In [5]:

```
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objs as go
import plotly.offline as py
import seaborn as sns

import matplotlib.ticker as mtick
plt.style.use('fivethirtyeight')
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import  ExtraTreesRegressor
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [6]:

```
import pandas as pd
data=pd.read_csv(r'C:\Users\ritvikpalvankar7\OneDrive - University of Florida\Desktop\Udemy Coursework Documents\2021 ML deployment Mastery\Flask Deployment\zomato.csv')
```

In [7]:

```
data.head()
```

Out[7]:

	url	address	name	online_order	book_tabl
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Ye
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	N
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	N
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	N
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	N

In [8]:

```
data.shape
```

Out[8]:

```
(51717, 17)
```

In [9]:

```
data.dtypes #checking the data types
```

Out[9]:

```
url                             object
address                         object
name                            object
online_order                    object
book_table                      object
rate                            object
votes                            int64
phone                           object
location                        object
rest_type                       object
dish_liked                      object
cuisines                        object
approx_cost(for two people)     object
reviews_list                    object
menu_item                       object
listed_in(type)                 object
listed_in(city)                 object
dtype: object
```

In [10]:

```
data.isna().sum() #Checking null values
```

Out[10]:

```
url                               0
address                           0
name                              0
online_order                      0
book_table                        0
rate                           7775
votes                             0
phone                          1208
location                         21
rest_type                       227
dish_liked                    28078
cuisines                         45
approx_cost(for two people)     346
reviews_list                      0
menu_item                         0
listed_in(type)                   0
listed_in(city)                   0
dtype: int64
```

In [11]:

```
df=data.drop(['url','phone'],axis=1) #Dropping the column like "phone" and "url" and sa
ving the new dataset as "df"
```

In [12]:

```
df.duplicated().sum()
```

Out[12]:

```
43
```

In [13]:

```
df.drop_duplicates(inplace=True)
```

In [14]:

```
df.duplicated().sum()
```

Out[14]:

```
0
```

In [15]:

```
#Remove the NaN values from the dataset
df.dropna(how='any',inplace=True)
df.isnull().sum()
```

Out[15]:

```
address                        0
name                           0
online_order                   0
book_table                     0
rate                           0
votes                          0
location                       0
rest_type                      0
dish_liked                     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
```

In [16]:

```
df.columns
```

Out[16]:

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

In [17]:

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

Out[17]:

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

In [18]:

```
df.head()
```

Out[18]:

	address	name	online_order	book_table	rate	votes	location	rest_type	c
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	Banashankari	Cafe, Casual Dining	C M
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashankari	Quick Bites	
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	Basavanagudi	Casual Dining	

Cleaning the dataset

In [19]:

```
df['cost'].unique()
```

Out[19]:

```
array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
       '750', '200', '850', '1,200', '150', '350', '250', '1,500',
       '1,300', '1,000', '100', '900', '1,100', '1,600', '950', '230',
       '1,700', '1,400', '1,350', '2,200', '2,000', '1,800', '1,900',
       '180', '330', '2,500', '2,100', '3,000', '2,800', '3,400', '40',
       '1,250', '3,500', '4,000', '2,400', '1,450', '3,200', '6,000',
       '1,050', '4,100', '2,300', '120', '2,600', '5,000', '3,700',
       '1,650', '2,700', '4,500'], dtype=object)
```

In [20]:

```
#zomato['cost'] = zomato['cost'].astype(str) #Changing the cost to string
df['cost'] = df['cost'].apply(lambda x: x.replace(',','')) #Using lambda function to re
place ',' from cost
df['cost'] = df['cost'].astype(float)
```

In [21]:

```
print(df['cost'].unique())

print('---'*10)

df.dtypes
```

```
[ 800.  300.  600.  700.  550.  500.  450.  650.  400.  750.  200.  850.
 1200.  150.  350.  250. 1500. 1300. 1000.  100.  900. 1100. 1600.  950.
  230. 1700. 1400. 1350. 2200. 2000. 1800. 1900.  180.  330. 2500. 2100.
 3000. 2800. 3400.   40. 1250. 3500. 4000. 2400. 1450. 3200. 6000. 1050.
 4100. 2300.  120. 2600. 5000. 3700. 1650. 2700. 4500.]
------------------------------
```

Out[21]:

```
address          object
name             object
online_order     object
book_table       object
rate             object
votes             int64
location         object
rest_type        object
dish_liked       object
cuisines         object
cost            float64
reviews_list     object
menu_item        object
type             object
city             object
dtype: object
```

In [22]:

```
#Reading uninque values from the Rate column
df['rate'].unique()
```

Out[22]:

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

In [23]:

```
df = df.loc[df.rate !='NEW'] #getting rid of "NEW"
```

In [24]:

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

Out[24]:

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

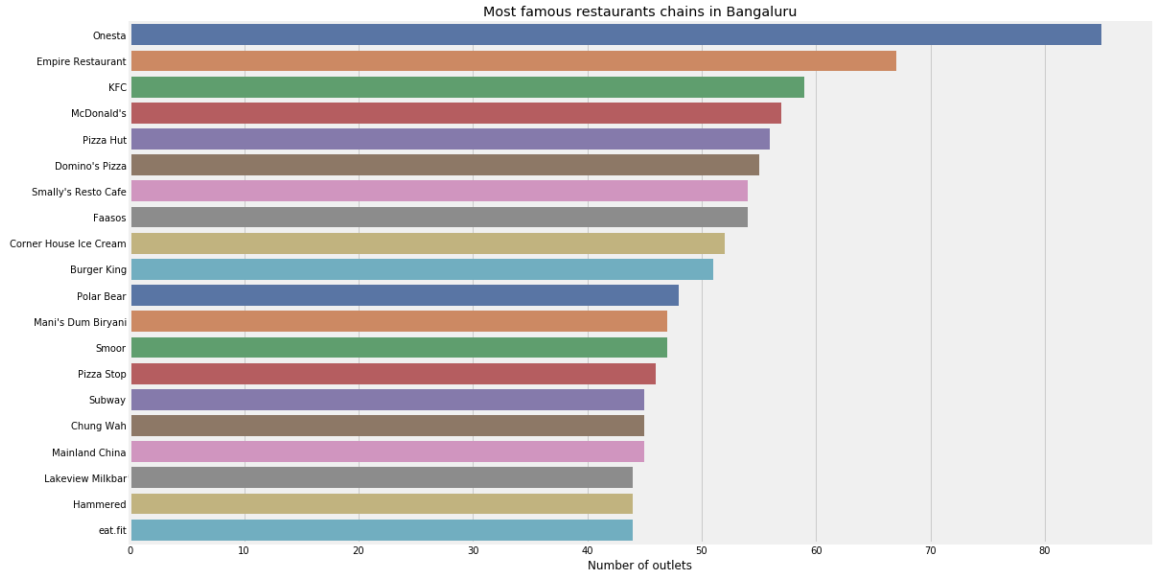
In [25]:

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

Visualizations

In [26]:

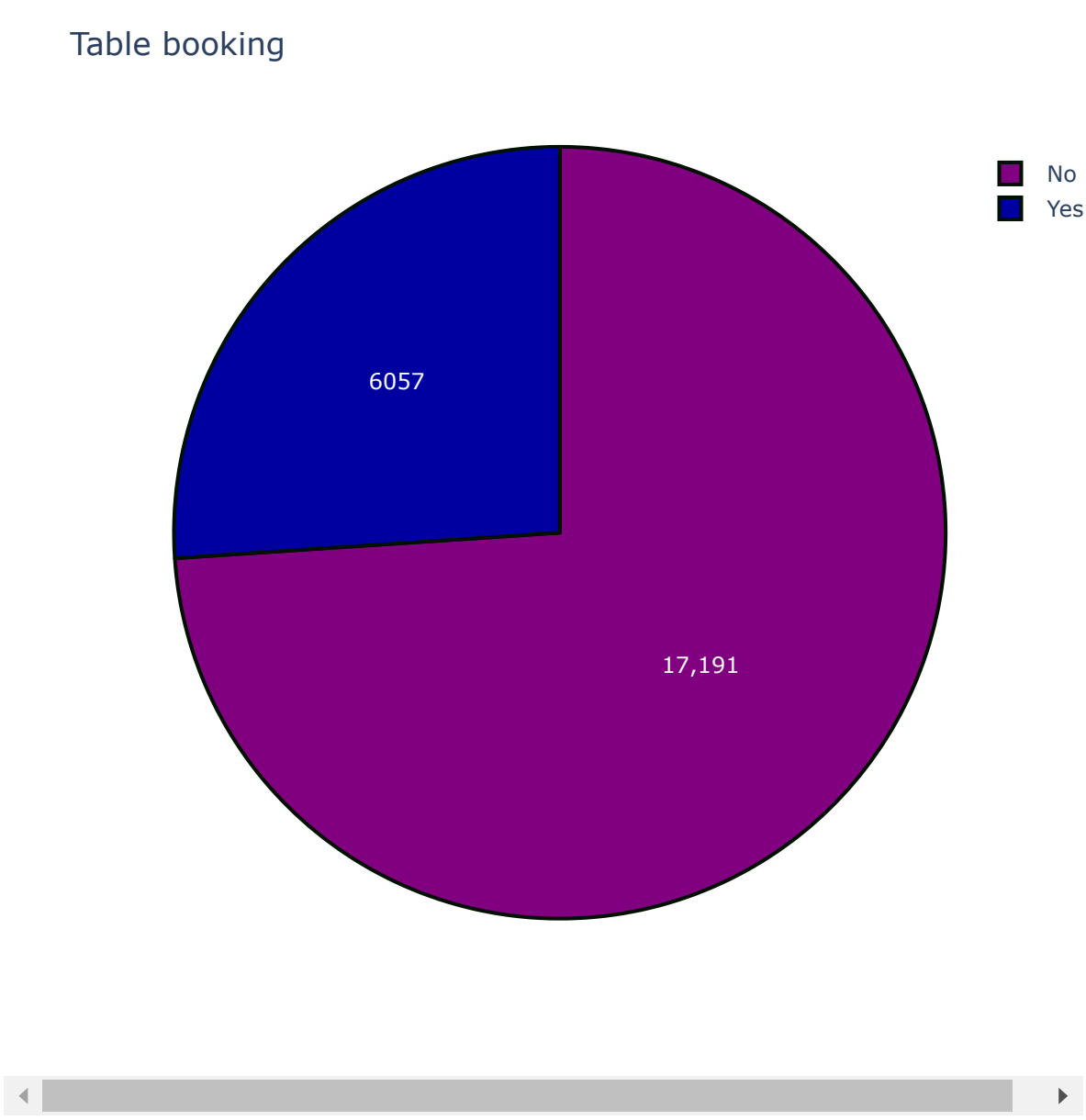
```
plt.figure(figsize=(17,10))
chains=df['name'].value_counts()[:20]
sns.barplot(x=chains,y=chains.index,palette='deep')
plt.title("Most famous restaurants chains in Bangaluru")
plt.xlabel("Number of outlets")
plt.show()
```



In [27]:

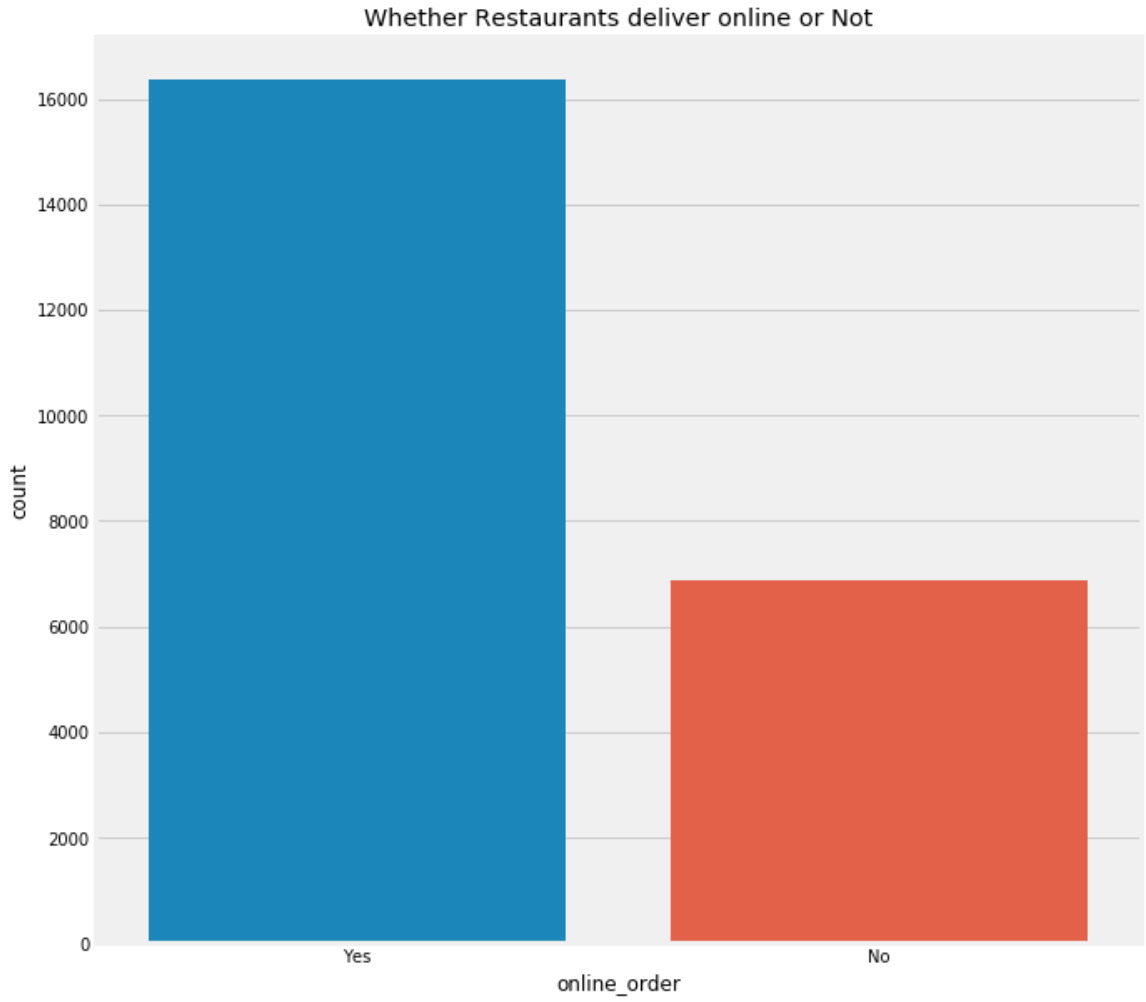
```
x=df['book_table'].value_counts()
colors = ['#800080', '#0000A0']

trace=go.Pie(labels=x.index,values=x,textinfo="value",
             marker=dict(colors=colors,
                         line=dict(color='#001000', width=2)))
layout=go.Layout(title="Table booking",width=600,height=600)
fig=go.Figure(data=[trace],layout=layout)
py.iplot(fig, filename='pie_chart_subplots')
```



In [28]:

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



Rating distribution

In [30]:

```
#How ratings are distributed
#plt.figure(figsize=(9,7))

#sns.distplot(df['rate'],bins=20)
```

Count of ratings as between "1 and 2", "2 and 3", "3 and 4", and "4 and 5"

In [31]:

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

Out[31]:

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

In [32]:

```
df['rate']=df['rate'].astype(float)
```

In [33]:

```
((df['rate']>=1) & (df['rate']<2)).sum()
```

Out[33]:

```
5
```

In [34]:

```
((df['rate']>=2) & (df['rate']<3)).sum()
```

Out[34]:

```
1179
```

In [35]:

```
((df['rate']>=3) & (df['rate']<4)).sum()
```

Out[35]:

```
10153
```

In [36]:

```
(df['rate']>=4).sum()
```

Out[36]:

```
11911
```

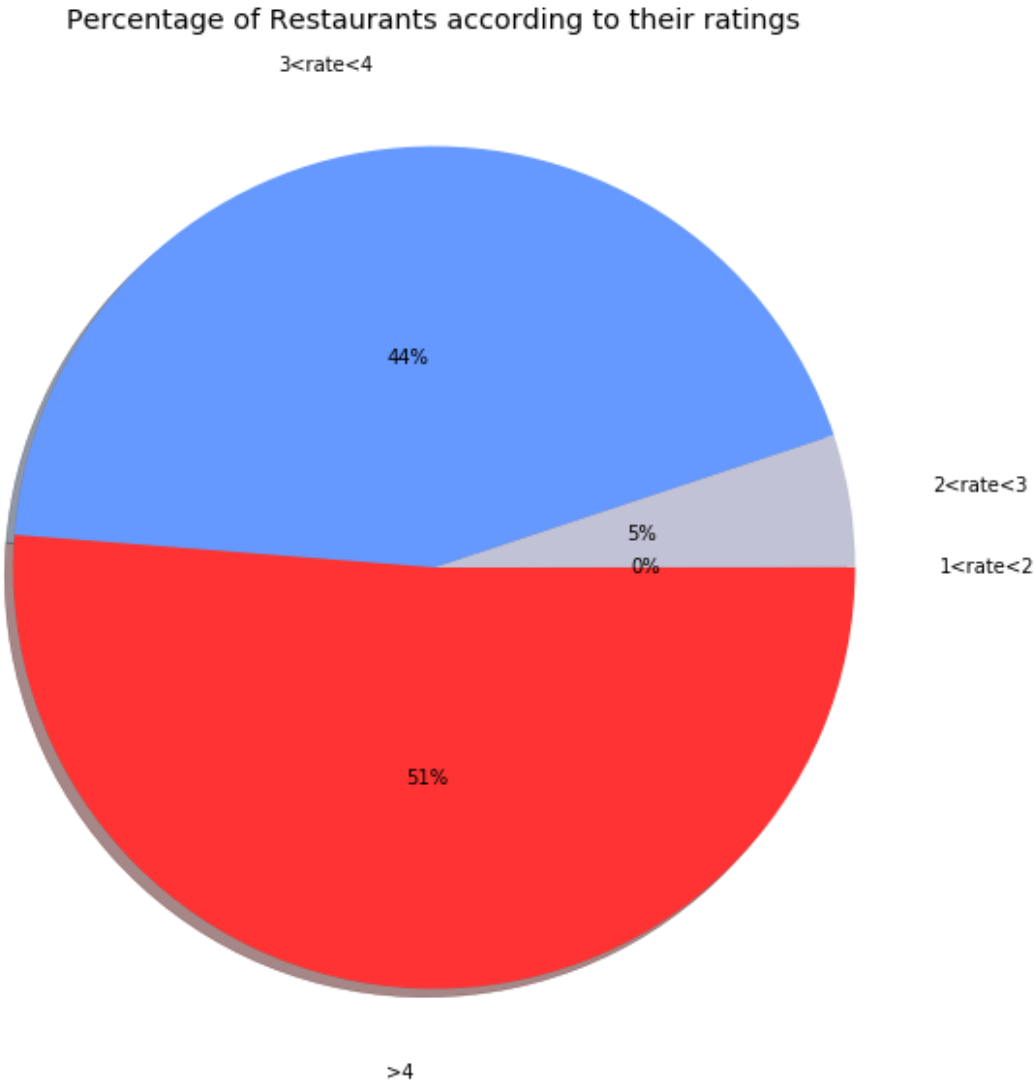
Plotting the counts with the help of pie chart

In [37]:

```
slices=[((df['rate']>=1) & (df['rate']<2)).sum(),
        ((df['rate']>=2) & (df['rate']<3)).sum(),
        ((df['rate']>=3) & (df['rate']<4)).sum(),
        (df['rate']>=4).sum()
        ]

labels=['1<rate<2','2<rate<3','3<rate<4','>4']
colors = ['#ff3333','#c2c2d6','#6699ff']
plt.pie(slices,colors=colors, labels=labels, autopct='%1.0f%%', pctdistance=.5, labeldi
stance=1.2,shadow=True)
fig = plt.gcf()
plt.title("Percentage of Restaurants according to their ratings")

fig.set_size_inches(10,10)
plt.show()
```



Service types

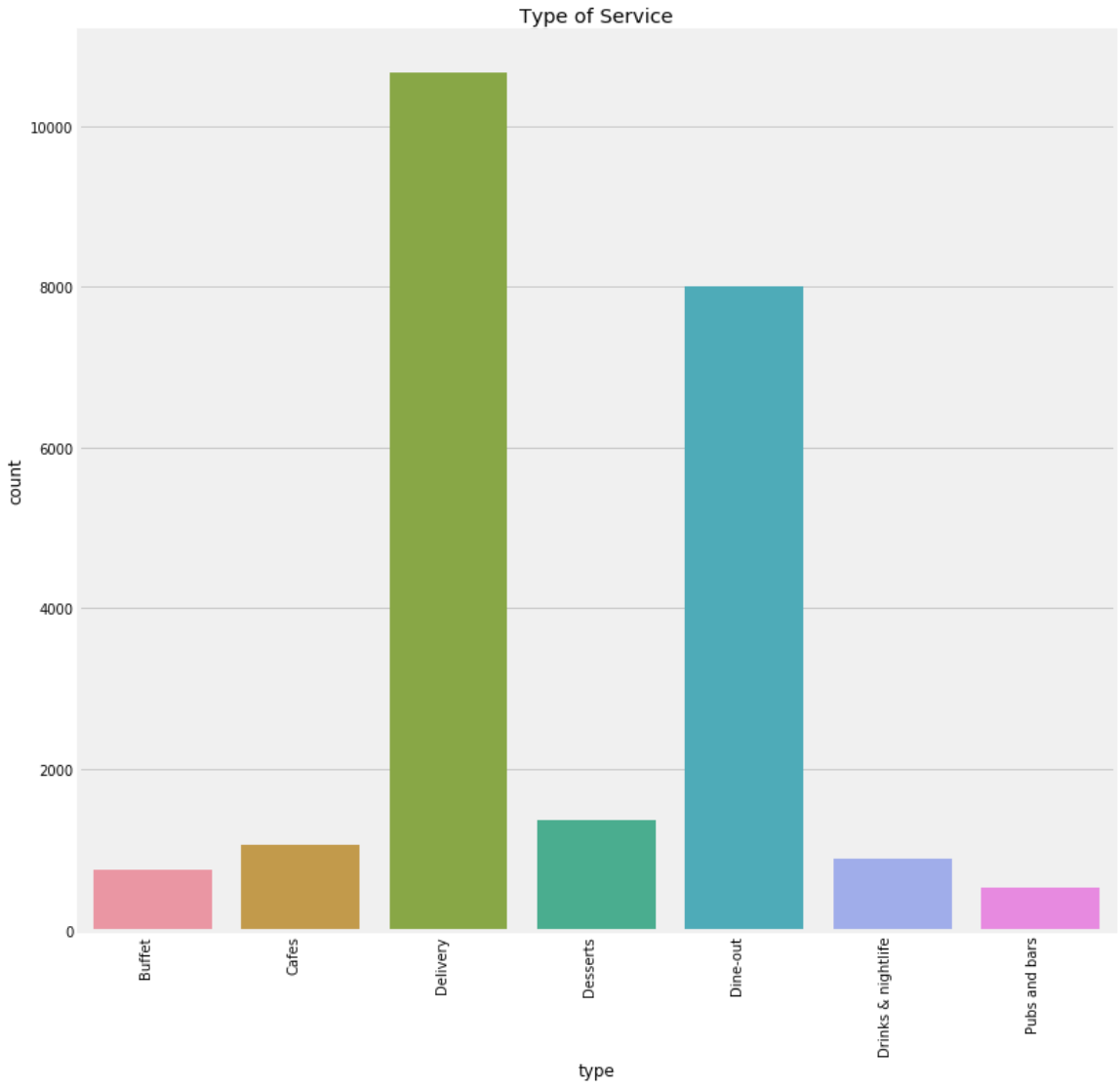
In [38]:

```
#Types of Services

sns.countplot(df['type']).set_xticklabels(sns.countplot(df['type']).get_xticklabels(),
rotation=90, ha="right")
fig = plt.gcf()
fig.set_size_inches(12,12)
plt.title('Type of Service')
```

Out[38]:

```
Text(0.5, 1.0, 'Type of Service')
```



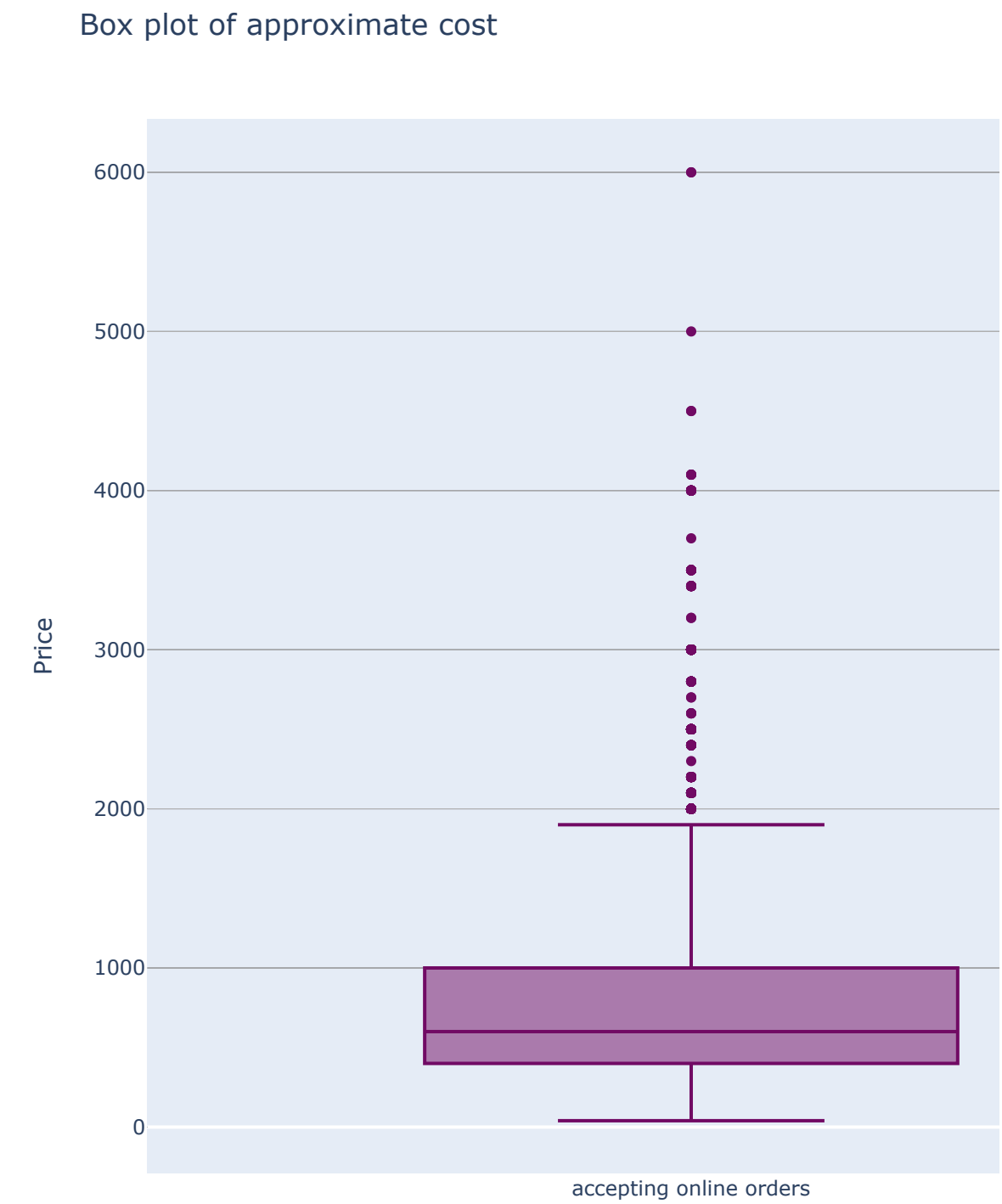
Distribution of Cost of Food for two People¶

In [39]:

```
from plotly.offline import iplot
```

In [40]:

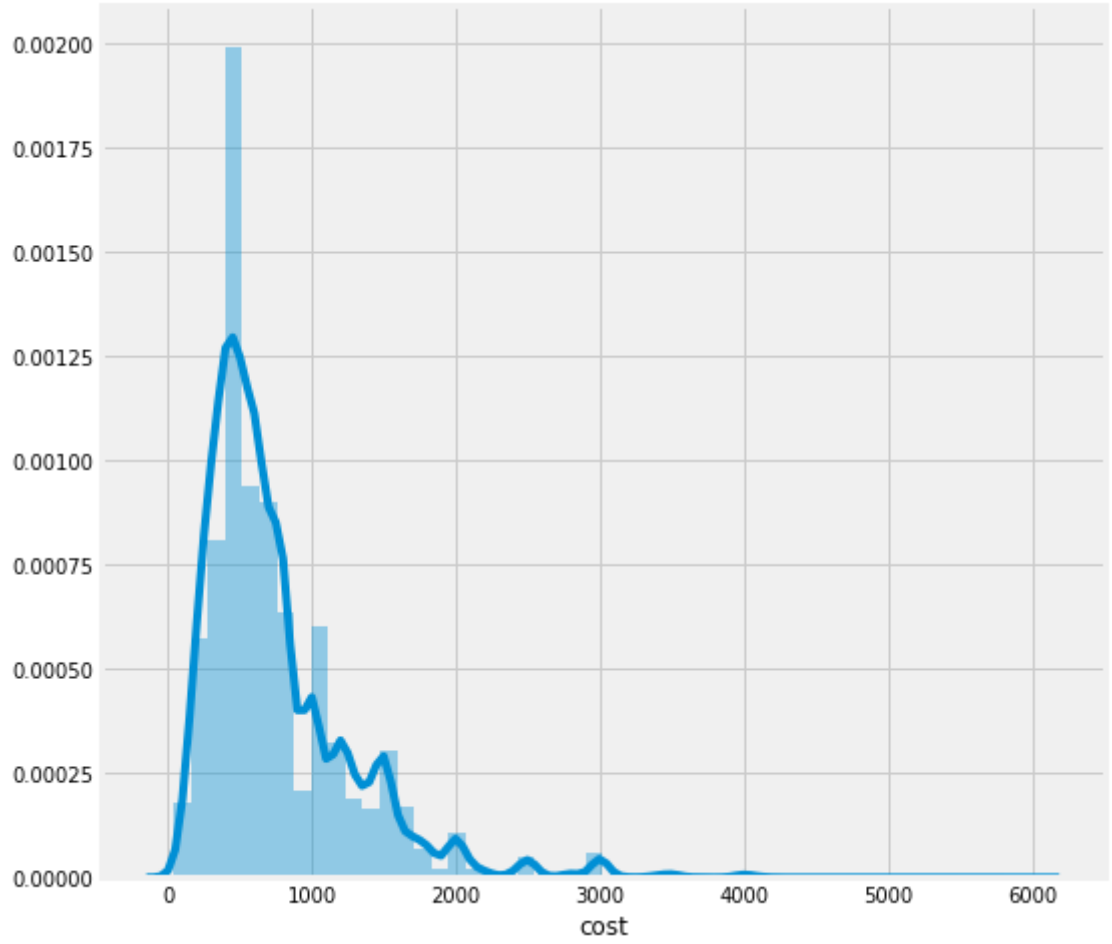
```
trace0=go.Box(y=df['cost'],name="accepting online orders",
              marker = dict(
        color = 'rgb(113, 10, 100)',
    ))
data=[trace0]
layout=go.Layout(title="Box plot of approximate cost",width=800,height=800,yaxis=dict(t
itle="Price"))
fig=go.Figure(data=data,layout=layout)
py.iplot(fig)
```



Distribution of charges

In [41]:

```
plt.figure(figsize=(8,8))
sns.distplot(df['cost'])
plt.show()
```



Most liked dishes

In [42]:

```
import re

df.index=range(df.shape[0])
likes=[]
for i in range(df.shape[0]):
    array_split=re.split(',',df['dish_liked'][i])
    for item in array_split:
        likes.append(item)
```

In [43]:

```
df.index=range(df.shape[0])
```

In [44]:

```
df.index
```

Out[44]:

```
RangeIndex(start=0, stop=23248, step=1)
```

In [45]:

```
print("Count of Most liked dishes in Bangalore")
favourite_food = pd.Series(likes).value_counts()
favourite_food.head(30)
```

```
Count of Most liked dishes in Bangalore
```

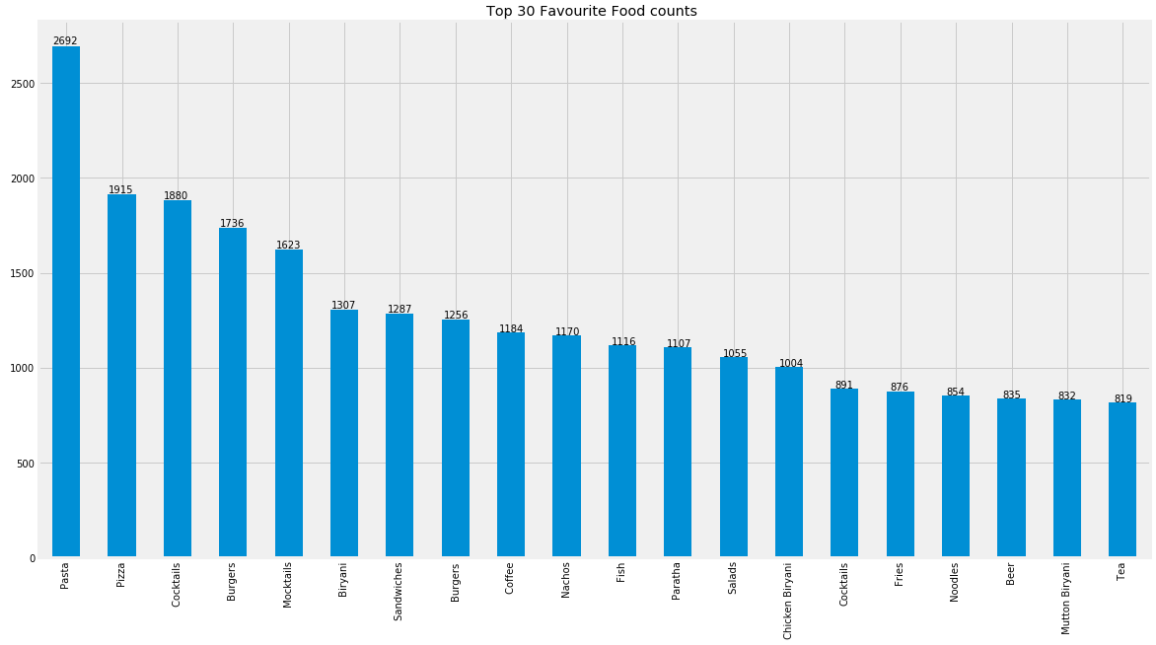
Out[45]:

```
 Pasta              2692
 Pizza              1915
 Cocktails          1880
 Burgers            1736
 Mocktails          1623
 Biryani            1307
 Sandwiches         1287
Burgers             1256
 Coffee             1184
 Nachos             1170
 Fish               1116
 Paratha            1107
 Salads             1055
 Chicken Biryani    1004
Cocktails            891
 Fries               876
 Noodles             854
 Beer                835
 Mutton Biryani      832
 Tea                 819
Coffee               801
 Sandwich            788
 Butter Chicken      782
 Thali               770
Biryani              749
Pizza                747
 Roti                729
 Brownie             726
 Salad               677
 Hot Chocolate       672
dtype: int64
```

In [46]:

```
ax = favourite_food.nlargest(n=20, keep='first').plot(kind='bar',figsize=(18,10),title
= 'Top 30 Favourite Food counts ')

for i in ax.patches:
    ax.annotate(str(i.get_height()), (i.get_x() * 1.005, i.get_height() * 1.005))
```



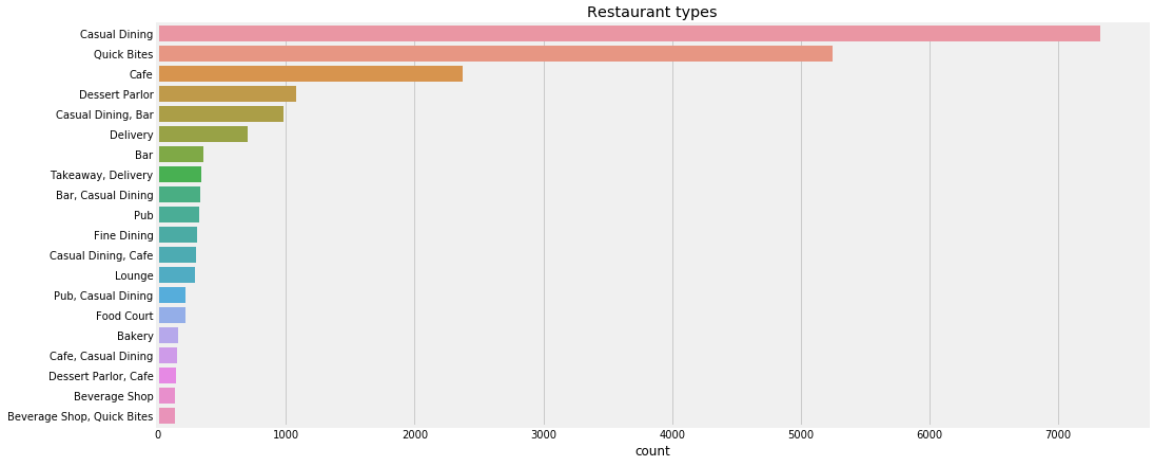
Restaurant and their types

In [49]:

```
plt.figure(figsize=(15,7))
rest=df['rest_type'].value_counts()[:20]
sns.barplot(rest,rest.index)
plt.title("Restaurant types")
plt.xlabel("count")
```

Out[49]:

```
Text(0.5, 0, 'count')
```



Model

In [50]:

```
df.head()
```

Out[50]:

	address	name	online_order	book_table	rate	votes	location	rest_type	d
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	(
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	C M
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	G

In [51]:

```
df.online_order[df.online_order == 'Yes'] = 1
df.online_order[df.online_order == 'No'] = 0
```

In [52]:

```
df.online_order.value_counts()
```

Out[52]:

```
1    16378
0     6870
Name: online_order, dtype: int64
```

In [53]:

```
df.online_order = pd.to_numeric(df.online_order)
```

In [54]:

```
df.book_table[df.book_table == 'Yes'] = 1
df.book_table[df.book_table == 'No'] = 0
```

In [55]:

```
df.book_table = pd.to_numeric(df.book_table)
```

In [56]:

```
df.book_table.value_counts()
```

Out[56]:

```
0    17191
1     6057
Name: book_table, dtype: int64
```

In [57]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [58]:

```
df.location = le.fit_transform(df.location)
df.rest_type = le.fit_transform(df.rest_type)
df.cuisines = le.fit_transform(df.cuisines)
df.menu_item = le.fit_transform(df.menu_item)
```

In [59]:

```
df.head()
```

Out[59]:

	address	name	online_order	book_table	rate	votes	location	rest_type	dish_li
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	1	1	4.1	775	1	20	Pa Lu Bu Ma Pa Pai La
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	1	0	4.1	787	1	20	Mor Lu Bu Chocc Nirv Thai
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	1	0	3.8	918	1	16	Chu Cannel Minest Soup, Ch
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	0	0	3.7	88	1	62	Ma D
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	0	0	3.8	166	4	20	Pani Gol Ga

In [60]:

```
my_data=df.iloc[:,[2,3,4,5,6,7,9,10,12]]
my_data.to_csv(r'C:\Users\ritvikpalvankar7\OneDrive - University of Florida\Desktop\Udemy Coursework Documents\2021 ML deployment Mastery\Flask Deployment\Zomato_df.csv')
```

In [61]:

```
x = df.iloc[:,[2,3,5,6,7,9,10,12]]
x.head()
```

Out[61]:

	online_order	book_table	votes	location	rest_type	cuisines	cost	menu_item
0	1	1	775	1	20	1386	800.0	5047
1	1	0	787	1	20	594	800.0	5047
2	1	0	918	1	16	484	800.0	5047
3	0	0	88	1	62	1587	300.0	5047
4	0	0	166	4	20	1406	600.0	5047

In [62]:

```
y = df['rate']
y
```

Out[62]:

```
0        4.1
1        4.1
2        3.8
3        3.7
4        3.8
        ...
23243    3.8
23244    3.9
23245    2.8
23246    2.5
23247    4.3
Name: rate, Length: 23248, dtype: float64
```

In [63]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=10)
```

Regression

In [64]:

```
lr_model=LinearRegression()
lr_model.fit(x_train,y_train)
```

Out[64]:

```
LinearRegression()
```

In [65]:

```
from sklearn.metrics import r2_score
y_pred=lr_model.predict(x_test)
r2_score(y_test,y_pred)
```

Out[65]:

```
0.22818828522967138
```

Random Forest

In [66]:

```
from sklearn.ensemble import RandomForestRegressor
RF_Model=RandomForestRegressor(n_estimators=650,random_state=245,min_samples_leaf=.0001
)
RF_Model.fit(x_train,y_train)
y_predict=RF_Model.predict(x_test)
r2_score(y_test,y_predict)
```

Out[66]:

```
0.8809706960047533
```

Extra Tree regressor

In [67]:

```
#Preparing Extra Tree Regression
from sklearn.ensemble import  ExtraTreesRegressor
ET_Model=ExtraTreesRegressor(n_estimators = 120)
ET_Model.fit(x_train,y_train)
y_predict=ET_Model.predict(x_test)


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

Out[67]:

```
0.9332757719670571
```

In [69]:

```
import pickle
# Saving model to disk
pickle.dump(ET_Model, open(r'C:\Users\ritvikpalvankar7\OneDrive - University of Florida
\Desktop\Udemy Coursework Documents\2021 ML deployment Mastery\Flask Deployment\model.p
kl','wb'))
model=pickle.load(open(r'C:\Users\ritvikpalvankar7\OneDrive - University of Florida\Des
ktop\Udemy Coursework Documents\2021 ML deployment Mastery\Flask Deployment\model.pkl',
'rb'))
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: