Internship Report - 2022

ALPHA

Zomato Bangalore Restaurants

A Predictive Model

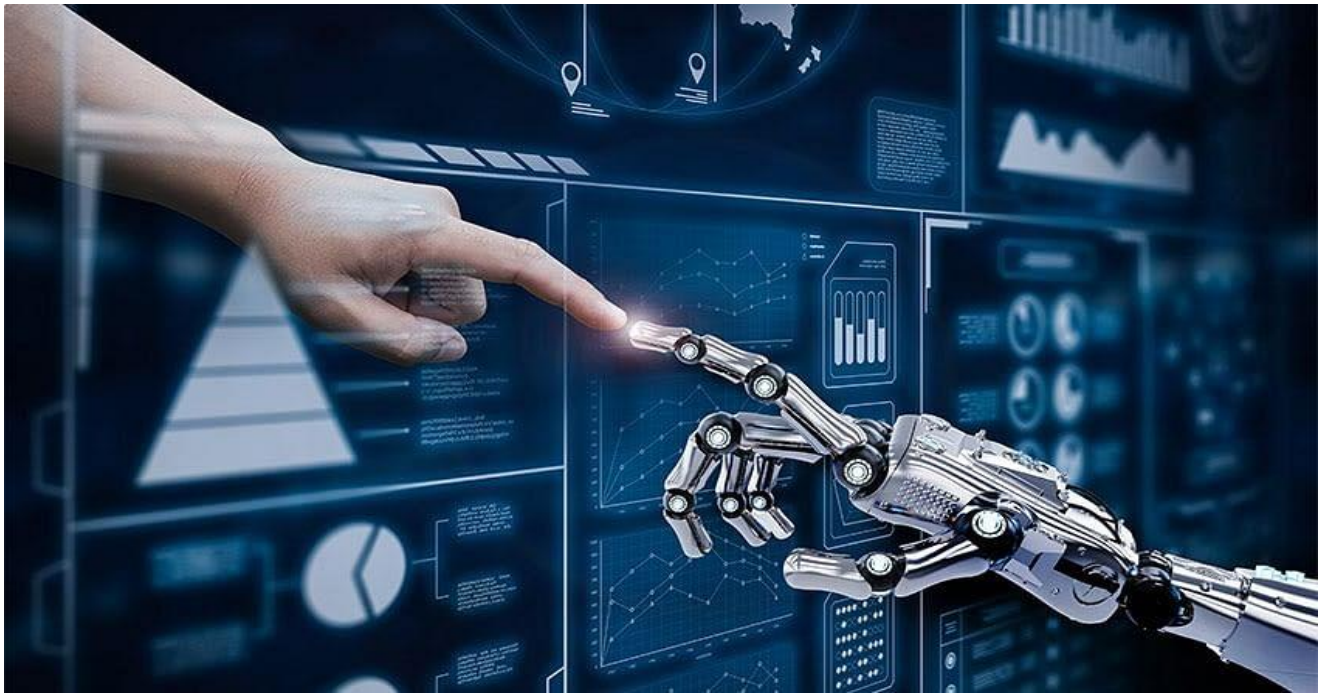
Team Name: ALPHA

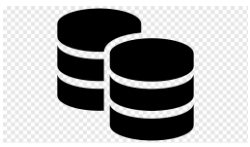
- **Pranesh B**
- **Akshay**
- **Nikith Kashyap**

Co-ordinator
Dr. Mahantesh K

Tequed Labs
Invent - Innovate - Iterate

INDEX





MACHINE LEARNING

CHAPTER 1:

ABSTRACT

Bangalore (officially known as Bengaluru) is the capital and largest city of the Indian state of Karnataka. With a population of over 15 million, Bangalore is the third largest city in India and 27th largest city in the world. Bangalore is one of the most ethnically diverse cities in the country, with over 51% of the city's population being migrants from other parts of India. Bangalore is sometimes referred to as the "Silicon Valley of India" (or "IT capital of India") because of its role as the nation's leading information technology (IT) exporter.

Whenever we visit a new place, we want to go to the best restaurant or the cheapest restaurant, but a decent one. Or we can first look at the ratings or the reviews if we want to try food in some new restaurants. Zomato is one such app that provides users with ratings and reviews of restaurants across India. Ratings or reviews are one of the most significant/decisive variables that decide how good a restaurant is.

Since it is a real time data, we would start with our Data Exploratory process like handling the Nan values, null values, dropping duplicates and other Transformations. Our target variable is the "Rates" column. We explore the relationship of the other features in the dataset with respect to Rates. we will the visualize the relation of all the other depend on features with respect to our target variable, and hence find the most correlated features which effects our target variable. We would then implement the data in various modelling structures such as Random Forest, Linear Regression, and SGD. These modelling will then give us the accuracy of prediction and then we could state which model gives us the most optimized and accurate readings

PROBLEM STATEMENT:

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 restaurants is increasing day by day. Currently which stands at approximately 12,000 restaurants. With such a 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 neighbourhoods of Bengaluru based on food.

Does demography of area matters? Does location of restaurant depend on people living in that area> Does theme of restaurant matters? Is food chain category restaurant likely to have more customers than its counterpart? Are any neighbourhood on similar based on the type of food? Are particular neighbours being famous for its own kind of food? If two neighbours are similar does that mean these are related or 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.

“Build a predictive model for determining prices of houses for users’ input”

DATASET DESCRIPTION

Zomato Bangalore Restaurants

URL : https://www.kaggle.com/datasets/himanshupoddar/zomato-bangalore-restaurants

The basic idea of analysing the Zomato dataset is to get a fair idea about the factors affecting the aggregate rating of each restaurant, establishment of different types of restaurants at different places, Bengaluru being one such city has more than 12,000 restaurants with restaurants serving dishes from all over the world. With each day new restaurants opening the industry hasn't been saturated yet and the demand is increasing day by day. Inspite of increasing demand it however has become difficult for new restaurants to compete with established restaurants. Most of them serving the same food. Bengaluru being an IT capital of India. Most of the people here are dependent mainly on the restaurant food as they don't have time to cook for themselves. With such an overwhelming demand of restaurants it has therefore become important to study the demography of a location. What kind of a food is more popular in a locality? Do the entire locality loves vegetarian food. If yes, then is that locality populated by a particular sect of people for eg. Jain, Marwaris, Gujaratis who are mostly vegetarian. This kind of analysis can be done using the data, by studying different factors.

```
<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
```

Column Description

1. url: contains the URL of the restaurant in the Zomato website
2. address: contains the address of the restaurant in Bengaluru
3. name: contains the name of the restaurant
4. online order: whether online ordering is available in the restaurant or not
5. book table: table book option available or not
6. rate: contains the overall rating of the restaurant out of 5
7. votes: contains total number of ratings for the restaurant as of the above-mentioned date
8. phone: contains the phone number of the restaurant
9. location: contains the neighbourhood in which the restaurant is located
10. rest type: restaurant type
11. dish liked: dishes people liked in the restaurant
12. cuisines: food styles, separated by comma
13. approx_cost(for two people): contains the approximate cost for meal for two people
14. reviews_list: list of tuples containing reviews for the restaurant, each tuple
15. menu_item: contains list of menus available in the restaurant
16. listed_in(type): type of meal
17. listed_in(city): contains the neighbourhood in which the restaurant is listed.

```
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')
```

Observation

1. Rate
2. dish_liked
3. phone
4. approx_cost for two people/entries 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
```

DATA PRE-PROCESSING

Data pre-processing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

Column G Analysis

1. Minimum votes value is 0. This can be interpreted as there are some restaurants which have 0 votes
2. Maximum votes value is 16832 There is a restaurant which has 16832.
3. Average votes value is 284.

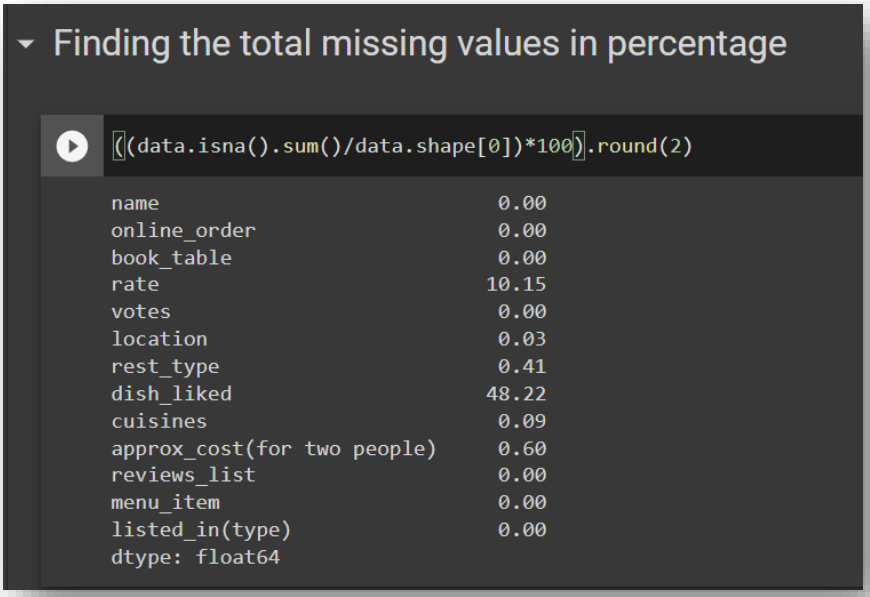
Check point

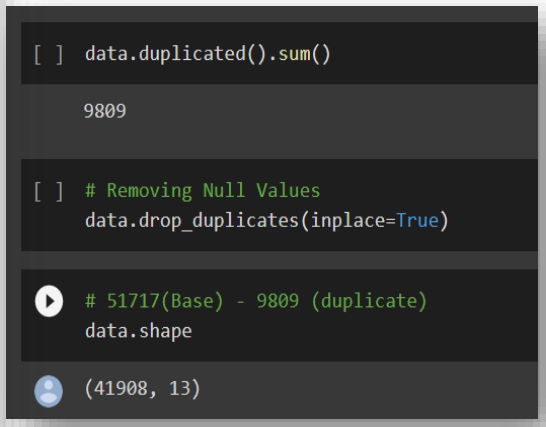
- Here, we can see that 3 columns are representing same information, so just dropping column which are not important.
- we are going to keep the location column and drop the address and listed_in(city) columns.

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

- To remove the rows by index all we have to do is pass the index number or list of index numbers in case of multiple drops. to drop rows by index simply use this code: df. drop(index)
- axis=1 (or axis='columns') is vertical axis. To take it further, if you use pandas method drop, to remove columns or rows, if you specify axis=1 you will be removing columns. If you specify axis=0 you will be removing rows from dataset.

+ Code + Text





The Dataset contained 17 Attributes.

• Records with null values were dropped from ratings columns and were replaced in the other columns with a numerical value.

• Values in the 'Rating' column were changed. The '/5' string was deleted. For eg. If the rating of a restaurant was 3.5/5, it was changed to 3.5.

• Using Label Encoding from sklearn library, encoding was done on columns like book_table, online_order, rest_type, listed_in (city).

Check point :

- We can observe 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.
- data.isna().sum() returns the number of missing values in each column.
- DataFrame. shape. Return a tuple representing the dimensionality of the DataFrame.

Analysis and Pre-processing of Ratings & Reviews:

```
# Rating Analysis
d=data.rate
d
```

```
0         4.1/5
1         4.1/5
2         3.8/5
3         3.7/5
4         3.8/5
          ...
51712    3.6 /5
51713       NaN
51714       NaN
51715    4.3 /5
51716    3.4 /5
Name: rate, Length: 41908, dtype: object
```

```
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)
```

```
## Traversing through the ratings
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
```

There are some points which has 'NEW' rating and '-' rating.

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

The above code is used to replace multiple values with multiple new values for an individual DataFrame column.

```
# Before
data['rate']
```

```
0         4.1/5
1         4.1/5
2         3.8/5
3         3.7/5
4         3.8/5
          ...
51712    3.6 /5
51713       NaN
51714       NaN
51715    4.3 /5
51716    3.4 /5
Name: rate, Length: 41908, dtype: object
```

```
# After
data['rate']
```

```
0        4.1
1        4.1
2        3.8
3        3.7
4        3.8
        ...
51712    3.6
51713    NaN
51714    NaN
51715    4.3
51716    3.4
Name: rate, Length: 41908, dtype: object
```

```
ast.literal_eval(data.reviews_list.values[1])
```

The ast. literal_eval method is one of the helper functions that helps traverse an abstract syntax tree. This function evaluates an expression node or a string consisting of a Python literal or container display.

Extract the rate value out of a string inside tuple

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

```
'Rated'
```

```
def extract_features_from_review_list(x):

    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)
```

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

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

To drop Null Vaues we need to use the below code and the reframed dataframe is of (36840 , 13)

```
[ ]  # 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.shape

     (36840, 13)
```

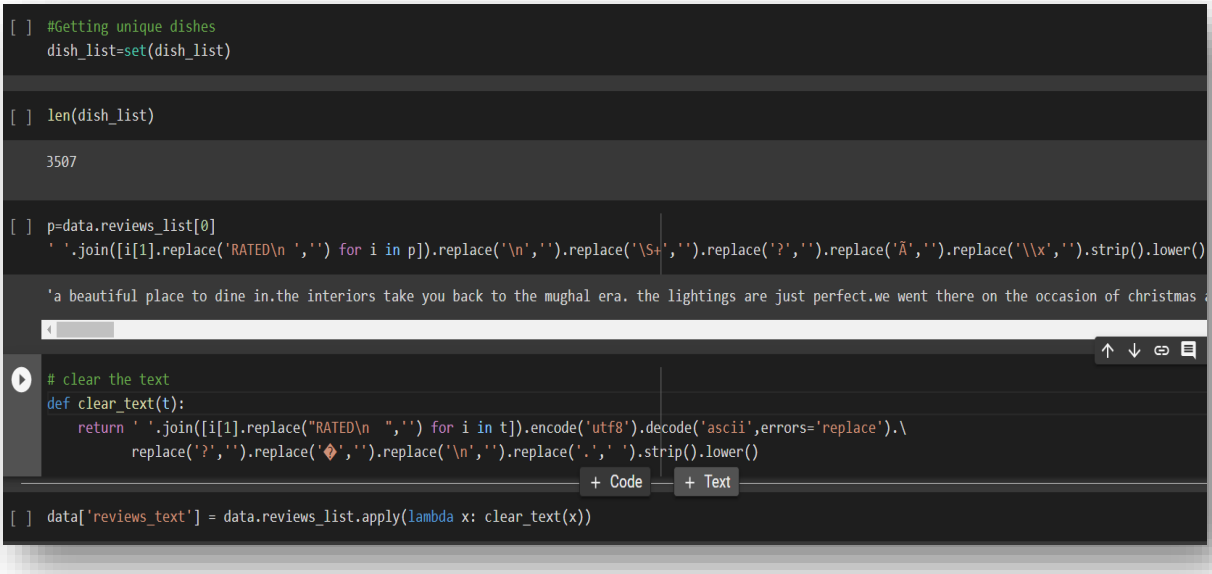
```
data.isna().sum()

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.rest_type.value_counts()

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

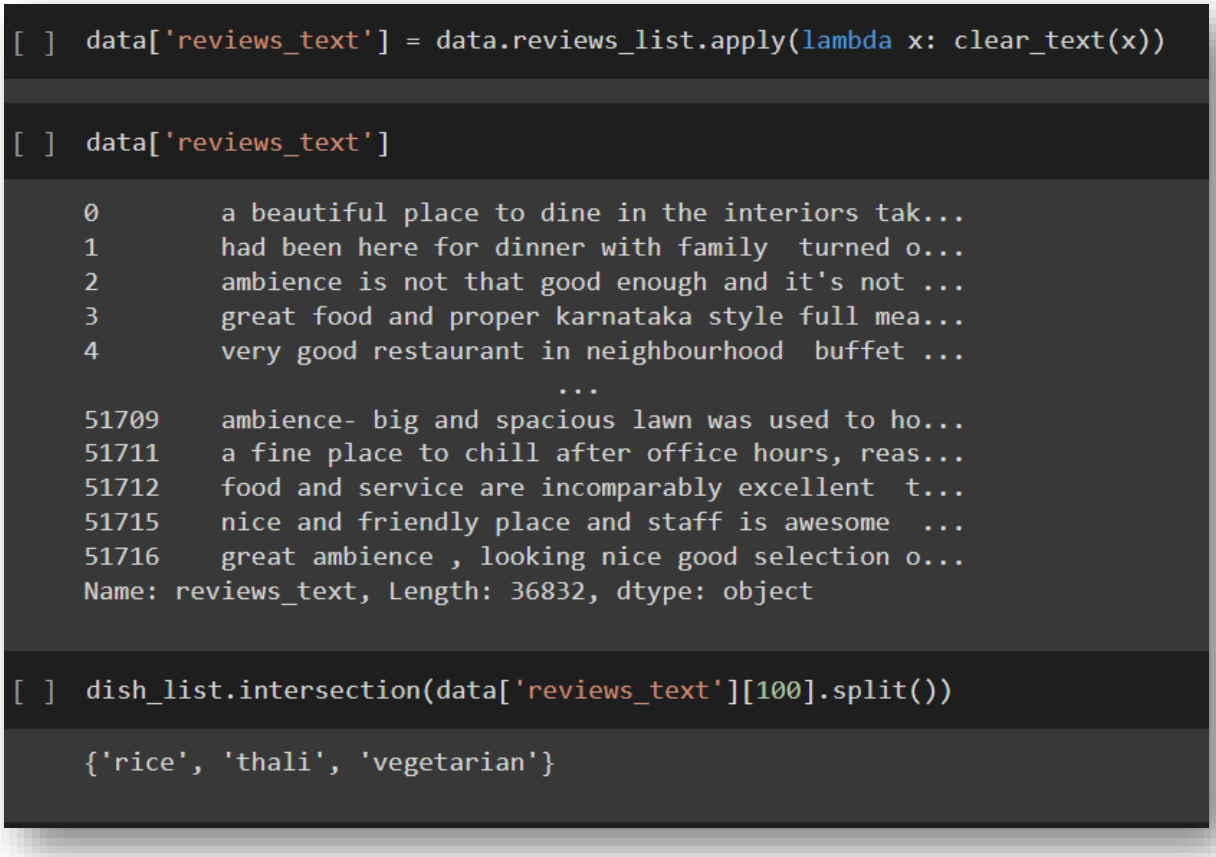
Check Point

- **Value_counts()** function returns object containing counts of unique values. The resulting object will be in descending order so that the first element is the most frequently occurring element.
- Filling missing values of **rest_type** with the most occurring value

Cleaning Reviews_text :



```
[ ] #Getting unique dishes
    dish_list=set(dish_list)

[ ] len(dish_list)

    3507

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

    '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

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

    + Code  + Text

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



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

[ ] data['reviews_text']

    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 ...
    3        great food and proper karnataka style full mea...
    4        very good restaurant in neighbourhood  buffet ...
                                ...
    51709    ambience- big and spacious lawn was used to ho...
    51711    a fine place to chill after office hours, reas...
    51712    food and service are incomparably excellent  t...
    51715    nice and friendly place and staff is awesome  ...
    51716    great ambience , looking nice good selection o...
    Name: reviews_text, Length: 36832, dtype: object

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

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

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

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

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)
```

Check Point

- As we can see dishes liked or disliked are mentioned in reviews,
- If we can extract these dishes, we can fill the nan values of dish_liked column
- we will start by getting a list of all the dishes available from our dataset

```
[ ] 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)
```

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

```
0
```

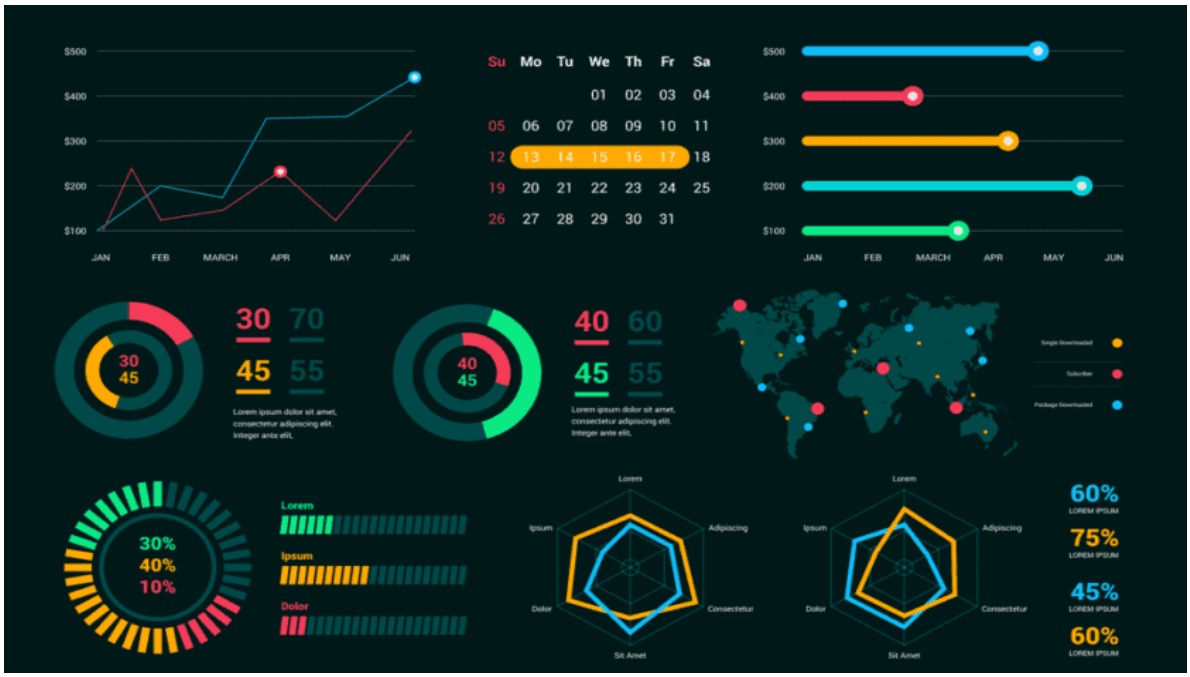
```
# get sample to compare
data.query('dish_liked != dish_liked')[['dish_liked','dish_liked_new']].sample(5,random_state=1)
```

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

```
data
```

	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	cost	reviews_list
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	pasta, lunch buffet, masala papad, paneer laja...	North Indian, Mughlai, Chinese	800	[(Rated 4.0, RATED\n A beautiful place to din...
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	momos, lunch buffet, chocolate nirvana, thai g...	Chinese, North Indian, Thai	800	[(Rated 4.0, RATED\n Had been here for dinner...
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	churros, cannelloni, minestrone soup, hot choc...	Cafe, Mexican, Italian	800	[(Rated 3.0, RATED\n Ambience is not that goo...
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	masala dosa	South Indian, North Indian	300	[(Rated 4.0, RATED\n Great food and proper Ka...
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	panipuri, gol gappe	North Indian, Rajasthani	600	[(Rated 4.0, RATED\n Very good restaurant in ...

CHAPTER 2: DATA VISUALIZATION



Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

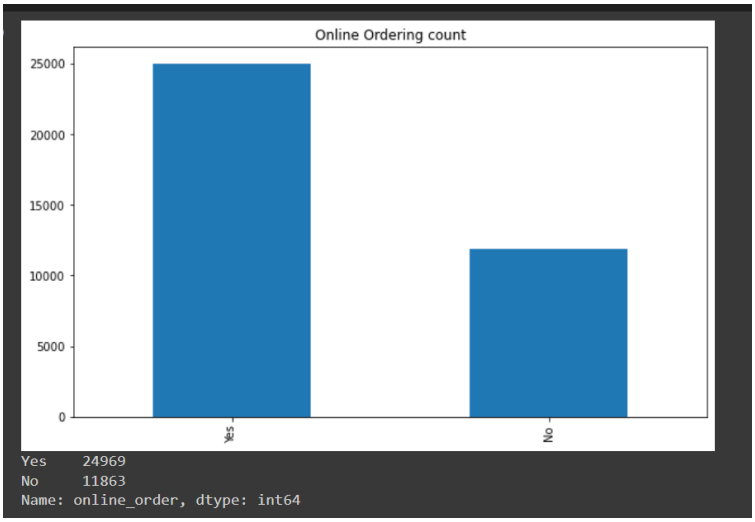
```
plt.figure(figsize=(10,6))

o.plot(kind='bar')
plt.title('Online Ordering count')
plt.show()
oo
```

Data Visualization

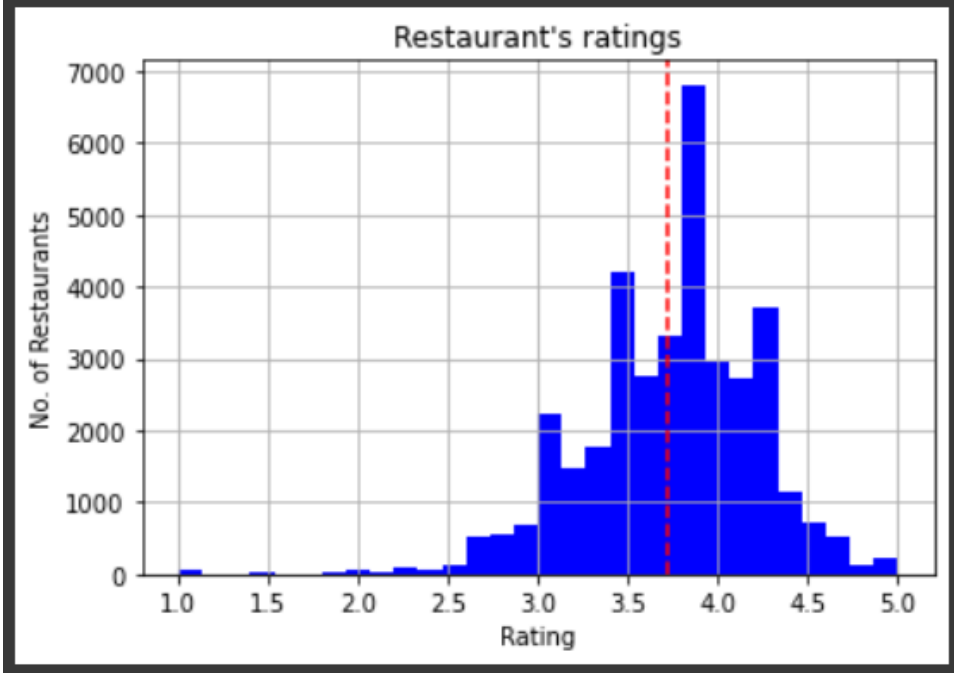
How many Restaurant accepting online orders?

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

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



```
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())
```



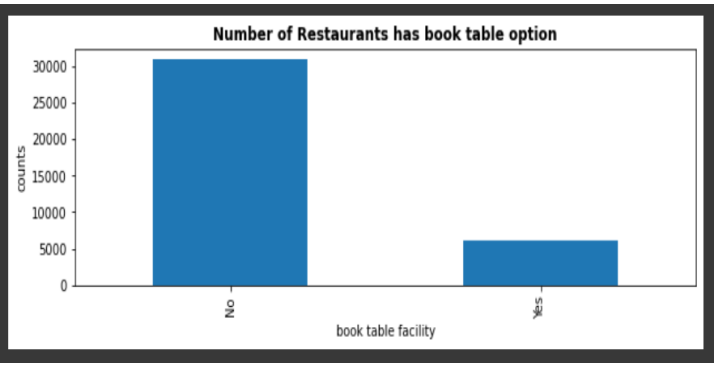

```
3.7208921589921835
```

```
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()
```

Table Booking Facility

```
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>
```

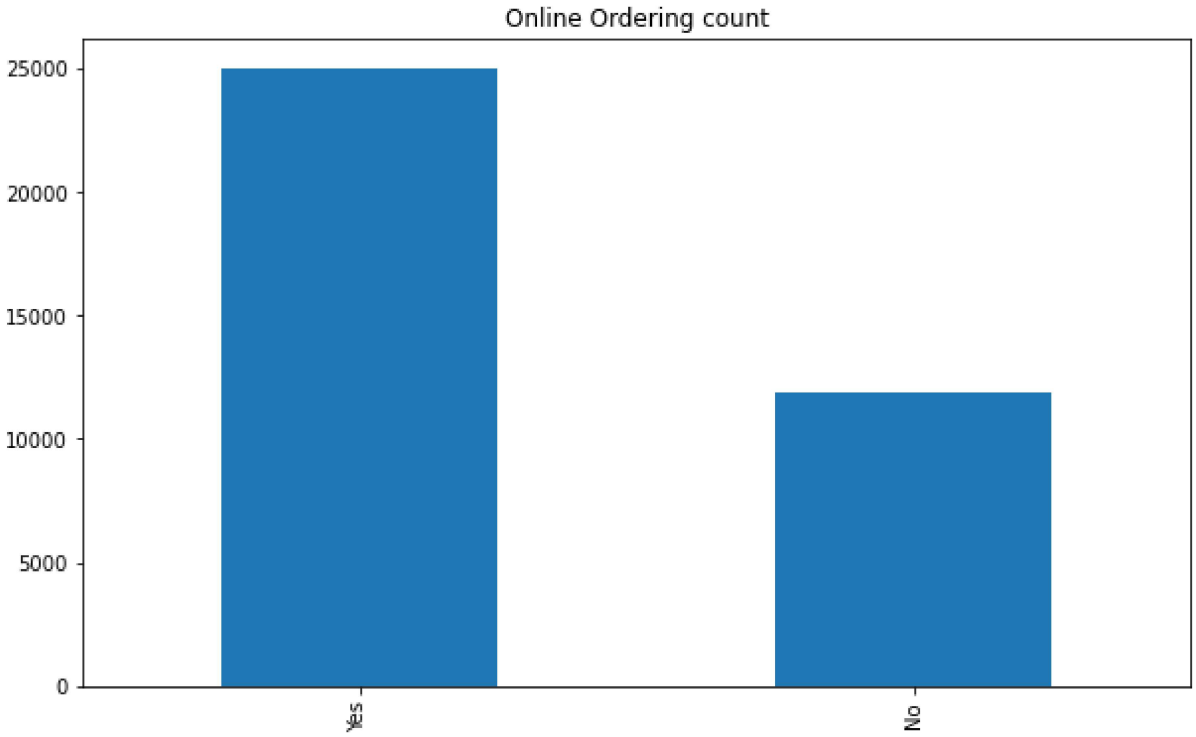


Data Visualization

How many Restaurant accepting online orders?

In [ ]:
```
oo=data.online_order.value_counts()
```

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

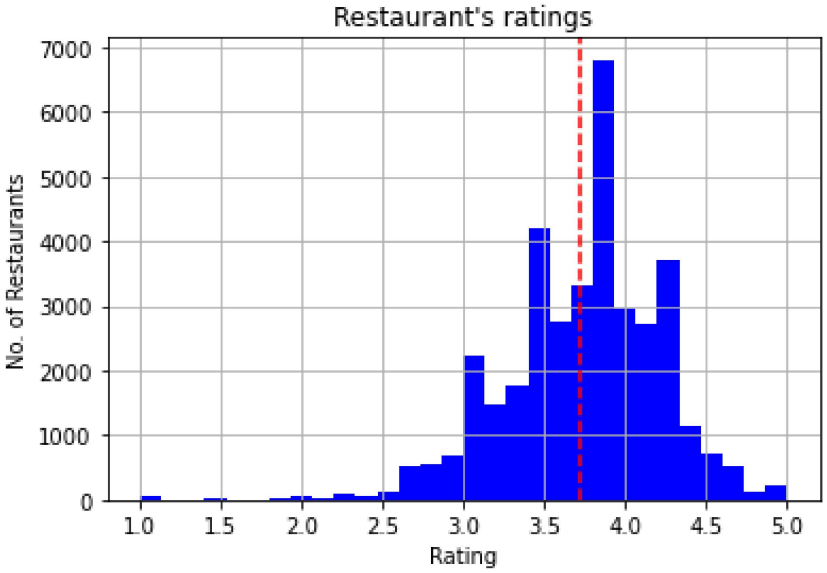


Out[ ]:
```
Yes    24969
No     11863
Name: online_order, dtype: int64
```

Ratings

In [ ]:
```
data.rate=data.rate.astype('float')
```

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



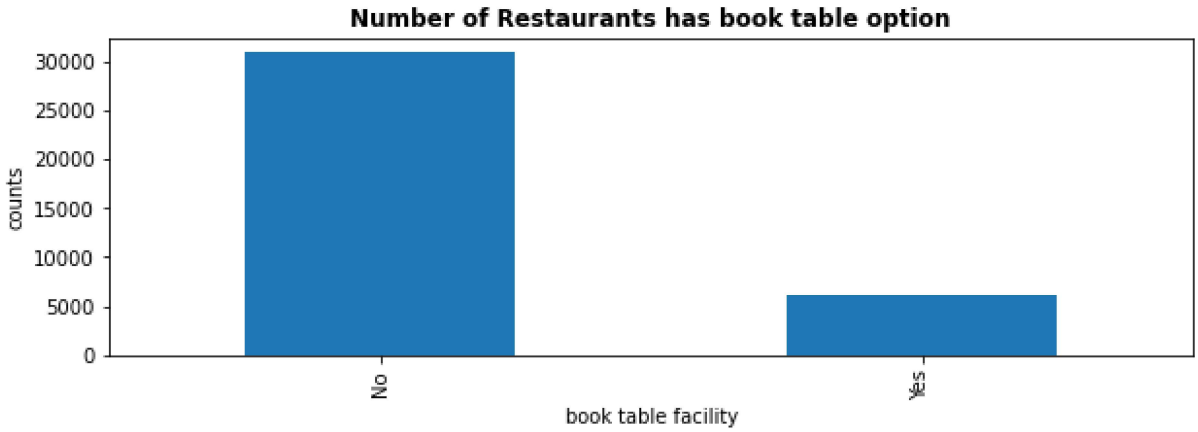

```
3.7208921589921835
```

Table Booking Facility

In [ ]:
```
data.book_table.value_counts
```

Out[ ]:
```
<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>
```

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



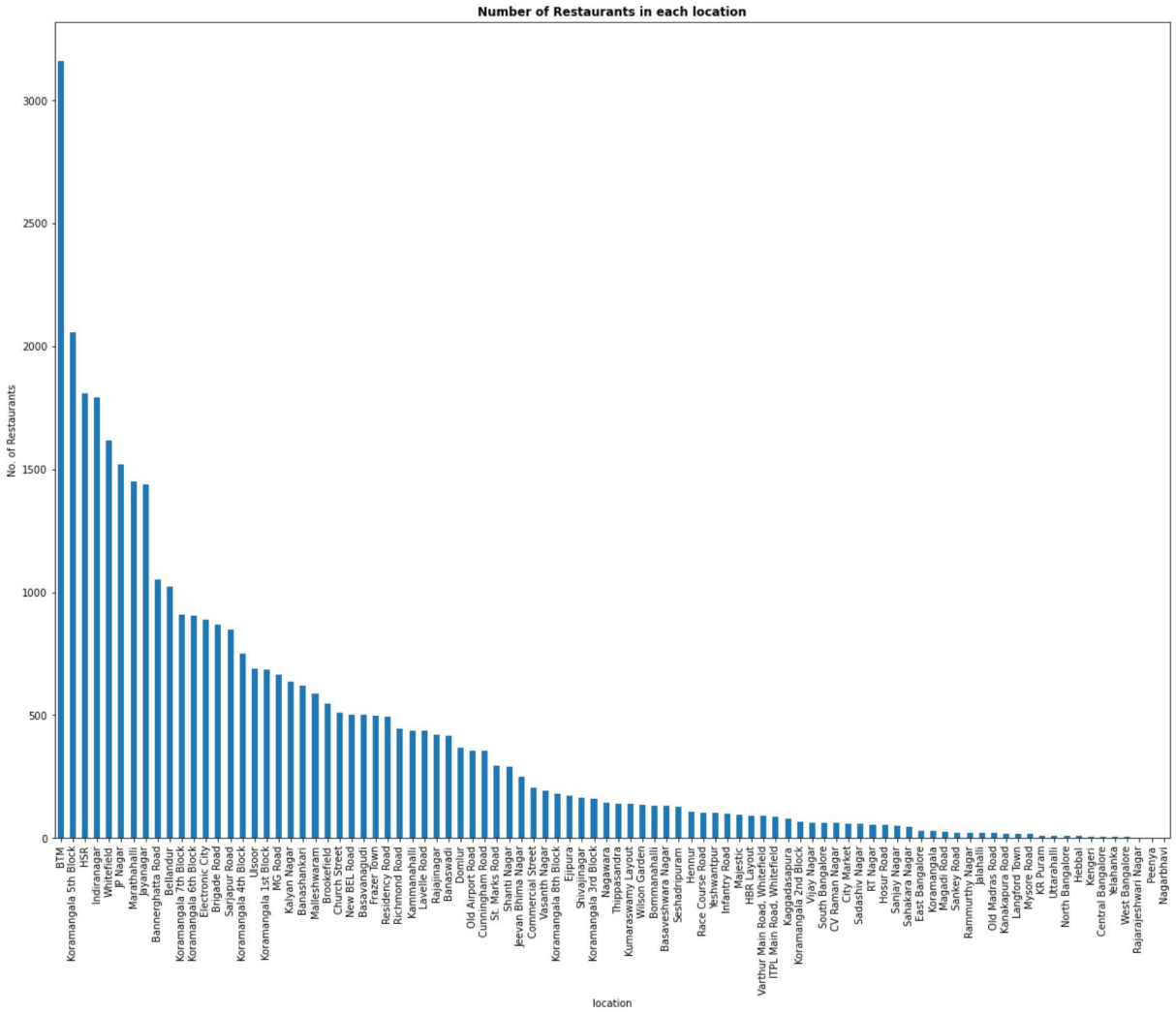
Number of Restaurants in each location (Bengalore)

In [ ]:
```
data.location.value_counts()
```

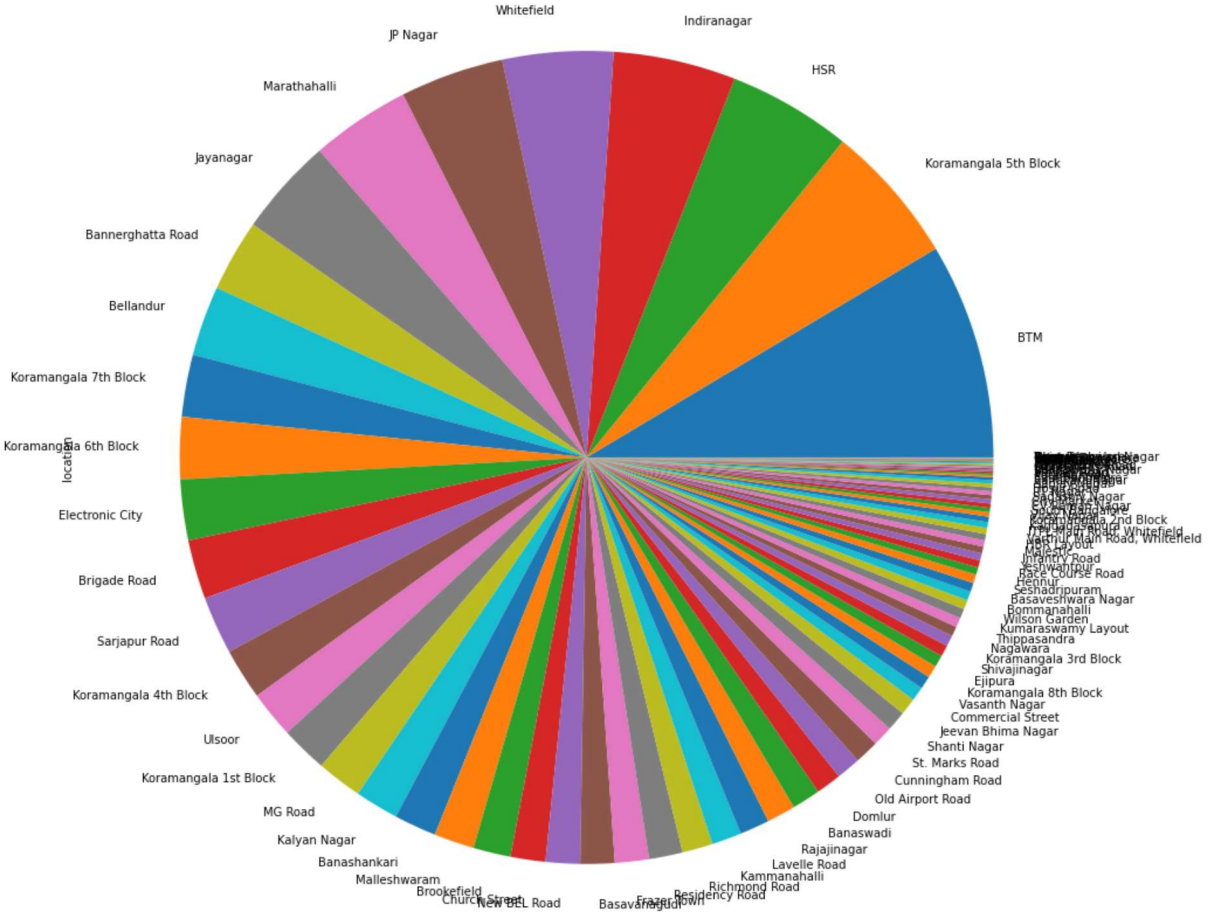
Out[ ]:
```
BTM                      3161
Koramangala 5th Block    2056
HSR                      1807
Indiranagar              1793
Whitefield               1616
                         ...
Yelahanka                   4
West Bangalore              4
Rajarajeshwari Nagar        2
Peenya             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

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

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

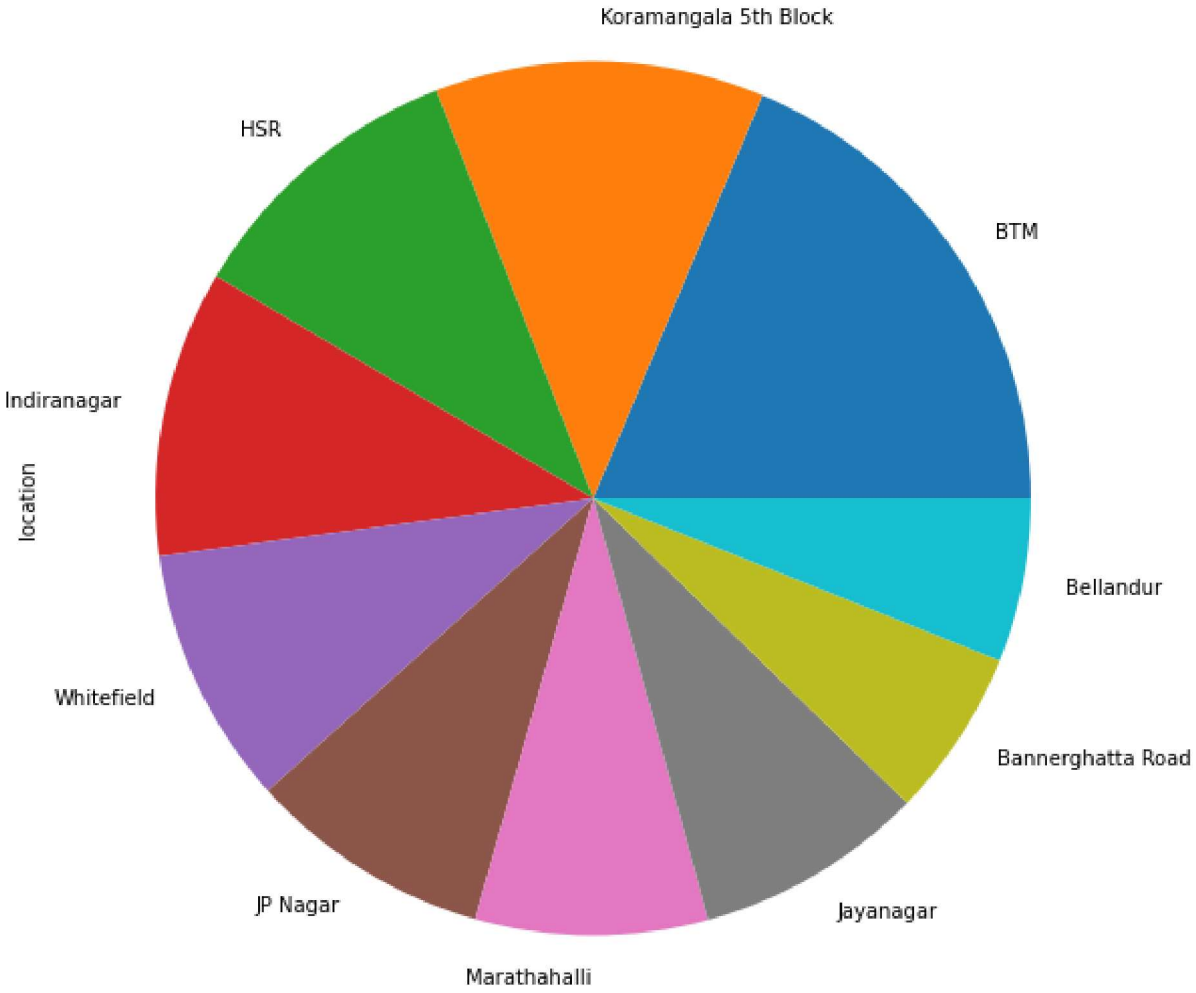


In [ ]:
```
#plotting pie chart
plt.figure(figsize=(20,16))
ax=data.location.value_counts().plot(kind='pie')
plt.show()
```



- Due to huge size and complexity...
- limitting to TOP 10 hotspots

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

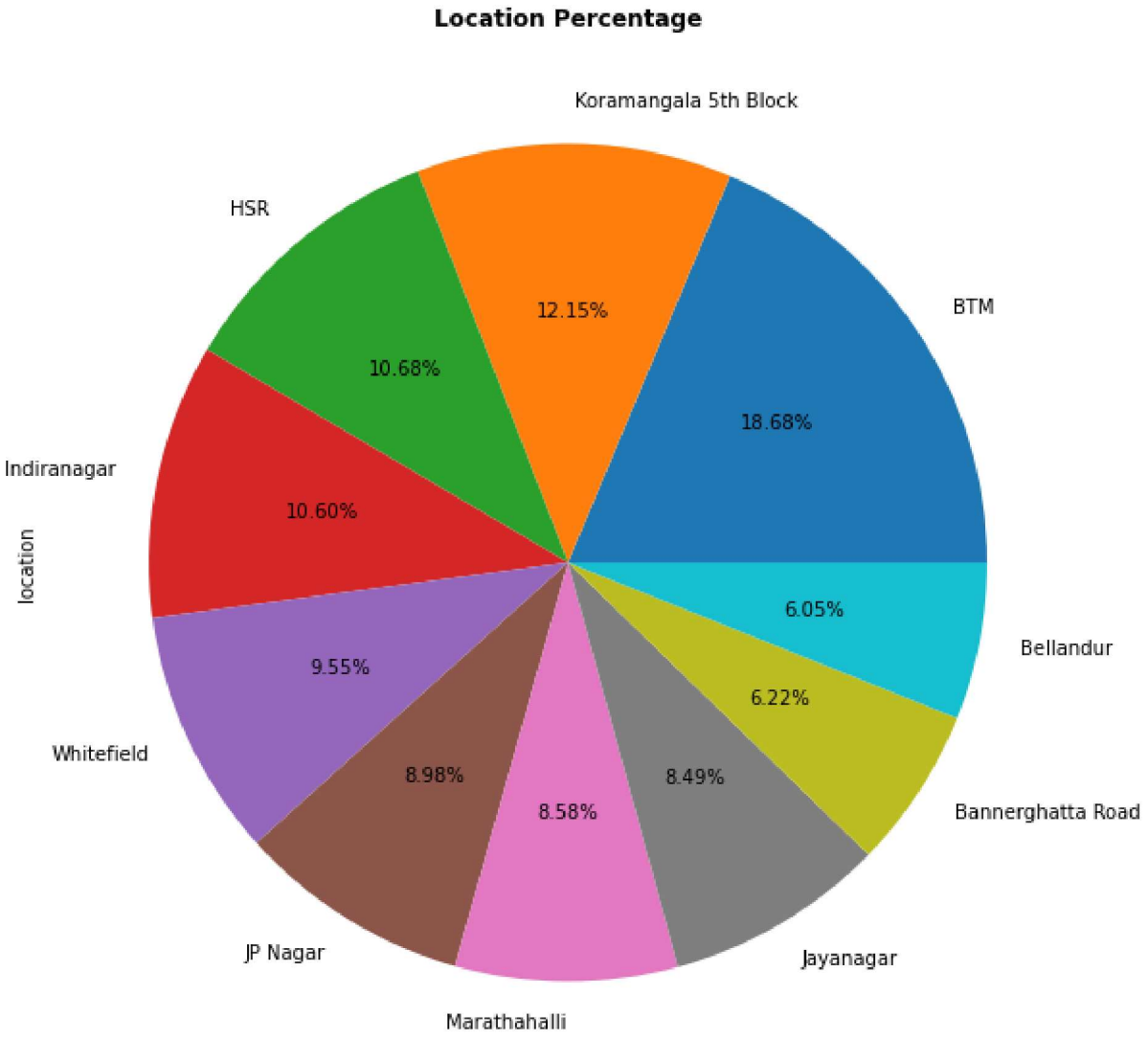


Inference

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

In [ ]:

```
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()
```

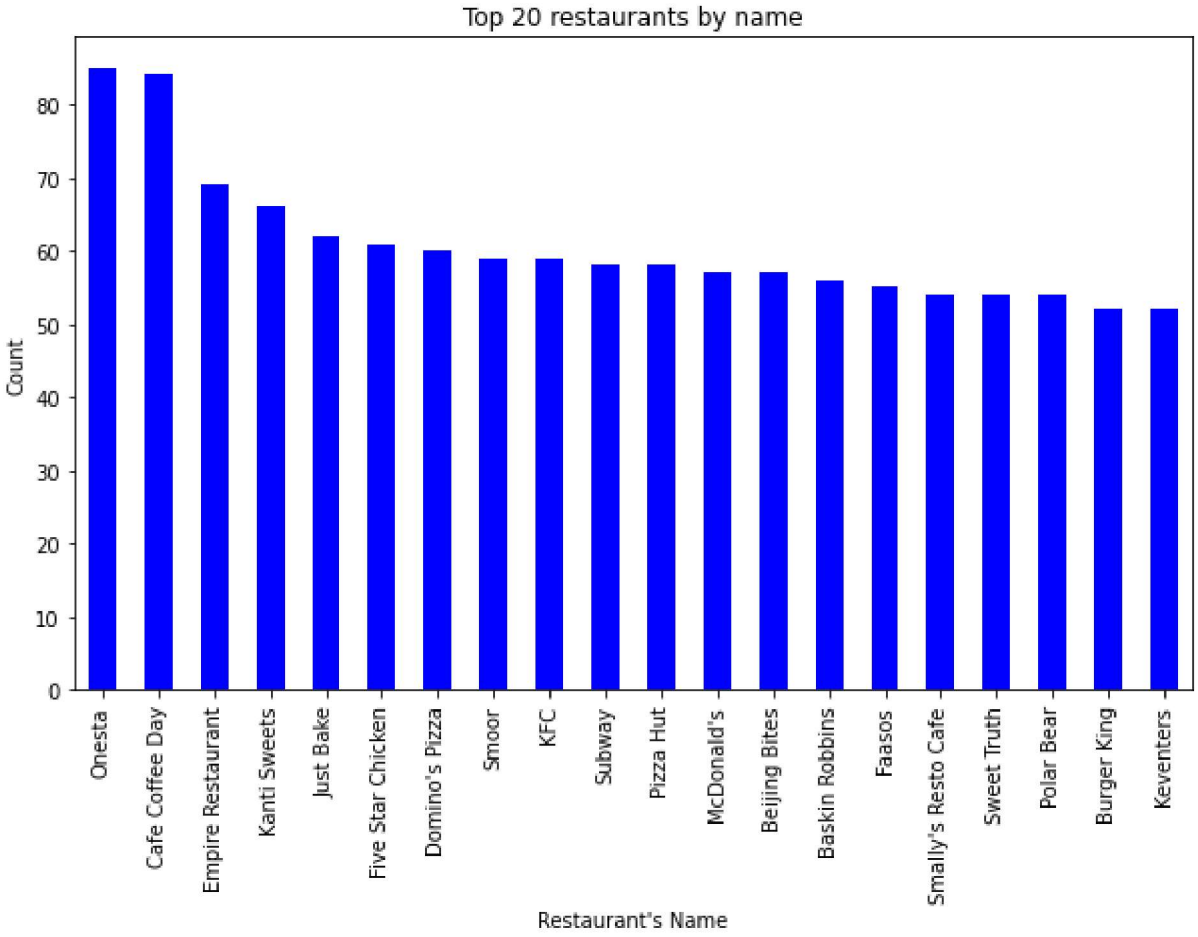


Top 20 restaurants by name

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

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

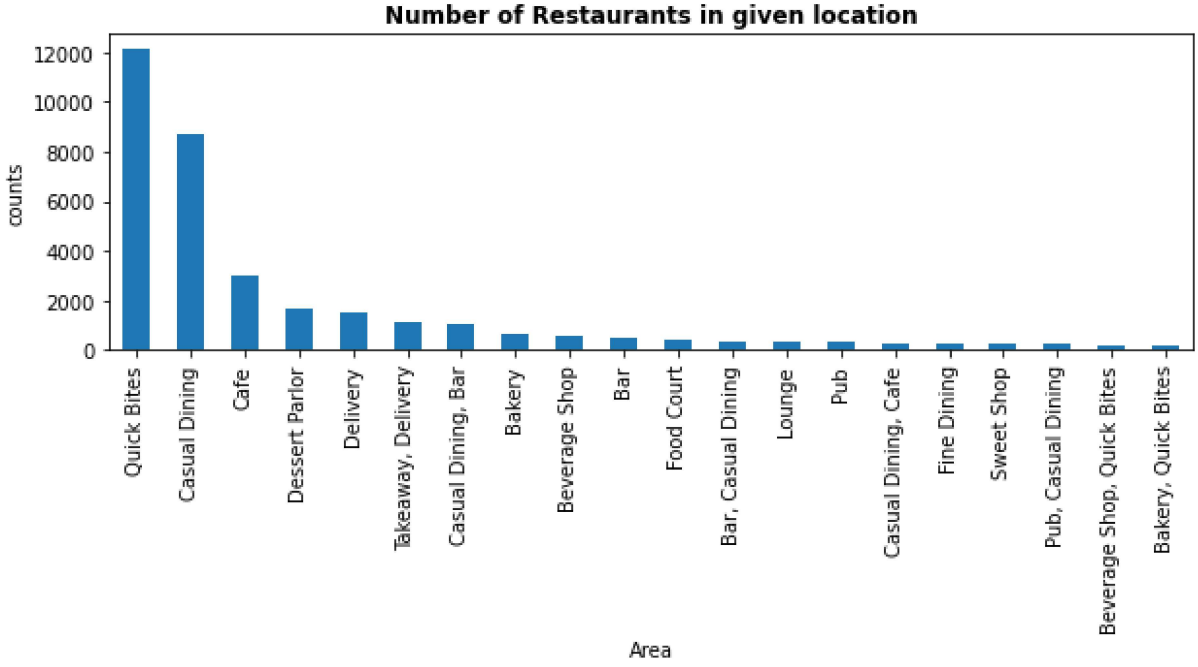
In [ ]:
```
plt.figure(figsize=(10,6))
ax=top_20.plot(kind='bar',color='blue')
plt.title('Top 20 restaurants by name')
plt.xlabel("Restaurant's Name")
plt.ylabel('Count')
plt.show()
```



In [ ]:

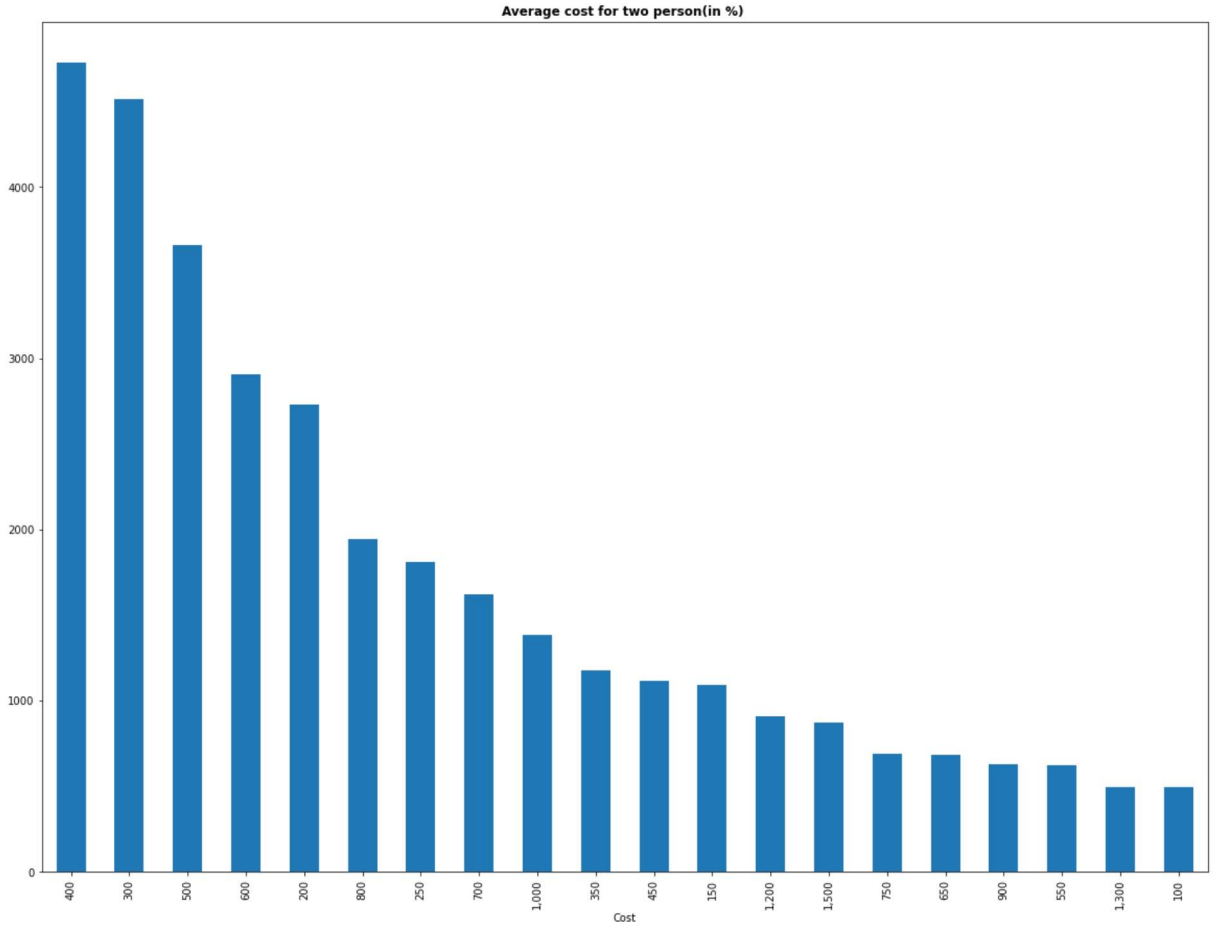
```
plt.figure(figsize=(10,3))
ax =data.rest_type.value_counts()[:20].plot(kind='bar')
plt.title('Number of Restaurants in given location', weight='bold')
plt.xlabel('Area')
plt.ylabel('counts')
```

Out[ ]: Text(0, 0.5, 'counts')



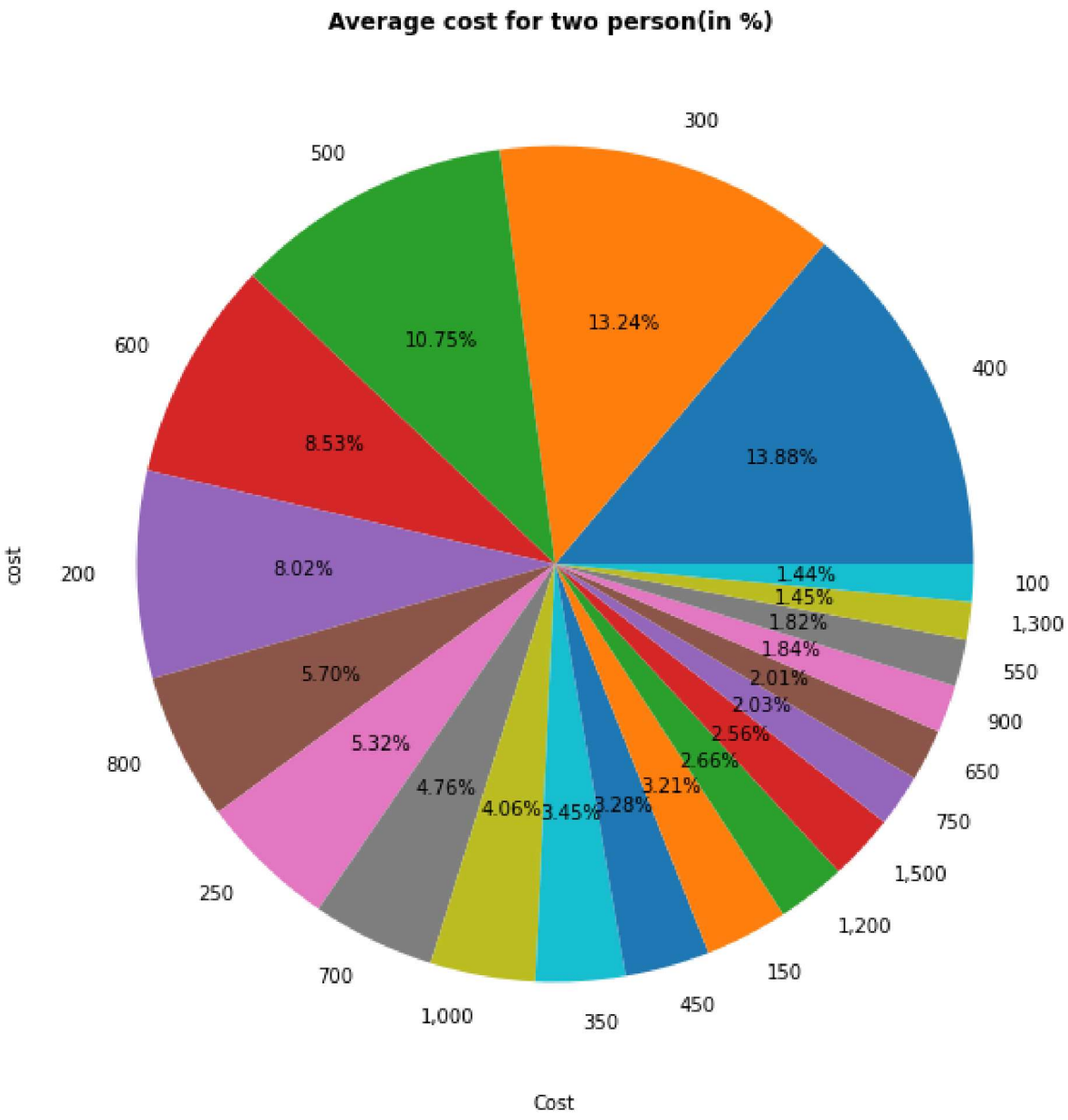
What is the Average cost in restaurants?

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



Average cost for two person(in %)

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

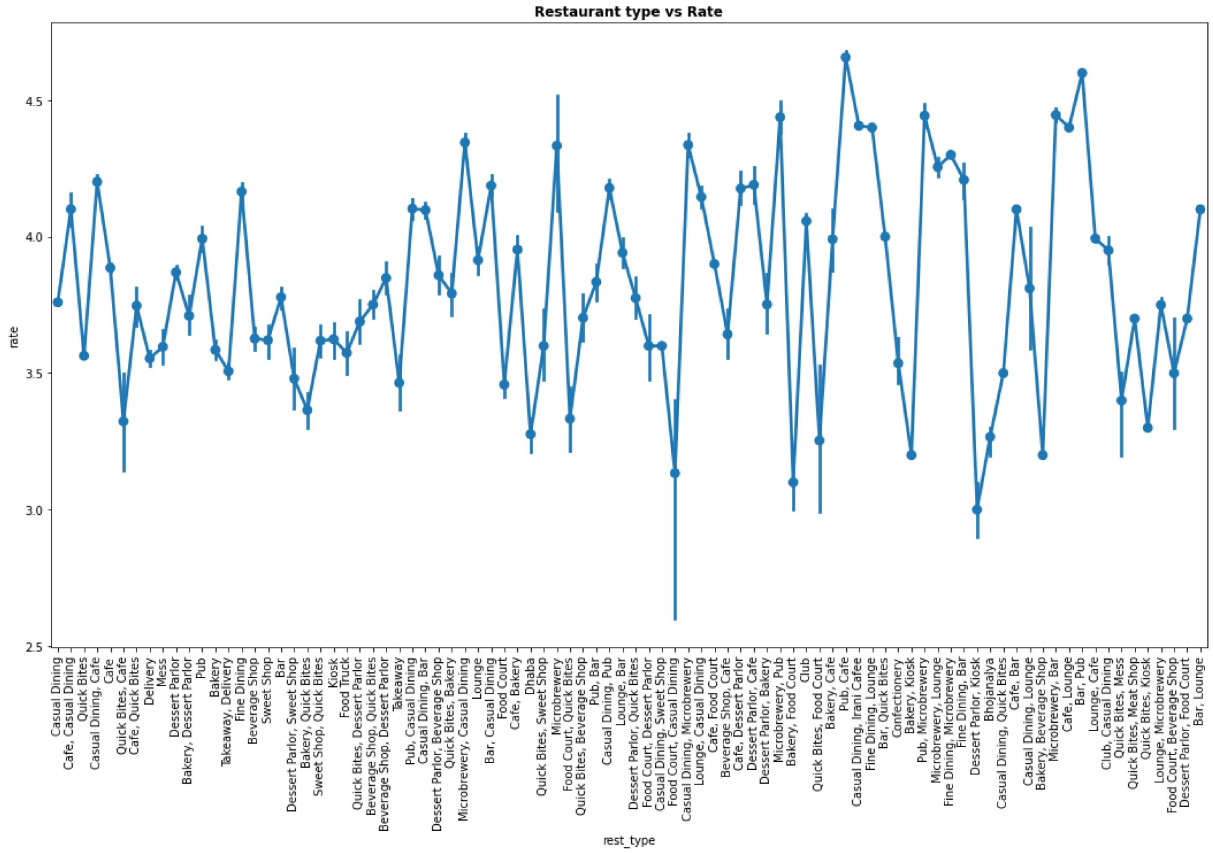


Average cost for 2 is around 300-400 for 27% restaurants and below 500 for approx 52% restaurants

Rate VS Restaurant Type graph

In [ ]:

```
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()
```

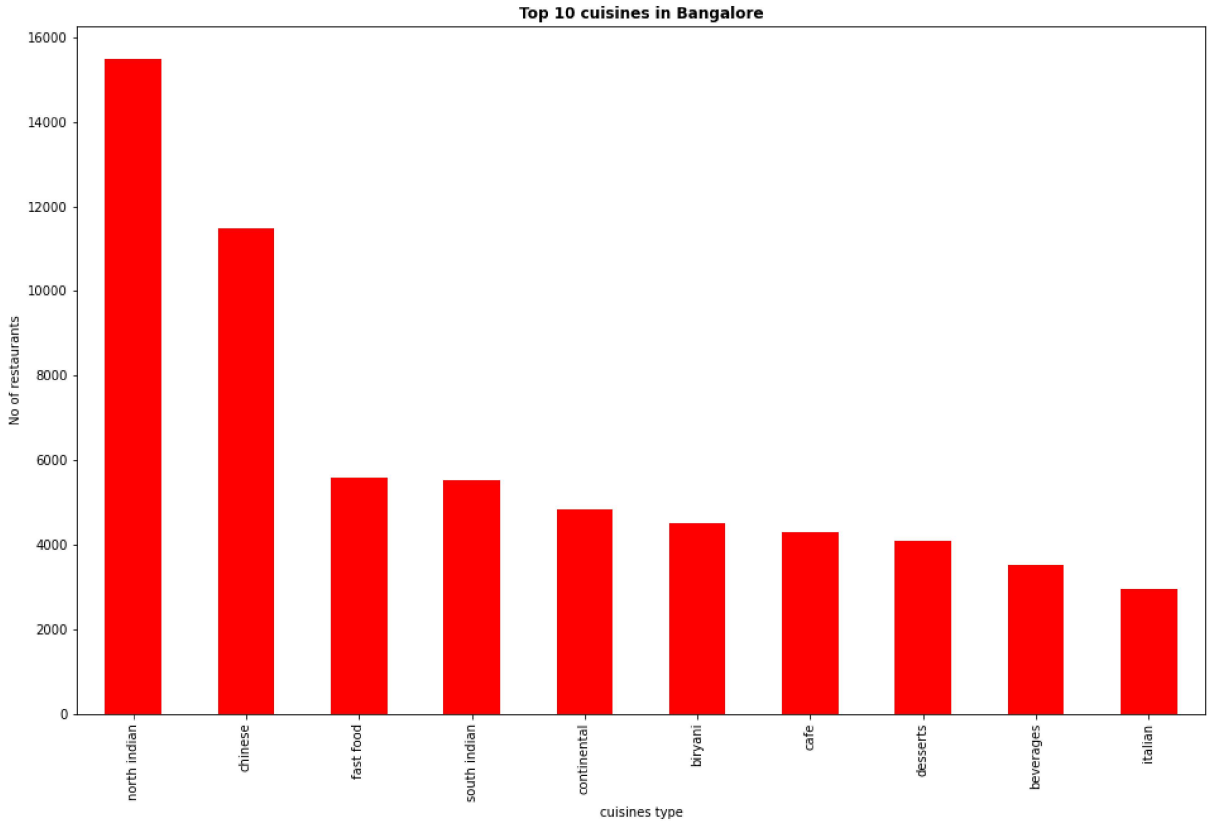


Top 10 Cuisines in Bangalore

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

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

Out[ ]:
```
Text(0, 0.5, 'No of restaurants')
```



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

we have considered various restaurant records with characteristics such as name, average cost, place, whether online order is accepted, we can book a table, restaurant type. This model would help business owners forecast their ranking on the criteria taken into account in our model and enhance the experience of the customer. Different algorithms have been used, but on Random Forest the final model is eventually chosen, which provides the highest precision compared to others.

CHAPTER 3

MACHINE LEARNING

RANDOMIZATION AND SPLITTING OF DATASET

The features selected in the preceding step were approved to develop classification models. Initially the dataset was randomized to obtain an arbitrary permutated sample. It was followed by splitting of the dataset into training (70% of the dataset) and test (30%) sets.

CLASSIFICATION ALGORITHMS

1. Linear Regression:

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is an explanatory variable, and the other is considered to be a dependent variable.

For example, a user might want to relate the weights of individuals to their heights using a linear regression model. Before attempting to fit a linear model to observed data, a user should first determine whether there is a relationship between the variables of interest. This does not necessarily imply that one variable causes the other (for example, higher SAT scores do not cause higher college grades), but that there is some significant association strength of the relationship between two variables. If there appears to be no association between the proposed explanatory and dependent, then fitting a linear regression model to the data probably will not provide a useful model.

A valuable numerical measure of association between two variables is the correlation coefficient, which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables. A linear regression line has an equation of the form Y = a + b X, where X is the explanatory variable and Y is the dependent variable. The slope of the line is b, and a is the intercept (the value of y when x = 0).

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

Splitting the data for Model Building

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

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

```
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)
```

```
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,))
```

```
# Instantiate and fit
lm2 = LinearRegression()
lm2.fit(X_train,y_train)
y_pred=lm2.predict(X_test)
# print the coefficients
print(lm2.intercept_)
print(lm2.coef_)
```

```
from sklearn.metrics import r2_score
a=r2_score(y_test,y_pred)
```

```
print(metrics.mean_absolute_error(y_test, y_pred))
print(metrics.mean_squared_error(y_test, y_pred))
```

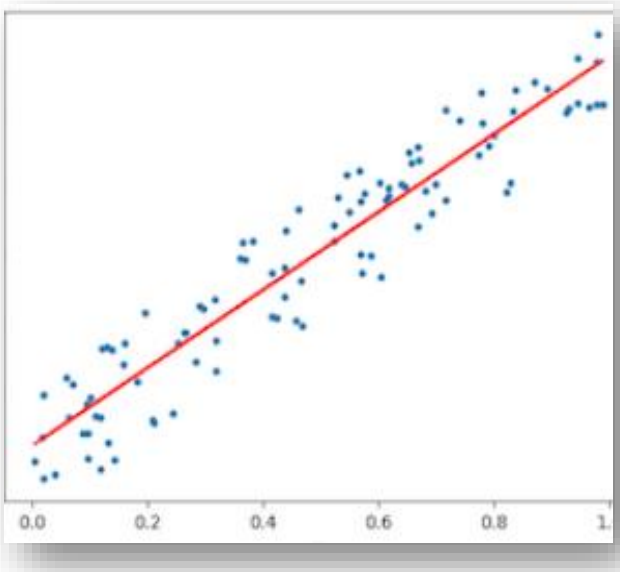
```
predictions = lm2.predict(X_test)
print('Predicted labels: ', predictions)
print('Actual labels:    ' ,y_test)
```

```
Predicted labels:  [3.84729105 3.68957192 3.34571912 ... 3.97695784 3.53371604 4.28525752]
Actual labels:     24008    3.8
6816     4.2
948      3.1
9048     4.1
35049    3.7
        ...
23895    3.8
44714    4.2
6538     4.2
7473     3.0
42856    4.5
Name: rate, Length: 11050, dtype: float64
```

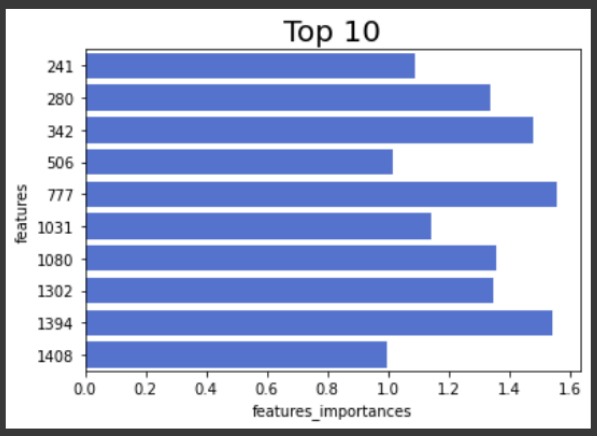
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.4662323649293568
```

```
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

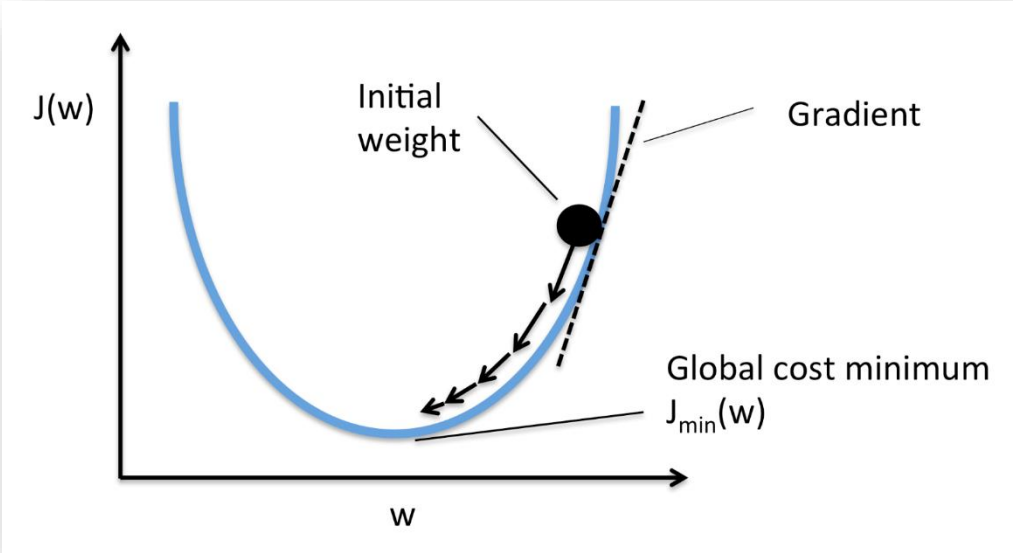



```
var_imp_plot(lm2[:10],'Top 10')
```




```
#Plot
import matplotlib.pyplot as plt
plt.plot(X_test,y_pred,c = 'r')
plt.show
```

2. Stochastic Gradient Descent (SGD)



Stochastic Gradient Descent is an iterative optimization technique that uses minibatches of data to form an expectation of the gradient, rather than the full gradient using all available data.

$$w_{t+1} = w_t - \eta \hat{\nabla}_w L(w_t)$$

Where 'n' is a learning rate. SGD reduces redundancy compared to batch gradient descent - which recomputes gradients for similar examples before each parameter update - so it is usually much faster.

SGD Model

```
[ ] !pip install scikit-learn
```

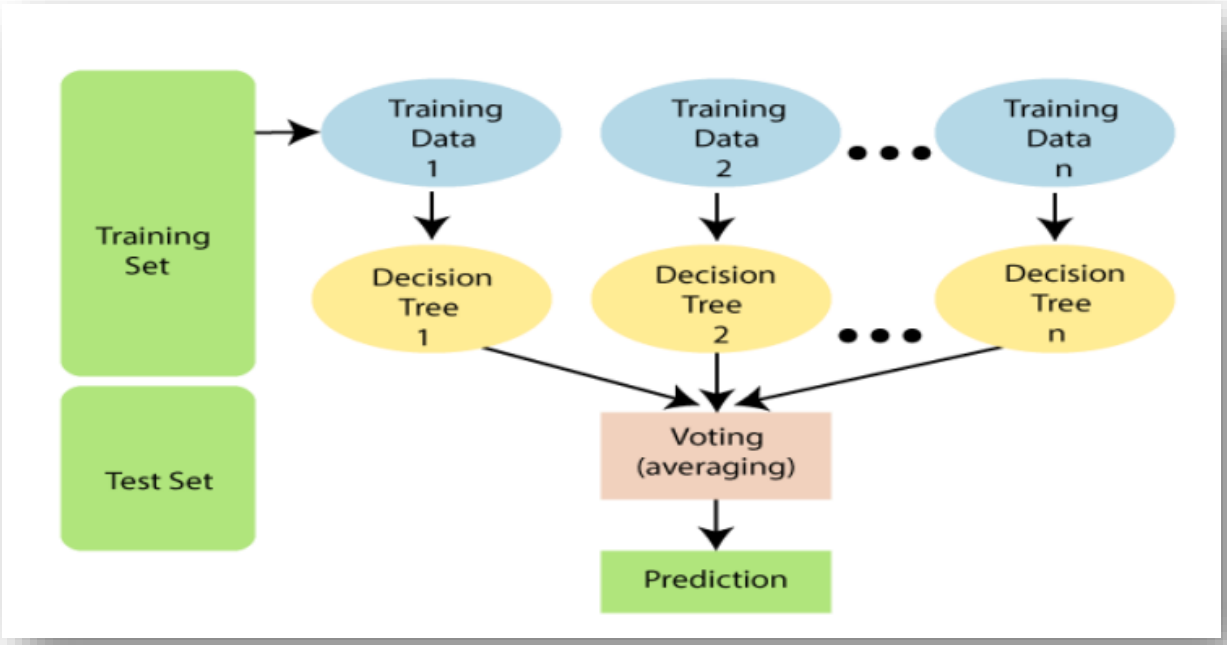
```
Requirement already satisfied: scikit-learn in c:\users\prane\anaconda3\lib\site-packages (0.24.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\prane\anaconda3\lib\site-packages (from scikit-learn) (1.20.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\prane\anaconda3\lib\site-packages (from scikit-learn) (2.1.0)
Requirement already satisfied: scipy>=0.19.1 in c:\users\prane\anaconda3\lib\site-packages (from scikit-learn) (1.6.2)
Requirement already satisfied: joblib>=0.11 in c:\users\prane\anaconda3\lib\site-packages (from scikit-learn) (1.0.1)
```

```
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)
```

```
[ ] # Accuracy Score
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_score(y_test,y_pred_sgd)
```

3. Random Forest

Random forests or random decision forests are an ensemble learning technique for classification. Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.



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

As the name suggests, “Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.” Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting. The diagram explains the working of the Random Forest algorithm.

```
print(metrics.mean_absolute_error(y_test, y_pred_rf))
print(metrics.mean_squared_error(y_test, y_pred_rf))
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred_rf)))
```

```
0.072716913812235 27
0.030483923387334 57
0.174596458690703 62
```

Working Of Random Forest Algorithm

We can understand the working of Random Forest algorithm with the help of following steps –

- Step 1 – First, start with the selection of random samples from a given dataset.
- Step 2 – Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- Step 3 – In this step, voting will be performed for every predicted result.
- Step 4 – At last, select the most voted prediction result as the final prediction result.

```
[ ]  from sklearn.metrics import r2_score
     b= r2_score(y_test,y_pred_rf)
     b

     0.8694231518687003
```

Results

Algorithm	Train Size In %	Test Size In %	Result In %
Linear Regression	70	30	46.42
SGD	70	30	75.42
Random Forest	70	30	86.94

In this model, we have considered various restaurant records with characteristics such as name, average cost, place, whether online order is accepted, we can book a table, restaurant type.

This model would help business owners forecast their ranking on the criteria considered in our model and enhance the experience of the customer. Different algorithms have been used, but on Random Forest the final model is eventually chosen, which provides the highest precision compared to others.

Conclusion

The analyses of various characteristics of current restaurants in different areas of a city and analyses them to predict restaurant ratings. This makes it an important thing to take into consideration before making a dining decision. Before creating a venture like that of a restaurant, such research is an important part of planning. There has been a lot of research into variables impacting revenue and the competition in the restaurant industry. To enhance customer satisfaction rates, numerous dine-scape variables have been analysed. If data is also collected for other citrics, such predictions could be made for accuracy.

NoteBook link :

https://github.com/pranesh-badarinath/Zomato-Bangalore