Recipe Recommender System

1. Create a virtual environment
2. pip install -r requirements.txt
3. python run main.py

Github Repo : https://github.com/SushOS/Recipe-Recommender

Overview

Dataset used : **Food.com Recipes and User Interactions**

This project implements a recipe recommendation system in which user can dynamically enter their preferences in a **terminal-based** UI, and the system **filters** + **ranks** recipes accordingly. If no recipes match, a **fallback** mechanism offers the "closest possible" matches.

- Builds a **content-based recommendation** approach using **TF-IDF.**
- **Incorporates** a **collaborative filtering** component using average ratings for recipes.
- Users can specify **dietary restrictions** (e.g., vegetarian, non_vegetarian, vegan, gluten_free, dairy_free), **cuisine preference**, **available ingredients**, **time**, **difficulty**, etc.
- **Collects** user feedback and applies **NLP** on textual reviews to **infer** additional preferences (e.g., "concerned about calories").
- **Visualizes** the dataset and recommendations via **Seaborn/Matplotlib**.

Project Structure

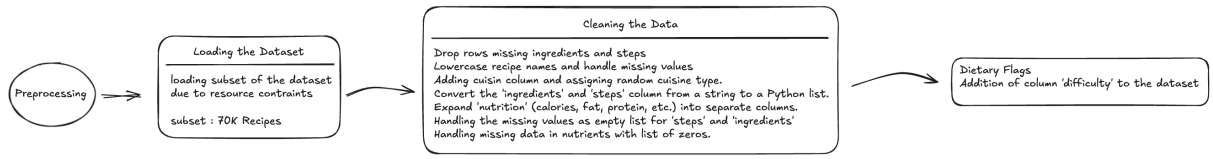
```
recipe_recommender/
├── data/
```

```
│   └── feedback.json            # JSON file storing user fee
dback
│   └── raw/
│       ├── RAW_recipes.csv      # The main recipes dataset
│       ├── RAW_interactions.csv # The user interactions/rati
ngs data
├── src/
│   ├── main.py
│   ├── data_preprocessing.py
│   ├── recommendation_engine.py
│   ├── feedback_manager.py
│   ├── nlp_preference_extraction.py
│   ├── utils.py
│   └── visualization.py
└── requirements.txt
```

What are each of the files doing?

1. main.py : The entry point. Prompts user for preferences, runs the pipeline, displays top recommendations, collects feedback.
2. data_preprocessing.py : Cleans the dataset, randomly assigns cuisines, handles nutrition info, and creates TF-IDF embeddings.
3. recommendation_engine.py : Core recommendation logic (content-based + CF), plus fallback filtering.
4. feedback_manager.py : Loads/saves user feedback in JSON.
5. nlp_preference_extraction.py : Simple keyword-based approach for scanning user comments (NLP).
6. utils.py : Common helper functions (validate preferences, format recipe output).
7. visualization.py : Functions for plotting distributions, correlation, top recommendations, etc.

Preprocessing Steps



Difficulty calculation : (number of ingredients)/10 + (cooking_time)/60 + (n_steps)/5

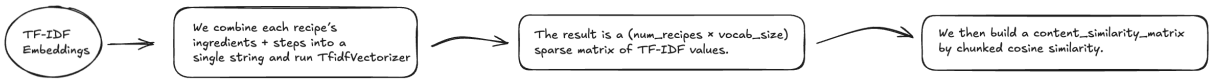
Dietary Flags:

- For 'vegetarian', 'vegan', 'gluten_free', 'dairy_free', we store large lists of “excluded” items (like 'beef', 'chicken', etc.) and check if each recipe mentions them.
- Mark each recipe as is_vegetarian, is_vegan, etc. if it **does not** contain any of those terms.
- Also create is_non_vegetarian = ~is_vegetarian for those wanting strictly non-veg.

TF-IDF Content Embeddings

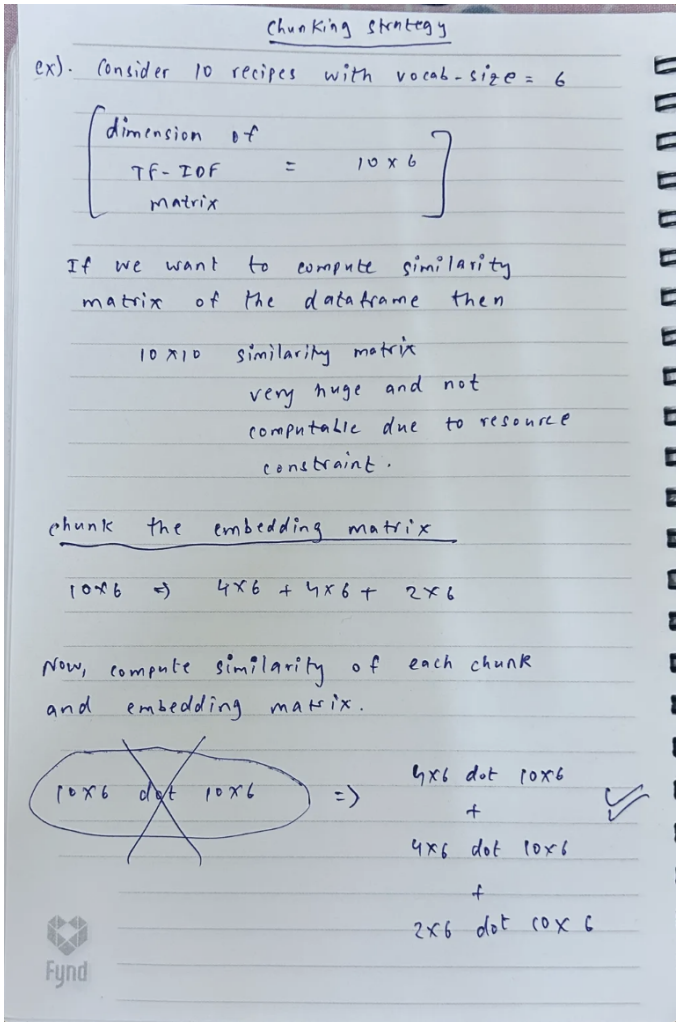
What is TF-IDF?

- **Term Frequency (TF)**: How often a word appears in a recipe’s text relative to the total words in that recipe. A recipe mentioning “chicken” 5 times out of 50 total words has a higher TF for “chicken” than one mentioning it once out of 100 words.
- **Inverse Document Frequency (IDF)**: Gives **less** weight to words that appear **very frequently** across all recipes (like “salt”, “bake”) and **more** weight to rarer, more distinguishing words (like “tamarind” or “mozzarella”)

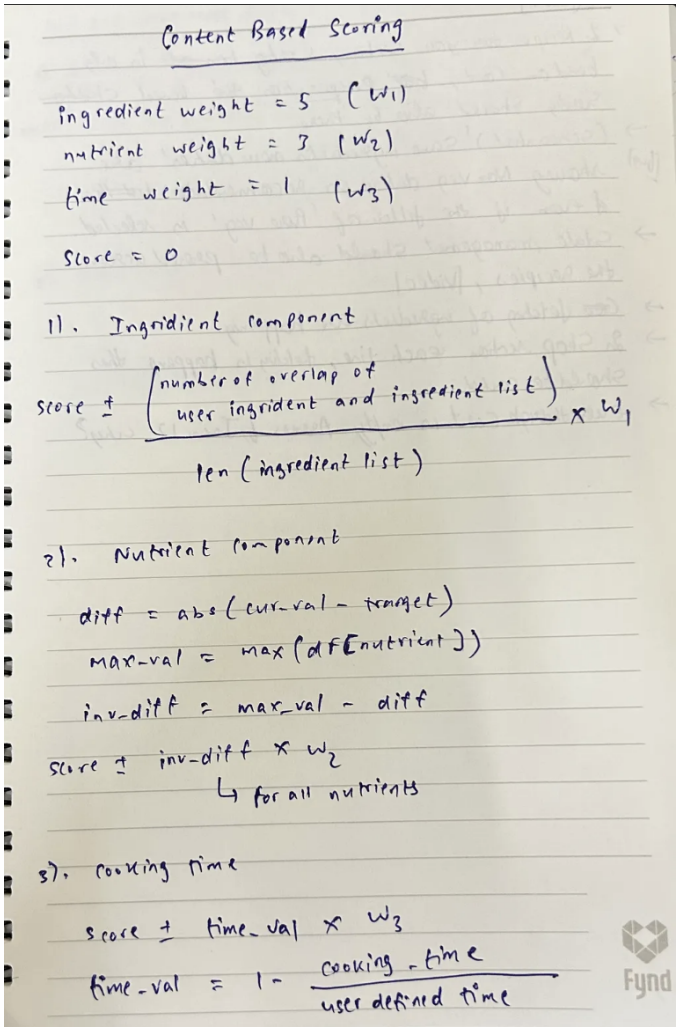


Chunking Strategy

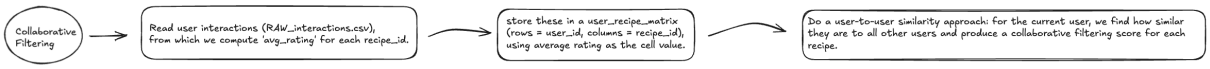
The below image shows how chunking strategy is used to compute the similarity matrix between the recipes.



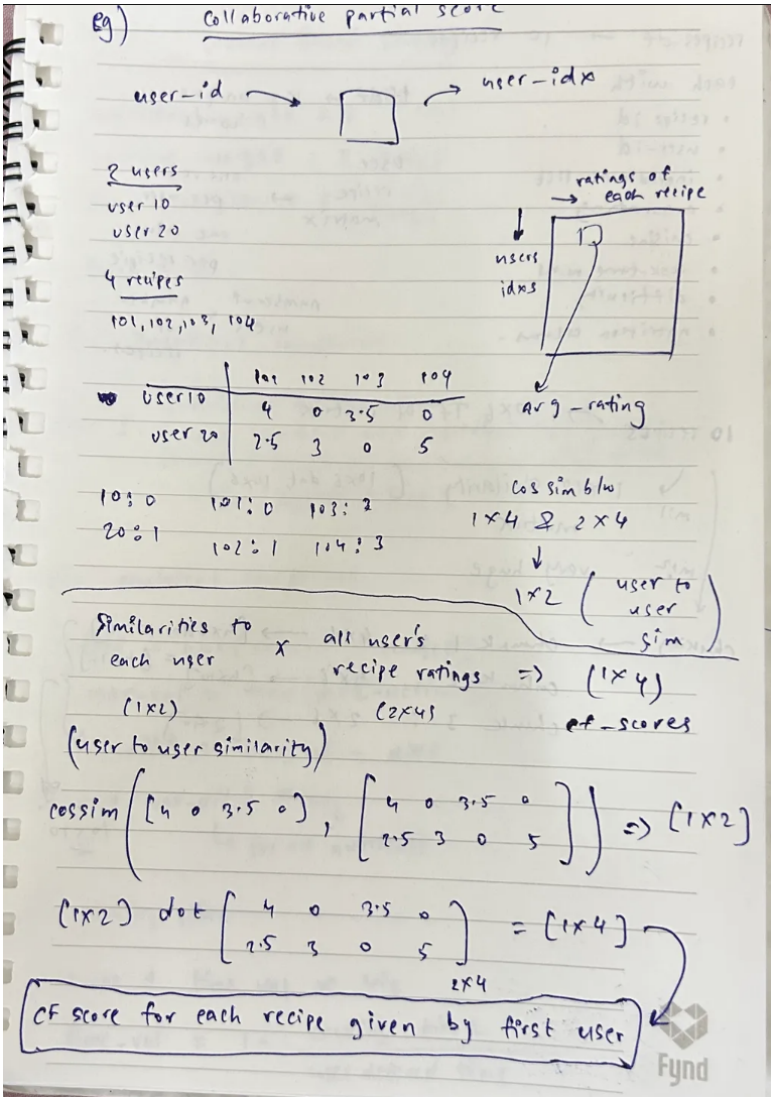
Content Based Score Computation



Collaborative Filtering Score Computation



The below image shows the collaborative scores of each recipe due to a particular user. Similarly the collaborative scores of the recipes due to other users also add up based on the user to user similarity.



Blending Scores + Filters

1. **Content-based** partial score: Weighted sum of ingredient overlap, nutrition closeness, and time-based factor.
2. **Collaborative** partial score: If the user is recognized, we find a user-based similarity score.
3. **Final** = 0.7 × content + 0.3 × CF. (We can change the weights as per our needs)
4. **Filter** out recipes not matching cuisine, dietary restrictions, time, difficulty, must-have ingredients.

5. **If** no match remains, we do a **fallback**: skip cuisine/dietary checks but keep time/difficulty/ingredients.

Collecting & Parsing User Feedback (NLP)

Feedback is stored in a JSON file (feedback.json) with entries like:

```
{
  "123": [
     { "recipe_id": 26067, "rating":2.0, "comments":"Has too
many calories than I need." },
     ...
  ]
}
```

- **NLP**: We have nlp_preference_extraction.py scanning each user's comment for keywords: e.g., "calorie", "healthy."
- If user mentions "calorie," we interpret that they're concerned about **low_calorie**.
- We update user_preferences['nutrition_preferences']['calories'] to something smaller (e.g. 400) automatically.
- This ensures **textual** feedback influences the final recommended recipes.

Visualization

We included several plotting functions in visualization.py:

1. **plot_cuisine_distribution**: Bar chart of how many recipes per cuisine.
2. **plot_top_recommendations**: Bar chart of recommended recipes by their average rating.
3. **show_recommendations_cuisine_breakdown**: Pie chart of recommended recipes by cuisine.

4. **plot_difficulty_distribution**: Histogram of difficulty from 1–5.
5. **plot_rating_distribution**: Distribution of avg_rating.
6. **plot_diet_flags_breakdown**: How many recipes meet is_vegetarian, is_vegan, etc.
7. **plot_correlation_matrix**: A heatmap correlation among numeric columns (e.g., avg_rating, difficulty, calories, etc.) ignoring cook_time_minutes.

Why This Approach?

Below is explanation of **why** we chose this **hybrid (content-based + collaborative filtering)** approach, **why it's a good fit** for this particular dataset, and **how it compares** to other possible methods.

1. **Rich Textual Data (Ingredients & Steps)**

- The dataset includes **detailed textual descriptions** (ingredients, cooking steps). A **content-based** method leveraging **TF-IDF** captures how recipes differ in terms of keywords (e.g., "tomato," "cumin," "frying," "baking"). This is **ideal** for recommending recipes that share similar **ingredient or step** patterns.

2. **User Ratings & Reviews**

- The dataset also has **ratings** (in RAW_interactions.csv), which reflect **user preferences**. A **collaborative filtering (CF)** module uses these ratings to push recipes that **similar users** enjoyed. This adds **personalization** on top of purely content-based matching.

A **hybrid system** of TF-IDF + average-rating CF is **straightforward** to implement and interpret.

TF-IDF embeddings and chunk-based cosine similarity are well-supported by scikit-learn. The CF step is also manageable if we keep a user–recipe matrix for average ratings which we have done. We can handle moderate data sizes without needing a huge GPU cluster.

3. **Fallback & Filtering**

- Because we can derive **scores** from both content and CF, we can also **filter** out recipes by time constraints, dietary needs, or must-have ingredients, and

still produce a final ranking. Many other methods (e.g., deep neural models) may not handle these constraints as transparently or easily.

Why Hybrid (TF-IDF + CF) Is Beneficial

1. **Handles Both Cold Start and Personalization**

- The content-based side helps recommend new or rarely rated recipes because they share textual similarity with known recipes.
- The CF side ensures personalization, surfacing recipes that **similar users** enjoyed.
- You can **explain** recommendations: "This dish is recommended because it shares ingredients with your favourite dish and has a decent average rating from similar users."

2. **Flexible & Expandable**

- Easy to **add** new constraints (time, dietary flags) or new rating data.
- If the dataset or user base grows, you can incorporate more advanced CF or deeper textual embeddings while still leveraging the same framework.

3. **Balance of Complexity & Performance**

- It's not overly complicated to implement or tune, yet it typically performs **much better** than a single method alone (pure content or pure CF).
- Tweaking and weighting between content-based and CF for each domain or user scenario.

Challenges Faced

1. **Data Inconsistency**: Some recipes have missing ingredients or steps. Filled them with empty instances or zero values.
2. **Random Cuisine Assignment**: The dataset didn't have a "cuisine" column, so we randomly assigned it. That's not truly reflective of real data, but we did so to show that the model is capable to filter based on the cuisines.

3. **Dietary Patterns**: The list of forbidden items for “vegetarian/vegan” can be endless. Typos or brand names might slip through. Tried to incorporate all the keywords.
4. **NLP** implemented is very **basic** (keyword-based). Real user comments may have more nuance.

Potential Improvements

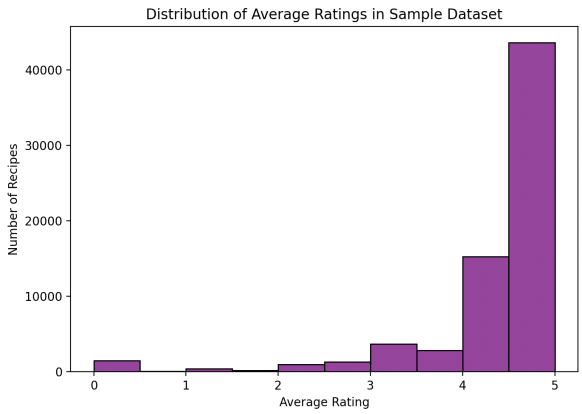
1. **Advanced NLP**: Use spaCy or a large language model to detect user preferences more accurately (sentiment, synonyms, brand names).
2. **Better Cuisine Labelling**: Instead of random assignment, classify recipes by actual ingredients (e.g., “soy sauce,” “rice vinegar” → “Chinese”).
3. **Matrix Factorization**: Replace the simple average rating user→user approach with an advanced SVD or neural CF method for better personalization.
4. **User Interface**: Create a **web app** or **Streamlit** dashboard so users can select preferences and see real-time updates.
5. **Scalability**: For extremely large datasets, consider approximate nearest neighbour methods for TF-IDF similarity, or store data in a specialized index.

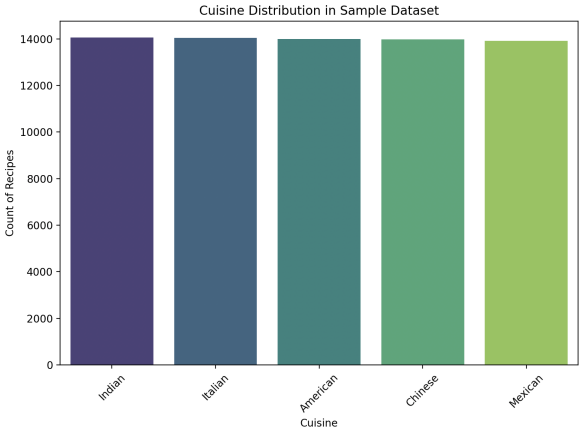
Conclusion

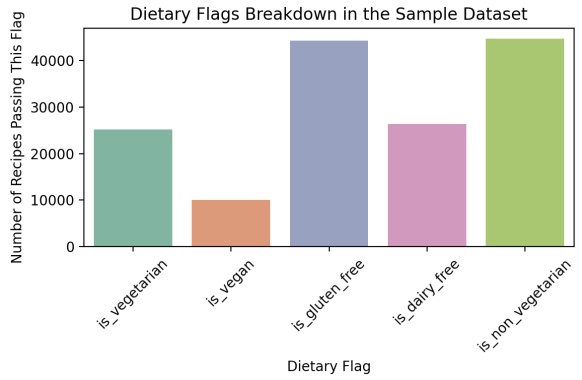
This project **blends** a **content-based** approach (TF-IDF of ingredients/steps) with a **collaborative filtering** approach (user rating data) and respects **user constraints** (time, difficulty, dietary flags). We incorporate **NLP** on textual reviews to infer hidden preferences like “concerned about calories.” The system is demonstrated through a **terminal-based** user flow, plus a suite of **visualizations** that help explore both the dataset and the final recommendations.

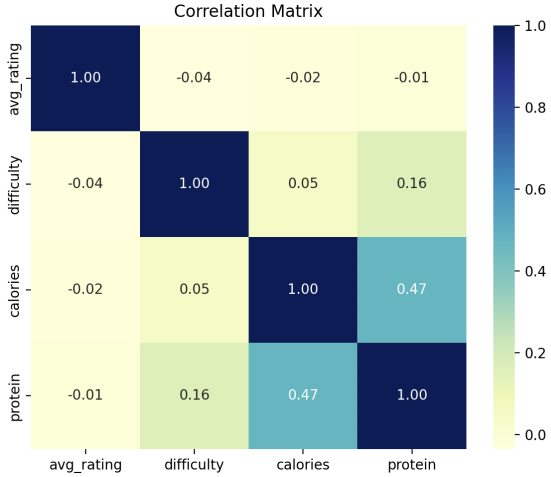
Overall, it serves as a foundation for a **hybrid** recipe recommender, with ample room for **further expansions** and **enhancements** in modeling, UI, and data integration.

Dataset Visualizations





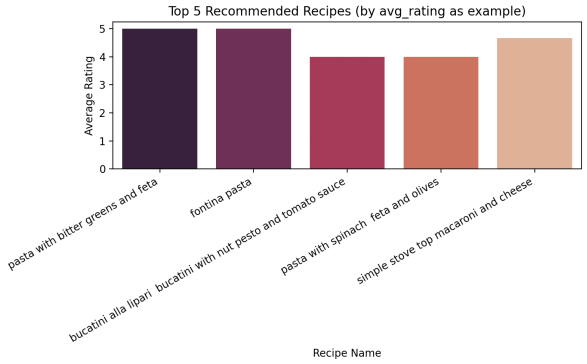


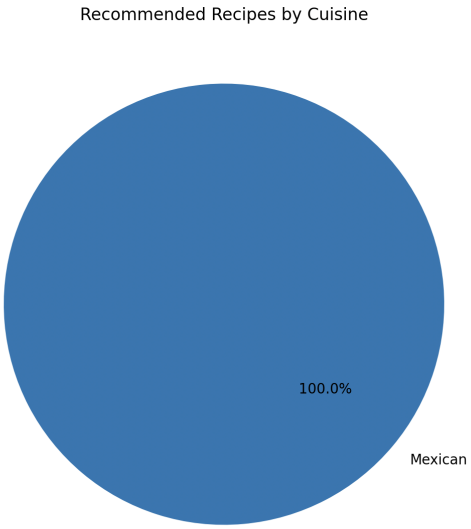


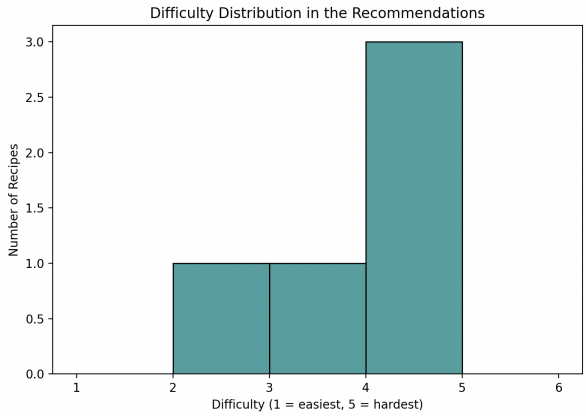
1. User Preference

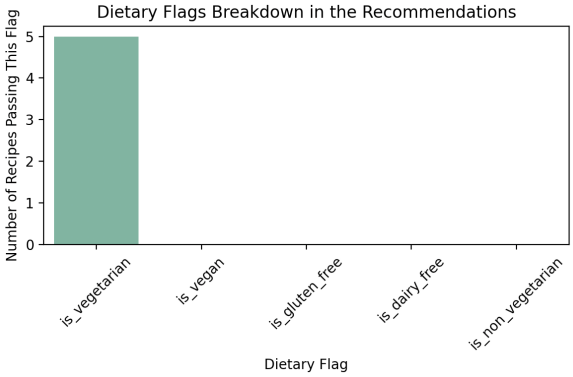
```
3. Please enter your user preferences.
Enter a user ID (integer): 5849
Enter cuisine preference (e.g., 'Indian', 'Italian', 'Mexican', 'Chinese', American'): Mexican
Enter dietary restrictions (comma-separated, e.g. 'vegetarian, non_vegetarian, gluten_free', 'vegan'): vegetarian
Enter available time in minutes (integer): 50
Enter available ingredients (comma-separated, e.g. 'tomatoes, onions, etc..'): pasta
Enter max difficulty (1-5, float allowed): 5
Enter target calories (float): 1500
Enter target protein in grams (float): 10
Enter target fat in grams (float): 10
```

Visualizations



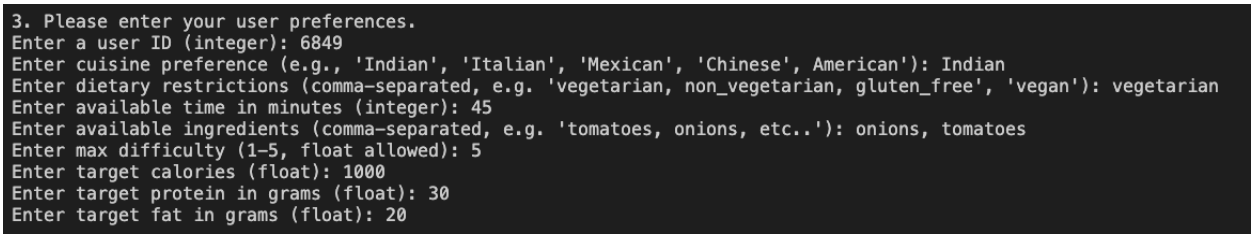




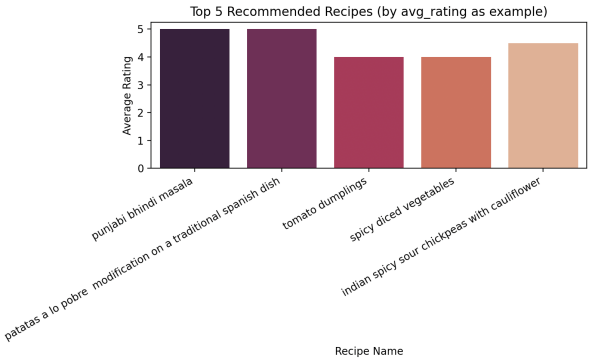


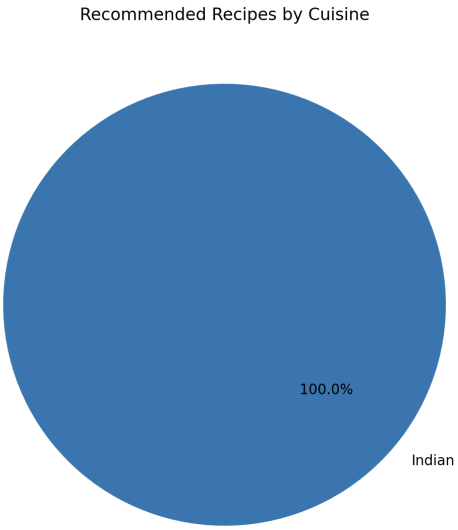
The top 5 recommendations and modified user preference using NLP based on the comments given to the recipe by the user

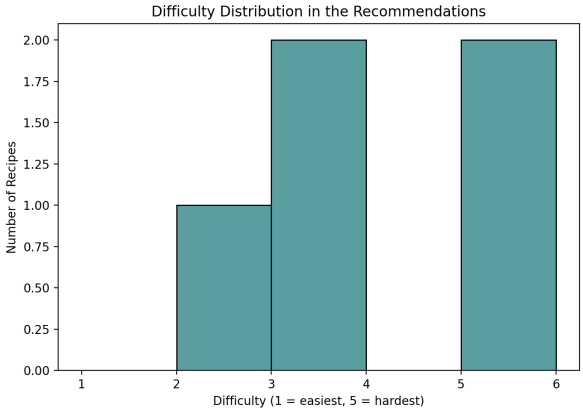
2. Preference

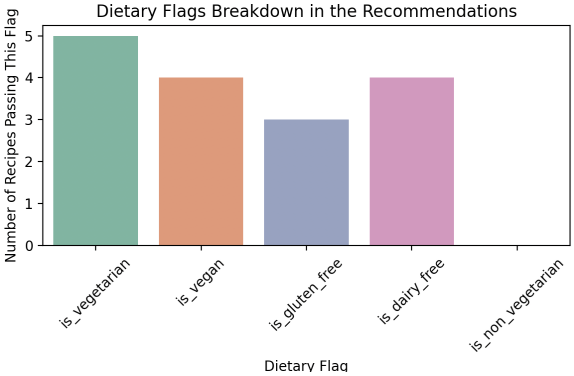


Visualizations







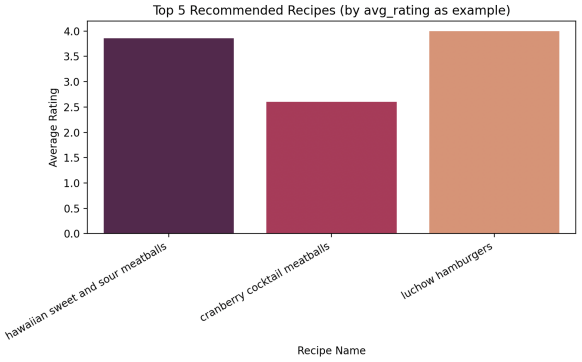


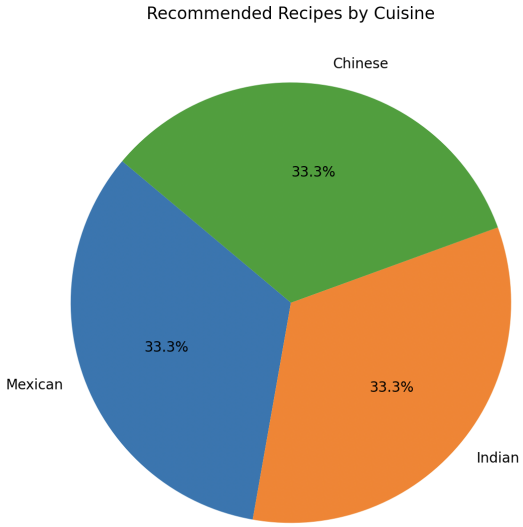
The top 5 recommendations and modified user preference using NLP based on the comments given to the recipe by the user

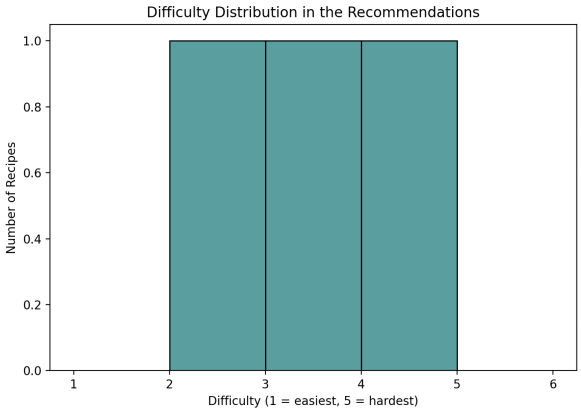
3. Preference

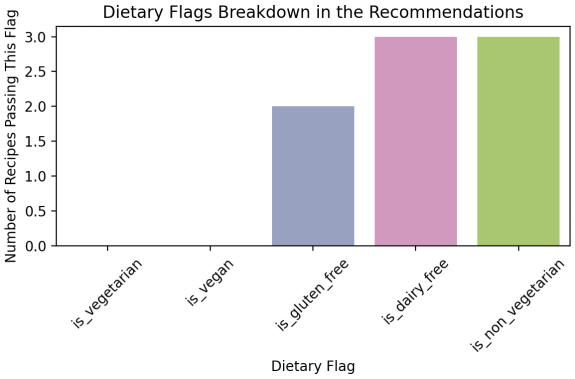
```
3. Please enter your user preferences.
Enter a user ID (integer): 9527
Enter cuisine preference (e.g., 'Indian', 'Italian', 'Mexican', 'Chinese', American'): American
Enter dietary restrictions (comma-separated, e.g. 'vegetarian, non_vegetarian, gluten_free', 'vegan'): non_vegetarian
Enter available time in minutes (integer): 50
Enter available ingredients (comma-separated, e.g. 'tomatoes, onions, etc..'): beef, eggs
Enter max difficulty (1-5, float allowed): 5
Enter target calories (float): 1500
Enter target protein in grams (float): 24
Enter target fat in grams (float): 14
```

visualizations









The user selected American but the model recommended dishes from Indian, Mexican and Chinese cuisine because there were no American dishes as per the user's constraints. Hence the model recommended three closest dishes using the fallback mechanism.

```
5. Top Recommendations:

Recipe 1:
name: hawaiian sweet and sour meatballs
cook_time: 30 minutes
ingredients: ['beef', 'cornstarch', 'nutmeg', 'salt', 'eggs', 'onion', 'pepper', 'garlic clove', 'brown sugar', 'fresh pineapple chunks', 'green pepper', 'garlic powder', 'pineapple juice', 'soy sauce', 'water', 'vinegar']
instructions: blend together beef , eggs , 1 t cornstarch , onion , pepper , nutmeg , salt , and garlic | form into 1-inch balls | heat oil in skillet | brown meatballs on all sides , or bake in oven until browned | in large saucepan add remaining cornstarch , soy sauce , vinegar , water , and brown sugar to pineapple juice | cook until thickened | stir constantly | add meatballs , fruit , and peppers | cook 5 minutes or until fruit is well heated | for oamc: make meatballs as directed and flash freeze | transfer to a ziploc bag , label and put back into freezer | to serve prepare sauce as directed and add defrosted meatballs and heat through
nutrition: {'calories': 1428.0, 'protein': 38.0, 'fat': 109.0, 'carbohydrates': 20.0}
difficulty: 4.5 (1=easiest, 5=hardest)
--------------------------------------------------
Please rate this recipe (1-5), or press Enter to skip:
Any comments? (optional):

Recipe 2:
name: cranberry cocktail meatballs
cook_time: 0 minutes
ingredients: ['beef', 'corn flakes', 'eggs', 'parsley', 'ketchup', 'onions', 'soy sauce', 'garlic powder', 'pepper', 'cranberry sauce', 'chili sauce', 'brown sugar', 'lemon juice']
instructions: in a large bowl , combine ground beef , cornflakes crumbs , parsley , eggs , ketchup , onion , soy sauce , garlic powder and pepper | mix well and form into small balls , from 1 / 2" to 3 / 4" in diameter | place in a casserole or baking pan | heat oven to 300 degrees f | meanwhile in a saucepan , combine cranberry sauce , chili sauce , brown sugar and lemon juice | cook stirring over medium heat until smooth | pour hot sauce over meatballs in casserole | bake for 30 to 45 minutes , depending on the size of the mea tballs | transfer to crock pot and keep on low for serving
nutrition: {'calories': 1264.1, 'protein': 35.8, 'fat': 168.8, 'carbohydrates': 17.0}
difficulty: 3.1 (1=easiest, 5=hardest)
--------------------------------------------------
Please rate this recipe (1-5), or press Enter to skip:
Any comments? (optional):

Recipe 3:
name: luchow hamburgers
cook_time: 30 minutes
ingredients: ['beef', 'fat', 'white bread', 'salt', 'pepper', 'nutmeg', 'eggs']
instructions: mix the meat and the fat | squeeze as much water as you can out of the bread | add to the meat and mix smoothly | add the seasoning and eggs | combine well | shape into large patties | broil or cook in a little hot fat in a pan until brown and done | there should be no pink showing in the burger
nutrition: {'calories': 1455.3, 'protein': 33.0, 'fat': 231.0, 'carbohydrates': 2.0}
difficulty: 2.8 (1=easiest, 5=hardest)
--------------------------------------------------
Please rate this recipe (1-5), or press Enter to skip:
Any comments? (optional):
No feedback/reviews found for user 9527.
```