

## ✓ **Project Name** - ZOMATO RESTURANT CLUSTERING AND SENTIMENT ANALYSIS

**Project Type** - Unsupervised Machine Learning

**Contribution** - Individual

**Team Member 1** - Sonali Singh

## ✓ **Project Summary** -

Zomato is an Indian restaurant aggregator and food delivery start-up founded by Deepinder Goyal and Pankaj Chaddah in 2008. Zomato provides information, menus and user-reviews of restaurants, and also has food delivery options from partner restaurants in select cities.

India is quite famous for its diverse multi cuisine available in a large number of restaurants and hotel resorts, which is reminiscent of unity in diversity. Restaurant business in India is always evolving. More Indians are warming up to the idea of eating restaurant food whether by dining outside or getting food delivered. The growing number of restaurants in every state of India has been a motivation to inspect the data to get some insights, interesting facts and figures about the Indian food industry in each city. So, this project focuses on analysing the Zomato restaurant data for each city in India.

There are two separate files, while the columns are self explanatory. Below is a brief description:

Restaurant names and Metadata - This could help in clustering the restaurants into segments. Also the data has valuable information around cuisine and costing which can be used in cost vs. benefit analysis Restaurant reviews - Data could be used for sentiment analysis. Also the metadata of reviewers can be used for identifying the critics in the industry.

Steps that are performed:

- Importing libraries
- Loading the dataset
- Shape of dataset
- Dataset information
- Handling the duplicate values
- Handling missing values.
- Understanding the columns
- Variable description
- Data wrangling
- Data visualization
- Story telling and experimenting with charts.
- Text preprocessing,
- Latent Dirichlet Allocation
- Sentiment analysis
- Challenges faced
- Conclusion.

## ✓ **GitHub Link** -

Provide your GitHub Link here.

## ✓ **Problem Statement**

The problem statement for this project is to analyze and understand the restaurant industry in India by utilizing data from the Indian restaurant aggregator and food delivery start-up, Zomato. The project aims to gain insights into the sentiments of customer reviews, cluster Zomato restaurants into different segments, and analyze the data to make useful conclusions in the form of visualizations. The data analyzed includes information on cuisine, costing, and customer reviews. The project aims to assist customers in finding the best restaurant in their locality and aid the company in identifying areas for growth and improvement in the industry. Additionally, the project aims to use the data for sentiment analysis and identifying critics in the industry through the metadata of reviewers.

## ✓ **General Guidelines : -**

1. Well-structured, formatted, and commented code is required.
2. Exception Handling, Production Grade Code & Deployment Ready Code will be a plus. Those students will be awarded some additional credits.

The additional credits will have advantages over other students during Star Student selection.

[ Note: - Deployment Ready Code is defined as, the whole .ipynb notebook should be executable in one go without a single error logged. ]

3. Each and every logic should have proper comments.
4. You may add as many number of charts you want. Make Sure for each and every chart the following format should be answered.

# Chart visualization code

- Why did you pick the specific chart?
  - What is/are the insight(s) found from the chart?
  - Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.
5. You have to create at least 15 logical & meaningful charts having important insights.

[ Hints : - Do the Vizualization in a structured way while following "UBM" Rule.

U - Univariate Analysis,

B - Bivariate Analysis (Numerical - Categorical, Numerical - Numerical, Categorical - Categorical)

M - Multivariate Analysis ]

6. You may add more ml algorithms for model creation. Make sure for each and every algorithm, the following format should be answered.
- Explain the ML Model used and it's performance using Evaluation metric Score Chart.
  - Cross- Validation & Hyperparameter Tuning
  - Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.
  - Explain each evaluation metric's indication towards business and the business impact pf the ML model used.

## ✓ ***Let's Begin !***

### ✓ ***1. Know Your Data***

#### ✓ Import Libraries

# Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from wordcloud import WordCloud

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize, RegexpTokenizer
from nltk.stem import PorterStemmer, LancasterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from textblob import TextBlob
from IPython.display import Image
from gensim import corpora
from gensim.models import LdaModel
from gensim.utils import simple_preprocess
import gensim

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

Dataset Loading

```
# Load Dataset
from google.colab import drive
drive.mount('/content/drive')

# importing dataset
meta_df = pd.read_csv('/content/drive/MyDrive/MACHINELEARNING/Zomato Restaurant names and Metadata (2) - Copy.csv')
review_df = pd.read_csv('/content/drive/MyDrive/MACHINELEARNING/Zomato Restaurant reviews (2).csv')

meta_df=meta_df_main
```

Traceback (most recent call last)

<ipython-input-85-538789c31861> in <cell line: 1>()  
----> 1 meta\_df=meta\_df\_main

NameError: name 'meta\_df\_main' is not defined

Next steps: [Explain error](#)

Dataset First View

```
# Dataset First Look
meta_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 & Restaurant	https://www.zomato.com/hyderabad/shah-ghouse-h...	800	Late Night Restaurants	Biryani, North Indian, Chinese, Seafood, Bever...	12 Noon to 2 AM
4	Over The Moon	https://www.zomato.com/hyderabad/over-the-moon...	1,200	Best Bars & Pubs, Food Hygiene Rated Restaura...	Asian, Continental, North Indian, Chinese, Mod...	12noon to 11pm (Mon, Tue, Wed, Thu, Sun),

Next steps: [Generate code with meta\\_df](#) [View recommended plots](#) [New interactive sheet](#)

```
# 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
3	Beyond Flavours	Swapnil Sarkar	Soumen das and Arun was a great guy. Only beca...	5	1 Review , 1 Follower	5/24/2019 22:11	0
4	Beyond Flavours	Dileep	Food is good.we ordered Kodi drumsticks and ba...	5	3 Reviews , 2 Followers	5/24/2019 21:37	0

Next steps: [Generate code with review\\_df](#) [View recommended plots](#) [New interactive sheet](#)

Dataset Rows & Columns count

```
# Dataset Rows & Columns count.

print(f' We have total {meta_df.shape[0]} rows and {meta_df.shape[1]} columns.')

print(f' We have total {review_df.shape[0]} rows and {review_df.shape[1]} columns.')
```

We have total 105 rows and 6 columns.  
We have total 10000 rows and 7 columns.

Dataset Information

```
# Dataset Info
meta_df.info()
review_df.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: 5.0+ KB
<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.
```

```
meta_df.duplicated(keep='last').sum()
```

```
0
```

```
review_df.duplicated(keep='last').sum()
```

```
36
```

Missing Values/Null Values

```
# Missing Values/Null Values Count
```

```
meta_df.isnull().sum()
```

```
0
Name      0
Links     0
Cost      0
Collections 54
Cuisines  0
Timings   1
dtype: int64
```

```
review_df.isnull().sum()
```

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

dtype: int64

```
# Checking for Null values.  
  
meta_df[meta_df['Collections'].isnull()].head()
```

	Name	Links	Cost	Collections	Cuisines	Timings
7	Shah Ghouse Spl Shawarma	https://www.zomato.com/hyderabad/shah-ghouse-s...	300	NaN	Lebanese	12 Noon to 12 Midnight
15	KFC	https://www.zomato.com/hyderabad/kfc-gachibowli	500	NaN	Burger, Fast Food	11 AM to 11 PM
16	NorFest - The Dhaba	https://www.zomato.com/hyderabad/norfest-the-d...	550	NaN	North Indian	12 Noon to 10:30 PM
17	Hotel Zara Hi-Fi	https://www.zomato.com/hyderabad/hotel-zara-ga...	400	NaN	Chinese, North Indian	11:30 AM to 1 AM
23	Amul	https://www.zomato.com/hyderabad/amul-gachibowli	150	NaN	Ice Cream, Desserts	10 AM to 5 AM

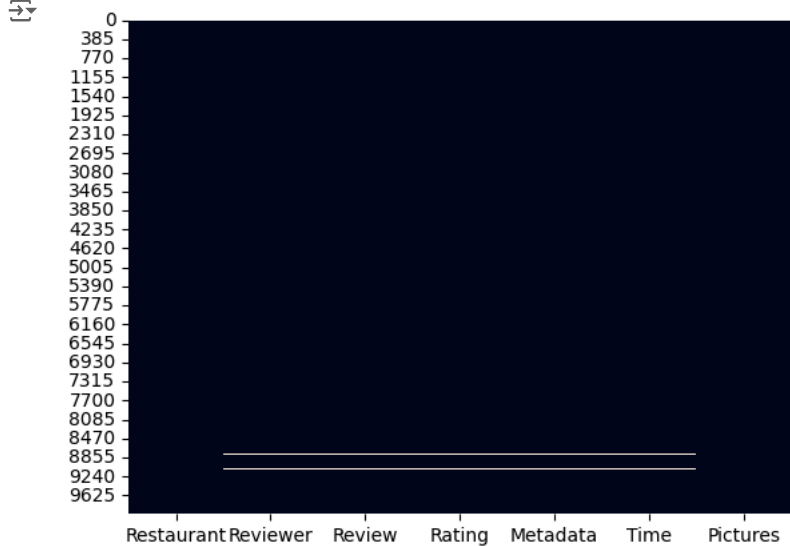
```
review_df[review_df['Metadata'].isnull()].head()
```

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures
8777	American Wild Wings	NaN	NaN	NaN	NaN	NaN	0
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

```
# Visualizing the missing values  
plt.figure(figsize=(15,5))  
sns.heatmap(meta_df.isnull(),cmap='plasma',annot=False,yticklabels=False)  
plt.title(" Visualising Missing Values");
```



```
# 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
meta_df.columns

Index(['Name', 'Links', 'Cost', 'Collections', 'Cuisines', 'Timings'], dtype='object')
```

```
review_df.columns

Index(['Restaurant', 'Reviewer', 'Review', 'Rating', 'Metadata', 'Time',
      'Pictures'],
      dtype='object')
```

```
# Dataset Describe
meta_df.describe()
```

	Name	Links	Cost	Collections	Cuisines	Timings
count	105	105	105	51	105	104
unique	105	105	29	42	92	77
top	Beyond Flavours	https://www.zomato.com/hyderabad/beyond-flavou...	500	Food Hygiene Rated Restaurants in Hyderabad	North Indian, Chinese	11 AM to 11 PM
freq	1	1	13	4	4	6

```
review_df.describe(include='all')
```

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures
count	10000	9962	9955	9962	9962	9962	10000.000000
unique	100	7446	9364	10	2477	9782	NaN
top	Beyond Flavours	Parijat Ray	good	5	1 Review	7/29/2018 20:34	NaN
freq	100	13	237	3832	919	3	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	0.748600
std	NaN	NaN	NaN	NaN	NaN	NaN	2.570381
min	NaN	NaN	NaN	NaN	NaN	NaN	0.000000
25%	NaN	NaN	NaN	NaN	NaN	NaN	0.000000
50%	NaN	NaN	NaN	NaN	NaN	NaN	0.000000
75%	NaN	NaN	NaN	NaN	NaN	NaN	0.000000
max	NaN	NaN	NaN	NaN	NaN	NaN	64.000000

Variables Description

Zomato Restaurant names and Metadata

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

Zomato Restaurant reviews

- 1. Restaurant : Name of the Restaurant
- 2. Reviewer : Name of the Reviewer
- 3. Review : Review Text
- 4. Rating : Rating Provided by Reviewer
- 5. MetaData : Reviewer Metadata - No. of Reviews and followers
- 6. Time: Date and Time of Review
- 7. Pictures : No. of pictures posted with review

Check Unique Values for each variable.

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

```
# Convert the 'Cost' column, deleting the comma and changing the data type into 'int64'.
```

```
meta_df['Cost'] = meta_df['Cost'].str.replace(",","").astype('int64')
```

Convert the 'Cost' column, deleting the comma and changing the data type into 'int64'

```
# Dataset Info.
```

```
meta_df.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    int64
 3   Collections     51 non-null     object
 4   Cuisines        105 non-null    object
 5   Timings         104 non-null    object
dtypes: int64(1), object(5)
memory usage: 5.0+ KB
```

```
review_df.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
```

## ✓ 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 further 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

```
# Chart - 1 visualization code
```

```
top10_res_by_cost = meta_df[['Name', 'Cost']].groupby('Name', as_index=False).sum().sort_values(by='Cost', ascending=False).head(10)
```





```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
```


```
# Extracting the stopwords from nltk library for English corpus.
sw = stopwords.words('english')

# Creating a function for removing stopwords.
def stopwords(text):
    '''a function for removing the stopword'''

    # removing the stop words and lowercasing the selected words
    text = [word.lower() for word in str(text).split() if word.lower() not in sw]

    # joining the list of words with space separator
    return " ".join(text)

# Removing stopwords from Cuisines.
meta_df['Cuisines'] = meta_df['Cuisines'].apply(lambda text: stopwords(text))
meta_df['Cuisines'].head()
```



	Cuisines
0	chinese, continental, kebab, european, south i...
1	biryani, north indian, chinese
2	asian, mediterranean, north indian, desserts
3	biryani, north indian, chinese, seafood, bever...
4	asian, continental, north indian, chinese, med...


**dtype:** object

Punctuations present in the text are removed successfully

```
# Cleaning and removing Numbers.
import re

# Writing a function to remove repeating characters.
def cleaning_repeating_char(text):
    return re.sub(r'(.+)1+', r'1', text)

# Removing repeating characters from Cuisines.
meta_df['Cuisines'] = meta_df['Cuisines'].apply(lambda x: cleaning_repeating_char(x))
meta_df['Cuisines'].head()
```



	Cuisines
0	chinese, continental, kebab, european, south i...
1	biryani, north indian, chinese
2	asian, mediterranean, north indian, desserts
3	biryani, north indian, chinese, seafood, bever...
4	asian, continental, north indian, chinese, med...

**dtype:** object

Removed repeated characters successfully

```
# Removing the Numbers from the data.
def cleaning_numbers(data):
    return re.sub('[0-9]+', '', data)

# Implementing the cleaning.
meta_df['Cuisines'] = meta_df['Cuisines'].apply(lambda x: cleaning_numbers(x))
meta_df['Cuisines'].head()
```



Cuisines

```
0  chinese, continental, kebab, european, south i...
1      biryani, north indian, chinese
2      asian, mediterranean, north indian, desserts
3  biryani, north indian, chinese, seafood, bever...
4  asian, continental, north indian, chinese, med...
```

**dtype:** object

We dont want numbers in the text Hence removed number successfully

```
# Top 20 Two word Frequencies of Cuisines.
from collections import Counter
text = ' '.join(meta_df['Cuisines'])

# separating each word from the sentences
words = text.split()

# Extracting the first word from the number for cuisines in the sentence.
two_words = { ' '.join(words):n for words,n in Counter(zip(words, words[1:])).items() if not words[0][-1]=='('}

# Extracting the most frequent cuisine present in the collection.
# Counting a frequency for cuisines.
two_words_dfc = pd.DataFrame(two_words.items(), columns=['Cuisine Words', 'Frequency'])

# Sorting the most frequent cuisine at the top and order by descending
two_words_dfc = two_words_dfc.sort_values(by = "Frequency", ascending = False)

# selecting first top 20 frequent cuisine.
two_words_20c = two_words_dfc[:20]
two_words_20c
```



	Cuisine Words	Frequency	
3	north indian,	51	
1	north indian	10	
17	fast food	9	
0	south indian,	6	
20	chinese north	6	
47	fast food,	6	
26	indian north	4	
24	food north	3	
10	south indian	3	
28	biryani north	3	
52	chinese chinese,	3	
41	asian north	2	
25	indian chinese,	2	
44	asian cafe,	2	
66	sushi north	2	
65	momos north	2	
48	desserts cafe,	2	
31	desserts north	2	
27	continental north	2	
14	ice cream,	2	

Next steps: [Generate code with two\\_words\\_20c](#)

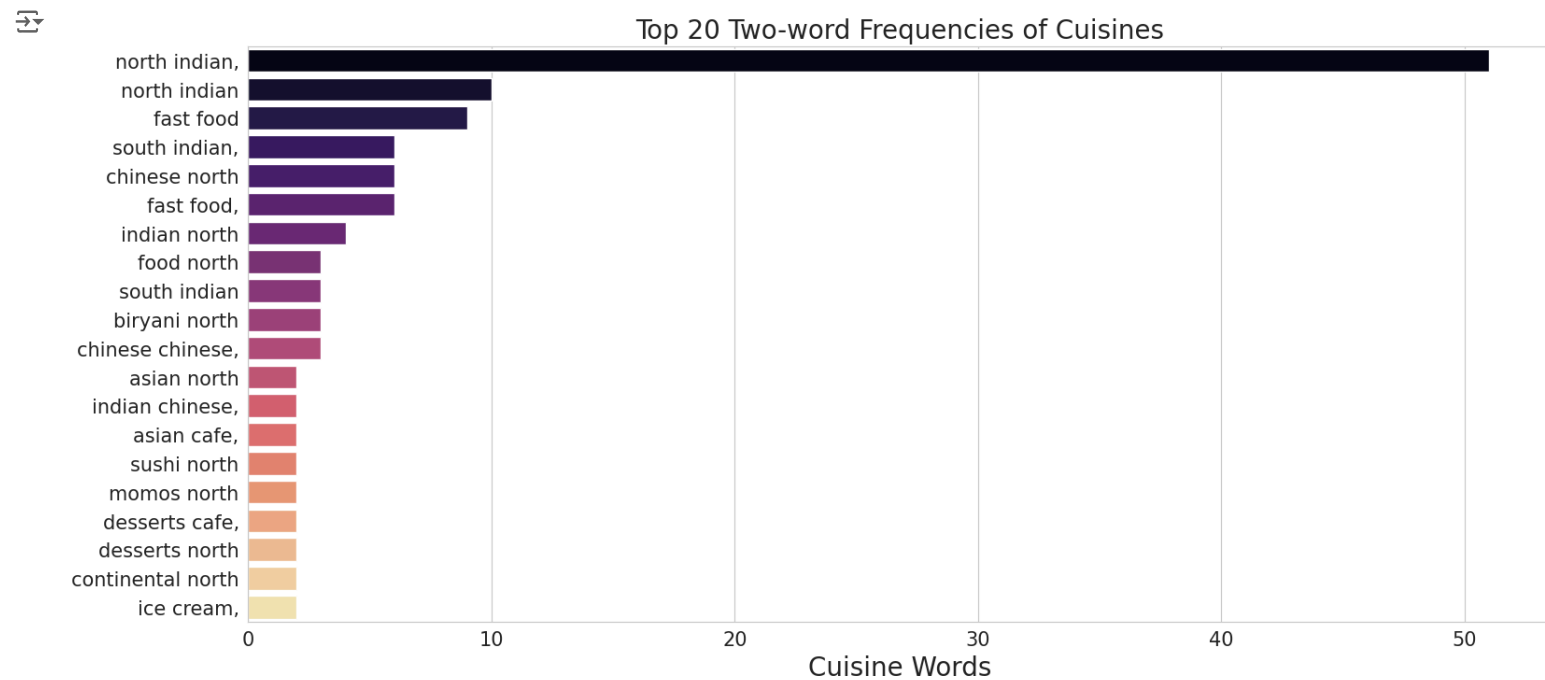
[View recommended plots](#)

[New interactive sheet](#)

Chart - 4

```
# Chart - 4 visualization code
# Visualizing the frequency of the Cuisines.

sns.set_style("whitegrid")
plt.figure(figsize = (18, 8))
sns.barplot(y = "Cuisine Words", x = "Frequency", data = two_words_20c, palette = "magma")
plt.title("Top 20 Two-word Frequencies of Cuisines", size = 20)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel("Cuisine Words", size = 20)
plt.ylabel(None)
plt.savefig("Top_20_Two-word_Frequencies_of_Cuisines.png")
plt.show()
```



> 1. Why did you pick the specific chart?

↳ 1 cell hidden

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

The DataFrame contains two columns: "Cuisine Words" and "Frequency." The "Cuisine Words" column lists the most frequent two-word cuisine terms, while the "Frequency" column shows the number of times each two-word cuisine term appears in the dataset. This information can be helpful in understanding the most common cuisine types in the dataset. It can also be used to identify trends and patterns in the types of cuisines that are popular or in demand among the customers.

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

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

Answer Here

```
# proportion or percentage of occurrences for each unique value in the Rating column.
review_df['Rating'].value_counts(normalize=True)
```



proportion

Rating	
5	0.384662
4	0.238205
1	0.174162
3	0.119755
2	0.068661
4.5	0.006926
3.5	0.004718
2.5	0.001907
1.5	0.000903
Like	0.000100

dtype: float64

```
# Removing like value and taking the mean in the rating column.
review_df.loc[review_df['Rating'] == 'Like'] = np.nan
```

```
# Changing the data type of rating column
review_df['Rating']= review_df['Rating'].astype('float64')
```

```
print(review_df['Rating'].mean())
```



3.601044071880333

```
# Filling mean in place of null value
review_df['Rating'].fillna(3.6, inplace=True)
```

```
# Changing the data type of review column.
review_df['Review'] = review_df['Review'].astype(str)
```

```
# Creating a review_length column to check the frequency of each rating.
review_df['Review_length'] = review_df['Review'].apply(len)
```

```
review_df['Rating'].value_counts(normalize=True)
```



proportion

Rating	
5.0	0.3832
4.0	0.2373
1.0	0.1735
3.0	0.1193
2.0	0.0684
4.5	0.0069
3.5	0.0047
3.6	0.0039
2.5	0.0019
1.5	0.0009

dtype: float64

The Ratings distribution 38% reviews are 5 rated,23% are 4 rated stating that people do rate good food high.

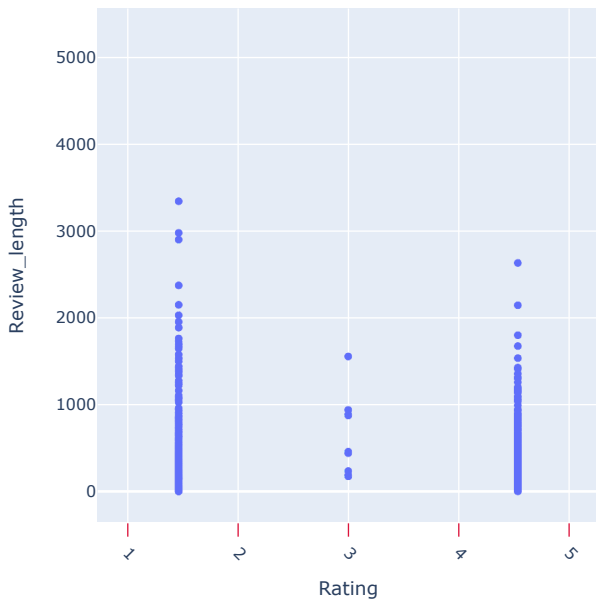
Chart - 5

```
# Chart - 5 visualization code
# Visualizing the rating column against the review length.
# Polting the frequency of the rating on scatter bar plot
```

```
import plotly.express as px
fig = px.scatter(review_df, x=review_df['Rating'], y=review_df['Review_length'])
fig.update_layout(title_text="Rating vs Review Length")
fig.update_xaxes(ticks="outside", tickwidth=1, tickcolor='crimson', tickangle=45, ticklen=10)
fig.show()
```



Rating vs Review Length



✓ 1. Why did you pick the specific chart?

The scatter plot confirms that length of review doesnt impact ratings.

Answer Here

✓ Chart - 6

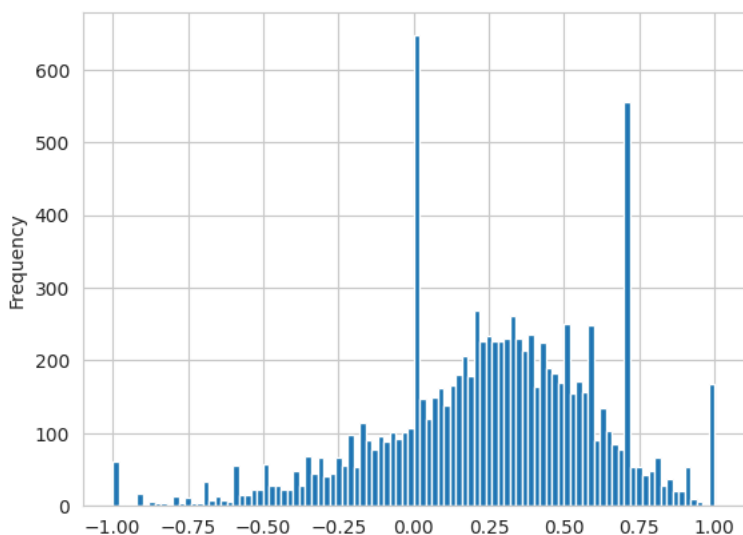
```
# Chart - 6 visualization code
```

```
# Creating polarity variable to see sentiments in reviews.(using textblob)
from textblob import TextBlob
review_df['Polarity'] = review_df['Review'].apply(lambda x: TextBlob(x).sentiment.polarity)
```

```
# Visualizing the polarity using histogram.
review_df['Polarity'].plot(kind='hist', bins=100)
```



<Axes: ylabel='Frequency'>



Answer Here.

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

Polarity is float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0,1].

### Removing Stop words

Stop words are used in a language to removed from text data during natural language processing. This helps to reduce the dimensionality of the feature space and focus on the more important words in the text.

```
# Importing dependancies and removing stopwords.
```

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
# Creating argument for stop words.
stop_words = stopwords.words('english')
```

```
print(stop_words)
rest_word=['order', 'restaurant', 'taste', 'ordered', 'good', 'food', 'table', 'place', 'one', 'also']
rest_word
```

```
⌕ ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselv
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
['order',
 'restaurant',
 'taste',
 'ordered',
 'good',
 'food',
 'table',
 'place',
 'one',
 'also']
```

## Chart - 7

```
# Chart - 7 visualization code
```

```
# We will extrapolate the 15 profiles that have made more reviews.
```

```
# Groupby on the basis of rivewer gives the fequency of the reviews
reviewer_list = review_df.groupby('Reviewer').apply(lambda x: x['Reviewer'].count()).reset_index(name='Review_Count')
```

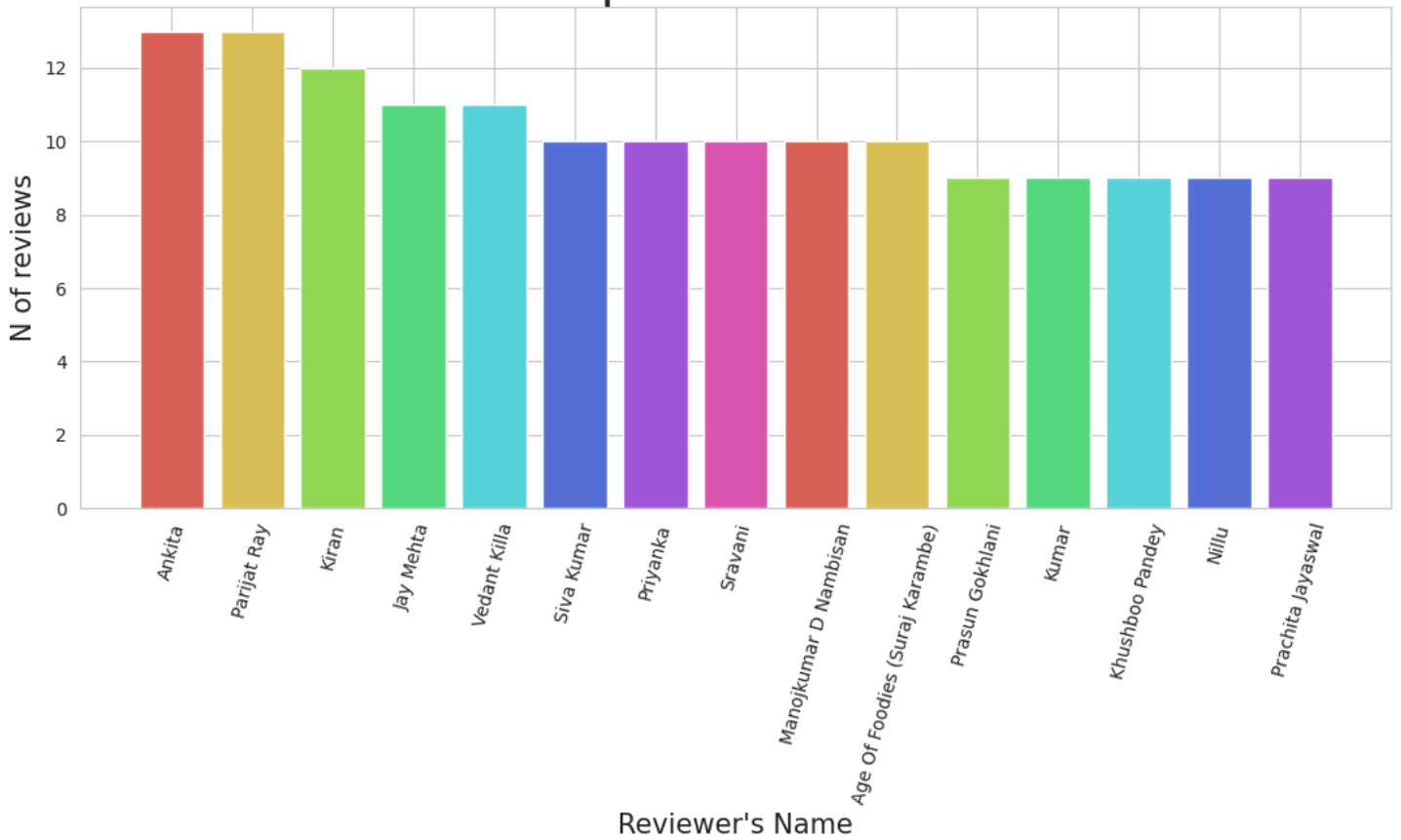
```
# Sorting the frequency of reviews decending
reviewer_list = reviewer_list.sort_values(by = 'Review_Count',ascending=False)
```

```
# Selecting the top 15 reviewrs
top_reviewers = reviewer_list[:15]
```

```
# Visualizing the top 15 reviewers.
plt.figure(figsize=(13,5))
plt.bar(top_reviewers['Reviewer'], top_reviewers['Review_Count'], color = sns.color_palette("hls", 8))
plt.xticks(rotation=75)
plt.title('Top 15 reviews',size=28)
plt.xlabel("Reviewer's Name",size=15)
plt.ylabel('N of reviews',size=15)
```

Text(0, 0.5, 'N of reviews')

## Top 15 reviews



✓ 1. Why did you pick the specific chart?

Answer Here.

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

The output of top 15 reviewers based on the number of reviews they have made in a given dataset. Analyzing the reviews made by these top reviewers can help in improving customer satisfaction and loyalty, ultimately leading to increased revenue and growth

✓ Chart - 8

# Chart - 8 visualization code

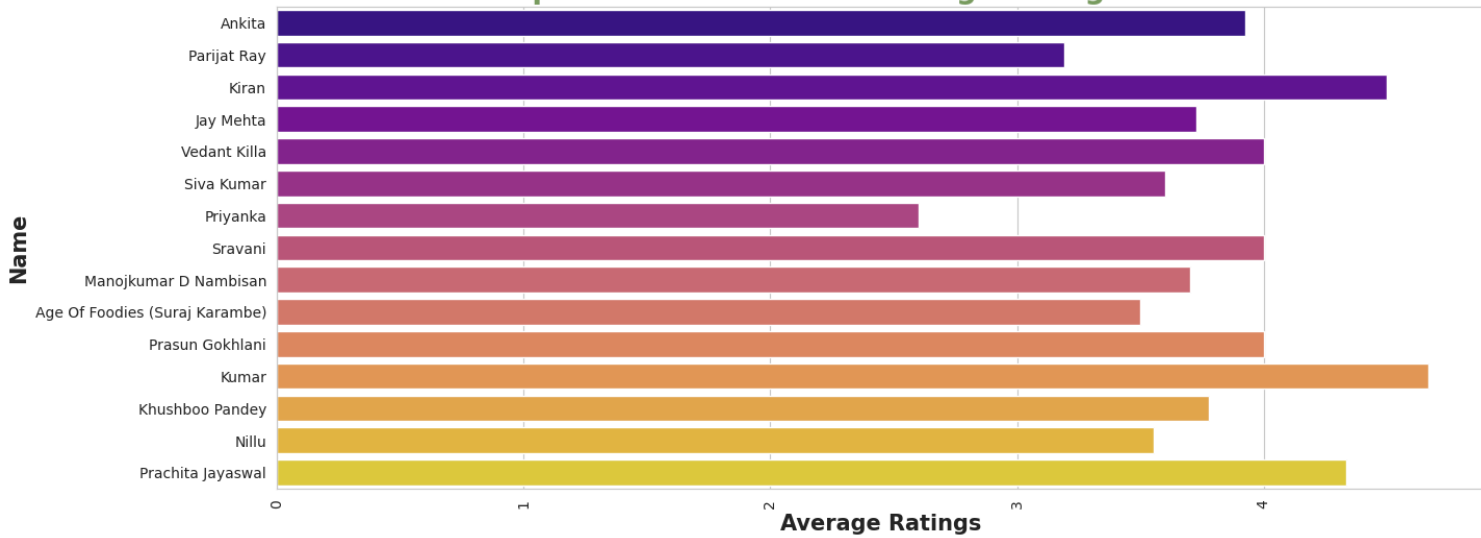
```
# Calculate the average of their ratings review.
review_ratings=review_df.groupby('Reviewer').apply(lambda x:np.average(x['Rating'])).reset_index(name='Average_Ratings')
review_ratings=pd.merge(top_reviewers,review_ratings,how='inner',left_on='Reviewer',right_on='Reviewer')
top_reviewers_ratings=review_ratings[:15]
```

```
# Average rating of top reviewers.
plt.figure(figsize=(15,6))
x = top_reviewers_ratings['Average_Ratings']
y = top_reviewers_ratings['Reviewer']
plt.title("Top 15 reviewers with average rating of review",fontsize=20, weight='bold',color=sns.cubehelix_palette(8, start=.5, rot=90)[-5])
plt.ylabel("Name",weight='bold',fontsize=15)
plt.xlabel("Average Ratings",weight='bold',fontsize=15)
plt.xticks(rotation=90)
sns.barplot(x=x, y=y,palette='plasma')
plt.show()
```





### Top 15 reviewers with average rating of review



▼ 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.

▼ Chart - 9

```
# Chart - 9 visualization code
```

```
# Removing Special characters and punctuation from review columns.
```

```
import re
review_df['Review']=review_df['Review'].map(lambda x: re.sub('[,\.\!?', '', x))
review_df['Review']=review_df['Review'].map(lambda x: x.lower())
review_df['Review']=review_df['Review'].map(lambda x: x.split())
review_df['Review']=review_df['Review'].apply(lambda x: [item for item in x if item not in stop_words])
review_df['Review']=review_df['Review'].apply(lambda x: [item for item in x if item not in rest_word])
```

```
# Word cloud for positive reviews.
```

```
from wordcloud import WordCloud
review_df['Review']=review_df['Review'].astype(str)
```

```
ps = PorterStemmer()
review_df['Review']=review_df['Review'].map(lambda x: ps.stem(x))
long_string = ', '.join(list(review_df['Review'].values))
long_string
wordcloud = WordCloud(background_color="white", max_words=100, contour_width=3, contour_color='steelblue')
wordcloud.generate(long_string)
wordcloud.to_image()
```



- ✓ 1. Why did you pick the specific chart?

Service, taste, time, starters are key to good review.

▼ Chart - 10



Plotting the top 10 most occurring words. Topic modeling is a process to automatically identify topics present in a text object and to assign text corpus to one category of topic.

# Assume that documents is a list of strings representing text documents

```
# Tokenize the documents
tokenized_docs = [simple_preprocess(doc) for doc in review_df['Review']]

# Create a dictionary from the tokenized documents
dictionary = corpora.Dictionary(tokenized_docs)

# Convert the tokenized documents to a bag-of-words corpus
bow_corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]

# Train an LDA model on the bag-of-words corpus
num_topics = 10 # The number of topics to extract
lda_model = LdaModel(bow_corpus, num_topics=num_topics, id2word=dictionary, passes=10)

# Print the topics and their top 10 terms
for topic in lda_model.show_topics(num_topics=num_topics, num_words=10, formatted=False):
    print('Topic {}: {}'.format(topic[0], ' '.join([term[0] for term in topic[1]])))
```

```
Topic 0: quantity, less, delivered, price, nan, time, main, ok, course, la
Topic 1: veg, ambience, service, non, menu, items, starters, burger, chinese, main
Topic 2: indian, north, lime, hyderabad, club, day, well, litti, banana, thali
Topic 3: service, time, even, worst, bad, delivery, zomato, experience, quality, us
Topic 4: coffee, cake, cafe, bread, cakes, old, bakery, cupcakes, tea, sandwich
Topic 5: ambience, service, music, great, best, awesome, drinks, amazing, nice, night
Topic 6: service, great, staff, nice, ambience, really, visit, time, buffet, experience
Topic 7: nice, bad, packing, chai, irani, excellent, biscuits, breakfast, arena, gobi
Topic 8: pizza, cheese, chocolate, cream, ice, try, wings, sauce, like, best
Topic 9: chicken, biryani, rice, paneer, fried, spicy, veg, tikka, noodles, try
```

pip install pyLDAvis

```
Collecting pyLDAvis
  Downloading pyLDAvis-3.4.1-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: numpy>=1.24.2 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.13.1)
Requirement already satisfied: pandas>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.1.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.4.2)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (3.1.4)
Requirement already satisfied: Numexpr in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.10.1)
Collecting funcy (from pyLDAvis)
  Downloading funcy-2.0-py2.py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.3.2)
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (4.3.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (71.0.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->pyLDAvis) (3.5.0)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from gensim->pyLDAvis) (7.0.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2->pyLDAvis) (2.1.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=2.0.0->pyLDAvis) (1.16.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.10/dist-packages (from smart-open>=1.8.1->gensim->pyLDAvis) (1.16.0)
Downloading pyLDAvis-3.4.1-py3-none-any.whl (2.6 MB)
2.6/2.6 MB 38.4 MB/s eta 0:00:00
Downloading funcy-2.0-py2.py3-none-any.whl (30 kB)
Installing collected packages: funcy, pyLDAvis
Successfully installed funcy-2.0 pyLDAvis-3.4.1
```

```
import gensim
import pyLDAvis.gensim
import pyLDAvis.sklearn
pyLDAvis.enable_notebook()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
```

`should\_run\_async` will not call `transform\_cell` automatically in the future. Please pass the result to `transformed\_cell` argument and any excep

```
-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-63-a140c1e9839b> in <cell line: 3>()
      1 import gensim
      2 import pyLDAvis.gensim
----> 3 import pyLDAvis.sklearn
      4 pyLDAvis.enable_notebook()
```

ModuleNotFoundError: No module named 'pyLDAvis.sklearn'

NOTE: If your import is failing due to a missing package, you can manually install dependencies using either `!pip` or `!apt`.

To view examples of installing some common dependencies, click the "Open Examples" button below.

OPEN EXAMPLES

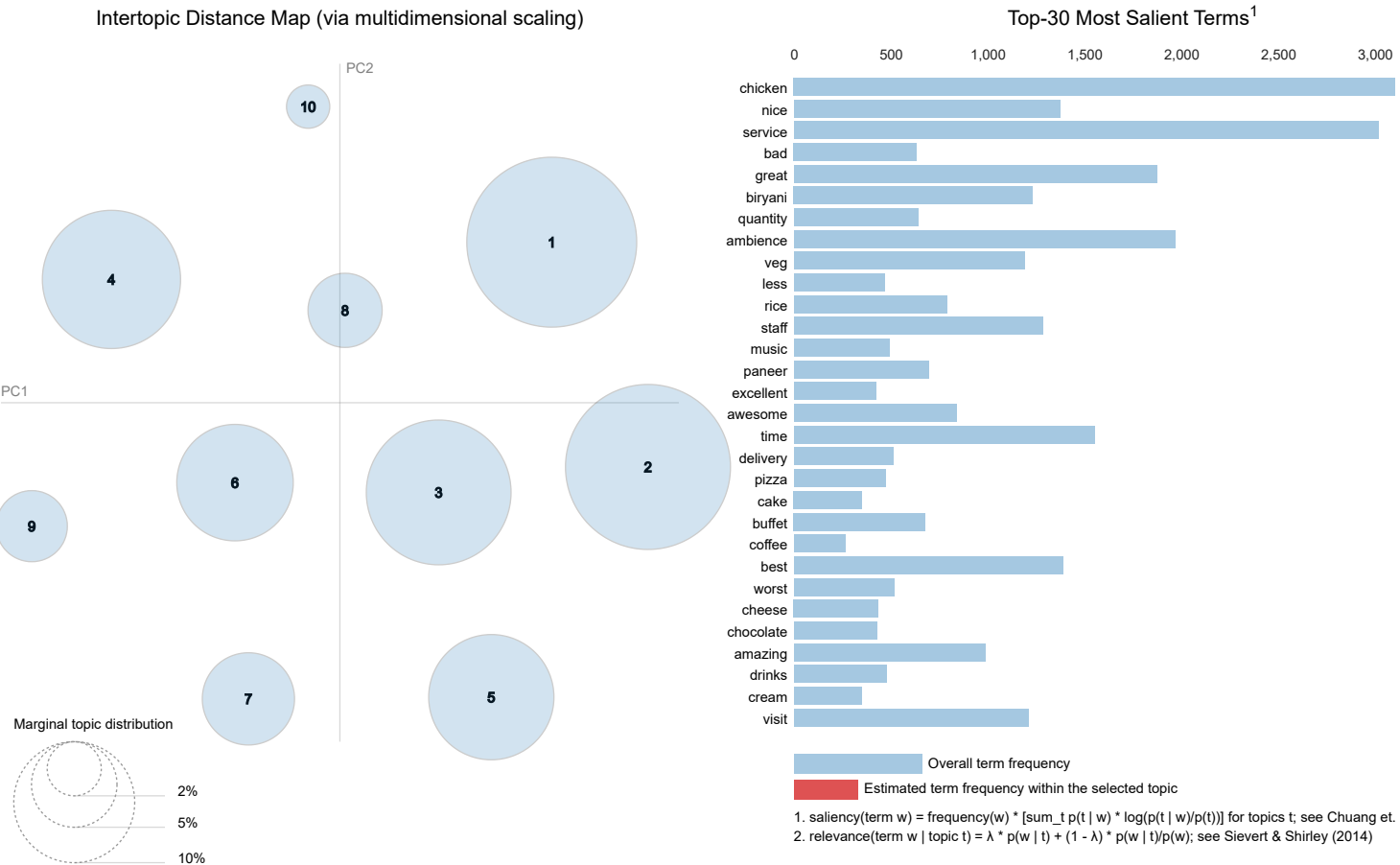
Next steps: [Explain error](#)

```
lda_visualization = pyLDAvis.gensim.prepare(lda_model, bow_corpus, dictionary, mds='tsne')
pyLDAvis.display(lda_visualization)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any excep
```

Selected Topic:  [Previous Topic](#) [Next Topic](#) [Clear Topic](#)

Slide to adjust relevance metric:<sup>(2)</sup>   $\lambda = 1$



The topics and topic terms can be visualised to help assess how interpretable the topic model is.

Sentiment Analysis

```
from textblob import TextBlob
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import plotly.express as px
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any excep
```

```
# Create a function to get the subjectivity
def subjectivity(text):
    return TextBlob(text).sentiment.subjectivity
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any excep
```

```
# Create a function to get the subjectivity
def subjectivity(text):
    return TextBlob(text).sentiment.subjectivity
```

 /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should\_run\_async` will not call `transform\_cell` automatically in the future. Please pass the result to `transformed\_cell` argument and any excep

```
# Applying subjectivity and the polarity function to the respective columns
review_df['Subjectivity'] = review_df['Review'].apply(subjectivity)
review_df['Polarity'] = review_df['Review'].apply(polarity)
```



```
-----
NameError                                Traceback (most recent call last)
<ipython-input-72-8442c8306925> in <cell line: 3>()
      1 # Applying subjectivity and the polarity function to the respective columns
      2 review_df['Subjectivity'] = review_df['Review'].apply(subjectivity)
----> 3 review_df['Polarity'] = review_df['Review'].apply(polarity)

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

Next steps: [Explain error](#)

```
# Checking for created columns
review_df['Polarity']
```



	Polarity
0	0.660000
1	0.606667
2	0.540000
3	0.500000
4	0.577500
...	...
9995	0.292500
9996	0.186174
9997	0.124535
9998	0.470000
9999	0.079861

10000 rows × 1 columns

dtype: float64

```
# Checking for created columns
review_df['Subjectivity']
```



	Subjectivity
0	0.900000
1	0.966667
2	0.740000
3	0.750000
4	0.450000
...	...
9995	0.646591
9996	0.710606
9997	0.501252
9998	0.620000
9999	0.630303

10000 rows × 1 columns

dtype: float64

```
fig.show()
```



```
# Plot for postive reviews
def build_corpus(data):
    "Creates a list of lists containing words from each sentence"
    corpus = []
    for col in ['Review']:
        for sentence in data[col].iteritems():
            word_list = sentence[1].split(" ")
            corpus.append(word_list)

    return corpus

# Display the first two elements of the corpus list
corpus = build_corpus(pos_rev)
corpus[0:2]
```

## ✓ 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. ( $H_0: \beta_1 = 0$ )
- Alternative hypothesis: There is a positive relationship between the cost of a restaurant and the rating it receives. ( $H_1: \beta_1 > 0$ )
- Test : Simple Linear Regression Analysis

#### ✓ 2. Perform an appropriate statistical test.

```
# Perform Statistical Test to obtain P-Value

# 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.")
```

#### ✓ 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 has no effect on the rating of a restaurant. ( $H_0: \beta_1 = 0$ )
- Alternative hypothesis: Alternative Hypothesis: The number of followers a reviewer has has a positive effect on the rating of a restaurant. ( $H_1: \beta_1 > 0$ )
- Test : Simple Linear Regression test

▼ 2. Perform an appropriate statistical test.

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

↳ 1 cell hidden

➤ Why did you choose the specific statistical test?

↳ 1 cell hidden

▼ 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.

↳ 1 cell hidden

➤ 2. Perform an appropriate statistical test.

[ ] ↳ 6 cells hidden

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

➤ 1. Handling Missing Values

[ ] ↳ 20 cells hidden

➤ 2. Handling Outliers

[ ] ↳ 17 cells hidden

▼ 3. Categorical Encoding

```

# Encode your categorical columns
# Encode your categorical columns

#categorical encoding using pd.getdummies
#new df with important categories
cluster_dummy = hotel[['Restaurant','Cuisines']]
#splitting cuisines as they are separated with comma and converting into list
cluster_dummy['Cuisines'] = cluster_dummy['Cuisines'].str.split(',')
#using explode converting 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()

#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)

#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)

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

```

Start coding or [generate](#) with AI.

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

↳ 1 cell hidden

> 4. Textual Data Preprocessing

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

[ ] ↳ 25 cells hidden

✓ 4. Feature Manipulation & Selection

> 1. Feature Manipulation

[ ] ↳ 1 cell hidden

> 2. Feature Selection

[ ] ↳ 5 cells hidden

✓ 5. Data Transformation

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

[ ] ↳ 1 cell hidden

✓ 6. Data Scaling

# Scaling your data

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

## 7. Dimesionality Reduction

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

[ ] ↳ 2 cells hidden

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

↳ 1 cell hidden

## 8. Data Splitting

# Split your data to train and test. Choose Splitting ratio wisely.

> What data splitting ratio have you used and why?

↳ 1 cell hidden

## 9. Handling Imbalanced Dataset

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

[ ] ↳ 2 cells hidden

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

↳ 1 cell hidden

## 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).

```
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
```

```
# Create a list of inertia scores for different numbers of clusters
scores = [KMeans(n_clusters=i+2, random_state=11).fit(df_cluster.drop('Name',axis=1)).inertia_
           for i in range(8)]
```

```
# Create a line plot of inertia scores versus number of clusters
plt.figure(figsize=(7,7))
sns.lineplot(x=np.arange(2, 10), y=scores)
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Inertia of k-Means versus number of clusters')
plt.show()
```



```
-----
NameError                                Traceback (most recent call last)
<ipython-input-87-eb92e04deeb5> in <cell line: 2>()
      1 # Create a list of inertia scores for different numbers of clusters
----> 2 scores = [KMeans(n_clusters=i+2, random_state=11).fit(df_cluster.drop('Name',axis=1)).inertia_
      3             for i in range(8)]
      4
      5 # Create a line plot of inertia scores versus number of clusters

<ipython-input-87-eb92e04deeb5> in <listcomp>(.0)
      1 # Create a list of inertia scores for different numbers of clusters
----> 2 scores = [KMeans(n_clusters=i+2, random_state=11).fit(df_cluster.drop('Name',axis=1)).inertia_
      3             for i in range(8)]
      4
      5 # Create a line plot of inertia scores versus number of clusters

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

Next steps: [Explain error](#)

```
# 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_)

# Fit the Algorithm

# Predict on the model
```



```
-----
NameError                                Traceback (most recent call last)
<ipython-input-88-5d786d392100> in <cell line: 7>()
      7 for i in range(1,11):
      8     km=KMeans(n_clusters=i,random_state = 20)
----> 9     km.fit(df_pca)
     10     wcss.append(km.inertia_)
     11

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

Next steps: [Explain error](#)

```
#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()
```



ValueError Traceback (most recent call last)

<ipython-input-89-3e72f17b83bc> in <cell line: 2>()

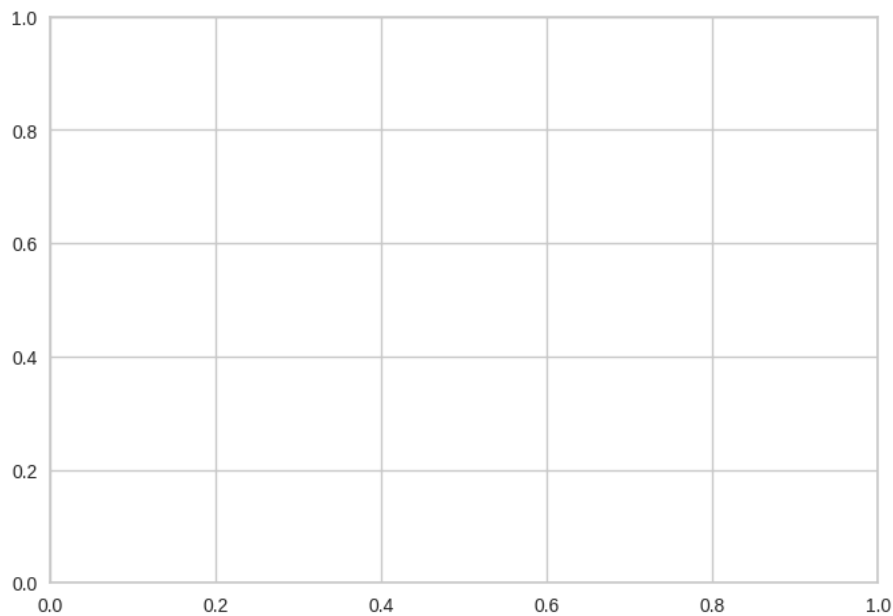
```
1 #elbow curve
----> 2 plt.plot(range(1,11),wcss)
3 plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="o")
4 plt.xlabel("K Value", size = 20, color = 'purple')
5 plt.xticks(np.arange(1,11,1))
```

3 frames

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/\_base.py in \_plot\_args(self, tup, kwargs, return\_kwargs, ambiguous\_fmt\_datakey)

```
502
503     if x.shape[0] != y.shape[0]:
--> 504         raise ValueError(f"x and y must have same first dimension, but "
505                           f"have shapes {x.shape} and {y.shape}")
506     if x.ndim > 2 or y.ndim > 2:
```

ValueError: x and y must have same first dimension, but have shapes (10,) and (0,)



Next steps:

[Explain error](#)

```

#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_xlabel("Feature space for the 2nd feature")
#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)
#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'])
clustering_result = hotel[['Restaurant', 'Cuisines']].merge(cluster_dummy[['Restaurant', 'Cost',
cluster_dummy.sample(2)

#creating df to store cluster data
clustering_result = cluster_dummy.copy().reset_index()

```

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