Water Potability in India

Statistical Learning (MOD. B) Exam

Gioele Ceccon 2079425, Pietro Renna 2089068

2023-05-30

Introduction

Water is a vital resource for human life, and safeguarding the quality of water bodies is crucial for preserving public health and the environment. The unregulated growth of industries and human activities over the past century has resulted in the widespread contamination and pollution of water bodies, with developing countries, such as India, facing particularly severe challenges due to limited resources for monitoring and regulating water quality. This project aims to tackle this issue by developing a binary classification model of the quality of freshwater bodies, such as lakes, rivers, and dams in India, which are classified as either "Potable" or "Non-Potable" based on various quality parameters provided in the discussed dataset. The primary objectives of this analysis are to build a model capable of determining the safety of water samples for human consumption, identify the factors affecting water quality, and evaluate the relative importance of each factor in determining potability. Overall, the classification analysis of water bodies is a critical step towards ensuring public health and environmental sustainability. Moreover, it has far-reaching implications for developing effective policies and regulations to protect water resources.

Dataset Presentation

The proposed dataset[1] is taken from *HydroShare*[2], an online, open-source, collaborative system developed for sharing hydrologic data and models, operated by *The Consortium of Universities for the Advancement of Hydrologic Science Inc.* (CUAHSI). It consists of 1361 observations that are characterized by 15 attributes. Each observation represents a water sample collected from a specific location in one of the 32 Indian states, with each attribute representing physical, chemical, and biological characteristics, as well as the presence of pollutants and organic compounds. The attribute that represents whether the water sample is suitable for human consumption is presented as a binary value and it's initially named "class." However, to make it a more informative name, it has been changed to "Potability."

The importation of the necessary libraries is carried out.

```
# Retrieval of the libraries necessary for the correct import of
# the data, its manipulation and its visualization.

library(dplyr)
library(pander)
library(tibble)
```

[1] M, G. J. (2023). Water Quality Dataset for water quality prediction tool, HydroShare, https://doi.org/10.4211/hs.4ab43e1b507b496b9b42749701daed5c

[2] Consortium of Universities for the Advancement of Hydrologic Science, Inc. (2023). HydroShare [Data sharing platform] https://help.hydroshare.org/about-hydroshare/

```
library(dlookr)
library(multiUS)
library(ggplot2)
library(reshape2)
library(knitr)
library(MASS)
library(car)
library(class)
library(rsample)
library(caret)
library(ggpubr)
library(pROC)
library(glmnet)
library(knitr)
library(kableExtra)
library(readxl)
```

The dataset is presented as an XLSX file and it's imported through the function *read_excel*.

```
# Upload the dataset "aquaattributes.xlsx".
data <- read_excel("aquaattributes.xlsx")
```

Let's take a look at its first 10 instances:

```
## # A tibble: 10 x 15
##    Stationcode Locations     Lat   Lon Capitalcity State Temperature   D.O    pH
##          <dbl> <chr>       <dbl> <dbl> <chr>       <chr>       <dbl> <dbl> <dbl>
##  1        1001 BEAS AT U/~  32.2  77.2 Shimla      HIMA~           9     9     8
##  2        1002 BEAS AT D/~  32.0  77.1 Shimla      HIMA~          10     9     8
##  3        1003 BEAS AT D/~  26.9  75.8 Shimla      HIMA~          11     9     8
##  4        1004 BEAS AT U/~  47.4  19.6 Shimla      HIMA~          13     9     8
##  5        1005 BEAS AT EX~  26.0  91.8 Shimla      HIMA~          14    10     8
##  6        1550 U/S MANDI    31.7  76.9 Shimla      HIMA~          16     9     8
##  7        1006 BEAS AT D/~  31.7  76.9 Shimla      HIMA~          16     9     8
##  8        2604 BEAS AT D/~  31.9  76.6 Shimla      HIMA~          19     8     8
##  9        1007 BEAS AT D/~  15.9  78.1 Shimla      HIMA~          19     8     8
## 10        1008 BEAS AT D/~  31.9  76.2 Shimla      HIMA~          19     8     8
## # i 6 more variables: Conductivity <chr>, B.O.D <dbl>, Nitrate <chr>,
## #   Fecalcaliform <chr>, Totalcaliform <dbl>, class <chr>
```

To solve some issues (spelling, presence of points) with the names of the variables, some changes are made:

```
# Removal of points
colnames(data)[8] <- "DO"
colnames(data)[11] <- "BOD"

# Replace the "class" name with a more informative one
colnames(data)[15] <- "Potability"

# Spell correction
colnames(data)[13] <- "Fecalcoliform"
colnames(data)[14] <- "Totalcoliform"
```

To gain a more comprehensive understanding of the impact of each of the 15 variables, they were examined in detail. The subsequent section presents a brief explanation for each variable.

Variable	Description
Stationcode	ID of the water sample
Locations	Location where the water sample was collected
Lat	Latitude of the location of the water sample
Lon	Longitude of the location of the water sample
Capitalcity	Capital city of the state in which the water sample was collected
State	Indian state in which the water sample was collected
Temperature	Temperature in °C of the water sample
DO	Dissolved Oxygen, it's a measure of how much oxygen is dissolved in the water. It's measured in milligrams of dissolved oxygen per liter of water (mg/L).
ph	It's a measure of how acidic/basic water is. The range goes from 0 - 14, with 7 being neutral. pHs of less than 7 indicate acidity, whereas a pH of greater than 7 indicates a base.
Conductivity	It's a measure of the ability of water to pass an electrical current. It's measured in microSiemens/cm (µS/cm).
BOD	Biological Oxygen Demand, it represents the amount of oxygen consumed by bacteria and other microorganisms while they decompose organic matter under aerobic (oxygen is present) conditions at a specified temperature. It's measured in milligrams of dissolved oxygen per liter (mg/L).
Nitrate	It's a chemical found in fertilizers, manure, agricultural runoff, dairy lagoons, and liquid waste discharged from septic tanks. It's measured in mg/L.
Fecalcoliform	They are microscopic organisms that live in the intestines of warm-blooded animals. The presence of fecal coliform in a drinking water sample often indicates recent fecal contamination. Its presence is measured in number of colony-forming units (CFUs) per 100 mL of sample water (#/100 mL).
Totalcoliform	It is a group of bacteria that are commonly found in the environment, including soil, water, and vegetation. They are not typically harmful to human health, but their presence in water can indicate the potential presence of other harmful bacteria, viruses, or parasites that can cause illness. Its presence is measured in number of colony-forming units (CFUs) per 100 mL of sample water (#/100 mL).
Potability	Response variable, it represents the water's quality (either Potable or Non-Potable).

The variables concerning the location of the observations and the IDs are removed since they can't be properly used on the kind of models that will be presented later on the analysis.

```
# Removing unnecessary data
working_data<-data[,-c(1,2,3,4,5,6)]
```

The response variable "Potability" is encoded as character, and so in order to treat it as a proper categorical variable, it is converted to the R factor type

```
# Transforming the response variable into a factor
working_data$Potability <- ifelse(working_data$Potability == "yes", 1, 0)
working_data$Potability <- factor(working_data$Potability)
```

For the same reason, some numeric variables that are initially encoded as char type are converted to numeric type.

```
# Transforming char variables into numerical variables
working_data$Conductivity <- as.numeric(working_data$Conductivity)
working_data$Nitrate <- as.numeric(working_data$Nitrate)
working_data$Fecalcoliform <- as.numeric(working_data$Fecalcoliform)
```

Let's take a quick overview at the dataset:

```
pandoc.title("Summary of Predictors")
```

```
## %
## Summary of Predictors
```

```
pander(summary(working_data), style = "rmarkdown")
```

Table 2: Table continues below

Temperature	DO	pH	Conductivity
Min. : 5.00	Min. : 0.000	Min. :5.000	Min. : 2.0
1st Qu.:23.50	1st Qu.: 6.329	1st Qu.:7.375	1st Qu.: 216.7
Median :26.21	Median : 7.200	Median :7.700	Median : 417.2
Mean :25.16	Mean : 7.054	Mean :7.605	Mean : 1232.1
3rd Qu.:27.76	3rd Qu.: 7.800	3rd Qu.:7.984	3rd Qu.: 781.6
Max. :65.00	Max. :30.367	Max. :9.575	Max. :65700.0
NA's :34	NA's :9	NA's :1	NA's :38

Table 3: Table continues below

BOD	Nitrate	Fecalcoliform	Totalcoliform
Min. : 0.000	Min. : 0.0000	Min. : 1	Min. : 0
1st Qu.: 1.000	1st Qu.: 0.3336	1st Qu.: 14	1st Qu.: 2
Median : 2.060	Median : 0.9825	Median : 120	Median : 251
Mean : 4.191	Mean : 9.1467	Mean : 429155	Mean : 1431256
3rd Qu.: 3.933	3rd Qu.: 2.5428	3rd Qu.: 774	3rd Qu.: 1600
Max. :222.000	Max. :920.0000	Max. :230000000	Max. :670000000
NA's :68	NA's :210	NA's :189	NA's :135

Potability
0: 298
1:1063
NA

Potability
NA
NA
NA
NA

Looking at the summary above, it's easy to see that there are many NA values and there is an evident presence of outliers. These problems are assessed further in the analysis.

Dataset Split

To evaluate the performance of the models presented in this analysis, the initial dataset is divided into two sets. The first set, called the training set, is utilized for model training and estimation, whereas the second set, the test set, solely comprises examples that are absent from the training set. The test set is used to assess the models' performance and effectiveness. The split is performed by randomly assigning 80% of the examples to the training set and the remaining 20% to the test set, which is a commonly employed choice.

```
# Train and Test split
set.seed(1429)
split <- initial_split(working_data, prop=0.8)
train <- training(split)
test <- testing(split)
```

After the split is done, it's worth checking the proportions for the response variable in both sets to see if the split has been properly performed.

```
train_prop<-round(prop.table(table(train$Potability)),3)
test_prop<-round(prop.table(table(test$Potability)),3)
```

Training Set

```
row.names(train_prop) <- c( "Non-Potable","Potable")
pandoc.title("Proportion of Potability in Train set")
```

```
## %
## Proportion of Potability in Train set
```

```
pander((train_prop), style = "rmarkdown")
```

Non-Potable	Potable
0.216	0.784

Test Set

```
row.names(test_prop) <- c("Non-Potable","Potable")
pandoc.title("Proportion of Potability in Test set")
```

```
## %
## Proportion of Potability in Test set
```

```
pander((test_prop), style = "rmarkdown")
```

Non-Potable	Potable
0.231	0.769

Pre-Processing

Pre-Processing represents a set of techniques and data manipulation to ensure a proper format with respect to the further analysis that will be performed.

Test data represent new unseen observations that the model does not know. Since they are used to assess the performance of the model, they must be representative of real examples which may also contain extreme or incorrect values. For this reason, outliers are not removed in the Test set. On the other hand, NA values are treated in the same way in both sets, since their removal would cause a big loss of information compared to the number of available observations.

Handling Outliers

Due to the fact that variables have not exact symmetric distributions, outliers are handled with the Modified Z-Score (Iglewicz and Hoaglin), which is defined as follows:

Modified Z-Score $= \frac{0.6745 \times (x - \text{median}(x))}{\text{MAD}}$

where:

- 0.6745 is a fixed constant;
- $median(x)$ is the median for the considered variable x;
- Median Absolute Deviation MAD $=$ median $(|x_i - median(x)|)$

```
iglewicz_hoaglin = function(x, threshold = 3.5, return_scores = F) {
  # check input
  if (!is.numeric(x))
    stop("could not identify outliers: input is not numeric")
  # calculate modified Z scores
  med = median(x, na.rm = T)
  MAD = median(abs(x - med), na.rm = T)
  Mi = 0.6745 * (x - med) / MAD
  if (return_scores) {
    return(Mi)
  } else {
    # mask vector
    x[abs(Mi) > threshold] = NA
    return(x)
  }
}
```

Once the Modified Z-Score is defined, it can be applied to remove outliers on all the variables of the training set.

```
train$Temperature <- iglewicz_hoaglin(train$Temperature)
train$DO <- iglewicz_hoaglin(train$DO)
train$pH <- iglewicz_hoaglin(train$pH)
train$BOD <- iglewicz_hoaglin(train$BOD)
train$Nitrate <- iglewicz_hoaglin(train$Nitrate)
train$Conductivity <- iglewicz_hoaglin(train$Conductivity)
train$Fecalcoliform <- iglewicz_hoaglin(train$Fecalcoliform)
train$Totalcoliform<- iglewicz_hoaglin(train$Totalcoliform)
```

Handling NA's

Following the Modified Z-Score application, numerous NA values emerged. Additionally, both the train and test sets already contained some NA values which were present in the original dataset. In order to retain all the examples and prevent the loss of many observations, it is necessary to handle these NA values. A resilient approach to accomplish this is to employ median imputing, which is a robust measure of centrality. This technique involves substituting the NA values with the median value of the corresponding variable.

```
train$Temperature[is.na(train$Temperature)]<- median(train$Temperature,
                                                    na.rm = TRUE)
train$DO[is.na(train$DO)]<- median(train$DO,
                                   na.rm = TRUE)
train$pH[is.na(train$pH)]<- median(train$pH,
                                   na.rm = TRUE)
train$Conductivity[is.na(train$Conductivity)]<- median(train$Conductivity,
                                                       na.rm = TRUE)
train$BOD[is.na(train$BOD)]<- median(train$BOD,
                                     na.rm = TRUE)
train$Nitrate[is.na(train$Nitrate)]<- median(train$Nitrate,
                                             na.rm = TRUE)
train$Fecalcoliform[is.na(train$Fecalcoliform)]<- median(train$Fecalcoliform,
                                                         na.rm = TRUE)
train$Totalcoliform[is.na(train$Totalcoliform)]<- median(train$Totalcoliform,
                                                         na.rm = TRUE)
train$Temperature[is.na(train$Temperature)]<- median(train$Temperature,
                                                    na.rm = TRUE)
train$DO[is.na(train$DO)]<- median(train$DO,
                                   na.rm = TRUE)
train$pH[is.na(train$pH)]<- median(train$pH,
                                   na.rm = TRUE)
train$Conductivity[is.na(train$Conductivity)]<- median(train$Conductivity,
                                                       na.rm = TRUE)
train$BOD[is.na(train$BOD)]<- median(train$BOD,
                                     na.rm = TRUE)
train$Nitrate[is.na(train$Nitrate)]<- median(train$Nitrate,
                                             na.rm = TRUE)
train$Fecalcoliform[is.na(train$Fecalcoliform)]<- median(train$Fecalcoliform,
                                                         na.rm = TRUE)
train$Totalcoliform[is.na(train$Totalcoliform)]<- median(train$Totalcoliform,
                                                         na.rm = TRUE)
```

Test

```r
#test
test$Temperature[is.na(test$Temperature)]<- median(test$Temperature,
                                                   na.rm = TRUE)
test$DO[is.na(test$DO)]<- median(test$DO,
                                 na.rm = TRUE)
test$pH[is.na(test$pH)]<- median(test$pH,
                                 na.rm = TRUE)
test$Conductivity[is.na(test$Conductivity)]<- median(test$Conductivity,
                                                     na.rm = TRUE)
test$BOD[is.na(test$BOD)]<- median(test$BOD,
                                   na.rm = TRUE)
test$Nitrate[is.na(test$Nitrate)]<- median(test$Nitrate,
                                           na.rm = TRUE)
test$Fecalcoliform[is.na(test$Fecalcoliform)]<- median(test$Fecalcoliform,
                                                       na.rm = TRUE)
test$Totalcoliform[is.na(test$Totalcoliform)]<- median(test$Totalcoliform,
                                                       na.rm = TRUE)
test$Temperature[is.na(test$Temperature)]<- median(test$Temperature,
                                                   na.rm = TRUE)
test$DO[is.na(test$DO)]<- median(test$DO,
                                 na.rm = TRUE)
test$pH[is.na(test$pH)]<- median(test$pH,
                                 na.rm = TRUE)
test$Conductivity[is.na(test$Conductivity)]<- median(test$Conductivity,
                                                     na.rm = TRUE)
test$BOD[is.na(test$BOD)]<- median(test$BOD,
                                   na.rm = TRUE)
test$Nitrate[is.na(test$Nitrate)]<- median(test$Nitrate,
                                           na.rm = TRUE)
test$Fecalcoliform[is.na(test$Fecalcoliform)]<- median(test$Fecalcoliform,
                                                       na.rm = TRUE)
test$Totalcoliform[is.na(test$Totalcoliform)]<- median(test$Totalcoliform,
                                                       na.rm = TRUE)
```

Exploratory Analysis

Exploratory analysis is a technique employed to examine fundamental informations about variables, such as their distributions, behavior and descriptive statistics. This is accomplished using tools such as graphical representations and tables.

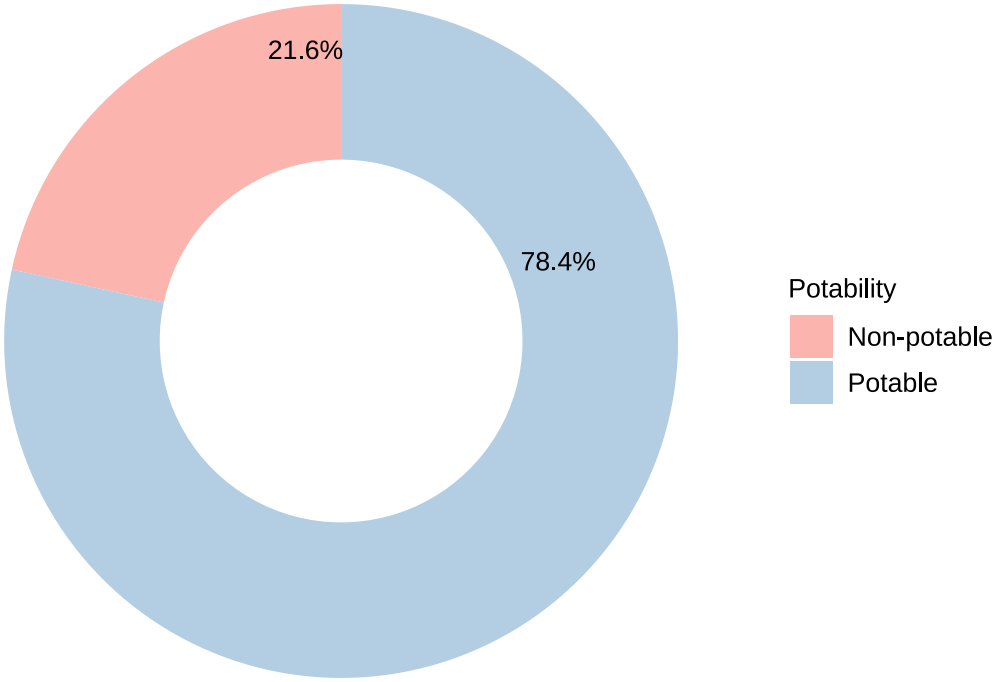
Univariate Analysis

The Univariate Analysis is the Exploratory Analysis performed on a single variable at time.

Potability (Response Variable)

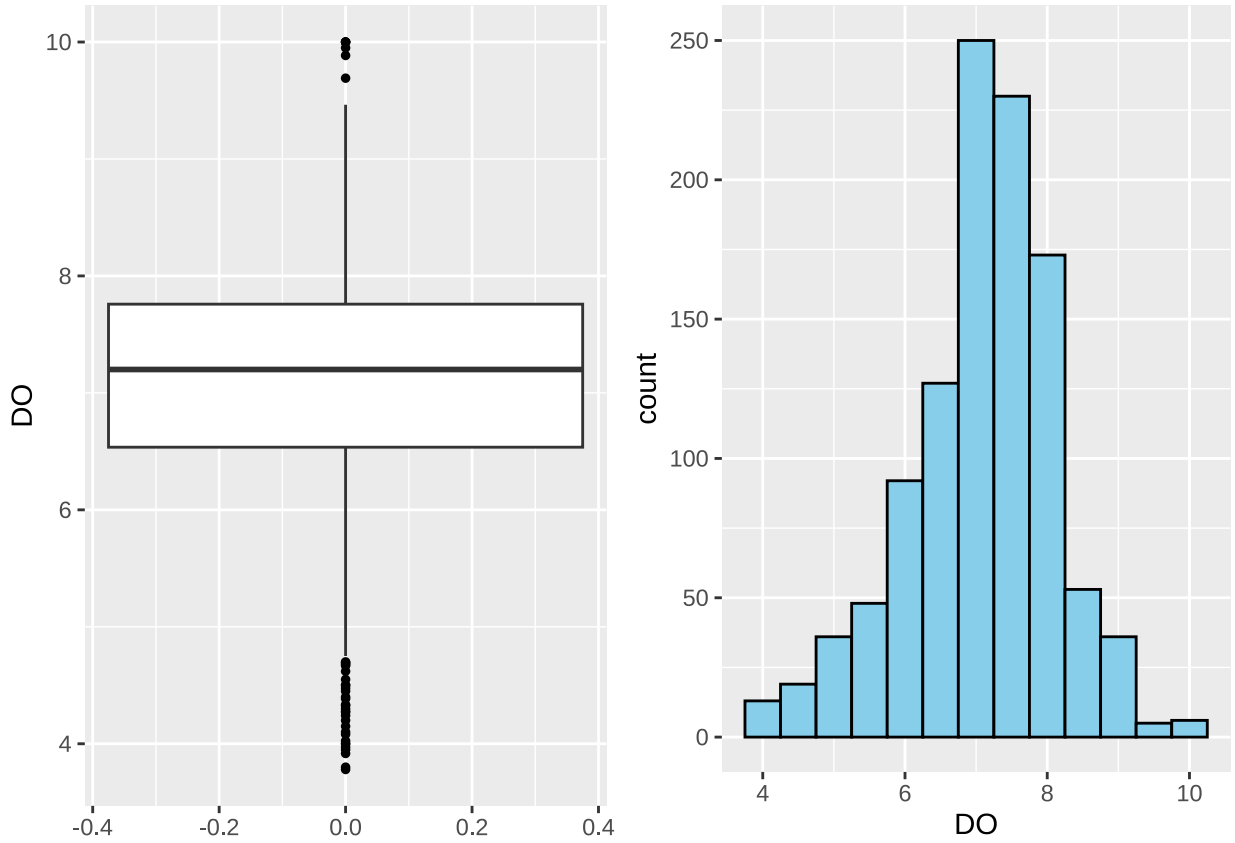
It's immediate to see that the two classes are imbalanced, having more Potable observations than Non-Potable ones.

```
#(Donutplot)
results = train %>%
  group_by(Potability) %>%  # specify categorical variable
  summarize(Frequency = n()) %>% # return counts / frequencies
  mutate(Percent = paste0(round(Frequency / dim(train)[1] * 100, 2), "%"))

ggplot(results, aes(x = 2, y = Frequency, fill = Potability)) +
geom_bar(stat = "identity") +coord_polar(theta = "y",start=0) + theme_void() +
  geom_text(aes(y = (Frequency/sum(Frequency)) * 2 * pi +
                  c(0, cumsum(Frequency)[-length(Frequency)]),
                label = paste0(round(Frequency / dim(train)[1] * 100, 2), "%")),
            color = "black", size = 3.5, hjust =1.0, vjust=-1.25) +
  theme(legend.title = element_text(size = 10),legend.text = element_text(size = 10)) +
    scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability") + xlim(0.5, 2.5)
```


21.6%
78.4%
Potability
Non-potable
Potable

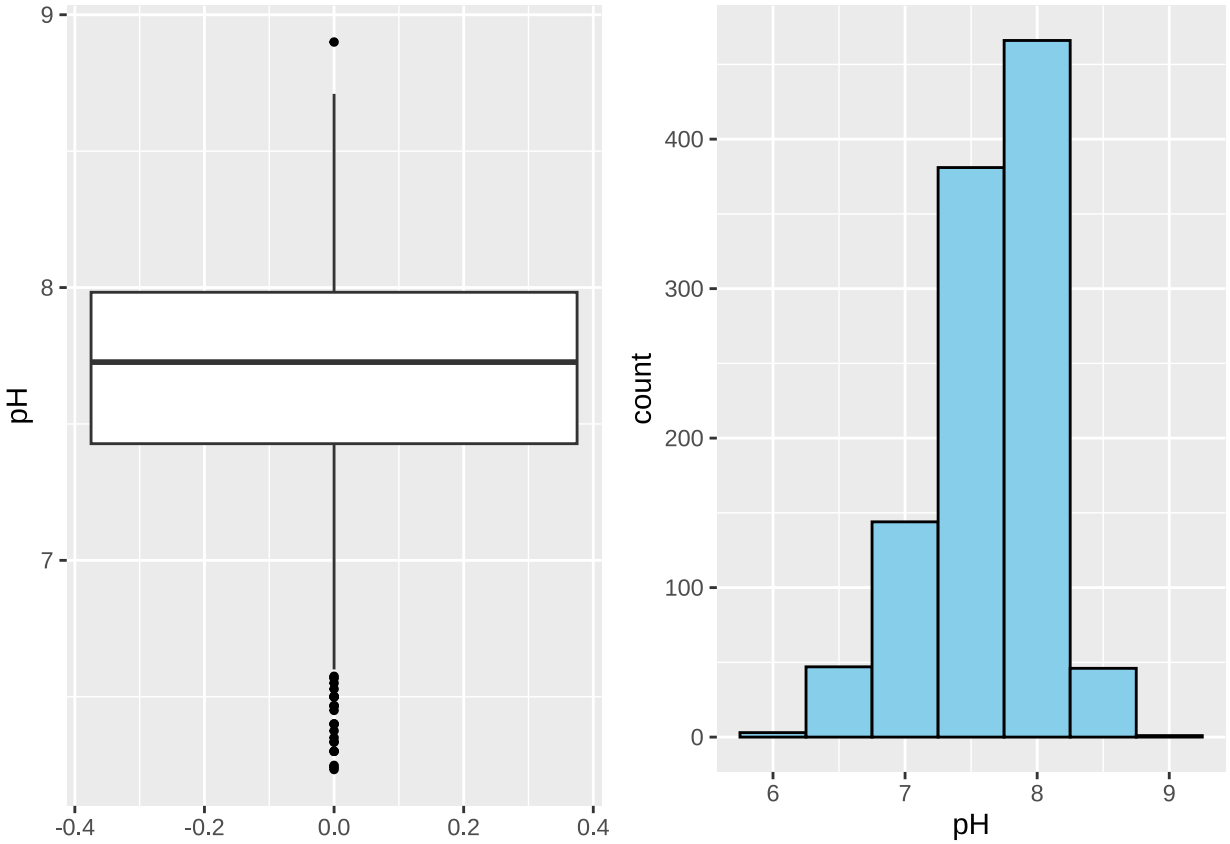
DO

```
box_plot <- ggplot(data=train, aes(x=DO)) +
  geom_boxplot(outlier.colour="black",
               outlier.size=1) + coord_flip()+
  labs(x = "DO")

hist_plot <- ggplot(train, aes(x = DO)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black")

ggarrange(box_plot, hist_plot, ncol = 2)
```



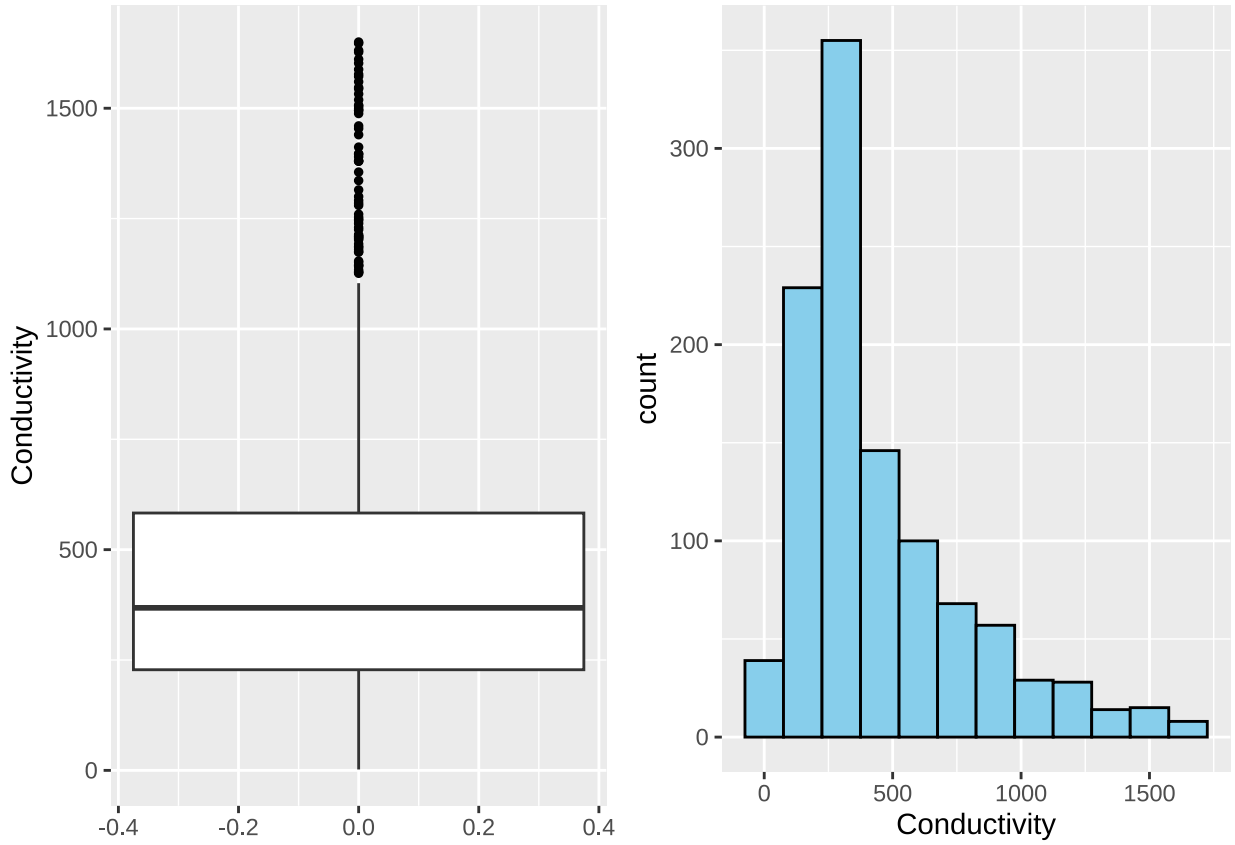
pH

```
box_plot <- ggplot(data=train, aes(x=pH)) +
  geom_boxplot(outlier.colour="black",
               outlier.size=1) + coord_flip()+
  labs(x = "pH")

hist_plot <- ggplot(train, aes(x = pH)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black")

ggarrange(box_plot, hist_plot, ncol = 2)
```



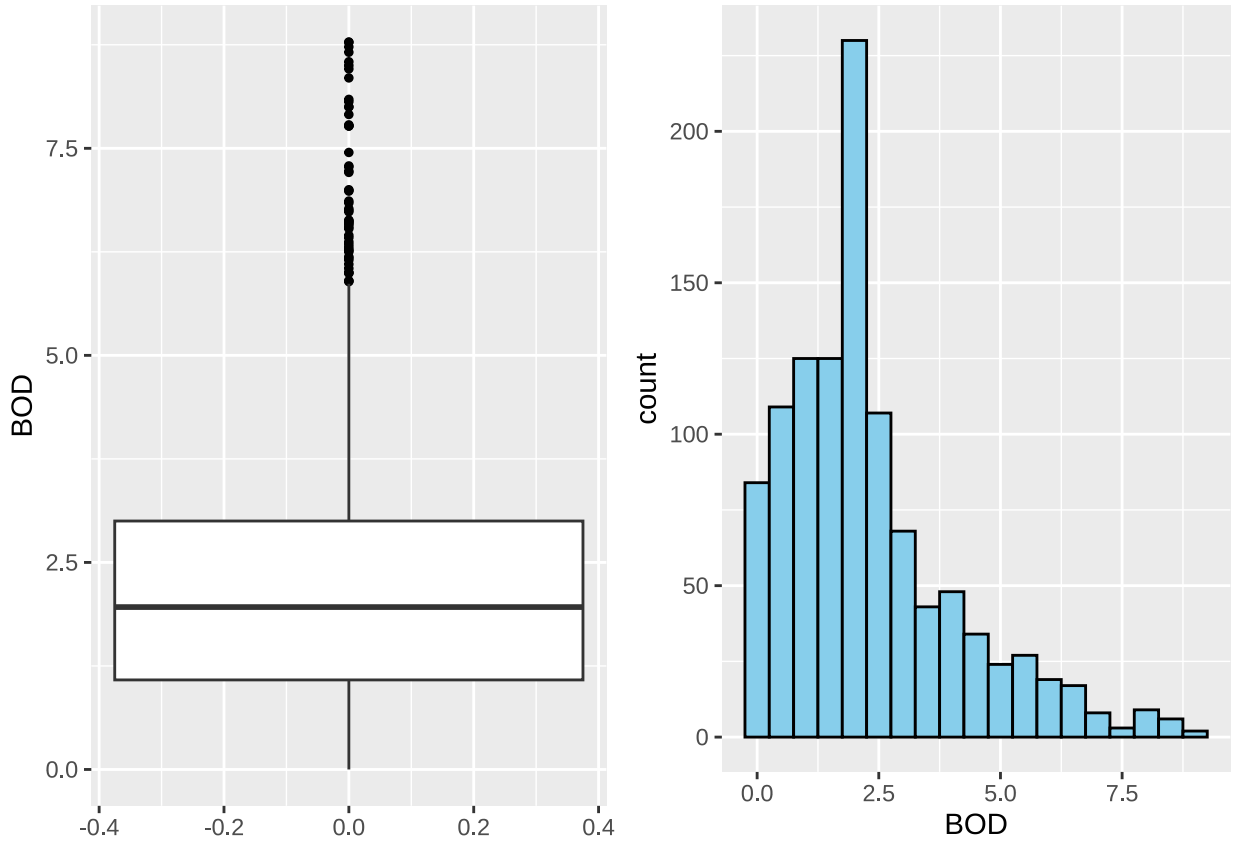
Conductivity

```
box_plot <- ggplot(data=train, aes(x=Conductivity)) +
  geom_boxplot(outlier.colour="black",
               outlier.size=1) + coord_flip()+
  labs(x = "Conductivity")

hist_plot <- ggplot(train, aes(x = Conductivity)) +
  geom_histogram(binwidth = 150, fill = "skyblue", color = "black")

ggarrange(box_plot, hist_plot, ncol = 2)
```



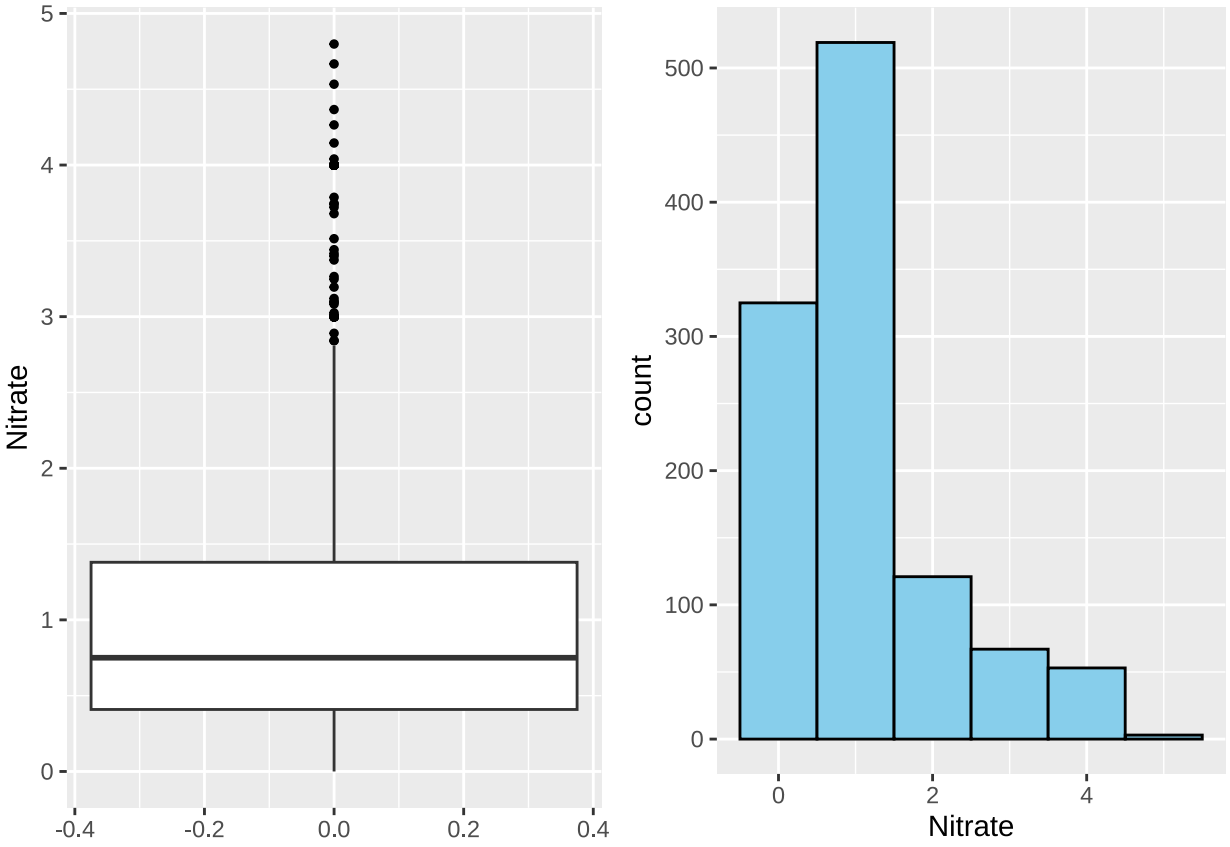
BOD

```
box_plot <- ggplot(data=train, aes(x=BOD)) +
  geom_boxplot(outlier.colour="black",
               outlier.size=1) + coord_flip()+
  labs(x = "BOD")

hist_plot <- ggplot(train, aes(x = BOD)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black")

ggarrange(box_plot, hist_plot, ncol = 2)
```



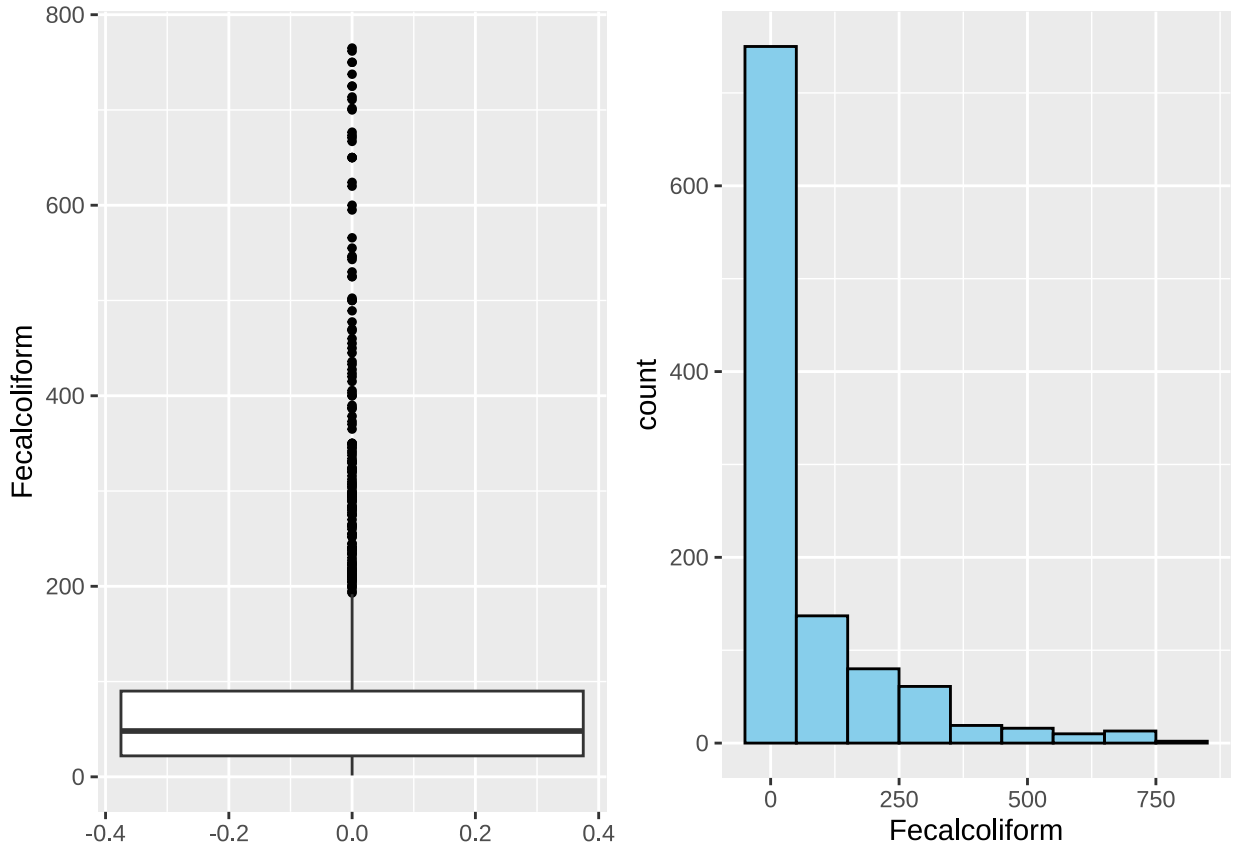
Nitrate

```
box_plot <- ggplot(data=train, aes(x=Nitrate)) +
  geom_boxplot(outlier.colour="black",
               outlier.size=1) + coord_flip()+
  labs(x = "Nitrate")

hist_plot <- ggplot(train, aes(x = Nitrate)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black")

ggarrange(box_plot, hist_plot, ncol = 2)
```



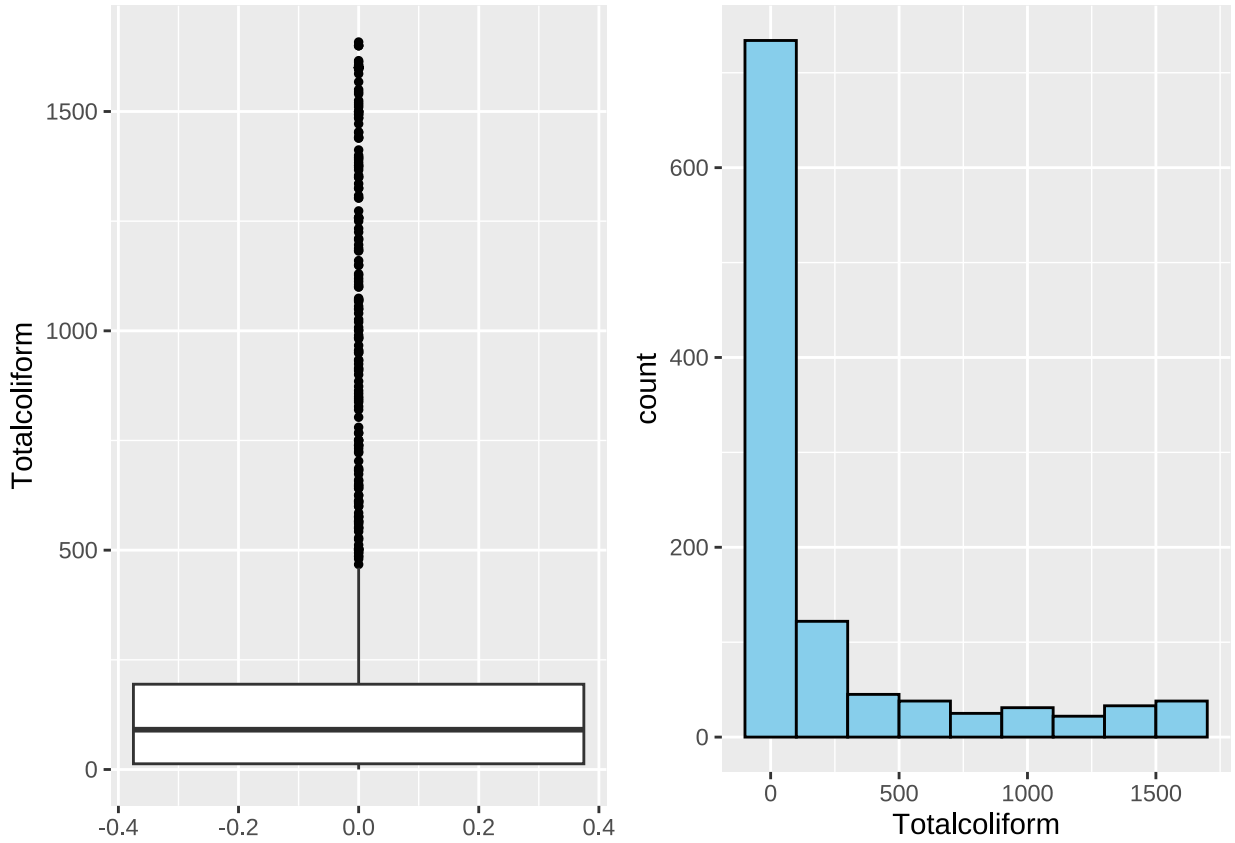
Fecalcoliform

```
box_plot <- ggplot(data=train, aes(x=Fecalcoliform)) +
  geom_boxplot(outlier.colour="black",
               outlier.size=1) + coord_flip()+
  labs(x = "Fecalcoliform")

hist_plot <- ggplot(train, aes(x = Fecalcoliform)) +
  geom_histogram(binwidth = 100, fill = "skyblue", color = "black")

ggarrange(box_plot, hist_plot, ncol = 2)
```



Totalcoliform

```
box_plot <- ggplot(data=train, aes(x=Totalcoliform)) +
  geom_boxplot(outlier.colour="black",
               outlier.size=1) + coord_flip()+
  labs(x = "Totalcoliform")

hist_plot <- ggplot(train, aes(x = Totalcoliform)) +
  geom_histogram(binwidth = 200, fill = "skyblue", color = "black")

ggarrange(box_plot, hist_plot, ncol = 2)
```



Summary

Predictors' behaviors which are worth mentioning are:

- Positive skewness: Conductivity, BOD, Nitrate.
- Strong Positive Skewness: Fecalcoliform, Totalcoliform

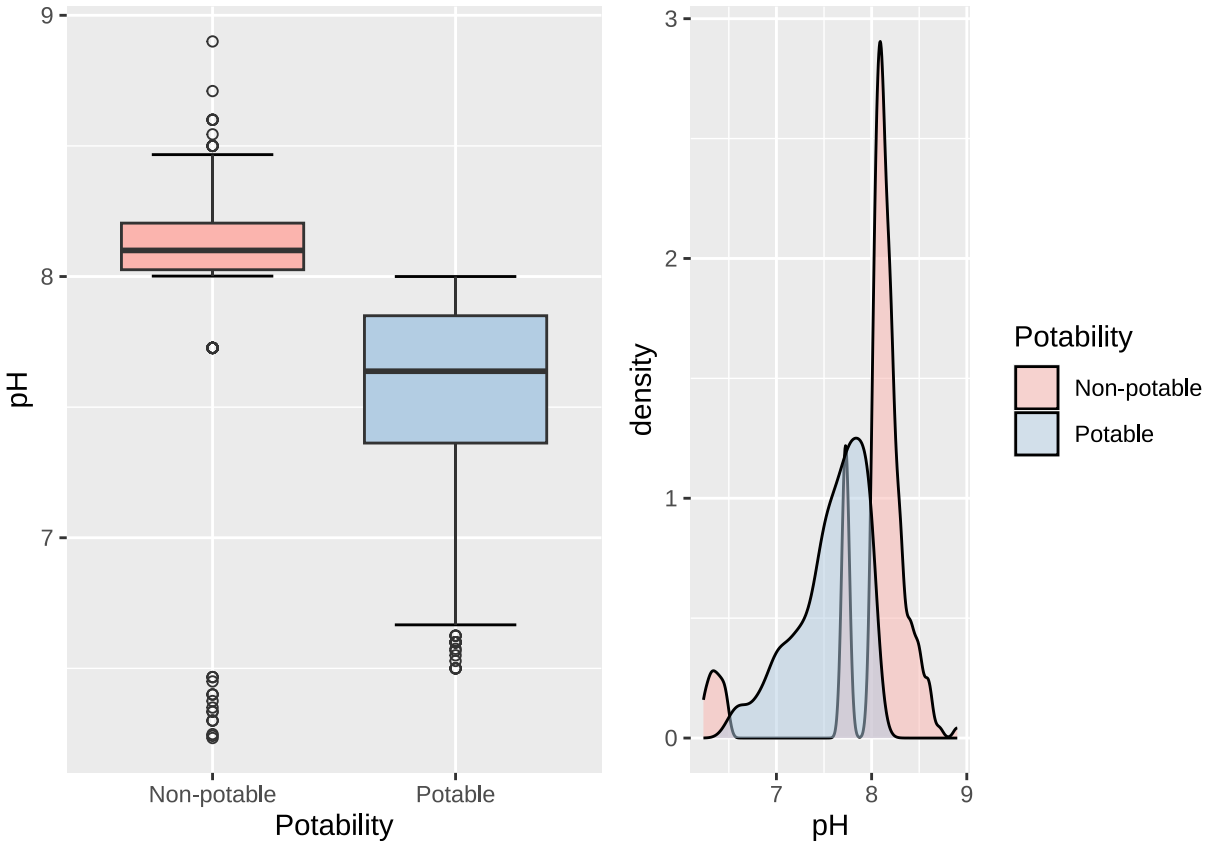
Bivariate Analysis

The goal of bivariate analysis is to visualize the distribution of each predictor for both Potable and Non-Potable Water cases. This analysis aims to determine if there is a significant difference in the distribution depending on the label's value, and ultimately, to determine whether the considered predictor is statistically significant in distinguishing between the two classes.

pH

It is significant, with different medians and different 1st and 3rd quartile in the two cases. Outliers are more evident for the Non-Potable distribution.

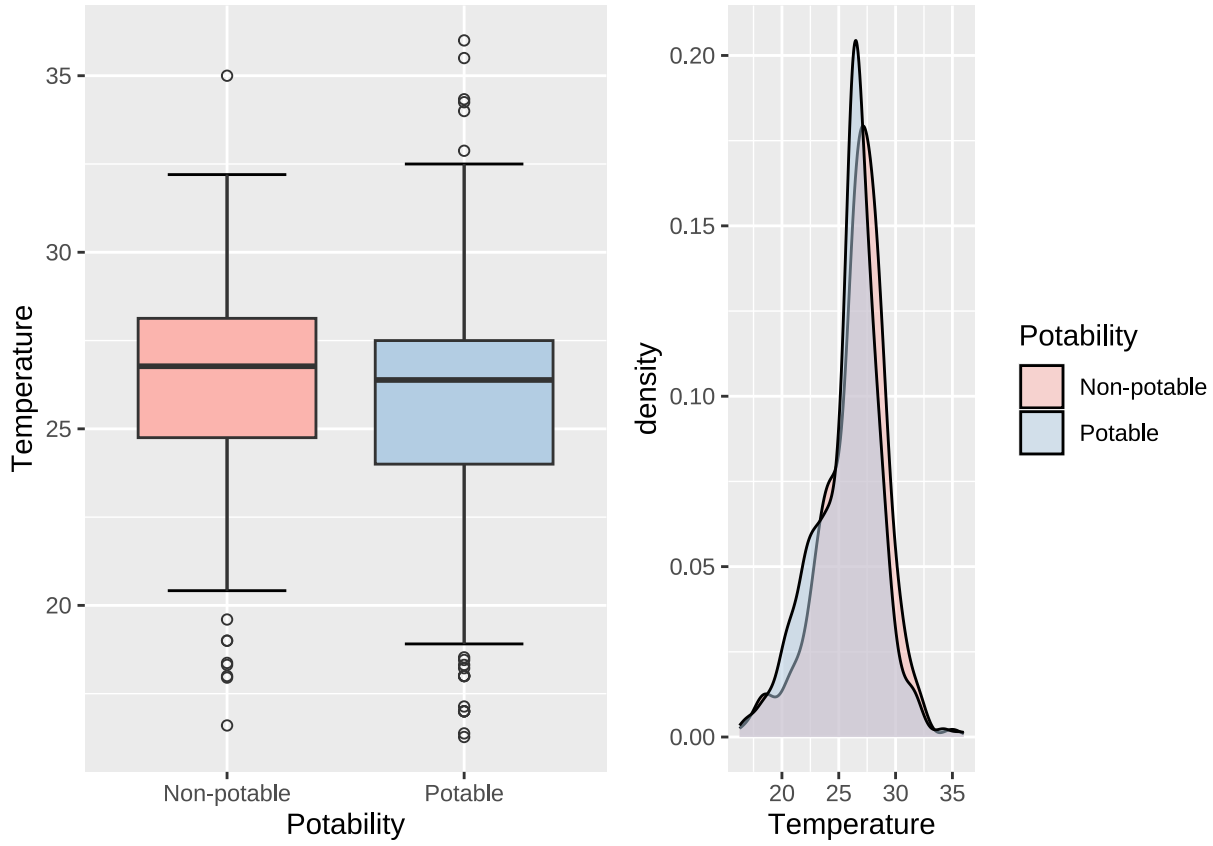
```
dens_plot<- ggplot(train, aes(x = pH, fill = Potability)) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))

box_plot<- ggplot(data = subset(train),
                  mapping = aes(x = Potability, y = pH, fill = Potability)) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))+
  stat_boxplot( aes(Potability, pH),
                geom='errorbar', linetype=1, width=0.5)+
  scale_x_discrete(labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability")+
  geom_boxplot(outlier.shape=1, na.rm=T)+
  guides(fill="none")

ggarrange(box_plot, dens_plot, ncol = 2)
```


9
8
7
pH
Non-potable
Potable
Potability
3
2
1
0
density
7
8
9
pH
Potability
Non-potable
Potable

Temperature

It is not significant, since the distribution is almost the same for the Potable and Non-Potable case. There are more outliers for the Potable distribution.

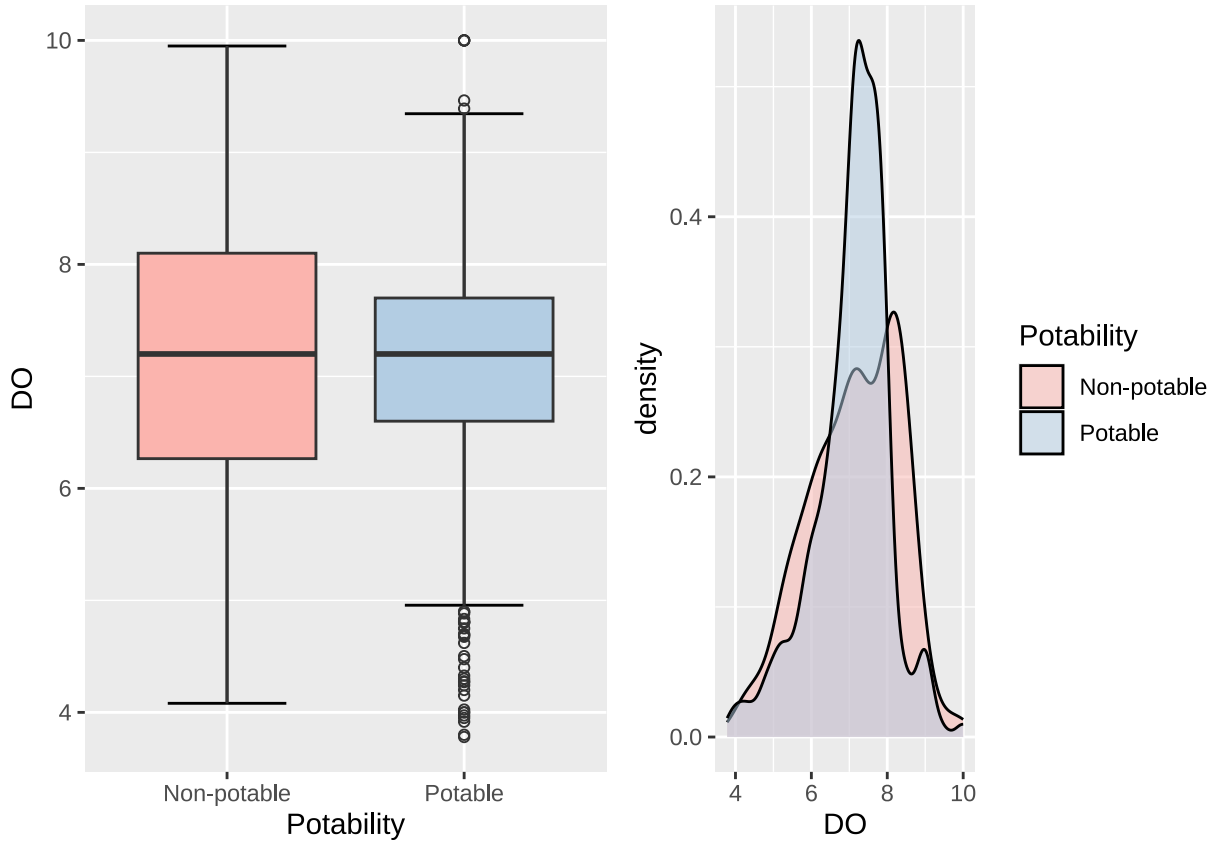
```
dens_plot<- ggplot(train, aes(x = Temperature, fill = Potability)) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))

box_plot<- ggplot(data = subset(train),
                  mapping = aes(x = Potability,
                                y = Temperature, fill = Potability)) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))+
  stat_boxplot( aes(Potability, Temperature),
                geom='errorbar', linetype=1, width=0.5)+
  scale_x_discrete(labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability")+
  geom_boxplot(outlier.shape=1, na.rm=T)+
  guides(fill="none")

ggarrange(box_plot, dens_plot, ncol = 2)
```



DO

The median is the same, but the distribution is slightly different for Potable and Non-Potable case. The Non-Potable distribution has more outliers.

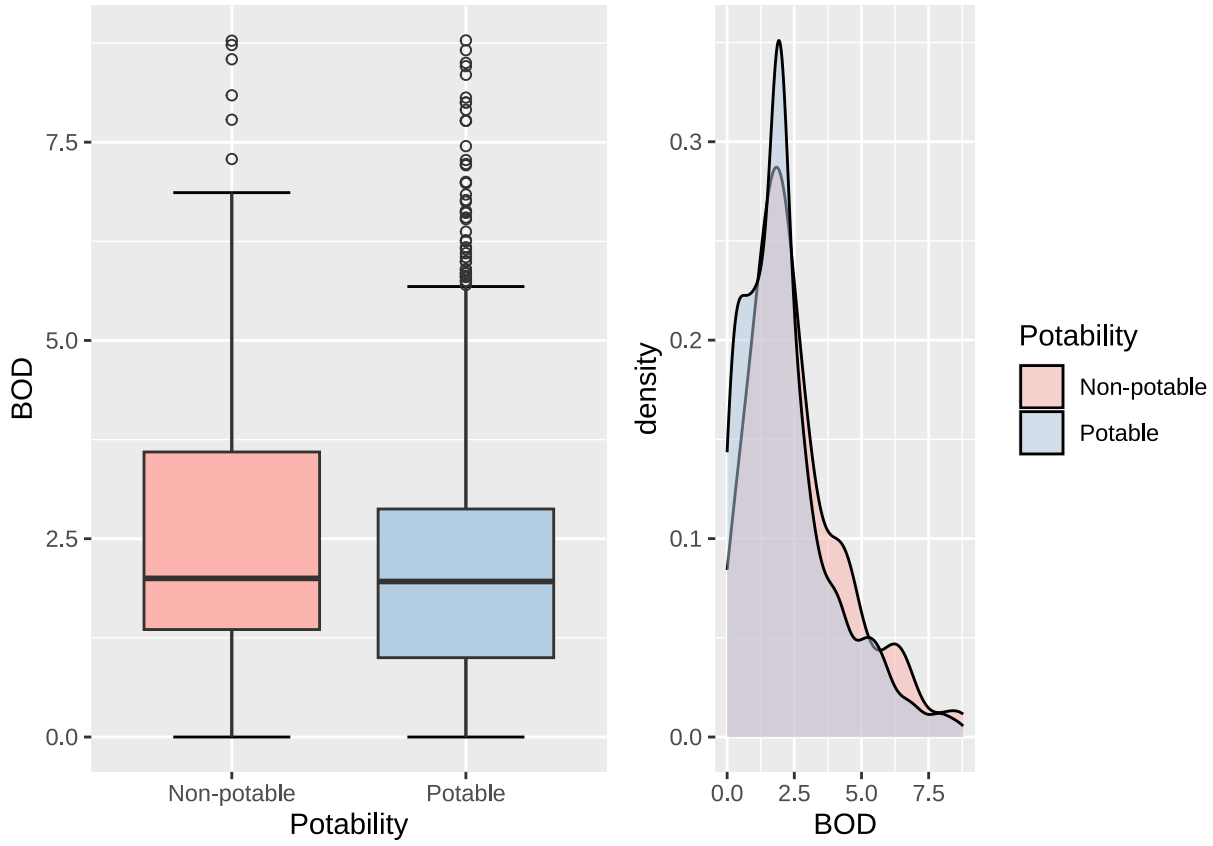
```
dens_plot<- ggplot(train, aes(x = DO, fill = Potability)) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))

box_plot<- ggplot(data = subset(train),
                  mapping = aes(x = Potability,
                                y = DO, fill = Potability)) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))+
  stat_boxplot( aes(Potability, DO),
                geom='errorbar', linetype=1, width=0.5)+
  scale_x_discrete(labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability")+
  geom_boxplot(outlier.shape=1, na.rm=T)+
  guides(fill="none")

ggarrange(box_plot, dens_plot, ncol = 2)
```



BOD

Similarly to the BOD variable, the median is the same while the distribution is slightly different for Potable and Non-Potable case. The Non-Potable distribution has more outliers.

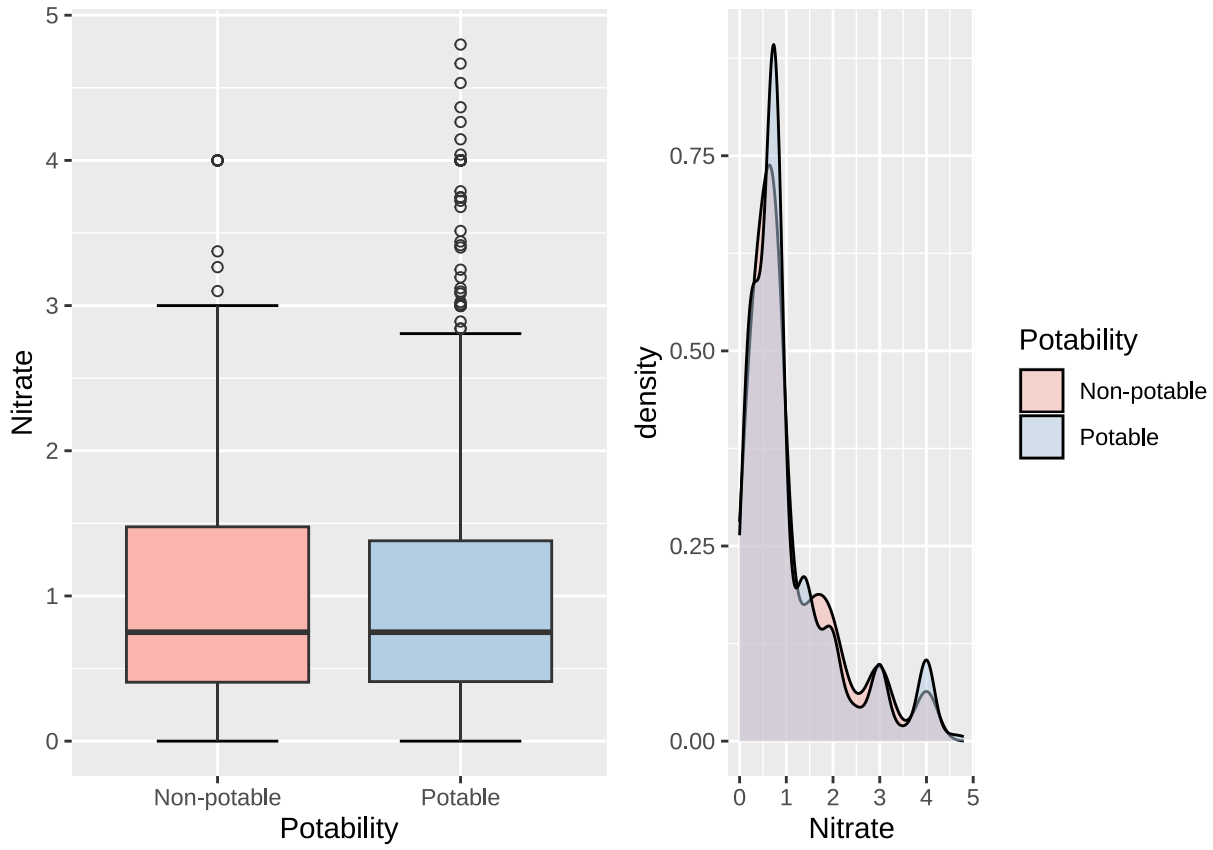
```
dens_plot<- ggplot(train, aes(x = BOD, fill = Potability)) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))

box_plot<- ggplot(data = subset(train),
                  mapping = aes(x = Potability,
                                y = BOD, fill = Potability)) +
  scale_fill_brewer(palette = "Pastel1",labels =c("Non-potable", "Potable"))+
  stat_boxplot( aes(Potability, BOD),
                geom='errorbar', linetype=1, width=0.5)+
  scale_x_discrete(labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability")+
  geom_boxplot(outlier.shape=1, na.rm=T)+
  guides(fill="none")

ggarrange(box_plot, dens_plot, ncol = 2)
```



Nitrate

It is not significant, the distribution is almost the same. There are more outliers for the Potable distribution.

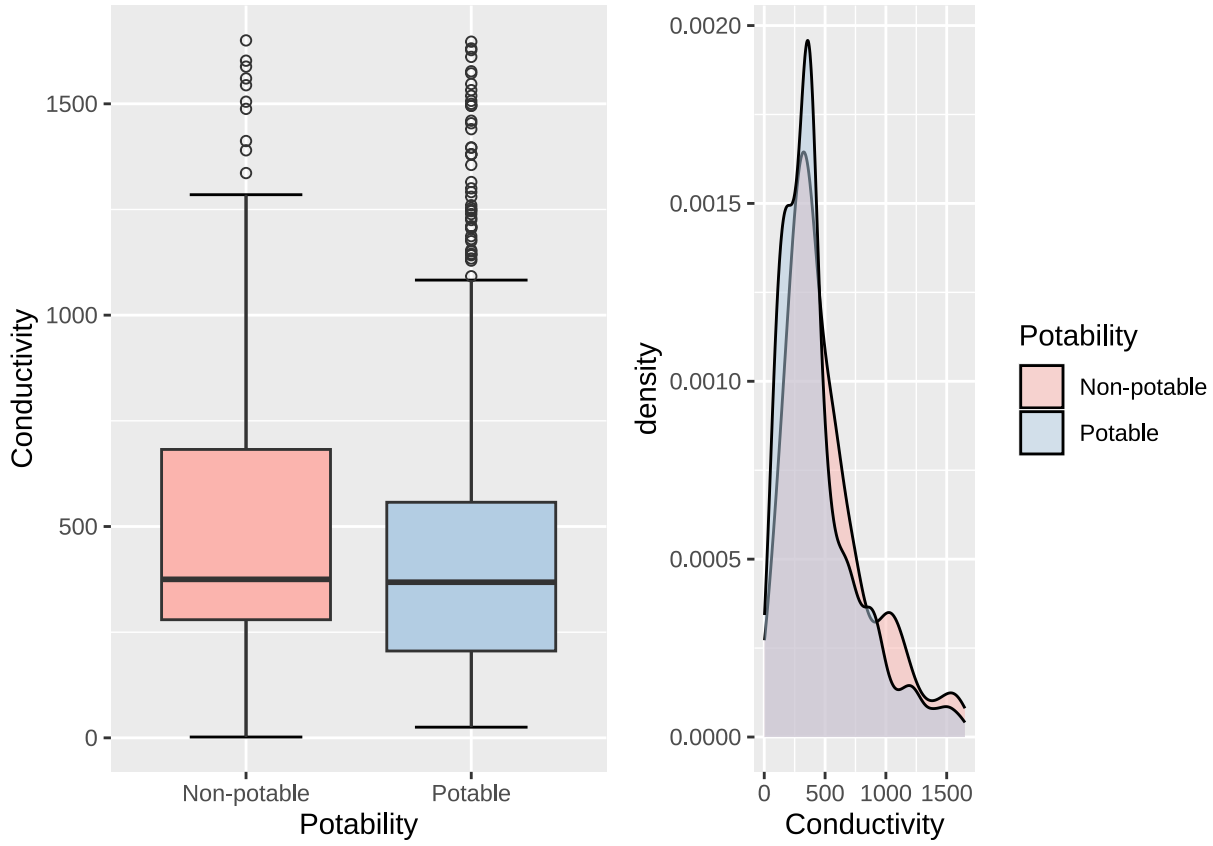
```
dens_plot<- ggplot(train, aes(x = Nitrate, fill = Potability)) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))

box_plot<- ggplot(data = subset(train),
                  mapping = aes(x = Potability,
                                y = Nitrate, fill = Potability)) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))+
  stat_boxplot( aes(Potability, Nitrate),
                geom='errorbar', linetype=1, width=0.5)+
  scale_x_discrete(labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability")+
  geom_boxplot(outlier.shape=1, na.rm=T)+
  guides(fill="none")

ggarrange(box_plot, dens_plot, ncol = 2)
```



Conductivity

The median is the same, but the distribution is slightly different in the two cases. The Potable distribution has more outliers.

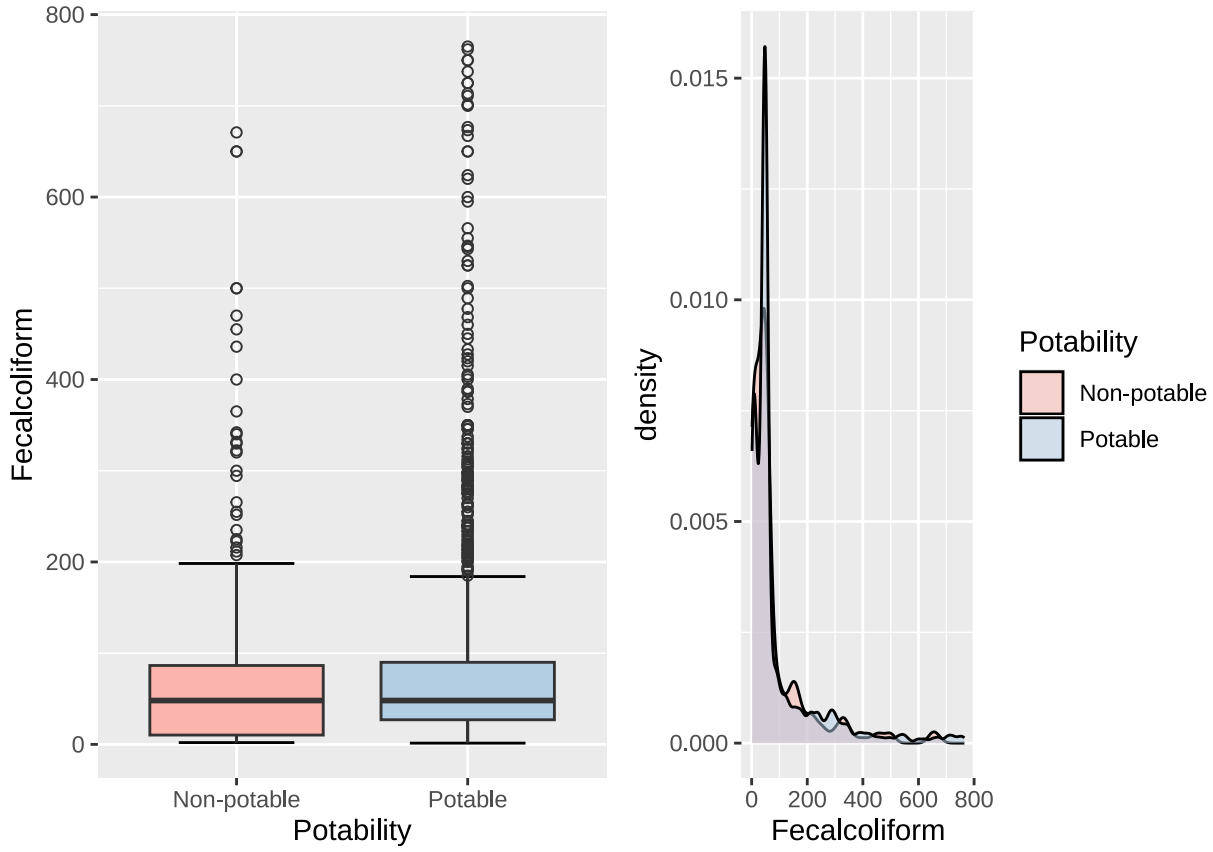
```
dens_plot<- ggplot(train, aes(x = Conductivity, fill = Potability)) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))

box_plot<- ggplot(data = subset(train),
                  mapping = aes(x = Potability,
                                y = Conductivity, fill = Potability)) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))+
  stat_boxplot( aes(Potability, Conductivity),
                geom='errorbar', linetype=1, width=0.5)+
  scale_x_discrete(labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability")+
  geom_boxplot(outlier.shape=1, na.rm=T)+
  guides(fill="none")

ggarrange(box_plot, dens_plot, ncol = 2)
```



Fecalcoliform

It is not significant, the distributions are almost the same. The Potable distribution has more outliers.

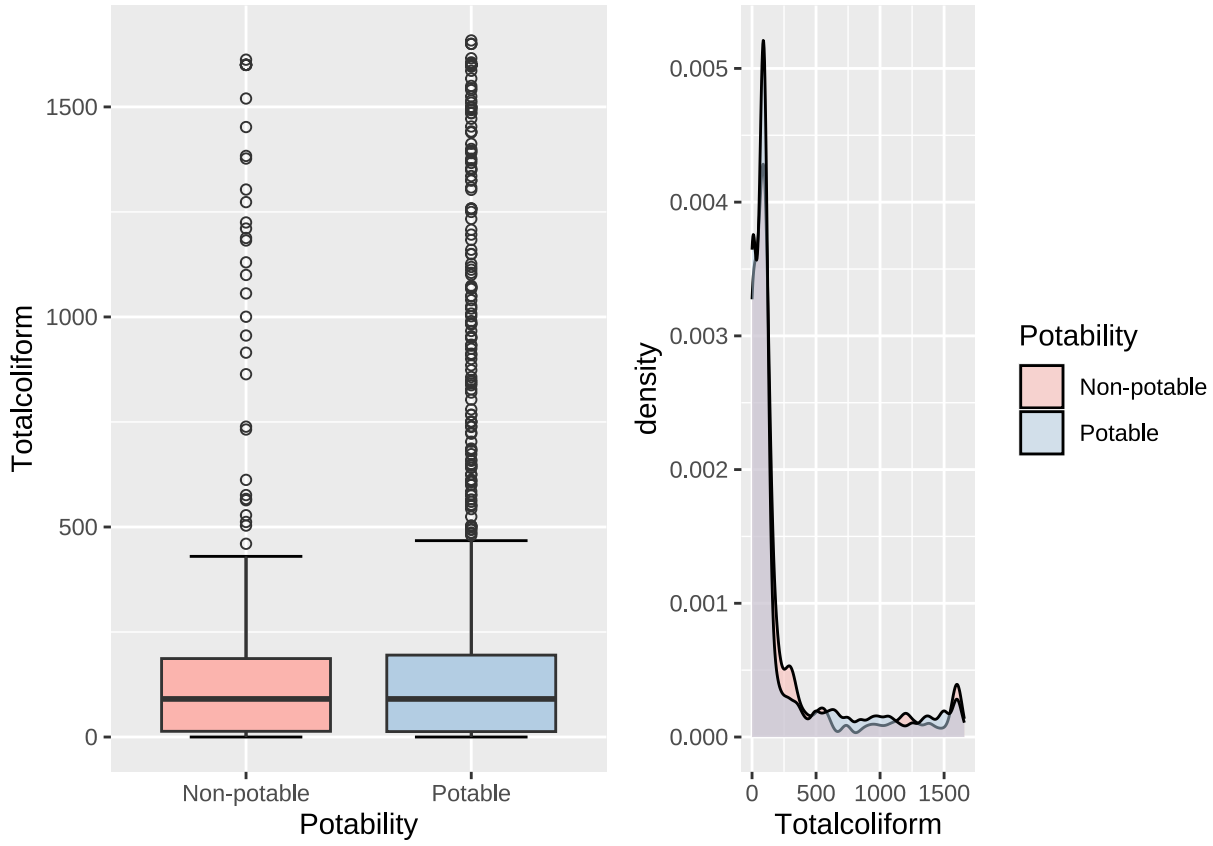
```
dens_plot<- ggplot(train, aes(x = Fecalcoliform, fill = Potability)) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))

box_plot<- ggplot(data = subset(train),
                  mapping = aes(x = Potability,
                                y = Fecalcoliform, fill = Potability)) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))+
  stat_boxplot( aes(Potability, Fecalcoliform),
                geom='errorbar', linetype=1, width=0.5)+
  scale_x_discrete(labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability")+
  geom_boxplot(outlier.shape=1, na.rm=T)+
  guides(fill="none")

ggarrange(box_plot, dens_plot, ncol = 2)
```



Totalcoliform

Similarly to Fecalcoliform, it is not significant and the distributions are almost the same. The Potable distribution has more outliers.

```
dens_plot<- ggplot(train, aes(x = Totalcoliform, fill = Potability)) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))

box_plot<- ggplot(data = subset(train),
                  mapping = aes(x = Potability,
                                y = Totalcoliform, fill = Potability)) +
  scale_fill_brewer(palette = "Pastel1",labels = c("Non-potable", "Potable"))+
  stat_boxplot( aes(Potability, Totalcoliform),
                geom='errorbar', linetype=1, width=0.5)+
  scale_x_discrete(labels = c("Non-potable", "Potable"))+
  labs(fill = "Potability")+
  geom_boxplot(outlier.shape=1, na.rm=T)+
  guides(fill="none")

ggarrange(box_plot, dens_plot, ncol = 2)
```

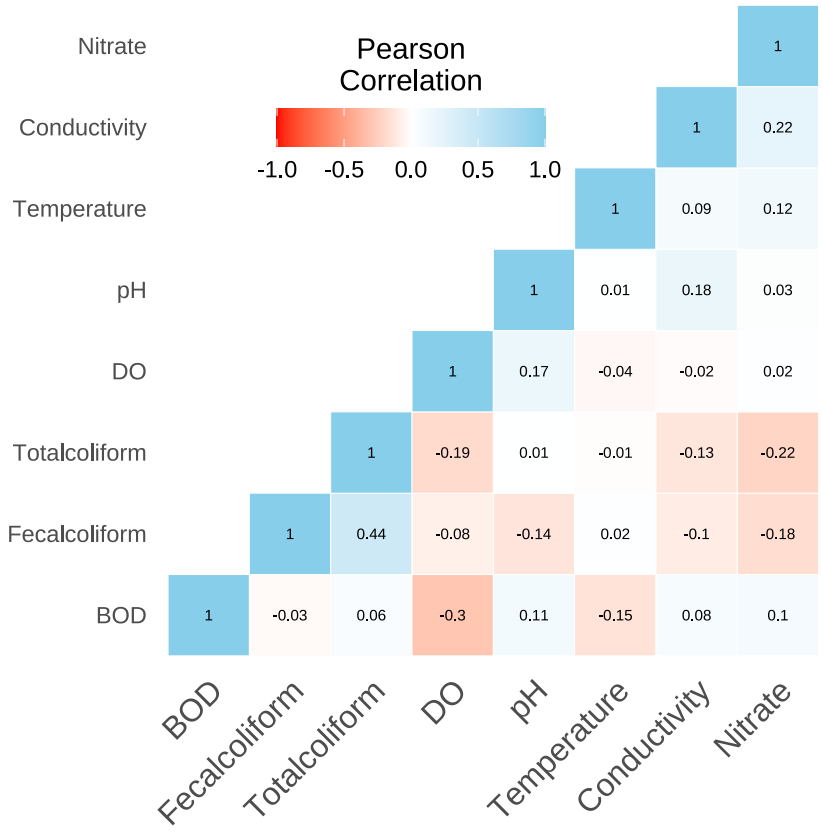


Correlations and Collinearity

Correlation helps understanding the relationship between two variables. By measuring the degree to which two variables are related, we can gain insights into how changes in one variable might affect the other. Focusing on the correlations between the predictors, it's possible to build the following matrix:

```
cormat <- round(cor(train[sapply(train, is.numeric)]),2)
melted_cormat <- melt(cormat)

reorder_cormat <- function(cormat){
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat <-cormat[hc$order, hc$order]
}

# Get lower triangle of the correlation matrix
get_lower_tri<-function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}

# Get upper triangle of the correlation matrix
get_upper_tri <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}

cormat <- reorder_cormat(cormat)
upper_tri <- get_upper_tri(cormat)
# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Create a ggheatmap
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low="red", high="skyblue",midpoint = 0,
                       limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+ # minimal theme
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()

# Printing the heatmap
ggheatmap +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 2) +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.ticks = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.6, 0.7),
```

```
  legend.direction = "horizontal")+
guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                             title.position = "top", title.hjust = 0.5))
```



Looking at the matrix, it's easy to see that most variables are not highly correlated. Anyway, it's worth exploring the relationships among some of them:

Variable 1	Variable 2	Correlation
Totalcoliform	Fecalcoliform	0.44
DO	BOD	-0.30

- Total coliform and Fecal coliform are both types of bacteria commonly used as indicators of water quality. Total coliform bacteria are found in the intestines of humans and animals, as well as in soil and vegetation. Fecal coliform bacteria, on the other hand, are a subset of total coliform bacteria that are found specifically in the feces of warm-blooded animals. The positive correlation between these two variables can be explained by the fact that both Total coliform and Fecal coliform are indicators of contamination from human or animal waste. Therefore, if there is a high level of Fecal coliform bacteria in a water sample, it is likely that there will also be a high level of Total coliform bacteria present.
- Dissolved Oxygen (DO) is a measure of the amount of oxygen that is dissolved in water. This is important because aquatic life require oxygen to survive. Biological Oxygen Demand (BOD), on the other hand, is a measure of the amount of oxygen that is required by microorganisms to break down organic matter in the water. The negative correlation between DO and BOD can be explained by the fact that high levels of BOD can reduce the amount of dissolved oxygen in the water. Therefore, if

there is a high level of BOD in a water sample, it is likely that there will be a lower level of dissolved oxygen present as well. However, the correlation between these two variables is relatively weak, which suggests that other factors may also be influencing the levels of DO and BOD in the water.

Models

As previously mentioned during the exploratory analysis, the response variable is imbalanced. To ensure effective modeling performance, a downsampling technique is applied to obtain a properly balanced dataset.

```
downsampled <- downSample(x = train[, -9], y = train$Potability,
                          yname="Potability")

downsampled_prop <- prop.table(table(downsampled$Potability))
row.names(downsampled_prop) <- c( "Non-Potable","Potable")
pander((downsampled_prop), style = "rmarkdown")
```

Non-Potable	Potable
0.5	0.5

Besides evaluating the models on the downsampled set, they will also be assessed on the original training set with imbalanced Potability to determine whether downsampling has indeed improved performance. Specifically, the proportion of the Potability variable in the original training set is as follows:

```
original_prop <- prop.table(table(train$Potability))
row.names(original_prop) <- c( "Non-Potable","Potable")
pander((original_prop), style = "rmarkdown")
```

Non-Potable	Potable
0.216	0.784

To assess and compare the performance of the models, the following metrics are mainly used:

- Accuracy, since we are dealing with a binary classification problem
- False Positive rate, which quantifies the proportion of Non-Potable samples that are erroneously identified as Potable. This metric holds particular importance in this analysis, as misclassifying a Non-Potable sample as Potable can have severe health and environmental implications.

Logistic Regression

The Logistic Regression is a binary classification model defined as

$$\pi(x) = \frac{e^z}{1 - e^z}$$

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

where $\pi(x)$ is the probability of the binary outcome given the predictors $x_1, x_2, \ldots, x_p$;

The following models have been selected through the stepwise selection with the AIC criterion starting from the null model. This criterion is able to select the best model putting a penalization on the complexity caused by the number of predictors, which are added or removed on each step.

Original Train Set

```
lr_i <- glm(Potability ~ . , data=train, family=binomial)
summary(lr_i)
```

```
##
## Call:
## glm(formula = Potability ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0106   0.0805   0.3692   0.6665   1.2990
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    3.637e+01  2.970e+00  12.243  < 2e-16 ***
## Temperature   -1.014e-01  3.101e-02  -3.270  0.00107 **
## DO             1.188e-01  8.604e-02   1.381  0.16734
## pH            -4.249e+00  3.613e-01 -11.760  < 2e-16 ***
## Conductivity  -4.859e-05  2.718e-04  -0.179  0.85811
## BOD           -7.891e-02  5.227e-02  -1.510  0.13109
## Nitrate        9.503e-02  9.585e-02   0.991  0.32148
## Fecalcoliform -7.489e-04  7.706e-04  -0.972  0.33110
## Totalcoliform  4.384e-04  2.339e-04   1.874  0.06091 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1135.41  on 1087  degrees of freedom
## Residual deviance:  862.11  on 1079  degrees of freedom
## AIC: 880.11
##
## Number of Fisher Scoring iterations: 6
```

```
# Using stepwise model selection to choose the best model
step_model_lr_i <- stepAIC(lr_i, direction = "both")
```

```
## Start:  AIC=880.11
## Potability ~ Temperature + DO + pH + Conductivity + BOD + Nitrate +
##     Fecalcoliform + Totalcoliform
##
##                 Df Deviance     AIC
## - Conductivity   1   862.15  878.15
## - Fecalcoliform  1   863.03  879.03
## - Nitrate        1   863.12  879.12
## - DO             1   864.01  880.01
## <none>               862.11  880.11
## - BOD            1   864.39  880.39
```

```
## - Totalcoliform  1   865.76  881.76
## - Temperature    1   873.32  889.32
## - pH             1  1098.63 1114.63
##
## Step:  AIC=878.15
## Potability ~ Temperature + DO + pH + BOD + Nitrate + Fecalcoliform +
##     Totalcoliform
##
##                 Df Deviance     AIC
## - Fecalcoliform  1   863.07  877.07
## - Nitrate        1   863.12  877.12
## - DO             1   864.08  878.08
## <none>               862.15  878.15
## - BOD            1   864.51  878.51
## - Totalcoliform  1   865.86  879.86
## + Conductivity   1   862.11  880.11
## - Temperature    1   873.59  887.59
## - pH             1  1106.42 1120.42
##
## Step:  AIC=877.07
## Potability ~ Temperature + DO + pH + BOD + Nitrate + Totalcoliform
##
##                 Df Deviance     AIC
## - Nitrate        1   864.24  876.24
## - DO             1   864.95  876.95
## <none>               863.07  877.07
## - BOD            1   865.47  877.47
## - Totalcoliform  1   865.90  877.90
## + Fecalcoliform  1   862.15  878.15
## + Conductivity   1   863.03  879.03
## - Temperature    1   874.60  886.60
## - pH             1  1108.58 1120.58
##
## Step:  AIC=876.24
## Potability ~ Temperature + DO + pH + BOD + Totalcoliform
##
##                 Df Deviance     AIC
## - BOD            1   866.09  876.09
## <none>               864.24  876.24
## - DO             1   866.39  876.39
## - Totalcoliform  1   866.40  876.40
## + Nitrate        1   863.07  877.07
## + Fecalcoliform  1   863.12  877.12
## + Conductivity   1   864.24  878.24
## - Temperature    1   874.86  884.86
## - pH             1  1110.28 1120.28
##
## Step:  AIC=876.09
## Potability ~ Temperature + DO + pH + Totalcoliform
##
##                 Df Deviance     AIC
## <none>               866.09  876.09
## - Totalcoliform  1   868.15  876.15
## + BOD            1   864.24  876.24
```

```
## + Fecalcoliform  1   864.99  876.99
## + Nitrate        1   865.47  877.47
## + Conductivity   1   866.05  878.05
## - DO             1   870.81  878.81
## - Temperature    1   875.58  883.58
## - pH             1  1125.76 1133.76
```

```
# Print the summary of the selected model
summary(step_model_lr_i)
```

```
##
## Call:
## glm(formula = Potability ~ Temperature + DO + pH + Totalcoliform,
##     family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0133   0.0840   0.3725   0.6723   1.2592
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   35.7726849  2.8978845  12.344  < 2e-16 ***
## Temperature   -0.0896771  0.0296888  -3.021  0.00252 **
## DO             0.1701627  0.0784077   2.170  0.02999 *
## pH            -4.2771782  0.3527004 -12.127  < 2e-16 ***
## Totalcoliform  0.0002919  0.0002070   1.410  0.15850
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1135.41  on 1087  degrees of freedom
## Residual deviance:  866.09  on 1083  degrees of freedom
## AIC: 876.09
##
## Number of Fisher Scoring iterations: 6
```

```
# predictions
fit_lr_i<- predict(step_model_lr_i, test, type = "response")
# label assignement
pred.lr_i<- ifelse(fit_lr_i > 0.5, 1, 0)

# Create confusion matrix
confusion_matrix <- table(pred.lr_i,test$Potability)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- confusion_matrix
```

```r
# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

cat("\\usepackage{longtable}\n")
```

```
## \usepackage{longtable}
```

```r
kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for Logistic Regression in Original Train Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 10: Confusion Matrix for Logistic Regression in Original Train Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	27	0	27
Pred. Potable	36	210	246
Total	63	210	273

```r
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.57
```

```r
#Accuracy
acc_lr_i <- round(mean(pred.lr_i == test$Potability),2)
cat("Accuracy:", acc_lr_i, "\n")
```

```
## Accuracy: 0.87
```

The final selected model contains the variables $Temperature$, DO, pH, $Totalcoliform$.

Downsampled Set

```r
down_lr1 <- glm(Potability ~ . , data=downsampled, family=binomial)
summary(down_lr1)
```

```
##
## Call:
## glm(formula = Potability ~ ., family = binomial, data = downsampled)
##
```

```
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.83877  -0.88845  -0.01855   1.01234   1.67001
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   21.5519512  2.7570010   7.817  5.4e-15 ***
## Temperature   -0.0795477  0.0360578  -2.206   0.0274 *
## DO             0.0878664  0.1052508   0.835   0.4038
## pH            -2.5607820  0.3270741  -7.829  4.9e-15 ***
## Conductivity   0.0002536  0.0003113   0.815   0.4153
## BOD           -0.1151150  0.0635921  -1.810   0.0703 .
## Nitrate        0.0788097  0.1138860   0.692   0.4889
## Fecalcoliform -0.0004998  0.0008995  -0.556   0.5785
## Totalcoliform  0.0002698  0.0002862   0.943   0.3457
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 651.56  on 469  degrees of freedom
## Residual deviance: 541.25  on 461  degrees of freedom
## AIC: 559.25
##
## Number of Fisher Scoring iterations: 4
```

```
# Using stepwise model selection to choose the best model
down_step_model_lr <- stepAIC(down_lr1, direction = "both")
```

```
## Start:  AIC=559.25
## Potability ~ Temperature + DO + pH + Conductivity + BOD + Nitrate +
##     Fecalcoliform + Totalcoliform
##
##                 Df Deviance    AIC
## - Fecalcoliform  1   541.56 557.56
## - Nitrate        1   541.73 557.73
## - Conductivity   1   541.92 557.92
## - DO             1   541.95 557.95
## - Totalcoliform  1   542.14 558.14
## <none>               541.25 559.25
## - BOD            1   544.60 560.60
## - Temperature    1   546.23 562.23
## - pH             1   634.50 650.50
##
## Step:  AIC=557.56
## Potability ~ Temperature + DO + pH + Conductivity + BOD + Nitrate +
##     Totalcoliform
##
##                 Df Deviance    AIC
## - Nitrate        1   542.11 556.11
## - Totalcoliform  1   542.19 556.19
## - Conductivity   1   542.21 556.21
## - DO             1   542.26 556.26
## <none>               541.56 557.56
```

```
## - BOD            1   544.89 558.89
## + Fecalcoliform  1   541.25 559.25
## - Temperature    1   546.55 560.55
## - pH             1   635.40 649.40
##
## Step:  AIC=556.11
## Potability ~ Temperature + DO + pH + Conductivity + BOD + Totalcoliform
##
##                 Df Deviance    AIC
## - Totalcoliform  1   542.54 554.54
## - DO             1   542.93 554.93
## - Conductivity   1   543.10 555.10
## <none>               542.11 556.11
## - BOD            1   545.14 557.14
## + Nitrate        1   541.56 557.56
## + Fecalcoliform  1   541.73 557.73
## - Temperature    1   546.65 558.65
## - pH             1   637.03 649.03
##
## Step:  AIC=554.54
## Potability ~ Temperature + DO + pH + Conductivity + BOD
##
##                 Df Deviance    AIC
## - DO             1   543.21 553.21
## - Conductivity   1   543.43 553.43
## <none>               542.54 554.54
## - BOD            1   545.42 555.42
## + Totalcoliform  1   542.11 556.11
## + Nitrate        1   542.19 556.19
## + Fecalcoliform  1   542.44 556.44
## - Temperature    1   546.99 556.99
## - pH             1   637.06 647.06
##
## Step:  AIC=553.21
## Potability ~ Temperature + pH + Conductivity + BOD
##
##                 Df Deviance    AIC
## - Conductivity   1   544.14 552.14
## <none>               543.21 553.21
## + DO             1   542.54 554.54
## + Nitrate        1   542.74 554.74
## + Totalcoliform  1   542.93 554.93
## + Fecalcoliform  1   543.07 555.07
## - BOD            1   547.52 555.52
## - Temperature    1   547.94 555.94
## - pH             1   638.52 646.52
##
## Step:  AIC=552.14
## Potability ~ Temperature + pH + BOD
##
##                 Df Deviance    AIC
## <none>               544.14 552.14
## + Conductivity   1   543.21 553.21
## + Nitrate        1   543.34 553.34
```

```
## + DO             1   543.43 553.43
## + Totalcoliform  1   543.94 553.94
## + Fecalcoliform  1   543.98 553.98
## - BOD            1   548.21 554.21
## - Temperature    1   548.65 554.65
## - pH             1   639.51 645.51
```

```
# Print the summary of the selected model
summary(down_step_model_lr)
```

```
##
## Call:
## glm(formula = Potability ~ Temperature + pH + BOD, family = binomial,
##     data = downsampled)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.82174  -0.89662  -0.01048   1.00967   1.66421
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 21.28446    2.64218   8.056 7.91e-16 ***
## Temperature -0.07382    0.03514  -2.101   0.0356 *
## pH          -2.43504    0.30888  -7.883 3.19e-15 ***
## BOD         -0.11863    0.05959  -1.991   0.0465 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 651.56  on 469  degrees of freedom
## Residual deviance: 544.14  on 466  degrees of freedom
## AIC: 552.14
##
## Number of Fisher Scoring iterations: 4
```

```
#predictions
down_fit_lr<- predict(down_step_model_lr, test, type = "response")
# label assignement
down_pred.lr<- ifelse(down_fit_lr > 0.5, 1, 0)

# Create confusion matrix
confusion_matrix <- table(down_pred.lr,test$Potability)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- confusion_matrix

# Add a total row to the data frame
```

```
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))


kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for Logistic Regression in Downsampled Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 11: Confusion Matrix for Logistic Regression in Downsampled Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	48	48	96
Pred. Potable	15	162	177
Total	63	210	273

```
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.24
```

```
#Accuracy
down_acc_lr <- round(mean(down_pred.lr == test$Potability),2)
cat("Accuracy:", down_acc_lr, "\n")
```

```
## Accuracy: 0.77
```

The final selected model contains the variables *Temperature*, *pH*, *BOD*.

Comparison The downsampled Logistic Regression model is the best one, with a lower AIC (552.14) compared to the model fitted on the Original Training set (876.09). Moreover, even if the latter scores a better accuracy value, it wrongly classifies 57% of the Non-Potable observations, while the downsampled one misclassifies only 24% of the samples. It's also worth mentioning that the *pH* variable is very significant in both models, as expected from the Bivariate Exploratory Analysis.

Regularized Logistic Regression

It is a class of models based on the Logistic Regression, with the addition of a penalty term witch is used to prevent overfitting. In both Ridge and Lasso regression, a 10-fold Cross-Validation is performed to check different λ values and select the one that best handle the variance-bias trade off. In order to explain the choice made, for each model is plotted the cross-validation error as a function of the logarithm of the lambda values tested. The plot will show the mean cross-validation error as a solid line, and the standard error of the mean as a shaded area around the solid line, highlighting the λ value chosen.

Ridge

It asymptotically shrinks the penalized coefficients towards zero, making it the best choice when most variables are known to be useful from the context and multicollinearity needs to be mitigated keeping all the variables anyway. It applies the L2 Penalty to Logistic Regression in order to estimate the coefficients.

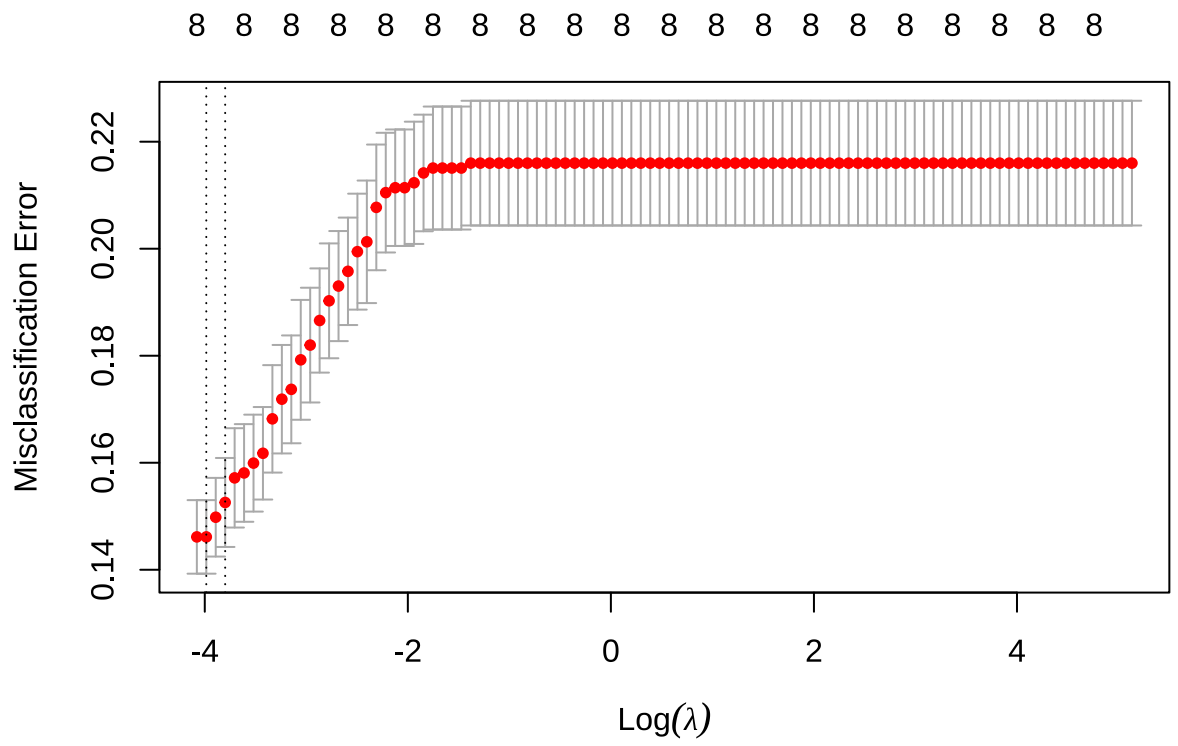
$$\hat{\beta}^R = \underset{\beta}{\operatorname{argmin}} \left[(y - X\beta)^\top (y - X\beta) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

where λ is an hyperparameter which is selected and tuned using Cross Validation.

Original Train Set

```
# Cross Validation for lambda values
ridge_cv_i <- cv.glmnet(as.matrix(train[,1:8]),
                        train$Potability,alpha = 0,
                        family = "binomial", type.measure = "class")
```

```
plot(ridge_cv_i)
```




```
cat("The value for the minimum lambda is ", ridge_cv_i$lambda.min)
```

```
## The value for the minimum lambda is  0.01859145
```

```
pred_ridge_i<- predict(ridge_cv_i, as.matrix(test[,1:8]), type = "class",
                       s = ridge_cv_i$lambda.min)

# Create confusion matrix
confusion_matrix <- table(test$Potability, pred_ridge_i)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- t(confusion_matrix)

# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for Ridge in Original Train Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 12: Confusion Matrix for Ridge in Original Train Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	16	13	29
Pred. Potable	47	197	244
Total	63	210	273

```
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.75
```

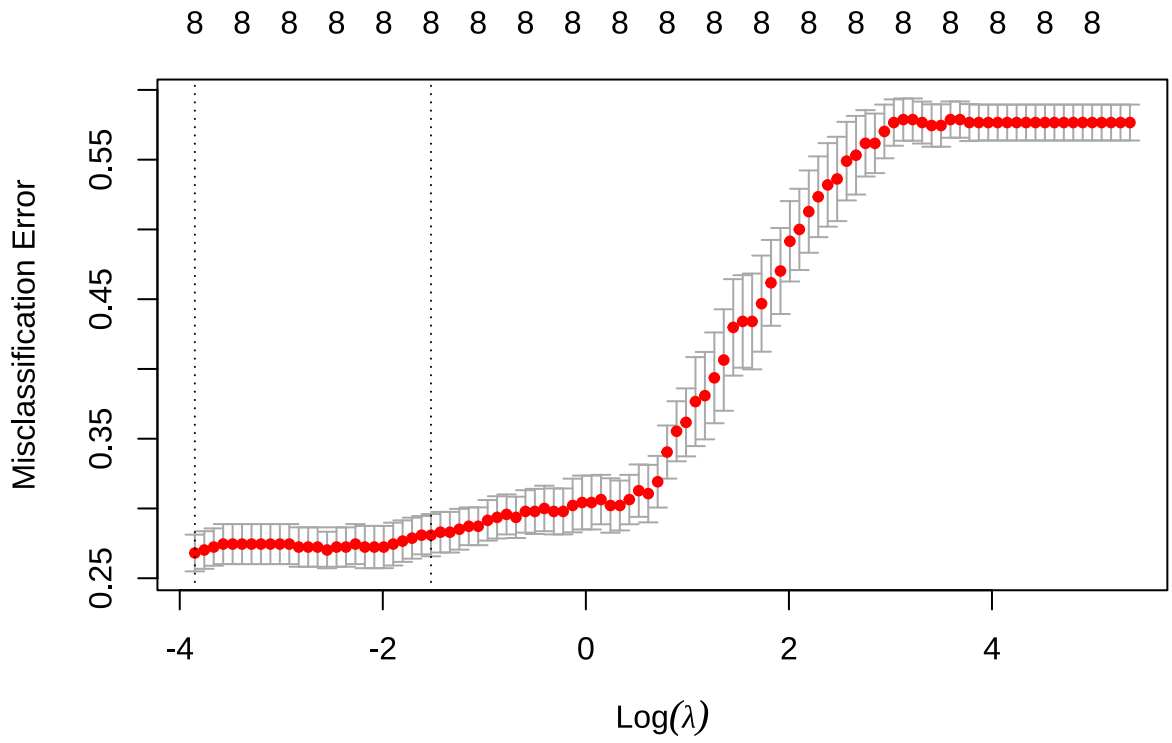
```
#Accuracy
acc_ridge_i <- round(mean(pred_ridge_i == test$Potability),2)
cat("Accuracy:", acc_ridge_i, "\n")
```

```
## Accuracy: 0.78
```

Downsampled Set

```r
# Cross Validation for lambda values
down_ridge_cv <- cv.glmnet(as.matrix(downsampled[,1:8]),
                      downsampled$Potability,alpha = 0,
                      family = "binomial", type.measure = "class")
```

```r
plot(down_ridge_cv)
```




```r
cat("The value for the minimum lambda is ", down_ridge_cv$lambda.min)
```

```
## The value for the minimum lambda is  0.02125082
```

```r
down_pred_ridge<- predict(down_ridge_cv, as.matrix(test[,1:8]), type = "class",
                     s = down_ridge_cv$lambda.min)

confusion_matrix <- table(test$Potability, down_pred_ridge)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- t(confusion_matrix)
```

```r
# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for Ridge in Downsampled Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 13: Confusion Matrix for Ridge in Downsampled Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	43	35	78
Pred. Potable	20	175	195
Total	63	210	273

```r
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.32
```

```r
#Accuracy
down_acc_ridge <- round(mean(down_pred_ridge == test$Potability),2)
cat("Accuracy:", down_acc_ridge, "\n")
```

```
## Accuracy: 0.8
```

Comparison The Ridge model trained over the downsampled set scores a slightly better accuracy value (0.80 vs 0.78) and at the same time achieves a much lower number of false positives (0.32 vs 0.75).

Lasso

Unlike Ridge, it exactly shrinks the penalized coefficients towards zero, performing variables exclusion for those which are not significant. It applies the L1 Penalty to Logistic Regression in order to estimate the coefficients.

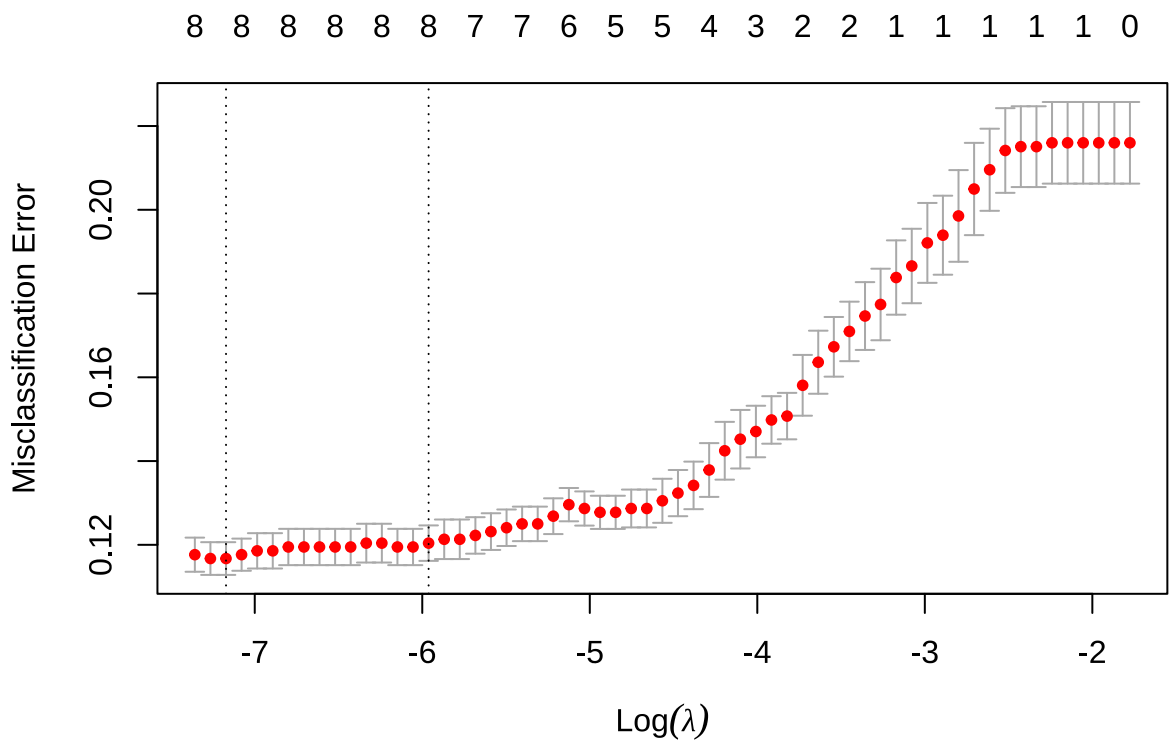
$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left[(y - X\beta)^\top (y - X\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right]$$

where, as before, λ is an hyperparameter which is selected and tuned using Cross-Validation.

Original Train Set

```
# Cross Validation for lambda values
lasso_cv_i <- cv.glmnet(as.matrix(train[,1:8]),
                        train$Potability,alpha = 1,
                       family = "binomial", type.measure = "class")
```

```
plot(lasso_cv_i)
```




```
cat("The value for the minimum lambda is ", lasso_cv_i$lambda.min)
```

```
## The value for the minimum lambda is  0.0007682011
```

```
pred_lasso_i<- predict(lasso_cv_i, as.matrix(test[,1:8]), type = "class",
                       s = lasso_cv_i$lambda.min)

confusion_matrix <- table(test$Potability, pred_lasso_i)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- t(confusion_matrix)
```

```r
# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for Lasso in Original Train Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 14: Confusion Matrix for Lasso in Original Train Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	28	8	36
Pred. Potable	35	202	237
Total	63	210	273

```r
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.56
```

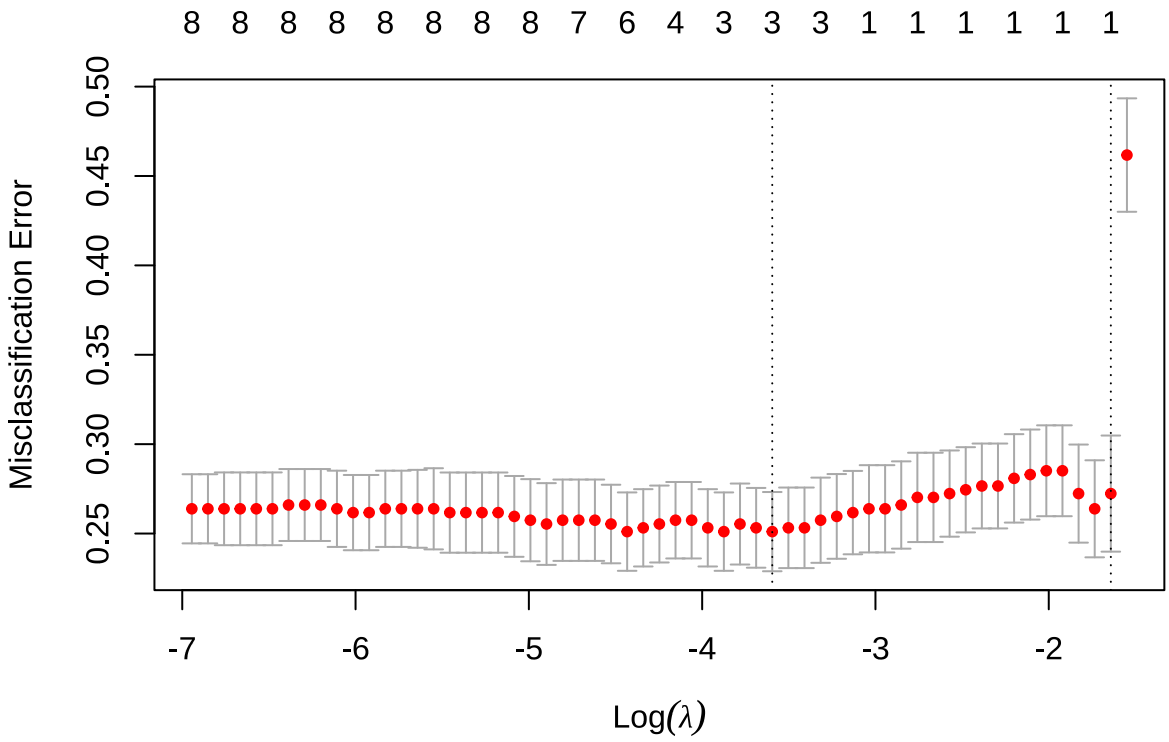
```r
#Accuracy
acc_lasso_i <- round(mean(pred_lasso_i == test$Potability),2)
cat("Accuracy:", acc_lasso_i, "\n")
```

```
## Accuracy: 0.84
```

Downsampled Set

```r
# Cross Validation for lambda values
down_lasso_cv <- cv.glmnet(as.matrix(downsampled[,1:8]),
                      downsampled$Potability,alpha = 1,
                      family = "binomial", type.measure = "class")
```

```r
plot(down_lasso_cv)
```




```
cat("The value for the minimum lambda is ", down_lasso_cv$lambda.min)
```

```
## The value for the minimum lambda is  0.02744649
```

```
down_pred_lasso<- predict(down_lasso_cv, as.matrix(test[,1:8]), type = "class",
                      s = down_lasso_cv$lambda.min)

confusion_matrix <- table(test$Potability, down_pred_lasso)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- t(confusion_matrix)

# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for Lasso in Downsampled Set") %>%
```

```
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 15: Confusion Matrix for Lasso in Downsampled Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	51	53	104
Pred. Potable	12	157	169
Total	63	210	273

```
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.19
```

```
#Accuracy
down_acc_lasso <- round(mean(down_pred_lasso == test$Potability),2)
cat("Accuracy:", down_acc_lasso, "\n")
```

```
## Accuracy: 0.76
```

Comparison The Lasso model trained over the original Train set scores a better accuracy value (0.84 vs 0.76). Yet, it has a false positive rate of 0.56 while the model trained over the downsampled set achieves a far better result with 0.19. Because of this, the second model represents the better choiche among the two.

Discriminant Analysis

Discriminant Analysis techniques require that the predictors are normally distributed. To assess the validity of this assumption, the Shapiro-Wilk test was performed. This test is commonly used to test statistical hypotheses, particularly the null hypothesis that a population is normally distributed. If the p-value is less than or equal to 0.05, the null hypothesis of normality is rejected, indicating that the data does not follow a normal distribution with 95% confidence. The test is applied to each covariate in both the original dataset and the downsampled one. As can be seen in the following cells, all the variables in both sets pass the test, resulting non-normally distributed, so Discriminant Analysis models will be performed keeping in mind that the Normality assumption is violated.

```
# Testing in the original Train set

shapiro.test(train$Temperature[train$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Temperature[train$Potability == 0]
## W = 0.95876, p-value = 2.796e-06
```

```
shapiro.test(train$BOD[train$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$BOD[train$Potability == 0]
## W = 0.91213, p-value = 1.531e-10
```

```
shapiro.test(train$pH[train$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$pH[train$Potability == 0]
## W = 0.67151, p-value < 2.2e-16
```

```
shapiro.test(train$Conductivity[train$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Conductivity[train$Potability == 0]
## W = 0.8911, p-value = 5.478e-12
```

```
shapiro.test(train$BOD[train$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$BOD[train$Potability == 0]
## W = 0.91213, p-value = 1.531e-10
```

```
shapiro.test(train$Nitrate[train$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Nitrate[train$Potability == 0]
## W = 0.8099, p-value = 3.092e-16
```

```
shapiro.test(train$Fecalcoliform[train$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Fecalcoliform[train$Potability == 0]
## W = 0.64509, p-value < 2.2e-16
```

```
shapiro.test(train$Totalcoliform[train$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Totalcoliform[train$Potability == 0]
## W = 0.59699, p-value < 2.2e-16
```

```
shapiro.test(train$Temperature[train$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Temperature[train$Potability == 1]
## W = 0.96733, p-value = 6.823e-13
```

```
shapiro.test(train$BOD[train$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$BOD[train$Potability == 1]
## W = 0.89886, p-value < 2.2e-16
```

```
shapiro.test(train$pH[train$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$pH[train$Potability == 1]
## W = 0.91639, p-value < 2.2e-16
```

```
shapiro.test(train$Conductivity[train$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Conductivity[train$Potability == 1]
## W = 0.86905, p-value < 2.2e-16
```

```
shapiro.test(train$BOD[train$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$BOD[train$Potability == 1]
## W = 0.89886, p-value < 2.2e-16
```

```r
shapiro.test(train$Nitrate[train$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Nitrate[train$Potability == 1]
## W = 0.79002, p-value < 2.2e-16
```

```r
shapiro.test(train$Fecalcoliform[train$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Fecalcoliform[train$Potability == 1]
## W = 0.62886, p-value < 2.2e-16
```

```r
shapiro.test(train$Totalcoliform[train$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train$Totalcoliform[train$Potability == 1]
## W = 0.62486, p-value < 2.2e-16
```

```r
# Testing in the downsampled set

shapiro.test(downsampled$Temperature[downsampled$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Temperature[downsampled$Potability == 0]
## W = 0.95876, p-value = 2.796e-06
```

```r
shapiro.test(downsampled$BOD[downsampled$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$BOD[downsampled$Potability == 0]
## W = 0.91213, p-value = 1.531e-10
```

```r
shapiro.test(downsampled$pH[downsampled$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$pH[downsampled$Potability == 0]
## W = 0.67151, p-value < 2.2e-16
```

```r
shapiro.test(downsampled$Conductivity[downsampled$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Conductivity[downsampled$Potability == 0]
## W = 0.8911, p-value = 5.478e-12
```

```r
shapiro.test(downsampled$BOD[downsampled$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$BOD[downsampled$Potability == 0]
## W = 0.91213, p-value = 1.531e-10
```

```r
shapiro.test(downsampled$Nitrate[downsampled$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Nitrate[downsampled$Potability == 0]
## W = 0.8099, p-value = 3.092e-16
```

```r
shapiro.test(downsampled$Fecalcoliform[downsampled$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Fecalcoliform[downsampled$Potability == 0]
## W = 0.64509, p-value < 2.2e-16
```

```r
shapiro.test(downsampled$Totalcoliform[downsampled$Potability==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Totalcoliform[downsampled$Potability == 0]
## W = 0.59699, p-value < 2.2e-16
```

```r
shapiro.test(downsampled$Temperature[downsampled$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Temperature[downsampled$Potability == 1]
## W = 0.97595, p-value = 0.0005008
```

```r
shapiro.test(downsampled$BOD[downsampled$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$BOD[downsampled$Potability == 1]
## W = 0.91026, p-value = 1.115e-10
```

```r
shapiro.test(downsampled$pH[downsampled$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$pH[downsampled$Potability == 1]
## W = 0.90666, p-value = 6.146e-11
```

```r
shapiro.test(downsampled$Conductivity[downsampled$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Conductivity[downsampled$Potability == 1]
## W = 0.85096, p-value = 2.715e-14
```

```r
shapiro.test(downsampled$BOD[downsampled$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$BOD[downsampled$Potability == 1]
## W = 0.91026, p-value = 1.115e-10
```

```r
shapiro.test(downsampled$Nitrate[downsampled$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Nitrate[downsampled$Potability == 1]
## W = 0.80913, p-value = 2.862e-16
```

```r
shapiro.test(downsampled$Fecalcoliform[downsampled$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Fecalcoliform[downsampled$Potability == 1]
## W = 0.59927, p-value < 2.2e-16
```

```
shapiro.test(downsampled$Totalcoliform[downsampled$Potability==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  downsampled$Totalcoliform[downsampled$Potability == 1]
## W = 0.62119, p-value < 2.2e-16
```

Linear Discriminant Analysis (LDA)

The LDA uses a linear decision boundary and assumes that the covariance matrix is the same for the two classes. The label assignment is done using a threshold of 0.5.

Original Train Set

```
lda_i <- lda(Potability~., data=train)
fit.lda_i <- predict(lda_i, newdata=test,type = "response")
```

```
post.lda_i <- fit.lda_i$posterior[,2]
pred.lda_i<- (ifelse(post.lda_i > 0.5, 1, 0))

confusion_matrix <- table(pred.lda_i,test$Potability)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- confusion_matrix

# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for LDA in Original Train Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 16: Confusion Matrix for LDA in Original Train Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	15	14	29
Pred. Potable	48	196	244
Total	63	210	273

```r
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.76
```

```r
#Accuracy
acc_lda_i <- round(mean(pred.lda_i == test$Potability),2)
cat("Accuracy:", acc_lda_i, "\n")
```

```
## Accuracy: 0.77
```

Downsampled Set

```r
down_lda1 <- lda(Potability~., data=downsampled)
down_fit.lda1 <- predict(down_lda1, newdata=test,
                         threshold=0.5,type = "response")
```

```r
down_post.lda1 <- down_fit.lda1$posterior[,2]
down_pred.lda<- (ifelse(down_post.lda1 > 0.5, 1, 0))

confusion_matrix <- table(down_pred.lda,test$Potability)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- confusion_matrix

# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for LDA in Downsampled Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

```r
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

Table 17: Confusion Matrix for LDA in Downsampled Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	42	40	82
Pred. Potable	21	170	191
Total	63	210	273

```
## False Positive rate: 0.33
```

```
#Accuracy
down_acc_lda <- round(mean(down_pred.lda == test$Potability),2)
cat("Accuracy:", down_acc_lda, "\n")
```

```
## Accuracy: 0.78
```

Comparison The model trained over the the downsampled set outperforms the one trained over the original Train set both in terms of accuracy (0.78 vs 0.77) and number of false positives (0.33 vs 0.76).

Quadratic Discriminant Analysis (QDA)

The QDA uses a non-linear, quadratic decision boundary, which is supposed to perform better in case of non-linearly separable data. The label assignment is done using a threshold of 0.6.

Original Train Set

```
qda_i <- qda(Potability~., data=train)
fit.qda_i <- predict(qda_i, newdata=test,type = "response")
```

```
post.qda_i <- fit.qda_i$posterior[,2]
pred.qda_i<- (ifelse(post.qda_i > 0.6, 1, 0))

confusion_matrix <- table(pred.qda_i,test$Potability)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- confusion_matrix

# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

kable(confusion_df, format = "latex", booktabs = TRUE,
      caption = "Confusion Matrix for QDA in Original Train Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 18: Confusion Matrix for QDA in Original Train Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	26	28	54
Pred. Potable	37	182	219
Total	63	210	273

```
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.59
```

```
#Accuracy
acc_qda_i <- round(mean(pred.qda_i == test$Potability),2)
cat("Accuracy:", acc_qda_i, "\n")
```

```
## Accuracy: 0.76
```

Downsampled Set

```
down_qda1 <- qda(Potability~., data=downsampled)
down_fit.qda1 <- predict(down_qda1, newdata=test,type = "response")
```

```
down_post.qda1 <- down_fit.qda1$posterior[,2]
down_pred.qda<- (ifelse(down_post.qda1 > 0.6, 1, 0))

confusion_matrix <- table(down_pred.qda,test$Potability)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- confusion_matrix

# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))


kable(confusion_df, format = "latex",
      booktabs = TRUE,
      caption = "Confusion Matrix for QDA in Downsampled Set") %>%
```

```
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 19: Confusion Matrix for QDA in Downsampled Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	38	52	90
Pred. Potable	25	158	183
Total	63	210	273

```
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.4
```

```
#Accuracy
down_acc_qda <- round(mean(down_pred.qda == test$Potability),2)
cat("Accuracy:", down_acc_qda, "\n")
```

```
## Accuracy: 0.72
```

Comparison The model trained over the the original Train set scores a better result in terms of accuracy (0.76 vs 0.72). Yet, it has a higher ratio of false positives (0.59 vs 0.4).

K-Nearest Neighbour (KNN)

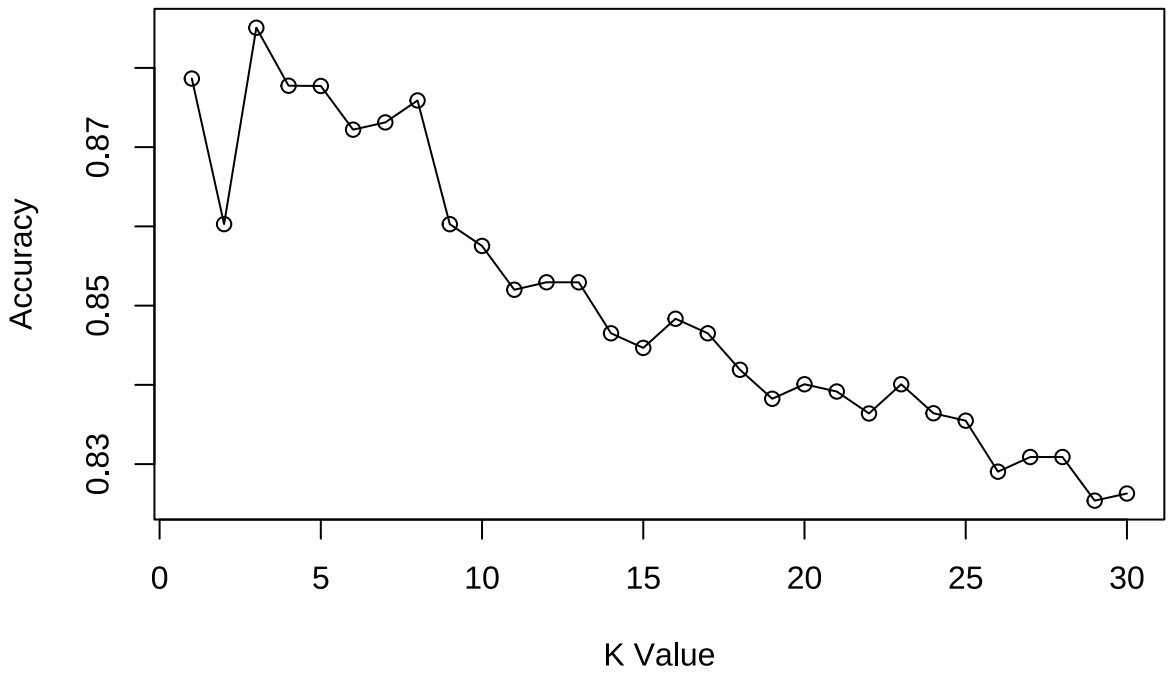
KNN is a non parametric algorithm used for classification, which assigns the most frequent category of the k-closest data points with respect to the considered unlabeled example. Unlike the previously defined parametric models, it doesn't make assumption on the data distribution.

In the following implementations, the Euclidean Distance is used as distance measure and the number of neighbors to consider is selected with the use of 10-fold cross validation, which will select the k for which the accuracy is higher. To avoid biased results in the distances due to the different scale of each predictor, a scaling is also applied.

Original Train Set

```
k_values <- 1:30
tuneGrid <- expand.grid(k = k_values)
ctrl <- trainControl(method = "cv", number = 10)

# Training the KNN model with 10-fold cross-validation, using
# scaled data
knn_cv_model <- train(x = as.data.frame(scale(train[, 1:8])),
                      y = train$Potability,
```

```
                      method = "knn",
                      tuneGrid = tuneGrid,
                      trControl = ctrl)

# Plot accuracy vs k values
plot(knn_cv_model$results$k,
     knn_cv_model$results$Accuracy,
     type="o", xlab = "K Value", ylab = "Accuracy")
```




```
#Best K value extraction
best_k <- knn_cv_model$bestTune$k


# Final model performed using the best K value
knn_pred_i <- knn(as.data.frame(scale(train[, 1:8])), as.data.frame(scale(test[, 1:8])), train$Potabili

confusion_matrix <- table(knn_pred_i, test$Potability)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")

# Fill the confusion matrix with values
confusion_df[,1:2] <- confusion_matrix
```

```r
# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))


kable(confusion_df, format = "latex",
      booktabs = TRUE,
      caption = "Confusion Matrix for KNN in Original Train Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 20: Confusion Matrix for KNN in Original Train Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	34	10	44
Pred. Potable	29	200	229
Total	63	210	273

```r
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.46
```

```r
# Accuracy
acc_knn_i <- round(mean(knn_pred_i == test$Potability),2)

# Print the accuracy and the best K value
cat("Best K value:", best_k, "\n")
```

```
## Best K value: 3
```

```r
cat("Accuracy:", acc_knn_i, "\n")
```

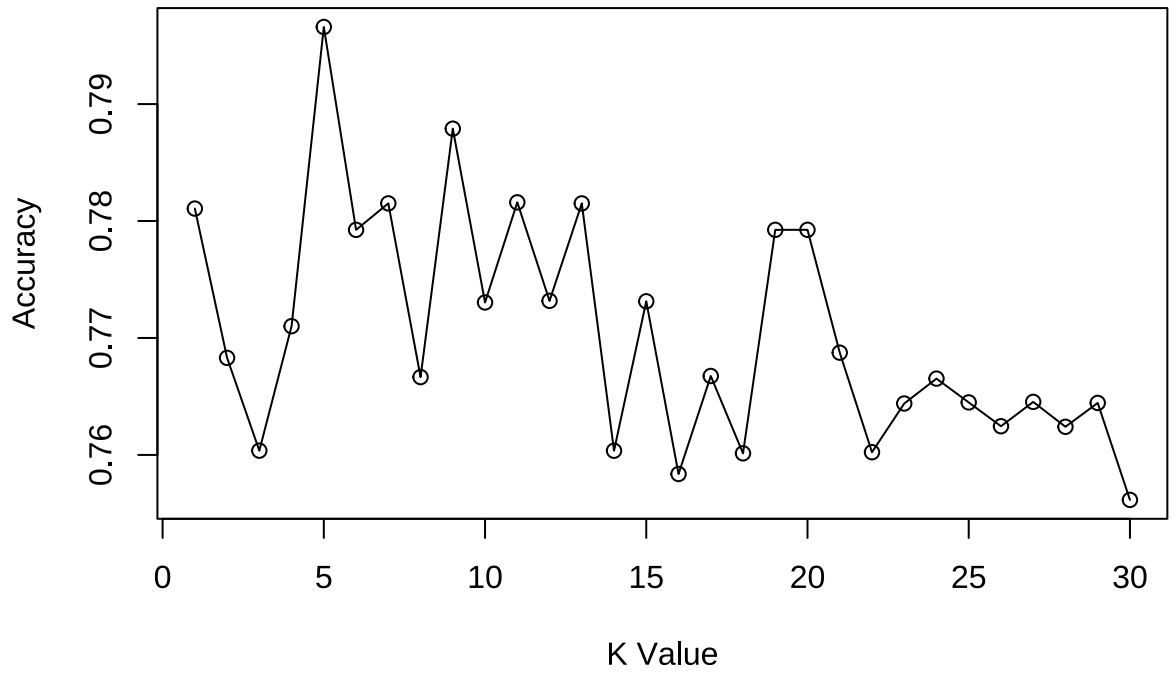
```
## Accuracy: 0.86
```

The CV has selected K=3 which has lead to an accuracy of 85.71%.

Downsampled Set

```r
k_values <- 1:30
tuneGrid <- expand.grid(k = k_values)
ctrl <- trainControl(method = "cv", number = 10)
```

```r
# Training the KNN model with 10-fold cross-validation, using
# scaled data
down_knn_cv_model <- train(x = as.data.frame(scale(downsampled[, 1:8])),
                       y = downsampled$Potability,
                       method = "knn",
                       tuneGrid = tuneGrid,
                       trControl = ctrl)

# Plot accuracy vs k values
plot(down_knn_cv_model$results$k,
     down_knn_cv_model$results$Accuracy,type="o", xlab = "K Value", ylab = "Accuracy")
```




```r
#Best K value extraction
down_best_k <- down_knn_cv_model$bestTune$k


# Final model performed using the best K value
down_knn_pred <- knn(as.data.frame(scale(downsampled[, 1:8])), as.data.frame(scale(test[, 1:8])), downsa

confusion_matrix <- table(down_knn_pred, test$Potability)

# Format confusion matrix as a data frame with 2 columns
confusion_df <- as.data.frame(matrix(0, nrow = 2, ncol = 2))
colnames(confusion_df) <- c("True Non-Potable", "True Potable")
rownames(confusion_df) <- c("Pred. Non-Potable", "Pred. Potable")
```

```r
# Fill the confusion matrix with values
confusion_df[,1:2] <- confusion_matrix

# Add a total row to the data frame
confusion_df["Total",] <- colSums(confusion_df)

# Add a total column to the data frame
confusion_df <- cbind(confusion_df, Total = rowSums(confusion_df))

kable(confusion_df, format = "latex",
      booktabs = TRUE, caption = "Confusion Matrix for KNN in Downsampled Set") %>%
  kable_styling(latex_options = "hold_position")%>%
column_spec(c(1,3), border_right = TRUE)
```

Table 21: Confusion Matrix for KNN in Downsampled Set

	True Non-Potable	True Potable	Total
Pred. Non-Potable	56	53	109
Pred. Potable	7	157	164
Total	63	210	273

```r
# Extract the confusion matrix elements for the Non-Potable class
non_potable_confusion <- confusion_df[,1]

# Compute the fraction of Non-Potable samples predicted as Potable
non_potable_predicted_as_potable <- round((non_potable_confusion[2] /non_potable_confusion[3]),2)
cat("False Positive rate:", non_potable_predicted_as_potable, "\n")
```

```
## False Positive rate: 0.11
```

```r
# Print the accuracy and the best K value
cat("Best K value:", down_best_k, "\n")
```

```
## Best K value: 5
```

```r
# Accuracy computation
down_acc_knn <- round(mean(down_knn_pred == test$Potability),2)
cat("Accuracy:", down_acc_knn, "\n")
```

```
## Accuracy: 0.78
```

For the downsampled set, K=5 is selected since it is a sufficiently good value for the bias-variance trade-off, with an accuracy of 78%. There are less misclassified labels compared to the Original Train Set.

Comparison For the two estimated KNN non-parametric models, the best Accuracy is achieved by the one fitted on the Original Training Set (85.71%). However, it's worth mentioning the 29 misclassified Non-Potable labels on a total of 63 that represents 46% wrong classification of Non-Potable labels. Furthermore, the K-NN is not able to tell which predictors are significantly influencing the probability of being Potable and to quantify this influence, so it will not be considered in the final models comparison.

Models comparison and conclusions

Accuracy

It is an indicator which measures the proportion of correctly classified instances out of the total number of instances, which is

$$Accuracy = \frac{TruePositives+TrueNegatives}{TruePositives+FalsePositives+TrueNegatives+FalseNegatives}$$

Original Train Set

```
table_data <- data.frame(
  Model = c("Logistic Regression", "Lasso", "Ridge", "LDA", "QDA", "KNN"),
  Accuracy = round(c(acc_lr_i, acc_lasso_i, acc_ridge_i, acc_lda_i, acc_qda_i, acc_knn_i), digits=3)
)

kable(table_data, format = "latex", booktabs = T,
      caption = "Model Accuracies for Original Train Set") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 22: Model Accuracies for Original Train Set

Model	Accuracy
Logistic Regression	0.87
Lasso	0.84
Ridge	0.78
LDA	0.77
QDA	0.76
KNN	0.86

Downsampled Set

```
table_data_2 <- data.frame(
  Model = c("Logistic Regression", "Lasso", "Ridge", "LDA", "QDA", "KNN"),
  Accuracy = round(c(down_acc_lr, down_acc_lasso, down_acc_ridge, down_acc_lda, down_acc_qda, down_acc_
)

kable(table_data_2, format = "latex", booktabs = T,
      caption = "Model Accuracies for Downsampled Set") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 23: Model Accuracies for Downsampled Set

Model	Accuracy
Logistic Regression	0.77
Lasso	0.76
Ridge	0.80
LDA	0.78
QDA	0.72
KNN	0.78

ROC Curves

Receiver Characteristic Curve (ROC) is used to visualize the trade-off between:

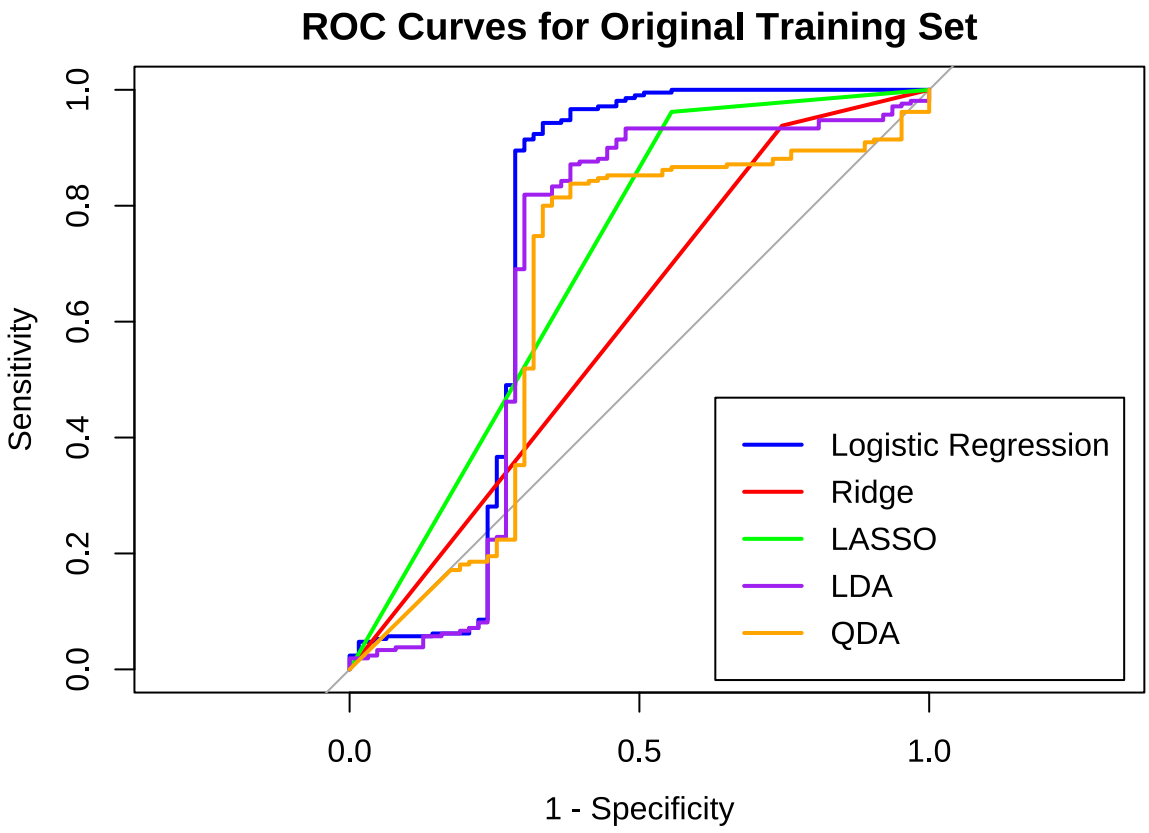
- True Positive Rate (proportion of Potable samples that are correctly identified as Potables);
- False Positive Rate (proportion of Non-Potable samples that are incorrectly identified as Potable).

It is a good tool to compare different models which have performed the same classification task, and the aim of each ROC curves is to show how the TPR and FPR change for classifying a sample as Potable using different threshold values.

The AUC is the area under a ROC curve. It ranges from 0 to 1, with higher values indicating better performance for the considered model.

Original Train Set

```
roc_lr_i <- roc(test$Potability, fit_lr_i)
roc_ridge_i <- roc(test$Potability, as.numeric(pred_ridge_i))
roc_lasso_i <- roc(test$Potability, as.numeric(pred_lasso_i))
roc_lda_i <- roc(test$Potability, post.lda_i)
roc_qda_i <- roc(test$Potability, post.qda_i)

# Plot the ROC curves for each model on the same graph
plot(roc_lr_i, col = "blue", main = "ROC Curves for Original Training Set",
     print.auc = FALSE, legacy.axes = TRUE)
lines(roc_ridge_i, col = "red")
lines(roc_lasso_i, col = "green")
lines(roc_lda_i, col = "purple")
lines(roc_qda_i, col = "orange")


# Add a legend to the graph
legend("bottomright", legend = c("Logistic Regression", "Ridge",
                                 "LASSO", "LDA", "QDA"),
       col = c("blue", "red", "green", "purple", "orange"),
       lwd = 2, inset = c(0.02, 0.02))
```




```
auc_data_1 <- data.frame(
  Model = c("Logistic Regression", "Lasso", "Ridge", "LDA", "QDA"),
  AUC = round(c(roc_lr_i$auc, roc_lasso_i$auc,roc_ridge_i$auc, roc_lda_i$auc, roc_qda_i$auc),digits=3)
)

kable(auc_data_1, format = "latex",
      booktabs = T, caption = "AUC for Original Train Set models") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

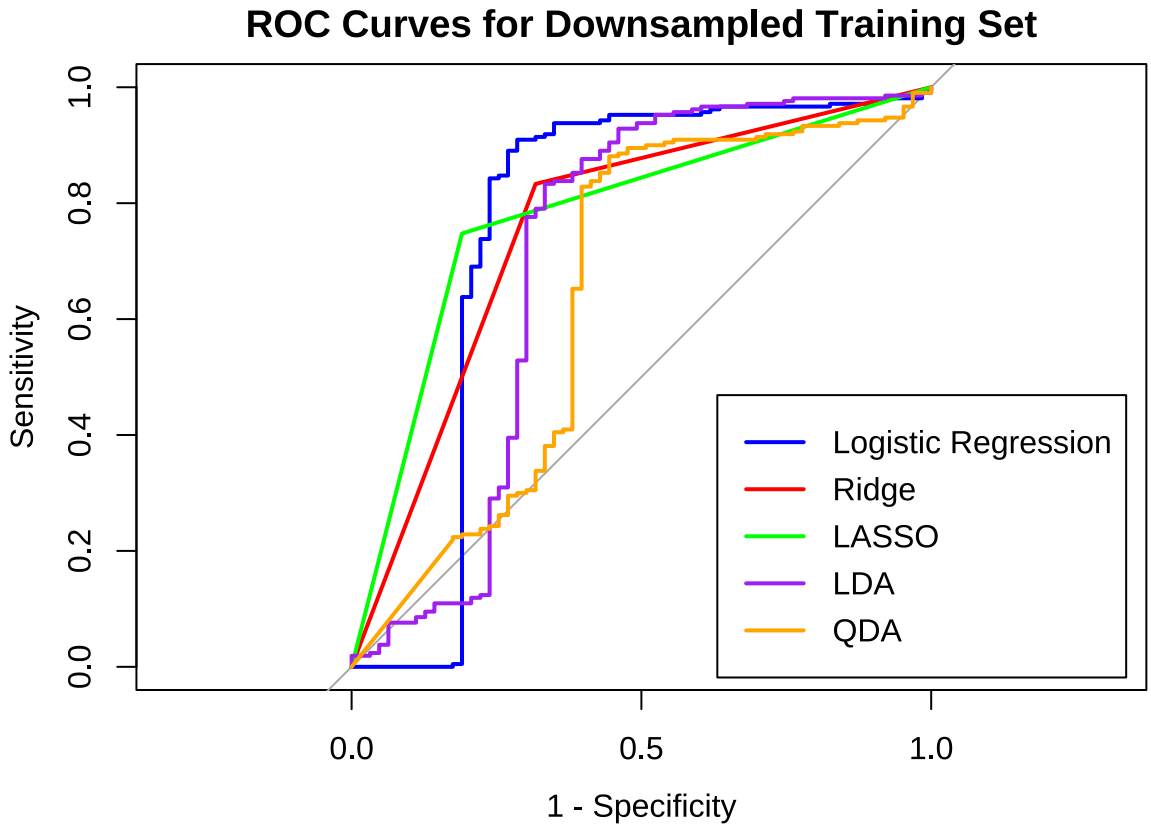
Table 24: AUC for Original Train Set models

Model	AUC
Logistic Regression	0.736
Lasso	0.703
Ridge	0.596
LDA	0.680
QDA	0.646

Downsampled set

```
down_roc_lr <- roc(test$Potability, down_fit_lr)
down_roc_ridge <- roc(test$Potability, as.numeric(down_pred_ridge))
down_roc_lasso <- roc(test$Potability, as.numeric(down_pred_lasso))
```

```r
down_roc_lda <- roc(test$Potability, down_post.lda1)
down_roc_qda <- roc(test$Potability, down_post.qda1)

# Plot the ROC curves for each model on the same graph
plot(down_roc_lr, col = "blue",
     main = "ROC Curves for Downsampled Training Set",
     print.auc = FALSE, legacy.axes = TRUE)
lines(down_roc_ridge, col = "red")
lines(down_roc_lasso, col = "green")
lines(down_roc_lda, col = "purple")
lines(down_roc_qda, col = "orange")


# Add a legend to the graph
legend("bottomright", legend = c("Logistic Regression", "Ridge",
                                 "LASSO", "LDA", "QDA"),
       col = c("blue", "red", "green", "purple", "orange"),
       lwd = 2, inset = c(0.02, 0.02))
```




```r
auc_data_2 <- data.frame(
  Model = c("Logistic Regression", "Lasso", "Ridge", "LDA", "QDA"),
  AUC = round(c(down_roc_lr$auc, down_roc_lasso$auc,down_roc_ridge$auc, down_roc_lda$auc, down_roc_qda$a
)

kable(auc_data_2, format = "latex", booktabs = T,
```

```
      caption = "AUC for Downsampled Set models") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 25: AUC for Downsampled Set models

Model	AUC
Logistic Regression	0.758
Lasso	0.779
Ridge	0.758
LDA	0.701
QDA	0.644

Conclusions

Even if the accuracy resulted higher in some of the parametric models fitted on the Original Training Set, in terms of AUC and ROC Curve the models fitted on the downsampled set are all better than the ones fitted on the Original Training set, confirming the necessity of using a downsampled set. This conclusion is made since AUC is robust to different thresholds and can also take in consideration how each model performed on False Positive and False Negative Values which are actually fundamental for this context, resulting in a better indicator to consider than Accuracy.

It can then be affirmed that the best parametric model is the Lasso Logistic Regression model fitted on the downsampled set, with an AUC of 0.779 and an accuracy of 0.762. In the upcoming subsection, the results obtained on this model will be discussed and final considerations will be addressed.

Results Interpretation

```
final_model <- glmnet(as.matrix(train[,1:8]), train$Potability,
                      alpha = 1, family = "binomial", lambda = down_lasso_cv$lambda.min)
beta_values <- coef(final_model)
beta_values <- as.vector(beta_values)
```

The coefficients for the predictors selected by the Lasso penalization are

```
variables <- c("Intercept",names(downsampled[,1:8]))


coefficients_df <- data.frame(Variables = variables, Coefficients = round(beta_values, digits=3))

# Summary Table for the model
kable(coefficients_df, "latex", booktabs = T,
      caption = "Coefficients Table") %>%
  kable_styling(latex_options = c("striped", "hold_position")) %>%
  column_spec(1:2)
```

Since the Lasso Penalization has given some coefficients equal to 0, the results can be simplified and the model is interpreted as:

Table 26: Coefficients Table

Variables	Coefficients
Intercept	24.196
Temperature	-0.017
DO	0.000
pH	-2.888
Conductivity	0.000
BOD	0.000
Nitrate	0.000
Fecalcoliform	0.000
Totalcoliform	0.000

$$log(\frac{\hat{\pi}(x)}{1-\hat{\pi}(x)}) = 24.196 - 0.017 * Temperature - 2.888 * pH$$

Considering the definition of odds as the probability of being Potable divided by the probability of being Non-potable

$$\frac{\hat{\pi}(x)}{1-\hat{\pi}(x)}$$

- The intercept term (24.196) represents the estimated **log-odds** of Potable water when both Temperature and pH are zero. In that case, the probability of being Potable would be exactly 1;
- The coefficient for Temperature (-0.017) indicates that, on average, when the Temperature is increased by one unit there is a decrease in the **log-odds** of Potable water, for a fixed value of pH. In terms of odds ratios, increasing the Temperature by one unity implies that the odds of water being potable decreases $e^{-0.017} = 0.983$ times.
- The coefficient for pH (-2.888) suggests that, on average, when the pH is increased by one unit there is decrease in the **log-odds** of water being potable, for a fixed value of Temperature. In terms of odds ratios, increasing by one unity the pH implies that the odds of water being potable decreases by $e^{-2.888} = 0.056$ times.