# Eating Habits and Physical Activities Effects on Obesity

Statistical Learning Project

Ngoc Diem Le (2009466) Marina Vicini (2021116)

10 July 2021

## Contents



# 1 Introduction

Obesity is one of the world's largest health problems. According to the World Health Organization, since 1975, obesity has nearly tripled worldwide [1]. In 2016, 13% of adults aged 18 years and over were obese. It is an alarming problem, because it is responsible for 4.7 million premature deaths each year [2]. The fundamental cause of obesity and overweight is an energy imbalance between calories consumed and calories expended. Therefore, the health impact of eating a healthy diet and being physically active cannot be underestimated. The goal of this project is to indicate which eating habit and physical condition variables are most related to obesity levels by evaluating on the obesity dataset.

# 2 Dataset

The dataset used is from UC Irvine's Machine Learning Repository [5] which contains 2111 records and 17 attributes for the estimation of obesity levels in individuals age 14 to 61 from Mexico, Peru and Colombia,

based on their eating habits and physical condition [3]. 77% of the data was generated synthetically using the Weka tool and the SMOTE filter, 23% of the data was collected directly from users through a web platform. The records are labeled with the class variable NObesity (Obesity level) which divides into 7 categories: Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III.

## 2.1 Data Features

The attributes related with eating habits are: Frequent consumption of high caloric food (FAVC), Frequency of consumption of vegetables (FCVC), Number of main meals (NCP), Consumption of food between meals (CAEC), Consumption of water daily (CH2O), and Consumption of alcohol (CALC). The attributes related with the physical condition are: Calories consumption monitoring (SCC), Physical activity frequency (FAF), Time using technology devices (TUE), Transportation used (MTRANS), other variables obtained were: Gender, Age, Height, Weight, Family history with overweight (family_history_with_overweight).

| Variable | Meaning |
|---|---|
| Gender | User's sex |
| Age | User's age |
| Height | Height in m |
| Weight | Weight in kg |
| FAVC | Frequent consumption of high caloric food |
| FCVC | Frequency of consumption of vegetables |
| NCP | Number of main meals |
| CAEC | Consumption of food between meals |
| SMOKE | If the user smokes |
| CH2O | Consumption of water daily |
| SCC | Calories consumption monitoring |
| FAF | Physical activity frequency |
| TUE | Time using technology devices |
| CALC | Consumption of alcohol |
| MTRANS | Transportation used |

```
# Overview the dataset
head(df)
```

```
##   Gender Age Height Weight family_history_with_overweight FAVC FCVC NCP
## 1 Female  21   1.62   64.0                             yes   no    2   3
## 2 Female  21   1.52   56.0                             yes   no    3   3
## 3   Male  23   1.80   77.0                             yes   no    2   3
## 4   Male  27   1.80   87.0                              no   no    3   3
## 5   Male  22   1.78   89.8                              no   no    2   1
## 6   Male  29   1.62   53.0                              no  yes    2   3
##        CAEC SMOKE CH2O SCC FAF TUE       CALC                MTRANS
## 1 Sometimes    no    2  no   0   1         no Public_Transportation
## 2 Sometimes   yes    3 yes   3   0  Sometimes Public_Transportation
## 3 Sometimes    no    2  no   2   1 Frequently Public_Transportation
## 4 Sometimes    no    2  no   2   0 Frequently              Walking
## 5 Sometimes    no    2  no   0   0  Sometimes Public_Transportation
## 6 Sometimes    no    2  no   0   0  Sometimes            Automobile
##            NObeyesdad
## 1       Normal_Weight
## 2       Normal_Weight
## 3       Normal_Weight
```

```
## 4  Overweight_Level_I
## 5 Overweight_Level_II
## 6       Normal_Weight
```

## 2.2 Data cleaning and preprocessing

For simplicity, target variable NObesity was changed into Obesity and the variable family_history_with_overweight was changed into family_history.

```
# Rename column NObeyesdad to Obesity
names(df)[names(df) == 'NObeyesdad'] <- 'Obesity'

# Rename column family_history_of_obesity to Family_history
names(df)[names(df) == 'family_history_with_overweight'] <- 'Family_history'

# Check the final column names
colnames(df)
```

```
##  [1] "Gender"         "Age"            "Height"         "Weight"
##  [5] "Family_history" "FAVC"           "FCVC"           "NCP"
##  [9] "CAEC"           "SMOKE"          "CH2O"           "SCC"
## [13] "FAF"            "TUE"            "CALC"           "MTRANS"
## [17] "Obesity"
```

Body mass index (BMI) is a value derived from the mass (weight) and height of a person which is widely used as a risk factor for the prevalence of several health issues, especially obesity.

$$BMI = \frac{Weight}{Height * Height}$$

This index has been used to classify the user's obesity status (only for the not synthetic data) [2].

WHO and the Mexican normativity classify as such the BMI (kg/m$^2$) [1, 4]:

- Underweight: $BMI < 18.5$
- Normal: $18.5 < BMI < 24.9$
- Overweight: $25.0 < BMI < 29.9$
- Obesity I: $30.0 < BMI < 34.9$
- Obesity II: $35.0 < BMI < 39.9$
- Obesity III: $BMI > 40$

We also create a new binary variable, that classifies between obesity and not, called Obesity_binary.

```
# Create a new column named BMI which calculates the BMI index
df$BMI = round(df$Weight/(df$Height)**2, 2)

# Categorize obesity level based on BMI with BMI > 30 is Obesity (1)
#and BMI <= 30 is Non-Obesity (0)
for (i in 1:nrow(df)){
  if (df$BMI[i] <= 30){
    df$Obesity_binary[i] = 'Not Obese' #Non-obesity
  } else {
    df$Obesity_binary[i] = 'Obese' #Obesity
  }
}

head(df)
```

```
##   Gender Age Height Weight Family_history FAVC FCVC NCP      CAEC SMOKE CH2O
```

```
## 1 Female  21   1.62   64.0            yes   no    2   3 Sometimes    no    2
## 2 Female  21   1.52   56.0            yes   no    3   3 Sometimes   yes    3
## 3   Male  23   1.80   77.0            yes   no    2   3 Sometimes    no    2
## 4   Male  27   1.80   87.0             no   no    3   3 Sometimes    no    2
## 5   Male  22   1.78   89.8             no   no    2   1 Sometimes    no    2
## 6   Male  29   1.62   53.0             no  yes    2   3 Sometimes    no    2
##   SCC FAF TUE       CALC                 MTRANS            Obesity   BMI
## 1  no   0   1         no Public_Transportation       Normal_Weight 24.39
## 2 yes   3   0  Sometimes Public_Transportation       Normal_Weight 24.24
## 3  no   2   1 Frequently Public_Transportation       Normal_Weight 23.77
## 4  no   2   0 Frequently               Walking  Overweight_Level_I 26.85
## 5  no   0   0  Sometimes Public_Transportation Overweight_Level_II 28.34
## 6  no   0   0  Sometimes            Automobile       Normal_Weight 20.20
##   Obesity_binary
## 1      Not Obese
## 2      Not Obese
## 3      Not Obese
## 4      Not Obese
## 5      Not Obese
## 6      Not Obese
```

One problem when dealing with cleaning data is duplication. Duplication can lead to incorrect conclusion by making people believe that some observations are more common than they really are. Therefore, removing duplicates is very important before moving on analyzing data.

```
# Remove duplicate rows in the data frame
df = unique(df)
nrow(df)
```

```
## [1] 2087
```

As the result, number of observations decreased to 2087, which means 24 observations have been removed from the dataset. Another important thing is to check whether there are any missing values in the dataset, because it can skew results.

```
# Check for number of missing values
sum(is.na(df))
```

```
## [1] 0
```

There is no missing values in the dataset.

# 3 Exploratory Data Analysis

For better understanding the data as well as knowing the relationship between variables, some graphs are visualized to make the trends and patterns to be more easily seen. We start by plotting the number of users for Obesity type.

```
# Bar plot for weight classification using Obesity column
count_Obesity <- df %>% group_by(Obesity) %>% summarise(counts = n())
ggplot(count_Obesity, aes(x = Obesity, y = counts)) +
  geom_bar(fill = "#0073C2FF", stat = "identity") +
  scale_x_discrete(limits = c('Insufficient_Weight', 'Normal_Weight',
                              'Overweight_Level_I', 'Overweight_Level_II',
                              'Obesity_Type_I', 'Obesity_Type_II',
                              'Obesity_Type_III')) +
  geom_text(aes(label = counts), vjust = -0.3) +
```

```
  ylab('Number of observation') +
  theme(axis.text.x = element_text(size = 5),
        axis.title.x = element_blank())
```

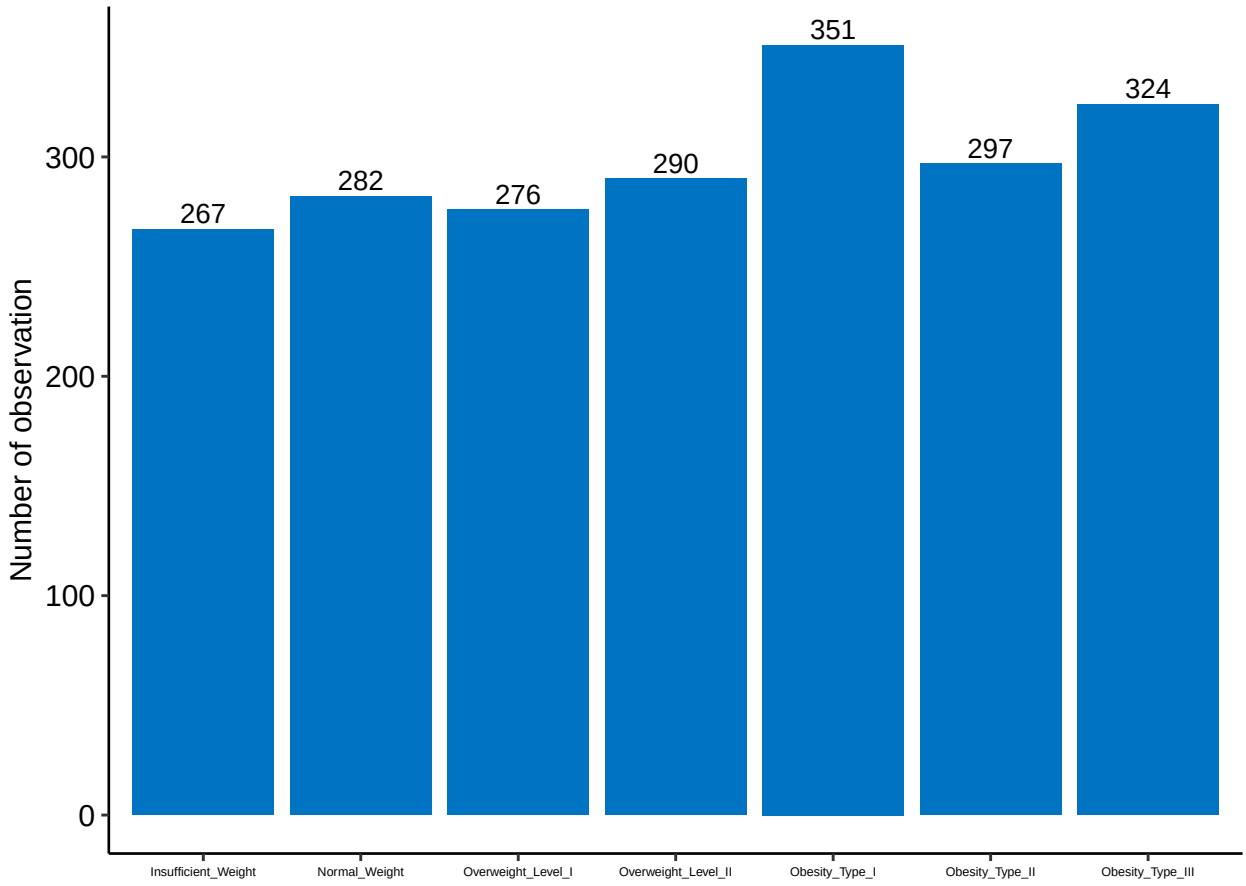



Figure 1: Number of observations per weight classification

Figure 1 shows that the most common type in the dataset is type I of obesity and the least common type is insufficient weight. Moreover, the dataset seems like balance because the number of observations between each category is not different significantly. This is the result of synthesizing the data [3].

## 3.1 Physical Activity

### 3.1.1 Means of Transportation

Let us plot the variable called Means of Transportation.

```
# Count number of observations base on weight classification and means of transportation
count_MTRANS <- df %>% group_by(Obesity, MTRANS) %>% tally()
# Bar plot for MTRANS
ggplot(data = count_MTRANS, aes(x=Obesity, y = n, fill = MTRANS)) +
  geom_bar(stat = 'identity') +
  scale_fill_brewer() +
  ylab('Count') +
  theme(axis.text.x = element_text(size = 5),
        axis.title.x = element_blank(),
        legend.title = element_blank())
```

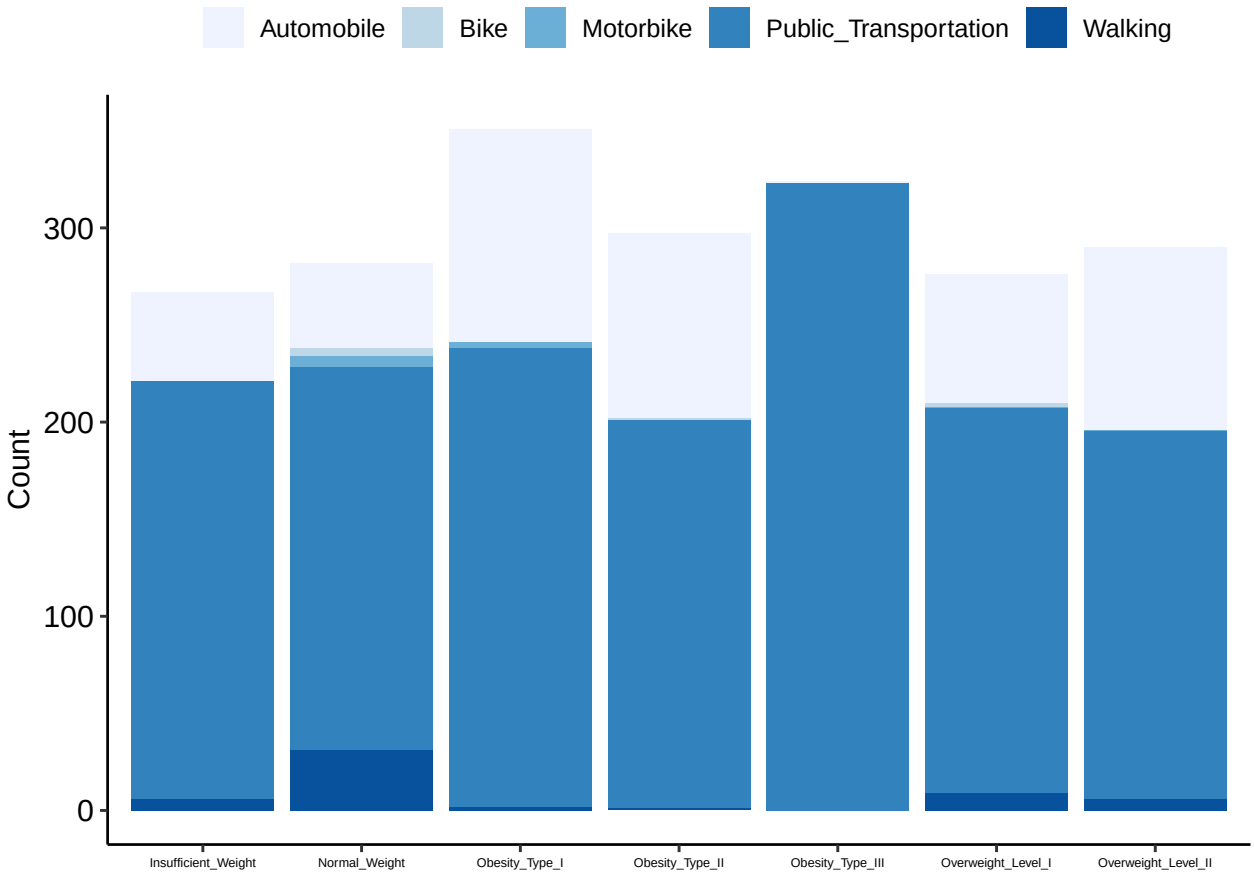



Figure 2: Means of transportation for each weight category

Means of transportation are significantly associated with the issues of weighting. Figure 2 shows that public transportation is the most common used vehicle beside with automobile. It can be seen that most of people who are in type III of obesity use only public transportation. In addition with public transportation, automobile is used more by obesity people. By contrast, people who are in normal weight walking more than any other type of weight classification. Therefore, traveling by automobile or by public transportation rather than by motorbike, bike or walking can encourage obesity by eliminating physical activity.

```
# Count number of observations base on weight classification and means of transportation
count_Obesity <- df %>% group_by(MTRANS, Obesity) %>% tally()
# Bar plot for Obesity
ggplot(data = count_Obesity, aes(x=MTRANS, y = n, fill = Obesity)) +
  geom_bar(stat = 'identity') +
  scale_fill_brewer() +
  ylab('Count') +
  theme(axis.text.x = element_text(size = 5),
        axis.title.x = element_blank(),
        legend.title = element_blank())
```

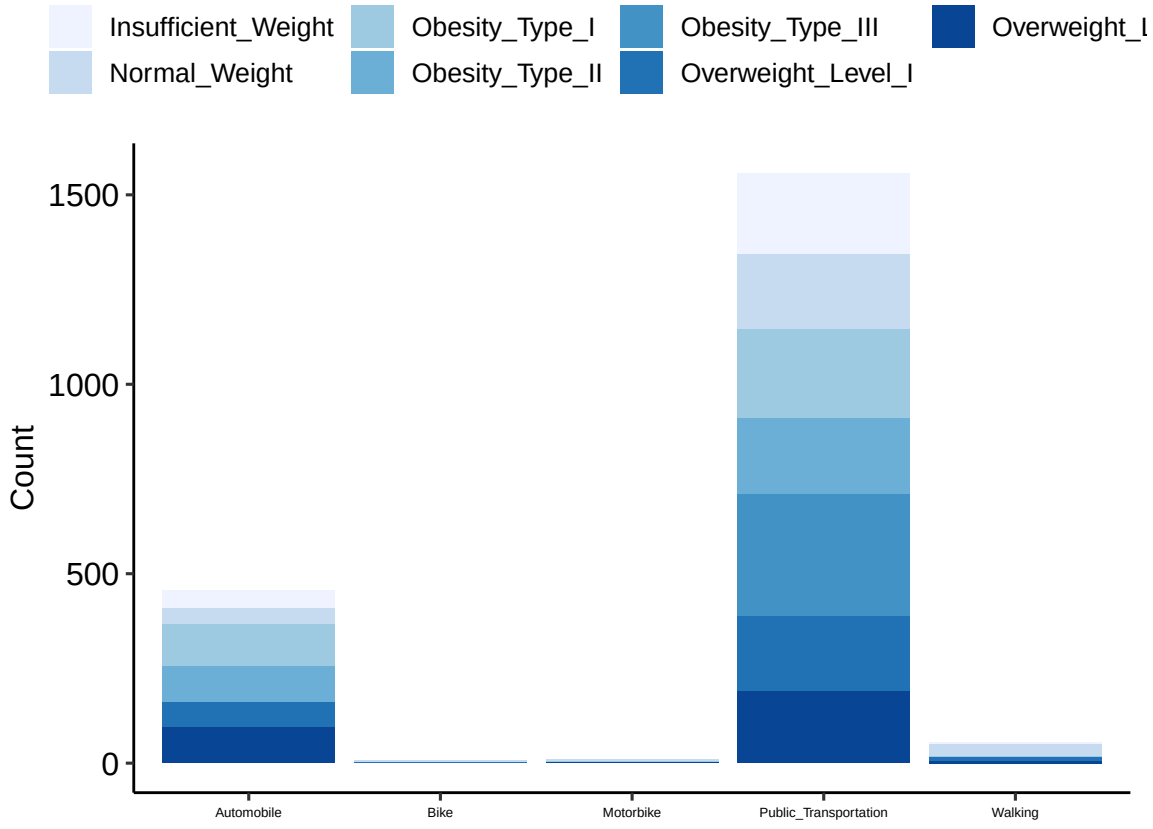



Figure 3: Weight cathegory for each means of transportation

### 3.1.2 Physical Activity Frequency

The importance of physical activity can be shown also by the following plot.

```
# Change the physical activity frequency into numeric
for (i in 1:nrow(df)){
  if (df$FAF[i] < 1) {
    df$FAF_1[i] = 0
```

```
  } else if (df$FAF[i] < 2 & df$FAF[i] >= 1) {
    df$FAF_1[i] = 1
  } else if (df$FAF[i] < 3 & df$FAF[i] >= 2) {
    df$FAF_1[i] = 2
  } else {
    df$FAF_1[i] = 3
  }
}
# Count number of observations base on weight classification
#and physical activity frequency
count_FAF <- df %>% group_by(Obesity, FAF_1) %>% tally()
# Bar plot for FAF
count_FAF$FAF_1 = as.factor(count_FAF$FAF_1)
ggplot(data = count_FAF, aes(x=Obesity, y = n, fill = FAF_1)) +
  geom_bar(stat = 'identity') +
  scale_fill_brewer() +
  ylab('Count') +
  scale_x_discrete(limits = c('Insufficient_Weight', 'Normal_Weight',
                              'Overweight_Level_I', 'Overweight_Level_II',
                              'Obesity_Type_I', 'Obesity_Type_II',
                              'Obesity_Type_III')) +
  theme(axis.text.x = element_text(size = 5),
        axis.title.x = element_blank(),
        legend.title = element_blank())
```

```
# Delete column FAF_1
df$FAF_1 <- NULL
```

As we can see on figure 4, overweight and obesity people do not active much. People who frequently take part in physical activities are usually have normal weight and some are insufficient weight with a little less intensity. Moreover, people with obesity type II or more are sedentary.

## 3.2 Eating Habits

Now we can analyze the relationship between BMI or Obesity with eating habits. We consider the consumption of high caloric food and calories consumption monitoring.

```
# Boxplot for frequent consumption of high caloric food
favc_boxplot <- ggplot(data = df, aes(x = FAVC, y = BMI, fill = FAVC)) +
  geom_boxplot() +
  theme(legend.position = "none") +
  xlab('Consumption of high caloric food')

# Boxplot for calories consumption monitoring
scc_boxplot <- ggplot(data = df, aes(x = SCC, y = BMI, fill = SCC)) +
  geom_boxplot() +
  theme(legend.position = "none") +
  xlab('Calories consumption monitoring')

grid.arrange(favc_boxplot, scc_boxplot, nrow = 1)
```

The cause of obesity is stated by the World Health Organization as an energy imbalance between calories consumed and calories expended. The result in the figure 5 clearly indicate that people who frequent consumption of high caloric food had a higher BMI than those who do not. The next plot shows the relationship between calories consumption monitoring and BMI index, which indicates that those people who

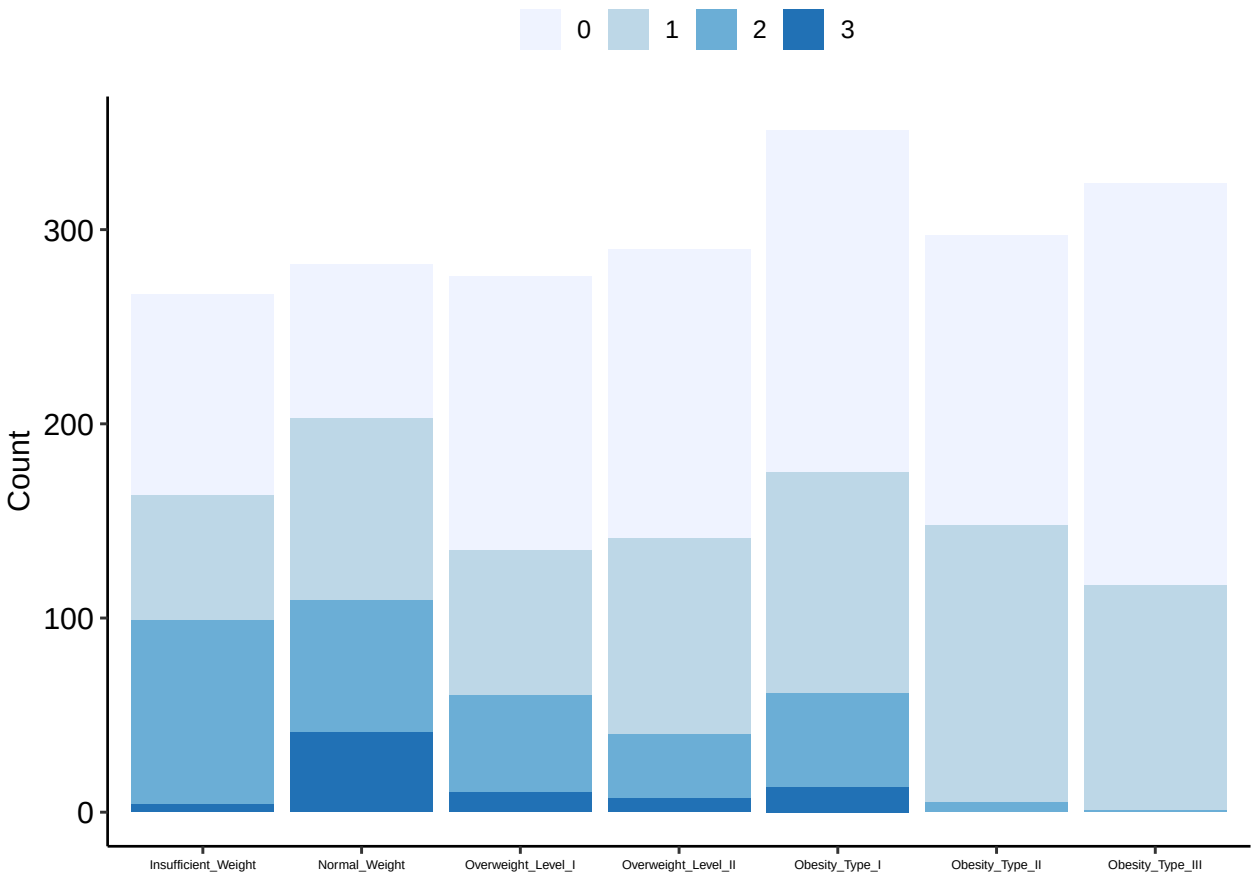



Figure 4: Physical Activity for Weight classification

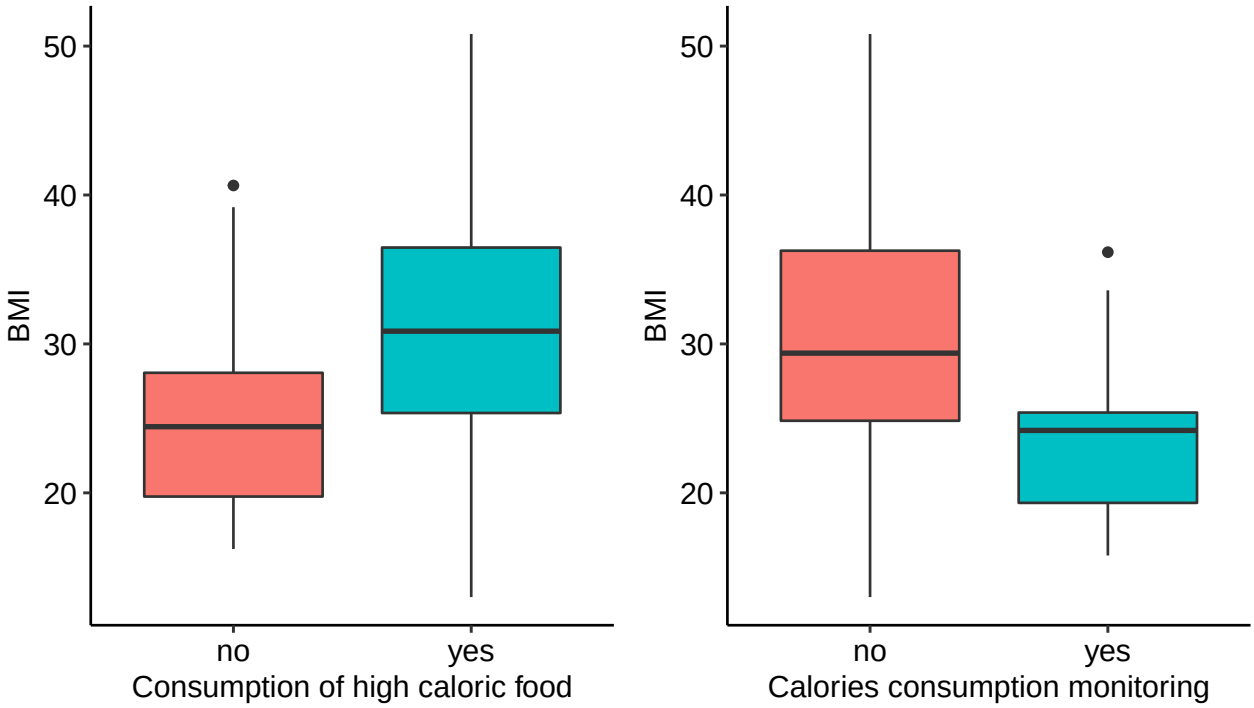



Figure 5: Eating Habits

do not control their calories consumption are more likely to have a higher BMI than those who control.

Then we can focus on the consumption of food between meals and on the consumption of alcohol.

```
# Boxplot for Consumption of food between meals
caec_boxplot <- ggplot(data = df, aes(x = CAEC, y = BMI, fill = CAEC)) +
  geom_violin() +
  scale_x_discrete(limits = c('no', 'Sometimes', 'Frequently', 'Always')) +
  theme(legend.position = "none", axis.text.x = element_text(size = 5)) +
  xlab('Consumption of food between meals')

# Boxplot for Consumption of alcohol
calc_boxplot <- ggplot(data = df, aes(x = CALC, y = BMI, fill = CALC)) +
  geom_violin() +
  scale_x_discrete(limits = c('no', 'Sometimes', 'Frequently', 'Always')) +
  theme(legend.position = "none", axis.text.x = element_text(size = 5)) +
  xlab('Consumption of alcohol')

grid.arrange(caec_boxplot, calc_boxplot, nrow = 1)
```

```
## Warning: Groups with fewer than two data points have been dropped.
```

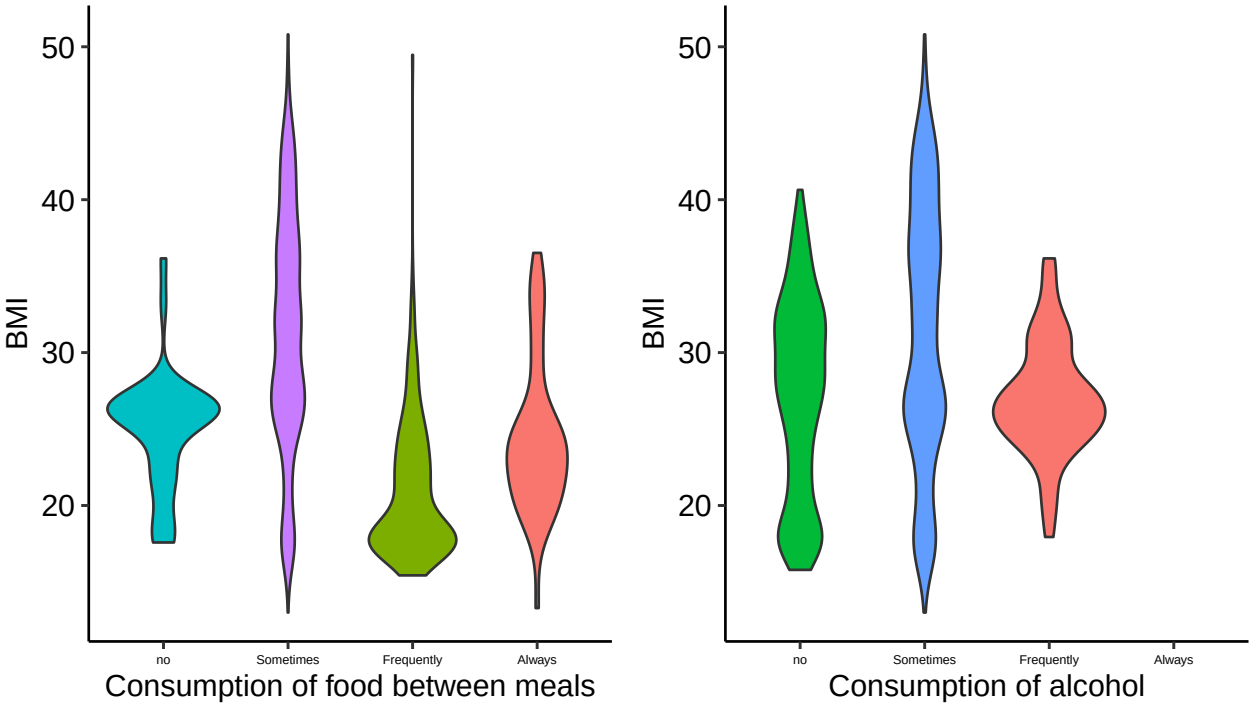



Figure 6: Eating Habits

In Figure 5, we do not observe a strong relation between BMI and those two cathegories. The same holds for consumption of vegetables, shown in 7.

```
# Boxplot for Frequency of consumption of vegetables
# Discretize the continuous variable
fcvc.discr = cut(df$FCVC, breaks = c(0.99, 1.66, 2.33, 3.1), labels = FALSE)
df$fcvc.discr = fcvc.discr
fcvc_boxplot <- ggplot(data = df, aes(x = as.factor(fcvc.discr), y = BMI,
                                      fill = as.factor(fcvc.discr))) +
```

```
  geom_boxplot() +
    xlab('Consumption of vegetables') +
  theme(legend.title = element_blank())
df$fcvc.discr = NULL

grid.arrange( fcvc_boxplot)
```

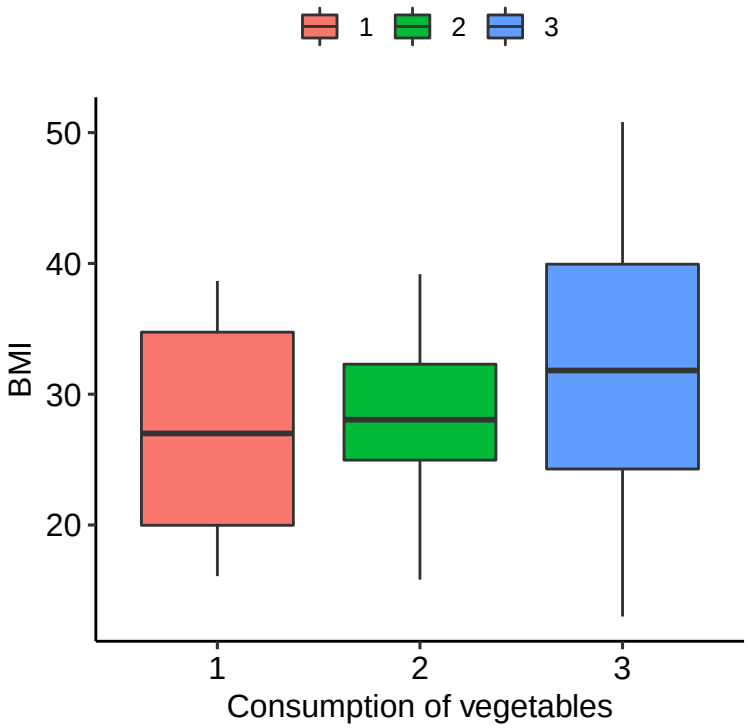



Figure 7: Frequency of consumption of vegetables

We consider the number of main meals.

```
ncp.discr = cut(df$NCP, breaks = c(0.99, 1.5, 2.5, 3.5, 4.1), labels = FALSE)
df$ncp.discr = ncp.discr
ggplot(data = df, aes(x = as.factor(ncp.discr), y = BMI, color = as.factor(ncp.discr))) +
  geom_jitter() +
  xlab('Number of Main Meals') +
  theme(legend.title = element_blank()) +
  geom_hline(yintercept=30, linetype="dashed", color = "black")
```

```
df$ncp.discr = NULL
```

From Figure 8, we notice that the majority of the users consume 3 main meals, there is not a clear explanation.

```
wat.discr = cut(df$CH2O, breaks = c(0.99, 1.5, 2.5, 3.1), labels = FALSE)
df$wat.discr = wat.discr
ggplot(data = df, aes(x = as.factor(wat.discr), color = Obesity_binary, fill = Obesity_binary)) +
  geom_bar(position = 'dodge') +
  xlab('Consumption of water') +
  theme(legend.title = element_blank())
```

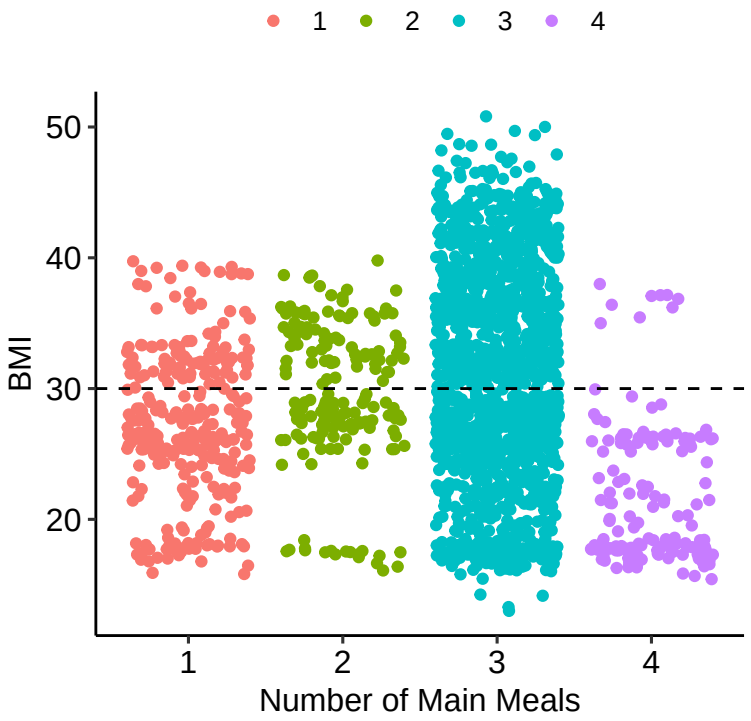



Figure 8: Number of Main Meals

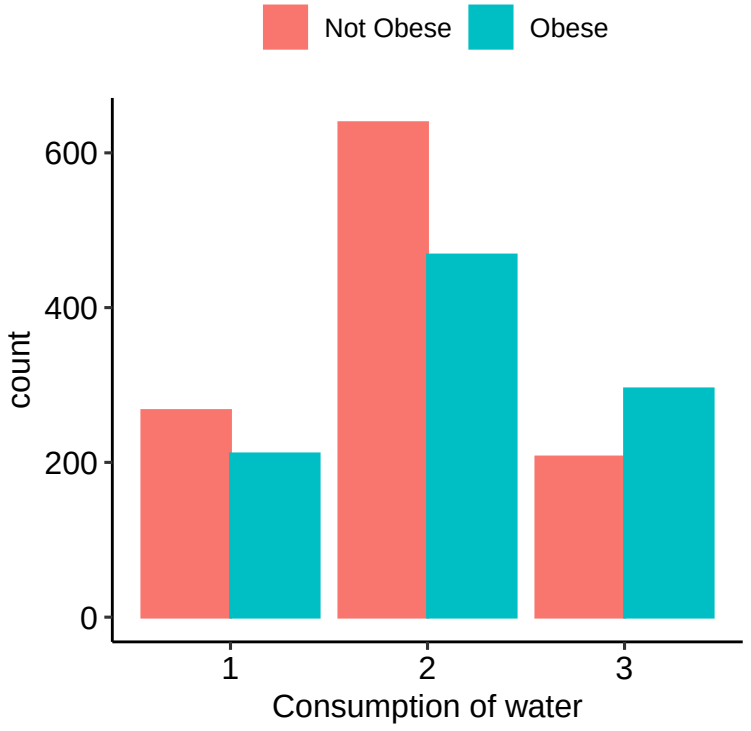



Figure 9: Consumption of Water

```
df$wat.discr = NULL
```

In Figure 9, we cannot indicate a relation between Obesity and the consumption of water.

## 3.3 Habits

Lastly, we want to investigate the relation between the obesity variable and the user's habits, such as the time using technology and smoking.

### 3.3.1 Time using Technology

Lastly, let us consider the time using technology.

```
# Change time using technology into numeric
for (i in 1:nrow(df)){
  if (df$TUE[i] < 1){
    df$TUE_1[i] = 0
  }
  else if (df$TUE[i] < 2 && df$TUE[i] >= 1) {
    df$TUE_1[i] = 1
  }
  else {
    df$TUE_1[i] = 2
  }
}

count_TUE <- df %>% group_by(Obesity_binary, TUE_1) %>% tally()
# Bar plot for TUE
count_TUE$TUE_1 = as.factor(count_TUE$TUE_1)
ggplot(data = count_TUE, aes(x=as.factor(Obesity_binary), y = n, fill = TUE_1)) +
  geom_bar(stat = 'identity') +
  scale_fill_brewer() +
  ylab('Count') +
  theme(axis.text.x = element_text(size = 5),
        axis.title.x = element_blank(),
        legend.title = element_blank())
```

```
# Delete TUE_1 column
df$TUE_1 <- NULL
```

In the Figure 10, we do not see a significant relation between use of technology and Obesity.

### 3.3.2 Smoking

```
ggplot(df, aes(as.factor(Obesity_binary), fill=SMOKE)) +
  geom_bar(position="dodge") +
  scale_x_discrete(element_blank())
```

We can see in the figure 11 that the majority of the users do not smoke, so we can see that it does not really have an impact for the data that we are considering.

## 3.4 Others

### 3.4.1 Height and Weight

Let us continue by computing the density distribution of the variable Height and Weight.

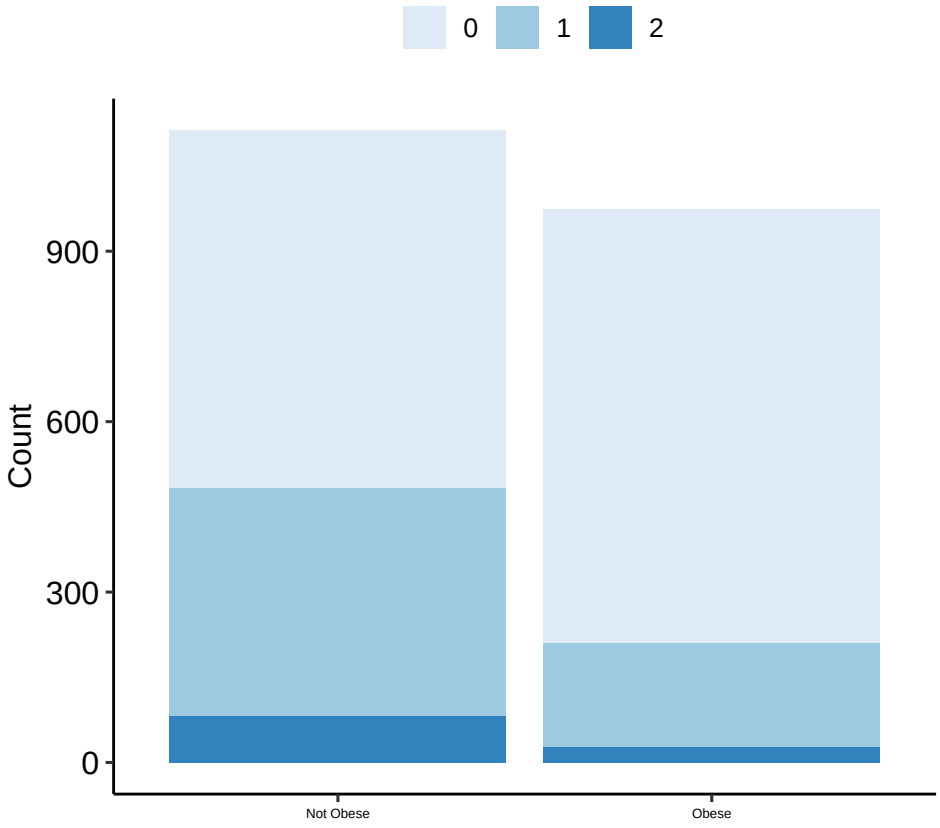



Figure 10: Time using Technology

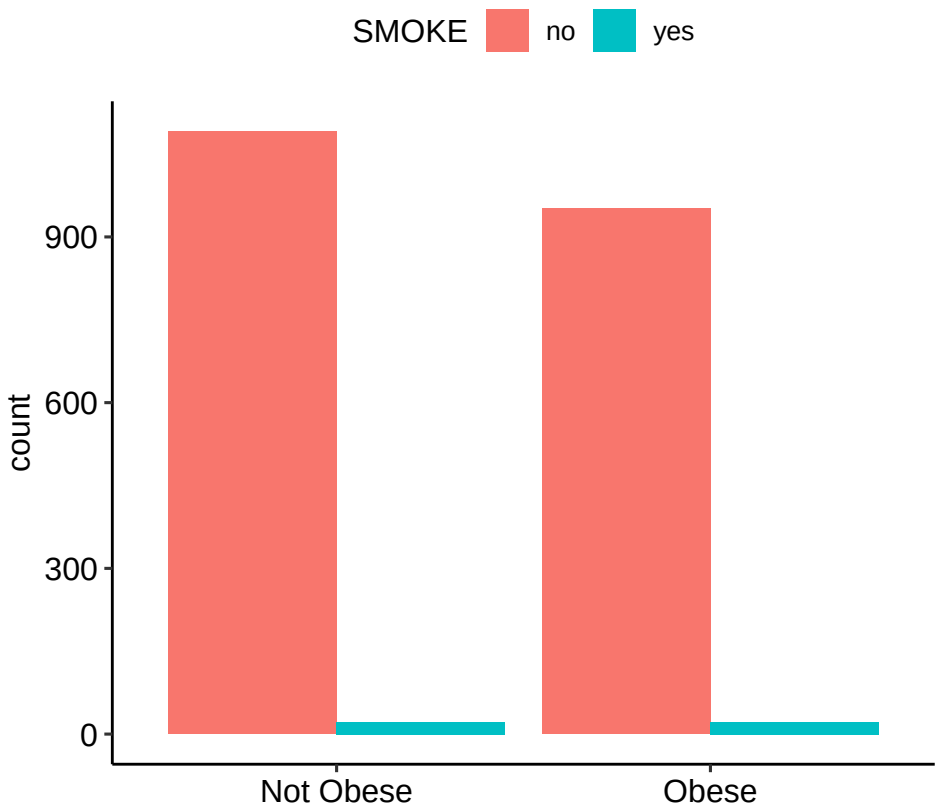



Figure 11: Smoking Habit

```
# Density plot of weight and height
weight_density <- ggplot(data = df, aes(x=Weight)) +
  geom_histogram((aes(y=..density..)), fill = 'lightblue', color = 'white', bins = 30) +
  geom_density(alpha = .4, color='darkblue') +
  geom_vline(aes(xintercept = mean(Weight)),
             color ='black', linetype='dashed', size = 0.5)
height_density <- ggplot(data = df, aes(x=Height)) +
  geom_histogram((aes(y=..density..)), fill = 'bisque', color = 'white', bins = 30) +
  geom_density(alpha = .4, color='darkorange') +
  geom_vline(aes(xintercept = mean(Height)),
             color ='black', linetype='dashed', size = 0.5)
grid.arrange(weight_density, height_density, nrow = 1)
```

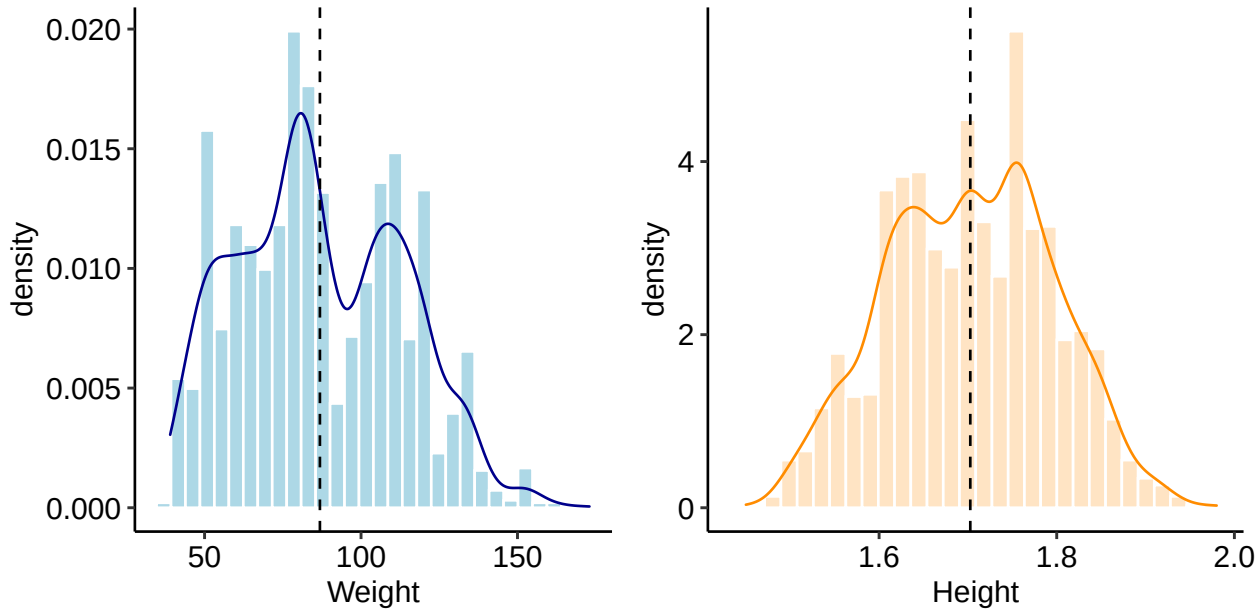



Figure 12: Weight and Height density

```
# Mean of the Height and Weight variable
cat('Mean of the Height:', round(mean(df$Height),2), 'm', '\n')
```

```
## Mean of the Height: 1.7 m
```

```
cat('Mean of the Weight:', round(mean(df$Weight), 2), 'kg')
```

```
## Mean of the Weight: 86.86 kg
```

By looking at the density curve in two plots in Figure 12, it can be seen that the density curve of the Weight variable has the bimodal shape with the mean is 86.86 kilogram, and also the Height variable shows a bimodel shape with the mean 1.7 meter.

Let us check whether we can explain the shape of the weight density plot could be explained by considering two different groups Obese and Non-Obese. From the definition of BMI, we know that the variable weight is strongly related to the obesity variable.

```
ggplot(df, aes(x=Weight, color=Obesity_binary, fill=Obesity_binary)) +
  geom_histogram(aes(y=..density..), position="identity", alpha=0.5, bins=30) +
  geom_density(alpha=0.6)
```


Figure 13: Density distributions of the weight of obese and not obese users.

```
tapply(df$Weight, df$Obesity_binary, mean)
```

```
## Not Obese     Obese
##  67.48073 109.00217
```

Then we check whether we can explain the bimodal shape of the density of the height in Figure 12 considering dividing the population according to the Gender.

```
ggplot(df, aes(x=Height, color=Gender, fill=Gender)) +
  geom_histogram(aes(y=..density..), position="identity", alpha=0.5, bins=30) +
  geom_density(alpha=0.6)
```

```
tapply(df$Height, df$Gender, mean)
```

```
##   Female     Male
## 1.643806 1.760591
```

### 3.4.2 BMI and Obesity Type

Now, we want to understand the relationship between BMI and Obesity Type. To do so we plot a boxplot of the BMI.

```
## Box plot for BMI per weight classification
ggplot(data = df, aes(x = Obesity, y = BMI, fill = Obesity)) +
  geom_boxplot() +
  scale_x_discrete(limits = c('Insufficient_Weight', 'Normal_Weight',
                              'Overweight_Level_I', 'Overweight_Level_II',
                              'Obesity_Type_I', 'Obesity_Type_II',
                              'Obesity_Type_III')) +
  theme(axis.text.x = element_text(size = 5),
        axis.title.x = element_blank())
```

Figure 15 shows the box plot for weight classification by BMI index. It can easily be seen from the graph that

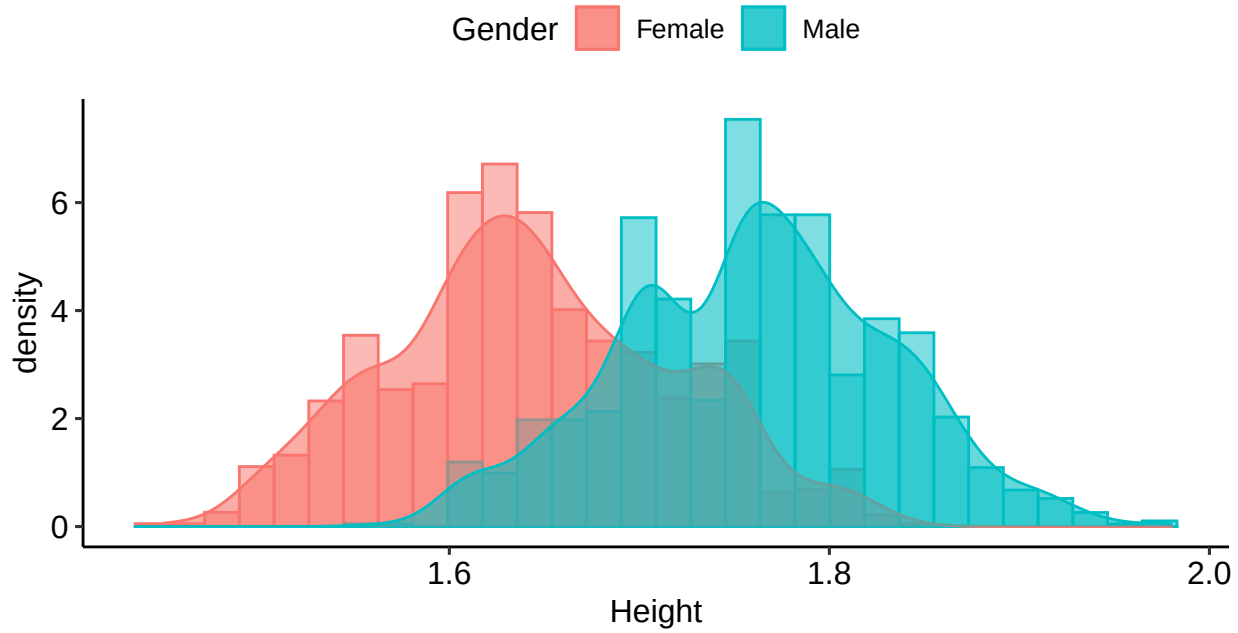



Figure 14: Height density of Male and Female

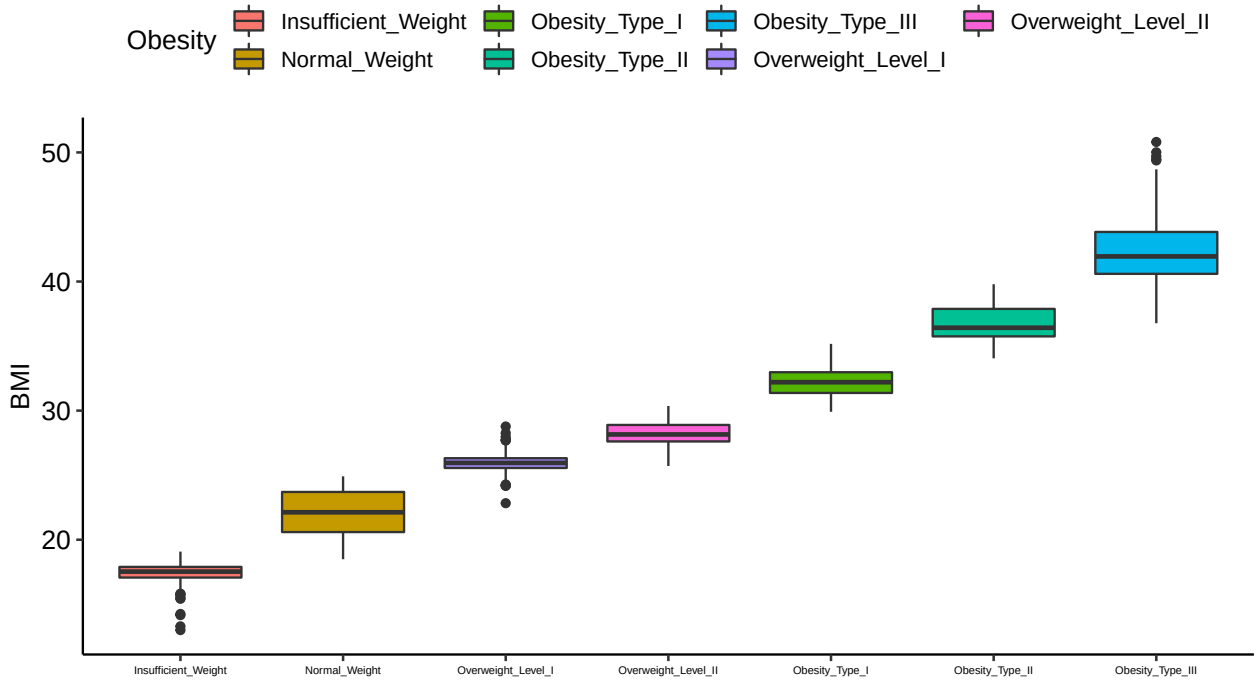



Figure 15: Box plot for BMI per weight classification

BMI has a strong relationship with the issues of weighting. According to WHO and Mexican Normativity, Obesity type has been classified from the BMI value. However, we seen that there are some outliers. Hence, we can plot another graph to visualize how many data have been classified differently from the proposed WHO classification.

```
## BMI per weight classification
ggplot(data = df, aes(x = Obesity, y = BMI, color = Obesity)) +
  geom_jitter() +
  geom_hline(yintercept=18.5, linetype="dashed", color = "black") +
  geom_hline(yintercept=25, linetype="dashed", color = "black") +
  geom_hline(yintercept=30, linetype="dashed", color = "black") +
  geom_hline(yintercept=35, linetype="dashed", color = "black") +
  geom_hline(yintercept=40, linetype="dashed", color = "black") +
  scale_x_discrete(limits = c('Insufficient_Weight', 'Normal_Weight',
                              'Overweight_Level_I', 'Overweight_Level_II',
                              'Obesity_Type_I', 'Obesity_Type_II',
                              'Obesity_Type_III')) +
  theme(axis.text.x = element_text(size = 5),
        axis.title.x = element_blank())
```


Figure 16: BMI per weight classification

According to WHO and the Mexican Normative, each pair of classes should be linearly separable. There are these mistakes, because most of these data have been synthetized. Moreover, we can notice two things, the normal weight cathegory is the one where all the elements belong within the thresholders and this cathegory does not contain synthetic data. The second element that we can consider is that between Overweight Level II and Obesity Type I, we have only a few elements outside the threshold, and so it does make sense to consider this as threshold for future analysis.

### 3.4.3 Age

Let us describe the relationship between Age and BMI, classified for Obesity.

```
ggplot(df, aes(Age, BMI, color=Obesity)) +
  geom_jitter()
```


Figure 17: BMI for age and Obesity type

Based on figure 17, we can see that there is not much observations for the age above 40. Most observations are concentrated in the age around 18 to 30 and this can be seen from the graph below.

```
# Density plot of age
age_density <- ggplot(data = df, aes(x=Age)) +
  geom_histogram((aes(y=..density..)), fill = 'lightgreen', color = 'white', bins = 30) +
  geom_density(alpha = .4, color = 'darkgreen')
plot(age_density)
```

### 3.4.4 Gender

Let us describe BMI classification divided by Gender. In this dataset for Gender is meant Sex.

```
ggplot(df, aes(Gender, BMI, color=Obesity)) +
  geom_count() +
  scale_x_discrete(element_blank())
```

We can see that the data are not balanced between male and female, maybe because the data are synthetic, but we should just consider the distribution of obesity (so grouping different classes) between male and female.

```
ggplot(df, aes(as.factor(Obesity_binary), fill=Gender)) +
  geom_bar(position="dodge") +
```

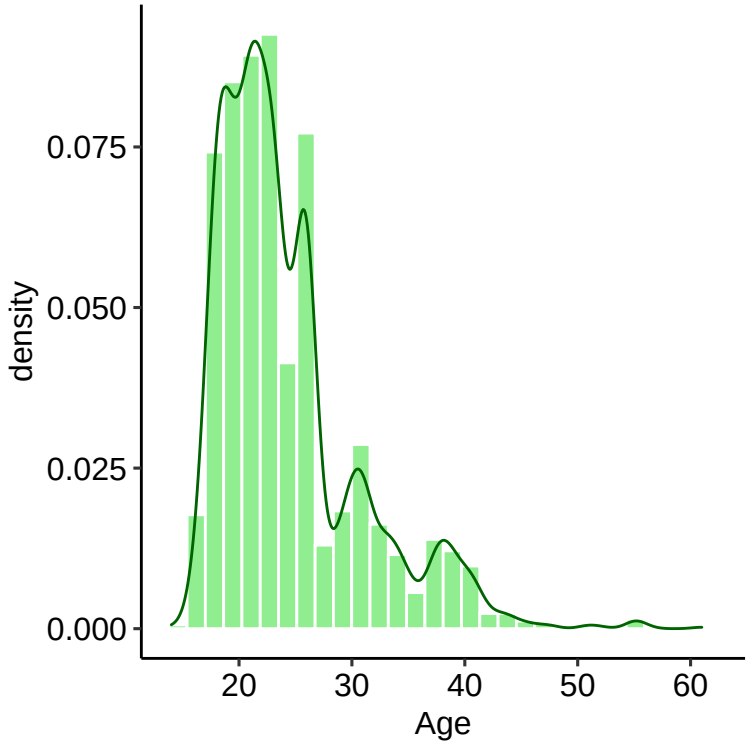



Figure 18: Age distribution

```
  scale_x_discrete(element_blank())
```

```
ggplot(df, aes(x=factor(1), fill=Gender))+
  geom_bar(width = 1)+
  coord_polar("y") +
  theme_void()
```

```
fem = length(df$Gender["Female"])
mal = length(df$Gender["Male"])

cat("Percentage of Female:", fem/(fem+mal)*100, "%\n")
```

```
## Percentage of Female: 50 %
```

```
cat("Percentage of Male:", mal/(fem+mal)*100, "%")
```

```
## Percentage of Male: 50 %
```

### 3.4.5 Family History

Let us now investigate how hereditary is obesity.

```
# Geom_bar for family history with overweight
ggplot(data = df, aes(x = Family_history, fill = Obesity_binary)) +
  geom_bar(position = 'dodge') +
  theme(legend.title = element_blank()) +
  xlab('Family history with overweight')
```

In figure 22, we can see the influence of the family history on obesity.

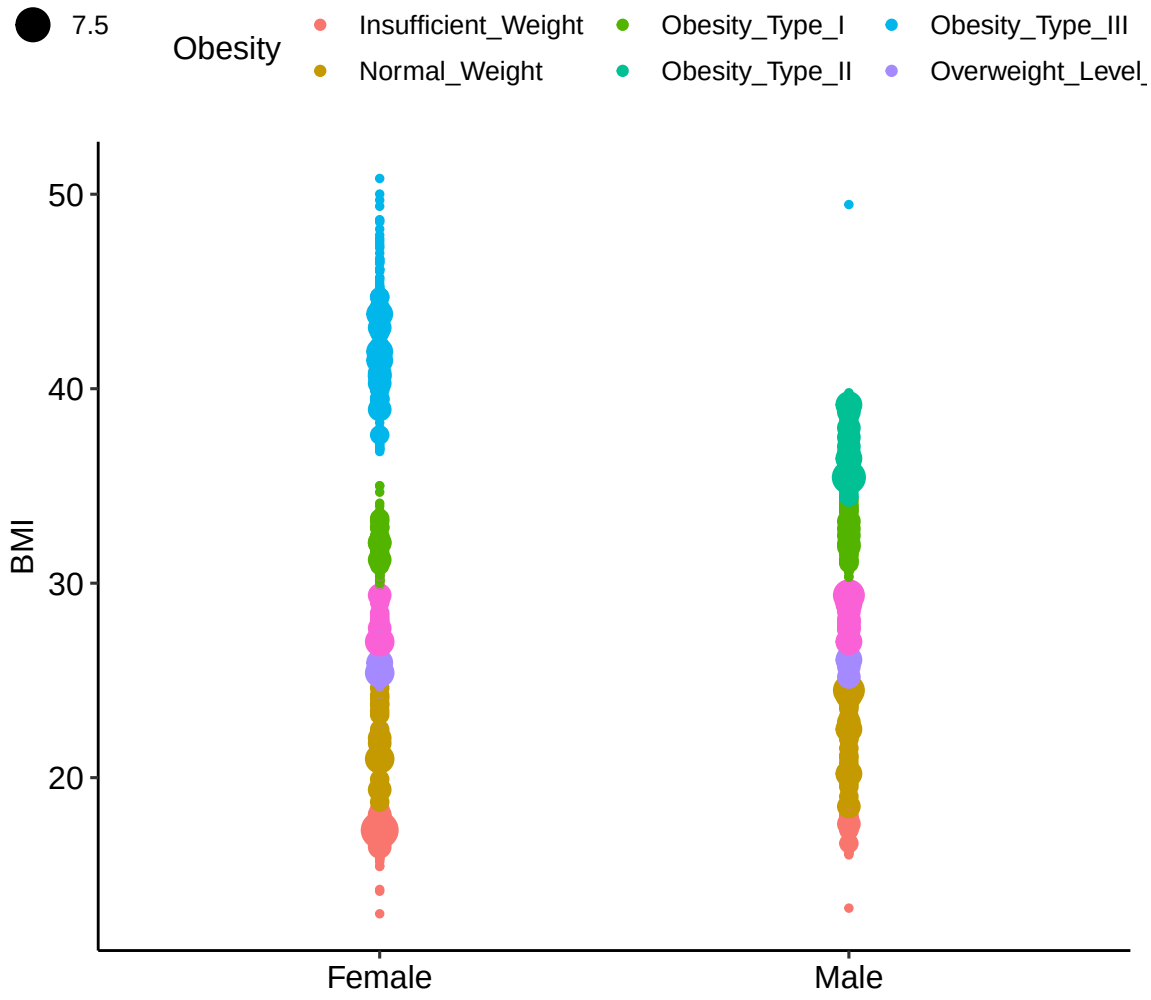



Figure 19: Age distribution

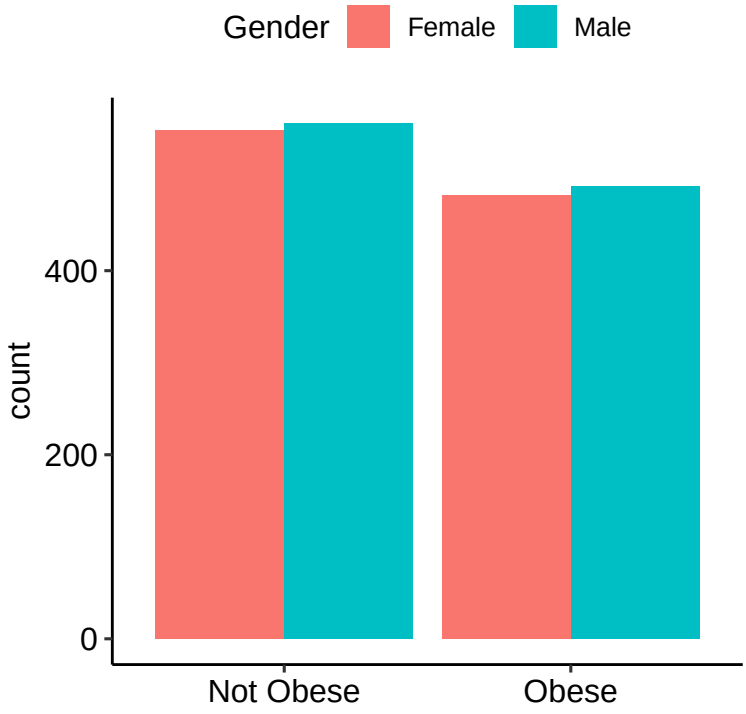



Figure 20: Obesity cathegory for gender

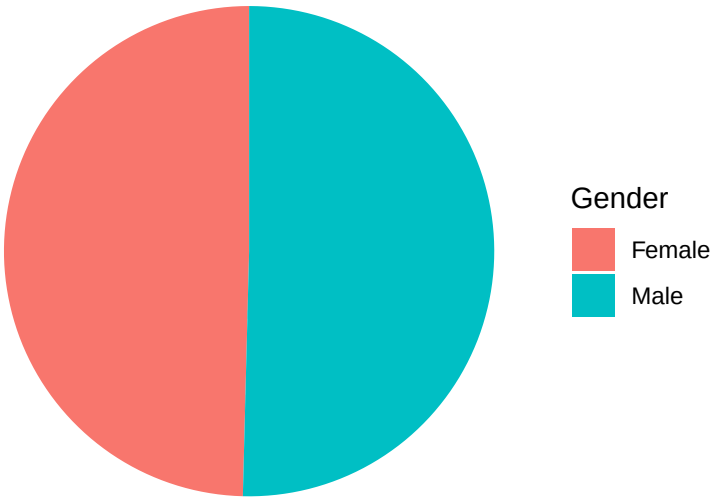



Figure 21: Percentage of Female and Female

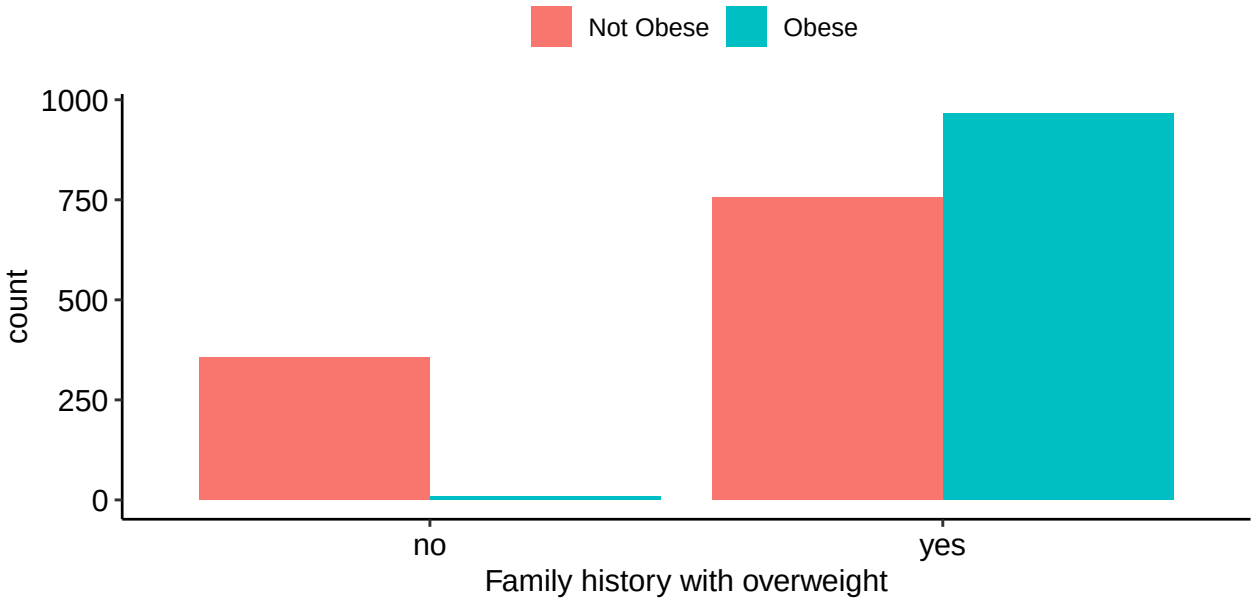



Figure 22: Family History with Obesity

# 4 Modelling

In this section, after normalizing the data and separating it into the appropriate training and test set, we will analyze it with a Logistic Regression model, given that we have a binary target. After this we will try to consider feature selection, to identify which feature are most relevant and if we can obtain better results symplifying the model. We will apply feature selection based on AIC and based on RSS. Furthermore, we will consider some regularization techniques, such as Lasso regularization and Ridge regularization.

In conclusion, we will analyze a different model based on Decision Trees, to have a comparison of two different models.

```
# Here we should first save the raw data
df_raw = copy(df)
```

```
# Drop current Obesity column
df$Obesity <- NULL

# Categorize obesity level based on BMI with BMI > 30 is Obesity (1) and
#BMI <= 30 is Non-Obesity (0)
for (i in 1:nrow(df)){
  if (df$BMI[i] <= 30){
    df$Obesity[i] = 0 #Non-obesity
  } else {
    df$Obesity[i] = 1 #Obesity
  }
}
# Dropping Obesity_binary
df$Obesity_binary <- NULL

# Dropping BMI
df$BMI <- NULL
```

```r
# Convert value in Obesity column into factor
df$Obesity = as.factor(df$Obesity)
# Convert Obesity into numeric without loss information (keep 0 and 1)
df$Obesity = as.numeric(levels(df$Obesity))[df$Obesity]
```

```r
# Female = 0, Male = 1
df$Gender = ifelse(df$Gender == 'Female', 0, 1)

# Family_history
df$Family_history = ifelse(df$Family_history == 'no', 0, 1)

# Smoke = 1, Non-smoke = 0
df$SMOKE = ifelse(df$SMOKE == 'no', 0, 1)

# FAVC
df$FAVC = ifelse(df$FAVC == 'no', 0, 1)

# SCC
df$SCC = ifelse(df$SCC == 'no', 0, 1)

# Convert CALC into number factor
for (i in 1:nrow(df)){
  if (df$CALC[i] == 'no'){
    df$CALC[i] = 0
  } else if (df$CALC[i] == 'Sometimes'){
    df$CALC[i] = 1
  } else if (df$CALC[i] == 'Frequently'){
    df$CALC[i] = 2
  } else {
    df$CALC[i] = 3
  }
}

# Convert CAEC into number factor
for (i in 1:nrow(df)){
  if (df$CAEC[i] == 'no'){
    df$CAEC[i] = 0
  } else if (df$CAEC[i] == 'Sometimes'){
    df$CAEC[i] = 1
  } else if (df$CAEC[i] == 'Frequently'){
    df$CAEC[i] = 2
  } else {
    df$CAEC[i] = 3
  }
}

# Convert MTRANS into number factor
for (i in 1:nrow(df)){
  if (df$MTRANS[i] == 'Public_Transportation'){
    df$MTRANS[i] = 0
  } else if (df$MTRANS[i] == 'Automobile'){
    df$MTRANS[i] = 1
  } else if (df$MTRANS[i] == 'Motorbike'){
    df$MTRANS[i] = 2
```

```
  } else if (df$MTRANS[i] == 'Bike'){
    df$MTRANS[i] = 3
  } else {
    df$MTRANS[i] = 4
  }
}
```

```
# Convert Gender into numeric without loss information (keep 0 and 1)
df$Gender = as.factor(df$Gender)
df$Gender = as.numeric(levels(df$Gender))[df$Gender]

# Convert Family_history into numeric without loss information (keep 0 and 1)
df$Family_history = as.factor(df$Family_history)
df$Family_history = as.numeric(levels(df$Family_history))[df$Family_history]

# Convert SMOKE into numeric without loss information (keep 0 and 1)
df$SMOKE = as.factor(df$SMOKE)
df$SMOKE = as.numeric(levels(df$SMOKE))[df$SMOKE]

# Convert FAVC into numeric without loss information (keep 0 and 1)
df$FAVC = as.factor(df$FAVC)
df$FAVC = as.numeric(levels(df$FAVC))[df$FAVC]

# Convert SCC into numeric without loss information (keep 0 and 1)
df$SCC = as.factor(df$SCC)
df$SCC = as.numeric(levels(df$SCC))[df$SCC]

# Convert CALC into numeric without loss information (keep 0 and 1)
df$CALC = as.factor(df$CALC)
df$CALC = as.numeric(levels(df$CALC))[df$CALC]

# Convert CAEC into numeric without loss information (keep 0 and 1)
df$CAEC = as.factor(df$CAEC)
df$CAEC = as.numeric(levels(df$CAEC))[df$CAEC]

# Convert MTRANS into numeric without loss information (keep 0 and 1)
df$MTRANS = as.factor(df$MTRANS)
df$MTRANS = as.numeric(levels(df$MTRANS))[df$MTRANS]
```

```
# Save df into data for modelling part
data = df

# Correlation matrix
corr_mat <- round(cor(data),2)
head(corr_mat)
```

```
##                Gender   Age Height Weight Family_history FAVC  FCVC   NCP  CAEC
## Gender           1.00  0.05   0.63   0.16           0.11 0.06 -0.27  0.08 -0.07
## Age              0.05  1.00  -0.03   0.20           0.20 0.06  0.01 -0.06 -0.09
## Height           0.63 -0.03   1.00   0.46           0.23 0.18 -0.04  0.23 -0.06
## Weight           0.16  0.20   0.46   1.00           0.49 0.27  0.22  0.09 -0.30
## Family_history   0.11  0.20   0.23   0.49           1.00 0.21  0.03  0.03 -0.21
## FAVC             0.06  0.06   0.18   0.27           0.21 1.00 -0.03 -0.01 -0.15
##                SMOKE  CH2O   SCC   FAF   TUE  CALC MTRANS Obesity
```

```
## Gender           0.05  0.10 -0.10  0.19  0.02 -0.01   0.13    0.00
## Age              0.09 -0.04 -0.12 -0.15 -0.30  0.05   0.28    0.21
## Height           0.05  0.22 -0.14  0.29  0.04  0.14   0.08    0.13
## Weight           0.02  0.20 -0.21 -0.06 -0.08  0.21  -0.11    0.79
## Family_history   0.01  0.17 -0.19 -0.06  0.00 -0.03  -0.02    0.41
## FAVC            -0.05  0.00 -0.19 -0.11  0.07  0.09  -0.14    0.27
```

```
# Reorder the correlation matrix
reorder_corr_mat <- function(corr_mat){
  dd <- as.dist((1-corr_mat)/2)
  hc <- hclust(dd)
  corr_mat <- corr_mat[hc$order, hc$order]
}

corr_mat <- reorder_corr_mat(corr_mat)

# Melt the correlation matrix
melted_corr_mat <- melt(corr_mat)

# Heatmap for correlation matrix
ggheatmap <- ggplot(melted_corr_mat, aes(Var2, Var1, fill = value)) +
  geom_tile(color = 'white') +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
  scale_fill_viridis() +
  theme(axis.text.x = element_text(angle = 90)) +
  scale_x_discrete(element_blank()) +
  scale_y_discrete(element_blank())

print(ggheatmap)
```

Figure 23 shows the heatmap with strong relationship between 2 pairs: Obesity and Weight, Height and Gender. To avoid multicollinearity, we will drop the Weight column because of high corelation with the target variable Obesity.

```
# Dropping weight column
data$Weight <- NULL
```

```
# TARGET VARIABLE: Obesity
count_Obesity <- data %>% group_by(Obesity) %>% summarise(counts = n())
count_Obesity
```

```
## # A tibble: 2 x 2
##   Obesity counts
##     <dbl>  <int>
## 1       0   1113
## 2       1    974
```

```
ggplot(count_Obesity, aes(x = as.factor(Obesity), y = counts)) +
  geom_bar(fill = "#0073C2FF", stat = "identity") +
  geom_text(aes(label = counts), vjust = -0.3) +
  xlab('Weight classification') +
  ylab('Number of observation') +
  scale_x_discrete(labels = c('Non-obesity', 'Obesity')) +
  theme_pubclean()
```

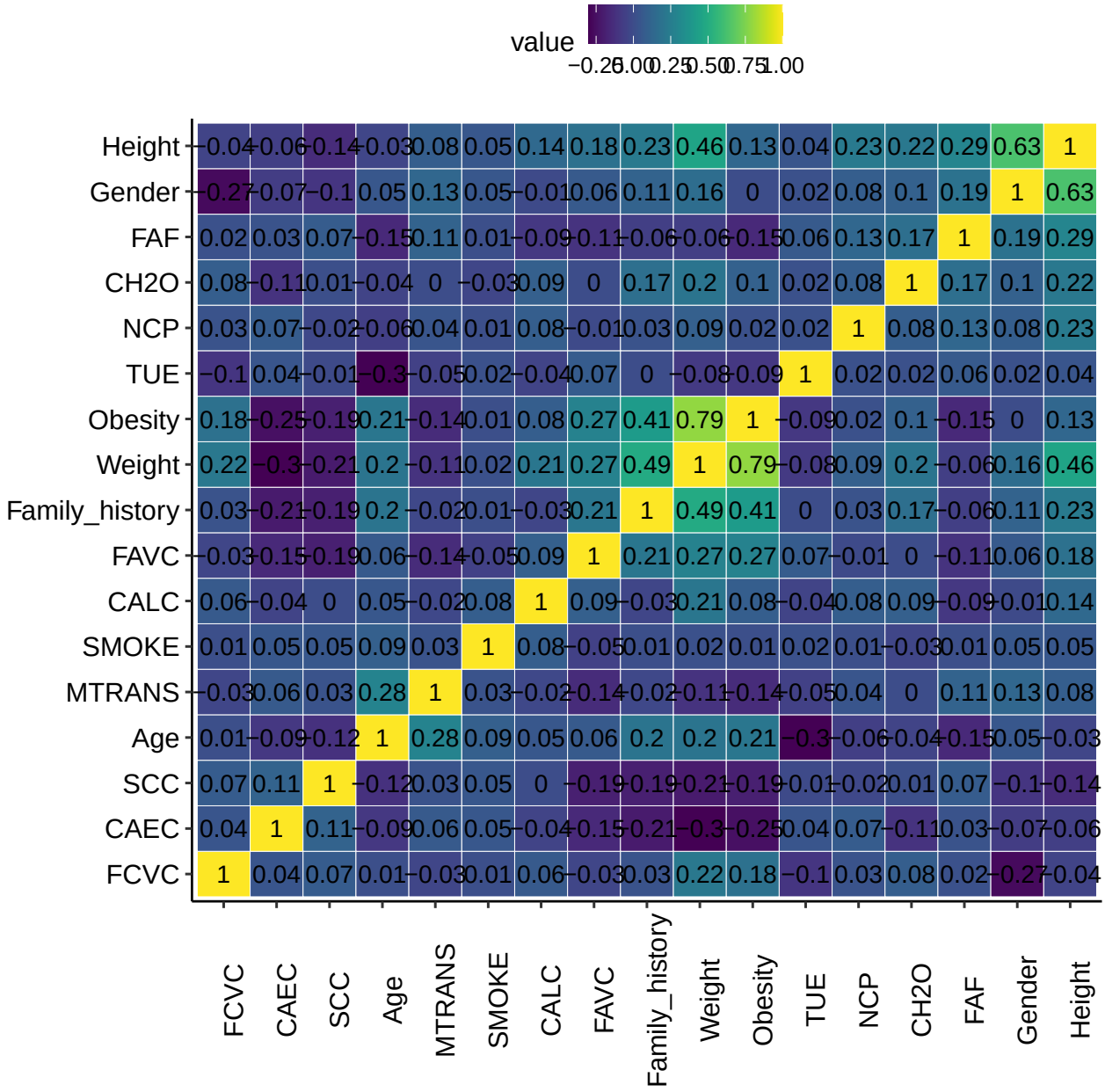



Figure 23: Correlation matrix

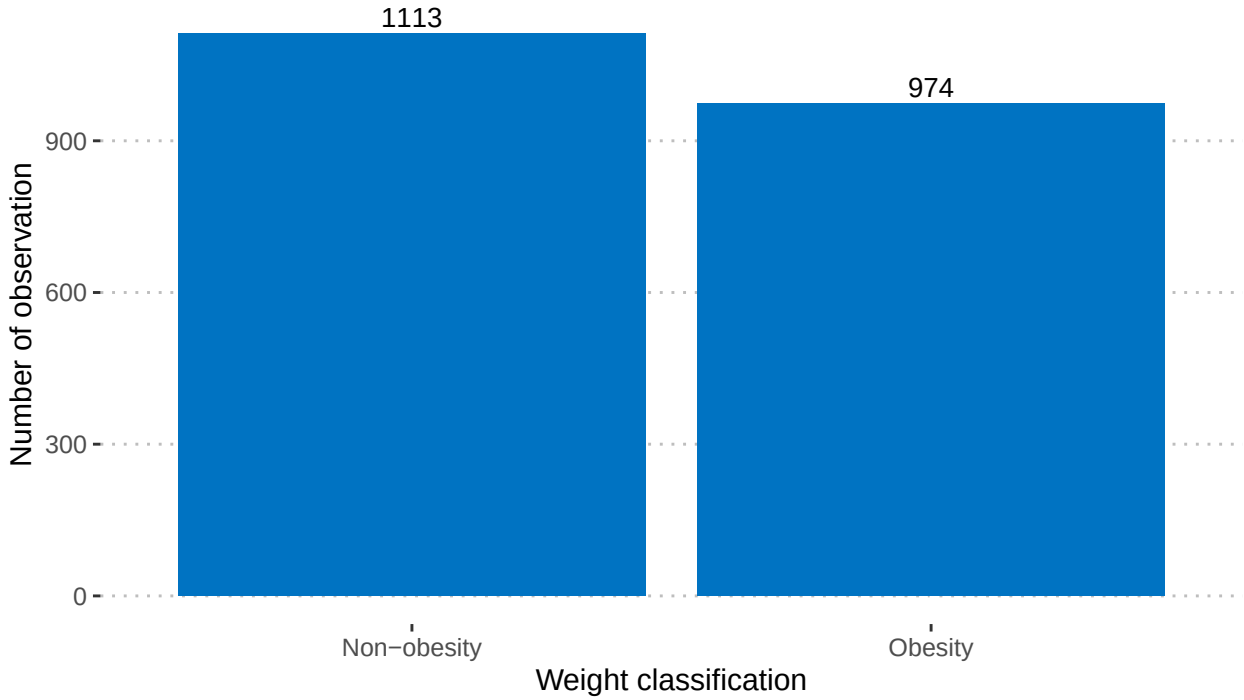



Figure 24: Number of observation per weight classification

## 4.1 Normalize the data

Before going deep into the model, we normalize the data.

```
# Normalize the data
preprocess <- preProcess(data, method = c('range'))
data <- predict(preprocess, data)
summary(data)
```

```
##      Gender            Age              Height       Family_history
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.1259   1st Qu.:0.3400   1st Qu.:1.0000
##  Median :1.0000   Median :0.1882   Median :0.4747   Median :1.0000
##  Mean   :0.5041   Mean   :0.2203   Mean   :0.4767   Mean   :0.8251
##  3rd Qu.:1.0000   3rd Qu.:0.2553   3rd Qu.:0.6028   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       FAVC             FCVC             NCP              CAEC
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:1.0000   1st Qu.:0.5000   1st Qu.:0.5658   1st Qu.:0.3333
##  Median :1.0000   Median :0.6981   Median :0.6667   Median :0.3333
##  Mean   :0.8836   Mean   :0.7107   Mean   :0.5671   Mean   :0.3820
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.6667   3rd Qu.:0.3333
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      SMOKE             CH2O             SCC              FAF
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.2955   1st Qu.:0.000   1st Qu.:0.0415
##  Median :0.00000   Median :0.5000   Median :0.000   Median :0.3333
##  Mean   :0.02108   Mean   :0.5024   Mean   :0.046   Mean   :0.3376
##  3rd Qu.:0.00000   3rd Qu.:0.7331   3rd Qu.:0.000   3rd Qu.:0.5594
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000
```

```
##       TUE              CALC            MTRANS           Obesity
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.3154   Median :0.3333   Median :0.00000   Median :0.0000
##  Mean   :0.3315   Mean   :0.2433   Mean   :0.08613   Mean   :0.4667
##  3rd Qu.:0.5000   3rd Qu.:0.3333   3rd Qu.:0.25000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
```

```
head(data)
```

```
##   Gender       Age    Height Family_history FAVC FCVC       NCP      CAEC SMOKE
## 1      0 0.1489362 0.3207547              1    0  0.5 0.6666667 0.3333333     0
## 2      0 0.1489362 0.1320755              1    0  1.0 0.6666667 0.3333333     1
## 3      1 0.1914894 0.6603774              1    0  0.5 0.6666667 0.3333333     0
## 4      1 0.2765957 0.6603774              0    0  1.0 0.6666667 0.3333333     0
## 5      1 0.1702128 0.6226415              0    0  0.5 0.0000000 0.3333333     0
## 6      1 0.3191489 0.3207547              0    1  0.5 0.6666667 0.3333333     0
##   CH2O SCC       FAF TUE      CALC MTRANS Obesity
## 1  0.5   0 0.0000000 0.5 0.0000000   0.00       0
## 2  1.0   1 1.0000000 0.0 0.3333333   0.00       0
## 3  0.5   0 0.6666667 0.5 0.6666667   0.00       0
## 4  0.5   0 0.6666667 0.0 0.6666667   1.00       0
## 5  0.5   0 0.0000000 0.0 0.3333333   0.00       0
## 6  0.5   0 0.0000000 0.0 0.3333333   0.25       0
```

Then we split the data into two sets: training and test with the ratio 7:3.

```
# Splitting test and training data (70% training and 30% testing)
set.seed(1234)

# Create a dummy indicator that indicates whether a row is assigned to the
#training or testing data set
split_dummy <- sample(c(rep(0, 0.7 * nrow(data)),
                        rep(1, 0.3 * nrow(data))))
print(table(split_dummy))
```

```
## split_dummy
##    0    1
## 1460  626
```

Creating training and test set based on variable split_dummy

```
# Create a train data set
data_train <- data[split_dummy == 0, ]
row.names(data_train) <- NULL # Reorder index
head(data_train)
```

```
##   Gender       Age     Height Family_history FAVC FCVC       NCP      CAEC
## 1      0 0.1489362 0.32075472              1    0  0.5 0.6666667 0.3333333
## 2      0 0.1489362 0.13207547              1    0  1.0 0.6666667 0.3333333
## 3      1 0.1914894 0.66037736              1    0  0.5 0.6666667 0.3333333
## 4      1 0.2765957 0.66037736              0    0  1.0 0.6666667 0.3333333
## 5      1 0.3191489 0.32075472              0    1  0.5 0.6666667 0.3333333
## 6      0 0.1914894 0.09433962              1    1  1.0 0.6666667 0.3333333
##   SMOKE CH2O SCC       FAF TUE      CALC MTRANS Obesity
## 1     0  0.5   0 0.0000000 0.5 0.0000000   0.00       0
## 2     1  1.0   1 1.0000000 0.0 0.3333333   0.00       0
```

```
## 3     0  0.5   0 0.6666667 0.5 0.6666667   0.00       0
## 4     0  0.5   0 0.6666667 0.0 0.6666667   1.00       0
## 5     0  0.5   0 0.0000000 0.0 0.3333333   0.25       0
## 6     0  0.5   0 0.3333333 0.0 0.3333333   0.50       0
```

```
x_train <- data_train %>% select(1:15)
y_train <- data_train %>% select(16) #Obesity column
y_train <- y_train[,1] # y_train should be a vector
```

```
# Create a test data set
data_test <- data[split_dummy == 1, ]
row.names(data_test) <- NULL # Reorder index
head(data_test)
```

```
##   Gender       Age    Height Family_history FAVC FCVC       NCP      CAEC SMOKE
## 1      1 0.1702128 0.6226415              0    0  0.5 0.0000000 0.3333333     0
## 2      1 0.1702128 0.3584906              0    0  0.5 0.6666667 0.3333333     0
## 3      1 0.2127660 0.6226415              1    1  1.0 0.6666667 0.3333333     0
## 4      0 0.8085106 0.4528302              1    1  1.0 0.0000000 0.3333333     1
## 5      0 0.1702128 0.2830189              1    1  0.0 0.0000000 0.3333333     0
## 6      1 0.5319149 0.6415094              0    0  0.5 0.0000000 0.3333333     0
##   CH2O SCC       FAF TUE      CALC MTRANS Obesity
## 1  0.5   0 0.0000000 0.0 0.3333333   0.00       0
## 2  0.5   0 1.0000000 0.0 0.3333333   0.00       0
## 3  0.5   0 0.3333333 0.5 0.6666667   0.00       0
## 4  0.5   0 0.0000000 0.0 0.0000000   0.25       1
## 5  0.5   0 0.0000000 1.0 0.3333333   0.00       1
## 6  0.5   0 0.0000000 0.0 0.3333333   0.00       0
```

```
x_test <- data_test %>% select(1:15)
y_test <- data_test %>% select(16) # Obesity column
y_test <- y_test[,1] # y_test should be a vector
```

## 4.2 Logistic Regression

```
model_lr <- glm(as.factor(Obesity) ~ ., data = data_train, family = binomial)

# Summary of model
summary(model_lr)
```

```
##
## Call:
## glm(formula = as.factor(Obesity) ~ ., family = binomial, data = data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2065  -0.7518  -0.0532   0.8220   3.9618
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -5.27406    0.60885  -8.662  < 2e-16 ***
## Gender         -0.11440    0.18704  -0.612 0.540787
## Age             4.14491    0.68361   6.063 1.33e-09 ***
## Height          1.16912    0.58943   1.983 0.047315 *
## Family_history  3.52252    0.40463   8.705  < 2e-16 ***
```

```
## FAVC            1.95178    0.31211   6.253 4.02e-10 ***
## FCVC            1.51724    0.28410   5.341 9.27e-08 ***
## NCP             0.23178    0.27606   0.840 0.401127
## CAEC           -4.75319    0.69900  -6.800 1.05e-11 ***
## SMOKE           0.14819    0.54568   0.272 0.785948
## CH2O           -0.03307    0.24033  -0.138 0.890564
## SCC            -2.66547    0.76790  -3.471 0.000518 ***
## FAF            -0.82816    0.26489  -3.126 0.001769 **
## TUE            -0.13651    0.23443  -0.582 0.560378
## CALC           -0.35648    0.41663  -0.856 0.392210
## MTRANS         -4.04000    0.72478  -5.574 2.49e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2016.6  on 1460  degrees of freedom
## Residual deviance: 1379.7  on 1445  degrees of freedom
## AIC: 1411.7
##
## Number of Fisher Scoring iterations: 6
```

As we can see on the coefficients column above, there are some variables which have strong connect with Obesity because of high values. These are Age, Family_history, CAEC and MTRANS. It definitely makes sense because based on the graph 17, elderly people are more likely to have problem with weight, the figure **??** shows people whose family has history with overweight are easily have obesity than those family do not, figure 2 proves that people who use less strenuous means of transport are more likely to have obesity and the more consumption of food between meals, the less likely you are to have weight problems.

Moreover, there are some variables which are not really significant in our model, they are Gender, NCP, SMOKE, CH2O, TUE and CALC.

```
# Calculate training and test accuracy based on the threshold 0.5
train_prob <- predict(model_lr, data_train, type = 'response')
train_pred <- ifelse(train_prob > 0.5, 1, 0)
train_acc <- mean(train_pred == data_train$Obesity)

test_prob <- predict(model_lr, data_test, type = 'response')
test_pred <- ifelse(test_prob > 0.5, 1, 0)
test_acc <- mean(test_pred == data_test$Obesity)

cat('Training accuracy: ', train_acc,'\n')
```

```
## Training accuracy:  0.7878166
```

```
cat('MSE of training data: ',mean((train_pred - y_train)^2),'\n')
```

```
## MSE of training data:  0.2121834
```

```
cat('Test accuracy: ', test_acc,'\n')
```

```
## Test accuracy:  0.7507987
```

```
cat('MSE of test data: ',mean((test_pred - y_test)^2))
```

```
## MSE of test data:  0.2492013
```

```r
# Define a function to compare two ROC curves of training and test set
plot_roc <- function(y_train, y_test, train_prob, test_prob){
  roc_train = AUC(obs = y_train, pred = train_prob,
                  main='ROC Curve on train set', plot=TRUE)
  roc_test = AUC(obs = y_test, pred = test_prob,
                 main='ROC Curve on test set', plot=TRUE)
}
```

```r
# Plot the ROC curve for training set
par(mfrow = c(1,1))
roc_lr = AUC(model_lr, main='ROC Curve on training set')
```

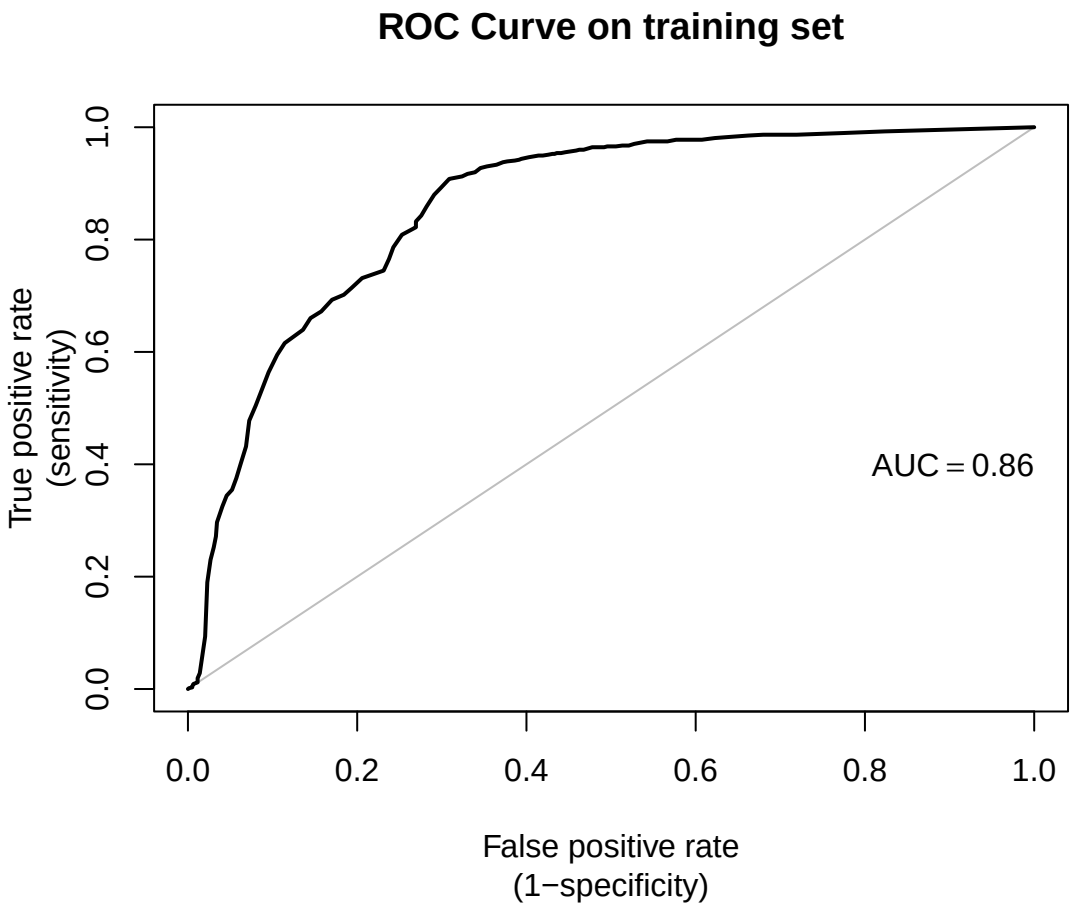



Figure 25:

Figure 25 shows the ROC curve on training set based on logistic regression. ROC curves and AUC tell how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. In this case, the AUC is 0.86, it means there is 86% chance that the model will be able to distinguish between obesity and non-obesity observations.

```r
# Calculate some metric based on train set
metric = data.frame(AUC(obs = data_train$Obesity, pred = train_prob,
                        main = 'ROC Curve', plot=FALSE))

metric_train = metric[, c('thresholds.thresholds', 'thresholds.sensitivity',
                          'thresholds.precision')]
```

```
accuracies_train = c()
for (i in metric_train$thresholds.thresholds){
  train_pred = ifelse(train_prob > i, 1, 0)
  train_acc = mean(train_pred == data_train$Obesity)
  accuracies_train = c(accuracies_train, train_acc)
}
metric_train$accuracies = accuracies_train
colnames(metric_train) = c('Threshold', 'Sensitivity', 'Precision', 'Accuracy')
head(metric_train)
```

```
##      Threshold Sensitivity Precision  Accuracy
## 0         0.00   1.0000000 0.4613279 0.4613279
## 0.01      0.01   0.9925816 0.5079727 0.5530459
## 0.02      0.02   0.9896142 0.5227273 0.5783710
## 0.03      0.03   0.9866469 0.5402112 0.6064339
## 0.04      0.04   0.9866469 0.5541667 0.6276523
## 0.05      0.05   0.9851632 0.5603376 0.6365503
```

```
# Find the optimal threshold
max = which.max(metric_train$Accuracy)
optimal_threshold = metric_train$Threshold[max]
best_accuracy = metric_train$Accuracy[max]
cat('Opimal threshold: ', optimal_threshold, '\n', 'Best accuracy: ', best_accuracy)
```

```
## Opimal threshold:  0.48
##  Best accuracy:  0.7912389
```

We can see that, based on the result in training set, the optimal threshold with the value 0.48 gives us the highest accuracy.

```
# Calculate the accuracy based on optimal threshold
train_prob_lr <- predict(model_lr, data_train, type = 'response')
train_pred_lr <- ifelse(train_prob_lr > optimal_threshold, 1, 0)
train_acc_lr <- mean(train_pred_lr == data_train$Obesity)

test_prob_lr <- predict(model_lr, data_test, type = 'response')
test_pred_lr <- ifelse(test_prob_lr > optimal_threshold, 1, 0)
test_acc_lr <- mean(test_pred_lr == data_test$Obesity)

cat('Training accuracy: ', train_acc,'\n',
    'Test accuracy: ', test_acc)
```

```
## Training accuracy:  0.5386721
##  Test accuracy:  0.7507987
```

```
#Confusion matrix for train set
conf_mat_train = confusionMatrix(as.factor(train_pred_lr), as.factor(y_train))
conf_mat_train
```

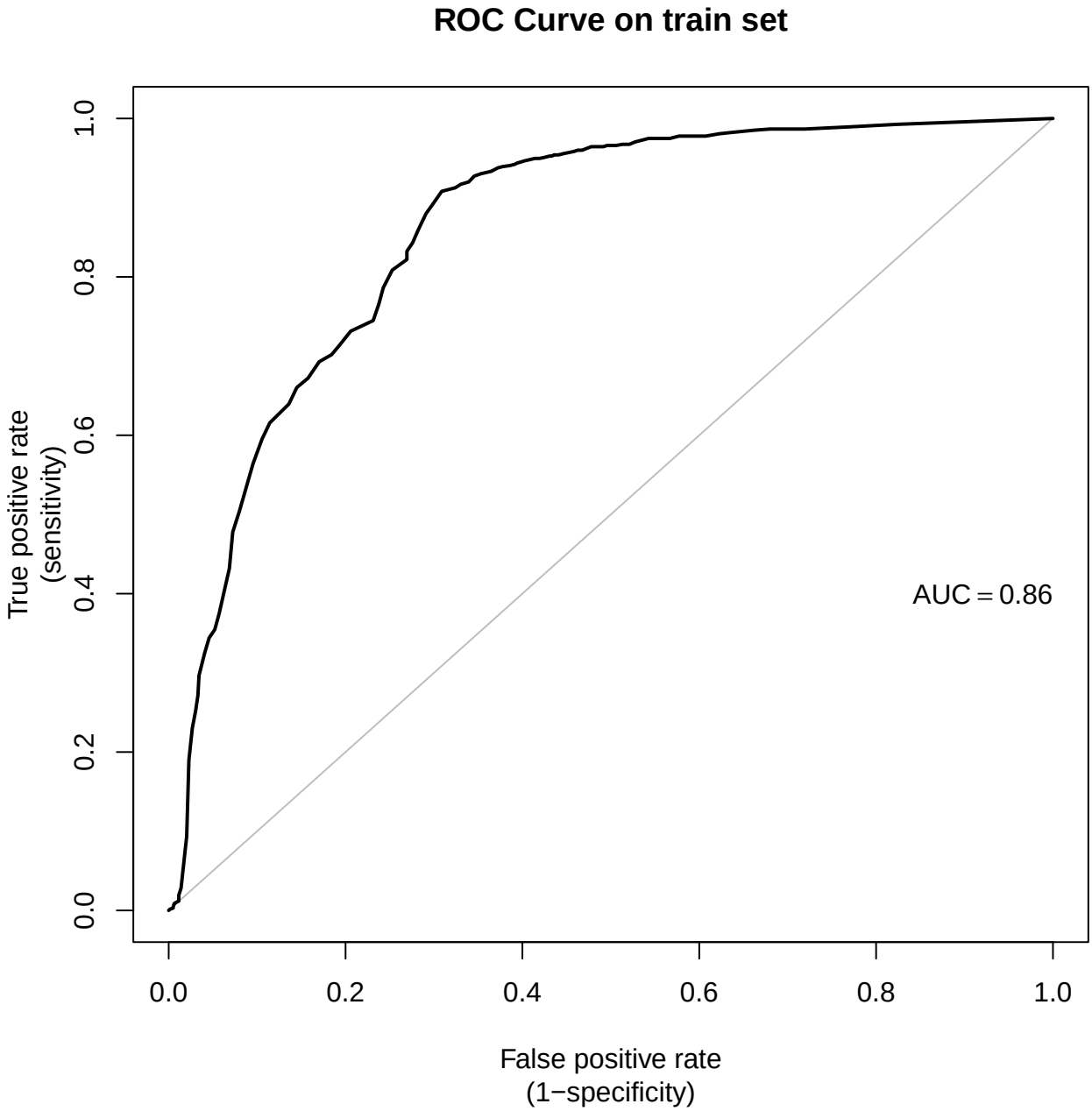
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 544  62
##          1 243 612
##
```

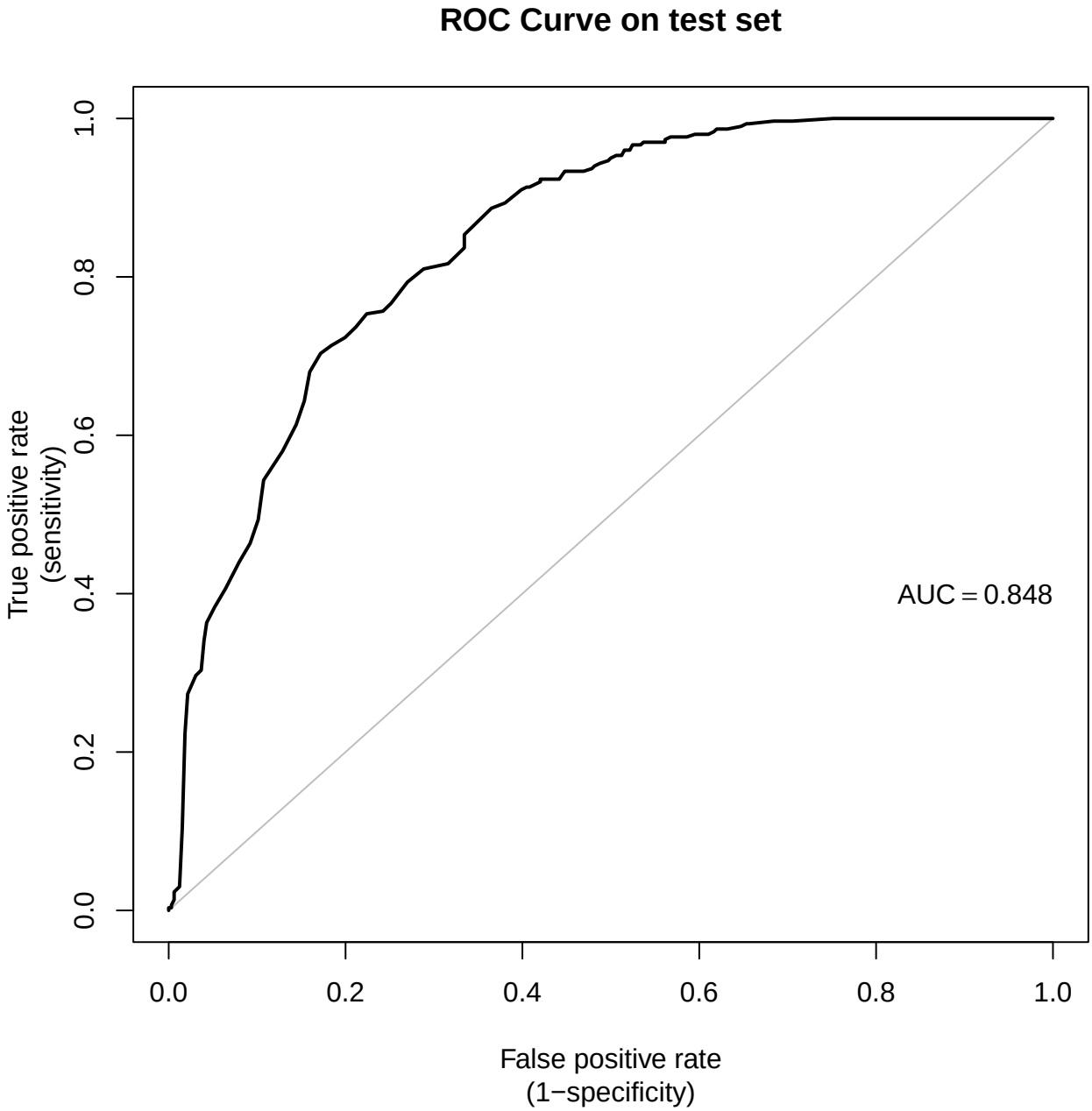
```
##                Accuracy : 0.7912
##                  95% CI : (0.7695, 0.8118)
##     No Information Rate : 0.5387
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5879
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6912
##             Specificity : 0.9080
##          Pos Pred Value : 0.8977
##          Neg Pred Value : 0.7158
##              Prevalence : 0.5387
##          Detection Rate : 0.3723
##    Detection Prevalence : 0.4148
##       Balanced Accuracy : 0.7996
##
##        'Positive' Class : 0
##
```

```
# Confusion matrix on the test set
conf_mat_test = confusionMatrix(as.factor(test_pred_lr), as.factor(y_test))
conf_mat_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 194  26
##          1 132 274
##
##                Accuracy : 0.7476
##                  95% CI : (0.7117, 0.7812)
##     No Information Rate : 0.5208
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5014
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5951
##             Specificity : 0.9133
##          Pos Pred Value : 0.8818
##          Neg Pred Value : 0.6749
##              Prevalence : 0.5208
##          Detection Rate : 0.3099
##    Detection Prevalence : 0.3514
##       Balanced Accuracy : 0.7542
##
##        'Positive' Class : 0
##
```

```
# ROC curve of train and test set
plot_roc(y_train, y_test, train_prob_lr, test_prob_lr)
```


ROC Curve on train set
True positive rate
(sensitivity)
False positive rate
(1−specificity)
AUC = 0.86
0.0
0.2
0.4
0.6
0.8
1.0




Beside the accuracy, we should care about the AUC, as we can see, on the training set, the AUC is 0.86 and on the test set the AUC is 0.848 which is slightly decrease, it shown that our model is stable.

## 4.3 Variable Selection

In this part, we use some variable selection method to find which variables have the meaning in our data.

### 4.3.1 AIC method

First of all is the AIC method.

```
# Implement stepwise selection based on AIC
step_model = step(model_lr)
```

```
## Start:  AIC=1411.73
```

```
## as.factor(Obesity) ~ Gender + Age + Height + Family_history +
##     FAVC + FCVC + NCP + CAEC + SMOKE + CH2O + SCC + FAF + TUE +
##     CALC + MTRANS
##
##                   Df Deviance    AIC
## - CH2O             1   1379.8 1409.8
## - SMOKE            1   1379.8 1409.8
## - TUE              1   1380.1 1410.1
## - Gender           1   1380.1 1410.1
## - NCP              1   1380.4 1410.4
## - CALC             1   1380.5 1410.5
## <none>                 1379.7 1411.7
## - Height           1   1383.7 1413.7
## - FAF              1   1389.6 1419.6
## - SCC              1   1401.9 1431.9
## - FCVC             1   1409.3 1439.3
## - Age              1   1420.6 1450.6
## - MTRANS           1   1426.7 1456.7
## - FAVC             1   1429.8 1459.8
## - CAEC             1   1442.1 1472.1
## - Family_history   1   1550.2 1580.2
##
## Step:  AIC=1409.75
## as.factor(Obesity) ~ Gender + Age + Height + Family_history +
##     FAVC + FCVC + NCP + CAEC + SMOKE + SCC + FAF + TUE + CALC +
##     MTRANS
##
##                   Df Deviance    AIC
## - SMOKE            1   1379.8 1407.8
## - TUE              1   1380.1 1408.1
## - Gender           1   1380.1 1408.1
## - NCP              1   1380.5 1408.5
## - CALC             1   1380.5 1408.5
## <none>                 1379.8 1409.8
## - Height           1   1383.7 1411.7
## - FAF              1   1390.0 1418.0
## - SCC              1   1401.9 1429.9
## - FCVC             1   1409.3 1437.3
## - Age              1   1420.8 1448.8
## - MTRANS           1   1426.7 1454.7
## - FAVC             1   1429.8 1457.8
## - CAEC             1   1442.6 1470.6
## - Family_history   1   1551.7 1579.7
##
## Step:  AIC=1407.83
## as.factor(Obesity) ~ Gender + Age + Height + Family_history +
##     FAVC + FCVC + NCP + CAEC + SCC + FAF + TUE + CALC + MTRANS
##
##                   Df Deviance    AIC
## - TUE              1   1380.2 1406.2
## - Gender           1   1380.2 1406.2
## - NCP              1   1380.5 1406.5
## - CALC             1   1380.6 1406.6
## <none>                 1379.8 1407.8
```

```
## - Height          1   1383.9 1409.9
## - FAF             1   1390.1 1416.1
## - SCC             1   1402.0 1428.0
## - FCVC            1   1409.4 1435.4
## - Age             1   1422.2 1448.2
## - MTRANS          1   1427.4 1453.4
## - FAVC            1   1430.0 1456.0
## - CAEC            1   1443.0 1469.0
## - Family_history  1   1551.7 1577.7
##
## Step:  AIC=1406.17
## as.factor(Obesity) ~ Gender + Age + Height + Family_history +
##     FAVC + FCVC + NCP + CAEC + SCC + FAF + CALC + MTRANS
##
##                   Df Deviance    AIC
## - Gender           1   1380.6 1404.6
## - CALC             1   1380.9 1404.9
## - NCP              1   1380.9 1404.9
## <none>                 1380.2 1406.2
## - Height           1   1384.3 1408.3
## - FAF              1   1390.6 1414.6
## - SCC              1   1402.5 1426.5
## - FCVC             1   1410.0 1434.0
## - MTRANS           1   1428.0 1452.0
## - Age              1   1428.7 1452.7
## - FAVC             1   1430.0 1454.0
## - CAEC             1   1444.0 1468.0
## - Family_history   1   1551.7 1575.7
##
## Step:  AIC=1404.57
## as.factor(Obesity) ~ Age + Height + Family_history + FAVC + FCVC +
##     NCP + CAEC + SCC + FAF + CALC + MTRANS
##
##                   Df Deviance    AIC
## - CALC             1   1381.1 1403.1
## - NCP              1   1381.3 1403.3
## <none>                 1380.6 1404.6
## - Height           1   1385.0 1407.0
## - FAF              1   1390.9 1412.9
## - SCC              1   1402.8 1424.8
## - FCVC             1   1418.0 1440.0
## - Age              1   1428.7 1450.7
## - MTRANS           1   1429.8 1451.8
## - FAVC             1   1431.3 1453.3
## - CAEC             1   1444.0 1466.0
## - Family_history   1   1554.3 1576.3
##
## Step:  AIC=1403.14
## as.factor(Obesity) ~ Age + Height + Family_history + FAVC + FCVC +
##     NCP + CAEC + SCC + FAF + MTRANS
##
##                   Df Deviance    AIC
## - NCP              1   1381.8 1401.8
## <none>                 1381.1 1403.1
```

```
## - Height          1   1385.0 1405.0
## - FAF             1   1390.9 1410.9
## - SCC             1   1403.7 1423.7
## - FCVC            1   1418.0 1438.0
## - Age             1   1428.8 1448.8
## - MTRANS          1   1429.9 1449.9
## - FAVC            1   1431.9 1451.9
## - CAEC            1   1444.0 1464.0
## - Family_history  1   1559.0 1579.0
##
## Step:  AIC=1401.81
## as.factor(Obesity) ~ Age + Height + Family_history + FAVC + FCVC +
##     CAEC + SCC + FAF + MTRANS
##
##                  Df Deviance    AIC
## <none>                1381.8 1401.8
## - Height          1   1386.4 1404.4
## - FAF             1   1391.5 1409.5
## - SCC             1   1404.8 1422.8
## - FCVC            1   1420.2 1438.2
## - Age             1   1428.8 1446.8
## - MTRANS          1   1430.1 1448.1
## - FAVC            1   1432.8 1450.8
## - CAEC            1   1444.5 1462.5
## - Family_history  1   1561.3 1579.3
```

```
summary(step_model)
```

```
##
## Call:
## glm(formula = as.factor(Obesity) ~ Age + Height + Family_history +
##     FAVC + FCVC + CAEC + SCC + FAF + MTRANS, family = binomial,
##     data = data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1469  -0.7554  -0.0541   0.8230   3.9755
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -5.3613     0.5872  -9.130  < 2e-16 ***
## Age              4.1069     0.6380   6.437 1.22e-10 ***
## Height           0.9234     0.4347   2.124 0.033662 *
## Family_history   3.5584     0.4031   8.827  < 2e-16 ***
## FAVC             1.9428     0.3081   6.306 2.86e-10 ***
## FCVC             1.5875     0.2613   6.076 1.23e-09 ***
## CAEC            -4.6084     0.6699  -6.879 6.03e-12 ***
## SCC             -2.6664     0.7598  -3.509 0.000449 ***
## FAF             -0.7969     0.2575  -3.095 0.001969 **
## MTRANS          -3.9879     0.7080  -5.633 1.77e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 2016.6  on 1460  degrees of freedom
## Residual deviance: 1381.8  on 1451  degrees of freedom
## AIC: 1401.8
##
## Number of Fisher Scoring iterations: 6
```

```
step_model$anova
```

```
##       Step Df   Deviance Resid. Df Resid. Dev      AIC
## 1          NA         NA      1445   1379.732 1411.732
## 2   - CH2O  1 0.01893654      1446   1379.751 1409.751
## 3  - SMOKE  1 0.08145431      1447   1379.832 1407.832
## 4    - TUE  1 0.33545071      1448   1380.168 1406.168
## 5 - Gender  1 0.40671926      1449   1380.575 1404.575
## 6   - CALC  1 0.56623988      1450   1381.141 1403.141
## 7    - NCP  1 0.66711179      1451   1381.808 1401.808
```

The model with the smallest AIC gives us 9 independent variables in total, which removes CH2O, SMOKE, TUE, Gender, CALC, NCP.

```
model_AIC = glm(Obesity ~ Age + Height + Family_history + FAVC + FCVC + CAEC + SCC + FAF + MTRANS, fami
```

```
# Calculate the accuracy based on optimal threshold
train_prob <- predict(model_AIC, data_train, type = 'response')
train_pred <- ifelse(train_prob > optimal_threshold, 1, 0)
train_acc <- mean(train_pred == data_train$Obesity)

test_prob <- predict(model_AIC, data_test, type = 'response')
test_pred <- ifelse(test_prob > optimal_threshold, 1, 0)
test_acc <- mean(test_pred == data_test$Obesity)

cat('Training accuracy: ', train_acc,'\n',
    'Test accuracy: ', test_acc)
```

```
## Training accuracy:  0.7864476
##  Test accuracy:  0.7412141
```

```
# Confusion matrix on training set
conf_mat_train = confusionMatrix(as.factor(train_pred), as.factor(y_train))
conf_mat_train
```

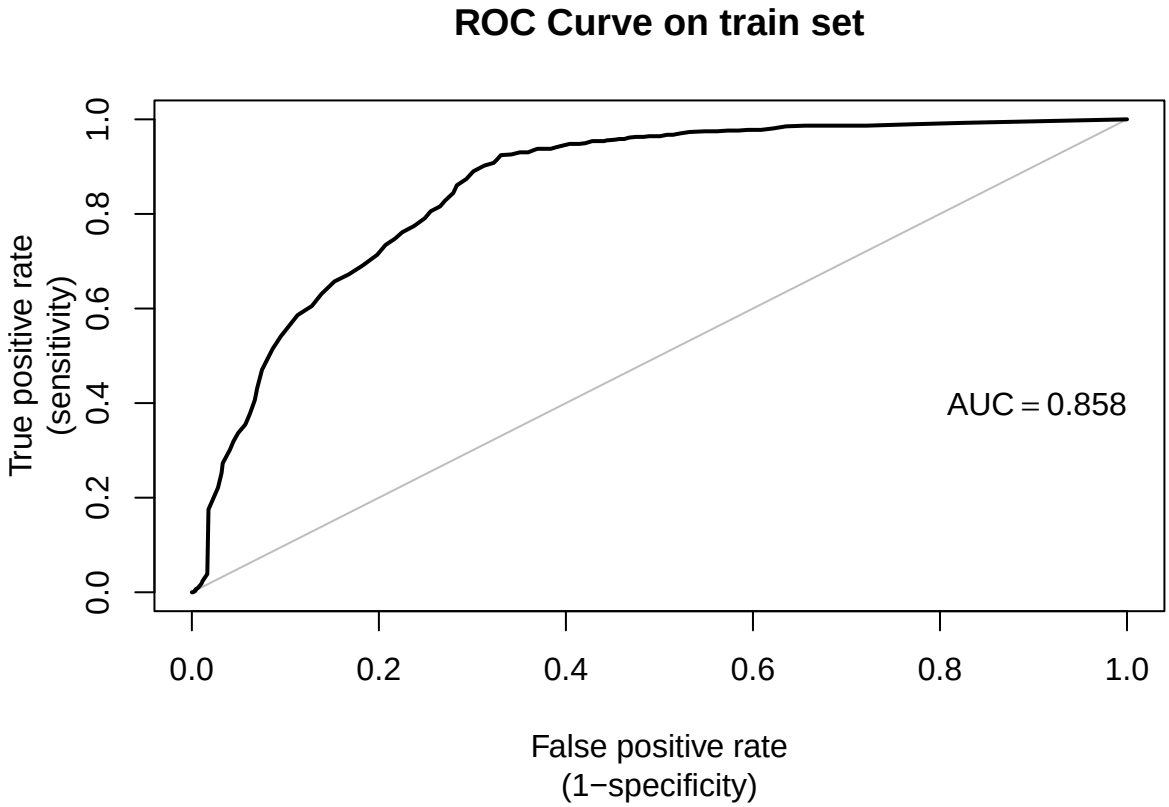
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 541  66
##          1 246 608
##
##                Accuracy : 0.7864
##                  95% CI : (0.7645, 0.8072)
##     No Information Rate : 0.5387
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5784
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

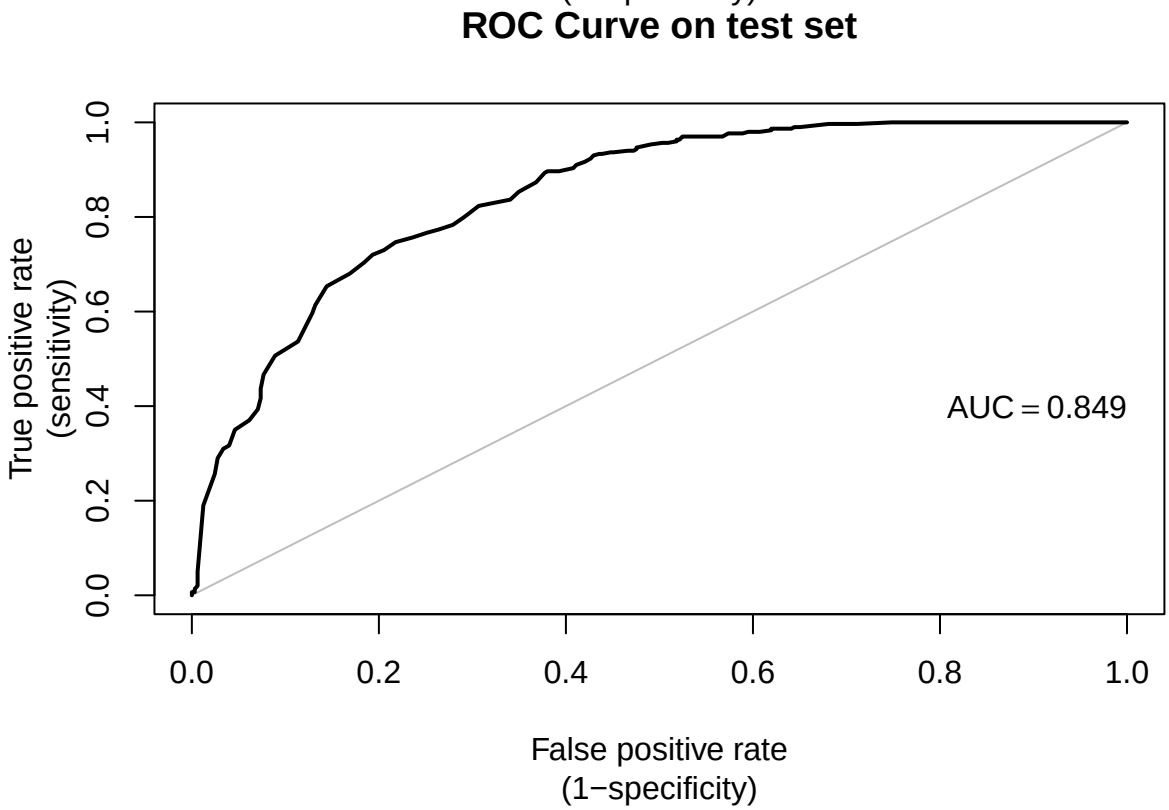
```
##             Sensitivity : 0.6874
##             Specificity : 0.9021
##          Pos Pred Value : 0.8913
##          Neg Pred Value : 0.7119
##              Prevalence : 0.5387
##          Detection Rate : 0.3703
##    Detection Prevalence : 0.4155
##       Balanced Accuracy : 0.7947
##
##        'Positive' Class : 0
##
```

```
# Confusion matrix on test set
conf_mat_test = confusionMatrix(as.factor(test_pred), as.factor(y_test))
conf_mat_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 193  29
##          1 133 271
##
##                Accuracy : 0.7412
##                  95% CI : (0.705, 0.7751)
##     No Information Rate : 0.5208
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4886
##
##  Mcnemar's Test P-Value : 5.848e-16
##
##             Sensitivity : 0.5920
##             Specificity : 0.9033
##          Pos Pred Value : 0.8694
##          Neg Pred Value : 0.6708
##              Prevalence : 0.5208
##          Detection Rate : 0.3083
##    Detection Prevalence : 0.3546
##       Balanced Accuracy : 0.7477
##
##        'Positive' Class : 0
##
```

```
# ROC curve for training and test set
plot_roc(y_train, y_test, train_prob, test_prob)
```




In this case, we can see that the AUC on the training set is 0.857 and it is lightly decreases compare with the AUC in the full model.

### 4.3.2 Best subsets method

In this part, we perfrom backward elimination by using regsubsets function. regsubsets() can be used to identify different best models of different sizes. In this case, we choose the maximum number of predictors, which is 15.

```
# Using regsubsets using backward method
regfit.full <- regsubsets(Obesity~., data=data_train, method ='backward', nvmax = 15)
reg.summary <- summary(regfit.full)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(Obesity ~ ., data = data_train, method = "backward",
##     nvmax = 15)
## 15 Variables  (and intercept)
##                 Forced in Forced out
## Gender              FALSE      FALSE
## Age                 FALSE      FALSE
## Height              FALSE      FALSE
## Family_history      FALSE      FALSE
## FAVC                FALSE      FALSE
## FCVC                FALSE      FALSE
## NCP                 FALSE      FALSE
## CAEC                FALSE      FALSE
## SMOKE               FALSE      FALSE
## CH2O                FALSE      FALSE
## SCC                 FALSE      FALSE
## FAF                 FALSE      FALSE
## TUE                 FALSE      FALSE
## CALC                FALSE      FALSE
## MTRANS              FALSE      FALSE
## 1 subsets of each size up to 15
## Selection Algorithm: backward
##           Gender Age Height Family_history FAVC FCVC NCP CAEC SMOKE CH2O SCC
## 1  ( 1 )  " "    " " " "    "*"            " "  " "  " " " " " "  " "  " "
## 2  ( 1 )  " "    " " " "    "*"            "*"  " "  " " " " " "  " "  " "
## 3  ( 1 )  " "    " " " "    "*"            "*"  " "  " " " "*"  " "  " "
## 4  ( 1 )  " "    " " " "    "*"            "*"  "*"  " " " "*"  " "  " "
## 5  ( 1 )  " "    "*" " "    "*"            "*"  "*"  " " " "*"  " "  " "
## 6  ( 1 )  " "    "*" " "    "*"            "*"  "*"  " " "*"  " "  " "
## 7  ( 1 )  " "    "*" " "    "*"            "*"  "*"  " " "*"  " "  " "
## 8  ( 1 )  " "    "*" " "    "*"            "*"  "*"  " " "*"  " "  " "
## 9  ( 1 )  " "    "*" "*"    "*"            "*"  "*"  " " "*"  " "  " "
## 10  ( 1 ) " "    "*" "*"    "*"            "*"  "*"  "*" "*"  " "  " "
## 11  ( 1 ) " "    "*" "*"    "*"            "*"  "*"  "*" "*"  " "  " "
## 12  ( 1 ) "*"    "*" "*"    "*"            "*"  "*"  "*" "*"  " "  " "
## 13  ( 1 ) "*"    "*" "*"    "*"            "*"  "*"  "*" "*"  " "  "*"
## 14  ( 1 ) "*"    "*" "*"    "*"            "*"  "*"  "*" "*"  "*"  "*"
## 15  ( 1 ) "*"    "*" "*"    "*"            "*"  "*"  "*" "*"  "*"  "*"
##           FAF TUE CALC MTRANS
## 1  ( 1 )  " " " " " "  " "
## 2  ( 1 )  " " " " " "  " "
## 3  ( 1 )  " " " " " "  " "
## 4  ( 1 )  " " " " " "  " "
## 5  ( 1 )  " " " " " "  " "
```

```
## 6  ( 1 )  " " " " " " "  "*"
## 7  ( 1 )  " " " " " " "  "*"
## 8  ( 1 )  "*" " " " " "  "*"
## 9  ( 1 )  "*" " " " " "  "*"
## 10  ( 1 ) "*" " " " " "  "*"
## 11  ( 1 ) "*" "*" " " "  "*"
## 12  ( 1 ) "*" "*" " " "  "*"
## 13  ( 1 ) "*" "*" " " "  "*"
## 14  ( 1 ) "*" "*" " " "  "*"
## 15  ( 1 ) "*" "*" "*" "  "*"
```

The output above shows that the most important variable is family_history, which indicates that the family has history of overweight or not. The following 4 important variables are FAVC (Frequent consumption of high caloric food), CAEC (Consumption of food between meals), FCVC (Frequency of consumption of vegetables) and Age. We can see that 3 out of 5 most important variables based on best subsets selection related to eating habits, which indicates that diet directly affect weight problem.

Moreover, the function return up to the best 15-variables model. To identify which of these best models should we finally choose for our predictive analytics, we need to use some statistical metrics to compare the overall performance of the models and to choose the best one. By using some model selection criteria like Adjusted R-squared, Cp and BIC, we have the following results.

```
# Plotting RSS, adjusted  R2,  Cp, and BIC for all of the models at once will help us
# decide which model to select.

par(mfrow=c(2,2))

# residual sum of squares
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")

# adjusted-R^2 with its largest value
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
which.max(reg.summary$adjr2)
```

```
## [1] 12
```

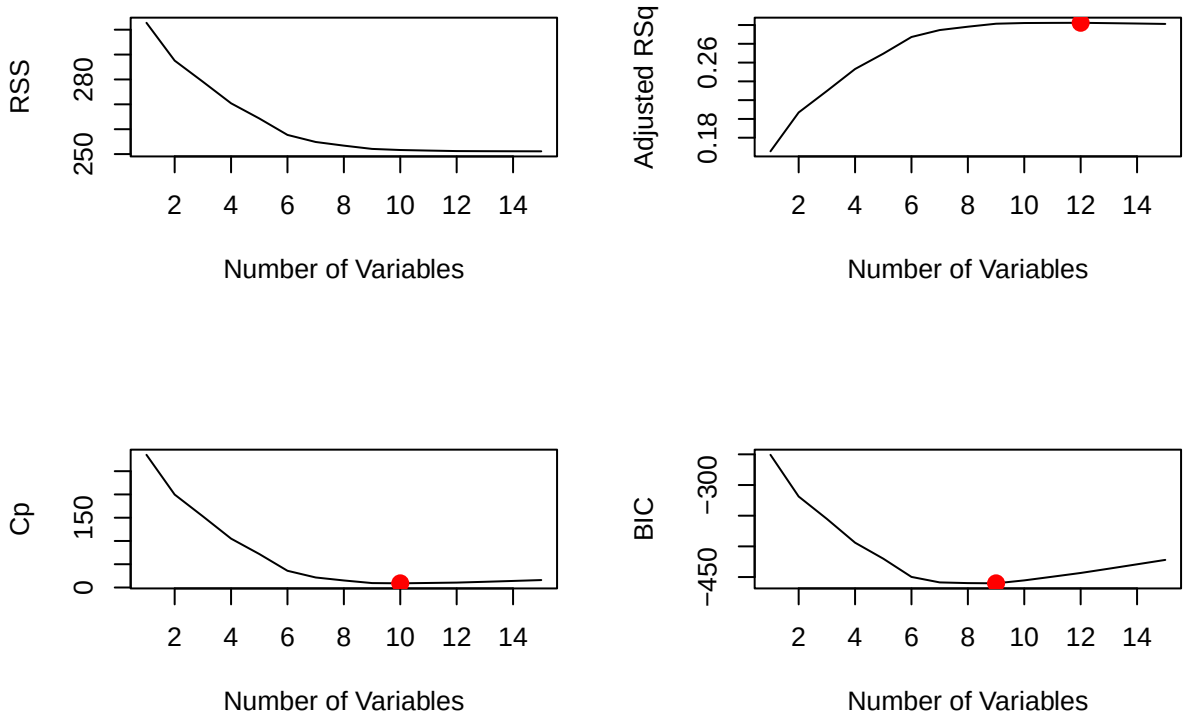
```
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.summary$adjr2)],
       col="red",cex=2,pch=20)

# Mallow's Cp with its smallest value
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
which.min(reg.summary$cp)
```

```
## [1] 10
```

```
points(which.min(reg.summary$cp),reg.summary$cp[which.min(reg.summary$cp)],
       col="red",cex=2,pch=20)

# BIC with its smallest value
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
which.min(reg.summary$bic)
```

```
## [1] 9
```

```
points(which.min(reg.summary$bic),reg.summary$bic[which.min(reg.summary$bic)],
       col="red",cex=2,pch=20)
```

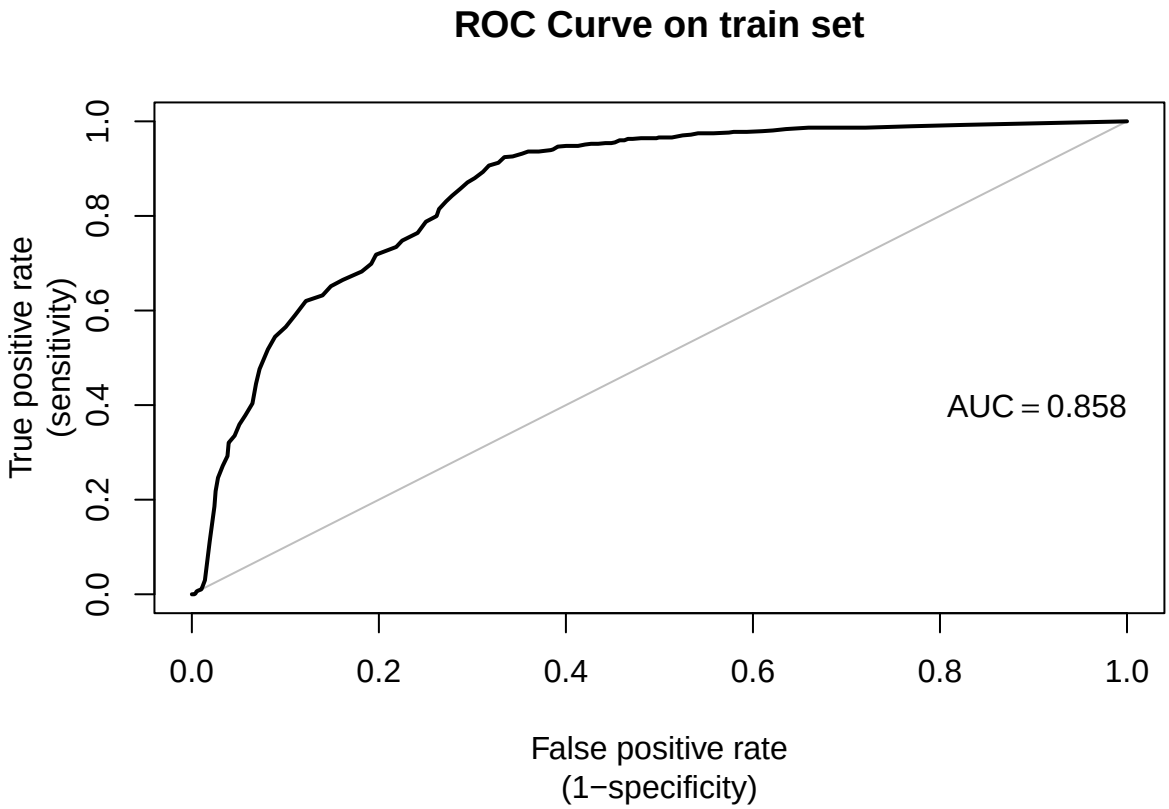


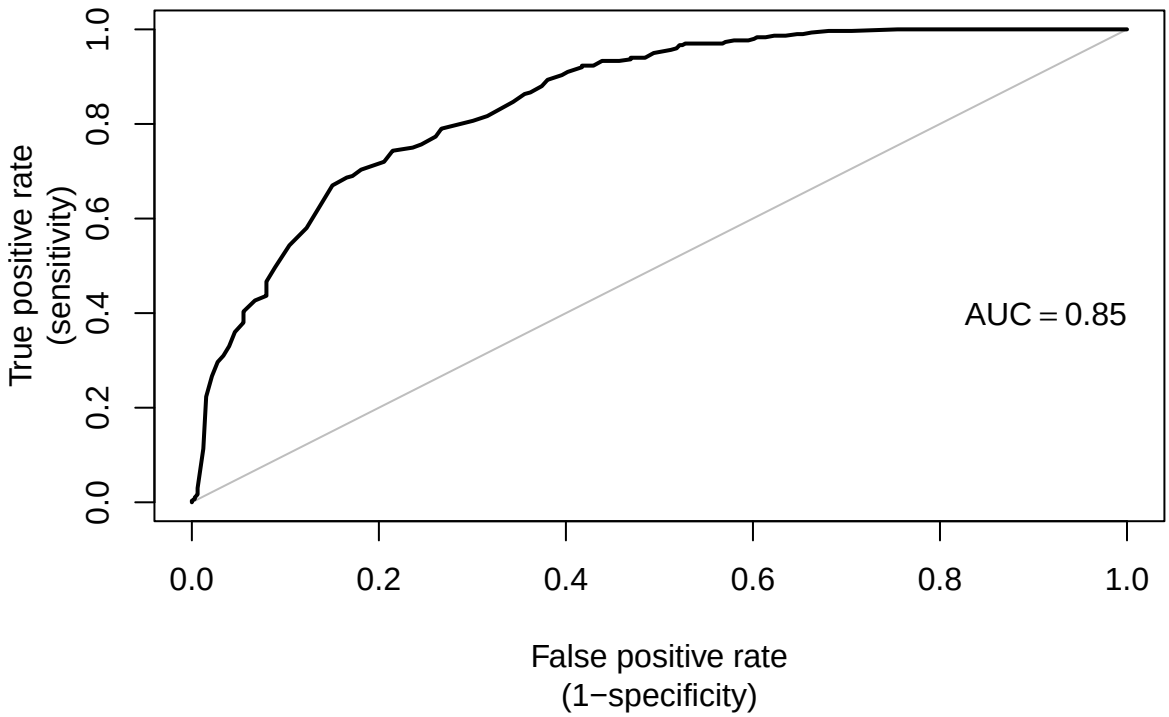

Based on the plot above, we can see that adjusted R-squared tells us that the best model is the one with 12 predictor variables. However, using the Cp criteria, we got a model with 10 predictor variables and with the BIC, we got 9 predictor variables. To evaluate the performance of each "best" model, we run the code below.

```
# Fit the model with 12 independent variables
train.glm12 <- glm(Obesity ~ Gender + Age + Height + Family_history + FAVC +
                     FCVC + NCP + SCC + FAF + TUE + CAEC + MTRANS,
                   family = binomial, data=data_train)
                 # remove NCP, SMOKE, CH2O, CALC

train_prob12 <- predict(train.glm12, data_train, type = 'response')
train_pred12 <- ifelse(train_prob12 > optimal_threshold, 1, 0)
train_acc12 <- mean(train_pred12 == data_train$Obesity)

test_prob12 <- predict(train.glm12, data_test, type = 'response')
test_pred12 <- ifelse(test_prob12 > optimal_threshold, 1, 0)
test_acc12 <- mean(test_pred12 == data_test$Obesity)

cat('Training accuracy: ', train_acc12,'\n',
    'Test accuracy: ', test_acc12)
```

```
## Training accuracy:  0.7830253
##  Test accuracy:  0.7476038
```

```
plot_roc(y_train, y_test, train_prob12, test_prob12)
```



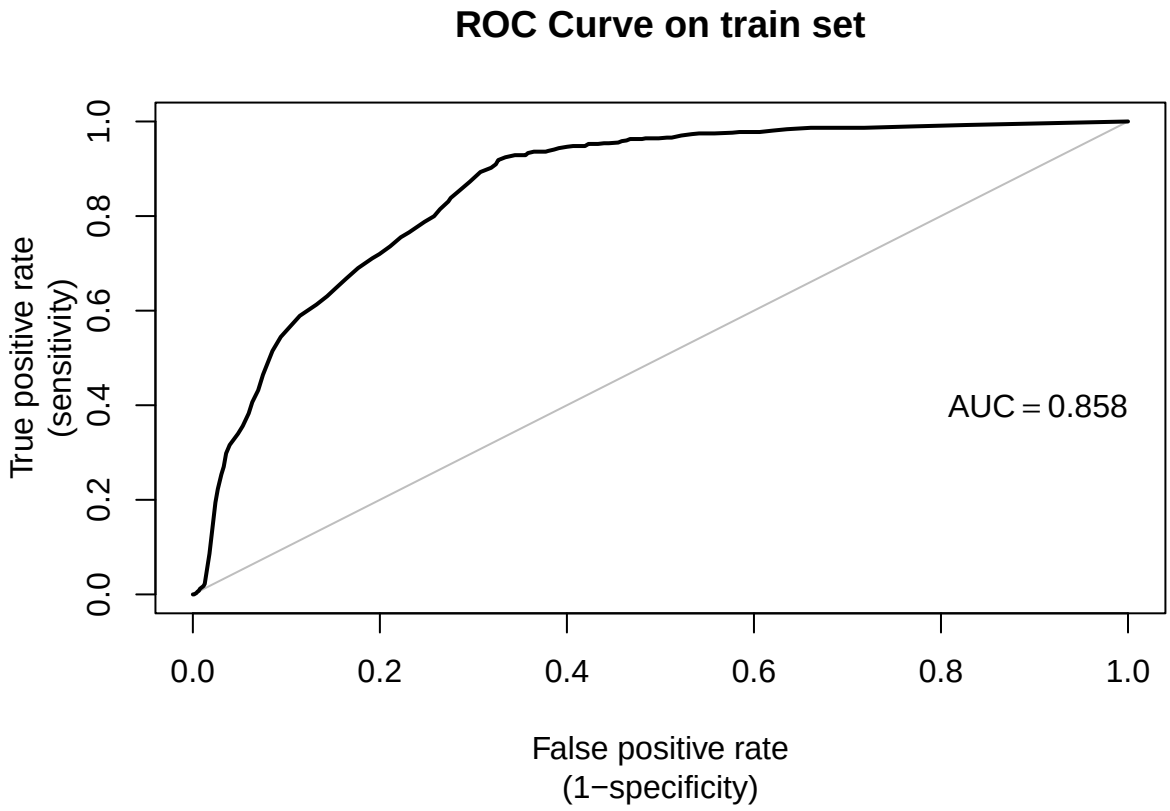



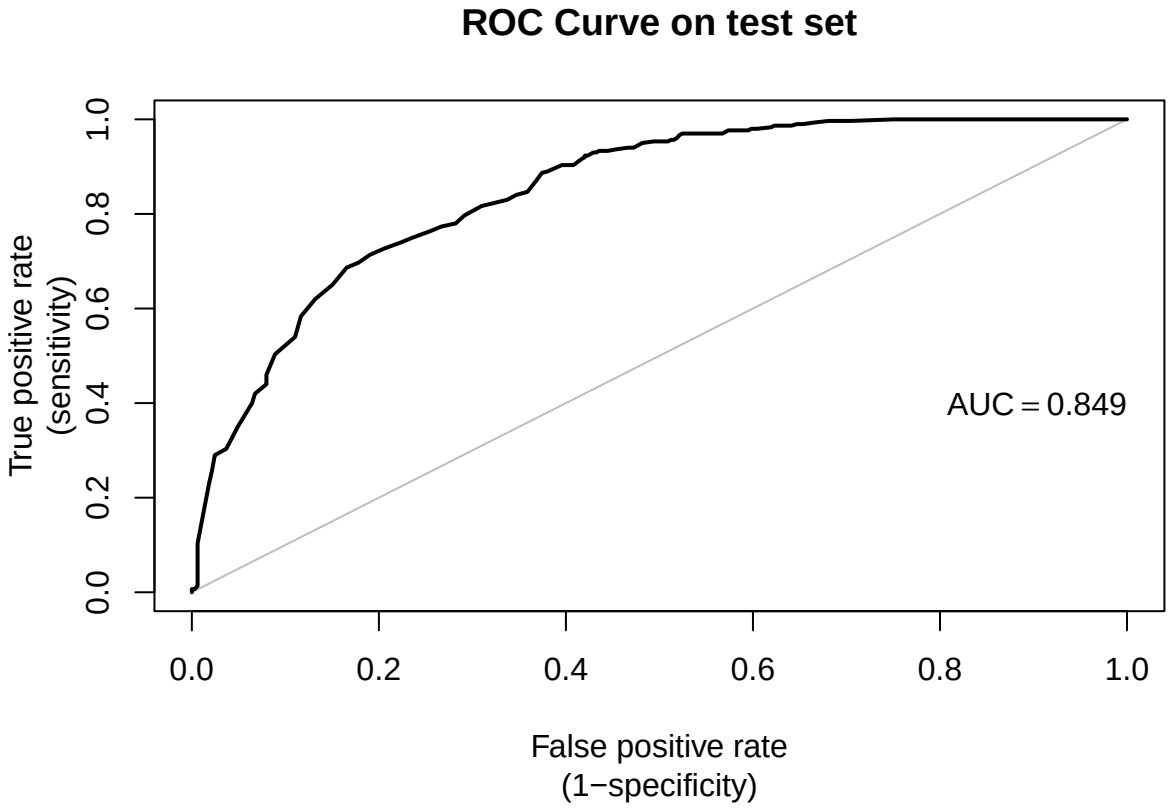

```
# Fit the model with 10 independent variables
train.glm10 <- glm(Obesity ~ Age + Height + Family_history + FAVC + FCVC +
                     NCP + SCC + FAF + CAEC + MTRANS,
```

```
                    family = binomial, data=data_train)

train_prob10 <- predict(train.glm10, data_train, type = 'response')
train_pred10 <- ifelse(train_prob10 > optimal_threshold, 1, 0)
train_acc10 <- mean(train_pred10 == data_train$Obesity)

test_prob10 <- predict(train.glm10, data_test, type = 'response')
test_pred10 <- ifelse(test_prob10 > optimal_threshold, 1, 0)
test_acc10 <- mean(test_pred10 == data_test$Obesity)

cat('Training accuracy: ', train_acc10,'\n',
    'Test accuracy: ', test_acc10)
```

```
## Training accuracy:  0.7830253
##  Test accuracy:  0.7412141
```

```
plot_roc(y_train, y_test, train_prob10, test_prob10)
```

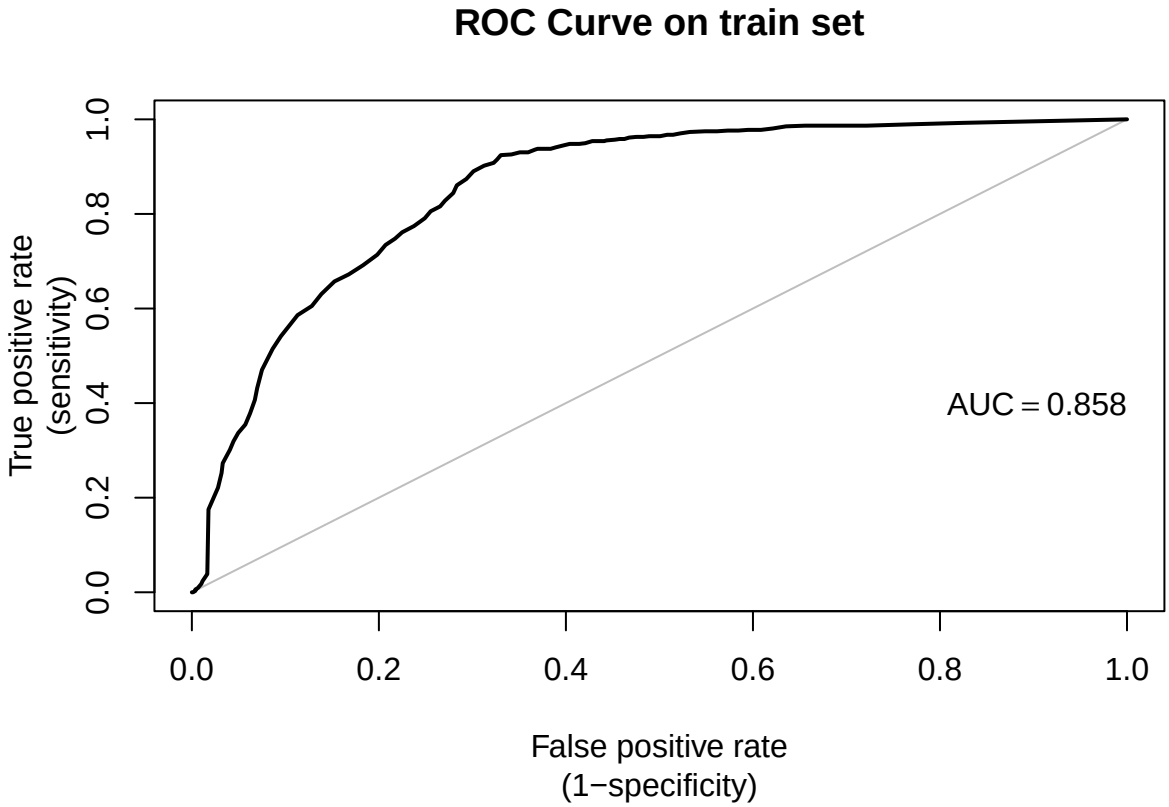


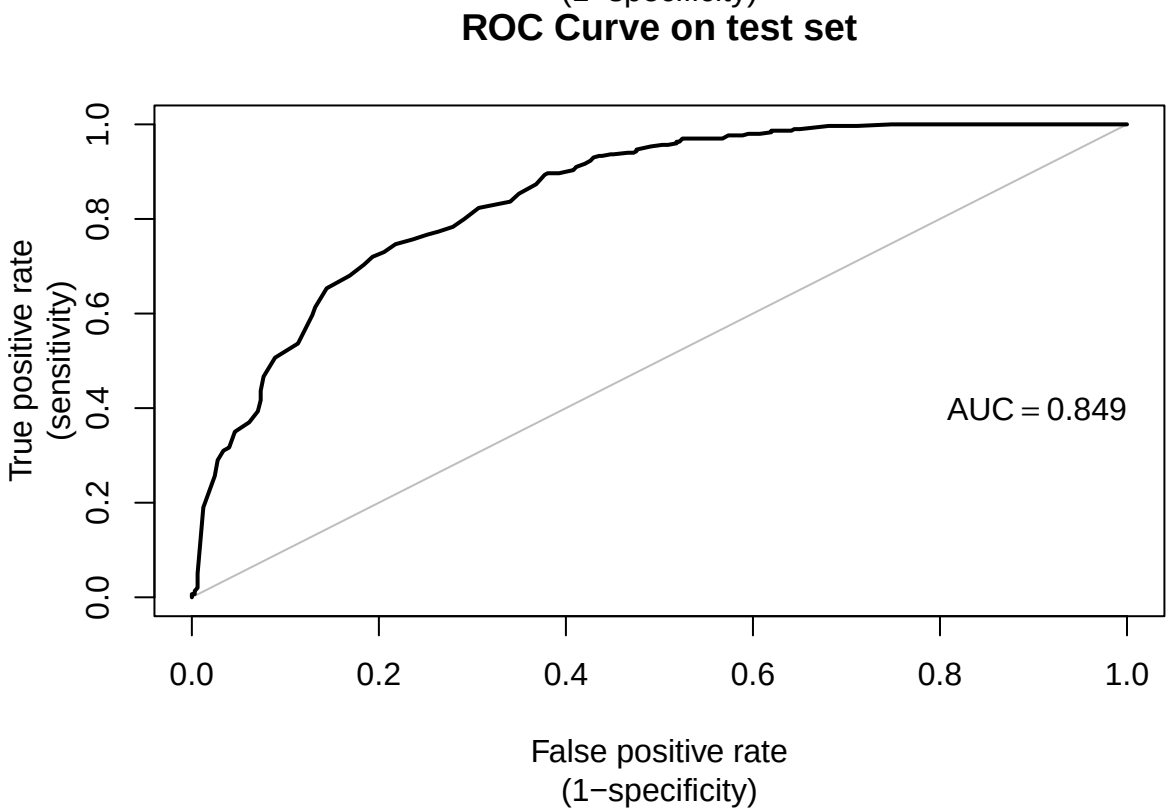



```
train.glm9 <- glm(Obesity ~ Age + Height + Family_history + FAVC + FCVC + CAEC +
                    SCC + FAF + MTRANS,
                  family = binomial, data=data_train)
                # remove Gender, NCP, SMOKE, Height, TUE, CALC

train_prob9 <- predict(train.glm9, data_train, type = 'response')
train_pred9 <- ifelse(train_prob9 > optimal_threshold, 1, 0)
train_acc9 <- mean(train_pred9 == data_train$Obesity)

test_prob9 <- predict(train.glm9, data_test, type = 'response')
test_pred9 <- ifelse(test_prob9 > optimal_threshold, 1, 0)
test_acc9 <- mean(test_pred9 == data_test$Obesity)

cat('Training accuracy: ', train_acc9,'\n',
    'Test accuracy: ', test_acc9)
```

```
## Training accuracy:  0.7864476
##  Test accuracy:  0.7412141
```

```
plot_roc(y_train, y_test, train_prob9, test_prob9)
```

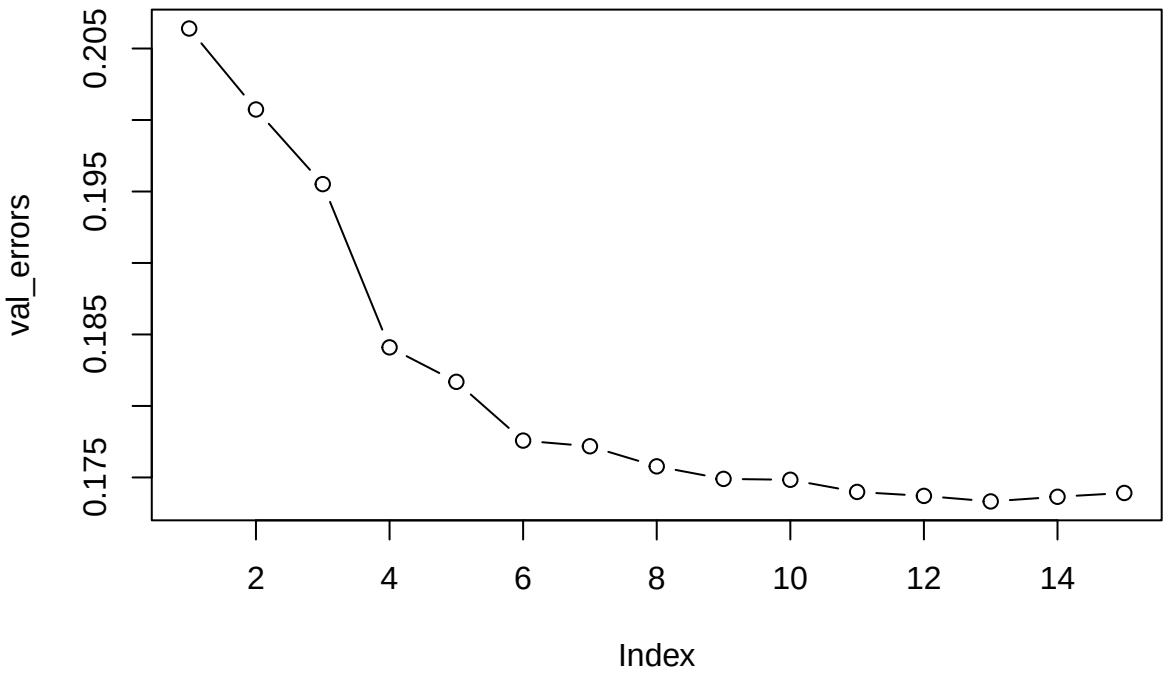



As the result, the model with 12 predictor variables gives us the highest accuracy among training and test set as well as the highest AUC value from the test set.

### 4.3.3 Cross-validation method

```
### CROSS-VALIDATION
model_cv <- regsubsets(Obesity ~ ., data = data_train, nvmax = 15)
model_matrix <- model.matrix(Obesity ~ ., data = data_test)
val_errors <- rep(NA, 15)
for (i in 1:15){
  coefi <- coef(model_cv, id = i)
  pred <- model_matrix[,names(coefi)] %*% coefi
  val_errors[i] <- mean((y_test - pred)^2)
}

plot(val_errors, type = 'b')
```




Above is the result by using cross-validation method. The model gives us the smallest validation error is the model with 13 variables and the corresponding coefficients.

```
# Fit the model with 13 variables
model_cv <- glm(Obesity ~ Gender + Age + Height + Family_history + FAVC +
                    FCVC + NCP + SCC + FAF + TUE + CAEC + MTRANS + CH2O,
                  data=data_train, family='binomial')
summary(model_cv)
```

```
##
## Call:
## glm(formula = Obesity ~ Gender + Age + Height + Family_history +
##     FAVC + FCVC + NCP + SCC + FAF + TUE + CAEC + MTRANS + CH2O,
##     family = "binomial", data = data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1596  -0.7538  -0.0532   0.8152   3.9878
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -5.33519    0.60467  -8.823  < 2e-16 ***
## Gender         -0.09242    0.18491  -0.500 0.617209
## Age             4.07674    0.66526   6.128 8.90e-10 ***
## Height          1.05680    0.56534   1.869 0.061579 .
## Family_history  3.54194    0.40371   8.773  < 2e-16 ***
## FAVC            1.94638    0.30981   6.282 3.33e-10 ***
## FCVC            1.50614    0.28329   5.317 1.06e-07 ***
## NCP             0.22170    0.27490   0.806 0.419970
## SCC            -2.65369    0.76157  -3.485 0.000493 ***
## FAF            -0.78765    0.26027  -3.026 0.002475 **
## TUE            -0.12760    0.23401  -0.545 0.585557
## CAEC           -4.66031    0.68498  -6.804 1.02e-11 ***
## MTRANS         -4.01857    0.71836  -5.594 2.22e-08 ***
## CH2O           -0.05816    0.23806  -0.244 0.807002
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2016.6  on 1460  degrees of freedom
## Residual deviance: 1380.5  on 1447  degrees of freedom
## AIC: 1408.5
##
## Number of Fisher Scoring iterations: 6
```

```
train_cv <- predict(model_cv, data_train, type = 'response')
train_pred_cv <- ifelse(train_cv > optimal_threshold, 1, 0)
train_acc_cv <- mean(train_pred_cv == data_train$Obesity)

test_cv <- predict(model_cv, data_test, type = 'response')
test_pred_cv <- ifelse(test_cv > optimal_threshold, 1, 0)
test_acc_cv <- mean(test_pred_cv == data_test$Obesity)

cat('Training accuracy: ', train_acc_cv,'\n',
    'Test accuracy: ', test_acc_cv)
```

```
## Training accuracy:  0.7830253
##  Test accuracy:  0.7476038
```

```
# Confusion matrix on train set
conf_mat_test = confusionMatrix(as.factor(train_pred_cv), as.factor(y_train))
conf_mat_test
```

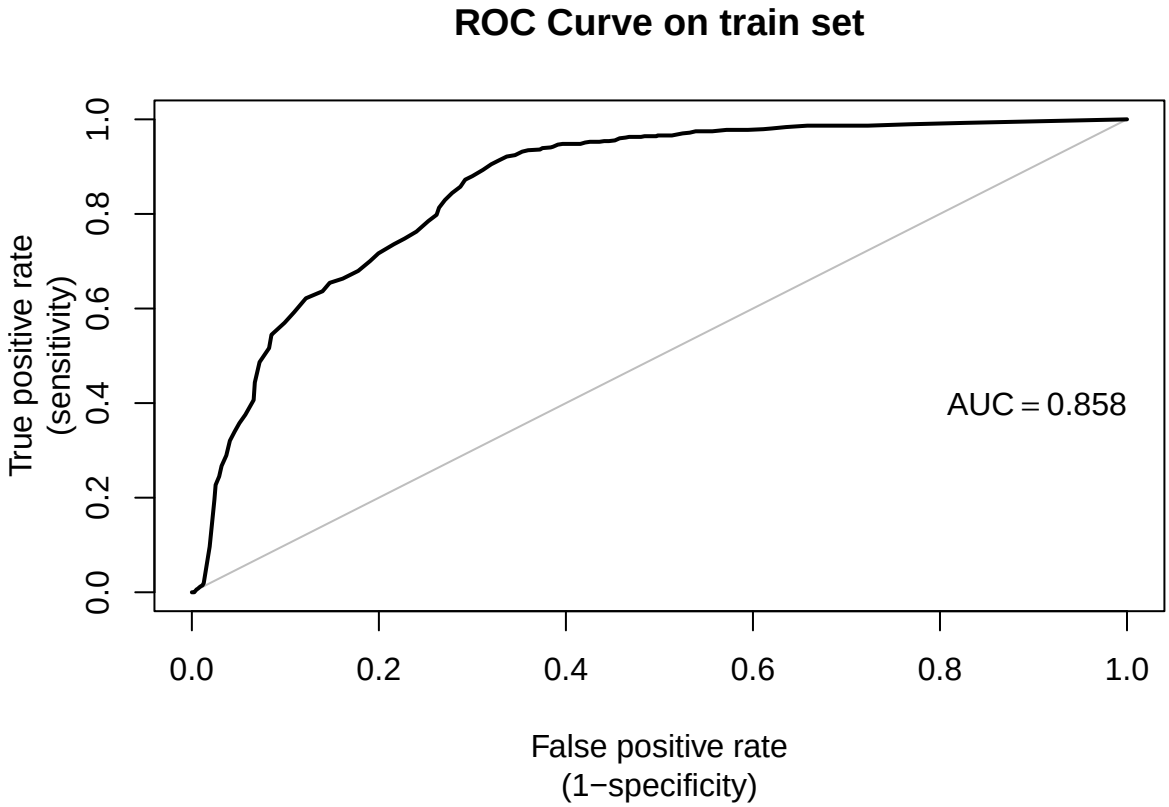
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 542  72
##          1 245 602
##
##                Accuracy : 0.783
##                  95% CI : (0.761, 0.8039)
##     No Information Rate : 0.5387
##     P-Value [Acc > NIR] : < 2.2e-16
```

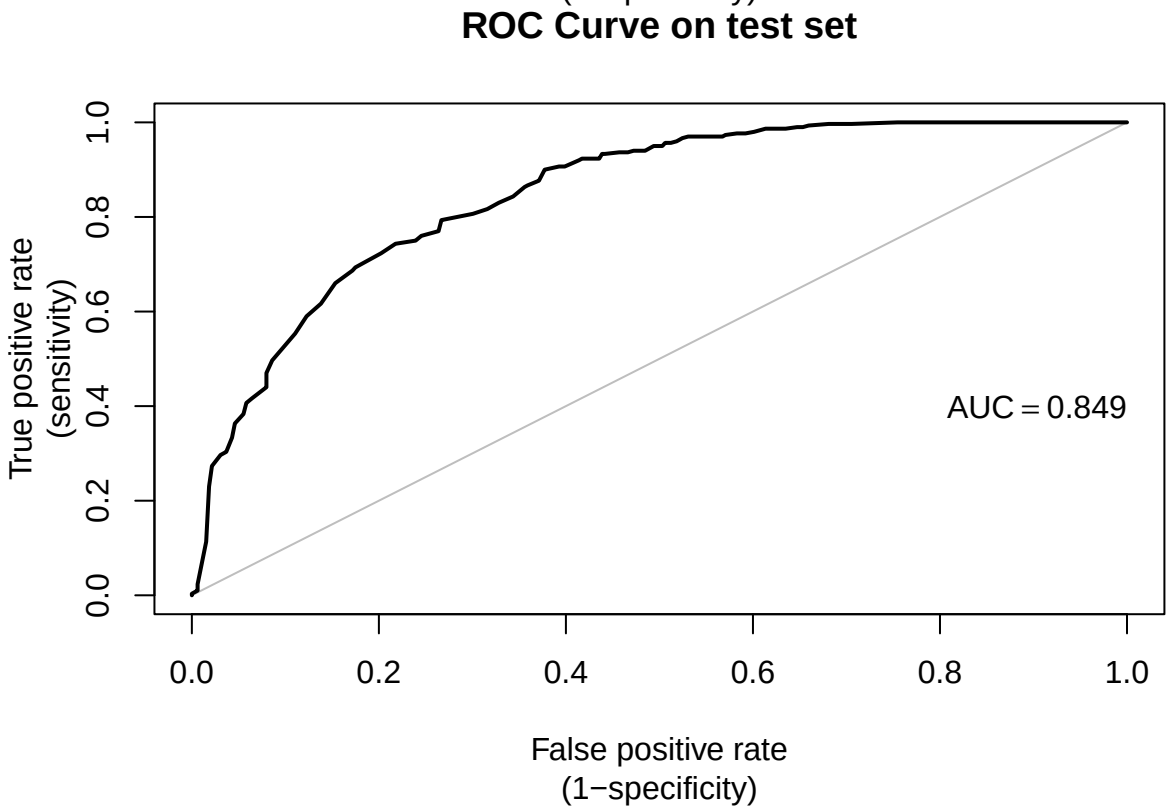
```
##
##                  Kappa : 0.5713
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6887
##             Specificity : 0.8932
##          Pos Pred Value : 0.8827
##          Neg Pred Value : 0.7107
##              Prevalence : 0.5387
##          Detection Rate : 0.3710
##    Detection Prevalence : 0.4203
##       Balanced Accuracy : 0.7909
##
##        'Positive' Class : 0
##
```

```
# Confusion matrix on test set
conf_mat_test = confusionMatrix(as.factor(test_pred_cv), as.factor(y_test))
conf_mat_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 196  28
##          1 130 272
##
##                Accuracy : 0.7476
##                  95% CI : (0.7117, 0.7812)
##     No Information Rate : 0.5208
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5011
##
##  Mcnemar's Test P-Value : 9.348e-16
##
##             Sensitivity : 0.6012
##             Specificity : 0.9067
##          Pos Pred Value : 0.8750
##          Neg Pred Value : 0.6766
##              Prevalence : 0.5208
##          Detection Rate : 0.3131
##    Detection Prevalence : 0.3578
##       Balanced Accuracy : 0.7539
##
##        'Positive' Class : 0
##
```

```
plot_roc(y_train, y_test, train_cv, test_cv)
```




### 4.4 Regularization

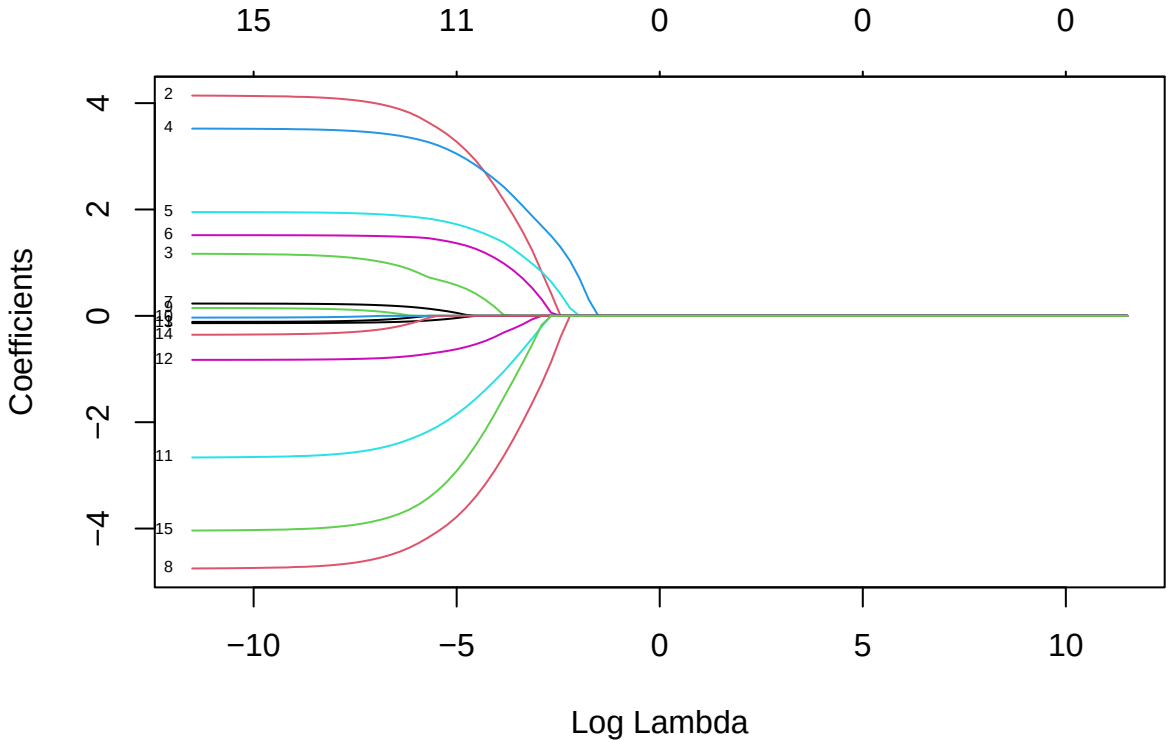
In this part, we will do some regularization methods to evaluate the performance on our data.

```r
# Prepare data for Regularization
X_train = as.matrix(x_train)
X_test = as.matrix(x_test)
X = as.matrix(data %>% select(1:15))
y = data %>% select(16)
y = y[,1]

grid = 10^seq(5, -5, length=100)
```

### 4.4.1 Lasso Regression (L1 Regularization)

We first illustrate Lasso regression, which can be fit using glmnet() with alpha = 1.

```r
# Fit lasso model on training data
lasso_mod = glmnet(X_train, as.factor(y_train),
                   alpha=1, #alpha=1 for lasso regression
                   lambda=grid, family="binomial")

# Plot the coefficients varies by lambda
plot(lasso_mod, label=TRUE, xvar = 'lambda') #draw plot of coefficients
```



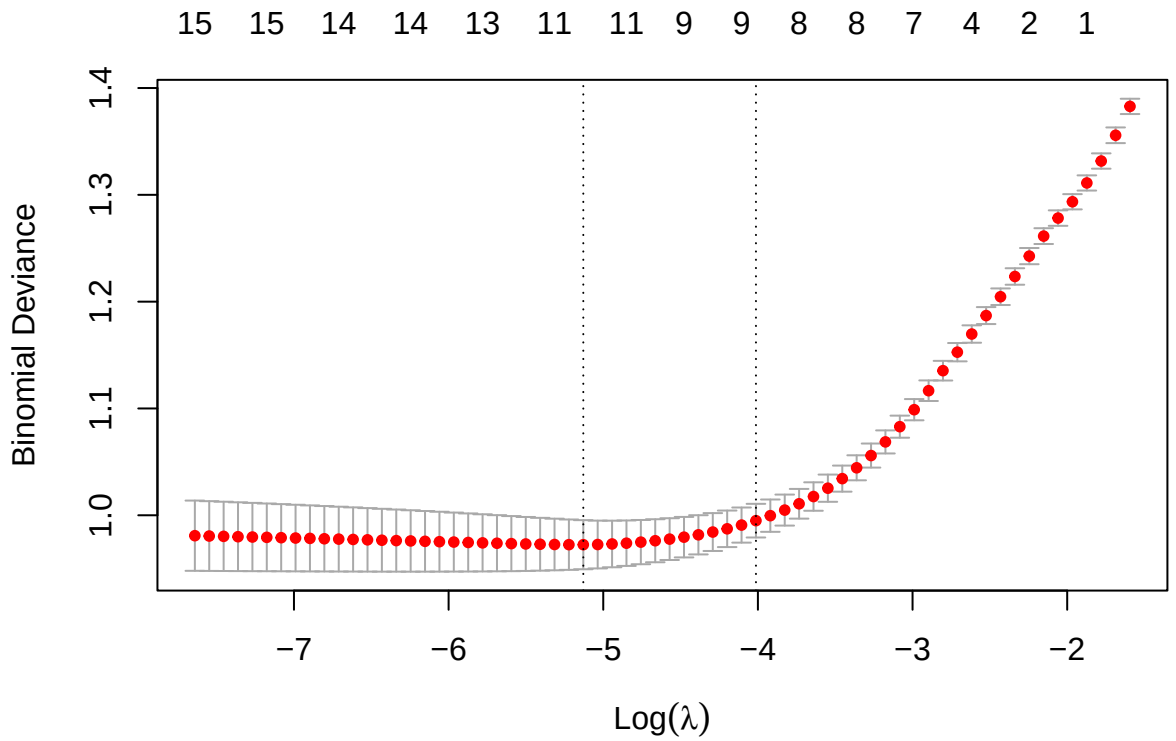

The graph above shows how much the coefficients of the model are penalized for different values of lambda. Each curve corresponds to a variable and the number begins at each curve corresponds to the position of that variable on the dataset. Notice along the top is the number of features in the model, in this case, the variables decrease.

We use cross-validation to select a good lambda value.

```r
cv_out_lasso = cv.glmnet(X_train, y_train, alpha=1, family='binomial')
best_lambda_lasso = cv_out_lasso$lambda.min
cat('Optimal lambda: ', best_lambda_lasso)
```

```
## Optimal lambda:  0.005922261
```

```
plot(cv_out_lasso)
```




This plots the cross-validation curve (red dotted line) along with upper and lower standard deviation curves along the lambda sequence. Two special values along the lambda sequence are indicated by the vertical dotted lines, the left indicates the value of lambda that gives minimum mean cross-validated error.

```
lasso_prob_train = predict(lasso_mod, s=best_lambda_lasso, newx=X_train, type='response')
lasso_pred_train = ifelse(lasso_prob_train > optimal_threshold, 1, 0)
acc_train = mean(lasso_pred_train == y_train)
cat('Accuracy of training data: ',acc_train,'\n')
```

```
## Accuracy of training data:  0.7864476
```

```
cat('MSE of training data: ',mean((lasso_pred_train - y_train)^2))
```

```
## MSE of training data:  0.2135524
```

```
lasso_prob_test = predict(lasso_mod, s=best_lambda_lasso, newx=X_test, type='response')
lasso_pred_test = ifelse(lasso_prob_test > optimal_threshold, 1, 0)
acc_test = mean(lasso_pred_test == y_test)
cat('Accuracy of test data: ',acc_test,'\n')
```

```
## Accuracy of test data:  0.7428115
```

```
cat('MSE of test data: ',mean((lasso_pred_test - y_test)^2))
```

```
## MSE of test data:  0.2571885
```

```
# Confusion matrix on train set
conf_mat_test = confusionMatrix(as.factor(lasso_pred_train), as.factor(y_train))
conf_mat_test
```

```
## Confusion Matrix and Statistics
```

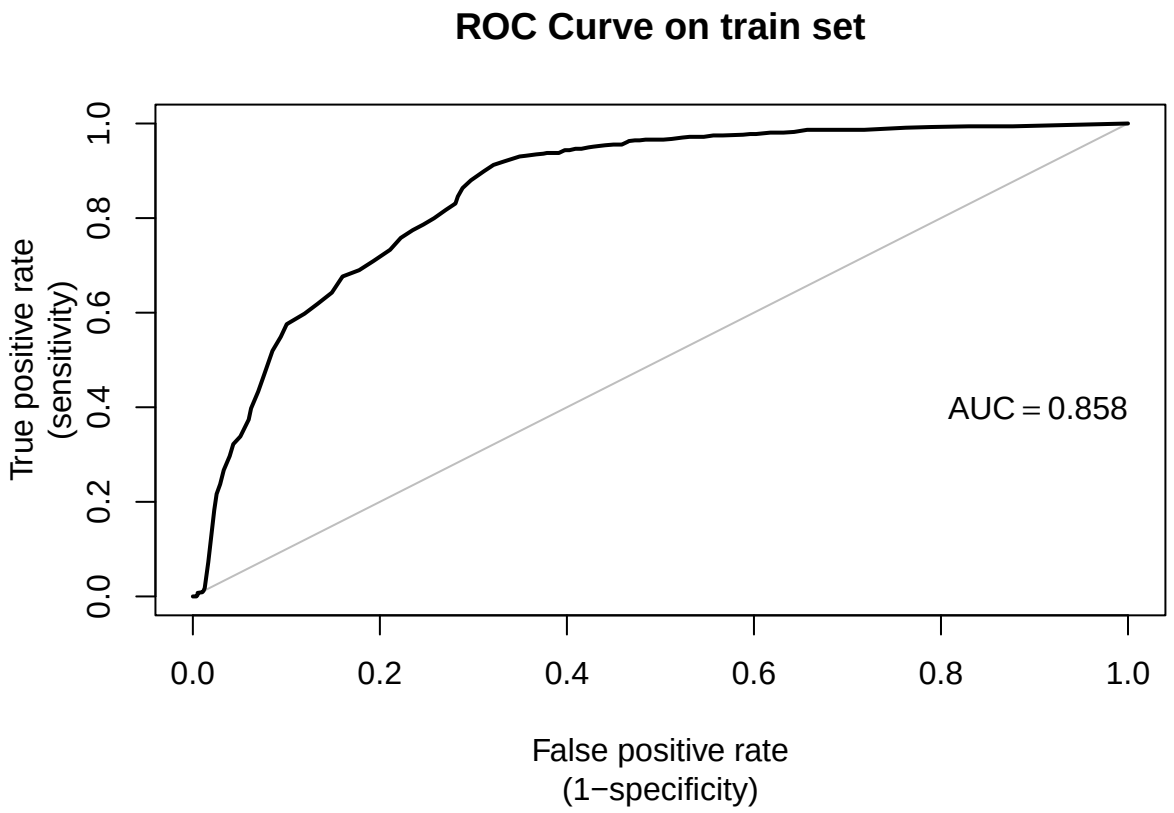
```
##
##           Reference
## Prediction   0   1
##          0 534  59
##          1 253 615
##
##                Accuracy : 0.7864
##                  95% CI : (0.7645, 0.8072)
##     No Information Rate : 0.5387
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.579
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6785
##             Specificity : 0.9125
##          Pos Pred Value : 0.9005
##          Neg Pred Value : 0.7085
##              Prevalence : 0.5387
##          Detection Rate : 0.3655
##    Detection Prevalence : 0.4059
##       Balanced Accuracy : 0.7955
##
##        'Positive' Class : 0
##
```

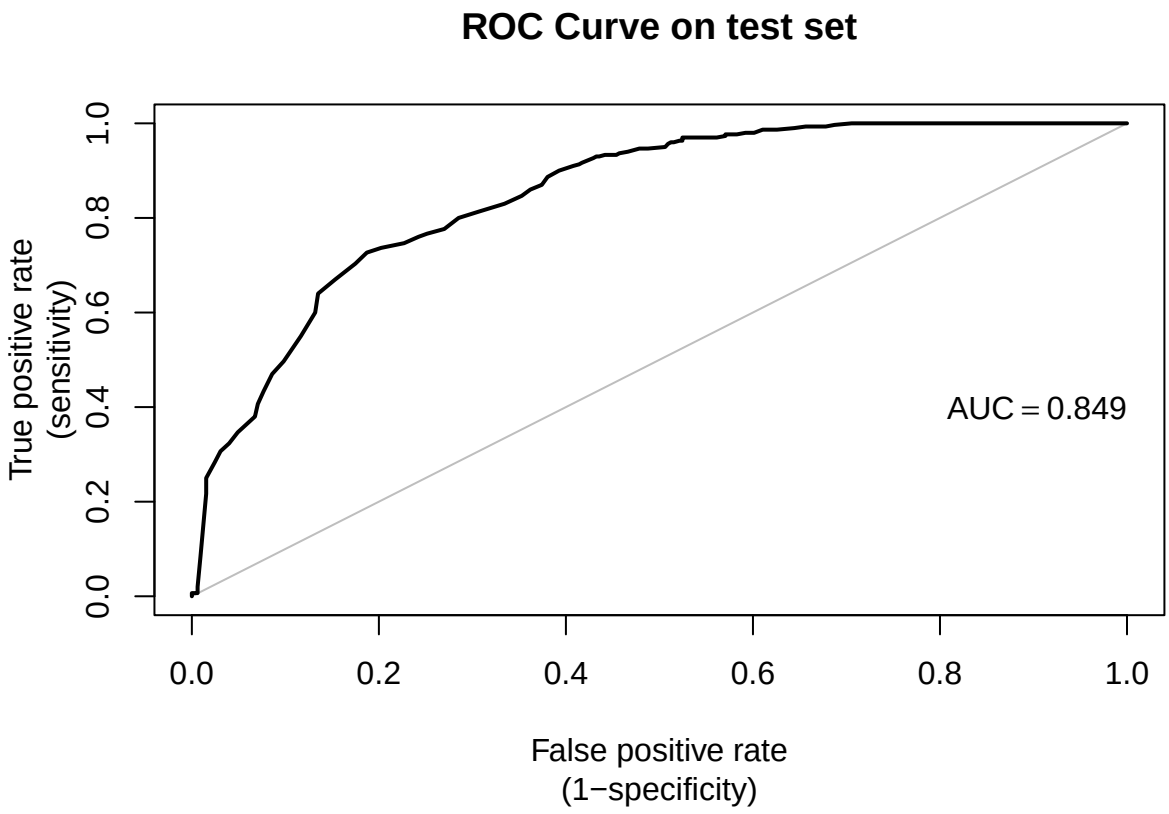
```
# Confusion matrix on test set
conf_mat_test = confusionMatrix(as.factor(lasso_pred_test), as.factor(y_test))
conf_mat_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 191  26
##          1 135 274
##
##                Accuracy : 0.7428
##                  95% CI : (0.7067, 0.7766)
##     No Information Rate : 0.5208
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4921
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5859
##             Specificity : 0.9133
##          Pos Pred Value : 0.8802
##          Neg Pred Value : 0.6699
##              Prevalence : 0.5208
##          Detection Rate : 0.3051
##    Detection Prevalence : 0.3466
##       Balanced Accuracy : 0.7496
```

```
##
##        'Positive' Class : 0
##
```

```
plot_roc(y_train, y_test, as.numeric(lasso_prob_train), as.numeric(lasso_prob_test))
```

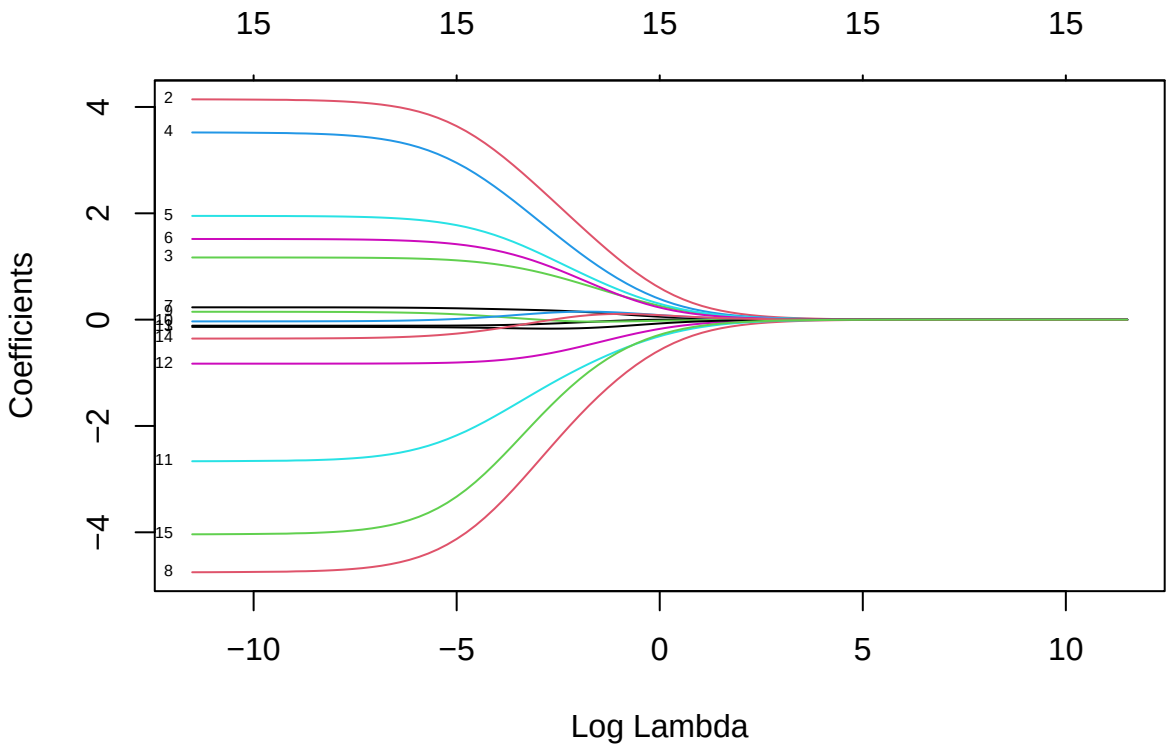





#### 4.4.2 Ridge regression (L2 regularization)

We now illustrate the Ridge regression, which can be fit using glmnet() with alpha = 0.

```
ridge_mod = glmnet(X_train, y_train, alpha=0, lambda=grid, family='binomial')
# Plot the coefficients varies by lambda
plot(ridge_mod, label=TRUE, xvar='lambda')
```

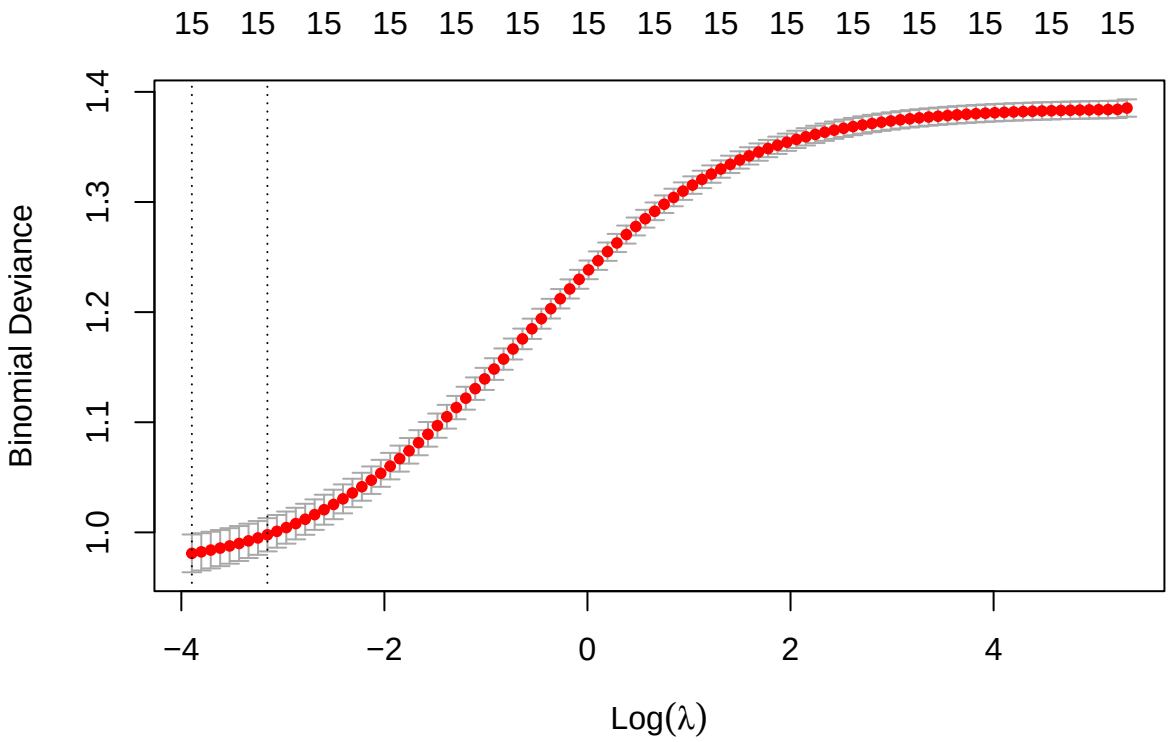



This plot illustrates how much the coefficients are penalized for different values of lambda. Again, to actually pick a lambda, we will use cross-validation. The plot is similar to the lasso plot. Notice along the top is the number of features in the model, in this case, all variables are kept, which makes it different with the lasso regression.

```
#Using CV to choose the tunning parameter lambda
cv.out = cv.glmnet(X_train, y_train, alpha=0, family='binomial')
best_lambda = cv.out$lambda.min #Select lambda that minimizes training MSE
cat('Optimal lambda: ', best_lambda)
```

```
## Optimal lambda:  0.02031613
```

```
plot(cv.out) #Draw plot of training MSE as a function of lambda
```




The plot displays the cross-validation error according to the log of lambda, the left dashed vertical line indicates that the log of the optimal value of lambda approximately -4, which is the one that minimizes the prediction error. This lambda value will give the most accurate model.

```
ridge_prob_train = predict(ridge_mod, s=best_lambda, newx=X_train, type='response')
ridge_pred_train = ifelse(ridge_prob_train > optimal_threshold, 1, 0)
acc_train = mean(ridge_pred_train == y_train)
cat('Accuracy of training data: ',acc_train,'\n')
```

```
## Accuracy of training data:  0.7754962
```

```
cat('MSE of training data: ',mean((ridge_pred_train - y_train)^2))
```

```
## MSE of training data:  0.2245038
```

```
ridge_prob_test = predict(ridge_mod, s=best_lambda, newx=X_test, type='response')
ridge_pred_test = ifelse(ridge_prob_test > optimal_threshold, 1, 0)
acc_test = mean(ridge_pred_test == y_test)
cat('Accuracy of test data: ',acc_test,'\n')
```

```
## Accuracy of test data:  0.7492013
```

```
cat('MSE of test data: ',mean((ridge_pred_test - y_test)^2))
```

```
## MSE of test data:  0.2507987
```

```
# Confusion matrix on train set
conf_mat_test = confusionMatrix(as.factor(ridge_pred_train), as.factor(y_train))
conf_mat_test
```

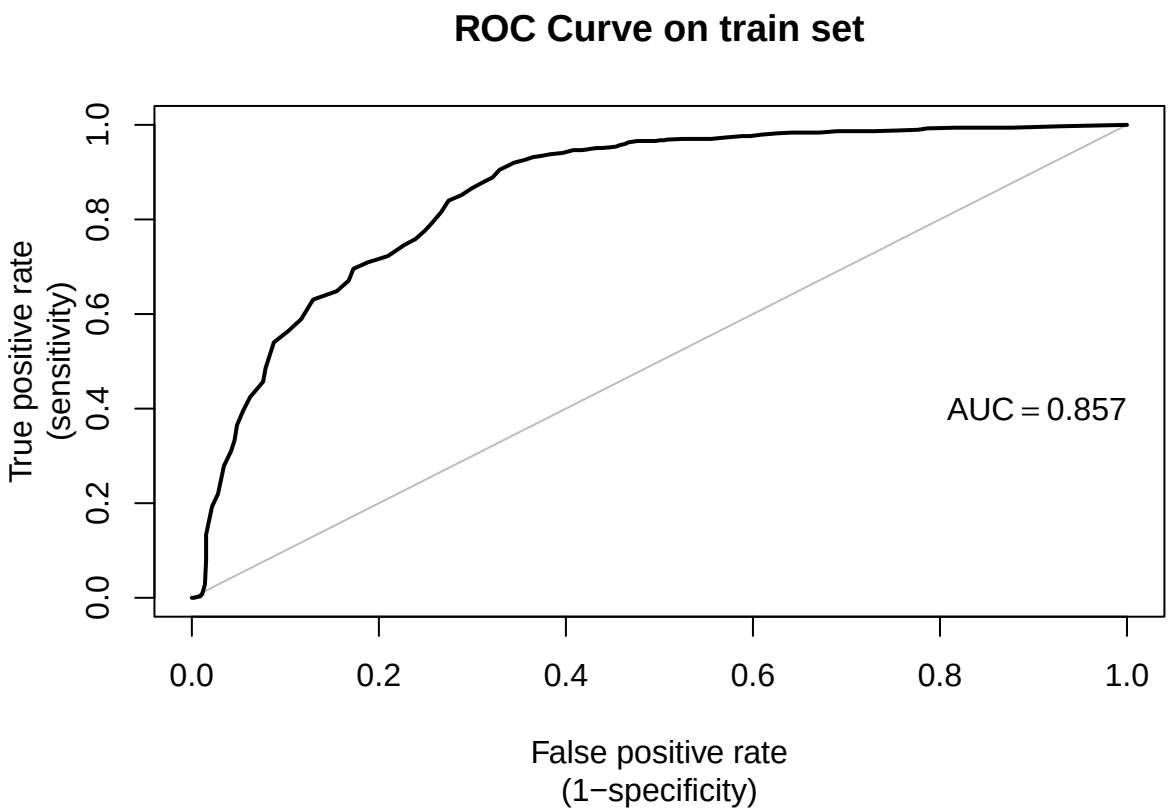
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

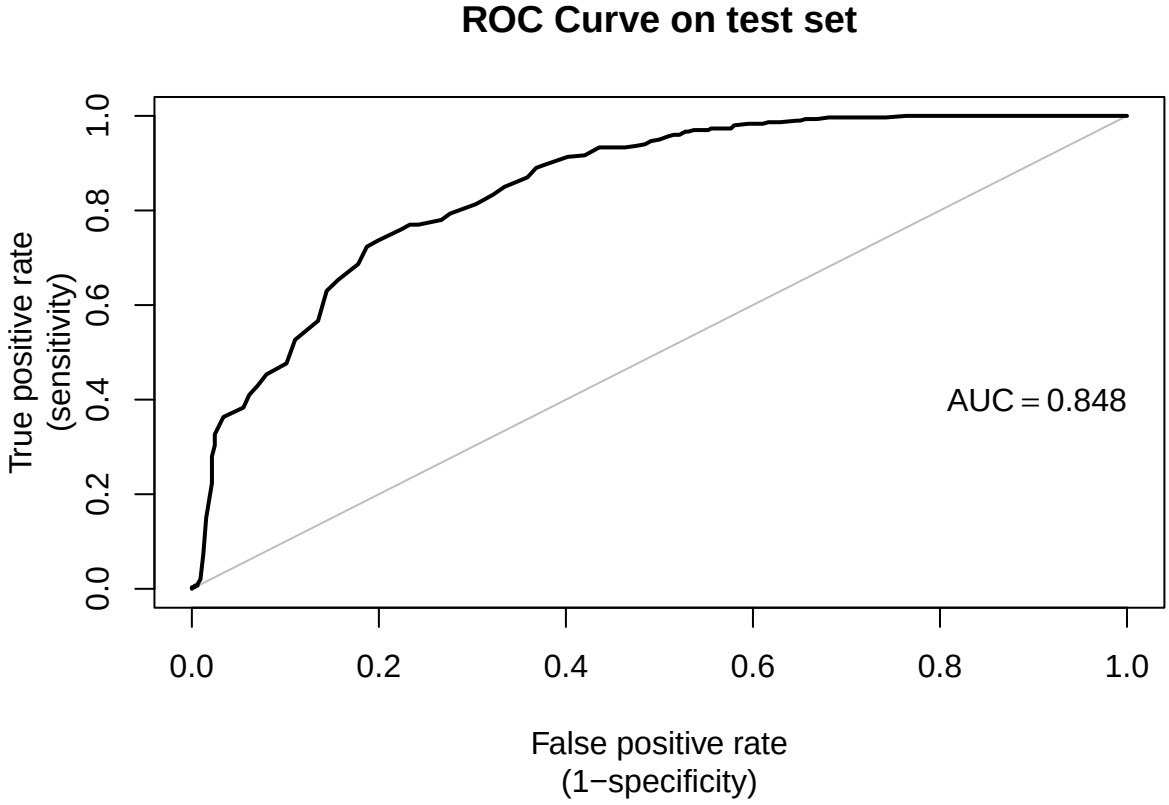
```
##          0 534  75
##          1 253 599
##
##                Accuracy : 0.7755
##                  95% CI : (0.7532, 0.7967)
##     No Information Rate : 0.5387
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5567
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6785
##             Specificity : 0.8887
##          Pos Pred Value : 0.8768
##          Neg Pred Value : 0.7031
##              Prevalence : 0.5387
##          Detection Rate : 0.3655
##    Detection Prevalence : 0.4168
##       Balanced Accuracy : 0.7836
##
##        'Positive' Class : 0
##
```

```
# Confusion matrix on test set
conf_mat_test = confusionMatrix(as.factor(ridge_pred_test), as.factor(y_test))
conf_mat_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 195  26
##          1 131 274
##
##                Accuracy : 0.7492
##                  95% CI : (0.7133, 0.7827)
##     No Information Rate : 0.5208
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5045
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5982
##             Specificity : 0.9133
##          Pos Pred Value : 0.8824
##          Neg Pred Value : 0.6765
##              Prevalence : 0.5208
##          Detection Rate : 0.3115
##    Detection Prevalence : 0.3530
##       Balanced Accuracy : 0.7557
##
##        'Positive' Class : 0
##
```

```
plot_roc(y_train, y_test, as.numeric(ridge_prob_train), as.numeric(ridge_prob_test))
```



## 4.5 Decision tree

To handle with classification problem, we apply a supervised machine learning algorithm called decision tree.

```
# Drop the Weight variable on df
df$Weight <- NULL

# Change the column type of character to factor
df$Gender = as.factor(df$Gender)
df$Family_history = as.factor(df$Family_history)
df$FAVC = as.factor(df$FAVC)
df$CAEC = as.factor(df$CAEC)
df$SMOKE = as.factor(df$SMOKE)
df$SCC = as.factor(df$SCC)
df$CALC = as.factor(df$CALC)
df$MTRANS = as.factor(df$MTRANS)
df$Obesity = as.factor(df$Obesity)
```

```
# Create a train data set
df_train <- df[split_dummy == 0, ]
row.names(df_train) <- NULL # Reorder index
head(df_train)
```

```
##   Gender Age Height Family_history FAVC FCVC NCP CAEC SMOKE CH2O SCC FAF TUE
## 1      0  21   1.62              1    0    2   3    1     0    2   0   0   1
## 2      0  21   1.52              1    0    3   3    1     1    3   1   3   0
## 3      1  23   1.80              1    0    2   3    1     0    2   0   2   1
## 4      1  27   1.80              0    0    3   3    1     0    2   0   2   0
## 5      1  29   1.62              0    1    2   3    1     0    2   0   0   0
## 6      0  23   1.50              1    1    3   3    1     0    2   0   1   0
##   CALC MTRANS Obesity
## 1    0      0       0
## 2    1      0       0
## 3    2      0       0
## 4    2      4       0
## 5    1      1       0
## 6    1      2       0
```

```
# library(dplyr)
x_train <- df_train %>% select(1:15)
y_train <- df_train %>% select(16)
y_train <- y_train[,1] # y_train should be a vector
```

```
# Create a test data set
df_test <- df[split_dummy == 1, ]
row.names(df_test) <- NULL # Reorder index
head(df_test)
```

```
##   Gender Age Height Family_history FAVC FCVC NCP CAEC SMOKE CH2O SCC FAF TUE
## 1      1  22   1.78              0    0    2   1    1     0    2   0   0   0
## 2      1  22   1.64              0    0    2   3    1     0    2   0   3   0
## 3      1  24   1.78              1    1    3   3    1     0    2   0   1   1
## 4      0  52   1.69              1    1    3   1    1     1    2   0   0   0
## 5      0  22   1.60              1    1    1   1    1     0    2   0   0   2
## 6      1  39   1.79              0    0    2   1    1     0    2   0   0   0
##   CALC MTRANS Obesity
## 1    1      0       0
```

```
## 2    1       0       0
## 3    2       0       0
## 4    0       1       1
## 5    1       0       1
## 6    1       0       0
```

```
x_test <- df_test %>% select(1:15)
y_test <- df_test %>% select(16)
y_test <- y_test[,1] # y_test should be a vector
```

In R language, we can use rpart library to perform a decision tree.

```
model_dt = rpart(Obesity ~ ., data=df_train, method='class')
fancyRpartPlot(model_dt, main='Classification Tree')
```

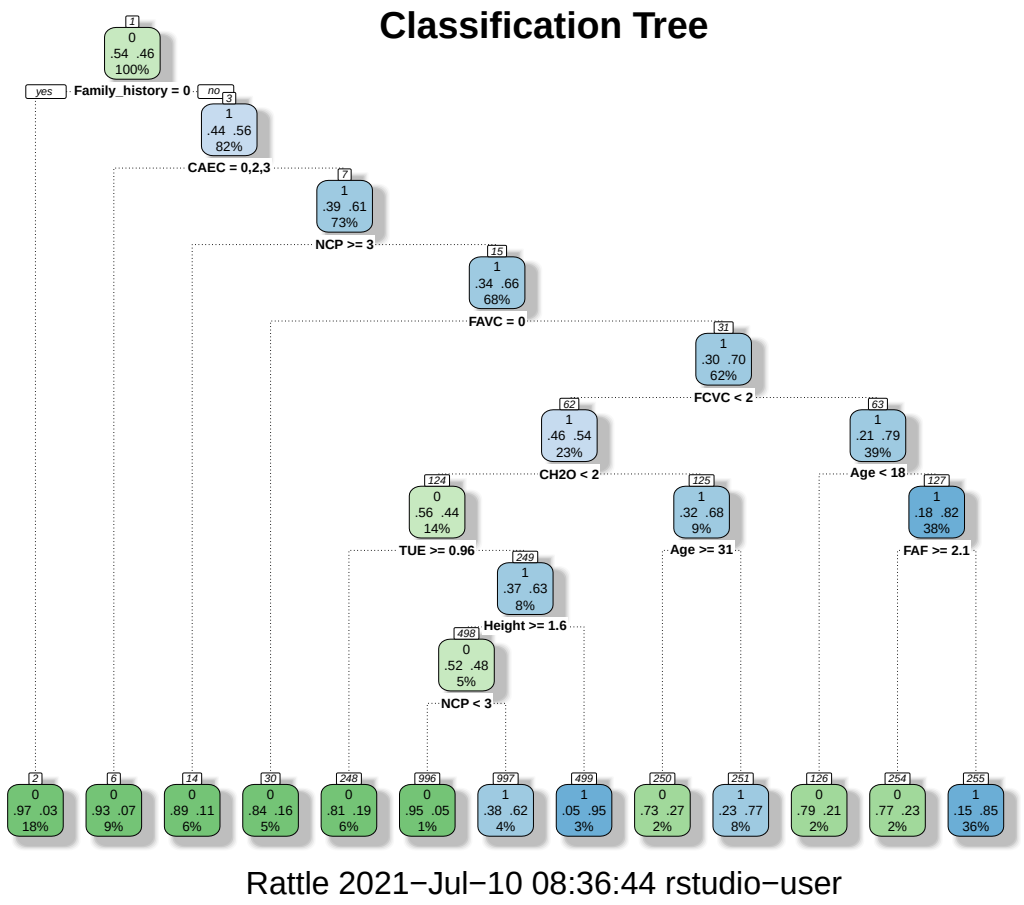



Figure 26:

At the top is the root node, it shows that 54% of observations are non-obesity while 46% are obesity and the number below indicates the proportion of the population that resides in this node, here at the top level so it is 100%. Travelling down the tree branches, if the family history with overweight does not happen, 97% of observation are non-obesity, and if it happens, 56% of observation are obesity and so on until we reach the leaf nodes.

In decision tree, nodes are arranged in order of priority, the top nodes have higher priority than the ones below. Based on figure 26, we can see that the first 5 nodes are Family_history, CAEC, NCP, FAVC and FCVC. Once again, as we saw on the best subsets method, the most important predictors of obesity are often related to dietary problems.

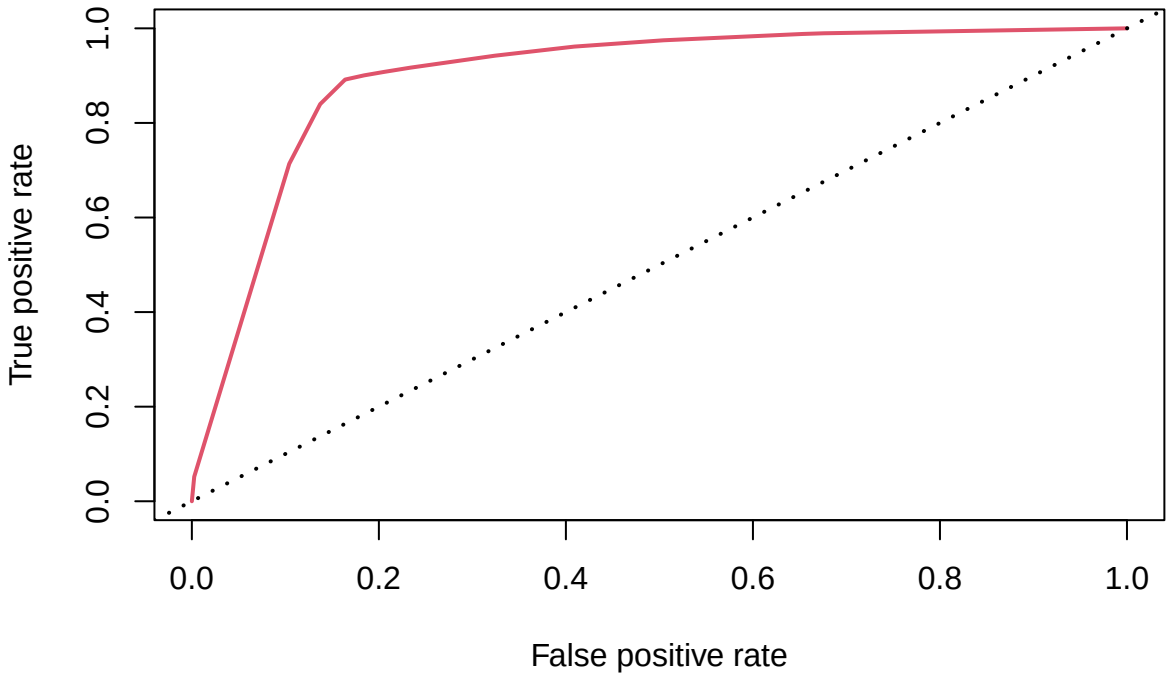
```r
# Evaluate the model on training set
pred_dt_train <- predict(model_dt, df_train, type='prob')
pred_train <- ifelse(pred_dt_train[,2] > optimal_threshold, 1, 0)
confusionMatrix(as.factor(y_train), as.factor(pred_train))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 658 129
##          1  73 601
##
##                Accuracy : 0.8617
##                  95% CI : (0.843, 0.879)
##     No Information Rate : 0.5003
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7235
##
##  Mcnemar's Test P-Value : 0.0001089
##
##             Sensitivity : 0.9001
##             Specificity : 0.8233
##          Pos Pred Value : 0.8361
##          Neg Pred Value : 0.8917
##              Prevalence : 0.5003
##          Detection Rate : 0.4504
##    Detection Prevalence : 0.5387
##       Balanced Accuracy : 0.8617
##
##        'Positive' Class : 0
##
```

```r
pred <- prediction(pred_dt_train[,2], y_train)
perf <- performance(pred, 'tpr', 'fpr')
auc <- performance(pred, measure='auc')
cat('AUC: ', round(auc@y.values[[1]], 4))
```

```
## AUC:  0.899
```

```r
plot(perf, main='ROC curve for training set', col=2, lwd=2)
abline(a=0, b=1, lwd = 2,lty = 3,col = "black")
```




```r
# Evaluate the model on test set
pred_dt_test <- predict(model_dt, df_test, type='prob')
pred_test <- ifelse(pred_dt_test[,2] > optimal_threshold, 1, 0)
confusionMatrix(as.factor(y_test), as.factor(pred_test))
```

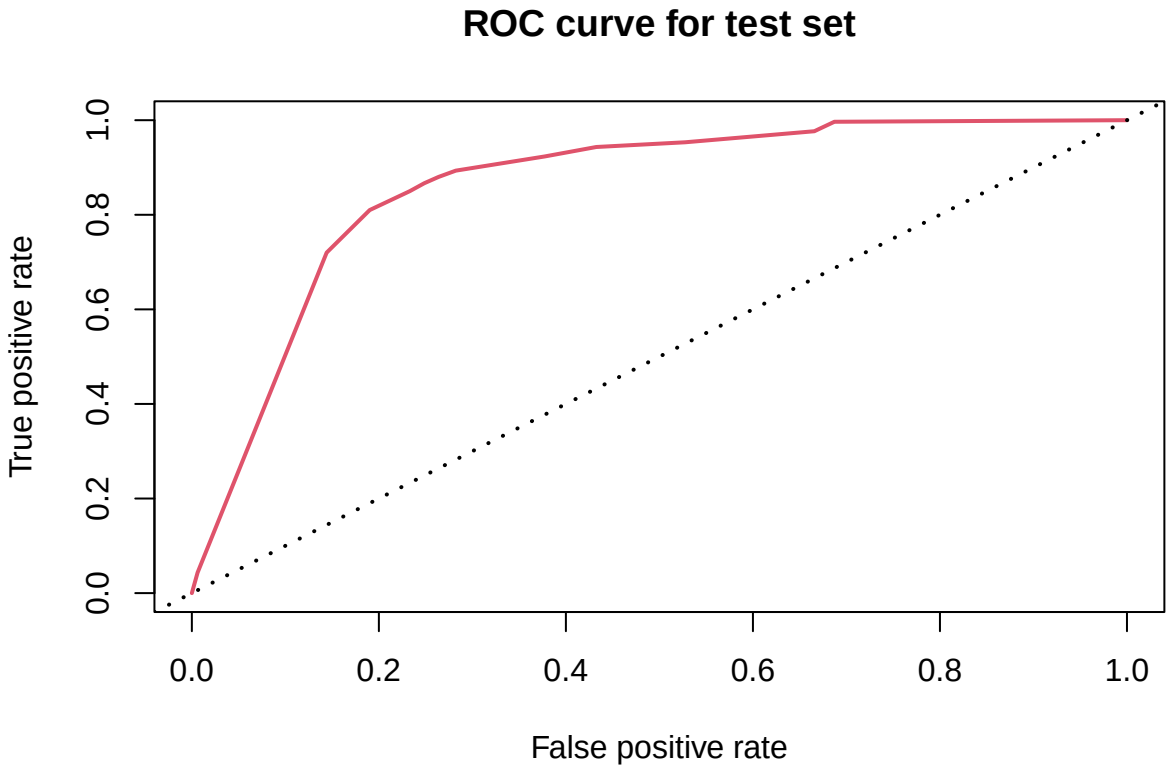
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 250  76
##          1  45 255
##
##                Accuracy : 0.8067
##                  95% CI : (0.7736, 0.8369)
##     No Information Rate : 0.5288
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6143
##
##  Mcnemar's Test P-Value : 0.006386
##
##             Sensitivity : 0.8475
##             Specificity : 0.7704
##          Pos Pred Value : 0.7669
##          Neg Pred Value : 0.8500
##              Prevalence : 0.4712
##          Detection Rate : 0.3994
##    Detection Prevalence : 0.5208
##       Balanced Accuracy : 0.8089
##
```

```
##        'Positive' Class : 0
##
```

```
pred <- prediction(pred_dt_test[,2], y_test)
perf <- performance(pred, 'tpr', 'fpr')
auc <- performance(pred, measure='auc')
cat('AUC: ', round(auc@y.values[[1]], 4))
```

```
## AUC:  0.8614
```

```
plot(perf, main='ROC curve for test set', col=2, lwd=2)
abline(a=0, b=1, lwd = 2,lty = 3,col = "black")
```




### 4.6 Result

| | Train | Train | Test | Test |
|---|---|---|---|---|
| | Accuracy | AUC | Accuracy | AUC |
| Logistic Regression | 0.791 | 0.86 | 0.746 | 0.848 |
| Variable Selection | | | | |
| AIC | 0.786 | 0.858 | 0.741 | 0.849 |
| Best Subsets | 0.783 | 0.858 | 0.758 | 0.85 |
| Cross-validation | 0.783 | 0.858 | 0.748 | 0.849 |
| Regularization | | | | |
| Lasso Regression | 0.786 | 0.858 | 0.743 | 0.849 |
| Ridge Regression | 0.775 | 0.857 | 0.749 | 0.848 |
| Decision Tree | 0.862 | 0.899 | 0.807 | 0.861 |

The table above shows the accuracy and AUC value based on 7 methods applied on the Obesity data. We can see that most of methods have the same performance with the accuracy and AUC value approximately

79% and 85% respectively. However, with supervised machine learning method Decision Tree, the result is outperform than other methods based on the accuracy, with the accuracy 86% compared with 79% of the logistic regression, it shows that with classification problem, decision tree has a better performance than traditional methods.

```
cat('Confusion matrix of Logistic Regression on test set:')
```

```
## Confusion matrix of Logistic Regression on test set:
```

```
table(as.factor(y_test), as.factor(test_pred_lr))
```

```
##
##       0   1
##   0 194 132
##   1  26 274
```

```
cat('Confusion matrix of Decision tree on test set:')
```

```
## Confusion matrix of Decision tree on test set:
```

```
table(as.factor(y_test), as.factor(pred_test))
```

```
##
##       0   1
##   0 250  76
##   1  45 255
```

We can see that, with decision tree method, the number of correctly predicted non-obesity observations are higher than logistic regression but the number of correctly predicted obesity observations are lower than logistic regression. However, in this problem, we want to focus on obesity observations, therefore, if an obese person is predicted to be non-obese is more dangerous than if a non-obese person is predicted to be obese. In this case, decision tree does a better job with 76 incorrectly predicted compared with 132 cases in logistic regression.

# 5 Conclusion

Obesity is becoming an alarming health problem, affecting an appalling percentage of the population and causing the death of many individuals. The impact of eating habits and physical activities can be shown after doing EDA and modelling using some basic methods. In conclusion, the most important predictor is Family_history, which directly affect an observation is obese or not. Moreover, certain eating habits play a key role in weight control, such as FAVC (the frequency consumption of high caloric food), CAEC (the consumption of food between meals), FCVC (the frequency consumption of vegetables). Besides, some physical conditions play an important part like MTRANS (Transportation used) and FAF (Physical activity frequency). Some other factors are not impactful across the board were Smoke, CH2O (The consumption of water daily) and CALC (The consumption of alcohol). There is a factor NCP (Number of main meals) which shows less important in Logistic Regression model but it plays a key role in Decision Tree model that can be explained by the fact that not all the independent variables are linearly related, some of which are not linearly related to each other. For the future work, we want to apply another method called Generalized Additive Model (GAM) which is allowed to learn non-linear features in order to evaluate which variable has nonlinear relationships with the Obesity type.

# 6 Bibliography

[1] WHO, 2021. World Health Organization. Obesity and overweight.

[2] Hannah Ritchie and Max Roser (2017) - "Obesity". Published online at OurWorldInData.org, 2017. 'https://ourworldindata.org/obesity'

[3] Fabio Mendoza Palechor, Alexis de la Hoz Manotas. Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico. Data in Brief, Volume 25, 2019, 104344, ISSN 2352-3409, https://doi.org/10.1016/j.dib.2019.104344.

[4] DO, NORMA Oficial Mexicana NOM-008-SSA3-2010, Para el tratamiento integral del sobrepeso y la obesidad, Diario Oficial, 2010.

[5] Dataset directory https://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+#