Import all the required libraries

**1.Data Understanding and Exploration**

**Step 1: Load the Dataset**

* Loads the dataset from the specified CSV file path into a pandas DataFrame named df.

**Step 2: Check Dataset Size**

* Prints the number of rows and columns in the dataset using the .shape attribute.

**Step 3: Examine Class Distribution (Meal\_Type)**

* Calculates and prints the distribution (count) of each category in the Meal\_Type column.
* Useful for identifying class imbalance in the dataset.

**Step 4: Visualize Class Distribution**

* Creates a bar chart to visually show how many samples exist for each Meal\_Type.
* Uses color and formatting for better readability.

**Step 5: Sample Entries from Each Class to Understand Inter-Class Variation**

* Randomly samples 3 records per Meal\_Type group.
* Helps in understanding the variation of records within and across different classes.
* Displays the Meal\_Type and the first few columns for inspection.

**Step 6: Check Missing Values (Noise Indicator)**

* Checks for missing (null) values in the dataset.
* Calculates both the absolute count and percentage of missing values for each column.
* Displays a summary to identify noisy or incomplete features.

**Step 7: Visualize Inter-Class Variation for Some Features**

* Selects the first four numeric (float64) features for visualization.
* Samples 200 rows to avoid clutter in the plot.
* Uses seaborn’s pairplot to plot pairwise feature relationships, color-coded by Meal\_Type.
* Includes kernel density estimates (kde) on the diagonals to show distribution per feature.

**2. Data Preprocessing**

**Step 1: Handle Missing Values**

* **Objective**: Remove all rows that contain any missing (NaN) values.
* **Reason**: Missing data can distort statistical analysis and machine learning models.
* **Note**: Alternatively, you could impute missing values using methods like mean/median imputation.

**Step 2: Detect and Remove Outliers Using IQR (Interquartile Range)**

**Objective**: Cap extreme values (outliers) instead of removing rows entirely.

**Method**:

* Calculate IQR for each numeric column.
* Define lower and upper bounds.
* Replace values below or above bounds with respective thresholds.

Why Capping? This preserves data size while reducing skew caused by outliers.

**Step 3: Normalize or Standardize Numeric Features**

* **Objective**: Standardize features to have zero mean and unit variance.
* **Why**? Many ML algorithms (e.g., KNN, SVM, Logistic Regression) perform better on standardized data.
* **Tool Used**: StandardScaler from sklearn.preprocessing.

**Step 4: Remove Duplicate Records**

* **Objective**: Eliminate identical duplicate rows from the dataset.
* **Impact**: Prevents data redundancy and model bias due to repetition.
* **Final Output:** Preview the Cleaned and Scaled Data

Displays the top 5 rows of the cleaned and normalized dataset for inspection.

**3.Feature Engineering**

**Step 1**: **Encode the Target Column (Meal\_Type)**

* **Objective**: Convert categorical labels in the Meal\_Type column into numeric values for model compatibility.
* **Method**: Uses LabelEncoder to assign an integer to each unique class (e.g., Breakfast → 0, Lunch → 1, etc.).
* **Output**: New column Meal\_Type\_Label added to the DataFrame.

**Step 2**: **Select Numeric Features**

* **Objective**: Select only the numerical columns (features) for PCA.
* **Why**? PCA requires numerical input to compute variances and correlations.

**Step 3**: **Standardize the Features**

* **Objective**: Standardize the features so that each has a mean of 0 and standard deviation of 1.
* **Reason**: PCA is sensitive to feature scale—unstandardized data may bias the results toward features with larger numeric ranges.

**Step 4**: **Apply PCA to Reduce Dimensionality**

* **Objective**: Reduce the dataset to 2 principal components using Principal Component Analysis (PCA).
* **Why PCA?** It helps visualize high-dimensional data in 2D while retaining the maximum variance possible.
* **Output**: Transformed matrix X\_pca with two columns (PC1, PC2).

**Step 5**: **Visualize the PCA Output**

* **Objective**: Plot the PCA-transformed data to visually inspect how well the classes separate in reduced dimensions.
* **Method**: Uses seaborn’s scatterplot to plot PC1 vs PC2, color-coded by Meal\_Type.

**Explained Variance**

* **Displays**: How much variance each principal component explains.
* **Use**: Helps interpret how well the PCA captures the dataset's variability.

**4. Model Selection and Training**

**Step 1**: **Feature Selection**

* **Objective**: Prepare feature matrix X and target vector y.
* **Note**: Includes only numeric features from the cleaned dataset for model input.

**Step 2: Feature Scaling**

* **Purpose**: Normalize feature values to a standard scale (mean = 0, std = 1).
* **Why**? Required for distance-based models (like KNN) and improves convergence of gradient-based models (like logistic regression).

**Step 3:** **Train-Test Split**

* **Objective**: Split data into training (80%) and test (20%) sets.
* **Why**? Ensures model is evaluated on unseen data, reducing overfitting risk.

**Step 4: Define Classifiers**

**Goal**: Define multiple classification algorithms for benchmarking.

**Included Models:**

* **Logistic Regression**: Linear classifier
* **Decision Tree**: Tree-based model using splitting rules
* **Random Forest**: Ensemble of decision trees (bagging)
* **KNN**: Distance-based classification
* **XGBoost:** Gradient boosting with regularization
* **Gradient Boosting**: Another boosting technique (non-XGB)

**Step 5: Train and Evaluate Models**

**Process**:

* Train each model on X\_train, y\_train.
* Predict labels on X\_test.

**Compute**:

* Accuracy
* Classification Report (Precision, Recall, F1-score per class)
* Confusion Matrix

Storage: Results are saved in a dictionary results for each model.

**Step 6: Visualize Confusion Matrices**

* Goal: Plot heatmaps for the confusion matrix of each classifier.
* Layout: 3x3 grid (to accommodate up to 9 models, with unused plots deleted).
* Axes Labels: Show predicted vs actual labels.
* Color Map: Blues used for visual clarity.

**Model Performance Insights:**

**1.Overall Accuracy is Low**

* All models show accuracy around 24–26%, which is only slightly better than random guessing for a 4-class classification problem (random would yield ~25% accuracy).
* This indicates:
  + Either features are not discriminative enough, or
  + The dataset is highly overlapping/imbalanced.

**2.Best Performing Model**

* **Gradient Boosting** performed marginally better with 26.4% accuracy.
* Followed closely by Random Forest and XGBoost (~25.5%).

**3.Class-Specific Observations**

* Precision and recall are quite evenly low across all classes.
* No class consistently dominates across models.
* Some fluctuation:
* KNN had better recall for breakfast (32%) but dropped for snack (17%).
* lunch and snack see slightly better F1-scores in tree-based models.

**4.Implications**

* More informative features are needed — consider domain-specific feature engineering or using text/image embeddings if applicable.
* Imbalanced class distribution or feature noise may be limiting performance.
* PCA visualization earlier hinted at low class separability, which matches this result.