# Zomato Restaurent Rating Prediction

## Introduction:

Zomato is a site where someone can give a review of a restaurant, how the restaurant is and someone's opinion about the restaurant. Restaurant customer satisfaction can be analyzed by their review on Zomato. Sometimes, restaurants see the reviews in Zomato, but they didn't get if the reviews are positive or negative to their restaurants. Review on Zomato is still in the form of text and can be classified with positive, negative, or neutral with their ratings. Zomato doesn't have an analysis of how users interact with the reviews and what words will indicate they like or not it. We need to extract the words in review and analysis it so we can know how users interact in Zomato and get customers satisfaction by their review.

## Objectives

- To predicting the rating of restaurants based on reviews given by the users.
- To determine is there any relationship between online booked and review score.
- To determine is there relationship between approximate cost and review score.
- To determine there is relationship between table booking and review score.
- To identify which location having the highest number of restaurants.
- To determine How many Restaurants are accepting online orders.
- To identify How many types of Restaurant are there.

## Scope of Study:

The solution to this rating prediction using reviews can be generalized to use as a model to give rating to restaurants in online platforms like zomato. And knowing online order facility has any effect on rating will help restaurants to consider online book facility for their future improvements. Similarly the result of test conducted to check the dependency of book table feature and rating score will help restaurants to improve their features.

## Performance metric:

The problem is Regression problem. The metric that we are using here is,

- **Mean squared error**

# Part 1:

# Data cleaning

In [ ]:

In [1]:
```
# Importing Libraries:
import numpy as np  #NumPy is a general-purpose array-processing package.
import pandas as pd  #It contains high-level data structures and manipulation tools
import matplotlib.pyplot as plt  #It is a Plotting Library.
```

```
import re
import seaborn as sns  #Seaborn is a Python data visualization library based on matp
from sklearn.linear_model import LinearRegression  #Linear Regression is a regressio
from sklearn.model_selection import train_test_split #Splitting of Dataset.

import warnings
warnings.filterwarnings("ignore")
```

# Read data

In [2]:
```
#Load the data
data=pd.read_csv("zomato.csv")
```

In [3]:
```
#Let us check first 5 data instances
data.head()
```

Out[3]:

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

In [4]:
```
#shape of the data
print("Number of data instaces that this dataset contains : ",data.shape[0])
print("Number of columns(including target variable) that this dataset contains : ",d
```

```
Number of data instaces that this dataset contains :  51717
Number of columns(including target variable) that this dataset contains :  17
```

In [5]:
```
# columns of dataset:
data.columns
```

Out[5]:
```
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')
```

- **url, address, phone these columns are not at all required for our analysis. Let us drop these columns at this point only.**

In [6]:
```
# drop unwanted columns:
data= data.drop(data.columns[[0,1,7]], axis=1)
data.head(2)
```

Out[6]:

| | name | online_order | book_table | rate | votes | location | rest_type | dish_liked | cuisines |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Jalsa | Yes | Yes | 4.1/5 | 775 | Banashankari | Casual Dining | Pasta, Lunch Buffet, Masala Papad, Paneer Laja... | North Indian, Mughlai, Chinese |
| 1 | Spice Elephant | Yes | No | 4.1/5 | 787 | Banashankari | Casual Dining | Momos, Lunch Buffet, Chocolate Nirvana, Thai G... | Chinese, North Indian, Thai |

In [7]:
```
# information about data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 14 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   name                         51717 non-null  object
 1   online_order                 51717 non-null  object
 2   book_table                   51717 non-null  object
 3   rate                         43942 non-null  object
 4   votes                        51717 non-null  int64 
 5   location                     51696 non-null  object
 6   rest_type                    51490 non-null  object
 7   dish_liked                   23639 non-null  object
 8   cuisines                     51672 non-null  object
 9   approx_cost(for two people)  51371 non-null  object
 10  reviews_list                 51717 non-null  object
 11  menu_item                    51717 non-null  object
 12  listed_in(type)              51717 non-null  object
 13  listed_in(city)              51717 non-null  object
dtypes: int64(1), object(13)
memory usage: 5.5+ MB
```

In [8]:
```
# data type:
data.dtypes
```

Out[8]:
```
name                           object
```

```
online_order                  object
book_table                    object
rate                          object
votes                          int64
location                      object
rest_type                     object
dish_liked                    object
cuisines                      object
approx_cost(for two people)   object
reviews_list                  object
menu_item                     object
listed_in(type)               object
listed_in(city)               object
dtype: object
```

# Data cleaning

In [9]:
```
# check unique value:
data['rate'].unique()
```

Out[9]:
```
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)
```

**Ratings are in object format. We need to clean this by removing "/5", And convert this to float values.**

In [10]:
```
def handlerate(value):
    """This function will remove "/5" from rating and converts value to float type
    if (value=='NEW' or value=='-'):
        return np.nan
    else:
        value=str(value).split('/')
        value= value[0]
        return float(value)

# convert ratings to required format
data['rate']=data['rate'].apply(lambda x: handlerate(x))
data['rate'].head()
```

Out[10]:
```
0    4.1
1    4.1
2    3.8
3    3.7
4    3.8
Name: rate, dtype: float64
```

In [11]:
```
# find null value
null_values=data.isnull().sum()
null_values
```

Out[11]:
```
name                        0
online_order                0
book_table                  0
rate                    10052
votes                       0
location                   21
rest_type                 227
```

```
dish_liked                    28078
cuisines                         45
approx_cost(for two people)     346
reviews_list                      0
menu_item                         0
listed_in(type)                   0
listed_in(city)                   0
dtype: int64
```

In [12]:
```
#calculating total null values
total_rows =data.shape[0]
total_null = null_values.sum()

# percent of data that is null:
print("Percentage of null value instanmces present are : ",round((total_null/total_r
```

```
Percentage of null value instanmces present are :  74.964 %
```

**In this data set we have nearly 75% null values. If we drop all null values, then we will be loosing 75% of the data.**

**So, we should not drop all null values. We can drop null values of some features which has less null values.**

In [13]:
```
# drop null values in perticular column which are less:
data.dropna(subset=["location","rest_type","approx_cost(for two people)","cuisines"]
data.reset_index(inplace=True)
```

- **After removing (Url, adress, Phone) columns, we have 14 features(including target variable).**
- **Also we removed (location, rest_type, approx_cost, cuisines) null values rows.**
- **we have 51148 observations and 14 columns.**

In [14]:
```
# Still we have null values, but we cannot drop these.
data.isnull().sum()
```

Out[14]:
```
index                         0
name                          0
online_order                  0
book_table                    0
rate                       9885
votes                         0
location                      0
rest_type                     0
dish_liked                27713
cuisines                      0
approx_cost(for two people)   0
reviews_list                  0
menu_item                     0
listed_in(type)               0
listed_in(city)               0
dtype: int64
```

**Rate and Dish_liked columns having lot of null values so fill null values by imputation method.**

In [15]:
```
# impute missing values in dish liked with "unknown", so that "unknown" also conside
data['dish_liked']=data['dish_liked'].fillna("unknown")
data['dish_liked']
```

Out[15]:
```
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
                              ...
51143                                              unknown
51144                                              unknown
51145                                              unknown
51146                          Cocktails, Pizza, Buttermilk
51147                                              unknown
Name: dish_liked, Length: 51148, dtype: object
```

In [16]:
```
# impute missing values with mean value
data['rate']=data['rate'].fillna(round(data['rate'].mean(),1))
data['rate'][:5]
```

Out[16]:
```
0    4.1
1    4.1
2    3.8
3    3.7
4    3.8
Name: rate, dtype: float64
```

In [ ]:

**Use the fillna() function to fill the null values in the dataset.**

**To fill the missing values, The possible ways are :**

**01)Filling the missing data with the mean or median value if it's a numerical variable.**

**02)Filling the missing data with mode or unknown if it's a categorical value.**

**in this case, dish_lied and menu_item is categorical variable, so we fill this variable as 'unknown'**

**and rate is numerical variable so we fill this variable with mean value.**

## approx_cost(for two people)

**removing "," from numbers and converting to float**

In [17]:
```
#removing "," from numbers and converting to float
data["approx_cost(for two people)"] = data["approx_cost(for two people)"].apply(lamb
data["approx_cost(for two people)"] = data["approx_cost(for two people)"].astype(flo
```

In [18]:
```
data["approx_cost(for two people)"]
```

Out[18]:
```
0         800.0
1         800.0
2         800.0
3         300.0
4         600.0
          ...
51143    1500.0
51144     600.0
51145    2000.0
51146    2500.0
51147    1500.0
Name: approx_cost(for two people), Length: 51148, dtype: float64
```

## online_order, book_table

In [19]:
```
#convrting to lower case
data["online_order"] = data["online_order"].apply(lambda x: x.lower())
data["book_table"] = data["book_table"].apply(lambda x: x.lower())
```

# Text data cleaning

In [20]:
```
# https://stackoverflow.com/a/47091490/4084039
def decontracted(phrase):
    """This function will converts shorthend form to full form of english words"""
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

## reviews text cleaning

In [21]:
```
data['reviews_list'].head(5)
```

Out[21]:
```
0    [('Rated 4.0', 'RATED\n  A beautiful place to ...
1    [('Rated 4.0', 'RATED\n  Had been here for din...
2    [('Rated 3.0', "RATED\n  Ambience is not that ...
3    [('Rated 4.0', "RATED\n  Great food and proper...
4    [('Rated 4.0', 'RATED\n  Very good restaurant ...
Name: reviews_list, dtype: object
```

In [22]:
```
# Combining all the above stundents
from tqdm import tqdm
def preprocess_text(text_data):
    """This function will clean the text data"""
    preprocessed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(text_data):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('NaN', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"',  ' ')
        sent = re.sub(r'[0-9]+',  ' ',sent)
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        sent = sent.replace('RATED',' ')
        sent = sent.replace('Rated',' ')
        sent= re.sub(' \w{1,2} ', ' ', sent)
        sent=' '.join([w for w in sent.split() if len(w)>=3])
        sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
        preprocessed_text.append(sent.lower().strip())
        # https://gist.github.com/sebleier/554280

    return preprocessed_text
```

In [ ]:
```
```

In [ ]:

In [ ]:

In [23]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not' because the
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that',
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', '
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'throu
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 't
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've",
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn'
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'm
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
            'won', "won't", 'wouldn', "wouldn't","NaN",]
```

In [24]:
```
data['reviews_list'] = preprocess_text(data['reviews_list'].values)
```

```
100%|████████████████████████████████████████████████████████████████████████| 51
148/51148 [07:40<00:00, 111.11it/s]
```

In [25]:
```
data['reviews_list'][0]
```

Out[25]:
```
'beautiful place dine interiors take back mughal era lightings perfect went occasion
christmas limited items available taste service not compromised complaint breads cou
ld better would surely like come dinner family weekday restaurant completely empty a
mbience good good old hindi music seating arrangement good ordered masala papad pann
er baby corn starters lemon corriander soup butter roti olive chilli paratha food fr
esh good service good good family hangout cheers restaurant near banashankari bda al
ong office friends visited buffet unfortunately provide veg buffet inquiring said pl
ace mostly visited vegetarians anyways ordered ala carte items took ages come food d
efinitely not visiting anymore went weekend one buffet two took ala carte firstly am
bience service place great buffet lot items good good pumpkin halwa intm dessert ama
zing must try kulchas great cheers best thing place ambiance second best thing yummy
food try buffet buffet food not disappointed test quality service staff professional
friendly overall experience excellent subirmajumder wixsite com great food pleasant
ambience expensive coll place chill relax service really good friendly staff food se
rvice ambience overall good ambience tasty food cheese chilli paratha bhutta palak m
ethi curry good combo lemon chicken starters must try item egg fried rice also quite
tasty mocktails recommend alice junoon not miss wrong jalsa never fan buffet thus al
ways order alacarte service times slower side food worth wait overdelighted service
food provided place royal ethnic atmosphere builds strong essence india also quality
taste food truly authentic would totally recommend visit place place nice comfortabl
e food wise jalea outlets maintain good standard soya chaap standout dish clearly on
e trademark dish per must try concern parking congested limited cars basement parkin
g steep makes cumbersome place nice comfortable food wise jalea outlets maintain goo
d standard soya chaap standout dish clearly one trademark dish per must try concern
parking congested limited cars basement parking steep makes cumbersome place nice co
mfortable food wise jalea outlets maintain good standard soya chaap standout dish cl
early one trademark dish per must try concern parking congested limited cars basemen
t parking steep makes cumbersome'
```

## cleaning dish_liked

In [26]:
```
def categorical_data(point):
    """Thgis function cleanse categorical text data"""

    lst_words = point.split(",")
```

```
    for i in range(len(lst_words)):
        splitted =lst_words[i].split()

        if len(splitted)>1:
            lst_words[i] = "_".join(splitted)


    point = "".join(lst_words).lower()
    return point
```

In [27]:
```
#before preprocessing
data['dish_liked'][5]
```

Out[27]: 'Onion Rings, Pasta, Kadhai Paneer, Salads, Salad, Roti, Jeera Rice'

In [28]:
```
data['dish_liked'] = data['dish_liked'].apply(lambda x: categorical_data(x))
```

In [29]:
```
data['dish_liked'][5]
```

Out[29]: 'onion_rings pastakadhai_paneer salads salad rotijeera_rice'

## cleaning cuisines

In [30]:
```
#before preprocessing
data['cuisines'][20]
```

Out[30]: 'Cafe, French, North Indian'

In [31]:
```
data['cuisines'] = data['cuisines'].apply(lambda x: categorical_data(x))
```

In [32]:
```
data['cuisines'][20]
```

Out[32]: 'cafe frenchnorth_indian'

## Cleaning rest_type

In [33]:
```
#before preprocessing
data['rest_type'][10]
```

Out[33]: 'Cafe'

In [34]:
```
data['rest_type'] = data['rest_type'].apply(lambda x: categorical_data(x))
```

In [35]:
```
data['rest_type'][10]
```

Out[35]: 'cafe'

## cleaning location

In [36]:
```
#before preprocessing
data['location'][60]
```

Out[36]: 'Banashankari'

In [37]:
```
data['location'] = data['location'].apply(lambda x: categorical_data(x))
```

In [38]:
```
#after preprocessing
data['location'][60]
```

Out[38]: 'banashankari'

In [ ]:
```

```

## Cleaning listed_in(type)

In [39]:
```
data['listed_in(type)'][0]
```

Out[39]: 'Buffet'

In [40]:
```
data['listed_in(type)'] = data['listed_in(type)'].apply(lambda x: categorical_data(x
```

In [41]:
```
data['listed_in(type)'][0]
```

Out[41]: 'buffet'

## Cleaning listed_in(city)

In [42]:
```
data['listed_in(city)'][0]
```

Out[42]: 'Banashankari'

In [43]:
```
data['listed_in(city)'] = data['listed_in(city)'].apply(lambda x: categorical_data(x
```

In [44]:
```
data['listed_in(city)'][0]
```

Out[44]: 'banashankari'

## Cleaning name

In [45]:
```
data['name'][0]
```

Out[45]: 'Jalsa'

In [46]:
```
data['name'] = preprocess_text(data['name'].values)
```

```
100%|████████████████████████████████████████████████████████████████████| 5114
8/51148 [00:02<00:00, 18269.66it/s]
```

In [47]:
```
data['name'][0]
```

Out[47]: 'jalsa'

## Cleaning menu_item

In [48]:
```
data['menu_item']
```

Out[48]:
```
0        []
1        []
2        []
3        []
4        []
         ..
51143    []
51144    []
```

```
51145    []
51146    []
51147    []
Name: menu_item, Length: 51148, dtype: object
```

In [49]:
```
#data['menu_item']=data['menu_item'].str.lower()
(data['menu_item'][1000])
```

Out[49]:
```
"['Combo 2', 'Combo 3', 'Combo 1', 'Combo 4', 'Crispy Chilly Baby Corn', 'American C
orn And Salt And Pepper', 'Chilly Babycorn', 'Mushroom Pepper', 'Pepper Chicken', 'C
hilli Chicken', 'Salt Pepper Fish', 'Fried Wontons-veg', 'Crispy Chilli Potato', 'Sz
echuan Paneer', 'Gobi Manchurian', 'Spring Rolls-non Veg', 'Thai Red Wings', 'Thai B
asil Chicken', 'Drums Of Heaven', 'Dragon Chicken', 'Sweet And Spicy Crispy Chicke
n', 'Lemon Basil Chicken', 'Thai Fried Chicken', 'Fried Chicken Wings', 'Chicken Lol
lypop', 'Chicken Meat Ball', 'Szechuan Chicken', 'Pan Fry Chilly Fish', 'Szechuan Pr
awn', 'Crispy Fish In Sweet Chilli Sauce And Basil', 'Chinese Veal', 'Spicy Lemon Co
rriander Soup(veg)', 'Lemon Pepper Soup(non Veg)', 'Spicy Lemon Corriander Soup(non
Veg)', 'Clear Soup(veg)', 'Noodles Soup(veg)', 'Lemon Pepper Soup(veg)', 'Sweetcorn
Soup(veg)', 'Manchow Soup(veg)', 'Hot & Sour Soup(veg)', 'Talumein Soup(veg)', 'Tom
Yam Soup(veg)', 'Chicken Mushroom Soup(veg)', 'Tom Kha Soup(veg)', 'Clear Soup(non V
eg)', 'Noodles Soup(non Veg)', 'Sweetcorn Soup(non Veg)', 'Manchow Soup(non Veg)',
'Hot & Sour Soup(non Veg)', 'Talumein Soup(non Veg)', 'Tom Yam Soup(non Veg)', 'Chic
ken Mushroom Soup(non Veg)', 'Tom Kha Soup(non Veg)', 'Chilli Garlic Veg Fried Ric
e', 'Chilli Garlic Egg Fried Rice', 'Chilli Garlic Prawns Fried Rice', 'Chilli Garli
c Mix Fried Rice', 'Veg Fried Rice', 'Mushroom Fried Rice', 'Szechuan Veg Fried Ric
e', 'Ginger Capscicum Fried Rice', 'Singapore Veg Fried Rice', 'Thai Basil Rice', 'N
asi Goreng Rice', 'Egg Fried Rice', 'Szechuan Egg Fried Rice', 'Ginger Capscicum Egg
fried Rice', 'Singapore Egg Fried Rice', 'Thai Basil Egg Rice', 'Nasi Goreng Egg Ric
e', 'Chicken Fried Rice', 'Szechuan Chicken Fried Rice', 'Ginger Capscicum Chicken F
ried Rice', 'Chilli Garlic Chicken Fried Rice', 'Singapore Chicken Fried Rice', 'Tha
i Basil Chicken Rice', 'Nasi Goreng Chicken Rice', 'Prawns Fried Rice', 'Szechuan Pr
awn Fried Rice', 'Ginger Capscicum Prawns Fried Rice', 'Singapore Prawns Fried Ric
e', 'Thai Basil Prawns Rice', 'Nasi Goreng Prawns Rice', 'Mix Fried Rice', 'Szechuan
Mix Fried Rice', 'Ginger Capscicum Mix Fried Rice', 'Singapore Mix Fried Rice', 'Tha
i Basil Mix Rice', 'Nasi Goreng Mix Rice', 'Chilli Garlic Veg Noodles', 'Chilli Garl
ic Egg Noodles', 'Chilli Garlic Prawns Noodles', 'Chilli Garlic Mix Noodles', 'Veg N
oodles', 'Mushroom Noodles', 'Szechuan Veg Noodles', 'Ginger Capsicum Veg Noodles',
'Singapore Veg Noodles', 'Malaysian Noodles-veg', 'Thai Basil Noodles', 'Nasi Gereng
Veg Noodles', 'Egg Noodles', 'Szechuan Egg Noodles', 'Ginger Capscicum Egg Noodles',
'Singapore Egg Noodles', 'Malaysian Noodles-egg', 'Thai Basil Egg Noodles', 'Nasi Ge
reng Egg Noodles', 'Chicken Noodles', 'Szechuan Chicken Noodles', 'Ginger Capscicum
Chicken Noodles', 'Chilli Garlic Chicken Noodles', 'Singapore Chicken Noodles', 'Mal
aysian Noodles-chicken', 'Thai Basil Chicken Noodles', 'Nasi Gereng Chicken Noodle
s', 'Prawns Noodles', 'Szechuan Prawns Noodles', 'Ginger Capscicum Prawns Noodles',
'Singapore Prawns Noodles', 'Malaysian Noodles-prawn', 'Thai Basil Prawns Noodles',
'Nasi Gereng Prawns Noodles', 'Mix Noodles', 'Szechuan Mix Noodles', 'Ginger Capscic
um Mix Noodles', 'Singapore Mix Noodles', 'Malaysian Noodles-mix', 'Thai Basil Mix N
oodles', 'Nasi Gereng Mix Noodles', 'Paneer Manchurian', 'Veg Manchurian', 'Szechuan
Vegetables', 'Babycorn Manchurian', 'Thai Red Curry-veg', 'Thai Red Curry-non Veg',
'Thai Green Curry-veg', 'Thai Green Curry-non Veg', 'Garlic Chicken', 'Chilli Chicke
n Gravy', 'Chicken Manchurian', 'Hunan Chicken', 'Szechuan Fish', 'Hunan Fish', 'Gar
lic Fish', 'Garlic Prawn', 'Paneer Steak Sizzler', 'Manchurian Sizzler', 'Paneer/mus
hroom Sizzler', 'Veg Mangolian Sizzler', 'Chicken Sizzler', 'Fish Sizzler', 'Prawn S
izzler', 'Veg Chinese Chopsuey', 'Chicken Chinese Chopsuey', 'Prawn Chinese Chopsue
y', 'Mixed Chinese Chosuey', 'Veg American Chopsuey', 'Chicken American Chopsuey',
'Prawn American Chopsuey', 'Mixed American Chopsuey', 'Veg Combo', 'Soup Combo Ric
e', 'Fujian Combo Veg', 'Chicken Combo']"
```

In [50]:
```
def cleaning(text):
    """This function will clean the menu item text"""
    #removing ] and [
    text = text.replace("[","").replace("]","")
    #removing numbers
    text = re.sub("[0-9]+","",text)
    #replacing '
    text = text.replace("'","")
    #replacing commas
    text = text.split(",")
```

```
    # _ joining single menu item
    for i in range(len(text)):
        words = text[i].split()
        if len(words)>1:
            text[i] = "_".join(words)

    #joining and lower case
    text = " ".join(text).lower()

    return text
```

In [51]:
```
data['menu_item'] = data['menu_item'].apply(lambda x: cleaning(x))
```

In [52]:
```
#let us fill empty values with 'unknown'
for i in range(len(data['menu_item'])):
    if data['menu_item'][i] == "":
        data['menu_item'][i] = "unknown"
```

In [53]:
```
data["reviews_list"].fillna("no reviews", inplace = True)
```

**We have done all cleaning part. Now data is ready for further analysis and featurizations.**

In [54]:
```
#let us see the cleaned data
data.head(4)
```

Out[54]:

| | index | name | online_order | book_table | rate | votes | location | rest_type | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | jalsa | yes | yes | 4.1 | 775 | banashankari | casual_dining | pastalu |
| 1 | 1 | spice elephant | yes | no | 4.1 | 787 | banashankari | casual_dining | momo |
| 2 | 2 | san churro cafe | yes | no | 3.8 | 918 | banashankari | cafecasual_dining | churr |
| 3 | 3 | addhuri udupi bhojana | no | no | 3.7 | 88 | banashankari | quick_bites | |

In [57]:
```
#Now we are saving the cleaned data, so that we don't need to re run the above cells
# We can use these cleaned saved file for further analysis.
data.to_csv("cleaned_data.csv")
```

**Let us continue the further analysis in the next part.**

In [ ]:

In [ ]: