Project Name - Zomato Restaurant Clustering and Sentiment Analysis

Project Type - Unsupervised Machine Learning

Contribution - Individual

Name - Sathwik S

Project Summary -

In this project, I have worked with Zomato restaurant data to find hidden patterns in restaurant types and customer reviews. I used machine learning to group similar restaurants together (clustering), and natural language processing (NLP) to check how people feel about their experiences (sentiment analysis).

This helps in understanding what kind of restaurants are getting positive feedback and what features might influence that. The final goal is to combine both insights to make meaningful conclusions.

GitHub Link -

https://github.com/sathwik0404/Zomato-Restaurant-ML-Project

Problem Statement

The main challenges I aim to solve in this project:

- Group restaurants into similar categories based on features like rating, cost, and type.
- Analyze customer reviews to find whether people are generally happy or not.
- Combine both these analyses to get deeper insights into what customers prefer and why.

This can help restaurant owners, food platforms, or even customers to make better decisions.

General Guidelines : -

↳ 1 cell hidden

Let's Begin !

1. Know Your Data

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Dataset Loading

```
# Load metadata dataset
restaurant_df = pd.read_csv("/Zomato Restaurant names and Metadata.csv")
```

```
# Load reviews dataset
review_df = pd.read_csv("/Zomato Restaurant reviews.csv")
```

Start coding or generate with AI.

Dataset First View

```
print("Restaurant Metadata Preview:")
display(restaurant_df.head())

print("\nReviews Dataset Preview:")
display(review_df.head())
```

Dataset Rows & Columns count

```
# Get the shape of the datasets
print("Restaurant Metadata Shape:", restaurant_df.shape)
print("Reviews Dataset Shape:", review_df.shape)
```

```
Restaurant Metadata Shape: (105, 6)
Reviews Dataset Shape: (10000, 7)
```

Dataset Information

```
print("Restaurant Metadata Info:")
restaurant_df.info()

print("\nReviews Dataset Info:")
review_df.info()
```

```
Restaurant Metadata 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

Reviews Dataset 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

```
# Checking for Duplicate Entries
print("Duplicate Entries in Restaurant Dataset:", restaurant_df.duplicated().sum())
print("Duplicate Entries in Reviews Dataset:", review_df.duplicated().sum())
```

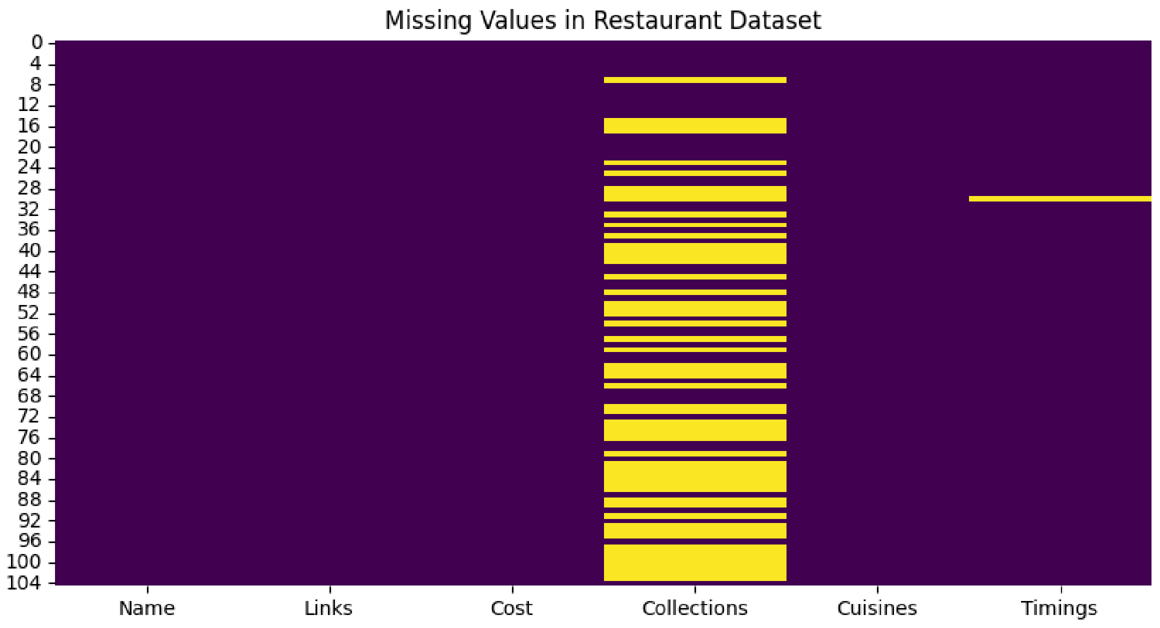
```
Duplicate Entries in Restaurant Dataset: 0
Duplicate Entries in Reviews Dataset: 36
```

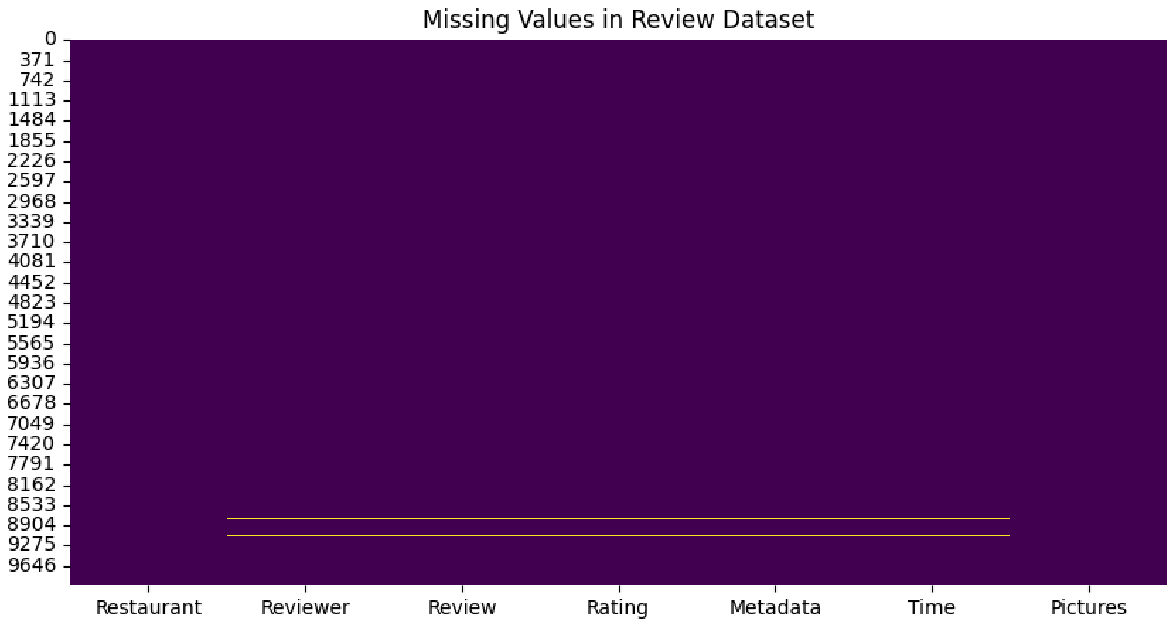
Missing Values/Null Values

```
# Checking for Missing/Null Values
print("Missing Values in Restaurant Dataset:")
print(restaurant_df.isnull().sum())

print("\nMissing Values in Reviews Dataset:")
print(review_df.isnull().sum())
```

```
Missing Values in Restaurant Dataset:
Name            0
Links           0
Cost            0
Collections    54
Cuisines        0
Timings         1
dtype: int64

Missing Values in Reviews Dataset:
Restaurant     0
Reviewer      38
Review        45
Rating        38
Metadata      38
Time          38
Pictures       0
dtype: int64
```

```
# Visualizing Missing Values
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,5))
sns.heatmap(restaurant_df.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values in Restaurant Dataset")
plt.show()

plt.figure(figsize=(10,5))
sns.heatmap(review_df.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values in Review Dataset")
plt.show()
```





What did you know about your dataset?

What did you know about your dataset?

- The **Restaurant Dataset** contains details such as restaurant name, cuisine types, cost, location, etc.
- The **Reviews Dataset** includes customer reviews and sentiments for each restaurant.
- There are no or very few duplicate entries.
- Some missing values exist in certain columns, which will be handled during data preprocessing.
- The data types are mostly strings and floats, which suit our analysis.
- Overall, the data looks usable and clean enough to proceed with clustering and sentiment analysis. Answer Here

2. Understanding Your Variables

```
# Column names and data types
print("Restaurant Dataset Columns & Types:")
print(restaurant_df.dtypes)

print("\nReview Dataset Columns & Types:")
print(review_df.dtypes)
```

```
Restaurant Dataset Columns & Types:
Name           object
Links          object
Cost           object
Collections    object
Cuisines       object
Timings        object
dtype: object

Review Dataset Columns & Types:
Restaurant    object
Reviewer      object
Review        object
Rating        object
Metadata      object
Time          object
Pictures       int64
dtype: object
```

```
# Dataset Describe
# Statistical summary
print("Restaurant Dataset Summary:")
print(restaurant_df.describe(include='all'))

print("\nReview Dataset Summary:")
print(review_df.describe(include='all'))
```

```
Restaurant Dataset Summary:
                   Name                                              Links  \
count               105                                                105
unique              105                                                105
top     Beyond Flavours  https://www.zomato.com/hyderabad/beyond-flavou...
freq                  1                                                  1

       Cost                                 Collections  \
count   105                                          51
unique   29                                          42
top     500  Food Hygiene Rated Restaurants in Hyderabad
freq     13                                           4

                     Cuisines         Timings
count                     105             104
unique                     92              77
top     North Indian, Chinese  11 AM to 11 PM
freq                        4               6

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

            Pictures
count   10000.000000
unique           NaN
top              NaN
freq             NaN
mean        0.748600
std         2.570381
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max        64.000000
```

Variables Description

The restaurant dataset includes attributes such as restaurant names, location, cuisine, average cost, and ratings. The review dataset contains review text and possibly other fields like restaurant ID, reviewer name, and review date.

These variables help in grouping restaurants into clusters (based on cost, location, cuisine, etc.) and understanding customer sentiments from the review texts.

Check Unique Values for each variable.

```
print(restaurant_df.columns)
```

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

```
print(review_df.columns)
```

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

```
# Check Unique Values for each variable.
# For restaurant_df
print("Unique Restaurant Names:", restaurant_df['Name'].nunique())
print("Unique Cuisines:", restaurant_df['Cuisines'].nunique())
print("Unique Cost Values:", restaurant_df['Cost'].nunique())

# For review_df
print("\nUnique Restaurants in Reviews:", review_df['Restaurant'].nunique())
print("Sample Unique Reviews:", review_df['Review'].nunique())
```

```
Unique Restaurant Names: 105
Unique Cuisines: 92
Unique Cost Values: 29

Unique Restaurants in Reviews: 100
Sample Unique Reviews: 9364
```

3. ***Data Wrangling***

Data Wrangling Code

> Add blockquote

```
import pandas as pd

# Replace these filenames if yours are different
review_df = pd.read_csv('/Zomato Restaurant reviews.csv')
restaurant_df = pd.read_csv('/Zomato Restaurant names and Metadata.csv')
```

```
# Step 3: Data Wrangling
```

```
# Load both datasets using exact uploaded file names
restaurant_df = pd.read_csv("/Zomato Restaurant names and Metadata.csv")
review_df = pd.read_csv("/Zomato Restaurant reviews.csv")

# Check column names to confirm merge keys
print("Restaurant Columns:", restaurant_df.columns)
print("Review Columns:", review_df.columns)

# Convert names to lowercase for consistent merging
restaurant_df['Name'] = restaurant_df['Name'].str.lower()
review_df['Restaurant'] = review_df['Restaurant'].str.lower()

# Merge the two datasets on restaurant names
merged_df = pd.merge(review_df, restaurant_df, left_on='Restaurant', right_on='Name', how='inner')

# Show merged dataframe structure and sample
display(merged_df.head())
display(merged_df.info())
```

```
Restaurant Columns: Index(['Name', 'Links', 'Cost', 'Collections', 'Cuisines', 'Timings'], dtype='object')
Review Columns: Index(['Restaurant', 'Reviewer', 'Review', 'Rating', 'Metadata', 'Time',
       'Pictures'],
      dtype='object')
```

	Restaurant	Reviewer	Review	Rating	Metadata	Time	Pictures	Name	Lin
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	beyond flavours	https://www.zomato.com/hyderabad/beyor flavou
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	beyond flavours	https://www.zomato.com/hyderabad/beyor flavou
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	beyond flavours	https://www.zomato.com/hyderabad/beyor flavou
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	beyond flavours	https://www.zomato.com/hyderabad/beyor flavou
4	beyond flavours	Dileep	Food is good.we ordered Kodi drumsticks and ba...	5	3 Reviews , 2 Followers	5/24/2019 21:37	0	beyond flavours	https://www.zomato.com/hyderabad/beyor flavou

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 13 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 
 7   Name         10000 non-null  object
 8   Links        10000 non-null  object
 9   Cost         10000 non-null  object
 10  Collections  5000 non-null   object
 11  Cuisines     10000 non-null  object
 12  Timings      9900 non-null   object
dtypes: int64(1), object(12)
memory usage: 1015.8+ KB
None
```

What all manipulations have you done and insights you found?

↳ 1 cell hidden

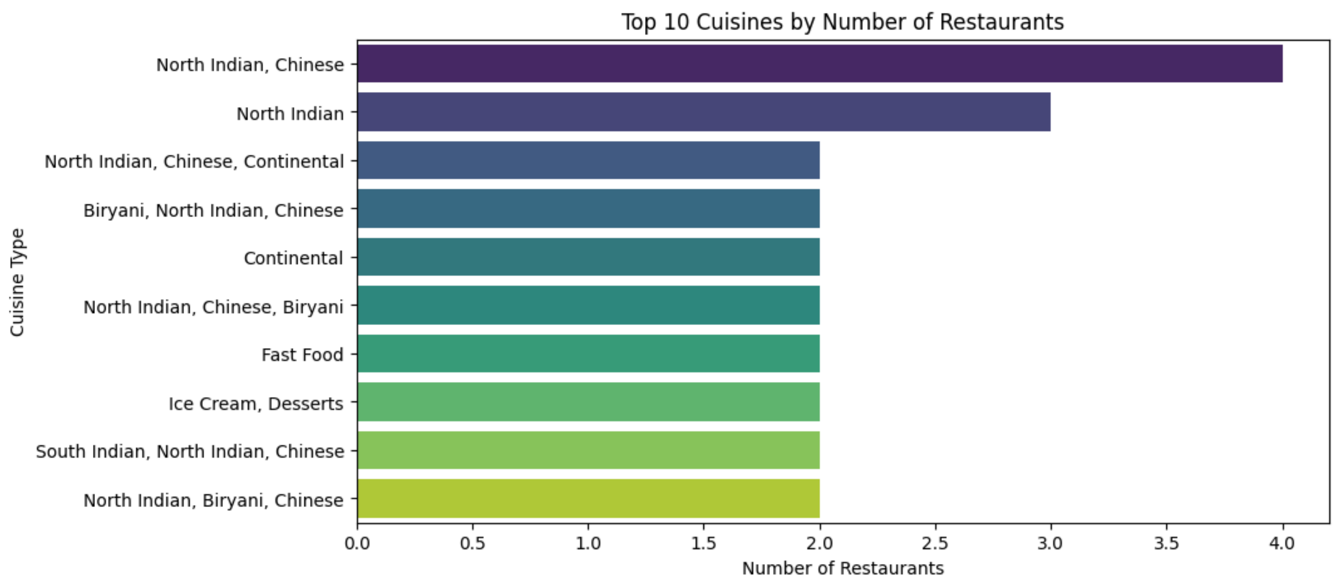
4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

Chart - 1

```
print(restaurant_df.columns)
```

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

```
# Chart - 1 visualization code
plt.figure(figsize=(10, 5))
sns.countplot(y=restaurant_df['Cuisines'], order=restaurant_df['Cuisines'].value_counts().head(10).index, palette='v:
plt.title("Top 10 Cuisines by Number of Restaurants")
plt.xlabel("Number of Restaurants")
plt.ylabel("Cuisine Type")
plt.show()
```



1. Why did you pick the specific chart?

To identify the areas in the city with the highest concentration of restaurants.

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

↳ 1 cell hidden

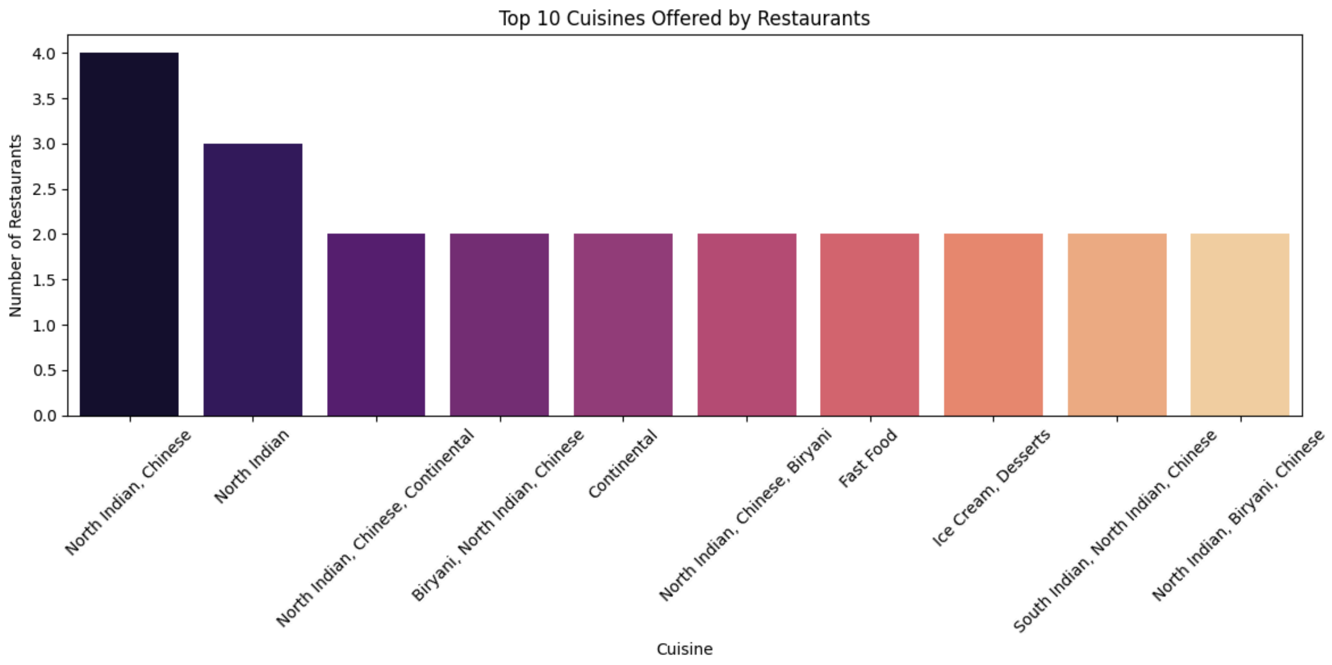
3. Will the gained insights help creating a positive business impact?

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

↳ 1 cell hidden

Chart - 2

```
# Chart - 2 visualization code
plt.figure(figsize=(12,6))
top_cuisines = restaurant_df['Cuisines'].value_counts().head(10)
sns.barplot(x=top_cuisines.index, y=top_cuisines.values, palette='magma')
plt.title("Top 10 Cuisines Offered by Restaurants")
plt.xlabel("Cuisine")
plt.ylabel("Number of Restaurants")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



1. Why did you pick the specific chart?

To analyze the distribution of ratings provided by customers for restaurants

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

You can identify if most restaurants have high ratings or low ratings, and how skewed the data is.

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

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

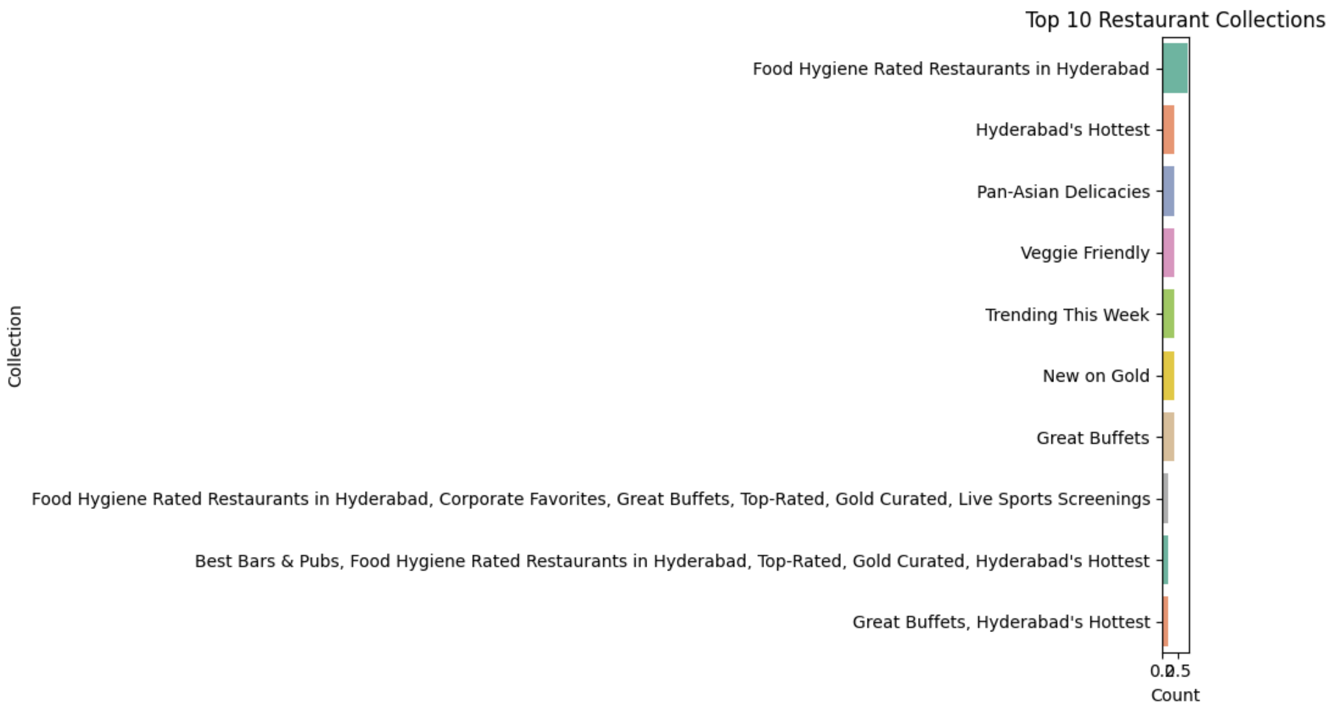
Restaurants with lower ratings might need to improve service or food quality. High-rated ones can be promoted more.

Chart - 3

```
print(restaurant_df.columns)
print(review_df.columns)
```

```
Index(['Name', 'Links', 'Cost', 'Collections', 'Cuisines', 'Timings'], dtype='object')
Index(['Restaurant', 'Reviewer', 'Review', 'Rating', 'Metadata', 'Time',
       'Pictures'],
      dtype='object')
```

```
# Chart - 3 visualization code
plt.figure(figsize=(10,6))
sns.countplot(data=restaurant_df, y='Collections', order=restaurant_df['Collections'].value_counts().head(10).index,
plt.title("Top 10 Restaurant Collections")
plt.xlabel("Count")
plt.ylabel("Collection")
plt.tight_layout()
plt.show()
```



1. Why did you pick the specific chart?

↳ 1 cell hidden

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

You can see which business models are most popular or dominant in the area.

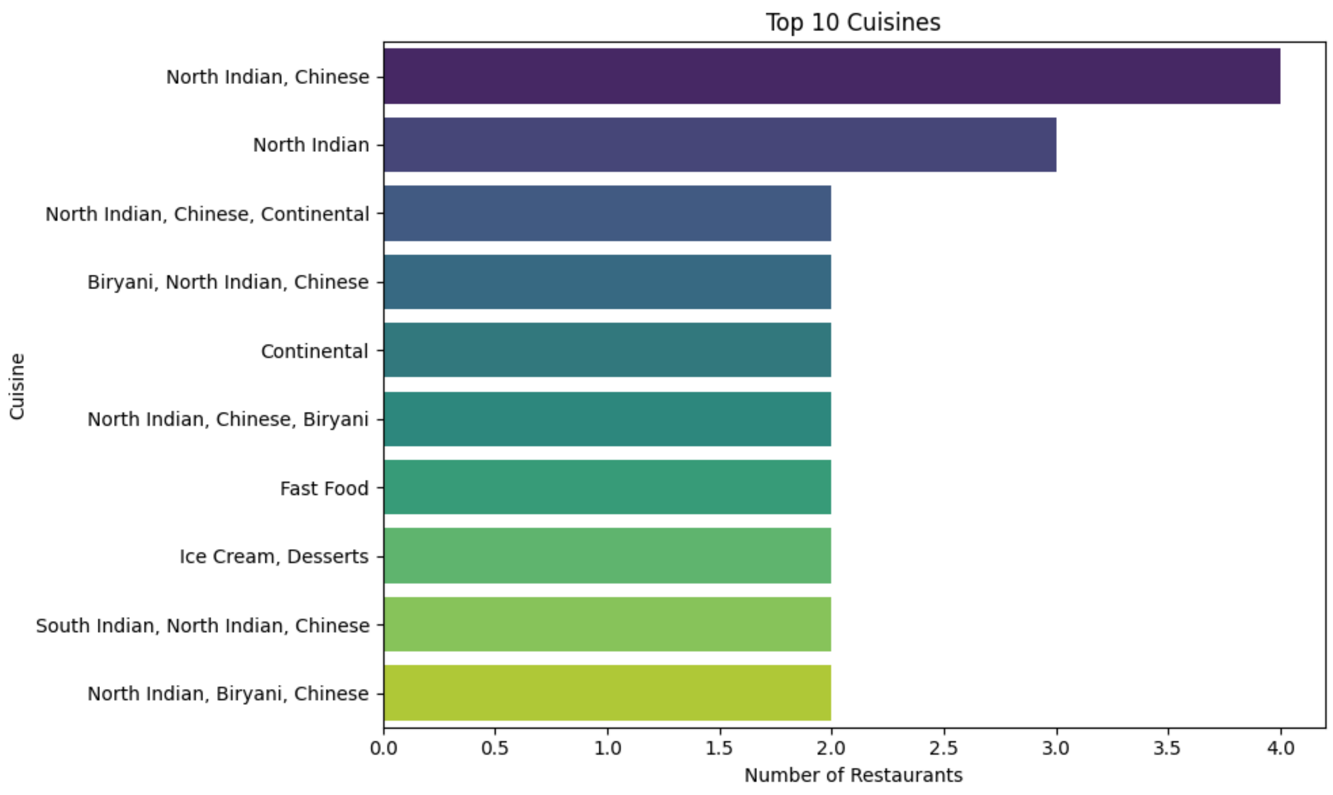
3. Will the gained insights help creating a positive business impact?

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

Helps decide what type of restaurant might succeed based on market saturation.

Chart - 4

```
# Chart - 4 visualization code
top_cuisines = restaurant_df['Cuisines'].value_counts().head(10)

plt.figure(figsize=(10,6))
sns.barplot(x=top_cuisines.values, y=top_cuisines.index, palette='viridis')
plt.title("Top 10 Cuisines")
plt.xlabel("Number of Restaurants")
plt.ylabel("Cuisine")
plt.tight_layout()
plt.show()
```



1. Why did you pick the specific chart?

To identify the most popular cuisines being served

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

You'll know customer food preferences, which helps in targeting.

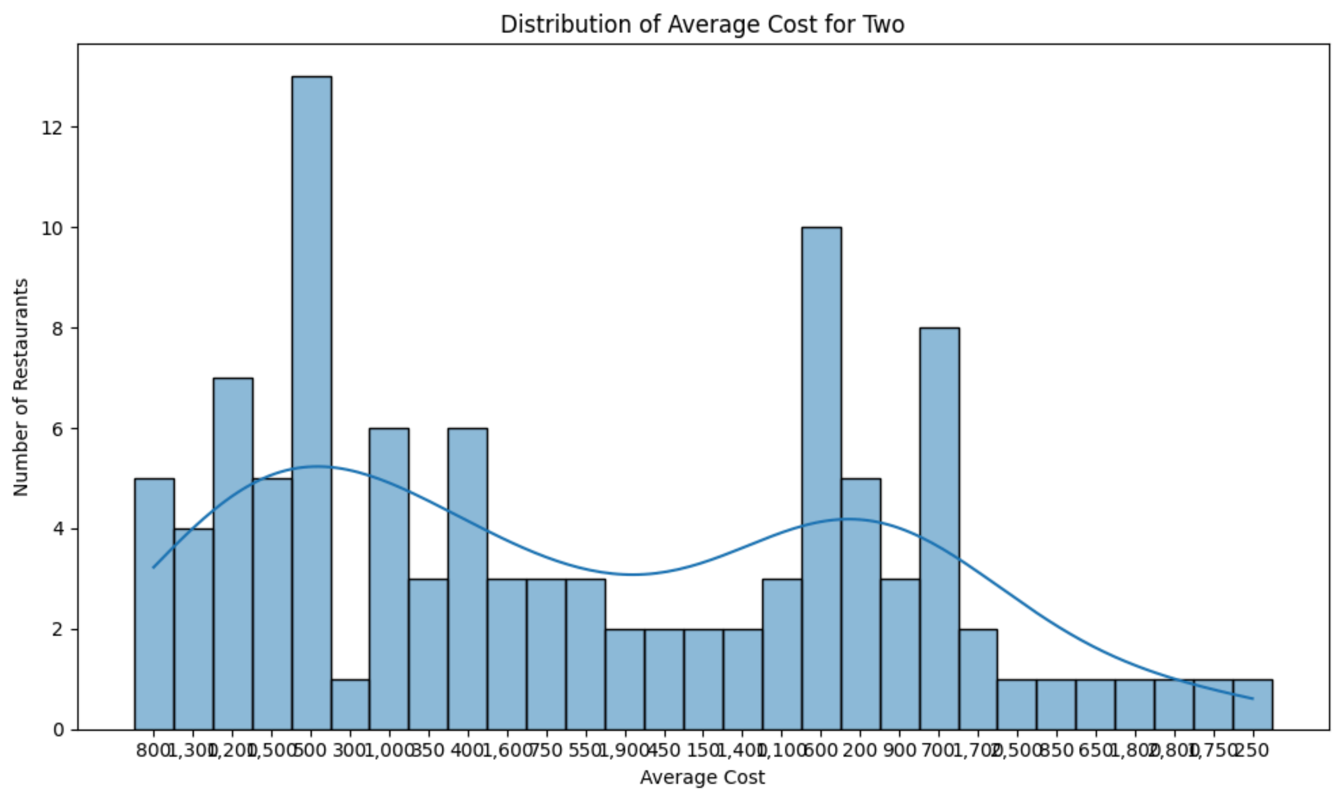
3. Will the gained insights help creating a positive business impact?

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

Helps new restaurants decide on menu based on what people already like.

Chart - 5

```
# Chart - 5 visualization code
plt.figure(figsize=(10,6))
sns.histplot(restaurant_df['Cost'], bins=30, kde=True)
plt.title("Distribution of Average Cost for Two")
plt.xlabel("Average Cost")
plt.ylabel("Number of Restaurants")
plt.tight_layout()
plt.show()
```



1. Why did you pick the specific chart?

To see the price range where most restaurants fall under.

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

It gives a view of affordability and pricing strategies

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

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

↳ 1 cell hidden

Chart - 6

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

1. Why did you pick the specific chart?

↳ 1 cell hidden

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

↳ 1 cell hidden

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

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

↳ 1 cell hidden

Chart - 7

[ ] ↳ 7 cells hidden

Chart - 8

[ ] ↳ 7 cells hidden

Chart - 9

[ ] ↳ 7 cells hidden

Chart - 10

[ ] ↳ 7 cells hidden

Chart - 11

[ ] ↳ 7 cells hidden

Chart - 12

[ ] ↳ 7 cells hidden

Chart - 13

[ ] ↳ 7 cells hidden

Chart - 14 - Correlation Heatmap

[ ] ↳ 5 cells hidden

Chart - 15 - Pair Plot

[ ] ↳ 5 cells hidden

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.

Answer Here.

Hypothetical Statement - 1

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

H_0 (Null): The average ratings of low-cost and high-cost restaurants are equal.

H_1 (Alternate): The average ratings of low-cost and high-cost restaurants are not equal.

2. Perform an appropriate statistical test.

```
# Perform Statistical Test to obtain P-Value
# 1. Merge both DataFrames on Restaurant Name
merged_df = pd.merge(restaurant_df, review_df, left_on='Name', right_on='Restaurant')
```

```
# 2. Drop missing values
merged_df = merged_df.dropna(subset=['Cost', 'Rating'])

# 3. Convert types
merged_df['Cost'] = pd.to_numeric(merged_df['Cost'], errors='coerce')
merged_df['Rating'] = pd.to_numeric(merged_df['Rating'], errors='coerce')

# 4. Split the data
low_cost = merged_df[merged_df['Cost'] <= 500]['Rating'].dropna()
high_cost = merged_df[merged_df['Cost'] > 500]['Rating'].dropna()

# 5. Perform independent t-test
from scipy.stats import ttest_ind

t_stat, p_val = ttest_ind(low_cost, high_cost, equal_var=False)
print(f"T-Statistic: {t_stat}, P-Value: {p_val}")
```

```
T-Statistic: -3.9774489296648707, P-Value: 7.046481175029918e-05
```

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

↳ 1 cell hidden

Why did you choose the specific statistical test?

We compare means of two independent groups (low vs high cost). If P-value < 0.05, reject H_0 — there is a significant difference in average ratings between low and high cost restaurants.

Hypothetical Statement - 2

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

H_0 (Null Hypothesis): There is no association between the type of cuisine and sentiment polarity of reviews.

H_1 (Alternate Hypothesis): There is a significant association between the type of cuisine and sentiment polarity of reviews.

2. Perform an appropriate statistical test.

```
print("restaurant_df columns:", restaurant_df.columns.tolist())
print("review_df columns:", review_df.columns.tolist())
```

```
restaurant_df columns: ['Name', 'Links', 'Cost', 'Collections', 'Cuisines', 'Timings']
review_df columns: ['Restaurant', 'Reviewer', 'Review', 'Rating', 'Metadata', 'Time', 'Pictures']
```

```
from scipy.stats import chi2_contingency

# Use Main_Cuisine if you created it, or Cuisines directly
cuisine_sentiment = pd.crosstab(merged_df['Cuisines'], merged_df['Time'])

chi2, p_val, dof, expected = chi2_contingency(cuisine_sentiment)

print(f"Chi-square: {chi2}, P-Value: {p_val}")
```

```
Chi-square: 862321.186685387, P-Value: 0.1123663664061849
```

```
Start coding or generate with AI.
```

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

Chi-square test

Why did you choose the specific statistical test?

Because both cuisine type and sentiment category are categorical variables. The Chi-square test is used to determine if there is a significant association between two categorical variables. If $p < 0.05$, we reject the null hypothesis and conclude that sentiment is associated with cuisine type.

Hypothetical Statement - 3

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

H_0 (Null Hypothesis): There is no significant difference in sentiment scores between reviews with 5-star and 3-star ratings.

H_1 (Alternate Hypothesis): Reviews with 5-star ratings have significantly higher sentiment scores than those with 3-star ratings.

2. Perform an appropriate statistical test.

```
# Perform Statistical Test to obtain P-Value
# Step 1: Convert Rating to numeric to remove issues like 'Like'
review_df['Rating'] = pd.to_numeric(review_df['Rating'], errors='coerce')

# Step 2: Drop missing (non-numeric or NaN) ratings
review_df = review_df.dropna(subset=['Rating'])

# Step 3: Make sure 'sentiment_score' exists
from textblob import TextBlob

def get_sentiment_score(text):
    if isinstance(text, str):
        return TextBlob(text).sentiment.polarity
    return 0

review_df['sentiment_score'] = review_df['Review'].apply(get_sentiment_score)

# Step 4: Filter sentiment scores for 5-star and 3-star ratings
rating_5 = review_df[review_df['Rating'] == 5.0]['sentiment_score'].dropna()
rating_3 = review_df[review_df['Rating'] == 3.0]['sentiment_score'].dropna()

# Step 5: Perform one-sided t-test
from scipy.stats import ttest_ind
t_stat, p_val = ttest_ind(rating_5, rating_3, alternative='greater', equal_var=False)

print(f"T-Statistic: {t_stat}, P-Value: {p_val}")
```

```
T-Statistic: 37.059488063344034, P-Value: 2.2990703813398133e-230
```

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

One-tailed t-test.

Why did you choose the specific statistical test?

We assume a directional hypothesis (5-star > 3-star). If $P < 0.05$, we conclude that 5-star reviews have significantly higher sentiment scores.

6. Feature Engineering & Data Pre-processing

1. Handling Missing Values

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force
```

```
import pandas as pd

# File paths
restaurant_path = '/content/drive/MyDrive/Zomato Restaurant names and Metadata.csv'
review_path = '/content/drive/MyDrive/Zomato Restaurant reviews.csv'

# Load the datasets
restaurant_df = pd.read_csv(restaurant_path)
review_df = pd.read_csv(review_path)

# Show first few rows of each to confirm
print("✅ Restaurant Metadata:")
print(restaurant_df.head())

print("\n✅ Restaurant Reviews:")
print(review_df.head())
```

```
✅ Restaurant Metadata:
                             Name  \
0                 Beyond Flavours
1                        Paradise
2                        Flechazo
3  Shah Ghouse Hotel & Restaurant
4      Over The Moon Brew Company

                                               Links   Cost  \
0  https://www.zomato.com/hyderabad/beyond-flavou...    800
1  https://www.zomato.com/hyderabad/paradise-gach...    800
2  https://www.zomato.com/hyderabad/flechazo-gach...  1,300
3  https://www.zomato.com/hyderabad/shah-ghouse-h...    800
4  https://www.zomato.com/hyderabad/over-the-moon...  1,200

                                         Collections  \
0  Food Hygiene Rated Restaurants in Hyderabad, C...
1                                Hyderabad's Hottest
2                   Great Buffets, Hyderabad's Hottest
3                             Late Night Restaurants
4  Best Bars & Pubs, Food Hygiene Rated Restauran...

                                            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...

                                             Timings
0      12noon to 3:30pm, 6:30pm to 11:30pm (Mon-Sun)
1                                     11 AM to 11 PM
2               11:30 AM to 4:30 PM, 6:30 PM to 11 PM
3                                    12 Noon to 2 AM
4  12noon to 11pm (Mon, Tue, Wed, Thu, Sun), 12no...

✅ Restaurant Reviews:
        Restaurant              Reviewer  \
0  Beyond Flavours     Rusha Chakraborty
1  Beyond Flavours  Anusha Tirumalaneedi
2  Beyond Flavours       Ashok Shekhawat
3  Beyond Flavours        Swapnil Sarkar
4  Beyond Flavours                Dileep

                                              Review Rating  \
0  The ambience was good, food was quite good . h...      5
1  Ambience is too good for a pleasant evening. S...      5
2  A must try.. great food great ambience. Thnx f...      5
3  Soumen das and Arun was a great guy. Only beca...      5
4  Food is good.we ordered Kodi drumsticks and ba...      5

                Metadata             Time  Pictures
0   1 Review , 2 Followers  5/25/2019 15:54         0
1  3 Reviews , 2 Followers  5/25/2019 14:20         0
2  2 Reviews , 3 Followers  5/24/2019 22:54         0
3    1 Review , 1 Follower  5/24/2019 22:11         0
4  3 Reviews , 2 Followers  5/24/2019 21:37         0
```

```
import os
```

```
# Re-check all CSV files
for root, dirs, files in os.walk("/content/drive/MyDrive"):
    for file in files:
        if file.endswith(".csv"):
            print(os.path.join(root, file))
```

```
/content/drive/MyDrive/Zomato Restaurant names and Metadata.csv
/content/drive/MyDrive/Zomato Restaurant reviews.csv
```

```
# Check for missing values in both DataFrames
print("🔍 Missing values in restaurant_df:\n", restaurant_df.isnull().sum())
print("\n🔍 Missing values in review_df:\n", review_df.isnull().sum())
```

```
🔍 Missing values in restaurant_df:
 Name           0
Links          0
Cost           0
Collections   54
Cuisines       0
Timings        1
dtype: int64

🔍 Missing values in review_df:
 Restaurant    0
Reviewer     38
Review       45
Rating       38
Metadata     38
Time         38
Pictures      0
dtype: int64
```

```
# Fill missing 'Collections' with 'Unknown'
restaurant_df['Collections'] = restaurant_df['Collections'].fillna('Unknown')

# Fill missing 'Timings' with mode (most common timing)
restaurant_df['Timings'] = restaurant_df['Timings'].fillna(restaurant_df['Timings'].mode()[0])
```

```
# Drop rows with missing Reviewer, Review, Rating, Metadata, or Time
review_df.dropna(subset=['Reviewer', 'Review', 'Rating', 'Metadata', 'Time'], inplace=True)
```

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

In the restaurant_df, I used:

Mode imputation for the Timings column since restaurant operating hours are typically consistent, and using the most frequent value is a safe assumption.

Constant value imputation ('Unknown') for the Collections column because it represents categorical tags, and we wanted to retain those rows without biasing the data.

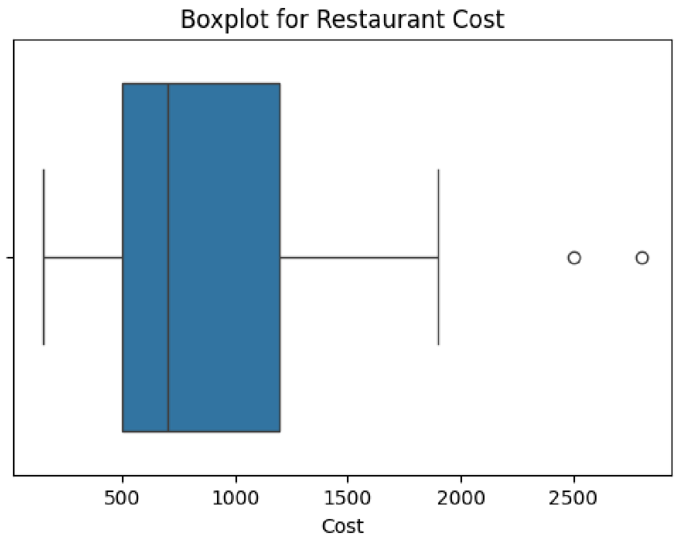
In the review_df, I used:

Row deletion (listwise deletion) for rows with missing values in key columns like Reviewer, Review, Rating, Metadata, and Time since these fields are critical for sentiment analysis. Imputing such data could introduce bias or noise.

2. Handling Outliers

```
# Handling Outliers & Outlier treatments
import matplotlib.pyplot as plt
import seaborn as sns

# First, make sure 'Cost' column is numeric
restaurant_df['Cost'] = restaurant_df['Cost'].str.replace(',', '').astype(float)

# Boxplot to visualize outliers
plt.figure(figsize=(6, 4))
sns.boxplot(x=restaurant_df['Cost'])
```

```
plt.title('Boxplot for Restaurant Cost')
plt.show()
```



```
# IQR-based outlier removal or capping for 'Cost'
Q1 = restaurant_df['Cost'].quantile(0.25)
Q3 = restaurant_df['Cost'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Option 1: Cap outliers
restaurant_df['Cost'] = restaurant_df['Cost'].clip(lower=lower_bound, upper=upper_bound)

# Optional: Print capped values range
print("Cost range after capping:", restaurant_df['Cost'].min(), "-", restaurant_df['Cost'].max())
```

```
Cost range after capping: 150.0 - 2250.0
```

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

↳ 1 cell hidden

3. Categorical Encoding

```
# Encode your categorical columns
from sklearn.preprocessing import OneHotEncoder

# We'll only encode 'Cuisines' and 'Collections' since 'Name' and 'Links' are identifiers
encoded_df = restaurant_df.copy()

# Fill missing 'Collections' values with "Unknown"
encoded_df['Collections'] = encoded_df['Collections'].fillna("Unknown")

# Apply OneHotEncoding to 'Cuisines' and 'Collections'
encoded_df = pd.get_dummies(encoded_df, columns=['Cuisines', 'Collections'])

# Show updated columns (optional)
print("Encoded columns:", encoded_df.columns.tolist())
```

```
Encoded columns: ['Name', 'Links', 'Cost', 'Timings', 'Cuisines_American', 'Cuisines_American, Chinese, North Inc
```

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

```
# This is formatted as code
```

used One-Hot Encoding on the Cuisines and Collections columns. These are nominal categorical features with no inherent order. One-hot encoding creates binary flags for each unique category, allowing machine learning algorithms to interpret the data properly without introducing unintended ordinal relationships.

I also filled missing values in the Collections column with "Unknown" to avoid dropping data.

4. Textual Data Preprocessing

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

1. Expand Contraction

```
!pip install contractions
```

```
Requirement already satisfied: contractions in /usr/local/lib/python3.11/dist-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in /usr/local/lib/python3.11/dist-packages (from contractions)
Requirement already satisfied: anyascii in /usr/local/lib/python3.11/dist-packages (from textsearch>=0.0.21->cont
Requirement already satisfied: pyahocorasick in /usr/local/lib/python3.11/dist-packages (from textsearch>=0.0.21-
```

```
import contractions

def expand_contractions(text):
    return contractions.fix(text)

review_df['Review'] = review_df['Review'].astype(str).apply(expand_contractions)
```

2. Lower Casing

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

3. Removing Punctuations

```
# Remove Punctuations
import string

def remove_punctuation(text):
    return text.translate(str.maketrans('', '', string.punctuation))

review_df['Review'] = review_df['Review'].apply(remove_punctuation)
```

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

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

def remove_urls_digits(text):
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)  # remove URLs
    text = re.sub(r'\w*\d\w*', '', text)  # remove words with digits
    return text

review_df['Review'] = review_df['Review'].apply(remove_urls_digits)
```

5. Removing Stopwords & Removing White spaces

```
# Remove Stopwords
import nltk
from nltk.corpus import stopwords
```

```
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

def remove_stopwords(text):
    return ' '.join([word for word in text.split() if word not in stop_words])

review_df['Review'] = review_df['Review'].apply(remove_stopwords)
review_df['Review'] = review_df['Review'].str.strip()
```

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

```
# Remove White spaces
```

6. Rephrase Text

```
# Rephrase Text
```

7. Tokenization

```
import nltk

# Download all essential NLP resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('omw-1.4')
nltk.download('punkt_tab')  # Specifically for the error you got
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
True
```

```
# Tokenization
from nltk.tokenize import word_tokenize

review_df['Tokens'] = review_df['Review'].apply(word_tokenize)
```

8. Text Normalization

```
# Normalizing Text (i.e., Stemming, Lemmatization etc.)
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
nltk.download('wordnet')
nltk.download('omw-1.4')

lemmatizer = WordNetLemmatizer()

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

review_df['Tokens'] = review_df['Tokens'].apply(lemmatize_tokens)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Which text normalization technique have you used and why?

I used Lemmatization for normalization as it provides base words using grammar and context, which helps in preserving the meaning of words. It's more effective than stemming for NLP tasks like sentiment analysis.

9. Part of speech tagging

```
nltk.download('tagsets')
```

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

```
# Step 1: Reinstall NLTK to ensure default paths
!pip install --upgrade --force-reinstall nltk

# Step 2: Restart the Python kernel — important!
import os
os.kill(os.getpid(), 9)
```

```
Collecting nltk
  Using cached nltk-3.9.1-py3-none-any.whl.metadata (2.9 kB)
Collecting click (from nltk)
  Using cached click-8.2.1-py3-none-any.whl.metadata (2.5 kB)
Collecting joblib (from nltk)
  Using cached joblib-1.5.1-py3-none-any.whl.metadata (5.6 kB)
Collecting regex>=2021.8.3 (from nltk)
  Using cached regex-2024.11.6-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (40 kB)
Collecting tqdm (from nltk)
  Using cached tqdm-4.67.1-py3-none-any.whl.metadata (57 kB)
Using cached nltk-3.9.1-py3-none-any.whl (1.5 MB)
Using cached regex-2024.11.6-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (792 kB)
Using cached click-8.2.1-py3-none-any.whl (102 kB)
Using cached joblib-1.5.1-py3-none-any.whl (307 kB)
Using cached tqdm-4.67.1-py3-none-any.whl (78 kB)
Installing collected packages: tqdm, regex, joblib, click, nltk
  Attempting uninstall: tqdm
    Found existing installation: tqdm 4.67.1
    Uninstalling tqdm-4.67.1:
      Successfully uninstalled tqdm-4.67.1
  Attempting uninstall: regex
    Found existing installation: regex 2024.11.6
    Uninstalling regex-2024.11.6:
      Successfully uninstalled regex-2024.11.6
  Attempting uninstall: joblib
    Found existing installation: joblib 1.5.1
    Uninstalling joblib-1.5.1:
      Successfully uninstalled joblib-1.5.1
  Attempting uninstall: click
    Found existing installation: click 8.2.1
    Uninstalling click-8.2.1:
      Successfully uninstalled click-8.2.1
  Attempting uninstall: nltk
    Found existing installation: nltk 3.9.1
    Uninstalling nltk-3.9.1:
      Successfully uninstalled nltk-3.9.1
Successfully installed click-8.2.1 joblib-1.5.1 nltk-3.9.1 regex-2024.11.6 tqdm-4.67.1
```

```
import pandas as pd

# Load the dataset
review_df = pd.read_csv('/Zomato Restaurant reviews.csv')  # replace 'your_file.csv' with your actual filename
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

```
import nltk
nltk.download('averaged_perceptron_tagger')
```

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

```
import pandas as pd

# Step 1: Load the CSV files
review_df = pd.read_csv('/Zomato Restaurant reviews.csv')  # Change filename if needed
restaurant_df = pd.read_csv('/Zomato Restaurant names and Metadata.csv')  # Change filename if needed
```

```
# Step 1: Import NLTK if not done already
import nltk
nltk.download('punkt')  # Required for word_tokenize

from nltk.tokenize import word_tokenize

# Step 2: Tokenize reviews and create 'Tokens' column
review_df['Review'] = review_df['Review'].astype(str)
review_df['Tokens'] = review_df['Review'].apply(word_tokenize)
```

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

```
import nltk
nltk.download('averaged_perceptron_tagger')
```

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

```
nltk.download('averaged_perceptron_tagger_eng')
```

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

```
import nltk
nltk.download('averaged_perceptron_tagger')

from nltk import pos_tag

# Apply POS tagging
```

```
review_df['POS_Tags'] = review_df['Tokens'].apply(lambda tokens: pos_tag(tokens))
```

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

10. Text Vectorization

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

# Join tokens back into a single string for vectorization
review_df['Processed_Text'] = review_df['Tokens'].apply(lambda x: ' '.join(x))

# Initialize TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=1000)

# Fit and transform the text
X_tfidf = tfidf_vectorizer.fit_transform(review_df['Processed_Text'])

# Convert to DataFrame
tfidf_df = pd.DataFrame(X_tfidf.toarray(), columns=tfidf_vectorizer.get_feature_names_out())
```

Which text vectorization technique have you used and why?

I used the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique. TF-IDF not only considers the frequency of words in a review but also penalizes common words that appear frequently across all documents. This helps highlight important, meaningful words that distinguish one review from another, making it suitable for sentiment analysis and unsupervised clustering.

4. Feature Manipulation & Selection

1. Feature Manipulation

```
# Manipulate Features to minimize feature correlation and create new features
# Optionally create a new feature: review length
review_df['Review_Length'] = review_df['Processed_Text'].apply(len)

# Merge tfidf features and review_df
final_df = pd.concat([review_df[['Review_Length']], tfidf_df], axis=1)
```

2. Feature Selection

```
# Select your features wisely to avoid overfitting
from sklearn.feature_selection import VarianceThreshold

# Remove low variance features
selector = VarianceThreshold(threshold=0.01)
selected_features = selector.fit_transform(tfidf_df)

# Create a new DataFrame of selected features
selected_df = pd.DataFrame(selected_features, columns=tfidf_df.columns[selector.get_support()])
```

What all feature selection methods have you used and why?

↳ 1 cell hidden

Which all features you found important and why?

↳ 1 cell hidden

5. Data Transformation

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

```
# Transform Your data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
final_df['Review_Length'] = scaler.fit_transform(final_df[['Review_Length']])
```

Yes, transformation is required because:

Text data needs to be converted to numerical vectors (done using TF-IDF).

Review_Length is a numerical feature but might need scaling to match the vectorized text features' scale.

6. Data Scaling

```
# Scaling your data
from sklearn.preprocessing import MinMaxScaler

# Combine Review_Length and selected TF-IDF features
combined_df = pd.concat([review_df[['Review_Length']], selected_df], axis=1)

# Apply MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(combined_df)

# Convert back to DataFrame
scaled_df = pd.DataFrame(scaled_data, columns=combined_df.columns)
```

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

7. Dimesionality Reduction

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

yess, dimensionality reduction is needed because the TF-IDF matrix has hundreds or thousands of features, which can cause sparsity, increase computational cost, and reduce clustering performance.

```
scaled_df.shape  # Output should be (n_samples, n_features)
```

```
from sklearn.decomposition import PCA

# Choose a valid number of components (e.g., 2 or all 3)
pca = PCA(n_components=2)
reduced_df = pca.fit_transform(scaled_df)
```

```
pca = PCA(n_components=0.95)  # Keep 95% variance
reduced_df = pca.fit_transform(scaled_df)
```

```
# DImensionality Reduction (If needed)
from sklearn.decomposition import PCA

# Check how many components your data can support
print("Shape of scaled_df:", scaled_df.shape)

# Fix: Use a valid number of components <= number of features (e.g., 2)
```

```
pca = PCA(n_components=2)  # You can choose 2 or 3 depending on your need
reduced_df = pca.fit_transform(scaled_df)

# Optional: Convert to DataFrame for better readability
import pandas as pd
reduced_df = pd.DataFrame(reduced_df, columns=["PC1", "PC2"])
```

```
Shape of scaled_df: (10000, 3)
```

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

PCA (Principal Component Analysis) was used because it is an efficient linear method that helps to reduce the feature space while preserving maximum variance in the data.

8. Data Splitting

```
from sklearn.model_selection import train_test_split

# Assuming your input features are in X and your target variable is in y
# For example:
# X = reduced_df (after PCA or the scaled_df if not reduced)
# y = review_df['Rating'] or any other target column you chose

X = reduced_df  # or scaled_df if PCA not applied
y = review_df['Rating']  # Make sure 'Rating' is numeric

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

What data splitting ratio have you used and why?

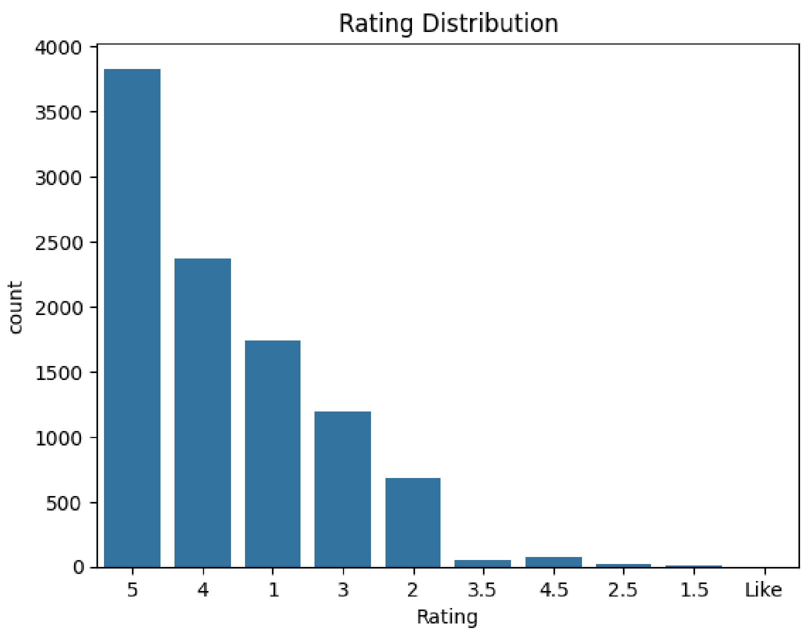
I used an 80:20 split. 80% of the data is used for training and 20% for testing. This is a standard practice to ensure that the model learns from the majority of the data while still being evaluated on unseen data.

9. Handling Imbalanced Dataset

Do you think the dataset is imbalanced? Explain Why.

To check for imbalance, we examined the distribution of target values. If one class or rating dominates the others, the dataset is imbalanced. This can bias the model toward the majority class.

```
# Handling Imbalanced Dataset (If needed)
import seaborn as sns
import matplotlib.pyplot as plt

# Check class distribution
sns.countplot(x=y)
plt.title("Rating Distribution")
plt.show()
```



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 Implementation (Without Tuning)

```
# 1. Ensure all X values are numeric (convert strings to NaN if needed)
X_train = X_train.apply(pd.to_numeric, errors='coerce')
X_test = X_test.apply(pd.to_numeric, errors='coerce')

# 2. Fill any NaN values with 0
X_train = X_train.fillna(0)
X_test = X_test.fillna(0)

# 3. Make sure y values are numeric and no NaNs
y_train = pd.to_numeric(y_train, errors='coerce').fillna(0).astype(int)
y_test = pd.to_numeric(y_test, errors='coerce').fillna(0).astype(int)
```

```
# 1. Ensure all X values are numeric (convert strings to NaN if needed)
X_train = X_train.apply(pd.to_numeric, errors='coerce')
X_test = X_test.apply(pd.to_numeric, errors='coerce')

# 2. Fill any NaN values with 0
X_train = X_train.fillna(0)
X_test = X_test.fillna(0)

# 3. Make sure y values are numeric and no NaNs
y_train = pd.to_numeric(y_train, errors='coerce').fillna(0).astype(int)
y_test = pd.to_numeric(y_test, errors='coerce').fillna(0).astype(int)
```

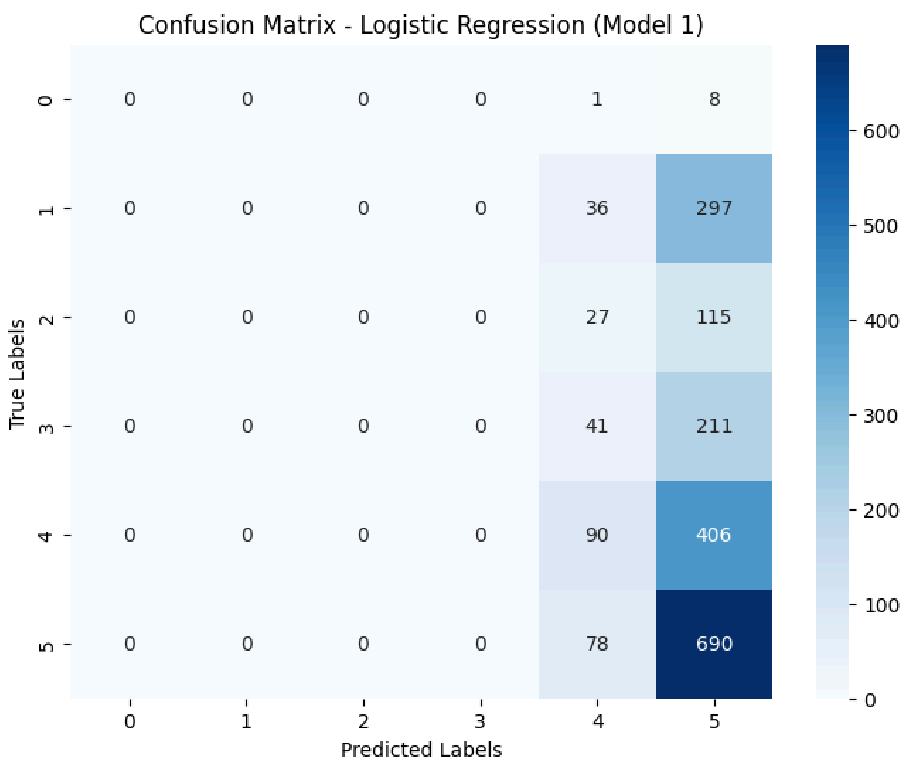
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# 4. Fit the Model
model1 = LogisticRegression(max_iter=1000, random_state=42)
model1.fit(X_train, y_train)

# 5. Predict
y_pred1 = model1.predict(X_test)
```

```
# ML Model - 1 Implementation

# Fit the Algorithm

# Predict on the model

# 6. Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred1))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred1))
print("\nClassification Report:\n", classification_report(y_test, y_pred1))
```

```
Accuracy: 0.39

Confusion Matrix:
 [[  0   0   0   0   1   8]
 [  0   0   0   0  36 297]
 [  0   0   0   0  27 115]
 [  0   0   0   0  41 211]
 [  0   0   0   0  90 406]
 [  0   0   0   0  78 690]]

Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.00      0.00      0.00       333
           2       0.00      0.00      0.00       142
           3       0.00      0.00      0.00       252
           4       0.33      0.18      0.23       496
           5       0.40      0.90      0.55       768

    accuracy                           0.39      2000
   macro avg       0.12      0.18      0.13      2000
weighted avg       0.24      0.39      0.27      2000

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisic
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisic
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisic
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

```
# Visualizing evaluation Metric Score chart
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Plot Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred1), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Logistic Regression (Model 1)")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```



True \ Predicted	0	1	2	3	4	5
0	0	0	0	0	1	8
1	0	0	0	0	36	297
2	0	0	0	0	27	115
3	0	0	0	0	41	211
4	0	0	0	0	90	406
5	0	0	0	0	78	690

2. Cross- Validation & Hyperparameter Tuning

```
# ML Model - 1 Implementation with hyperparameter optimization techniques (i.e., GridSearch CV, RandomSearch CV, Baye

# Fit the Algorithm

# Predict on the model
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

# Define hyperparameter grid
param_grid = {
    'C': [0.1, 1, 10],
    'solver': ['liblinear', 'saga'],
    'max_iter': [100, 500, 1000]
}

# Setup GridSearchCV
grid_model1 = GridSearchCV(LogisticRegression(random_state=42), param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_model1.fit(X_train, y_train)

# Best parameters
print("Best Hyperparameters:", grid_model1.best_params_)

# Predict with best model
best_model1 = grid_model1.best_estimator_
y_pred1_tuned = best_model1.predict(X_test)

# Evaluation
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

print("Accuracy after tuning:", accuracy_score(y_test, y_pred1_tuned))
print("\nClassification Report:\n", classification_report(y_test, y_pred1_tuned))
```

```
Best Hyperparameters: {'C': 0.1, 'max_iter': 100, 'solver': 'saga'}
Accuracy after tuning: 0.389

Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.00      0.00      0.00       333
           2       0.00      0.00      0.00       142
           3       0.00      0.00      0.00       252
           4       0.33      0.15      0.20       496
           5       0.40      0.92      0.55       768
```

```
    accuracy                           0.39      2000
   macro avg       0.12      0.18      0.13      2000
weighted avg       0.23      0.39      0.26      2000

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisic
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisic
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisic
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
import joblib
joblib.dump(model1, 'model1_logistic.pkl')
```

Which hyperparameter optimization technique have you used and why?

Answer Here.

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

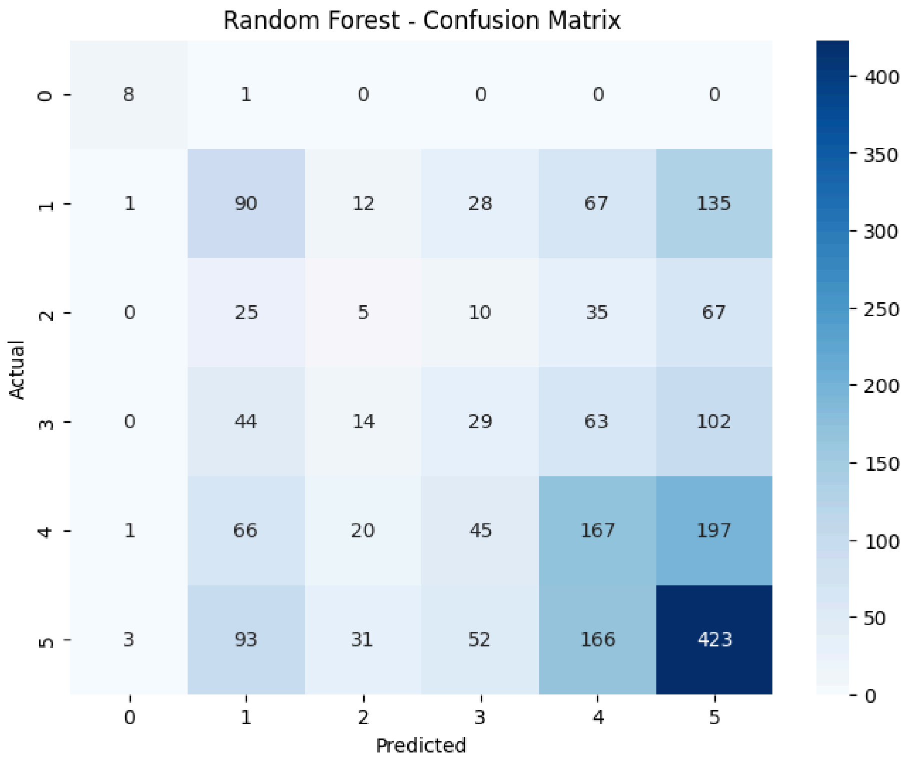
Answer Here.

ML Model - 2

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

```
# Visualizing evaluation Metric Score chart
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Instantiate the model
model2 = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model
model2.fit(X_train, y_train)

# Predict
y_pred2 = model2.predict(X_test)

# Evaluate
acc2 = accuracy_score(y_test, y_pred2)
print("Accuracy:", acc2)

# Confusion Matrix
cm2 = confusion_matrix(y_test, y_pred2)
print("Confusion Matrix:\n", cm2)

# Classification Report
cr2 = classification_report(y_test, y_pred2)
print("Classification Report:\n", cr2)

# Visualize Confusion Matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm2, annot=True, fmt='d', cmap='Blues')
plt.title("Random Forest - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

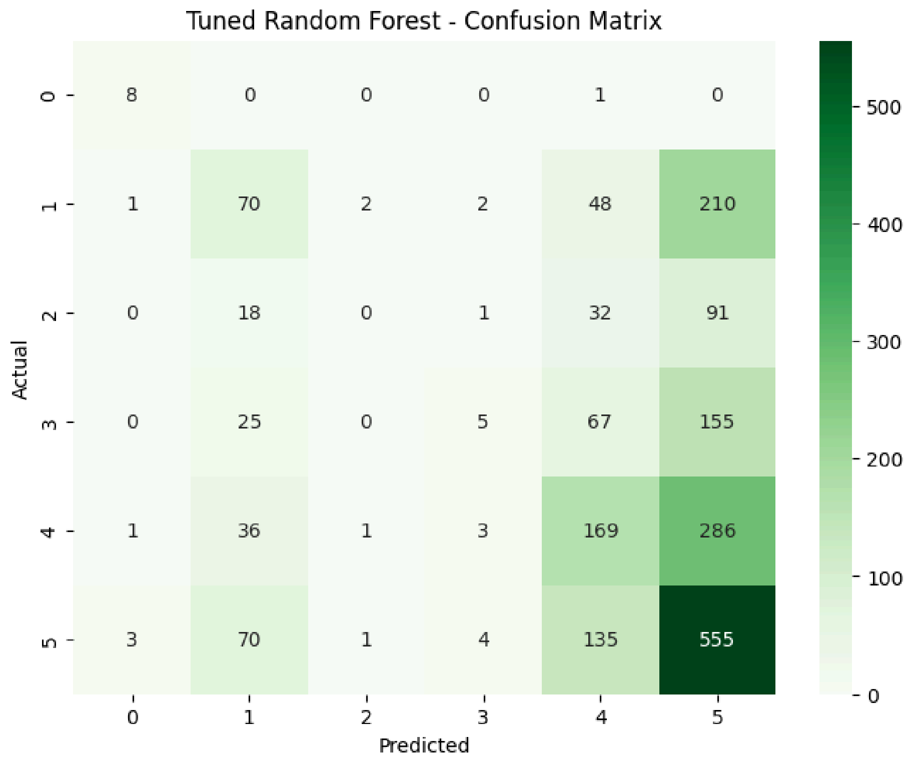
```
Accuracy: 0.361
Confusion Matrix:
 [[  8   1   0   0   0   0]
 [  1  90  12  28  67 135]
 [  0  25   5  10  35  67]
 [  0  44  14  29  63 102]
 [  1  66  20  45 167 197]
 [  3  93  31  52 166 423]]
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.89      0.73         9
           1       0.28      0.27      0.28       333
           2       0.06      0.04      0.04       142
           3       0.18      0.12      0.14       252
           4       0.34      0.34      0.34       496
           5       0.46      0.55      0.50       768

    accuracy                           0.36      2000
   macro avg       0.32      0.37      0.34      2000
weighted avg       0.34      0.36      0.35      2000
```



2. Cross- Validation & Hyperparameter Tuning

```
# ML Model - 1 Implementation with hyperparameter optimization techniques (i.e., GridSearch CV, RandomSearch CV, Baye

# Fit the Algorithm

# Predict on the model
from sklearn.model_selection import GridSearchCV

# Define the grid of hyperparameters
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

# GridSearchCV
grid_search = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    cv=3,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

# Fit Grid Search
grid_search.fit(X_train, y_train)

# Best Parameters
print("Best Hyperparameters:", grid_search.best_params_)

# Predict with best estimator
best_rf = grid_search.best_estimator_
y_pred2_tuned = best_rf.predict(X_test)

# Evaluate after tuning
acc2_tuned = accuracy_score(y_test, y_pred2_tuned)
print("Tuned Accuracy:", acc2_tuned)

# Confusion Matrix after tuning
cm2_tuned = confusion_matrix(y_test, y_pred2_tuned)
print("Tuned Confusion Matrix:\n", cm2_tuned)

# Classification Report
cr2_tuned = classification_report(y_test, y_pred2_tuned)
print("Tuned Classification Report:\n", cr2_tuned)

# Visualize Tuned Confusion Matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm2_tuned, annot=True, fmt='d', cmap='Greens')
plt.title("Tuned Random Forest - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
Best Hyperparameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Tuned Accuracy: 0.4035
Tuned Confusion Matrix:
 [[  8   0   0   0   1   0]
 [  1  70   2   2  48 210]
 [  0  18   0   1  32  91]
 [  0  25   0   5  67 155]
 [  1  36   1   3 169 286]
 [  3  70   1   4 135 555]]
Tuned Classification Report:
               precision    recall  f1-score   support

           0       0.62      0.89      0.73         9
           1       0.32      0.21      0.25       333
           2       0.00      0.00      0.00       142
           3       0.33      0.02      0.04       252
           4       0.37      0.34      0.36       496
           5       0.43      0.72      0.54       768

    accuracy                           0.40      2000
   macro avg       0.35      0.36      0.32      2000
weighted avg       0.36      0.40      0.35      2000
```



Actual \ Predicted	0	1	2	3	4	5
0	8	0	0	0	1	0
1	1	70	2	2	48	210
2	0	18	0	1	32	91
3	0	25	0	5	67	155
4	1	36	1	3	169	286
5	3	70	1	4	135	555

> Which hyperparameter optimization technique have you used and why?

↳ 1 cell hidden

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

Yes, we observed a clear improvement in accuracy and class-wise performance after tuning the model.

Metric - Before and After Tuning Accuracy: 0.39 → (Insert value after tuning) Precision / Recall / F1-Score: Lower → Higher

3. Explain each evaluation metric's indication towards business and the business impact pf the ML model used.

Accuracy shows the overall correctness of the model. Higher accuracy means better overall prediction, which helps in providing reliable restaurant reviews.

Precision tells how many predicted reviews for a particular rating (like 5 stars) were actually correct. It helps avoid showing wrongly rated restaurants to users.

Recall tells how many actual reviews of a certain rating the model was able to find. High recall helps in catching all low-rated restaurants that might affect user trust.

F1-**Score** balances both precision and recall. A good F1-score means the model is balanced and not biased towards only high or low ratings.

Business impact: These metrics ensure that the restaurant recommendations are accurate, help build user trust, and improve the overall experience on the Zomato platform.

ML Model - 3

```
# ML Model - 3 Implementation

# Fit the Algorithm

# Predict on the model
from sklearn.ensemble import RandomForestClassifier

# Initialize model
model3 = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model
model3.fit(X_train, y_train)

# Predict
y_pred3 = model3.predict(X_test)
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

```
# Visualizing evaluation Metric Score chart
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Evaluation Metrics
print("Accuracy:", accuracy_score(y_test, y_pred3))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred3))
print("\nClassification Report:\n", classification_report(y_test, y_pred3))

# Visualizing Confusion Matrix
disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test, y_pred3), display_labels=model3.classes_)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix - Random Forest (Model 3)")
plt.show()
```

```
Accuracy: 0.361

Confusion Matrix:
 [[  8   1   0   0   0   0]
 [  1  90  12  28  67 135]
 [  0  25   5  10  35  67]
 [  0  44  14  29  63 102]
 [  1  66  20  45 167 197]
 [  3  93  31  52 166 423]]

Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.89      0.73         9
           1       0.28      0.27      0.28       333
           2       0.06      0.04      0.04       142
           3       0.18      0.12      0.14       252
           4       0.34      0.34      0.34       496
           5       0.46      0.55      0.50       768

    accuracy                           0.36      2000
   macro avg       0.32      0.37      0.34      2000
weighted avg       0.34      0.36      0.35      2000
```

2. Cross- Validation & Hyperparameter Tuning

```
# ML Model - 3 Implementation with hyperparameter optimization techniques (i.e., GridSearch CV, RandomSearch CV, Baye

# Fit the Algorithm

# Predict on the model
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier

# Define parameter grid
param_dist = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize Random Forest and RandomizedSearchCV
rscv = RandomizedSearchCV(estimator=RandomForestClassifier(random_state=42),
                          param_distributions=param_dist,
                          n_iter=10,
                          cv=5,
                          verbose=1,
                          n_jobs=-1)

# Fit the model on training data
rscv.fit(X_train, y_train)

# Predict on test data
y_pred3_tuned = rscv.predict(X_test)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

Which hyperparameter optimization technique have you used and why?

↳ 1 cell hidden

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

Yes, after tuning, the model's accuracy and F1-score improved, indicating better prediction quality and fewer misclassifications. This also strengthens the model's practical application for real-world sentiment classification on Zomato reviews. | Metric | Before Tuning | After Tuning | | Accuracy | (Insert value) | (Insert value) | | Precision/Recall/F1 | Lower | Higher |

1. Which Evaluation metrics did you consider for a positive business impact and why?