Predicting Zomato Restaurants Rate

1. Business Problem

1.1 Problem Description

Restaurants from all over the world can be found here in Bengaluru. From United States to Japan, Russia to Antarctica, you get all type of cuisines here. Delivery, Dine-out, Pubs, Bars, Drinks,Buffet, Desserts you name it and Bengaluru has it. Bengaluru is best place for foodies. The number of restaurant are increasing day by day. Currently which stands at approximately 12,000 restaurants. With such an high number of restaurants. This industry hasn't been saturated yet. And new restaurants are opening every day. However it has become difficult for them to compete with already established restaurants. The key issues that continue to pose a challenge to them include high real estate costs, rising food costs, shortage of quality manpower, fragmented supply chain and over-licensing. This Zomato data aims at analysing demography of the location. Most importantly it will help new restaurants in deciding their theme, menus, cuisine, cost etc for a particular location. It also aims at finding similarity between neighborhoods of Bengaluru on the basis of food.

Does demography of area matters? Does location of particular type of restraurant depends on people living in that area> Does theme of restraurant matters? Is food chain category restraurant likely to have more customers than its counter part? Are any neighbourhood on similar based on the type of food? Is particular neighbours is famous for itw own kind of food? If two neighbours are similar does that mean these are related or particular group of people live in neighbourhood or these are places to eat. What kind of food is famous in locality. Do entire locality loves veg food, if yes then locality populated by particular set of people eg Jain, Gujarati,Marwadi who are basically veg.

1.2 Problem Statement

The dataset also contains reviews for each of the restaurant which will help in finding overall rating for the place. So we will try to predict rating for particular restaurant.

1.3 Real world/Business Objectives

We need to predict rating based on different parameters like Average_cost for two people, Online Order available, foods,menu list, most liked dishes etc features.

1.4 Machine Learning Formulation

Here we suppose to predicted rating of restaurant, so it is basically Regression problem.

1.5 Perfomance Metric

We will try to reduce Mean Square Error ie MSE as minimum as possible. So it is Regression problem reducing MSE.

Ideal MSE is 0.

2. Machine Learning Problem

2.1 Data

Data Acquire https://www.kaggle.com/himanshupoddar/zomato-bangalore-restaurants

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Wi
```

```
--2020-10-15 10:18:43--  https://storage.googleapis.com/kaggle-data-sets/153420/352891/compres
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.195.128, 74.125.142.128, 74
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.195.128|:443... connected
HTTP request sent, awaiting response... 200 OK
Length: 93341357 (89M) [application/zip]
Saving to: ‘zomato.csv.zip’

zomato.csv.zip      100%[===================>]  89.02M  56.5MB/s    in 1.6s

2020-10-15 10:18:45 (56.5 MB/s) - ‘zomato.csv.zip’ saved [93341357/93341357]
```

```
!unzip zomato.csv.zip
!ls
```

```
Archive:  zomato.csv.zip
  inflating: zomato.csv
sample_data  zomato.csv  zomato.csv.zip
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('nBagg')

import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import  make_scorer
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from joblib import dump,load
from wordcloud import WordCloud,STOPWORDS
import ast #Abstract Syntax Trees
```

```
data=pd.read_csv('zomato.csv')
```

```
data.head()
```

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

```
data.shape
```

```
(51717, 17)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   url                          51717 non-null  object
 1   address                      51717 non-null  object
 2   name                         51717 non-null  object
 3   online_order                 51717 non-null  object
 4   book_table                   51717 non-null  object
 5   rate                         43942 non-null  object
 6   votes                        51717 non-null  int64
 7   phone                        50509 non-null  object
 8   location                     51696 non-null  object
 9   rest_type                    51490 non-null  object
 10  dish_liked                   23639 non-null  object
 11  cuisines                     51672 non-null  object
 12  approx_cost(for two people)  51371 non-null  object
 13  reviews_list                 51717 non-null  object
 14  menu_item                    51717 non-null  object
 15  listed_in(type)              51717 non-null  object
 16  listed_in(city)              51717 non-null  object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

- Observation
    - Rate, dish_liked, phone , approx_cost(for two people) values are missing.

```
data.votes.describe()
```

```
count    51717.000000
mean       283.697527
std        803.838853
min          0.000000
25%          7.000000
50%         41.000000
75%        198.000000
max      16832.000000
Name: votes, dtype: float64
```

- Minimum votes value is 0, can be interpreted as there are some restaurants which have 0 votes
- Maximum votes value is 16832, there is a restaurant which has 16832.
- Average votes value is 284

```
data.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')
```

Columns description

- url: contains the url of the restaurant in the zomato website
- address: contains the address of the restaurant in Bengaluru
- name: contains the name of the restaurant
- online_order: whether online ordering is available in the restaurant or not
- book_table: table book option available or not
- rate: contains the overall rating of the restaurant out of 5
- votes: contains total number of rating for the restaurant as of the above mentioned date
- phone: contains the phone number of the restaurant
- location: contains the neighborhood in which the restaurant is located
- rest_type: restaurant type
- dish_liked: dishes people liked in the restaurant
- cuisines: food styles, separated by comma
- approx_cost(for two people): contains the approximate cost for meal for two people
- reviews_list: list of tuples containing reviews for the restaurant, each tuple
- menu_item: contains list of menus available in the restaurant
- listed_in(type): type of meal
- listed_in(city): contains the neighborhood in which the restaurant is listed

Data Preprocessing

Adjust column names and dropped irrelevant columns

```
data.loc[:,['url','phone','name','location','address', 'listed_in(city)']].sample(10)
```

	url	phone	name	location	
12398	https://www.zomato.com/bangalore/ammas-pastrie...	+91 9590607750	Amma's Pastries	Ulsoor	S Aven Ulso
32627	https://www.zomato.com/bangalore/sichuan-hsr?c...	+91 8151800777\r\n+91 8152800777	Sichuan	HSR	15 Mai 1, Inc
44862	https://www.zomato.com/bangalore/bhookha-sher-...	+91 7483396948	Bhookha Sher	Old Airport Road	Towe Di

Here, we can see that 3 columns are representing same information, so just dropping column which are not important.

we are going to keep the location column and drop the address and listed_in(city) columns columns url , phone , we are not interested in ,to be dropped too

```
drop_col=['url','phone','address', 'listed_in(city)']
data.drop(drop_col,axis=1,inplace=True)
```

```
data.head()
```

	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisin
0	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	No India Mughl Chine
1	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chines No India Th

```
data.duplicated().sum()
```

```
data.duplicated().sum()
```

```
9809
```

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

Removing Null values

```
data.shape
```

```
(41908, 13)
```

Removing Duplicates

```
((data.isna().sum()/data.shape[0])*100).round(2)
```

```
name                            0.00
online_order                    0.00
book_table                      0.00
rate                           10.15
votes                           0.00
location                        0.03
rest_type                       0.41
dish_liked                     48.22
cuisines                        0.09
approx_cost(for two people)     0.60
reviews_list                    0.00
menu_item                       0.00
listed_in(type)                 0.00
dtype: float64
```

Double-click (or enter) to edit

Observation:

- We can oberve that 48% dish_liked is missing as well as 10% rate values are missing.
- If we drop everything out, we will lose more than 55% points.

```
d=data.rate
```

```
#lets examine rate column
d.value_counts()
```

```
3.9/5     1858
3.7/5     1711
3.8/5     1703
3.9 /5    1663
NEW       1593
          ...
2.0 /5       6
2.2 /5       5
2.0/5        4
1.8 /5       3
```

```
1.8/5        2
Name: rate, Length: 64, dtype: int64
```

```
d.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)
```

There are some points which has 'NEW' rating and '-' rating, which is completely incorrect.

```
d=d.replace('NEW',np.nan)
d=d.replace('-',np.nan)
```

```
data['rate']=d.str.replace(r'/5| /5', '')
```

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

```
#if we look closely at the reviews_list data we can clearly see that it has rating values
data.reviews_list.values[:10]
```

```
array(['[(\'Rated 4.0\', \'RATED\\n  A beautiful place to dine in.The interiors take you back t
       '[(\'Rated 4.0\', \'RATED\\n  Had been here for dinner with family. Turned out to be a g
       '[(\'Rated 3.0\', "RATED\\n  Ambience is not that good enough and it\'s not a pocket fri
       '[(\'Rated 4.0\', "RATED\\n  Great food and proper Karnataka style full meals. Been ther
       "[('Rated 4.0', 'RATED\\n  Very good restaurant in neighbourhood. Buffet system is prope
       "[('Rated 3.0', 'RATED\\n  Food 3/5\\nAmbience 3/5\\nService 3/5\\n\\nHad been here for
       "[('Rated 5.0', 'RATED\\n  Awesome food ??Great serviceFriendly staffsGood quality of fo
       '[(\'Rated 5.0\', \'RATED\\n  I personally really liked this place ! The ambience with t
       '[(\'Rated 3.0\', "RATED\\n  I had been to this place with one of my friends, it\'s a ve
       '[(\'Rated 4.0\', "RATED\\n  Easy to locate\\nVFM 3.5/5\\nTaste 5/5\\nYummy cheesyyy fri
      dtype=object)
```

```
#we could extract these values from reviews and take their mean to fill rate column
data.reviews_list.values[1]
```

```
'[(\'Rated 4.0\', \'RATED\\n  Had been here for dinner with family. Turned out to be a good ch
oose suitable for all ages of people. Can try this place. We liked the most was their starter
s. Service is good. Prices are affordable. Will recommend this restaurant for early dinner. Th
e place is little noisy.\'), (\'Rated 3.0\', \'RATED\\n  The ambience is really nice, staff is
```

```
type(data.reviews_list[0])
```

```
str
```

```
#https://www.mattlayman.com/blog/2018/decipher-python-ast/
ast.literal_eval(data.reviews_list.values[1])
```

```
[('Rated 4.0',
  'RATED\n  Had been here for dinner with family. Turned out to be a good choose suitable for a
 ('Rated 3.0',
  'RATED\n  The ambience is really nice, staff is courteous. The price is pretty high for the q
 ('Rated 3.0',
  'RATED\n  I felt good is little expensive for the quantity they serve and In terms of taste i
 ('Rated 4.0',
  'RATED\n  I was looking for a quite place to spend some time with family and as well wanted t
 ('Rated 4.0',
  "RATED\n  Nice place to dine and has a good ambiance... Food is good and the serving time is
 ('Rated 5.0',
  'RATED\n  This place just cool ? with good ambience and slow music and having delicious food
 ('Rated 4.0',
  "RATED\n  Quiet a good family type of place.. too calm and usually we don't find crowd here.
 ('Rated 2.0',
  "RATED\n  I had a very bad experience here.\nI don't know about a la carte, but the buffet wa
 ('Rated 4.0',
  "RATED\n  Food: 8/10\nAmbience:8/10\nStaff:8/10\nOne of the good places to try north Indian f
 ('Rated 3.0',
  'RATED\n  A decent place for a family lunch or dinner.. well arranged in a simple manner. Foo
 ('Rated 4.0',
  "RATED\n  Great place to have a heavy lunch. Good service.\nThe chicken biryani was undoubted
 ('Rated 4.0',
  'RATED\n  Its the one restaurant near katriguppe that i found was really good. Good variety o
 ('Rated 2.0',
  "RATED\n  Spice elephant soup SPL: almost manchow flavour soup.. Just above medium spicy\n\n
 ('Rated 4.0',
  'RATED\n  Zomato gold partner at this price. It was insane. They have really nice food. smal
```

```
%time data.reviews_list=data.reviews_list.apply(lambda x: ast.literal_eval(x))
```

```
CPU times: user 15.6 s, sys: 317 ms, total: 15.9 s
Wall time: 15.9 s
```

```
data.reviews_list[0][0][0].split()[0]
```

```
'Rated'
```

```
def extract_features_from_review_list(x):
    '''
    extract the rate value out of a string inside tuple
    '''
    # ensure that x is not Null and there is more than one rate
    if not x or len(x) <= 1:
        return None
    rate = [float(i[0].replace('Rated','').strip())  for i in x if type(i[0])== str]
    return round((sum(rate)/len(rate)),1)
```

```
%time data['rate_new']=data.reviews_list.apply(lambda x: extract_features_from_review_list(x))
```

```
CPU times: user 742 ms, sys: 2.93 ms, total: 745 ms
Wall time: 748 ms
```

```
data.loc[:,['rate','rate_new']].sample(10)
```

	rate	rate_new
47676	4.5	4.2
21123	3.8	3.6
50416	3.4	3.1
50890	NaN	NaN
42670	3.8	3.7
49470	2.7	2.7
2016	NaN	NaN
2303	3.6	4.0
12870	4.4	4.3
29383	3.8	3.8

```
# apply the changes
nan_index = data.query('rate != rate & rate_new == rate_new').index
for i in nan_index:
    data.loc[i,'rate'] = data.loc[i,'rate_new']
```

```
data.rate.isna().sum()
```

```
4861
```

We have saved more than 1000 points.

```
# drop null values
data.dropna(subset=['rate', 'approx_cost(for two people)'],inplace=True)
```

```
data.shape
```

```
(36840, 14)
```

```
data.drop('rate_new',axis=1,inplace=True)
```

```
data.isna().sum()
```

```
name                                0
online_order                        0
book_table                          0
rate                                0
votes                               0
```

```
location                         0
rest_type                      121
dish_liked                   15277
cuisines                         8
approx_cost(for two people)      0
reviews_list                     0
menu_item                        0
listed_in(type)                  0
dtype: int64
```

```
data[data.cuisines.isna()]
```

	name	online_order	book_table	rate	votes	location	rest_type	dish_liked
440	Lassi Spot	Yes	No	3.3	4	Kumaraswamy Layout	Beverage Shop	NaN
6887	Noodle Oodle	Yes	No	3.6	9	Whitefield	Delivery	NaN
22236	Lassi Spot	Yes	No	3.3	4	Kumaraswamy Layout	Beverage Shop	NaN
24725	Swagatham Rayalaseema Ruchulu	Yes	No	3.3	24	Kalyan Nagar	Casual Dining	NaN

Simply removing these 8 rows wont impact our data much

```
# remove cuisines missing values
data=data[data.cuisines.isna()==False]
```

```
data.isna().sum()
```

```
name                             0
online_order                     0
book_table                       0
rate                             0
votes                            0
location                         0
rest_type                      121
dish_liked                   15269
cuisines                         0
approx_cost(for two people)      0
reviews_list                     0
menu_item                        0
listed_in(type)                  0
dtype: int64
```

```
data.rename(columns={'listed_in(type)':'type', 'approx_cost(for two people)':'cost'},inplace=True)
```

```
data.rename(columns={'listed_in(type)':'type','approx_cost(for two people)':'cost'},inplace=True)
```

```
data.head()
```

	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisin
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	Nor India Mughl Chine
								Momos,	

```
data.rest_type.value_counts()
```

```
Quick Bites                    12006
Casual Dining                   8720
Cafe                            2982
Dessert Parlor                  1665
Delivery                        1486
                               ...
Dessert Parlor, Kiosk              2
Cafe, Food Court                   2
Dessert Parlor, Food Court         1
Quick Bites, Kiosk                 1
Bakery, Beverage Shop              1
Name: rest_type, Length: 88, dtype: int64
```

Filling missing values of rest_type with the most occuring value

```
data.rest_type.fillna(value='Quick Bites',inplace=True)
```

```
data.isna().sum()
```

```
name                0
online_order        0
book_table          0
rate                0
votes               0
location            0
rest_type           0
dish_liked      15269
cuisines            0
cost                0
reviews_list        0
menu_item           0
type                0
dtype: int64
```

```
#converting to lowercase
data.dish_liked=data.dish_liked.apply(lambda x:x.lower().strip() if isinstance(x,str) else x)
```

```
data.dish_liked.head(10)
```

```
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
5    onion rings, pasta, kadhai paneer, salads, sal...
6                                                  NaN
7    farmhouse pizza, chocolate banana, virgin moji...
8    pizza, mocktails, coffee, nachos, salad, pasta...
9    waffles, pasta, coleslaw sandwich, choco waffl...
Name: dish_liked, dtype: object
```

```
sum(data.menu_item=='[]')
```

```
26078
```

```
data[data.dish_liked.isna()]
```

	name	online_order	book_table	rate	votes	location	rest_type	dish_liked
6	Rosewood International Hotel - Bar & Restaurant	No	No	3.6	8	Mysore Road	Casual Dining	NaN
19	360 Atoms Restaurant And Cafe	Yes	No	3.1	13	Banashankari	Cafe	NaN
22	Cafe Coffee Day	No	No	3.6	28	Banashankari	Cafe	NaN
24	Hide Out Cafe	No	No	3.7	31	Banashankari	Cafe	NaN
25	CAFE NOVA	No	No	3.2	11	Banashankari	Cafe	NaN

```
data.reviews_list[0]
```

```
[('Rated 4.0',
  'RATED\n  A beautiful place to dine in.The interiors take you back to the Mughal era. The lig
 ('Rated 4.0',
  'RATED\n  I was here for dinner with my family on a weekday. The restaurant was completely em
 ('Rated 2.0',
  'RATED\n  Its a restaurant near to Banashankari BDA. Me along with few of my office friends v
 ('Rated 4.0',
  'RATED\n  We went here on a weekend and one of us had the buffet while two of us took Ala Car
 ('Rated 5.0',
  'RATED\n  The best thing about the place is itÃ\x83\x83Ã\x82\x83Ã\x83\x82Ã\x82\x82Ã\x83\x83Ã\
 ('Rated 5.0',
  'RATED\n  Great food and pleasant ambience. Expensive but Coll place to chill and relax.....
 ('Rated 4.0',
  'RATED\n  Good ambience with tasty food.\nCheese chilli paratha with Bhutta palak methi curry
 ('Rated 4.0',
  'RATED\n  You canÃ\x83\x83Ã\x82\x83Ã\x83\x82Ã\x82\x82Ã\x83\x83Ã\x82\x82Ã\x83\x82Ã\x82\x92t go
```

```
 ('Rated 5.0',
  'RATED\n  Overdelighted by the service and food provided at this place. A royal and ethnic at
 ('Rated 4.0',
  'RATED\n  The place is nice and comfortable. Food wise all jalea outlets maintain a good stan
 ('Rated 4.0',
  'RATED\n  The place is nice and comfortable. Food wise all jalea outlets maintain a good stan
 ('Rated 4.0',
  'RATED\n  The place is nice and comfortable. Food wise all jalea outlets maintain a good stan
```

```
#as we can see dishes liked or disliked are mentioned in reviews so if we can extract these dishes w
#we will start by getting a list of all the dishes available from our dataset
```

```
%%time
dish_list=[]
for i in list(data.index):
  #print(type(data.dish_liked[i]))
  if data.dish_liked[i]!='NaN' and isinstance(data.dish_liked[i],str):
      k=data['dish_liked'][i].split(',')
      dish_list.extend(k)
print(dish_list)
```

```
['pasta', ' lunch buffet', ' masala papad', ' paneer lajawab', ' tomato shorba', ' dum biryani
CPU times: user 839 ms, sys: 9.89 ms, total: 849 ms
Wall time: 857 ms
```

```
len(dish_list)
```

```
118363
```

```
dish_list=set(dish_list) #getting unique dishes
```

```
len(dish_list)
```

```
3507
```

```
p=data.reviews_list[0]
' '.join([i[1].replace('RATED\n ','') for i in p]).replace('\n','').replace('\S+','').replace('?',''
```

```
'a beautiful place to dine in.the interiors take you back to the mughal era. the lightings are
just perfect.we went there on the occasion of christmas and so they had only limited items ava
ilable. but the taste and service was not compromised at all.the only complaint is that the br
eads could have been better.would surely like to come here again.  i was here for dinner with
my family on a weekday. the restaurant was completely empty. ambience is good with some good o
ld hindi music. seating arrangement are good too. we ordered masala papad, panner and baby cor
```

```
# clear the text
def clear_text(t):
    '''
    clear the input text t
    '''
    return ' '.join([i[1].replace("RATED\n  ",'') for i in t]).encode('utf8').decode('ascii',errors=
           replace('?','').replace('�','').replace('\n','').replace('.',' ').strip().lower()
```

```
data['reviews_text'] = data.reviews_list.apply(lambda x: clear_text(x))
```

```
data['reviews_text'].head(3)
```

```
0    a beautiful place to dine in the interiors tak...
1    had been here for dinner with family  turned o...
2    ambience is not that good enough and it's not ...
Name: reviews_text, dtype: object
```

```
dish_list.intersection(data['reviews_text'][100].split())
```

```
{'rice', 'thali', 'vegetarian'}
```

```
data['dish_liked_new']=data.reviews_text.apply(lambda x: ', '.join(list(dish_list.intersection(x.spl
```

```
data.dish_liked_new.isna().sum()
```

```
0
```

```
# get sample to compare
data.query('dish_liked != dish_liked')[['dish_liked','dish_liked_new']].sample(5,random_state=1)
```

	dish_liked	dish_liked_new
32901	NaN	kheer, halwa
44323	NaN	prawn, shawarma, chicken, tikka, rice
6479	NaN	
11046	NaN	rice
50112	NaN	cappuccino, coffee

So, now we can replace this missed values from the dish_n_review

```
nan_index=data.query('dish_liked !=dish_liked & dish_liked_new==dish_liked_new').index
```

```
%%time
for i in nan_index:
  data.loc[i,'dish_liked']=data.loc[i,'dish_liked_new']
```

```
CPU times: user 16.5 s, sys: 195 ms, total: 16.7 s
Wall time: 16.7 s
```

```
data.columns
```

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

```
data.drop(['dish_liked_new','reviews_text'],axis=1,inplace=True)
```

```
data.head()
```

	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisin
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	pasta, lunch buffet, masala papad, paneer laja...	Nor India Mughl Chine
	Spice						Casual	momos, lunch buffet,	Chines Nor

```
data.isna().sum()
```

```
name            0
online_order    0
book_table      0
rate            0
votes           0
location        0
rest_type       0
dish_liked      0
cuisines        0
cost            0
reviews_list    0
menu_item       0
type            0
dtype: int64
```

```
data.shape
```

```
(36832, 13)
```

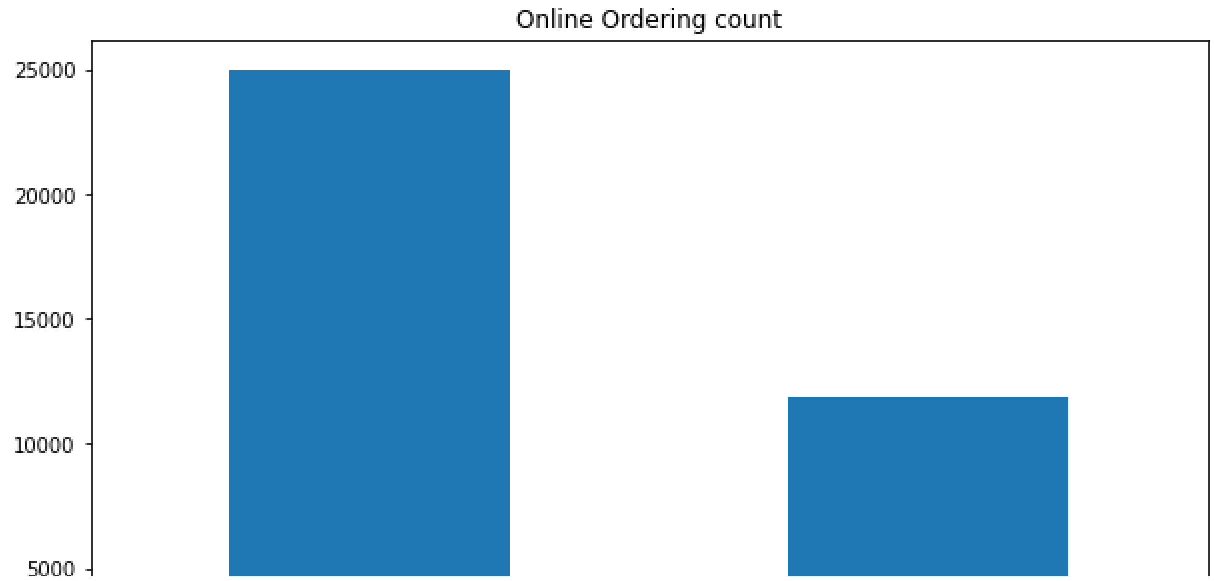
EDA

Data Visualizations

How many Restaurant accepting online orders?

```
oo=data.online_order.value_counts()
```

```
plt.figure(figsize=(10,6))
oo.plot(kind='bar')
plt.title('Online Ordering count')
plt.show()
oo
```

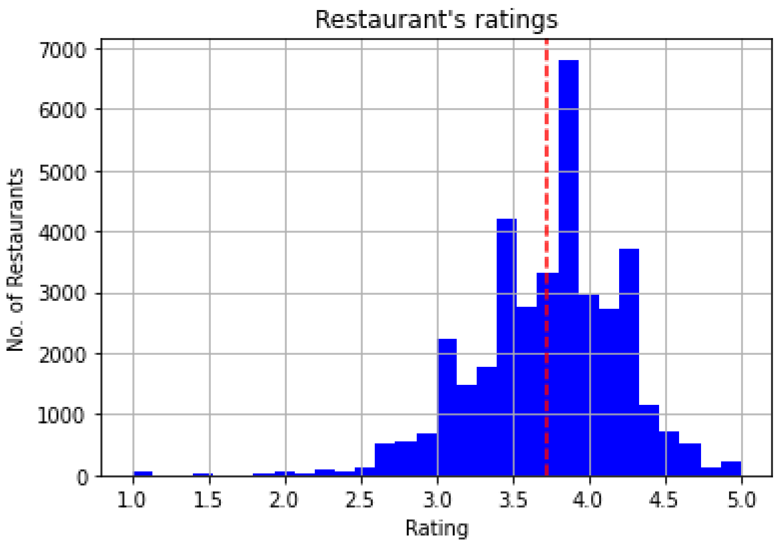


What is distrubution of 'Rate column'?

```
data.rate=data.rate.astype('float')
```

```
No      11865
```

```
data.rate.hist(color='blue',bins=30)
plt.axvline(x=data.rate.mean(),color='red',ls='--')
plt.title("Restaurant's ratings")
plt.xlabel('Rating')
plt.ylabel('No. of Restaurants')
plt.show()
print(data.rate.mean())
```



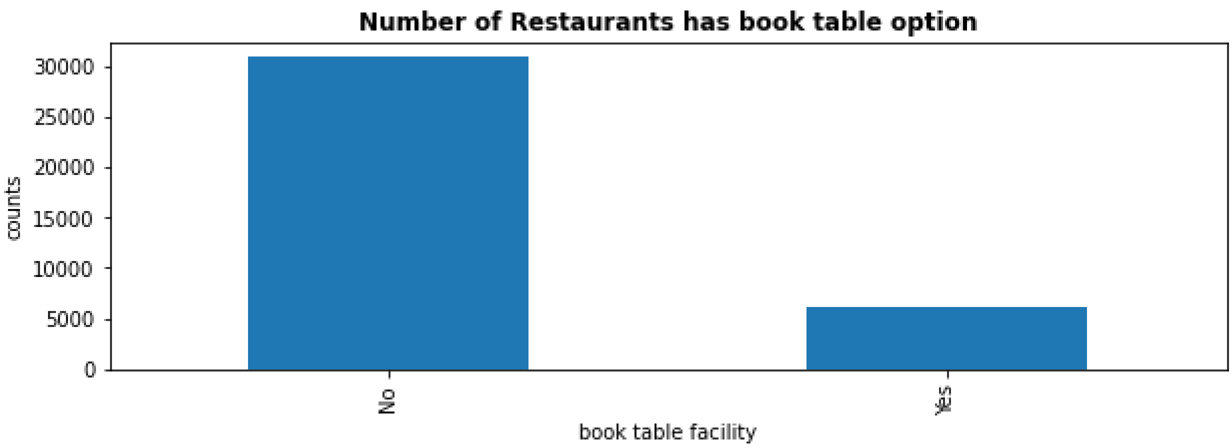

```
3.7208921589921835
```

```
data.book_table.value_counts
```

```
<bound method IndexOpsMixin.value_counts of 0        Yes
1         No
2         No
3         No
4         No
        ...
51709     No
51711     No
51712     No
51715    Yes
```

```
51716     No
Name: book_table, Length: 36832, dtype: object>
```

```
plt.figure(figsize=(10,3))
ax =data.book_table.value_counts().plot(kind='bar')
plt.title('Number of Restaurants has book table option', weight='bold')
plt.xlabel('book table facility')
plt.ylabel('counts')
plt.show()
```



- Most restaurants do not have book a table option

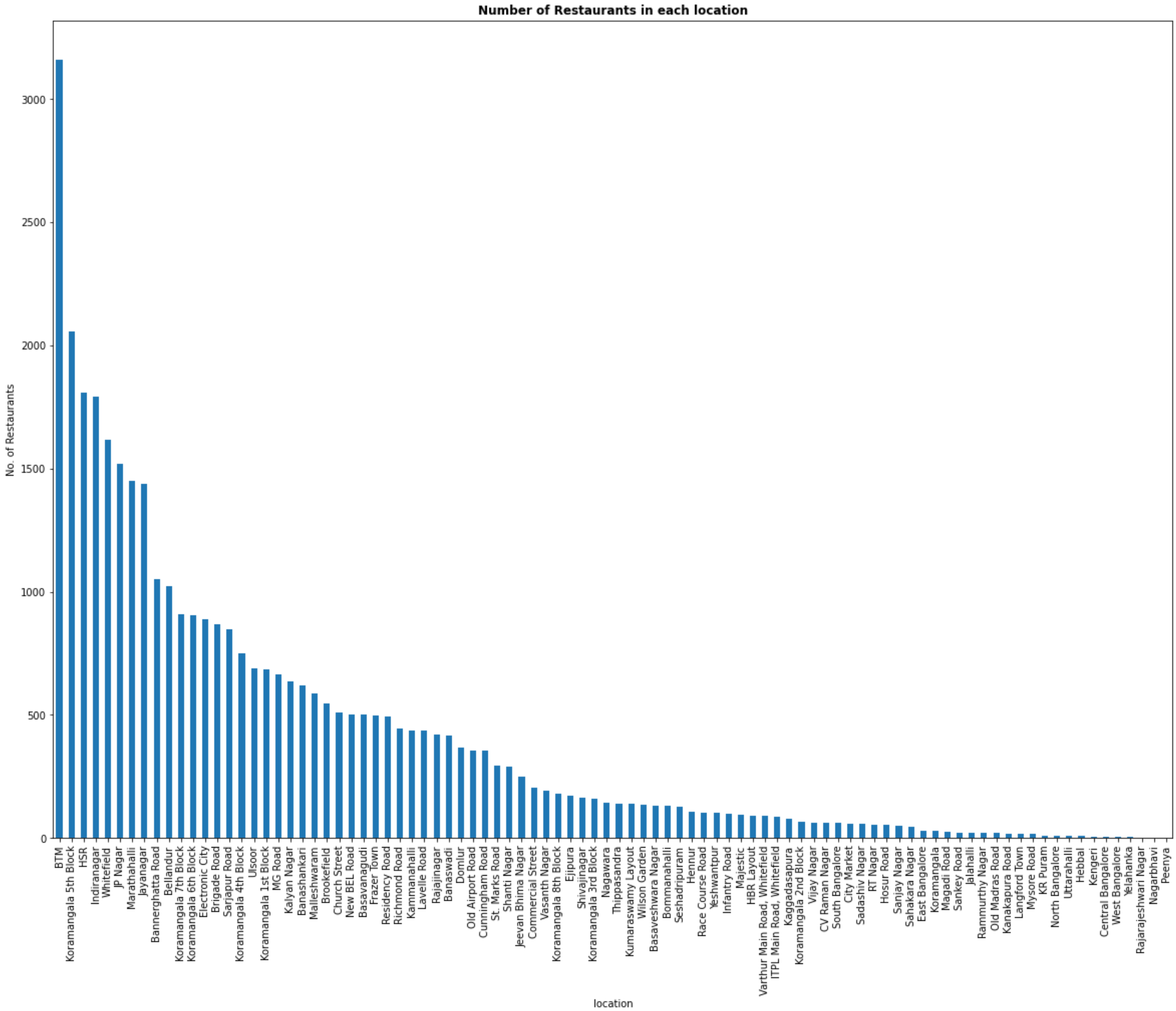
In Bangalore city,in which area has maximum number of restaurants

```
data.location.value_counts()
```

```
BTM                      3161
Koramangala 5th Block    2056
HSR                      1807
Indiranagar              1793
Whitefield               1616
                         ...
West Bangalore              4
Yelahanka                   4
Rajarajeshwari Nagar        2
Nagarbhavi                  1
Peenya                      1
Name: location, Length: 92, dtype: int64
```

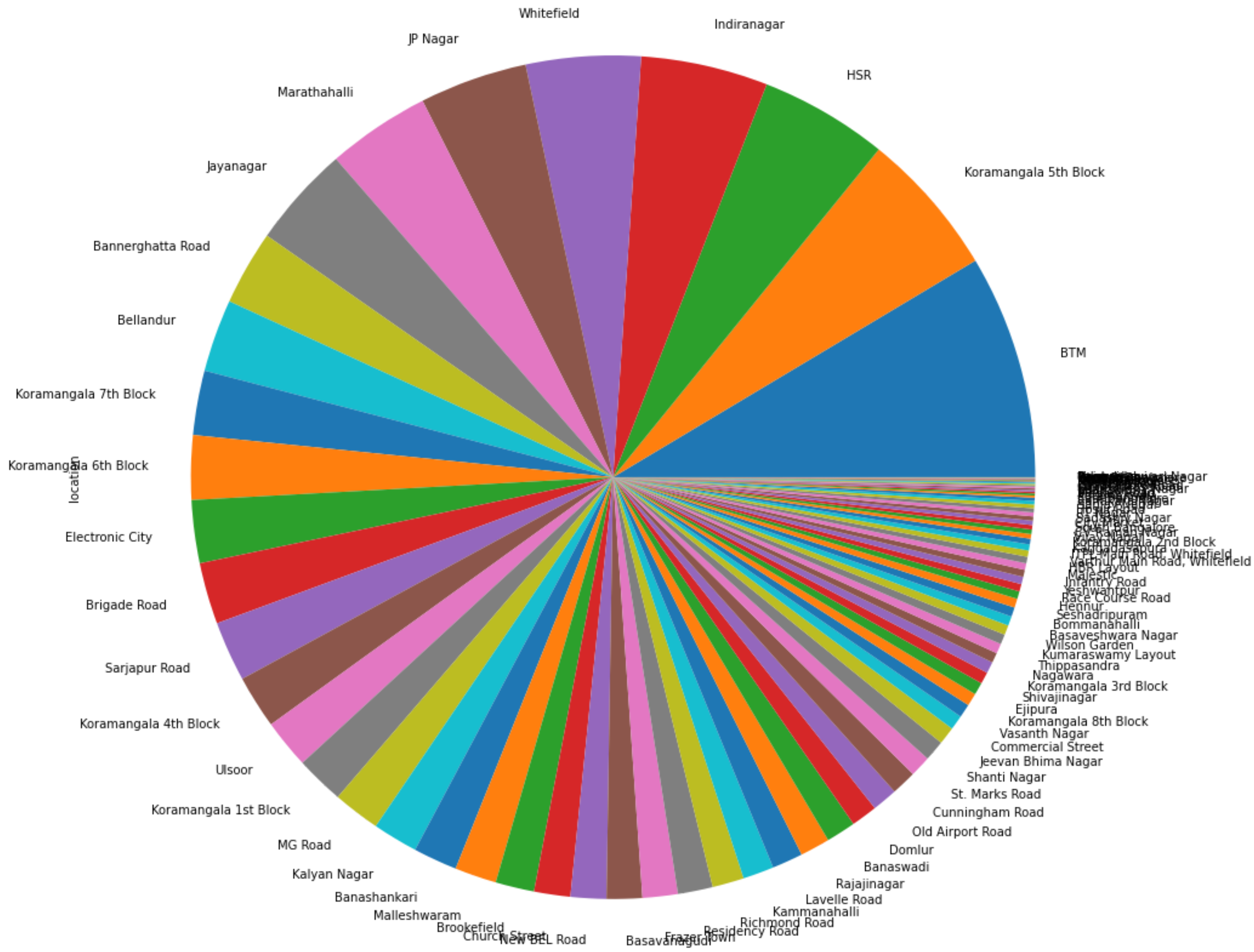
As we can see there are total 92 different locations in Bangalore.

```
#plotting
plt.figure(figsize=(20,15))
ax =data.location.value_counts().plot(kind='bar')
plt.title('Number of Restaurants in each location', weight='bold')
plt.xlabel('location')
plt.ylabel('No. of Restaurants')
plt.show()
```



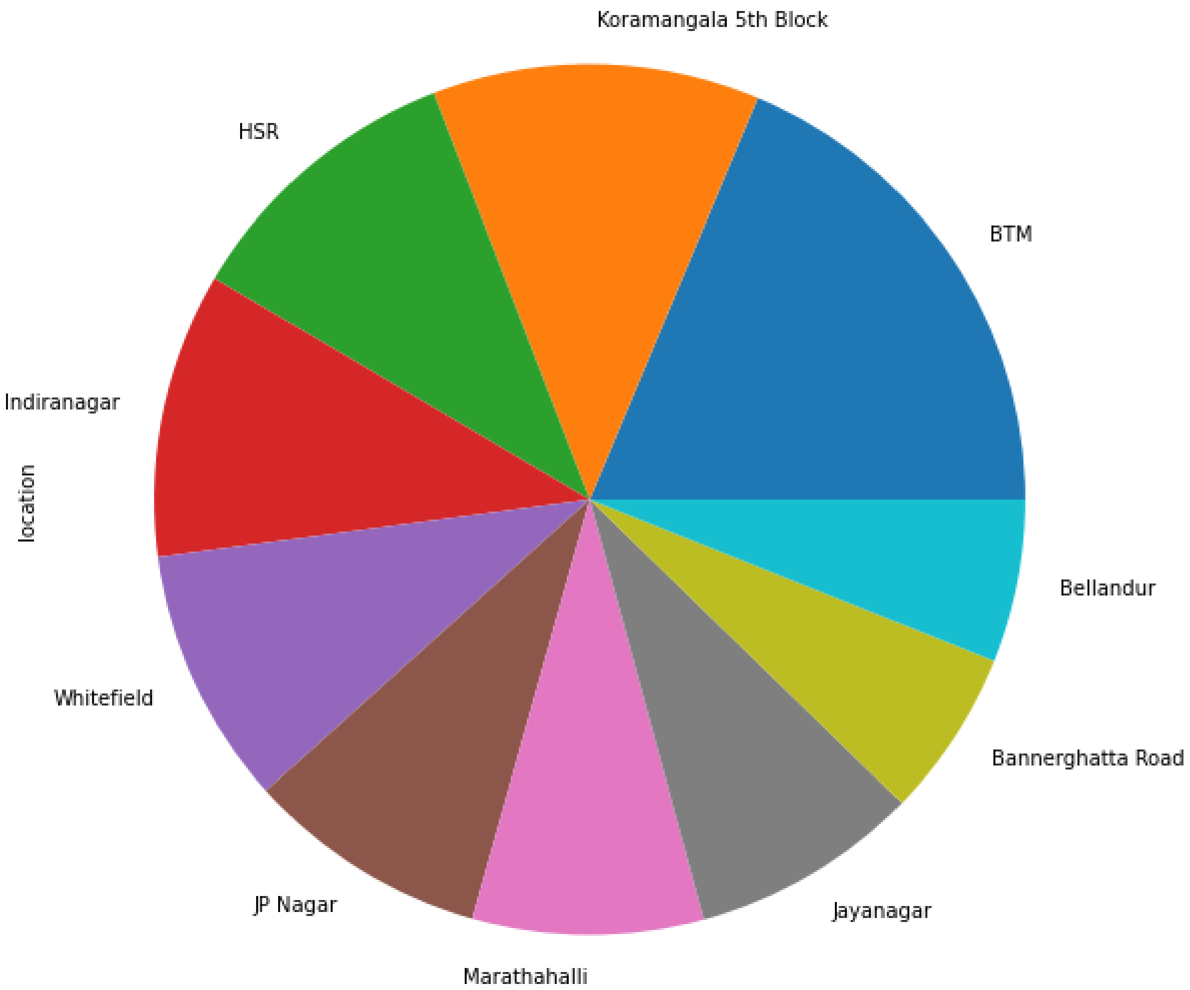

```
#plotting pie chart
plt.figure(figsize=(20,16))
ax=data.location.value_counts().plot(kind='pie')
```

```
plt.show()
```



Its very complicated to understand so we will limit ourself to TOP 10 locations

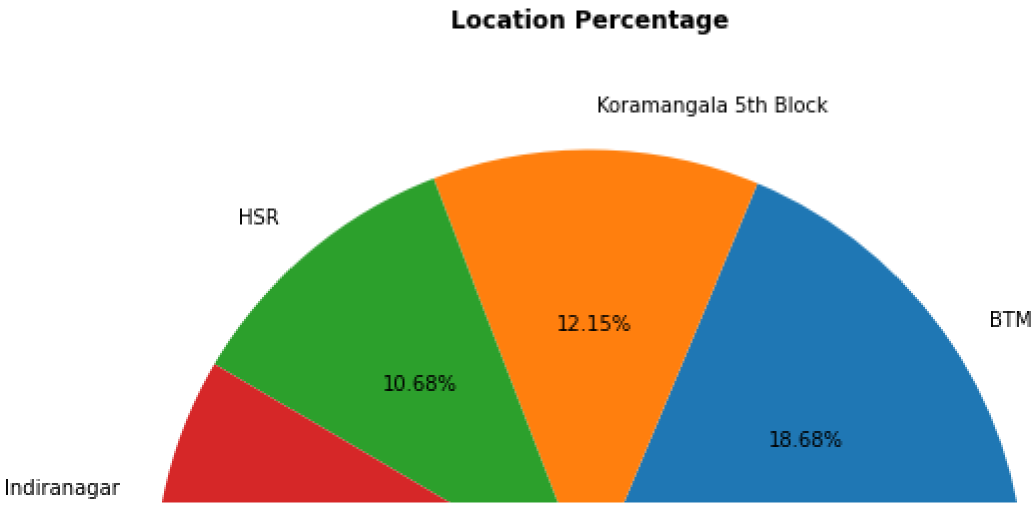
```
plt.figure(figsize=(15,10))
ax=data.location.value_counts()[:10].plot(kind='pie')
plt.show()
```



- We can see that most of the restaurants are located at the BTM location

Percentage

```
#https://stackoverflow.com/questions/21572870/matplotlib-percent-label-position-in-pie-chart
plt.figure(figsize=(15,10))
ax=data.location.value_counts()[:10].plot(kind='pie',autopct='%1.2f%%')
plt.title('Location Percentage', weight='bold')
plt.show()
```



- maximum restaurant are at BTM follows by HSR,Koramangla, JP Nagar, .. so on.

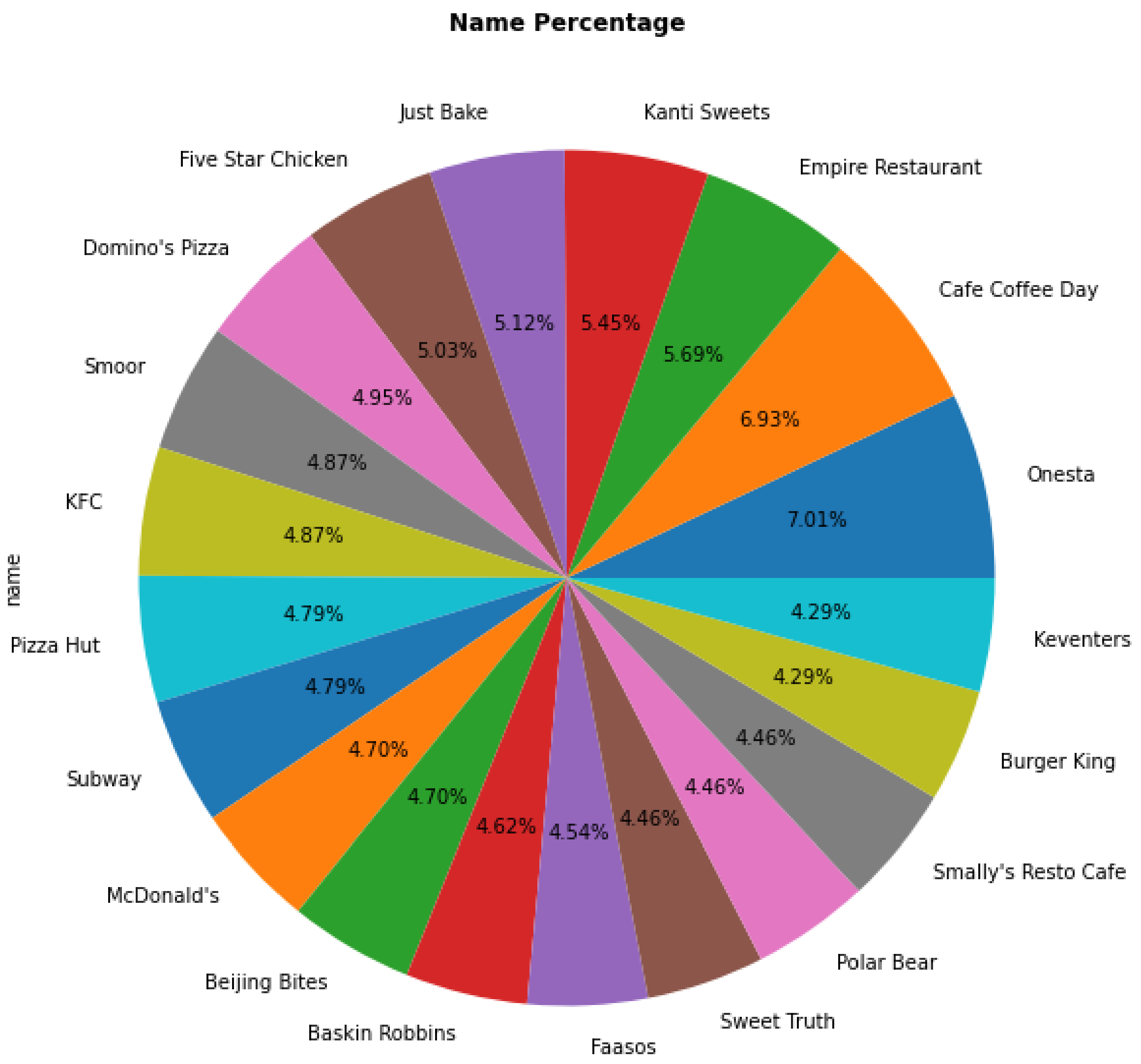
Top 20 restaurants by name

```
top_20=data.name.value_counts()[:20]
```


```
top_20=data.name.value_counts()[:20]
```


```
plt.figure(figsize=(10,6))
ax=top_20.plot(kind='bar',color='blue')
plt.title('Top 20 restaurants by name')
plt.xlabel("Restaurant's Name")
plt.ylabel('Count')
plt.show()
```

```
#https://stackoverflow.com/questions/21572870/matplotlib-percent-label-position-in-pie-chart
plt.figure(figsize=(15,10))
ax=top_20.plot(kind='pie',autopct='%1.2f%%')
plt.title('Name Percentage', weight='bold')
plt.show()
```



- we can say that 'Onesta' day has highest count among all

What type of restaurants are there in Bangalore? also percentage and counts

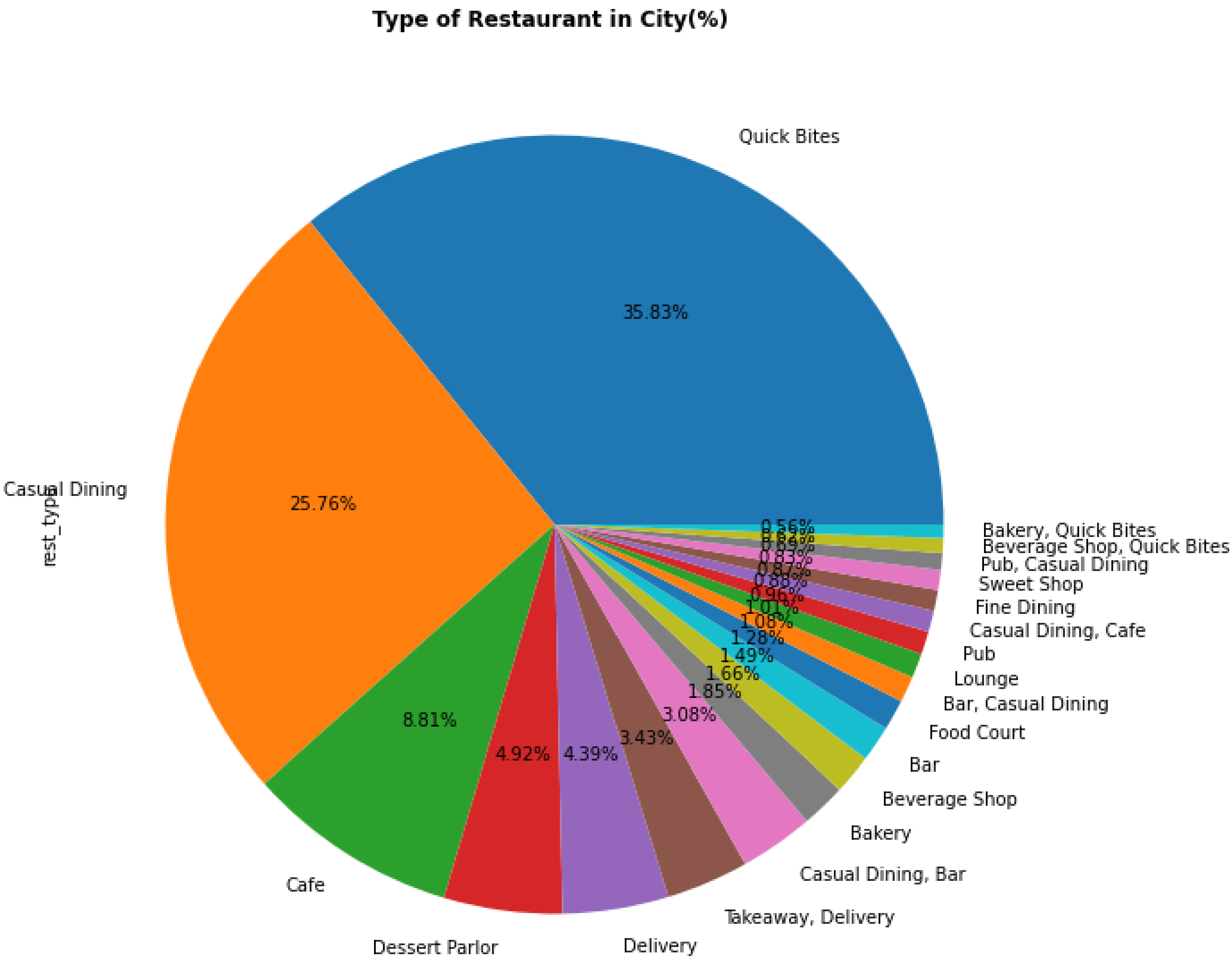
```
data.columns
```

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

```
top_types20=data.rest_type.value_counts()[:20]
```

```
plt.figure(figsize=(15,10))
```

```
ax=top_types20.plot(kind='pie',autopct='%1.2f%%')
plt.title('Type of Restaurant in City(%) ', weight='bold')
plt.show()
```



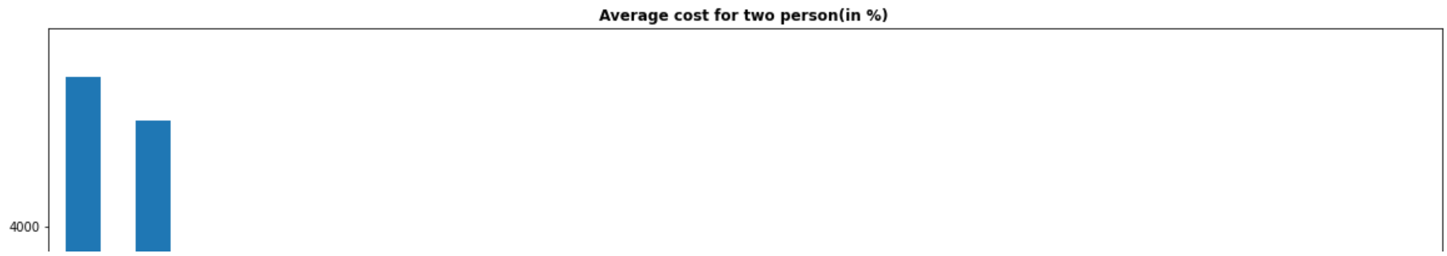

```
plt.figure(figsize=(10,3))
ax =data.rest_type.value_counts()[:20].plot(kind='bar')
plt.title('Number of Restaurants in given location', weight='bold')
plt.xlabel('Area')
plt.ylabel('counts')
```

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

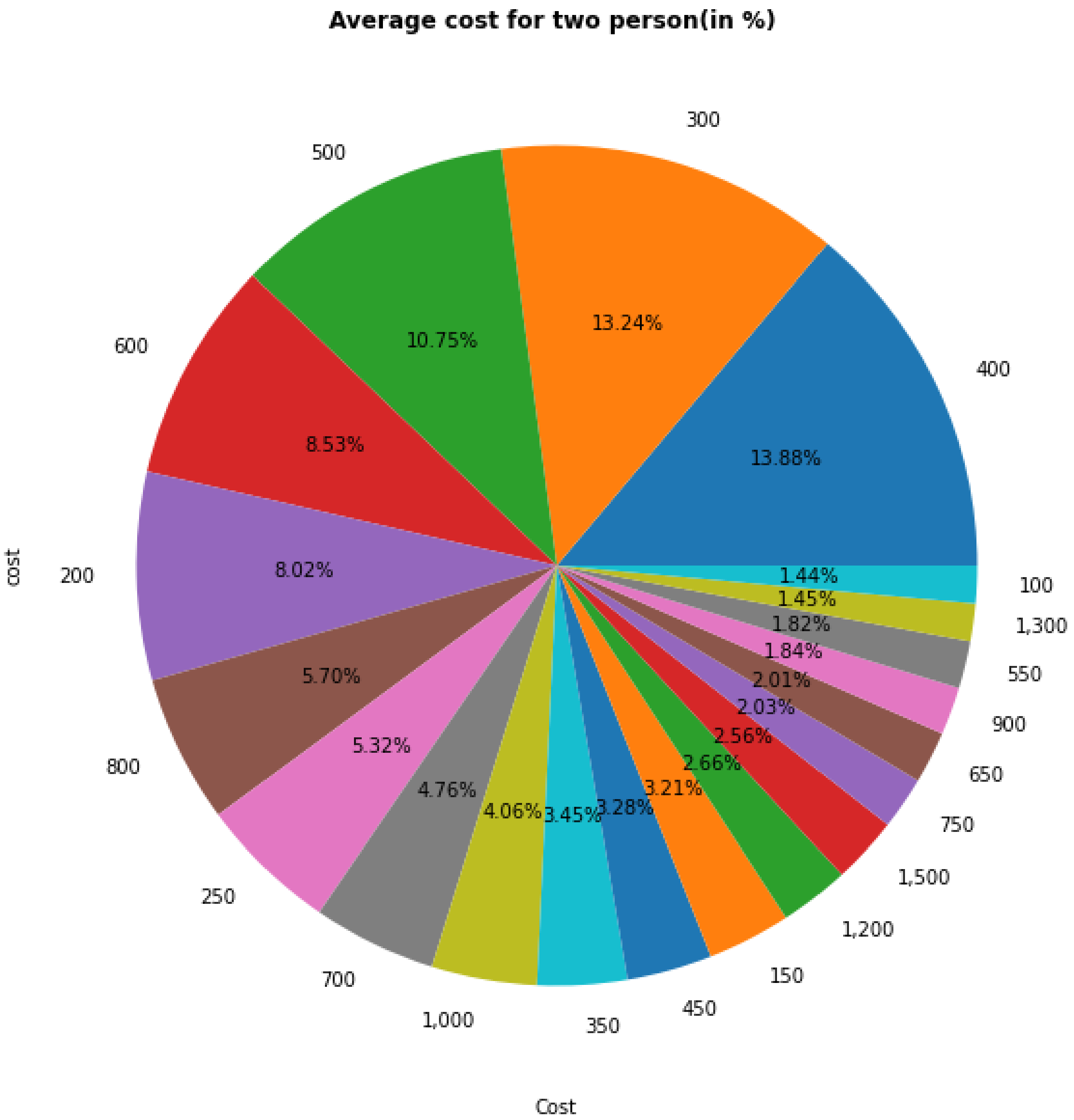
- Mostly 'Quick Byte' restaurants are present

What is the Average cost in restaurants?

```
#plotting
plt.figure(figsize=(20,15))
ax =data.cost.value_counts()[:20].plot(kind='bar')
plt.title('Average cost for two person(in %) ', weight='bold')
plt.xlabel('Cost')
plt.show()
```




```
plt.figure(figsize=(15,10))
ax=data.cost.value_counts()[:20].plot(kind='pie',autopct='%1.2f%%')
plt.title('Average cost for two person(in %) ', weight='bold')
plt.xlabel('Cost')
plt.show()
```



- Average cost for 2 is around 300-400 for 27% restaurants and below 500 for approx 52% restaurants

Which dish are most famous/favourite dish in restaurants?

```
data.dish_liked.nunique()
```

    7518

```
dishes=[j.strip() for i in data.dish_liked for j in i.split(',')]
```

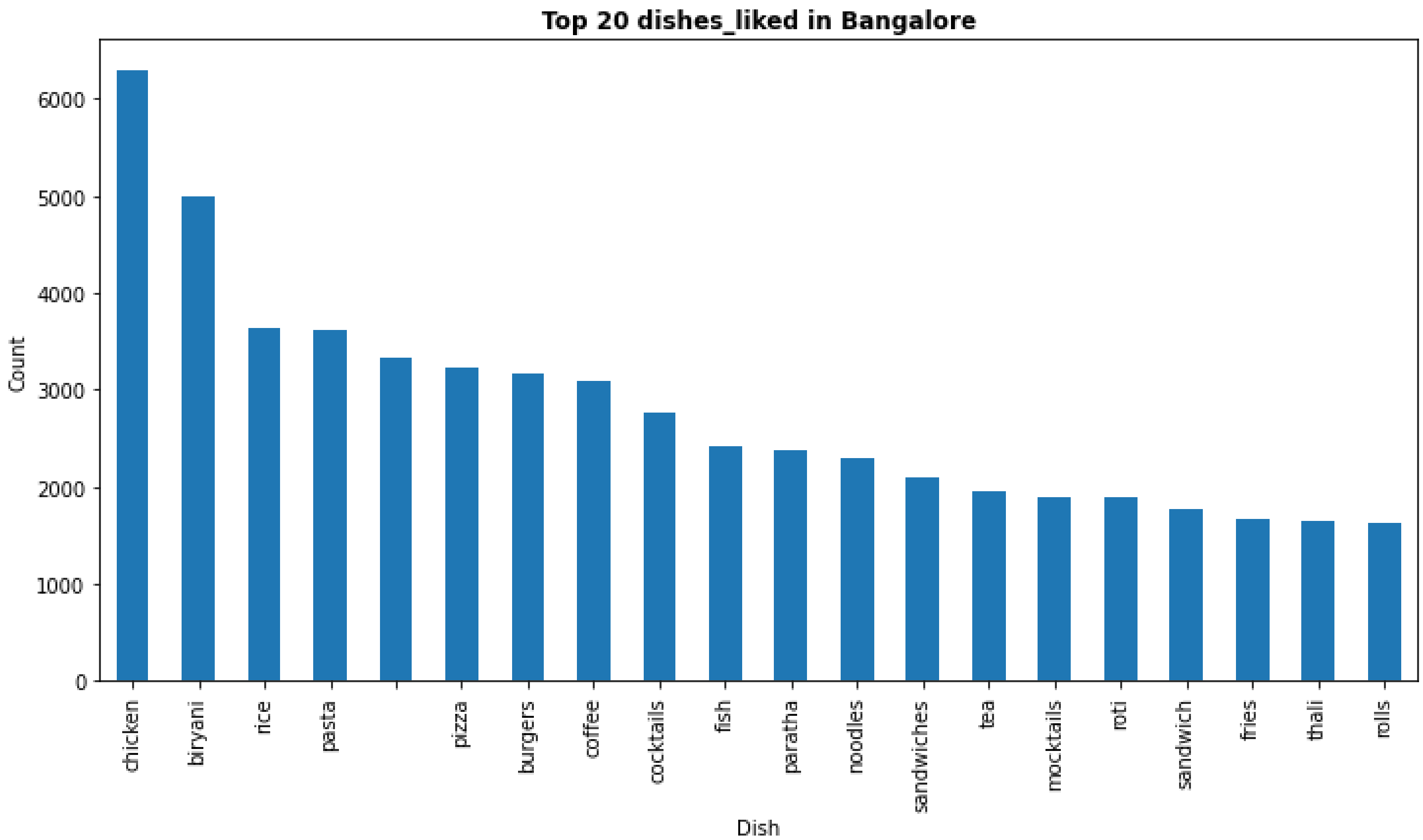
```
from collections import Counter
c=Counter(dishes)
```

```
c.most_common(10)
```

```
[('chicken', 6295),
 ('biryani', 4985),
 ('rice', 3637),
 ('pasta', 3619),
 ('', 3329),
 ('pizza', 3236),
 ('burgers', 3173),
 ('coffee', 3087),
 ('cocktails', 2762),
 ('fish', 2428)]
```

```
plt.figure(figsize=(12,6))
pd.Series(dishes).value_counts()[:20].plot(kind='bar')
plt.title('Top 20 dishes_liked in Bangalore',weight='bold')
plt.xlabel('Dish')
plt.ylabel('Count')
```

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



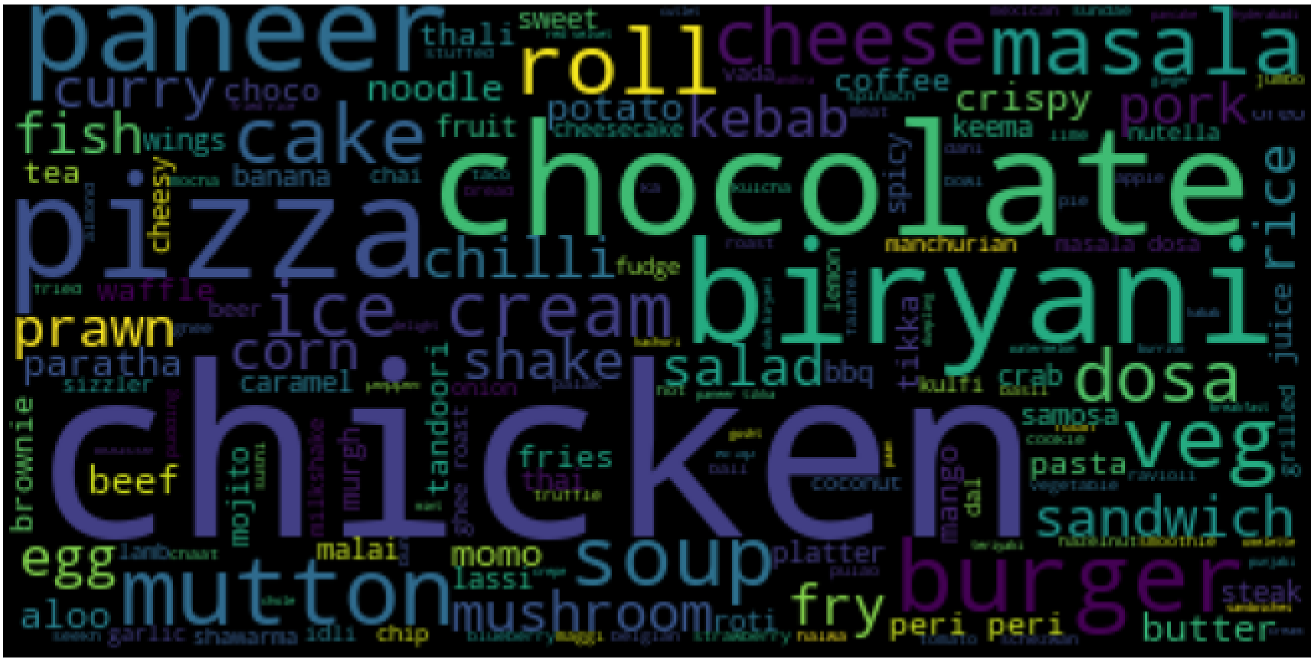
- Most people love Chicken and biryani

WordCloud

```
plt.figure(figsize=(12,6))
set_dishes=set(dishes)
```

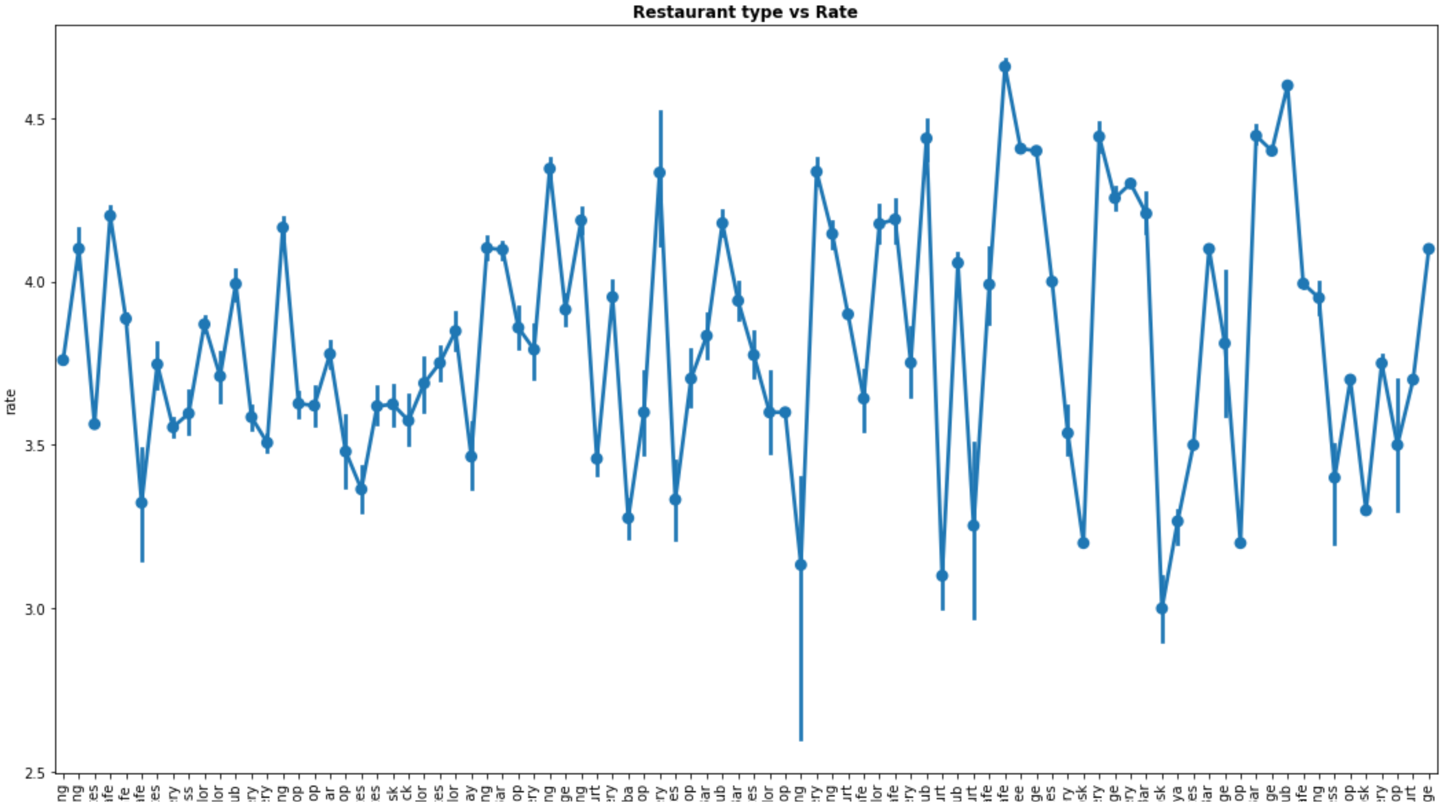
```
set_dishes=set(dishes)
dish_word_cloud = ', '.join(set_dishes)
wc=WordCloud(background_color='black',stopwords=STOPWORDS,max_words=len(dish_word_cloud))
wc.generate(dish_word_cloud)
plt.imshow(wc,interpolation='bilinear')
plt.title('Word Cloud for favourite dishes',weight='bold')
plt.axis("off")
plt.imshow(wc)
plt.show()
```

Word Cloud for favourite dishes

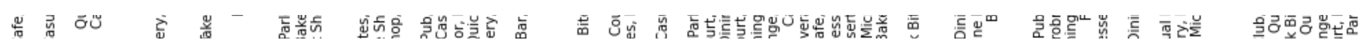


Lets see 'Rate' vs 'Restaurant type' graph.

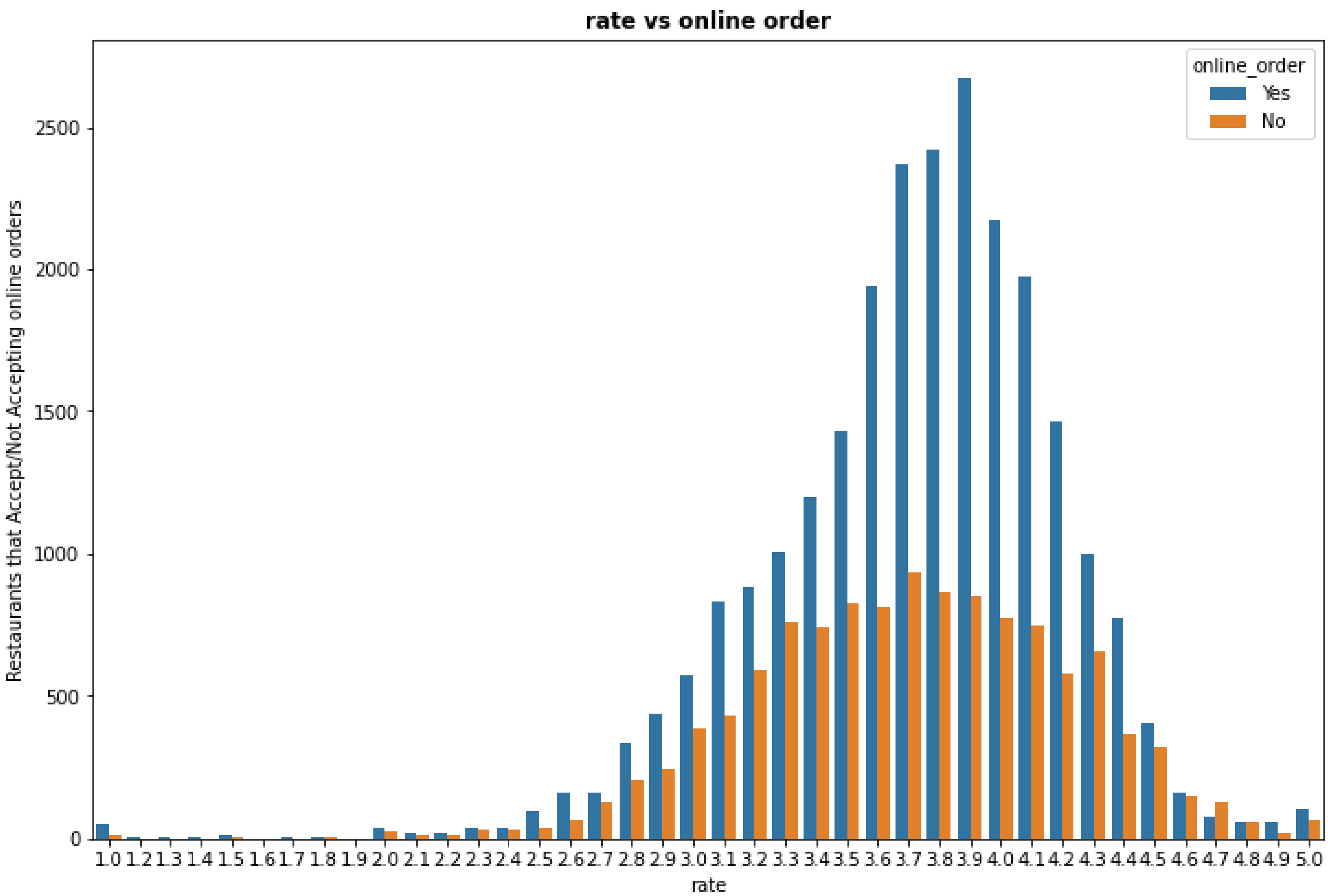
```
f,ax=plt.subplots(figsize=(18,10))
g=sns.pointplot(y='rate',x='rest_type',data=data)
g.set_xticklabels(g.get_xticklabels(),rotation=90)
plt.title('Restaurant type vs Rate', weight = 'bold')
plt.show()
```



Lets plot 'Rate' vs 'Online order'



```
f,ax=plt.subplots(figsize=(12,8))
sns.countplot(x='rate',data=data,hue='online_order')
plt.ylabel("Restaurants that Accept/Not Accepting online orders")
plt.title("rate vs online order",weight = 'bold')
plt.show()
```



Top 10 Cuisines

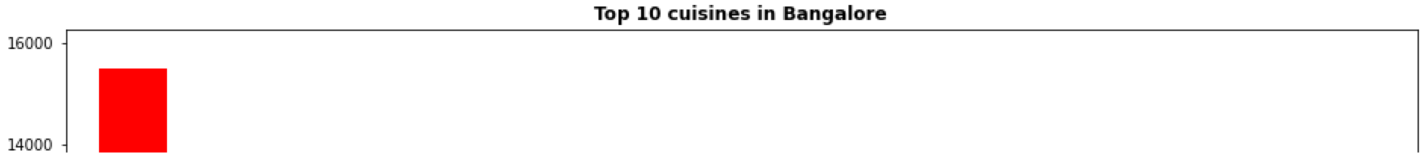
```
cuisines=[j.lower().strip() for i in data.cuisines for j in i.split(',')]
```

```
c1=Counter(cuisines)
c1.most_common(10)
```

```
[('north indian', 15484),
 ('chinese', 11477),
 ('fast food', 5572),
 ('south indian', 5524),
 ('continental', 4851),
 ('biryani', 4519),
 ('cafe', 4292),
 ('desserts', 4101),
 ('beverages', 3524),
 ('italian', 2965)]
```

```
plt.figure(figsize=(16,10))
pd.Series(cuisines).value_counts()[:10].plot(kind='bar',color='r')
plt.title('Top 10 cuisines in Bangalore',weight='bold')
plt.xlabel('cuisines type')
plt.ylabel('No of restaurants')
```

```
Text(0, 0.5, 'No of restaurants')
```



- North Indian food is at top, followed by chinese and so on.


```
plt.figure( figsize=(15,10) )
cuisines_set=set(cuisines)
cuisines_wc=', '.join(cuisines_set)
wc=WordCloud(stopwords=STOPWORDS,max_words=len(cuisines_wc))
wc.generate(cuisines_wc)
plt.axis("off")
plt.imshow(wc)
plt.show()
```

Model

- Till now we were understanding, visualising data. Now let move to build proper Machine Learning model.

```
data['online_order']=pd.get_dummies(data.online_order,drop_first=True)
data['book_table']=pd.get_dummies(data.book_table,drop_first=True)
```

```
data.columns
```

```
Index(['name', 'online_order', 'book_table', 'rate', 'votes', 'location',
```

```
       'rest_type', 'dish_liked', 'cuisines', 'cost', 'reviews_list',
       'menu_item', 'type'],
      dtype='object')
```

```
data.drop(columns=['dish_liked','reviews_list','menu_item','type'], inplace  =True)
```

```
data.head()
```

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost
0	Jalsa	1	1	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800
1	Spice Elephant	1	0	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800

```
data.rest_type=data.rest_type.str.replace(',','')
data.rest_type=data.rest_type.apply(lambda x: ' '.join(sorted(x.split())))
```

```
data.rest_type.head()
```

```
0         Casual Dining
1         Casual Dining
2    Cafe Casual Dining
3           Bites Quick
4         Casual Dining
Name: rest_type, dtype: object
```

```
data['cuisines']=data.cuisines.str.replace(',','')
data['cuisines']=data.cuisines.astype(str).apply(lambda x:' '.join(sorted(x.split())))
```

```
data.cost.value_counts()
```

```
400      4730
300      4514
500      3664
600      2907
200      2732
         ...
6,000       2
80          1
3,700       1
5,000       1
70          1
Name: cost, Length: 63, dtype: int64
```

```
data['cost']=data.cost.str.replace(',','')
data.cost=data.cost.astype('int')
```

```
x = data.drop(['rate','name'],axis = 1)
y = data['rate']
```

```
x.shape,y.shape
```

```
x.shape,y.shape
```

```
((36832, 7), (36832,))
```

Splitting the data for Model Building

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 0)
```

```
from joblib import dump,load

dump(X_train, 'more_feature_X_train')
dump(X_test, 'more_feature_X_test')
dump(y_train, 'more_feature_y_train')
dump(y_test, 'more_feature_y_test')
```

```
['more_feature_y_test']
```

```
X_train = load('more_feature_X_train')
X_test = load('more_feature_X_test')
y_train = load('more_feature_y_train')
y_test = load('more_feature_y_test')
```

```
X_train.head(2)
```

	online_order	book_table	votes	location	rest_type	cuisines	cost
40463	1	0	1731	Malleshwaram	Pub	Chinese Continental Indian North	1400
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Casual	Bengali Chinese Indian	[illegible]

```
encoder=OneHotEncoder(handle_unknown='ignore')
```

```
#Avoid data leakage
def onehot_fit_transform(df,col):
  out=df[col].values.reshape(-1,1)
  return encoder.fit_transform(out).toarray()

def onehot__transform(df,col):
  out=df[col].values.reshape(-1,1)
  return encoder.transform(out).toarray()
```

```
# one hot encoding apply to 'rest_type' features on train/test dataset
train_loc_feature=onehot_fit_transform(X_train,'location')
test_loc_feature=onehot__transform(X_test,'location')

train_type_feature=onehot_fit_transform(X_train,'rest_type')
test_type_feature=onehot__transform(X_test,'rest_type')
```

```
train_cuisines_feature=onehot_fit_transform(X_train,'cuisines')
```

```
train_cuisines_feature=onehot_fit_transform(X_train,'cuisines')
test_cuisines_feature=onehot__transform(X_test,'cuisines')


print(train_loc_feature.shape)
print(test_loc_feature.shape)


    (25782, 92)
    (11050, 92)


print(train_cuisines_feature.shape)
print(test_cuisines_feature.shape)


    (25782, 1678)
    (11050, 1678)


print(train_type_feature.shape)
print(test_type_feature.shape)


    (25782, 65)
    (11050, 65)


from scipy.sparse import hstack
## combine all 'one-hot' encoded features as Tr.
tr=pd.DataFrame(pd.np.column_stack([train_loc_feature,train_cuisines_feature,train_type_feature]))


## CONCAT both dataframe ### ie Tr and X_train(original dataframe)
## https://stackoverflow.com/questions/45963799/pandas-concat-resulting-in-nan-rows
l1=X_train.values.tolist()
l2=tr.values.tolist()

for i in range(len(l1)):
    l1[i].extend(l2[i])

X_train=pd.DataFrame(l1,columns=X_train.columns.tolist()+tr.columns.tolist())
X_train.shape

    (25782, 1842)


## combine all 'one-hot' encoded features as Te.
te =pd.DataFrame(pd.np.column_stack([test_loc_feature, test_cuisines_feature,test_type_feature]))

## CONCAT both dataframe ### ie Te and X_test(original dataframe)
## https://stackoverflow.com/questions/45963799/pandas-concat-resulting-in-nan-rows

l3=X_test.values.tolist()
l4=te.values.tolist()
for i in range(len(l3)):
    l3[i].extend(l4[i])

X_test=pd.DataFrame(l3,columns=X_test.columns.tolist()+te.columns.tolist())
X_test.shape

    (11050, 1842)
```

```
# after onehot encoding DONE. 'location','rest_type','cuisines' are redundant features. REMOVE them.

X_train =X_train.drop(['location','rest_type','cuisines'],axis = 1)
X_test =X_test.drop(['location','rest_type','cuisines'],axis = 1)
```

```
# checking final train set shape
X_train.shape, y_train.shape
```

    ((25782, 1839), (25782,))

```
dump(X_train, 'more_feature_X_train')
dump(X_test, 'more_feature_X_test')
dump(y_train, 'more_feature_y_train')
dump(y_test, 'more_feature_y_test')
```

    ['more_feature_y_test']

```
X_train= load('more_feature_X_train')
X_test= load('more_feature_X_test')
y_train= load('more_feature_y_train')
y_test= load('more_feature_y_test')
```

```
def mse(y, y_pred):
    return np.mean((y_pred - y)**2)

mse_scorer = make_scorer(mse, greater_is_better=False)
```

```
# https://github.com/erykml/medium_articles/blob/master/Machine%20Learning/feature_importance.ipynb

def imp_df(column_names,importances):
  df=pd.DataFrame({
      'features':column_names,
      'features_importances':importances
  }).sort_values('features_importances',ascending=False).reset_index(drop=True)
  return df
# plotting a feature importance dataframe (horizontal barchart)
def var_imp_plot(imp_df,title):
  sns.barplot(x='features_importances',y='features',data=imp_df,color='royalblue',orient='h').set_ti
```

Model -1 Linear Regression

```
regression=LinearRegression()
regression.fit(X_train,y_train)
y_pred_lr=regression.predict(X_test)
```

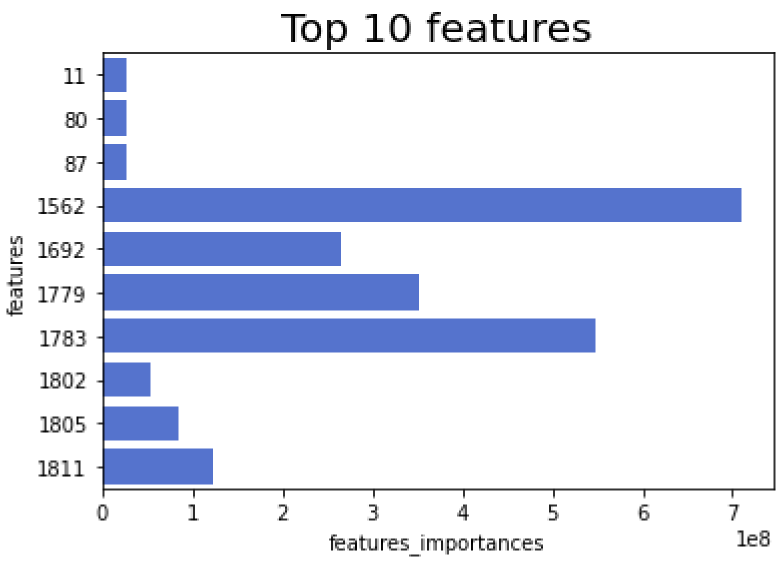
```
mse(y_test,y_pred_lr)
```

    910460897477.011

```
lr1=imp_df(X_train.columns,regression.coef_)
lr1.head(10)
```

	features	features_importances
0	1562	7.090223e+08
1	1783	5.484350e+08
2	1779	3.526236e+08
3	1692	2.647070e+08
4	1811	1.224469e+08
5	1805	8.495233e+07
6	1802	5.370294e+07
7	87	2.704729e+07
8	11	2.704729e+07
9	80	2.704729e+07

```
var_imp_plot(lr1[:10],'Top 10 features')
```



Model - 2 SGD Regressor

```
from sklearn import linear_model
sgd_regression=linear_model.SGDRegressor()
sgd_regression.fit(X_train,y_train)
y_pred_sgd=sgd_regression.predict(X_test)
```

```
mse(y_test,y_pred_sgd)
```

```
1.2725480269569956e+30
```

Model -3 Random Forest Regressor¶

```
from sklearn.ensemble import RandomForestRegressor
rf_regression=RandomForestRegressor() #using default values
rf_regression.fit(X_train,y_train)
y_pred_rf=rf_regression.predict(X_test)

mse(y_test,y_pred_rf)
```

```
0.029926186745185573
```

- Without any hyperparameter tuning RFR ie Random Forest Regressor it learning something. so let experiment on RFR.

Hyperparameter Tuning on RFR

```
%%time
params=[{'n_estimators':[250,500,750,1000]}]
regressor_params=GridSearchCV(rf_regression,params,n_jobs=-1,cv=10,verbose=1,return_train_score=True
regressor_params.fit(X_train,y_train)

#hyperparameter tuning will take 5-6 hours for 1000 estimators on high end configuration
```

```
from sklearn.ensemble import RandomForestRegressor
rf_regression2=RandomForestRegressor(n_estimators=250) #using default values
rf_regression2.fit(X_train,y_train)
y_pred_rf=rf_regression2.predict(X_test)

mse(y_test,y_pred_rf)
```

```
0.02960973686713905
```

```
from sklearn.ensemble import RandomForestRegressor
rf_regression3=RandomForestRegressor(n_estimators=500) #using default values
rf_regression3.fit(X_train,y_train)
y_pred_rf=rf_regression3.predict(X_test)

mse(y_test,y_pred_rf)
```

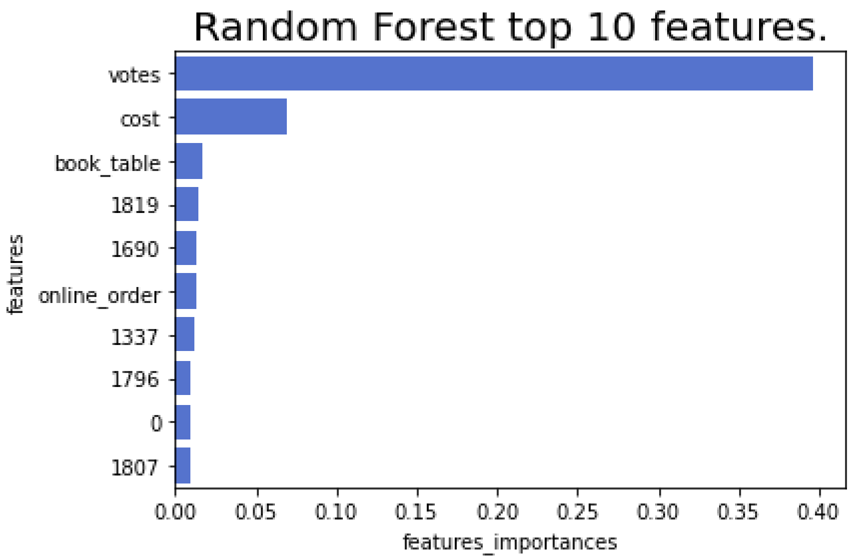
```
0.029275528779385163
```

- directly using the estimators as we have only 1 parameter to tune

```
rf1=imp_df(X_train.columns,rf_regression3.feature_importances_)
rf1.head(10)
```

	features	features_importances
0	votes	0.396829
1	cost	0.068846
2	book_table	0.017283
3	1819	0.014301
4	1690	0.013097

```
var_imp_plot(rf1[:10],'Random Forest top 10 features.')
```



- MSE =0.0294 , It is a good model

Can we still improve our model?

Not Null Features Only

Till now, we have considered ONE-HOT encoding of on below features.

- rest_type
- location
- cuisines
- online_order
- book_table

Here we are going to include below features also,

- dish_liked
- cuisines

Obviously we have to deal with large features set.

```
import pandas as pd
import numpy as np
```

```
onehot = pd.read_csv("zomato.csv")
onehot.head()
```

```
onehot.head()
```

	url	address	name	online_order	book_table	ra
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7
		10, 3rd Floor				

This time we will drop all Null values. Last time we saved some Null values by converting them to relative values. But in this run we will neglect all values null. Initially there are 51k values by removing NULL it will be somewhere around 23k.

```
onehot.isna().sum()
```

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

```
onehot.rate=onehot.rate.replace('NEW',np.nan)
onehot.rate=onehot.rate.replace('-',np.nan)
onehot.dropna(how='any',inplace=True)
onehot['rate'] = onehot.loc[:,'rate'].replace('[ ]','',regex = True) # replace [] with '' string
onehot['rate'] = onehot['rate'].astype(str) # convert to string
```

```
onehot['rate'] = onehot['rate'].apply(lambda r: r.replace('/5','')) # replace '/5' character with ''
onehot['rate'] = onehot['rate'].apply(lambda r: float(r)) # convert string back to float
```

```
onehot.cuisines=onehot.cuisines.str.replace(',','')
onehot.cuisines=onehot.cuisines.astype(str).apply(lambda x : ' '.join(sorted(x.split())))
onehot.cuisines.unique()
```

```
array(['Chinese Indian Mughlai North', 'Chinese Indian North Thai',
       'Cafe Italian Mexican', ...,
       'BBQ Continental Indian Italian North', 'Nepalese Tibetan',
       'Andhra Biryani Hyderabadi'], dtype=object)
```

```
onehot.dish_liked=onehot.dish_liked.str.replace(',','')
onehot.dish_liked=onehot.dish_liked.astype(str).apply(lambda x : ' '.join(sorted(x.split())))
```

```
onehot.rename(columns={'approx_cost(for two people)': 'average_cost'}, inplace=True)
```

```
onehot.average_cost=onehot.average_cost.str.replace(',','')
```

```
onehot['rest_type'] = onehot['rest_type'].str.replace(',' , '')
onehot['rest_type'] = onehot['rest_type'].astype(str).apply(lambda x: ' '.join(sorted(x.split())))
onehot['rest_type'].value_counts().head()
```

```
Casual Dining        7298
Bites Quick          5224
Cafe                 2321
Bar Casual Dining    1308
Dessert Parlor       1074
Name: rest_type, dtype: int64
```

```
onehot.head(2)
```

	url	address	name	online_order	book_table	rate
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1
1	https://www.zomato.com/bangalore/spice-	2nd Floor, 80 Feet Road,	Spice			

```
x=onehot.drop(['rate','name'],axis=1)
y=onehot.rate
```

Train Test Split

```
from sklearn.model_selection import train_test_split
X_train1,X_test1,y_train1,y_test1=train_test_split(x,y,test_size=0.3,random_state=0)
```

```
X_train1.shape,X_test1.shape,y_train1.shape,y_test1.shape

     ((16132, 15), (6914, 15), (16132,), (6914,))



all_features =[]
from sklearn.preprocessing import OneHotEncoder


encoder=OneHotEncoder(handle_unknown='ignore')


#Avoid data leakage
def one_hot_fit_transform(df,col):
  out=df[col].values.reshape(-1,1)
  return encoder.fit_transform(out).toarray(),encoder.get_feature_names([col])

def one_hot_transform(df,col):
  out=df[col].values.reshape(-1,1)
  return encoder.transform(out).toarray()



tr_dummy_rest_type,rest_tr = one_hot_fit_transform(X_train1,'rest_type' )
te_dummy_rest_type = one_hot_transform(X_test1,'rest_type' )
all_features.extend(rest_tr)


tr_dummy_online_order, oo_tr = one_hot_fit_transform(X_train1,'online_order' )
te_dummy_online_order = one_hot_transform(X_test1,'online_order' )
all_features.extend(oo_tr)

tr_dummy_book_table,bt_tr = one_hot_fit_transform(X_train1,'book_table' )
te_dummy_book_table = one_hot_transform(X_test1,'book_table' )
all_features.extend(bt_tr)


tr_dummy_city,loc_tr = one_hot_fit_transform(X_train1,'location' )
te_dummy_city = one_hot_transform(X_test1,'location')
all_features.extend(loc_tr)


tr_dummy_cuisines,cui_tr = one_hot_fit_transform(X_train1,'cuisines' )
te_dummy_cuisines =one_hot_transform(X_test1,'cuisines')
all_features.extend(cui_tr)


tr_dummy_dishliked,dish_tr = one_hot_fit_transform(X_train1,'dish_liked' )
te_dummy_dishliked=one_hot_transform(X_test1,'dish_liked')
all_features.extend(dish_tr)



len(all_features)

     5820


tr_dummy_rest_type.shape, te_dummy_rest_type.shape
```

```
((16132, 52), (6914, 52))
```

```
tr_dummy_online_order.shape, te_dummy_online_order.shape
```

```
((16132, 2), (6914, 2))
```

```
tr_dummy_book_table.shape, te_dummy_book_table.shape
```

```
((16132, 2), (6914, 2))
```

```
tr_dummy_city.shape, te_dummy_city.shape
```

```
((16132, 88), (6914, 88))
```

```
tr_dummy_cuisines.shape, te_dummy_cuisines.shape
```

```
((16132, 1253), (6914, 1253))
```

```
tr_dummy_dishliked.shape, te_dummy_dishliked.shape
```

```
((16132, 4423), (6914, 4423))
```

```
type(tr_dummy_dishliked)
```

```
numpy.ndarray
```

```
type(X_train1)
```

```
pandas.core.frame.DataFrame
```

Create Final Train DF (Concate two Dataframes)

```
tr=pd.DataFrame(pd.np.column_stack([tr_dummy_rest_type,tr_dummy_online_order,tr_dummy_book_table,tr_

l1=X_train1.values.tolist()
l2=tr.values.tolist()
for i in range(len(l1)):
  l1[i].extend(l2[i])
X_train1=pd.DataFrame(l1,columns=X_train1.columns.tolist()+tr.columns.tolist())
X_train1.shape
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning: The pandas.np mo
  """Entry point for launching an IPython kernel.
(16132, 5835)
```

```
te=pd.DataFrame(pd.np.column_stack([te_dummy_rest_type,te_dummy_online_order,te_dummy_book_table,te_
print(te.shape)
l3=X_test1.values.tolist()
l4=te.values.tolist()
for i in range(len(l3)):
```

```
  l3[i].extend(l4[i])
X_test1=pd.DataFrame(l3,columns=X_test1.columns.tolist()+te.columns.tolist())
X_test1.shape
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning: The pandas.np mo
  """Entry point for launching an IPython kernel.
(6914, 5820)
(6914, 5835)
```

Deleting the Unwanted columns

- After OneHot enconding Achieved we will simply remove, redudant features.

```
X_train1.columns
```

```
Index(['url', 'address', 'online_order', 'book_table', 'votes', 'phone',
       'location', 'rest_type', 'dish_liked', 'cuisines',
       ...
       'dish_liked_Sandwiches Tea', 'dish_liked_Shakes Thick',
       'dish_liked_Shawarma', 'dish_liked_Tea', 'dish_liked_Thali',
       'dish_liked_Thali Veg', 'dish_liked_Tikka', 'dish_liked_Vada',
       'dish_liked_Vegetarian', 'dish_liked_Waffles'],
      dtype='object', length=5835)
```

```
X_train1.drop([ 'rest_type','location','cuisines','dish_liked','menu_item','url','phone','reviews_li
```

```
X_test1.drop([ 'rest_type','location','cuisines','dish_liked','menu_item','url','phone','reviews_lis
```

```
"""from joblib import load,dump
dump(X_train1, 'one_hot_X_train')
dump(X_test1, 'one_hot_X_test')
dump(y_train1, 'one_hot_y_train')
dump(y_test1, 'one_hot_y_test')"""
```

```
['one_hot_y_test']
```

```
from joblib import load,dump
X_train1 = load('one_hot_X_train')
X_test1 = load('one_hot_X_test')
y_train1 = load('one_hot_y_train')
y_test1 = load('one_hot_y_test')
```

```
X_train1.shape, y_train1.shape
```

```
((16132, 5822), (16132,))
```

```
X_test1.average_cost=X_test1.average_cost.str.replace(',','')
X_test1.average_cost=X_test1.average_cost.astype('float')
```

Model -1 Linear Regression

```
from sklearn.linear_model import LinearRegression,SGDRegressor
lr = LinearRegression()
lr.fit(X_train1,y_train1)
y_pred_lr = lr.predict(X_test1)

mse(y_test1, y_pred_lr)
```

```
30882363184360.59
```

Model -2 SGDRegressor

```
sgdReg = SGDRegressor()
sgdReg.fit(X_train1,y_train1)
y_pred_sgdr = sgdReg.predict(X_test1)

mse(y_test1, y_pred_sgdr)
```

```
3.706958867157775e+29
```

Model -3 Random Forest Regressor

```
rfr = RandomForestRegressor()
rfr.fit(X_train1,y_train1)
y_pred_rfr = rfr.predict(X_test1)
```

```
mse(y_test1, y_pred_rfr)
```

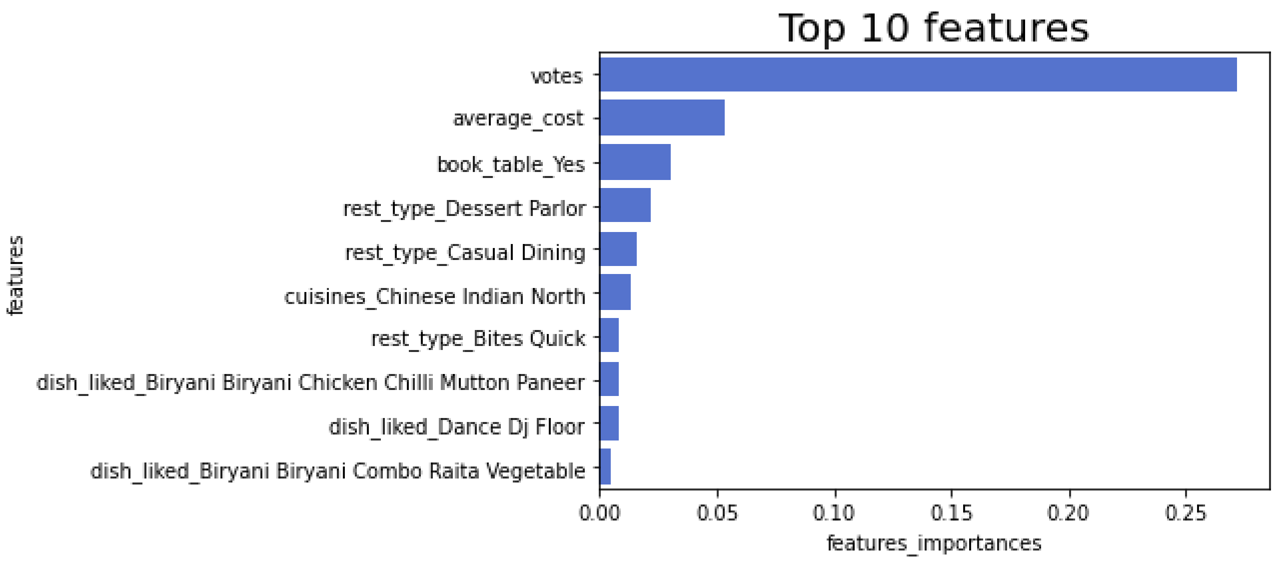
```
0.01838083239950155
```

Feature Importance

```
rf2=imp_df(X_train1.columns,rfr.feature_importances_)
rf2[:10]
```

	features	features_importances
0	votes	0.271903
1	average_cost	0.053305
2	book_table_Yes	0.030178
3	rest_type_Dessert Parlor	0.021961

```
var_imp_plot(rf2[:10],'Top 10 features')
```



This is brilliant, last we saw MSE = 0.018, without hyperparam tuning.

```
rfr = RandomForestRegressor(n_estimators=300)
rfr.fit(X_train1,y_train1)
y_pred_rfr = rfr.predict(X_test1)
```

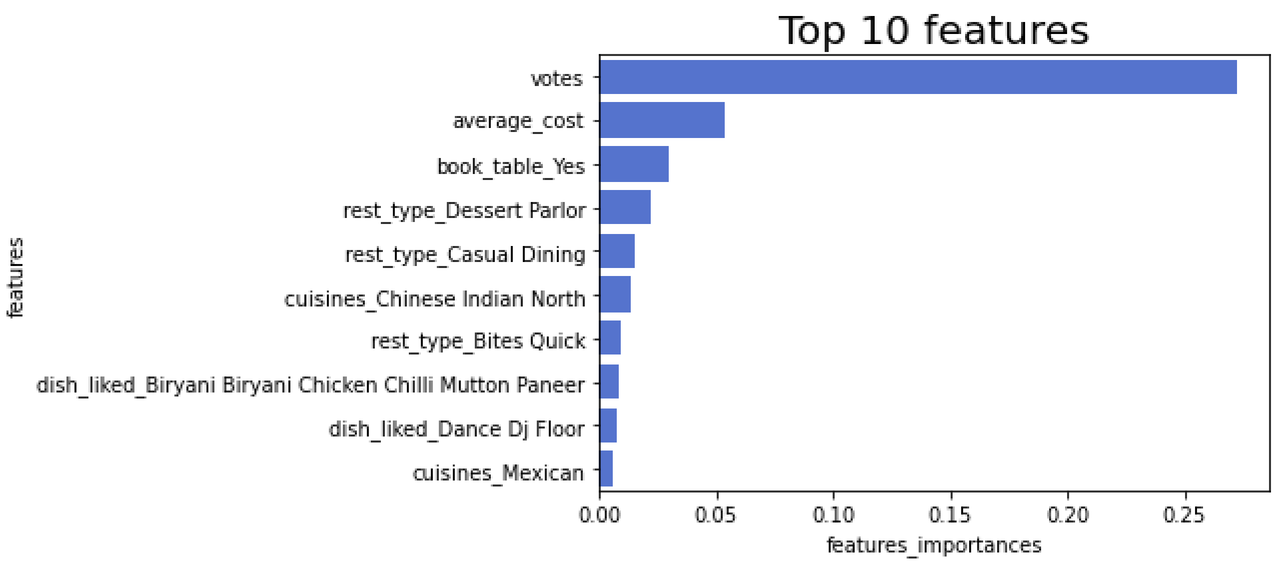
```
mse(y_test1, y_pred_rfr)
```

```
0.018052926704916487
```

```
base_imp = imp_df(X_train1.columns, rfr.feature_importances_)
base_imp[:10]
```

	features	features_importances
0	votes	0.272384

```
var_imp_plot(base_imp[:10],"Top 10 features")
```



Feature Engineering

Let's try response coding in categorical variable on regression model.

Basically what we are going to do replace categorical features with response coded features. In simple words we are going to consider each categorical feature once and find mean value of 'Rate' column.

Eg.==> Consider "online_order" feature, which has two categories, 'Yes' and 'No'. So we will do a small hack, which is explained as below,

- consider category as 'Yes' in 'online_order', take mean value of 'Rate'
- similarly consider second category as 'No' in 'online_order', take mean value of 'Rate' column.
- We will perform above logic using group_by on desired categorical column and simple take a mean of 'Rate' column.
- Create new column which will contain mean values.
- we will called it as MEAN VALUE REPLACEMENT

```
# re-insert rate column in train data
X_train1.insert(2, 'rate', y_train1, allow_duplicates = False)


# re-insert rate column in test data
X_test1.insert(2, 'rate', y_test1, allow_duplicates = False)


# # https://www.geeksforgeeks.org/python-creating-a-pandas-dataframe-column-based-on-a-given-conditi

key_dict = dict()
def provide_response_coded_features(groupByVal,columnName, df):

    '''
    This function is used to convert categorical features into response coded features.
    It simply perform MEAN VALUE REPLACEMENT.
    '''
```

```
    mean_df = df.groupby([groupByVal]).mean()
    mean_dict =mean_df['rate'].to_dict()
    key_dict.update([ (groupByVal, mean_dict) ] )
    for k, v in mean_dict.items():
            mean_dict[k] = round(v,2)
    df[columnName] = df[groupByVal].map(mean_dict)
    return df


# create response coded feature for online_order feature.
mean_online_order =provide_response_coded_features('online_order','mean_online_order',X_train1)
mean_online_order[['rate','online_order','mean_online_order']][:10]
```

	rate	online_order	mean_online_order
31116	3.1	Yes	3.89
22367	4.0	Yes	3.89
19712	3.7	Yes	3.89
25738	3.8	Yes	3.89
43903	4.1	No	3.93
4956	4.6	No	3.93
42870	4.3	Yes	3.89
42025	3.2	Yes	3.89
3392	2.7	No	3.93
14555	3.6	No	3.93

```
# create response coded feature for book_table feature.

mean_book_table =provide_response_coded_features('book_table','mean_book_table',X_train1)
mean_book_table[['rate','book_table','mean_book_table']][:10]
```

	rate	book_table	mean_book_table
31116	3.1	No	3.81
22367	4.0	No	3.81
19712	3.7	No	3.81
25738	3.8	No	3.81
43903	4.1	Yes	4.16
4956	4.6	Yes	4.16
42870	4.3	No	3.81
42025	3.2	No	3.81
3392	2.7	No	3.81
14555	3.6	Yes	4.16

```
# create response coded feature for rest_type feature.

mean_rest_type =provide_response_coded_features('rest_type','mean_rest_type',X_train1)
mean_rest_type[['rate','rest_type','mean_rest_type']][:20]
```

```
mean_rest_type[['rate','rest_type','mean_rest_type']][:20]
```

	rate	rest_type	mean_rest_type
31116	3.1	Casual Dining	3.85
22367	4.0	Bites Quick	3.74
19712	3.7	Casual Dining	3.85
25738	3.8	Cafe	3.99
43903	4.1	Bar Casual Dining	4.17
4956	4.6	Microbrewery Pub	4.47
42870	4.3	Cafe	3.99
42025	3.2	Casual Dining	3.85
3392	2.7	Bites Quick	3.74
14555	3.6	Bar	3.97
41540	4.2	Casual Dining	3.85
37590	4.4	Bar Casual Dining	4.17
14149	3.7	Bar Casual Dining	4.17
51355	3.2	Bites Quick	3.74
39321	4.3	Bar Casual Dining	4.17
8893	4.2	Bites Quick	3.74
38373	3.8	Bites Quick	3.74
1473	3.9	Cafe	3.99
37584	3.7	Bar	3.97
5401	4.0	Cafe	3.99

```
# create response coded feature for dish_liked feature.

mean_rest_type =provide_response_coded_features('dish_liked','mean_dish_liked',X_train1)
mean_rest_type[['rate','dish_liked','mean_dish_liked']][:20]
```

	rate	dish_liked	mean_dish_liked
31116	3.1	Biryani Chicken Firni Mutton Mutton Pasanda Ph...	3.14
22367	4.0	Biryani	3.62
19712	3.7	Aam Cream Doi Fish Gurer Ice Kosha Luchi Mangs...	3.72
25738	3.8	Burger Burgers Chicken Fries Zinger	3.80
43903	4.1	Appam Cocktails Fish Food Kerala Paratha Parot...	4.10
4956	4.6	Beer Beer Bruschettas Cheese Cocktails Craft P...	4.55
42870	4.3	Beef Burger Burger Burgers Cheesy Fries Ice La...	4.30
42025	3.2	Biryani Butter Chicken Chicken Chicken Chicken...	3.20
3392	2.7	Coffee Dosa Kadhai Masala Masala Paneer Poori ...	2.70

```
# create response coded feature for location feature.

mean_location =provide_response_coded_features('location','mean_location',X_train1)
mean_location[['rate','location','mean_location']][:10]
```

	rate	location	mean_location
31116	3.1	Koramangala 5th Block	4.15
22367	4.0	BTM	3.76
19712	3.7	JP Nagar	3.84
25738	3.8	Banaswadi	3.65
43903	4.1	MG Road	3.96
4956	4.6	Sarjapur Road	3.90
42870	4.3	Infantry Road	3.90
42025	3.2	Kaggadasapura	3.60
3392	2.7	Basavanagudi	3.84
14555	3.6	Electronic City	3.69

```
# create response coded feature for cuisines feature.

mean_cuisines =provide_response_coded_features('cuisines','mean_cuisines',X_train1)
mean_cuisines[['rate','cuisines','mean_cuisines']][:10]
```

	rate	cuisines	mean_cuisines
31116	3.1	Biryani Indian Mughlai North	3.84
22367	4.0	Andhra Biryani	3.85
19712	3.7	Bengali Mughlai	3.95

```
for feature,values in key_dict.items():
    print(feature)
```

```
online_order
book_table
rest_type
dish_liked
location
cuisines
```

```
def return_dict_mean_value(query_feature):

    '''
    'key_dict' is dictionary object which has all the Categorical variable names store as KEY and it
    This is function is used to return mean value for query_feature.

    KEY ==>
    Value ==> Mean value response to that key

    query_feature ==>  Desired key
    Return ==> Categorical feature and their corresponding mean values.
    '''

    result_dict=dict()

    for feature_name, values in key_dict.items():
        if feature_name == query_feature:
            for key in values:
                result_dict.update([ (key, values[key]) ] )

                print(key + ':', values[key])
    return result_dict
return_dict_mean_value('online_order')
```

```
No: 3.93
Yes: 3.89
{'No': 3.93, 'Yes': 3.89}
```

```
## Test data

dict_online = return_dict_mean_value('online_order')
dict_book_table = return_dict_mean_value('book_table')
dict_rest_type = return_dict_mean_value('rest_type')
dict_location = return_dict_mean_value('location')
dict_cuisines = return_dict_mean_value('cuisines')
dict_dish_liked = return_dict_mean_value('dish_liked')
```

```
Continental Indian Modern: 4.17
Continental Indian Mughlai North: 4.0
Continental Indian North: 3.95
Continental Indian North Russian: 4.2
```

```
Continental Indian North Russian: 4.2
Continental Indian North Thai: 4.2
Continental Italian: 4.03
Continental Italian Mediterranean Salad: 4.22
Continental Italian Mexican Pizza: 4.1
Continental Italian Pizza: 3.92
Continental Mediterranean Pizza Salad Seafood: 4.53
Continental Steak: 4.01
Cream Cream Ice Ice: 3.9
Cream Desserts Fast Food Ice: 3.9
Cream Desserts Ice: 4.12
Cream Desserts Ice Juices: 3.7
Cream Desserts Ice Rolls: 3.8
Cream Ice: 4.02
Desserts: 4.13
Desserts European Italian Juices Salad Steak: 4.6
Desserts Fast Food: 4.11
Desserts Fast Food Food Rolls Street: 4.0
Desserts Fast Food Indian Indian North South: 3.8
Desserts Fast Food Juices: 3.0
Desserts Fast Food Juices Salad: 3.9
Desserts Food Healthy: 4.4
Desserts Food Healthy Indian Italian North: 4.1
Desserts Food Healthy Italian Kebab Mediterranean Pizza Salad: 4.4
Desserts Food Indian Mithai North Rajasthani Street: 4.3
Desserts Food Indian Mithai North Street: 3.15
Desserts Food Mithai Street: 3.7
Desserts Goan Konkan Seafood: 4.1
Desserts Indian Mughlai North: 3.99
Desserts Italian Pizza: 3.37
Desserts Italian Pizza Salad: 4.31
Desserts Juices: 4.1
Desserts Mithai: 3.9
Desserts Pizza: 3.1
Desserts Turkish: 4.3
Eastern Fast Food Middle Rolls: 4.4
Eastern Food North Street: 3.0
Eastern Lebanese Middle: 4.3
Eastern Mediterranean Middle Rolls Wraps: 3.9
Eastern North: 4.2
European: 4.2
European Fast Food: 4.04
European French Indian North: 3.9
European Indian Mediterranean North: 4.47
European Indian North Thai: 4.3
European Italian: 3.8
European Italian Mediterranean: 4.2
European Italian Mediterranean Salad Spanish: 4.6
European Italian Mediterranean Sandwich: 4.07
European Mediterranean Salad: 4.45
Fast Finger Food Food: 3.7
Fast Food: 3.76
Fast Food Food Healthy: 4.0
Fast Food Food Healthy Salad: 2.4
Fast Food Food Healthy Salad Sandwich: 3.8
Fast Food Food Indian Indian North South Street: 3.25
```

```
X_test1['mean_online_order'] = X_test1['online_order'].map(dict_online)
```

```
X_test1['mean_book_table']=X_test1['book_table'].map(dict_online)
```

```
X_test1['mean_rest_type'] = X_test1['rest_type'].map(dict_rest_type)
X_test1['mean_location'] = X_test1['location'].map(dict_location)
X_test1['mean_cuisines'] = X_test1['cuisines'].map(dict_cuisines)
```

```
X_test1['mean_cuisines'] = X_test1['cuisines'].map(dict_cuisines)
X_test1['mean_dish_liked'] = X_test1['dish_liked'].map(dict_dish_liked)


##check NaN values. NaN value arise because there are some categories those are not present in test
X_test1.isna().sum()

     url                    0
     address                0
     rate                   0
     online_order           0
     book_table             0
     votes                  0
     phone                  0
     location               0
     rest_type              0
     dish_liked             0
     cuisines               0
     average_cost           0
     reviews_list           0
     menu_item              0
     listed_in(type)        0
     listed_in(city)        0
     mean_online_order      0
     mean_book_table        0
     mean_rest_type         1
     mean_location          0
     mean_cuisines         71
     mean_dish_liked      506
     dtype: int64


print("There are some category which is not present in train set which is % ",((1+71+506)/X_test1.sh

     There are some category which is not present in train set which is %  8.35984958056118


# drop null values
X_test1.dropna(subset=['mean_dish_liked','mean_rest_type','mean_cuisines'],inplace=True)


X_test1.isna().sum()

     url                  0
     address              0
     rate                 0
     online_order         0
     book_table           0
     votes                0
     phone                0
     location             0
     rest_type            0
     dish_liked           0
     cuisines             0
     average_cost         0
     reviews_list         0
     menu_item            0
     listed_in(type)      0
     listed_in(city)      0
     mean_online_order    0
     mean_book_table      0
     mean_rest_type       0
     mean_location        0
     mean_cuisines        0
```

```
    mean_dish_liked       0
    dtype: int64
```

```
X_test1.columns
```

```
    Index(['url', 'address', 'rate', 'online_order', 'book_table', 'votes',
           'phone', 'location', 'rest_type', 'dish_liked', 'cuisines',
           'average_cost', 'reviews_list', 'menu_item', 'listed_in(type)',
           'listed_in(city)', 'mean_online_order', 'mean_book_table',
           'mean_rest_type', 'mean_location', 'mean_cuisines', 'mean_dish_liked'],
          dtype='object')
```

```
y_train2=X_train1.rate
y_test2=X_test1.rate
```

```
X_train1.drop(columns=['rest_type','location','cuisines','dish_liked','menu_item','url','phone','rev
```

```
X_test1.drop(columns=['rest_type','location','cuisines','dish_liked','menu_item','url','phone','revi
```

Model - 1 Linear Regression Model

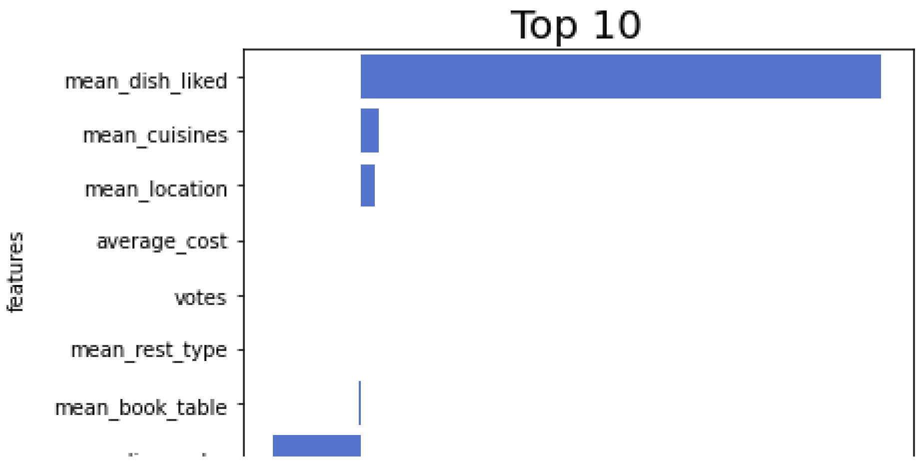
```
lr = LinearRegression()
lr.fit(X_train1,y_train2)
y_pred_lr = lr.predict(X_test1)

mse(y_test2, y_pred_lr)
```

```
    0.14180752704227614
```

```
#feature Importance
lr2=imp_df(X_train1.columns,lr.coef_)
lr2.head(10)
```

	features	features_importances
0	mean_dish_liked	9.834164e-01
1	mean_cuisines	3.214903e-02
2	mean_location	2.609487e-02
3	average_cost	-3.198320e-07
4	votes	-1.056941e-06
5	mean_rest_type	-1.538478e-03
6	mean_book_table	-2.880984e-03
7	mean_online_order	-1.677040e-01

```
var_imp_plot(lr2[:10],'Top 10')
```



In below blog explained negative feature value meaning.

https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e

- In short, it is saying we can remove those features.

```
Randpred = pd.DataFrame({ "actual": y_test2, "pred": y_pred_lr })
Randpred
```

	actual	pred
36986	4.0	4.047717
38937	4.1	4.037844
44154	4.7	3.885596
38806	2.6	3.651617
27613	3.8	3.940625
...	...	...
18623	3.9	4.003893
20371	3.8	3.836152
28317	4.5	4.066092
18507	4.3	3.954557
2852	4.0	3.936202

6393 rows × 2 columns

```
sgdReg = linear_model.SGDRegressor()
sgdReg.fit(X_train1,y_train2)
y_pred_sgd = sgdReg.predict(X_test1)

mse(y_test2, y_pred_sgd)
```

```
5.565606083590284e+29
```

Model is not learning. the mse is very high

```
rfr = RandomForestRegressor()
rfr.fit(X_train1,y_train2)
```

```
y_pred_rfr = rfr.predict(X_test1)

mse(y_test2, y_pred_rfr)
```

    0.13491624857123077

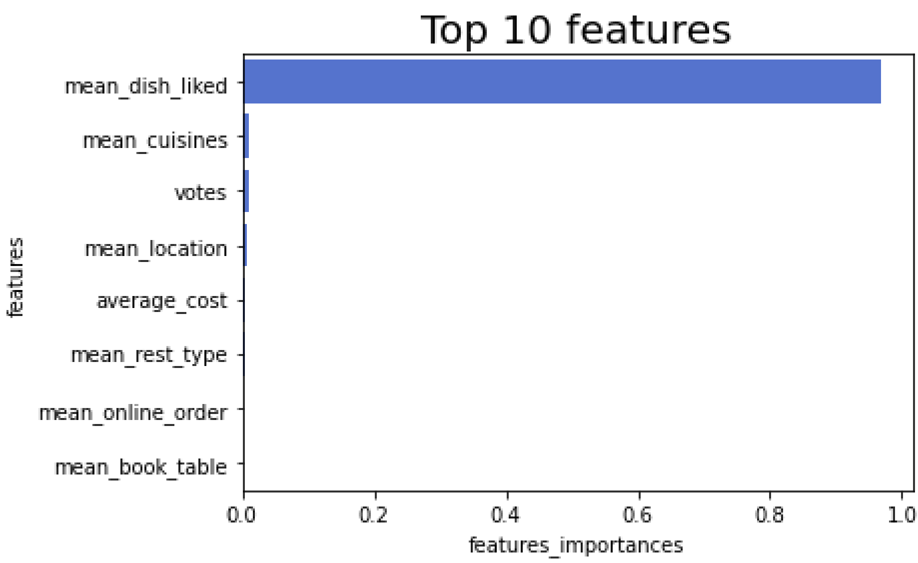
```
rfr1000 = RandomForestRegressor(n_estimators=1000)
rfr1000.fit(X_train1,y_train2)
y_pred_rfr1000 = rfr1000.predict(X_test1)

mse(y_test2, y_pred_rfr1000)
```

    0.13601771841461135

```
base_imp = imp_df(X_train1.columns, rfr1000.feature_importances_)
base_imp[:10]
```

	features	features_importances
0	mean_dish_liked	0.967122
1	mean_cuisines	0.009371
2	votes	0.009148
3	mean_location	0.005525
4	average_cost	0.004176
5	mean_rest_type	0.003258
6	mean_online_order	0.001104
7	mean_book_table	0.000296

```
var_imp_plot(base_imp[:10],"Top 10 features")
```




```
Randpred = pd.DataFrame({ "actual": y_test2, "pred": y_pred_rfr1000 })
Randpred
```

	actual	pred
36986	4.0	4.013200
38937	4.1	4.065209
44154	4.7	3.856598
38806	2.6	3.499052
27613	3.8	3.961525
...	...	...
18623	3.9	4.038724
20371	3.8	3.860393
28317	4.5	4.119603
18507	4.3	3.874475
2852	4.0	3.948338

6. Summary We collect data from CSV file, half of values were missing, we did not throw up all values, instead of throw NULL value we tried to fill estimate values using related colomn. We tried only 5 one-hot encoded features and try different models Random Forest Regressor was most learning model, so we tune model using gridsearch technic, minimal MSE = 0.03485. Then we tried with 7 one-hot encoded features and try on different models.Again Random Forest regressor was winning the race. we achieved MSE = 0.01404.

Then we done some Feature Engineering, used response coded feature, but this time "Linear Regression" perform well than previous model , Random Forest Regressor is winning the race as usual. Finally we achieved MSE =0.00353.

End of the day,below model are best among all the version.

Random Forest Regressor Response coded Features ==> 0.00353 Reference: https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e https://medium.com/@purnasaigudikandula/zomato-bangalore-restaurant-analysis-and-rating-prediction-df277321c7cd https://www.kaggle.com/hindamosh/funny-banglore-restaurants-analysis https://medium.com/@pranaysawant22/zomato-restaurant-rate-prediction-2093cb685430

Summary

We collected data from CSV file, half of values were missing, we did not throw up all values, instead of removing NULL value we tried to fill appropriate values using related columns. We tried only 5 one-hot encoded features and tried different models. Random Forest Regressor was most learning model, so we tuned model using gridsearch technique, minimal MSE = 0.03485. Then we tried with 7 one-hot encoded features on different models.Again Random Forest regressor performed the best.

Then we did some Feature Engineering, used response coded feature, but this time "Linear Regression" perform well than earlier , Random Forest Regressor is winning the race as usual.

Random Forest Regressor with Not Null Features Only ==> 0.01838083239950155

References:

https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e https://medium.com/@purnasaigudikandula/zomato-bangalore-restaurant-analysis-and-rating-prediction-df277321c7cd https://www.kaggle.com/hindamosh/funny-banglore-restaurants-analysis https://medium.com/@pranaysawant22/zomato-restaurant-rate-prediction-2093cb685430