Statistical Learning Project

Samuele Samonini

May 2025

Obesity

1 Introduction

1.1 Project objective

This statistical learning project analyzes a dataset about obesity. The main goal is to use different machine learning techniques to find useful insights and build predictive models. The project is divided into two parts: *unsupervised learning* and *supervised learning*.

In the unsupervised learning section, the aim is to discover hidden patterns in the data without using labeled outcomes. The analysis starts with an exploratory data analysis (EDA) to explore the distribution and relationships between variables. Then, principal component analysis (PCA) is used to reduce the number of variables and highlight the most important ones. Finally, K-means clustering is applied to group similar observations, helping to find possible subgroups in the dataset.

The supervised learning section focuses on prediction using labeled data. Like the unsupervised part, it starts with EDA to get the data ready. Then, several classification models are tested, such as support vector machines (SVM), classification and regression trees (CART), random forest (RF), multinomial logistic regression, gradient boosting, and neural networks.

The models are evaluated using different metrics, including accuracy, Cohen's kappa, root mean squared error (RMSE), F1 score, area under the curve (AUC), and receiver operating characteristic (ROC) curves. The model with the best performance is chosen for final interpretation.

This two-step approach offers a complete way to understand the data, combining exploratory analysis with powerful predictive modeling.

1.2 Dataset Description and Variables Overview

The dataset used in this project aims to estimate obesity levels based on people's eating habits and physical condition. It includes data from 2,111 individuals aged 14 to 61, coming from Colombia, Peru, and Mexico. At first, 485 records were collected through an online survey open for 30 days. Participants gave anonymous answers about their lifestyle, diet, and physical activity. Using this information, each person's Body Mass Index (BMI) was calculated with the following formula:

$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height (m)}^2} \tag{1}$$

BMI values were then used to assign categories following WHO and Mexican guidelines:

- *Underweight*: BMI < 18.5
- *Normal weight*: $18.5 \leq$ BMI < 24.9
- *Overweight*: $25.0 \leq$ BMI < 29.9
- *Obesity Type I*: $30.0 \leq$ BMI < 34.9
- *Obesity Type II*: $35.0 \leq$ BMI < 39.9
- *Obesity Type III*: BMI ≥ 40

Since the original data was imbalanced, 77% of the final dataset was created synthetically using the SMOTE algorithm in Weka, resulting in a balanced dataset of 2,111 records. All entries are complete and have been cleaned to remove outliers and normalized for analysis.

This dataset is well-suited for different statistical learning tasks like classification, clustering, and pattern recognition. It includes 17 input variables and one target variable. Below is a summary of all variables:

- **Gender (Categorical)** – Biological gender of the person.
- **Age (Continuous)** – Age in years.
- **Height (Continuous)** – Height in meters.
- **Weight (Continuous)** – Weight in kilograms.
- **family_history_with_overweight (Binary)** – Indicates if the person has family members who are or were overweight.
- **FAVC (Binary)** – Frequent intake of high-calorie food.
- **FCVC (Integer)** – Frequency of eating vegetables in meals.
- **NCP (Continuous)** – Number of main meals per day.

- **CAEC (Categorical)** – Frequency of eating between meals.
- **SMOKE (Binary)** – Indicates if the person smokes.
- **CH2O (Continuous)** – Daily water intake.
- **SCC (Binary)** – Whether the person tracks their daily calories.
- **FAF (Continuous)** – Frequency of physical activity.
- **TUE (Integer)** – Time spent on electronic devices (e.g., TV, phone, computer).
- **CALC (Categorical)** – Frequency of alcohol consumption.
- **MTRANS (Categorical)** – Main type of transportation used.
- **NObeyesdad (Categorical – Target)** – Obesity level (7 classes from underweight to obesity type III).

All variables are ready for analysis and provide a strong, multidimensional view of obesity prediction.

2 Unsupervised Learning

2.1 EDA

In this section, unsupervised learning methods are used to explore the structure of the data without using predefined class labels. The main goal is to find hidden patterns, similarities, or differences among individuals based on their lifestyle, eating habits, and physical condition. This helps us better understand how the data is distributed and spot possible groups of similar individuals.

The analysis is divided into three steps: an initial exploratory data analysis (EDA), dimensionality reduction using principal component analysis (PCA), and clustering with the K-means algorithm.

As a first step, we check the structure of the dataset to get a general overview. This is done using the following R command:

```
head(data)
```

This command shows the first few rows of the dataset, giving a quick view of the variables and their values. It also helps confirm that the data has been loaded correctly and is ready for analysis.

head(data)

Gender	Age	Height	Weight	family_history	FAVC	FCVC	NCP	CAEC
Female	21	1.62	64.0	yes	no	2	3	Sometimes
Female	21	1.52	56.0	yes	no	3	3	Sometimes
Male	23	1.80	77.0	yes	no	2	3	Sometimes
Male	27	1.80	87.0	no	no	3	3	Sometimes
Male	22	1.78	89.8	no	no	2	1	Sometimes
Male	29	1.62	53.0	no	yes	2	3	Sometimes

SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad
no	2	no	0	1	no	Public_Transportation	Normal_Weight
yes	3	yes	3	0	Sometimes	Public_Transportation	Normal_Weight
no	2	no	2	1	Frequently	Public_Transportation	Normal_Weight
no	2	no	2	0	Frequently	Walking	Overweight_Level_I
no	2	no	0	0	Sometimes	Public_Transportation	Overweight_Level_II
no	2	no	0	0	Sometimes	Automobile	Normal_Weight

To analyze the distribution of the numerical variables, we use histograms. These plots give a visual summary of how values are spread across the dataset and help identify key features like skewness, outliers, and overall variability. This is an important step to decide if further data transformations are needed and to guide the interpretation of the next phases.

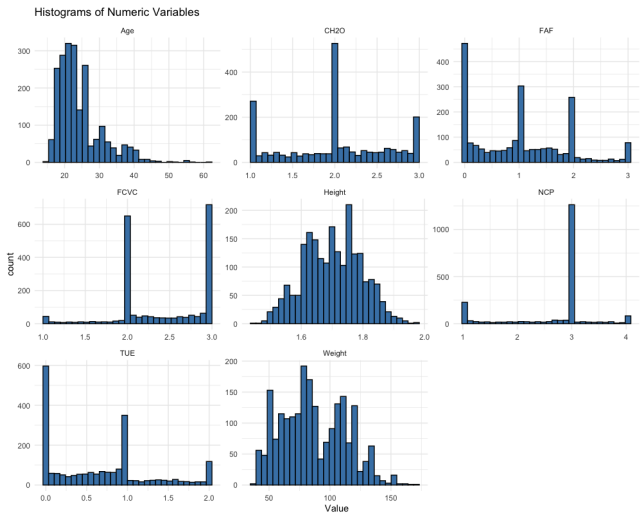

Figure 1: Histogram of numerical variables in the dataset.

To better understand the categorical variables, we use bar plots showing the frequency of each category. These plots make it easier to see class imbalances and dominant groups. This is useful for analyzing lifestyle and demographic factors and supports choosing the right modeling strategies later.

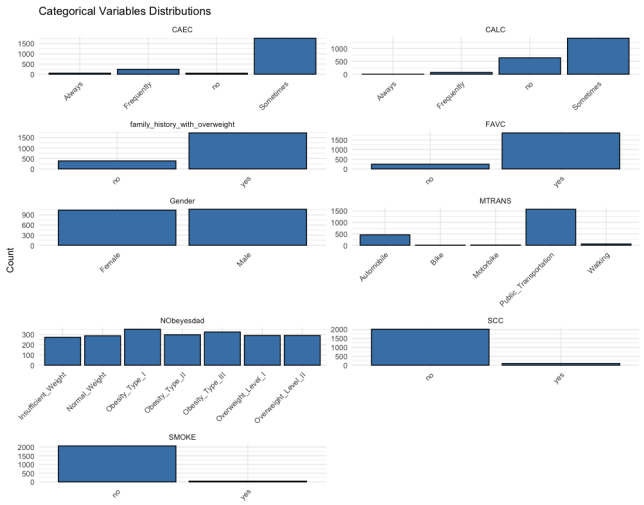

Figure 2: Histogram of numerical variables in the dataset.

We also include a correlation heatmap to explore relationships between variables. This plot shows pairwise correlation coefficients, from -1 (strong nega-

tive) to +1 (strong positive).

The color scale indicates the strength and direction of correlation:

- Dark blue shows a strong positive correlation.
- Dark red shows a strong negative correlation.
- Light colors mean weak or no correlation.

The heatmap helps identify multicollinearity, redundant features, and meaningful relationships. For example, one-hot encoded categorical variables like `CALC`, `CAEC`, or `MTRANS` show perfect negative correlations among levels. Also, numerical variables like `Height`, `Weight`, and `Age` show moderate positive correlations, as expected.

This step is important to understand how variables are connected and supports future steps like dimensionality reduction.

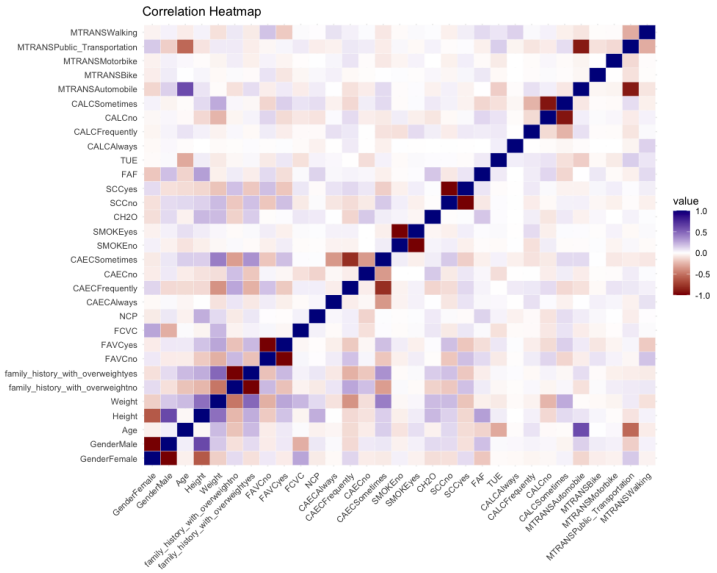

Figure 3: Correlation heatmap of the dataset.

2.2 Principal Component Analysis (PCA)

To reduce the number of variables while keeping most of the information, we apply Principal Component Analysis (PCA). This method transforms the original variables into new ones called principal components, which are linear combinations of the original features and are uncorrelated. The first few components usually explain most of the variation in the data, making it easier to visualize and work with.

PCA is especially helpful in unsupervised learning to uncover patterns and relationships that are not obvious in higher dimensions.

2.2.1 Scree Plot – Explained Variance

The first plot (Figure 4) is a scree plot, which shows how much variance each principal component explains. The first component accounts for 13.6% of the total variance, and the second one explains 9%. The rest explain smaller portions. This means that the first few components keep a good amount of the information from the original data.

The sharp drop after the first components suggests that using only 2 or 3 dimensions could still give a useful summary of the dataset, while keeping it easy to interpret.

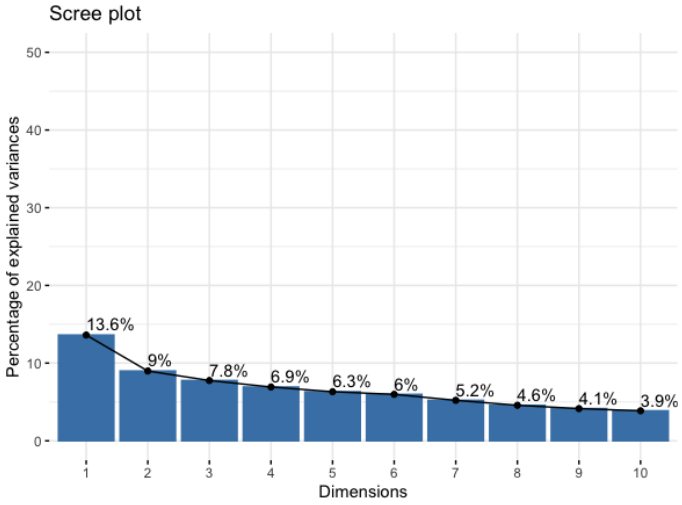

Figure 4: Scree plot showing the explained variance by each principal component.

2.2.2 Variable Contribution – PCA Biplot

The second plot (Figure 5) is a variable contribution plot, also known as a biplot. It shows how the original variables influence the first two principal components (Dim1 and Dim2). Each arrow represents the direction and strength of a variable's contribution.

Variables like `Weight`, `Height`, and `family_history_with_overweight` contribute strongly to the first component (Dim1). Variables related to gender and transportation influence the second component (Dim2).

The color scale indicates the strength of the contribution: warmer colors (red/orange) mean stronger influence.

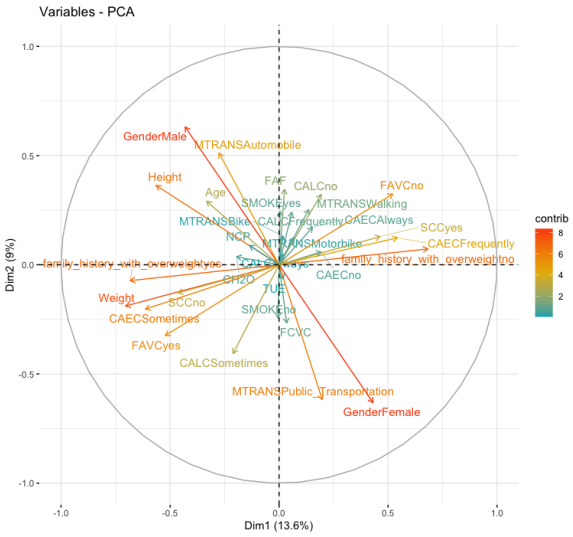

Figure 5: PCA biplot showing variable contributions to the first two principal components.

This plot helps understand which variables are most important in shaping the main patterns in the data and gives insight into possible correlations and groupings.

2.2.3 Individuals – PCA Colored by Gender

The third plot (Figure 6) shows individuals placed in the space of the first two principal components, colored and shaped by gender: blue circles for females and yellow triangles for males. The ellipses show confidence intervals for the two gender groups.

There is a clear separation between males and females along the vertical axis (Dim2), meaning gender-related variables are important drivers of variability in the dataset.

Even though there is some overlap, the group separation is clear. This shows that PCA has effectively captured underlying structure related to gender.

This plot shows how PCA can be useful not only for reducing dimensions but also for revealing group differences and patterns in the data.

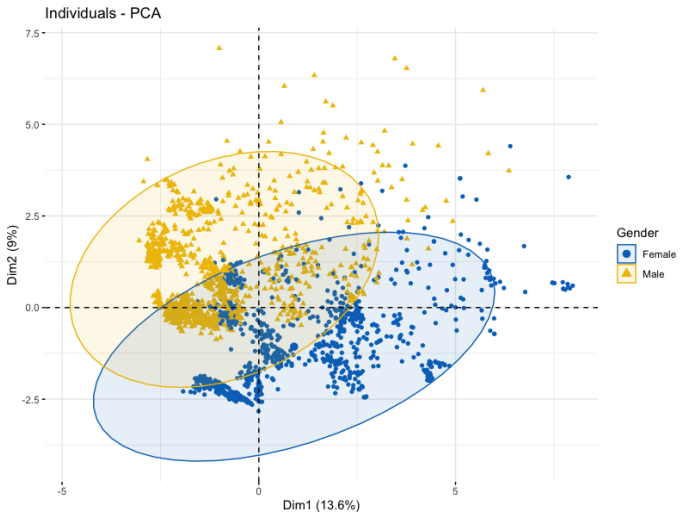

Figure 6: PCA plot of individuals colored by gender.

2.3 Clustering and K-means Analysis

After reducing the number of variables with PCA, we perform clustering to find groups of individuals with similar traits. Clustering is an unsupervised learning method that groups observations based on similarity, without using predefined labels. Among various algorithms, K-means is widely used because it is simple and effective. It aims to divide the data into k clusters by minimizing the variation within each cluster.

An important step in K-means is choosing the best number of clusters (k). Too few or too many can lead to poor results.

2.3.1 Silhouette Method for Optimal k

To choose the best k, we use the Silhouette Method. This method measures how well each point fits in its own cluster compared to others. The silhouette score ranges from -1 to 1:

- Scores near 1 mean the point fits well in its cluster.
- Scores around 0 suggest overlapping clusters.
- Negative scores may mean poor clustering.

We compute the average silhouette score for different k values. As shown below, the score is highest at $k = 7$, suggesting this is the best choice.

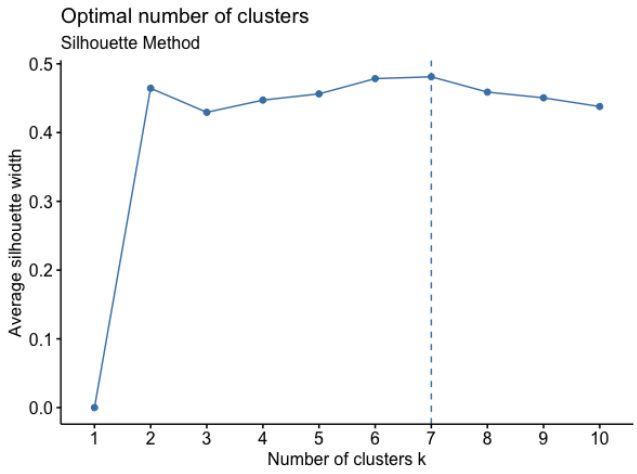

Figure 7: Silhouette score by number of clusters (k). Peak value observed at $k = 7$.

Based on this result, we set $k = 7$ for the K-means analysis.

2.3.2 K-means Clustering Results

The next plot shows the K-means results with $k = 7$, projected onto the first two principal components (PC1 and PC2) from PCA. Each point is an individual, colored by cluster. Polygons outline the space covered by each cluster.

Most clusters are clearly separated, meaning the algorithm grouped the data well. Some clusters (e.g., 1 and 6) are compact and distinct, while others (e.g., 3 and 5) overlap more, which may mean they share some patterns.

This confirms that PCA helped preserve useful structure in the data, making the clusters easier to see and interpret.

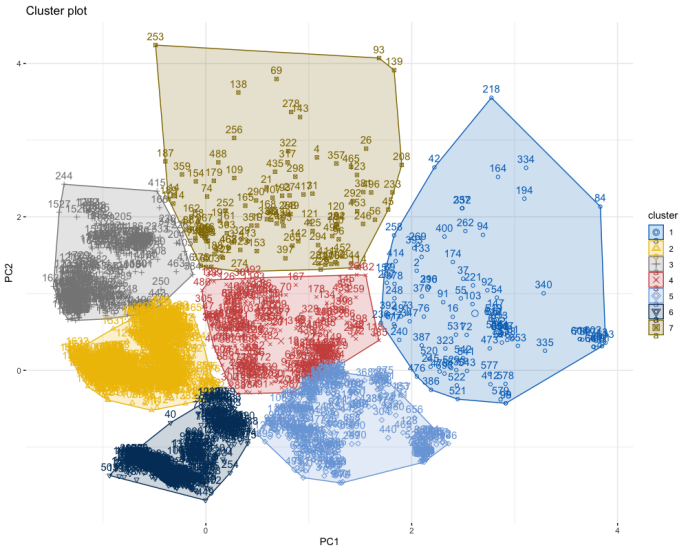

Figure 8: K-means clustering results ($k = 7$) projected onto the first two principal components.

We also include a 3D version of the plot, which offers another view of the group separation using three principal components.

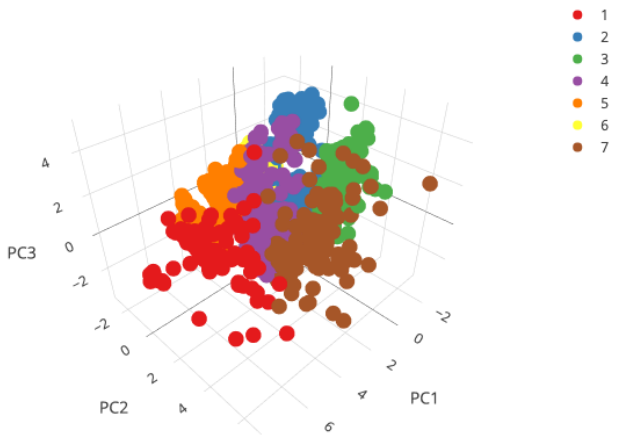

Figure 9: 3D visualization of K-means clusters using the first three principal components.

2.3.3 Cluster Profile Analysis

To understand the characteristics of each cluster, we calculate the average values of all numerical variables for each group.

We first group the data by cluster and compute the mean of each numeric variable. Then, to compare variables with different scales, we standardize the values (z-scores):

- Positive values (blue) mean the cluster is above the overall average.
- Negative values (red) mean the cluster is below average.

We reshape the data and create a heatmap to visualize the standardized values across clusters and variables.

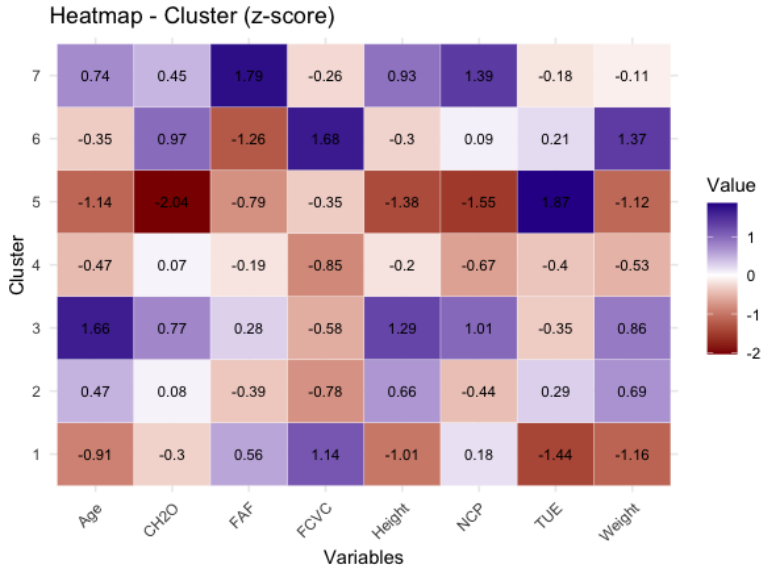

Figure 10: Heatmap of standardized mean values (z-scores) for numerical variables across clusters.

This heatmap lets us quickly compare clusters. For example, cluster 5 has low water intake (`CH2O`) and fewer meals (`NCP`), while cluster 7 scores high in physical activity (`FAF`) and number of meals.

This analysis gives useful insights into how behaviors and characteristics differ across groups.

2.3.4 Age Distribution by Cluster

To explore demographic differences, we analyze how `Age` is distributed in each cluster. The plot below shows density curves by cluster.

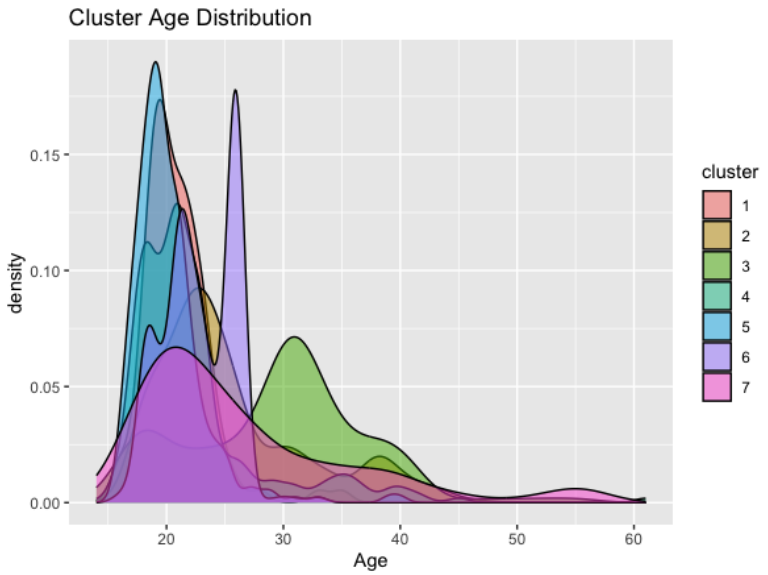

Figure 11: Age distribution by cluster. Density curves highlight differences in age profiles among clusters.

From the plot, we can see:

- Clusters 4 and 6 mostly include younger individuals (around 18–20 years old).
- Cluster 3 includes more people aged 30–40.
- Cluster 7 has a wide age range.
- Clusters 1 and 2 have smaller populations but similar age ranges.

These age patterns help explain the behavioral profiles found earlier and may support targeted health strategies or recommendations.

2.4 Conclusion – Unsupervised Learning

The unsupervised learning section allowed us to explore the obesity dataset without using the target variable. We started by examining the structure of the data and visualizing the distribution of numerical and categorical variables through histograms and bar plots. These initial steps helped us better understand the dataset.

Next, we used Principal Component Analysis (PCA) to reduce the number of variables while keeping most of the information. The scree plot helped choose the number of components, the biplot showed the most important variables, and

the projection of individuals revealed a clear separation by gender. A 3D version gave an even deeper view of the group structure.

To find hidden patterns in the data, we applied K-means clustering. Using the Silhouette Method, we found that $k = 7$ was the best number of clusters. The cluster plot based on the first two PCA components confirmed good separation between groups.

We also looked inside each cluster:

- A heatmap of z-scores showed how each cluster differs from the average in variables like age, water intake, and physical activity.
- A density plot of age by cluster showed different age profiles, with some groups mostly young people and others with a wider range.

Overall, the results show that we can group individuals meaningfully based on behavior and demographics alone. These insights will be useful in the next phase of the project, where we will use supervised learning to predict obesity levels. The clusters and key variables found here may help guide feature selection and model building.

3 Supervised Learning

After exploring the structure of the dataset with unsupervised methods, we now move to supervised learning. This part focuses on building models that can predict the obesity level (`NObeyesdad`), which has seven categories based on BMI.

The goal is to train models that can learn from the input features and accurately predict the obesity class of new individuals. This has practical value for early detection, personalized health advice, and risk assessment based on habits and physical condition.

We use the `caret` package in R to manage the modeling process. `caret` provides a unified way to train and compare models, handle preprocessing, tune hyperparameters with cross-validation, and evaluate performance using various metrics.

In this section, we will:

- Prepare the dataset for training and testing;
- Apply classification models (SVM, CART, Random Forest, Logistic Regression, Gradient Boosting, Neural Network);
- Compare performance using Accuracy, F1 Score, Kappa, RMSE, AUC, and ROC curves;
- Choose the best model for predicting obesity level.

This phase builds on the insights from the unsupervised analysis and aims to create a clear and reliable prediction system.

3.1 Data Preparation and Exploratory Analysis

First, we load all the required libraries for modeling, visualization, and evaluation. These include `caret`, `e1071`, `randomForest`, `nnet`, `xgboost`, `gbm`, and tools like `tidyverse`, `ggplot2`, `reshape2`, `MLmetrics`, `pROC`, and `ROCR`.

The dataset is read into the object `obesity_df`. The target variable `NObeyesdad`, which indicates the obesity level, is converted into an ordered factor from *Insufficient Weight* to *Obesity Type III*. This helps models treat the outcome in a structured way.

All categorical predictors are transformed into dummy variables using `dummyVars()` from `caret`, so that models can work with numeric inputs. These transformed features are then combined with the target variable.

The data is split into training (60%) and test (40%) sets using `createDataPartition()`, making sure the class proportions remain balanced in both sets.

Next, we remove any variables with zero variance, as they don't provide useful information for prediction.

Finally, we run an initial EDA to check the structure of the data and how both predictors and target variable are distributed.

Graphical Exploration

We begin by plotting a bar chart of the target variable to see the distribution of obesity levels. This helps spot class imbalance, which can affect model training.

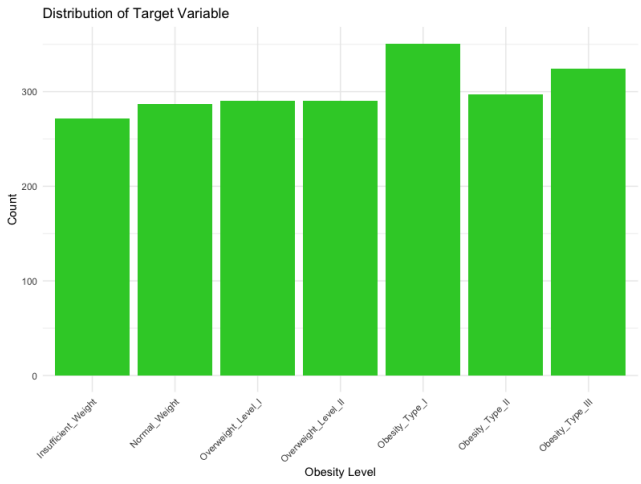

Figure 12: Distribution of obesity levels in the dataset.

Then, we use facet bar plots to explore how obesity levels are distributed across different categorical features (e.g., smoking, family history, transportation). Each subplot shows how classes vary within a category, helping to detect strong relationships.

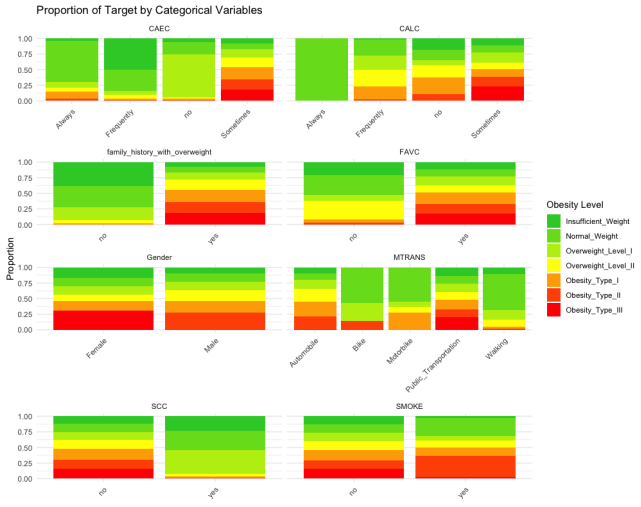

Figure 13: Obesity levels across selected categorical variables.

Finally, we use boxplots to examine the numeric features. These plots help detect outliers and understand variable distributions. Variables like `Age`, `Height`, `Weight`, `NCP`, `FAF`, and `CH2O` are included. If any variables are skewed or have extreme values, they may need transformation.

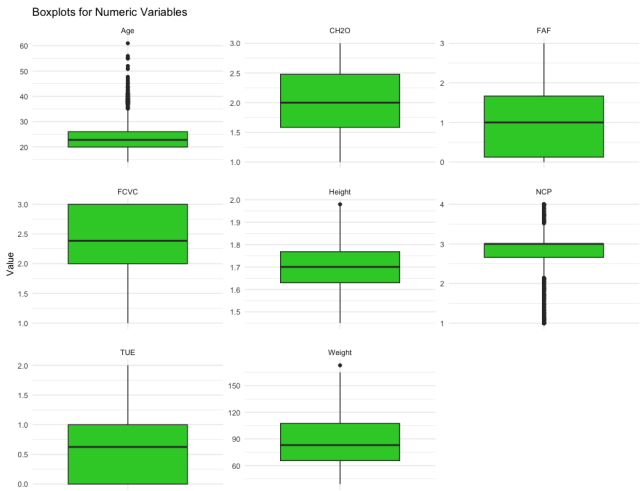

Figure 14: Boxplots of numeric predictor variables.

This initial step ensures that the data is clean and ready for modeling, while also revealing important trends and relationships that may impact prediction.

3.2 Support Vector Machine (Radial Kernel)

The first supervised model used is a Support Vector Machine (SVM) with a radial basis function (RBF) kernel. SVM is a powerful algorithm that finds the best boundary to separate classes in a high-dimensional space. The radial kernel is especially good at handling non-linear patterns, making it suitable for this multiclass obesity classification.

The model was trained using the `caret` package. Predictions were made on the test set and evaluated using a confusion matrix and performance metrics.

3.2.1 Confusion Matrix – SVM

The confusion matrix shows how well the model predicted each class compared to the actual values. Key findings:

- The model classifies *Insufficient Weight* well, with 95 correct predictions, 87.96% sensitivity, and 92.41% balanced accuracy.
- *Normal Weight* and *Overweight Level I* are also predicted reasonably well, though there is some confusion with similar categories.

- The model struggles with *Overweight Level II* and *Obesity Type II*, with sensitivities below 10%. These are often predicted as less severe cases.
- *Obesity Type III* is better handled, with 84 correct predictions, 65.12% sensitivity, and 100% precision (no false positives).

Overall accuracy is 54.34%, with a Kappa statistic of 0.4646, indicating moderate agreement. Since the No Information Rate is only 16.65%, the model performs much better than random guessing.

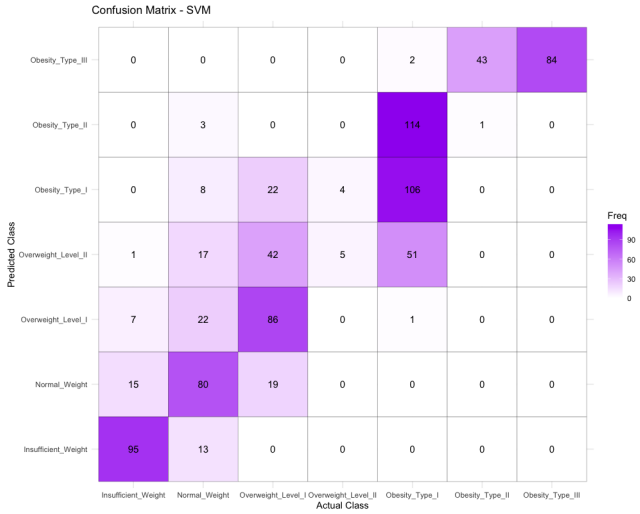

Figure 15: Confusion matrix for the SVM (RBF kernel) classifier.

3.2.2 Variable Importance – SVM

To see which features influenced the model most, we used variable importance from `caret`. The plot shows:

- `Weight` and `Age` are the most important features.
- `family_history_with_overweight`, `CAEC` (eating between meals), and `FCVC` (vegetable intake) are also key.
- Other useful variables include `Gender`, `FAF` (physical activity), and `CH2O` (water intake).
- Less important features are `MTRANS` (transportation), `SMOKE`, and some dietary habits.

This confirms that obesity levels depend mostly on biometric and eating behavior variables, with lifestyle playing a secondary role.

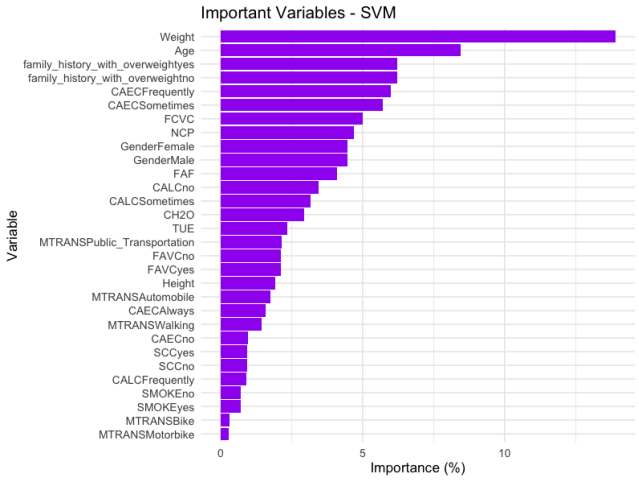

Figure 16: Variable importance plot for the SVM model.

Table 1: Overall performance metrics – SVM (RBF kernel)

Metric	**Value**
Accuracy	54.34%
95% Confidence Interval	[50.90%, 57.75%]
Kappa	0.4646
No Information Rate (NIR)	16.65%
P-value [Accuracy ¿ NIR]	$< 2.2 \times 10^{-16}$

The model reaches 54.34% accuracy with a Kappa of 0.4646. Since the NIR is only 16.65%, the model performs much better than random guessing, and the difference is statistically significant ($p < 2.2 \times 10^{-16}$).

3.2.3 Final Remarks – SVM

The SVM with RBF kernel shows moderate performance. It performs well for extreme cases like *Insufficient Weight* and *Obesity Type III*, but struggles with middle categories like *Overweight Level II* and *Obesity Type II*.

It relies mainly on `Weight`, `Age`, and `family_history_with_overweight`, showing a good mix of biometric and behavioral features. While the model is a solid baseline, its uneven class performance suggests that improvements may come from better tuning or switching to ensemble methods.

3.3 CART – Classification and Regression Tree

The second model is a CART (Classification and Regression Tree), which splits the data step-by-step into groups using the most informative features. CART is easy to interpret because it shows how predictions are made through a tree structure.

Here, the model was trained on the same data as the SVM using `rpart` via `caret`. Predictions were made on the test set and evaluated with a confusion matrix.

However, while CART is interpretable, it often lacks accuracy in complex, high-dimensional tasks like this multiclass classification. We'll now see how its performance compares.

3.3.1 Confusion Matrix – CART

The confusion matrix compares predicted vs actual classes for the CART model. Key observations:

- The model performs very well for *Obesity Type III*, with 128 correct predictions and 99.22% sensitivity.
- It also classifies *Obesity Type II* and *Overweight Level II* with high sensitivity (99.15% and 94.83%).
- It fails completely to detect *Normal Weight*, *Overweight Level I*, and *Obesity Type I*, all with 0% sensitivity.
- For *Insufficient Weight*, sensitivity is high (97.22%) but precision is low (61.05%), meaning many false positives.

Table 2: Overall performance metrics – CART

Metric	**Value**
Accuracy	54.70%
95% Confidence Interval	[51.26%, 58.10%]
Kappa	0.4739
No Information Rate (NIR)	16.65%
P-value [Accuracy ¿ NIR]	$< 2.2 \times 10^{-16}$

The model reaches 54.70% accuracy and a Kappa of 0.4739, indicating moderate agreement. As with SVM, it performs much better than random guessing (NIR = 16.65%), with a highly significant p-value. However, the overall accuracy hides poor performance for many classes.

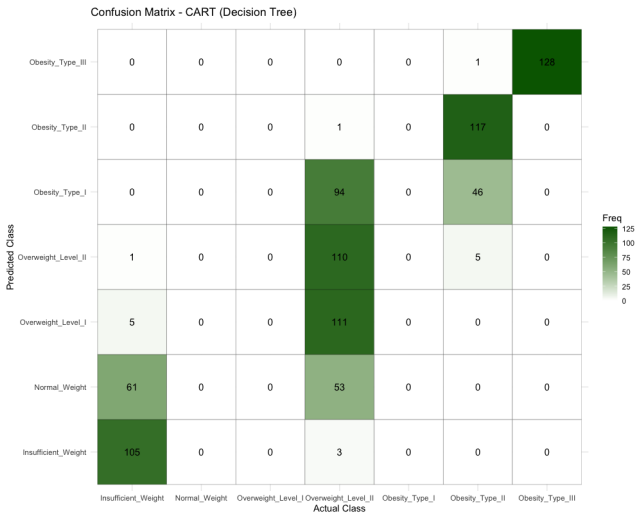

Figure 17: Confusion matrix for the CART classifier.

3.3.2 Variable Importance – CART

The importance plot shows that the CART model focuses heavily on just a few variables:

- `Weight` is by far the most important feature.
- `Gender`, `FCVC`, and `family_history_with_overweight` also contribute.
- `Age` and `Height` are less important.
- Other variables have very little influence.

This narrow focus may lead to underfitting, especially in a complex multi-class problem like this.

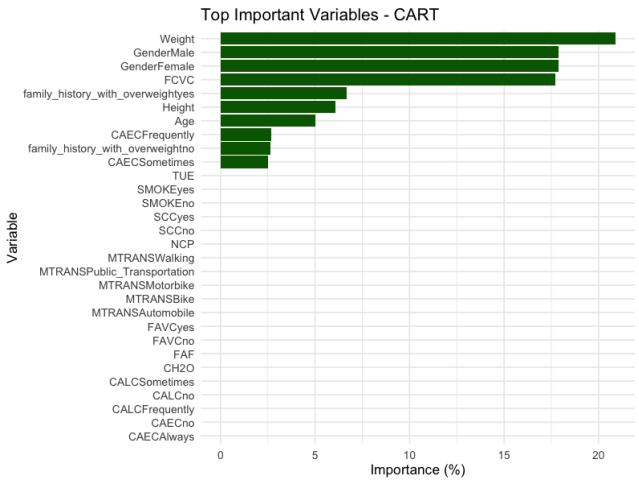

Figure 18: Variable importance plot for the CART model.

CART Decision Tree Visualization

The tree diagram confirms the model's simplicity. The first split is based on `Weight`, which drives most decisions:

- Weight ¡ 60 kg → *Insufficient Weight*
- Weight 60–100 kg → *Overweight Level II*
- Weight ¿ 100 kg → further split by `Gender`, leading to *Obesity Type II* or *Obesity Type III*

Classes like *Normal Weight* and *Overweight Level I* are not represented in the tree, consistent with their 0% sensitivity.

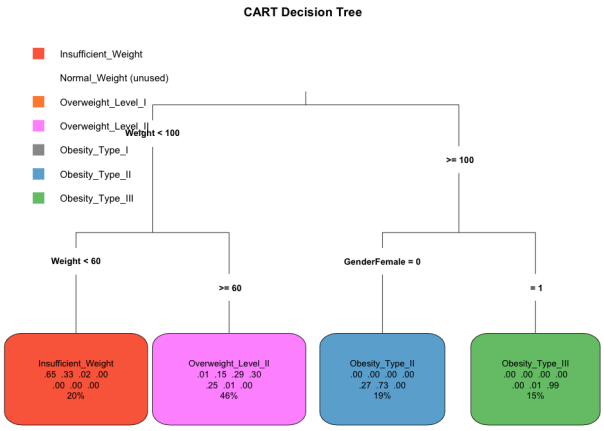

Figure 19: CART decision tree visualization.

3.3.3 Final Remarks

CART is easy to understand but doesn't perform well in this multiclass context. It relies too much on `Weight` and simple splits, which don't capture the complexity of predicting obesity levels.

While it's useful for interpretation or quick analysis, CART is not suitable as a final predictive model for this task.

3.4 Random Forest

Random Forest is an ensemble method that builds many decision trees and combines their results to improve accuracy and reduce overfitting. It is well-suited for complex, high-dimensional tasks like this multiclass obesity classification.

The model was trained using the `caret` framework, and predictions were evaluated on the test set using a confusion matrix and performance metrics.

3.4.1 Confusion Matrix – Random Forest

The Random Forest model performs exceptionally well, reaching 95.01% accuracy and a Kappa of 0.94, showing near-perfect agreement between predicted and actual labels.

Key results:

- Very high sensitivity across all classes, especially:
  - *Obesity Type II*: 100%
  - *Obesity Type I*: 98.57%

  - *Obesity Type III*: 99.22%

- Strong results also for:
  - *Insufficient Weight*: 96.30%
  - *Normal Weight*: 91.23%
  - *Overweight Level I*: 85.34%
  - *Overweight Level II*: 93.10%

Precision, specificity, and balanced accuracy are high for all classes. Most errors occur between similar categories, which is common in progressive labels like obesity levels.

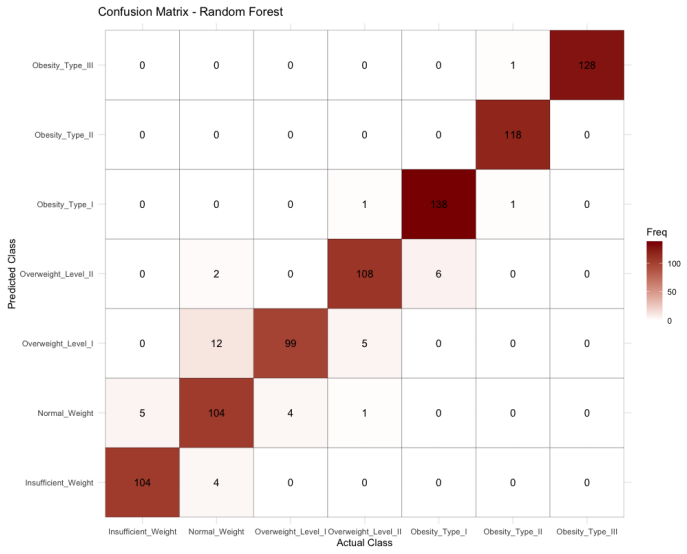

Figure 20: Confusion matrix for the Random Forest classifier.

Table 3: Overall performance metrics – Random Forest

Metric	**Value**
Accuracy	95.01%
95% Confidence Interval	[93.31%, 96.38%]
Kappa	0.9417
No Information Rate (NIR)	16.65%
P-value [Accuracy ¿ NIR]	$< 2.2 \times 10^{-16}$

The model clearly outperforms the No Information Rate (16.65%) and shows excellent generalization ability.

3.4.2 Variable Importance – Random Forest

The importance plot shows that Random Forest uses a wide and balanced set of features. Top contributors include:

- `Weight`, the most important predictor;
- `Height`, `Age`, and `FCVC` (vegetable consumption);
- `Gender`, `NCP` (number of meals), `TUE` (technology use), and `CH2O` (water intake).

Unlike simpler models like CART, Random Forest makes use of both biometric and lifestyle factors, improving prediction quality and robustness.

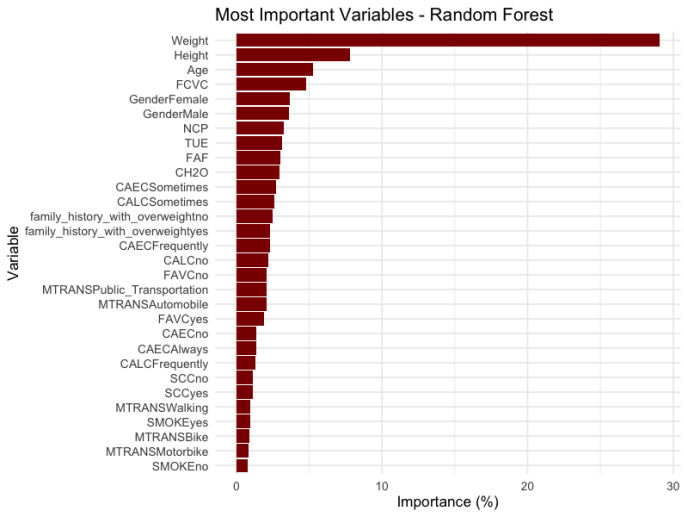

Figure 21: Variable importance plot for the Random Forest model.

3.4.3 Final Remarks – Random Forest

Random Forest is the best-performing model in this analysis. It shows excellent accuracy, consistent results across all classes, and balanced use of features.

Its ability to handle non-linear interactions and reduce overfitting makes it ideal for this kind of multiclass task. These results suggest it could be used in practical applications or as a benchmark for future models.

3.5 Multinomial Logistic Regression

Multinomial Logistic Regression (MLR) extends binary logistic regression to handle more than two classes. While simpler than models like Random Forest, MLR is still a valuable baseline due to its interpretability and solid theoretical base.

Here, the model was trained using the `caret` package and evaluated through a confusion matrix and performance metrics.

Confusion Matrix – Multinomial Logistic Regression

The MLR model performs very well, with 94.41% accuracy and a Kappa of 0.9347, indicating excellent agreement between predicted and actual labels.

Key points:

- All classes are predicted with high accuracy (balanced accuracy $> 93\%$).
- Especially strong performance for:
  - *Insufficient Weight*: 99.07%
  - *Normal Weight*: 94.74%
  - *Obesity Type I*: 95.00%
  - *Obesity Type II*: 96.61%
  - *Obesity Type III*: 98.45%
- Minor confusion appears between *Overweight Level I* and *Level II*, but sensitivity remains high (87.93% and 88.79%).

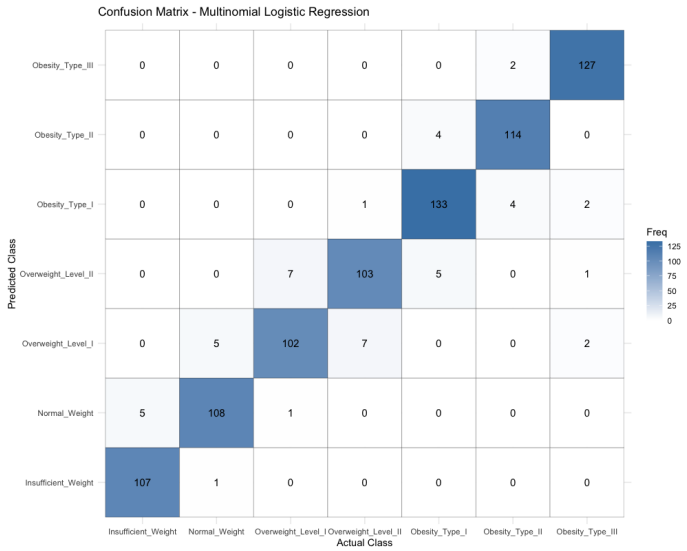

Figure 22: Confusion matrix for the Multinomial Logistic Regression model.

Table 4: Overall performance metrics – Multinomial Logistic Regression

Metric	**Value**
Accuracy	94.41%
95% Confidence Interval	[92.64%, 95.87%]
Kappa	0.9347
No Information Rate (NIR)	16.65%
P-value [Accuracy ¿ NIR]	$< 2.2 \times 10^{-16}$

The model clearly outperforms random guessing (NIR = 16.65%), with strong statistical significance. MLR proves to be a highly reliable and interpretable classifier.

Variable Importance – Multinomial Logistic Regression

The variable importance plot shows that the model relies mainly on a few predictors:

- `Weight` is the most important feature, followed by `Height`.
- `MTRANSBike` and `FCVC` (vegetable consumption) also contribute.
- Most other variables have little effect, showing the model depends mostly on biometric data.

This makes the model easy to interpret, but less flexible in learning complex patterns compared to more advanced models.

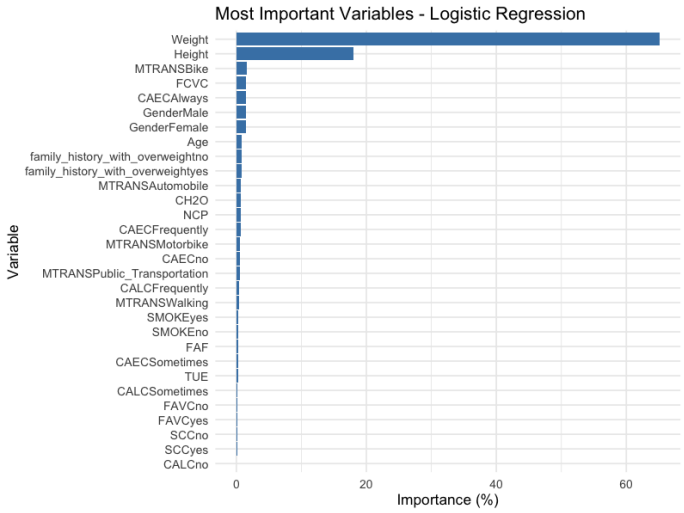

Figure 23: Variable importance plot for the Multinomial Logistic Regression model.

Final Remarks – Multinomial Logistic Regression

The MLR model performs very well, almost matching Random Forest in accuracy and class balance. Its strengths are simplicity, speed, and transparency, making it ideal when model interpretability is important.

Although it's not as powerful in capturing non-linear patterns, it clearly outperforms CART and shows more consistent results than SVM, especially in terms of balanced accuracy.

3.6 Gradient Boosting

Gradient Boosting is an ensemble method that builds a model step-by-step, combining many weak learners (usually decision trees) by minimizing errors at each stage. It is known for high accuracy and works well for complex multiclass problems.

Confusion Matrix – Gradient Boosting

The model performs very well, with 94.41% accuracy and a Kappa of 0.9347—identical to Multinomial Logistic Regression and close to the best-performing Random Forest.

Class-level results show:

- Very high sensitivity for all classes:
  - *Obesity Type III*: 99.22%
  - *Obesity Type II*: 99.15%
  - *Obesity Type I*: 97.86%
  - *Overweight Level II*: 90.52%
  - *Overweight Level I*: 87.07%
  - *Normal Weight*: 87.72%
  - *Insufficient Weight*: 98.15%
- Errors are minimal and mostly between similar (adjacent) classes.

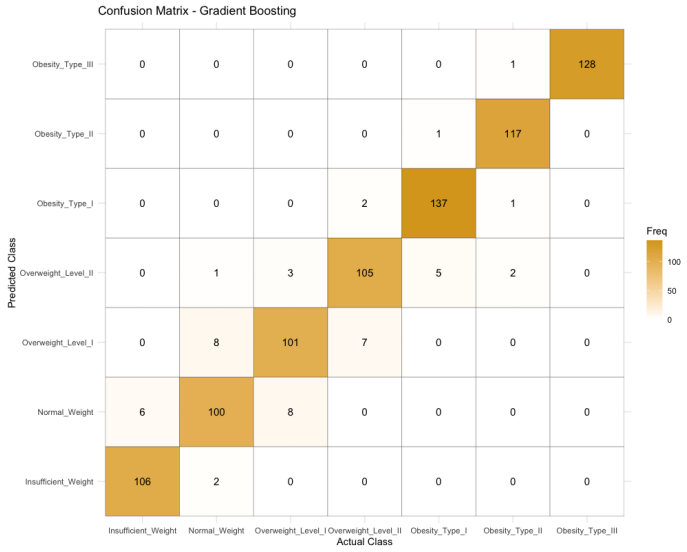

Figure 24: Confusion matrix for the Gradient Boosting model.

Table 5: Overall performance metrics – Gradient Boosting

Metric	**Value**
Accuracy	94.41%
95% Confidence Interval	[92.64%, 95.87%]
Kappa	0.9347
No Information Rate (NIR)	16.65%
P-value [Accuracy ¿ NIR]	$< 2.2 \times 10^{-16}$

The model clearly outperforms random guessing and shows strong, balanced performance across all obesity categories.

Variable Importance – Gradient Boosting

The importance plot shows that:

- `Weight` is the most important variable.
- Other top features include `Height`, `FCVC` (vegetables), and `Age`.
- `Gender`, `CH2O` (water intake), and `FAF` (physical activity) also contribute.

While many important features overlap with those used by Random Forest and Logistic Regression, Gradient Boosting spreads the importance more evenly, suggesting it captures a broader range of interactions.

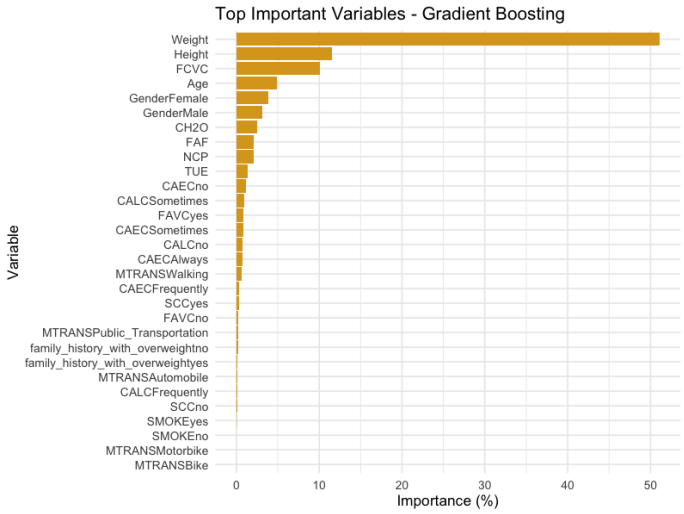

Figure 25: Variable importance plot for the Gradient Boosting model.

Final Remarks – Gradient Boosting

Gradient Boosting is one of the top models in this analysis. It matches Logistic Regression in accuracy and balance and comes close to Random Forest in predictive power.

It combines the interpretability of decision trees with strong performance, making it a reliable choice when both accuracy and transparency are needed. Even if it doesn't surpass Random Forest, it is highly effective and well-balanced across all obesity levels.

3.7 Neural Network (nnet)

Artificial Neural Networks (ANN) are flexible models that can capture complex non-linear relationships between features and outcomes. Here, a simple feed-forward neural network was trained using the `nnet` package to classify obesity levels. Neural networks often require more tuning and preprocessing compared to tree-based or regression models.

Confusion Matrix – Neural Network

The network reached 87.51% accuracy and a Kappa of 0.8542. This is strong, but slightly below the performance of models like Random Forest, Gradient Boosting, and Logistic Regression.

Key results by class:

- Very good sensitivity for:
  - *Insufficient Weight*: 98.15%

  - *Obesity Type II*: 100%
  - *Obesity Type I*: 95%
  - *Obesity Type III*: 98.45%
- Weaker performance for middle-range categories:
  - *Normal Weight*: 58.77%
  - *Overweight Level I*: 72.41%
  - *Overweight Level II*: 87.07%

There is some confusion between *Normal Weight* and *Insufficient Weight*, and between *Overweight Level I* and *Level II*, more than in the best-performing models.

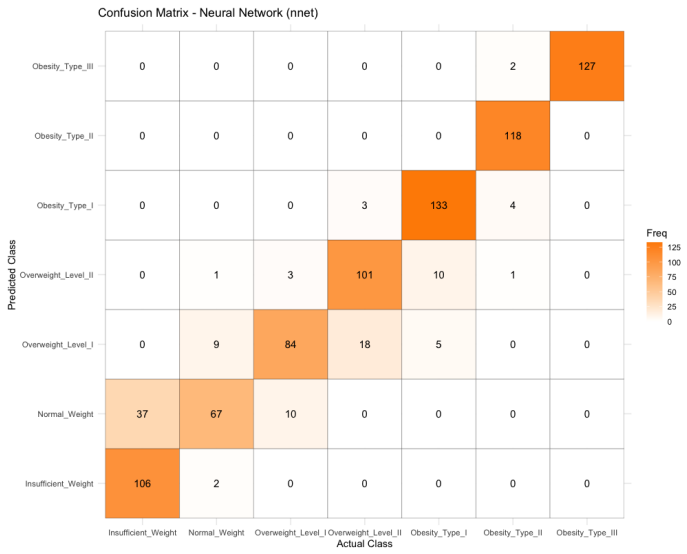

Figure 26: Confusion matrix for the Neural Network model.

Table 6: Overall performance metrics – Neural Network

Metric	**Value**
Accuracy	87.51%
95% Confidence Interval	[85.09%, 89.67%]
Kappa	0.8542
No Information Rate (NIR)	16.65%
P-value [Accuracy ¿ NIR]	$< 2.2 \times 10^{-16}$

The model performs well overall and clearly beats random guessing, but falls short compared to the best models in this analysis.

Variable Importance – Neural Network

The importance plot shows that the network uses a broad range of features:

- Top features are `Weight`, `Height`, and `Gender`.
- Other important predictors include `Transportation methods` (especially motorbike use), `eating habits`, and `physical activity`.

Unlike Logistic Regression, which focuses on a few features, the Neural Network spreads importance more evenly. This gives it flexibility but makes interpretation less straightforward.

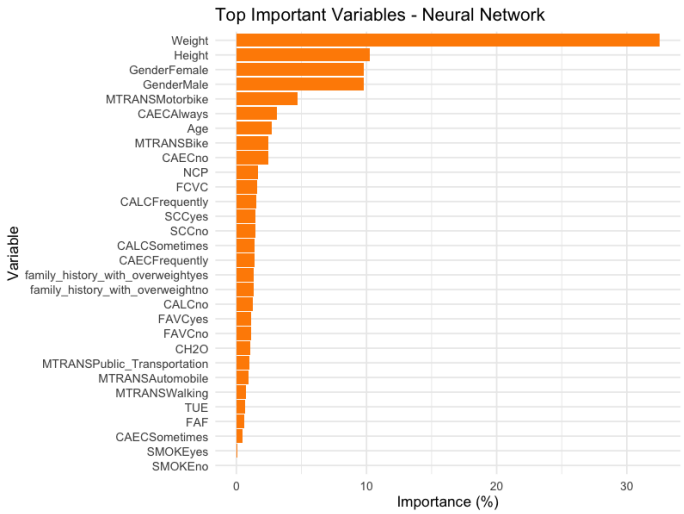

Figure 27: Variable importance plot for the Neural Network model.

Final Remarks – Neural Network

The neural network shows strong performance, especially for extreme obesity levels. However, it struggles with middle categories like *Normal Weight* and *Overweight*.

While its overall accuracy and Kappa are good, it doesn't outperform Random Forest or Gradient Boosting. It also requires more computation without clear gains. Deeper architectures or better tuning could improve results, but in its current form, it remains reliable but not top-performing.

3.8 Comparison of Supervised Learning Models

To compare the supervised learning models, we focus on identifying the most accurate and reliable one for predicting obesity levels. The ideal model should show:

- High **accuracy** (correct predictions),
- High **Kappa** (agreement beyond chance),
- Strong **F1-macro** (balanced performance across classes),
- Low **RMSE** (low prediction error),
- Consistent performance across cross-validation.

This is especially important due to the multiclass and imbalanced nature of the dataset.

3.8.1 Graphical Comparison of Models

The **Accuracy Comparison Boxplot** shows that *Random Forest (RF)*, *Gradient Boosting (GBM)*, and *Logistic Regression (LogReg)* have the highest and most consistent accuracy across cross-validation. *SVM* and *CART* show the lowest and most variable accuracy, meaning they are less reliable.

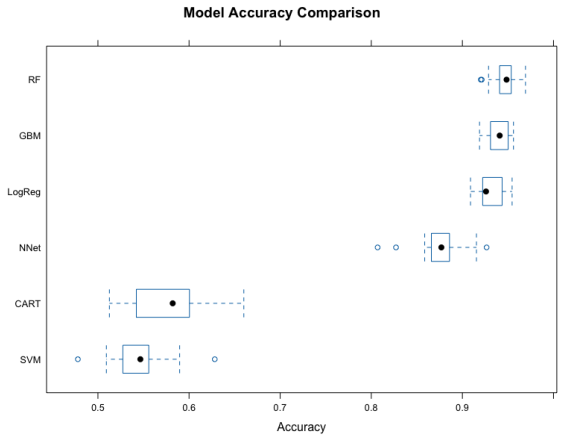

Figure 28: Accuracy comparison of supervised learning models across cross-validation.

The **Kappa Comparison Boxplot** measures how well predictions match actual values beyond random chance. RF, GBM, and LogReg again rank at the top, confirming strong and stable performance. SVM and CART show much weaker agreement.

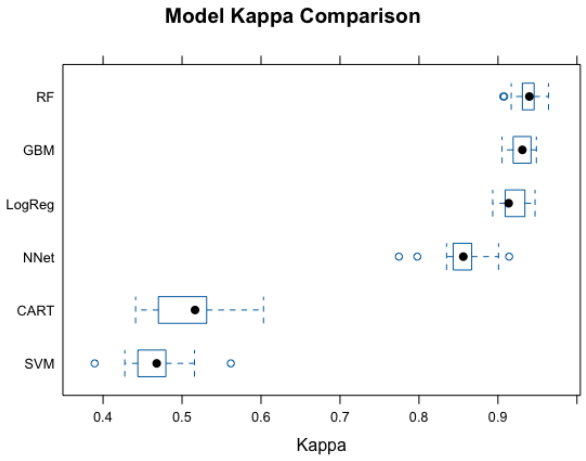

Figure 29: Kappa comparison of supervised learning models across cross-validation.

3.8.2 Numerical Summary of Model Performance

Table 7 summarizes all main metrics:

Table 7: Model performance comparison summary

Model	Accuracy	Kappa	F1_Macro	RMSE
Random Forest (RF)	0.950	0.942	0.992	0.246
Logistic Regression (LogReg)	0.944	0.935	0.991	0.329
Gradient Boosting (GBM)	0.944	0.935	0.991	0.258
Neural Network (NNet)	0.875	0.854	0.979	0.387
CART	0.547	0.474	0.921	0.848
SVM	0.543	0.465	0.922	0.878

Random Forest (RF) is the best model across all metrics, with the highest accuracy, Kappa, and F1-macro, and the lowest RMSE. **Logistic Regression** and **GBM** also perform very well but have slightly higher RMSE.

Neural Network (NNet) performs well in terms of F1, but its RMSE is higher. **CART** and **SVM** clearly underperform across all metrics.

3.8.3 Performance Summary Plot

The final plot combines all four metrics into a single view. RF, GBM, and LogReg lead across the board. NNet is solid but slightly behind. CART and SVM are consistently the weakest.

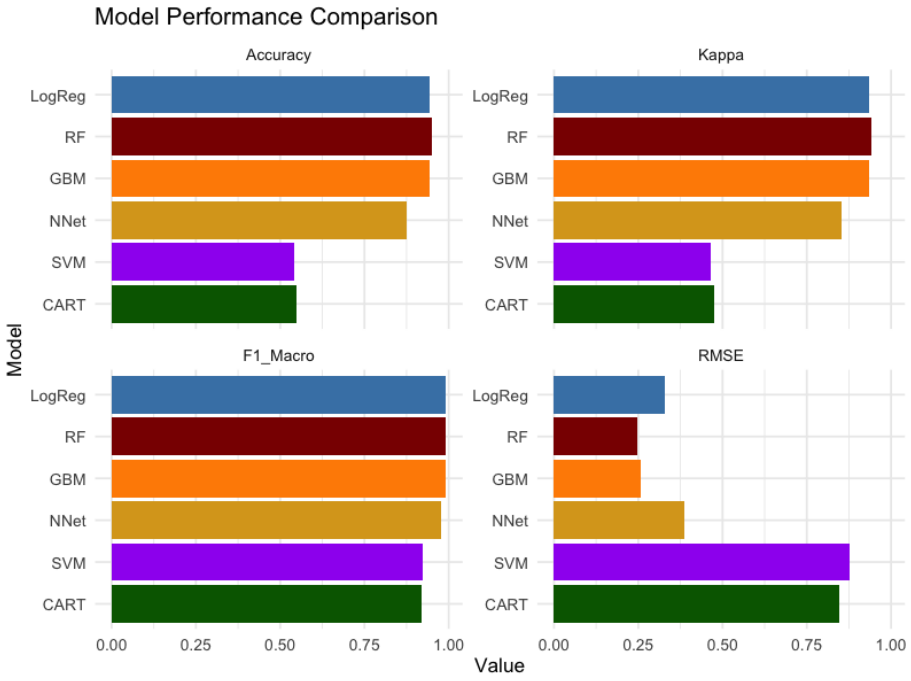

Figure 30: Overall model performance comparison across Accuracy, Kappa, F1-macro, and RMSE.

3.9 ROC Curve Comparison

Beyond accuracy and Kappa, ROC curves give insight into how well each model can distinguish between classes. In multiclass problems, ROC curves show the trade-off between sensitivity (true positive rate) and 1-specificity (false positive rate) for each class.

Here we compare the ROC curves for four models: *Random Forest (RF)*, *Gradient Boosting (GBM)*, *Logistic Regression (LogReg)*, and *CART*. Each figure includes curves for all seven obesity categories.

Random Forest

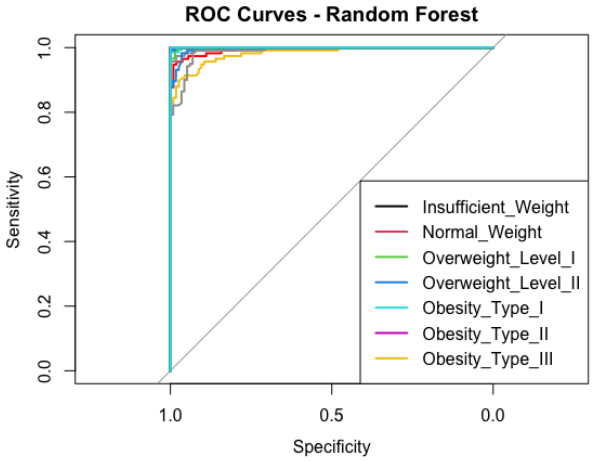

Figure 31: ROC curves for Random Forest.

Random Forest shows excellent performance, with most curves in the top-left area—meaning high sensitivity and low false positives across all classes. The strong separation confirms that RF is the best at distinguishing between obesity levels.

Gradient Boosting

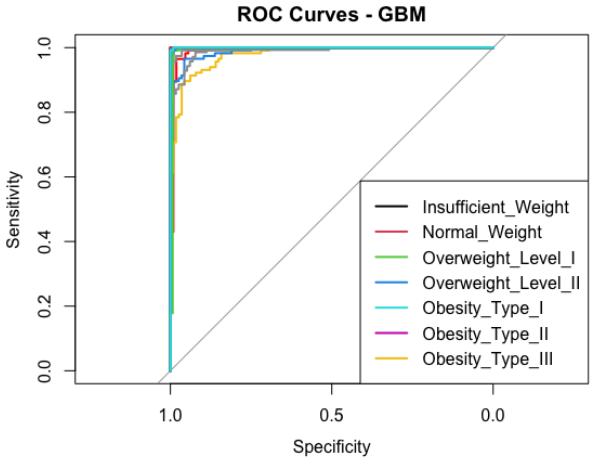

Figure 32: ROC curves for Gradient Boosting.

Gradient Boosting also performs very well. Its ROC curves are close to those of Random Forest, slightly less sharp for some classes but still showing strong and consistent classification quality.

Logistic Regression

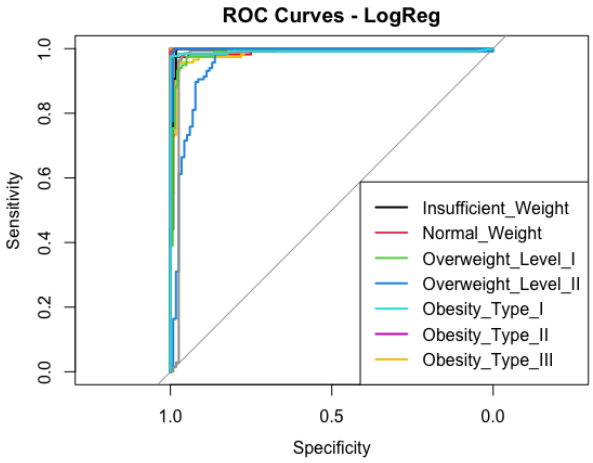

Figure 33: ROC curves for Logistic Regression.

Logistic Regression produces well-structured ROC curves with good sensitivity and specificity. There's slightly more variation for minority classes, but overall, the model shows strong discriminative power.

CART

Figure 34: ROC curves for CART.

CART performs poorly. Many curves are near the diagonal line, meaning it struggles to separate classes properly. This confirms what we saw in the earlier metrics—CART is not effective for this task.

Conclusion

ROC curves confirm that **Random Forest** is the best model for distinguishing between obesity levels, followed closely by **Gradient Boosting** and **Logistic Regression**. **CART** performs the worst. These results match the accuracy and Kappa scores, showing the value of ROC analysis in evaluating multiclass models with imbalanced data.

3.10 Conclusions – Supervised Learning

The supervised learning phase showed how important model selection is when dealing with complex multiclass problems like predicting obesity levels using lifestyle and biometric data. We tested six models — *SVM*, *CART*, *Random Forest*, *Logistic Regression*, *Gradient Boosting*, and *Neural Networks* — to compare their performance.

Random Forest was the best overall. It had the highest accuracy (95%), the best Kappa score (0.94), and strong balance across all classes. Its ROC curves showed excellent separation between categories. The variable importance plot confirmed that `Weight`, `Height`, and `Age` are key features in predicting obesity.

Gradient Boosting and **Multinomial Logistic Regression** also performed very well. Their accuracy was just below Random Forest, but they

showed stable results across all classes and strong interpretability. Their ROC curves supported these findings, with only small drops in performance for some categories.

Neural Networks gave solid results, especially for extreme obesity levels. However, it was less stable on middle categories and had a slightly higher RMSE, showing some inconsistency. On the other hand, **SVM** and **CART** performed poorly. SVM had trouble handling multiple classes, and CART, while easy to understand, failed to model the data effectively.

In summary, ensemble models — especially *Random Forest* and *Gradient Boosting* — were the most effective for this task. They combined strong accuracy, good generalization, and useful feature insights, making them ideal for real-world use in health prediction and personalized recommendations.

This analysis also shows the importance of using multiple evaluation metrics — not just accuracy, but also Kappa, F1-macro, RMSE, and ROC curves — to fully understand model performance in multiclass and imbalanced problems.

4 R Code

This section provides the full R code used for the implementation of both the unsupervised and supervised learning phases of the analysis.

4.1 Unsupervised Learning Script

```r
# ---------------------------
# Unsupervised Learning
# ---------------------------

# Load required libraries
library(tidyverse)
library(caret)
library(FactoMineR)
library(factoextra)
library(plotly)
library(reshape2)
library(pheatmap)
library(RColorBrewer)
library(fmsb)

# Load the dataset
obesity_df <- read.csv("data/obesity.csv", header = TRUE)

head(obesity_df)
summary(obesity_df)

# Save 'Gender' column as factor for plotting
gender_col <- obesity_df$Gender

# Remove the target variable
df_data <- dplyr::select(obesity_df, -NObeyesdad)

# Convert categorical variables into dummy variables
df_dummy <- dummyVars(~ ., data = df_data)
df_transformed <- predict(df_dummy, newdata = df_data) %>%
    as.data.frame()

head(df_transformed)

# Standardize the data
scaled_data <- scale(df_transformed)

head(scaled_data)

# Check for missing values
sum(is.na(scaled_data))

```

```r
# ------------------------------
# Exploratory Outlier Analysis
# ------------------------------

# Histogram of numeric variables
numeric_data <- df_data %>% select(where(is.numeric))

numeric_data %>%
  gather(key = "Variable", value = "Value") %>%
  ggplot(aes(x = Value)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
  facet_wrap(~ Variable, scales = "free") +
  theme_minimal() +
  labs(title = "Histograms of Numeric Variables")

# Boxplot of numeric variables
numeric_data %>%
  gather(key = "Variable", value = "Value") %>%
  ggplot(aes(x = Variable, y = Value, fill = Variable)) +
  geom_boxplot() +
  theme_minimal() +
  scale_fill_brewer(palette = "Set3") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Boxplots of Numeric Variables")

# Histogram of numeric variables
obesity_df %>%
  dplyr::select_if(~!is.numeric(.)) %>% # select the categorical variable
  gather(key = "variable", value = "value") %>%
  ggplot(aes(x = value)) +
  geom_bar(fill = "steelblue", color = "black") +
  facet_wrap(~ variable, scales = "free", ncol = 2) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1)) +
  labs(title = "Categorical Variables Distributions", x = "", y =
      "Count")

# ------------------------------
# Correlation Heatmap
# ------------------------------

cor_matrix <- cor(scaled_data, use = "complete.obs")
melted_cor <- melt(cor_matrix)

ggplot(melted_cor, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "darkred", mid = "white", high =
      "darkblue", midpoint = 0) +
  theme_minimal() +
  labs(title = "Correlation Heatmap", x = "", y = "") +
```

```r
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# -------------------------------
# Principal Component Analysis
# -------------------------------

pca_result <- prcomp(scaled_data, center = TRUE, scale. = TRUE)

# Explained variance
fviz_eig(pca_result, addlabels = TRUE, ylim = c(0, 50))

# Contribution of variables to PCs
fviz_pca_var(pca_result,
             col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)

# PCA by gender
fviz_pca_ind(pca_result,
             geom.ind = "point",
             col.ind = gender_col,
             palette = "jco",
             addEllipses = TRUE,
             legend.title = "Gender")

# Extract first two PCs
pca_data <- pca_result$x[, 1:2]

# -------------------------------
# Optimal Number of Clusters
# -------------------------------

fviz_nbclust(pca_data, kmeans, method = "wss") +
  labs(subtitle = "Elbow Method")

fviz_nbclust(pca_data, kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette Method")

# -------------------------------
# K-Means Clustering (k = 7)
# -------------------------------

set.seed(123)
kmeans_7 <- kmeans(pca_data, centers = 7, nstart = 25)
df_clustered <- df_data

# Add cluster column
df_clustered$cluster <- as.factor(kmeans_7$cluster)

# Cluster visualization
```

```r
fviz_cluster(kmeans_7, data = pca_data,
             ellipse.type = "convex",
             palette = "jco",
             ggtheme = theme_minimal())

# -------------------------------
# 3D PCA Cluster Plot
# -------------------------------

df_pca_3d <- as.data.frame(pca_result$x[, 1:3])
colnames(df_pca_3d) <- c("PC1", "PC2", "PC3")
df_pca_3d$Cluster <- as.factor(kmeans_7$cluster)

plot_ly(df_pca_3d, x = ~PC1, y = ~PC2, z = ~PC3,
        color = ~Cluster,
        colors = c('#E41A1C', '#377EB8', '#4DAF4A', '#984EA3',
                   '#FF7F00', '#FFFF33', '#A65628'),
        type = "scatter3d", mode = "markers")

# -------------------------------
# Cluster Profile Summary
# -------------------------------

summary_7 <- df_clustered %>%
  group_by(cluster) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE))

# Normalize for heatmap
norm_7 <- summary_7 %>%
  column_to_rownames("cluster") %>%
  scale(center = TRUE, scale = TRUE) %>%
  as.data.frame() %>%
  rownames_to_column("cluster") %>%
  pivot_longer(-cluster, names_to = "Variable", values_to = "Value")

# Heatmap

ggplot(norm_7, aes(x = Variable, y = cluster, fill = Value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(Value, 2)), size = 3) +
  scale_fill_gradient2(low = "darkred", mid = "white", high =
      "darkblue", midpoint = 0) +
  labs(title = "Heatmap - Cluster (z-score)", x = "Variables", y =
      "Cluster") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# -------------------------------
# Distribution Plots by Cluster
# -------------------------------
```

```r

# Age
ggplot(df_clustered, aes(x = Age, fill = cluster)) +
  geom_density(alpha = 0.5) +
  labs(title = "Cluster Age Distribution")
```

Listing 1: R Script for Unsupervised Learning

4.2 Supervised Learning Script

```r
# ---------------------------
# Supervised Learning
# ---------------------------

# Load all necessary libraries
libraries <- c("tidyverse", "caret", "e1071", "randomForest", "nnet",
               "xgboost", "ggplot2", "ROCR", "MASS", "mlbench",
               "reshape2", "gbm", "MLmetrics", "pROC", "rpart.plot")

lapply(libraries, library, character.only = TRUE)

# Load the dataset
obesity_df <- read.csv("data/obesity.csv", header = TRUE)

# Convert the target variable to a factor (multiclass) with correct
    level order
obesity_df$NObeyesdad <- factor(obesity_df$NObeyesdad,
                                levels = c("Insufficient_Weight",
                                     "Normal_Weight",
                                           "Overweight_Level_I",
                                               "Overweight_Level_II",
                                           "Obesity_Type_I",
                                               "Obesity_Type_II",
                                               "Obesity_Type_III"))

# Create dummy variables for predictors (excluding target)
dummies <- dummyVars(NObeyesdad ~ ., data = obesity_df)
df_features <- predict(dummies, newdata = obesity_df) %>% as.data.frame()

# Add the target variable (as factor)
df_features$target <- obesity_df$NObeyesdad

# Split into training and testing sets
set.seed(123)
train_index <- createDataPartition(df_features$target, p = 0.6, list =
    FALSE)
train_data <- df_features[train_index, ]
test_data <- df_features[-train_index, ]

```

```r
# Remove zero variance variables (they don't contribute to prediction)
nzv <- nearZeroVar(train_data, saveMetrics = TRUE)
zero_var_cols <- rownames(nzv[nzv$zeroVar == TRUE, ])

# print the deleted column
cat("Variable with zero variance:", zero_var_cols, "\n")

# remove from train and test
train_data <- train_data[, !names(train_data) %in% zero_var_cols]
test_data <- test_data[, !names(test_data) %in% zero_var_cols]

# Quick EDA: overview
str(obesity_df)
summary(obesity_df)

# Class distribution
ggplot(obesity_df, aes(x = NObeyesdad)) +
  geom_bar(fill = "limegreen") +
  labs(title = "Distribution of Target Variable", x = "Obesity Level", y
      = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1))

# Numeric variables distribution
obesity_df %>%
  select_if(is.numeric) %>%
  gather(key = "variable", value = "value") %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 30, fill = "limegreen", color = "black") +
  facet_wrap(~ variable, scales = "free") +
  theme_minimal() +
  labs(title = "Numeric Variables Distributions")

# Categorical variable
obesity_df %>%
  dplyr::select_if(~!is.numeric(.)) %>% # select the categorical variable
  gather(key = "variable", value = "value") %>%
  ggplot(aes(x = value)) +
  geom_bar(fill = "limegreen", color = "black") +
  facet_wrap(~ variable, scales = "free", ncol = 2) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1)) +
  labs(title = "Categorical Variables Distributions", x = "", y =
      "Count")

# Distribution of the category in respect to the target
obesity_df %>%
  dplyr::select_if(~!is.numeric(.)) %>%
  gather(key = "variable", value = "value", -NObeyesdad) %>%
  ggplot(aes(x = value, fill = NObeyesdad)) +
```

```r
  geom_bar(position = "fill") +
  facet_wrap(~ variable, scales = "free_x", ncol = 2) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1)) +
  labs(title = "Proportion of Target by Categorical Variables", y =
      "Proportion", x = "") +
  scale_fill_manual(
    values = colorRampPalette(c("limegreen", "yellow",
        "red"))(length(levels(obesity_df$NObeyesdad))),
    name = "Obesity Level"
  )

# Boxplot for numeric variables (to detect outliers visually)
obesity_df %>%
  select_if(is.numeric) %>%
  gather(key = "variable", value = "value") %>%
  ggplot(aes(x = "", y = value)) +
  geom_boxplot(fill = "limegreen") +
  facet_wrap(~ variable, scales = "free") +
  theme_minimal() +
  labs(title = "Boxplots for Numeric Variables", y = "Value", x = "")

# ---------------------------
# SVM
# ---------------------------

# Train the SVM model (radial kernel) using caret
set.seed(123)
model_svm <- train(
  target ~ .,
  data = train_data,
  method = "svmRadial",
  trControl = trainControl(classProbs = TRUE),
  preProcess = c("center", "scale") # Standardize features
)

# Predict on test set
pred_svm <- predict(model_svm, test_data)

# Confusion Matrix
cat("\nConfusion Matrix - SVM\n")
conf_mat_svm <- confusionMatrix(pred_svm, test_data$target)
print(conf_mat_svm)

# Convert confusion matrix to dataframe for plotting
cm_df_svm <- as.data.frame(conf_mat_svm$table)
colnames(cm_df_svm) <- c("Reference", "Prediction", "Freq")

# Plot Confusion Matrix
ggplot(cm_df_svm, aes(x = Reference, y = Prediction, fill = Freq)) +
```

```
  geom_tile(color = "black") +
  geom_text(aes(label = Freq), size = 4) +
  scale_fill_gradient(low = "white", high = "purple") +
  labs(
    title = "Confusion Matrix - SVM",
    x = "Actual Class",
    y = "Predicted Class"
  ) +
  theme_minimal()

# Plot variable importance for SVM
var_imp_svm <- varImp(model_svm)$importance
var_imp_svm$Overall <- rowMeans(var_imp_svm)
var_imp_svm$ImportancePercent <- 100 * var_imp_svm$Overall /
    sum(var_imp_svm$Overall)
var_imp_svm$Variable <- rownames(var_imp_svm)

var_imp_svm <- var_imp_svm %>%
  dplyr::arrange(desc(ImportancePercent))

# Plot
ggplot(var_imp_svm, aes(x = reorder(Variable, ImportancePercent), y =
    ImportancePercent)) +
  geom_col(fill = "purple") +
  coord_flip() +
  labs(
    title = "Important Variables - SVM",
    x = "Variable",
    y = "Importance (%)"
  ) +
  theme_minimal(base_size = 13)

# ---------------------------
# CART
# ---------------------------

# Train the CART model for multiclass classification
set.seed(123)
model_cart <- train(
  target ~ .,
  data = train_data,
  method = "rpart",
  trControl = trainControl(classProbs = TRUE)
)

# Predict on test set
pred_cart <- predict(model_cart, test_data)

# Print confusion matrix
cat("\nConfusion Matrix - CART\n")
```

```r
conf_mat_cart <- confusionMatrix(pred_cart, test_data$target)
print(conf_mat_cart)

# Visualize the confusion matrix with ggplot2
cm_df_cart <- as.data.frame(conf_mat_cart$table)
colnames(cm_df_cart) <- c("Reference", "Prediction", "Freq")

ggplot(cm_df_cart, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "black") +
  geom_text(aes(label = Freq), size = 4) +
  scale_fill_gradient(low = "white", high = "darkgreen") +
  labs(
    title = "Confusion Matrix - CART (Decision Tree)",
    x = "Actual Class",
    y = "Predicted Class"
  ) +
  theme_minimal()

# Plot variable importance
var_imp_cart <- varImp(model_cart)$importance
var_imp_cart$Overall <- rowMeans(var_imp_cart)
var_imp_cart$ImportancePercent <- 100 * var_imp_cart$Overall /
    sum(var_imp_cart$Overall)
var_imp_cart$Variable <- rownames(var_imp_cart)

var_imp_cart <- var_imp_cart %>%
  arrange(desc(ImportancePercent))

ggplot(var_imp_cart, aes(x = reorder(Variable, ImportancePercent), y =
    ImportancePercent)) +
  geom_col(fill = "darkgreen") +
  coord_flip() +
  labs(
    title = "Top Important Variables - CART",
    x = "Variable",
    y = "Importance (%)"
  ) +
  theme_minimal(base_size = 13)

# Visualize the tree
rpart.plot(model_cart$finalModel, type = 3, extra = 104, fallen.leaves =
    TRUE,
          main = "CART Decision Tree")

# ---------------------------
# Random Forest
# ---------------------------

# Train the Random Forest model for multiclass classification
set.seed(123)
```

```r
model_rf <- train(
  target ~ .,
  data = train_data,
  method = "rf",
  trControl = trainControl(classProbs = TRUE),
  importance = TRUE
)

# Predict on test set
pred_rf <- predict(model_rf, test_data)

# Print confusion matrix
cat("\nConfusion Matrix - RF\n")
conf_mat_rf <- confusionMatrix(pred_rf, test_data$target)
print(conf_mat_rf)

# Visualize the confusion matrix with ggplot2
cm_df_rf <- as.data.frame(conf_mat_rf$table)
colnames(cm_df_rf) <- c("Reference", "Prediction", "Freq")

ggplot(cm_df_rf, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "black") +
  geom_text(aes(label = Freq), size = 4) +
  scale_fill_gradient(low = "white", high = "darkred") +
  labs(
    title = "Confusion Matrix - Random Forest",
    x = "Actual Class",
    y = "Predicted Class"
  ) +
  theme_minimal()

# Plot variable importance
# Extract importance and calculate average across all classes
var_imp <- varImp(model_rf)$importance
var_imp$Overall <- rowMeans(var_imp)

# Convert to percentage
var_imp$ImportancePercent <- 100 * var_imp$Overall / sum(var_imp$Overall)

# Add variable name
var_imp$Variable <- rownames(var_imp)

# Top 20
var_imp <- var_imp %>%
  arrange(desc(ImportancePercent))

# Plot
ggplot(var_imp, aes(x = reorder(Variable, ImportancePercent), y =
    ImportancePercent)) +
  geom_col(fill = "darkred") +
```

```
  coord_flip() +
  labs(
    title = "Most Important Variables - Random Forest",
    x = "Variable",
    y = "Importance (%)"
  ) +
  theme_minimal(base_size = 13)

# ----------------------------------
# Multinomial Logistic Regression
# ----------------------------------

# Train the Multinomial Logistic Regression model
set.seed(123)
model_log <- train(
  target ~ .,
  data = train_data,
  method = "multinom",
  trControl = trainControl(classProbs = TRUE),
  preProcess = c("center", "scale"),
  trace = FALSE
)

# Predict on test set
pred_log <- predict(model_log, test_data)

# Print confusion matrix
cat("\nConfusion Matrix - Logistic Regression\n")
conf_mat_log <- confusionMatrix(pred_log, test_data$target)
print(conf_mat_log)

# Visualize the confusion matrix with ggplot2
cm_df_log <- as.data.frame(conf_mat_log$table)
colnames(cm_df_log) <- c("Reference", "Prediction", "Freq")

ggplot(cm_df_log, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "black") +
  geom_text(aes(label = Freq), size = 4) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(
    title = "Confusion Matrix - Multinomial Logistic Regression",
    x = "Actual Class",
    y = "Predicted Class"
  ) +
  theme_minimal()

# Plot variable importance
# Extract importance and calculate average across all classes
var_imp_log <- varImp(model_log)$importance
var_imp_log$Overall <- rowMeans(var_imp_log)
```

```r

# Convert to percentage
var_imp_log$ImportancePercent <- 100 * var_imp_log$Overall /
    sum(var_imp_log$Overall)
var_imp_log$Variable <- rownames(var_imp_log)

# Top 20
var_imp_log <- var_imp_log %>%
  arrange(desc(ImportancePercent))

ggplot(var_imp_log, aes(x = reorder(Variable, ImportancePercent), y =
    ImportancePercent)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Most Important Variables - Logistic Regression",
    x = "Variable",
    y = "Importance (%)"
  ) +
  theme_minimal(base_size = 13)

# ----------------------------
# Gradient Boosting
# ----------------------------

# Train the Gradient Boosting model for multiclass classification
set.seed(123)
model_gbm <- train(
  target ~ .,
  data = train_data,
  method = "gbm",
  trControl = trainControl(classProbs = TRUE),
  verbose = FALSE
)

# Predict on test set
pred_gbm <- predict(model_gbm, test_data)

# Print confusion matrix
cat("\nConfusion Matrix - Gradient Boosting\n")
conf_mat_gbm <- confusionMatrix(pred_gbm, test_data$target)
print(conf_mat_gbm)

# Visualize the confusion matrix with ggplot2
cm_df_gbm <- as.data.frame(conf_mat_gbm$table)
colnames(cm_df_gbm) <- c("Reference", "Prediction", "Freq")

ggplot(cm_df_gbm, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "black") +
  geom_text(aes(label = Freq), size = 4) +
```

```
  scale_fill_gradient(low = "white", high = "goldenrod") +
  labs(
    title = "Confusion Matrix - Gradient Boosting",
    x = "Actual Class",
    y = "Predicted Class"
  ) +
  theme_minimal()

# Plot variable importance
var_imp_gbm <- varImp(model_gbm)$importance
var_imp_gbm$Overall <- rowMeans(var_imp_gbm)
var_imp_gbm$ImportancePercent <- 100 * var_imp_gbm$Overall /
    sum(var_imp_gbm$Overall)
var_imp_gbm$Variable <- rownames(var_imp_gbm)

var_imp_gbm <- var_imp_gbm %>%
  arrange(desc(ImportancePercent))

ggplot(var_imp_gbm, aes(x = reorder(Variable, ImportancePercent), y =
    ImportancePercent)) +
  geom_col(fill = "goldenrod") +
  coord_flip() +
  labs(
    title = "Top Important Variables - Gradient Boosting",
    x = "Variable",
    y = "Importance (%)"
  ) +
  theme_minimal(base_size = 13)

# ----------------------------
# Neural Network (nnet)
# ----------------------------

# Train the Neural Network model for multiclass classification
set.seed(123)
model_nnet <- train(
  target ~ .,
  data = train_data,
  method = "nnet",
  trControl = trainControl(classProbs = TRUE),
  preProcess = c("center", "scale"),
  trace = FALSE,
  MaxNWts = 5000,
  linout = FALSE
)

# Predict on test set
pred_nnet <- predict(model_nnet, test_data)

# Print confusion matrix
```

```r
cat("\nConfusion Matrix - Neural Network\n")
conf_mat_nnet <- confusionMatrix(pred_nnet, test_data$target)
print(conf_mat_nnet)

# Visualize the confusion matrix with ggplot2
cm_df_nnet <- as.data.frame(conf_mat_nnet$table)
colnames(cm_df_nnet) <- c("Reference", "Prediction", "Freq")

ggplot(cm_df_nnet, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "black") +
  geom_text(aes(label = Freq), size = 4) +
  scale_fill_gradient(low = "white", high = "darkorange") +
  labs(
    title = "Confusion Matrix - Neural Network (nnet)",
    x = "Actual Class",
    y = "Predicted Class"
  ) +
  theme_minimal()

# Plot variable importance
var_imp_nnet <- varImp(model_nnet)$importance
var_imp_nnet <- as.data.frame(var_imp_nnet)
var_imp_nnet$Overall <- rowMeans(var_imp_nnet)
var_imp_nnet$ImportancePercent <- 100 * var_imp_nnet$Overall /
    sum(var_imp_nnet$Overall)
var_imp_nnet$Variable <- rownames(var_imp_nnet)

var_imp_nnet <- var_imp_nnet %>%
  dplyr::arrange(desc(ImportancePercent))

# Plot
ggplot(var_imp_nnet, aes(x = reorder(Variable, ImportancePercent), y =
    ImportancePercent)) +
  geom_col(fill = "darkorange") +
  coord_flip() +
  labs(
    title = "Top Important Variables - Neural Network",
    x = "Variable",
    y = "Importance (%)"
  ) +
  theme_minimal(base_size = 13)

# ---------------------------
# Compare Model Performance
# ---------------------------

# Create a resamples object
model_list <- resamples(
  list(
    SVM = model_svm,
```

```
    CART = model_cart,
    RF = model_rf,
    LogReg = model_log,
    NNet = model_nnet,
    GBM = model_gbm
  )
)

# Summary of cross-validated metrics
summary(model_list)

# Boxplot of Accuracy and Kappa
bwplot(model_list, metric = "Accuracy", main = "Model Accuracy
    Comparison")
bwplot(model_list, metric = "Kappa", main = "Model Kappa Comparison")

# Dotplot for overall comparison
dotplot(model_list, metric = "Accuracy", main = "Dotplot - Accuracy by
    Model")

# ---------------------------
# F1 Macro Score Function
# ---------------------------

f1_macro <- function(true, pred) {
  classes <- levels(true)
  f1_list <- sapply(classes, function(class) {
    F1_Score(y_true = ifelse(true == class, 1, 0),
             y_pred = ifelse(pred == class, 1, 0))
  })
  mean(f1_list, na.rm = TRUE)
}

# ---------------------------
# RMSE Score Function
# ---------------------------

# Convert target classes to numeric values
true_numeric <- as.numeric(test_data$target)

rmse_score <- function(pred) {
  pred_numeric <- as.numeric(pred)
  RMSE(pred_numeric, true_numeric)
}

# ---------------------------
# Create Final Comparison Table
# ---------------------------

# Function to extract Accuracy from a model
```

```r
conf_matrix <- function(model, test_data) {
  pred <- predict(model, test_data)
  cm <- confusionMatrix(pred, test_data$target)
  cm$overall["Accuracy"]
}

# Build the comparison dataframe
comparison_df <- data.frame(
  Model = c("SVM", "CART", "RF", "LogReg", "NNet", "GBM"),
  Accuracy = c(
    conf_matrix(model_svm, test_data),
    conf_matrix(model_cart, test_data),
    conf_matrix(model_rf, test_data),
    conf_matrix(model_log, test_data),
    conf_matrix(model_nnet, test_data),
    conf_matrix(model_gbm, test_data)
  ),
  Kappa = c(
    confusionMatrix(pred_svm, test_data$target)$overall["Kappa"],
    confusionMatrix(pred_cart, test_data$target)$overall["Kappa"],
    confusionMatrix(pred_rf, test_data$target)$overall["Kappa"],
    confusionMatrix(pred_log, test_data$target)$overall["Kappa"],
    confusionMatrix(pred_nnet, test_data$target)$overall["Kappa"],
    confusionMatrix(pred_gbm, test_data$target)$overall["Kappa"]
  ),
  F1_Macro = c(
    f1_macro(test_data$target, pred_svm),
    f1_macro(test_data$target, pred_cart),
    f1_macro(test_data$target, pred_rf),
    f1_macro(test_data$target, pred_log),
    f1_macro(test_data$target, pred_nnet),
    f1_macro(test_data$target, pred_gbm)
  ),
  RMSE = c(
    rmse_score(pred_svm),
    rmse_score(pred_cart),
    rmse_score(pred_rf),
    rmse_score(pred_log),
    rmse_score(pred_nnet),
    rmse_score(pred_gbm)
  )
)

# Round for clarity
comparison_df <- comparison_df %>%
  mutate(across(where(is.numeric), round, 3)) %>%
  arrange(desc(Accuracy))

# Print the final comparison table
print(comparison_df)
```

```r

# Convert the comparison dataframe to long format
comparison_long <- melt(comparison_df, id.vars = "Model")

model_colors <- c(
  "SVM" = "purple",
  "CART" = "darkgreen",
  "RF" = "darkred",
  "LogReg" = "steelblue",
  "NNet" = "goldenrod",
  "GBM" = "darkorange"
)

# Plot: one chart with facets for each metric
ggplot(comparison_long, aes(x = reorder(Model, value), y = value, fill = Model)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ variable, scales = "free_y") +
  coord_flip() +
  labs(title = "Model Performance Comparison",
       x = "Model",
       y = "Value") +
  scale_fill_manual(values = model_colors) +
  theme_minimal(base_size = 13)

# ---------------------------
# ROC curve
# ---------------------------

# Predict class probabilities - Random Forest
prob_rf <- predict(model_rf, test_data, type = "prob")

# Compute multiclass AUC
roc_rf <- multiclass.roc(test_data$target, prob_rf)
cat("Multiclass AUC - Random Forest:", round(roc_rf$auc, 3), "\n")

# Get individual one-vs-all ROC curves
rs_rf <- multiclass.roc(test_data$target, prob_rf)
rs_list <- rs_rf$rocs

# Plot ROC curves
plot.roc(rs_list[[1]][[1]], col = "red", main = "ROC Curves - Random Forest")
for (i in 2:length(rs_list)) {
  lines.roc(rs_list[[i]][[1]], col = i)
}
legend(
  "bottomright",
  legend = levels(test_data$target),
  col = 1:length(rs_list),
```

```
  lwd = 2
)

# Predict class probabilities - GBM
prob_gbm <- predict(model_gbm, test_data, type = "prob")

# Compute multiclass AUC
roc_gbm <- multiclass.roc(test_data$target, prob_gbm)
cat("Multiclass AUC - GBM:", round(roc_gbm$auc, 3), "\n")

# Get individual one-vs-all ROC curves
rs_gbm <- multiclass.roc(test_data$target, prob_gbm)
rs_list_gbm <- rs_gbm$rocs

# Plot ROC curves
plot.roc(rs_list_gbm[[1]][[1]], col = "red", main = "ROC Curves - GBM")
for (i in 2:length(rs_list_gbm)) {
  lines.roc(rs_list_gbm[[i]][[1]], col = i)
}
legend(
  "bottomright",
  legend = levels(test_data$target),
  col = 1:length(rs_list_gbm),
  lwd = 2
)

# Predict class probabilities - LogReg
prob_log <- predict(model_log, test_data, type = "prob")

# Compute multiclass AUC
roc_log <- multiclass.roc(test_data$target, prob_log)
cat("Multiclass AUC - LogReg:", round(roc_log$auc, 3), "\n")

# Get individual one-vs-all ROC curves
rs_log <- multiclass.roc(test_data$target, prob_log)
rs_list_log <- rs_log$rocs

# Plot ROC curves
plot.roc(rs_list_log[[1]][[1]], col = "red", main = "ROC Curves -
    LogReg")
for (i in 2:length(rs_list_log)) {
  lines.roc(rs_list_log[[i]][[1]], col = i)
}
legend(
  "bottomright",
  legend = levels(test_data$target),
  col = 1:length(rs_list_log),
  lwd = 2
)

```

```r
# Predict class probabilities - CART
prob_cart <- predict(model_cart, test_data, type = "prob")

# Compute multiclass AUC
roc_cart <- multiclass.roc(test_data$target, prob_cart)
cat("Multiclass AUC - CART:", round(roc_cart$auc, 3), "\n")

# Get individual one-vs-all ROC curves
rs_cart <- multiclass.roc(test_data$target, prob_cart)
rs_list_cart <- rs_cart$rocs

# Plot ROC curves
plot.roc(rs_list_cart[[1]][[1]], col = "red", main = "ROC Curves - CART")
for (i in 2:length(rs_list_cart)) {
  lines.roc(rs_list_cart[[i]][[1]], col = i)
}
legend(
  "bottomright",
  legend = levels(test_data$target),
  col = 1:length(rs_list_cart),
  lwd = 2
)
```

Listing 2: R Script for Supervised Learning