Food Recommender System with Justification Generator

Project Documentation

GitHub Repository

Ester Molinari, AY 2024/25

Sommario

Introduction

Project Overview

The **Food Recommender System with Justification Generator** is a personalized recommendation system designed to help users select meals based on their individual preferences, dietary restrictions, nutritional goals, and seasonal food availability. In the context of increasing awareness about healthy eating and sustainability, this project aims to provide a convenient tool for users to make informed food choices. The system suggests meal combinations that respect nutritional balance while also considering seasonality and environmental impact, such as greenhouse gas emissions related to food transportation.

The core idea is to recommend ingredients and recipes tailored to the user's profile, which is created by collecting information such as food preferences and intolerances. This information is then used to generate meal suggestions that not only match the user's taste but also meet their nutritional needs, ensuring that the meals adhere to guidelines for balanced macronutrients.

A unique aspect of this project is the **justification generator**, which provides users with "explanations" for why certain meals or ingredients are recommended. These justifications can include aspects such as nutritional benefits, and the seasonal availability of ingredients, helping users make informed decisions about their food choices.

Motivation

The motivation for this project arises from growing concern about personal health, environmental sustainability, and food waste. As more people are looking to improve their diets and reduce their environmental footprint, the need for intelligent systems that can provide personalized recommendations based on these factors becomes increasingly apparent.

- **Health and Nutrition:** A well-balanced diet is essential for maintaining good health, and yet many individuals struggle to understand what foods and meals align with their health goals. This system aims to guide users in making better dietary choices by providing them with meals that meet their nutritional needs while considering individual preferences and dietary restrictions, such as lactose intolerance or gluten sensitivity.
- **Sustainability:** With the global push toward reducing environmental impact, the system also factors in the seasonality of ingredients to promote local, seasonal foods, which tend to have a lower carbon footprint than out-of-season or imported alternatives. By integrating information about the environmental impact of food sourcing and transportation, the system helps users make choices that are both health-conscious and environmentally friendly.

Objectives

The project has several key objectives:

1. **Develop a Personalized Food Recommender System:** The main objective is to create a system that can recommend meals based on user preferences, dietary restrictions, and nutritional goals. This recommendation engine will be powered by algorithms such as cosine similarity, which will analyze user profiles and ingredient data.
2. **Generate Justifications for Meal Recommendations:** Alongside generating meal suggestions, the system will provide explanations for why those meals were recommended. These justifications will include nutritional values, environmental impact, and seasonality, allowing users to make more informed food choices.

3. **Incorporate Seasonality and Environmental Impact:** To align with sustainability goals, the system will consider the seasonality of ingredients and the associated environmental impact of their sourcing. This can help users reduce their carbon footprint by choosing locally grown, seasonal ingredients.
4. **Evaluate the System's Effectiveness:** The system's performance will be evaluated by user feedback and testing, focusing on how well the recommendations meet the users' nutritional needs and preferences, as well as the clarity and usefulness of the justifications.

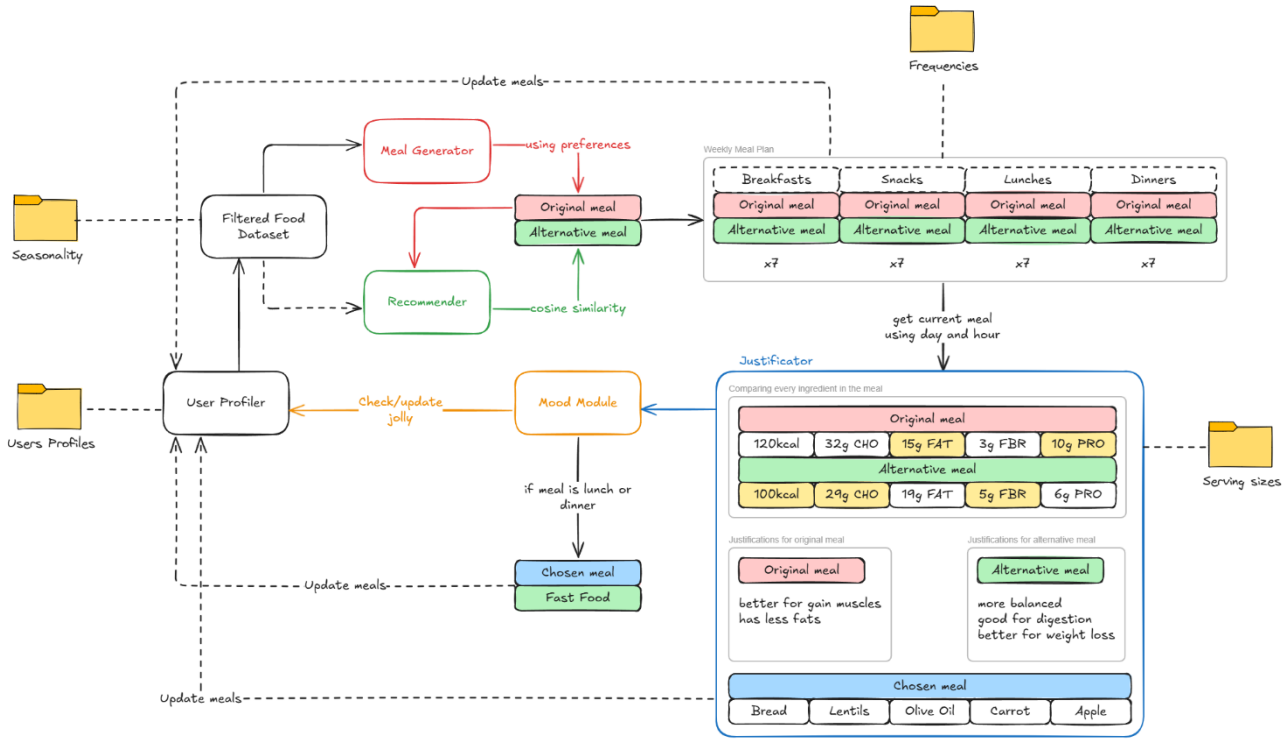
System Design and Architecture

System Architecture

The architecture of the **Food Recommender System** is designed to support the following high-level functionalities:

1. **User Profiling and Preferences Collection** – Users input personal preferences and intolerances (e.g., gluten-free, or lactose-free). This data is stored by the system in JSON files and is used to personalize meal recommendations.
2. **Meal Generation** – The system generates meal suggestions by combining ingredients in a way that meets the user's nutritional needs, preferences, and dietary restrictions.
3. **Food Recommendation Engine** – Based on user preferences and dietary guidelines, this engine generates recommendations for generated meals. The engine uses algorithms like **cosine similarity** to match the user's preferences with available ingredients and recipes considering food energy density.
4. **Justification Generator** – The system explains why each meal or ingredient is recommended. This includes justifications based on nutritional value, seasonality, and tips for buying and storing goods.

Here is a high-level architecture diagram that represents the flow of data and interactions between different system components:



Implementation Details

The system has been implemented using Python 3.12 along with standard libraries such as NumPy, Sci-Kit Learn and Pandas. The project can be reached at this GitHub Repository[1].

System Components

User Profiler

User Profiler is responsible for **creating and loading user profiles**. It has getter and setter methods to read and modify the user profile that is stored in the project's data/processed folder as a **JSON file**. Each user presents: a **diet** style (currently, the system only supports omnivorous style), a list of **intolerances** (currently, the system only supports lactose intolerance) that can also be empty if there are none, a list of the user's food **preferences**, a list of seasonal food preferences (mainly fruits and vegetables), a **meal** dictionary organized into "Breakfast," "Snack," "Lunch," and "Dinner," and a flag indicating whether the user has made use of the **wildcard meal** during the week. This flag value resets every beginning of the week.

An example of user profile creation is shown in Figure 1.

```
Enter username to create or load your profile: lactose
Profile not found. Creating a new one...
Empty profile created at C:\Users\Ester\Documents\SIIA\food-recommender-system\data\processed\lactose.json

Are you lactose intolerant? (Yes/no): yes
To create your profile, please follow the instructions below:

Baked Products:
1. French bread
2. Italian bread
3. Pita bread
4. Rye bread
5. Bread crumbs
6. Bread sticks
7. Cracker
8. Crouton
9. Filo
10. Focaccia
Press Enter to use default preferences or select Baked Products (1-10):
```

Figure 1

Meal Generator

Meal Generator is responsible for generating all the meals for the week using the user's preferences and the foods proposed by the Recommender, which we will talk about later. This module can generate **seven breakfasts, seven lunches, seven dinners and seven snacks** (initially fourteen snacks). The logic that the module uses is as follows:

1. To generate **breakfasts**, it randomly selects a drink, a baked product, a jam, or a cream based on nuts and fruit.
2. To generate **snacks**, use the same logic as breakfasts excluding the drink.
3. To generate the **lunches** and **dinners**, consult the JSON file that contains the recommended weekly portions of each food category to be able to generate them and, subsequently, choose seven meals for lunches and seven meals for dinners, without reconsidering the meals extracted for lunches.

An example of a full week meal generation is shown in Figure 2.

[1] https://github.com/molinari135/food-recommender-system

```
Do you want to print the meals in the terminal or save them to a CSV file? (print/csv): print

🗓️ Monday:

🍽️ Breakfast:
👉 Main option:
- 200g of Tea (a medium glass)
- 50g of White Bread (e.g. 2-3 biscuits, a slice of bread)
- 30g of Almond paste (e.g. 2 full teaspoons)
- 150g of Orange (e.g. a medium fruit or two small fruits)

🔁 Alternative option:
- 200g of Herbal tea (a medium glass)
- 50g of Multigrain bread (e.g. 2-3 biscuits, a slice of bread)
- 30g of Peanut butter (e.g. 2 full teaspoons)
- 150g of Clementine (e.g. a medium fruit or two small fruits)
------------------------------

🍽️ Snack:
👉 Main option:
- 50g of Biscuit (e.g. 2-3 biscuits, a slice of bread)
- 20g of Fruit preserves (e.g. 2 full teaspoons)
- 150g of Pear (e.g. a medium fruit or two small fruits)

🔁 Alternative option:
- 50g of White Bread (e.g. 2-3 biscuits, a slice of bread)
- 20g of Apricot jam (e.g. 2 full teaspoons)
- 150g of Orange (e.g. a medium fruit or two small fruits)

✅ Chosen option:
- 50g of White Bread (e.g. 2-3 biscuits, a slice of bread)
- 20g of Apricot jam (e.g. 2 full teaspoons)
- 150g of Pear (e.g. a medium fruit or two small fruits)
------------------------------

🍽️ Lunch:
👉 Main option:
- 80g of Pasta (weight them with a cooking scale)
- 100g of Cheese (weight them with a cooking scale)
- 10g of Olive oil (a tablespoon)
- 10g of Marinara sauce (optional, a tablespoon)
- 200g of Pumpkin (raw or cooked, weight them)
- 150g of Orange (e.g. a medium fruit or two small fruits)

🔁 Alternative option:
- 80g of Couscous (weight them with a cooking scale)
- 100g of Cottage cheese (weight them with a cooking scale)
- 10g of Margarine (a tablespoon)
- 10g of Tomato sauce (optional, a tablespoon)
- 200g of Chicory (raw or cooked, weight them)
- 150g of Clementine (e.g. a medium fruit or two small fruits)
------------------------------
```

Figure 2

Recommender

The Recommender intervenes after the generation of meals obtained by combining the foods for which the user expressed a preference during the profile creation phase. Using the **cosine similarity** between the nutritional values of the preferred foods and those of the same category belonging to the dataset, the recommender suggests a food that is nutritionally equivalent but with a **lower energy density**, which is the amount of energy in a particular weight of food (calories per gram).

In this way, the system offers the user two meals: one obtained by combining only the foods he likes and one by combining only similar foods but with a lower energy density.

The Recommender also has a seasonal function that can help the user choose fresh seasonal fruits and vegetables for every meal. An example is shown in Figure 3.

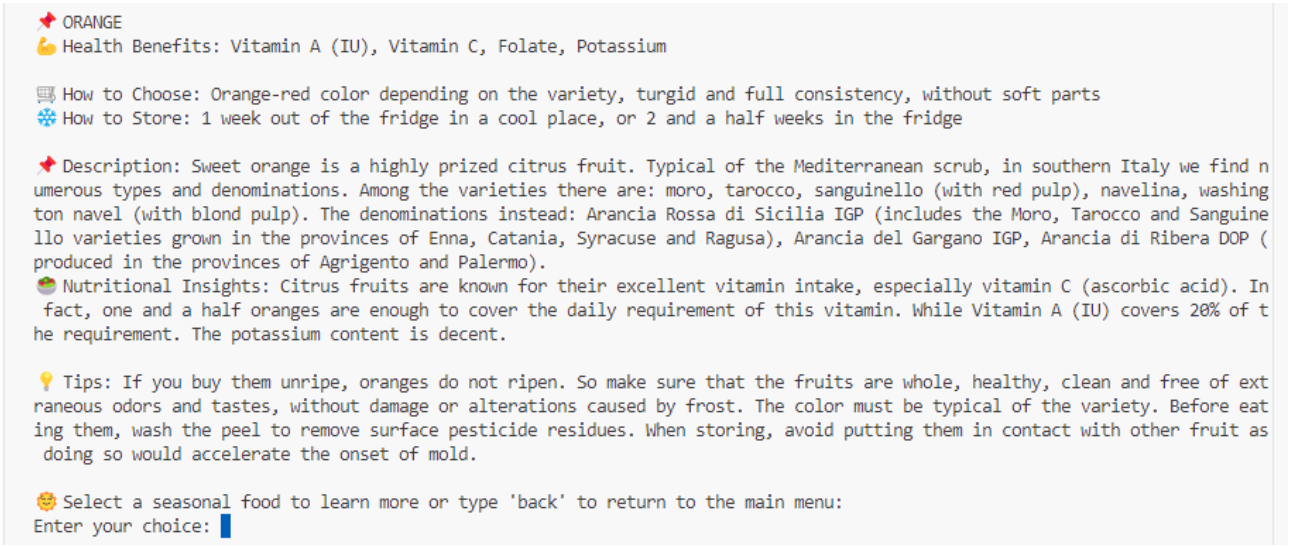


Figure 3

Justificator

When the user asks the system for the meal of the day, using the **calendar** and the **system clock** of his computer, the Justificator goes into action by showing the user the nutritional differences between the preferred foods and those recommended by the Recommender, **trying to persuade the user** to choose those suggested by the Recommender.

The logic implemented in the Justifier is based on the EFSA guidelines on healthy eating: a food that has a **lower** value in calories, fat and carbohydrates and a **higher** value of proteins and fibers is "rewarded" by assigning it a positive score, otherwise the **score** is assigned to the other food with which it is compared.

When all the comparisons are completed, the Justificator will display the "winning" food and along with it the strengths of both compared foods. The user is still invited to make a flexible choice as the system does not intend to mark certain foods as good or bad.

An example of the Justificator at work is shown in Figure 4.

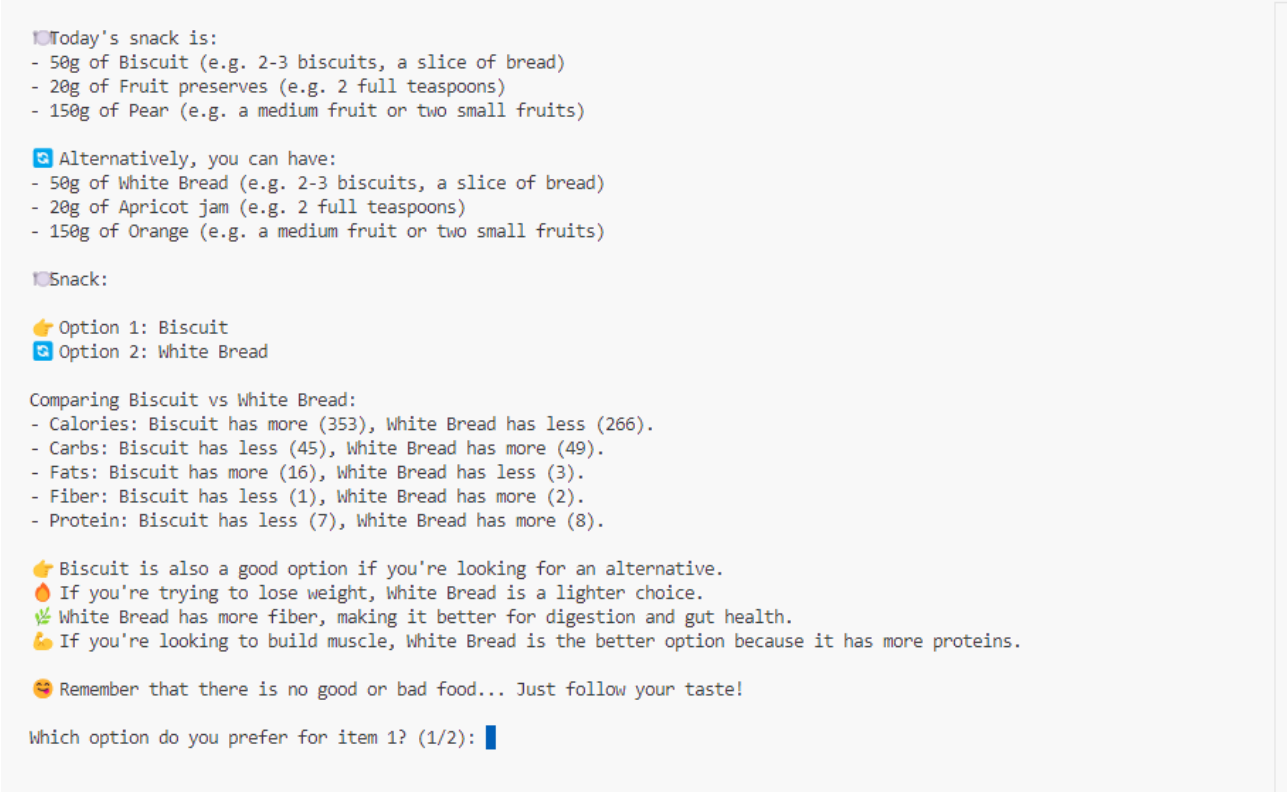


Figure 4

Mood Module

Another module of the system is the one dedicated to the user's mood. A **stressed user** will tend to eat in a disorderly manner to balance his feelings, so the system will take this into account by modifying only one of the two main meals, namely lunch and dinner, proposing **fast food**. This operation can only happen once a week and is reset only on Monday of the following week.

The mood module will be triggered, for example, during lunch or dinner, as is shown in Figure 5.

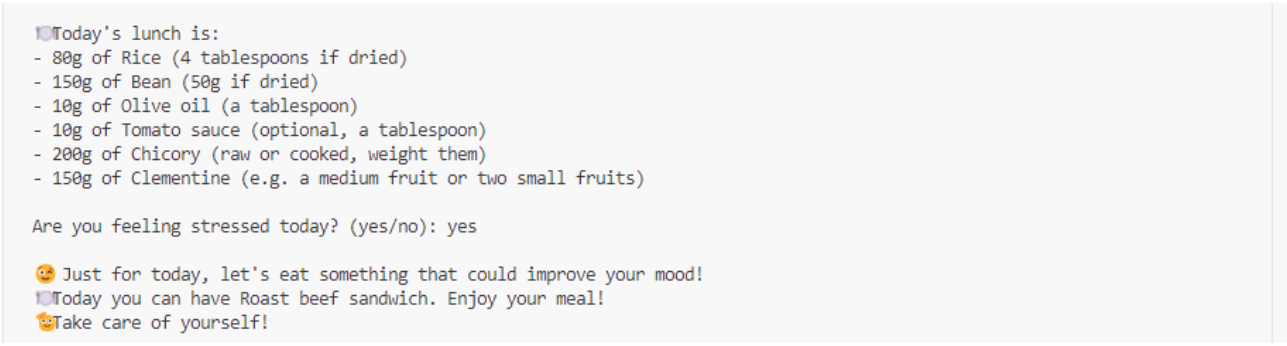


Figure 5

Data Storage and Management

The data used by the system to perform the generations and justifications are present in the **"data/raw" folder** and are never modified by the system. The user profiles, however, are stored in the **"data/processed" folder** which is automatically loaded by the system at its startup.

The original dataset with all nutritional values is in **CSV format** and has undergone heavy modifications during the design phase as it did not present a precise distinction between foods. The modified version can be found in the repository on GitHub and is necessary for the model to function.

The support files for the Justificator and the Meal Generator are in **JSON format** and were created following the European (EFSA) and Italian (LARN) guidelines for food that can be found in the GitHub Repository.

The system can print the weekly meal plan in a **CSV file** providing generated meals based on preferences and alternative meals based on the recommender's proposals (Figure 6 and Figure 7).

```
Do you want to print the meals in the terminal or save them to a CSV file? (print/csv): csv

Weekly meals have been saved to 'weekly_meals.csv'.
```

Figure 6

Column1	Column2	Column3	Column4	
Meal Type	Option Type	Friday	Monday	Saturday
Breakfast	Alternative	Herbal tea, White Bread, Apricot jam, Orange	Herbal tea, Multigrain bread, Apricot jam, Orange	Herbal tea, Multigra
Breakfast	Main	Orange juice, Biscuit, Fruit preserves, Apple	Orange juice, White Bread, Marmalade, Pear	Espresso, White Bre
Dinner	Alternative	Buckwheat, Egg white, Margarine, Tomato sauce, Chico...	Couscous, Turkey ham, Margarine, Salsa, Chicory, Orange	Buckwheat, Surimi,
Dinner	Main	Rice, Egg, Olive oil, Marinara sauce, Broccoli, Apple	Wheat Bread, Mortadella, Olive oil, Tomato sauce, Caul...	Rice, Salmon, Olive
Lunch	Alternative	Couscous, Rabbit Meat, Margarine, Tomato sauce, Chic...	Buckwheat, Bean, Margarine, Salsa, Chicory, Clementine	Couscous, Roast be
Lunch	Chosen		Roast beef sandwich	
Lunch	Main	Wheat Bread, Chicken meat, Olive oil, Marinara sauce,...	Rice, Lentil, Olive oil, Tomato sauce, Turnip, Orange	Pasta, Pork, Olive o
Snack	Alternative	White Bread, Apricot jam, Orange | White Bread, Apric...	Multigrain bread, Apricot jam, Clementine | Multigrain...	White Bread, Apric
Snack	Chosen		Multigrain bread, Marmalade, Clementine | Multigrain...	
Snack	Main	Biscuit, Marmalade, Kiwifruit | Biscuit, Marmalade, Kiw...	White Bread, Marmalade, Orange | White Bread, Mar...	Biscuit, Marmalade

Figure 7

Evaluation and Results

Evaluation Criteria

A **questionnaire** will be made to evaluate the system.

Testing

The system will be tested by a **limited number of users**.

Challenges and Limitations

Challenges

The system has undergone **several scaling** during development, one of many was to exclude the various diets (omnivorous, vegetarian, vegan) and allergies. The diets were excluded for dataset construction reasons, in fact the data within it did not allow a balanced generation of foods for the various diets. Allergies are **not considered** by the system because the dataset was not designed for that purpose and would have required an additional mapping operation.

Limitations

Currently, the system is not able to handle **gluten intolerance** despite it being implemented. For this reason, the function has been deactivated.

Conclusion

The implementation of such a system requires a deep knowledge of the domain and a lot of time to implement all the information that is available. Both EFSA and LARN publish huge scientific dossiers that are not understandable by those who do not know the domain and are often difficult to implement with **crisp logic**, requiring greater effort in the implementation of the modules of these systems.

The system proves capable of **creating nutritional plans** and **making pertinent suggestions** but requires a more in-depth study of the information available.