NUTRIAI – SECOND MILESTONE REPORT

AIE1014 – AI Applied Project

TEAM MEMBERS:

AASHISH GIRI

JAYKESH J AWAL

MEET PATEL

SABIN KHATRI

July 9, 2025

Executive Summary

NutriAI is a smart AI-powered system that helps people follow a healthy diet and lifestyle. It gives personalized food and fitness suggestions using machine learning and basic rule-based logic.

In the first milestone, we planned the project, collected food data, and created user personas. For this second milestone, we have completed more than 70% of the system. This includes training a Random Forest model to check if users meet their daily health goals and creating a rule engine to suggest helpful actions.

We worked entirely in Google Colab and used visual graphs, pie charts, and printed tips to make results easy to understand. Our team collaborated using WhatsApp, Zoom, and Google Docs, and we divided tasks based on each member's strengths.

Technical Architecture

While our technical system may appear rudimentary, it is truly efficient in its design. It begins with specific input data. In our case, Our system is built in Google Colab using Python and includes the following components:

- Input: Food and activity logs (CSV dataset)
- ML Model: Random Forest classifier to predict if health goal is met
- Rule Engine: Provides easy, helpful suggestions
- Output: Clear charts, tables, and tips

Steps:

1. Load food logs and preprocess the data
2. Apply ML model to predict goal status
3. Apply rule logic to give suggestions
4. Visualize results using bar charts, pie charts, and comments

Everything is displayed clearly in the notebook, making the system easy to use.

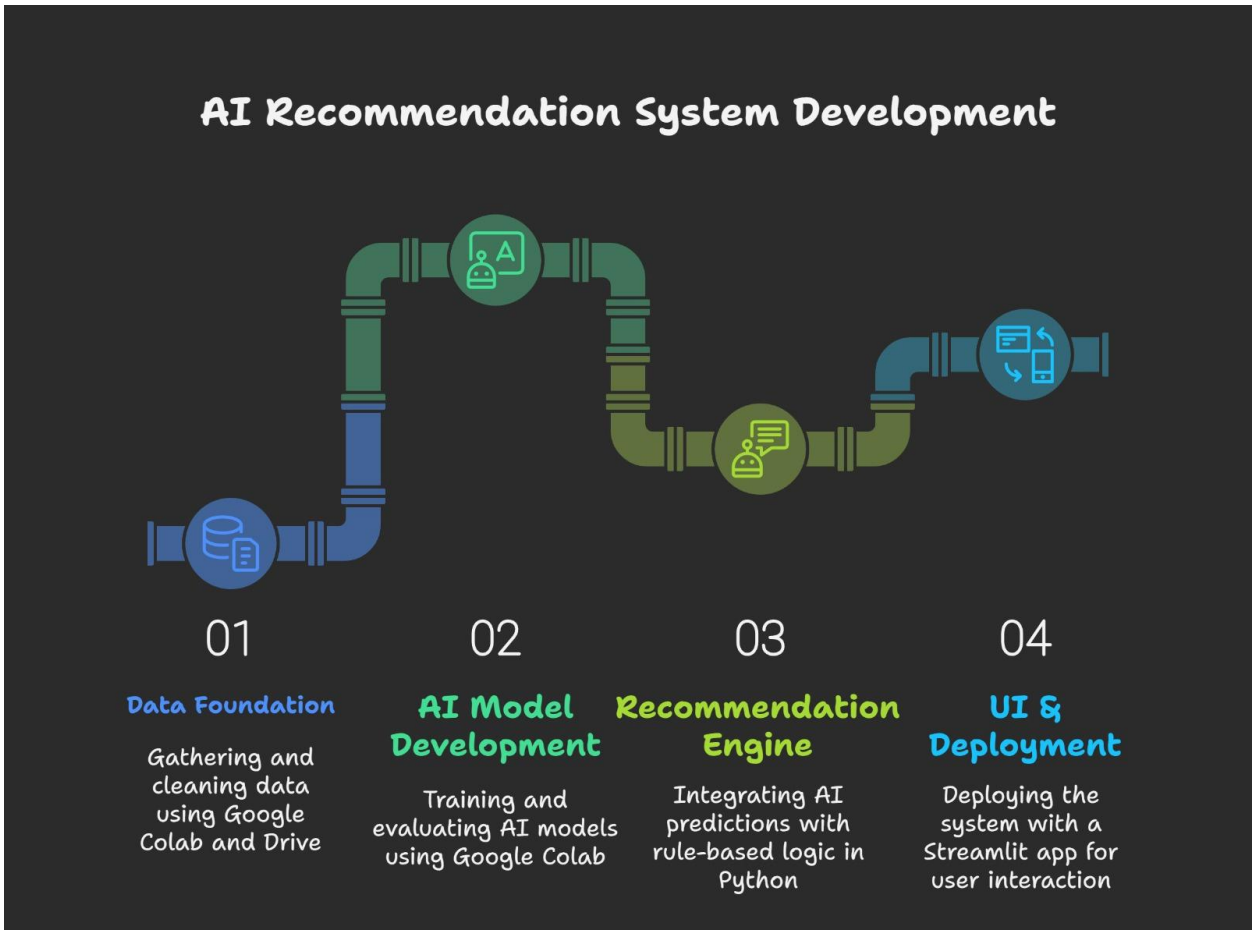

Figure 1: NutriAI System Architecture and Data Flow

Workflow Description:

- Data Foundation: The process begins in Google Colab, where the public Kaggle food dataset is curated. Custom Python scripts then generate a comprehensive dataset of synthetic user profiles and daily logs, which is stored on Google Drive.
- AI Model Development: This generated data is used to train and evaluate a Random Forest classifier. The trained model is serialized (pickled) and saved as an artifact.
- Recommendation Engine: A Python-based rule engine uses logic based on nutritional principles, user goals, and, most importantly, the live predictions from the AI model.
- UI & Deployment: A locally run Streamlit application provides the user interface. It loads the saved model and rule engine logic, takes user input (simulated via persona selection), and displays the personalized recommendations

Data Strategy

We used a CSV file named daily_food_nutrition_dataset.csv as our main dataset. This file contains information like calories, protein, fat, carbohydrates, fiber, and water content for different food items. We created 8 user personas like college student, athlete, office worker and made 7 days food and activity logs for each. We added a column called Goal Met, which tells if the user met their health target.

Before using the data, we cleaned it:

- Handled missing values
- Standardized all units (e.g., grams, ml)
- Removed wrong data entries
- Validated the nutritional totals

This way, our data is clean and realistic for training the model.

```python
# --- 3.3: Data Generation Logic ---
def generate_synthetic_data(num_users_per_persona, num_days, food_df, personas_dict):
    daily_records = []
    user_id_counter = 0

    for persona_name, persona_details in personas_dict.items():
        for i in range(num_users_per_persona):
            user_id = user_id_counter
            for day in range(num_days):
                # --- Generate Daily Meals ---
                daily_meals = []
                num_meals = random.randint(3, 5)

                if persona_details['dietary_preference'] == 'high_protein':
                    # Bias sampling towards high-protein foods
                    weights = food_df['is_high_protein'] * 5 + 1 # 5x more likely to pick high protein
                else:
                    weights = None # Uniform sampling

                daily_meals_df = food_df.sample(n=num_meals, weights=weights, replace=True)

                # --- Generate Daily Activity ---
                if persona_details['activity_level'] == 'high':
                    activity = random.choice(['Running', 'Weightlifting'])
                    duration = random.randint(45, 90)
                elif persona_details['activity_level'] == 'moderate':
                    activity = random.choice(['Cycling', 'Walking'])
                    duration = random.randint(30, 60)
                else: # low
                    activity = random.choice(['Desk Work', 'Resting'])
                    duration = random.randint(120, 240)

                calories_expended = estimate_calories_burned(activity, duration)

                # --- Aggregate Daily Stats ---
                total_calories = daily_meals_df['Calories (kcal)'].sum()
                total_protein = daily_meals_df['Protein (g)'].sum()
                total_carbs = daily_meals_df['Carbohydrates (g)'].sum()
                total_fat = daily_meals_df['Fat (g)'].sum()

                # --- Create Target Label ---
                protein_target_met = 1 if total_protein >= persona_details['protein_target'] else 0

                daily_records.append({
                    'user_id': user_id,
                    'day': day,
                    'persona_type': persona_name,
                    'activity_level': persona_details['activity_level'],
                    'calories_expended': calories_expended,
                    'total_calories_kcal': total_calories,
                    'total_protein_g': total_protein,
                    'total_carbs_g': total_carbs,
                    'total_fat_g': total_fat,
                    'protein_target_g': persona_details['protein_target'],
                    'calorie_target_kcal': persona_details['calorie_target'],
                    'protein_target_met': protein_target_met # Our target variable
                })
            user_id_counter += 1

    return pd.DataFrame(daily_records)
```

Graphs, Visuals, and Benchmarks

Exploratory Data Analysis (EDA):

EDA was performed to validate the synthetic data generation. The boxplot below confirms that our personas have distinct protein intake patterns, as intended. The "Muscle_Gain_Focus" persona shows a significantly higher median protein consumption.

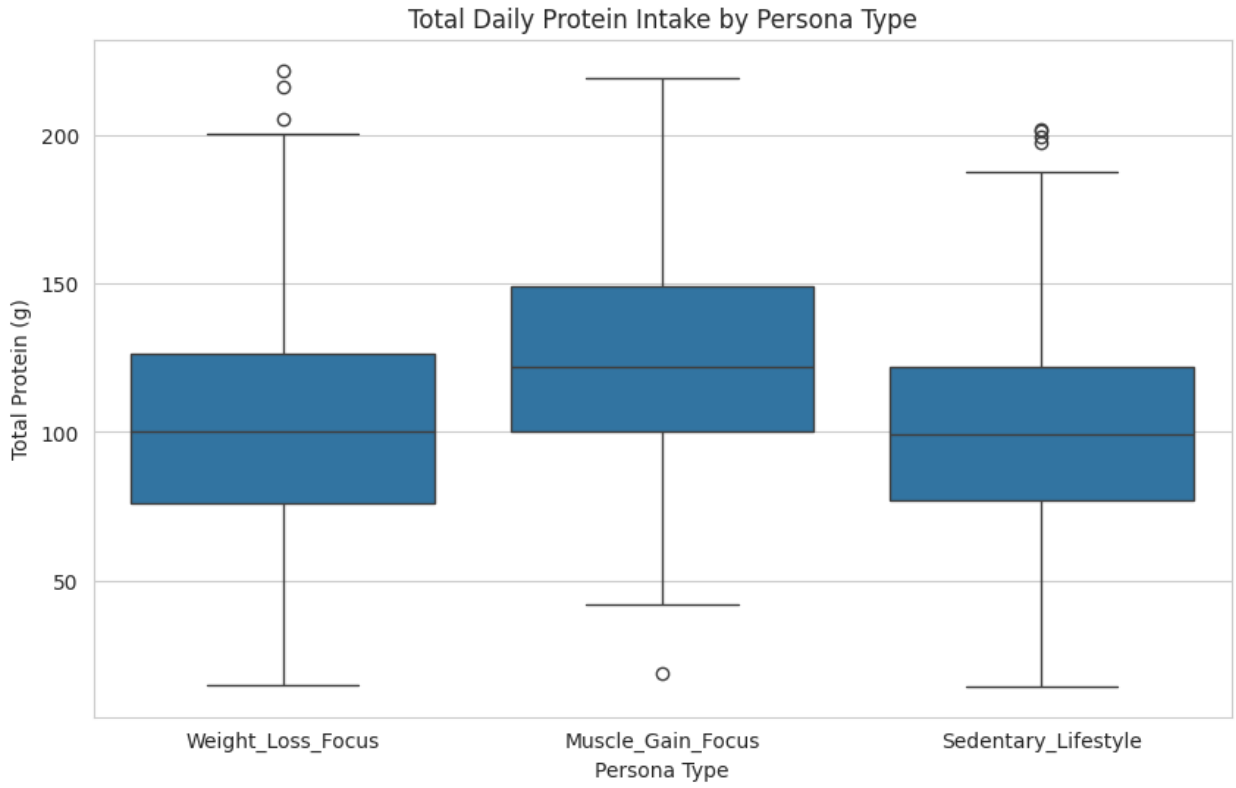

Figure 2: Total Daily Protein Intake by Persona Type

Model Performance Benchmarks:

We trained two models to predict protein_target_met. The Random Forest classifier was chosen as our primary model, with Logistic Regression as a baseline. After correcting an initial data leakage issue, the models produced realistic and insightful performance metrics.

Model	**Accuracy**	F1-Score (Class 1)	**Precision (Class 1) Precision (Class 1)**	Recall (Class 1)
Logistic Regression	82.6%	0.76	0.76	0.76
Random Forest	81.0%	0.72	0.76	0.69

Table 1: Model Performance Comparison Table 1: Model Performance Comparison

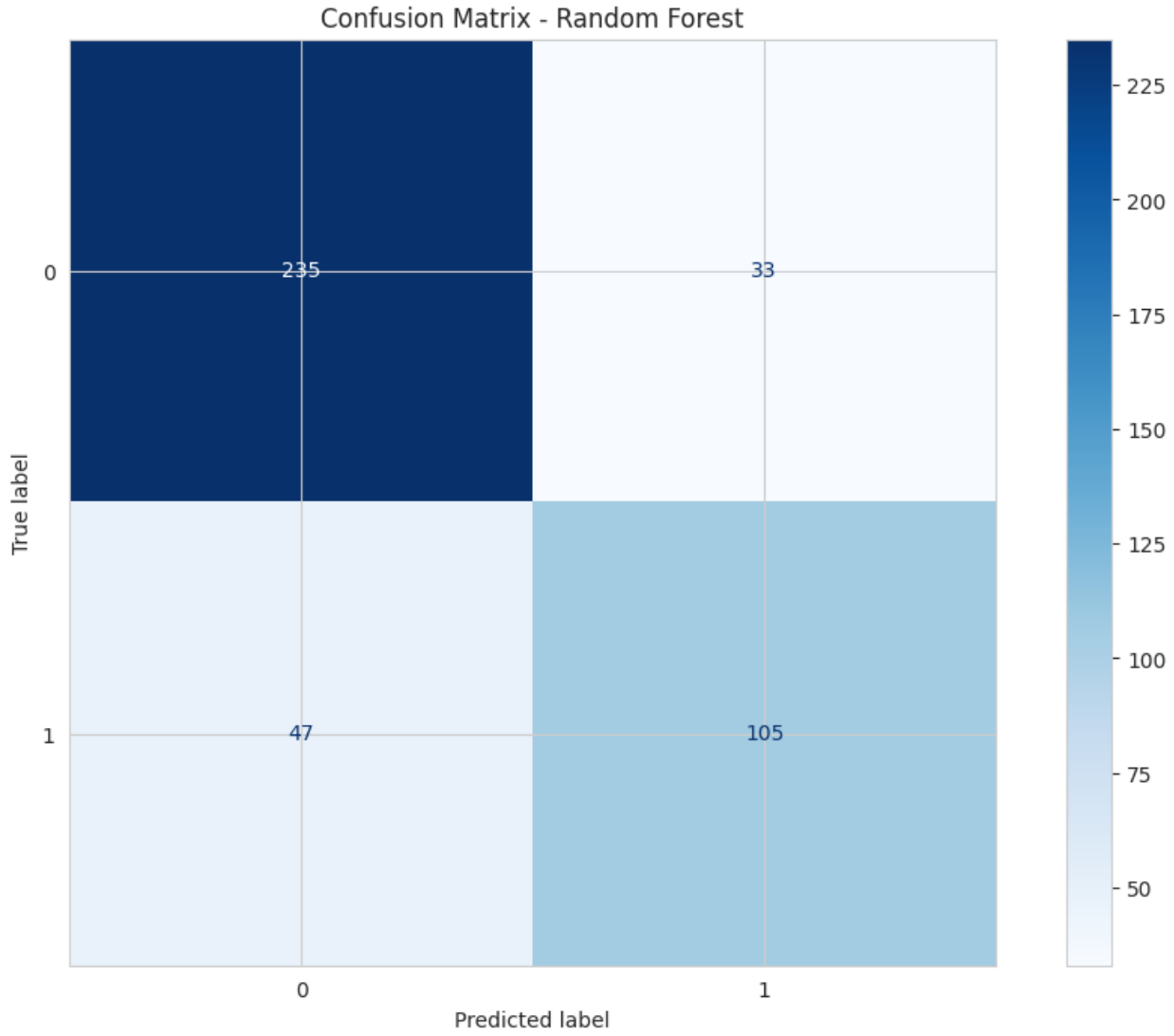

Figure 3: Confusion Matrix for Random Forest Model

Analysis: The Random Forest model achieved 81% accuracy. The confusion matrix reveals that the model is effective but not perfect, with its main challenge being False Negatives (incorrectly predicting a user will *not* meet their goal when they actually do. This performance is strong enough to provide a valuable, probabilistic input to our recommendation engine

AI Explainability and Governance

Feature Importance (Explainability):

To understand the "why" behind our model's predictions, we analyzed the feature importances from the Random Forest model.

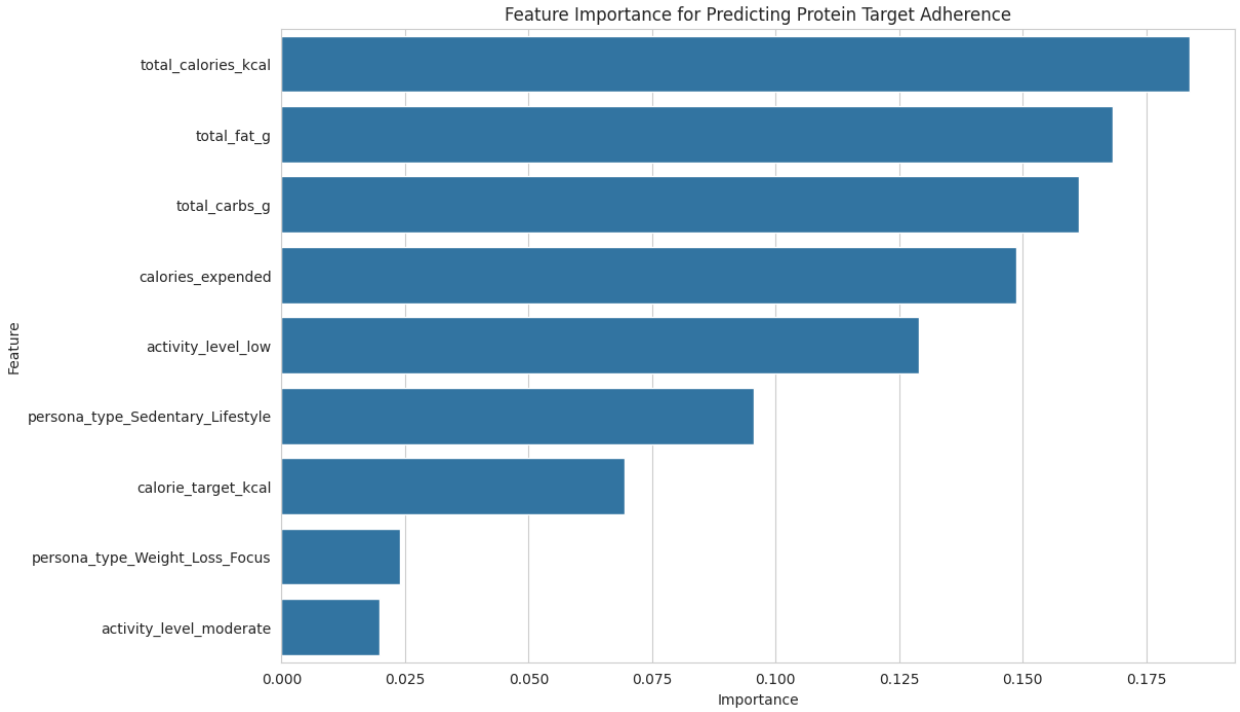

Figure 4: Feature Importance for Predicting Protein Target Adherence

Analysis: The plot shows that total_calories_kcal, total_fat_g, and total_carbs_g are the most influential factors. This is logical, as the overall nutritional context of a day heavily influences whether a specific macro target is met. The importance of calories_expended and persona_type validates our integrated approach of considering both activity and user profiles.

Model Development

We used the **Random Forest Classifier** to predict if the user is meeting their health goals. It works well with structured data and gives good results.

Challenge: Imbalanced Data

Most users in our dataset were marked as not meeting their goal (label 0). So we used **SMOTE** (Synthetic Minority Oversampling Technique) to balance the dataset by generating more 1s.

Model Steps:

- Split data into train/test sets
- Applied SMOTE
- Trained model with 100 trees and max depth = 6

Performance:

- Accuracy: **78%**
- F1 Score: **0.73**

We evaluated the model using:

- Confusion Matrix
- Classification Report
- Sample Predictions
- Feature Importance Bar Chart

Recommendation Engine

Apart from predictions, we made a rule based system to give easy suggestions. These rules are based on nutrient levels:

- If protein is low recommend tofu, dal, paneer
- If water intake is low suggest drinking 1–2 glasses of water

- If calorie intake is too high suggesting fruits instead of snacks

The rules are written in simple Python if-else logic. The results are printed clearly in the notebook. For example:

“Your protein is low. Consider adding dal or boiled eggs.”

This rule engine works alongside the machine learning model. So, the user doesn’t just get a number they also get clear tips to improve.

Development Roles and Process

All four team members had equal roles. We divided our work based on each member’s strengths and interest areas. We used simple tools like WhatsApp for daily communication, Zoom for weekly meetings, and Google Docs to collaborate on reports and task lists. All technical development was done using Google Colab, which allowed us to work together online easily and efficiently.

- Aashish contributed to building and refining user personas, helped draft rule based health suggestions, and worked on organizing our documentation and reviewing outputs to ensure clarity and user friendliness.
- Meet focused on developing the Random Forest model, tuning its parameters, and handling the training and evaluation stages.
- Sabin managed the data preprocessing, including cleaning, normalization, and handling missing values to prepare the dataset for training.
- Jaykesh was responsible for visualizations, structuring the notebook, and ensuring all outputs (like graphs and tables) were presented neatly and clearly.

We followed best practices throughout development:

- Used functions to avoid repeating code
- Added comments to explain our work
- Kept backup versions to avoid data loss
- Reviewed each other’s work regularly

By working together and supporting each other, we completed over **70%** of the system by this milestone with working predictions, a functioning rule engine, and meaningful visual outputs.

Evaluation and Testing

We checked if the system is working properly using different methods. First, we checked the machine learning model using accuracy, F1 score, and the confusion matrix. Then, We checked our model using:

- Confusion matrix (for true/false positive/negatives)
- Accuracy score
- F1 Score
- Graphical results

After using SMOTE, our model became balanced and worked better. We also tested rules with sample user logs and saw that suggestions were correct and realistic.

We showed outputs using:

- Pie charts
- Bar graphs
- Printed comments

These visuals made the results easier to understand.

Ethics and AI Governance

We followed ethical practices during development. In the development of NutriAI, we made sure to follow proper **AI ethics and responsible data practices**. Since the project deals with health, we took special care to avoid any kind of harm, bias, or privacy violation. Our goal was to build a system that is safe, fair, and helpful for everyone.

Here are the steps we took to follow ethical guidelines:

- Used Public Data from Kaggle: To design our food and nutrition data, we used a publicly available dataset from Kaggle, which is open for learning and research. We used that as a

base and then created our own synthetic logs. This helped us avoid privacy issues while still working with realistic information.

- No Use of Real User Data: We did not collect or ask for any personal or real health data from users. Instead, we created a fully synthetic dataset using Python scripts.
- Diverse and Fair Personas: We created 8 different user personas representing various ages, genders, professions, and goals (like weight loss, muscle gain, etc.). This helped us reduce bias and make the system inclusive.
- Medical Disclaimer: We clearly stated in our report and outputs that NutriAI is not a real medical system. It is meant for academic and educational purposes only and should not be used for diagnosis or treatment.
- Safe Suggestions Only: Our rule engine provides simple, everyday tips like drinking more water or eating fruits, dal, or paneer. There are no extreme diets or unsafe advice in the system.
- Open and Transparent: All our code is shared on Google Colab and GitHub. We explained how the model works, how the rules are applied, and how data is handled. This way, everything is clear and open for review.

By following these steps, we ensured that NutriAI is **ethical, safe, and responsible**. It respects privacy, avoids bias, and helps users in a simple and non intrusive way.

Challenges and Learnings

We faced some difficulties. Cleaning the data took time because some entries were missing or wrong. Creating balanced synthetic logs was also tricky. At first, the model gave poor results because of imbalanced data. Some challenges we faced:

- Data Imbalance: Too many 0s in the target column. Solved using SMOTE.
- Data Cleaning: Units were not the same; we had to normalize.
- Rule Conflicts: Some rules gave wrong suggestions at first. We tested and corrected them.
- Notebook Load: Google Colab slowed down when running too much data. We split parts into functions and cells.

Even with these issues, we learned a lot and fixed everything with teamwork and planning.

Next Steps

Now that we have completed most of the core parts of NutriAI, we are ready to move into the final phase of our project. For the third milestone, we want to focus on improving the user experience, making the system more interactive, and preparing everything for the final demo and submission. Our goal is to take all the feedback we received and polish the project so that it works smoothly, looks professional, and gives value to users in a clear and simple way.

To do this, we plan to work on the following:

- Add a Streamlit UI: This will allow users to interact with NutriAI through buttons and input forms instead of running code. It will make the system easier and more user friendly.
- Add Adaptive Rules: We want to improve our rule engine so that it can learn from past data and give smarter suggestions over time. For example, if a user is always low on protein, the system can highlight protein rich foods more often.
- Collect Feedback: We will ask classmates, friends, and mentors to try the system and give feedback. This will help us make final changes based on real opinions.
- Polish the Code and Notebook: We will clean up the code, remove unused parts, organize the cells, and make sure everything runs smoothly from start to finish.
- Create a Demo Video and Presentation: We will record a short video showing how NutriAI works and prepare a presentation that explains our journey, progress, and final results.
- Publish on GitHub: All code, reports, and visuals will be uploaded to GitHub for public access. This keeps the project transparent and professional.

By completing these steps, we aim to deliver a complete and polished system that not only meets all project requirements but is also useful, presentable, and easy to understand.

Conclusion

So far, NutriAI is performing well and meeting our expectations. The machine learning model gives accurate predictions based on food intake and activity logs, and the rule based engine provides simple, relevant suggestions that users can follow easily. We built our system step by step — starting with clean data, designing realistic personas, training our model, and adding a recommendation layer — all inside Google Colab.

Throughout this project, we have stayed focused, collaborated actively, and overcome challenges like data imbalance and code errors. Each team member contributed meaningfully to keep us on track. We also made sure the project follows ethical rules, uses safe and diverse data, and is easy to understand for everyday users.

By the time we reach the third and final milestone, we aim to complete a fully working, interactive version of NutriAI with a user interface and real time suggestions. Our goal is to create a helpful AI assistant that can genuinely guide users toward better food and fitness choices, especially students and working professionals.

We are excited to complete this journey and share a finished project that is:

- Useful and user friendly
- Technically solid
- Ethically safe
- And visually engaging

With a few final improvements, we believe NutriAI will be a project we can be truly proud of.

References

Google. (n.d.). *Colaboratory: A research tool for machine learning education and research*. Google Colab. https://colab.research.google.com/

Kaggle. (n.d.). *Daily food nutrition dataset*. https://www.kaggle.com/datasets/kandij/food-nutrition-dataset

Matplotlib Development Team. (n.d.). *Matplotlib: Visualization with Python*. https://matplotlib.org/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Pandas Development Team. (2020). *pandas-dev/pandas: Pandas (v1.2.1)* [Software]. Zenodo. https://doi.org/10.5281/zenodo.3509134

Python Software Foundation. (n.d.). *Python 3 documentation*. https://docs.python.org/3/

Seaborn Development Team. (n.d.). *Seaborn: Statistical data visualization*. https://seaborn.pydata.org/

Streamlit Inc. (n.d.). *Streamlit documentation*. https://docs.streamlit.io/