Personalized Nutrition Assistant: Leveraging Food Data Analytics for Customized Dietary Recommendations

📄 Introduction

This project leverages nutritional data to provide precise dietary recommendations:

- Motivation: Rising global health issues such as obesity and cardiovascular diseases highlight the importance of nutrition management. Many lack a comprehensive understanding of nutritional values, leading to imbalanced diets.
- Goal: Analyze dietary habits combined with nutritional data to generate personalized recommendations and recipes.

📊 Dataset

1. Food Nutritional Facts: Nutritional data for 1,174 food items, including calories, protein, fats, carbs, fiber, sugar, vitamins, and minerals. Used for dietary analysis and nutrient gap identification.
2. EpiRecipes Dataset: Recipes with ingredients, preparation steps, and nutritional details. Used to recommend recipes tailored to users' nutritional needs.

🛠 System Architecture

The system consists of:

1. Input Layer: Users provide personal data (age, weight, height, gender) and dietary records.
2. Analysis Layer: Calculates BMR and TDEE, evaluates nutritional intake, and identifies deficiencies/excesses.
3. Recommendation Layer: Uses XGBoost to suggest the top 5 recipes, including detailed cooking instructions.

📋 How to Reproduce

Prerequisites

1. Python 3.9 or higher.
2. Install the required Python libraries listed in requirements.txt.

Steps to Run

1. Clone the Repository:

```bash
git clone https://github.com/vicky0619/Introduction-to-Data-Science.git
cd Introduction-to-Data-Science/final\ project
```

2. Set Up a Virtual Environment:

```bash
python3 -m venv env
source env/bin/activate  # For Windows: env\Scripts\activate
```

3. Install Dependencies:

```bash
pip install -r requirements.txt
```

4. Run the Backend (Flask):

```bash
python app.py
```

5. Run the Frontend (Streamlit):

```bash
streamlit run frontend.py
```

6. Access the Application:

- Open the Streamlit interface in your browser (e.g., `http://localhost:8501`).

🗁 File Structure

1. app.py: Backend Flask API for processing user data, calculating BMR/TDEE, and generating recommendations.
2. frontend.py: Streamlit-based frontend for user input and displaying results.
3. analysis.py: Core analysis logic, including calculations for nutritional gaps, BMR, TDEE, and recipe recommendations using XGBoost.
4. requirements.txt: List of required Python libraries for the project.
5. cleaned_food_data.csv: Preprocessed dataset for nutritional analysis.

📈 Model Description

1. Model Selection: XGBoost was chosen for its regularization and sparse data optimization capabilities.
2. Training: Data preprocessed using LabelEncoder and MinMaxScaler. 80-20 train-test split to evaluate performance.
3. Prediction: Predicts health scores for recipes based on key nutritional features.
4. Recommendation: Top 5 recipes suggested based on similarity scores.