Knowledge-Based Restaurant Recommender System

Objective

Develop a **restaurant recommendation system** that leverages **explicit user preferences** (e.g., cuisine, location, budget) and detailed **restaurant attributes** (from the Zomato dataset) to suggest relevant dining options. This approach eliminates the cold start problem by not relying on past user behavior.

Requirements

Functional Requirements

1. Data Ingestion and Preprocessing

- **Dataset Loading:**
  - Import the Zomato Restaurants Dataset from Kaggle using `pandas`.
- **Cleaning Steps:**
  - Remove duplicates and irrelevant columns (e.g., URL, address if not used).
  - Handle missing values through imputation or removal.
  - Normalize categorical values (e.g., lowercase cuisine types, unify synonyms like "Chinese"/"chinese").
  - Convert price and rating fields to numerical formats.
- **Feature Engineering:**
  - Categorize cost into buckets (low, medium, high).
  - Extract primary cuisine in case of multi-cuisine listings.

    - Map user ratings to a consistent scale (e.g., 1–5).

2. Recommendation Engine

- **Filtering Logic:**
    - Apply filters to match user inputs with dataset:
        - Cuisine: Match any of the selected cuisines.
        - Budget: Use price buckets to match within user-defined range.
        - Location: Exact or approximate match.
- **Ranking Strategy:**
    - Score restaurants based on:
        - User Rating (weighted)
        - Number of votes (credibility of rating)
        - Optional: Proximity or popularity
    - Normalize scores and rank top-N results.
- **Explainability:**
    - Include a short explanation (e.g., “Matched on Indian cuisine and ₹500 budget with 4.3 rating”).

3. User Interface

- **Platform:** Build using **Streamlit** for simplicity and interactivity.
- **Components:**
    - Input sidebar for user preferences
    - Result cards displaying:
        - Restaurant name

    - Cuisine, cost, rating
    - Map link or approximate location
- **Optional Enhancements:**
  - Map view using Folium or Streamlit’s map component
  - Filters to re-rank on the fly

4. Evaluation Metrics

- **Qualitative Evaluation:**
  - Conduct short **user surveys** (Likert scale or yes/no) after recommendation.
- **A/B Testing:**
  - Compare different filtering or ranking strategies with multiple users.
- **Metrics to collect:**
  - User satisfaction score
  - Relevance of recommendation
  - Usability feedback

Deliverables

Module	Description
Data Processing Scripts	Python scripts to load, clean, and preprocess the dataset (Jupyter or standalone Python).
User Preference Interface	Form using Streamlit sidebar or main section to capture preferences.
Recommendation Engine	`filter_and_rank()` function that handles logic and returns top recommendations.

User Interface	Streamlit-based GUI with dynamic updates and explanations for each recommendation.
Evaluation Report	PDF/Markdown file documenting evaluation results, user feedback, and proposed improvements.
Deployment	Host the app on **Streamlit Cloud** or **Render** with a public link.

Resources

- Dataset: Zomato Restaurants Dataset on Kaggle: https://www.kaggle.com/datasets/shrutimehta/zomato-restaurants-data