Project: EDA on Zomato Restaurant Rating & Predict the Zomato Restaurant Ratings using Machine Learning

Submitted By

ADAIKKAPPAN A (EBEON0522601032)

ABSTRACT

Zomato was launched in 2010. Zomato's technology platform connects customers, restaurant partners and delivery partners, serving their multiple needs. Customers use the Zomato platform to search and discover restaurants, read and write customer-generated reviews and view and upload photos, order food delivery, book a table and make payments while dining out at restaurants. On the other hand, Zomato provides restaurant partners with industry-specific marketing tools which enable them to engage and acquire customers to grow their business while also providing a reliable and efficient last-mile delivery service. Zomato also operates a one-stop procurement solution, Hyperpure, which supplies high quality ingredients and kitchen products to restaurant partners. Zomato also provides our delivery partners with transparent and flexible earning opportunities. Now I am going to analyze the Bangalore Zomato Restaurants Dataset and predict the ratings of a restaurants by using machine learning algorithms.

CONTENTS

TABLE OF CONTENTS

Chapter 1

Introduction

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 various features a user would look into regarding a restaurant. We will be considering Bangalore City in this analysis.

Problem Statement:

The basic idea of analysing the Zomato dataset is to get a fair idea about the factors affecting the establishment of different types of restaurants 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 hasn't been saturated yet and the demand is increasing day by day. In spite 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 e.g., Jain, Marwaris, Gujaratis who are mostly vegetarian. This kind of analysis can be done using the data.

Study of Existing Systems:

1. EDA: Bangalore restaurants:

- Author: **FF RANKUSHA -** In this project, she did the analysis part in three different stages, that is Univariate Analysis, Bivariate Analysis & Automated Analysis.

2. Zomato [EDA - FE - Model Building]:

- Author-**ADITYA RAWAT -** In this project, he did the analysis part and applied the machine learning algorithms to predict the ratings of a restaurant.

Identification of gaps in existing systems:

1. EDA: Bangalore restaurants:

- Author: **FF RANKUSHA -** Analysis Part should be more in detail.

2. Zomato [EDA - FE - Model Building]:

- Author-**ADITYA RAWAT -** In this project, he applied only two machine learning algorithms to predict the ratings of a restaurant.

Discussed on proposed solution:

1. EDA: Bangalore restaurants:

- Author: **FF RANKUSHA -** Analysis Part should be more in detail like which service types are more popular in Bangalore and which is the most popular cuisines available in Bangalore and check the Top Rating Restaurants in Bangalore having online order or not and check table booking option available or not and check most liked dishes available or not.

2. Zomato [EDA - FE - Model Building]:

- Author: **ADITYA RAWAT –** We can apply additional two more Machine learning algorithms Random Forest Regressor and Linear Regression and compare the best model by using the accuracy of each model and we can generate the Feature Importance to understand which feature is used to make key decisions.

Tools/Technology used to implement proposed solution:

- Python
- Pandas
- Numpy
- Matplotlib
- Seaborn
- Plotly
- Sklearn
- Jupyter Notebook

Chapter 2

Features & Predictor

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

Note

- Total - 17 columns
- Numerical – 3 - Continuous: Which is quantitative data that can be measured.
- String – 14 - Ordinal Data: Categorical data that has an order to it.

Chapter 3
Methodology

Data Cleaning and Pre-processing:

The datasets which were collected from Zomato Restaurant Rating Dataset from Kaggle website contain unfiltered data which must be filtered before the final data set can be used to do analysis. Also, data has some categorical variables which must be modified into numerical values for which we used Panda’s library of Python. In data cleaning step, first we checked whether there are any missing or junk values in the dataset for which we used the is null () function.

Machine Learning Algorithms:

a) ExtraTree Regressor:

An extra-trees regressor, this class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

b) Random Forest Regressor:

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

c) Decision Tree Regressor:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

d) Linear Regression:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used. There are many names for a regression's dependent variable. It may be called an outcome variable, criterion variable, endogenous variable, or regressand. The independent variables can be called exogenous variables, predictor variables, or regressors.

Implementation Steps:

As we already discussed in the methodology section about some of the implementation details. So, the language used in this project is Python programming. We're running python code in anaconda navigator's Jupyter notebook. Jupyter notebook is much faster than Python IDE tools like PyCharm or Visual studio for implementing ML algorithms. The advantage of Jupyter notebook is that while writing code, it's really helpful for Data visualization and plotting some graphs like histogram and heatmap of correlated matrices. Let's revise implementation steps: a) Dataset collection. b) Importing Libraries: NumPy, Pandas, Matplotlib, Seaborn and Sklearn libraries were used. c) Exploratory data analysis: For getting more insights about data. d) Data cleaning and pre-processing: Checked for null and junk values using isnull() and isna().sum() functions of python. In Pre-processing phase, we did feature engineering on our dataset. As we converted categorical variables into numerical variables using function of Pandas library. All our datasets contains some categorical variables.

LIBRARIES:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing import LabelEncoder
import re
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import  ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor

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

Reading the Dataset and assigning that to the variable df:

df=pd.read_csv("zomato.csv")

Access the first 5 rows of a dataframe:

df.head()

	url	address	name	online_order	book_table	rate	votes	phone
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
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
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
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
4	https://www.zomato.com/bangalore/grand-village	10, 3rd Floor, Lakshmi Associates	Grand Village	No	No	3.8/5	166	+91 8026612447\r\n+91

Access the last 5 rows of a dataframe:

df.tail()

51712	https://www.zomato.com/bangalore/best-brews-fo...	Four Points by Sheraton Bengaluru, 43/3, White...	Best Brews - Four Points by Sheraton Bengaluru...	No	No	3.6 /5	27	080 40301477	V
51713	https://www.zomato.com/bangalore/vinod-bar-and...	Number 10, Garudachar Palya, Mahadevapura, Whi...	Vinod Bar And Restaurant	No	No	NaN	0	+91 8197675843	V
51714	https://www.zomato.com/bangalore/plunge-sherat...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Plunge - Sheraton Grand Bengaluru Whitefield H...	No	No	NaN	0	NaN	V
51715	https://www.zomato.com/bangalore/chime-sherato...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Chime - Sheraton Grand Bengaluru Whitefield Ho...	No	Yes	4.3 /5	236	080 49652769	V
51716	https://www.zomato.com/bangalore/the-nest-the-...	ITPL Main Road, KIADB Export Promotion Industr...	The Nest - The Den Bengaluru	No	No	3.4 /5	13	+91 8071117272	V

Prints information about the DataFrame:

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23248 entries, 0 to 23247
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   address        23248 non-null  object
 1   name           23248 non-null  object
 2   online_order   23248 non-null  int64
 3   book_table     23248 non-null  int64
 4   rate           23248 non-null  float64
 5   votes          23248 non-null  int64
 6   location       23248 non-null  int32
 7   rest_type      23248 non-null  int32
 8   dish_liked     23248 non-null  object
 9   cuisines       23248 non-null  int32
 10  cost           23248 non-null  float64
 11  reviews_list   23248 non-null  object
 12  menu_item      23248 non-null  int32
 13  type           23248 non-null  object
 14  city           23248 non-null  object
dtypes: float64(2), int32(4), int64(3), object(6)
memory usage: 2.3+ MB
```

Dimension of the Dataset:

```
In [5]:  df.shape
Out[5]: (51717, 17)
```

Checking the data type for each column:

```
In [6]: df.dtypes
```

```
Out[6]: 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
```

Delete the Unnnecessary Columns:

df=df.drop(['url','phone'],axis=1)

Checking for duplicate values:

```
In [8]: df.duplicated().sum()
```

```
Out[8]: 43
```

Drop the duplicate values:

```
In [9]: df.drop_duplicates(inplace=True)
```

```
In [10]: df.duplicated().sum() #Make sure that the duplicate values are droped
```

```
Out[10]: 0
```

Checking for null values:

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
address                         0
name                            0
online_order                    0
book_table                      0
rate                         7767
votes                           0
location                       21
rest_type                     227
dish_liked                  28047
cuisines                       45
approx_cost(for two people)   345
reviews_list                    0
menu_item                       0
listed_in(type)                 0
listed_in(city)                 0
dtype: int64
```

Drop the null values:

In [12]:
```
df.dropna(how='any',inplace=True)
df.isnull().sum()
```

Out[12]:
```
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 the columns appropriately:

```
In [13]: df.columns
```

```
Out[13]: 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 [14]: df = df.rename(columns={'approx_cost(for two people)':'cost','listed_in(type)':'type',
                                 'listed_in(city)':'city'})
```

Replace the comma (',') from cost:

```
In [16]: df['cost'].unique()
```

```
Out[16]: 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 [17]: df['cost'] = df['cost'].apply(lambda x: x.replace(',',''))
         df['cost'].unique()
```

```
Out[17]: array(['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'], dtype=object)
```

Removing '/5' from Rates:

```
In [18]: df['rate'].unique()
```

```
Out[18]: 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 [19]: df = df.loc[df.rate !='NEW'] #getting rid of "NEW"
```

```
In [20]: df['rate'] = df['rate'].apply(lambda x: x.replace('/5',''))
         df['rate'].unique()
```

```
Out[20]: 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)
```

Convert the cost column datatype to float:

```
In [21]: df['cost'] = df['cost'].astype(float)
```

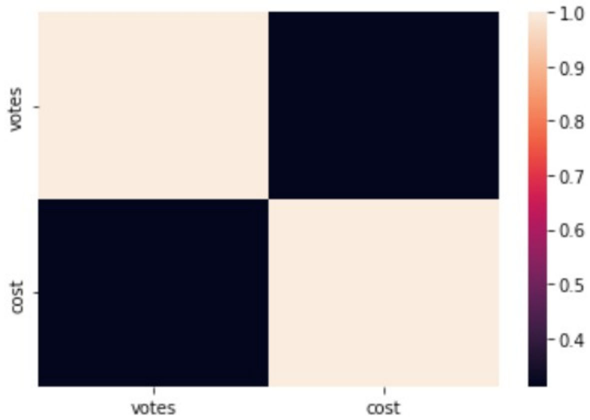
Correlation:

```
In [22]: df.corr()
```

Out[22]:

	votes	cost
votes	1.000000	0.311097
cost	0.311097	1.000000

```
In [23]: sns.heatmap(df.corr())
         plt.show()
```



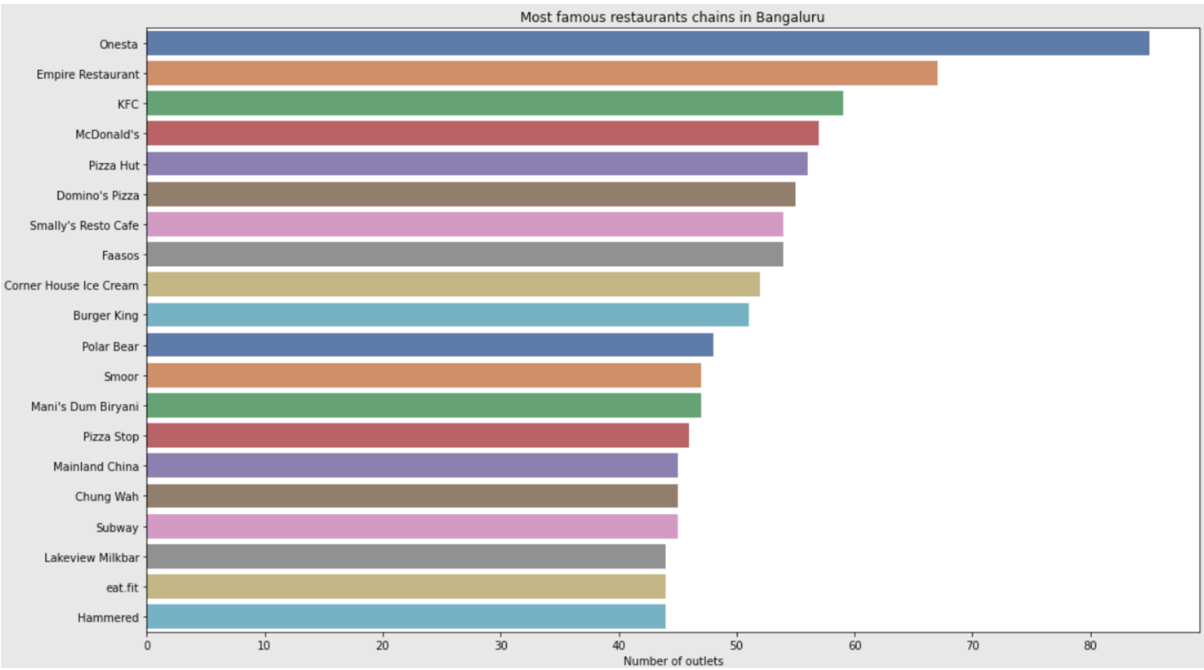
Observation:

- Votes are highly correlated with cost
- Cost is highly correlated with votes

Most famous restaurants chains in Bangaluru:

In [24]:
```
famous_restaurants=df['name'].value_counts()[:20]
```

In [25]:
```
plt.figure(figsize=(17,10))
sns.barplot(x=famous_restaurants,y=famous_restaurants.index,palette='deep')
plt.title("Most famous restaurants chains in Bangaluru")
plt.xlabel("Number of outlets")
plt.show()
```



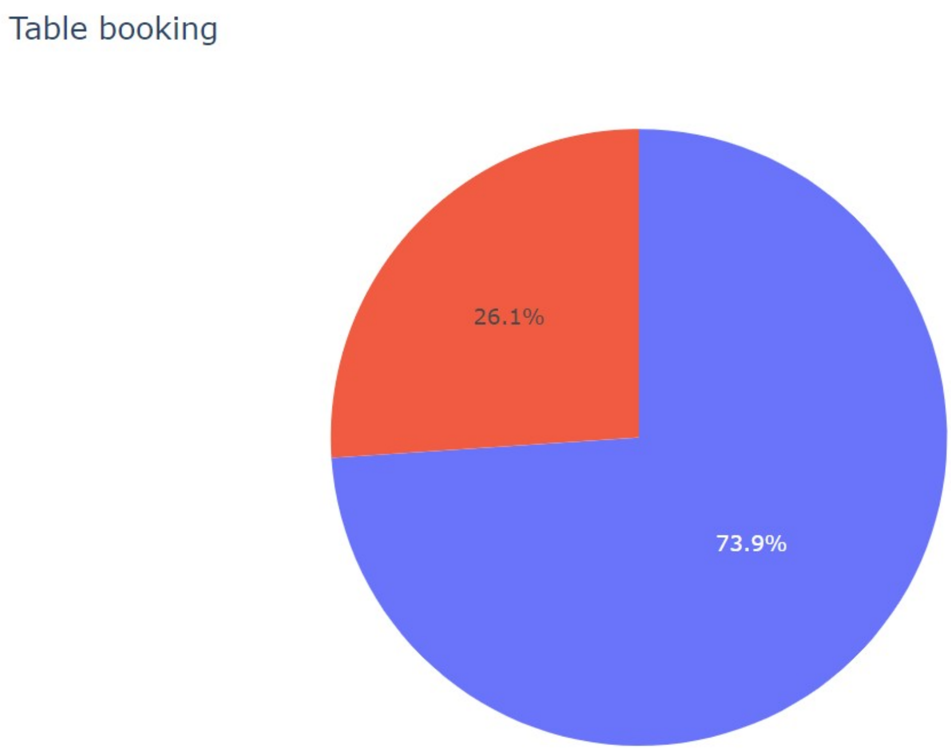
Observation

The Top five famous restaurants in Bangalore are

- Onesta
- Empire Restaurant
- KFC
- McDonald's
- Pizza Hut

Whether restaurant offer Table booking or not:

```
In [26]:  book_table=df['book_table'].value_counts()
          book_table

Out[26]:  No      17191
          Yes      6057
          Name: book_table, dtype: int64

In [27]:  fig = px.pie(labels=book_table.index,values=book_table,title = "Table booking")
          fig.show()
```

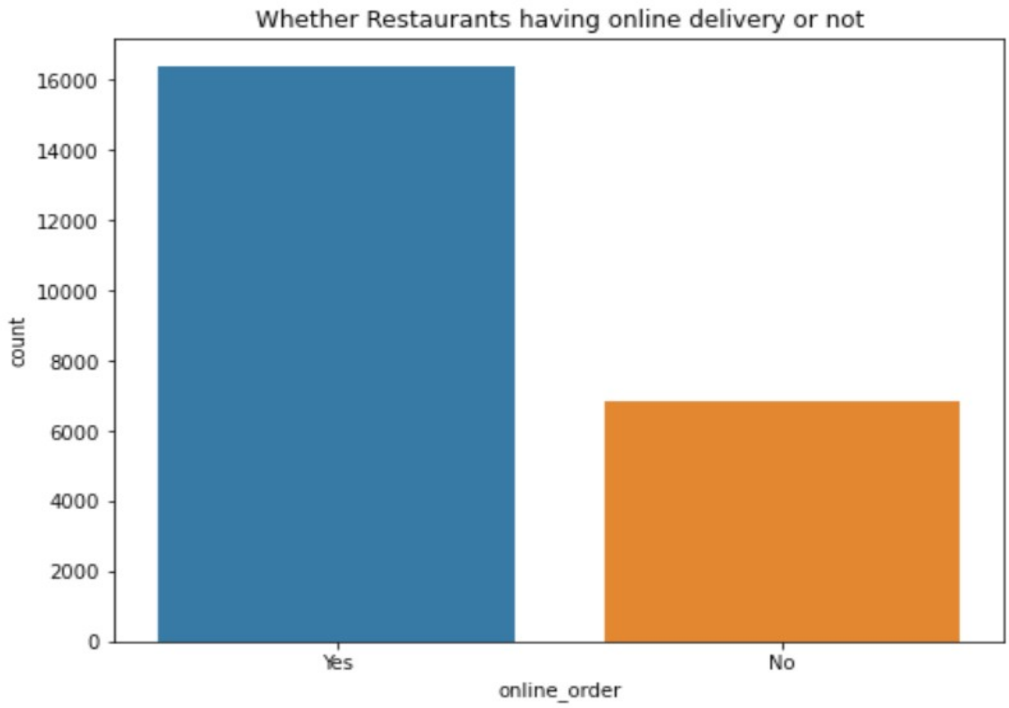


Observation

- 73.9% of the Restaurants do not offer table booking
- 26.1% of the Restaurants offer table booking

Whether Restaurants having online delivery or not:

```
In [28]:  plt.figure(figsize=(8,6))
          sns.countplot(df['online_order'])

          plt.title("Whether Restaurants having online delivery or not")
          plt.show()
```

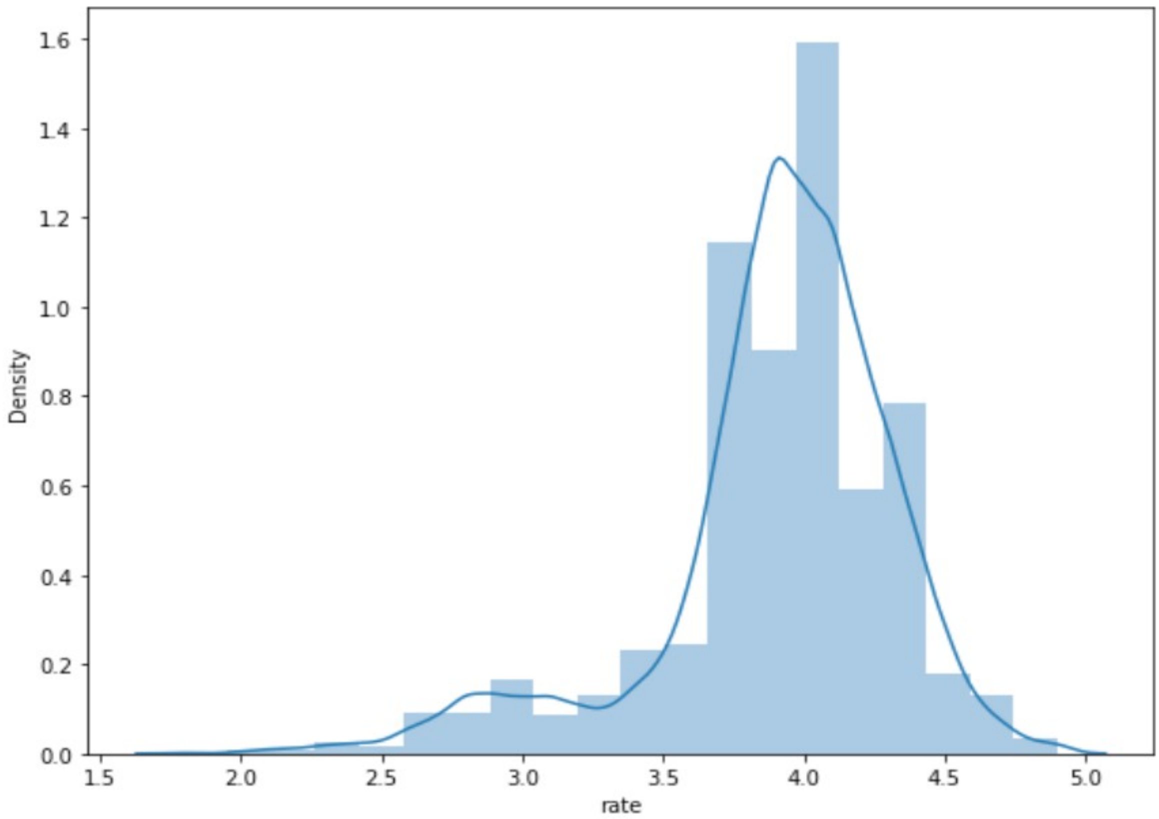


<u>Observation:</u>

- Most of the Restaurants offer option for online order and delivery

<u>Rating Distributions:</u>

```
In [29]:  plt.figure(figsize=(9,7))
          sns.distplot(df['rate'],bins=20)
          plt.show()
```



Observation:

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

Checking the count of ratings as between "1 and 2", "2 and 3", "3 and 4", and "4 and 5":

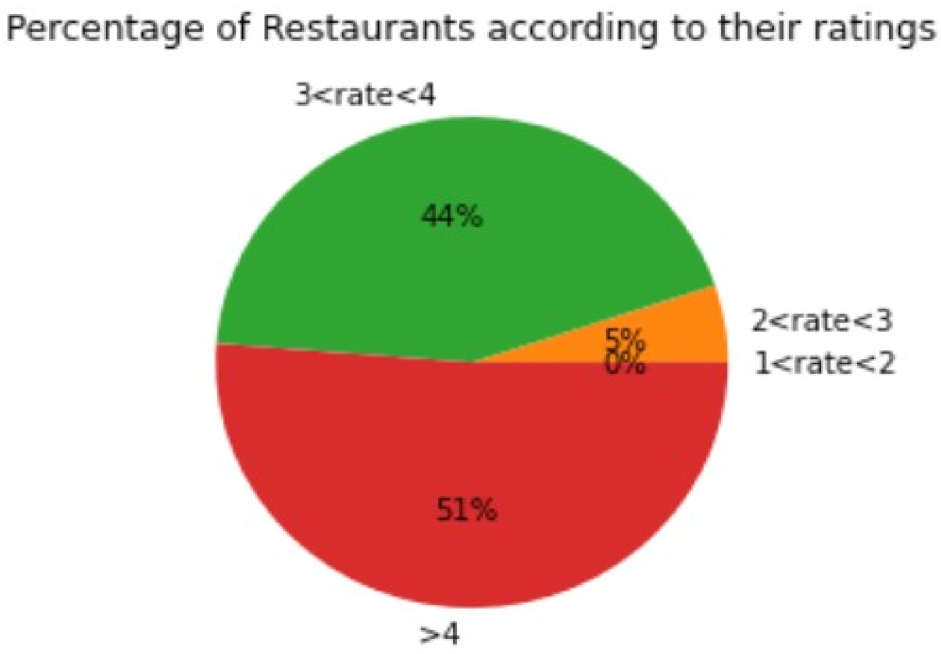
```
In [30]:  df['rate'].unique()

Out[30]: 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 [31]:  df['rate'].min()

Out[31]: '1.8'

In [32]:  df['rate'].max()

Out[32]: '4.9 '

In [33]:  #Convert the Rate column datatype to float
          df['rate']=df['rate'].astype(float)
```

Plotting the counts with the help of pie chart:

In [38]:
```
split_rating=[((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()
        ]
```

In [39]:
```
labels=['1<rate<2','2<rate<3','3<rate<4','>4']
```

In [40]:
```
plt.pie(split_rating, labels=labels, autopct='%1.0f%%')
plt.title("Percentage of Restaurants according to their ratings")
plt.show()
```

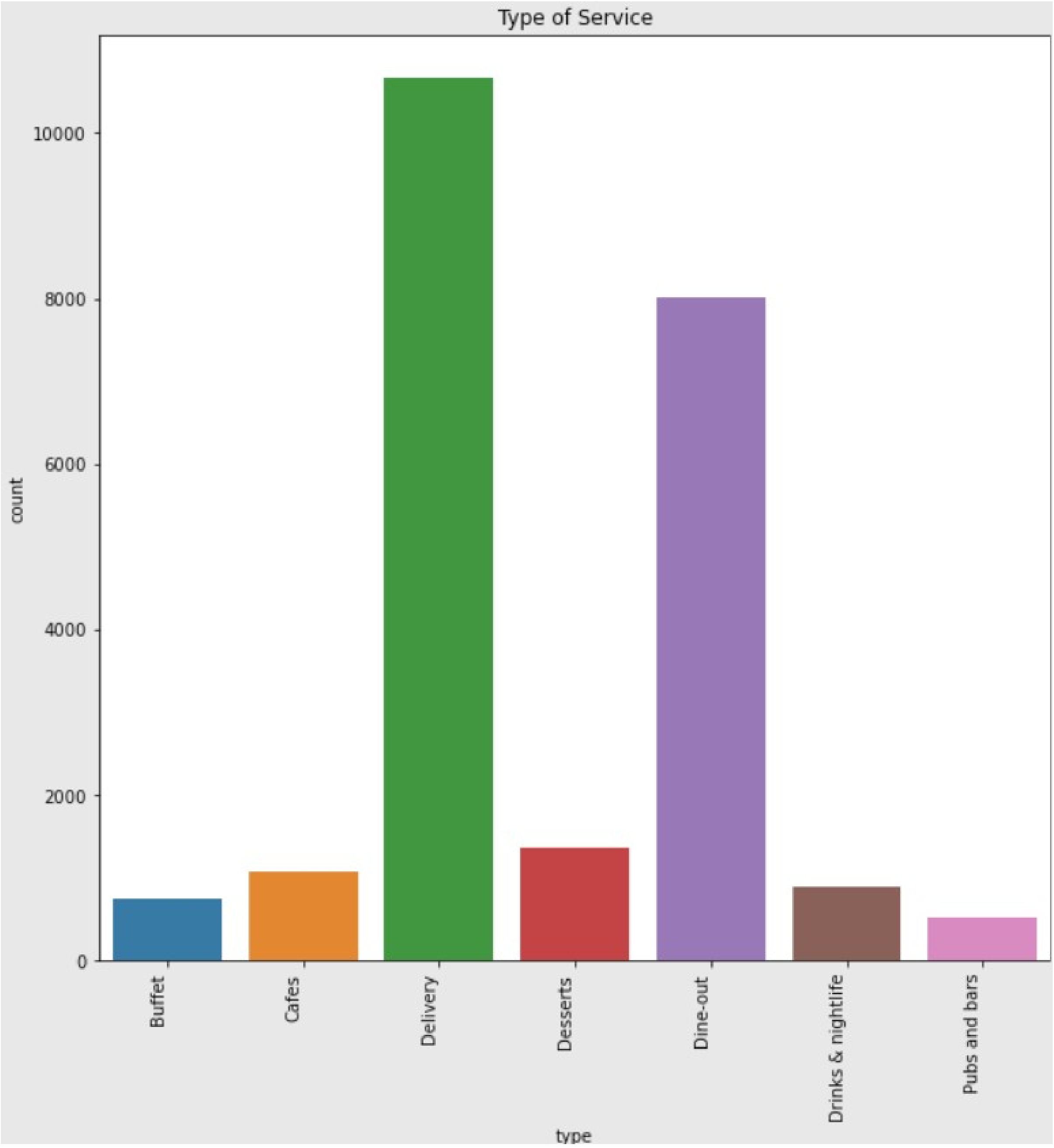


Observation:

- 51% of Restaurants ratings are greater than 4
- 44% of Restaurants ratings between 3 to 4
- 5% of Restaurants ratings between 2 to 3

Services Types:

```
plt.figure(figsize=(10,10))
sns.countplot(df['type']).set_xticklabels(sns.countplot(df['type']).get_xticklabels(), rotation=90
fig = plt.gcf()
plt.title('Type of Service')
plt.show()
```



Observation:

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

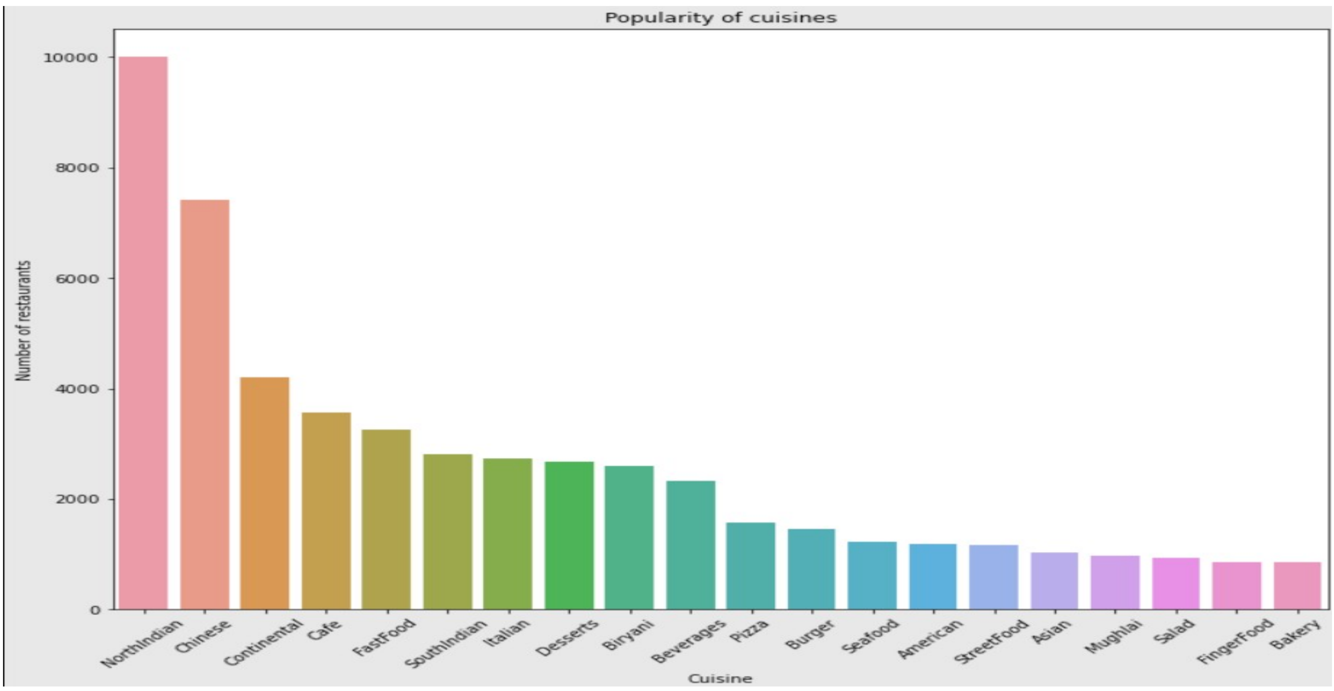
Cuisines:

In [42]:
```
df['cuisines'] = df['cuisines'].str.replace(' ', '')
all_cuisines = pd.unique(df['cuisines'].str.split(',', expand=True).stack())

# Count number of restaurants based on the cuisines
no_of_restaurants = []
for cuisine in all_cuisines:
    no_of_restaurants.append(df['cuisines'].dropna()
                             .apply(lambda x: 1 if cuisine in x else 0).sum())

# Make a dataframe using the two lists
df_cuisines = pd.DataFrame(list(zip(all_cuisines, no_of_restaurants)), columns=['All_cuisines', 'N
df_cuisines = df_cuisines.sort_values(by='No_of_restaurants', ascending=False)
df_cuisines.head()
```

Out[42]:

	All_cuisines	No_of_restaurants
0	NorthIndian	10011
2	Chinese	7422
10	Continental	4204
4	Cafe	3572
13	FastFood	3249

In [43]:
```
plt.figure(figsize=(12,10))
bar_plot = sns.barplot(x=df_cuisines['All_cuisines'][:20], y=df_cuisines['No_of_restaurants'][:20]

bar_plot.set_xticklabels(bar_plot.get_xticklabels(),
                         rotation=45)
plt.title('Popularity of cuisines')
plt.ylabel('Number of restaurants')
plt.xlabel('Cuisine')
plt.show()
```



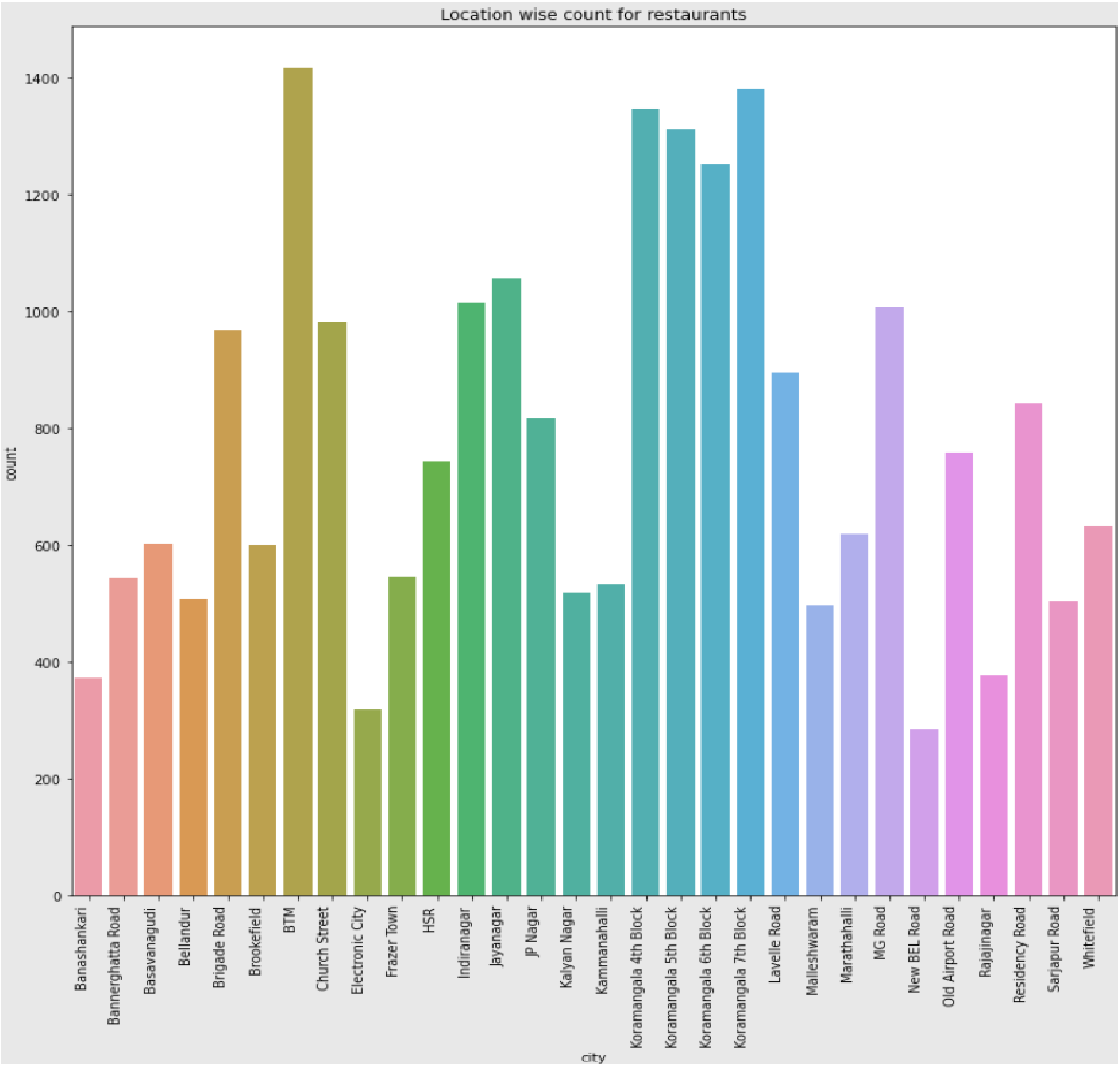
Observation:

- Here the three popularity cuisines are NorthIndian,Chinese and Continental

Location wise count for restaurants:

In [44]:

```
sns.countplot(df['city'])
sns.countplot(df['city']).set_xticklabels(sns.countplot(df['city']).get_xticklabels(), rotation=9
fig = plt.gcf()
fig.set_size_inches(13,13)
plt.title('Location wise count for restaurants')
plt.show()
```



Observation:

- We can infer from the analysis that the top 3 Locations are BTM, Koramangala 7th Block and Koramangala 4th Block

Most Liked Dishes:

In [45]:

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

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 [46]:

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

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

Out[46]:

```
 Pasta              2692
 Pizza              1915
 Cocktails          1880
 Burgers            1736
 Mocktails          1623
 Biryani            1307
 Sandwiches         1287
Burgers             1256
 Coffee             1184
```

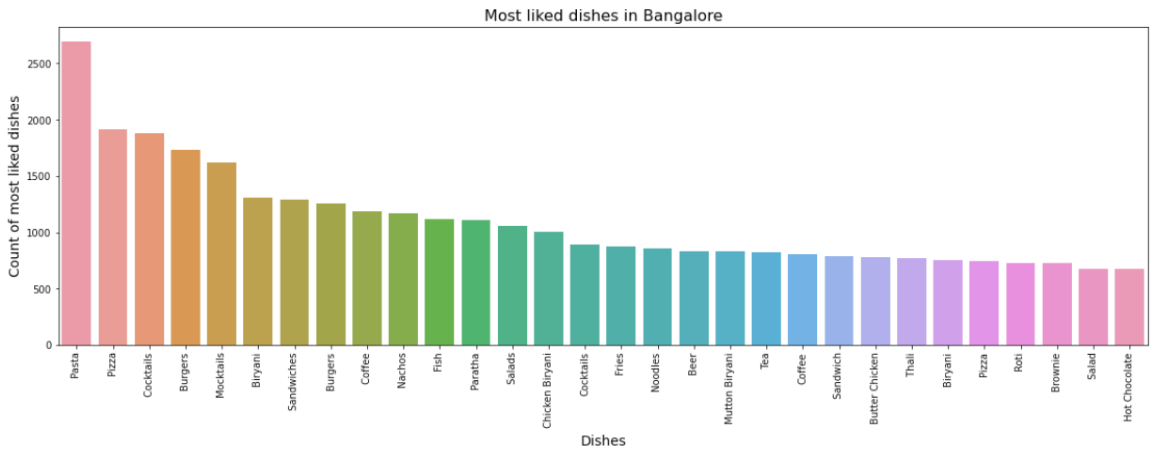
In [47]:

```
favourite_food=favourite_food.reset_index()
favourite_food.rename(columns = {'index':'Dishes',0:'Count_of_most_liked_dish'}, inplace = True)
favourite_food
```

Out[47]:

	Dishes	Count_of_most_liked_dish
0	Pasta	2692
1	Pizza	1915
2	Cocktails	1880
3	Burgers	1736
4	Mocktails	1623
5	Biryani	1307
6	Sandwiches	1287
7	Burgers	1256
8	Coffee	1184

```
In [48]:
plt.figure(figsize=(20,6))
sns.barplot(x=favourite_food['Dishes'],y=favourite_food['Count_of_most_liked_dish'])
plt.title('Most liked dishes in Bangalore',size=16)
plt.xticks(rotation=90)
plt.xlabel('Dishes',size=14)
plt.ylabel('Count of most liked dishes',size=14)
plt.show()
```

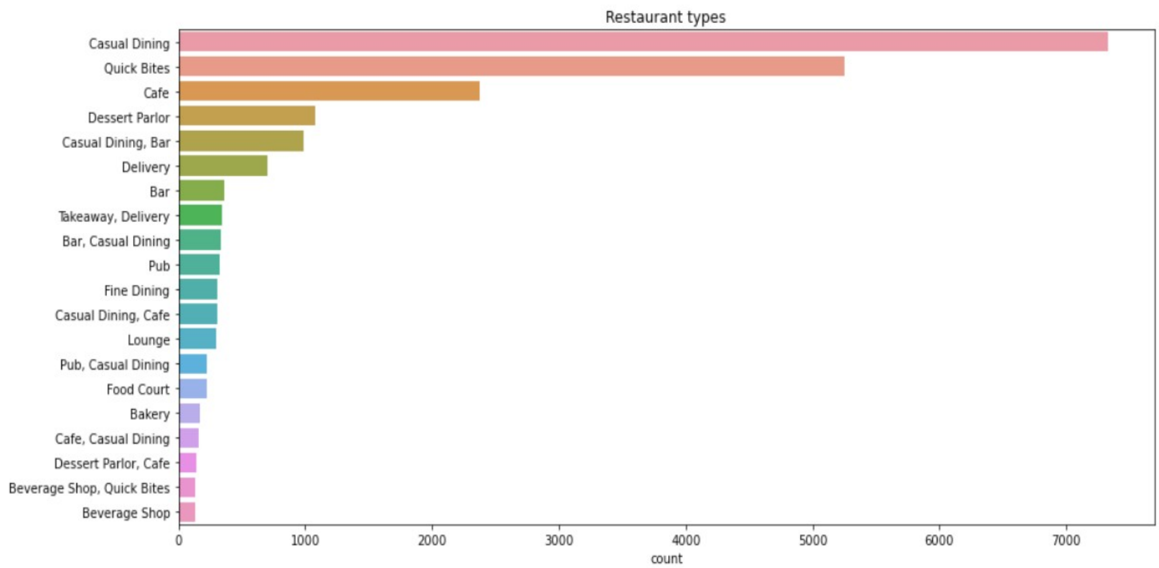


Observation:

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

Restaurants types and their counts:

```
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")
plt.show()
```



Observation:

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

Top Rating Restuarants in Bangalore:

In [50]:
```
top_rating_res=df.groupby(['name'])['rate'].sum().sort_values(ascending=False)[:30]
top_rating_res
```

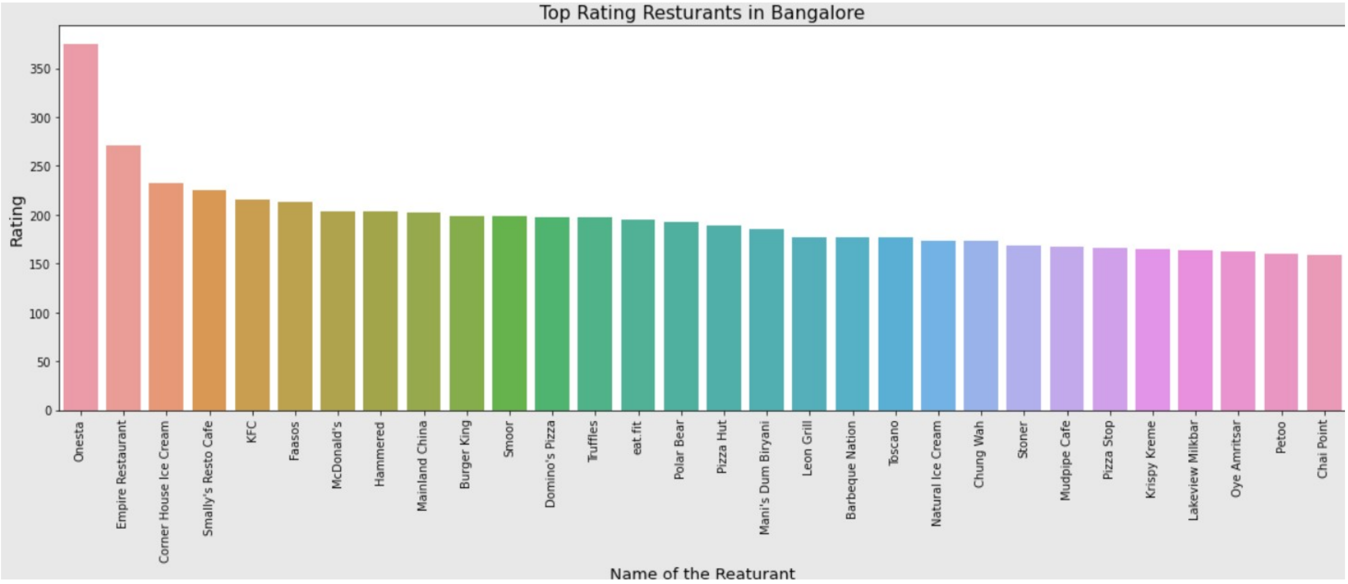
Out[50]:
```
name
Onesta                    374.9
Empire Restaurant         270.9
Corner House Ice Cream    232.2
Smally's Resto Cafe       225.3
KFC                       215.3
Faasos                    212.8
McDonald's                203.7
Hammered                  203.5
Mainland China            202.3
Burger King               198.6
Smoor                     198.1
Domino's Pizza            197.7
Truffles                  197.6
eat.fit                   194.4
Polar Bear                192.3
Pizza Hut                 189.3
Mani's Dum Biryani        185.4
Leon Grill                176.8
Barbeque Nation           176.6
Toscano                   176.5
Natural Ice Cream         173.3
```

In [51]:
```
top_rating_res=top_rating_res.reset_index()
top_rating_res
```

Out[51]:

	name	rate
0	Onesta	374.9
1	Empire Restaurant	270.9
2	Corner House Ice Cream	232.2
3	Smally's Resto Cafe	225.3
4	KFC	215.3
5	Faasos	212.8
6	McDonald's	203.7
7	Hammered	203.5
8	Mainland China	202.3
9	Burger King	198.6
10	Smoor	198.1
11	Domino's Pizza	197.7
12	Truffles	197.6
13	eat.fit	194.4

```
In [52]: plt.figure(figsize=(20,6))
         sns.barplot(x=top_rating_res['name'],y=top_rating_res['rate'])
         plt.title('Top Rating Resturants in Bangalore',size=16)
         plt.xticks(rotation=90)
         plt.xlabel('Name of the Reaturant',size=14)
         plt.ylabel('Rating',size=14)
         plt.show()
```



<u>Observation:</u>

- We can infer from the analysis that the 5 most top restuarants are Onesta, Empire Restaurant, Corner House Ice Cream, Smally's Resto Cafe, KFC

<u>Check Top Rating Restaurants having online order:</u>

```
In [53]: res_online_order=df.groupby(['name','online_order'])['rate'].sum().sort_values(ascending=False)
         res_online_order

Out[53]: name                online_order
         Onesta              Yes             374.9
         Empire Restaurant   Yes             263.2
         Smally's Resto Cafe Yes             225.3
         KFC                 Yes             211.4
         Faasos              Yes             208.7
                                              ...
         Nalaas Aapakadai    No                2.7
         Delight Food        Yes               2.6
         Swad                Yes               2.6
         Gongura             No                2.6
         Night Spoon         No                2.4
         Name: rate, Length: 3342, dtype: float64
```

In [54]:
```
res_online_order=res_online_order.reset_index()
res_online_order
```

Out[54]:

	name	online_order	rate
0	Onesta	Yes	374.9
1	Empire Restaurant	Yes	263.2
2	Smally's Resto Cafe	Yes	225.3
3	KFC	Yes	211.4
4	Faasos	Yes	208.7
...	...	...	...
3337	Nalaas Aapakadai	No	2.7
3338	Delight Food	Yes	2.6
3339	Swad	Yes	2.6
3340	Gongura	No	2.6
3341	Night Spoon	No	2.4

3342 rows × 3 columns

In [55]:
```
top_rating_res=["Onesta","Empire Restaurant","Corner House Ice Cream","Smally's Resto Cafe","KFC"]
indx=[]
for i in range(len(res_online_order.name)):
    for rest in top_rating_res:
        if rest == res_online_order.name[i]:
            indx.append(i)
```

In [56]:
```
df_rest = pd.DataFrame()
for indexes in indx:
    df_rest = df_rest.append(res_online_order.iloc[indexes])
```

In [57]:
```
df_rest[:5]
```

Out[57]:

	name	online_order	rate
0	Onesta	Yes	374.9
1	Empire Restaurant	Yes	263.2
2	Smally's Resto Cafe	Yes	225.3
3	KFC	Yes	211.4
20	Corner House Ice Cream	Yes	170.7

Observation:

- We can infer from the analysis that the 5 most top restuarants are Onesta, Empire Restaurant, Corner House Ice Cream, Smally's Resto Cafe, KFC having online order.

Check Top Rating Restaurants having table booking option:

In [58]:
```
res_table_book=df.groupby(['name','book_table'])['rate'].sum().sort_values(ascending=False)
res_table_book
```

Out[58]:
```
name                    book_table
Onesta                  Yes           374.9
Empire Restaurant       No            270.9
Corner House Ice Cream  No            232.2
KFC                     No            215.3
Faasos                  No            212.8
                                      ...
Foodie                  No              2.9
Nalaas Aapakadai        No              2.7
Tuk-Tuk                 No              2.7
Gongura                 No              2.6
Delight Food            No              2.6
Name: rate, Length: 3228, dtype: float64
```

In [59]:
```
res_table_book=res_table_book.reset_index()
res_table_book
```

Out[59]:

	name	book_table	rate
0	Onesta	Yes	374.9
1	Empire Restaurant	No	270.9
2	Corner House Ice Cream	No	232.2
3	KFC	No	215.3
4	Faasos	No	212.8
...	...	...	...
3223	Foodie	No	2.9
3224	Nalaas Aapakadai	No	2.7
3225	Tuk-Tuk	No	2.7
3226	Gongura	No	2.6
3227	Delight Food	No	2.6

3228 rows × 3 columns

In [60]:
```python
top_rating_res=["Onesta","Empire Restaurant","Corner House Ice Cream","Smally's Resto Cafe","KFC"]
indx=[]
df_tab_rest = pd.DataFrame()
for i in range(len(res_table_book.name)):
    for rest in top_rating_res:
        if rest == res_table_book.name[i]:
            indx.append(i)


for indexes in indx:
    df_tab_rest = df_tab_rest.append(res_table_book.iloc[indexes])

df_tab_rest[:5]
```

Out[60]:

	name	book_table	rate
0	Onesta	Yes	374.9
1	Empire Restaurant	No	270.9
2	Corner House Ice Cream	No	232.2
3	KFC	No	215.3
61	Smally's Resto Cafe	No	116.6

Observation:

- We can infer from the analysis that the top most restaurant Onesta having Table booking option.

Check Top Rating Restaurants having most liked dishes or not:

For this Analysis part, I have taken each Top Rating Restaurants separately and then check whether the most liked dishes are available or not

In [61]:
```python
res_liked_dish=df.groupby(['name','dish_liked'])['rate'].sum().sort_values(ascending=False)
```

In [62]:
```python
res_liked_dish=res_liked_dish.reset_index()
res_liked_dish
```

Out[62]:

	name	dish_liked	rate
0	Dolci Desserts	Cappuccino, Eclair, Pizza, Coffee, Blueberry C...	107.5
1	Lot Like Crepes	Paneer Peri Peri, Pancakes, Fajitas, Sweet Cre...	92.3
2	Stoner	Burgers, Spinach Pizza, Butter Chicken Pizza, ...	89.9
3	Oh! Calcutta	Fish, Curry Crab, Mocktails, Luchi, Kosha Mang...	88.6
4	Smally's Resto Cafe	Burgers, Pasta, Chocolate Mousse, Potato Wedge...	88.2
...	...	...	...
5507	Biryani Treats	Chicken, Pot Biryani, Mutton Biryani	2.5
5508	Lazeez	Rolls, Phirni, Mutton Biryani, Kadhai Murgh, T...	2.4
5509	American Bites	Burgers, Chicken Fries, Hot Wings	2.4
5510	Mast Kalandar	Lassi, Raita, Chole, Dal Makhani, Gulab Jamun,...	2.4
5511	Night Spoon	Sandwiches, Shawarma, Pasta	2.4

5512 rows × 3 columns

In [63]:

```python
top_rating_res=["Onesta","Empire Restaurant","Corner House Ice Cream","Smally's Resto Cafe","KFC"]
indx=[]
df_liked_rest = pd.DataFrame()
for i in range(len(res_liked_dish.name)):
    for rest in top_rating_res:
        if rest == res_liked_dish.name[i]:
            indx.append(i)


for indexes in indx:
    df_liked_rest = df_liked_rest.append(res_liked_dish.iloc[indexes])
```

Onesta Restaurant:

In [65]:

```python
df_Onesta = pd.DataFrame()
df_Onesta = df_liked_rest.where(df_liked_rest.name == 'Onesta')
df_Onesta=df_Onesta.dropna()

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

print("Count of Most liked dishes in Onesta Restuarant")
favourite_food = pd.Series(likes).value_counts()[:5]

favourite_food=favourite_food.reset_index()
favourite_food.rename(columns = {'index':'Dishes',0:'Count_of_most_liked_dish'}, inplace = True)
favourite_food
```

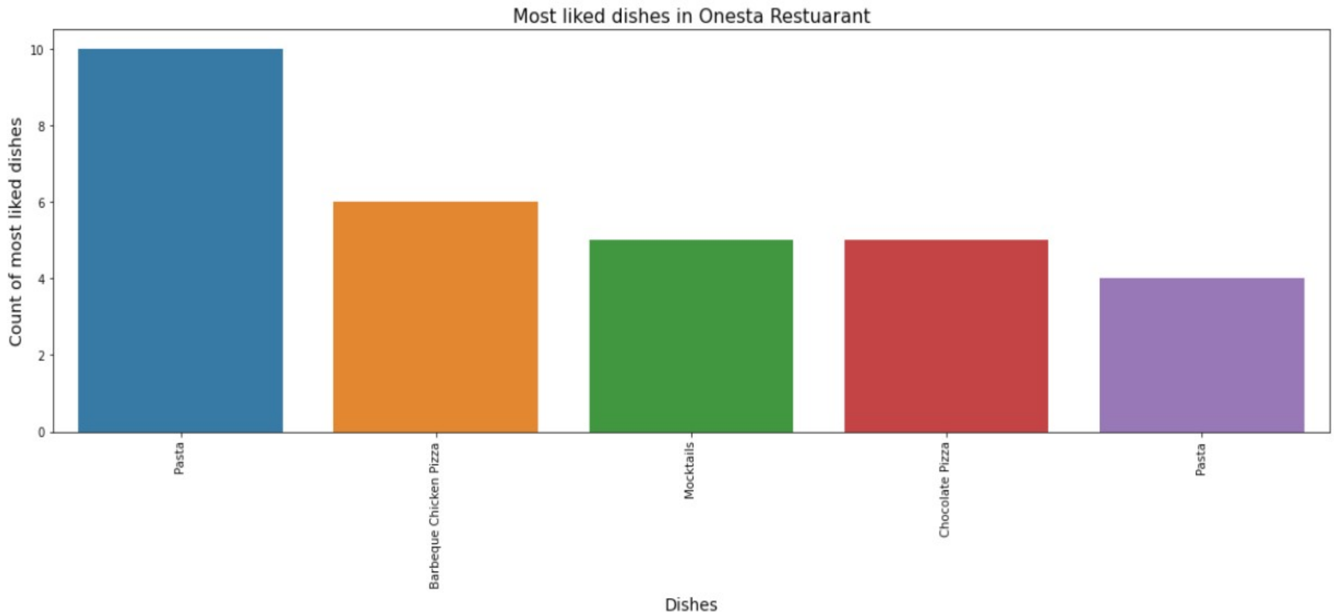
```
Count of Most liked dishes in Onesta Restuarant
```

Out[65]:

	Dishes	Count_of_most_liked_dish
0	Pasta	10
1	Barbeque Chicken Pizza	6
2	Mocktails	5
3	Chocolate Pizza	5
4	Pasta	4

In [66]:

```
plt.figure(figsize=(20,6))
sns.barplot(x=favourite_food['Dishes'],y=favourite_food['Count_of_most_liked_dish'])
plt.title('Most liked dishes in Onesta Restuarant',size=16)
plt.xticks(rotation=90)
plt.xlabel('Dishes',size=14)
plt.ylabel('Count of most liked dishes',size=14)
plt.show()
```



Empire Restaurant:

In [67]:

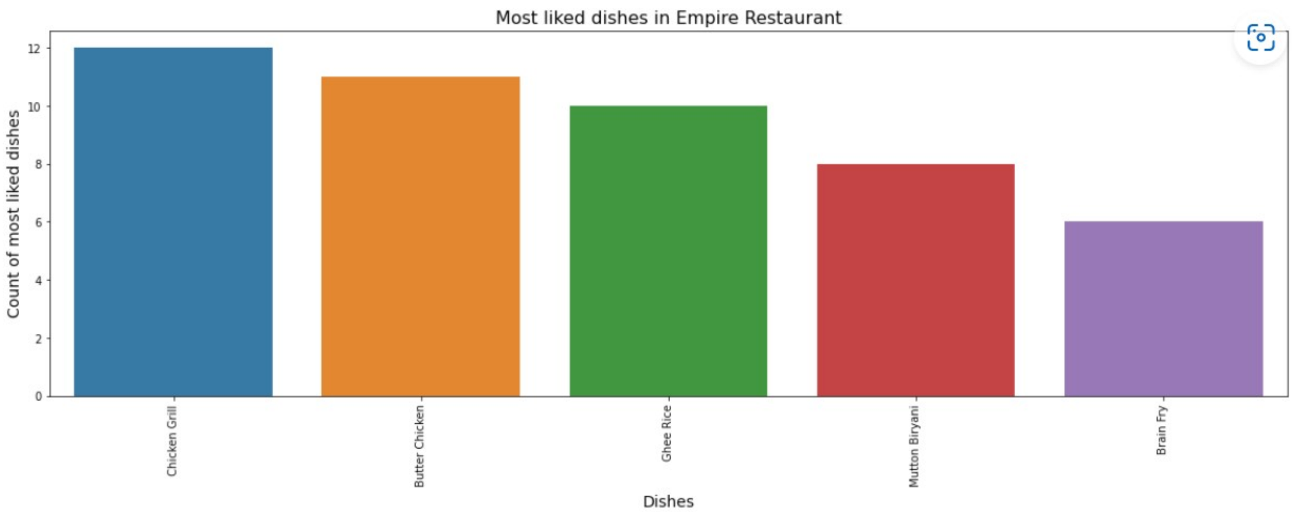
```
df_Empire = pd.DataFrame()
df_Empire = df_liked_rest.where(df_liked_rest.name == 'Empire Restaurant')
df_Empire=df_Empire.dropna()

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

print("Count of Most liked dishes in Empire Restaurant")
favourite_food = pd.Series(likes).value_counts()[:5]

favourite_food=favourite_food.reset_index()
favourite_food.rename(columns = {'index':'Dishes',0:'Count_of_most_liked_dish'}, inplace = True)
favourite_food
```

```
Count of Most liked dishes in Empire Restaurant
```

Out[67]:

	Dishes	Count_of_most_liked_dish
0	Chicken Grill	12
1	Butter Chicken	11
2	Ghee Rice	10
3	Mutton Biryani	8
4	Brain Fry	6

In [68]:
```python
plt.figure(figsize=(20,6))
sns.barplot(x=favourite_food['Dishes'],y=favourite_food['Count_of_most_liked_dish'])
plt.title('Most liked dishes in Empire Restaurant',size=16)
plt.xticks(rotation=90)
plt.xlabel('Dishes',size=14)
plt.ylabel('Count of most liked dishes',size=14)
plt.show()
```



Corner House Ice Cream Restaurant:

In [69]:
```python
df_ice_cream = pd.DataFrame()
df_ice_cream = df_liked_rest.where(df_liked_rest.name == 'Corner House Ice Cream')
df_ice_cream=df_ice_cream.dropna()

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

print("Count of Most liked dishes in Corner House Ice Cream Restaurant")
favourite_food = pd.Series(likes).value_counts()[:5]

favourite_food=favourite_food.reset_index()
favourite_food.rename(columns = {'index':'Dishes',0:'Count_of_most_liked_dish'}, inplace = True)
favourite_food
```

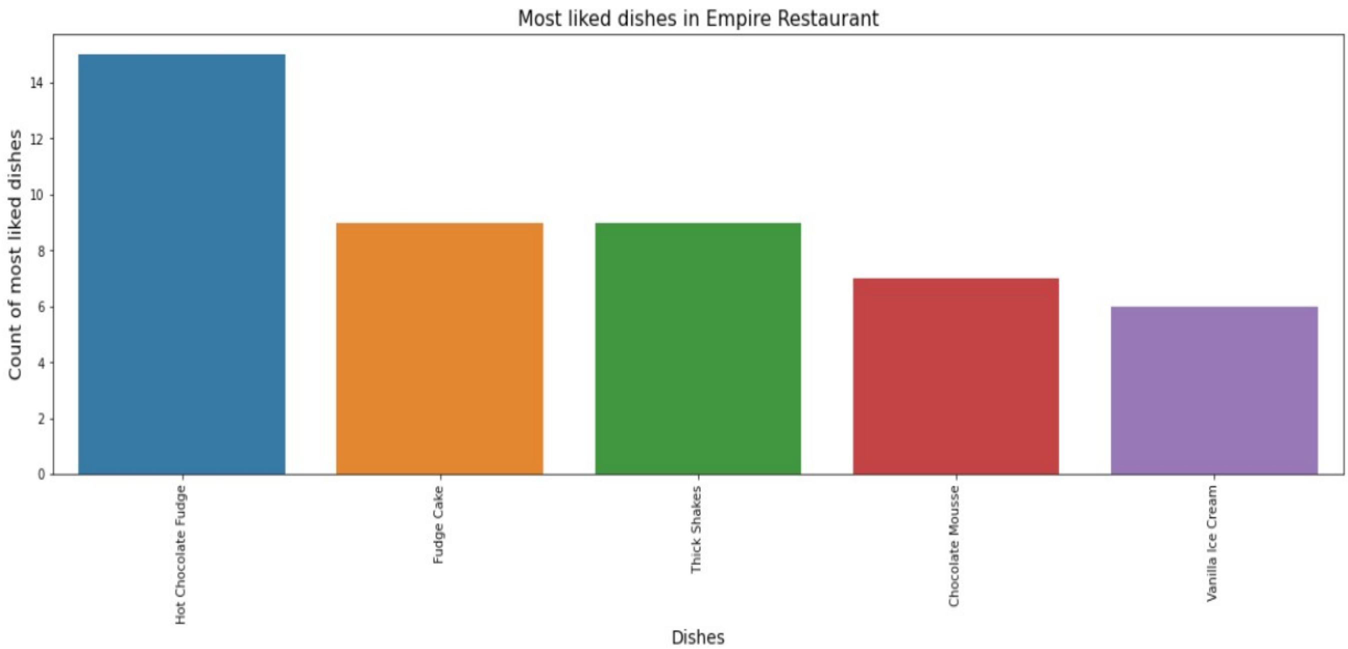
```
Count of Most liked dishes in Corner House Ice Cream Restaurant
```

Out[69]:

	Dishes	Count_of_most_liked_dish
0	Hot Chocolate Fudge	15
1	Fudge Cake	9
2	Thick Shakes	9
3	Chocolate Mousse	7
4	Vanilla Ice Cream	6

In [70]:

```python
plt.figure(figsize=(20,6))
sns.barplot(x=favourite_food['Dishes'],y=favourite_food['Count_of_most_liked_dish'])
plt.title('Most liked dishes in Empire Restaurant',size=16)
plt.xticks(rotation=90)
plt.xlabel('Dishes',size=14)
plt.ylabel('Count of most liked dishes',size=14)
plt.show()
```



Smally's Resto Cafe Cream Restaurant:

In [71]:

```python
df_smallys = pd.DataFrame()
df_smallys = df_liked_rest.where(df_liked_rest.name == "Smally's Resto Cafe")
df_smallys=df_smallys.dropna()

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

print("Count of Most liked dishes in Smally's Resto Cafe")
favourite_food = pd.Series(likes).value_counts()[:5]

favourite_food=favourite_food.reset_index()
favourite_food.rename(columns = {'index':'Dishes',0:'Count_of_most_liked_dish'}, inplace = True)
favourite_food
```

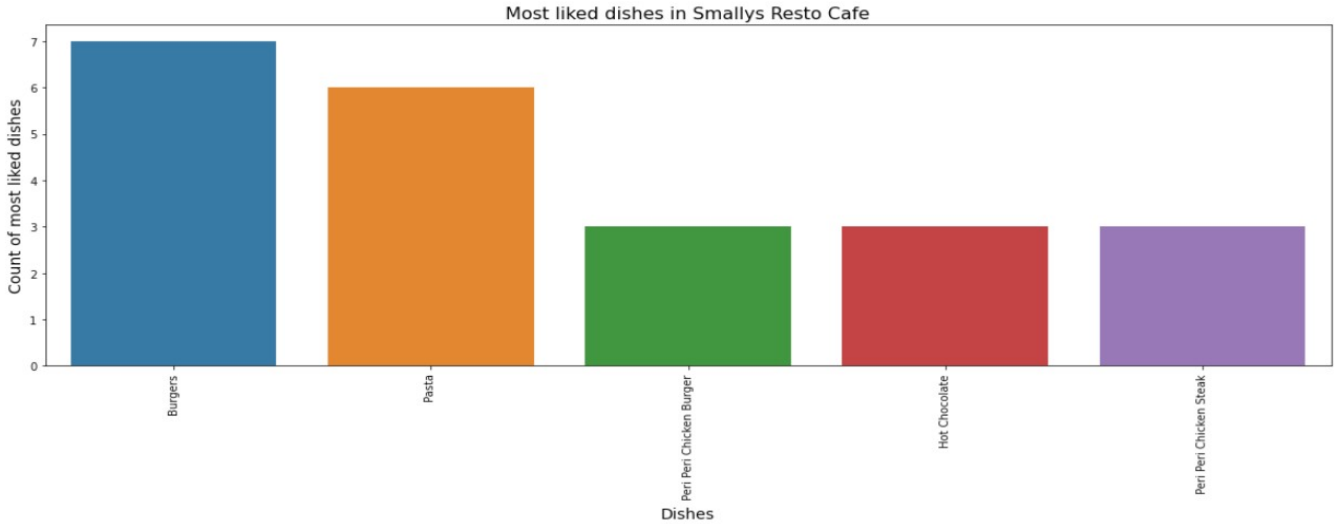
```
Count of Most liked dishes in Smally's Resto Cafe
```

Out[71]:

	Dishes	Count_of_most_liked_dish
0	Burgers	7
1	Pasta	6
2	Peri Peri Chicken Burger	3
3	Hot Chocolate	3
4	Peri Peri Chicken Steak	3

In [72]:

```
plt.figure(figsize=(20,6))
sns.barplot(x=favourite_food['Dishes'],y=favourite_food['Count_of_most_liked_dish'])
plt.title('Most liked dishes in Smallys Resto Cafe',size=16)
plt.xticks(rotation=90)
plt.xlabel('Dishes',size=14)
plt.ylabel('Count of most liked dishes',size=14)
plt.show()
```



KFC Restaurant:

In [73]:

```
df_kfc = pd.DataFrame()
df_kfc = df_liked_rest.where(df_liked_rest.name == 'KFC')
df_kfc=df_kfc.dropna()

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

print("Count of Most liked dishes in KFC Restaurant")
favourite_food = pd.Series(likes).value_counts()[:5]

favourite_food=favourite_food.reset_index()
favourite_food.rename(columns = {'index':'Dishes',0:'Count_of_most_liked_dish'}, inplace = True)
favourite_food
```

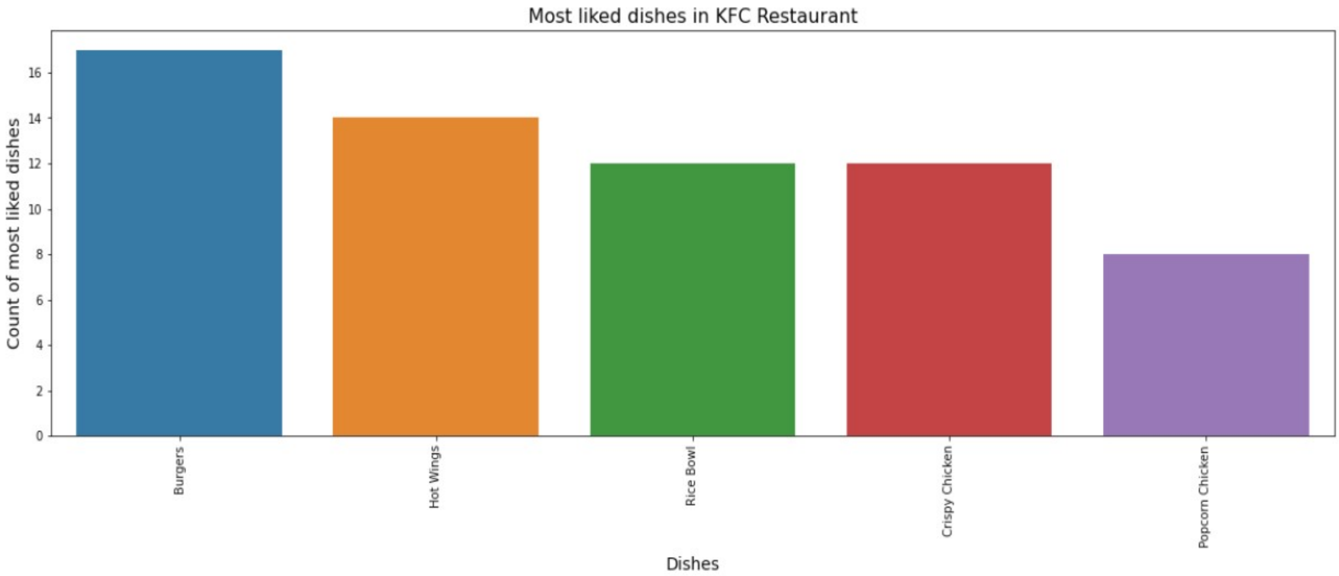
Count of Most liked dishes in KFC Restaurant

Out[73]:

	Dishes	Count_of_most_liked_dish
0	Burgers	17
1	Hot Wings	14
2	Rice Bowl	12
3	Crispy Chicken	12
4	Popcorn Chicken	8

In [74]:

```
plt.figure(figsize=(20,6))
sns.barplot(x=favourite_food['Dishes'],y=favourite_food['Count_of_most_liked_dish'])
plt.title('Most liked dishes in KFC Restaurant',size=16)
plt.xticks(rotation=90)
plt.xlabel('Dishes',size=14)
plt.ylabel('Count of most liked dishes',size=14)
plt.show()
```



Observation:

- We can infer from the analysis and compare with that previously created barplot, "Count of Most Liked Dishes in Bangalore," that the top most restaurants have the favourite dishes.

Creation of a Model:

Convert the online order column categorical variables into a numeric format:

In [75]:

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

In [76]:

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

Out[76]:

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

In [77]:
```
df.online_order = pd.to_numeric(df.online_order)
```

Convert the book_table column categorical variables into a numeric format:

In [78]:
```
df.book_table[df.book_table == 'Yes'] = 1
df.book_table[df.book_table == 'No'] = 0
```

In [79]:
```
df.book_table = pd.to_numeric(df.book_table)
```

In [80]:
```
df.book_table.value_counts()
```

Out[80]:
```
0    17191
1     6057
Name: book_table, dtype: int64
```

Convert the Object Datatypes to int using label encoder:

In [81]:
```
le = LabelEncoder()
```

In [82]:
```
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)
```

Create a copy of Dataset:

In [83]:
```
my_data=df.iloc[:,[2,3,4,5,6,7,9,10,12]]
my_data.to_csv('Zomato_df.csv')
```

Take X and Y:

In [84]:
```
x = df.iloc[:,[2,3,5,6,7,9,10,12]]
x.head()
```

Out[84]:

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

```
Out[85]: 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
```

Split the train and test part:

```
In [86]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=10)
```

Model 1:- Linear Regression:

```
In [87]: lr_model=LinearRegression()
         lr_model.fit(x_train,y_train)
```

```
Out[87]: LinearRegression()
```

```
In [88]: y_pred=lr_model.predict(x_test)
         score_lr=round(r2_score(y_test,y_pred)*100,2)
         score_lr
```

```
Out[88]: 22.82
```

Observation:

- 22.82% of Accuracy in Linear Regression Model.

Model 2:- RandomForest Regressor:

```
In [89]: RF_Model=RandomForestRegressor(n_estimators=10,random_state=10)
         RF_Model.fit(x_train,y_train)
         y_predict=RF_Model.predict(x_test)
         score_rf=round(r2_score(y_test,y_predict)*100,2)
         score_rf
```

```
Out[89]: 89.03
```

Observation:

- 89.03% of Accuracy in Random Forest Regressor Model.

Model 3:- ExtraTree Regressor:

```
In [90]:
ET_Model=ExtraTreesRegressor(n_estimators = 120)
ET_Model.fit(x_train,y_train)
y_predict=ET_Model.predict(x_test)
score_etr=round(r2_score(y_test,y_predict)*100,2)
score_etr

Out[90]: 93.3
```

Observation:

- 93.3% of Accuracy in ExtraTree Regressor Model.

Model 4:- Decision Tree Regressor:

```
In [91]:
DTree=DecisionTreeRegressor(min_samples_leaf=.0001)
DTree.fit(x_train,y_train)
y_predict=DTree.predict(x_test)
score_dtr=round(r2_score(y_test,y_predict)*100,2)
score_dtr

Out[91]: 85.54
```

Observation:

- 85.54% of Accuracy in Decision Tree Regressor Model.

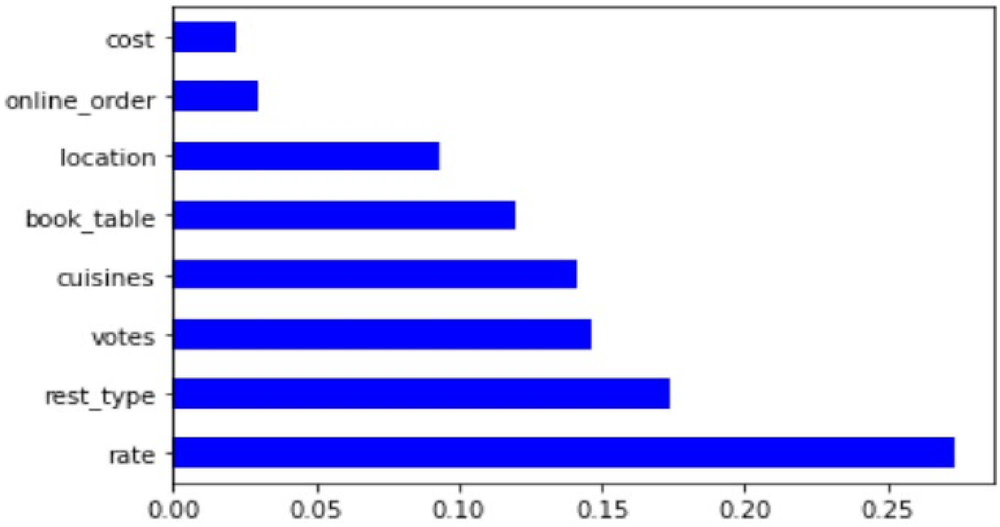
Feature Importance For a ExtraTree Regressor Model:

- Feature Importance provides a score that indicates how helpful each feature was in our model.
- The higher the Feature Score, the more that feature is used to make key decisions & thus the more important it is.

In [97]:
```
importance = ET_Model.feature_importances_

for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: 0.02975
Feature: 1, Score: 0.11946
Feature: 2, Score: 0.27380
Feature: 3, Score: 0.14643
Feature: 4, Score: 0.09346
Feature: 5, Score: 0.17400
Feature: 6, Score: 0.14147
Feature: 7, Score: 0.02163
```

In [98]:
```
index=my_data.columns[:-1]
importance = pd.Series(ET_Model.feature_importances_,index)
importance.nlargest(8).plot(kind='barh',colormap='winter')
plt.show()
```



Observation:

- From the above feature importance graph, we can conclude that the top 5 significant features were rate, rest_type, votes, cuisines, book_table.

Chapter 4

Analysis of the Result

1. We can infer from the analysis that the top 5 restaurants are Onesta, Empire Restaurant, Corner House Ice Cream, Smally's Resto Cafe, KFC.

2. The two main service types are Delivery and Dine-out.

3. The three popularity cuisines are NorthIndian, Chinese and Continental.

4. We can infer from the analysis that the top 3 famous locations in Bangalore are BTM, Koramangala 7th Block and Koramangala 4th Block.

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

6. Casual Dining, Quick Bites and Cafe are the 3 most common types of Restaurants in Bangalore.

Find the Best Model:

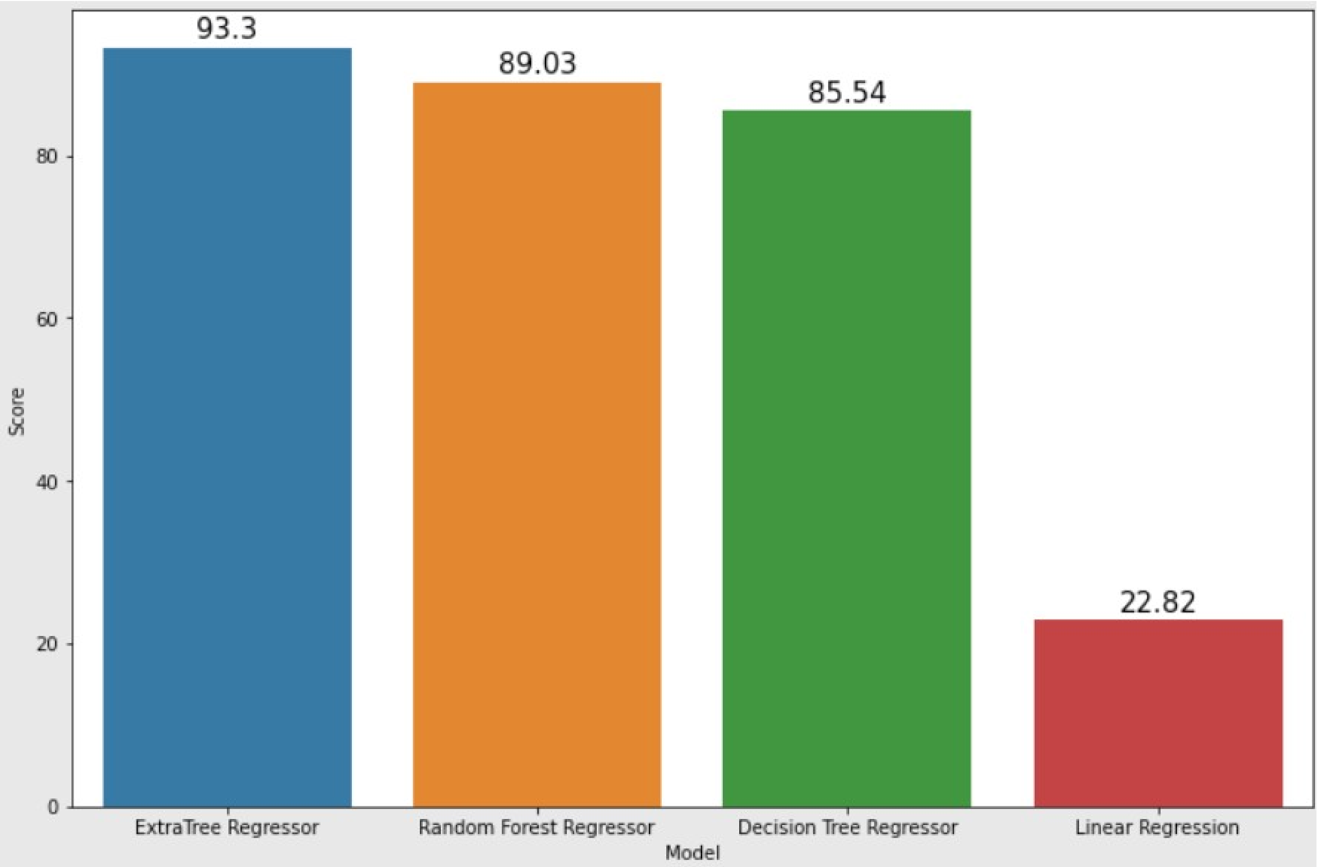
In [92]:

```
results = pd.DataFrame({
    'Model': [ 'Linear Regression',
               'Random Forest Regressor',
               'ExtraTree Regressor',
               'Decision Tree Regressor'
               ],
    'Score': [ score_lr,
               score_rf,
               score_etr,
               score_dtr]
             })
result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.reset_index(drop=True)
result_df.head()
```

Out[92]:

	Model	Score
0	ExtraTree Regressor	93.30
1	Random Forest Regressor	89.03
2	Decision Tree Regressor	85.54
3	Linear Regression	22.82

In [93]:
```
plt.subplots(figsize=(12,8))
ax=sns.barplot(x='Model',y="Score",data=result_df)
labels = (result_df["Score"])
# add result numbers on barchart
for i, v in enumerate(labels):
    ax.text(i, v+1, str(v), horizontalalignment = 'center', size = 15, color = 'black')
```



<u>Observation:</u>

- ExtraTree Regressor Model having the highest Accuracy Score 93.28%
- Random Forest Regressor Model having the Second Highest Accuracy Score 89.03%
- Decision Tree Regressor Model having the Third Highest Accuracy Score 85.26%
- Linear Regression Model Provide the lowest Accuracy Score 22.82%

Chapter 5

Conclusion

- Our ExtraTree Regressor algorithm yields the highest accuracy, 93.28%. Any accuracy above 70% is considered good, but be careful because if your accuracy is extremely high, it may be too good to be true (an example of Overfitting). Thus, 80% is the ideal accuracy!
- Out of the 8 features we examined, the top 5 significant features that helped us to predict the rating (i.e.) rate, rest_type, votes, cuisines and book_table.
- Our machine learning algorithm will able to predict the restaurant rating.
- Based upon the analysis of the result, we can suggest the restaurant owners to improve the service type and what type of cuisines need to be provided and what type of dishes need to be included in the menu.
- The two main service types are Delivery and Dine-out
- The three popularity cuisines are NorthIndian, Chinese and Continental
- The Below Dishes need to be included in the Menu
  Pasta, Pizza, Cocktails, Burgers, Mocktails, Biryani, Sandwiches, Coffee, Nachos, Barbeque Chicken Pizza, Chocolate Pizza, Chicken Grill, Butter Chicken, Ghee Rice, Mutton Biryani, Brain Fry, Hot Chocolate Fudge, Fudge Cake, Thick Shakes, Chocolate Mousse, Vanilla Ice Cream, Peri Peri Chicken Burger, Hot Chocolate, Peri Peri Chicken Steak, Hot Wings, Rice Bowl, Crispy Chicken, Popcorn Chicken

Here is access to the data set & code from my GitHub page:

https://github.com/Adaikkkappan/ZomatoRestaurantRating-EDA-and-MachineLearning

References

1. Titanic dataset from Kaggle Created by **HIMANSHU PODDAR**.

2. Titanic dataset from Kaggle, Author-**FFRANKUSHA.**

https://www.kaggle.com/code/anatosly/eda-bangalore-restaurants

3. Titanic dataset from Kaggle, Author-**ADITYA RAWAT.**

https://www.kaggle.com/code/adityarawat10/zomato-eda-fe-model-building

4. Zomato official website: https://www.zomato.com/