Department of Electrical and Computer Engineering

CSE 445: Machine Learning

Sec – 4

Spring 2025

Project : Phase # 1

Submitted By : GROUP – 04

Submitted To: Silvia Ahmed(sva)

Date : 08.04.2025

Contents:Error! Bookmark not defined.

1. Problem Definition and Dataset Description:

1.1 Problem Statement

This dataset is designed to train a model that classifies various types of food as either healthy or unhealthy. The model will analyze multiple columns of data, each containing nutrient information for the respective food items. By evaluating factors such as protein, vitamins, carbohydrates, and more, the model will determine whether a food item meets the criteria for being classified as healthy or unhealthy. The goal is to build a reliable classifier that can predict the health status of food items based on the nutrient content provided in the dataset.

1.2 Dataset Description

This dataset combined national open databases to analyze various aspects of food production, consumption, environmental impact and public health in Brazil.It gives a comprehensive approach to the food system,connecting agriculture,nutrition and sustainability.Some of the key issues analyzed are water usage,pesticide monitoring and food trends with the purpose of understanding the more far-reaching impacts of food choices on both human health and the environment.The data has been harmonized from multiple government sources to ensure consistency over time. Important data include food name, nutrient contentand sustainability indicators, making this dataset a valuavle resourse for agrifood system research. It also provide support for future food policies and climate models.

- **Source**: https://data.mendeley.com/datasets/mt4mj23j73/9
- **Size**: 439299
- **Features**: 51 (e.g., food name, energy content, protein content,etc.).
- **Target Variable**: Healthy food
- **Missing Values**: Our dataset includes some missing values in certain columns. These missing values will be handled through imputation techniques or by removing the rows or columns with excessive missing data, depending on the extent of the missing values (such as 50% of the data). Imputation methods such as replacing missing values with the mean, median, or mode for numerical columns, or using the most frequent category for categorical columns will be applied. If the missing data is minimal, affected rows or columns may be dropped to avoid skewing the analysis.

1.3 Type of Task

This project involves a classification problem where our goal is to categorize various food items into categories of "healthy" and "unhealthy" based on their nutrient composition. In this case, categories are binary(healthy= 1, unhealthy= 0). The model will differentiate between these two categories using features such as protein content, vitamins, fats, and minerals.

Significance: Classifying food items into categories of healthy and unhealthy has significant implications for public health, nutrition and sustainable food systems. Furthermore, classifying food items based on their nutrient data, this model can help researchers, policymakers and consumers make informed decisions about food choices. Additionally, it supports efforts to promote healthier diets and sustainable food production practices.

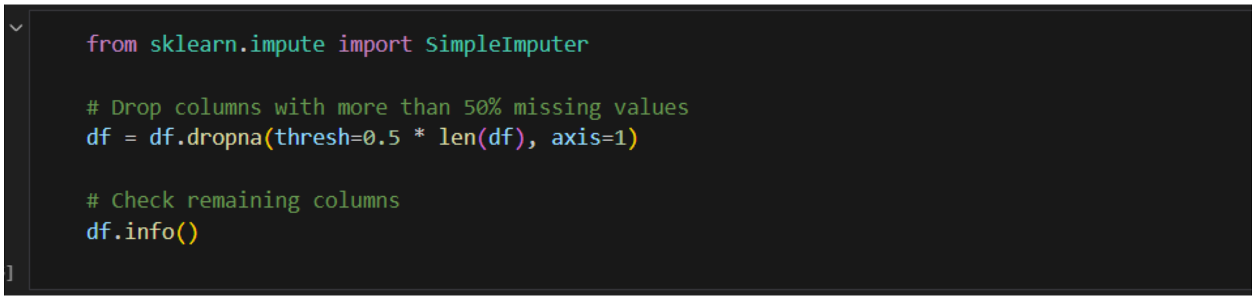
2.Data Preprocessing and Exploratory Data - Analysis(EDA):

2.1 Handling Missing Values:

In our dataset, the presence of missing values was initially checked using the **isnull()** and **sum()** functions, which helped us identify the features with missing data. We first printed the missing values for each column and then dropped the columns where more than 50% of the data was missing.
For the remaining missing values, we used imputation as a technique to fill in the gaps. In numerical columns, missing values were imputed with the mean of each respective column, a common approach where missing values are replaced with the average value of the feature.

For categorical columns, we imputed missing values with the mode, i.e., the most frequent value in the column, as this is a reasonable assumption that the majority class would likely be the most appropriate replacement.

Imputation was chosen to avoid data loss, ensuring that valuable data is retained for modeling without introducing significant bias.



```python
from sklearn.impute import SimpleImputer

# Drop columns with more than 50% missing values
df = df.dropna(thresh=0.5 * len(df), axis=1)

# Check remaining columns
df.info()
```

Figure: Code for dropping columns with more than 50% missing value

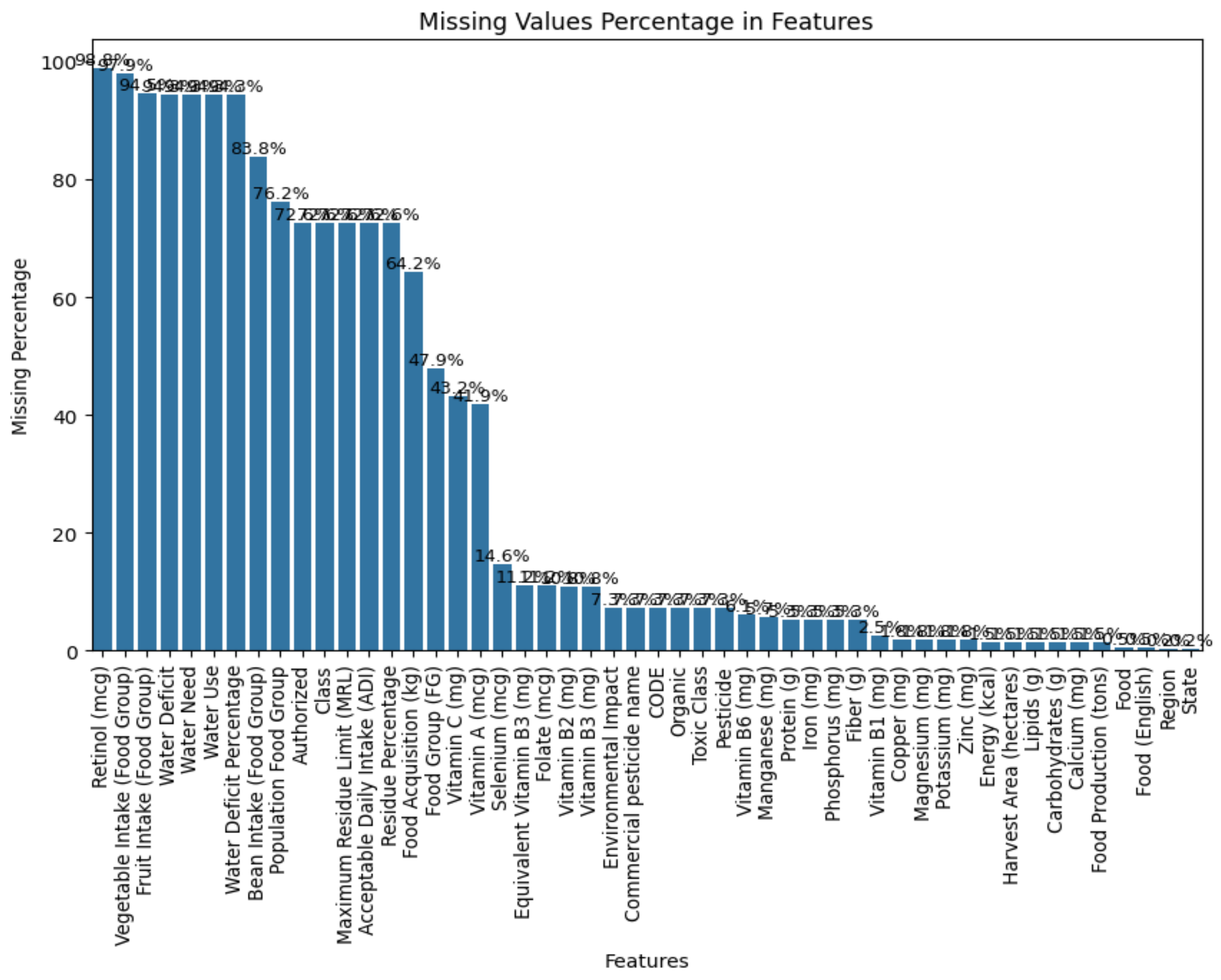


Figure: Missing Values Percentage in Dataset

2.2 Feature Engineering:

In the feature engineering step, several transformations were applied to the dataset to prepare it for model training:

i) Feature Creation (Target Variable Creation):

A new target variable, healthy, was created based on the health_score feature. The healthy column was defined as a binary classification target, where foods with a health score greater than or equal to the median health score were labeled as healthy (1), and those below the median were labeled as unhealthy (0).

ii) Feature Encoding:

The healthy column, which is the target variable, was encoded into a binary format (0 for unhealthy, 1 for healthy). Since this is already a numeric feature, no further encoding was necessary.

After identifying categorical columns in the dataset, we applied one-hot encoding to convert them into numerical format.

These transformations ensure that the data is in an appropriate format and scale for the models, improving the performance and accuracy of the classification algorithms.

iii) Feature Scaling:

Normalization of numerical features was performed using StandardScaler to scale the features to have a mean of 0 and a standard deviation of 1. This step ensures that all numerical features are on the same scale, which is important for machine learning algorithms that are sensitive to the scale of the data (such as Logistic Regression and Decision Trees).

The data was split into training and testing sets before scaling.The scaling was applied only to the numerical columns

2.3 Outlier Detection and Handling:

Outlier detection is an important step in data preprocessing as outliers can significantly affect model performance. In our project, we used the Interquartile Range (IQR) method to detect and handle outliers in the dataset. This method is effective in identifying values that fall outside the expected range, which can distort the analysis and model results.

i) Outlier Detection Using IQR

The IQR method calculates the difference between the 25th percentile (Q1) and 75th percentile (Q3) of the data. Any data point that falls below **Q1-1.5X IQR** or above **Q3+1.5XIQR** is considered as an outlier.

Step 1: Calculate the **Q1** (25th percentile) and **Q3** (75th percentile) values for each numerical feature.

Step 2: Calculate the IQR for each feature: **IQR=Q3−Q1**

Step 3: Identify outliers based on the threshold defined by 1.5 times the IQR:

Lower bound: **Q1 - 1.5 X IQR**

Upper bound: **Q3 + 1.5 X IQR**

ii) Handling Outliers: In our approach, we handled outliers by removing the rows that had extreme values identified by the IQR method. This ensures that the model training process is not skewed by data points that do not represent the general trend.

By removing outliers, we reduce the risk of overfitting and ensure that the models generalize better. The IQR method is chosen because it is a non-parametric approach, which doesn't make assumptions about the distribution of the data, making it robust for this type of task.

This method is simple, yet effective in improving model performance by ensuring that the data used for training is clean and representative of the underlying patterns in the dataset.

Number of Outliers detected by IQR method:

Feature	**Number of Outliers**
Energy (kcal)	81123
Protein (g)	84322
Lipids (g)	82879
Carbohydrate (g)	78737
Fiber (g)	83056
Calcium (mg)	73681
Magnesium (mg)	80781
Manganese (mg)	79407
Phospohorus (mg)	83125
Iron (mg)	82429
Potassium (mg)	75221
Copper (mg)	76400
Zinc (mg)	82717
Selenium (mcg)	35482
Vitamin A (mcg)	23701
Vitamin B1 (mg)	77081

Vitamin B2 (mg)	94241
Vitamin B3 (mg)	97672
Equivalent Vitamin B3 (mg)	91699
Vitamin B6 (mg)	79096
Folate (mcg)	35446
Vitamin C (mg)	21765

Original shape: (439299, 23)
Shape after removing outliers (IQR method): (291761, 23)
Total rows removed: 147538

2.4 Data Distribution and Summary Statistics:

Understanding the data distribution is essential for analyzing patterns, identifying skewness, and determining the appropriate preprocessing steps. In our project, we computed key statistical measures, including mean, median, standard deviation, and quartiles, to summarize the dataset.

Summary Statistics Calculation To obtain an overview of the dataset, we used the **describe() function** from pandas, which provides essential statistics for each numerical column, such as mean, median,standard deviation,minimum and maximum, percentiles(25%,50%,75%).

Standard deviation helps to measure how spread out the data points are while percentiles representes quartiles that help in understanding data distribution.

From the summary statistics, we observed:

- Some features had a high standard deviation, indicating significant variability.
- The mean and median were close for most features, suggesting a roughly symmetrical distribution.
- The minimum and maximum values helped identify potential outliers or data entry errors.

Summary statistics and visualizations help us:

- Understand the data spread and identify features with large variations.
- Detect skewed distributions, which may require transformations like log scaling.
- Confirm the presence of outliers, reinforcing the need for appropriate handling.

By analyzing these statistics, we ensured that our dataset was well-prepared for model training, reducing the chances of biases and errors in predictions.

2.5 Visualization:

Visualizations are crucial for understanding data before modeling. They help in:

- Identifying data distribution to apply appropriate transformations.
- Detecting outliers that can skew model predictions.
- Recognizing feature relationships to reduce multicollinearity.
- Understanding patterns in the dataset to choose the right algorithms.

We used histograms, box plots, correlation heatmaps and scatter plots to visualize our obtained data after preprocessing it.

i) Histograms

Histograms provide a graphical representation of the distribution of numerical features. By plotting histograms, we can determine whether a feature follows a normal distribution, is skewed, or has multiple peaks.

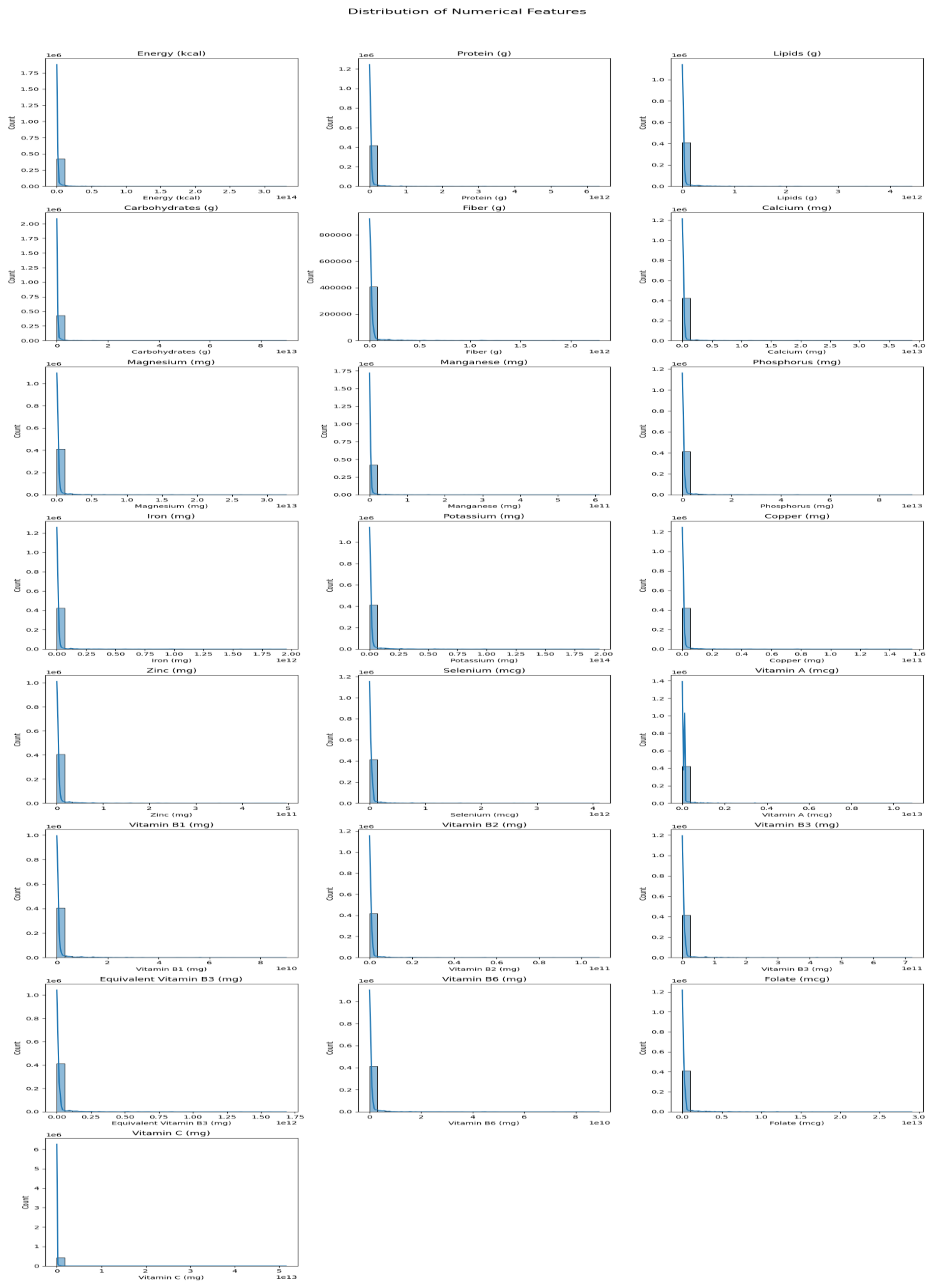


Figure : Histogram of all numerical columns

Insight:

Histogram obtained was observed to have the following:

- Data distribution is highly right-skewed. Almost all histograms show that most values are connected near 0 with a long tail towards the right. This means that most food have low values for these nutrients but a few have extremely high values.
- Many historgrams have unrealistic values such as 1e12 or 1e14. Energy in the range of 1e14 is way too high. It might occur due to scaling issues.
- Since our data has extreme high values, our model will struggle to predict the proper values. In order to fix this, feature scaling is required. The extreme long tail suggest the presence of outliers. Hence, we need to eliminate the outliers.

ii) Box Plot

Box plots help visualize the spread of numerical features and detect outliers using the interquartile range (IQR).

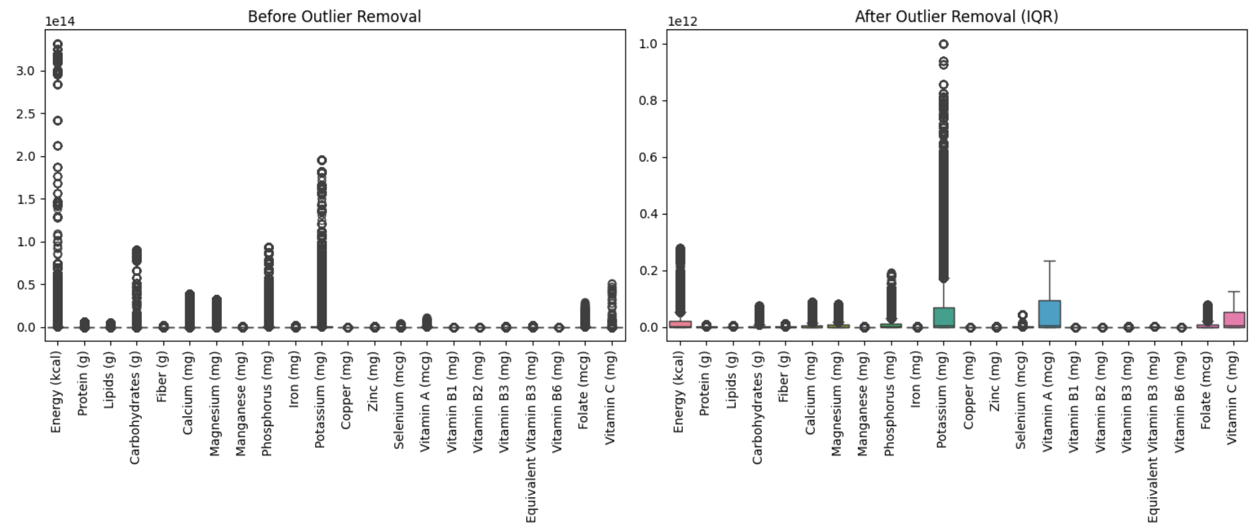


Figure: Box Plot Before and After Outlier Removal

Insight:

Before Outlier Removal: The dataset contains extreme values, especially in "Energy (kcal)", "Potassium (mg)", and "Iron (mg)". Many features have values that are significantly higher than others, indicating the presence of outliers.

After Outlier Removal (IQR Method): The extreme values have been significantly reduced. The distribution of the data looks more controlled, suggesting better generalization for machine learning models. Features such as "Potassium (mg)" and "Vitamin A (mcg)" still have some high values, but overall, the data looks more normalized.

Removing outliers helps in improving model performance by preventing extreme values from skewing predictions. If important data points were removed, we need to ensure that these outliers were truly anomalies and not valuable information.

iii) Correlation Heatmap:

A correlation heatmap helps us understand the relationships between numerical features. High correlation between two features may indicate redundancy.

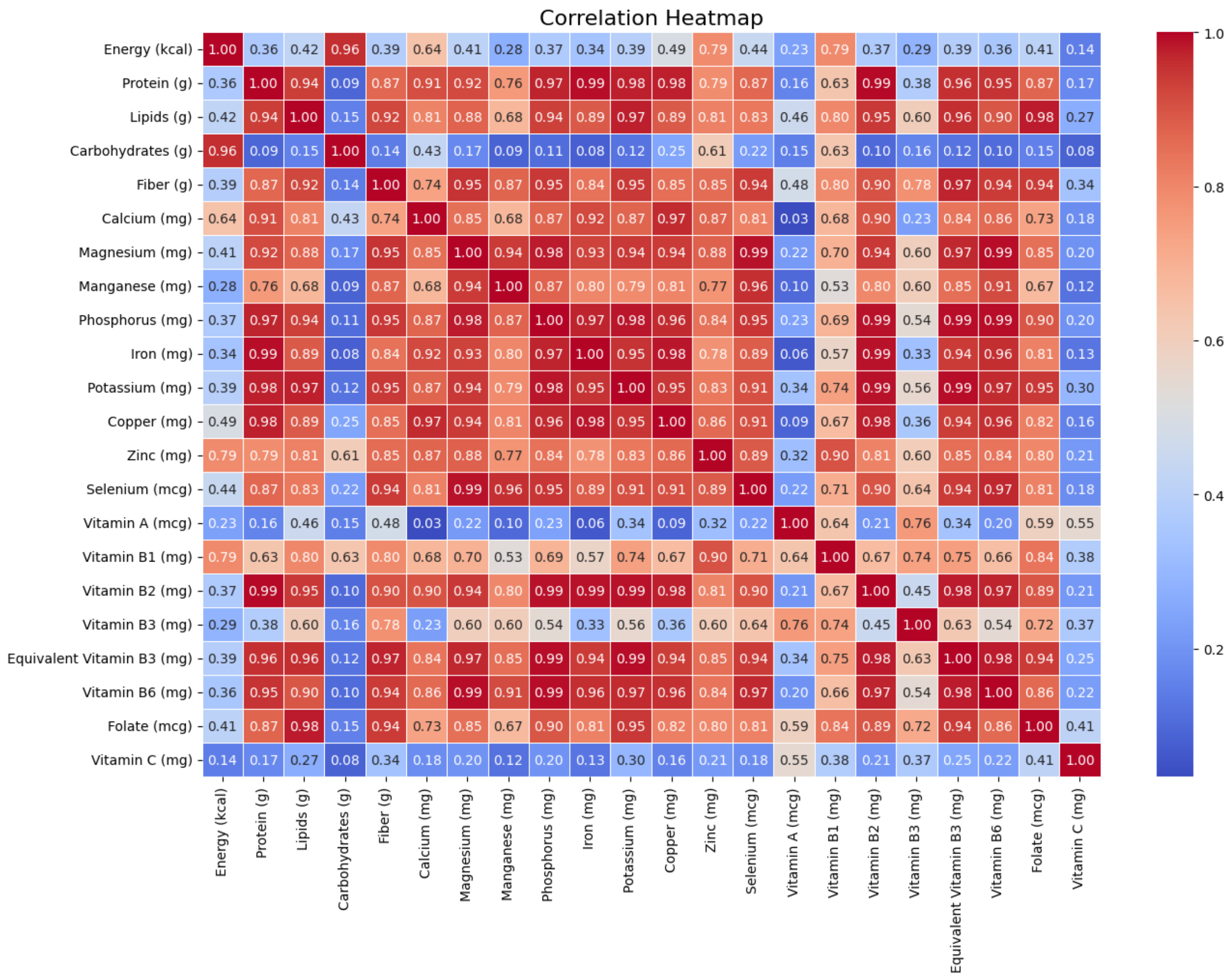


Figure: Correlation Heatmap

Insight: The heatmap shows the correlation between various features in the dataset.

Key Observations: Highly correlated features (Red color, close to 1.0): "Protein (g)" is highly correlated with "Lipids (g)" and "Phosphorus (mg)". "Carbohydrates (g)" and "Fiber (g)" have a moderate positive correlation. "Energy (kcal)" has strong correlations with "Lipids (g)" and "Protein (g)", which makes sense as macronutrients contribute to energy intake. Negative correlation (Blue color, closer to -1.0): "Vitamin C (mg)" shows negative correlations with multiple features, suggesting it behaves differently from macronutrient-based features.

Interpretation:

Highly correlated features may indicate redundancy, meaning we might need to drop some of them to avoid multicollinearity in machine learning models. A low correlation between certain features and the target variable could mean they might not be important predictors.

iv) Scatter Plot:

Scatter plots help visualize relationships between two numerical features and identify patterns, clusters, or linear relationships.

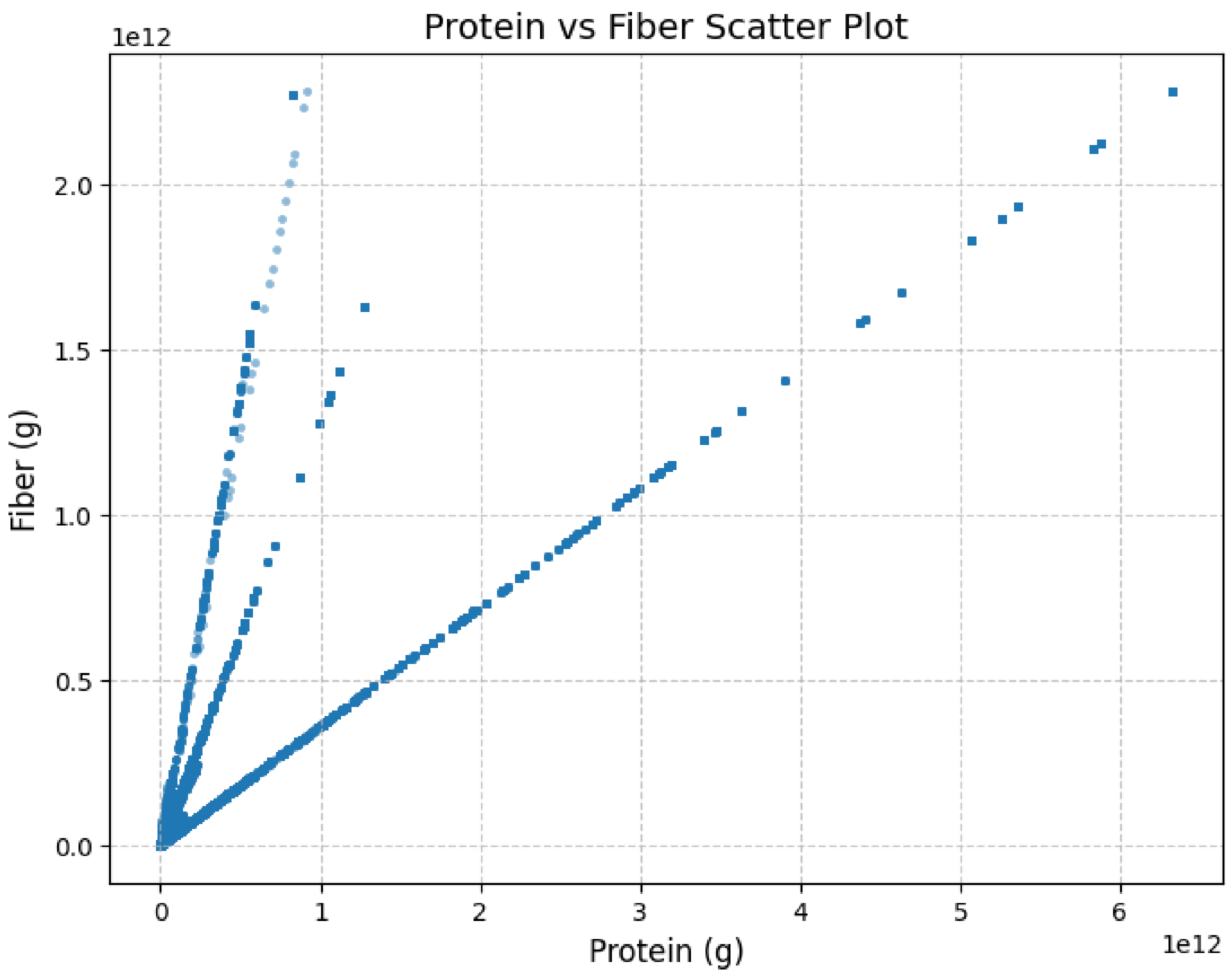


Figure: Scatter Plot between Protein and Fiber

Insight: The scatter plot compares "Protein (g)" and "Fiber (g)".

Key Observations: A distinct pattern suggests that higher protein values correspond with a range of fiber values. Some extreme outliers exist where fiber is unusually high or protein values are extreme. The presence of linear patterns indicates a potential relationship between the two variables.

Interpretation:

There might be specific food types that have both high protein and fiber, such as plant-based foods. If the trend is too sharp, it could indicate a data issue or incorrect unit conversions. If outliers exist, they should be investigated to determine whether they are true outliers or important variations. The IQR method successfully reduced extreme values, improving data distribution. Certain features are highly correlated, meaning we may need dimensionality reduction techniques like PCA or feature selection. Protein and Fiber show a relationship, but outliers need further examination. By leveraging these visualizations, we ensured that our preprocessing decisions were data-driven and enhanced model performance.

3. Model Training and Evaluation:

3.1 Classification Models:

In order to train the model, we created a healthy_score based on the key features from our remaining columns of our dataset which are vital in classification process. We split the data into training data(80% of the data) and testing data(20% of the data).

i) Logistic Regression: Logistic Regression is a linear classifier that models the probability of a binary outcome, making it well-suited for classification tasks like determining if a food item is healthy or unhealthy based on various features.

We used the preprocessed dataset with scaled numerical features to train the Logistic Regression model. The target variable indicates whether a food item is healthy (1) or unhealthy (0). The model computes the relationship between the features and the target, and then predicts the probability of each food being healthy.

This model was chosen due to its simplicity, interpretability, and efficiency. It works well for linear relationships and provides insight into how different features influence the prediction.

ii) Decision Tree Classifier: The Decision Tree Classifier builds a tree-like model of decisions and their possible consequences. It splits the data at each node based on feature values, making it easier to visualize and interpret.

This model was trained on the same preprocessed and scaled dataset, learning to classify food items into healthy or unhealthy categories by recursively splitting the dataset at each node based on the best feature.

We chose this model for its ability to handle both linear and non-linear relationships in the data. It also has the advantage of being more interpretable compared to other complex models. Moreover, it can capture interactions between features, which might be valuable for classifying foods based on their characteristics.

We considered the default hyperparameter, **max_depth,** to be 42.

3.2 Evaluation Metrics:

Classification Metrics:

After training the Logistic Regression and Decision Tree Classifier models, we evaluated their performance using several classification metrics to assess their effectiveness in predicting whether a food item is healthy or unhealthy.

- **Confusion Matrix:** A confusion matrix was generated to visualize the performance of both models in classifying the food items. It shows the true positive(TP), false positive(FP), true negative(TN), and false negative(FN) values. This matrix allows us to assess how well the models correctly predicted the healthy and unhealthy categories.

  We obtained two confusion metrix from Logistic Regression and Decision Tree Classifier:

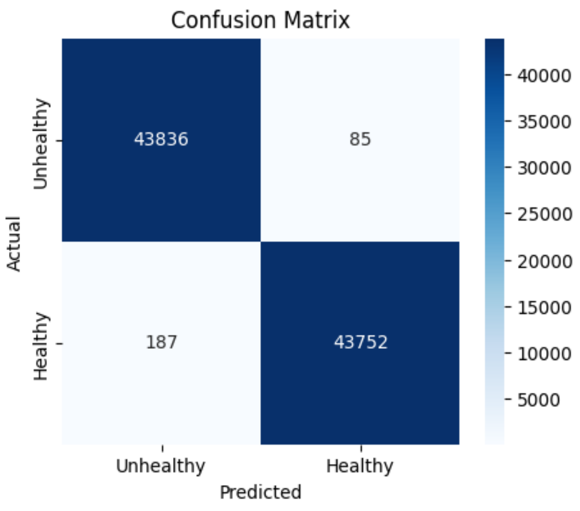


Figure: Logistic Regression

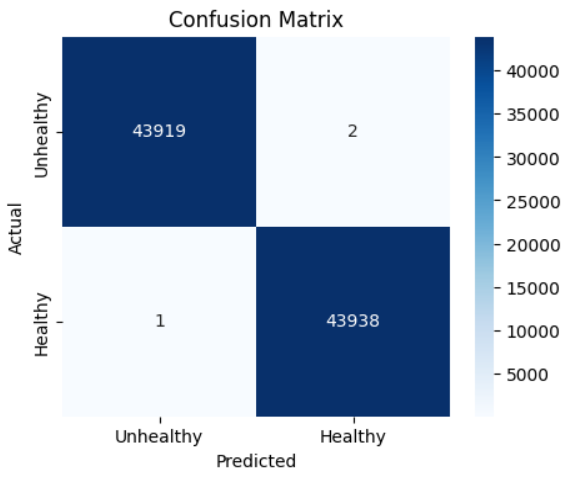


Figure: Decision Tree Classifier

- **Accuracy:** Accuracy is the proportion of correct predictions (both healthy and unhealthy) to the total number of predictions. It provides a general overview of how well the model performed. Higher accuracy indicates that the model made more correct predictions overall.

$$Accuracy = \frac{correct\ predictions}{total\ predictions} = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Precision:** Precision measures the proportion of true positives (correctly predicted healthy items) out of all the predicted positives. It answers the question, "Of all the items predicted as healthy, how many were actually healthy?" Precision is important when the cost of false positives (predicting unhealthy items as healthy) is high.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:** Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of actual positives (healthy items) that were correctly predicted. It answers the question, "Of all the actual healthy items, how many were correctly identified as healthy?" Recall is critical when false negatives (predicting healthy items as unhealthy) are costly.

$$Recall = \frac{TP}{TP + FN}$$

- **F1 Score:** The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall and is useful when you need a single metric to compare models, especially when there is an imbalance between the classes. A higher F1 score indicates a better balance between precision and recall.

$$F1\ score = 2 \times \frac{Precision\ \times Recall}{Precision + Recall}$$

Model Performance during Model Training and Evaluation(Before Hyperparameter Tuning)

Metric	Logistic Regression	Decision Tree Classifier
Model Accuracy	0.9918	0.9999
Precision	0.9936	1.0000
Recall	0.9899	0.9999
F1- Score	0.9917	0.9999

In our project, the classification models (Logistic Regression and Decision Tree Classifier) were evaluated using these metrics to determine their effectiveness in classifying food items as healthy or unhealthy. The confusion matrix was visualized to understand the number of correct and incorrect predictions, and the other metrics provided a detailed view of each model's performance, helping us compare their strengths and weaknesses.

4. Hyperparameter Tuning:

4.1 Tuning Methods:

Hyperparameter tuning is the process of finding the optimal set of hyperparameters for a machine learning model to maximize its performance. These hyperparameters control the learning process and directly impact the model's ability to generalize well to unseen data.

Without proper hyperparameter tuning,models might underfit or overfit the dataset. Proper tuning helps balance the tradeoff between underfitting and overfitting by adjusting model complexity. Effective tuning often results in significant performance gains, sometimes even more than using a more complex model. Well-tuned models generalize better to unseen data, making them more reliable in real-world applications.

To improve model performance, **Grid Search Cross-Validation (GridSearchCV)** was used for hyperparameter tuning of both the Logistic Regression and Decision Tree Classifier models. Grid Search systematically tests different combinations of hyperparameters and selects the best configuration based on cross-validation performance.

- **Logistic Regression**:
    - **Tuned Parameters**:
        - C: [0.01, 0.1, 1, 10, 100]
        - penalty: ['l2']
        - solver: ['liblinear']
        - max_iter: [500]
    - **Other Settings**: class_weight='balanced'
    - **Tuning Method**: GridSearchCV with 5-fold cross-validation and accuracy as the scoring metric
- **Decision Tree Classifier**:
    - **Tuned Parameters**:
        - criterion: ['gini', 'entropy']
        - max_depth: [None, 10, 20, 30]
        - min_samples_split: [10,20]
        - min_samples_leaf: [5,10]
    - **Tuning Method**: GridSearchCV with 5-fold cross-validation and accuracy as the scoring metric
    - **Preprocessing**: All categorical features were encoded using Label Encoding

The results obtained for both logistic regression model and decision tree classifier model are represented below:

Model Performance during Model Training and Evaluation(After Hyperparameter Tuning)

Metric	Logistic Regression	Decision Tree Classifier
Best Parameters	{'C': 100, 'max_iter': 500, 'penalty': 'l2', 'solver': 'liblinear'}	{'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 5, 'min_samples_split': 20}
Best CV Accuracy	0.9989	0.9999
Test Accuracy	0.9990	0.9999
Test Precision	0.9993	1.0000
Test Recall	0.9987	0.9999
Test F1 Score	0.9990	0.9999

As the results of decision tree model was giving almost perfect scores suggesting overfitting of the model, we performed cross-validation of

4.2 Model Performance Comparison:

After applying **Grid Search Cross-Validation (GridSearchCV)** to tune the hyperparameters for both the Logistic Regression and Decision Tree models, we evaluated their performance on the **unseen test dataset**. Below are the results:

Cross-Validation Results for Logistic Regression Model and Decision Tree Classifier Model

	Logistic Regression	**Decision Tree Classifier**
CV Accuracy	0.9990 ± 0.0001	0.9999 ± 0.0000

Key Observations:

1. **Both Models Perform Exceptionally Well:**

   The cross-validation accuracies of both model is high suggesting both are fitting the training data effectively. Logistic Regression has an accuracy of 99.90% while Decision tree Classifier has a slightly higher accuracy of 99.99%.

2. **Stability of Performance:**

   The standard deviation for Logistic regression is ±0.0001 which means there is very little variation in accuracy across the folds. This indicates a stable performance. For Decision Tree, the standard deviation is ± 0.0000 which shows perfect consistency in cross-validation folds. This suggests the model may be perfectly fitting each an every data points on all folds indicating **overfitting.**

3. **Interpretation:** Decision Tree is overfitting the data.

We represented these data with the aid of visualization techniques.A learning curve was plotted for the logistic regression whereas a validation curve was plotted for the decision tree.

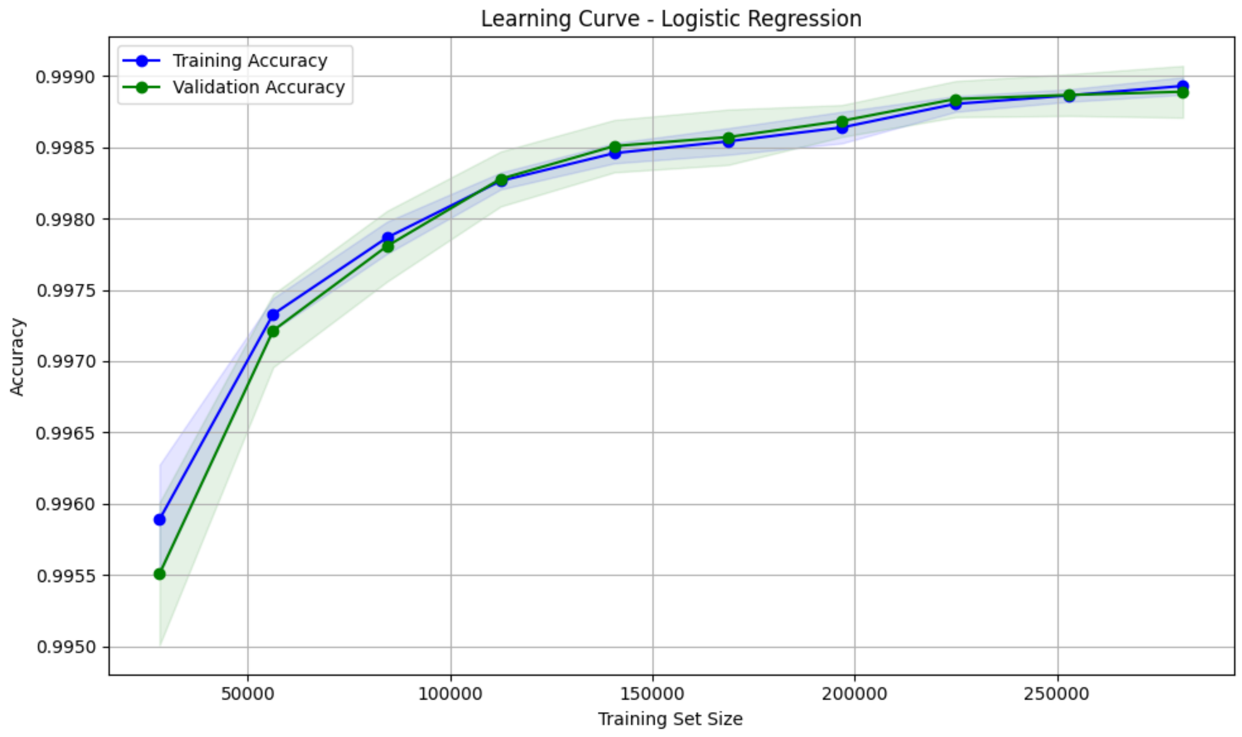


Figure: Learning Curve for Logistic Regression Model

We can observe from the learning curve that the training accuracy and the validation accuracy is close to eachother as the rate of change of accuracy between both the curves is minimal. This shows our hyperparameter tuning and cross-validation for logistic regression is working properly. There is no sign of overfitting or underfitting.

Key Observations:

- **High Accuracy:** Both curves are close to 99.9%
- **Convergence of Training and Validation Accuracy:** The blue and green line converge as training data increases which shows model is generalizing well and is not overfitting.
- **Small Gap between Two Curves:** This shows low variance.
- **Accuracy Increases with More Data:** This shows adding more data can be beneficial.

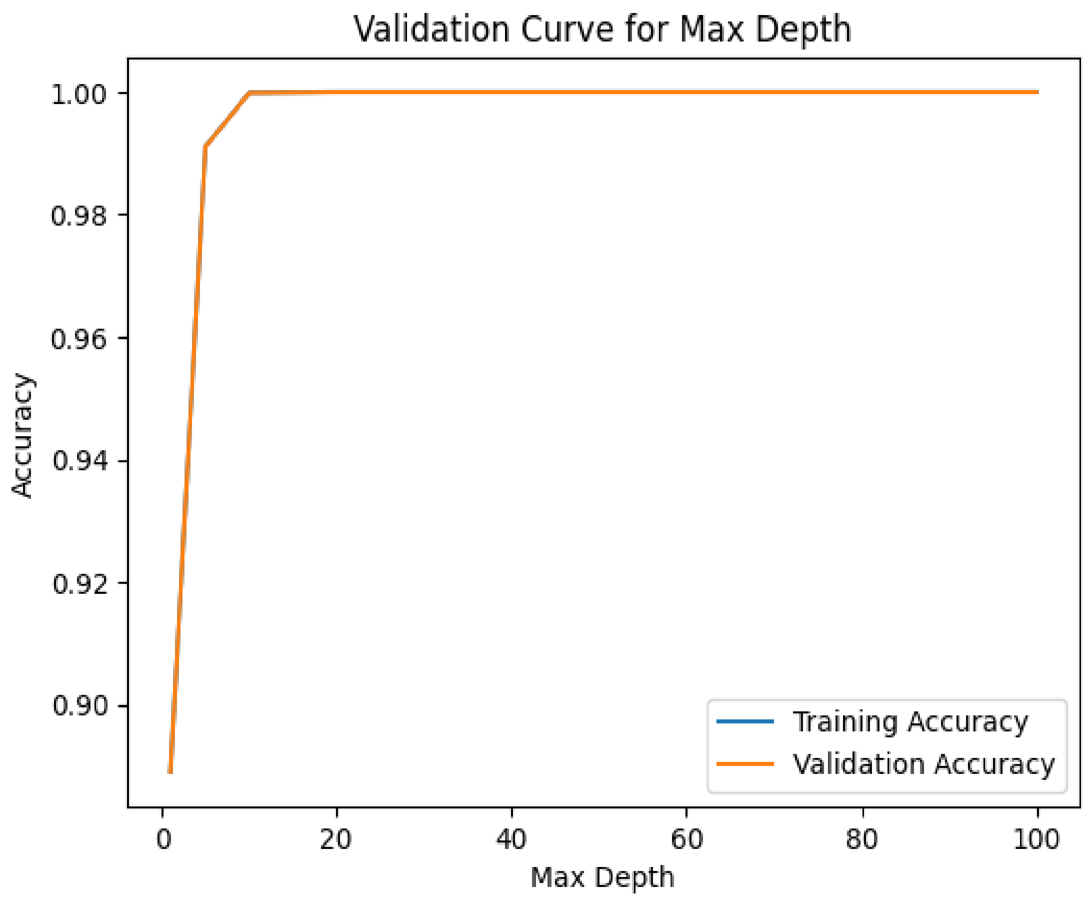


Figure: Validation Curve for max_depth in Decision Tree Model

We can observe from the validation curve that both the training accuracy and validation accuracy are exactly same which suggests **overfitting** of the data. Overfitting in a model is highly discouraged .

Key Observations:

- **Very High Accuracy:** Both training accuracy and validation accuracy are almost at 1.0.This shows model is working well on both training and validation data.
- **Low Bias(No Underfitting):** At low depth, accuracy is low but as depth increases, accuracy improves rapidly.
- **High Variance Risk(Overfitting):** Validation accuracy remains flat at 1.0 which suggests potential overfitting.

5. Final Comparison and Conclusion:

5.1 Model Comparison:

In this project, we implemented two classification models: **Logistic Regression** and **Decision Tree Classifier** based on our selected dataset to predict whether a food item is healthy or unhealthy. Both models were evaluated using standard classification metrics including **accuracy**, **precision**, **recall**, and **F1-score**.

- **Logistic Regression** showed consistent performance with a balanced trade-off between precision and recall. It generalized well on unseen data, making it a reliable baseline model.
- **Decision Tree Classifier**, although effective at capturing complex patterns in the dataset, initially exhibited overfitting on the training data. However, after a second round of hyperparameter tuning, the model showed a slight improvement in reducing overfitting.

After performing hyperparameter tuning using GridSearchCV, the Decision Tree model achieved almost perfect results compared to the Logistic Regression model which indicated overfitting.

The Final Performance Results for Logistic Regression Model and Decision Tree Model:

Metric	Logistic Regression	Decision Tree Classifier
Best Parameters	{'C': 100, 'max_iter': 500, 'penalty': 'l2', 'solver': 'liblinear'}	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 15, 'min_samples_leaf': 5, 'min_samples_split': 10}
Best CV Accuracy	0.9989	0.9999
Test Accuracy	0.9990	0.9999
Test Precision	0.9993	1.0000
Test Recall	0.9987	0.9998
Test F1 Score	0.9990	0.9999

5.2 Key Insights:

The overall workflow of the project revealed several important insights at each stage—data preprocessing, exploratory data analysis (EDA), model training, and hyperparameter tuning.

- **Data Preprocessing:** Significant attention was given to handling missing values to retain as much useful data as possible. Columns with more than 50% missing data were dropped, while numerical values were imputed with their mean and categorical ones with the mode. Outliers were effectively managed using the Interquartile Range (IQR) method, which significantly reduced data noise and improved overall model performance.

- **Feature Engineering and Scaling:** The creation of the binary target variable ‘healthy’ based on the health score allowed for effective classification. One-hot encoding and normalization helped standardize the dataset, ensuring the machine learning models performed optimally without bias due to scale or categorical format.
- **Exploratory Data Analysis (EDA):** Through visualizations and statistical summaries, we gained a deep understanding of feature distributions, correlations, and the dataset’s overall structure. This informed our model selection and tuning strategies.
- **Model Training and Evaluation:** Both Logistic Regression and Decision Tree classifiers were used to model the binary classification task. Evaluation through accuracy, precision, recall, and F1-score revealed that both models performed well, but the Decision Tree model showed overfitting nature of the model.
- **Hyperparameter Tuning:** Grid Search was used to optimize the models, resulting in improved evaluation scores. Tuning helped avoid overfitting and ensured that the final models were well-generalized on unseen data.

Logistic Regression performed slightly better than Decision Tree Classifier across all evaluation metrics.While Decision Tree is more interpretable and works well with non-linear relationships, Logistic Regression showed more promising results on this dataset.

Decision tree showed perfect consistency in the hyperparameter tuning results suggesting overfitting of data. Models that overfit the data are generally discouraged, as they perform well on memorized training data but fail to generalize to unseen data.

As our goal is to train a model which works well on unseen data, we can conclude that logistic regression model is more preferrable in this case.

5.3 Final Recommendation:

The evaluation metrics indicate that the Decision Tree model tends to overfit the training data, performing well on known inputs but struggling to generalize to unseen examples. In contrast, Logistic Regression shows better overall performance, is computationally efficient, and handles unseen data more effectively.

Considering these factors, **Logistic Regression** is recommended as the preferred model for this classification task**.**

The full incremental progress of each of the group members can be found in our github repository:

LINK: https://github.com/XavierRolex/Food-Classification-Model.git

Contribution Summary:

Name(Id)	**Contribution**
1. Suhana Islam (2131550642) & Nafis Anzum(2211554042)	• Handled complete data preprocessing—missing value treatment, outlier removal using IQR, feature encoding, and scaling. • Conducted comprehensive exploratory data analysis (EDA) with scatterplots, boxplots,histograms and correlation heatmaps. • Implemented and evaluated both Logistic Regression and Decision Tree Classifier models using appropriate classification metrics. • Performed hyperparameter tuning via GridSearchCV and documented the performance improvements. • Concluded the most effective model • Documentated the **final PDF** with results, plots, and insights. • Also managed GitHub version control and coordinated code contributions.
2. Tasfia Anjum Zuairia(2221233642)	• Handled cleaning and preparing the data including fixing missing values • Created initial scatter plots for preliminary data analysis. • Continued working on code and attempted to address overfitting, although final implementations were not included in the final submission. • Took Feedback from teammates during preprocessing stage.
3. Ekfat Jahan Ashrafy(2132236642)	Unable to participate due to family health reasons (as communicated via email)