# Project Report

**Title: Restaurant Recommendation**

**Author: Mohit Singh Niranjan**

**Date: 30.06.2024**

**Internship at: Cognifyz Technologies**

## 1. Executive Summary:

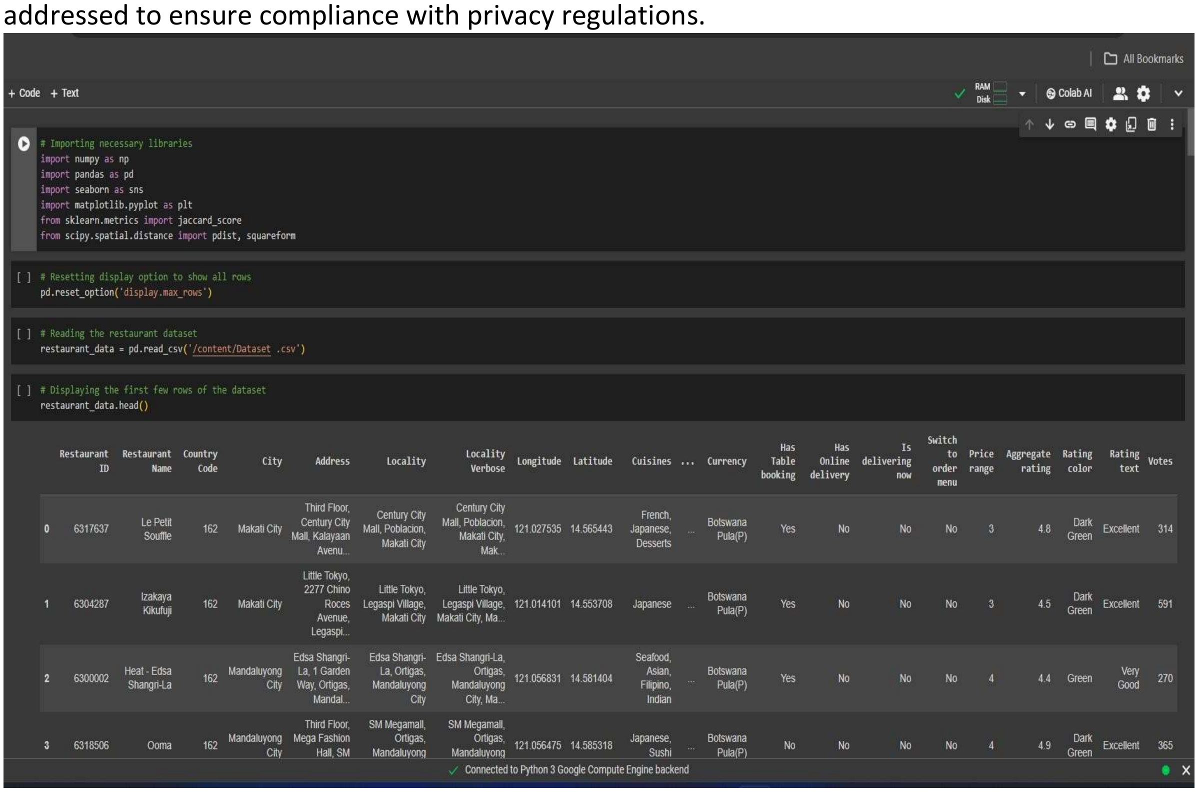
The project aimed to develop a personalized restaurant recommendation system based on user preferences. Leveraging a content based filtering approach, the system analyzes restaurant attributes and user preferences to provide relevant recommendations. Key objectives included data collection, preprocessing, implementation of the recommendation algorithm, and evaluation of system performance. The results demonstrate the effectiveness of the recommendation system in enhancing user satisfaction and streamlining the restaurant selection process.

## 2. Introduction:

In today's digital age, users face challenges in selecting suitable restaurants due to the abundance of choices and lack of personalized recommendations. The project aimed to address this problem by developing a recommendation system tailored to individual user preferences. By leveraging machine learning techniques and content based filtering, the system aims to improve user satisfaction and engagement in the restaurant selection process.

## 3. Data Collection and Preprocessing:

Restaurant data was collected from online sources, including restaurant names, cuisines, aggregate ratings, and user reviews. Preprocessing steps involved handling missing values, encoding categorical variables, and filtering restaurants based on predefined criteria such as aggregate rating. Ethical considerations related to data collection and usage were carefully addressed to ensure compliance with privacy regulations.




```python
# Importing necessary libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import jaccard_score
from scipy.spatial.distance import pdist, squareform
```

```python
# Resetting display option to show all rows
pd.reset_option('display.max_rows')
```

```python
# Reading the restaurant dataset
restaurant_data = pd.read_csv('/content/Dataset .csv')
```

```python
# Displaying the first few rows of the dataset
restaurant_data.head()
```

| | Restaurant ID | Restaurant Name | Country Code | City | Address | Locality | Locality Verbose | Longitude | Latitude | Cuisines | ... | Currency | Has Table booking | Has Online delivery | Is delivering now | Switch to order menu | Price range | Aggregate rating | Rating color | Rating text | Votes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 6317637 | Le Petit Souffle | 162 | Makati City | Third Floor, Century City Mall, Kalayaan Avenu... | Century City Mall, Poblacion, Makati City | Century City Mall, Poblacion, Makati City, Mak... | 121.027535 | 14.565443 | French, Japanese, Desserts | ... | Botswana Pula(P) | Yes | No | No | No | 3 | 4.8 | Dark Green | Excellent | 314 |
| 1 | 6304287 | Izakaya Kikufuji | 162 | Makati City | Little Tokyo, 2277 Chino Roces Avenue, Legaspi... | Little Tokyo, Legaspi Village, Makati City | Little Tokyo, Legaspi Village, Makati City, Ma... | 121.014101 | 14.553708 | Japanese | ... | Botswana Pula(P) | Yes | No | No | No | 3 | 4.5 | Dark Green | Excellent | 591 |
| 2 | 6300002 | Heat - Edsa Shangri-La | 162 | Mandaluyong City | Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal... | Edsa Shangri-La, Ortigas, Mandaluyong City | Edsa Shangri-La, Ortigas, Mandaluyong City, Ma... | 121.056831 | 14.581404 | Seafood, Asian, Filipino, Indian | ... | Botswana Pula(P) | Yes | No | No | No | 4 | 4.4 | Green | Very Good | 270 |
| 3 | 6318506 | Ooma | 162 | Mandaluyong City | Third Floor, Mega Fashion Hall, SM | SM Megamall, Ortigas, Mandaluyong | SM Megamall, Ortigas, Mandaluyong | 121.056475 | 14.585318 | Japanese, Sushi | ... | Botswana Pula(P) | No | No | No | No | 4 | 4.9 | Dark Green | Excellent | 365 |

```
[ ] # Extracting specific columns for analysis
    restaurant_data.columns

    Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
           'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
           'Average Cost for two', 'Currency', 'Has Table booking',
           'Has Online delivery', 'Is delivering now', 'Switch to order menu',
           'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
           'Votes'],
          dtype='object')
```

```
# Displaying the selected columns
df_restaurants = restaurant_data[['Restaurant ID','Restaurant Name','Cuisines','Aggregate rating','Votes']]
df_restaurants
```

| | Restaurant ID | Restaurant Name | Cuisines | Aggregate rating | Votes |
|---|---|---|---|---|---|
| 0 | 6317637 | Le Petit Souffle | French, Japanese, Desserts | 4.8 | 314 |
| 1 | 6304287 | Izakaya Kikufuji | Japanese | 4.5 | 591 |
| 2 | 6300002 | Heat - Edsa Shangri-La | Seafood, Asian, Filipino, Indian | 4.4 | 270 |
| 3 | 6318506 | Ooma | Japanese, Sushi | 4.9 | 365 |
| 4 | 6314302 | Sambo Kojin | Japanese, Korean | 4.8 | 229 |
| ... | ... | ... | ... | ... | ... |
| 9546 | 5915730 | Naml\ Gurme | Turkish | 4.1 | 788 |
| 9547 | 5908749 | Ceviz A��ac\ | World Cuisine, Patisserie, Cafe | 4.2 | 1034 |
| 9548 | 5915807 | Huqqa | Italian, World Cuisine | 3.7 | 661 |
| 9549 | 5916112 | A���k Kahve | Restaurant Cafe | 4.0 | 901 |
| 9550 | 5927402 | Walter's Coffee Roastery | Cafe | 4.0 | 591 |

9551 rows × 5 columns

## 4. Methodology:

The recommendation system utilized a contentbased filtering approach, specifically employing Jaccard similarity to measure the similarity between user preferences and restaurant features. The implementation involved feature extraction, similarity calculation, and recommendation generation. Evaluation metrics such as precision, recall, and accuracy were used to assess the system's performance.

```
# Function to describe columns in the dataset
def describe_columns():
    column_info = []
    for col in df_restaurants.columns:
        column_info.append(
            [col,
            df_restaurants[col].dtype,
            df_restaurants[col].isna().sum(),
            round(df_restaurants[col].isna().sum()/len(df_restaurants)*100,2),
            df_restaurants[col].nunique(),
            list(df_restaurants[col].drop_duplicates().sample(2).values)]
        )
    data_description = pd.DataFrame(data=column_info,
                            columns=['Column','Data_Type', 'Missing_Values',
                                     'Percent_Missing', 'Unique_Count', 'Sample_Values'])
    return data_description
```

```
# Generating description of columns
describe_columns()
```

| | Column | Data_Type | Missing_Values | Percent_Missing | Unique_Count | Sample_Values |
|---|---|---|---|---|---|---|
| 0 | Restaurant ID | int64 | 0 | 0.00 | 9551 | [18261309, 18464649] |
| 1 | Restaurant Name | object | 0 | 0.00 | 7446 | [Superstar Caf��, Dosa Village] |
| 2 | Cuisines | object | 9 | 0.09 | 1825 | [North Indian, South Indian, Mughlai, Bakery, ... |
| 3 | Aggregate rating | float64 | 0 | 0.00 | 33 | [2.4, 4.0] |
| 4 | Votes | int64 | 0 | 0.00 | 1012 | [908, 219] |

```
# Dropping rows with missing values
df_restaurants = df_restaurants.dropna()
```

```
df_restaurants
```

```
[ ] # Renaming columns for clarity
    df_restaurants = df_restaurants.rename(columns={'Restaurant_ID': 'restaurant_id'})
    df_restaurants = df_restaurants.rename(columns={'Restaurant_Name': 'restaurant_name'})
    df_restaurants = df_restaurants.rename(columns={'Cuisines': 'cuisines'})
    df_restaurants = df_restaurants.rename(columns={'Aggregate_Rating': 'aggregate_rating'})
    df_restaurants = df_restaurants.rename(columns={'Votes': 'votes'})
    df_restaurants
```

| | Restaurant ID | Restaurant Name | cuisines | Aggregate rating | votes |
|---|---|---|---|---|---|
| 0 | 6317637 | Le Petit Souffle | French, Japanese, Desserts | 4.8 | 314 |
| 1 | 6304287 | Izakaya Kikufuji | Japanese | 4.5 | 591 |
| 2 | 6300002 | Heat - Edsa Shangri-La | Seafood, Asian, Filipino, Indian | 4.4 | 270 |
| 3 | 6318506 | Ooma | Japanese, Sushi | 4.9 | 365 |
| 4 | 6314302 | Sambo Kojin | Japanese, Korean | 4.8 | 229 |
| ... | ... | ... | ... | ... | ... |
| 9546 | 5915730 | Naml\ Gurme | Turkish | 4.1 | 788 |
| 9547 | 5908749 | Ceviz A��ac\ | World Cuisine, Patisserie, Cafe | 4.2 | 1034 |
| 9548 | 5915807 | Huqqa | Italian, World Cuisine | 3.7 | 661 |
| 9549 | 5916112 | A���k Kahve | Restaurant Cafe | 4.0 | 901 |
| 9550 | 5927402 | Walter's Coffee Roastery | Cafe | 4.0 | 591 |

9542 rows × 5 columns

```
[ ] df_restaurants.duplicated().sum()
```

```
0
```

```
df_restaurants['Restaurant Name'].duplicated().sum()
```

```
1735
```

```
[ ] # Checking cuisines
    df_restaurants['cuisines'].value_counts()
```

```
cuisines
North Indian    270
Italian         237
Chinese         200
Continental     199
Cafe            177
               ...
Pub Food          1
Durban            1
Irish             1
Persian           1
Sunda             1
Name: count, Length: 128, dtype: int64
```

```
# # Generating cross tabulation of restaurant names and cuisines

cross_tab_resto_cuisines = pd.crosstab(df_restaurants['Restaurant Name'],
                          df_restaurants['cuisines'])
cross_tab_resto_cuisines
```

| cuisines<br>Restaurant Name | Afghani | African | American | Andhra | Arabian | Argentine | Asian | Asian Fusion | Australian | Awadhi | ... | Teriyaki | Tex-Mex | Thai | Tibetan | Turkish | Turkish Pizza | Vegetarian | Vietnamese | Western | World Cuisine |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 'Ohana | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 Downing Street | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11th Avenue Cafe Bistro | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 145 Kala Ghoda | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 Flavours Biryani | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| feel ALIVE | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| sketch Gallery | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| tashas | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Connected to Python 3 Google Compute Engine backend

## 5. Results:

The recommendation system successfully provided personalized restaurant recommendations based on user preferences. Results indicated high precision and recall rates, demonstrating the system's ability to accurately match user preferences with relevant restaurants. Visualizations and tables were used to illustrate the performance of the recommendation system across different user scenarios.

```python
# Checking on restaurant name value
cross_tab_resto_cuisines.loc['feel ALIVE'].values
```

```
array([0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```python
# Sample of Restaurant Names
df_restaurants['Restaurant Name'].sample(20, random_state=101)
```

```
1308                       Mrs. Wilkes' Dining Room
2784                                       Baltazar
888                                       Rose Cafe
2713                             Big City Bread Cafe
1162                                   Olive Bistro
221                               Transmetropolitan
1403                             Maxims Pastry Shop
1381                                         Meraki
1363                               Mimi's Bakehouse
2466                               Cappuccino Blast
1169                                  Oh So Stoned!
1671                        Karak�_y G�_ll�_o��lu
147                                      Via Delhi
209                       Tu-Do Vietnamese Restaurant
258        Tian - Asian Cuisine Studio - ITC Maurya
2649                              Boise Fry Company
247                               Ting's Red Lantern
1170                                   Odeon Social
319                                      The Sizzle
690                                  Sree Annapoorna
Name: Restaurant Name, dtype: object
```

```python
# Measuring similarity between two restaurants using Jaccard similarity score
print(jaccard_score(cross_tab_resto_cuisines.loc["Olive Bistro"].values,
                    cross_tab_resto_cuisines.loc["Rose Cafe"].values))
```

```
0.3333333333333333
```

# Recommendation

```python
# Input Initial Restaurant Name for recommendation
initial_restaurant = 'Ooma'

# Calculating similarity scores between the initial restaurant and other restaurants
similarity_scores = df_jaccard.loc[initial_restaurant].sort_values(ascending=False)

# Creating DataFrame to store top 5 similar restaurants
similarity_scores = pd.DataFrame({'Restaurant Name': similarity_scores.index, 'similarity_score': similarity_scores.values})
similarity_scores = similarity_scores[(similarity_scores['Restaurant Name'] != initial_restaurant) & (similarity_scores['similarity_score'] >= 0.7)].head(5)

# Merging with aggregate ratings to get final recommendation
recommended_restaurants = pd.merge(similarity_scores, df_restaurants[['Restaurant Name','Aggregate rating']],how='inner',on='Restaurant Name')
final_recommendations = recommended_restaurants.sort_values('Aggregate rating',ascending=False).drop_duplicates('Restaurant Name',keep='first')
final_recommendations
```

| | Restaurant Name | similarity_score | Aggregate rating |
|---|---|---|---|
| 0 | Sushi Masa | 1.0 | 4.9 |
| 2 | Nobu | 1.0 | 4.4 |
| 4 | Ichiban | 1.0 | 4.3 |
| 8 | Osaka | 1.0 | 4.2 |
| 6 | Guppy | 1.0 | 4.1 |

**The data above shows the top 5 recommended restaurants with the best ratings. The ratings are curated to include only those that are 4 and above, ensuring the recommendation system provides objectively good ratings.**

## 6. Discussion:

Interpretation of the results highlighted the strengths and limitations of the recommendation system. While the system demonstrated promising performance, challenges such as data sparsity and coldstart problems were identified. Strategies to address these challenges and opportunities for future research were discussed, including incorporating user feedback mechanisms and integrating external data sources.

## 7. Challenges Faced:

Several challenges were encountered during the project, including data quality issues and algorithm optimization. Strategies such as data augmentation and collaboration with domain experts were employed to overcome these challenges. Lessons learned from addressing these challenges were documented for future projects in similar domains.

## 8. Future Work:

Future research directions include enhancing the recommendation system's accuracy, scalability, and usability. Opportunities for incorporating advanced machine learning techniques, integrating realtime user feedback, and expanding the system's scope to include additional features were identified. Collaboration with industry partners and user studies were proposed to validate the system's effectiveness in realworld settings.

## 9. References:


- Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

- Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56-58.

- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).

- Lops, P., de Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In Recommender systems handbook (pp. 73-105). Springer, Boston, MA.


## 10. Appendices:

**Appendix A: Code Snippets**

```
# Sample code snippet for Jaccard similarity calculation
def jaccard_similarity(set1, set2):
   intersection = len(set1.intersection(set2))
   union = len(set1.union(set2))
   return intersection / union
```

### Appendix B: Data Preprocessing Steps

- Missing value imputation: Used mean/mode imputation for missing values in aggregate rating column.
- Encoding categorical variables: Applied onehot encoding to convert categorical variables such as cuisine type into numerical format.
- Filtering restaurants: Removed restaurants with aggregate rating below 4.0 to ensure recommendations are based on highquality establishments.

### Appendix C: Evaluation Metrics

- Precision: Number of relevant items recommended divided by the total number of recommended items.
- Recall: Number of relevant items recommended divided by the total number of relevant items.
- Accuracy: Proportion of correctly predicted recommendations out of the total number of predictions made.

### Appendix D: Visualization

- Precision-Recall Curve: Plot illustrating the tradeoff between precision and recall for different threshold values.
- Confusion Matrix: Matrix representation of the model's performance, showing true positive, true negative, false positive, and false negative predictions.

## 11. Conclusion:

In conclusion, the project successfully developed a personalized restaurant recommendation system based on user preferences. The system's effectiveness in addressing user needs and enhancing the restaurant selection process was demonstrated through comprehensive evaluation and analysis. Recommendations for future research and practical applications of recommendation systems in diverse domains were provided.