Swiggy Restaurant Recommendation System using Machine Learning and Streamlit

Problem statement:

Food delivery platforms like Swiggy offer thousands of restaurants across multiple cities, making it challenging for users to discover suitable dining options based on their individual preferences. With increasing choices, customers expect personalized recommendations that match their tastes, budget, and location. A recommendation system solves this challenge by analyzing restaurant features and suggesting the most relevant options to the user.

The objective of this project is to build an intelligent restaurant recommendation system using the Swiggy dataset. The system recommends restaurants based on key factors such as city, cuisine type, average rating, rating count, and cost for two. The approach includes data cleaning, preprocessing, feature encoding, similarity computation, clustering, and deploying a user-friendly Streamlit application.

The dataset contains more than 148,000+ restaurant entries, each described with attributes such as name, city, cuisine, cost, rating, rating count, address, and menu link. However, the raw data contains inconsistencies such as missing values, non-numeric fields (e.g., "1K+ ratings", "Too Few Ratings", "₹ 200"), and unstructured cuisine information. These issues must be resolved through data cleaning and preprocessing to ensure accurate model performance.

After cleaning, categorical features (city and cuisine) are converted into numerical format using One-Hot Encoding and Multi-Hot Encoding. Numerical features such as rating, cost, and rating count are scaled using MinMaxScaler to standardize the feature range. The processed data is then used to build two recommendation models:

1. **Cosine Similarity Model** – Identifies restaurants that are most similar to the user's preferences.
2. **K-Means Clustering Model** – Groups restaurants into clusters based on patterns and recommends within the closest cluster.

Finally, a fully interactive Streamlit web application is developed to provide real-time restaurant recommendations. Users can select their preferred city, enter a cuisine type, choose a rating threshold, set cost limits, and instantly view the top recommended restaurants. The app presents details such as restaurant name, rating, cost, address, and Swiggy link.

This project demonstrates practical skills in data preprocessing, encoding, clustering, similarity modeling, and full-stack ML application deployment, making it valuable for real-world recommendation system development.

Data set Description:

The dataset consists of **148,541 restaurant entries** collected from Swiggy and spans multiple cities across India, offering a diverse and extensive foundation for building a recommendation system. It includes a wide variety of cuisines, with more than **100 cuisine types**, often appearing as combined entries such as "North Indian, Chinese" or "Fast Food, Pizza." The rating data contains both valid numerical values and placeholders like "--", indicating missing ratings. Similarly, the rating count field appears in multiple formats, including "50+", "100+", "1K+", and "Too Few Ratings", requiring careful preprocessing to extract meaningful numeric values. The cost column also includes currency symbols (e.g., "₹ 200"), which need to be cleaned and converted into numerical format for analysis. Due to its rich attributes and real-world inconsistencies, this dataset is ideal for developing recommendation systems based on content filtering, similarity measures, clustering, and machine learning techniques.

Data Cleaning:

The raw Swiggy dataset contained several inconsistencies and formatting issues that needed to be addressed before analysis and model development. The first step in the data cleaning process was handling missing values. A small number of restaurants had missing names or ratings, and these entries were removed since they lacked essential identifying information. The rating column contained non-numeric placeholders such as "--", which were converted to NaN and replaced with the median rating value to maintain consistency. The rating_count column required additional preprocessing because values appeared in formats like "50+ ratings", "1K+ ratings", and "Too Few Ratings". These were converted into appropriate numeric values by extracting the numbers and converting "K" values into thousands, while "Too Few Ratings" was treated as zero.

The **cost** column also contained currency symbols such as "₹ 200". These symbols were removed, and the values were converted into numeric format to allow mathematical operations during recommendation calculations. The cuisine column, which often included multiple cuisines separated by commas, was cleaned and standardized by trimming extra spaces and preparing it for later encoding. Duplicate rows were checked, but none were found in the dataset. The final cleaned dataset contained structured, reliable, and machine-readable values, making it suitable for the preprocessing and modeling stages. This cleaned file was saved as cleaned_data.csv for subsequent steps.

Data Preprocessing:

After completing the data cleaning stage, the next step was to transform the dataset into a machine-readable numerical format suitable for modeling. This involved encoding categorical features, handling multi-label fields, and scaling numerical values. The **city** column, which contains textual location names, was converted into multiple binary columns using One-Hot Encoding. Each city became its own column (e.g., `city_Chennai`, `city_Mumbai`), and restaurants were assigned a value of 1 or 0 depending on whether they belonged to that city. This method allows machine learning models to interpret city information numerically without implying any ranking between cities.

The cuisine column required a more advanced transformation because many restaurants list multiple cuisines (e.g., "Chinese, Fast Food, North Indian"). To handle this, the text was split into

individual cuisines and encoded using Multi-Label Binarization, generating multiple columns such as `cuisine_chinese`, `cuisine_fast food`, and `cuisine_north indian`. This ensures that restaurants offering multiple cuisines are represented accurately in the feature space.

Numerical columns such as rating, rating_count, and cost were converted into comparable scales using the MinMaxScaler technique. Scaling ensures that all numeric values fall between 0 and 1, preventing features with larger numerical ranges from dominating the recommendation model's calculations. Finally, all transformed components—scaled numeric values, one-hot encoded cities, and multi-hot encoded cuisines—were combined into a unified dataset. This preprocessed dataset was saved as encoded_data.csv, while the encoders (MultiLabelBinarizer, scaler, and city column information) were saved in encoder.pkl to ensure the same transformations can be applied to user inputs in the Streamlit application. This preprocessing step prepares the data for clustering and similarity-based recommendation algorithms.

Recommendation Engine:

The recommendation engine is the core component of this project, responsible for suggesting the most relevant restaurants to users based on their preferences. To achieve this, two different recommendation approaches were implemented: Cosine Similarity–based recommendations and K-Means Clustering–based recommendations. Both methods rely on the fully preprocessed numerical dataset to measure similarity and group restaurants effectively.

1. Cosine Similarity Model

Cosine similarity is a widely used technique in recommendation systems because it measures how similar two data points are, regardless of their absolute values. In this project, each restaurant is represented as a numeric vector containing information such as cuisine type, city encoding, rating, rating count, and cost. When a user enters their preferences (city, cuisine, rating, and cost), the input is also converted into a similar vector using the same preprocessing steps (encoder.pkl). The cosine similarity score is then calculated between the user vector and every restaurant vector in the dataset. Restaurants with the highest similarity scores are considered the best matches and are returned as recommendations. This method provides highly accurate suggestions because it focuses on the direction of similarity rather than magnitude.

2. K-Means Clustering Model

K-Means clustering offers another approach to generating recommendations by grouping similar restaurants together. Using the encoded dataset, restaurants are clustered into a predefined number of groups based on their features. Each cluster represents a unique combination of cuisine, cost, rating tendencies, and city patterns. When a user submits their preferences, the user vector is assigned to the closest cluster using the trained K-Means model. Only restaurants within that cluster are considered for recommendations, and cosine similarity is then applied within the cluster to produce the top results. This method greatly improves efficiency because it reduces the search space to a smaller group of similar restaurants.

3. User Input Mapping

To ensure accurate recommendations, user inputs must undergo the same transformations applied to the dataset. The system uses the stored encoders and scaler (in **encoder.pkl**) to convert the user's city, cuisine, rating, and cost into a consistent numerical format. This ensures that the recommendation engine interprets the user inputs correctly and compares them fairly against restaurant vectors.

4. Output Generation

The recommendation engine returns results by mapping the selected restaurant indices back to the original **cleaned_data.csv**, ensuring the output contains meaningful human-readable details such as restaurant name, cost, rating, address, and Swiggy link. This allows the final recommendation list to be displayed clearly in the Streamlit interface.

Streamlit Application:

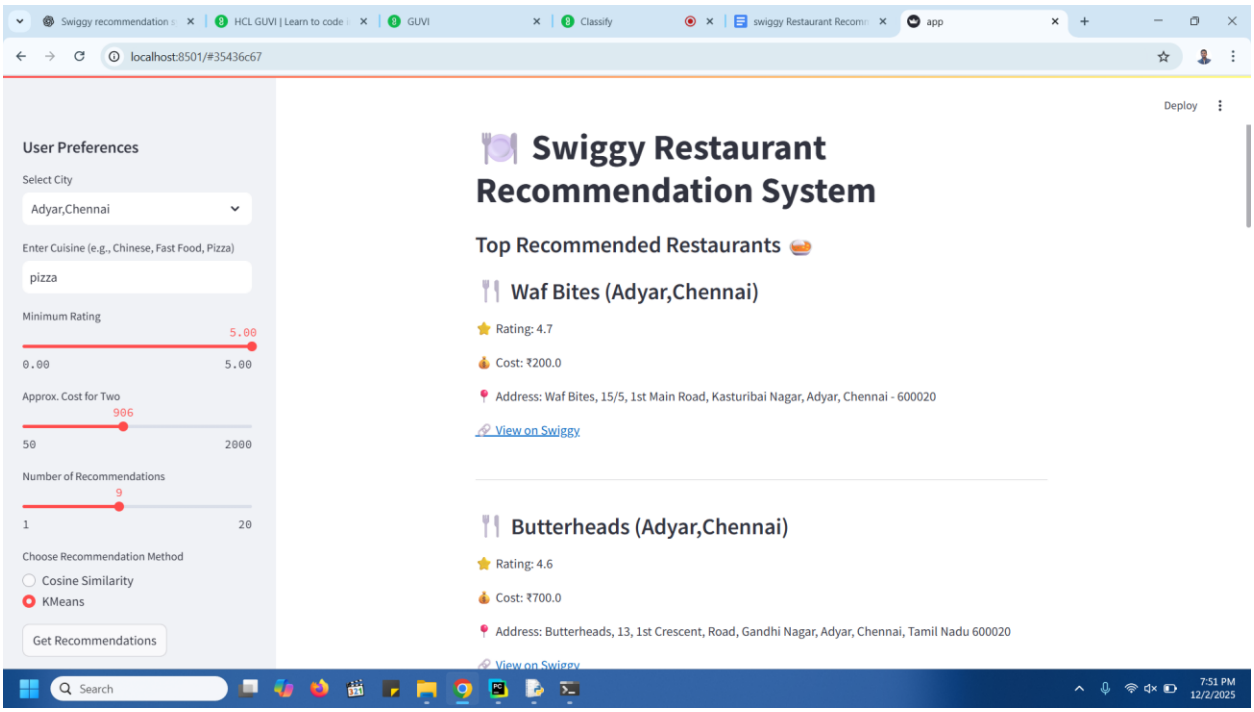
A Streamlit web application was developed to provide an interactive interface for the restaurant recommendation system. The application allows users to input their preferences in a simple and user-friendly manner, without requiring any technical knowledge. Users can select a city from a dropdown list, enter one or more cuisines, and adjust sliders to set their preferred minimum rating and approximate cost for two. Additionally, they can choose between two recommendation algorithms—Cosine Similarity and K-Means Clustering. When the user clicks the “Get Recommendations” button, the Streamlit app communicates with the backend recommendation engine. The user inputs are first transformed using the same encoders and scalers that were applied during preprocessing, ensuring consistency with the encoded dataset. The recommendation engine then generates the most relevant restaurant suggestions based on the selected method.

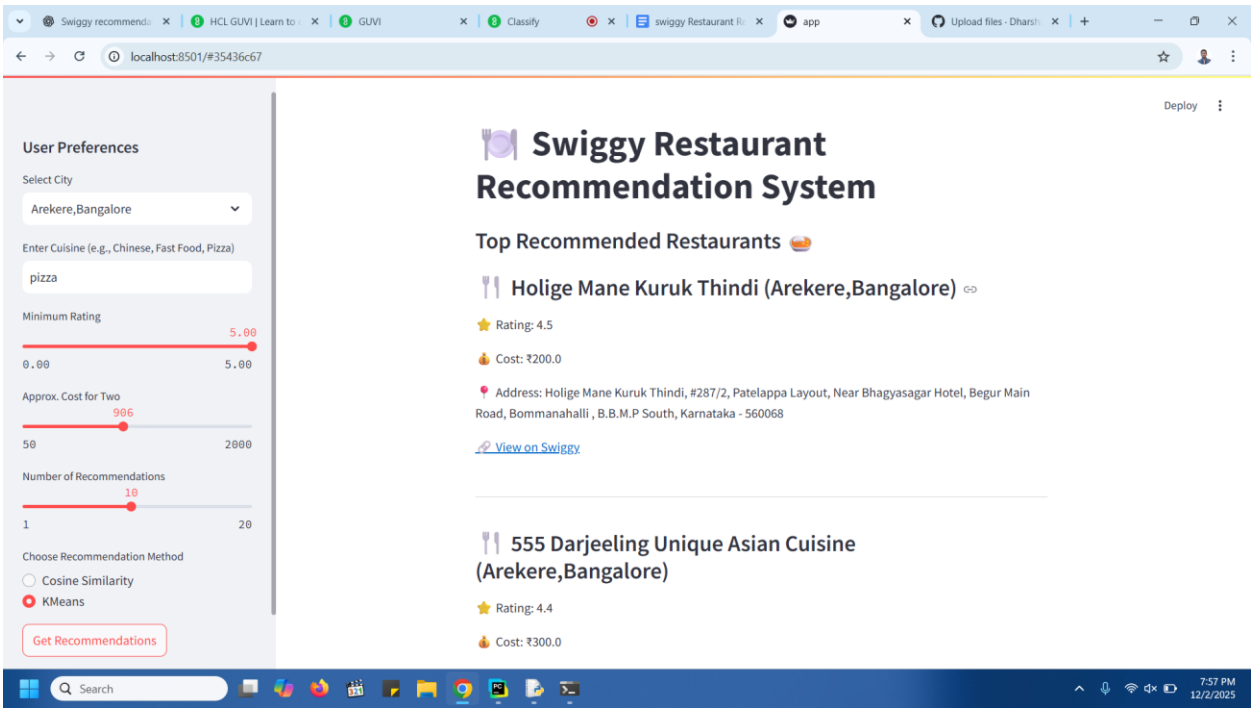
The recommended restaurants are displayed in an organized format within the app, including details such as the restaurant name, city, rating, cost, and address. Each result also includes a clickable Swiggy link to allow users to directly view the restaurant online. Streamlit updates the recommendations in real time based on changes in user preferences, making the interface highly responsive and similar to modern food-delivery applications. Overall, the Streamlit application serves as the final user-facing layer of the project, converting the machine learning model into a functional, easy-to-use product suitable for real-world usage and project evaluation.

Conclusion:

The Swiggy Restaurant Recommendation System successfully demonstrates the end-to-end development of practical machine learning application, starting from raw data processing to real-time user interaction through a Streamlit interface. The project addressed several real-world data challenges, including inconsistent rating formats, multi-label cuisine fields, and categorical city data, all of which were cleaned and transformed into a machine-readable format. By applying both Cosine Similarity and K-Means Clustering, the system is able to provide personalized restaurant recommendations tailored to user preferences such as cuisine type, city, rating, and cost. The Streamlit application enhances usability by allowing users to interact with the model intuitively and receive instant recommendations. Overall, this project highlights the importance of data preprocessing, feature engineering, and interactive deployment in building effective recommendation systems. It demonstrates a complete workflow integrating data analytics, machine learning, and user interface development, showcasing meaningful skills for real-world industry applications.

OUTPUT:




localhost:8501/#35436c67
Deploy
User Preferences
Select City
Arekere,Bangalore
Enter Cuisine (e.g., Chinese, Fast Food, Pizza)
pizza
Minimum Rating
5.00
0.00
5.00
Approx. Cost for Two
906
50
2000
Number of Recommendations
10
1
20
Choose Recommendation Method
Cosine Similarity
KMeans
Get Recommendations
Swiggy Restaurant Recommendation System
Top Recommended Restaurants
Holige Mane Kuruk Thindi (Arekere,Bangalore)
Rating: 4.5
Cost: ₹200.0
Address: Holige Mane Kuruk Thindi, #287/2, Patelappa Layout, Near Bhagyasagar Hotel, Begur Main Road, Bommanahalli , B.B.M.P South, Karnataka - 560068
View on Swiggy
555 Darjeeling Unique Asian Cuisine (Arekere,Bangalore)
Rating: 4.4
Cost: ₹300.0
Search
7:57 PM
12/2/2025