This data is of Zomato-bangalore-restaurants

****Importing Libraries in python****

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier,
RandomForestRegressor
from sklearn.metrics import accuracy_score, classification_report,
mean_absolute_error, mean_squared_error
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
```

****Reading a csv file****

```
df =
pd.read_csv("/kaggle/input/zomato-bangalore-restaurants/zomato.csv")
df.head()
```

```
                                                 url  \
0  https://www.zomato.com/bangalore/jalsa-banasha...
1  https://www.zomato.com/bangalore/spice-elephan...
2  https://www.zomato.com/SanchurroBangalore?cont...
3  https://www.zomato.com/bangalore/addhuri-udupi...
4  https://www.zomato.com/bangalore/grand-village...

                                          address
name  \
0  942, 21st Main Road, 2nd Stage, Banashankari, ...
Jalsa
1  2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...         Spice
Elephant
2  1112, Next to KIMS Medical College, 17th Cross...        San Churro
Cafe
3  1st Floor, Annakuteera, 3rd Stage, Banashankar...  Addhuri Udupi
Bhojana
4  10, 3rd Floor, Lakshmi Associates, Gandhi Baza...          Grand
Village

  online_order book_table   rate  votes
phone  \
0          Yes        Yes  4.1/5    775    080 42297555\r\n+91
9743772233
```

```
1          Yes         No  4.1/5    787                       080
41714161
2          Yes         No  3.8/5    918                     +91
9663487993
3           No         No  3.7/5     88                     +91
9620009302
4           No         No  3.8/5    166  +91 8026612447\r\n+91
9901210005

       location           rest_type  \
0  Banashankari        Casual Dining
1  Banashankari        Casual Dining
2  Banashankari  Cafe, Casual Dining
3  Banashankari          Quick Bites
4  Basavanagudi        Casual Dining

                                        dish_liked  \
0  Pasta, Lunch Buffet, Masala Papad, Paneer Laja...
1  Momos, Lunch Buffet, Chocolate Nirvana, Thai G...
2  Churros, Cannelloni, Minestrone Soup, Hot Choc...
3                                     Masala Dosa
4                            Panipuri, Gol Gappe

                        cuisines approx_cost(for two people)  \
0  North Indian, Mughlai, Chinese                         800
1     Chinese, North Indian, Thai                         800
2          Cafe, Mexican, Italian                         800
3     South Indian, North Indian                          300
4       North Indian, Rajasthani                          600

                                       reviews_list menu_item  \
0  [('Rated 4.0', 'RATED\n  A beautiful place to ...        []
1  [('Rated 4.0', 'RATED\n  Had been here for din...        []
2  [('Rated 3.0', "RATED\n  Ambience is not that ...        []
3  [('Rated 4.0', "RATED\n  Great food and proper...        []
4  [('Rated 4.0', 'RATED\n  Very good restaurant ...        []

  listed_in(type) listed_in(city)
0          Buffet    Banashankari
1          Buffet    Banashankari
2          Buffet    Banashankari
3          Buffet    Banashankari
4          Buffet    Banashankari
```

****Size of the dataframe****

```
df.shape
```

```
(51717, 17)
```

Knowing the columns

```
df.columns
```

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

****droping or removing certain columns to make data more readable****

```
df = df.drop(['url',
'address','phone','dish_liked','reviews_list','menu_item'], axis = 1 )
df.head()
```

```
                    name online_order book_table   rate  votes
location  \
0                  Jalsa          Yes        Yes  4.1/5    775
Banashankari
1         Spice Elephant          Yes         No  4.1/5    787
Banashankari
2        San Churro Cafe          Yes         No  3.8/5    918
Banashankari
3  Addhuri Udupi Bhojana           No         No  3.7/5     88
Banashankari
4          Grand Village           No         No  3.8/5    166
Basavanagudi

             rest_type                        cuisines  \
0        Casual Dining  North Indian, Mughlai, Chinese
1        Casual Dining     Chinese, North Indian, Thai
2  Cafe, Casual Dining          Cafe, Mexican, Italian
3          Quick Bites      South Indian, North Indian
4        Casual Dining        North Indian, Rajasthani

  approx_cost(for two people) listed_in(type) listed_in(city)
0                        800          Buffet    Banashankari
1                        800          Buffet    Banashankari
2                        800          Buffet    Banashankari
3                        300          Buffet    Banashankari
4                        600          Buffet    Banashankari
```

getting the desired information

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   name                         51717 non-null  object
 1   online_order                 51717 non-null  object
 2   book_table                   51717 non-null  object
 3   rate                         43942 non-null  object
 4   votes                        51717 non-null  int64 
 5   location                     51696 non-null  object
 6   rest_type                    51490 non-null  object
 7   cuisines                     51672 non-null  object
 8   approx_cost(for two people)  51371 non-null  object
 9   listed_in(type)              51717 non-null  object
 10  listed_in(city)              51717 non-null  object
dtypes: int64(1), object(10)
memory usage: 4.3+ MB
```

Deleting the duplicated data in the dataframe

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

```
(51609, 11)
```

```
df.head()
```

```
                    name online_order book_table   rate  votes
location  \
0                  Jalsa          Yes        Yes  4.1/5    775
Banashankari
1         Spice Elephant          Yes         No  4.1/5    787
Banashankari
2        San Churro Cafe          Yes         No  3.8/5    918
Banashankari
3  Addhuri Udupi Bhojana           No         No  3.7/5     88
Banashankari
4          Grand Village           No         No  3.8/5    166
Basavanagudi

             rest_type                        cuisines  \
0        Casual Dining  North Indian, Mughlai, Chinese
1        Casual Dining     Chinese, North Indian, Thai
2  Cafe, Casual Dining          Cafe, Mexican, Italian
3          Quick Bites      South Indian, North Indian
4        Casual Dining        North Indian, Rajasthani

  approx_cost(for two people) listed_in(type) listed_in(city)
0                        800          Buffet    Banashankari
```

```
1                          800          Buffet    Banashankari
2                          800          Buffet    Banashankari
3                          300          Buffet    Banashankari
4                          600          Buffet    Banashankari
```

Removing "/5" from the column and also finding the unique value in the column 'rate'

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

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

To remove /5 from the column we need to have following code:

- In this code we had replaced NEW and - with nan
- splitted the / between actual float values and 5 eg:- **"4.8/5" = "4.8"**
- Now it will return the float value only

```
def handlerate(value):
    if(value == 'NEW' or value == "-"):
        return np.nan
    else:
        value = str(value).split('/')
        value = value[0]
        return float(value)

df['rate'] = df['rate'].apply(handlerate)
df['rate'].head()
```

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

Handling the null values

finding the null values

```
df.rate.isnull().sum()
```

```
10019
```

- **Filling the null values in Rate column with Mean**
- **Now the null values are reduced to 0**

```
df['rate'].fillna(df['rate'].mean(), inplace = True)
df['rate'].isnull().sum()
```

```
0
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51609 entries, 0 to 51716
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   name                         51609 non-null  object 
 1   online_order                 51609 non-null  object 
 2   book_table                   51609 non-null  object 
 3   rate                         51609 non-null  float64
 4   votes                        51609 non-null  int64  
 5   location                     51588 non-null  object 
 6   rest_type                    51382 non-null  object 
 7   cuisines                     51564 non-null  object 
 8   approx_cost(for two people)  51265 non-null  object 
 9   listed_in(type)              51609 non-null  object 
 10  listed_in(city)              51609 non-null  object 
dtypes: float64(1), int64(1), object(9)
memory usage: 4.7+ MB
```

Removing all the null values

```
df.dropna(inplace = True)
df.head()
```

```
                  name online_order book_table  rate  votes
location  \
0                Jalsa          Yes        Yes   4.1    775
Banashankari
1         Spice Elephant          Yes         No   4.1    787
Banashankari
2        San Churro Cafe          Yes         No   3.8    918
Banashankari
3  Addhuri Udupi Bhojana           No         No   3.7     88
Banashankari
```

```
4          Grand Village           No         No   3.8    166
Basavanagudi

             rest_type                        cuisines  \
0        Casual Dining  North Indian, Mughlai, Chinese
1        Casual Dining     Chinese, North Indian, Thai
2  Cafe, Casual Dining          Cafe, Mexican, Italian
3          Quick Bites      South Indian, North Indian
4        Casual Dining        North Indian, Rajasthani

  approx_cost(for two people) listed_in(type) listed_in(city)
0                         800          Buffet    Banashankari
1                         800          Buffet    Banashankari
2                         800          Buffet    Banashankari
3                         300          Buffet    Banashankari
4                         600          Buffet    Banashankari
```

renaming the columns to make the data easily accessible

```
df.rename(columns = {'approx_cost(for two people)':'Cost2plates',
'listed_in(type)':'Type'}, inplace = True)
df.head()
```

```
                    name online_order book_table  rate  votes
location  \
0                  Jalsa          Yes        Yes   4.1    775
Banashankari
1         Spice Elephant          Yes         No   4.1    787
Banashankari
2        San Churro Cafe          Yes         No   3.8    918
Banashankari
3  Addhuri Udupi Bhojana           No         No   3.7     88
Banashankari
4          Grand Village           No         No   3.8    166
Basavanagudi

             rest_type                        cuisines Cost2plates
Type  \
0        Casual Dining  North Indian, Mughlai, Chinese         800
Buffet
1        Casual Dining     Chinese, North Indian, Thai         800
Buffet
2  Cafe, Casual Dining          Cafe, Mexican, Italian         800
Buffet
3          Quick Bites      South Indian, North Indian         300
Buffet
4        Casual Dining        North Indian, Rajasthani         600
Buffet

  listed_in(city)
```

```
0    Banashankari
1    Banashankari
2    Banashankari
3    Banashankari
4    Banashankari
```

Finding the Unique value various columns

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

```
array(['Banashankari', 'Basavanagudi', 'Mysore Road', 'Jayanagar',
       'Kumaraswamy Layout', 'Rajarajeshwari Nagar', 'Vijay Nagar',
       'Uttarahalli', 'JP Nagar', 'South Bangalore', 'City Market',
       'Nagarbhavi', 'Bannerghatta Road', 'BTM', 'Kanakapura Road',
       'Bommanahalli', 'CV Raman Nagar', 'Electronic City', 'HSR',
       'Marathahalli', 'Wilson Garden', 'Shanti Nagar',
       'Koramangala 5th Block', 'Koramangala 8th Block', 'Richmond
Road',
       'Koramangala 7th Block', 'Jalahalli', 'Koramangala 4th Block',
       'Bellandur', 'Sarjapur Road', 'Whitefield', 'East Bangalore',
       'Old Airport Road', 'Indiranagar', 'Koramangala 1st Block',
       'Frazer Town', 'RT Nagar', 'MG Road', 'Brigade Road',
       'Lavelle Road', 'Church Street', 'Ulsoor', 'Residency Road',
       'Shivajinagar', 'Infantry Road', 'St. Marks Road',
       'Cunningham Road', 'Race Course Road', 'Commercial Street',
       'Vasanth Nagar', 'HBR Layout', 'Domlur', 'Ejipura',
       'Jeevan Bhima Nagar', 'Old Madras Road', 'Malleshwaram',
       'Seshadripuram', 'Kammanahalli', 'Koramangala 6th Block',
       'Majestic', 'Langford Town', 'Central Bangalore', 'Sanjay
Nagar',
       'Brookefield', 'ITPL Main Road, Whitefield',
       'Varthur Main Road, Whitefield', 'KR Puram',
       'Koramangala 2nd Block', 'Koramangala 3rd Block',
'Koramangala',
       'Hosur Road', 'Rajajinagar', 'Banaswadi', 'North Bangalore',
       'Nagawara', 'Hennur', 'Kalyan Nagar', 'New BEL Road', 'Jakkur',
       'Rammurthy Nagar', 'Thippasandra', 'Kaggadasapura', 'Hebbal',
       'Kengeri', 'Sankey Road', 'Sadashiv Nagar', 'Basaveshwara
Nagar',
       'Yeshwantpur', 'West Bangalore', 'Magadi Road', 'Yelahanka',
       'Sahakara Nagar', 'Peenya'], dtype=object)
```

```
df['listed_in(city)'].unique()
```

```
array(['Banashankari', 'Bannerghatta Road', 'Basavanagudi',
'Bellandur',
       'Brigade Road', 'Brookefield', 'BTM', 'Church Street',
       'Electronic City', 'Frazer Town', 'HSR', 'Indiranagar',
       'Jayanagar', 'JP Nagar', 'Kalyan Nagar', 'Kammanahalli',
       'Koramangala 4th Block', 'Koramangala 5th Block',
```

```
       'Koramangala 6th Block', 'Koramangala 7th Block', 'Lavelle
Road',
       'Malleshwaram', 'Marathahalli', 'MG Road', 'New BEL Road',
       'Old Airport Road', 'Rajajinagar', 'Residency Road',
       'Sarjapur Road', 'Whitefield'], dtype=object)
```

Removing the listed_in(city) as the cities and location are one and the same thing

```
df = df.drop(['listed_in(city)'], axis = 1)
```

Finding the unique values in cost2plates column

- **defining a function to handle the unique values**
- **replacing the "," with ""**

```
df['Cost2plates'].unique()
```

```
array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
       '900', '200', '750', '150', '850', '100', '1,200', '350',
'250',
       '950', '1,000', '1,500', '1,300', '199', '80', '1,100', '160',
       '1,600', '230', '130', '50', '190', '1,700', '1,400', '180',
       '1,350', '2,200', '2,000', '1,800', '1,900', '330', '2,500',
       '2,100', '3,000', '2,800', '3,400', '40', '1,250', '3,500',
       '4,000', '2,400', '2,600', '120', '1,450', '469', '70',
'3,200',
       '60', '560', '240', '360', '6,000', '1,050', '2,300', '4,100',
       '5,000', '3,700', '1,650', '2,700', '4,500', '140'],
dtype=object)
```

```
def handlecomma(value):
    value = str(value)
    if ',' in value:
        value = value.replace(',','')
        return float(value)
    else:
        return float(value)

df['Cost2plates'] = df['Cost2plates'].apply(handlecomma)
df['Cost2plates'].unique()
```

```
array([ 800.,  300.,  600.,  700.,  550.,  500.,  450.,  650.,  400.,
        900.,  200.,  750.,  150.,  850.,  100., 1200.,  350.,  250.,
        950., 1000., 1500., 1300.,  199.,   80., 1100.,  160., 1600.,
        230.,  130.,   50.,  190., 1700., 1400.,  180., 1350., 2200.,
       2000., 1800., 1900.,  330., 2500., 2100., 3000., 2800., 3400.,
         40., 1250., 3500., 4000., 2400., 2600.,  120., 1450.,  469.,
         70., 3200.,   60.,  560.,  240.,  360., 6000., 1050., 2300.,
       4100., 5000., 3700., 1650., 2700., 4500.,  140.])
```

```
df.head()
```

```
                    name online_order book_table  rate  votes
location  \
0                  Jalsa          Yes        Yes   4.1    775
Banashankari
1         Spice Elephant          Yes         No   4.1    787
Banashankari
2        San Churro Cafe          Yes         No   3.8    918
Banashankari
3  Addhuri Udupi Bhojana           No         No   3.7     88
Banashankari
4          Grand Village           No         No   3.8    166
Basavanagudi

             rest_type                        cuisines  Cost2plates
Type
0        Casual Dining  North Indian, Mughlai, Chinese        800.0
Buffet
1        Casual Dining     Chinese, North Indian, Thai        800.0
Buffet
2  Cafe, Casual Dining          Cafe, Mexican, Italian        800.0
Buffet
3          Quick Bites      South Indian, North Indian        300.0
Buffet
4        Casual Dining        North Indian, Rajasthani        600.0
Buffet
```

Cleaning the rest_type

Counting the different values in rest_type

```
df['rest_type'].value_counts()
```

```
Quick Bites                   19010
Casual Dining                 10253
Cafe                           3682
Delivery                       2574
Dessert Parlor                 2242
                              ...
Dessert Parlor, Kiosk             2
Food Court, Beverage Shop         2
Dessert Parlor, Food Court        2
Quick Bites, Kiosk                1
Sweet Shop, Dessert Parlor        1
Name: rest_type, Length: 93, dtype: int64
```

- **sorting the data in ascending manner**
- **making the less than 1000 rest_type frequency under others**
- **defining a function to do all these stuff**

```
rest_types = df['rest_type'].value_counts(ascending = False)
rest_types
```

```
Quick Bites                   19010
Casual Dining                 10253
Cafe                           3682
Delivery                       2574
Dessert Parlor                 2242
                              ...
Dessert Parlor, Kiosk             2
Food Court, Beverage Shop         2
Dessert Parlor, Food Court        2
Quick Bites, Kiosk                1
Sweet Shop, Dessert Parlor        1
Name: rest_type, Length: 93, dtype: int64
```

```
rest_types_lessthan1000 = rest_types[rest_types < 1000]
rest_types_lessthan1000
```

```
Beverage Shop                 863
Bar                           686
Food Court                    616
Sweet Shop                    468
Bar, Casual Dining            411
                             ...
Dessert Parlor, Kiosk           2
Food Court, Beverage Shop       2
Dessert Parlor, Food Court      2
Quick Bites, Kiosk              1
Sweet Shop, Dessert Parlor      1
Name: rest_type, Length: 85, dtype: int64
```

```
def handlerest_type(value):
    if(value in rest_types_lessthan1000):
        return 'others'
    else:
        return value

df['rest_type'] = df['rest_type'].apply(handlerest_type)
df['rest_type'].value_counts()
```

```
Quick Bites           19010
Casual Dining         10253
others                 9003
Cafe                   3682
Delivery               2574
Dessert Parlor         2242
Takeaway, Delivery     2008
Bakery                 1140
Casual Dining, Bar     1130
Name: rest_type, dtype: int64
```

```
df.head()
```

```
                    name online_order book_table  rate  votes
location  \
0                  Jalsa          Yes        Yes   4.1    775
Banashankari
1         Spice Elephant          Yes         No   4.1    787
Banashankari
2        San Churro Cafe          Yes         No   3.8    918
Banashankari
3  Addhuri Udupi Bhojana           No         No   3.7     88
Banashankari
4          Grand Village           No         No   3.8    166
Basavanagudi

       rest_type                        cuisines  Cost2plates    Type

0  Casual Dining  North Indian, Mughlai, Chinese        800.0  Buffet

1  Casual Dining     Chinese, North Indian, Thai        800.0  Buffet

2         others          Cafe, Mexican, Italian        800.0  Buffet

3    Quick Bites      South Indian, North Indian        300.0  Buffet

4  Casual Dining        North Indian, Rajasthani        600.0  Buffet
```

Doing the same thing with location as rest_type

```
df['location'].value_counts()
```

```
BTM                    5056
HSR                    2494
Koramangala 5th Block  2479
JP Nagar               2218
Whitefield             2105
                       ...
West Bangalore            6
Yelahanka                 5
Jakkur                    3
Rajarajeshwari Nagar      2
Peenya                    1
Name: location, Length: 93, dtype: int64
```

- **making the less than 1000 rest_type frequency under others**

```
location = df['location'].value_counts(ascending = False)
location_lessthan300 = location[location < 300]
```

```
def handle_location(value):
    if(value in location_lessthan300):
        return 'others'
    else:
        return value

df['location'] = df['location'].apply(handle_location)
df['location'].value_counts()
```

```
BTM                    5056
others                 4954
HSR                    2494
Koramangala 5th Block  2479
JP Nagar               2218
Whitefield             2105
Indiranagar            2026
Jayanagar              1916
Marathahalli           1805
Bannerghatta Road      1609
Bellandur              1268
Electronic City        1246
Koramangala 1st Block  1236
Brigade Road           1210
Koramangala 7th Block  1174
Koramangala 6th Block  1127
Sarjapur Road          1047
Koramangala 4th Block  1017
Ulsoor                 1011
Banashankari            902
MG Road                 893
Kalyan Nagar            841
Richmond Road           803
Malleshwaram            721
Frazer Town             714
Basavanagudi            684
Residency Road          671
Brookefield             656
New BEL Road            644
Banaswadi               640
Kammanahalli            639
Rajajinagar             591
Church Street           566
Lavelle Road            518
Shanti Nagar            508
Shivajinagar            498
Cunningham Road         490
Domlur                  482
Old Airport Road        437
Ejipura                 433
Commercial Street       370
```

```
St. Marks Road            343
Name: location, dtype: int64
```

```
df.head()
```

```
                    name online_order book_table  rate  votes
location  \
0                  Jalsa          Yes        Yes   4.1    775
Banashankari
1         Spice Elephant          Yes         No   4.1    787
Banashankari
2        San Churro Cafe          Yes         No   3.8    918
Banashankari
3  Addhuri Udupi Bhojana           No         No   3.7     88
Banashankari
4          Grand Village           No         No   3.8    166
Basavanagudi

       rest_type                        cuisines  Cost2plates    Type

0  Casual Dining  North Indian, Mughlai, Chinese        800.0  Buffet

1  Casual Dining     Chinese, North Indian, Thai        800.0  Buffet

2         others          Cafe, Mexican, Italian        800.0  Buffet

3    Quick Bites      South Indian, North Indian        300.0  Buffet

4  Casual Dining        North Indian, Rajasthani        600.0  Buffet
```

Cleaning cuisines column

```
cuisines = df['cuisines'].value_counts(ascending = False)

cuisines_lessthan100 = cuisines[cuisines<100]

def handle_cuisines(value):
    if value in cuisines_lessthan100:
        return 'others'
    else:
        return value

df['cuisines'] = df['cuisines'].apply(handle_cuisines)
df['cuisines'].value_counts()
```

```
others                                  26159
North Indian                             2852
North Indian, Chinese                    2351
```

```
South Indian                                1820
Biryani                                      903
                                            ...
South Indian, Chinese, North Indian          105
North Indian, Mughlai, Chinese               104
South Indian, Fast Food                      104
Italian, Pizza                               102
North Indian, Chinese, Seafood               102
Name: cuisines, Length: 70, dtype: int64
```

```
df.head()
```

```
                    name online_order book_table  rate  votes
location  \
0                  Jalsa          Yes        Yes   4.1    775
Banashankari
1         Spice Elephant          Yes         No   4.1    787
Banashankari
2        San Churro Cafe          Yes         No   3.8    918
Banashankari
3  Addhuri Udupi Bhojana           No         No   3.7     88
Banashankari
4          Grand Village           No         No   3.8    166
Basavanagudi

       rest_type                        cuisines  Cost2plates    Type

0  Casual Dining  North Indian, Mughlai, Chinese        800.0  Buffet

1  Casual Dining                          others        800.0  Buffet

2         others                          others        800.0  Buffet

3    Quick Bites      South Indian, North Indian        300.0  Buffet

4  Casual Dining                          others        600.0  Buffet
```

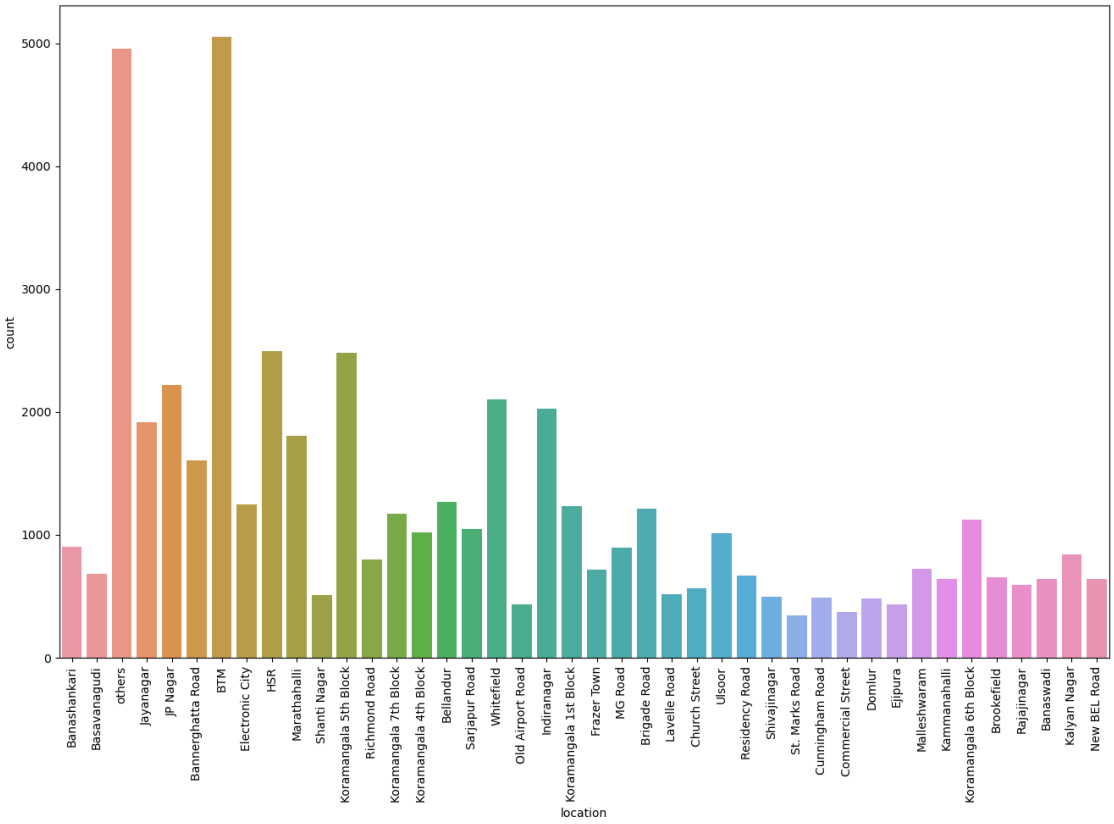
Data is Clean, let's Jump to visualization !!

Count plot of Various Locations

```
plt.figure(figsize = (16,10))
ax = sns.countplot(x = 'location', data = df)
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
```

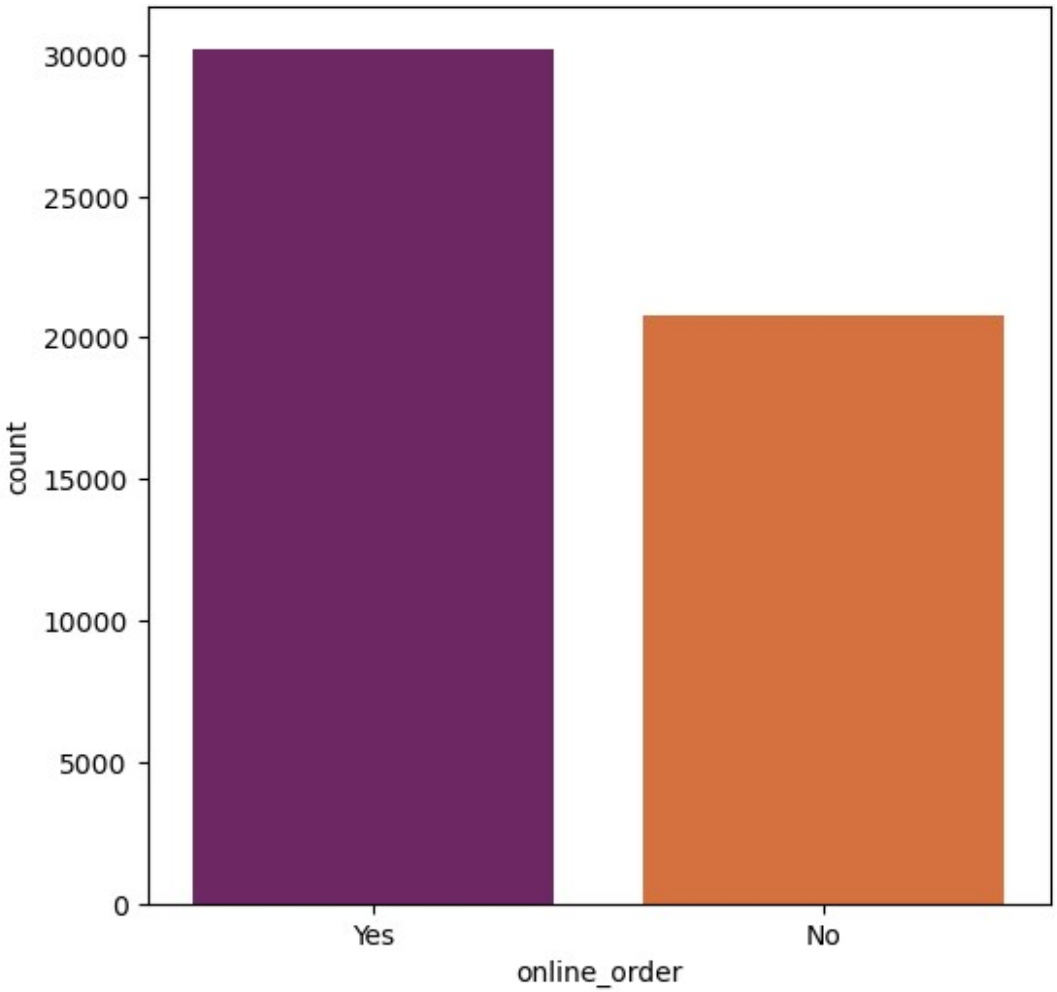
```
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33,
        34, 35, 36, 37, 38, 39, 40, 41]),
 [Text(0, 0, 'Banashankari'),
  Text(1, 0, 'Basavanagudi'),
  Text(2, 0, 'others'),
  Text(3, 0, 'Jayanagar'),
  Text(4, 0, 'JP Nagar'),
  Text(5, 0, 'Bannerghatta Road'),
  Text(6, 0, 'BTM'),
  Text(7, 0, 'Electronic City'),
  Text(8, 0, 'HSR'),
  Text(9, 0, 'Marathahalli'),
  Text(10, 0, 'Shanti Nagar'),
  Text(11, 0, 'Koramangala 5th Block'),
  Text(12, 0, 'Richmond Road'),
  Text(13, 0, 'Koramangala 7th Block'),
  Text(14, 0, 'Koramangala 4th Block'),
  Text(15, 0, 'Bellandur'),
  Text(16, 0, 'Sarjapur Road'),
  Text(17, 0, 'Whitefield'),
  Text(18, 0, 'Old Airport Road'),
  Text(19, 0, 'Indiranagar'),
  Text(20, 0, 'Koramangala 1st Block'),
  Text(21, 0, 'Frazer Town'),
  Text(22, 0, 'MG Road'),
  Text(23, 0, 'Brigade Road'),
  Text(24, 0, 'Lavelle Road'),
  Text(25, 0, 'Church Street'),
  Text(26, 0, 'Ulsoor'),
  Text(27, 0, 'Residency Road'),
  Text(28, 0, 'Shivajinagar'),
  Text(29, 0, 'St. Marks Road'),
  Text(30, 0, 'Cunningham Road'),
  Text(31, 0, 'Commercial Street'),
  Text(32, 0, 'Domlur'),
  Text(33, 0, 'Ejipura'),
  Text(34, 0, 'Malleshwaram'),
  Text(35, 0, 'Kammanahalli'),
  Text(36, 0, 'Koramangala 6th Block'),
  Text(37, 0, 'Brookefield'),
  Text(38, 0, 'Rajajinagar'),
  Text(39, 0, 'Banaswadi'),
  Text(40, 0, 'Kalyan Nagar'),
  Text(41, 0, 'New BEL Road')])
```



Visualizing the online order

```
plt.figure(figsize = (6,6))
sns.countplot(x = 'online_order', data = df, palette = 'inferno')
```

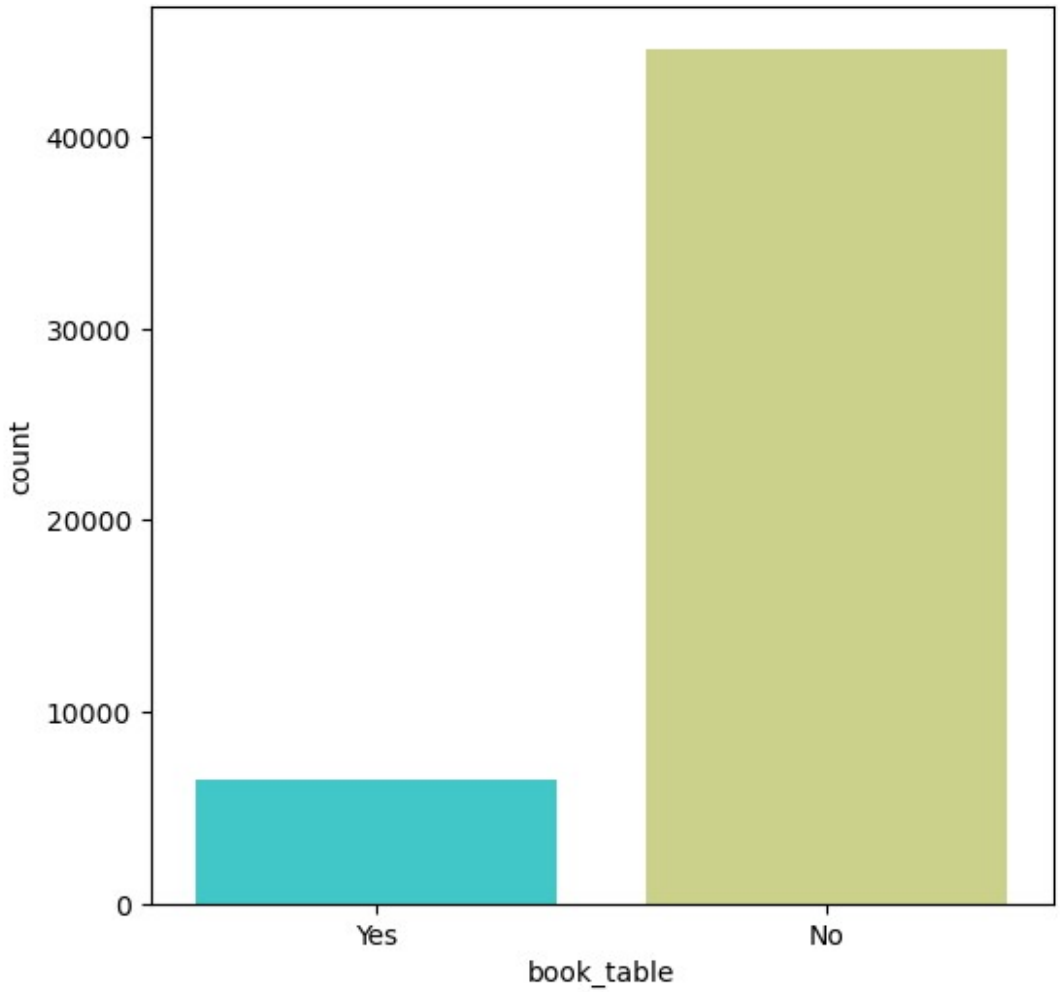
```
<AxesSubplot:xlabel='online_order', ylabel='count'>
```



Visualizing the book table

```
plt.figure(figsize = (6,6))
sns.countplot(x = 'book_table', data = df, palette = 'rainbow')
```

```
<AxesSubplot:xlabel='book_table', ylabel='count'>
```



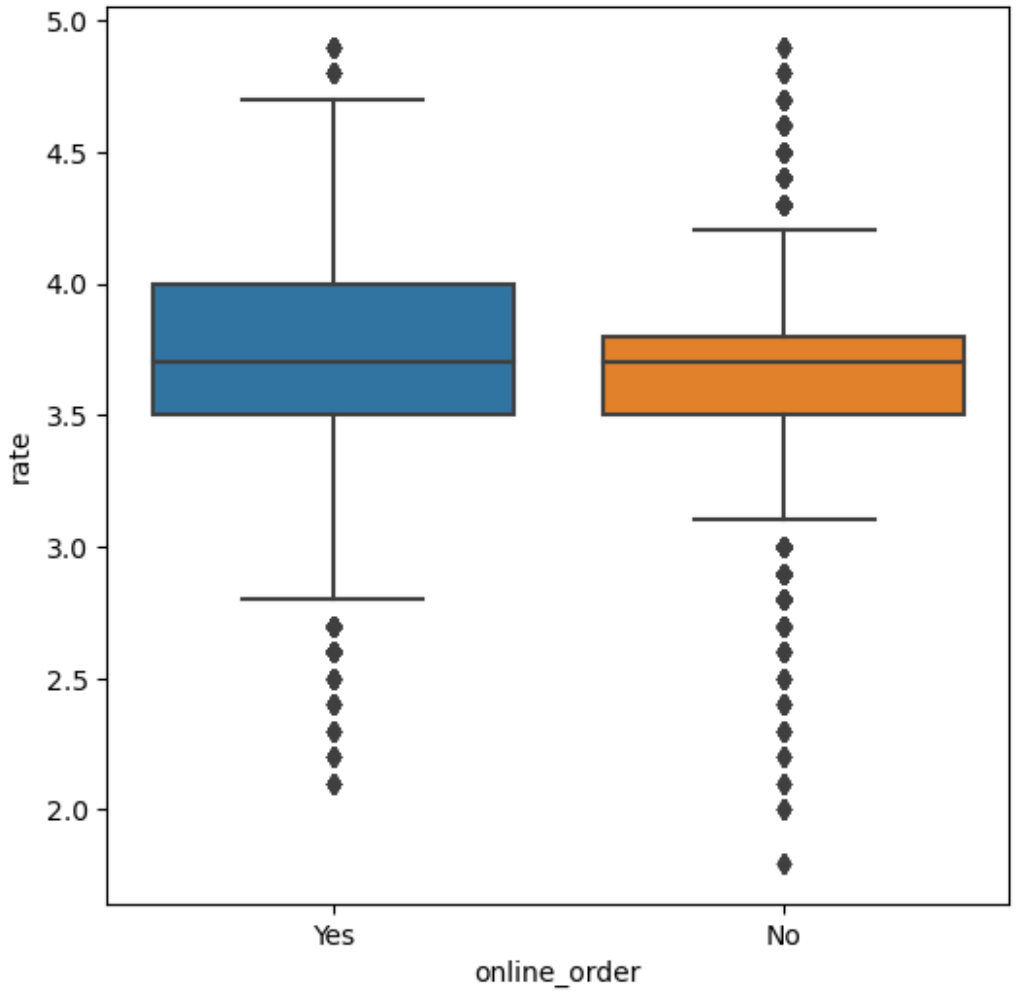
- **So we can conclude that most of the restaurants in India don't offer the facility of booking a table**
- **And the many restaurants do offer the facility of online ordering**

Visualizing Online order vs Rate

rate is not the price but actually the review rating

```
plt.figure(figsize = (6,6))
sns.boxplot(x = 'online_order', y = 'rate', data = df)
```

```
<AxesSubplot:xlabel='online_order', ylabel='rate'>
```



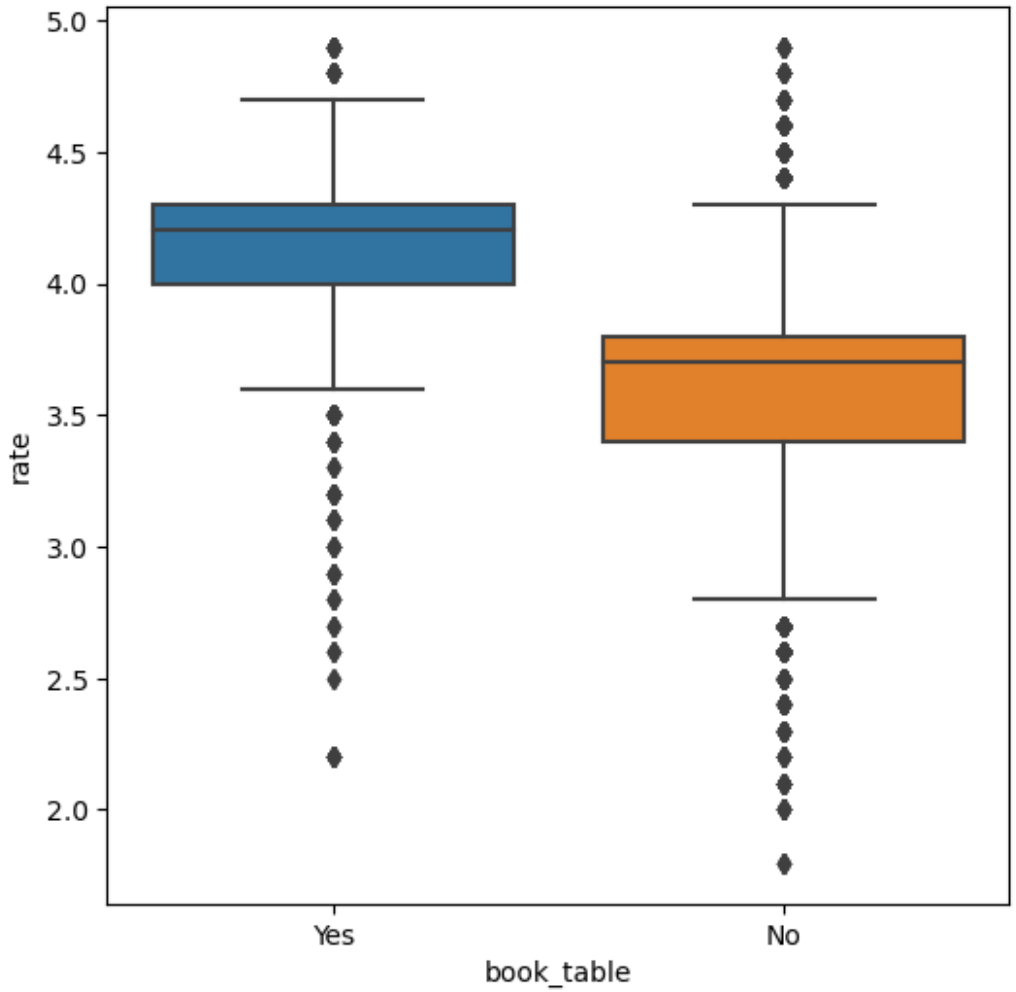
The conclusion

- The maximum rating of the restaurants having the online order facility is **higher(4.8 or 4.7)** as compared to restaurants not having this online ordering facility **(around 4.2 or 4.3)**...
- so yeah we can infer that restaurants having the online order facility are ***tend to outrank*** the ones with no such facility.

Visualizing the book_table vs rate

```
plt.figure(figsize = (6,6))
sns.boxplot(x = 'book_table', y = 'rate', data = df)
```

```
<AxesSubplot:xlabel='book_table', ylabel='rate'>
```



- **Here is the big difference as we can see here restaurants with book_table facility, there average rating is higher as compared to restaurants with no such facility**
- **Therefore If opening a restaurant one should offer the facility to have good business**

Visualizing the online_Order facility, location-wise

```
df1 = df.groupby(['location','online_order'])['name'].count()
df1.to_csv('location_online.csv')
df1 = pd.read_csv('location_online.csv')
df1 = pd.pivot_table(df1,values = None, index = ['location'],columns =
['online_order'], fill_value = 0, aggfunc = np.sum)
df1
```

```
                      name
online_order            No   Yes
location
BTM                   1763  3293
```

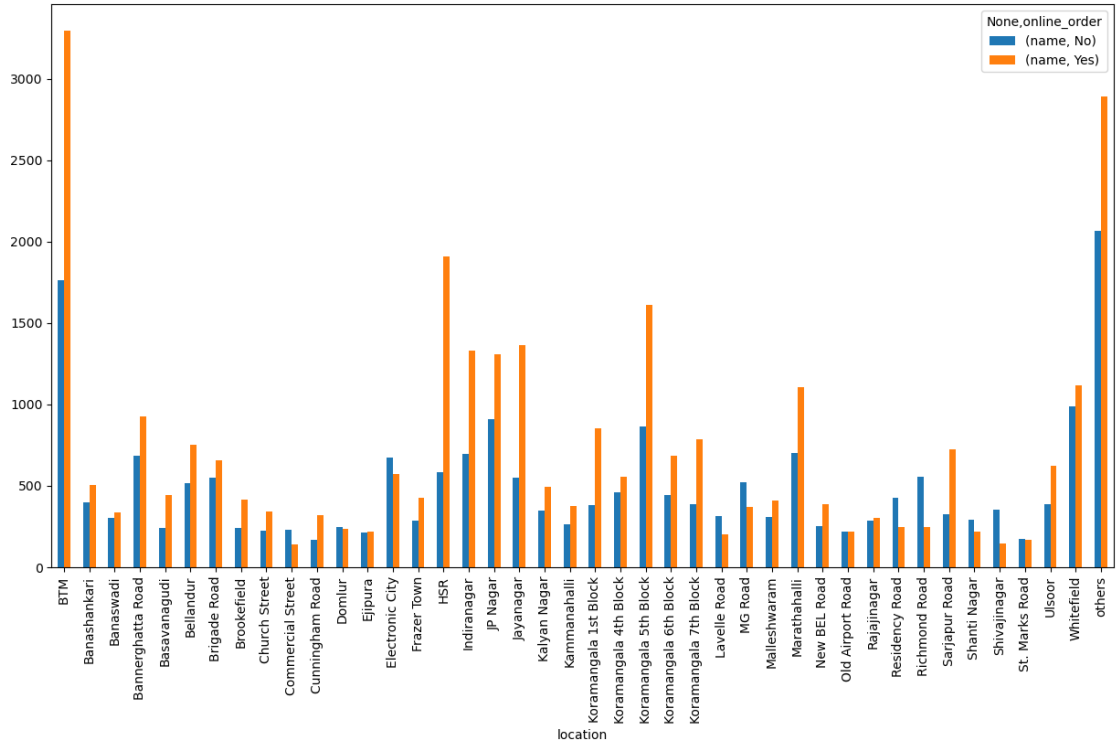
```
Banashankari             397   505
Banaswadi                302   338
Bannerghatta Road        685   924
Basavanagudi             243   441
Bellandur                517   751
Brigade Road             552   658
Brookefield              239   417
Church Street            226   340
Commercial Street        228   142
Cunningham Road          168   322
Domlur                   247   235
Ejipura                  214   219
Electronic City          676   570
Frazer Town              287   427
HSR                      584  1910
Indiranagar              697  1329
JP Nagar                 911  1307
Jayanagar                552  1364
Kalyan Nagar             350   491
Kammanahalli             264   375
Koramangala 1st Block    384   852
Koramangala 4th Block    459   558
Koramangala 5th Block    866  1613
Koramangala 6th Block    445   682
Koramangala 7th Block    389   785
Lavelle Road             315   203
MG Road                  520   373
Malleshwaram             309   412
Marathahalli             701  1104
New BEL Road             255   389
Old Airport Road         221   216
Rajajinagar              286   305
Residency Road           424   247
Richmond Road            557   246
Sarjapur Road            323   724
Shanti Nagar             289   219
Shivajinagar             354   144
St. Marks Road           176   167
Ulsoor                   389   622
Whitefield               986  1119
others                  2064  2890
```

* Here by looking we can conclude that which region restaurants are having more chances of online ordering restaurants

- **we can visualize the data using below plot**

```
df1.plot(kind = 'bar', figsize = (15,8))
```

```
<AxesSubplot:xlabel='location'>
```



- **We can conclude that in BTM there are mostly online ordering restaurants**
- **In Lavelle road the online ordering facility is less as compared to other places**
- **Also we do know that online ordering restaurants outshines the other ones so....**
- **If anyone needs to open a new restaurant he will open a restaurant in places like lavelle road and with online ordering facility to grow the business in the market**

let us Now Visualize book table facility, Location wise

```
df2 = df.groupby(['location','book_table'])['name'].count()
df2.to_csv('location_booktable.csv')
df2 = pd.read_csv('location_booktable.csv')
df2 = pd.pivot_table(df2,values = None, index = ['location'],columns =
['book_table'], fill_value = 0, aggfunc = np.sum)
df2
```

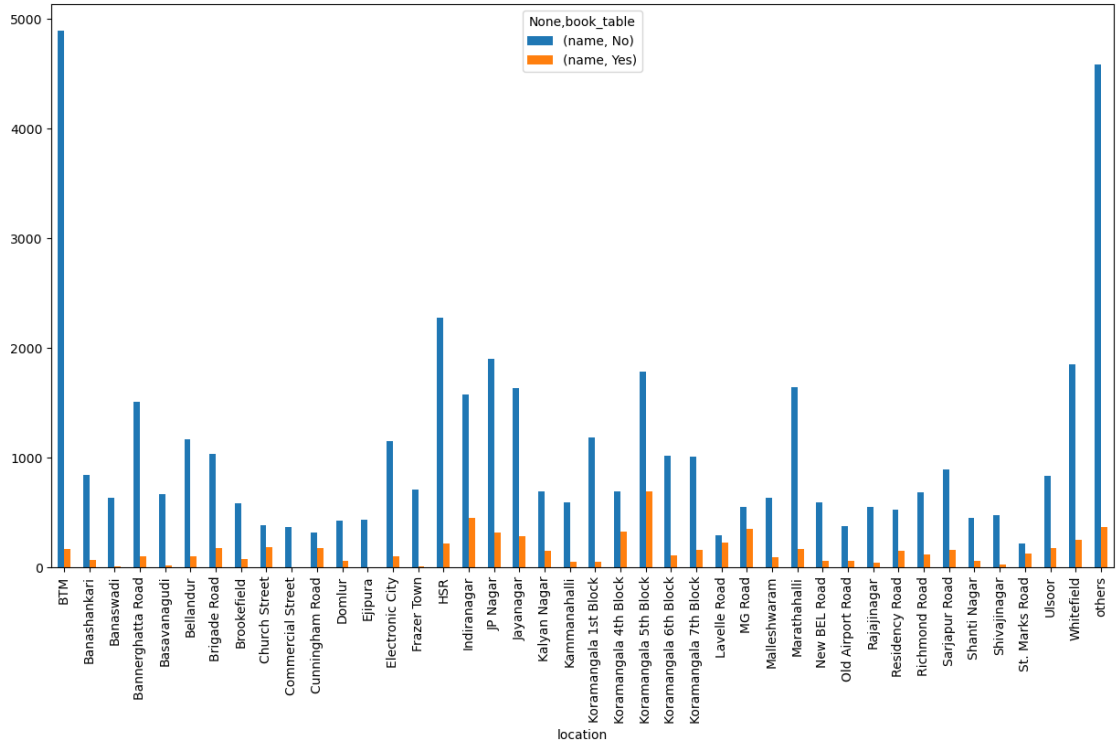
```
                      name
book_table              No  Yes
location
BTM                   4889  167
Banashankari           839   63
Banaswadi              632    8
Bannerghatta Road     1510   99
Basavanagudi           668   16
Bellandur             1170   98
```

```
Brigade Road           1034  176
Brookefield             582   74
Church Street           385  181
Commercial Street       370    0
Cunningham Road         315  175
Domlur                  427   55
Ejipura                 433    0
Electronic City        1148   98
Frazer Town             706    8
HSR                    2277  217
Indiranagar            1578  448
JP Nagar               1903  315
Jayanagar              1637  279
Kalyan Nagar            692  149
Kammanahalli            590   49
Koramangala 1st Block  1186   50
Koramangala 4th Block   695  322
Koramangala 5th Block  1787  692
Koramangala 6th Block  1015  112
Koramangala 7th Block  1012  162
Lavelle Road            290  228
MG Road                 546  347
Malleshwaram            632   89
Marathahalli           1642  163
New BEL Road            588   56
Old Airport Road        378   59
Rajajinagar             550   41
Residency Road          522  149
Richmond Road           687  116
Sarjapur Road           893  154
Shanti Nagar            451   57
Shivajinagar            475   23
St. Marks Road          219  124
Ulsoor                  834  177
Whitefield             1852  253
others                 4587  367
```

so let's visualize it with a plot

```
df2.plot(kind = 'bar', figsize = (15,8))
```

```
<AxesSubplot:xlabel='location'>
```



- **In BTM many restaturants don't provide the booking facility and thus it is anice chance to open arestaurant there with booking facility**
- **But there's a twist BTM already have ample amount of restaurants there so we might not get a chance there as a new brand**
- **Instead we can try opening a restaurant in following places as there also restaurants with booking facility are relatively low and we can find opportunity to grow our business :-**

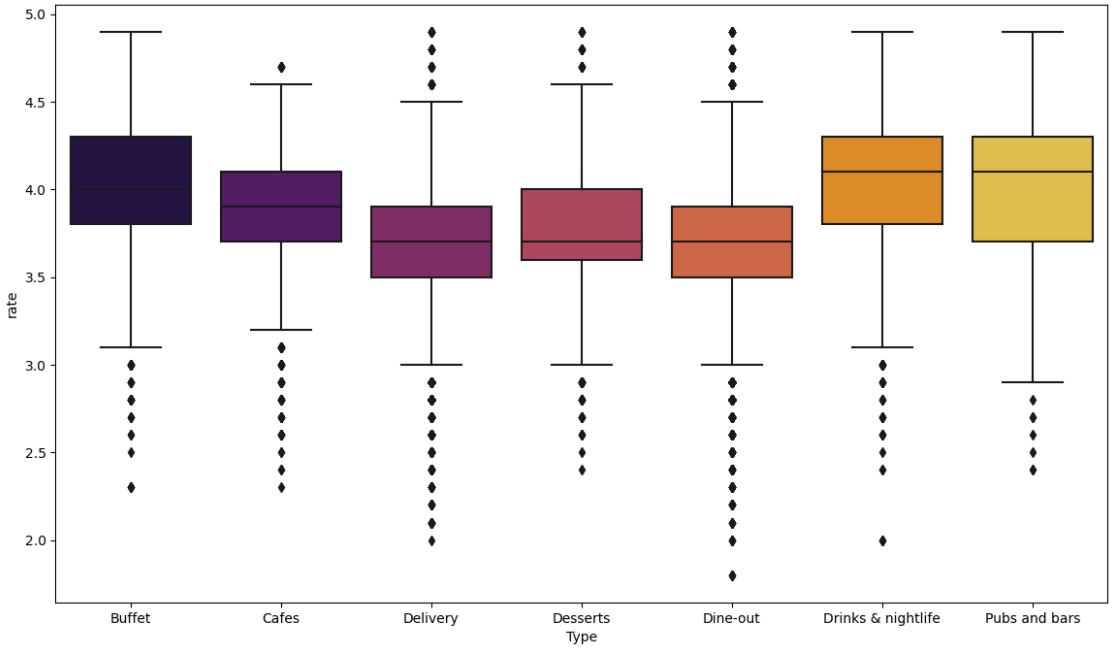
1. **HSR**
2. **Electronic city**
3. **Whitefield**

- **Less competition = More Profit**
- **If a coouple wants to book a table they will be buying from restaurant setup in HSR**

Visualizing types of restaurants VS Rate

```
plt.figure(figsize = (14,8))
sns.boxplot(x = 'Type', y = 'rate', data = df ,palette = 'inferno')
```

```
<AxesSubplot:xlabel='Type', ylabel='rate'>
```



- **Drinks and Nightlife restaurants are the most ranked and delivery restaurants are the least ranked**
- **So if we need to open a restaurant in any field we are more likely to open up as :-**
    a. ***Drinks and nightlife***
    b. ***Pubs and bars***
    c. ***Buffet***

Grouping types of restaurants, location wise

```
df3 = df.groupby(['location','Type'])['name'].count()
df3.to_csv('location_Type.csv')
df3 = pd.read_csv('location_Type.csv')
df3 = pd.pivot_table(df3, values = None, index = ['location'], columns
= ['Type'], fill_value = 0, aggfunc = np.sum)
df3
```

	name					\
Type	Buffet	Cafes	Delivery	Desserts	Dine-out	
location						
BTM	21	83	3053	198	1660	
Banashankari	7	36	418	71	356	
Banaswadi	0	24	310	37	262	
Bannerghatta Road	9	46	828	137	578	
Basavanagudi	7	11	344	66	251	
Bellandur	28	36	617	75	479	
Brigade Road	25	46	497	108	455	
Brookefield	6	17	339	45	245	
Church Street	19	51	193	29	215	

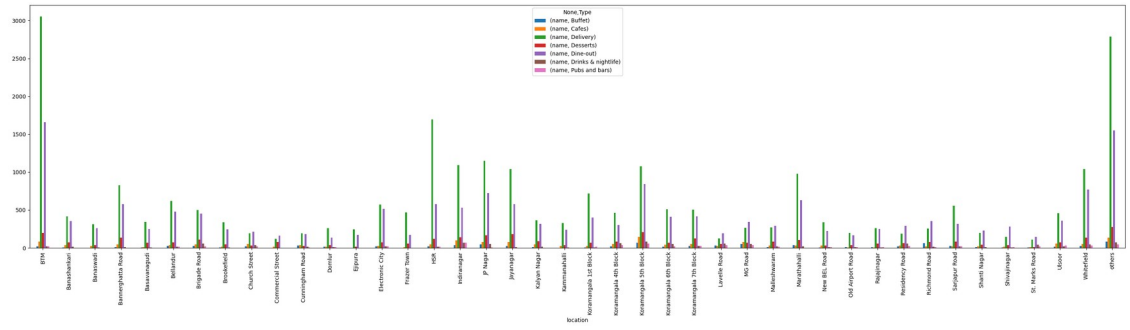
Commercial Street	0	13	121	77	159
Cunningham Road	29	34	194	26	184
Domlur	15	13	261	35	135
Ejipura	0	0	245	16	172
Electronic City	23	24	570	71	516
Frazer Town	1	11	470	56	172
HSR	19	49	1694	120	580
Indiranagar	38	97	1091	140	529
JP Nagar	45	76	1151	166	722
Jayanagar	27	77	1043	182	575
Kalyan Nagar	9	45	366	88	315
Kammanahalli	2	27	329	35	240
Koramangala 1st Block	3	26	716	70	398
Koramangala 4th Block	21	53	464	81	302
Koramangala 5th Block	65	146	1075	209	842
Koramangala 6th Block	18	43	511	70	411
Koramangala 7th Block	25	52	503	127	417
Lavelle Road	30	27	127	50	191
MG Road	51	76	266	68	343
Malleshwaram	11	31	269	85	291
Marathahalli	34	32	980	105	630
New BEL Road	4	29	338	33	224
Old Airport Road	12	5	200	35	164
Rajajinagar	10	4	258	55	251
Residency Road	20	31	187	63	289
Richmond Road	63	21	257	78	356
Sarjapur Road	25	22	558	82	319
Shanti Nagar	9	22	198	39	229
Shivajinagar	6	17	143	37	280
St. Marks Road	5	10	111	10	145
Ulsoor	16	56	456	71	359
Whitefield	28	51	1041	137	768
others	83	133	2787	276	1553

Type	Drinks & nightlife	Pubs and bars
location		
BTM	22	19
Banashankari	14	0
Banaswadi	6	1
Bannerghatta Road	9	2
Basavanagudi	5	0
Bellandur	17	16
Brigade Road	57	22
Brookefield	4	0
Church Street	36	23
Commercial Street	0	0
Cunningham Road	16	7
Domlur	12	11
Ejipura	0	0

```
Electronic City                  21           21
Frazer Town                       2            2
HSR                              14           18
Indiranagar                      65           66
JP Nagar                         51            7
Jayanagar                        12            0
Kalyan Nagar                     18            0
Kammanahalli                      6            0
Koramangala 1st Block             7           16
Koramangala 4th Block            62           34
Koramangala 5th Block            84           58
Koramangala 6th Block            51           23
Koramangala 7th Block            25           25
Lavelle Road                     59           34
MG Road                          53           36
Malleshwaram                     20           14
Marathahalli                     22            2
New BEL Road                      8            8
Old Airport Road                 12            9
Rajajinagar                       3           10
Residency Road                   55           26
Richmond Road                    16           12
Sarjapur Road                    19           22
Shanti Nagar                      9            2
Shivajinagar                      7            8
St. Marks Road                   40           22
Ulsoor                           23           30
Whitefield                       47           33
others                           75           47
```

```
df3.plot(kind = 'bar', figsize = (36,8))
```

```
<AxesSubplot:xlabel='location'>
```



So if we want to open a pub and bar we can easily go for shantinagar / shivajinagar as they are having the least amount of pubs and bar

- **hence less competition = more profit**

No. of votes, location wise

```
df4 = df[['location','votes']]
df4.drop_duplicates()
df5 = df4.groupby(['location'])['votes'].sum()
df5 = df5.to_frame()
df5 = df5.sort_values('votes', ascending = False)
df5.head()
```

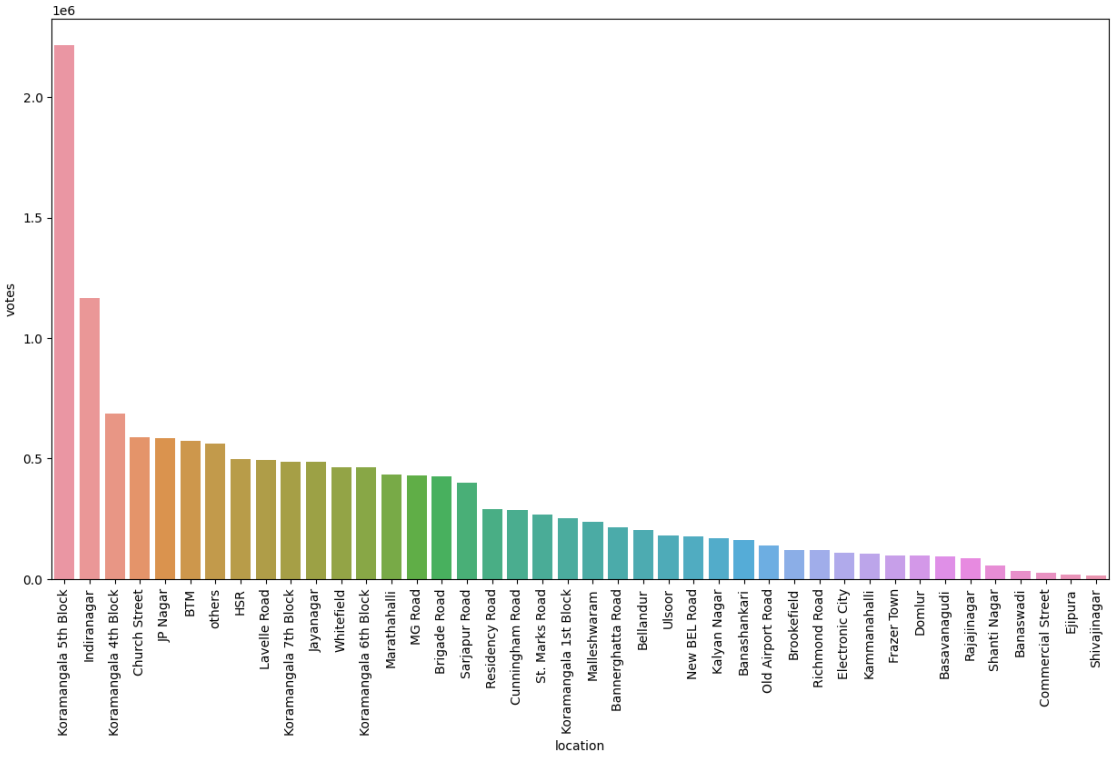
```
                         votes
location
Koramangala 5th Block  2214083
Indiranagar            1165909
Koramangala 4th Block   685156
Church Street           590306
JP Nagar                586522
```

I am interested in which location people are actually voting as people's feedback is the most crucial thing in this type of market

```
plt.figure(figsize=(15, 8))
sns.barplot(x=df5.index, y='votes', data=df5)
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33,
        34, 35, 36, 37, 38, 39, 40, 41]),
 [Text(0, 0, 'Koramangala 5th Block'),
  Text(1, 0, 'Indiranagar'),
  Text(2, 0, 'Koramangala 4th Block'),
  Text(3, 0, 'Church Street'),
  Text(4, 0, 'JP Nagar'),
  Text(5, 0, 'BTM'),
  Text(6, 0, 'others'),
  Text(7, 0, 'HSR'),
  Text(8, 0, 'Lavelle Road'),
  Text(9, 0, 'Koramangala 7th Block'),
  Text(10, 0, 'Jayanagar'),
  Text(11, 0, 'Whitefield'),
  Text(12, 0, 'Koramangala 6th Block'),
  Text(13, 0, 'Marathahalli'),
  Text(14, 0, 'MG Road'),
  Text(15, 0, 'Brigade Road'),
  Text(16, 0, 'Sarjapur Road'),
  Text(17, 0, 'Residency Road'),
  Text(18, 0, 'Cunningham Road'),
  Text(19, 0, 'St. Marks Road'),
  Text(20, 0, 'Koramangala 1st Block'),
```

```
Text(21, 0, 'Malleshwaram'),
Text(22, 0, 'Bannerghatta Road'),
Text(23, 0, 'Bellandur'),
Text(24, 0, 'Ulsoor'),
Text(25, 0, 'New BEL Road'),
Text(26, 0, 'Kalyan Nagar'),
Text(27, 0, 'Banashankari'),
Text(28, 0, 'Old Airport Road'),
Text(29, 0, 'Brookefield'),
Text(30, 0, 'Richmond Road'),
Text(31, 0, 'Electronic City'),
Text(32, 0, 'Kammanahalli'),
Text(33, 0, 'Frazer Town'),
Text(34, 0, 'Domlur'),
Text(35, 0, 'Basavanagudi'),
Text(36, 0, 'Rajajinagar'),
Text(37, 0, 'Shanti Nagar'),
Text(38, 0, 'Banaswadi'),
Text(39, 0, 'Commercial Street'),
Text(40, 0, 'Ejipura'),
Text(41, 0, 'Shivajinagar')])
```



- **By this above plot we got to know that people are giving votes mostly to Koramangala 5th block and then reducing the votes to least amount in Shivajinagar and Ejipura**
- **So we can conclude that people are not interested in shivajinagar/Ejipura/Banaswadi like places, rather they are interested in**

places like Koramangala 5th block/ Koramangala 4th block / Church street / Indiranagar

- **So if we want to open a restaurant it is more likely to be opening at the places with most likely votes and thus Koramnagla 5th block area suits the best to get people's attention**
- **Thus if we open a restaurnat in koramangala 5th block we would most likely to get the most customer feedback**

```
df.head()
```

```
                    name online_order book_table  rate  votes
location  \
0                  Jalsa          Yes        Yes   4.1    775
Banashankari
1         Spice Elephant          Yes         No   4.1    787
Banashankari
2        San Churro Cafe          Yes         No   3.8    918
Banashankari
3  Addhuri Udupi Bhojana           No         No   3.7     88
Banashankari
4          Grand Village           No         No   3.8    166
Basavanagudi

       rest_type                        cuisines  Cost2plates    Type

0  Casual Dining  North Indian, Mughlai, Chinese        800.0  Buffet

1  Casual Dining                          others        800.0  Buffet

2         others                          others        800.0  Buffet

3    Quick Bites      South Indian, North Indian        300.0  Buffet

4  Casual Dining                          others        600.0  Buffet
```

Now we are interested in finding out which cuisine restaurant is suitable for opening

visualizing Top Cuisines

```
df6 = df[['cuisines','votes']]
df6.drop_duplicates()
df7 = df6.groupby(['cuisines'])['votes'].sum()
df7 = df7.to_frame()
df7 = df7.sort_values('votes',ascending = False)
df7.head()
```

```
                          votes
cuisines
others                  11542182
North Indian              516310
North Indian, Chinese     258225
South Indian              161975
North Indian, Mughlai     103706
```

- **North Indian cuisine is having the highest no.of votes**
- **And restaurants with mixed varities of not=rth indian and chinese food are the 2nd highest**
- **South indian cuisine is geting the 3rd highest no.of votes**

To visualize this data we are not taking others in our plotting as it doesn't provide us any data in particular , So we use iloc function to start our plotting from index 1

```
df7 = df7.iloc[1:, :]
df7.head()
```

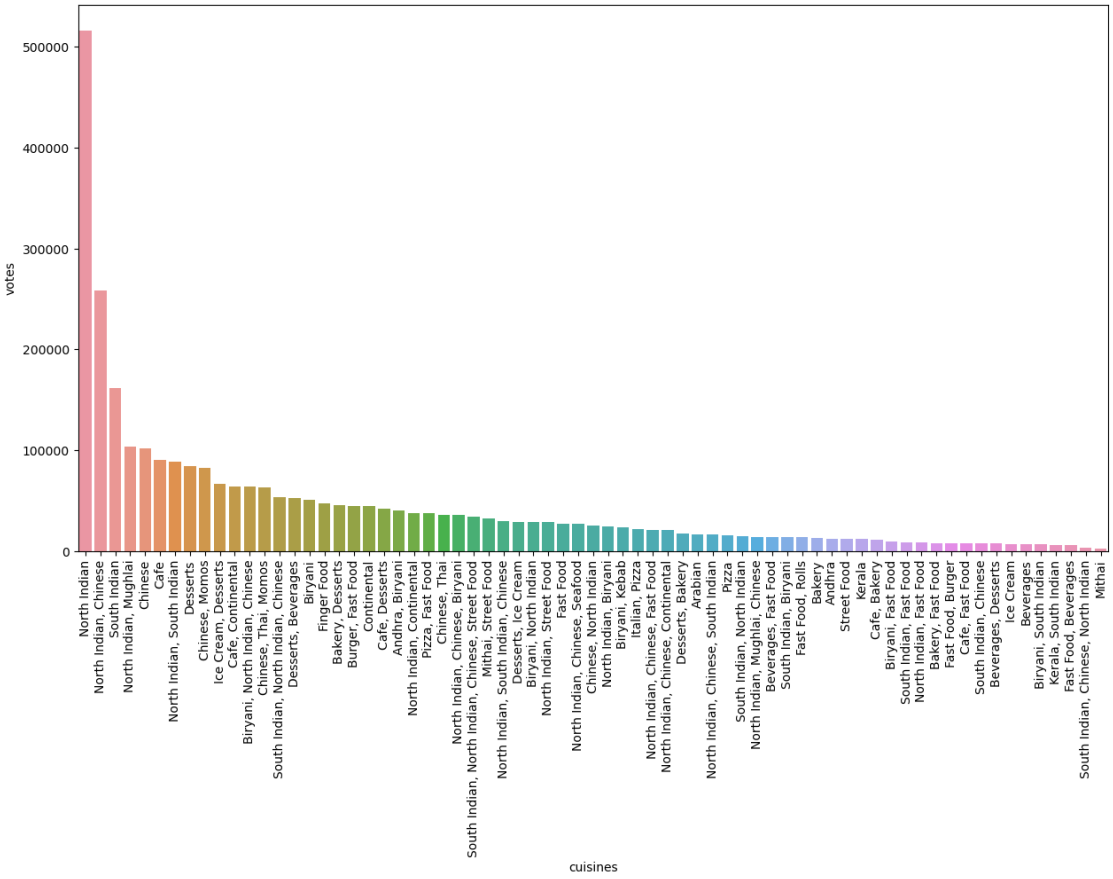
```
                         votes
cuisines
North Indian            516310
North Indian, Chinese   258225
South Indian            161975
North Indian, Mughlai   103706
Chinese                 101728
```

```
plt.figure(figsize = (15,8))
sns.barplot(x = df7.index, y = 'votes', data = df7)
plt.xticks(rotation = 90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
66, 67,
        68]),
 [Text(0, 0, 'North Indian'),
  Text(1, 0, 'North Indian, Chinese'),
  Text(2, 0, 'South Indian'),
  Text(3, 0, 'North Indian, Mughlai'),
  Text(4, 0, 'Chinese'),
  Text(5, 0, 'Cafe'),
  Text(6, 0, 'North Indian, South Indian'),
  Text(7, 0, 'Desserts'),
```

```
Text(8, 0, 'Chinese, Momos'),
Text(9, 0, 'Ice Cream, Desserts'),
Text(10, 0, 'Cafe, Continental'),
Text(11, 0, 'Biryani, North Indian, Chinese'),
Text(12, 0, 'Chinese, Thai, Momos'),
Text(13, 0, 'South Indian, North Indian, Chinese'),
Text(14, 0, 'Desserts, Beverages'),
Text(15, 0, 'Biryani'),
Text(16, 0, 'Finger Food'),
Text(17, 0, 'Bakery, Desserts'),
Text(18, 0, 'Burger, Fast Food'),
Text(19, 0, 'Continental'),
Text(20, 0, 'Cafe, Desserts'),
Text(21, 0, 'Andhra, Biryani'),
Text(22, 0, 'North Indian, Continental'),
Text(23, 0, 'Pizza, Fast Food'),
Text(24, 0, 'Chinese, Thai'),
Text(25, 0, 'North Indian, Chinese, Biryani'),
Text(26, 0, 'South Indian, North Indian, Chinese, Street Food'),
Text(27, 0, 'Mithai, Street Food'),
Text(28, 0, 'North Indian, South Indian, Chinese'),
Text(29, 0, 'Desserts, Ice Cream'),
Text(30, 0, 'Biryani, North Indian'),
Text(31, 0, 'North Indian, Street Food'),
Text(32, 0, 'Fast Food'),
Text(33, 0, 'North Indian, Chinese, Seafood'),
Text(34, 0, 'Chinese, North Indian'),
Text(35, 0, 'North Indian, Biryani'),
Text(36, 0, 'Biryani, Kebab'),
Text(37, 0, 'Italian, Pizza'),
Text(38, 0, 'North Indian, Chinese, Fast Food'),
Text(39, 0, 'North Indian, Chinese, Continental'),
Text(40, 0, 'Desserts, Bakery'),
Text(41, 0, 'Arabian'),
Text(42, 0, 'North Indian, Chinese, South Indian'),
Text(43, 0, 'Pizza'),
Text(44, 0, 'South Indian, North Indian'),
Text(45, 0, 'North Indian, Mughlai, Chinese'),
Text(46, 0, 'Beverages, Fast Food'),
Text(47, 0, 'South Indian, Biryani'),
Text(48, 0, 'Fast Food, Rolls'),
Text(49, 0, 'Bakery'),
Text(50, 0, 'Andhra'),
Text(51, 0, 'Street Food'),
Text(52, 0, 'Kerala'),
Text(53, 0, 'Cafe, Bakery'),
Text(54, 0, 'Biryani, Fast Food'),
Text(55, 0, 'South Indian, Fast Food'),
Text(56, 0, 'North Indian, Fast Food'),
Text(57, 0, 'Bakery, Fast Food'),
```

```
Text(58, 0, 'Fast Food, Burger'),
Text(59, 0, 'Cafe, Fast Food'),
Text(60, 0, 'South Indian, Chinese'),
Text(61, 0, 'Beverages, Desserts'),
Text(62, 0, 'Ice Cream'),
Text(63, 0, 'Beverages'),
Text(64, 0, 'Biryani, South Indian'),
Text(65, 0, 'Kerala, South Indian'),
Text(66, 0, 'Fast Food, Beverages'),
Text(67, 0, 'South Indian, Chinese, North Indian'),
Text(68, 0, 'Mithai')])
```



- **So from here we got to know that as we had already mentioned North indian is the highly voted one following up chinese + north indian is the 2nd and south indian is 3rd**
- **if anyone wanted to open a restaurant as certain speciality they most likely to go for north indian cuisines due to its high demand**
- **We can also see that cuisines like mithai and biryani are not that famous along with icecreams which shows us that people tend to move towards a healthy lifestyle where sugar is not there test and they also reduced the fast food consumption**

- **However chinese food is growing in demand**

But all in all the restaurants are dominated by the North indian cuisines

Now we are moving ahead to apply classifiers to complete our prediction

Convert the 'Aggregate rating' column to binary values indicating whether the restaurant is good or not (good = rating > 3.5)

```
df['Good Restaurant'] = df['rate'].apply(lambda x: 1 if x > 3.5 else
0)
```

Splitting the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(df[['votes',
'Cost2plates', 'online_order', 'book_table', 'cuisines', 'location']],
df['Good Restaurant'], test_size=0.2, random_state=42)
```

Preprocessing the Data we have

```
numeric_features = ['votes', 'Cost2plates']
numeric_transformer = StandardScaler()

categorical_features = ['online_order', 'book_table', 'cuisines',
'location']
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

preprocessor = ColumnTransformer(transformers=[('num',
numeric_transformer, numeric_features),('cat',
categorical_transformer, categorical_features)])

X_train_preprocessed = preprocessor.fit_transform(X_train)
X_test_preprocessed = preprocessor.transform(X_test)
```

Using Decision tree classifier and training the classifier

```
dt = DecisionTreeClassifier(random_state = 42)
dt.fit(X_train_preprocessed, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
dt.score(X_train_preprocessed , y_train)
```

```
0.9949060808659662
```

Evaluating the decision tree classifier on the testing set

```
pred_dt = dt.predict(X_test_preprocessed)
print('Decision Tree Classifier Accuracy:', accuracy_score(y_test,
pred_dt))
print(classification_report(y_test, pred_dt))
```

```
Decision Tree Classifier Accuracy: 0.9710059751199922
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      2772
           1       0.98      0.98      0.98      7437

    accuracy                           0.97     10209
   macro avg       0.96      0.96      0.96     10209
weighted avg       0.97      0.97      0.97     10209
```

The accuracy of Decision Tree model is 97.1%

```
dt = DecisionTreeClassifier(max_depth=5, random_state=42)
dt.fit(X_train_preprocessed, y_train)
```

```
DecisionTreeClassifier(max_depth=5, random_state=42)
```

Using Random forest classifier and training the classifier

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_preprocessed, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
rf.score(X_train_preprocessed, y_train)
```

```
0.9949060808659662
```

Evaluating the Random forest classifier on the testing set

```
pred_rf = rf.predict(X_test_preprocessed)
rf.score(X_test_preprocessed, y_test)
```

```
0.9766872367518856
```

```
print('Random Forest Classifier Accuracy:', accuracy_score(y_test,
pred_rf))
print(classification_report(y_test, pred_rf))
```

```
Random Forest Classifier Accuracy: 0.9766872367518856
              precision    recall  f1-score   support

           0       0.96      0.95      0.96      2772
           1       0.98      0.99      0.98      7437

    accuracy                           0.98     10209
```

```
   macro avg       0.97      0.97      0.97     10209
weighted avg       0.98      0.98      0.98     10209
```

The accuracy of Random forest model is 97.6%

Inference

* Decision tree Accuracy = 97.1%

* Random Forest Accuracy = 97.6%

Hence Random forest is comparitively better

Using Regressor to classify our data

Choose a decision tree regressor and train the regressor

```
dt = DecisionTreeRegressor(random_state=42)
dt.fit(X_train_preprocessed, y_train)
```

```
DecisionTreeRegressor(random_state=42)
```

Evaluate the decision tree regressor on the testing set

```
pred_dt = dt.predict(X_test_preprocessed)
print('Decision Tree Regressor Mean Absolute Error:',
mean_absolute_error(y_test, pred_dt))
print('Decision Tree Regressor Mean Squared Error:',
mean_squared_error(y_test, pred_dt))
```

```
Decision Tree Regressor Mean Absolute Error: 0.030145025484921654
Decision Tree Regressor Mean Squared Error: 0.02685341356244725
```

* Here we can see that Mean Absolute error is 3.01%

* and Mean Squared error is 2.68%

* As we know that lower the MAE & MSE then, Higher the chances of accuracy of the Model

```
dt = DecisionTreeRegressor(max_depth=5, random_state=42)
dt.fit(X_train_preprocessed, y_train)
```

```
DecisionTreeRegressor(max_depth=5, random_state=42)
```

Choose a random forest regressor and train the regressor

```
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train_preprocessed, y_train)
```

```
RandomForestRegressor(random_state=42)
```

Evaluate the random forest regressor on the testing set

```
pred_rf = rf.predict(X_test_preprocessed)
print('Random Forest Regressor Mean Absolute Error:',
mean_absolute_error(y_test, pred_rf))
print('Random Forest Regressor Mean Squared Error:',
mean_squared_error(y_test, pred_rf))
```

```
Random Forest Regressor Mean Absolute Error: 0.04897758978375219
Random Forest Regressor Mean Squared Error: 0.02146084987114553
```

* **Here we can see that Mean Absolute error is 4.89%**

* **and Mean Squared error is 2.14%**

* **As we know that lower the MAE & MSE then, Higher the chances of accuracy of the Model**

Important Conclusions :-

1. In case of Decision tree regressor, the MAE is 3.01% ### 2. In case of Random forest regressor, the MAE is 4.89% ### 3. The lower the MAE, the better the model is performing ### 4. MAE(Decision tree) < MAE(random forest) # Hence, Decision tree Regressor model is more accurate than Random forest Regressor model

Accuracy of model

1.) As a classifier, Random forest model is the best

2.) As a Regressor, Decision tree model is the best