An Ensemble Model to Predict the Obesity levels based on eating habits and physical conditions

DA5030

Bhargava Udayagiri Raghunath

2023-12-02

Contents

Loading the dataset:

The dataset was obtained from: https://www.kaggle.com/datasets/aravindpcoder/obesity-or-cvd-risk-classifyregressorcluster/ (Obesity or CVD Risk (Classify/Regressor/Cluster), 2023)

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(e1071)
#install.packages("missMethods")
library(missMethods)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(kernlab)
```

Data Acquisition:

```
# Loading dataset from a url directly
obesity_data <- read.csv(url("https://drive.google.com/uc?id=1ONIAiOM0uNKt4eskYbOnAGtRqf0Y_1RO&export=d

str(obesity_data) # Examining datatypes, number of columns and rows
```

```
## 'data.frame':    2111 obs. of  17 variables:
##  $ Gender                         : chr  "Female" "Female" "Male" "Male" ...
##  $ Age                            : num  21 21 23 27 22 29 23 22 24 22 ...
##  $ Height                         : num  1.62 1.52 1.8 1.8 1.78 1.62 1.5 1.64 1.78 1.72 ...
##  $ Weight                         : num  64 56 77 87 89.8 53 55 53 64 68 ...
```

```
##  $ family_history_with_overweight: chr  "yes" "yes" "yes" "no" ...
##  $ FAVC                          : chr  "no" "no" "no" "no" ...
##  $ FCVC                          : num  2 3 2 3 2 2 3 2 3 2 ...
##  $ NCP                           : num  3 3 3 3 1 3 3 3 3 3 ...
##  $ CAEC                          : chr  "Sometimes" "Sometimes" "Sometimes" "Sometimes" ...
##  $ SMOKE                         : chr  "no" "yes" "no" "no" ...
##  $ CH2O                          : num  2 3 2 2 2 2 2 2 2 2 ...
##  $ SCC                           : chr  "no" "yes" "no" "no" ...
##  $ FAF                           : num  0 3 2 2 0 0 1 3 1 1 ...
##  $ TUE                           : num  1 0 1 0 0 0 0 0 1 1 ...
##  $ CALC                          : chr  "no" "Sometimes" "Frequently" "Frequently" ...
##  $ MTRANS                        : chr  "Public_Transportation" "Public_Transportation" "Public_Trans
##  $ NObeyesdad                    : chr  "Normal_Weight" "Normal_Weight" "Normal_Weight" "Overweight_L
```

```
unique(obesity_data$NObeyesdad) # Identifying different categories for target variable
```

```
## [1] "Normal_Weight"       "Overweight_Level_I"  "Overweight_Level_II"
## [4] "Obesity_Type_I"      "Insufficient_Weight" "Obesity_Type_II"
## [7] "Obesity_Type_III"
```

Data shaping - Introducing NA values, since there are none and then imputing values.

```
# Introducing missing values, since there are none
set.seed(1234)
obesity_data <- delete_MCAR(obesity_data, 0.01, c("Gender", "Age", "CH2O", "Weight"))

sum(is.na(obesity_data))
```

```
## [1] 84
```

```
# Random Forest for feature importance
obesity_data$NObeyesdad <- as.factor(obesity_data$NObeyesdad)
rf_model <- randomForest(NObeyesdad ~ ., data = obesity_data)
importance(rf_model)
```

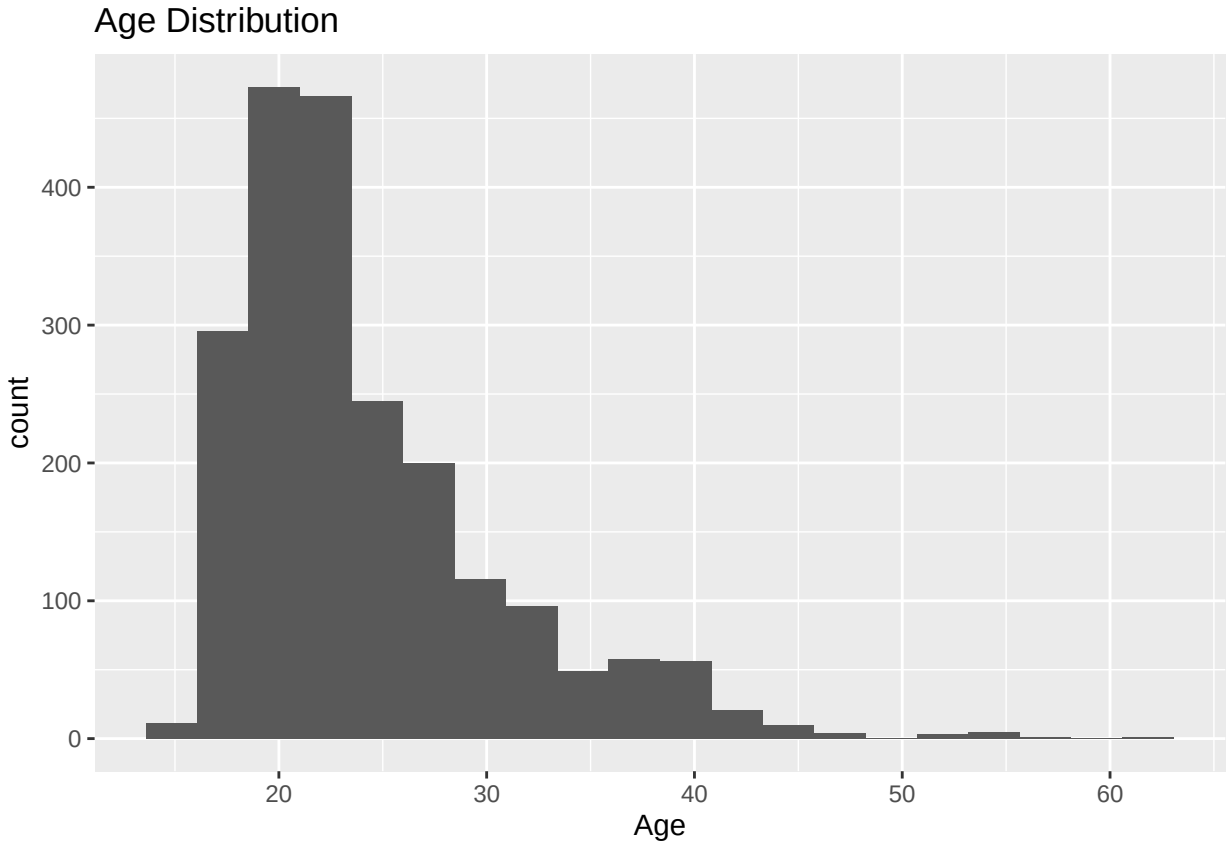
```
##                                  MeanDecreaseGini
## Age                                     167.576917
## Height                                  185.581850
## Weight                                  632.337312
## FCVC                                    161.442672
## NCP                                      89.117904
## CH2O                                     71.875686
## FAF                                      71.228830
## TUE                                      80.607834
## Gender                                   97.676812
## family_history_with_overweight           50.620525
## FAVC                                     28.116115
## CAEC                                     59.145683
## SMOKE                                     3.955759
```

```
## SCC                                10.201208
## CALC                               53.836551
## MTRANS                             33.355485
```

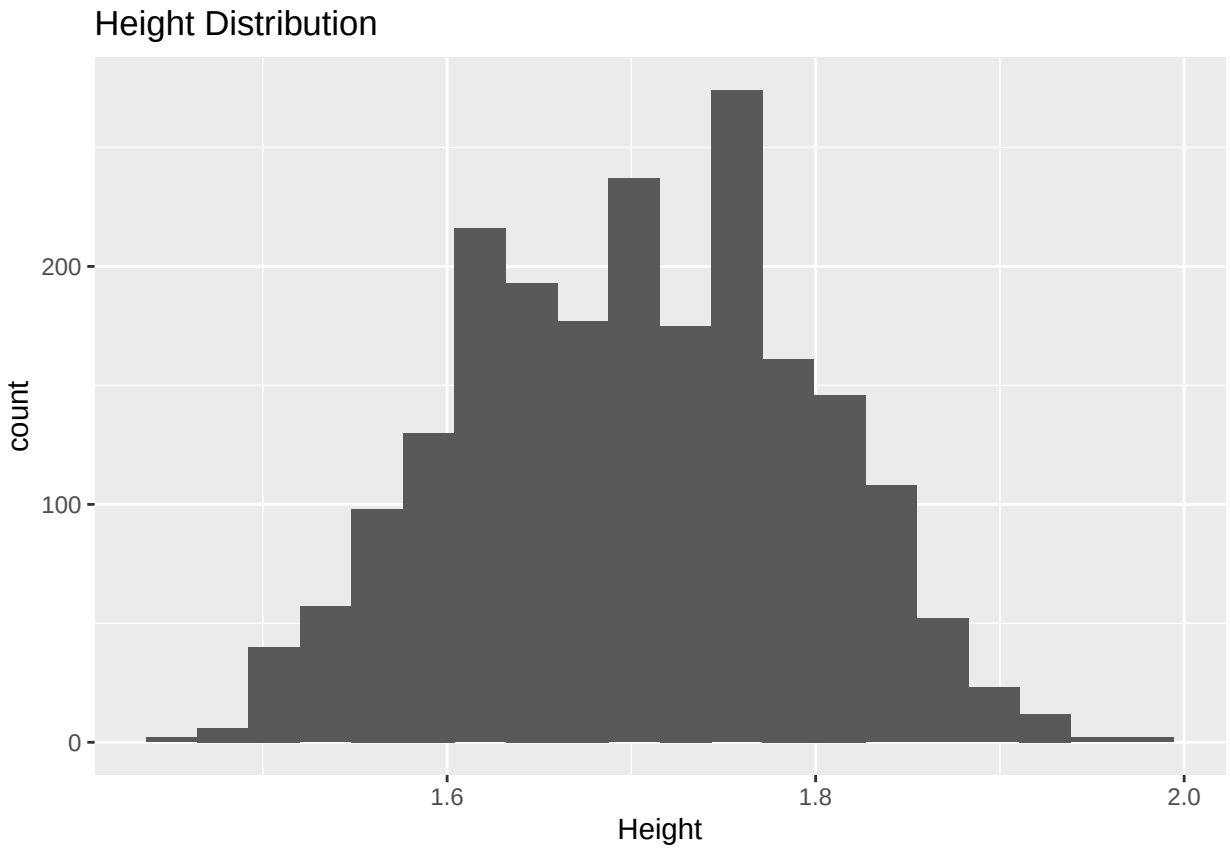
To select relevant features for classification models, based on the Gini Impurity values and domain knowledge from the research conducted for the dataset, we can see that features like SMOKE and SCC (Calories consumption monitoring) contribute very less to the analysis and therefore, I will be removing these columns for our analysis. Based solely on Gini impurity values, we could remove FAVC (Frequent consumption of high caloric food), however, going by domain knowledge, high calorific food intake is a major causal factor in determining people turning overweight and later obese.

Data Exploration:

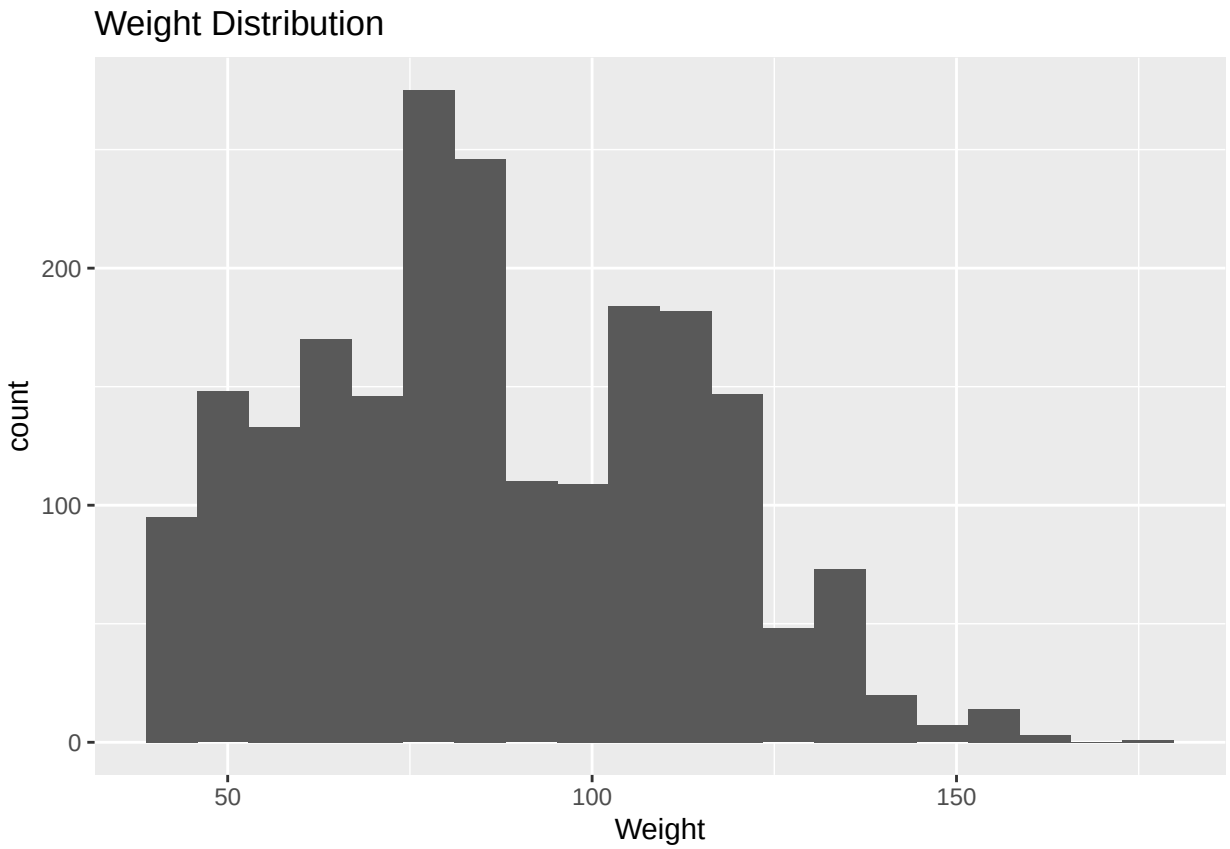
Data plots:



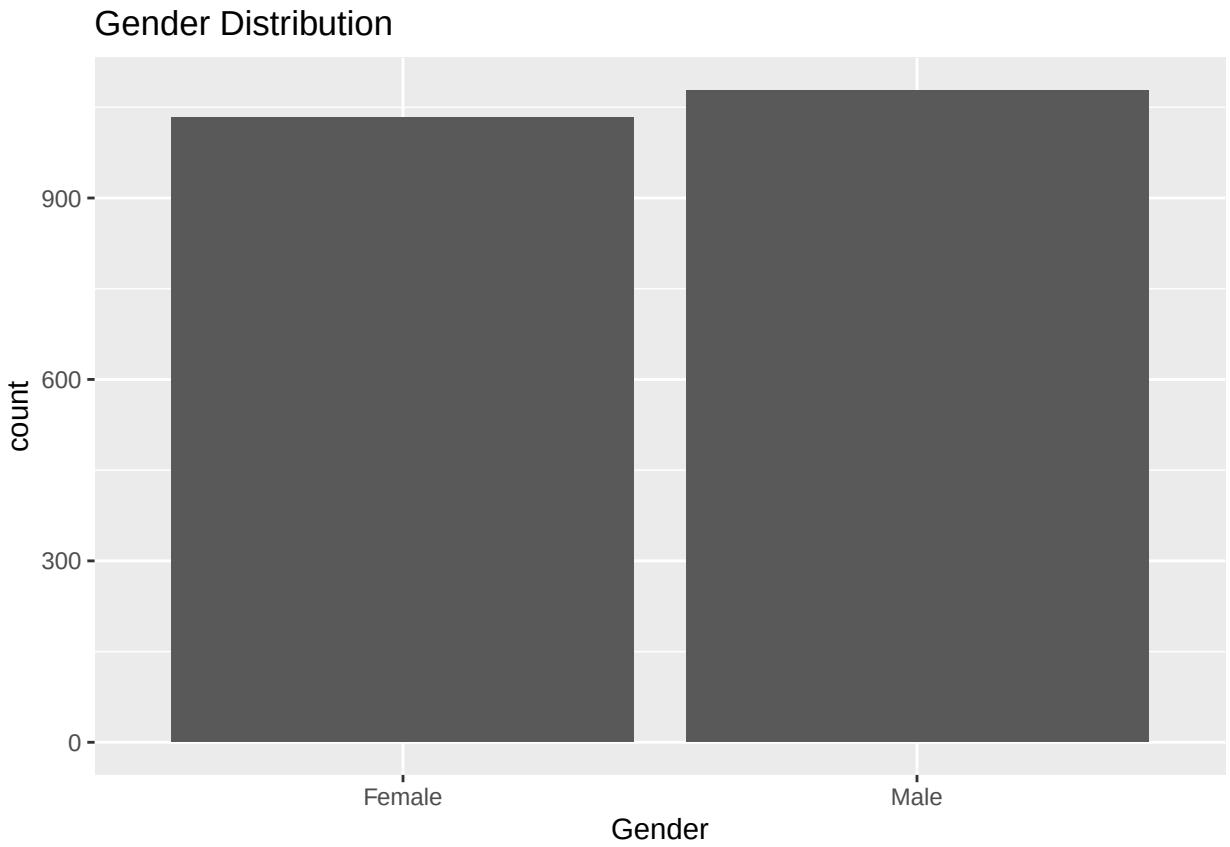
The histogram suggests that the age distribution is skewed right, with a higher frequency of younger age groups and a lower frequency of older demographic. The peak of the distribution seems to be in the range of around 20 years of age, indicating that this is the most common age in the data set.



This height distribution suggests a multi-modal pattern, which is indicative of several subgroups with distinct average heights within the dataset. Heights range from 1.5 to 2.0 meters, aligning with typical adult heights. Importantly, a large number of individuals have heights concentrated around 1.6 to 1.8 meters. The spread of the histogram bars shws a considerable variability in heights. Several height intervals show higher frequencies, reaching towards the upper end of the count scale on the y-axis, therefore, they are still relevant to the analysis. The distribution also shows very little skewness, mostly showing a normal distribution.

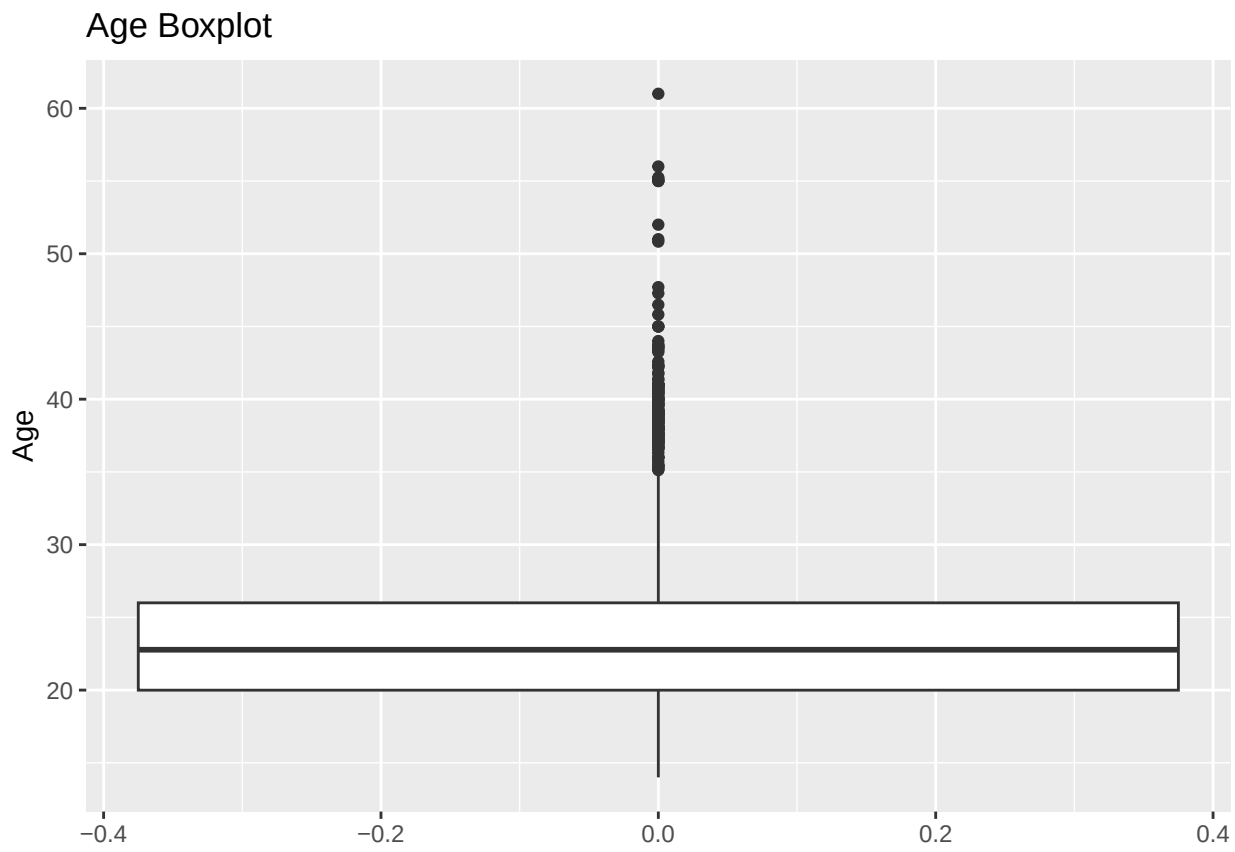


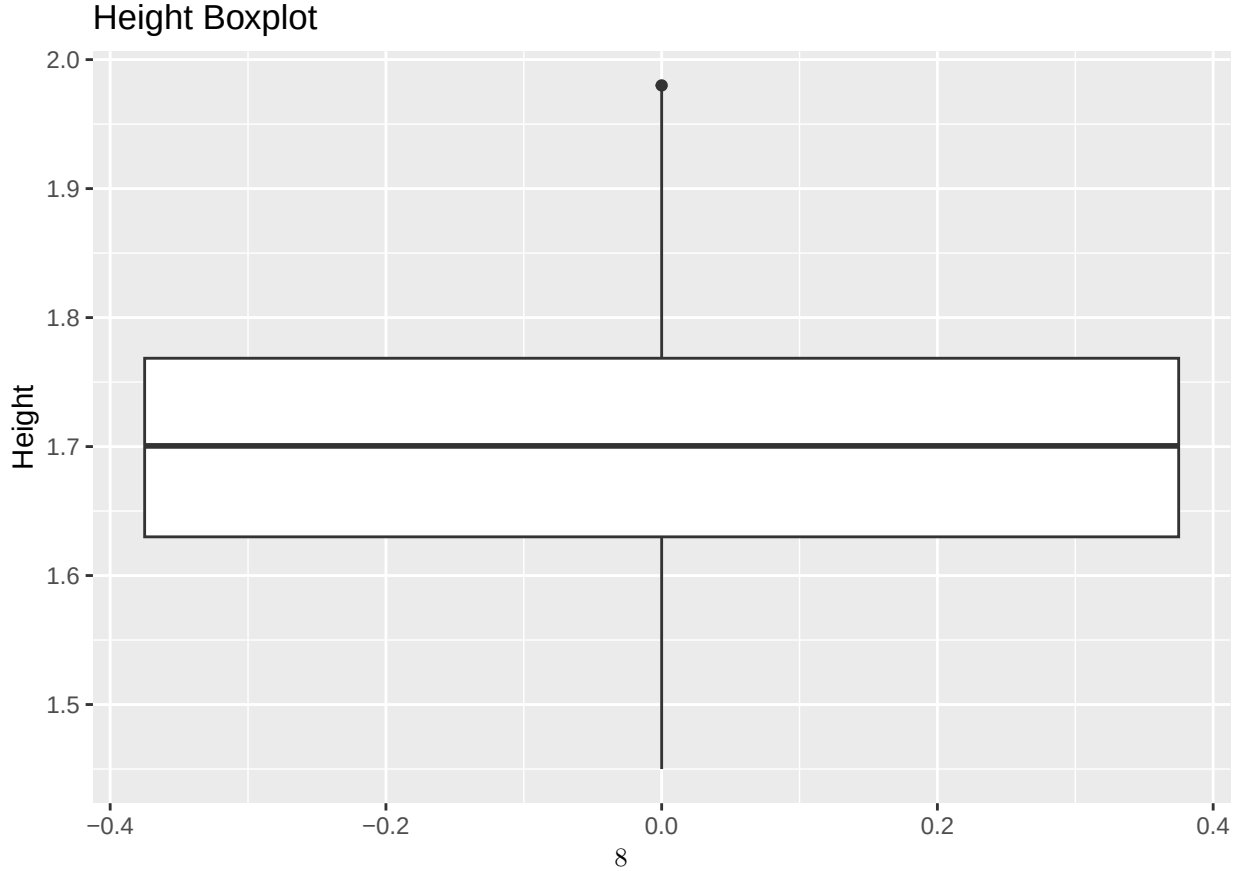
Based on the histogram above, the weight distribution appears multi-modal similar to height distribution, with several peaks indicating that there are multiple groups within the population that have different average weights. The tallest peak around the 50-60 range suggests that a significant number of individuals fall within this weight category, making it the most common weight range in the dataset. After this, there is a gradual decrease in frequency as weight increases, with fewer individuals in the higher weight ranges. Importantly, there is a small peak again around the 100 mark, which could indicate another, less populous group with higher weight values. The distribution is slightly right-skewed, meaning there are fewer individuals with weights on the higher end of the scale.

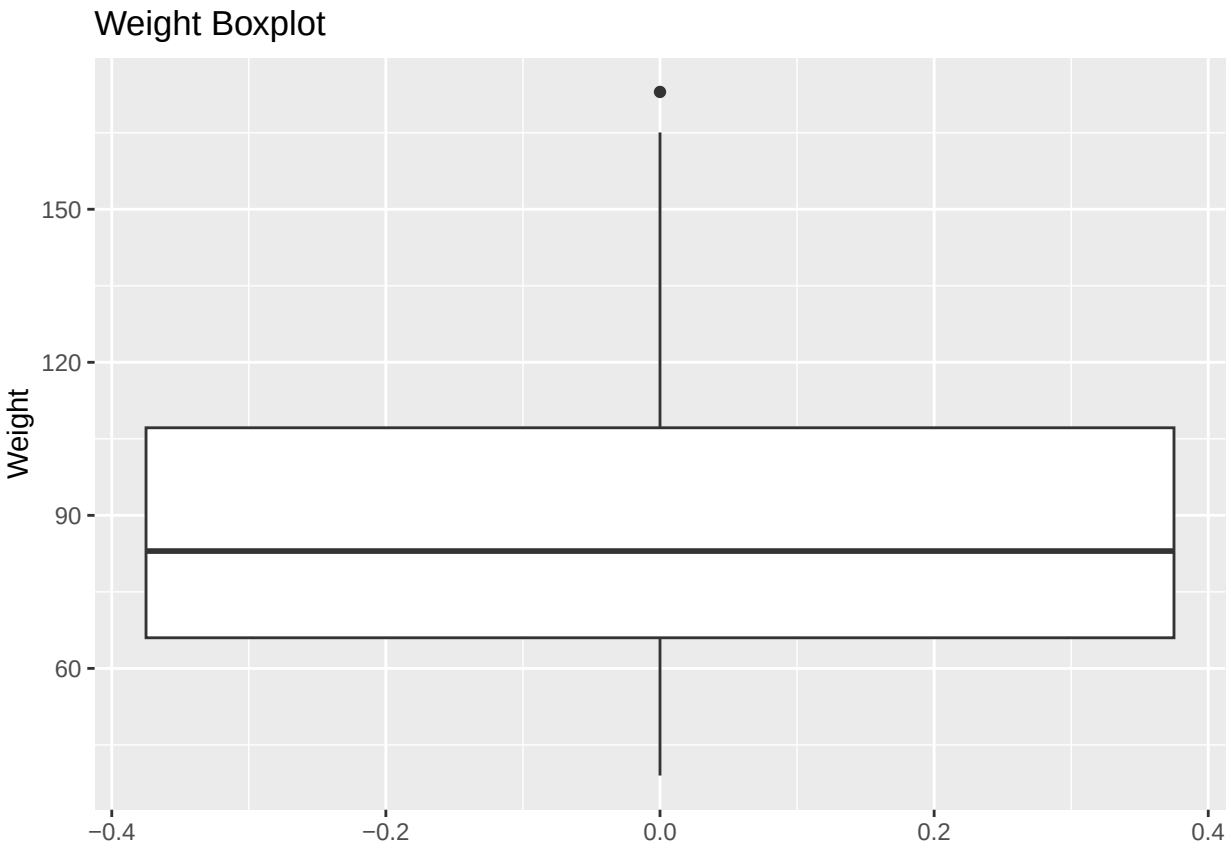


Looking at the Gender ds=istribution barplot, we can see that the data is almost evenly balanced, with the male population being only 1% higher than the female, at around ~ 1077 vs ~1035 females.

Box plots to see outliers:







Examining the age boxplot, I can see that the age distribution among the individuals in this dataset has a median value around the late 20s to early 30s, as indicated by the line within the box. The box itself, which represents the interquartile range, stretches from approximately the early 20s to late 30s, showing where the middle 50% of ages fall. There is a presence of outliers, as shown by the individual points above the upper whisker, which suggests that there are individuals who are significantly older than the rest of the population in the dataset. These outliers are all above the age of 50, which is well outside the upper quartile. The whiskers extend from around 20 years to approximately 50 years, indicating the range within which most of the data falls. Lower outliers are not seen, suggesting that the lower age range is tightly clustered.

Looking at the boxplot for the height distribution, we can first notice only one upper outlier, which is a person who is nearly 2 meters tall. The median height is about 1.7m, shown by the line in the box. The interquartile ranges varies from about 1.63m to about 1.77m and this is where 50% of the population's height lies in. The whiskers show height variation from 1.45m to 1.98m. There are no lower outliers.

Looking at this weight distribution boxplot, I can discern that the median weight is around 90 units, with the middle 50% of the data spread between roughly 70 and 110 units, as indicated by the edges of the box. The whiskers, which represent the range of the data excluding outliers, extend from approximately 60 units to just over 120 units. This suggests that most individuals have weights within this range. There's a notable outlier indicated by the dot above the upper whisker, showing that at least one individual has a weight significantly higher than the rest of the population in the dataset. The distribution of weight within the interquartile range appears to be fairly symmetrical around the median, indicating a relatively even spread of weight values above and below the median. The absence of outliers below the lower whisker suggests there are no individuals with exceptionally low weights compared to the rest of the dataset.

Outlier Analysis:

```
# Function to remove outliers from dataset
remove_outliers <- function(df, cols = names(df)) {
  is_outlier <- function(x) {
    stddev <- sd(x)
    avg <- mean(x)
    lower <- avg - (3 * stddev)
    upper <- avg + (3 * stddev)
    x > upper | x < lower
  }

  for (col in cols) {
    outlier_indices <- is_outlier(df[[col]])
    df <- df[!outlier_indices, ]
  }

  df
}

# Using the function to filter out outliers
obe_without_outliers <- remove_outliers(obesity_data, cols = 1:8)

# Getting the dimensions of the dataset without outliers
dim(obe_without_outliers)
```

```
## [1] 2086   15
```

```
# Getting only the outliers
obesity_only_outliers <- anti_join(obesity_data, obe_without_outliers)
```

```
## Joining with `by = join_by(Age, Height, Weight, FCVC, NCP, CH2O, FAF, TUE,
## Gender, family_history_with_overweight, FAVC, CAEC, CALC, MTRANS, NObeyesdad)`
```

```
dim(obesity_only_outliers)
```

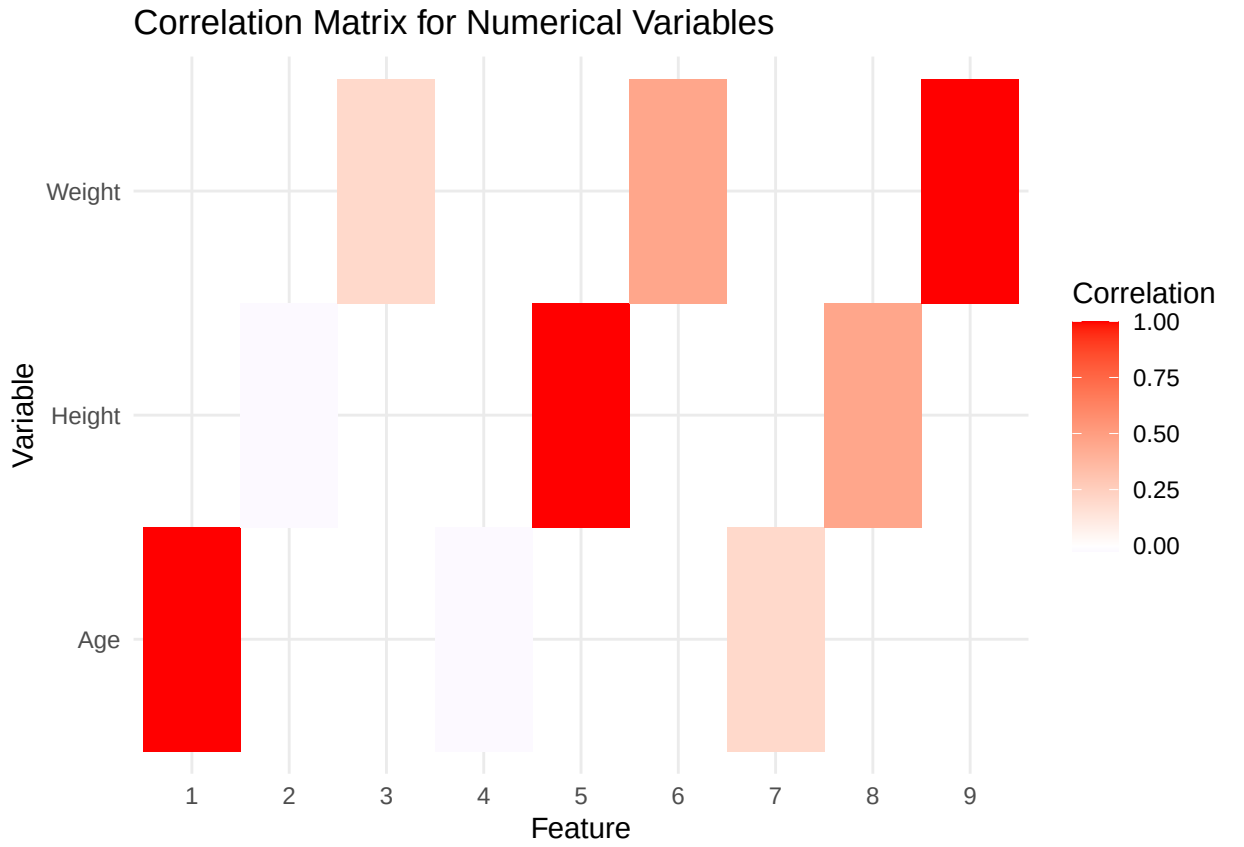
```
## [1] 25 15
```

We can see by implementing a function to determine outliers and removing them, with a standard default threshold value of 3 standard deviations from the mean, there are only about 25 outliers in the dataset.

Correlation:

```
# Correlation Analysis
correlation_matrix <- obesity_data %>%
  select(Age, Height, Weight) %>%
  cor()

# Correlation matrix to long format using pivot_longer
```

```r
correlation_long <- as.data.frame(correlation_matrix) %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Correlation") %>%
  mutate(Feature = rownames(.))
```



The above image is a correlation matrix generated for weight, height, and age. The correlation matrix shows us that there is a positive correlation between all three variables, with the strongest correlation being between weight and height (0.75). This means that people who are taller tend to be heavier and people who are shorter tend to be lighter. The correlation between weight and age is also positive (0.50), meaning that older people tend to be heavier than younger people. This is not always true in every case and that is why it has a correlation of 0.5. However, the correlation between height and age is weaker (0.25), meaning that there is not as strong of a relationship between height and age. Hence, the correlation matrix suggests that there is a relationship between weight, height, and age. But, the relationship is not perfectly linear and we know that there are other factors that influence these variables.

Evaluation of distribution:

```r
# Evaluation of Distribution - skewness:
obesity_data %>% summarise(Skewness_Age = skewness(Age),
                   Skewness_Height = skewness(Height),
                   Skewness_Weight = skewness(Weight))
```

```
##   Skewness_Age Skewness_Height Skewness_Weight
## 1      1.53153     -0.01283638       0.2528764
```

From the skewness distribution we can see that Age is highly skewed right, while the Weight and Height distribution show little skewness (0.25 and -0.012, respectively).

Data Cleaning and Shaping:

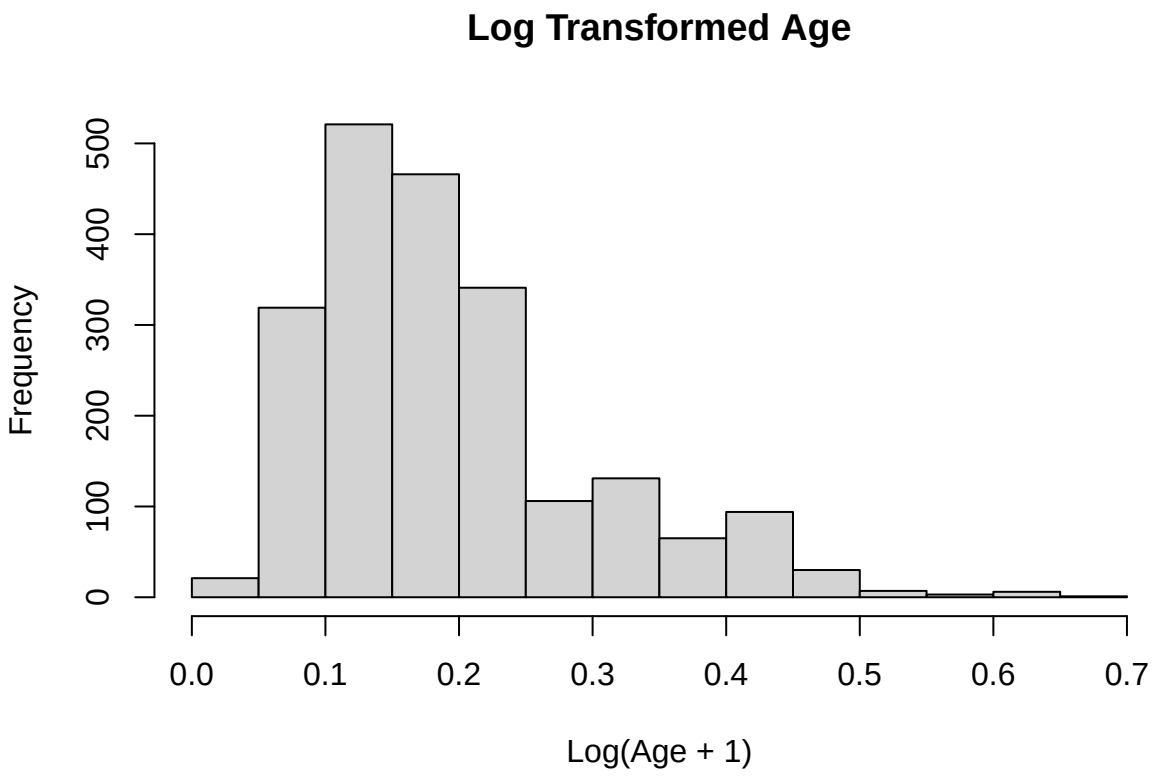
Normalization - Min Max:

```
# Normalization of Feature Values
normalized_obesity_data <- obesity_data

# Data Normalization
normalize <- function(x) {
  (x - min(x))/(max(x) - min(x))
}

normalized_obesity_data[1:8] <- as.data.frame(lapply(obesity_data[1:8], normalize))
```

Min-max method of normalization is applied by creating a simple function to normalize the data.

Feature transformation:

```
# Transformation of Features - Based on skewness analysis
# Applying log transformation to the Age feature
normalized_obesity_data$Age_log <- log(normalized_obesity_data$Age + 1)
# Check the distribution after the transformation
hist(normalized_obesity_data$Age_log, main="Log Transformed Age", xlab="Log(Age + 1)")
```

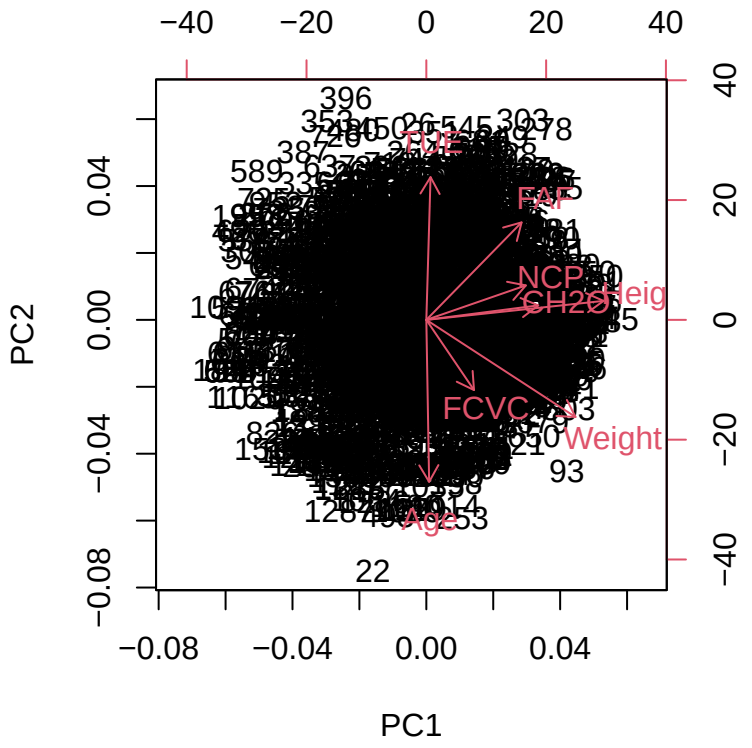



```
# Removing original skewed Age column:
normalized_obesity_data <- subset(normalized_obesity_data, select = -normalized_obesity_data$Age)


# normalized_obesity_data will be used for SVM model training
```

Based on the skewness distribution, we saw that the Age column is highly skewed. Therefore, log transformation of the feature is performed and the resulting distribution is viewed as a histogram. Then, the log transformed feature is kept and the original Age feature is removed in the new normalized dataset, which is then used later for SVM model training. This is because SVM is very sensitive to normalization of the continuous features and also highly skewed feature from the normal could impact the model.

PCA:

```
# Identification of Principal Components (PCA)
# Applying PCA
obesity_data_pca <- prcomp(obesity_data[1:8], center = TRUE, scale. = TRUE)
#summary(obesity_data_pca)
biplot(obesity_data_pca)
```




```
obesity_data_pca <- as.data.frame(obesity_data_pca$x)
```

From the biplot, we can infer that Height and Weight have a strong influence on the principal components, as indicated by the longer vectors pointing in the direction of these variables. These vectors suggest that Height and Weight explain a significant portion of the variance in the dataset. The scatterplot of the points shows how the data is distributed across the first two principal components, with most data points clustered near the center but with some spread along the directions of the Height and Weight vectors. The presence of numerous data points across the plot also indicates that the dataset has a good deal of diversity or variability captured by these two principal components. Another observation from the biplot is the direction of the vectors of Age and TUE(Time using technology devices). They are pointing completely opposite to each other, indicating they are negatively correlated, this means that as the Age of the person increases, the TUE tends to decrease and vice-versa. PCA is performed as a way to reduce dimensionality and see how features are related in a less complex map of relationships.

Feature engineering to create BMI feature:

```
# Feature Engineering: New feature addition
# BMI (Body Mass Index)
obesity_data$BMI <- obesity_data$Weight / (obesity_data$Height^2)
#head(obesity_data)
normalized_obesity_data$BMI <- obesity_data$Weight / (obesity_data$Height^2)
#head(normalized_obesity_data)
```

Feature engineering was done based on domain knowledge to derive a new feature BMI (Body Mass Index) which could prove to be a useful feature and play an important role for prediction.

Model Construction:

Naive Bayes:

```
# Copy of original dataset
obesity_data_NB <- obesity_data
obesity_data_NB$Age_log <- log(obesity_data_NB$Age + 1)

# Data encoding - Binning continuous variables
obesity_data_NB$Age <- as.numeric(cut(obesity_data_NB$Age, c(10,15,20,25,30,35,40,45,50,55,60,65)))
obesity_data_NB$Height <- as.numeric(cut(obesity_data_NB$Height, c(1.40, 1.45, 1.50, 1.55, 1.60, 1.65,
obesity_data_NB$Weight <- as.numeric(cut(obesity_data_NB$Weight, c(30, 40, 50, 60, 70, 80, 90, 100, 110
obesity_data_NB$FCVC <- as.numeric(cut(obesity_data_NB$FCVC, c(0, 1, 2, 3, 4)))
obesity_data_NB$NCP <- as.numeric(cut(obesity_data_NB$NCP, c(0, 1, 2, 3, 4, 5)))
obesity_data_NB$CH2O <- as.numeric(cut(obesity_data_NB$CH2O, c(0, 1, 2, 3, 4)))
obesity_data_NB$FAF <- as.numeric(cut(obesity_data_NB$FAF, c(-1, 0, 1, 2, 3, 4)))
obesity_data_NB$TUE <- as.numeric(cut(obesity_data_NB$TUE, c(-1, 0, 1, 2, 3)))
obesity_data_NB$BMI <- as.numeric(cut(obesity_data_NB$BMI, c(5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55,
obesity_data_NB$Age_log <- as.numeric(cut(obesity_data_NB$Age_log, c(2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8,
```

The continuous columns in the dataset are converted to categorical columns by binning, where bin thresholds are set.

```
set.seed(1000)
# Splitting into training and test set
split_data <- sample(c(rep(0, 0.7 * nrow(obesity_data_NB)), rep(1, 0.3 * nrow(obesity_data_NB))))
obesity_data_NB_train <- obesity_data_NB[split_data == 0,]
obesity_data_NB_validation <- obesity_data_NB[split_data == 1,]
```

Then the binned data columns are split into training and testing datasets. This dataset is used for Naive Bayes as proper data encoding has been performed.

```
# Naive Bayes model construction and prediction
set.seed(1000)
model_NB <- naiveBayes(NObeyesdad~., data = obesity_data_NB_train)
pred_NB <- predict(model_NB, obesity_data_NB_validation)
```

The model is trained using the naiveBayes() function and the predictions are stored in a variable, which will be used later for evaluation.

Support Vector Machines (SVM):

```
# Preparing the data for SVM model by copying to new variable
svm_data <- normalized_obesity_data

# Calculating the minimum and maximum values of the BMI column
```

```r
min_bmi <- min(svm_data$BMI, na.rm = TRUE)
max_bmi <- max(svm_data$BMI, na.rm = TRUE)
# Applying min-max normalization to the BMI column
svm_data$BMI <- (svm_data$BMI - min_bmi) / (max_bmi - min_bmi)


# Converting categorical character variables to factors
categorical_vars <- sapply(svm_data, is.character)
svm_data[categorical_vars] <- lapply(svm_data[categorical_vars], as.factor)


# Splitting the dataset into training and testing sets
set.seed(1000)
svm_split <- sample(c(TRUE, FALSE), nrow(svm_data), replace = TRUE, prob = c(0.7, 0.3))
svm_train <- svm_data[svm_split, ]
svm_test <- svm_data[!svm_split, ]


#sum(is.na(svm_data))
#infinite_values <- sapply(svm_data, function(x) sum(is.infinite(x)))
#infinite_values

# Constructing the SVM model using the svm() with radial kernel
svm_model <- svm(NObeyesdad ~ ., data = svm_train, method = "C-classification", kernel = "radial")
# Making predictions using the SVM model
pred_svm <- predict(svm_model, svm_test)
```

SVM model was constructed using svm() function using the kernel - radial, suited for non-linear relationships.

Decision Tree:

```r
dt_data <- normalized_obesity_data

cat_vars <- sapply(dt_data, is.character)
dt_data[cat_vars] <- lapply(dt_data[cat_vars], as.factor)

# Split the dataset into training and testing sets
set.seed(1000)
train_index <- sample(1:nrow(dt_data), 0.7 * nrow(dt_data))
dt_train_data <- dt_data[train_index, ]
dt_test_data <- dt_data[-train_index, ]

# Construct a decision tree model
decision_tree_model <- rpart(NObeyesdad ~ ., data = dt_train_data, method = "class")

# Make predictions on the test data
pred_dt <- predict(decision_tree_model, newdata = dt_test_data, type = "class")
```

Decision tree model constructed using rpart library, method was explicitly given as class for classification task.

Appropriateness of model:

Appropriateness of the model can be determined by many factors, like even choosing the right classification models for a classification task like this project, various metrics like Precision, F1 scores, viewing confusion matrix statistics and so on. All of these have been performed later on, however, appropriateness can also be checked by using model specific tests as shown below:

```
# Using different kernels to check which performs better - SVM:
svm_linear <- svm(NObeyesdad ~ ., data = svm_train, kernel = 'linear')
pred_svm_linear <- predict(svm_linear, svm_test)
conf_svm_linear <- confusionMatrix(as.factor(pred_svm_linear), svm_test$NObeyesdad)
conf_svm_linear$overall["Accuracy"]
```

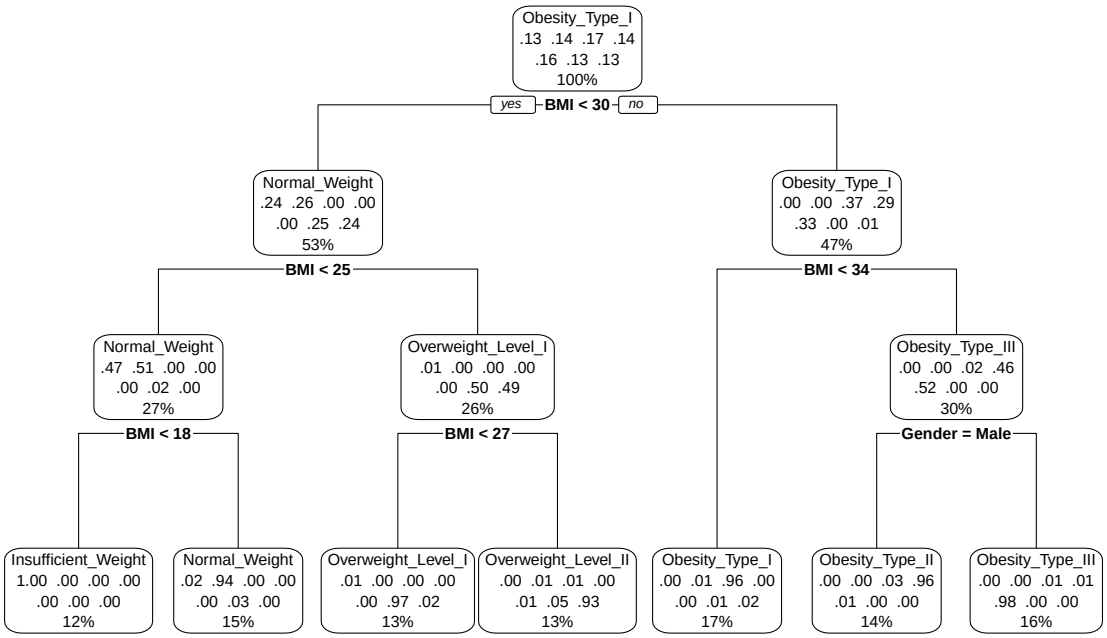
```
##  Accuracy
## 0.9367089
```

```
svm_radial <- svm(NObeyesdad ~ ., data = svm_train, kernel = 'radial')
pred_svm_radial <- predict(svm_radial, svm_test)
conf_svm_radial <- confusionMatrix(as.factor(pred_svm_radial), svm_test$NObeyesdad)
conf_svm_radial$overall["Accuracy"]
```

```
##  Accuracy
## 0.9287975
```

The output of the two models with use of linear and radial kernels show that the linear kernel in the svm model has a higher accuracy. Even then, I have chosen the radial kernel with a slightly lower accuracy. This is because accuracy is only one of the many factors in determining the right model for classification using SVM and there are other important factors to consider. For example, the relationship between consumption of high-calorie food and obesity might not be perfectly linear. There might be thresholds or other factors that influence the relationship. Also, the influence of transportation mode on obesity/overweight nature of the person is more complex to determine and not easily captured by a simple linear equation. Therefore, the radial kernel which is suited for non-linear relationships, while being more robust in handling unknonw data is chosen as appropriate model.

```
# Tree pruning to avoid overfitting - Decision Tree:
tree_model <- rpart(NObeyesdad ~ ., data = dt_train_data, method = "class")
pruned_tree <- prune(tree_model, cp = tree_model$cptable[which.min(tree_model$cptable[,"xerror"]),"CP"])
rpart.plot(pruned_tree, box.palette = 0)
```



The pruned decision tree shows that the BMI as the primary predictor for categorizing weight status, ranging from normal weight to various obesity levels. Therefore, feature enginerring a BMI feature proved to be quite useful in the predictions. The tree suggests that individuals with a BMI below 25 are generally classified as normal weight, those with a BMI between 25 and 27 are considered overweight, and a BMI over 30 leads to different classifications of obesity. Also, a BMI over 34 coupled with male gender indicates a higher likelihood of being classified as 'Obesity_Type_III'. The probabilities and percentages at each leaf node indicate the certainty of the model about each classification and the proportion of samples from the trained data within each category. The pruned tree shows us that the model gives a clear set of rules, with BMI serving as a key indicator for weight classification, and also we can infer gender to be a secondary predictor in distinguishing obesity types. The pruned tree mainly also can be indicator or model appropriateness if we observe the certainty of model and the percentages of samples. The percentages of samples in each leaf node does appear to be low, but it is mostly evenly divided, and not too low in one of the nodes and too high in others. The certainty values (or percentages) should also show mostly one class to be highly certain, thereby indicating that the model is predicting each class with high certainty. We can see that this is true, since all leaf nodes have one certainty value very close to 1 and others negligibly small. This effectibvely shows us that the chosen decision tree model is appropriate for the dataset used for classification task.

Model Evaluation:

Confusion matrices for each model:

```
# Confusion matrix for Naive Bayes model:
conf_NB <- confusionMatrix(as.factor(pred_NB), as.factor(obesity_data_NB_validation$NObeyesdad))
conf_NB$overall["Accuracy"]
```

```
##  Accuracy
## 0.8120063
```

```
conf_NB_tab <- table(obesity_data_NB_validation$NObeyesdad, pred_NB)
```

```
# Confusion matrix for SVM model:
conf_svm <- confusionMatrix(as.factor(pred_svm), svm_test$NObeyesdad)
conf_svm$overall["Accuracy"]
```

```
##  Accuracy
## 0.9287975
```

```
conf_svm_tab <- table(svm_test$NObeyesdad, pred_svm)
```

```
# Confusion matrix for Decision Tree model:
conf_dt <- table(dt_test_data$NObeyesdad, pred_dt)
# Accuracy of Decisionn Tree model:
accuracy_dt_mod <- sum(diag(conf_dt))/sum(conf_dt)
print(paste("Accuracy:", accuracy_dt_mod))
```

```
## [1] "Accuracy: 0.963722397476341"
```

The above evaluation metrics show accuracies from the confusion matrices from each of the models.

Comparison of models and interpretation - By use of Precision, Recall and F1 scores:

Metrics such as Precision, Recall and F1 score have been calculated by using function.

```
##
## Insufficient_Weight       Normal_Weight      Obesity_Type_I     Obesity_Type_II
##                 272                 287                 351                 297
##    Obesity_Type_III  Overweight_Level_I Overweight_Level_II
##                 324                 290                 290
```

Since the instances in each of the classes for the target variable is mostly equivalent, a simple average of the scores for each metric can be taken to assess model performance and evaluation.

```
## [1] "Naive Bayes - Overall Average Scores:"
```

```
## $`Overall Average Precision`
## [1] 0.8309729
##
## $`Overall Average Recall`
## [1] 0.8130763
##
## $`Overall Average F1 Score`
## [1] 0.8091725
```

```
## [1] "Support Vector Machines - Overall Average Scores:"
```

```
## $`Overall Average Precision`
## [1] 0.9269402
##
## $`Overall Average Recall`
## [1] 0.9245463
##
## $`Overall Average F1 Score`
## [1] 0.9247995
```

```
## [1] "Decision Tree - Overall Average Scores:"
```

```
## $`Overall Average Precision`
## [1] 0.9652871
##
## $`Overall Average Recall`
## [1] 0.9644454
##
## $`Overall Average F1 Score`
## [1] 0.9645526
```

The performance of the Naive Bayes, Support Vector Machines (SVM), and Decision Tree models in a multi-class classification task shows differences in their predictive capabilities. The Naive Bayes model demonstrates a good level of precision (83.09%) and recall (81.3%), with an F1 Score of 80.91%. These figures suggest that while the model is fairly accurate and reliable in predicting the correct class, there is some scope for improvement, especially in comparison to the other models. The SVM model exhibits much higher performance, achieving high precision (92.69%) and recall (92.45%), alongside an impressive F1 Score of 92.48%. These scores indicate that the SVM model is highly effective in making accurate class predictions with a low rate of false positives and a strong ability to detect most positive instances. Among the three, the Decision Tree model stands out with the highest metrics, marking it as the most effective model in this analysis. It achieves a very high precision of 96.53% and a recall of 96.44%, therefore resulting in an F1 Score of 96.46%. This great performance suggests that the Decision Tree model is highly capable at correctly classifying instances with minimal false positives and negatives.

K-fold cross-validation:

```
# Average accuracy from the cross-validation test _ SVM:
avg_accuracy <- mean(sapply(cv_results_svm, function(x) x$overall['Accuracy']))
print(avg_accuracy)
```

```
## [1] 0.9332
```

From the k-fold cross-validation test on the SVM model, we see an average accuracy of 93.3%. This is a high accuracy level and suggests a robust model especially as it is performing well on unseen data. The slight increase in accuracy from cross-validation compared to test data from before (92.8%) suggests that when the model is trained on different subsets of data, it does not suffer from a significant performance drop. This indicates good generalization, which is preferred over a model that might have been overly optimized for a single test set.

```
# Average accuracy from the cross-validation test - Naive Bayes:
avg_accuracy_NB <- mean(sapply(cv_results_NB, function(x) x$overall['Accuracy']))
print(avg_accuracy_NB)
```

```
## [1] 0.832781
```

From the k-fold cross-validation test on the Naive Bayes model, we see an average accuracy of 83.28%. This is a reasonably good accuracy level and suggests that its a good model especially as it is performing well on unseen data. Because the accuracy is lower than SVM, this model should not be considered as inaccurate or less reliable, sicne this algorithm is well suited for classification and accuracy is not the only factor.

```
# Average accuracy from the cross-validation test:
avg_accuracy_DT <- mean(sapply(cv_results_DT, function(x) x$overall['Accuracy']))
print(avg_accuracy_DT)
```

```
## [1] 0.9611598
```

The cross-validation output showing an average accuracy of 96.11% for the decision tree model indicates a high level of predictive performance. This result implies that when the decision tree is trained and tested on different subsets of the dataset, it consistently makes correct predictions on the test folds. A high average accuracy across folds, like 96.11%, suggests that the decision tree model is very effective at classifying the correct level of obesity and is likely to generalize well to unseen data. This high level of accuracy also indicates that the decision tree has captured the underlying patterns in the dataset without overfitting, as the process of cross-validation helps to reduce the overfitting risk by evaluating the model on multiple independent test sets.

Failure analysis of the models:

```
##                       NB_predicted_failures
## NB_actual_failures    Insufficient_Weight Normal_Weight Obesity_Type_I
##   Insufficient_Weight                   0             2              0
##   Normal_Weight                        15             0              2
##   Obesity_Type_I                        0             0              0
##   Obesity_Type_II                       0             0              0
##   Obesity_Type_III                      0             0              0
##   Overweight_Level_I                    0             7              0
##   Overweight_Level_II                   0             1              2
##                       NB_predicted_failures
## NB_actual_failures    Obesity_Type_II Obesity_Type_III Overweight_Level_I
##   Insufficient_Weight               0                0                  0
##   Normal_Weight                     0                0                  0
##   Obesity_Type_I                    1               13                  0
##   Obesity_Type_II                   0               26                  0
##   Obesity_Type_III                  0                0                  0
##   Overweight_Level_I                0                0                  0
##   Overweight_Level_II               0                6                 10
##                       NB_predicted_failures
## NB_actual_failures    Overweight_Level_II
##   Insufficient_Weight                   0
##   Normal_Weight                         0
##   Obesity_Type_I                        1
##   Obesity_Type_II                       0
##   Obesity_Type_III                      0
##   Overweight_Level_I                   33
##   Overweight_Level_II                   0
```

The confusion matrix from the Naive Bayes model failure analysis shows slight difficulties in distinguishing between certain weight categories. 'Obesity_Type_II' is often misclassified as 'Obesity_Type_III', and 'Overweight_Level_I' is frequently mistaken for 'Overweight_Level_II'. These errors suggest that the model confuses categories that are adjacent to each other, likely due to overlapping features. However, categories like 'Insufficient_Weight' and 'Obesity_Type_III' show clear definition with no misclassifications, indicating distinct separation from other categories.

```
##                      svm_predicted_failures
## svm_actual_failures   Insufficient_Weight Normal_Weight Obesity_Type_I
##   Insufficient_Weight                   0             0              0
##   Normal_Weight                         4             0              3
##   Obesity_Type_I                        0             0              0
##   Obesity_Type_II                       0             0              2
##   Obesity_Type_III                      0             0              0
##   Overweight_Level_I                    0             7              2
##   Overweight_Level_II                   0             0              2
##                      svm_predicted_failures
## svm_actual_failures   Obesity_Type_II Obesity_Type_III Overweight_Level_I
##   Insufficient_Weight               0                0                  0
##   Normal_Weight                     0                0                 12
##   Obesity_Type_I                    1                0                  0
##   Obesity_Type_II                   0                0                  0
##   Obesity_Type_III                  1                0                  0
##   Overweight_Level_I                0                0                  0
##   Overweight_Level_II               0                0                  4
##                      svm_predicted_failures
## svm_actual_failures   Overweight_Level_II
##   Insufficient_Weight                   0
##   Normal_Weight                         0
##   Obesity_Type_I                        4
##   Obesity_Type_II                       0
##   Obesity_Type_III                      0
##   Overweight_Level_I                    3
##   Overweight_Level_II                   0
```

The SVM model confusion matrix shows it mainly confuses 'Normal_Weight' with 'Overweight_Level_I' and 'Obesity_Type_I' with 'Overweight_Level_II'. The clear separation of 'Insufficient_Weight' and 'Obesity_Type_III' from other classes, with no misclassifications, indicates clear and distinct features for these categoriess. However, the overlap in predictions for categories that are close to each other suggests that more discrimination should be introduced in the model so it can better decide between differences.

```
##                      dt_predicted_failures
## dt_actual_failures    Insufficient_Weight Normal_Weight Obesity_Type_I
##   Insufficient_Weight                   0             0              0
##   Normal_Weight                         0             0              2
##   Obesity_Type_I                        0             0              0
##   Obesity_Type_II                       0             0              3
##   Obesity_Type_III                      0             0              0
##   Overweight_Level_I                    0             3              0
##   Overweight_Level_II                   0             0              5
##                      dt_predicted_failures
## dt_actual_failures    Obesity_Type_II Obesity_Type_III Overweight_Level_I
##   Insufficient_Weight               0                0                  0
```

```
##   Normal_Weight                    0                1                0
##   Obesity_Type_I                   3                0                0
##   Obesity_Type_II                  0                0                0
##   Obesity_Type_III                 0                0                0
##   Overweight_Level_I               0                0                0
##   Overweight_Level_II              0                0                2
##                       dt_predicted_failures
## dt_actual_failures    Overweight_Level_II
##   Insufficient_Weight                   0
##   Normal_Weight                         0
##   Obesity_Type_I                        0
##   Obesity_Type_II                       0
##   Obesity_Type_III                      0
##   Overweight_Level_I                    4
##   Overweight_Level_II                   0
```

From the output above showing the decision tree model confusion matrix, we see that there are very few misclassifications. Mainly the model misclassifies 'Overweight_Level_II' and 'Obesity_Type_I' in certain instances and 'Overweight_Level_II' vs 'Overweight_Level_I'. But overall, the model is very capable at distinguishing classes, even the ones that are close to each other in terms of their differences.

Model Tuning & Performance Improvement:

Hyperparameter tuning:

```
# Naive Bayes model with Laplace smoothing
set.seed(1000)
model_NB_hp <- naiveBayes(NObeyesdad ~ ., data = obesity_data_NB_train, laplace = 1)
pred_NB_hp <- predict(model_NB_hp, obesity_data_NB_validation)
conf_NB_hp <- confusionMatrix(as.factor(pred_NB_hp), as.factor(obesity_data_NB_validation$NObeyesdad))
conf_NB_hp$overall["Accuracy"]
```

Naive Bayes:

```
##  Accuracy
## 0.7835703
```

As we can see, by adding a laplace = 1 for Laplace smoothing in order to do hyperparamter tuning (considering this is a small dataset) does not seem to be that effective and there is not much difference in accuracy, even with other values for laplace parameter.

```
# Extended hyperparameter tuning for SVM
set.seed(1000)
c_values <- c(0.1, 1, 10)
gamma_values <- c(0.001, 0.01, 0.1)

# Store results
```

```r
svm_results_extended <- list()

for (C in c_values) {
  for (gamma in gamma_values) {
    svm_model_extended <- svm(NObeyesdad ~ ., data = svm_train, cost = C, gamma = gamma, kernel = "radia
    pred_svm_extended <- predict(svm_model_extended, svm_test)
    conf_svm_extended <- confusionMatrix(as.factor(pred_svm_extended), svm_test$NObeyesdad)

    # Store results
    svm_results_extended[[paste("C=", C, "gamma=", gamma)]] <- conf_svm_extended
  }
}

# Different combinations of C and gamma
lapply(svm_results_extended, function(x) x$overall["Accuracy"])
```

SVM:

```
## $`C= 0.1 gamma= 0.001`
##  Accuracy
## 0.1693038
##
## $`C= 0.1 gamma= 0.01`
##  Accuracy
## 0.5474684
##
## $`C= 0.1 gamma= 0.1`
## Accuracy
## 0.846519
##
## $`C= 1 gamma= 0.001`
##  Accuracy
## 0.5411392
##
## $`C= 1 gamma= 0.01`
##  Accuracy
## 0.8623418
##
## $`C= 1 gamma= 0.1`
##  Accuracy
## 0.9351266
##
## $`C= 10 gamma= 0.001`
##  Accuracy
## 0.8655063
##
## $`C= 10 gamma= 0.01`
##  Accuracy
## 0.9382911
##
## $`C= 10 gamma= 0.1`
##  Accuracy
## 0.9446203
```

From the output, we can observe a general trend that increasing the values of both C and gamma leads to higher accuracy scores. The lowest accuracy is at C=0.1 and gamma=0.001, suggesting underfitting, where the model is too simple to capture the patterns. As C and gamma increase, the model becomes more complex and fits the data better, shown by the highest accuracy observed at C=10 and gamma=0.1. Overfitting of the data is also possible at the highest accuracy level. The best hyperparameters seem to be where C=10 and gamma is either 0.01 or 0.1, indicating a robust model with strong predictive power.

```
# Extended hyperparameter tuning for Decision Trees - use of cp and maxdepth parameters:
cp_values <- c(0.01, 0.05)
maxdepth_values <- c(3, 5, 10)

dt_results_extended <- list()

for (cp in cp_values) {
  for (maxdepth in maxdepth_values) {
    decision_tree_model_extended <- rpart(NObeyesdad ~ ., data = dt_train_data, method = "class", cp = 
    pred_dt_extended <- predict(decision_tree_model_extended, dt_test_data, type = "class")
    conf_dt_extended <- confusionMatrix(as.factor(pred_dt_extended), dt_test_data$NObeyesdad)
    dt_results_extended[[paste("cp=", cp, "maxdepth=", maxdepth)]] <- conf_dt_extended
  }
}

# Different combinations of the parameters:
lapply(dt_results_extended, function(x) x$overall["Accuracy"])
```

Decision Tree:

```
## $`cp= 0.01 maxdepth= 3`
##  Accuracy
## 0.9637224
##
## $`cp= 0.01 maxdepth= 5`
##  Accuracy
## 0.9637224
##
## $`cp= 0.01 maxdepth= 10`
##  Accuracy
## 0.9637224
##
## $`cp= 0.05 maxdepth= 3`
##  Accuracy
## 0.9637224
##
## $`cp= 0.05 maxdepth= 5`
##  Accuracy
## 0.9637224
##
## $`cp= 0.05 maxdepth= 10`
##  Accuracy
## 0.9637224
```

The output from the hyperparameter tuning of the decision tree model reveals that varying the complexity parameter and maximum depth has no significant impact on the accuracy, which remains constant at 0.9637224 across all combinations of these hyperparameters. This consistency suggests that the decision tree model is quite robust to changes in these parameters within the tested ranges. It could indicate that the model has reached a point of optimal complexity for the given data, where further increasing the depth or decreasing the complexity parameter does not yield any additional benefit in terms of accuracy. This result might also imply that the model performance is more influenced by other factors in the data rather than the depth and complexity of the tree itself.

Regularization:

Regularization is an important step in model tuning and improvement, especially for the models, SVM and Decision Trees, where overfitting is one of the problems. Regularization techniques simplify the model, thus preventing overfitting and improving the generalization ability of the model.

```
# Results for different values of C
lapply(svm_results, function(x) x$overall["Accuracy"])
```

For SVM:

```
## $`0.1`
##  Accuracy
## 0.7689873
##
## $`1`
##  Accuracy
## 0.9287975
##
## $`10`
## Accuracy
## 0.943038
##
## $`100`
## Accuracy
## 0.931962
##
## $`1000`
##  Accuracy
## 0.9303797
```

The output from regularization of the SVM model with varying C values indicates that as C increases from 0.1 to 10, the accuracy improves, peaking at C=10 with an accuracy of 0.943038. However, further increasing C to 100 and 1000 leads to a slight decrease in accuracy. This pattern suggests that increasing the regularization strength up to a certain point (C=10) helps the model perform better by finding a good balance between fitting the training data and maintaining generalizability. However, beyond this optimal point, excessive regularization (higher C values) does not improve the performance and starts to negatively impact it, most probably because of overfitting to the training data.

```r
# Results for different values of cp
lapply(dt_results, function(x) x$overall["Accuracy"])
```

For Decision Tree:

```
## $`0.001`
##  Accuracy
## 0.9637224
##
## $`0.005`
##  Accuracy
## 0.9637224
##
## $`0.01`
##  Accuracy
## 0.9637224
##
## $`0.02`
##  Accuracy
## 0.9637224
##
## $`0.05`
##  Accuracy
## 0.9637224
```

Like before with hyperparameter tuning for decision tree, we see a similar result here, as the accuracy remains the same for various values of cp. This is most likely because the complexity parameter has a very little impact on the model performance.

Bagging with homogenous learners:

```r
set.seed(1000)
# Bagging using decision trees
bagging_model <- train(NObeyesdad ~ ., data = dt_train_data, method = "treebag")
bagging_predictions <- predict(bagging_model, dt_test_data)
bagging_conf_matrix <- confusionMatrix(bagging_predictions, dt_test_data$NObeyesdad)
print(bagging_conf_matrix$overall["Accuracy"])
```

```
##  Accuracy
## 0.9700315
```

The 97% accuracy achieved with the bagging model using decision trees indicates a high level of predictive accuracy. Bagging, which stands for Bootstrap Aggregating, involves creating multiple decision trees which are homogeneous learners on different subsets of the training data and then aggregating their predictions. This method reduces the variance of the model, thereby decreasing the risk of overfitting and improving generalizability. The high accuracy score suggests that this ensemble approach effectively captures the underlying patterns in the dataset, leading to more accurate predictions on the test data. The improvement in accuracy compared to a single decision tree model highlights the benefit of using ensemble techniques like bagging to boost the performance of machine learning models.

Heterogenous Ensemble model as a function:

```
# Function to create and evaluate a heterogeneous ensemble model
heterogeneous_ensemble <- function(train_data, test_data, new_data, target_column) {

  # Training individual models
  model_NB <- naiveBayes(reformulate(termlabels = names(train_data)[!names(train_data) %in% target_colu

  svm_model <- svm(reformulate(termlabels = names(train_data)[!names(train_data) %in% target_column], re

  decision_tree_model <- rpart(reformulate(termlabels = names(train_data)[!names(train_data) %in% targe

  # Predictions:
  predictions_NB <- predict(model_NB, test_data)
  predictions_svm <- predict(svm_model, test_data)
  predictions_dt <- predict(decision_tree_model, newdata = test_data, type = "class")
  # Combining predictions::
  combined_predictions <- data.frame(predictions_NB, predictions_svm, predictions_dt)
  final_predictions <- apply(combined_predictions, 1, function(x) names(sort(table(x), decreasing = TRU
  # Confusion matrix
  conf_ensemble <- confusionMatrix(as.factor(final_predictions), test_data[[target_column]])

  # Prediction for new data
  new_data_predictions_NB <- predict(model_NB, new_data)
  new_data_predictions_svm <- predict(svm_model, new_data)
  new_data_predictions_dt <- predict(decision_tree_model, newdata = new_data, type = "class")

  comb_nd_pred <- data.frame(new_data_predictions_NB, new_data_predictions_svm, new_data_predictions_dt
  final_nd_pred <- apply(comb_nd_pred, 1, function(x) names(sort(table(x), decreasing = TRUE))[1])

  return(list(confusion_matrix = conf_ensemble, new_data_predictions = final_nd_pred))
}
```

The above chuck is a heterogeneous ensemble model function combining predictions from all the three models based on majority voting system.

Comparison of ensemble with individual models

```
ensemble_results <- heterogeneous_ensemble(dt_train_data, dt_test_data, dt_test_data, "NObeyesdad")
conf_ensemble <- ensemble_results$confusion_matrix

# Print out accuracy for comparison
cat("Naive Bayes Accuracy:", conf_NB$overall["Accuracy"], "\n")
```

```
## Naive Bayes Accuracy: 0.8120063
```

```
cat("SVM Accuracy:", conf_svm$overall["Accuracy"], "\n")
```

```
## SVM Accuracy: 0.9287975
```

```r
cat("Decision Tree Accuracy:", accuracy_dt_mod, "\n")
```

```
## Decision Tree Accuracy: 0.9637224
```

```r
cat("Heterogeneous Ensemble Accuracy:", conf_ensemble$overall["Accuracy"], "\n")
```

```
## Heterogeneous Ensemble Accuracy: 0.9605678
```

Ultimately at the end of the analysis, comparing just the accuracies of all the models, we see that Naive Bayes performs reasonably good with 81% accuracy, while the other two models, SVM and Decision Tree perform significantly better at 93% and 96.37% respectively. The heterogeneous ensemble function built does not seem to impact the accuracy further, giving us accuracy of 96.05%. While the acuracies do show whether a model is performing well in predicting a certain target, it is not always the conclusive determining factor in choosing the model to go ahead with any insights and implementation later on. For example, Naive Bayes shows only 81% accuracy and F1 score of 0.809, which is only reasonably good compared to the other two models, but it can still be used in cases where simplicity of the model might be required, in which case, we do not need to dismiss the Naive Bayes model in this example.

Application of ensemble model for new random data

```r
# Applying ensemble model to new data
ensemble_prediction_results <- heterogeneous_ensemble(dt_train_data, dt_test_data, new_data, "NObeyesda

# New data predictions:
new_data_predictions <- ensemble_prediction_results$new_data_predictions
new_data_predictions
```

```
##                     1                     2                     3
## "Insufficient_Weight"       "Normal_Weight"       "Normal_Weight"
##                     4                     5                     6
##       "Normal_Weight"       "Normal_Weight"       "Normal_Weight"
##                     7                     8                     9
##       "Normal_Weight"       "Normal_Weight"       "Normal_Weight"
##                    10                    11                    12
##       "Normal_Weight"       "Normal_Weight"       "Normal_Weight"
##                    13                    14                    15
##       "Normal_Weight" "Insufficient_Weight" "Insufficient_Weight"
```

The output from applying the ensemble model to the new data indicates that, of the 15 new instances, the model predicts that the majority fall into the 'Normal_Weight' category, with 13 out of 15 instances classified as such. In contrast, only 2 instances are predicted to be 'Insufficient_Weight'. This suggests that, based on the features provided in the new data the ensemble model predominantly identifies these instances as falling within the normal weight range. The presence of 'Insufficient_Weight' predictions also shows the model ability to identify different weight statuses.

References:

Obesity or CVD risk (Classify/Regressor/Cluster). (2023, November 20). Kaggle. https://www.kaggle.com/datasets/aravindpcoder/obesity-or-cvd-risk-classifyregressorcluster/