⌄ *1. Know Your Data*

⌄ Import Libraries

```
!pip uninstall -y numpy scipy gensim
```

```
⤓   Found existing installation: numpy 1.26.4
    Uninstalling numpy-1.26.4:
      Successfully uninstalled numpy-1.26.4
    Found existing installation: scipy 1.13.1
    Uninstalling scipy-1.13.1:
      Successfully uninstalled scipy-1.13.1
    Found existing installation: gensim 4.3.3
    Uninstalling gensim-4.3.3:
      Successfully uninstalled gensim-4.3.3
```

```
!pip install numpy==1.26.4 scipy==1.13.1 gensim==4.3.3
```

```
⤓   Collecting numpy==1.26.4
      Using cached numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
    Collecting scipy==1.13.1
      Using cached scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
    Collecting gensim==4.3.3
      Using cached gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (8.1 kB)
    Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.11/dist-packages (from gensim==4.3.3) (7.1.0)
    Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open>=1.8.1->gensim==4.3.3)
    Using cached numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
    Using cached scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (38.6 MB)
    Using cached gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26.7 MB)
    Installing collected packages: numpy, scipy, gensim
    Successfully installed gensim-4.3.3 numpy-1.26.4 scipy-1.13.1
```

```python
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import matplotlib.cm as cm
import seaborn as sns
import math
import time
from wordcloud import WordCloud

from scipy.stats import norm
from scipy import stats
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_score,recall_score,f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler, StandardScaler

#importing kmeans
from sklearn.cluster import KMeans

#importing random forest and XgB
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

#Non-negative matrix Factorization
from sklearn.decomposition import NMF


from sklearn.naive_bayes import MultinomialNB

#principal component analysis
from sklearn.decomposition import PCA


#silhouette score
from sklearn.metrics import silhouette_score
from sklearn.model_selection import ParameterGrid

#importing stopwords
import nltk
nltk.download('stopwords')
```

```python
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.corpus import stopwords
#for tokenization
from nltk.tokenize import word_tokenize
# for POS tagging(Part of speech in NLP sentiment analysis)
nltk.download('averaged_perceptron_tagger')

#import stemmer
from nltk.stem.snowball import SnowballStemmer

#import tfidf
from sklearn.feature_extraction.text import TfidfVectorizer

#LDA


from sklearn.decomposition import LatentDirichletAllocation


#importing contraction
!pip install contractions
!pip install gensim
import gensim
from gensim import corpora

#importing shap for model explainability
!pip install shap
import shap

#download small spacy model
# !python -m spacy download en_core_web_sm
# import spacy

# The following lines adjust the granularity of reporting.
pd.options.display.float_format = "{:.2f}".format

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]    Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]    Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]    Unzipping taggers/averaged_perceptron_tagger.zip.
Collecting contractions
  Downloading contractions-0.1.73-py2.py3-none-any.whl.metadata (1.2 kB)
Collecting textsearch>=0.0.21 (from contractions)
  Downloading textsearch-0.0.24-py2.py3-none-any.whl.metadata (1.2 kB)
Collecting anyascii (from textsearch>=0.0.21->contractions)
  Downloading anyascii-0.3.2-py3-none-any.whl.metadata (1.5 kB)
Collecting pyahocorasick (from textsearch>=0.0.21->contractions)
  Downloading pyahocorasick-2.1.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Downloading contractions-0.1.73-py2.py3-none-any.whl (8.7 kB)
Downloading textsearch-0.0.24-py2.py3-none-any.whl (7.6 kB)
Downloading anyascii-0.3.2-py3-none-any.whl (289 kB)
                                   289.9/289.9 kB 14.3 MB/s eta 0:00:00
Downloading pyahocorasick-2.1.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (118 kB)
                                   118.3/118.3 kB 10.5 MB/s eta 0:00:00
Installing collected packages: pyahocorasick, anyascii, textsearch, contractions
Successfully installed anyascii-0.3.2 contractions-0.1.73 pyahocorasick-2.1.0 textsearch-0.0.24
Requirement already satisfied: gensim in /usr/local/lib/python3.11/dist-packages (4.3.3)
Requirement already satisfied: numpy<2.0,>=1.18.5 in /usr/local/lib/python3.11/dist-packages (from gensim) (1.26.4)
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in /usr/local/lib/python3.11/dist-packages (from gensim) (1.13.1)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.11/dist-packages (from gensim) (7.1.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open>=1.8.1->gensim) (1.17.2
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages (0.47.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from shap) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from shap) (1.13.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from shap) (1.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.11/dist-packages (from shap) (4.67.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.11/dist-packages (from shap) (24.2)
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.11/dist-packages (from shap) (0.0.8)
Requirement already satisfied: numba>=0.54 in /usr/local/lib/python3.11/dist-packages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.11/dist-packages (from shap) (3.1.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from shap) (4.13.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba>=0.54->
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2025.2)
```

```
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->shap) (1.4.2
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->shap)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas-
```

Dataset Loading

```python
# Load Dataset
hotel_df = pd.read_csv('/content/drive/MyDrive/Zomato data/Zomato Restaurant names and Metadata.csv')
review_df = pd.read_csv('/content/drive/MyDrive/Zomato data/Zomato Restaurant reviews.csv')
```

Dataset First View

```python
# Dataset First Look restaurant
hotel_df.head()
```

	Name	Links	Cost	Collections	Cuisines	Timings
0	Beyond Flavours	https://www.zomato.com/hyderabad/beyond-flavou...	800	Food Hygiene Rated Restaurants in Hyderabad, C...	Chinese, Continental, Kebab, European, South I...	12noon to 3:30pm, 6:30pm to 11:30pm (Mon-Sun)
1	Paradise	https://www.zomato.com/hyderabad/paradise-gach...	800	Hyderabad's Hottest	Biryani, North Indian, Chinese	11 AM to 11 PM
2	Flechazo	https://www.zomato.com/hyderabad/flechazo-gach...	1,300	Great Buffets, Hyderabad's Hottest	Asian, Mediterranean, North Indian, Desserts	11:30 AM to 4:30 PM, 6:30 PM to 11 PM
3	Shah Ghouse Hotel &	https://www.zomato.com/hyderabad/shah-ghouse-h	800	Late Night Restaurants	Biryani, North Indian, Chinese, Seafood,	12 Noon to 2 AM

Next steps: [Generate code with `hotel_df`] [👁 View recommended plots] [New interactive sheet]

```python
# Dataset First Look review
review_df.head()
```

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures
0	Beyond Flavours	Rusha Chakraborty	The ambience was good, food was quite good . h...	5	1 Review , 2 Followers	5/25/2019 15:54	0
1	Beyond Flavours	Anusha Tirumalaneedi	Ambience is too good for a pleasant evening. S...	5	3 Reviews , 2 Followers	5/25/2019 14:20	0
2	Beyond Flavours	Ashok Shekhawat	A must try.. great food great ambience. Thnx f...	5	2 Reviews , 3 Followers	5/24/2019 22:54	0

Next steps: [Generate code with `review_df`] [👁 View recommended plots] [New interactive sheet]

Dataset Rows & Columns count

```python
# Dataset Rows(Observation) & Columns count(Feature)
print(f'Total observation and feature for restaurant: {hotel_df.shape}')
print(f'Total observation and feature for review: {review_df.shape}')
```

```
Total observation and feature for restaurant: (105, 6)
Total observation and feature for review: (10000, 7)
```

Dataset Information

```python
# Dataset Info
print('Restaurant Info')
print('\n')
hotel_df.info()
print('='*120)
print('\n')
print('Review Info')
print('\n')
review_df.info()
```

```
Restaurant Info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105 entries, 0 to 104
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         105 non-null    object
 1   Links        105 non-null    object
 2   Cost         105 non-null    object
 3   Collections  51 non-null     object
 4   Cuisines     105 non-null    object
 5   Timings      104 non-null    object
dtypes: object(6)
memory usage: 5.1+ KB
========================================================================================================
```

Review Info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Restaurant  10000 non-null  object
 1   Reviewer    9962 non-null   object
 2   Review      9955 non-null   object
 3   Rating      9962 non-null   object
 4   Metadata    9962 non-null   object
 5   Time        9962 non-null   object
 6   Pictures    10000 non-null  int64
dtypes: int64(1), object(6)
memory usage: 547.0+ KB
```

⌄ Duplicate Values

```
# Dataset Duplicate Value Count
print('For Restaurant')
print('\n')
print(f"Data is duplicated ? {hotel_df.duplicated().value_counts()},unique values with {len(hotel_df[hotel_df.duplicated()])}
print('\n')
print('='*120)
print('\n')
print('For Reviews')
print('\n')
print(f"Data is duplicated ? {review_df.duplicated().value_counts()},unique values with {len(review_df[review_df.duplicated(
```

⇄ For Restaurant

```
    Data is duplicated ? False    105
    Name: count, dtype: int64,unique values with 0 duplication


    ========================================================================================================


    For Reviews


    Data is duplicated ? False    9964
    True          36
    Name: count, dtype: int64,unique values with 36 duplication
```

```
#getting duplicate values
print(f' Duplicate data count = {review_df[review_df.duplicated()].shape[0]}')
review_df[review_df.duplicated()]
```

Duplicate data count = 36

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures
8778	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8779	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8780	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8781	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8782	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8783	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8784	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8785	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8786	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8787	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8788	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8789	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8790	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8791	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8792	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8793	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8794	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8795	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8796	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8797	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8798	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
8799	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
9086	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9087	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9088	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9089	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9090	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9091	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9092	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9093	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9094	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9095	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9096	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9097	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0
9098	Arena Eleven	NaN	NaN	NaN	NaN	NaN	0

```python
#checking values for Anerican Wild Things
review_df[(review_df['Restaurant'] == 'American Wild Wings')].shape
```

(100, 7)

```python
#checking values for Arena Eleven
review_df[(review_df['Restaurant'] == 'Arena Eleven')].shape
```

(100, 7)

Missing Values/Null Values

```python
# Missing Values/Null Values Count for hotel data
hotel_df.isnull().sum()
```

	0
Name	0
Links	0
Cost	0
Collections	54
Cuisines	0
Timings	1

```python
# Missing Values/Null Values Count for review data
review_df.isnull().sum()
```

	0
Restaurant	0
Reviewer	38
Review	45
Rating	38
Metadata	38
Time	38
Pictures	0

```python
# Visualizing the missing values for restaurant
# Checking Null Value by plotting Heatmap
sns.heatmap(hotel_df.isnull(), cbar=False);
```
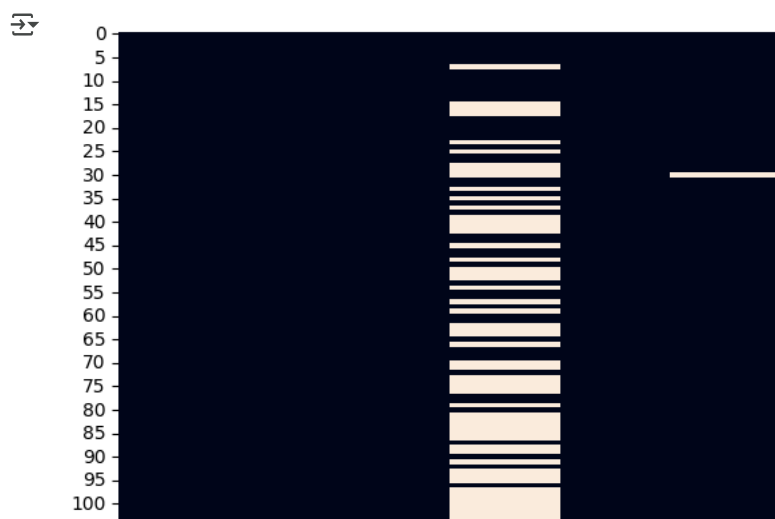


```python
# Visualizing the missing values for reviews
# Checking Null Value by plotting Heatmap
sns.heatmap(review_df.isnull(), cbar=False);
```

What did you know about your dataset?

Restaurant DataSet

- There are 105 total observation with 6 different features.
- Feature like collection and timing has null values.
- There is no duplicate values i.e., 105 unique data.
- Feature cost represent amount but has object data type because these values are separated by comma ','.
- Timing represent operational hour but as it is represented in the form of text has object data type.

Review DataSet

- There are total 10000 observation and 7 features.
- Except picture and restaurant feature all others have null values.
- There are total of 36 duplicate values for two restaurant - American Wild Wings and Arena Eleven, where all these duplicate values generally have null values.
- Rating represent ordinal data, has object data type should be integer.
- Timing represent the time when review was posted but show object data time, it should be converted into date time.

2. Understanding Your Variables

```
# Dataset Columns restaurant
print(f'Features : {hotel_df.columns.to_list()}')
```

```
Features : ['Name', 'Links', 'Cost', 'Collections', 'Cuisines', 'Timings']
```

```
# Dataset Columns review
print(f'Features : {review_df.columns.to_list()}')
```

```
Features : ['Restaurant', 'Reviewer', 'Review', 'Rating', 'Metadata', 'Time', 'Pictures']
```

```
# Dataset Describe restaurant
hotel_df.describe().T
```

	count	unique	top	freq
Name	105	105	Beyond Flavours	1
Links	105	105	https://www.zomato.com/hyderabad/beyond-flavou...	1
Cost	105	29	500	13
Collections	51	42	Food Hygiene Rated Restaurants in Hyderabad	4
Cuisines	105	92	North Indian, Chinese	4

```
# Dataset Describe review
review_df.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Restaurant	10000	100	Beyond Flavours	100	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Reviewer	9962	7446	Parijat Ray	13	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Review	9955	9364	good	237	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rating	9962	10	5	3832	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Metadata	9962	2477	1 Review	919	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Time	9962	9782	7/29/2018 20:34	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Variables Description

Attributes ▶

Zomato Restaurant

- Name : Name of Restaurants
- Links : URL Links of Restaurants
- Cost : Per person estimated Cost of dining
- Collection : Tagging of Restaurants w.r.t. Zomato categories
- Cuisines : Cuisines served by Restaurants
- Timings : Restaurant Timings

Zomato Restaurant Reviews

- Restaurant : Name of the Restaurant
- Reviewer : Name of the Reviewer
- Review : Review Text
- Rating : Rating Provided by Reviewer
- MetaData : Reviewer Metadata - No. of Reviews and followers
- Time: Date and Time of Review
- Pictures : No. of pictures posted with review

Check Unique Values for each variable.

```
# Check Unique Values for each variable for restaurant
for i in hotel_df.columns.tolist():
  print("No. of unique values in ",i,"is",hotel_df[i].nunique(),".")
```

```
No. of unique values in  Name is 105 .
No. of unique values in  Links is 105 .
No. of unique values in  Cost is 29 .
No. of unique values in  Collections is 42 .
No. of unique values in  Cuisines is 92 .
No. of unique values in  Timings is 77 .
```

```
# Check Unique Values for each variable for reviews
for i in review_df.columns.tolist():
  print("No. of unique values in ",i,"is",review_df[i].nunique(),".")
```

```
No. of unique values in  Restaurant is 100 .
No. of unique values in  Reviewer is 7446 .
No. of unique values in  Review is 9364 .
No. of unique values in  Rating is 10 .
No. of unique values in  Metadata is 2477 .
No. of unique values in  Time is 9782 .
No. of unique values in  Pictures is 36 .
```

3. *Data Wrangling*

Data Wrangling Code

```
#creating copy of both the data
hotel = hotel_df.copy()
review = review_df.copy()
```

⌄ Restaurant

```
#before changing data type for cost checking values
hotel['Cost'].unique()
```

```
array(['800', '1,300', '1,200', '1,500', '500', '300', '1,000', '350',
       '400', '1,600', '750', '550', '1,900', '450', '150', '1,400',
       '1,100', '600', '200', '900', '700', '1,700', '2,500', '850',
       '650', '1,800', '2,800', '1,750', '250'], dtype=object)
```

```
# Write your code to make your dataset analysis ready.
# changing the data type of the cost function
hotel['Cost'] = hotel['Cost'].str.replace(",","").astype('int64')
```

```
#top 5 costlier restaurant
hotel.sort_values('Cost', ascending = False)[['Name','Cost']][:5]
```

	Name	Cost
92	Collage - Hyatt Hyderabad Gachibowli	2800
56	Feast - Sheraton Hyderabad Hotel	2500
21	Jonathan's Kitchen - Holiday Inn Express & Suites	1900
18	10 Downing Street	1900

```
#top 5 economy restaurant
hotel.sort_values('Cost', ascending = False)[['Name','Cost']][-5:]
```

	Name	Cost
85	Momos Delight	200
29	Hunger Maggi Point	200
101	Sweet Basket	200
89	Mohammedia Shawarma	150

```
#hotels that share same price
hotel_dict = {}
amount = hotel.Cost.values.tolist()

#adding hotel name based on the price by converting it into list
for price in amount:
    # Get all the rows that have the current price
    rows = hotel[hotel['Cost'] == price]
    hotel_dict[price] = rows["Name"].tolist()

#converting it into dataframe
same_price_hotel_df=pd.DataFrame.from_dict([hotel_dict]).transpose().reset_index().rename(
    columns={'index':'Cost',0:'Name of Restaurants'})

#alternate methode to do the same
#same_price_hotel_df = hotel.groupby('Cost')['Name'].apply(lambda x: x.tolist()).reset_index()

#getting hotel count
hotel_count = hotel.groupby('Cost')['Name'].count().reset_index().sort_values(
    'Cost', ascending = False)

#merging together
same_price_hotel_df = same_price_hotel_df.merge(hotel_count, how = 'inner',
                      on = 'Cost').rename(columns = {'Name':'Total_Restaurant'})

#max hotels that share same price
same_price_hotel_df.sort_values('Total_Restaurant', ascending = False)[:5]
```

	Cost	Name of Restaurants	Total_Restaurant	
4	500	[eat.fit, KFC, Kritunga Restaurant, Karachi Ba...	13	
17	600	[Behrouz Biryani, Karachi Cafe, Hyderabad Chef...	10	
20	700	[Marsala Food Company, Green Bawarchi Restaura...	8	
2	1200	[Over The Moon Brew Company, The Glass Onion, ...	7	

```
#hotels which has max price
same_price_hotel_df.sort_values('Cost', ascending = False)[:5]
```

	Cost	Name of Restaurants	Total_Restaurant	
26	2800	[Collage - Hyatt Hyderabad Gachibowli]	1	
22	2500	[Feast - Sheraton Hyderabad Hotel]	1	
12	1900	[10 Downing Street, Jonathan's Kitchen - Holid...	2	
25	1800	[Cascade - Radisson Hyderabad Hitec City]	1	

```
# spliting the cusines and storing in list
cuisine_value_list = hotel.Cuisines.str.split(', ')


# storing all the cusines in a dict
cuisine_dict = {}
for cuisine_names in cuisine_value_list:
    for cuisine in cuisine_names:
        if (cuisine in cuisine_dict):
            cuisine_dict[cuisine]+=1
        else:
            cuisine_dict[cuisine]=1


# converting the dict to a data frame
cuisine_df=pd.DataFrame.from_dict([cuisine_dict]).transpose().reset_index().rename(
    columns={'index':'Cuisine',0:'Number of Restaurants'})


#top 5 cuisine
cuisine_df.sort_values('Number of Restaurants', ascending =False)[:5]
```

	Cuisine	Number of Restaurants	
5	North Indian	61	
0	Chinese	43	
1	Continental	21	
6	Biryani	16	

```
# spliting the cusines and storing in list
Collections_value_list = hotel.Collections.dropna().str.split(', ')


# storing all the cusines in a dict
Collections_dict = {}
for collection in Collections_value_list:
    for col_name in collection:
        if (col_name in Collections_dict):
            Collections_dict[col_name]+=1
        else:
            Collections_dict[col_name]=1


# converting the dict to a data frame
Collections_df=pd.DataFrame.from_dict([Collections_dict]).transpose().reset_index().rename(
    columns={'index':'Tags',0:'Number of Restaurants'})


#top 5 collection
Collections_df.sort_values('Number of Restaurants', ascending =False)[:5]
```

	Tags	Number of Restaurants	
2	Great Buffets	11	
0	Food Hygiene Rated Restaurants in Hyderabad	8	
5	Live Sports Screenings	7	
6	Hyderabad's Hottest	7	

⌄ Reviews

```
#in order to change data type for rating checking values
review.Rating.value_counts()
```

	count
Rating	
5	3832
4	2373
1	1735
3	1193
2	684
4.5	69
3.5	47
2.5	19
1.5	9
Like	1

```
#changing data type for each rating since had value as interger surrounded by inverted comma
#since there is one rating as like converting it to 0 since no rating is 0 then to median
review.loc[review['Rating'] == 'Like'] = 0
#changing data type for rating in review data
review['Rating'] = review['Rating'].astype('float')


#since there is one rating as like converting it to median
review.loc[review['Rating'] == 0] = review.Rating.median()
```

```
review
```

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures	
0	Beyond Flavours	Rusha Chakraborty	The ambience was good, food was quite good . h...	5.00	1 Review , 2 Followers	5/25/2019 15:54	0	
1	Beyond Flavours	Anusha Tirumalaneedi	Ambience is too good for a pleasant evening. S...	5.00	3 Reviews , 2 Followers	5/25/2019 14:20	0	
2	Beyond Flavours	Ashok Shekhawat	A must try.. great food great ambience. Thnx f...	5.00	2 Reviews , 3 Followers	5/24/2019 22:54	0	
3	Beyond Flavours	Swapnil Sarkar	Soumen das and Arun was a great guy. Only beca...	5.00	1 Review , 1 Follower	5/24/2019 22:11	0	
4	Beyond Flavours	Dileep	Food is good.we ordered Kodi drumsticks and ba...	5.00	3 Reviews , 2 Followers	5/24/2019 21:37	0	
...	...	...	...	...	...	...	...	
9995	Chinese Pavilion	Abhishek Mahajan	Madhumathi Mahajan Well to start with nice cou...	3.00	53 Reviews , 54 Followers	6/5/2016 0:08	0	
9996	Chinese Pavilion	Sharad Agrawal	This place has never disappointed us.. The foo...	4.50	2 Reviews , 53 Followers	6/4/2016 22:01	0	

Next steps: [Generate code with `review`]   [👁 View recommended plots]   [New interactive sheet]

⌄ changing date and extracting few feature for manipulation

review

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures
0	Beyond Flavours	Rusha Chakraborty	The ambience was good, food was quite good . h...	5.00	1 Review , 2 Followers	5/25/2019 15:54	0
1	Beyond Flavours	Anusha Tirumalaneedi	Ambience is too good for a pleasant evening. S...	5.00	3 Reviews , 2 Followers	5/25/2019 14:20	0
2	Beyond Flavours	Ashok Shekhawat	A must try.. great food great ambience. Thnx f...	5.00	2 Reviews , 3 Followers	5/24/2019 22:54	0
3	Beyond Flavours	Swapnil Sarkar	Soumen das and Arun was a great guy. Only beca...	5.00	1 Review , 1 Follower	5/24/2019 22:11	0
4	Beyond Flavours	Dileep	Food is good.we ordered Kodi drumsticks and ba...	5.00	3 Reviews , 2 Followers	5/24/2019 21:37	0
...	...	...	...	...	...	...	...
9995	Chinese Pavilion	Abhishek Mahajan	Madhumathi Mahajan Well to start with nice cou...	3.00	53 Reviews , 54 Followers	6/5/2016 0:08	0
9996	Chinese Pavilion	Sharad Agrawal	This place has never disappointed us.. The foo...	4.50	2 Reviews , 53 Followers	6/4/2016 22:01	0

Next steps: [Generate code with `review`]   [View recommended plots]   [New interactive sheet]

```
review['Reviewer_Total_Review']=review['Metadata'].str.split(',',expand=True)[0]
review['Reviewer_Followers']=review['Metadata'].str.split(',', expand=True)[1]

review['Reviewer_Total_Review'] = pd.to_numeric(review['Reviewer_Total_Review'].str.split(' ',expand=True)[0])
review['Reviewer_Followers'] = pd.to_numeric(review['Reviewer_Followers'].str.split(' ').str[1])
```

review

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures	Reviewer_Total_Review	Reviewer_Followers
0	Beyond Flavours	Rusha Chakraborty	The ambience was good, food was quite good . h...	5.00	1 Review , 2 Followers	5/25/2019 15:54	0	1.00	2.00
1	Beyond Flavours	Anusha Tirumalaneedi	Ambience is too good for a pleasant evening. S...	5.00	3 Reviews , 2 Followers	5/25/2019 14:20	0	3.00	2.00
2	Beyond Flavours	Ashok Shekhawat	A must try.. great food great ambience. Thnx f...	5.00	2 Reviews , 3 Followers	5/24/2019 22:54	0	2.00	3.00
3	Beyond Flavours	Swapnil Sarkar	Soumen das and Arun was a great guy. Only beca...	5.00	1 Review , 1 Follower	5/24/2019 22:11	0	1.00	1.00
4	Beyond Flavours	Dileep	Food is good.we ordered Kodi drumsticks and ba...	5.00	3 Reviews , 2 Followers	5/24/2019 21:37	0	3.00	2.00
...	...	...	...	...	...	...	...	...	...

Next steps: [Generate code with `review`]   [View recommended plots]   [New interactive sheet]

```
review['Time']=pd.to_datetime(review['Time'],errors='coerce')
review['Review_Year'] = pd.DatetimeIndex(review['Time']).year
review['Review_Month'] = pd.DatetimeIndex(review['Time']).month
review['Review_Hour'] = pd.DatetimeIndex(review['Time']).hour


#Average engagement of restaurants
avg_hotel_rating = review.groupby('Restaurant').agg({'Rating':'mean',
```

```
     'Reviewer': 'count'}).reset_index().rename(columns = {'Reviewer': 'Total_Review'})
avg_hotel_rating
```

	Restaurant	Rating	Total_Review
0		4.00	4.00	1
1	10 Downing Street	3.80	100
2	13 Dhaba	3.48	100
3	3B's - Buddies, Bar & Barbecue	4.76	100
4	AB's - Absolute Barbecues	4.88	100
...	...	...	...
96	Urban Asia - Kitchen & Bar	3.65	100
97	Yum Yum Tree - The Arabian Food Court	3.56	100
98	Zega - Sheraton Hyderabad Hotel	4.45	100
99	Zing's Northeast Kitchen	3.65	100
100	eat.fit	3.20	100

Next steps: [Generate code with `avg_hotel_rating`] [◯ View recommended plots] [New interactive sheet]

```
#usless data
review[review['Restaurant'] == 4.0]
```

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures	Reviewer_Total_Review	Reviewer_Followers	Review_Yea
7601	4.00	4.00	4.00	4.00	4.00	NaT	4	NaN	NaN	Na

```
#checking hotel count as total hotel in restaurant data was 105
review.Restaurant.nunique()
```

```
101
```

```
#finding hotel without review
hotel_without_review = [name for name in hotel.Name.unique().tolist()
        if name not in review.Restaurant.unique().tolist()]
hotel_without_review
```

```
['IndiBlaze',
 'Sweet Basket',
 'Angaara Counts 3',
 'Wich Please',
 'Republic Of Noodles – Lemon Tree Hotel']
```

```
#top 5 most engaging or rated restaurant
avg_hotel_rating.sort_values('Rating', ascending = False)[:5]
```

	Restaurant	Rating	Total_Review
4	AB's - Absolute Barbecues	4.88	100
12	B-Dubs	4.81	100
3	3B's - Buddies, Bar & Barbecue	4.76	100
68	Paradise	4.70	100

```
#top 5 lowest rated restaurant
avg_hotel_rating.sort_values('Rating', ascending = True)[:5]
```

	Restaurant	Rating	Total_Review
42	Hotel Zara Hi-Fi	2.40	100
11	Asian Meal Box	2.58	100
67	Pakwaan Grand	2.71	100
58	Mathura Vilas	2.82	100

```
#Finding the most followed critic
most_followed_reviewer = review.groupby('Reviewer').agg({'Reviewer_Total_Review':'max',
        'Reviewer_Followers':'max', 'Rating':'mean'}).reset_index().rename(columns = {
            'Rating':'Average_Rating_Given'}).sort_values('Reviewer_Followers', ascending = False)
most_followed_reviewer[:5]
```

	Reviewer	Reviewer_Total_Review	Reviewer_Followers	Average_Rating_Given
5464	Satwinder Singh	186.00	13410.00	3.67
1702	Eat_vth_me	60.00	13320.00	5.00
5236	Samar Sardar	8.00	11329.00	3.50
1788	Foodies Hyderabad	31.00	9494.00	4.50

```
#finding which year show maximum engagement
hotel_year = review.groupby('Review_Year')['Restaurant'].apply(lambda x: x.tolist()).reset_index()
hotel_year['Count']= hotel_year['Restaurant'].apply(lambda x: len(x))
hotel_year
```

	Review_Year	Restaurant	Count
0	2016.00	[Labonel, Labonel, Labonel, Labonel, Labonel, ...	43
1	2017.00	[KS Bakers, KS Bakers, KS Bakers, KS Bakers, K...	213
2	2018.00	[Shah Ghouse Spl Shawarma, Shah Ghouse Spl Sha...	4903

Next steps: [Generate code with `hotel_year`]   [⊙ View recommended plots]   [New interactive sheet]

```
#merging both data frame
hotel = hotel.rename(columns = {'Name':'Restaurant'})
merged = hotel.merge(review, on = 'Restaurant')
merged.shape
```

(9999, 17)

```
#Price point of restaurants
price_point = merged.groupby('Restaurant').agg({'Rating':'mean',
        'Cost': 'mean'}).reset_index().rename(columns = {'Cost': 'Price_Point'})
```

```
#price point for high rated restaurants
price_point.sort_values('Rating',ascending = False)[:5]
```

	Restaurant	Rating	Price_Point
3	AB's - Absolute Barbecues	4.88	1500.00
11	B-Dubs	4.81	1600.00
2	3B's - Buddies, Bar & Barbecue	4.76	1100.00
67	Paradise	4.70	800.00

```
#price point for lowest rated restaurants
price_point.sort_values('Rating',ascending = True)[:5]
```

	Restaurant	Rating	Price_Point
41	Hotel Zara Hi-Fi	2.40	400.00
10	Asian Meal Box	2.58	200.00
66	Pakwaan Grand	2.71	400.00
57	Mathura Vilas	2.82	500.00

```
#rating count by reviewer
rating_count_df = pd.DataFrame(review.groupby('Reviewer').size(), columns=[
                                                    "Rating_Count"])
rating_count_df.sort_values('Rating_Count', ascending = False)[:5]
```

	Rating_Count	
Reviewer		
Parijat Ray	13	
Ankita	13	
Kiran	12	
Vedant Killa	11	

⌄ What all manipulations have you done and insights you found?

Firstly, I started with changing data types for cost and rating. In rating there was only one rating which was string or has value of like so I change it into median of the rating. This was done to make data consistent.

Restaurant data : In this dataset I first figured out 5 costlier restaurant in which Collage - Hyatt Hyderabad Gachibowli has maximum price of 2800 and then found the lowest which is Amul with price of 150. Then I found how many hotel share same price i.e., 13 hotel share 500 price. North indian cuisine with great buffet tags is mostly used in hotels.

Review data : In this dataset I found famous or restaurant that show maximum engagement. Followed by that I found most followed critic which was Satwinder Singh who posted total of 186 reviews and had followers of 13410 who gives and average of 3.67 rating for each order he makes. Lastly I also found in year 2018 4903 hotels got reviews.

Then I merged the two dataset together to figure out the price point for the restaurant, top rated restaurant AB's - Absolute Barbecues has a price point of 1500 and the low rated Hotel Zara Hi-Fi has price point of 400.

In order to exactly understand why even with price point of 1500 these hotel has maximum number of rating and sentiment of those rating need to extract words from the text and do futher analysis of the review and then followed by forming clusters so that one can get recommendation about top quality restaurants.

⌄ *4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables*

⌄ Chart - 1 Distplot for Distribution

```
# Chart — 1 visualization code
plt.figure(figsize = (18,8));
for i,col in enumerate(['Cost','Rating','Review_Year']) :
    # plt.figure(figsize = (8,5));
    plt.subplot(2,2,i+1);
    sns.distplot(merged[col], color = '#055E85', fit = norm);
    feature = merged[col]
    plt.axvline(feature.mean(), color='#ff033e', linestyle='dashed', linewidth=3,label= 'mean');  #red
    plt.axvline(feature.median(), color='#A020F0', linestyle='dashed', linewidth=3,label='median'); #cyan
    plt.legend(bbox_to_anchor = (1.0, 1), loc = 'upper left')
    plt.title(f'{col.title()}');
    plt.tight_layout();
```
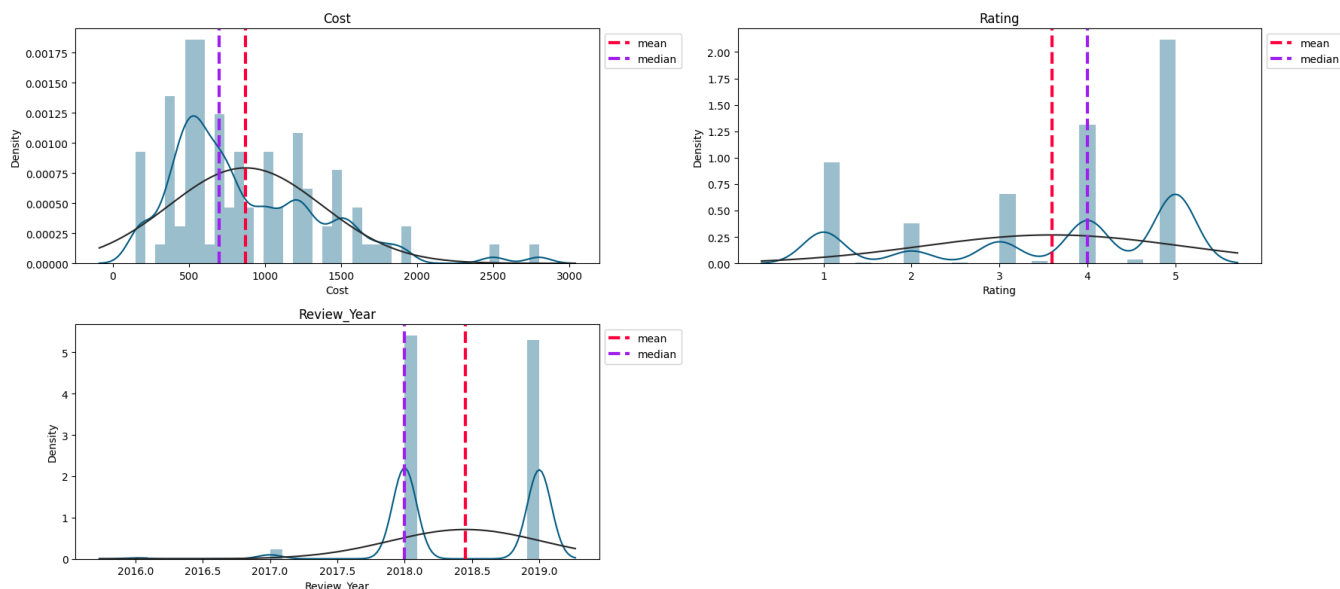
1. Why did you pick the specific chart?

Distplot is helpful in understanding the distribution of the feature.

2. What is/are the insight(s) found from the chart?

- All three are show skewness.
- Maximum restaurant show price range for 500.
- In 2018 number of reviews are more.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Price always place important role in any business alongwith rating which show how much engagement are made for the product.

But in this chart it is unable to figure any impact on business when plotted all alone.

Price Point and Maximum Engagement

Chart - 2 Maximum Engagement and Lowest Engagement

```
#geting the top 10 hotel that show maximum engagement
most_engaged_hotel = price_point.sort_values('Rating', ascending = False)


# Chart – 2 visualization code for most liked
sns.barplot(data = most_engaged_hotel[:10], x = 'Rating', y = 'Restaurant')
plt.title('Most Engaged Restaurant')
plt.show()
```

Most Engaged Restaurant
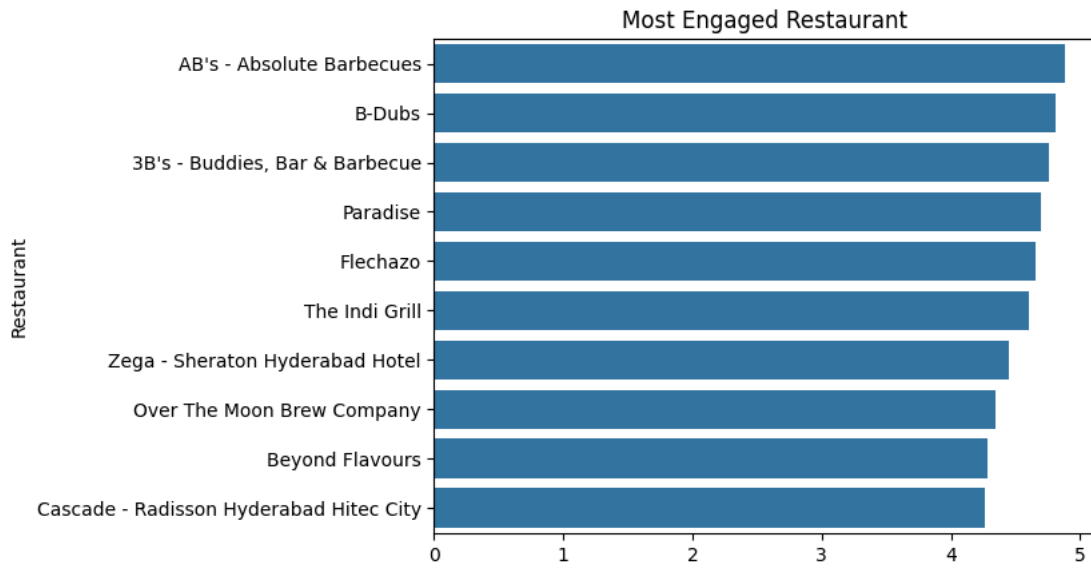

```
#chart for less liked hotels
sns.barplot(data = most_engaged_hotel[-10:], x = 'Rating', y = 'Restaurant')
plt.title('Less Engaged Restaurant')
plt.show()
```

Less Engaged Restaurant
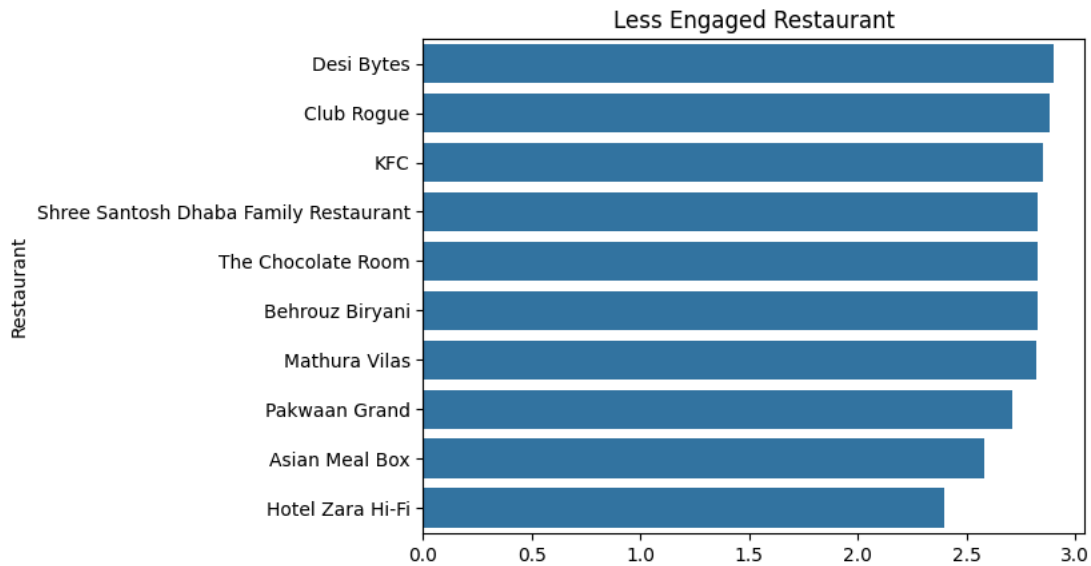
⌄   1. Why did you pick the specific chart?

I picked barplot for the above graph because it show frequency level for different category.

⌄   2. What is/are the insight(s) found from the chart?

AB's - Absolute Barbecues, show maximum engagement and retention as it has maximum number of rating on average and Hotel Zara Hi-Fi show lowest engagement as has lowest average rating.

⌄   3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Engagement and retention for any business is very much important as profit and scalability for any business depend upon retention of customers. Maximum retention means people prefer to use the same brand over others.

Some restaurant show less rating which can show negative growth if not monitored why they recieve less order for example KFC is listed in low rated it is sure they have different outlet and their own outsourcing and lised here because of the popularity of the app and to increase their sale and demand but are not giving 100% dedication to the platform to generate revenue.

∨   Chart - 3 Price Point for High Rated and Low Rated Hotels

```
# Chart — 3 visualization code for price point of high rated restaurant
sns.barplot(data = most_engaged_hotel[:10], x = 'Price_Point', y = 'Restaurant')
plt.title('Price Point for Top Rated Restaurant')
plt.show()
```
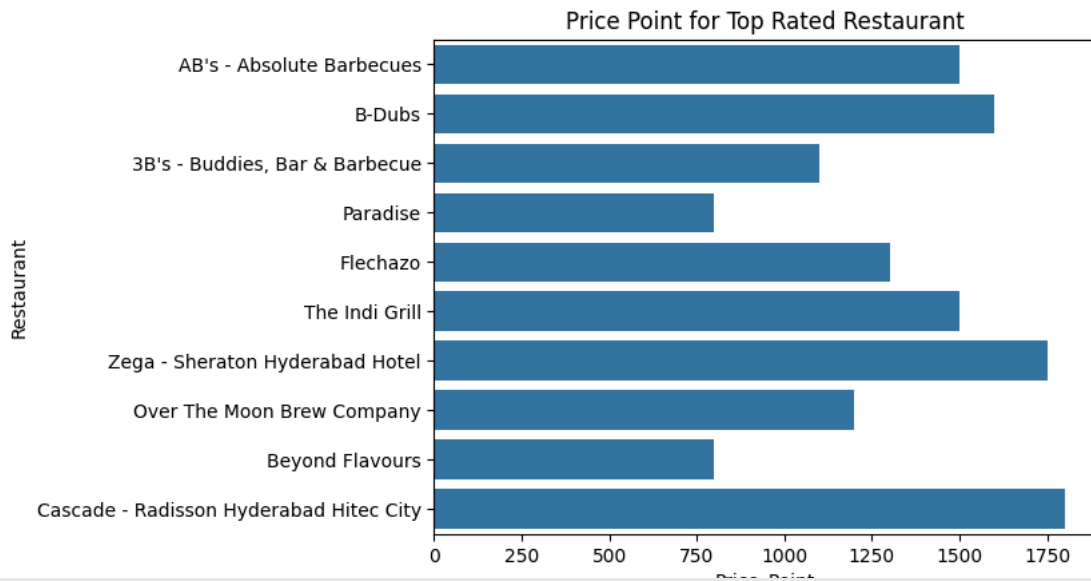


```
#visualization code for price point of low rated restaurant
sns.barplot(data = most_engaged_hotel[-10:], x = 'Price_Point',
            y = 'Restaurant',palette = 'hsv')
plt.title('Price Point for Low Rated Restaurant', size = 15)
# Setting the background color of the plot
# using set_facecolor() method
ax = plt.axes()
ax.set_facecolor("black")
plt.show()
```
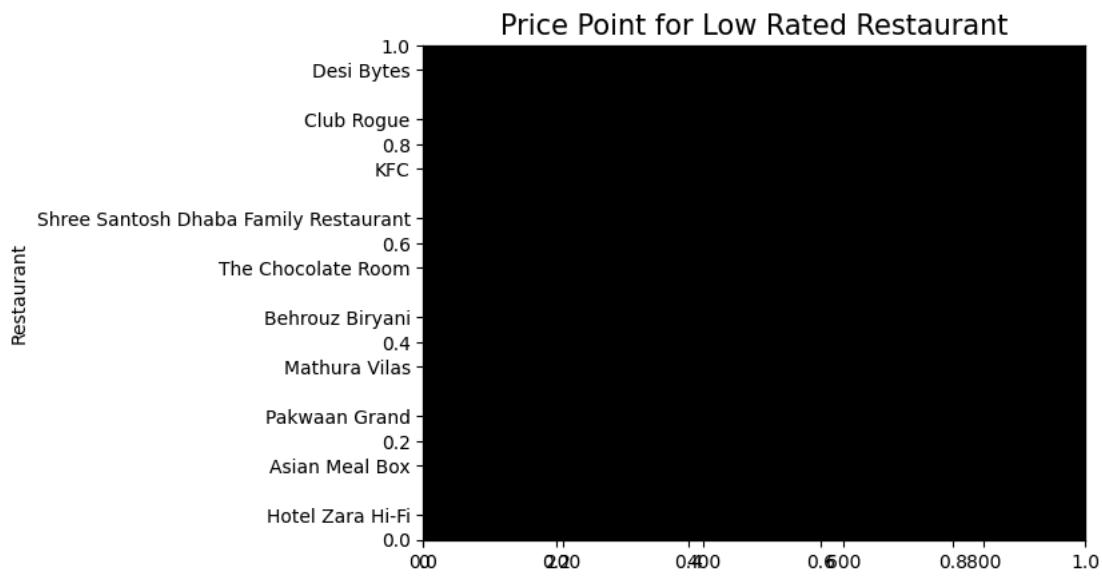

∨   1. Why did you pick the specific chart?

Here I choose barplot because bar plot is a good choice for plotting hotel name and price point as it is a simple and effective way to display the comparison of different categories (hotel names) and their corresponding values (price points) on the same chart. Also, it allow to have a sense of the price range of each hotel and how they compare to each other.

∨   2. What is/are the insight(s) found from the chart?

Price point for high rated hotel AB's= Absolute Barbecues is 1500 and price point for low rated restaurant Hotel Zara Hi-Fi is 400.

∨   3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Since it is customer centered business i.e., direct to consumer it is important to understand price point which makes this business more affordable for evryone, therefore it is important for business to crack the price point.

Here most liked restaurant has a price point of 1500 which is even though a little high than average but as this business is all about food quality and taste it show maximum engagement which means it serve best quality of food, however deep dive on analysing review text can exactly give why this price point is prefered most.

Some restaurant with lowest rating even with low price point is not making engagement, this may create a negative impact on business.

However it can not be finalized that this hotel should unlisted as there may be chance of different cuisine they both serve and it also depend upon the locality they both serve, therefore based on that small promotional offers can also be given for low rated restaurant to increase sales.

∨   Commoditized Cuisine

∨   Chart - 4 Proportion of Cuisine Sold by Most Restaurant

```
#list of all cuisine
cuisine_list = cuisine_df.sort_values('Number of Restaurants', ascending = False)['Cuisine'].tolist()[:5]


# Chart – 4 visualization code pie chart for top 5 mpst selling cuisine
data = cuisine_df.sort_values('Number of Restaurants', ascending = False)[
    'Number of Restaurants'].tolist()[:5]
labels = cuisine_list

#define Seaborn color palette to use
colors = sns.color_palette('Paired')[4:9]

#create pie chart
plt.pie(data, labels = labels, colors = colors, autopct='%.0f%%')
plt.title('Top 5 Most Selling Cuisine', size =22, color= 'blue')
plt.show()
```


Top 5 Most Selling Cuisine
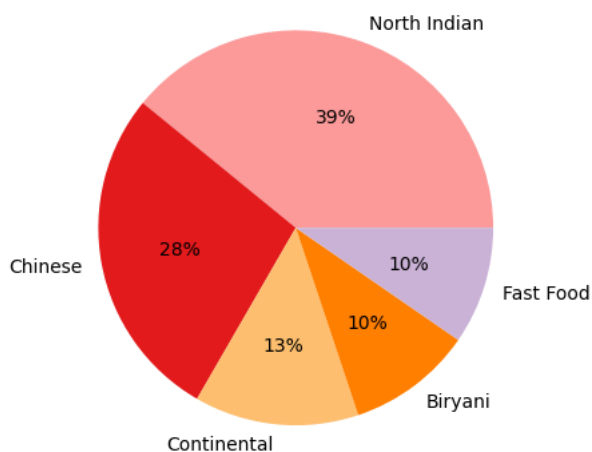
```
#wordcloud for Cuisine
# storind all cuisine in form of text
plt.figure(figsize=(15,8))
text = " ".join(name for name in cuisine_df.Cuisine )


# Creating word_cloud with text as argument in .generate() method

word_cloud = WordCloud(width = 2000, height = 2000,collocations = False,
                       colormap='rainbow',background_color = 'black').generate(text)

# Display the generated Word Cloud
```

```
plt.imshow(word_cloud, interpolation='bilinear');
```

```
plt.axis("off");
```

Aternate way to do the same tast

```
# #creating variable to store restaurant and cuisine from hotel dataset
# cproduct = hotel[['Restaurant','Cost',    'Cuisines']].copy()
# #splitting cuisines
# cproduct['Cuisines'] = cproduct['Cuisines'].str.split(',')
# #exploding the cuisine list from above to separate row
# cproduct = cproduct.explode('Cuisines')
# #removing trailing spaces
# cproduct['Cuisines'] = cproduct['Cuisines'].apply(lambda x: x.strip())
# #grouping cuisines and then making list of restaurants
# cprod = cproduct.groupby('Cuisines')['Restaurant'].apply(lambda x: x.tolist()).reset_index()
# # cproduct['Cuisines'].unique()
# cprod['Restaurant_Count'] = cprod['Restaurant'].apply(lambda x: len(x))
# cprod[cprod['Restaurant_Count']==1].sort_values('Restaurant_Count', ascending = False)
```

∨   1. Why did you pick the specific chart?

Here I choose to use pie chart because it show proportion of each quantity and used wordcloud because it show all text and highlight the most frequent words.

∨   2. What is/are the insight(s) found from the chart?

Based on the above chart it is clear that most of the hotel sold North Indian food followed by chinese.

∨   3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Identifying the Commoditized Cuisine plays an important role as it helps in identifying the challenge or Competitive Advantage i.e., Knowing which cuisines are commoditized allows a restaurant or food business to differentiate themselves from their competitors by offering unique and non-commoditized options.

If a cuisine is commoditized, the prices for ingredients and labor for that cuisine may be higher than for non-commoditized cuisines. Identifying these commoditized cuisines can help a business to control costs by focusing on non-commoditized options or finding ways to lower the cost of commoditized items.

Identifying commoditized cuisines can also provide insight into consumer preferences, which can be used to make informed decisions about menu offerings, pricing, and promotions.

Plotting a pie chart of cuisine types can help to identify the most popular cuisine types among its customers. This information can be used to make strategic decisions about which cuisines to focus on promoting and expanding. For example, as the significant portion of customers are searching for north indian restaurants, Zomato could focus on adding more north indian restaurants to its platform and promoting them to customers.

Similarly, a word cloud of cuisine can help Zomato identify the most frequently mentioned cuisine types in customer reviews. This can provide insight into which cuisines are most popular and well-regarded among customers, and which cuisines may need improvement.

However, these types of charts do not provide all the information about the business, and can not be the only decision making factor. For example, a pie chart showing that a certain cuisine is popular does not tell us about the profitability of that cuisine or the competition in that category. The same goes for word cloud, it only shows us the frequency of the cuisine mentioned, it can not tell us if the mentions are positive or negative.

Additionally, these charts do not provide information about the other factors that can impact the business such as market trends, consumer preferences, and economic conditions. Therefore, it's important for Zomato to consider other data and information when making strategic decisions.

⌄ Chart - 5 Most used Tags

```
#list of all collection
collection_list = Collections_df.sort_values('Number of Restaurants',
                            ascending = False)['Tags'].tolist()[:5]
```

```
# Chart – 5 visualization code pie chart for top 5 mpst selling cuisine
data = cuisine_df.sort_values('Number of Restaurants', ascending = False)[
    'Number of Restaurants'].tolist()[:5]
labels = collection_list

#define Seaborn color palette to use
colors = sns.color_palette('Paired')[:5]

#create pie chart
plt.pie(data, labels = labels, colors = colors, autopct='%.0f%%')
plt.title('Top 5 Most Selling Cuisine', size =22, color= 'red')
plt.show()
```
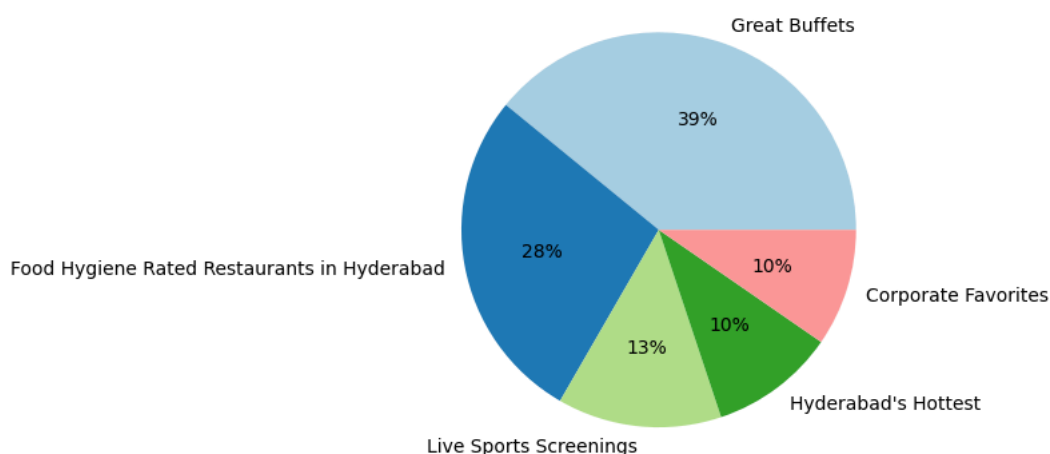


```
#wordcloud for Cuisine
# storind all cuisine in form of text
plt.figure(figsize=(15,8))
text = " ".join(name for name in Collections_df.Tags )


# Creating word_cloud with text as argument in .generate() method

word_cloud = WordCloud(width = 1400, height = 1400,collocations = False,
                    colormap='rainbow', background_color = 'black').generate(text)
```

```
# Display the generated Word Cloud

plt.imshow(word_cloud, interpolation='bilinear');

plt.axis("off");
```


1. Why did you pick the specific chart?

The pie chart provides a clear and simple way to see the proportion of different food attributes, making it easy to identify the most popular attributes and compare them to one another. It also allows for a quick comparison of the popularity of different attributes, and can be useful in identifying patterns or trends in the data.

On the other hand, a word cloud displays the most frequently mentioned attributes in a way that is visually striking and easy to understand. It is useful for identifying the most frequently mentioned attributes and can be used to quickly identify patterns and trends in customer reviews.

Both charts, when used together, can provide a comprehensive understanding of customer reviews and can be used to identify customer preferences, which can help Zomato to make strategic decisions to improve their business.

2. What is/are the insight(s) found from the chart?

Great Buffets is the most frequently used tags and other tags like great, best, north, Hyderabad is also used in large quantity.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Plotting a pie chart of tags used to describe food can help a restaurant review and food delivery platform Zomato to identify the most popular adjectives used to describe the food. This information can be used to make strategic decisions about which food attributes to focus on promoting and expanding. For example, if a significant portion of customers are describing the food as "delicious" or "fresh", Zomato could focus on adding more restaurants that are known for their delicious and fresh food and promoting them to customers.

Similarly, a word cloud of tags used to describe food can help Zomato identify the most frequently mentioned food attributes in customer reviews. This can provide insight into which attributes are most popular and well-regarded among customers, and which attributes may need improvement.

However, it's important to note that these types of charts do not provide all the information about the business, and can not be the only decision making factor. For example, a pie chart showing that a certain adjective is popular does not tell us about the profitability of that

adjective or the competition in that category. The same goes for word cloud, it only shows us the frequency of the adjective mentioned, it can not tell us if the mentions are positive or negative.

Additionally, these charts do not provide information about the other factors that can impact the business such as market trends, consumer preferences, and economic conditions. Therefore, it's important for Zomato to consider other data and information when making strategic decisions. Also, it's important to note that the data used for creating these charts should be cleaned and validated, as the results may be biased if the data is not accurate or complete.

˅   Most Popular Critics

Chart - 6 Learn about Reviewers

˅   Chart - 6 visualization code for most review

```
# Chart — 6 visualization code for most review
sns.barplot(data = most_followed_reviewer[:10], x = 'Reviewer_Total_Review',
            y = 'Reviewer', palette='bright')
plt.title('Reviewer given Maximum Review')
plt.show()
```
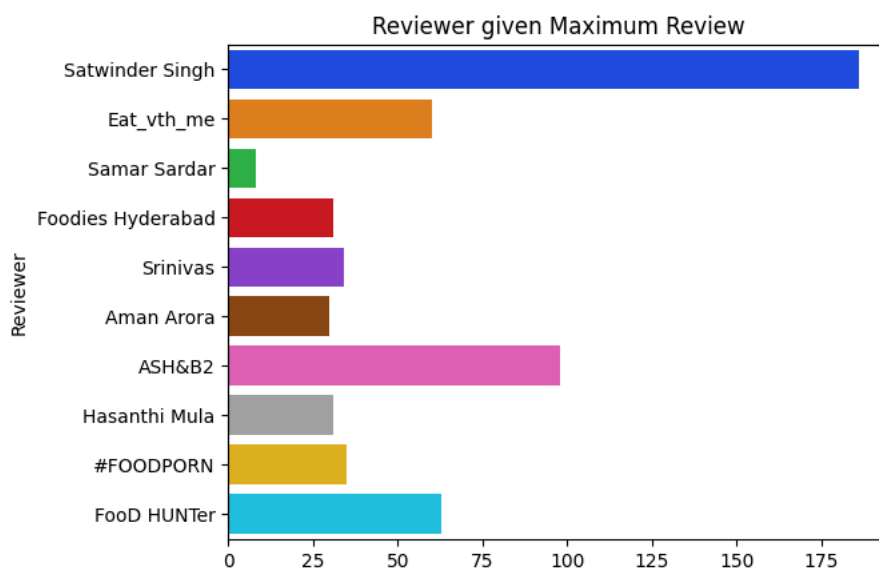


```
# visualization code for most review follower
sns.barplot(data = most_followed_reviewer[:10], x = 'Reviewer_Followers',
            y = 'Reviewer',palette='bright')
plt.title('Most followed Reviewer')
plt.show()
```
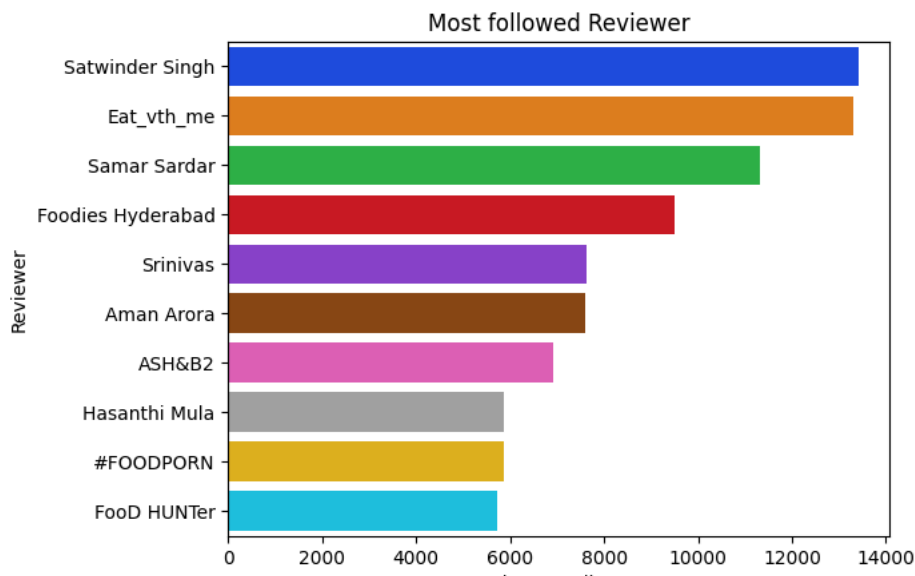
```
# visualization code for average rating given by most followed reviewer
sns.barplot(data = most_followed_reviewer[:10], x = 'Average_Rating_Given',
            y = 'Reviewer',palette='bright')
plt.title('Average Ratings give by Most followed Reviewer')
plt.show()
```


Average Ratings give by Most followed Reviewer
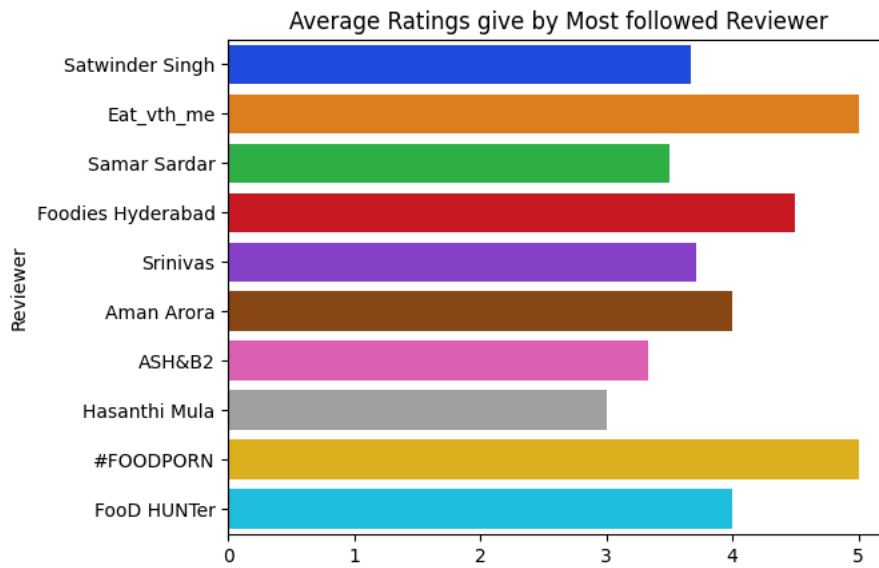
1. Why did you pick the specific chart?

Barplot helps in understanding the frequency of rating, follower and total reviews with respect to reviewer. Plotting total review, average reviewer rating, and total follower allows to see the correlation between these variables and how they relate to one another for each reviewer. It can also give insight on how reviewers with more followers tend to get more reviews, how their ratings tend to be, etc.

2. What is/are the insight(s) found from the chart?

Satwinder singh is the most popular critic who has maximum number of follower and on an average he give 3.5 rating.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

This information can be used to make strategic decisions about which reviewers to focus on promoting and expanding. For example, if a certain reviewer has a high average rating and a large number of followers, Zomato could focus on promoting their reviews to customers.

It's important to note that this chart does not provide all the information about the business, and can not be the only decision making factor. However it can help on promotions food based on reviews.
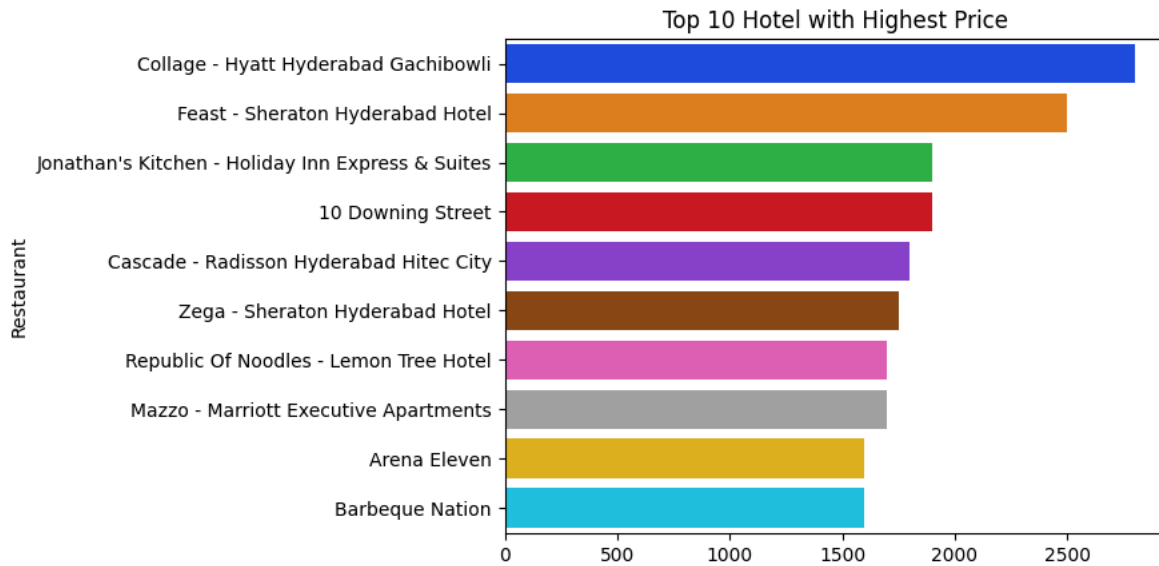
Most Expensive Restaurant

**Chart - 7 Hotel with Highest Price and Lowest Price**

```
#extracting name and price

price_of_hotel = hotel.sort_values('Cost', ascending = False)[['Restaurant','Cost']]
```

```
# Chart — 7 visualization code for howtel with maximum price
sns.barplot(data = price_of_hotel[:10], x = "Cost", y='Restaurant', palette = 'bright')
plt.title('Top 10 Hotel with Highest Price')
plt.show()
```

Top 10 Hotel with Highest Price

```
#hotel with lowest price
sns.barplot(data = price_of_hotel[-10:], x = "Cost", y='Restaurant', palette = 'hsv')
plt.title('Top 10 Hotel with Lowest Price', size =15, color = 'red')
plt.show()
```


Top 10 Hotel with Lowest Price
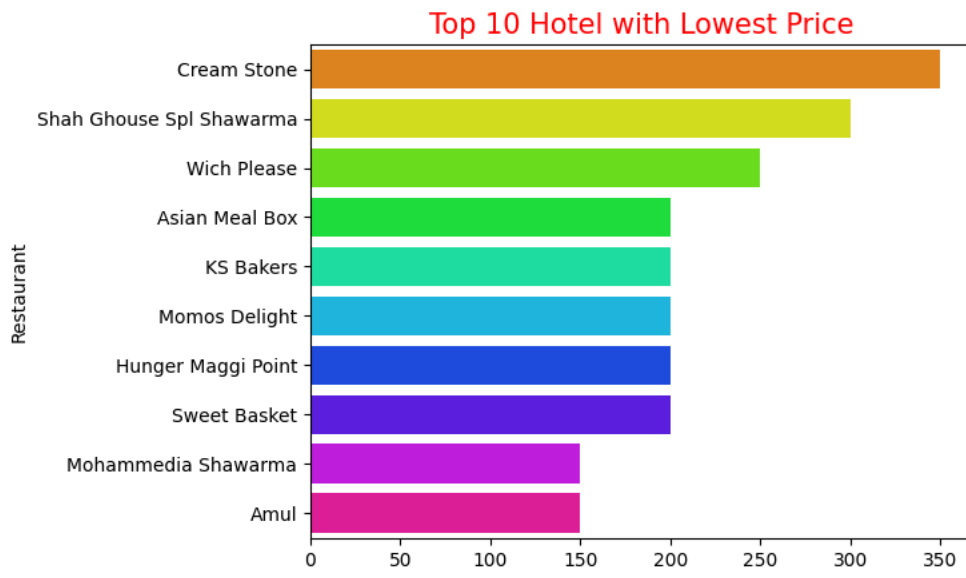
```
#merging average rating and cost to find rating for expensive hotel
expected_revenue = avg_hotel_rating.merge(hotel[['Restaurant','Cost']], on = 'Restaurant')
#calculating expected revenue based on total review recieved
expected_revenue['Expected_Revenue'] = expected_revenue['Total_Review'] * expected_revenue['Cost']


#chart for rating based on price and hotel
plt.figure(figsize=(16,6))
data = expected_revenue.sort_values('Cost', ascending  = False)
sns.scatterplot(data= data, x= "Restaurant", y="Rating", size="Cost",
                hue = 'Cost',legend=True, sizes=(20, 2000),palette ="icefire")
plt.xticks(rotation=90)
plt.title('Restaurants based on Price and Ratings',size=20,color = 'red')
plt.show()
```

Restaurants based on Price and Ratings
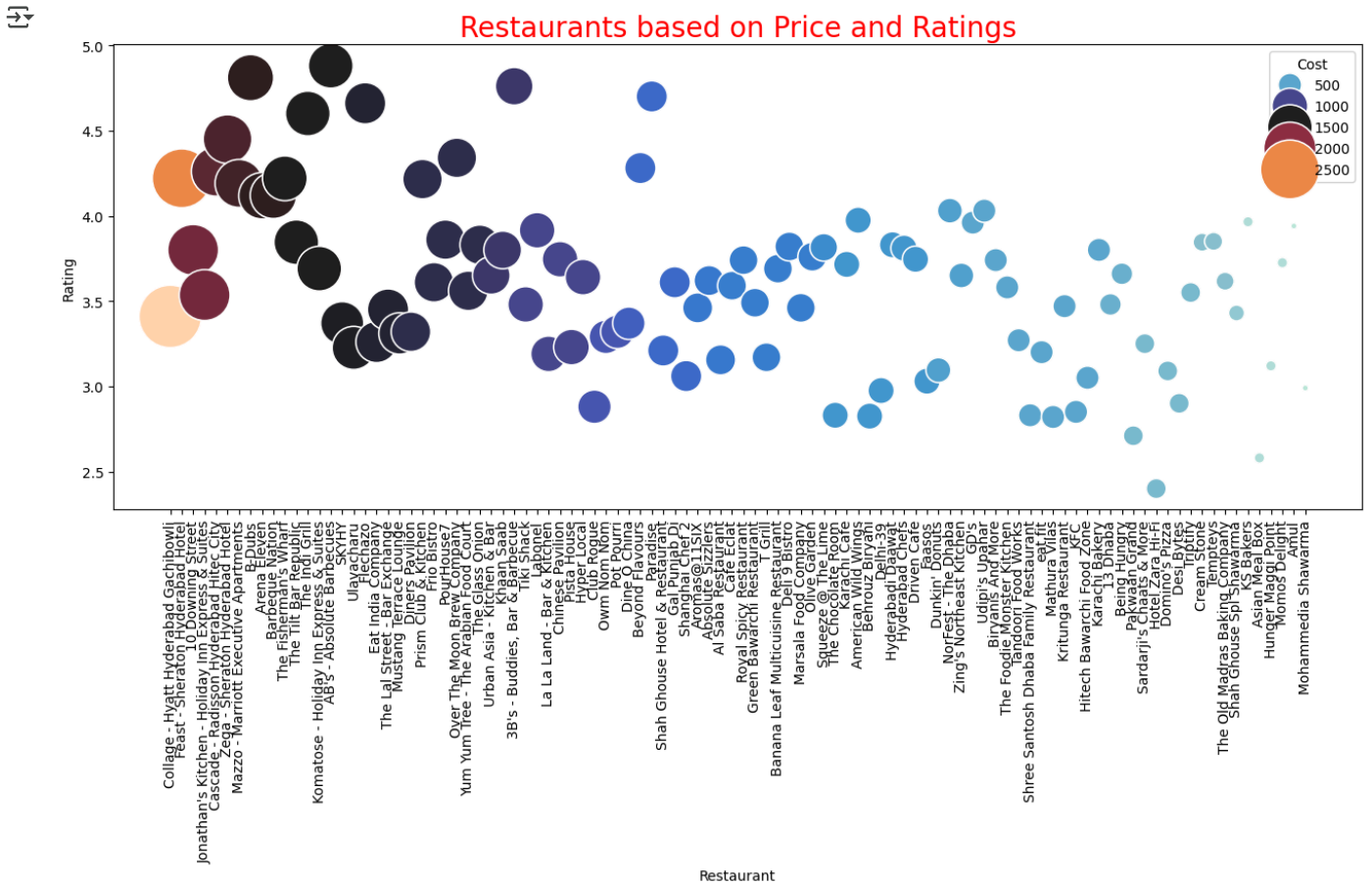
```
#chart to understand expected revenue
fig = plt.figure(figsize=[20,6])
sns.barplot(data= data, x='Restaurant', y= 'Expected_Revenue', palette ="rainbow")
plt.title("Expected_Revenue from each Restaurant", size = 22)
plt.xlabel('Restaurant Name', size = 22)
plt.xticks(rotation=90)
plt.ylabel('Expected_Revenue', size = 22)
plt.show()
```

Expected_Revenue from each Restaurant
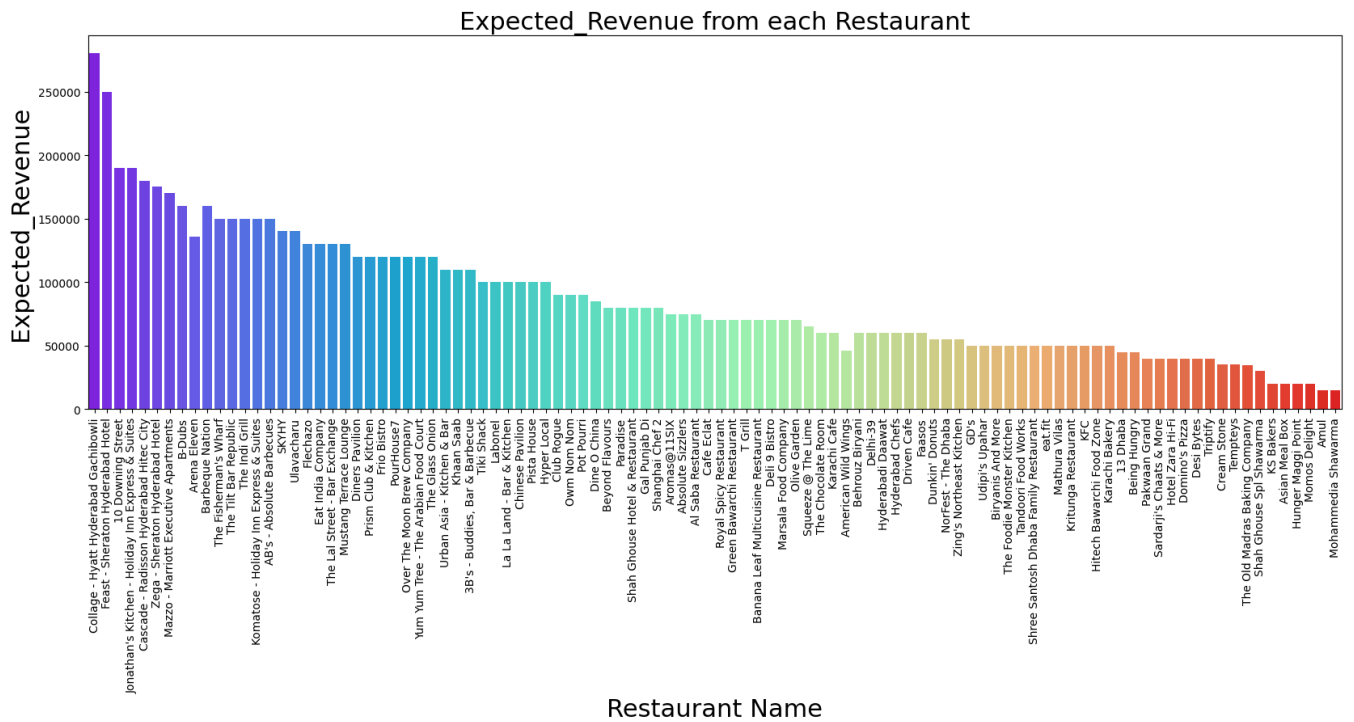
1. Why did you pick the specific chart?

Barplot helps in plotting the frquency of cost for each hotel.

2. What is/are the insight(s) found from the chart?

Based on the above chart it is clear that restaurant Collage - Hyatt Hyderabad Gachibowli is most expensive restaurant in the locality which has a price of 2800 for order and has 3.5 average rating.

Hotels like Amul and Mohammedia Shawarma are least expensive with price of 150 and has 3.9 average rating.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Most expensive product are always center of attraction for a niche market (subset of the market on which a specific product is focused) at the same time for a business purpose, this product are preffered to be most revenue generating market.

Definetly for food delivery platform Zomato, it is very important to focus and improve sales based on these hotels.

Based on the average rating of 3.4 these product should increase their engagement as this may cause negative brand impact. However true behaviour can only be inspected through analysing of reviews.

Chart - 8 - Correlation Heatmap

```
# Convert all columns to numeric, non-numeric columns will become NaN
merged_numeric = merged.apply(pd.to_numeric, errors='coerce')

# Drop columns that were entirely non-numeric (i.e., all NaN after conversion)
merged_numeric = merged_numeric.dropna(axis=1, how='all')


# Correlation Heatmap visualization code
# checking heatmap/correlation matrix to see the how the colums are correlated with each other
f, ax = plt.subplots(figsize = (20, 10))
sns.heatmap(merged_numeric.corr(),ax = ax, annot=True, cmap = 'icefire', linewidths = 1)
```

```
plt.show()
```
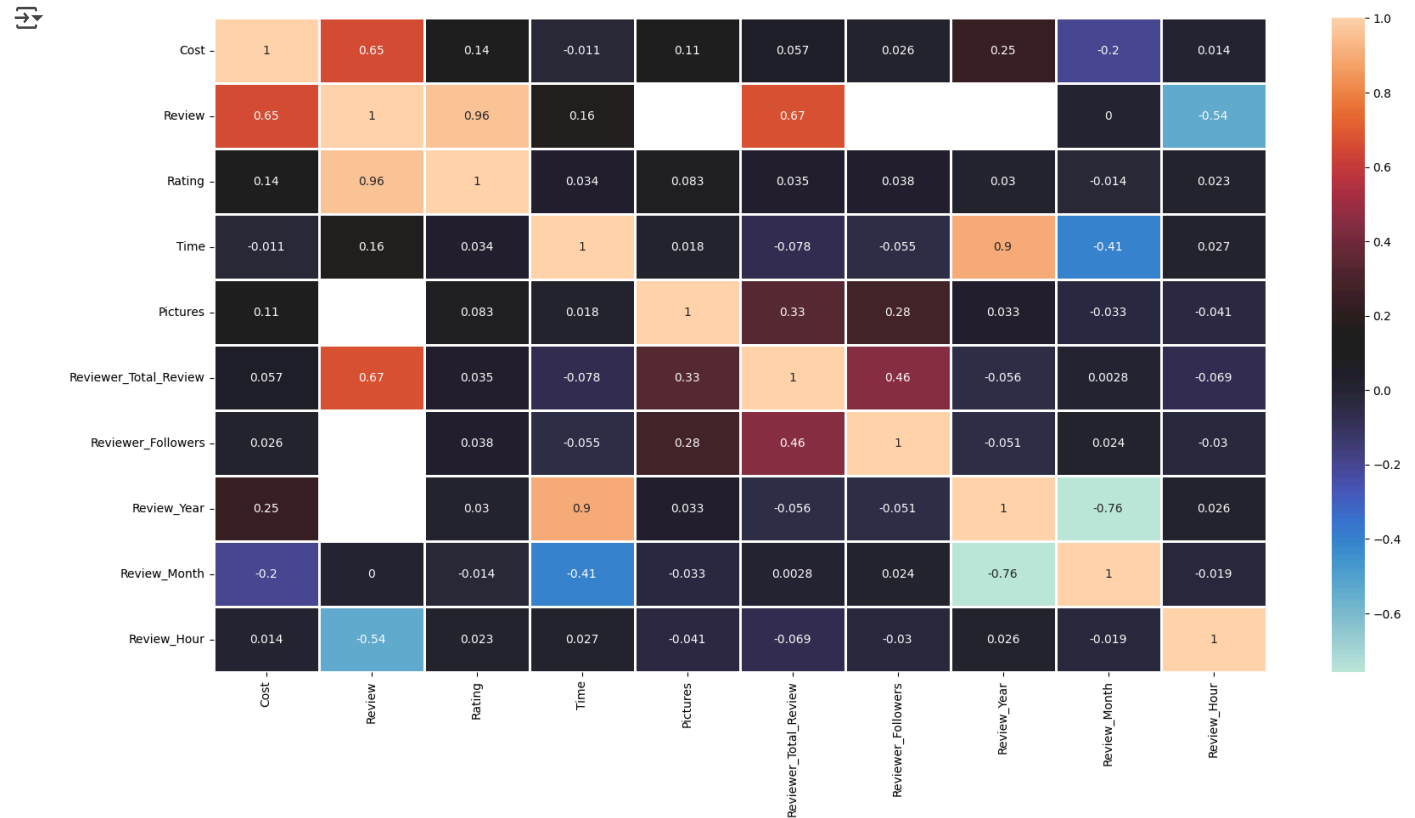

1. Why did you pick the specific chart?

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses. The range of correlation is [-1,1].

Thus to know the correlation between all the variables along with the correlation coeficients, i used correlation heatmap.

2. What is/are the insight(s) found from the chart?

From the above correlation heatmap, it can be depicted that few features are correlated, like reviewer total review is related to reviewer follower and again reviewer total review is related to pictures.

Rest all correlation can be depicted from the above chart.

Chart - 9 - Pair Plot

```
# Pair Plot visualization code
sns.pairplot(merged);
```

1. Why did you pick the specific chart?

Pair plot is used to understand the best set of features to explain a relationship between two variables or to form the most separated clusters. It also helps to form some simple classification models by drawing some simple lines or make linear separation in our data-set.

Thus, I used pair plot to analyse the patterns of data and realationship between the features. It's exactly same as the correlation map but here you will get the graphical representation.

2. What is/are the insight(s) found from the chart?

It can be seen that there is no significant correlation between the given features in the merged dataframe.

✓ *5. Hypothesis Testing*

✓ Based on your chart experiments, define three hypothetical statements from the dataset. In the next three questions, perform hypothesis testing to obtain final conclusion about the statements through your code and statistical testing.

- The cost of a restaurant is positively correlated with the rating it receives.
- Restaurants that are reviewed by reviewers with more followers will have a higher rating.
- Restaurants that offer a wider variety of cuisines will have a higher rating.

✓ Hypothetical Statement - 1

The cost of a restaurant is positively correlated with the rating it receives.

✓ 1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

- Null hypothesis: There is no relationship between the cost of restaurant and the rating it receives. (H0: $\beta 1 = 0$)
- Alternative hypothesis: There is a positive relationship between the cost of a restaurant and the rating it receives. (H1: $\beta 1 > 0$)
- Test : Simple Linear Regression Analysis

✓ 2. Perform an appropriate statistical test.

```
# Perform Statistical Test to obtain P-Value
import statsmodels.formula.api as smf


# fit the linear model
model = smf.ols(formula='Rating ~ Cost', data= merged).fit()

# Check p-value of coefficient
p_value = model.pvalues[1]
if p_value < 0.05:
    print("Reject Null Hypothesis - There is no relationship between the cost of\
 restaurant and the rating it receives.")
else:
    print("Fail to reject Null Hypothesis - There is a positive relationship \
 between the cost of a restaurant and the rating it receives.")
```

⇥ Reject Null Hypothesis - There is no relationship between the cost of restaurant and the rating it receives.

✓ Which statistical test have you done to obtain P-Value?

I have used Linear regression test for checking the relationship between the cost of a restaurant and its rating

✓ Why did you choose the specific statistical test?

I chose this test because it is a common and straightforward method for testing the relationship between two continuous variables. This would involve fitting a linear model with the rating as the dependent variable and the cost as the independent variable. The p-value of the coefficient for the cost variable can then be used to determine if there is a statistically significant relationship between the two variables.

✓ Hypothetical Statement - 2

Restaurants that are reviewed by reviewers with more followers will have a higher rating.

✓ 1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

- Null hypothesis: The number of followers a reviewer has no effect on the rating of a restaurant. (H0: $\beta 1 = 0$)
- Alternative hypothesis: Alternative Hypothesis: The number of followers a reviewer has has a positive effect on the rating of a restaurant. (H1: $\beta 1 > 0$)
- Test : Simple Linear Regression test

∨ 2. Perform an appropriate statistical test.

```
# Perform Statistical Test to obtain P-Value
import statsmodels.formula.api as smf


# fit the linear model
model = smf.ols(formula='Rating ~ Reviewer_Followers', data = merged).fit()

# print the summary of the model
# print(model.summary())

# extract p-value of coefficient for Reviewer_Followers
p_value = model.pvalues[1]

if p_value < 0.05:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
```

⇥ Reject Null Hypothesis

∨ Which statistical test have you done to obtain P-Value?

For the second hypothesis, I have used Simple Linear Regression Test.

∨ Why did you choose the specific statistical test?

I choose this test because it is a straightforward method for testing the relationship between two continuous variables. It assumes that there is a linear relationship between the independent variable (Reviewer_Followers) and the dependent variable (Rating) and it allows us to estimate the strength and direction of that relationship. It also allows us to test the null hypothesis that there is no relationship between the two variables by testing the p-value of the coefficient of the independent variable.

∨ Hypothetical Statement - 3

Restaurants that offer a wider variety of cuisines will have a higher rating.

∨ 1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

- Null hypothesis: The variety of cuisines offered by a restaurant has no effect on its rating. (H0: $\beta3 = 0$)
- Alternative hypothesis: The variety of cuisines offered by a restaurant has a positive effect on its rating. (H1: $\beta3 > 0$)
- Test : Chi-Squared Test

∨ 2. Perform an appropriate statistical test.

```
pd.crosstab(merged['Cuisines'], merged['Rating'])[:1]
```

⇥   **Rating 1.00 1.50 2.00 2.50 3.00 3.50 4.00 4.50 5.00**   ⊞

   Cuisines

```
# Perform Statistical Test to obtain P-Value
from scipy.stats import chi2_contingency

# create a contingency table
ct = pd.crosstab(merged['Cuisines'], merged['Rating'])

# perform chi-squared test
chi2, p, dof, expected = chi2_contingency(ct)

# Check p-value
if p < 0.05:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
```

```
Reject Null Hypothesis
```

∨   Which statistical test have you done to obtain P-Value?

For the third hypothesis, I have used chi-squared test for independence to test the relationship between the variety of cuisines offered by a restaurant and its rating.

∨   Why did you choose the specific statistical test?

I choose this test because it is suitable for comparing the relationship between two categorical variables. This would involve creating a contingency table with the number of restaurants that offer each cuisine as the rows and the rating of the restaurant as the columns.

∨ *6. Feature Engineering & Data Pre-processing*

∨   1. Handling Missing Values

∨   Treating Duplicates

- Since all the duplicated data has NaN values, hence dropping the entire values as it will not help and will create unnecessary noise.

```
#deleting duplicate value from review dataset
review = review.drop_duplicates()


#final check after dropping duplicates
print(f"Anymore duplicate left ? {review.duplicated().value_counts()}, unique values with {len(review[review.duplicated()])})
```

```
Anymore duplicate left ? False    9964
Name: count, dtype: int64, unique values with 0 duplication
```

∨   Treating Missing Values

Restaurant Dataset

```
# Handling Missing Values & Missing Value Imputation
hotel.isnull().sum()
```

	0
Restaurant	0
Links	0
Cost	0
Collections	54
Cuisines	0
Timings	1

```
#checking the null value in timing
hotel[hotel['Timings'].isnull()]
```

Restaurant	Links	Cost	Collections	Cuisines	Timings	

```
#filling null value in timings column
hotel.Timings.fillna(hotel.Timings.mode()[0], inplace = True)


#checking null values in Collections
missing_percentage = ((hotel['Collections'].isnull().sum())/(len(hotel['Collections'])))*100
print(f'Percentage of missing value in Collections is {round(missing_percentage, 2)}%')
```

```
Percentage of missing value in Collections is 51.43%
```

```
#dropping collection column since has more than 50% of null values
hotel.drop('Collections', axis = 1, inplace = True)
```

```
#final checking of missing value
hotel.isnull().sum()
```

	0
Restaurant	0
Links	0
Cost	0
Cuisines	0
Timings	0

Review Dataset

```
#review missing value
review.isnull().sum()
```

	0
Restaurant	0
Reviewer	2
Review	9
Rating	2
Metadata	2
Time	3
Pictures	0
Reviewer_Total_Review	3
Reviewer_Followers	1581
Review_Year	3
Review_Month	3
Review_Hour	3

```
#checking null reviewer
review[review['Reviewer'].isnull()]
```

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures	Reviewer_Total_Review	Reviewer_Followers	Review_Yea
8777	American Wild Wings	NaN	NaN	NaN	NaN	NaT	0	NaN	NaN	Na
9085	Arena Eleven	NaN	NaN	NaN	NaN	NaT	0	NaN	NaN	Na

```
#checking null Reviewer_Total_Review
review[review['Reviewer_Total_Review'].isnull()]
```

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures	Reviewer_Total_Review	Reviewer_Followers	Review_Yea
7601	4.00	4.00	4.00	4.00	4.00	NaT	4	NaN	NaN	Na
8777	American Wild Wings	NaN	NaN	NaN	NaN	NaT	0	NaN	NaN	Na
9085	Arena Eleven	NaN	NaN	NaN	NaN	NaT	0	NaN	NaN	Na

```
# dropping null values in reviewer and Reviewer_Total_Review column as all values are null for those column
review = review.dropna(subset=['Reviewer','Reviewer_Total_Review'])
```

```
#again checking the remaining values
null_counts = [(x, a) for x, a in review.isnull().sum().items() if a > 0]

# Print the columns with null values
null_counts
```

[('Review', 7), ('Reviewer_Followers', 1578)]

```
#filling null values in review and reviewer follower column
review = review.fillna({"Review": "No Review", "Reviewer_Followers": 0})

# final checking null values
review.isnull().sum()
```

	0
Restaurant	0
Reviewer	0
Review	0
Rating	0
Metadata	0
Time	0
Pictures	0
Reviewer_Total_Review	0
Reviewer_Followers	0
Review_Year	0
Review_Month	0
Review_Hour	0

```
#merging both dataset
merged = hotel.merge(review, on = 'Restaurant')
merged.shape
```

(9961, 16)

What all missing value imputation techniques have you used and why did you use those techniques?

I started treating missing values by first removing the duplicate data where all other values were NaN or null values except had restaurant name, so instead of replacing each null value I removed it as it was only 36 duplicate data which had no unique identity.

Dataset that contains details about hotel, had 1 null value in timing feature and more than 50% null value in collection feature. In order to treat with those I first replaced the null value for timing with mode since there was only one null and mode is robust to outliers plus that hotel name was one unique feature which had all other feature except timing and collection so it was better to preserve that data. Since there was more than 50% null values in collection feature, I removed the entire column because columns with a high percentage of null values are likely to have a lot of missing data, which can make it difficult to accurately analyze or make predictions based on the data.

In the dataset tha has details of reviewer had Reviewer - 2, Review - 9, Rating - 2, Metadata - 2, Time - 2, Reviewer_Total_Review- 3, Reviewer_Followers - 1581, Review_Year - 2, Review_Month - 2, Review_Hour - 2. On analysing I found that feature like reviewer and reviewer total review had all null values, therefore I removed those two columns which made null values in other feature to zero except in review and reviewer followers columns. Since review was textual data, I changed those 7 null values to 'no review' and reviewer followers to 0 as follower is the meta data for reviewer and it can be 0.

And thus all the null values were treated, at the end I then again merged both the dataset hotel and review dataset.

2. Handling Outliers

Detecting Anamoly

```
#Anamoly detection
from sklearn.ensemble import IsolationForest
#checking for normal distribution
print("Skewness – Cost: %f" % merged['Cost'].skew())
print("Kurtosis – Cost: %f" % merged['Cost'].kurt())
print("Skewness – Reviewer_Followers: %f" % merged['Reviewer_Followers'].skew())
```

```
print("Kurtosis - Reviewer_Followers: %f" % merged['Reviewer_Followers'].kurt())
```

```
Skewness - Cost: 1.153637
Kurtosis - Cost: 1.571590
Skewness - Reviewer_Followers: 10.092703
Kurtosis - Reviewer_Followers: 151.312960
```

```
#plotting graph for cost
plt.scatter(range(merged.shape[0]), np.sort(merged['Cost'].values))
plt.xlabel('index')
plt.ylabel('Cost')
plt.title("Cost distribution")
sns.despine()
```
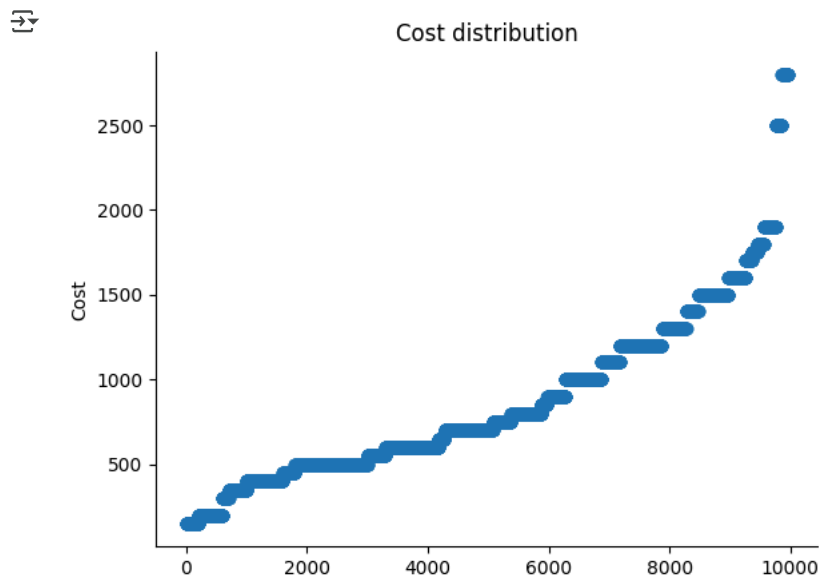


```
#distribution of cost
sns.distplot(merged['Cost'])
plt.title("Distribution of Cost")
sns.despine()
```
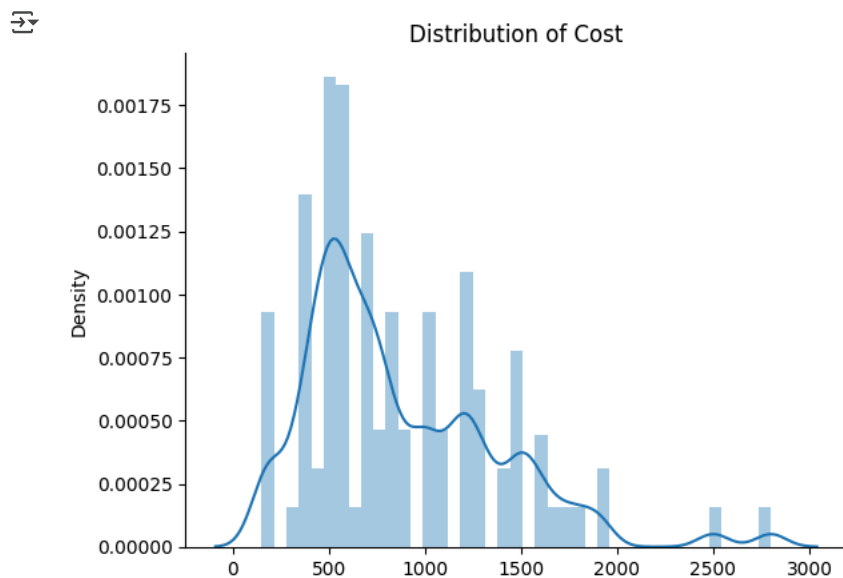


```
#plot for reviewer follower
plt.scatter(range(merged.shape[0]), np.sort(merged['Reviewer_Followers'].values))
plt.xlabel('index')
plt.ylabel('Reviewer_Followers')
plt.title("Reviewer_Followers distribution")
sns.despine()
```

Reviewer_Followers distribution
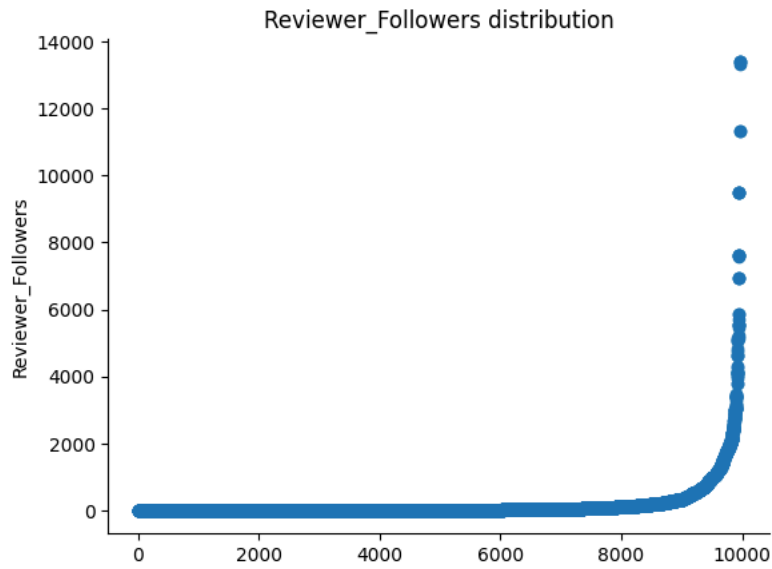
```python
#distribution of Reviewer_Followers
sns.distplot(merged['Reviewer_Followers'])
plt.title("Distribution of Reviewer_Followers")
sns.despine()
```


Distribution of Reviewer_Followers
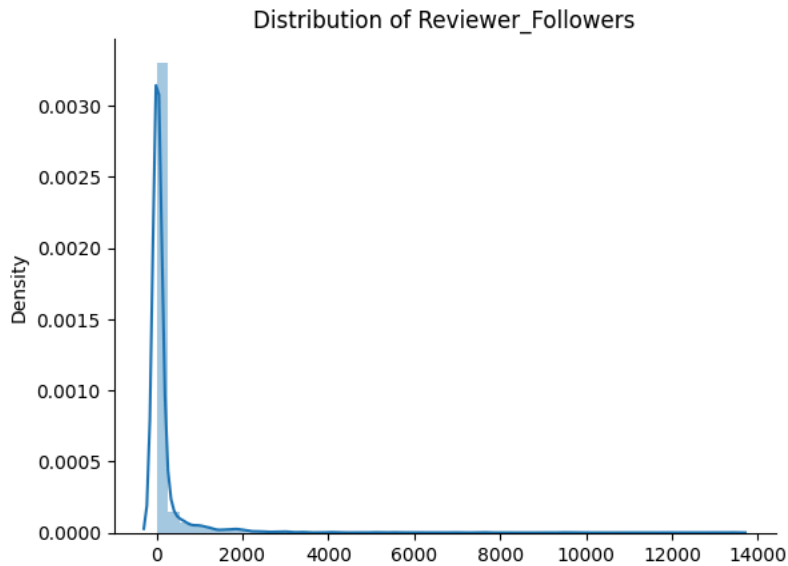
```python
#isolation forest for anamoly detection on cost
isolation_forest = IsolationForest(n_estimators=100, contamination=0.01)
isolation_forest.fit(merged['Cost'].values.reshape(-1, 1))
merged['anomaly_score_univariate_Cost'] = isolation_forest.decision_function(merged['Cost'].values.reshape(-1, 1))
merged['outlier_univariate_Cost'] = isolation_forest.predict(merged['Cost'].values.reshape(-1, 1))


#chart to visualize outliers
xx = np.linspace(merged['Cost'].min(), merged['Cost'].max(), len(merged)).reshape(-1,1)
anomaly_score = isolation_forest.decision_function(xx)
outlier = isolation_forest.predict(xx)
plt.figure(figsize=(10,4))
plt.plot(xx, anomaly_score, label='anomaly score')
plt.fill_between(xx.T[0], np.min(anomaly_score), np.max(anomaly_score),
where=outlier==-1, color='r',
alpha=.4, label='outlier region')
plt.legend()
plt.ylabel('anomaly score')
plt.xlabel('Cost')
plt.show();
```
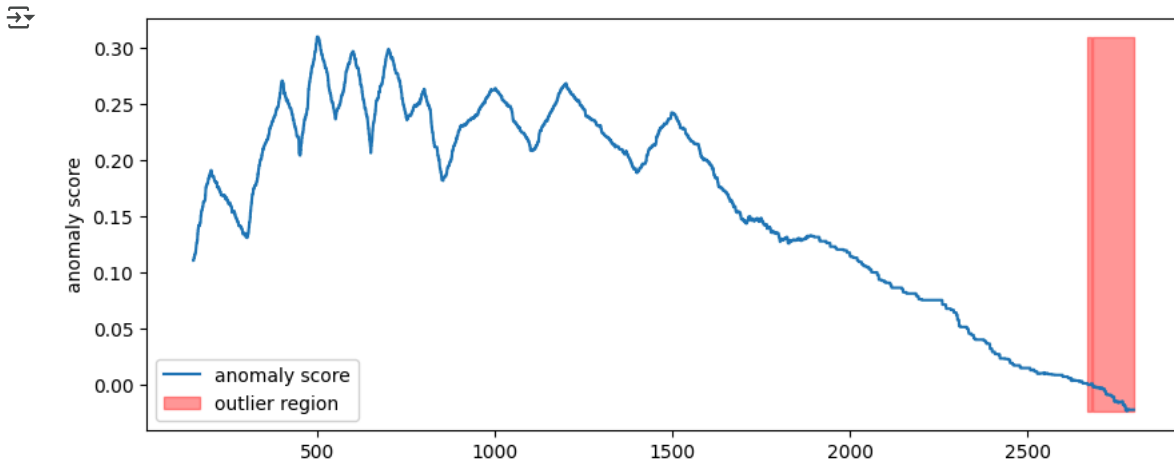
```
#isolation forest for anamoly detection of reviewer follower
isolation_forest = IsolationForest(n_estimators=100, contamination=0.01)
isolation_forest.fit(merged['Reviewer_Followers'].values.reshape(-1, 1))
merged['anomaly_score_univariate_follower'] = isolation_forest.decision_function(
    merged['Reviewer_Followers'].values.reshape(-1, 1))
merged['outlier_univariate_follower'] = isolation_forest.predict(
    merged['Reviewer_Followers'].values.reshape(-1, 1))
```

```
#chat to visualize outliers in reviwer follower column
xx = np.linspace(merged['Reviewer_Followers'].min(), merged['Reviewer_Followers'].max(), len(merged)).reshape(-1,1)
anomaly_score = isolation_forest.decision_function(xx)
outlier = isolation_forest.predict(xx)
plt.figure(figsize=(10,4))
plt.plot(xx, anomaly_score, label='anomaly score')
plt.fill_between(xx.T[0], np.min(anomaly_score), np.max(anomaly_score),
where=outlier==-1, color='r',
alpha=.4, label='outlier region')
plt.legend()
plt.ylabel('anomaly score')
plt.xlabel('Reviewer_Followers')
plt.show();
```
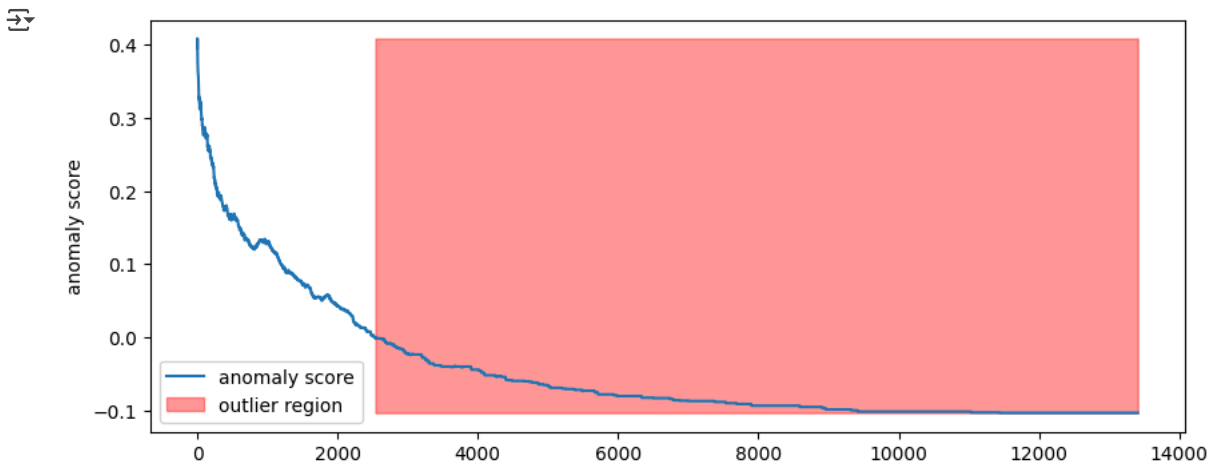

⌄ Treating Outlier

```
# Handling Outliers & Outlier treatments
# To separate the symmetric distributed features and skew symmetric distributed features
symmetric_feature=[]
non_symmetric_feature=[]
for i in merged.describe().drop("Time",axis=1).columns:
  print(abs(merged[i].mean()-merged[i].median()))
  if abs(merged[i].mean()-merged[i].median())<0.2:
    symmetric_feature.append(i)
  else:
    non_symmetric_feature.append(i)

# Getting Symmetric Distributed Features
print("Symmetric Distributed Features : -",symmetric_feature)
```

```
# Getting Skew Symmetric Distributed Features
print("Skew Symmetric Distributed Features : -",non_symmetric_feature)
```

```
171.58417829535188
0.39895592811966685
0.7514305792591105
19.992069069370544
156.00843288826422
0.4520630458789583
1.0179700833249674
0.8122678445939169
0.02089105350586723
0.020078305391024953
0.036827805527310464
0.01947595622929421
Symmetric Distributed Features : - ['anomaly_score_univariate_Cost', 'outlier_univariate_Cost', 'anomaly_score_univariat
Skew Symmetric Distributed Features : - ['Cost', 'Rating', 'Pictures', 'Reviewer_Total_Review', 'Reviewer_Followers', 'R
```

```
# For Skew Symmetric features defining upper and lower boundry
#Outer Fence
def outlier_treatment_skew(df,feature):
  IQR= df[feature].quantile(0.75)- df[feature].quantile(0.25)
  lower_bridge =df[feature].quantile(0.25)- 1.5*IQR
  upper_bridge =df[feature].quantile(0.75)+ 1.5*IQR
  # print(f'upper : {upper_bridge} lower : {lower_bridge}')
  return upper_bridge,lower_bridge
```

```
# Restricting the data to lower and upper boundary for cost in hotel dataset
#lower limit capping
hotel.loc[hotel['Cost']<= outlier_treatment_skew(df=hotel,
  feature='Cost')[1], 'Cost']=outlier_treatment_skew(df=hotel,feature='Cost')[1]

#upper limit capping
hotel.loc[hotel['Cost']>= outlier_treatment_skew(df=hotel,
  feature='Cost')[0], 'Cost']=outlier_treatment_skew(df=hotel,feature='Cost')[0]
```

```
# Restricting the data to lower and upper boundary for Reviewer followers in review dataset
#lower limit capping
review.loc[review['Reviewer_Followers']<= outlier_treatment_skew(df=review,
  feature='Reviewer_Followers')[1], 'Reviewer_Followers']=outlier_treatment_skew(
      df=review,feature='Reviewer_Followers')[1]

#upper limit capping
review.loc[review['Reviewer_Followers']>= outlier_treatment_skew(df=review,
  feature='Reviewer_Followers')[0], 'Reviewer_Followers']=outlier_treatment_skew(
      df=review,feature='Reviewer_Followers')[0]
```

```
#dropping the columns created while outliers treatment
merged.drop(columns =['anomaly_score_univariate_Cost','outlier_univariate_Cost',
  'anomaly_score_univariate_follower','outlier_univariate_follower'], inplace = True)
```

∨   What all outlier treatment techniques have you used and why did you use those techniques?

Since cost and reviewer follower feature or column show positive skewed distribution and using isolation forest found they have outliers,
hence using the capping technique instead of removing the outliers, capped outliers with the highest and lowest limit using IQR method.

∨   3. Categorical Encoding

```
# Encode your categorical columns

#categorial encoding using pd.getdummies
#new df with important categories
cluster_dummy = hotel[['Restaurant','Cuisines']]
#spliting cuisines as they are separted with comma and converting into list
cluster_dummy['Cuisines'] = cluster_dummy['Cuisines'].str.split(',')
#using explode convverting list to unique individual items
cluster_dummy = cluster_dummy.explode('Cuisines')
#removing extra trailing space from cuisines after exploded
cluster_dummy['Cuisines'] = cluster_dummy['Cuisines'].apply(lambda x: x.strip())
#using get dummies to get dummies for cuisines
cluster_dummy = pd.get_dummies(cluster_dummy, columns=["Cuisines"], prefix=["Cuisines"])

#checking if the values are correct
# cluster_dummy.loc[:, cluster_dummy.columns.str.startswith('Cuisines_')].eq(1)[:5].T
```

```
cluster_dummy.loc[:, cluster_dummy.columns.str.startswith('Cuisines_')].idxmax(1)[:6]
```

```
#replacing cuisines_ from columns name – for better understanding run seperatly

# cluster_dummy.columns = cluster_dummy.columns.str.replace(" ","")
cluster_dummy.columns = cluster_dummy.columns.str.replace("Cuisines_","")
# cluster_dummy = cluster_dummy.groupby(cluster_dummy.columns, axis=1).sum()

#grouping each restaurant as explode created unnecessary rows
cluster_dummy = cluster_dummy.groupby("Restaurant").sum().reset_index()


#total cuisine count
hotel['Total_Cuisine_Count'] = hotel['Cuisines'].apply(lambda x : len(x.split(',')))


#adding average rating – will remove 5 unrated restaurant from 105 restaurant
avg_hotel_rating.rename(columns = {'Rating':'Average_Rating'}, inplace =True)
hotel = hotel.merge(avg_hotel_rating[['Average_Rating','Restaurant']], on = 'Restaurant')
hotel.head(1)
```

	Restaurant	Links	Cost	Cuisines	Timings	Total_Cuisine_Count	Average_Rating	
				Chinese,	12noon to			

Next steps: (Generate code with `hotel`) (👁 View recommended plots) (New interactive sheet)

```
#adding cost column to the new dataset
cluster_dummy = hotel[['Restaurant','Cost','Average_Rating','Total_Cuisine_Count'
                      ]].merge(cluster_dummy, on = 'Restaurant')


cluster_dummy.shape
```

(100, 48)

```
#creating data frame for categorial encoding
cluster_df = hotel[['Restaurant','Cuisines','Cost','Average_Rating','Total_Cuisine_Count']]


#creating new dataframe for clustering
cluster_df = pd.concat([cluster_df,pd.DataFrame(columns=list(cuisine_dict.keys()))])


#creating categorial feature for cuisine
#iterate over every row in the dataframe
for i, row in cluster_df.iterrows():
  # iterate over the new columns
  for column in list(cluster_df.columns):
      if column not in ['Restaurant','Cost','Cuisines','Average_Rating','Total_Cuisine_Count']:
        # checking if the column is in the list of cuisines available for that row
        if column in row['Cuisines']:
          #assign it as 1 else 0
          cluster_df.loc[i,column] = 1
        else:
          cluster_df.loc[i,column] = 0


#result from encoding
cluster_df.head(2).T
```

	0	1
Restaurant	Beyond Flavours	Paradise
Cuisines	Chinese, Continental, Kebab, European, South I...	Biryani, North Indian, Chinese
Cost	800.00	800.00
Average_Rating	4.28	4.70
Total_Cuisine_Count	6.00	3.00
Chinese	1	1
Continental	1	0
Kebab	1	0
European	1	0
South Indian	1	0
North Indian	1	1
Biryani	0	1
Asian	0	0
Mediterranean	0	0
Desserts	0	0
Seafood	0	0
Beverages	0	0
Goan	0	0
Healthy Food	0	0
Lebanese	0	0
American	0	0
Ice Cream	0	0
Street Food	0	0
Fast Food	0	0
BBQ	0	0
Italian	0	0
Finger Food	0	0
Burger	0	0
Japanese	0	0
Salad	0	0
Sushi	0	0
Mexican	0	0
Mughlai	0	0
Andhra	0	0
Bakery	0	0
Juices	0	0
Arabian	0	0
Hyderabadi	0	0
Cafe	0	0
Spanish	0	0
Wraps	0	0
Thai	0	0
Indonesian	0	0
Modern Indian	0	1
Momos	0	0
Pizza	0	0
North Eastern	0	0
Mithai	0	0

⌄   What all categorical encoding techniques have you used & why did you use those techniques?

I have used one hot encoding on the cuisine category and based on the cuisine if present i gave value to 1 and if absent gave value of 0.
Benefit of using one hot encoding:

- Handling categorical variables with no ordinal relationship:

  > One-hot encoding does not assume any ordinal relationship between the categories, making it suitable for categorical
  > features that do not have a natural ordering.

- Handling categorical variables with many unique values

  > One-hot encoding can handle categorical features with a high cardinality, which can be useful when there are many
  > unique categories.

- Handling categorical variables with multiple levels

  > One-hot encoding can handle categorical features with multiple levels, such as "state" and "city". This can be useful
  > when there are many unique combinations of levels.

- Handling categorical variables with missing values

  > One-hot encoding can handle missing values by creating a new category for them.

- Model interpretability

  > One-hot encoded features are easy to interpret as the encoded values are binary, thus making it easy to understand the
  > relationship between the categorical feature and the target variable.

- Compatibility with many machine learning models

  > One-hot encoded features are compatible with most machine learning models, including linear and logistic regression,
  > decision trees, and neural networks.

⌄   4. Textual Data Preprocessing - Review

(It's mandatory for textual dataset i.e., NLP, Sentiment Analysis, Text Clustering etc.)

⌄   1. Expand Contraction

```
#creating new df for text processing of sentiment analysis
sentiment_df = review[['Reviewer','Restaurant','Rating','Review']]
#analysing two random sample
sentiment_df.sample(2)
```

	Reviewer	Restaurant	Rating	Review
5518	Shafia Shams	Cafe Eclat	4.00	This place has one of the most amazing menu , ...

```
#setting index
sentiment_df = sentiment_df.reset_index()
sentiment_df['index'] = sentiment_df.index
```

```
sentiment_df.sample(2)
```

	index	Reviewer	Restaurant	Rating	Review
4632	4632	Ganesh Kumar	Ulavacharu	1.00	Ordered Nellore Chapala Pulusu and Andhra Kodi...

```
# Expand Contraction
import contractions
# applying fuction for contracting text
sentiment_df['Review']=sentiment_df['Review'].apply(lambda x:contractions.fix(x))
```

2. Lower Casing

```
# Lower Casing
sentiment_df['Review'] = sentiment_df['Review'].str.lower()
```

```
sentiment_df.head()
```

	index	Reviewer	Restaurant	Rating	Review
0	0	Rusha Chakraborty	Beyond Flavours	5.00	the ambience was good, food was quite good . h...
1	1	Anusha Tirumalaneedi	Beyond Flavours	5.00	ambience is too good for a pleasant evening. s...
2	2	Ashok Shekhawat	Beyond Flavours	5.00	a must try.. great food great ambience. thnx f...
3	3	Swapnil Sarkar	Beyond Flavours	5.00	soumen das and arun was a great guy. only beca...

Next steps: [Generate code with `sentiment_df`] [👁 View recommended plots] [New interactive sheet]

3. Removing Punctuations

```
# Remove Punctuations
import string
def remove_punctuation(text):
  '''a function for removing punctuation'''

  # replacing the punctuations with no space,
  # which in effect deletes the punctuation marks
  translator = str.maketrans('', '', string.punctuation)
  # return the text stripped of punctuation marks
  return text.translate(translator)
```

```
#remove punctuation using function created
sentiment_df['Review'] = sentiment_df['Review'].apply(remove_punctuation)
sentiment_df.sample(5)
```

	index	Reviewer	Restaurant	Rating	Review
6444	6444	Damoder Volavoju	Hyderabad Chefs	2.00	1
5104	5104	Harish Kandikatla	Hitech Bawarchi Food Zone	1.00	i ordered a chicken biryani in this restaurant...
4996	4996	Tarun Singh	Mathura Vilas	1.00	bad experience first they denied zomato gold b...
3632	3632	Arun	Banana Leaf Multicuisine Restaurant	5.00	i like the place\nwelcome drink was coca cola ...

4. Removing URLs & Removing words and digits contain digits.

```
# Remove URLs & Remove words and digits contain digits
import re

# Remove links
sentiment_df["Review"] = sentiment_df["Review"].apply(lambda x: re.sub(r"http\S+", "", x))

# Remove digits
sentiment_df["Review"] = sentiment_df["Review"].apply(lambda x: re.sub(r"\d+", "", x))


#function to extract location of the restaurant
def get_location(link):
  link_elements = link.split("/")
  return link_elements[3]

#create a location feature
hotel['Location'] = hotel['Links'].apply(get_location)
hotel.sample(2)
```

⇄	Restaurant	Links	Cost	Cuisines	Timings	Total_Cuisine_Count	Average_Rating	Locatio
92	Collage - Hyatt Hyderabad Gachibowli	https://www.zomato.com/hyderabad/collage-hyatt...	2250	Continental, Italian, North Indian, Chinese, A...	24 Hours (Mon-Sun)	5	3.41	hyderaba

⌄ 5. Removing Stopwords & Removing White spaces

```python
# Remove Stopwords
# extracting the stopwords from nltk library
sw = stopwords.words('english')


#function to remove stopwords
def delete_stopwords(text):
  '''a function for removing the stopword'''
  # removing the stop words and lowercasing the selected words
  text = [word.lower() for word in text.split() if word.lower() not in sw]
  # joining the list of words with space separator
  return " ".join(text)


#calling function to remove stopwords
sentiment_df['Review'] = sentiment_df['Review'].apply(delete_stopwords)


# Remove White spaces
sentiment_df['Review'] =sentiment_df['Review'].apply(lambda x: " ".join(x.split()))


#random sample
sentiment_df.sample(2)
```

⇄	index	Reviewer	Restaurant	Rating	Review	⊞
4301	4301	Anuj Rajani	The Foodie Monster Kitchen	4.00	tried rice bowls shawarma non veg green slimy ...	⏸

⌄ 6. Rephrase Text

- **Not using** as it was not giving result as expected.

```python
# # Rephrase Text
# from nltk.corpus import wordnet

# #function to create rephrase sentence
# def rephrase_sentence(sentence):
#     # Tokenize the sentence
#     tokens = nltk.word_tokenize(sentence)

#     # Replace each token with its synonyms
#     new_sentence = []
#     for token in tokens:
#         synonyms = wordnet.synsets(token)
#         if synonyms:
#             new_sentence.append(synonyms[0].lemmas()[0].name())
#         else:
#             new_sentence.append(token)

#     # Join the tokens back into a sentence
#     rephrased_sentence = " ".join(new_sentence)

#     return rephrased_sentence

# # apply the function to the 'Review' column
# sentiment_df['Review'] = sentiment_df['Review'].apply(rephrase_sentence)


sentiment_df.sample(2)
```

	index	Reviewer	Restaurant	Rating	Review	
5422	5422	P Arora	Asian Meal Box	4.00	mini chinese cravings place order ordered gobi...	

7. Tokenization

```
!rm -rf /root/nltk_data
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import nltk
from nltk.tokenize import word_tokenize

# Ensure required NLTK resources are downloaded
nltk.download('punkt')

# Apply tokenization
sentiment_df['Review'] = sentiment_df['Review'].apply(word_tokenize)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
<ipython-input-150-e18700fbf94a> in <cell line: 0>()
      6
      7 # Apply tokenization
----> 8 sentiment_df['Review'] = sentiment_df['Review'].apply(word_tokenize)

                        ↕ 10 frames
lib.pyx in pandas._libs.lib.map_infer()

/usr/local/lib/python3.11/dist-packages/nltk/data.py in find(resource_name, paths)
    577         sep = "*" * 70
    578         resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579         raise LookupError(resource_not_found)
    580
    581

LookupError:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

Next steps: (Explain error)

```
sentiment_df.sample(2)
```

8. Text Normalization

```
# Normalizing Text (i.e., Stemming, Lemmatization etc.)
#stemming using snowballstemmer
# from nltk.stem import SnowballStemmer

# # Create a stemmer
```

```
# stemmer = SnowballStemmer("english")

# def stem_tokens(tokens):
#     stemmed_tokens = [stemmer.stem(token) for token in tokens]
#     return stemmed_tokens

# # Stem the 'Review' column
# sentiment_df['Review'] = sentiment_df['Review'].apply(stem_tokens)


import nltk
nltk.download('wordnet')


#applying Lemmatization
from nltk.stem import WordNetLemmatizer

# Create a lemmatizer
lemmatizer = WordNetLemmatizer()

def lemmatize_tokens(tokens):
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return lemmatized_tokens

# Lemmatize the 'Review' column
sentiment_df['Review'] = sentiment_df['Review'].apply(lemmatize_tokens)
```

```
sentiment_df.sample(2)
```

⌄ Which text normalization technique have you used and why?

I have used **Lemmatization** as a text normalization technique.

Lemmatization is the process of reducing words to their base or root form, similar to stemming. However, lemmatization uses a dictionary-based approach and considers the context of the word in order to determine its base form, while stemming uses simple heuristics and does not consider the context of the word. Lemmatization is a more accurate way of finding the root form of a word as it takes into account the context of the word as well as its grammatical structure.

I have used lemmatization because it is a more accurate way of reducing words to their base form than stemming. Lemmatization considers the context of the word and its grammatical structure to determine its base form, which can help to improve the performance of natural language processing models. Lemmatization is often used in tasks such as text classification and information retrieval, where the meaning of the words is important.

Other Method for Normalization

Tokenization is the process of breaking down a sentence or a piece of text into individual words or tokens. Tokenization is an important step in natural language processing as it allows us to work with individual words rather than the entire text.

Stemming is the process of reducing words to their base or root form. This is useful in natural language processing because it allows us to reduce the dimensionality of the data by converting words to their common form. This can help improve the performance of models by reducing the number of unique words that need to be processed.

Stemming can be used because they are common normalization techniques used in natural language processing to preprocess text data before it is fed into a model. Tokenization is the first step and it breaks down the text into individual words, which is necessary for most NLP tasks. Stemming is used to reduce the dimensionality of the data by converting words to their common form, this is useful for text classification and other NLP tasks where the meaning of the words is important.

⚡ In general, stemming is a more aggressive technique that can remove more of the original word form, which may make it difficult for a lemmatizer to accurately identify the base form of the word. Additionally, some stemming algorithms may create non-real words, which are difficult for a lemmatizer to handle. Therefore, if the goal is to maintain the meaning of the text and preserve more of the original word forms, it would be more appropriate to apply lemmatization before stemming. However, if the goal is to reduce all words to their base forms and to group together different forms of the same word, it may be useful to try both ways and compare the results.

⌄ 9. Part of speech tagging

```
# sentiment_tfid = sentiment_df.copy()


# POS Taging
# sentiment_tfid['Review'] = sentiment_tfid['Review'].apply(nltk.pos_tag)
# sentiment_tfid.head()
```

Here I am **not performing POS** tagging as it was taking longer time when training.

Part-of-speech (POS) tagging can be important for sentiment analysis in some cases, as it can provide additional information about the structure and meaning of the text.

For example, certain POS tags, such as adjectives and adverbs, are often used to express sentiment. By identifying these POS tags in the text, a sentiment analysis model can gain a better understanding of the sentiment being expressed. Additionally, certain grammatical structures, such as negations or modals, can change the sentiment of a sentence. By identifying these structures through POS tagging, a sentiment analysis model can take them into account when determining the overall sentiment of the text.

However, it's worth noting that POS tagging is not always necessary for sentiment analysis. In some cases, a model may be able to achieve good performance without using POS tagging. Additionally, the complexity of a model that uses POS tagging increases, which could lead to longer training time and higher computational cost.

⌄ 10. Text Vectorization

Tfidf

```python
# Vectorizing Text
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(tokenizer=lambda x: x, lowercase=False)
vectorizer.fit(sentiment_df['Review'].values)
#creating independent variable for sentiment analysis
X_tfidf = vectorizer.transform(sentiment_df['Review'].values)
```

```
    ---------------------------------------------------------------------------
    NameError                                 Traceback (most recent call last)
    <ipython-input-1-a89706bd4495> in <cell line: 0>()
          3
          4 vectorizer = TfidfVectorizer(tokenizer=lambda x: x, lowercase=False)
    ----> 5 vectorizer.fit(sentiment_df['Review'].values)
          6 #creating independent variable for sentiment analysis
          7 X_tfidf = vectorizer.transform(sentiment_df['Review'].values)

    NameError: name 'sentiment_df' is not defined
```

Next steps: (Explain error)

Bag of Words

```python
#Bag of Words
tokenized_text = []
for token in sentiment_df['Review']:
    tokenized_text.append(token)

#creating token dict
tokens_dict = gensim.corpora.Dictionary(tokenized_text)

#print token dict
#tokens_dict.token2id


#using tokens_dict.doc2bow() to generate BoW features for each tokenized course
texts_bow = [tokens_dict.doc2bow(text) for text in tokenized_text]

#creating a new text_bow dataframe based on the extracted BoW features
tokens = []
bow_values = []
doc_indices = []
doc_ids = []
for text_idx, text_bow in enumerate(texts_bow):
    for token_index, token_bow in text_bow:
        token = tokens_dict.get(token_index)
        tokens.append(token)
        bow_values.append(token_bow)
        doc_indices.append(text_idx)
        doc_ids.append(sentiment_df["Restaurant"][text_idx])

bow_dict = {"doc_index": doc_indices,
            "doc_id": doc_ids,
            "token": tokens,
            "bow": bow_values,
            }
bows_df = pd.DataFrame(bow_dict)
bows_df.head()
```

﹀ Which text vectorization technique have you used and why?

Here I have used Tf-idf Vectorization technique.

TF-IDF (term frequency-inverse document frequency) is a technique that assigns a weight to each word in a document. It is calculated as the product of the term frequency (tf) and the inverse document frequency (idf).

The term frequency (tf) is the number of times a word appears in a document, while the inverse document frequency (idf) is a measure of how rare a word is across all documents in a collection. The intuition behind tf-idf is that words that appear frequently in a document but not in many documents across the collection are more informative and thus should be given more weight.

The mathematical formula for tf-idf is as follows:

tf-idf(t, d, D) = tf(t, d) * idf(t, D)

where t is a term (word), d is a document, D is a collection of documents, tf(t, d) is the term frequency of t in d, and idf(t, D) is the inverse document frequency of t in D.

The tf component of the weight assigns a value to a word based on how often it appears in the document, while the idf component assigns a value based on how rare the word is in the entire collection of documents. Tf-idf is commonly used in text classification and information retrieval tasks because it can help to down-weight the effect of common words and up-weight the effect of rare words which are more informative.

It also helps to reduce the dimensionality of the data and increases the weight of important words, thus providing more informative and robust feature set for the model to work on.

Text vectorization is the process of converting text data into numerical vectors that can be used as input for machine learning models.

There are several ways to vectorize text data, one of the most common methods is using Tf-idf Vectorization, other methods are bag-of-words (BoW - uses CountVectorizer), word2vec, or doc2vec model.

﹀ 4. Feature Manipulation & Selection

﹀ 1. Feature Manipulation

﹀ Restaurant

```
# Manipulate Features to minimize feature correlation and create new features
hotel.shape
```

```
# extracting Timings when hotel was opened and when was closed
#q = hotel.copy()
# import re

# def extract_timings(timings_str):
#     days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
#     timings = {}
#     for day in days:
#         match = re.search(f"{day}.*?(\\d{{1,2}}:\\d{{2}}(am|pm))", timings_str)
#         if match:
#             open_time, close_time = match.groups()
#             timings[day] = (open_time, close_time)
#     return timings
```

```
# q['timings_data'] = q['Timings'].apply(extract_timings)
```

```
#columns for dataset
hotel.columns
```

- Dropping column like link as from link location was extracted, dropping location as it does not have variability only had hyderabad as 99% values. Then dropping Cuisines column as cuisine from the feature are extracted as new feature.

```
#dropping columns
hotel = hotel.drop(columns = ['Links','Location'], axis = 1)
```

```
hotel.shape
```

```
#creating new dataframe which will be used for clustering i.e dropping the unimportant column
#this dataset was used earlier while categorial encoding hence using it for clustering
```

```
cluster_df.shape
```

```
cluster_df.sample(1)
```

```
#alternatively using other variable created earlier during categorial creation
cluster_dummy.shape
```

∨   Review Data

```
#review data shape
review.shape
```

```
#review column
review.columns
```

- Since this dataset is used for sentiment analysis, therefore using only specific feature for sentiment analysis like Restaurant, Review and Ratings.

```
#creating new binary feature called sentiment based on rating which has 1 = positive and 0 = negative
sentiment_df['Sentiment'] = sentiment_df['Rating'].apply(
    lambda x: 1 if x >=sentiment_df['Rating'].mean() else 0)  #1 = positive # 0 = negative
```

```
#sentiment data frame
sentiment_df.sample(2)
```

∨   2. Feature Selection

```
hotel.columns
```

```
#feature selcted for clustering
cluster_df.columns
```

```
cluster_dummy.columns
```

```
review.columns
```

```
#feature selected for sentiment analysis
sentiment_df.columns
```

```
# Select your features wisely to avoid overfitting
```

∨   What all feature selection methods have you used and why?

I will be usign **PCA for feature selection**, which will be again beneficial for dimensional reduction, therefore will do the needfull in the precedding step.

The goal of PCA is to identify the most important variables or features that capture the most variation in the data, and then to project the data onto a lower-dimensional space while preserving as much of the variance as possible.

∨   Which all features you found important and why?

Answer Here.

∨   5. Data Transformation

∨   Do you think that your data needs to be transformed? If yes, which transformation have you used. Explain Why?

```
#detecting outliers using zcsore method
# from scipy import stats
# z = np.abs(stats.zscore(cluster_df[['Cost']]))
#print(z)
# cluster_df=cluster_df[(z<3).all(axis=1)]
# cluster_df.shape


# Getting symmetric and skew symmetric features from the cplumns
symmetric_feature=[]
non_symmetric_feature=[]
for i in cluster_df.describe().columns:
  if abs(cluster_df[i].mean()-cluster_df[i].median())<0.1:
    symmetric_feature.append(i)
  else:
    non_symmetric_feature.append(i)

# Getting Symmetric Distributed Features
print("Symmetric Distributed Features : -",symmetric_feature)

# Getting Skew Symmetric Distributed Features
print("Skew Symmetric Distributed Features : -",non_symmetric_feature)


#using log transformation to transform Cost as using capping tends to change median and mean
cluster_df['Cost'] = np.log1p(cluster_df['Cost'])
cluster_dummy['Cost'] = np.log1p(cluster_dummy['Cost'])


# Transform Your data
for i,col in enumerate(['Cost']) :
    sns.distplot(cluster_df[col], color = '#055E85', fit = norm);
    feature = cluster_df[col]
    plt.axvline(feature.mean(), color='#ff033e', linestyle='dashed', linewidth=3,label= 'mean');  #red
    plt.axvline(feature.median(), color='#A020F0', linestyle='dashed', linewidth=3,label='median'); #cyan
    plt.legend(bbox_to_anchor = (1.0, 1))
    plt.title(f'{col.title()}');
    plt.tight_layout();
```

- Since I have applied capping method, it changes mean and median, hence trying to achieve normal distribution using log transformation which is a method for treating positive skewness.

Gaussian transformation generally used to convert data distribution into normal distribution.

6. Data Scaling

```
# Scaling your data
cluster_dummy.sample(5)


#normalizing numerical columns
numerical_cols = ['Cost','Total_Cuisine_Count','Average_Rating']
scaler = StandardScaler()
scaler.fit(cluster_dummy[numerical_cols])
scaled_df = cluster_dummy.copy()
scaled_df[numerical_cols] = scaler.transform(cluster_dummy[numerical_cols])
```

∨   Which method have you used to scale you data and why?

- Here I have used standard scaler as those numerical columns where normally distributed.

7. Dimesionality Reduction

```
# Dimensionality Reduction (If needed)
#applying pca
#setting restaurant feature as index as it still had categorial value
scaled_df.set_index(['Restaurant'],inplace=True)
features = scaled_df.columns
# features = features.drop('Restaurant')
# create an instance of PCA
pca = PCA()

# fit PCA on features
pca.fit(scaled_df[features])
```

```
#explained variance v/s no. of components
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker ='o', color = 'orange')
plt.xlabel('number of components',size = 15, color = 'red')
plt.ylabel('cumulative explained variance',size = 14, color = 'blue')
plt.title('Variance v/s No. of Components',size = 20, color = 'green')
plt.xlim([0, 8])
plt.show()


#using n_component as 3
pca = PCA(n_components=3)

# fit PCA on features
pca.fit(scaled_df[features])

# explained variance ratio of each principal component
print('Explained variation per principal component: {}'.format(pca.explained_variance_ratio_))
# variance explained by three components
print('Cumulative variance explained by 3 principal components: {:.2%}'.format(
                                        np.sum(pca.explained_variance_ratio_)))

# transform data to principal component space
df_pca = pca.transform(scaled_df[features])
```

Dimensionality reduction is the process of reducing the number of features in a dataset while preserving as much of the relevant information as possible. It is a technique used to overcome the curse of dimensionality, which refers to the problem of increased computational complexity and decreased performance of machine learning models as the number of features increases.

There are two main types of dimensionality reduction techniques: feature selection and feature extraction.

Feature selection is the process of selecting a subset of the most relevant features from the original feature set. It is a technique that helps to reduce the dimensionality of the data by removing irrelevant and redundant features. Common feature selection techniques include:

- Correlation-based feature selection
- Mutual information-based feature selection
- Recursive feature elimination
- SelectKBest

Feature extraction is the process of creating new features from the original feature set by combining or transforming the existing features. It is a technique that helps to reduce the dimensionality of the data by creating a new feature space that is more compact and informative than the original feature space. Common feature extraction techniques include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Independent Component Analysis (ICA)
- Non-Negative Matrix Factorization (NMF)
- Autoencoder

Both feature selection and feature extraction can be used to reduce the dimensionality of the data and improve the performance of machine learning models. However, the choice of technique depends on the specific task, the data, and the computational resources available.

⌄　　Do you think that dimensionality reduction is needed? Explain Why?

Yes, it is important to use dimensionality reduction techniques as dataset has 40 or more features. This is because, as the number of features increases, the computational cost of clustering algorithms also increases. In addition, high dimensionality can lead to the "curse of dimensionality", where the data becomes sparse and the clusters become harder to identify. Dimensionality reduction techniques such as PCA, t-SNE, or LLE can help reduce the number of features while maintaining the important information in the data, making it easier to cluster and interpret the results.

⌄　　Which dimensionality reduction technique have you used and why? (If dimensionality reduction done on dataset.)

I have used PCA as dimension reduction technique, because PCA (Principal Component Analysis) is a widely used dimensionality reduction technique because it is able to identify patterns in the data that are responsible for the most variation. These patterns, known as principal components, are linear combinations of the original features that are uncorrelated with each other. By using the first few principal components, which account for the majority of the variation in the data, one can effectively reduce the dimensionality of the data while maintaining most of the important information.

Another advantage of PCA is that it is a linear technique, which means it can be applied to data that have a linear relationship between features. It is also easy to interpret the results as the principal components can be thought of as new, uncorrelated features. Additionally, PCA can be used for data visualization by projecting high-dimensional data onto a 2D or 3D space for easy visualization.

When PCA is applied before k-means, it is used to reduce the dimensionality of the data by transforming the original feature space into a new feature space of uncorrelated principal components. The k-means algorithm is then applied to the transformed data, resulting in clusters that are defined in the new feature space. The advantage of this approach is that it can help to remove noise and correlated features from the data, which can make the clustering results more interpretable. However, it also means that the clusters may be harder to interpret in the original feature space.

When PCA is applied after k-means, it is used to visualize the clusters in a lower-dimensional space. The k-means algorithm is applied to the original data, resulting in clusters that are defined in the original feature space. PCA is then used to project the data into a lower-dimensional space, making it easier to visualize and interpret the clusters. The advantage of this approach is that the clusters can be easily interpreted in the original feature space. However, it may not be as effective in removing noise and correlated features from the data.

```
### 8. Data Splitting
```

```
# Split your data to train and test. Choose Splitting ratio wisely.
# for sentiment analysis using sentiment_df dataframe
X = X_tfidf #from text vectorization
y = sentiment_df['Sentiment']
```

```
sentiment_df.shape
```

```
#spliting test train
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

```
# describes info about train and test set
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

∨   What data splitting ratio have you used and why?

I have used 80:20 split which is one the most used split ratio. Since there was only 9961 data, therefore I have used more in training set.

∨   9. Handling Imbalanced Dataset

∨   Do you think the dataset is imbalanced? Explain Why.

Using class imbalance ratio (CIR) to measure data imbalance. The CIR is calculated as the ratio of the number of observations in the majority class (Nm) to the number of observations in the minority class (Nm). The CIR can be expressed as: CIR = Nm / Ns, where Nm is the number of observations in the majority class and Ns is the number of observations in the minority class.

```
#getting the value count for target class
vc = sentiment_df.Sentiment.value_counts().reset_index().rename(columns =
            {'index':'Sentiment','Sentiment':'Count'})
```

```
#defining majority and minority class value
majority_class = vc.Count[0]
minority_class = vc.Count[1]
```

```
#calculating cir value for checking class imbalance
CIR = majority_class / minority_class
CIR
```

```
# Dependant Variable Column Visualization
sentiment_df['Sentiment'].value_counts().plot(kind='pie',
                        figsize=(15,6),
                         autopct="%1.1f%%",
                         startangle=90,
                         shadow=True,
                         labels=['Positive Sentiment','Negative Sentiment'],
                         colors=['red','blue'],
                         explode=[0.01,0.02]
                         )
plt.show()
```

```
# Handling Imbalanced Dataset (If needed)
```

⌄   What technique did you use to handle the imbalance dataset and why? (If needed to be balanced)

Yes, there is imbalance in dataset with 60: 40 ratio, where 60 is the majaority class and 40 is the minority class. Even the CIR score suggest that majority class is 1.73 times greater than minority class. However it is considered as slight imbalance, therefore not performing any under or over sampling technique i.e., **not required** to treat class imabalance.

7. ML Model Implementation

ML Model – 1 Clustering

####KMeans Clustering

K-Means Clustering is an Unsupervised Learning algorithm.The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The k-means clustering algorithm mainly performs two tasks:

Determines the best value for K center points or centroids by an iterative process.

Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

ELBOW METHOD

> This method uses the concept of WCSS value. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster.

SILHOUETTE METHOD

> The silhouette coefficient or silhouette score kmeans is a measure of how similar a data point is within-cluster (cohesion) compared to other clusters (separation).

```
# ML Model – 1 Implementation
#importing kmeans
from sklearn.cluster import KMeans


#Within Cluster Sum of Squared Errors(WCSS) for different values of k
wcss=[]
for i in range(1,11):
    km=KMeans(n_clusters=i,random_state = 20)
    km.fit(df_pca)
    wcss.append(km.inertia_)


#elbow curve
plt.plot(range(1,11),wcss)
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="o")
plt.xlabel("K Value", size = 20, color = 'purple')
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS", size = 20, color = 'green')
plt.title('Elbow Curve', size = 20, color = 'blue')
plt.show()


#silhouette score
from sklearn.metrics import silhouette_score
from sklearn.metrics import silhouette_samples
from sklearn.model_selection import ParameterGrid
# candidates for the number of cluster
parameters = list(range(2,10))
#parameters
parameter_grid = ParameterGrid({'n_clusters': parameters})
best_score = -1
#visualizing Silhouette Score for individual clusters and the clusters made
for n_clusters in parameters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # 1st subplot is the silhouette plot
    # silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
```

```
        # (n_clusters+1)*10 is for inserting blank space between silhouette
        # plots of individual clusters, to demarcate them clearly.
        ax1.set_ylim([0, len(df_pca) + (n_clusters + 1) * 10])

        # Initialize the clusterer with n_clusters value and a random generator
        # seed of 10 for reproducibility.
        clusterer = KMeans(n_clusters=n_clusters, random_state=10)
        cluster_labels = clusterer.fit_predict(df_pca)

        # silhouette_score gives the average value for all the samples.
        # This gives a perspective into the density and separation of the formed
        # clusters
        silhouette_avg = silhouette_score(df_pca, cluster_labels)
        print("For n_clusters =", n_clusters,
              "average silhouette_score is :", silhouette_avg)

        # Compute the silhouette scores for each sample
        sample_silhouette_values = silhouette_samples(df_pca, cluster_labels)

        y_lower = 10
        for i in range(n_clusters):
            # Aggregate the silhouette scores for samples belonging to
            # cluster i, and sort them
            ith_cluster_silhouette_values = \
                sample_silhouette_values[cluster_labels == i]

            ith_cluster_silhouette_values.sort()

            size_cluster_i = ith_cluster_silhouette_values.shape[0]
            y_upper = y_lower + size_cluster_i

            color = cm.nipy_spectral(float(i) / n_clusters)
            ax1.fill_betweenx(np.arange(y_lower, y_upper),
                              0, ith_cluster_silhouette_values,
                              facecolor=color, edgecolor=color, alpha=0.7)

            # Label the silhouette plots with their cluster numbers at the middle
            ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

            # Compute the new y_lower for next plot
            y_lower = y_upper + 10  # 10 for the 0 samples

        ax1.set_title("silhouette plot for the various clusters.")
        ax1.set_xlabel("silhouette coefficient values")
        ax1.set_ylabel("Cluster label")

        # vertical line for average silhouette score of all the values
        ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

        ax1.set_yticks([])  # Clear the yaxis labels / ticks
        ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

        # 2nd Plot showing the actual clusters formed
        colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
        ax2.scatter(df_pca[:, 0], df_pca[:, 1], marker='.', s=30, lw=0, alpha=0.7,
                    c=colors, edgecolor='k')

        # Labeling the clusters
        centers = clusterer.cluster_centers_
        # Draw white circles at cluster centers
        ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                    c="white", alpha=1, s=200, edgecolor='k')
        #marker='$%d$' % i will give numer in cluster in 2 plot
        for i, c in enumerate(centers):
            ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                        s=50, edgecolor='k')

        ax2.set_title("visualization of the clustered data.")
        ax2.set_xlabel("Feature space for the 1st feature")
        ax2.set_ylabel("Feature space for the 2nd feature")
        plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "
                      "with n_clusters = %d" % n_clusters),
                     fontsize=14, fontweight='bold')


    #vizualizing the clusters and the datapoints in each clusters
    plt.figure(figsize = (10,6), dpi = 120)

    kmeans= KMeans(n_clusters = 5, init= 'k-means++', random_state = 42)
    kmeans.fit(df_pca)

    #predict the labels of clusters.
    label = kmeans.fit_predict(df_pca)
```

```python
#Getting unique labels
unique_labels = np.unique(label)

#plotting the results:
for i in unique_labels:
    plt.scatter(df_pca[label == i , 0] , df_pca[label == i , 1] , label = i
plt.legend()
plt.show()


#making df for pca
kmeans_pca_df = pd.DataFrame(df_pca,columns=['PC1','PC2','PC3'],index=scaled_df.index)
kmeans_pca_df["label"] = label
kmeans_pca_df.sample(2)


#joining the cluster labels to names dataframe
cluster_dummy.set_index(['Restaurant'],inplace=True)
cluster_dummy = cluster_dummy.join(kmeans_pca_df['label'])
cluster_dummy.sample(2)


#changing back cost value to original from log1p done during transformation
cluster_dummy['Cost'] = np.expm1(cluster_dummy['Cost'])
cluster_dummy.sample(2)


#creating df to store cluster data
clustering_result = cluster_dummy.copy().reset_index()
clustering_result = hotel[['Restaurant','Cuisines']].merge(clustering_result[['Restaurant','Cost',
                'Average_Rating', 'Total_Cuisine_Count','label']], on = 'Restaurant')
clustering_result.head()


# Counting content in each cluster
cluster_count = cluster_dummy['label'].value_counts().reset_index().rename(
    columns={'index':'label','label':'Total_Restaurant'}).sort_values(by='Total_Restaurant')
cluster_count


#creating new df for checkign cuising in each cluster
new_cluster_df = clustering_result.copy()
new_cluster_df['Cuisines'] = new_cluster_df['Cuisines'].str.split(',')
new_cluster_df = new_cluster_df.explode('Cuisines')
#removing extra trailing space from cuisines after exploded
new_cluster_df['Cuisines'] = new_cluster_df['Cuisines'].apply(lambda x: x.strip())
new_cluster_df.sample(5)


#printing cuisine list for each cluster
for cluster in new_cluster_df['label'].unique().tolist():
  print('Cuisine List for Cluster :', cluster,'\n')
  print(new_cluster_df[new_cluster_df["label"]== cluster]['Cuisines'].unique(),'\n')
  print('='*120)
```

Agglomerative Hierarchical Clustering

Hierarchial clustering algorithms group similar objects into groups called clusters. There are two types of hierarchical clustering algorithms:

Agglomerative — Bottom up approach. Start with many small clusters and merge them together to create bigger clusters. Divisive — Top down approach. Start with a single cluster than break it up into smaller clusters.

Agglomerative hierarchical clustering

> The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the bottom-up approach. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.