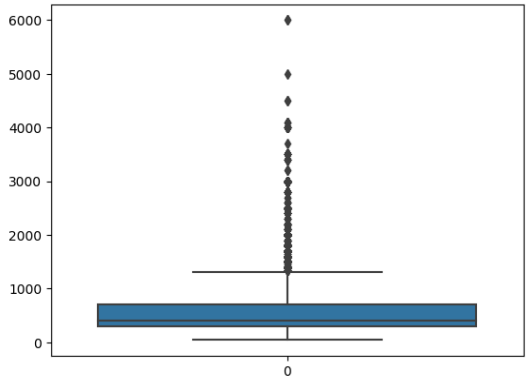
```
# checking for outliers in 'votes' column
sns.boxplot(df['votes'])
print(df['votes'].skew())
```

```
7.3912132212160415
```



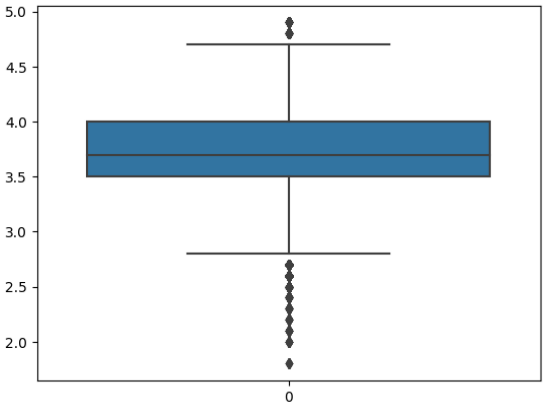

```
# checking for outliers in 'cost' column
sns.boxplot(df['cost'])
print(df['cost'].skew())
```

```
2.5825486058125673
```




```
# checking for outliers in 'rate' column
sns.boxplot(df['rate'])
print(df['rate'].skew())
```

```
-0.3545252165185422
```



g. Treating outliers

```
# calculating interquartile range
q1=df.quantile(.25)
q3=df.quantile(.75)
iqr=q3-q1
```

```
iqr
```

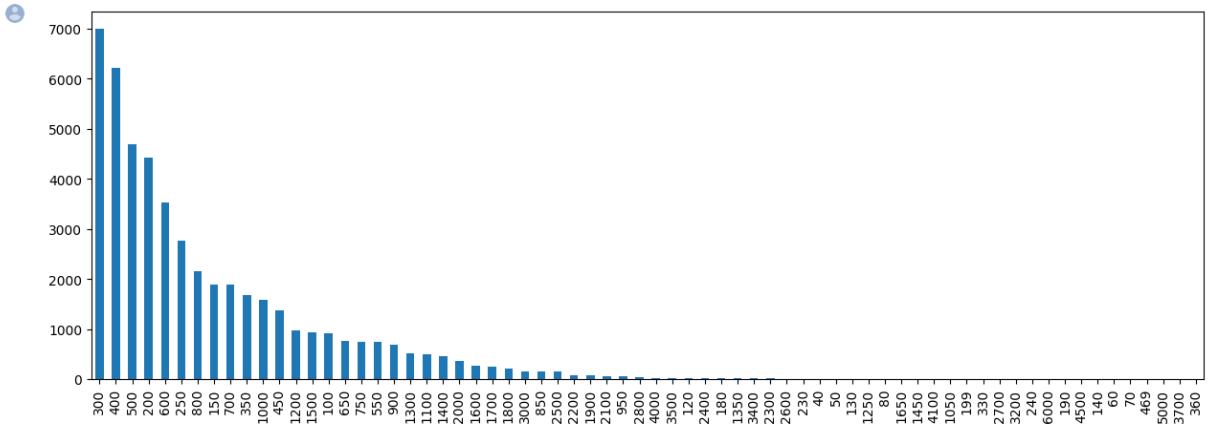
```
rate      0.5
votes   205.0
cost    400.0
dtype: float64
```

```
[ ]  # treating outliers
     df[((df < (q1-1.5*iqr)) | (df > (q3+1.5*iqr))).any(axis=1)]
```

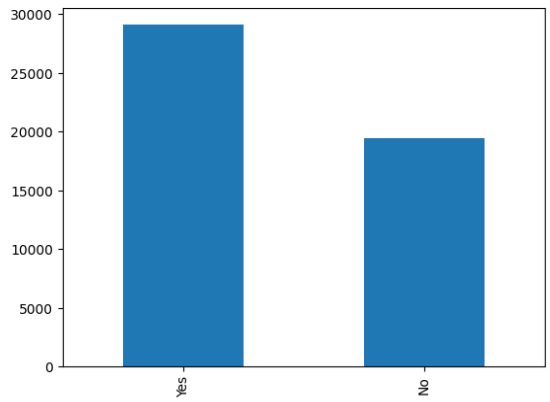
	online_order	book_table	rate	votes	location
0	Yes	Yes	4.1	775	Banashankari
1	Yes	No	4.1	787	Banashankari
2	Yes	No	3.8	918	Banashankari
7	Yes	Yes	4.6	2556	Banashankari
14	Yes	No	3.8	918	Banashankari

h. Univariate Data Analysis

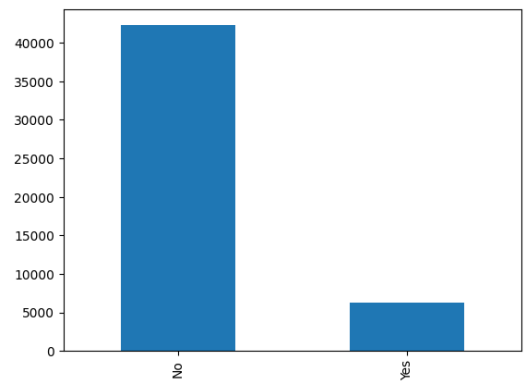
```
# plotting count of unique values of 'cost' column
figure=plt.figure(figsize=(15,5))
df['cost'].value_counts().plot(kind='bar')
plt.show()
```



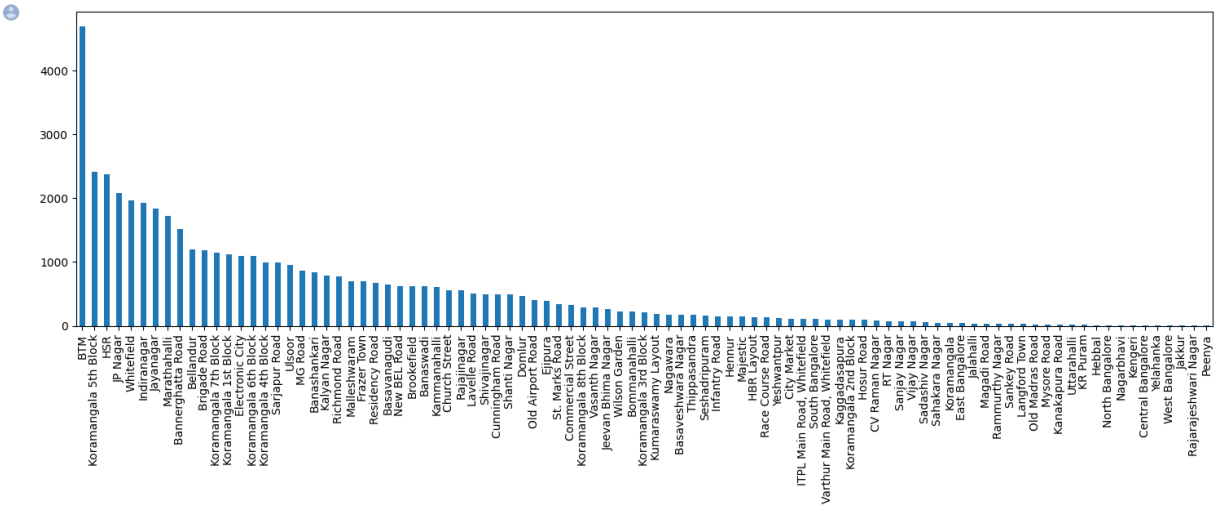

```
[ ] # plotting count of unique values of 'online_order' column
    df['online_order'].value_counts().plot(kind='bar')
    plt.show()
```



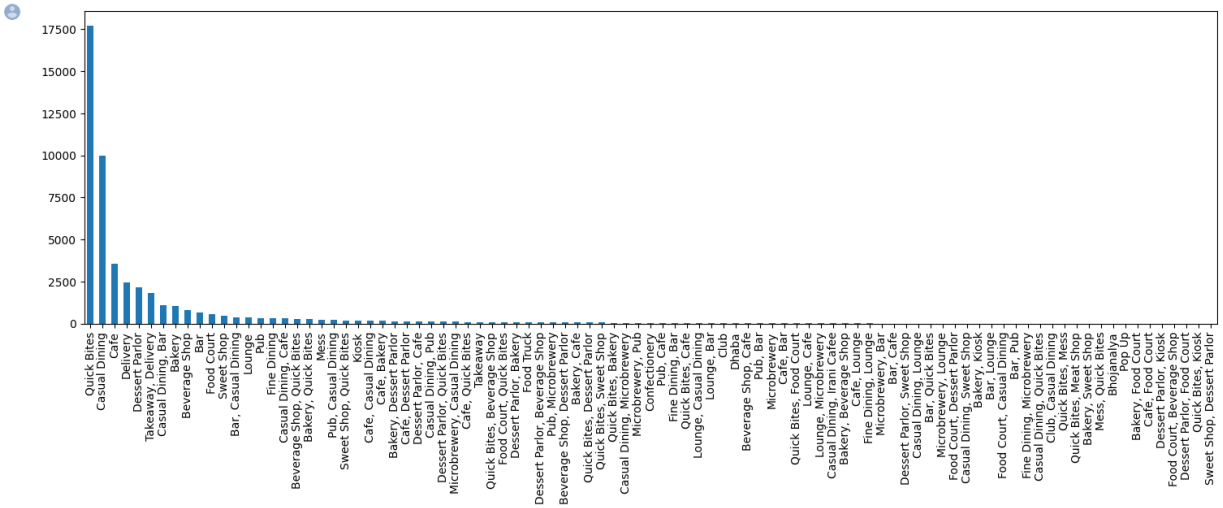

```
# plotting count of unique values of 'book_table' column
df['book_table'].value_counts().plot(kind='bar')
plt.show()
```



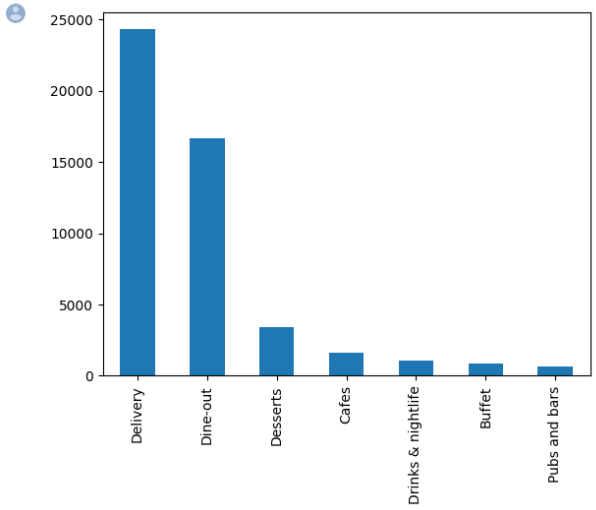

```
# plotting count of unique values of 'location' column
figure=plt.figure(figsize=(18,5))
df['location'].value_counts().plot(kind='bar')
plt.show()
```



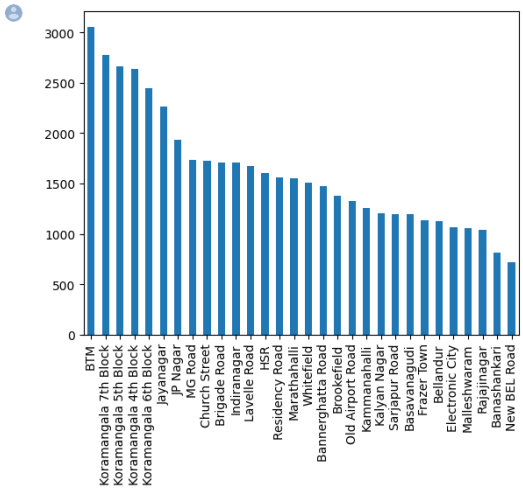
```
# plotting count of unique values of 'rest_type' column
figure=plt.figure(figsize=(18,5))
df['rest_type'].value_counts().plot(kind='bar')
plt.show()
```




```
# plotting count of unique values of 'service' column
df['service'].value_counts().plot(kind='bar')
plt.show()
```

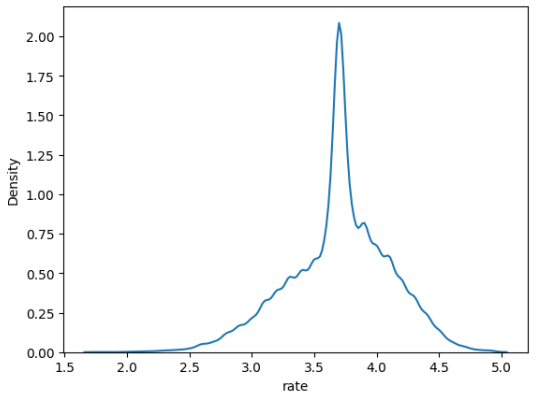



```
# plotting count of unique values of 'city' column
df['city'].value_counts().plot(kind='bar')
plt.show()
```



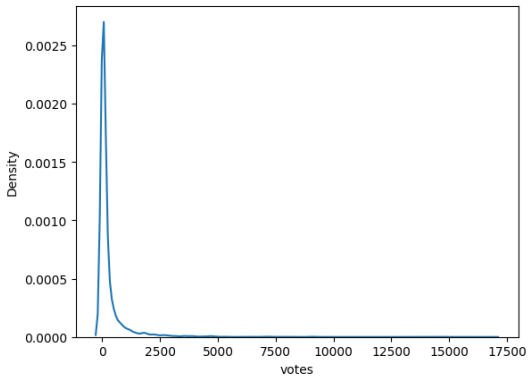
```
# checking skewness of 'rate' column
print("Skewness:", df.rate.skew())
sns.distplot(df.rate, hist=False)
plt.show()
```

```
Skewness: -0.3545252165185422
```



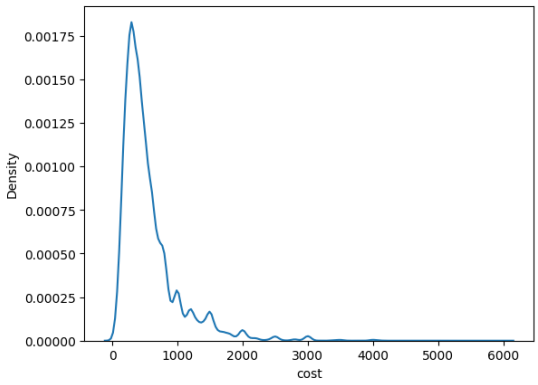

```
# checking skewness of 'votes' column
print("Skewness:", df.votes.skew())
sns.distplot(df['votes'], hist=False)
plt.show()
```

```
Skewness: 7.3912132212160415
```

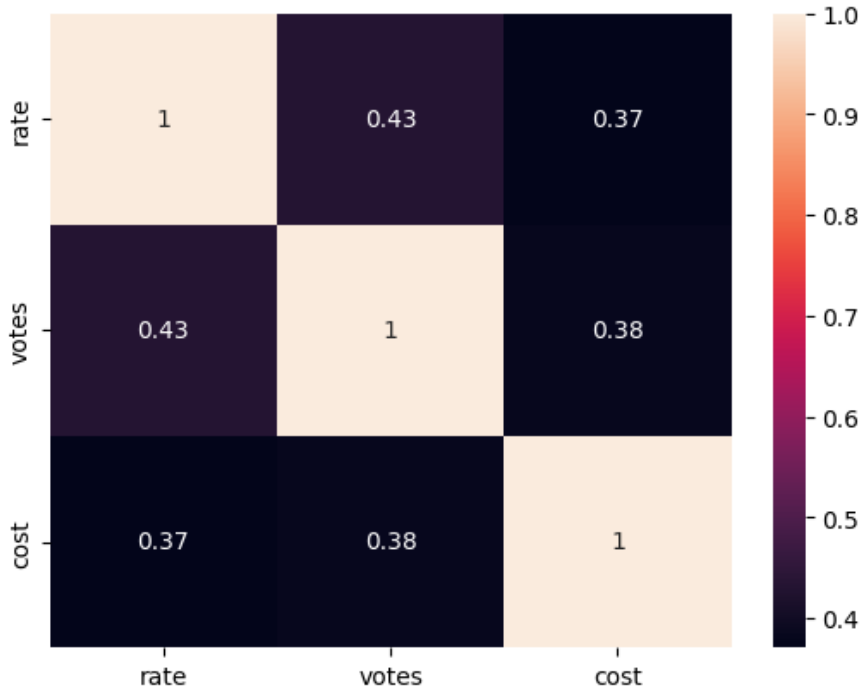



```
# checking skewness of 'cost' column
print("Skewness:", df.cost.skew())
sns.distplot(df['cost'], hist=False)
plt.show()
```

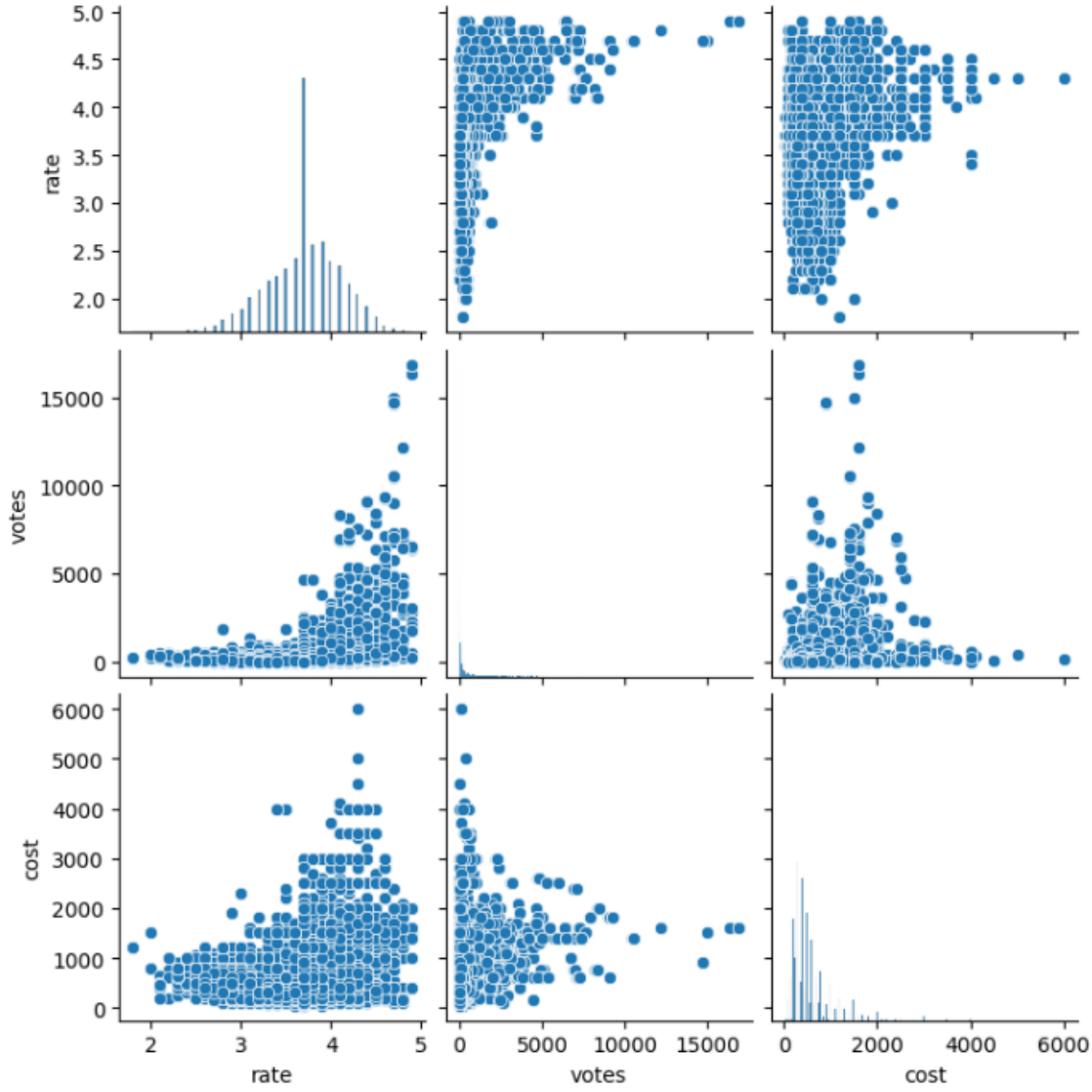
```
Skewness: 2.5825486058125673
```



```
[ ]  # plotting heatmap of numerical columns to check correlation between numerical columns
     sns.heatmap(data=df.corr(),annot=True)
     plt.show()
```

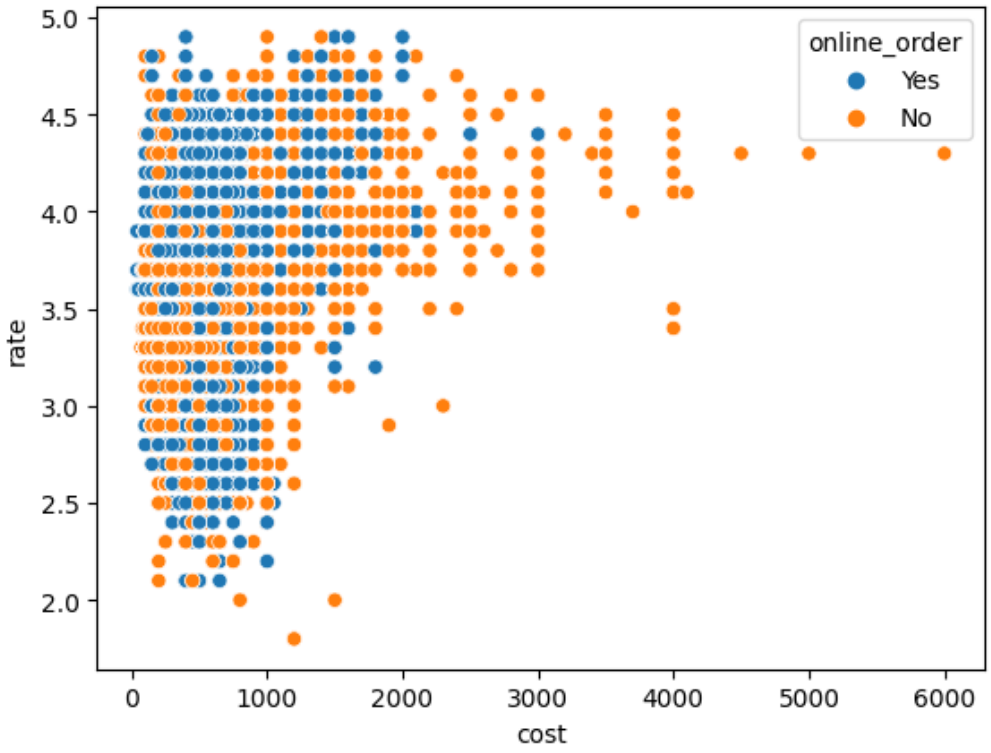


```
# plotting pairplot of numerical columns
sns.pairplot(data=df)
plt.show()
```



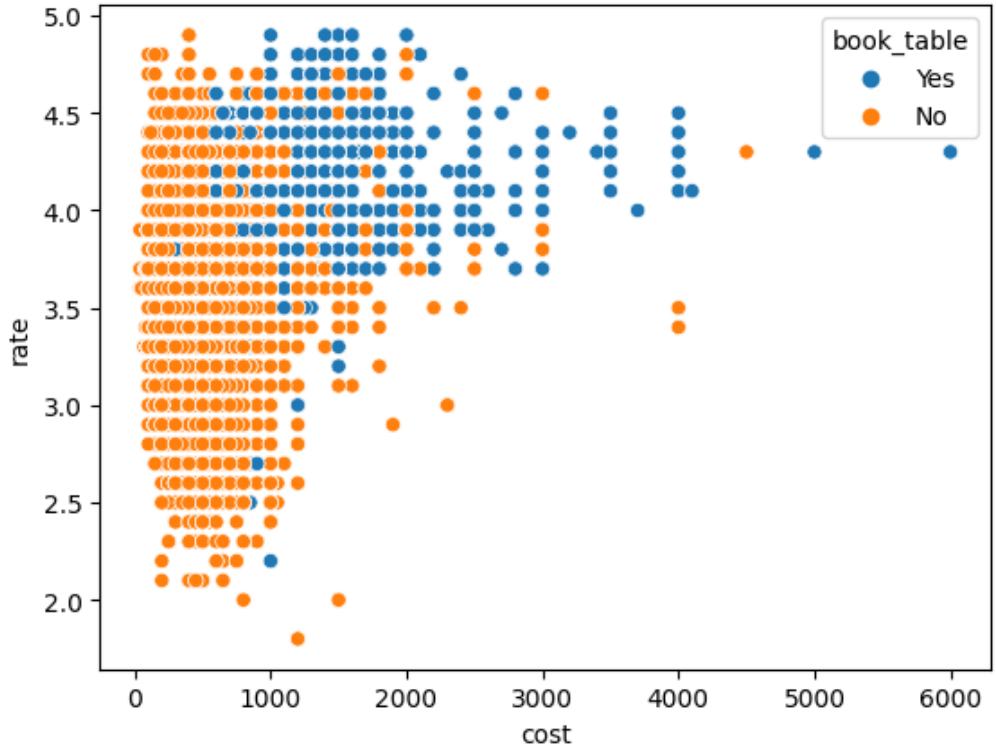
```
# ploting scatter plot between 'cost' and 'rate' columns with 'online_order' column
sns.scatterplot(x='cost', y='rate', hue='online_order', data=df)
```

```
<Axes: xlabel='cost', ylabel='rate'>
```



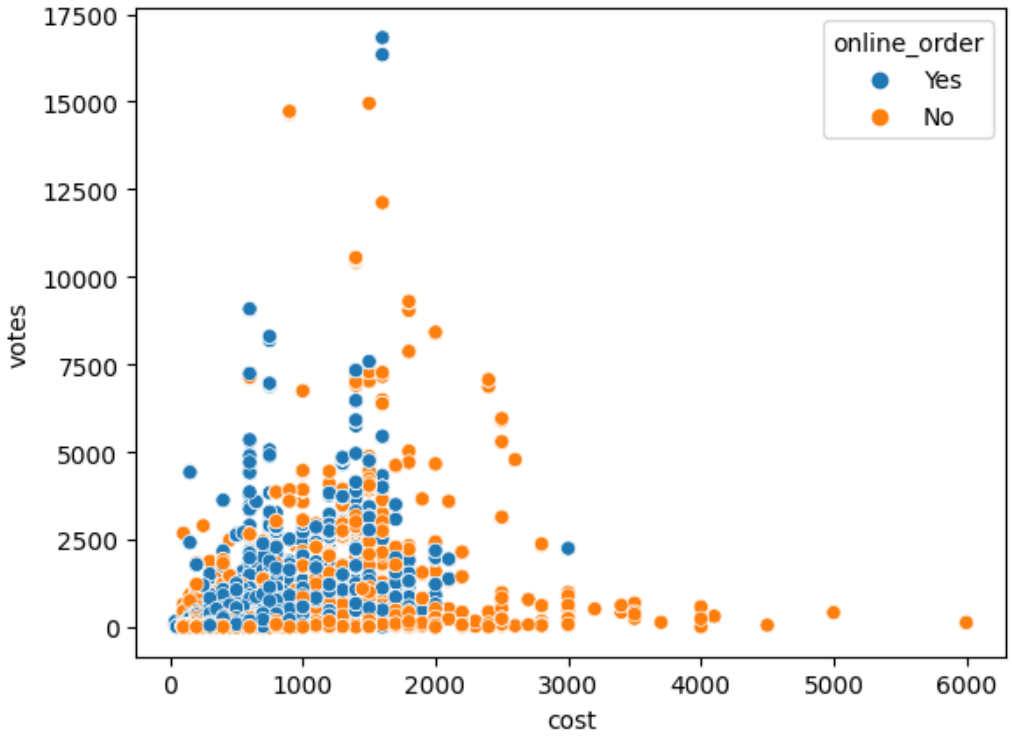
```
# ploting scatter plot between 'cost' and 'rate' columns with 'book_table' column
sns.scatterplot(x='cost', y='rate', hue='book_table', data=df)
```

```
<Axes: xlabel='cost', ylabel='rate'>
```



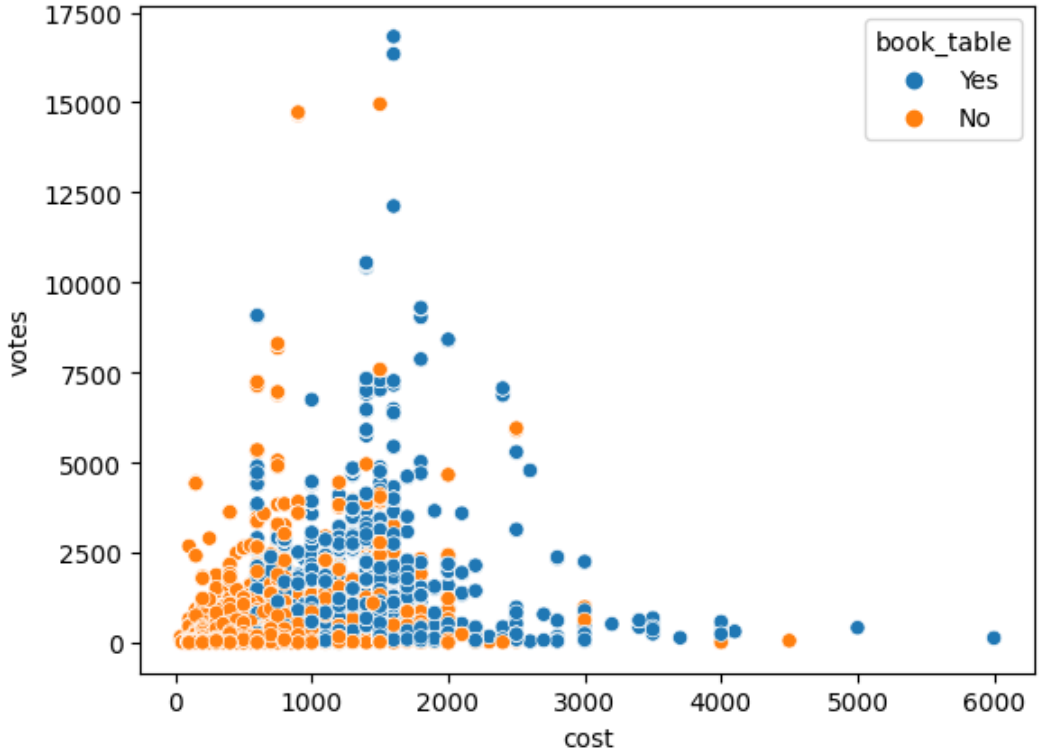
```
# ploting scatter plot between 'cost' and 'votes' columns with 'online_order' column
sns.scatterplot(x='cost', y='votes', hue='online_order', data=df)
```

```
<Axes: xlabel='cost', ylabel='votes'>
```

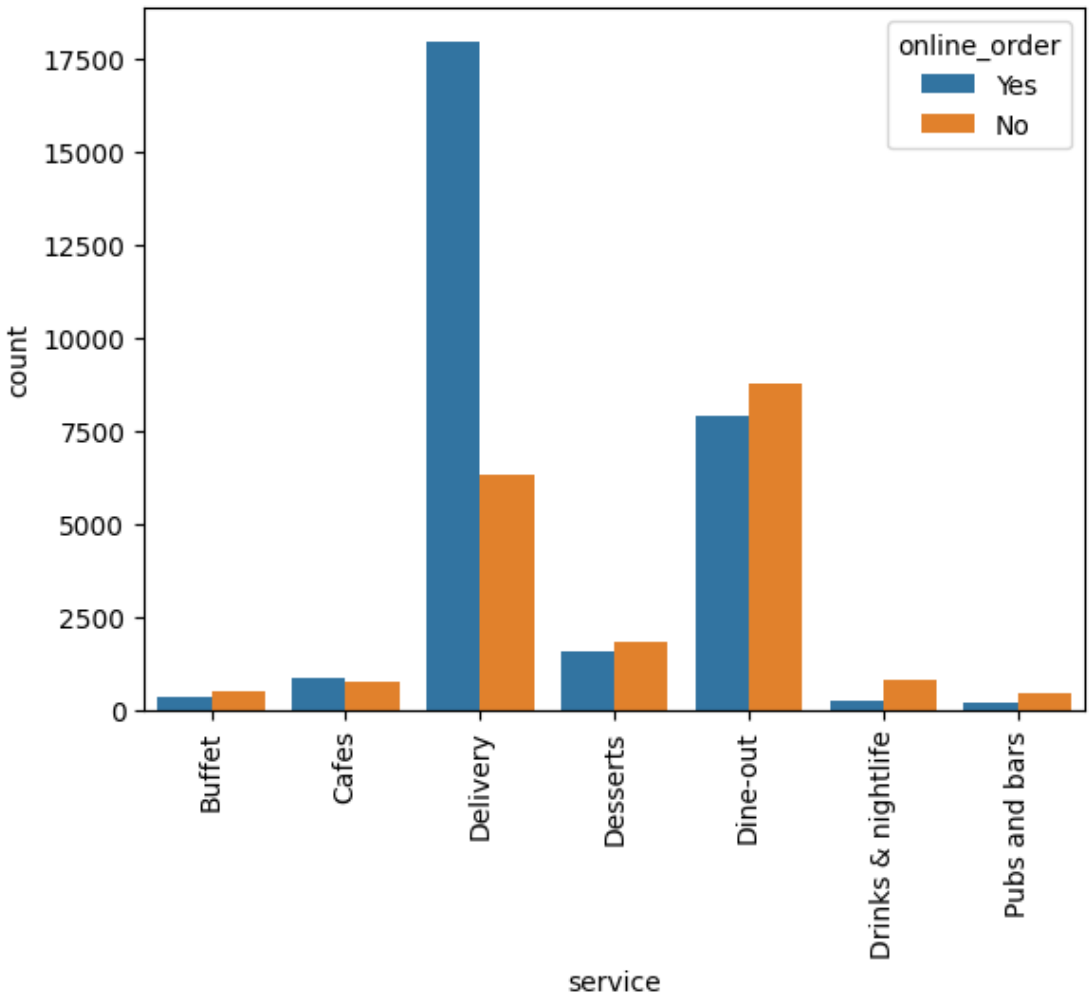


```
# ploting scatter plot between 'cost' and 'votes' columns with 'book_table' column
sns.scatterplot(x='cost', y='votes', hue='book_table', data=df)
```

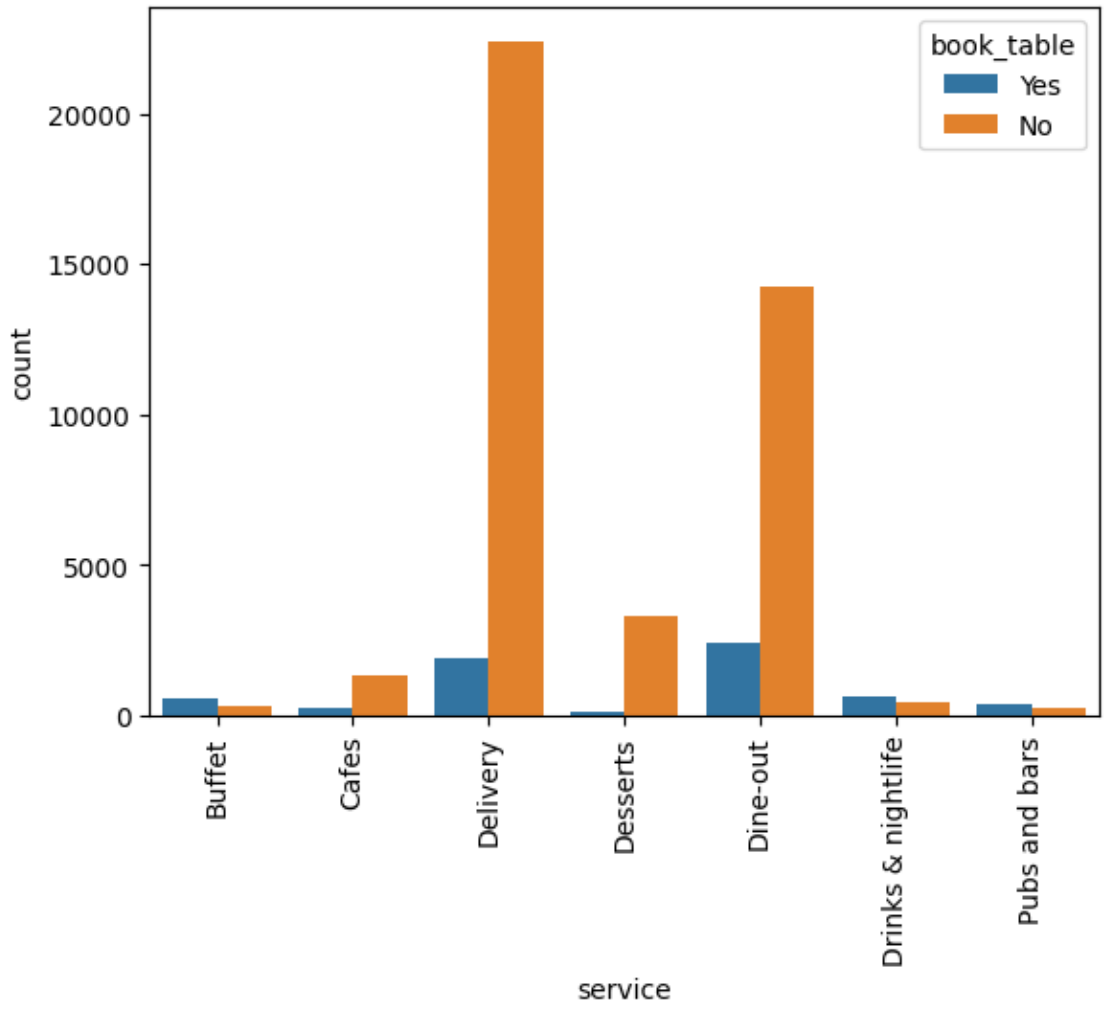
```
<Axes: xlabel='cost', ylabel='votes'>
```



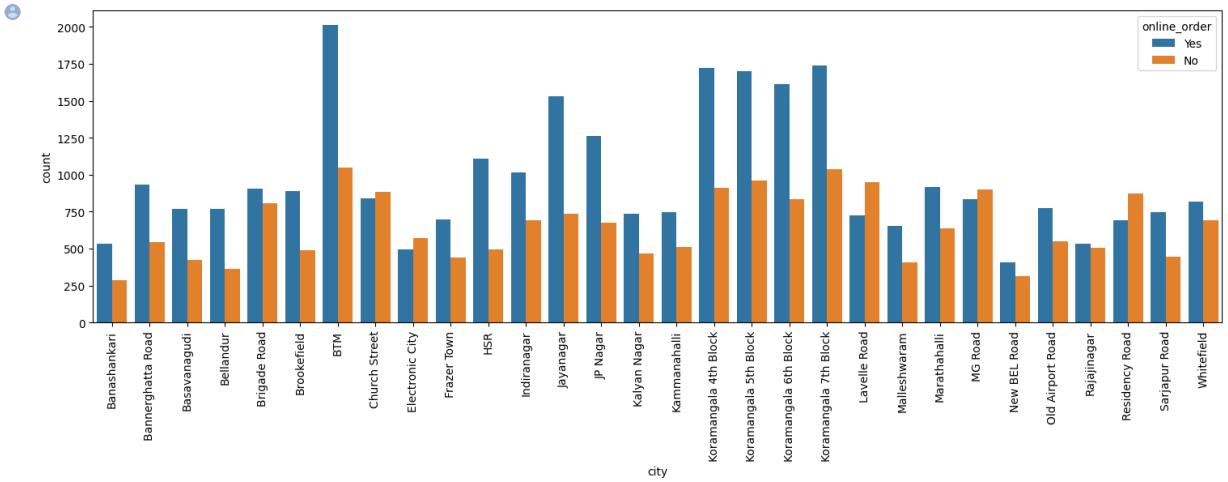
```
# ploting count plot of 'service' column with 'online_order' column
sns.countplot(x=df.service, hue=df.online_order)
plt.xticks(rotation=90)
plt.show()
```



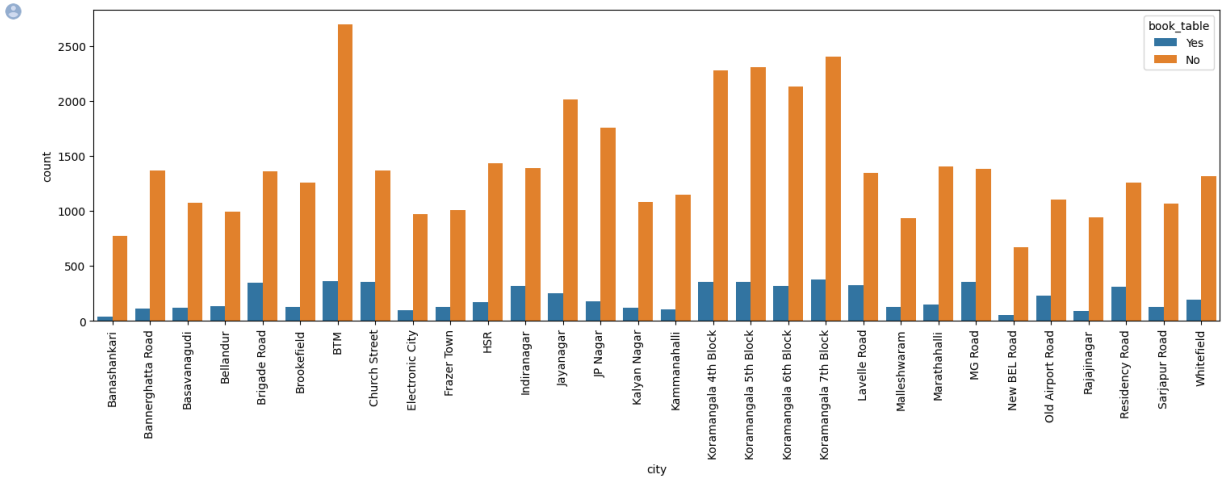
```
# ploting count plot of 'service' column with 'book_table' column
sns.countplot(x=df.service, hue=df.book_table)
plt.xticks(rotation=90)
plt.show()
```



```
# ploting count plot of 'city' column with 'online_order' column
figure=plt.figure(figsize=(18,5))
sns.countplot(x=df.city, hue=df.online_order)
plt.xticks(rotation=90)
plt.show()
```




```
# ploting count plot of 'city' column with 'book_table' column
figure=plt.figure(figsize=(18,5))
sns.countplot(x=df.city, hue=df.book_table)
plt.xticks(rotation=90)
plt.show()
```



PART A

Objective:- To help new and upcoming restaurants by letting them know the various reasons that customers look for and build a model which is able to predict the cost for two people.

Statistical Tests

We have checked the variation inflation factor for each variable in our dataset excluding the target variable i.e. cost, to check if there is any multicollinearity.

Below screenshot shows the VIF factor for each variable.

	VIF_Factor	Features
0	2.370882	cuisines
1	2.345675	rest_type
2	1.737804	book_table
3	1.386206	rate
4	1.376801	votes
5	1.265810	service
6	1.183315	location
7	1.091175	online_order
8	1.019180	city

From the above table we can see that the VIF of all the variables is less than 10 (we set 10 as threshold). So,we can conclude that there is no multicollinearity in the data.

Below screenshot shows the OLS Regression Summary result.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   cost   R-squared:                       0.830
Model:                            OLS   Adj. R-squared:                  0.830
Method:                 Least Squares   F-statistic:                 1.846e+04
Date:                Tue, 09 May 2023   Prob (F-statistic):               0.00
Time:                        15:46:35   Log-Likelihood:                -17871.
No. Observations:               33983   AIC:                         3.576e+04
Df Residuals:                   33973   BIC:                         3.585e+04
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.0001      0.002     -0.055      0.957      -0.004       0.004
online_order    0.0097      0.002      4.168      0.000       0.005       0.014
book_table      0.1107      0.003     37.996      0.000       0.105       0.116
rate            0.0308      0.003     11.783      0.000       0.026       0.036
votes          -0.0078      0.003     -3.038      0.002      -0.013      -0.003
location        0.0724      0.002     30.071      0.000       0.068       0.077
rest_type       0.4398      0.003    128.159      0.000       0.433       0.446
cuisines        0.4248      0.003    123.655      0.000       0.418       0.432
service         0.0076      0.002      3.039      0.002       0.003       0.012
city            0.0132      0.002      5.862      0.000       0.009       0.018
==============================================================================
Omnibus:                    20042.536   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           654592.920
Skew:                           2.286   Prob(JB):                         0.00
Kurtosis:                      24.009   Cond. No.                         3.45
==============================================================================
```

From the above summary we can conclude the below points:

1. Did a hypothesis for Prob(F-statistic), the alternate and null hypothesis are as shown below.

```
#H0: none of the features are useful/significant while making the prediction
#H1: atleast one of the features is useful while making the prediction
```

   a. As Prob(F-statistic) < 0.05, we reject null hypothesis.

   b. Hence, we can infer that at least one of the features is useful/significant while making the prediction

2. Did a hypothesis for p|t| which when pval > 0.05 indicates that feature probability is a junk feature. The alternate and null hypothesis are as shown below.

```
#H0:Feature is not contributing towards the prediction
#H1:Feature is contributing towards the prediction
```

a. As pval < 0.05 for all the independent variables, we reject the null hypothesis.
b. Hence, we can conclude that these features are impacting cost.

3. As Cond.no. < 100, we can say that the data has no multicollinearity.

4. Did a hypothesis to check normality of the data. The alternate and null hypothesis are as shown below.

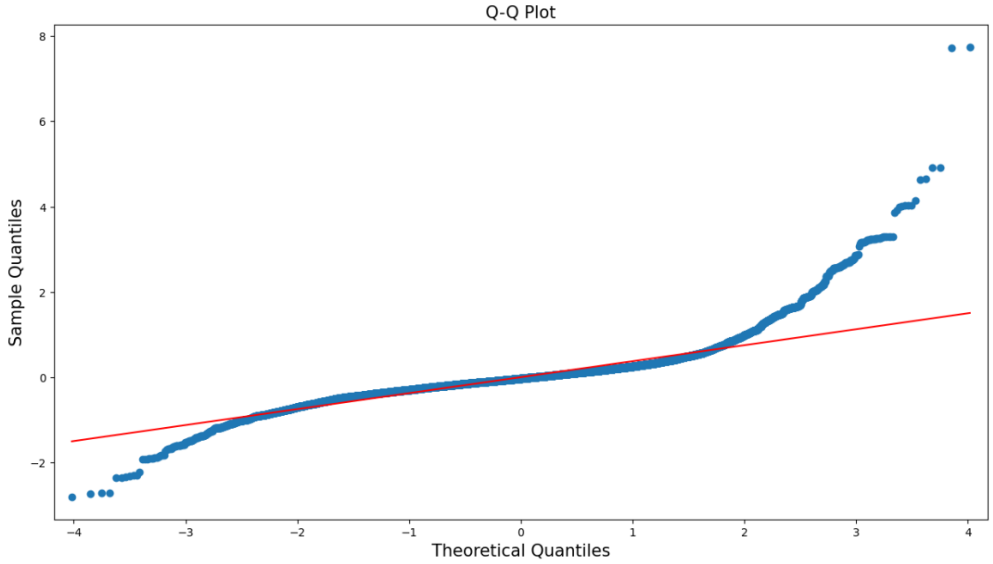
```
#H0:data is normal and kurtosis is mesokurtic
#H1:data is not normal and kurtosis is not mesokurtic
```

a. As Prob(JB) < 0.05, we can say that the data is not normal and kurtosis is not mesokurtic

b. As kurtosis = +ive, we can say that data is leptokurtic.

5. As Durbin-Watson is close to 2,there is no autocorrelation.

6. We have even tested the normality of the data using the qq plot as shown below.



a. The diagonal line (red line) is the regression line and the blue points are the cumulative distribution of the residuals. As some of the points are close to the diagonal line, we conclude that the residuals do not follow a normal distribution

Model Building

We have tried multiple linear regression models to check the accuracies and conclude with one best model out of all of them. Below are the models that we have built.

1. Linear Regression
2. Support Vector Machine
3. Decision Tree
4. Random Forest Extra Tree Regressor
5. KNN Regressor
6. Gradient Boosting
7. Xtreme Gradient Boosting
8. Ada Boosting

We have checked the RMSE values with train and test data with respective R2 score as shown in below screenshot.

Model	RMSE Train Value	RMSE Test Value	R2 SCORE Train	R2 SCORE Test
Ada Boosting regressor	248.01	252.67	67.77	67.28
Decision Tree Regressor	209.14	216.06	77.08	76.07
Gradient Boosting Regressor	174.97	178.41	83.96	83.68
KNN regressor	115.36	149.64	93.03	88.52
Linear regression	213.90	217.10	76.02	75.84
Random Forest Extra Trees Regressor	32.96	82.51	99.43	96.51
Support Vector Machine	246.36	250.42	68.20	67.86
Xtreme Gradient Boosting	128.86	138.44	91.30	90.18

From the above table we have observed there was overfitting and high multicollinearity occurring. Hence we have changed a bit for EDA and the issue is solved as shown in the below screenshot.

Below screenshot gives all the RMSE values that were achieved with train and test data set with respective R2 score as well.

Model	RMSE Train Value	RMSE Test Value	R2 SCORE Train	R2 SCORE Test
Ada Boosting regressor	0.59	0.60	64.51	65.06
Decision Tree Regressor	0.48	0.48	76.87	77.83
Gradient Boosting Regressor	0.41	0.41	82.65	83.32
KNN regressor	0.25	0.33	93.49	89.60
Linear regression	0.41	0.41	83.02	83.85
Random Forest Extra Trees Regressor	0.09	0.19	99.16	96.40
Support Vector Machine	0.44	0.44	80.41	81.21
Xtreme Gradient Boosting	0.29	0.30	91.36	91.12

PART B

Objective:-The aim is to classify the orders that have been ordered online and offline.And identify the patterns that lead to order online as well as offline orders.

Step 1:- We have done the required data processing and cleaning.

Step 2:- Splitting the data into train data and test data.

```
X_train_norm (33983, 9)
y_train (33983,)
X_test_norm (14565, 9)
y_test (14565,)
```

Step 3:- Building a KNN model on a training dataset using hamming distance

```
[ ]  knn_classification = KNeighborsClassifier(n_neighbors = 3, metric='hamming')

[ ]  knn_model = knn_classification.fit(X_train_norm, y_train)
```

Step 4:- Calculating performance measures

```
#Calculate performance measures on the test set.
test_report = get_test_report(knn_model, test_data = X_test_norm)

# print the performace measures
print(test_report)
```

```
              precision    recall  f1-score   support

           0       0.82      0.84      0.83      5871
           1       0.89      0.88      0.88      8694

    accuracy                           0.86     14565
   macro avg       0.86      0.86      0.86     14565
weighted avg       0.86      0.86      0.86     14565
```

This shows that our KNN model has 86% accuracy.

Building model based on Naive-Bayes Theorem:-

```
# compute the performance measures on test data
# call the function 'get_test_report'
# pass the gaussian naive bayes model to the function
test_report = get_test_report(gnb_model, test_data=X_test_norm)

# print the performace measures
print(test_report)
```

```
              precision    recall  f1-score   support

           0       0.57      0.23      0.32      5871
           1       0.63      0.89      0.74      8694

    accuracy                           0.62     14565
   macro avg       0.60      0.56      0.53     14565
weighted avg       0.61      0.62      0.57     14565
```

In the naive-bayes classifier, we got 62% accuracy.

Hence,we can conclude that KNN Classifier gives better accuracy for the given dataset as compared to Naive-Bayes Classifier.

REFERENCES

- https://www.w3schools.com/python/
- https://www.geeksforgeeks.org/python-programming-examples/