

Restaurant Recommendation System

Prepared For

Smart-Internz

Applied Data Science Guided project

By

Swaroop Kalagouda Patil

D Y Patil Agriculture and Technical University Talsande

On

1 June 2025

Abstract

This project develops a personalized restaurant recommendation system based on user preferences, location, and dining history. It analyzes factors such as cuisine, price, and ratings to suggest suitable dining options. Machine learning techniques like collaborative and content-based filtering are used for accurate suggestions. The system enhances the dining experience by offering relevant and location-aware recommendations

Final Project Report

Contents

1. **Introduction**
 - 1.1 Project Overviews
 - 1.2 Objectives
2. **Project Initialization and Planning Phase**
 - 2.1 Define Problem Statement
 - 2.2 Project Proposal (Proposed Solution)
 - 2.3 Initial Project Planning
3. **Data Collection and Preprocessing Phase**
 - 3.1 Data Collection Plan and Raw Data Sources Identified
 - 3.2 Data Quality Report
 - 3.3 Data Preprocessing
4. **Model Development Phase**
 - 4.1 Model Selection Report
 - 4.2 Initial Model Training Code, Model Validation and Evaluation Report
5. **Model Optimization and Tuning Phase**
 - 5.1 Tuning Documentation
 - 5.2 Final Model Selection Justification
6. **Results**
 - 6.1 Output Screenshots
7. **Advantages & Disadvantages**
 - Advantages
 - Disadvantages
8. **Conclusion**
9. **Future Scope**
10. **Appendix**
 - 10.1 Source Code
 - 10.2 GitHub & Project Video Demo Link

1 Introduction

1.1 Project overviews

The **Restaurant Recommendation System** is a smart, data-driven solution designed to help users efficiently discover restaurants that align with their unique preferences and situational contexts. As urbanization and mobile technologies continue to reshape consumer behavior, users are often overwhelmed by the sheer volume of available dining choices across platforms such as Google, Yelp, and Zomato. This leads to **decision fatigue** and suboptimal dining experiences. To solve this, the proposed system leverages a **hybrid recommendation model** combining collaborative filtering, content-based filtering, and geolocation-aware services. The **collaborative filtering component** analyzes historical user behavior, including past restaurant visits, ratings, and interaction patterns, to identify users with similar tastes and recommend restaurants favored by like-minded individuals. Meanwhile, the **content-based filtering module** evaluates restaurant attributes—such as cuisine type, price range, ambiance, and dietary offerings—to match them with explicit user preferences. To enhance practicality, **geolocation data** is integrated using GPS APIs or IP-based location tracking. This allows the system to dynamically adapt its recommendations based on the user's current position or a specified location, ensuring that results are both **relevant and accessible**. For example, a user seeking budget-friendly vegan food in a new city would receive highly localized and personalized recommendations.

Furthermore, the system is designed with **adaptive learning capabilities**. Using techniques like reinforcement learning or preference feedback loops, the recommendation engine improves over time by understanding user behavior patterns, modifying weightage of features, and incorporating real-time feedback such as likes, bookmarks, or direct reviews.

1.2 Objectives

1. **To design and implement a recommendation engine** that effectively filters and ranks restaurants based on individual user preferences, including food type, cost, ambiance, and dietary needs.
2. **To apply machine learning models**, such as collaborative filtering (user-based and item-based) and content-based filtering, to identify patterns in user behavior and restaurant attributes.
3. **To incorporate location-aware features** using GPS or user-inputted location data, ensuring that recommended restaurants are conveniently accessible to the user.
4. **To gather and analyze restaurant reviews and ratings** from public sources (e.g., Yelp, Google Reviews, or internal datasets) to improve the trustworthiness and relevance of suggestions.
5. **To create a user-friendly interface** that allows users to input preferences, view recommended restaurants, and interact with the system seamlessly.
6. **To develop a feedback mechanism** that collects user satisfaction data post-visit to refine future recommendations and enhance personalization over time.
7. **To ensure scalability and adaptability** of the system for use in different geographic regions or for integration into existing food delivery or travel applications.

2 Project Initialization and Planning Phase

2.1 Define Problem Statement

2 Project Initialization and Planning Phase

Problem Statements (Restaurant Recommendation system):

| PS No. | I am (Customer) | I'm trying to | But | Because | Which makes me feel |
|--------|-----------------------------|---|--|--|-------------------------------------|
| PS- 1 | A tourist in a new city | Find good local restaurants | I don't know the area well | I lack local knowledge or reviews | Confused and unsure of where to eat |
| PS- 2 | A vegetarian diner | Get recommendations for veg-only restaurants | Most apps show mixed cuisine places | I want strict dietary options | Frustrated and unsupported |
| PS- 3 | A restaurant owner | Attract more customers through recommendation platforms | My restaurant is not being recommended often | The system doesn't promote new or small businesses | Invisible and discouraged |
| PS- 4 | A student on a tight budget | Find affordable but tasty restaurants | Expensive options are shown first | Filters don't prioritize price or value | Overwhelmed and discouraged |
| PS- 5 | A delivery app user | Get suggestions based on past orders | It doesn't adapt to my taste | The system lacks learning | Frustrated by repetition |
| PS- 6 | A parent of young kids | Find kid-friendly and hygienic restaurants | No way to filter for child-friendly | Lack of safety and family-focused features amenities | Anxious about experience |

| | | | | | |
|-------------|--------------------------|--|--|---------------------------------------|---------------------------------|
| PS-7 | A small restaurant owner | Increase customer footfall via platforms | My business is buried under chain listings | Ranking algorithms favor large brands | Discouraged and invisible guide |
|-------------|--------------------------|--|--|---------------------------------------|---------------------------------|

| | | | | | |
|--------------|----------------------------------|--|--|--|--|
| PS-8 | A new-in-town resident | Explore culturally diverse food options | Unaware of hidden gems in my area | No cultural/ethnic tags or user reviews | Disconnected and bored of same cuisine |
| PS-9 | A food delivery platform analyst | Monitor food safety and restaurant quality | Can't verify ingredient safety from menus | Platforms lack AI food item scanners or trackers | Concerned about consumer trust |
| PS-10 | A data scientist | Analyze food trends from reviews | Datasets are messy, biased, or unavailable | Lack of structured sentiment and metadata | Blocked in model building and research |
| PS-11 | A foodie traveler | Find top-rated local restaurants in new cities | Recommendations don't match my taste or location | Generic, irrelevant suggestions | Frustrated and unsure where to eat |
| PS-12 | A restaurant owner | Improve my visibility on food apps | My reviews are outdated or low-rated | I can't easily respond or update info | Powerless and misrepresented |
| PS-13 | A health-conscious customer | Find healthy eating options nearby | Menus and calorie info are missing | I can't make informed decisions | Disconnected from my health goals |

| | | | | | |
|--------|------------------------------|--|--|--|--|
| PS- 14 | A healthconscious individual | Track the health benefits of different mushrooms | I can't identify what's in the store or dish | There's no easy app for instant scanning | Disappointed and disconnected from my health goals |
|--------|------------------------------|--|--|--|--|

1.1 Project Proposal (Proposed Solution)

Project Proposal (Proposed Solution)

This project proposal outlines a solution to address a specific problem. With a clear objective, defined scope, and a concise problem statement, the proposed solution details the approach, key features, and resource requirements, including hardware, software, and personnel

| Project Overview | |
|-------------------|--|
| Objective | To develop a system that provides personalized and efficient restaurant recommendations by analyzing user preferences, dietary requirements, location, and budget. |
| Scope | The project aims to serve users seeking restaurant suggestions that match their individual lifestyle choices and dining preferences. It will operate across various regions, considering real-time data and qualitative reviews. |
| Problem Statement | |
| Description | Finding restaurants tailored to specific needs is often time-consuming and inefficient. Users frequently revisit the same places, missing diverse options that better match their preferences. |
| Impact | Solving this problem improves user satisfaction, encourages exploration of new dining options, and reduces time spent on decision-making. |
| Proposed Solution | |
| Approach | The solution employs innovative recommendation algorithms that factor in both user input and external data like ambiance, ratings, and reviews. It adapts dynamically to user feedback and real-time changes. |
| Key Features | <ul style="list-style-type: none"> • Personalized recommendations • Real-time data analysis • Integration of user reviews • Consideration of dietary and budget constraints • Scalable infrastructure |

Resource Requirements

| Resource Type | Description | Specification/Allocation |
|-------------------------|--|--|
| Hardware | | |
| Computing Resources | 8-core CPUs and optional GPU | 2 x NVIDIA V100 GPUs |
| Memory | RAM | Minimum 8 GB RAM |
| Storage | SSD | 1 TB SSD for storing user data and restaurant metadata |
| Software | | |
| Frameworks | Python frameworks | Python, Flask |
| Libraries | Additional libraries | Pandas, NumPy, Scikit-learn, TensorFlow, BeautifulSoup (for scraping), and NLTK (for review analysis) |
| Development Environment | IDE, version control | Jupyter Notebook |
| Data | | |
| Data | <p>Size: - Approx. 50,000–100,000 records initially; scalable based on user growth,</p> <p>Format: - CSV for tabular datasets, Text/HTML for scraped reviews</p> | Aggregated from crowdsourced restaurant platforms (e.g., Yelp, Zomato APIs), user feedback, and public review datasets |

1.2 Initial Project Planning

Product Backlog, Sprint Schedule, and Estimation

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Sprint Start Date | Sprint End Date (Planned) |
|----------|-------------------------------|-------------------|--|--------------|----------|-------------------|---------------------------|
| Sprint-1 | User Preferences Input | USN-1 | As a user, I can enter my food or Hotel preferences. | 2 | High | 01 June 2025 | 02 June 2025 |
| Sprint-1 | Recommendation Engine | USN-2 | As a user, I can get restaurant recommendations based on my preferences. | 3 | High | 02 June 2025 | 02 June 2025 |
| Sprint-2 | Review & Rating Integration | USN-3 | As a user, I can view restaurant reviews and ratings fetched from the dataset. | 2 | Medium | 03 June 2025 | 04 June 2025 |
| Sprint-2 | UI/UX Enhancement | USN-4 | As a user, I can view results in a user-friendly interface with filters and sorting. | 2 | Medium | 04 June 2025 | 05 June 2025 |

2 Data Collection and Preprocessing Phase

2.1 Data Collection Plan and Raw Data Sources Identified

Data Collection Plan

| Section | Description |
|-----------------------------|---|
| Project Overview | Develop a restaurant recommendation system to assist users in finding dining options based on their preferences, location, and other relevant factors. By analyzing user preferences, restaurant ratings, and location data, this project aims to provide personalized recommendations that enhance the dining experience for users. |
| Data Collection Plan | The dataset used for this project was sourced from Kaggle and contains detailed information on over 9,000 restaurants in Bangalore, including attributes like name, location, cuisine, ratings, and pricing. This publicly available dataset was collected to support analysis and predictive modeling related to restaurant ratings and customer preferences. |
| Raw Data Sources Identified | The raw data for this project was obtained from the Kaggle dataset titled “Zomato Bangalore Restaurants” by Himanshu Poddar. The dataset is publicly available at https://www.kaggle.com/datasets/himanshupoddar/zomato-bangalore-restaurants and includes key restaurant-related attributes such as restaurant names, locations, cuisines, average costs, online delivery availability, and user ratings. |

Raw Data Sources

| Source Name | Description | Location/URL | Format | Size | Access Permissions |
|-------------------------------|--|---|--------|--------|--------------------|
| SmartInternz Provided Dataset | Restaurant-level data including name, location, cuisines, rating and cost. | Data-Set zomato-bangalore-restaurants | CSV | ~ 93MB | Public |

2.2 Data Quality Report

| Data Source | Data Quality Issue | Severity | Resolution Plan |
|---|---|----------|--|
| Dataset (Restaurant reviews and metadata) | Missing values in fields like restaurant name, location, or ratings | Moderate | Perform data imputation using techniques like mean/mode for numeric values and most frequent value for categorical data. Alternatively, remove rows with critical missing fields. |
| Dataset (User reviews) | Duplicate user review entries | Low | Remove duplicate records using <code>drop_duplicates()</code> in pandas or SQL <code>DISTINCT</code> queries. Use datetime parsing libraries (e.g., <code>pandas.to_datetime</code>) to standardize all date/time fields. |
| Dataset (Restaurant metadata) | Inconsistent formats (e.g., location written in different ways like "NY", "New York") | Moderate | Apply data standardization techniques, using string functions or regex patterns to unify the format. |
| Dataset (User preferences) | Sparse data or insufficient user history | High | Implement fallback strategies such as popularity-based or content-based recommendations when user data is lacking. |

2.3 Data Preprocessing

Data Preprocessing

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

| Section | Description |
|--|--|
| Data Overview | The dataset contains restaurant information from Zomato, including name, reviews, ratings, cuisines, cost, and more. The data is cleaned, deduplicated, and preprocessed for building a content-based recommendation system. |
| Resizing | <i>Not applicable for text data.</i> |
| Normalization | Ratings are normalized to a 1-5 scale using MinMaxScaler. Text is lowercased and punctuation is removed. |
| Data Augmentation | Not applicable for text data. |
| Denoising | Text is cleaned by removing newline characters and punctuation. |
| Edge Detection | Not applicable for text data. |
| Color Space Conversion | Not applicable for text data. |
| Image Cropping | Not applicable for text data. |
| Batch Normalization | Not applicable for text data. |
| Data Preprocessing Code Screenshots | |

| | |
|------------------------|---|
| Loading Data | <pre> # Mounting Google Drive #from google.colab import drive #drive.mount('/content/drive') import csv # Specifying the path to the dataset file file_path = '/content/zomato.csv' # Reading the dataset into a Pandas DataFrame #df = pd.read_csv(file_path,encoding = 'ISO-8859-1', low_memory = False) df = pd.read_csv(file_path, encoding='ISO-8859-1', on_bad_lines='skip', engine='python') # Displaying the first few rows of the dataset to ensure it's loaded correctly df.head() </pre> <p>Python</p> |
| Resizing | <i>Not applicable</i> |
| Normalization | <pre> # Computing Mean Rating restaurants = list(df['name'].unique()) df['Mean Rating'] = 0 for i in range(len(restaurants)): df['Mean Rating'][df['name'] == restaurants[i]] = df['rate'][df['name'] == restaurants[i]].mean() #Scaling the mean rating values from sklearn.preprocessing import MinMaxScaler scaler = MinMaxScaler (feature_range = (1,5)) df[['Mean Rating']] = scaler.fit_transform(df[['Mean Rating']]).round(2) </pre> |
| Data Augmentation | <i>Not applicable</i> |
| Denoising | <pre> ## Lower Casing df["reviews_list"] = df["reviews_list"].str.lower() ## Removal of Punctuations import string PUNCT_TO_REMOVE = string.punctuation def remove_punctuation(text): """custom function to remove the punctuation""" return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE)) df["reviews_list"] = df["reviews_list"].apply(lambda text: remove_punctuation (text)) </pre> <p>Python</p> |
| Edge Detection | <i>Not applicable</i> |
| Color Space Conversion | <i>Not applicable</i> |
| Image Cropping | <i>Not applicable</i> |
| Batch Normalization | <i>Not applicable</i> |

4. Model Development Phase

2.4 Model Selection Report

| Model | Description |
|--|--|
| Content-Based Filtering | Content-based filtering recommends restaurants by comparing user preferences (e.g., cuisine type, price range, dietary restrictions) with restaurant attributes. It focuses on similarities between items and the user's profile without relying on other users' data. This method is effective for users with unique tastes but may struggle with limited user profiles (cold start). |
| Collaborative Filtering | Collaborative filtering leverages the preferences of similar users to make recommendations. It uses historical ratings and reviews to identify patterns. This model is effective in discovering new items but can suffer from sparsity and cold start problems if data is limited. |
| Hybrid Recommendation Model | This combines content-based and collaborative filtering to overcome the limitations of each method. By integrating both user preference data and behavior of similar users, hybrid models improve recommendation accuracy, diversity, and scalability. It is particularly useful in scenarios with large, sparse datasets like restaurant recommendations. |
| Matrix Factorization | Matrix factorization techniques decompose the user-item interaction matrix into latent features, capturing underlying patterns in user preferences. Singular Value Decomposition (SVD) is a common approach. It is computationally efficient and works well for large datasets but requires enough ratings. |
| Deep Learning (Neural Networks) | Neural networks can be used to build recommendation systems by learning complex, non-linear relationships between users and restaurants from rich feature sets including reviews, preferences, and metadata. While powerful, they require large datasets and are computationally intensive. |

Conclusion:

| Model Selected | |
|------------------------------------|--|
| Hybrid Recommendation Model | The hybrid model was selected because it addresses the limitations of both content-based and collaborative filtering approaches. It effectively handles the cold start and sparsity issues by integrating multiple data sources such as user profiles, restaurant attributes, and behavioral data. This results in more personalized, diverse, and accurate recommendations, making it highly suitable for a restaurant recommendation system with varying user preferences and data availability. |

2.5

Initial Model Training Code, Model Validation and Evaluation Report

Initial Model Training Code, Model Validation and Evaluation Report

Initial Model Training Code (5 marks):

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
df_percent.set_index('name', inplace=True)
indices = pd.Series(df_percent.index)

# Creating tf-idf matrix
tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0, stop_words=None)
tfidf_matrix = tfidf.fit_transform(df_percent['reviews_list'])

cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
```

Model Validation and Evaluation Report (5 marks):

| Model | Summary | Training and Validation Performance Metrics |
|---------|------------------------------|---|
| Model 1 | Content-based Recommendation | Training Metrics -None (unsupervised, no explicit training phase) Validation Metrics - None (recommendations are inspected manually) |

3 Model Optimization and Tuning Phase

3.1 Tuning Documentation

Hyperparameter Tuning

| Model | Tuned Hyperparameters |
|--|--|
| Model 1: Content-Based Filtering | <p>- Similarity Metric: Cosine similarity was used as the primary metric to compute similarity between restaurants based on features like cuisines, rating, and cost.</p> <p>- Top N Recommendations: The number of top similar restaurants returned was tested with values like 5, 10, and 15.</p> <pre> 1 def recommend(name, cosine_similarities = cosine_similarities): 2 3 # Create a list to put top restaurants 4 recommend_restaurant = [] 5 6 # Find the index of the hotel entered 7 idx = indices[indices == name].index[0] 8 9 # Find the restaurants with a similar cosine-sim value and order them from biggest number 10 score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending=False) 11 12 # Extract top 30 restaurant indexes with a similar cosine-sim value 13 top30_indexes = list(score_series.iloc[0:31].index) 14 15 # Names of the top 30 restaurants 16 for each in top30_indexes: 17 recommend_restaurant.append(list(df_percent.index)[each]) 18 19 # Creating the new data set to show similar restaurants 20 df_new = pd.DataFrame(columns=['cuisines', 'Mean Rating', 'cost']) 21 22 # Create the top 30 similar restaurants with some of their columns 23 for each in recommend_restaurant: 24 df_new = df_new.append(pd.DataFrame(df_percent[['cuisines', 'Mean Rating', 'cost']][df_percent.index == each].sample())) 25 26 # Drop the same named restaurants and sort only the top 10 by the highest rating 27 df_new = df_new.drop_duplicates(subset=['cuisines', 'Mean Rating', 'cost'], keep=False) 28 df_new = df_new.sort_values(by='Mean Rating', ascending=False).head(10) 29 30 print('TOP %s RESTAURANTS LIKE %s WITH SIMILAR REVIEWS: ' % (str(len(df_new)), name)) 31 df_new.index = df_new.index.str.lower() 32 return df_new </pre> |

| | |
|--|---|
| Model 2: Collaborative Filtering | <ul style="list-style-type: none"> - Algorithm: SVD (Singular Value Decomposition) from the Surprise library. - Learning Rate: Tuned values such as 0.005, 0.01, and 0.02 were tested. - Regularization: Parameters such as 0.02, 0.05 were tried to avoid overfitting. - Number of Epochs: Adjusted between 20 and 100 epochs. |
|--|---|

Final Model Selection Justification

Final Model Selection Justification:

| Final Model | Reasoning |
|--|---|
| Model 1: Content-Based Filtering | Selected due to its simplicity and good performance without requiring detailed user history. It gave interpretable and relevant results using restaurant features like cuisines, ratings, and cost. |

4 Results

4.1 Output Screenshots

Home Page:


Restaurant Recommendation SystemHomeRecommend

Build Recommendation System with ease!!

In this age of information overload, people use a variety of strategies to make choices about what to buy, how to spend their leisure time, and even where to go. Recommender systems automate some of these strategies with the goal of providing affordable food items. The aim is to create a content-based recommender system in which when we write a restaurant name, the recommender system will look at the reviews of other restaurants, and the system will recommend us other restaurants with similar reviews and sort them from the highest-rated.

Test the System

[Recommend](#)



Input Page:

Example :-

Restaurant Recommendation SystemHomeRecommend

Restaurant Name

[Click to see the recommendation](#)

Output:

Restaurant Recommendation System

[Home](#)[Recommend](#)

Here are the top recommended restaurants

| Name | Cuisines | Mean Rating (out of 5) | Cost (in thousands) |
|-------------------------------|---|------------------------|---------------------|
| The Black Pearl | north indian european mediterranean bbq | 4.85 | 1.5 |
| Barbeque Nation | north indian european mediterranean bbq kebab | 4.7 | 1.6 |
| Hunger Camp | north indian south indian chinese seafood | 4.56 | 1.3 |
| Hakuna Matata | north indian asian seafood chinese | 4.41 | 1.2 |
| Jalsa Gold | north indian mughlai italian | 4.41 | 1.3 |
| Deja Vu Resto Bar | north indian italian | 4.26 | 900.0 |
| Tipsy Bull - The Bar Exchange | north indian chinese continental mexican | 4.26 | 1.4 |
| Dhaba Estd 1986 Delhi | north indian | 4.26 | 1.1 |
| Float | north indian japanese | 4.26 | 1.5 |
| nu.tree | north indian healthy food beverages | 4.26 | 400.0 |

5 Advantages & Disadvantages

Advantages:

- **Personalized User Experience:** Tailors dining options based on user preferences, dietary needs, and previous behaviour.
- **Time-saving:** Reduces the effort needed to search and choose a restaurant.
- **Improved Discoverability:** Helps smaller or new restaurants gain visibility through recommendations.
- **Data-Driven Decisions:** Uses user ratings, reviews, and location data to make informed suggestions.
- **Enhanced Customer Satisfaction:** Users are more likely to enjoy their meals when recommendations align with their preferences

Disadvantages:

- **Privacy Concerns:** Collecting and analyzing user data (location, preferences) can raise privacy issues.
- **Bias in Recommendations:** Algorithms might favor sponsored listings or high-traffic restaurants, reducing diversity.
- **Dependence on User Data:** Inaccurate or limited data can lead to poor recommendations.
- **Over-Personalization:** Users might be confined to similar choices, missing out on new or diverse dining experiences.
- **Scalability Issues:** Maintaining system accuracy and performance can become challenging as the user base grows.

6 Conclusion

A restaurant recommendation system is a powerful tool for enhancing the dining experience by delivering tailored suggestions based on user behavior, preferences, and location. While it offers significant benefits such as convenience, personalization, and efficient decision-making, it also presents challenges including data privacy, system bias, and the risk of user data dependency. Future advancements in AI, real-time analytics, and user interface technologies promise to make such systems more intelligent, inclusive, and immersive. With careful implementation and ethical considerations, this system can transform how users explore and enjoy culinary options.

7 Future Scope

- **Integration with AR/VR:** In the future, users could take virtual tours of restaurants or view their ambiance in AR before booking.
- **Voice Assistant Compatibility:** Integration with Siri, Alexa, or Google Assistant to provide hands-free restaurant suggestions.
- **Enhanced Personalization:** Use deep learning and behavioral analytics to refine suggestions based on dietary restrictions, allergies, and eating habits.
- **Real-time Data Utilization:** Incorporating real-time factors like wait times, special offers, and crowd density for more dynamic recommendations.
- **Multilingual Support:** Expanding the system to support various languages to cater to a global audience.
- **Social Media Integration:** Use of social media trends and check-ins to improve recommendation relevance.
- **Sustainability Preferences:** Factoring in eco-conscious dining choices (e.g., locally sourced, plant-based, or low-waste restaurants).

8 Appendix

8.1 Source Code

[<https://github.com/swaroop897/restaurant-recommendation-system>]

8.2 Project Video Demo Link :

Video Demo Link:

[https://drive.google.com/file/d/14yk4WJaUf_zjX9R067jbVg1uD-2ks697/view?usp=sharing]