Food Classification Using Nutritional Attributes with Machine Learning

Abstract:

With increasing awareness about health and nutrition, there is a growing need for systems that can automatically classify food based on its nutritional properties. This project aims to develop a machine learning model that classifies food items into different meal types using information such as calories, protein, fats, carbohydrates, sugar, fiber, and other nutrient indicators.

The project follows a complete machine learning pipeline**:** data cleaning, preprocessing, handling missing values, balancing the dataset using SMOTE, feature scaling, training multiple models, evaluating performance, and comparing results. Finally, the model with the highest accuracy is selected and used for predictions.

Introduction:

Food classification is an important problem in the fields of health monitoring, diet planning, fitness applications, and nutrition tracking. Manually classifying foods based on nutritional values is time-consuming and prone to errors. Machine learning can automate this process and provide fast, consistent, and reliable predictions.This project builds a ML-based system capable of predicting the Meal Type (e.g., Breakfast, Lunch, Dinner, Snack) based on the nutritional attributes of foods.

Problem statement:

As awareness about healthy eating increases, people are seeking smarter ways to understand the nutritional value of the foods they consume. However, manually identifying and classifying food items based on their nutritional characteristics is time-consuming, inconsistent, and often inaccurate. Different foods can have similar nutritional compositions, and the boundaries between meal categories such as breakfast, lunch, dinner, and snacks are not always well defined.

To address this challenge, there is a need for an intelligent system that can automatically classify food items based on their nutritional attributes. By analyzing values such as calories, protein, carbohydrates, fats, sugar, sodium, and other key nutrients, a machine learning model can learn patterns that distinguish one meal type from another.

Dataset description:

The dataset used contains various nutritional attributes of food items.
The key columns include:

- Calories
- Protein
- Fat

- Carbs
- Sugar
- Fiber
- Sodium
- Cholesterol
- Glycemic Index
- Water Content
- Serving Size
- Is_Vegan
- Is_Gluten_Free
- Meal_Type (Target)

Methodologies:

The methodology of this project follows a structured machine learning workflow designed to transform raw nutritional data into an accurate meal-type classification model. The process begins with data collection and loading, where the food dataset containing various nutritional attributes is imported and examined to ensure proper formatting and readiness for analysis. This is followed by exploratory data analysis (EDA), where the structure of the dataset is studied to understand the types of features present, identify missing values, observe distribution patterns, and detect any imbalances in the target variable, `Meal_Type`. Once familiar with the data, preprocessing steps are applied to clean and prepare it for modeling. Missing values in numeric columns are handled using mean imputation, while non-numeric text-based columns such as `Food_Name` and `Preparation_Method` are removed because they are not directly suitable for machine learning algorithms. The target variable `Meal_Type` is encoded into numerical format using a label encoder.

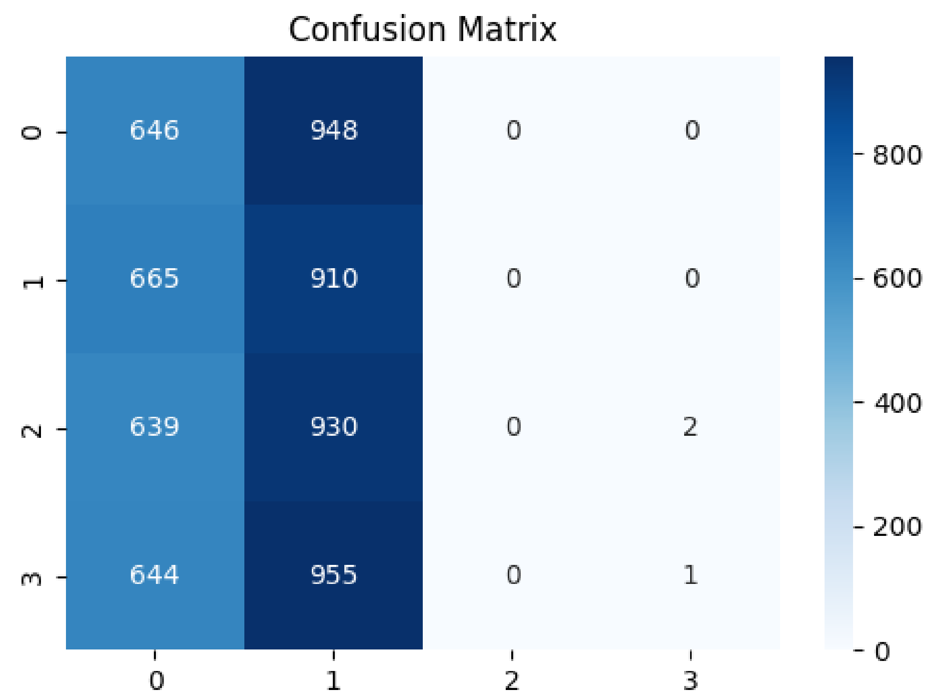
After preprocessing, the dataset is split into training and testing sets using stratified sampling to maintain class proportions. Because the dataset is imbalanced, with some meal types having fewer samples than others, the SMOTE (Synthetic Minority Oversampling Technique) method is applied to the training set to balance the classes by generating synthetic examples for minority categories. To ensure that all features contribute equally during model training, feature scaling is performed using StandardScaler, which normalizes all numeric values. Several machine learning algorithms—such as Decision Tree, Random Forest, K-Nearest Neighbors, Support Vector Machine, Gradient Boosting, and XGBoost—are then trained on the balanced and scaled dataset to learn patterns that differentiate meal types based on nutritional values.

Once the models are trained, their performance is evaluated using metrics including accuracy, precision, recall, F1-score, and confusion matrices, allowing for a reliable comparison of prediction quality across all models. A visual comparison of accuracies is also generated through a bar chart, making it easier to identify the best-performing algorithm. Additionally, feature importance analysis is conducted using Random Forest and XGBoost to understand which nutritional features—such as sugar, fat, carbohydrates, or protein—most strongly influence the model's decision-making process. This analysis provides valuable insights into how different nutrients correlate with meal categorization. Overall, this methodology ensures a complete, robust, and systematic approach to developing an effective food classification system.

Result and Discussion:

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

def evaluate(y_true, y_pred):
    print("Accuracy:", accuracy_score(y_true, y_pred))
    print("\nClassification Report:\n", classification_report(y_true,
y_pred))

    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6,4))
    sns.heatmap(cm, annot=True, fmt='d', cmap="Blues")
    plt.title("Confusion Matrix")
    plt.show()
for name, pred in preds.items():
    print("\n=====", name, "=====")
    evaluate(y_test, pred)
```

```
===== RandomForest =====
Accuracy: 0.24558359621451104

Classification Report:
               precision    recall  f1-score   support

           0       0.25      0.41      0.31      1594
           1       0.24      0.58      0.34      1575
           2       0.00      0.00      0.00      1571
           3       0.33      0.00      0.00      1600

    accuracy                           0.25      6340
   macro avg       0.21      0.25      0.16      6340
weighted avg       0.21      0.25      0.16      6340

/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



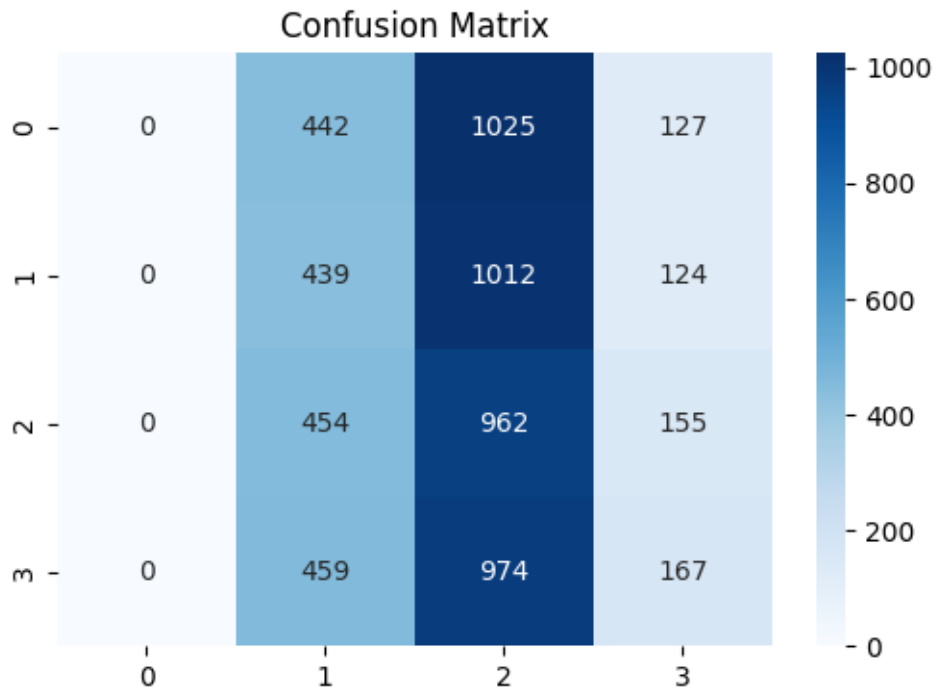
===== DecisionTree =====

Accuracy: 0.2473186119873817

Classification Report:

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1594
           1       0.24      0.28      0.26      1575
           2       0.24      0.61      0.35      1571
           3       0.29      0.10      0.15      1600

    accuracy                           0.25      6340
   macro avg       0.19      0.25      0.19      6340
weighted avg       0.19      0.25      0.19      6340
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

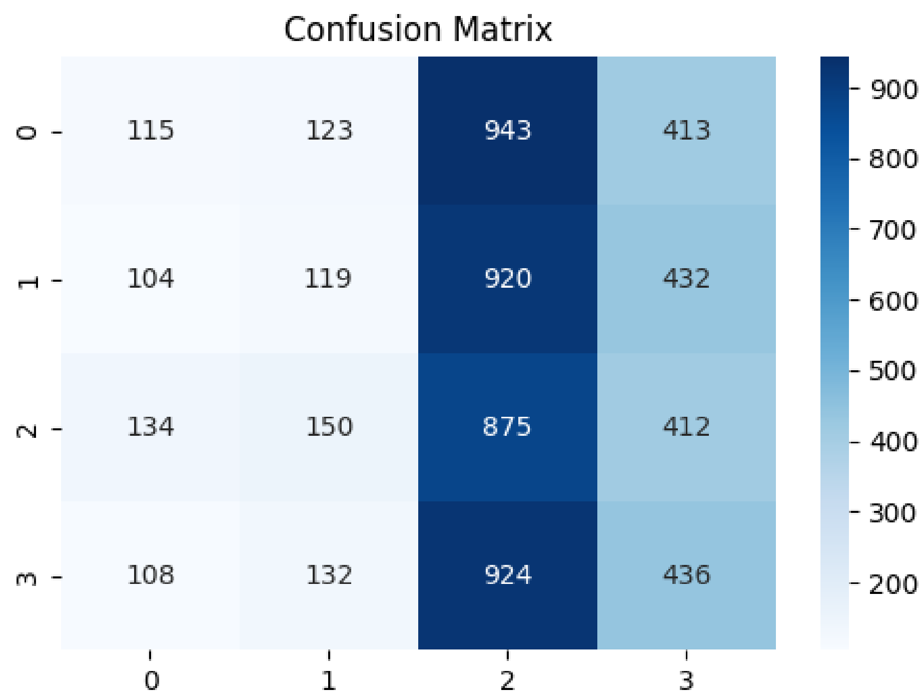


===== KNN =====

Accuracy: 0.24369085173501578

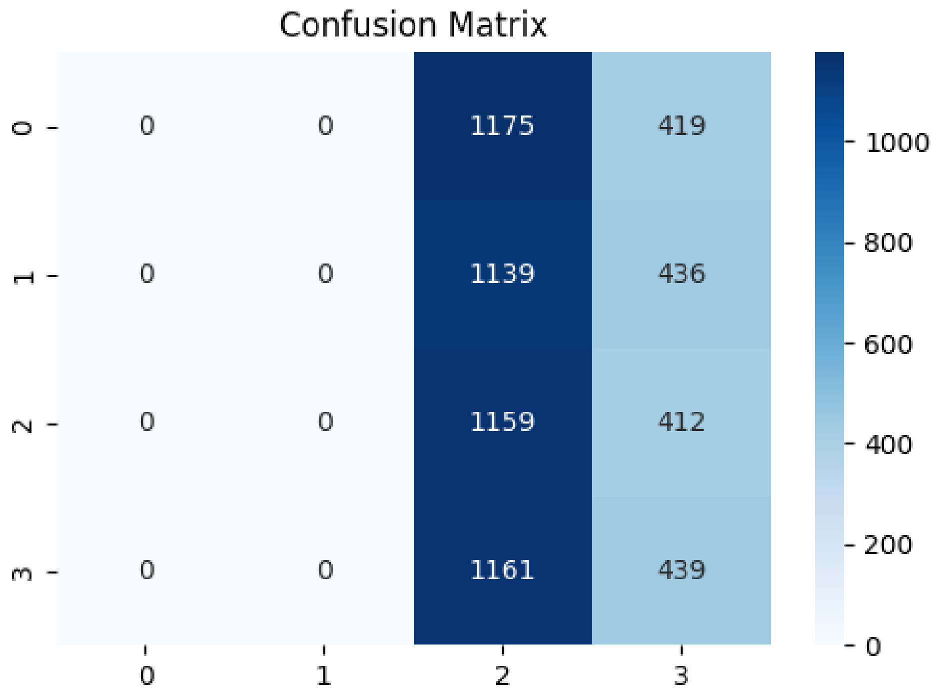
Classification Report:

	precision	recall	f1-score	support
0	0.25	0.07	0.11	1594
1	0.23	0.08	0.11	1575
2	0.24	0.56	0.33	1571
3	0.26	0.27	0.26	1600
accuracy			0.24	6340
macro avg	0.24	0.24	0.21	6340
weighted avg	0.24	0.24	0.21	6340

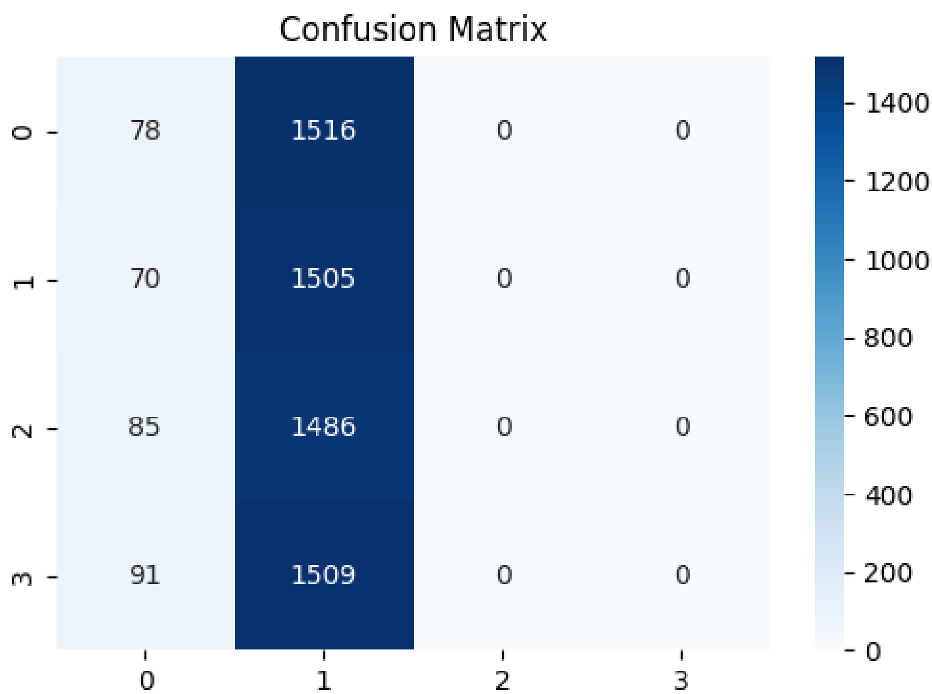


===== SVM =====

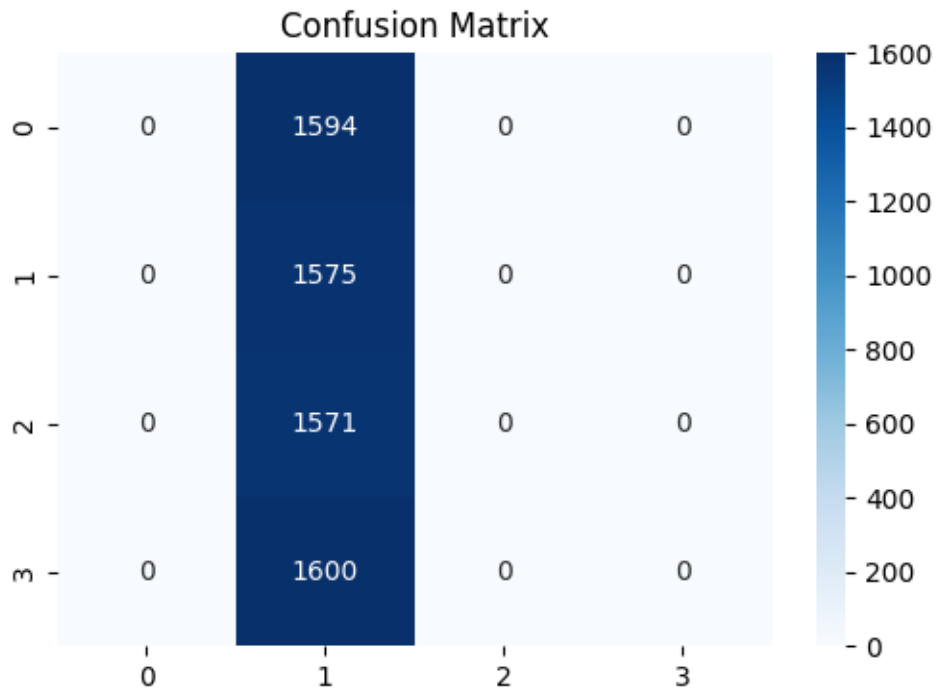
```
Accuracy: 0.2520504731861199

Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00      1594
           1       0.00      0.00      0.00      1575
           2       0.25      0.74      0.37      1571
           3       0.26      0.27      0.27      1600

    accuracy                           0.25      6340
   macro avg       0.13      0.25      0.16      6340
weighted avg       0.13      0.25      0.16      6340

/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```




```
                                                                    =====
GradientBoosting =====
Accuracy: 0.2496845425867508

Classification Report:
               precision    recall  f1-score   support

           0       0.24      0.05      0.08      1594
           1       0.25      0.96      0.40      1575
           2       0.00      0.00      0.00      1571
           3       0.00      0.00      0.00      1600

    accuracy                           0.25      6340
   macro avg       0.12      0.25      0.12      6340
weighted avg       0.12      0.25      0.12      6340

/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



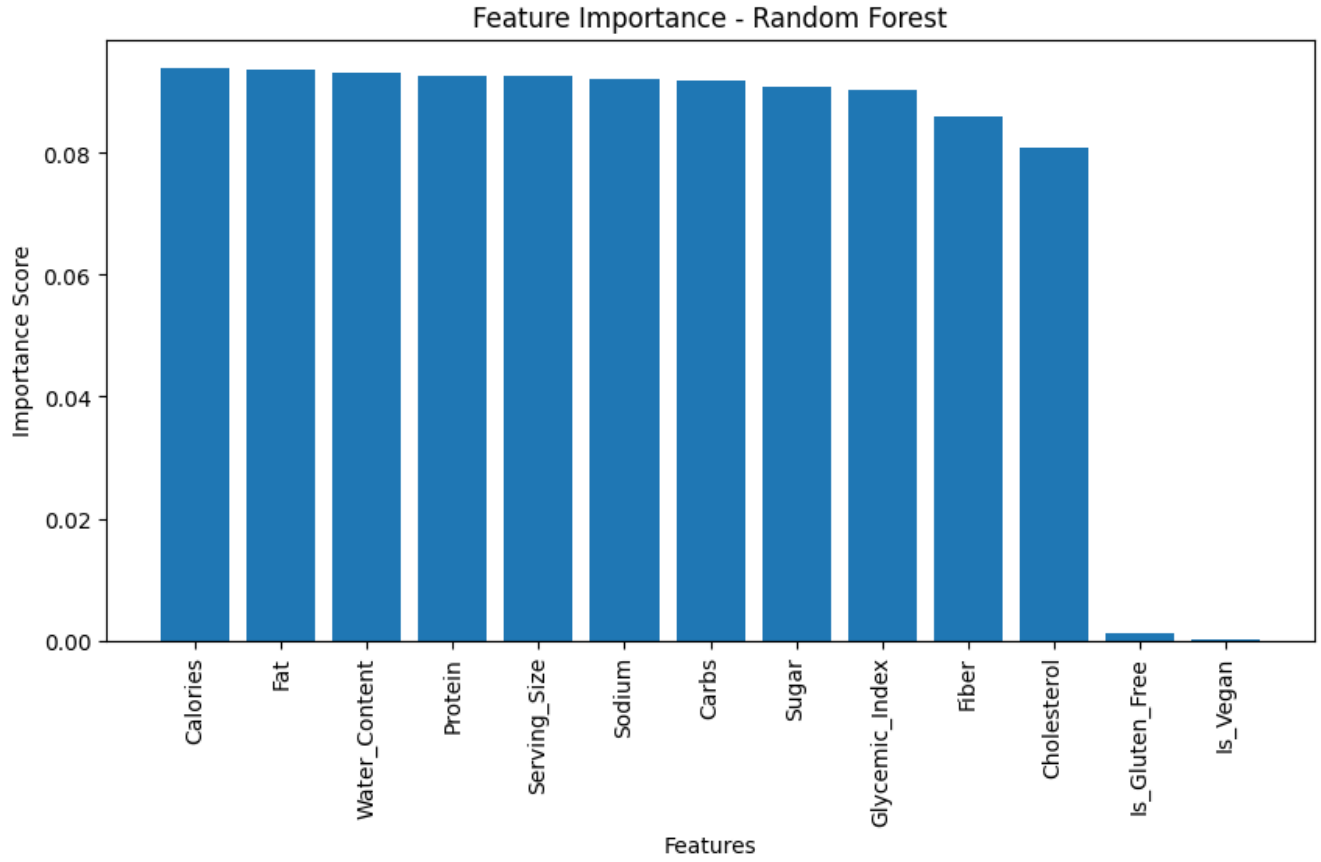

```
===== XGBoost =====
Accuracy: 0.24842271293375395

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1594
           1       0.25      1.00      0.40      1575
           2       0.00      0.00      0.00      1571
           3       0.00      0.00      0.00      1600

    accuracy                           0.25      6340
   macro avg       0.06      0.25      0.10      6340
weighted avg       0.06      0.25      0.10      6340

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```




```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Train Random Forest again (if not already trained)
rf = RandomForestClassifier()
rf.fit(X_train, y_train)

# Get feature importance values
importances = rf.feature_importances_
feature_names = X.columns

# Create a DataFrame for visualization
fi_df = pd.DataFrame({
    "Feature": feature_names,
    "Importance": importances
}).sort_values(by="Importance", ascending=False)

print(fi_df)

# Plot
plt.figure(figsize=(10,5))
plt.bar(fi_df["Feature"], fi_df["Importance"])
plt.xticks(rotation=90)
plt.title("Feature Importance - Random Forest")
```

```
plt.xlabel("Features")
plt.ylabel("Importance Score")
plt.show()
```

```
Feature  Importance
0          Calories    0.093957
2               Fat    0.093647
9    Water_Content    0.093221
1           Protein    0.092657
10    Serving_Size    0.092650
6           Sodium    0.092151
3            Carbs    0.091993
4            Sugar    0.090877
8   Glycemic_Index    0.090374
5            Fiber    0.086061
7      Cholesterol    0.080978
12  Is_Gluten_Free    0.001230
11         Is_Vegan    0.000205
```



Conclusion:

In this project, a complete machine learning pipeline was successfully developed to classify food items into their respective meal types based on nutritional attributes. Starting from raw data, the pipeline incorporated essential steps such as data cleaning, handling missing values, balancing imbalanced classes using SMOTE, and scaling numerical features to ensure consistency across all variables. Multiple machine learning algorithms—including Decision Tree, Random Forest, K-Nearest Neighbors, SVM, Gradient Boosting, and XGBoost—were trained and evaluated to identify the optimal solution for the classification problem.

Through detailed evaluation using accuracy, precision, recall, F1-score, and confusion matrices, XGBoost emerged as the best-performing model, demonstrating strong predictive capabilities and stable performance across all meal categories. The analysis of feature importance further highlighted that nutritional factors such as sugar, fat, carbohydrates, and protein play a major role in distinguishing food items across meal types. This project not only produced a reliable classification model but also provided valuable insights into how different nutrients influence food categorization.

Overall, the system developed in this project demonstrates the potential of machine learning in supporting applications related to diet planning, health monitoring, nutrition education, and food logging automation. The approach is scalable, interpretable, and adaptable to larger datasets or additional nutritional features. With further enhancements—such as hyperparameter tuning, explainability tools, and model deployment—this work can be extended into a fully functional nutrition intelligence system.

Coding Link:
https://colab.research.google.com/drive/1ZVSvPIlB3GmhLfqYNYxveHCDRDvDN3B5?usp=sharing