iNeuron

Project Title -> Restaurant Rating Prediction

Technologies -> Machine Learning Technology

Domain -> E-commerce

Project Difficulties -> level Intermediate

A look into the data

```
#Importing Libraries
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
```

```
zomato_real=pd.read_csv("/kaggle/input/zomato/zomato.csv")
zomato_real.head()
```

	url	address	name	online_order	bo
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	

```
zomato_real.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
```

```
zomato=zomato_real.drop(['url','dish_liked','phone'],axis=1)
```

```
zomato.head()
```

```
         address     name  online_order  book_table  rate  votes      location  rest_t
```

```
d=zomato.duplicated().sum()
zomato.drop_duplicates(inplace=True)
d
```

```
    0
```

```
          2nd Floor, 80
           Feet Road      Spice                                                    Ca
```

```
n=zomato.isnull().sum()
zomato.dropna(how='any',inplace=True)
zomato.info()
n
```

```
    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 43499 entries, 0 to 51716
    Data columns (total 14 columns):
     #   Column                       Non-Null Count  Dtype 
    ---  ------                       --------------  ----- 
     0   address                      43499 non-null  object
     1   name                         43499 non-null  object
     2   online_order                 43499 non-null  object
     3   book_table                   43499 non-null  object
     4   rate                         43499 non-null  object
     5   votes                        43499 non-null  int64 
     6   location                     43499 non-null  object
     7   rest_type                    43499 non-null  object
     8   cuisines                     43499 non-null  object
     9   approx_cost(for two people)  43499 non-null  object
     10  reviews_list                 43499 non-null  object
     11  menu_item                    43499 non-null  object
     12  listed_in(type)              43499 non-null  object
     13  listed_in(city)              43499 non-null  object
    dtypes: int64(1), object(13)
    memory usage: 5.0+ MB
    address                        0
    name                           0
    online_order                   0
    book_table                     0
    rate                           0
    votes                          0
    location                       0
    rest_type                      0
    cuisines                       0
    approx_cost(for two people)    0
    reviews_list                   0
    menu_item                      0
    listed_in(type)                0
    listed_in(city)                0
    dtype: int64
```

```
zomato.columns
```

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

```
zomato = zomato.rename(columns={'approx_cost(for two people)':'cost','listed_in(type)':'ty
                                  'listed_in(city)':'city'})
zomato.columns
```

```
Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'location', 'rest_type', 'cuisines', 'cost', 'reviews_list',
       'menu_item', 'type', 'city'],
      dtype='object')
```

```
zomato['cost'] = zomato['cost'].astype(str) #Changing the cost to string
zomato['cost'] = zomato['cost'].apply(lambda x: x.replace(',','.')) #Using lambda function
zomato['cost'] = zomato['cost'].astype(float) # Changing the cost to Float
zomato.info() # looking at the dataset information after transformation
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 43499 entries, 0 to 51716
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   address       43499 non-null  object 
 1   name          43499 non-null  object 
 2   online_order  43499 non-null  object 
 3   book_table    43499 non-null  object 
 4   rate          43499 non-null  object 
 5   votes         43499 non-null  int64  
 6   location      43499 non-null  object 
 7   rest_type     43499 non-null  object 
 8   cuisines      43499 non-null  object 
 9   cost          43499 non-null  float64
 10  reviews_list  43499 non-null  object 
 11  menu_item     43499 non-null  object 
 12  type          43499 non-null  object 
 13  city          43499 non-null  object 
dtypes: float64(1), int64(1), object(12)
memory usage: 5.0+ MB
```

```
#Reading unique values from the Rate column
u=zomato['rate'].unique()
u
```

```
array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
       '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
       '4.3/5', 'NEW', '2.9/5', '3.5/5', '2.6/5', '3.8 /5', '3.4/5',
       '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5',
       '3.4 /5', '-', '3.6 /5', '4.8/5', '3.9 /5', '4.2 /5', '4.0 /5',
       '4.1 /5', '3.7 /5', '3.1 /5', '2.9 /5', '3.3 /5', '2.8 /5',
       '3.5 /5', '2.7 /5', '2.5 /5', '3.2 /5', '2.6 /5', '4.5 /5',
       '4.3 /5', '4.4 /5', '4.9/5', '2.1/5', '2.0/5', '1.8/5', '4.6 /5',
       '4.9 /5', '3.0 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5',
       '2.1 /5', '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)
```

Modifying and Encoding

```
#Removing '/5' from Rates
zomato = zomato.loc[zomato.rate !='NEW']
zomato = zomato.loc[zomato.rate !='-'].reset_index(drop=True)
remove_slash = lambda x: x.replace('/5', '') if type(x) == np.str else x
#zomato.rate = zomato.rate.apply(remove_slash).str.strip().astype('float')
zomato['rate'].head() # looking at the dataset after transformation
```

```
0    4.1
1    4.1
2    3.8
3    3.7
4    3.8
Name: rate, dtype: float64
```

```
# Adjust the column names
zomato.name = zomato.name.apply(lambda x:x.title())
zomato.online_order.replace(('Yes','No'),(True, False),inplace=True)
zomato.book_table.replace(('Yes','No'),(True, False),inplace=True)
zomato.head() # looking at the dataset after transformation
```

	address	name	online_order	book_table	rate	votes	location	rest_t
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	True	True	4.1	775	Banashankari	Ca Di
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	True	False	4.1	787	Banashankari	Ca Di
2	1112, Next to KIMS Medical	San Churro	True	False	3.8	918	Banashankari	C Ca

```
c=zomato.cost.unique() # cheking the unique costs
c
```

```
array([800.  , 300.  , 600.  , 700.  , 550.  , 500.  , 450.  , 650.  ,
       400.  , 900.  , 200.  , 750.  , 150.  , 850.  , 100.  ,   1.2 ,
       350.  , 250.  , 950.  ,   1.  ,   1.5 ,   1.3 , 199.  ,   1.1 ,
         1.6 , 230.  , 130.  ,   1.7 ,   1.35,   2.2 ,   1.4 ,   2.  ,
         1.8 ,   1.9 , 180.  , 330.  ,   2.5 ,   2.1 ,   3.  ,   2.8 ,
         3.4 ,  50.  ,  40.  ,   1.25,   3.5 ,   4.  ,   2.4 ,   2.6 ,
         1.45,  70.  ,   3.2 , 240.  ,   6.  ,   1.05,   2.3 ,   4.1 ,
       120.  ,   5.  ,   3.7 ,   1.65,   2.7 ,   4.5 ,  80.  ])
```

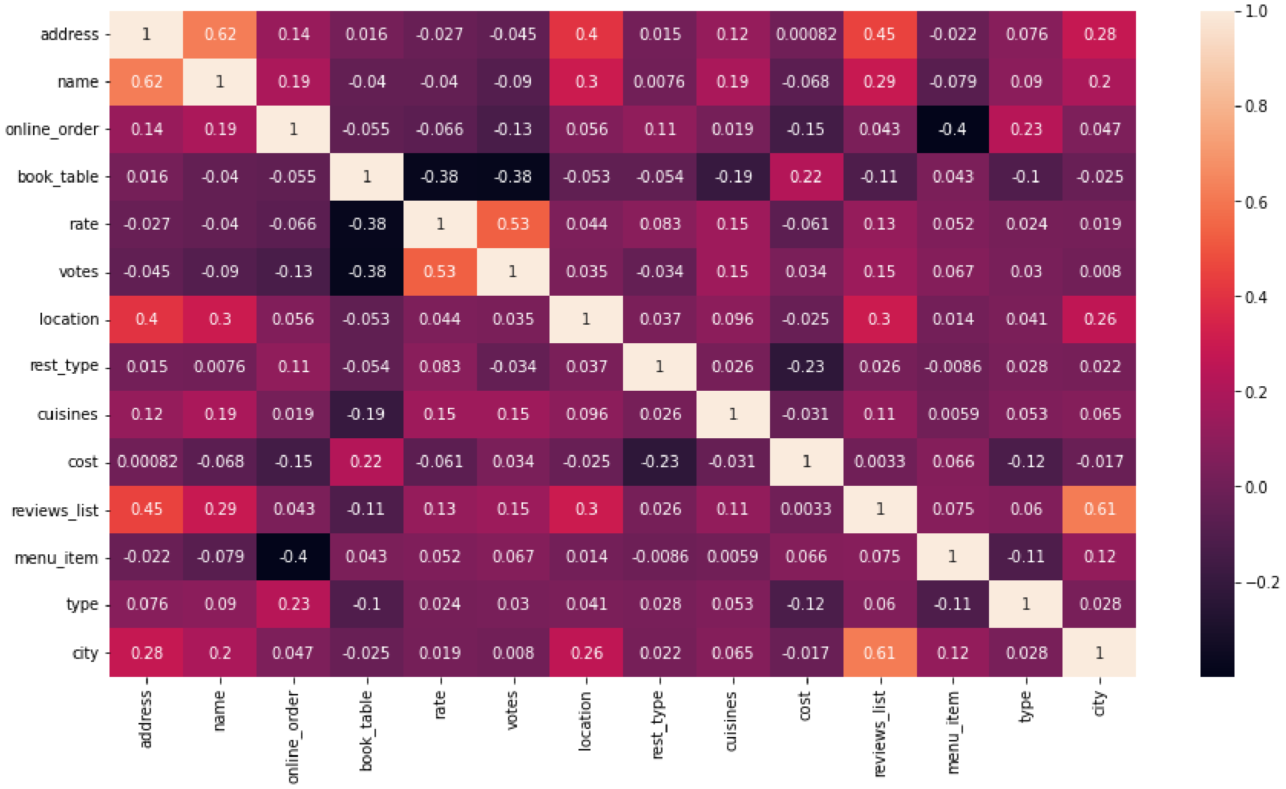
```
#Encode the input Variables
def Encode(zomato):
    for column in zomato.columns[~zomato.columns.isin(['rate', 'cost', 'votes'])]:
        zomato[column] = zomato[column].factorize()[0]
    return zomato

zomato_en = Encode(zomato.copy())
zomato_en.head() # looking at the dataset after transformation
```

	address	name	online_order	book_table	rate	votes	location	rest_type	cuisine
0	0	0	0	0	4.1	775	0	0	
1	1	1	0	1	4.1	787	0	0	
2	2	2	0	1	3.8	918	0	1	
3	3	3	1	1	3.7	88	0	2	
4	4	4	1	1	3.8	166	1	0	

```
#Get Correlation between different variables
corr = zomato_en.corr(method='kendall')
plt.figure(figsize=(15,8))
sns.heatmap(corr, annot=True)
zomato_en.columns
```

```
Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'location', 'rest_type', 'cuisines', 'cost', 'reviews_list',
       'menu_item', 'type', 'city'],
      dtype='object')
```



	address	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost	reviews_list	menu_item	type	city
address	1	0.62	0.14	0.016	-0.027	-0.045	0.4	0.015	0.12	0.00082	0.45	-0.022	0.076	0.28
name	0.62	1	0.19	-0.04	-0.04	-0.09	0.3	0.0076	0.19	-0.068	0.29	-0.079	0.09	0.2
online_order	0.14	0.19	1	-0.055	-0.066	-0.13	0.056	0.11	0.019	-0.15	0.043	-0.4	0.23	0.047
book_table	0.016	-0.04	-0.055	1	-0.38	-0.38	-0.053	-0.054	-0.19	0.22	-0.11	0.043	-0.1	-0.025
rate	-0.027	-0.04	-0.066	-0.38	1	0.53	0.044	0.083	0.15	-0.061	0.13	0.052	0.024	0.019
votes	-0.045	-0.09	-0.13	-0.38	0.53	1	0.035	-0.034	0.15	0.034	0.15	0.067	0.03	0.008
location	0.4	0.3	0.056	-0.053	0.044	0.035	1	0.037	0.096	-0.025	0.3	0.014	0.041	0.26
rest_type	0.015	0.0076	0.11	-0.054	0.083	-0.034	0.037	1	0.026	-0.23	0.026	-0.0086	0.028	0.022
cuisines	0.12	0.19	0.019	-0.19	0.15	0.15	0.096	0.026	1	-0.031	0.11	0.0059	0.053	0.065
cost	0.00082	-0.068	-0.15	0.22	-0.061	0.034	-0.025	-0.23	-0.031	1	0.0033	0.066	-0.12	-0.017
reviews_list	0.45	0.29	0.043	-0.11	0.13	0.15	0.3	0.026	0.11	0.0033	1	0.075	0.06	0.61
menu_item	-0.022	-0.079	-0.4	0.043	0.052	0.067	0.014	-0.0086	0.0059	0.066	0.075	1	-0.11	0.12
type	0.076	0.09	0.23	-0.1	0.024	0.03	0.041	0.028	0.053	-0.12	0.06	-0.11	1	0.028
city	0.28	0.2	0.047	-0.025	0.019	0.008	0.26	0.022	0.065	-0.017	0.61	0.12	0.028	1

The highest correlation is between name and address which is 0.62 which is not of very much concern

Splitting the Dataset into train and test data used for modelling later

```
#Defining the independent variables and dependent variables
x = zomato_en.iloc[:,[2,3,5,6,7,8,9,11]]
y = zomato_en['rate']
#Getting Test and Training Set
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.1,random_state=353)
x_train.head()
```

	online_order	book_table	votes	location	rest_type	cuisines	cost	menu_it
16950	0	1	0	8	2	5	250.0	
767	0	1	131	8	4	278	400.0	1
6750	0	1	137	45	2	1295	250.0	
9471	0	1	74	16	0	537	1.0	
25162	0	1	61	12	2	1860	350.0	

```
y_train.head()
```

```
16950    3.9
767      3.7
6750     4.0
9471     3.8
25162    3.7
Name: rate, dtype: float64
```

```
zomato_en['menu_item'].unique() # seeing the unique values in 'menu_item'
```

```
array([   0,    1,    2, ..., 8240, 8241, 8242])
```

```
zomato_en['location'].unique() # seeing the unique values in 'location'
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91])
```

```
zomato_en['cuisines'].unique() # seeing the unique values in 'cusines'
```

```
    array([   0,    1,    2, ..., 2364, 2365, 2366])
```

```
zomato_en['rest_type'].unique() # seeing the unique values in 'rest_type'
```

```
    array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
           17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
           34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
           51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
           68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
           85, 86])
```

```
x.head()
```

	online_order	book_table	votes	location	rest_type	cuisines	cost	menu_item
0	0	0	775	0	0	0	800.0	0
1	0	1	787	0	0	1	800.0	0
2	0	1	918	0	1	2	800.0	0
3	1	1	88	0	2	3	300.0	0
4	1	1	166	1	0	4	600.0	0

```
y.head()
```

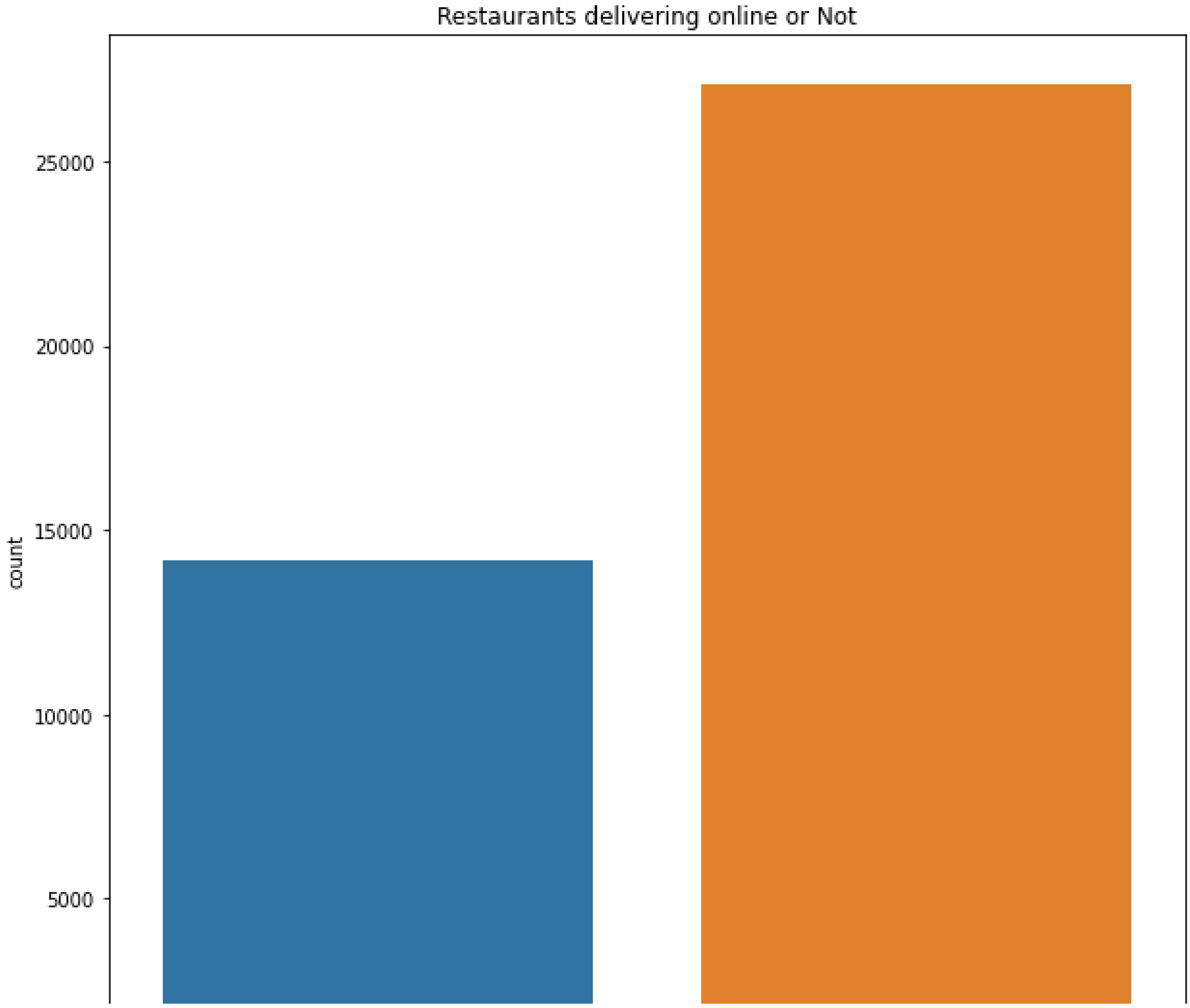
```
    0    4.1
    1    4.1
    2    3.8
    3    3.7
    4    3.8
    Name: rate, dtype: float64
```

Data Visualization

Restaurants delivering Online or not

```
#Restaurants delivering Online or not
sns.countplot(zomato['online_order'])
fig = plt.gcf()
fig.set_size_inches(10,10)
plt.title('Restaurants delivering online or Not')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
Text(0.5, 1.0, 'Restaurants delivering online or Not')
```



```
import plotly.graph_objs as go
df2 = zomato['online_order'].value_counts()
colors = ['#FEBFB3', '#E1396C']

trace = go.Pie(labels=df2.index, values=df2.values, textinfo="value",
              marker=dict(colors=colors, line=dict(width=2)))
layout = go.Layout(title='Accepting vs not accepting online orders', width=500, height=500
fig = go.Figure(data=[trace], layout=layout)
fig.show()
```

Restaurants allowing table booking or not

```
sns.countplot(zomato['book_table'])
fig = plt.gcf()
fig.set_size_inches(10,10)
plt.title('Restaurants allowing table booking or not')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
Text(0.5, 1.0, 'Restaurants allowing table booking or not')
```

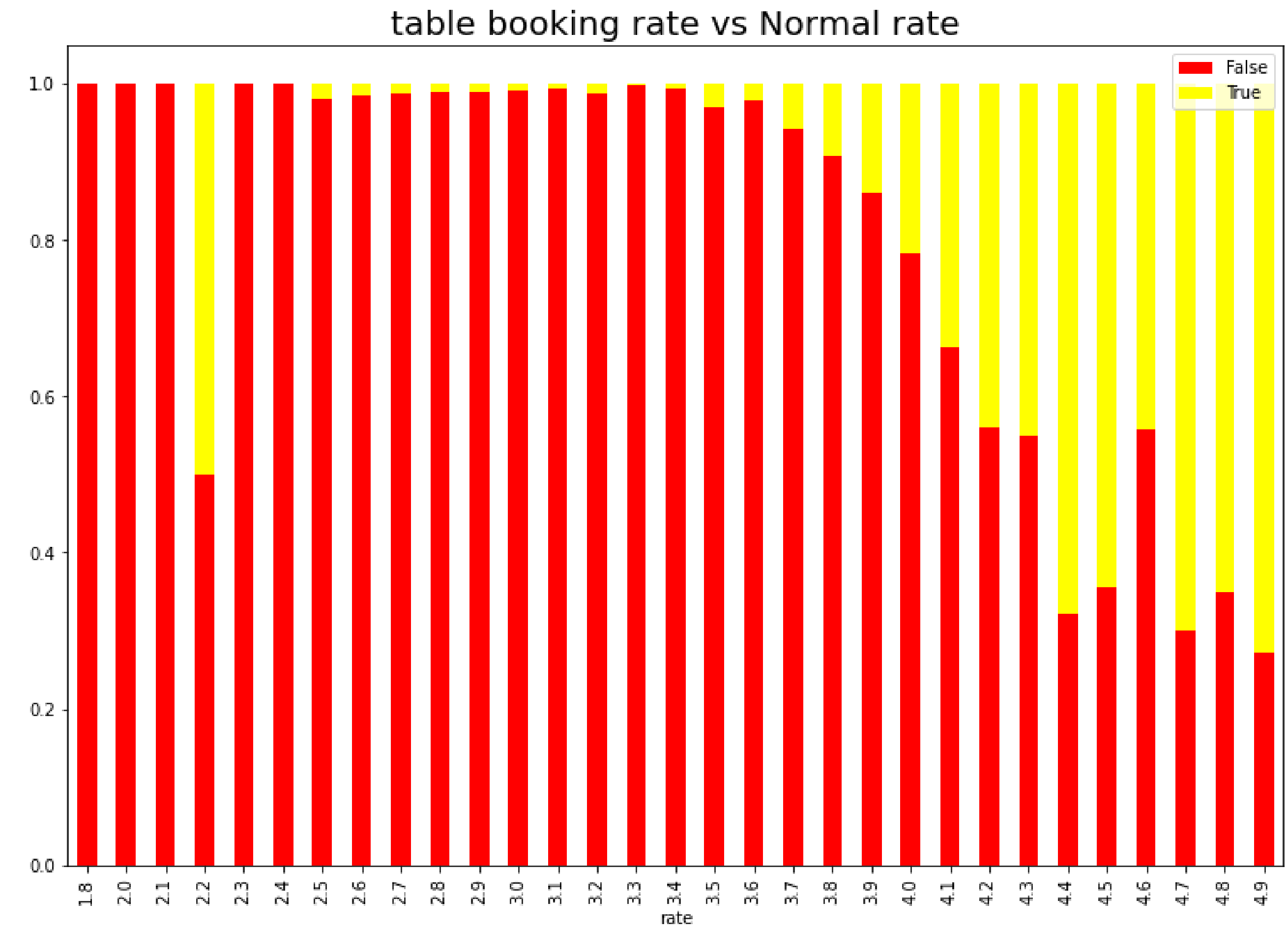
Restaurants allowing table booking or not

```
df3 = zomato['book_table'].value_counts()
colors = ['#96D38C', '#D0F9B1']

trace = go.Pie(labels=df3.index, values=df3.values, textinfo="value",
              marker=dict(colors=colors, line=dict(width=2)))
layout = go.Layout(title='Accepting vs not accepting table bookings', width=500, height=50
fig = go.Figure(data=[trace], layout=layout)
fig.show()
```

Table booking Rate vs Normal Rate

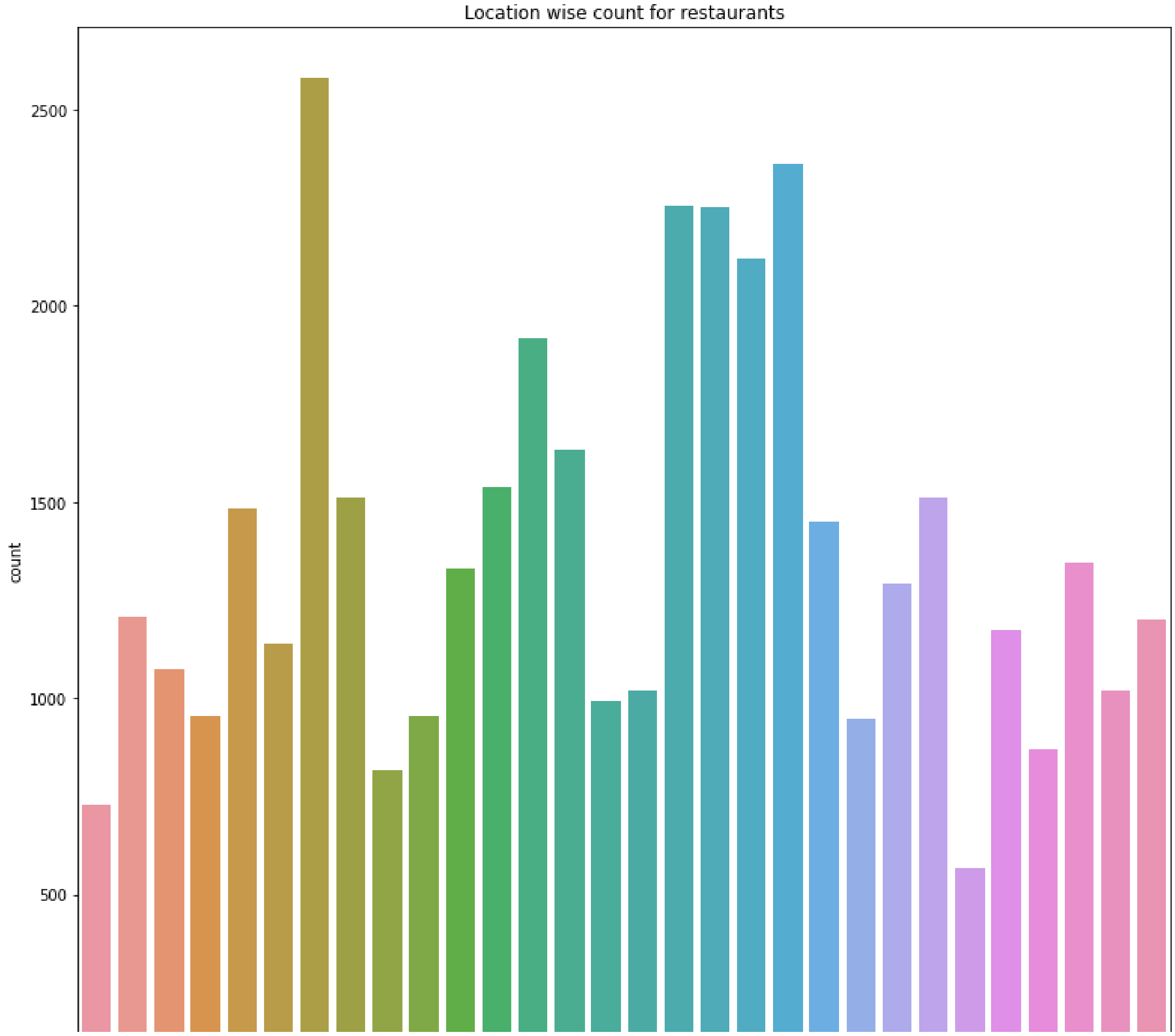
```
plt.rcParams['figure.figsize'] = (13, 9)
Y = pd.crosstab(zomato['rate'], zomato['book_table'])
Y.div(Y.sum(1).astype(float), axis = 0).plot(kind = 'bar', stacked = True,color=['red','ye
plt.title('table booking rate vs Normal rate', fontweight = 30, fontsize = 20)
plt.legend(loc="upper right")
plt.show()
```



Location

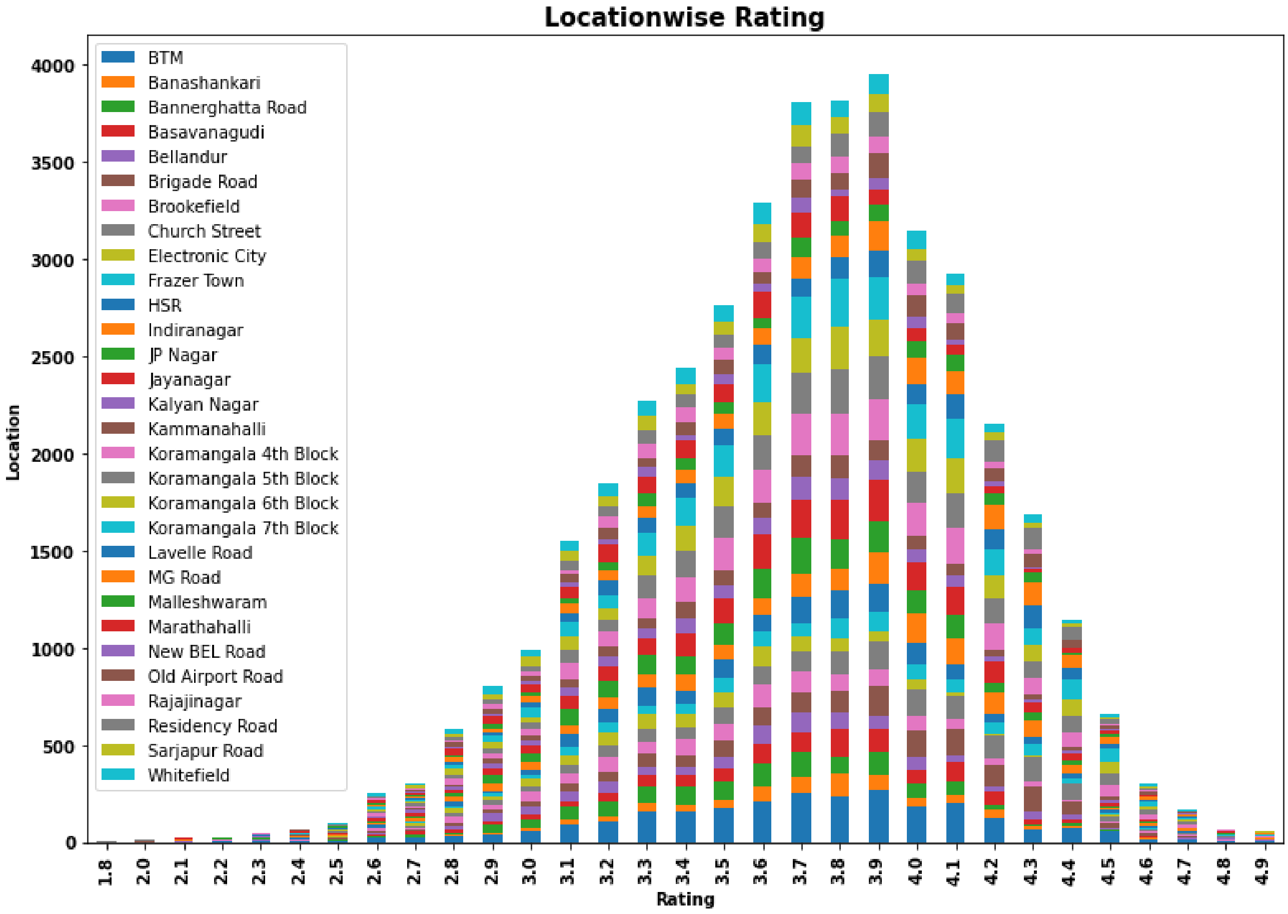
```
sns.countplot(zomato['city'])
sns.countplot(zomato['city']).set_xticklabels(sns.countplot(zomato['city']).get_xticklabel
fig = plt.gcf()
fig.set_size_inches(13,13)
plt.title('Location wise count for restaurants')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
Text(0.5, 1.0, 'Location wise count for restaurants')
```



Location and Rating

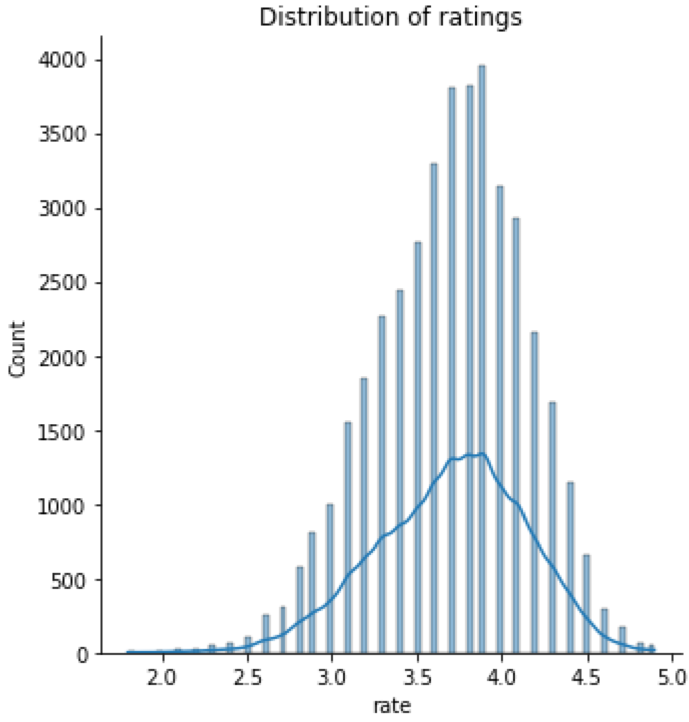
```
loc_plt=pd.crosstab(zomato['rate'],zomato['city'])
loc_plt.plot(kind='bar',stacked=True);
plt.title('Locationwise Rating',fontsize=15,fontweight='bold')
plt.ylabel('Location',fontsize=10,fontweight='bold')
plt.xlabel('Rating',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
plt.legend();
```



distribution of rating

```
plt.figure(figsize=(7, 5))
sns.displot(zomato['rate'], kde=True)
plt.title('Distribution of ratings')
plt.show()
```

```
<Figure size 504x360 with 0 Axes>
```

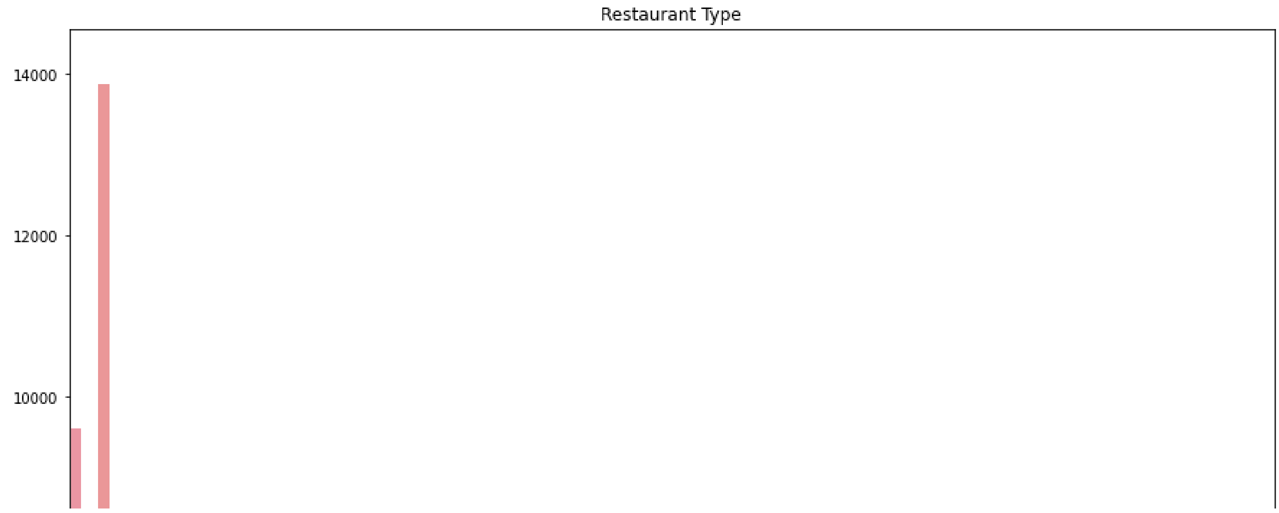


most of the restaurants have rating between 3.5 and 4.0

Restaurant Type

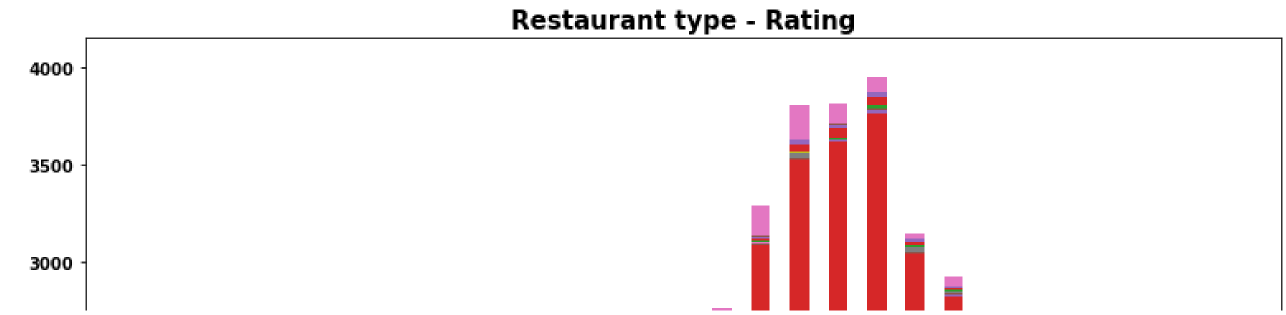
```
sns.countplot(zomato['rest_type'])
sns.countplot(zomato['rest_type']).set_xticklabels(sns.countplot(zomato['rest_type']).get_
fig = plt.gcf()
fig.set_size_inches(15,15)
plt.title('Restaurant Type')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
Text(0.5, 1.0, 'Restaurant Type')
```



Gaussian Rest type and Rating

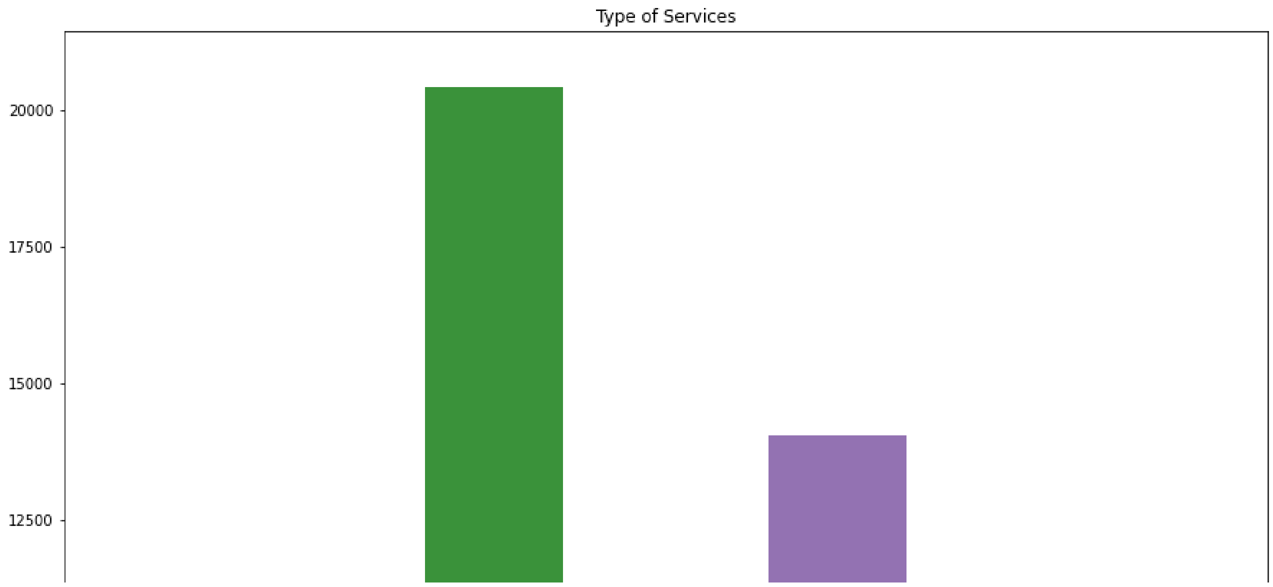
```
loc_plt=pd.crosstab(zomato['rate'],zomato['rest_type'])
loc_plt.plot(kind='bar',stacked=True);
plt.title('Restaurant type - Rating',fontsize=15,fontweight='bold')
plt.ylabel('Restaurant type',fontsize=10,fontweight='bold')
plt.xlabel('Rating',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
plt.legend().remove();
```



Types of Services

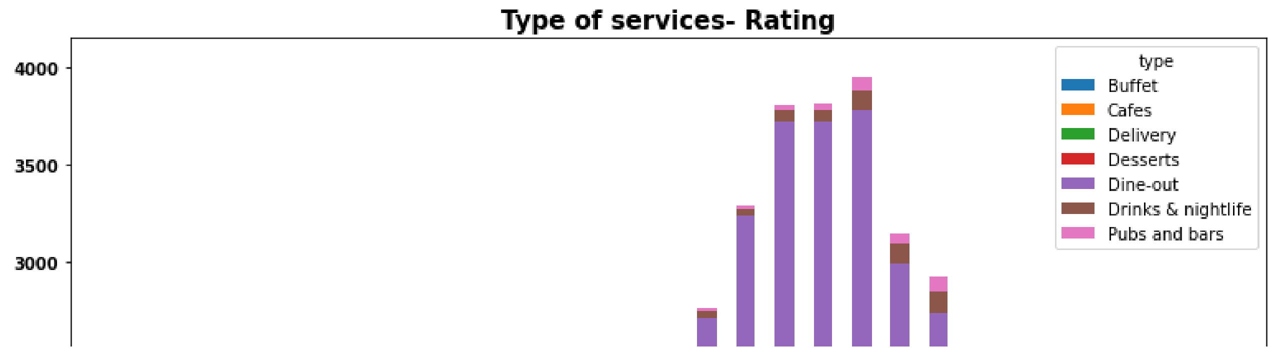
```
sns.countplot(zomato['type'])
sns.countplot(zomato['type']).set_xticklabels(sns.countplot(zomato['type']).get_xticklabel
fig = plt.gcf()
fig.set_size_inches(15,15)
plt.title('Type of Services')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
Text(0.5, 1.0, 'Type of Services')
```



Type and Rating

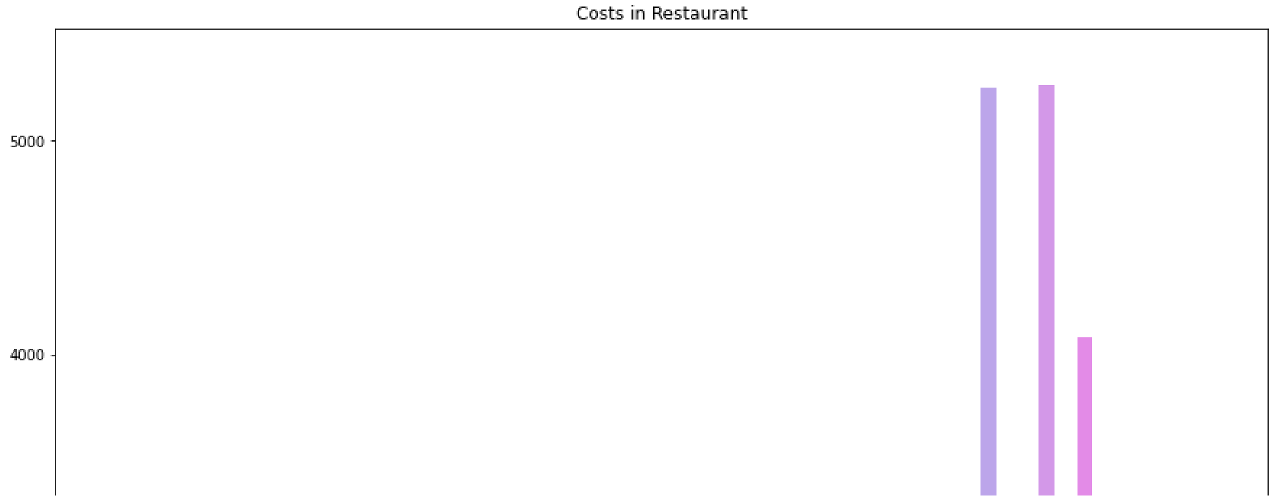
```
type_plt=pd.crosstab(zomato['rate'],zomato['type'])
type_plt.plot(kind='bar',stacked=True);
plt.title('Type of services- Rating',fontsize=15,fontweight='bold')
plt.ylabel('Type of services',fontsize=10,fontweight='bold')
plt.xlabel('Rating',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
```



Cost in Restaurant

```
sns.countplot(zomato['cost'])
sns.countplot(zomato['cost']).set_xticklabels(sns.countplot(zomato['cost']).get_xticklabel
fig = plt.gcf()
fig.set_size_inches(15,15)
plt.title('Costs in Restaurant')
```

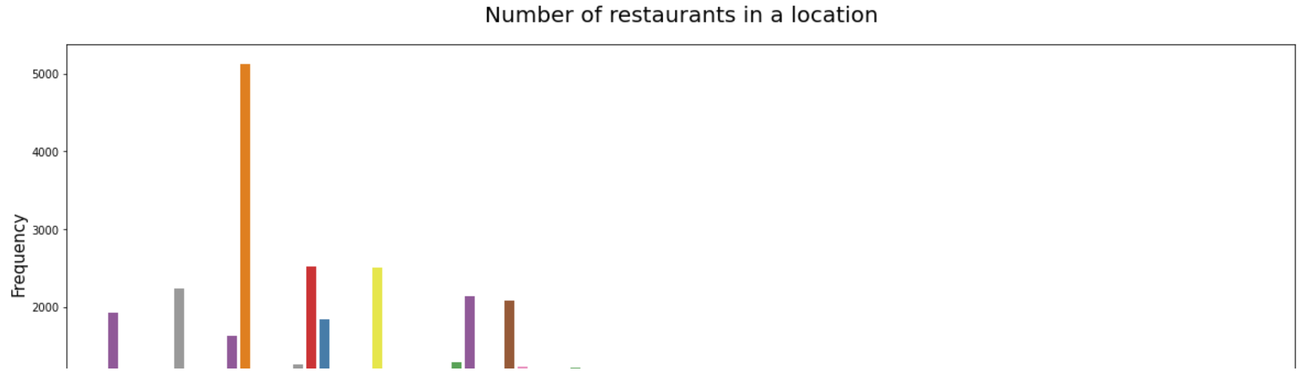
```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
Text(0.5, 1.0, 'Costs in Restaurant')
```



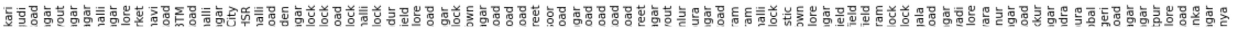
No. of Restaurants in a Location

```
fig = plt.figure(figsize=(20,7))
loc = sns.countplot(x="location",data=zomato_real, palette = "Set1")
loc.set_xticklabels(loc.get_xticklabels(), rotation=90, ha="right")
plt.ylabel("Frequency",size=15)
plt.xlabel("Location",size=18)
loc
plt.title('Number of restaurants in a location',size = 20,pad=20)
```

```
Text(0.5, 1.0, 'Number of restaurants in a location')
```

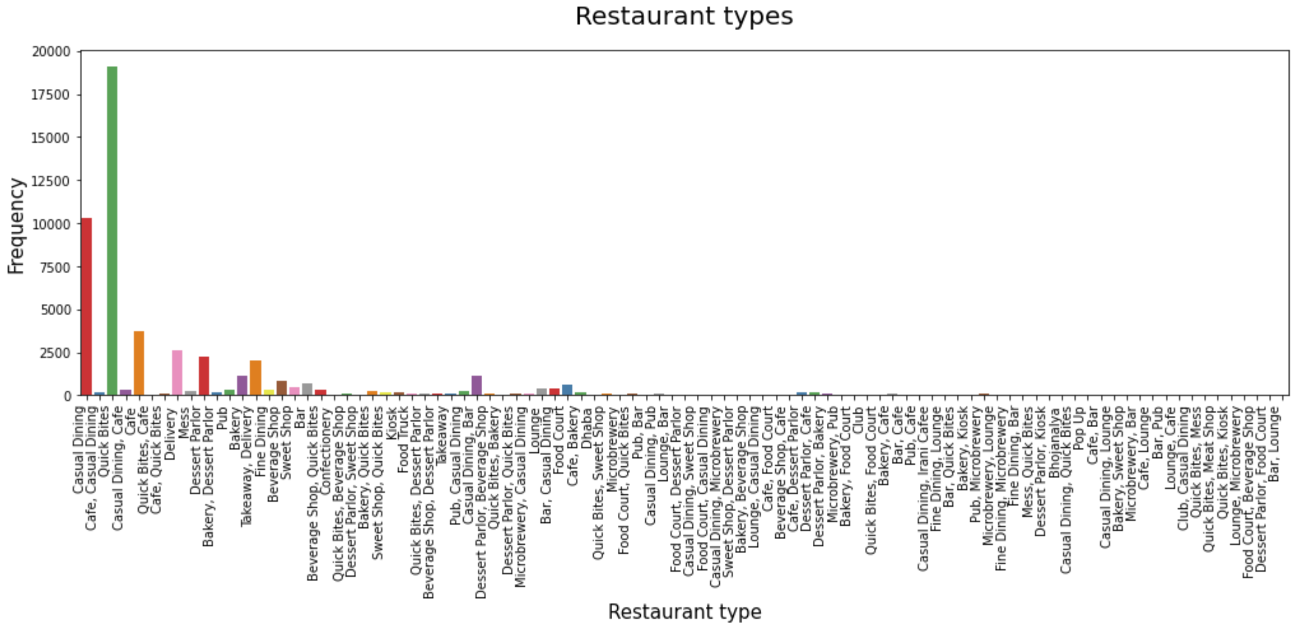


Restaurant type



```
fig = plt.figure(figsize=(17,5))
rest = sns.countplot(x="rest_type",data=zomato_real, palette = "Set1")
rest.set_xticklabels(rest.get_xticklabels(), rotation=90, ha="right")
plt.ylabel("Frequency",size=15)
plt.xlabel("Restaurant type",size=15)
rest
plt.title('Restaurant types',fontsize = 20 ,pad=20)
```

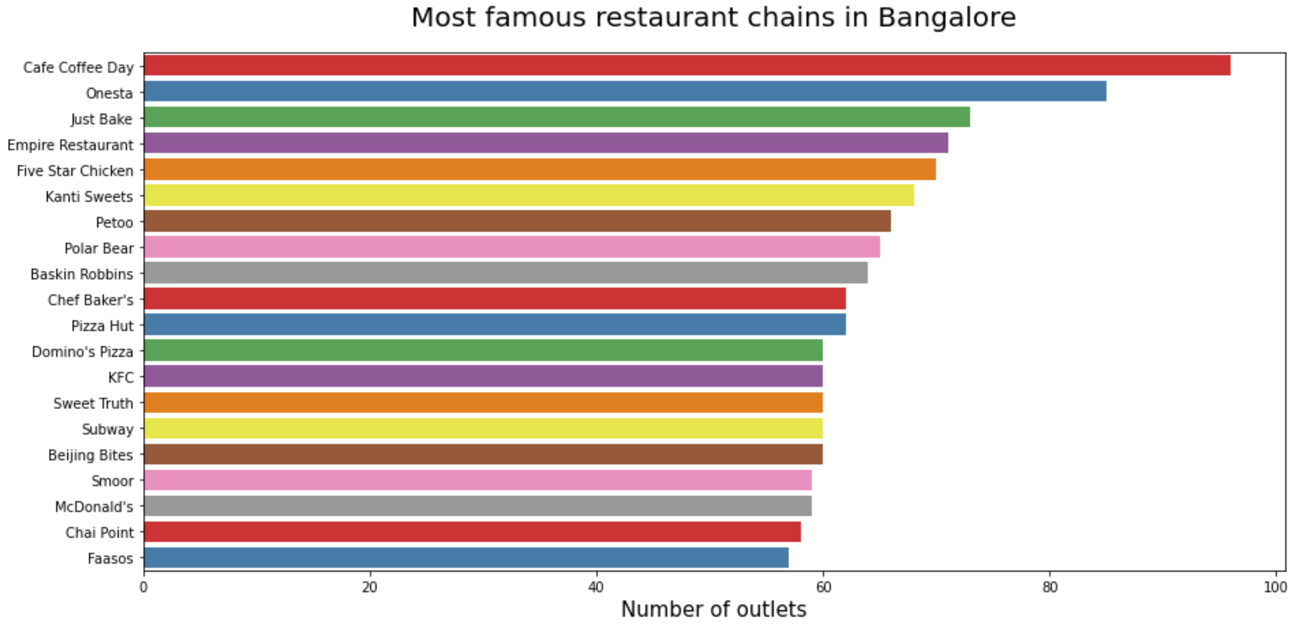
```
Text(0.5, 1.0, 'Restaurant types')
```



Most famous Restaurant chains in Bengalore

```
plt.figure(figsize=(15,7))
chains=zomato_real['name'].value_counts()[:20]
sns.barplot(x=chains,y=chains.index,palette='Set1')
plt.title("Most famous restaurant chains in Bangalore",size=20,pad=20)
plt.xlabel("Number of outlets",size=15)
```

```
Text(0.5, 0, 'Number of outlets')
```



Predicting by training the following models

Linear Regression

```
#Prepare a Linear Regression Model
reg=LinearRegression()
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.2736233722103949
```

Decision Tree Regression

```
#Prepairng a Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.1,random_state=105)
DTree=DecisionTreeRegressor(min_samples_leaf=.0001)
DTree.fit(x_train,y_train)
y_predict=DTree.predict(x_test)
from sklearn.metrics import r2_score
r2_score(y_test,y_predict)
```

```
     0.853799630505989
```

Random Forest Regression

```
#Preparing Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
RForest=RandomForestRegressor(n_estimators=500,random_state=329,min_samples_leaf=.0001)
RForest.fit(x_train,y_train)
y_predict=RForest.predict(x_test)
from sklearn.metrics import r2_score
r2_score(y_test,y_predict)
```

```
     0.8774282743423502
```

Extra Tree Regressor

```
#Preparing Extra Tree Regression
from sklearn.ensemble import  ExtraTreesRegressor
ETree=ExtraTreesRegressor(n_estimators = 100)
ETree.fit(x_train,y_train)
y_predict=ETree.predict(x_test)
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_predict)
```

```
     0.9400541490803134
```

```
import pickle
# Saving model to disk
pickle.dump(ETree, open('model.pkl','wb'))
```

It can be observed that we have got the best accuracy for Extra tree regressor