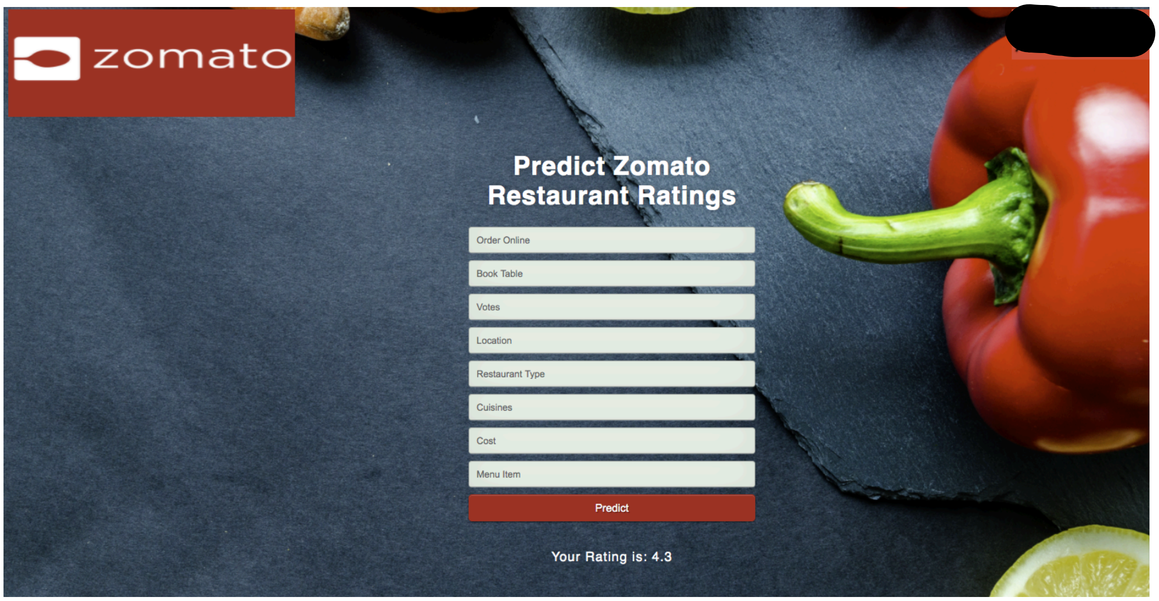
End-To-End Deployment of Zomato Restaurant Ratings


[redacted]

Predict Zomato Restaurant Ratings

Your Rating is: 4.3

ABSTRACT

Zomato is one of the best online food delivery apps which gives the users the ratings and the reviews on restaurants all over india.These ratings and the Reviews are considered as one of the most important deciding factors which determine how good a restaurant is.

We will therefore use the real time Data set with variuos features a user would look into regarding a restaurant. We will be considering Banglore City in this analysis.

Content The basic idea of analyzing the Zomato dataset is to get a fair idea about the factors affecting the establishment of different types of restaurant at different places in Bengaluru, aggregate rating of each restaurant, Bengaluru being one such city has more than 12,000 restaurants with restaurants serving dishes from all over the world.

With each day new restaurants opening the industry has’nt been saturated yet and the demand is increasing day by day. Inspite of increasing demand it however has become difficult for new restaurants to compete with established restaurants. Most of them serving the same food. Bengaluru being an IT capital of India. Most of the people here are dependent mainly on the restaurant food as they don’t have time to cook for themselves.

With such an overwhelming demand of restaurants it has therefore become important to study the demography of a location. What kind of a food is more popular in a locality. Do the entire locality loves vegetarian food. If yes then is that locality populated by a particular sect of people for eg. Jain, Marwaris, Gujaratis who are mostly vegetarian. These kind of analysis can be done using the data, by studying the factors such as

```
• Location of the restaurant
• Approx Price of food
• Theme based restaurant or not
• Which locality of that city serves that cuisines with
maximum number of restaurants
• The needs of people who are striving to get the best
cuisine of the neighborhood
• Is a particular neighborhood famous for its own kind of
food.
```

“Just so that you have a good meal the next time you step out”

The data is accurate to that available on the zomato website until 15 March 2019. The data was scraped from Zomato in two phase. After going through the structure of the website I found that for each neighborhood there are 6-7 category of restaurants viz. Buffet, Cafes, Delivery, Desserts, Dine-out, Drinks & nightlife, Pubs and bars.

Phase I,

In Phase I of extraction only the URL, name and address of the restaurant were extracted which were visible on the front page. The URl's for each of the restaurants on the zomato were recorded in the csv file so that later the data can be extracted individually for each restaurant. This made the extraction process easier and reduced the extra load on my machine. The data for each neighborhood and each category can be found here

Phase II,

In Phase II the recorded data for each restaurant and each category was read and data for each restaurant was scraped individually. 15 variables were scraped in this phase. For each of the neighborhood and for each category their onlineorder, booktable, rate, votes, phone, location, resttype, dishliked, cuisines, approxcost(for two people), reviewslist, menu_item was extracted. See section 5 for more details about the variables.

Acknowledgements The data scraped was entirely for educational purposes only. Note that I don’t claim any copyright for the data. All copyrights for the data is owned by Zomato Media Pvt. Ltd..

```
Source: Kaggle
```

Main Objective:

The main agenda of this project is:

> > Perform extensive **Exploratory Data Analysis(EDA)** on the Zomato Dataset.

> > Build an appropriate **Machine Learning Model** that will help various Zomato Restaurants to predict their respective Ratings based on certain features

> > **DEPLOY** the Machine learning model via **Flask** that can be used to make live predictions of restaurants ratings

A step by step guide is attached to this documnet as well as a video explanation of each concpet.

In [ ]:

```python
#IMPORT THE NECESSARY LIBRARIES

import pandas as pd
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
```

MOUNT DRIVE

In [ ]:

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?clien
t_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.co
m&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=
email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2f
www.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%
2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleap
i.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

In [ ]:
```
#!ls /content/
```

LOAD DATASET

In [ ]:
```
data = pd.read_csv('/content/drive/My Drive/zomato.csv')
```

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

Out[ ]:

	url	address	name	online_orc
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	

Columns description

1. **url** contains the url of the restaurant in the zomato website

2. **address** contains the address of the restaurant in Bengaluru

3. **name** contains the name of the restaurant

4. **online_order** whether online ordering is available in the restaurant or not

5. **book_table** table book option available or not

6. **rate** contains the overall rating of the restaurant out of 5

7. **votes** contains total number of rating for the restaurant as of the above mentioned date

8. **phone** contains the phone number of the restaurant

9. **location** contains the neighborhood in which the restaurant is located

10. **rest_type** restaurant type

11. **dish_liked** dishes people liked in the restaurant

12. **cuisines** food styles, separated by comma

13. **approx_cost**(for two people) contains the approximate cost of meal for two people

14. **reviews_list** list of tuples containing reviews for the restaurant, each tuple

15. **menu_item** contains list of menus available in the restaurant

16. **listed_in**(type) type of meal

17. **listed_in**(**city**) contains the neighborhood in which the restaurant is listed

In [ ]:

```
data.shape
```

Out[ ]:

```
(51717, 17)
```

In [ ]:

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

Out[ ]:

```
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 [ ]:

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

Out[ ]:

```
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 [ ]:

```
#You can use pandas profiling to get an over all overview of the dataset
# import pandas_profiling as pf

# pf.ProfileReport(df)
```

In [ ]:

```
#Deleting Unnnecessary Columns
df=data.drop(['url','phone'],axis=1) #Dropping the column like "phone" and "
```

Checking for duplicate values

In [ ]:

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

Out[ ]:

```
43
```

```
In [ ]: df.drop_duplicates(inplace=True)
```

```
In [ ]: df.duplicated().sum()
```

```
Out[ ]: 0
```

Drop null values

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

```
Out[ ]: 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
```

Renaming columes appropriately

```
In [ ]: df.columns
```

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

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

Out[ ]:

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

Cleaning the dataset

In [ ]:

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

Out[ ]:

```
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)
```

replacing the "," with nothing and converting the results to float

In [ ]:

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

In [ ]:

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

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

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

```
Out[ ]: 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)
```

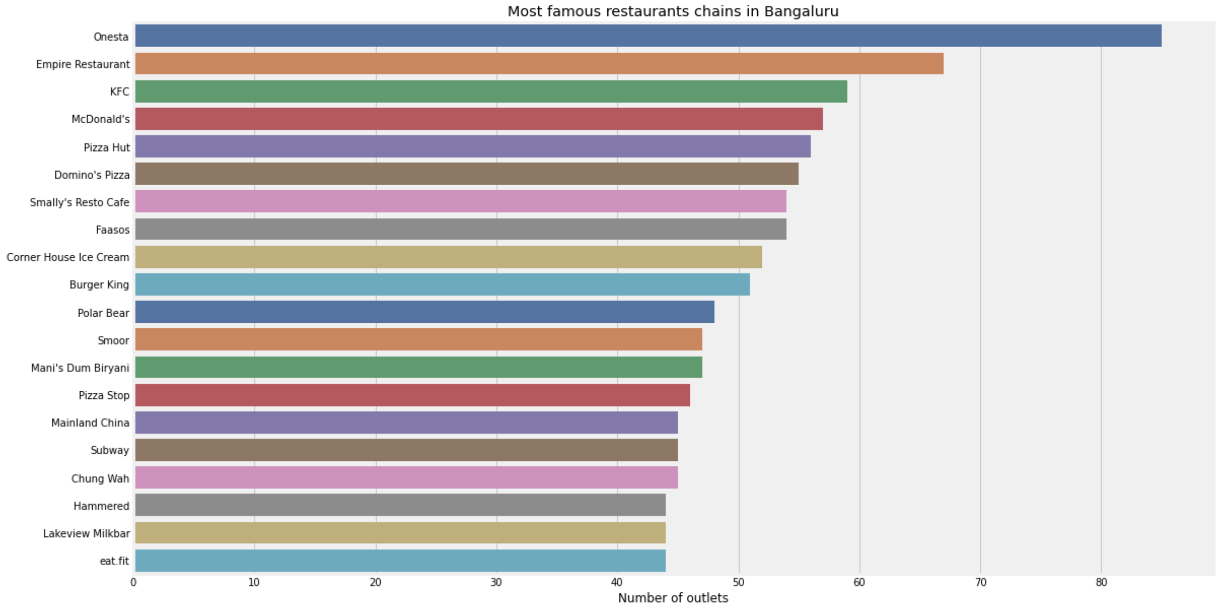
```
In [ ]: #Removing '/5' from Rates

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

Visualisations

Most famous restaurants chains in Bangaluru

```
In [ ]: 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()
```



Whether restaurant offer Table booking or not

```
In [ ]: 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')
```

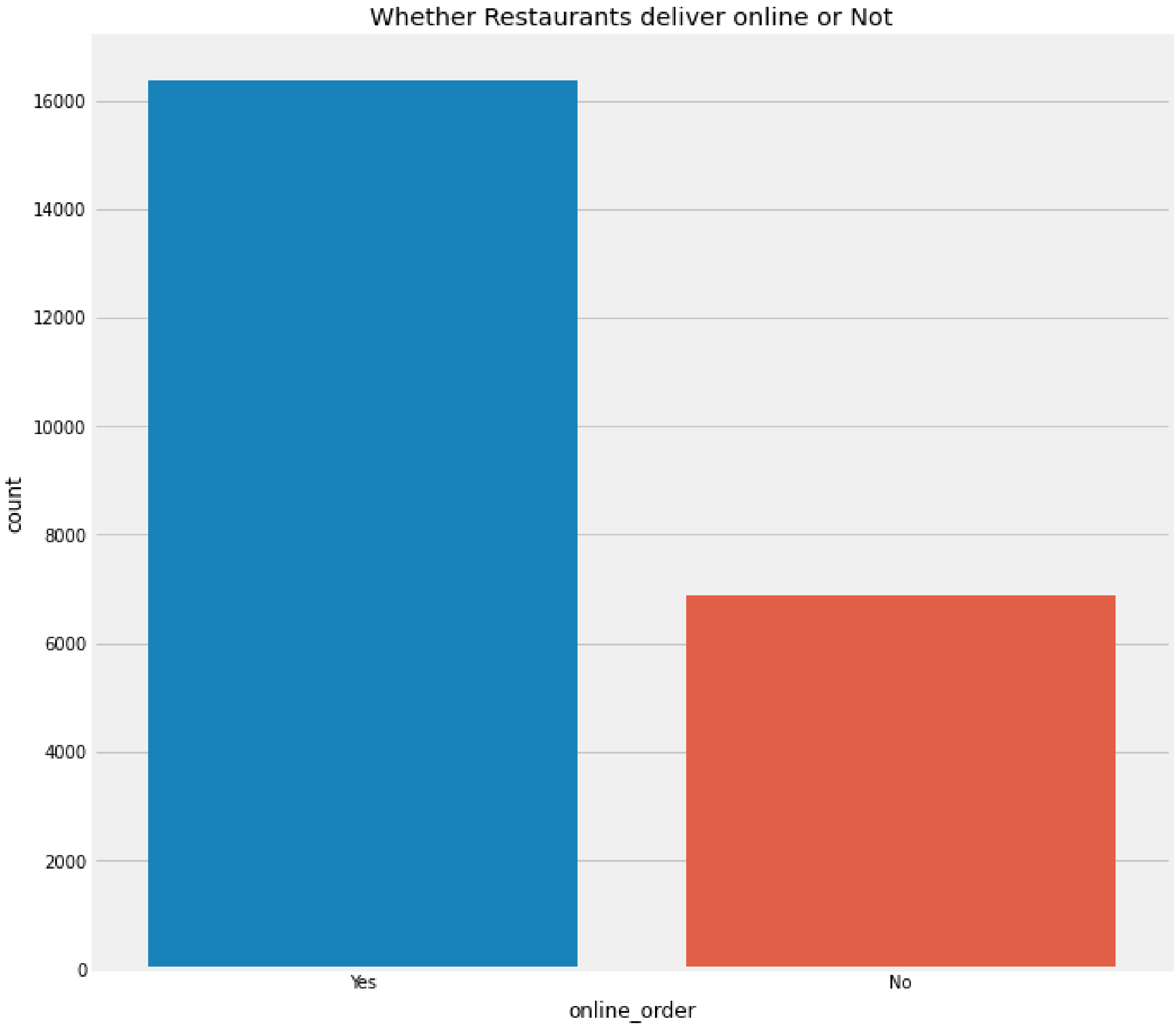
Insight

Most of the Restaurants do not offer table booking

Whether Restaurants deliver online or Not

In [ ]:

```
#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()
```



Insight:

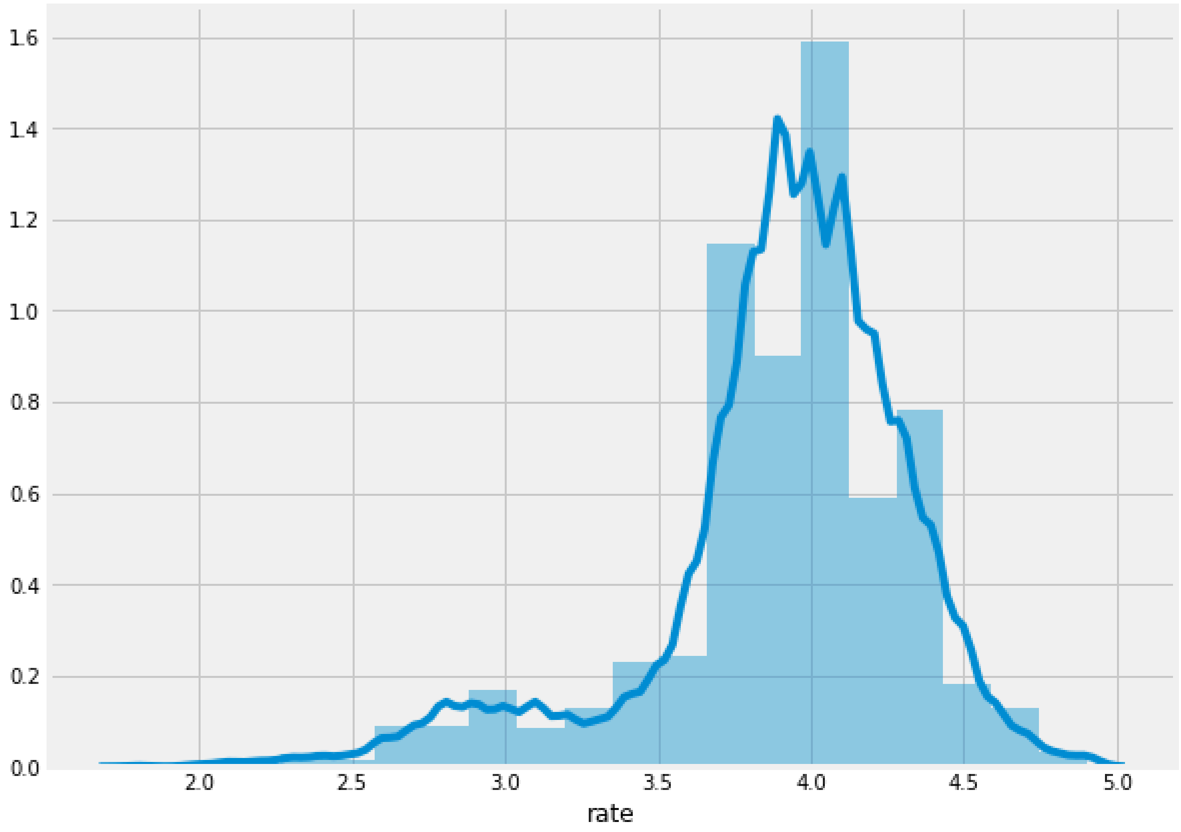
Most Restaurants offer option for online order and delivery

Rating Distributions

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

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

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f714ef32828>
```



- **Insight**

We can infer from above that most of the ratings are within 3.5 and 4.5

In [ ]:
```
# #Distribution of the cost Vs ratings in parallel with online order
# plt.figure(figsize=(10,7))
# sns.scatterplot(x="rate",y='cost',hue='online_order',data=df)
# plt.show()
```

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

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

Out[ ]:
```
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 [ ]:
```
df['rate'].min()
```

Out[ ]:
```
'1.8'
```

In [ ]:

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

Out[ ]:

```
'4.9 '
```

In [ ]:

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

In [ ]:

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

Out[ ]:

```
5
```

In [ ]:

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

Out[ ]:

```
1179
```

In [ ]:

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

Out[ ]:

```
10153
```

In [ ]:

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

Out[ ]:

```
11911
```

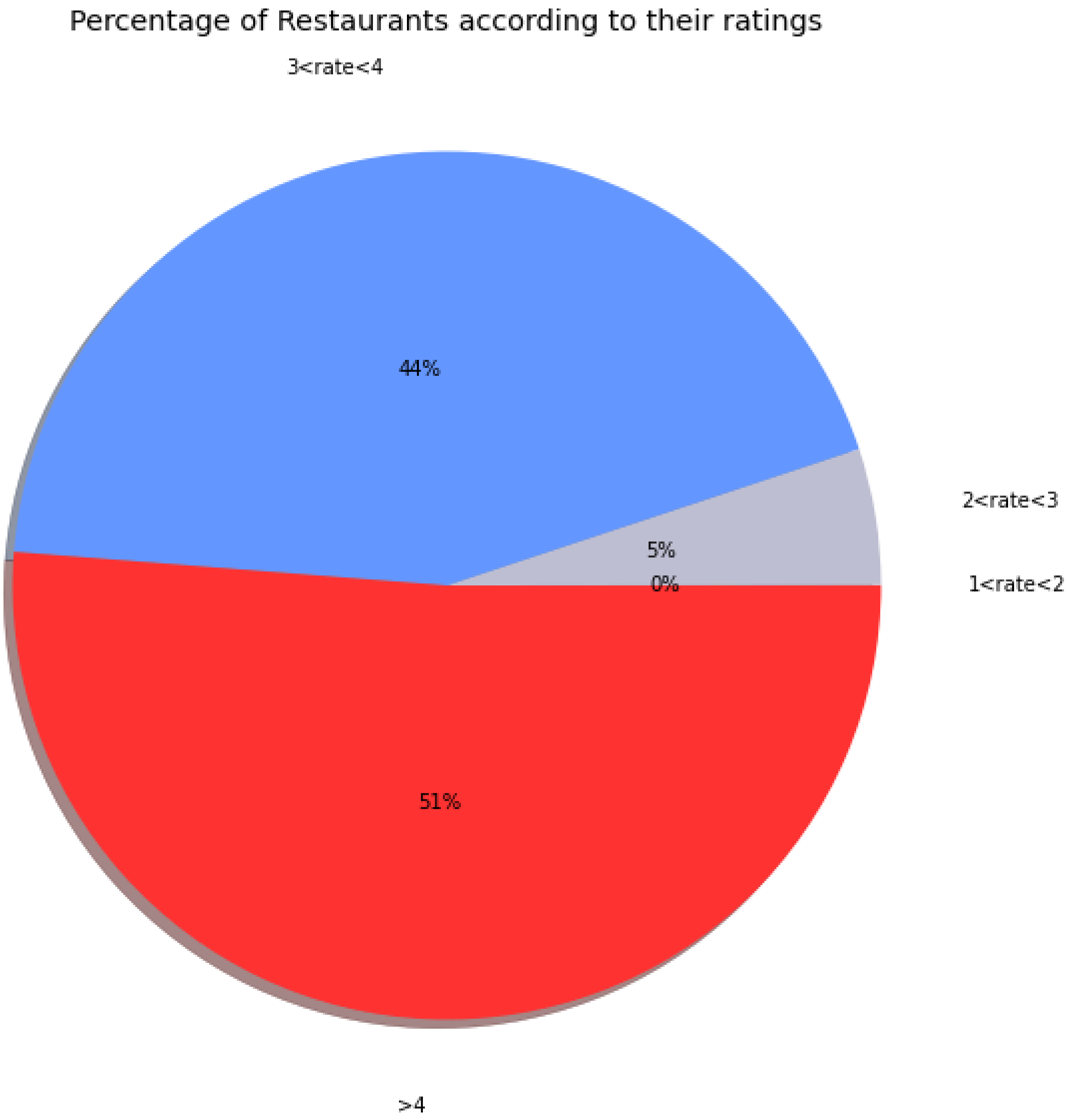
Plotting the counts with the help of pie chart

In [ ]:

```
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=
fig = plt.gcf()
plt.title("Percentage of Restaurants according to their ratings")

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



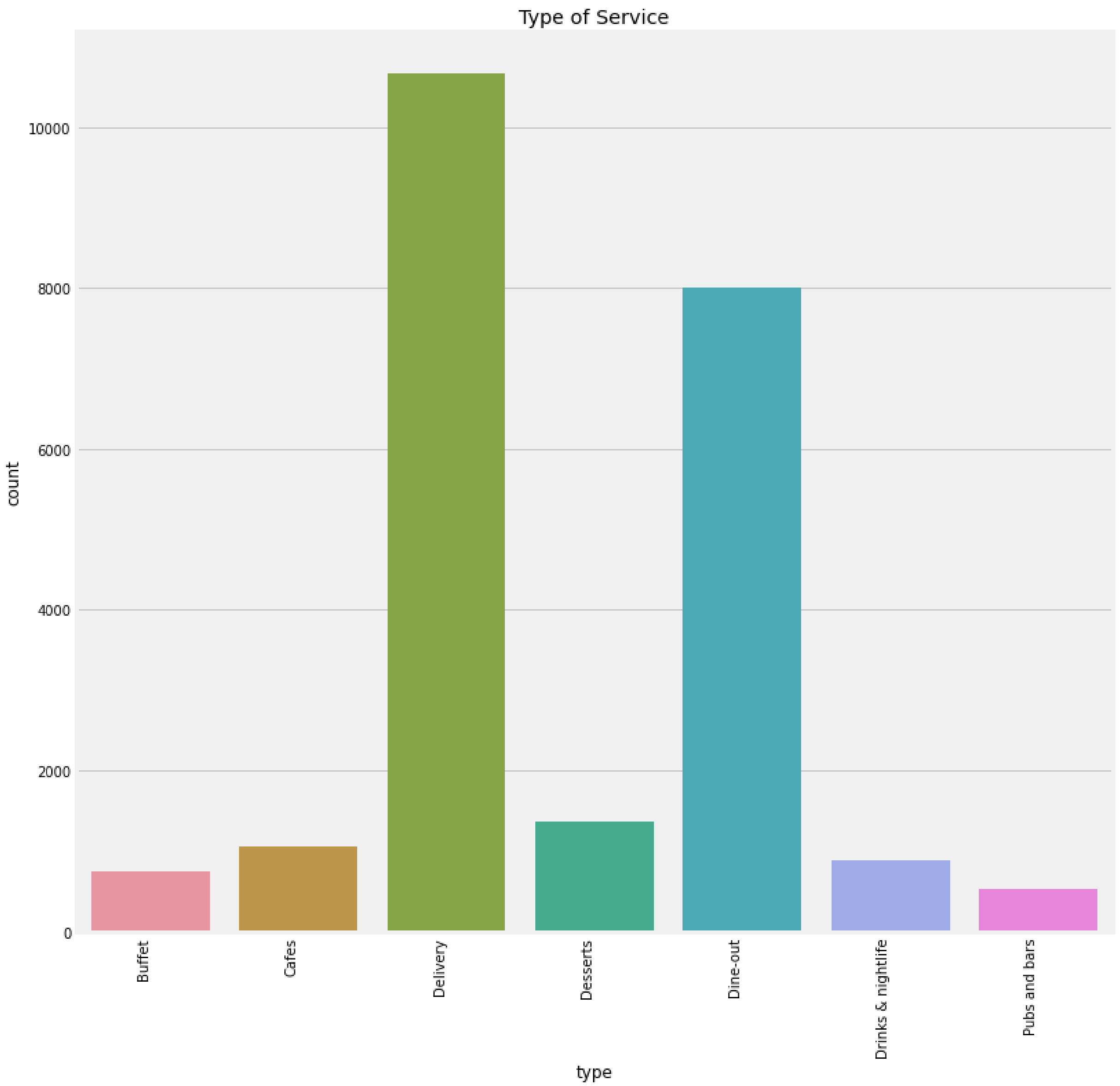
```
In [ ]:
```

Services Types

```
In [ ]: #Types of Services

        sns.countplot(df['type']).set_xticklabels(sns.countplot(df['type']).get_xtic
        fig = plt.gcf()
        fig.set_size_inches(12,12)
        plt.title('Type of Service')
```

```
Out[ ]: Text(0.5, 1.0, 'Type of Service')
```



Here the two main service types are **Delivery** and **Dine-out**

Distribution of Cost of Food for two People

In [ ]:

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

In [ ]:

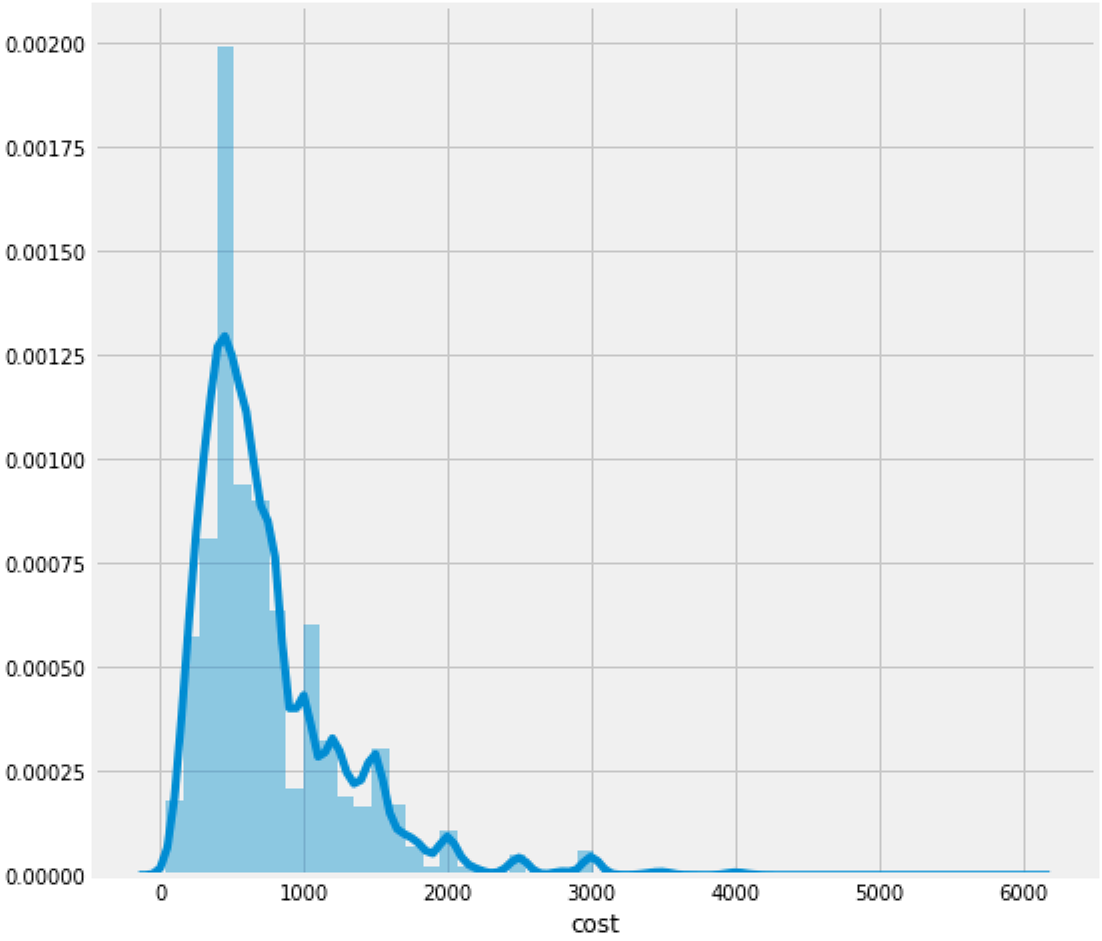
```
trace0=go.Box(y=df['cost'],name="accepting online orders",
              marker = dict(
        color = 'rgb(113, 10, 100)',
    ))
data=[trace0]
layout=go.Layout(title="Box plot of approximate cost",width=800,height=800,y
fig=go.Figure(data=data,layout=layout)
py.iplot(fig)
```

Distribution of charges

In [ ]:

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

```
plt.show()
```



Most Liked Dishes

In [ ]:

```
#re=regular expression (use for splitting words)

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 [ ]:

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

In [ ]:

```
df.index
```

In [ ]:

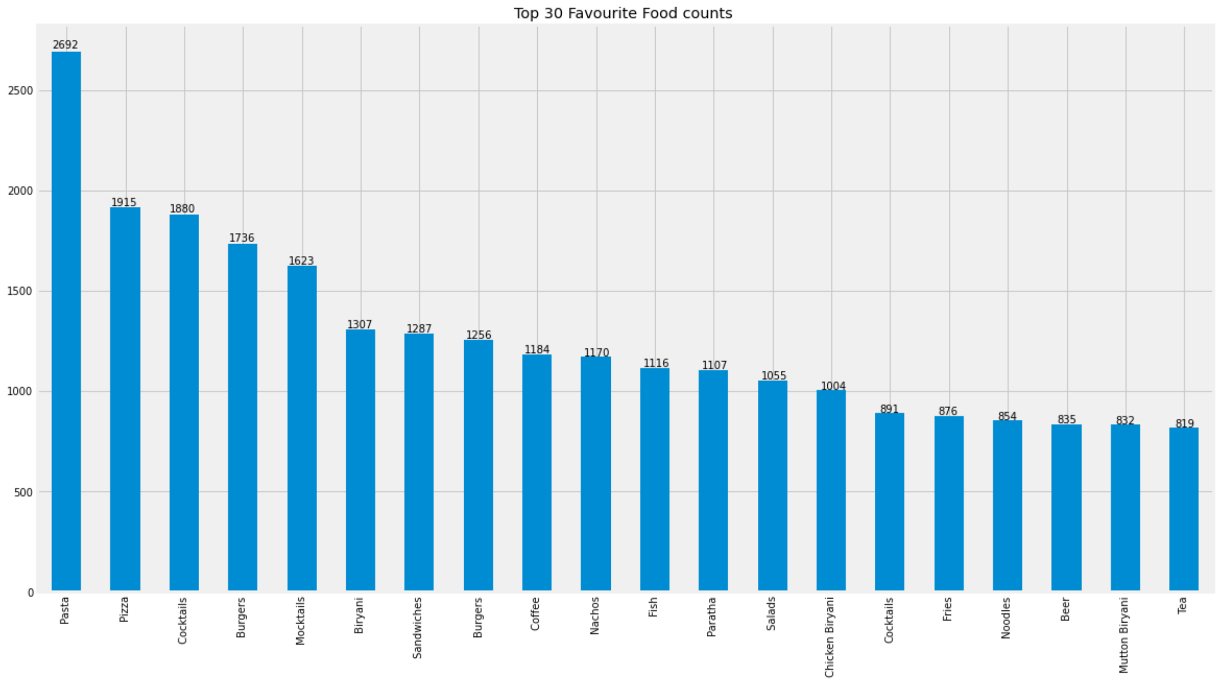
```
print("Count of Most liked dishes in Bangalore")
favourite_food = pd.Series(likes).value_counts()
favourite_food.head(30)
```

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

```
Out[ ]:  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 [ ]: ax = favourite_food.nlargest(n=20, keep='first').plot(kind='bar',figsize=(18

        for i in ax.patches:
            ax.annotate(str(i.get_height()), (i.get_x() * 1.005, i.get_height() * 1.
```



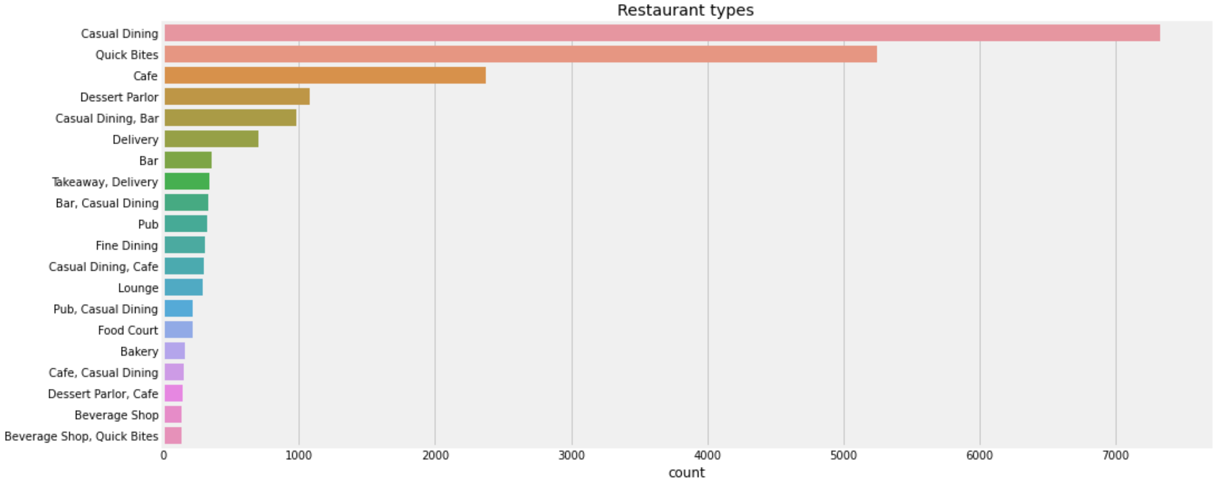
We can infer from the analysis that the 5 most liked dishes are **Pasta**,**Pizza**,**Cocktails**,**Burgers**,and **Mocktails**

```
In [ ]:
```

Restaurant and their counts

```
In [ ]: 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[ ]: Text(0.5, 0, 'count')
```



Casual Dining, **Quick Bites** and **Cafe** are the 3 most common types of Restaurants in Banglore

In [ ]:

```

```

Most famous Restaurants

In [ ]:

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

Building Our Model

In [ ]:

```
df.head()
```

Out[ ]:

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

In [ ]:

```

```

Convert the online categorical variables into a numeric format

In [ ]:

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

In [ ]:

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

Out[ ]:

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

In [ ]:

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

change the string categorical into to a categorical int

In [ ]:

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

In [ ]:

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

In [ ]:

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

Out[ ]:

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

In [ ]:

```

```

Label encode the categorical variables to make it easier to build algorithm

In [ ]:

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

In [ ]:

```
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 [ ]:

```
df.head()
```

Out[ ]:

	address	name	online_order	book_table	rate	votes	location	res
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	1	1	4.1	775	1	
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	1	0	4.1	787	1	
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	1	0	3.8	918	1	
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	0	0	3.7	88	1	
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	0	0	3.8	166	4	

In [ ]:

```
my_data=df.iloc[:,[2,3,4,5,6,7,9,10,12]]
my_data.to_csv('Zomato_df.csv')
```

In [ ]:

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

Out[ ]:

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

In [ ]:

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

Out[ ]:
```
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 [ ]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state
```

Linear Regression

In [ ]:
```
lr_model=LinearRegression()
lr_model.fit(x_train,y_train)
```

Out[ ]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fa
lse)
```

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

Out[ ]:
```
0.2281882852296705
```

Random Forest

In [ ]:
```
#from sklearn.tree import DecisionTreeRegressor
```

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor
RF_Model=RandomForestRegressor(n_estimators=650,random_state=245,min_samples
RF_Model.fit(x_train,y_train)
y_predict=RF_Model.predict(x_test)
r2_score(y_test,y_predict)
```

Out[ ]:
```
0.8812525999137639
```

ExtraTree Regressor

In [ ]:

```
#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[ ]:

```
0.9326010735721576
```

Extra Tree Regressor gives us the best model

Pickle: https://bit.ly/38MGdgn

In [ ]:

```
Use pickle to save our model so that we can use it later

import pickle
# Saving model to disk
pickle.dump(ET_Model, open('model.pkl','wb'))
model=pickle.load(open('model.pkl','rb'))
```

In [ ]:

```
```

This notebook was converted with convert.ploomber.io