**Project Title:** Prediction of Daily Calorie Intake Using Machine Learning on Food Nutrition Data

Name -> Rajat Gupta Roll no.->220107102

**Introduction**

Provide a brief overview of the context and relevance:

* Accurate estimation of calorie intake is critical for personalized diet planning, public health monitoring, and nutritional research.
* Manual dietary assessment is time‑consuming and error‑prone; machine learning offers a scalable way to predict caloric values from food composition.
* A robust predictive model can support dietitians, food engineers, and health applications by automating calorie estimation based on nutrient profiles.

**Objectives**

* Assess the performance of machine learning models in predicting calorie content from food nutrition data.
* Develop a predictive pipeline—from data cleaning through model evaluation—for daily calorie estimation.
* Identify key nutritional features (protein, carbohydrates, fat, fiber, sugars, etc.) driving prediction accuracy.
* Compare baseline (e.g., linear) versus ensemble methods (Random Forest) for regression performance.

**Problem Statement**

“To design a reliable machine learning model that predicts the caloric value of daily food items using their nutritional composition, achieving better accuracy than simple statistical baselines.”

**Significance of the Study**

* Enables automated dietary tracking and personalized nutrition recommendations.
* Supports food engineers in product formulation by estimating energy density from ingredient profiles.
* Aids chemical engineers in designing nutritionally optimized food processes.
* Contributes to public health by improving large‑scale dietary assessment accuracy.

**Data Description and Preprocessing**

**Data Sources:**

* Kaggle “Daily Food & Nutrition” dataset (daily\_food\_nutrition\_dataset.csv), containing ~1,000 entries with Date, Food Item, Category, Meal Type, and nutrient columns (Calories, Protein, Carbs, Fat, Fiber, Sugars, Sodium, Cholesterol, Water Intake) .

**Key Steps:**

1. **Data Cleaning**
   * Dropped missing values and duplicate rows.
   * Parsed Date column to datetime.
2. **Outlier Removal**
   * Filtered out entries with Calories ≥ 3,000 kcal as unrealistic outliers.
3. **Encoding & Feature Engineering**
   * Label‑encoded categorical features (Meal\_Type, Category).
   * Selected features: Protein (g), Carbohydrates (g), Fat (g), Fiber (g), Sugars (g), Sodium (mg), Cholesterol (mg), Meal\_Type, Category.
4. **Scaling & Splitting**
   * Standardized features using StandardScaler.
   * Split data into 80% training / 20% testing sets (random\_state=42).

**Methodology Flowchart**

Data Ingestion (CSV)

↓

Data Cleaning & Outlier Removal

↓

Encoding & Feature Engineering

↓

Train/Test Split & Scaling

↓

Model Training (Random Forest)

↓

Evaluation (MAE, MSE, R²)

**Model Selection & Rationale**

* **Random Forest Regressor** (n\_estimators=100, random\_state=42):
  + Handles nonlinear relationships and interactions between nutrients.
  + Robust to outliers and scales well with feature dimensionality.
* **Baseline Comparison:** A simple linear regression baseline was considered but deferred to Stage 2 for formal benchmarking.

**Model Training & Validation**

* **Training:** Fitted RandomForestRegressor on training set (X\_train, y\_train).
* **Validation:**
  + Predictions generated on X\_test.
  + No hyperparameter tuning yet—using default RF settings as a first pass.
  + Will incorporate grid/random search in subsequent stages.

**Evaluation Metrics**

* **Mean Absolute Error (MAE):** 139.53 kcal
* **Mean Squared Error (MSE):** 26,094.63 (kcal²)
* **R‑Squared Score (R²):** –0.039 (indicates model underperforms a horizontal mean predictor) .

**Deployment Strategy**

* Package the trained model in a RESTful API (e.g., Flask or FastAPI).
* Integrate with nutrition‑tracking web or mobile applications.
* Develop a simple dashboard for visualizing actual vs. predicted calories and feature importances.

**Tools and Libraries Used**

* **Data Manipulation:** NumPy, Pandas
* **Visualization:** Matplotlib, Seaborn
* **Preprocessing & Modeling:** scikit‑learn (LabelEncoder, StandardScaler, RandomForestRegressor)
* **Environment:** Python 3.9+, Jupyter Notebook

**Scalability and Optimization**

* Containerize with Docker for reproducible deployments.
* Employ batch processing for large‑scale nutritional datasets.
* Use GPU acceleration (e.g., with RAPIDS) for faster tree‑based model training.
* Plan to implement hyperparameter tuning (GridSearchCV) and feature selection.

**Use Case in Chemical Engineering**

* Formulating nutrient‑rich food products by predicting caloric density from ingredient profiles.
* Process control: adjusting reactor conditions in food manufacturing to achieve target energy content.
* Waste reduction: optimizing feedstock blends for consistent nutritional output.

**Expected Impact**

* Automates calorie estimation, reducing manual entry errors.
* Empowers personalized diet plans, improving health outcomes.
* Enhances food engineering workflows with predictive nutrition modeling.
* Lays groundwork for AI‑driven nutritional analytics in process industries.

**Conclusion**

* **Preliminary Findings:**
  + Data cleaning and feature engineering pipeline established.
  + Initial Random Forest model yields MAE ≈ 140 kcal but R² is negative, indicating room for improvement.
* **Challenges Faced & Solutions Explored:**
  + **Data Quality:** Inconsistent nutrient units and outliers; addressed via strict filtering and standardization.
  + **Model Performance:** Negative R² suggests overfitting or missing predictors; will explore additional features and models.
* **Next Steps:**
  + Perform hyperparameter tuning and cross‑validation.
  + Compare multiple algorithms (e.g., XGBoost, SVR, deep neural nets).
  + Conduct feature importance analysis and possibly engineer interaction terms.
  + Expand dataset or integrate external nutrition databases for richer training data.

**References**

* Kaggle. “Daily Food & Nutrition Dataset.” Available at: daily\_food\_nutrition\_dataset.csv.
* Pedregosa, F., et al. “Scikit‑learn: Machine Learning in Python.” Journal of Machine Learning Research, 2011.
* Chen, T., & Guestrin, C. “XGBoost: A Scalable Tree Boosting System.” KDD 2016.