Statistical Learning Project

Mazza Davide (2089074), Mancosu Bustos Davide Christian (2089208), Hamdar Karim (2092041)

2023-07-25

Introduction

Problem presentation

Credit risk and corporate bankruptcy are two of the most relevant topics in the financial and business environment. In general corporate bankruptcy occurs when a company is unable to meet its financial obligations and can no longer sustain its business operations, it is a state of financial insolvency where the company's debts exceed its available assets for paying creditors. In particular, business failure represents one of the biggest risks that businesses may face during their operations (Brealey, Myers & Allen, 2021). The bankruptcy of a company, in addition to generating financial losses for creditors due to insolvency, can generate a series of ripple effects that have repercussions on a social, economic, and financial level, such as unemployment. Depending on the size and turnover of the bankrupt company, it can also lead to instability in the entire economic system. The ability to predict and assess these risks has become increasingly essential for all economic players operating in the market, such as financial institutions, investors, and stakeholders. In recent years, the use of statistical models and machine learning algorithms has opened up new possibilities for financial analysts and market analysts to predict bankruptcy.

Our project deals with trying to predict the possible bankruptcy of a company through statistical models starting from the indices created with the data of its balance.

Dataset presentation

The Compustat dataset, a renowned and widely utilized source of financial information for companies, is not freely accessible. It is a commercial database that requires a subscription or institutional access to obtain the data. Maintained and provided by S&P Global, a leading provider of financial market intelligence, Compustat offers comprehensive coverage of publicly traded companies in the United States and Canada. Its extensive collection encompasses essential financial variables, including balance sheet data, income statements, cash flow statements, and a wide array of ratios and metrics crucial for financial analysis. Access to the dataset is typically obtained through paid subscriptions, making it a valuable resource for financial institutions, academic researchers, and professionals in the finance industry. Its rich and comprehensive nature enables in-depth analysis, modeling, and research related to various aspects of company financial performance. It can be found at the following web page: Compustat Database.

The bankruptcy dataset we used has 5436 observations with 13 variables each. Each observation represents a company, and of these 13 variables, 9 are accounting variables calculated from the balance sheet of the company in question, and 1 is a market variable. The 9 financial variables, which are indices calculated with balance sheet items commonly called ratios, are calculated as follows:

R1: WC/TA (Working Capital/Total Assets) - This continuous ratio measures the ratio between working capital, which represents the current assets minus current liabilities, and the total assets of a company. It indicates how much the company's current assets contribute to its total assets.

R2: RE/TA (Retained Earnings/Total Assets) - This continuous ratio represents the proportion of retained earnings, which are the accumulated profits that have not been distributed to shareholders, in

relation to the total assets of a company. It provides information on how much the total assets of the company are financed by its retained earnings.

R3: EBIT/TA (Earnings Before Interest and Tax/Total Assets) - This continuous ratio shows the relation between earnings before interest and taxes, which represents the company's operating profits, and its total assets. It indicates the level of efficiency of the company's asset utilization in generating operating income, demonstrating how much company-owned assets are capable of generating income.

R4: ME/TL (Market Value of Equity/Total Liability) - This continuous ratio compares the market value of a company's equity (the total value of its outstanding/circulating shares) to its total liability. It shows an indication of how much the market value of the equity of a company covers its total liabilities.

R5: S/TA (Sales/Total Assets) - This continuous ratio measures the proportion of a company's sales revenue to its total assets. It underlines the company's ability to generate sales compared to its assets.

R6: TL/TA (Total Liability/Total Assets) - This continuous ratio represents the ratio between the company's total liabilities to its total assets. It indicates the proportion between the company's assets and its liabilities, giving information on how the assets are financed by the liabilities.

R7: CA/CL (Current Assets/Current Liability) - This continuous ratio compares a company's current assets (the part of the assets that can be easily converted into cash within a year) to its current liabilities (obligations due to be refunded by the company within a year). It helps assess the liquidity of a company and its financial balance in the short term.

R8: NI/TA (Net Income/Total Assets) - This continuous ratio shows the proportion of a company's net income (income after removing all expenses and taxes) to its total assets. It's another way, alternative to the R3 ratio, to see the company's profitability compared to its asset base.

R9: Bankruptcy cost, Log(sales) - This continuous variable represents the bankruptcy cost, which is calculated as the logarithm of the company's sales. It serves as an indicator of the financial impact or consequences associated with bankruptcy.

R10: Market capitalization, log(abs(price)*numbers of shares outstanding/1000) - This continuous variable represents the market capitalization of a company, which is calculated as the logarithm of the absolute value of the price per share multiplied by the number of shares outstanding, divided by 1000. Market capitalization reflects the total value of a company's outstanding shares in the market.

CUSIP (Committee on Uniform Securities Identification Procedures) - This code is a unique nine-character alphanumeric identifier assigned to securities traded in the United States and Canada. It helps identify and track individual securities, facilitating efficient trading, settlement, and record-keeping processes in the financial industry.

DLRSN - This variable represents our target variable that we have to predict; it's a binary value (0 = No Bankruptcy; 1 = Bankruptcy).

FYEAR - This categorical variable represents the year in which the index values were collected for each observation.

Libraries

```
library(foreign)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(ggplot2)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-7
```

```
library(correlation)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

Exploratory Data Analysis

We can start our analysis loading the Bankruptcy dataset.

```
Bankruptcy <- read.csv("Bankruptcy.csv")

# Sample data: first six rows
head(Bankruptcy)
```

```
##   FYEAR DLRSN    CUSIP         R1        R2        R3          R4         R5
## 1  1999     0 00036020  0.3071395 0.8870057 1.6476808 -0.19915760  1.0929640
## 2  1999     0 00036110  0.7607365 0.5924934 0.4530028 -0.36989014  0.1861539
## 3  1999     0 00037520 -0.5135961 0.3376148 0.2990145 -0.02907996 -0.4326047
## 4  1994     1 00078110 -0.4661293 0.3707467 0.4960667 -0.37342897 -0.2674240
## 5  1999     0 00079X10  2.0234223 0.2148764 0.1825948  6.69536040 -1.1483381
## 6  1999     0 00086T10  0.9074985 0.3868797 0.4778914 -0.34715872  1.4079893
##            R6          R7        R8         R9         R10
## 1 -0.31328867 -0.19679332 1.2067628  0.2824709  0.15889625
## 2  0.03961907  0.32749732 0.4284181  1.1069652  0.79344276
## 3  0.82999324 -0.70778613 0.4761533  2.1791755  2.48458450
## 4  0.97779888 -0.61097507 0.4568098  0.1519511  0.04778851
## 5 -1.50588927  2.87647687 0.2873749 -0.9864421  0.79107673
## 6 -0.48390230  0.07025944 0.5278110  0.5024659 -0.16464802
```

Data Cleaning

As first step we checked for some missing values:

```
sum(is.na(Bankruptcy))
```

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

Since this is not the case, we could proceed with the analysis of the dataset. To make sure that we didn't have some redundant rows, we checked for repeating ID in the ID column.

```
sum(duplicated(Bankruptcy$CUSIP))
```

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

There are no multiple observations. Given that, we decided to remove the *CUSIP* column from the entire dataset as it's not helpful in our analysis.

```
Bank_clean <- Bankruptcy[, -c(1, 3)]
```

Based on the results obtained, we did not find any missing values (NA) in the dataset. This is a positive outcome as it allows us to proceed with our analysis without having to make significant changes to the dataset content. Therefore, we have a dataframe without any missing values. We can now continue our analysis by considering multicollinearity.

Data Exploration

In this section, our aim is to visually represent the distribution of the data. We will see how the data is distributed.

Structure of the data

To begin let's start with some general statistics of the dataset:

```
str(Bankruptcy)
```

```
## 'data.frame':    5436 obs. of  13 variables:
##  $ FYEAR: int  1999 1999 1999 1994 1999 1999 1999 1999 1987 1999 ...
##  $ DLRSN: int  0 0 0 1 0 0 0 0 1 0 ...
##  $ CUSIP: chr  "00036020" "00036110" "00037520" "00078110" ...
##  $ R1   : num  0.307 0.761 -0.514 -0.466 2.023 ...
##  $ R2   : num  0.887 0.592 0.338 0.371 0.215 ...
##  $ R3   : num  1.648 0.453 0.299 0.496 0.183 ...
##  $ R4   : num  -0.1992 -0.3699 -0.0291 -0.3734 6.6954 ...
##  $ R5   : num  1.093 0.186 -0.433 -0.267 -1.148 ...
##  $ R6   : num  -0.3133 0.0396 0.83 0.9778 -1.5059 ...
##  $ R7   : num  -0.197 0.327 -0.708 -0.611 2.876 ...
##  $ R8   : num  1.207 0.428 0.476 0.457 0.287 ...
##  $ R9   : num  0.282 1.107 2.179 0.152 -0.986 ...
##  $ R10  : num  0.1589 0.7934 2.4846 0.0478 0.7911 ...
```

```
dim(Bankruptcy)
```

```
## [1] 5436   13
```

```
Bank_clean$DLRSN=as.factor(Bank_clean$DLRSN)
summary(Bank_clean)
```

```
##  DLRSN          R1                R2                R3
##  0:4660   Min.   :-4.3828   Min.   :-2.2418   Min.   :-2.06423
##  1: 776   1st Qu.:-0.7501   1st Qu.:-1.0805   1st Qu.:-1.07887
##           Median :-0.2220   Median : 0.1337   Median : 0.06858
##           Mean   :-0.2352   Mean   :-0.2915   Mean   :-0.24411
##           3rd Qu.: 0.4821   3rd Qu.: 0.5103   3rd Qu.: 0.50070
##           Max.   : 2.0234   Max.   : 1.4854   Max.   : 2.14246
##        R4                R5                R6                R7
##  Min.   :-0.42712   Min.   :-1.3639   Min.   :-1.505889   Min.   :-1.23340
##  1st Qu.:-0.38752   1st Qu.:-0.8748   1st Qu.:-0.656827   1st Qu.:-0.77996
##  Median :-0.30754   Median :-0.3508   Median : 0.003493   Median :-0.43205
##  Mean   : 0.23903   Mean   :-0.1338   Mean   : 0.195707   Mean   :-0.09612
##  3rd Qu.: 0.02746   3rd Qu.: 0.2878   3rd Qu.: 0.608376   3rd Qu.: 0.17578
##  Max.   : 6.69536   Max.   : 4.0362   Max.   : 5.110424   Max.   : 2.87648
```

```
##        R8                R9                R10
##  Min.   :-2.2082   Min.   :-2.76356   Min.   :-2.2140
##  1st Qu.:-1.0054   1st Qu.:-0.66606   1st Qu.:-0.6413
##  Median : 0.2053   Median : 0.06219   Median : 0.1230
##  Mean   :-0.2272   Mean   : 0.02538   Mean   : 0.1806
##  3rd Qu.: 0.5289   3rd Qu.: 0.81215   3rd Qu.: 0.9878
##  Max.   : 2.0006   Max.   : 2.17918   Max.   : 2.4846
```

Distribution of bankruptcy - 0's and 1's in the data.

```
table(Bank_clean$DLRSN)
```

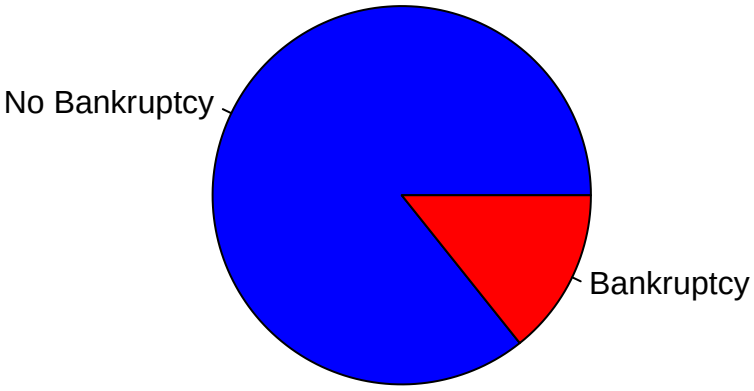
```
##
##    0    1
## 4660  776
```

```
table(Bank_clean$DLRSN)/length(Bank_clean$DLRSN)*100  # proportion of 0 and 1
```

```
##
##       0       1
## 85.7248 14.2752
```

We can observe a significant class imbalance in the binary response variable DLRSN. Specifically, there are 4660 cases (85.73%) classified as "no failure," and 776 cases (14.27%) classified as "failure." To obtain a clearer view and understand better this division, we have created a pie chart to visualize these proportions.

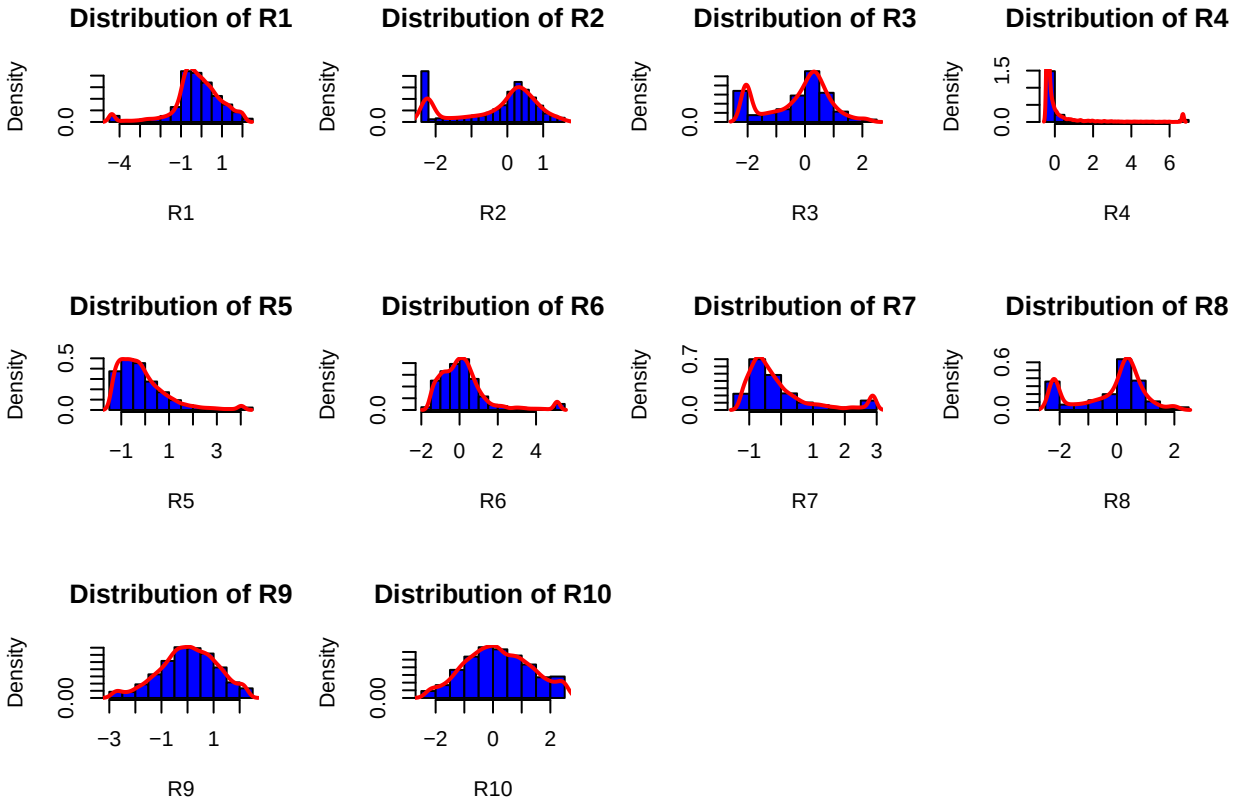
```
dati <- c(4660, 776)
labels <- c("No Bankruptcy", "Bankruptcy")

# Creazione del pie chart
pie(x = dati, labels = labels, col=c("blue", "red"))
```



Univariate Analysis

```
par(mfrow = c(3, 4))

for (i in 2:11) {
  hist(Bank_clean[, i], main = paste("Distribution of", colnames(Bank_clean[i])), xlab = colnames(Bank_
  dens <- density(Bank_clean[, i])
  lines(dens, col = "red", lwd = 2)
  4

  frequenze_relative <- table(Bank_clean[, i]) / length(Bank_clean[, i])
  freq_relative_ord <- frequenze_relative[as.character(unique(Bank_clean[, i]))]
```

```
}
```

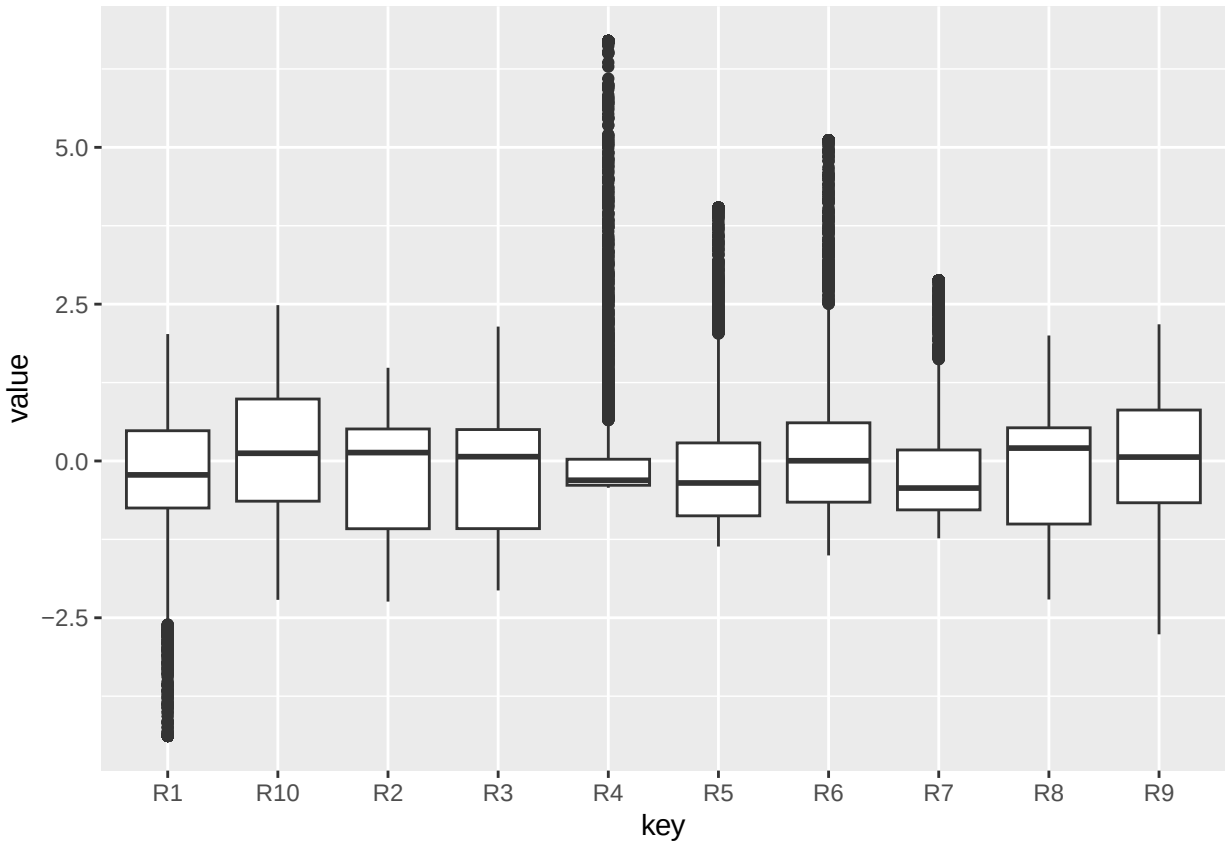


From the plots we can see that:

- **R4, R5, R6, R7** follow a right skewed distribution.
- **R1, R2** follow a left skewed distribution.
- **R10, R3, R8, and R9** follow an almost normal distribution with a few extreme values.

We make a boxplot to verify the outliers and extreme values.

```
Long <- Bank_clean[,-1] %>% gather()
ggplot(Long, aes(x = key, y = value)) + geom_boxplot()
```

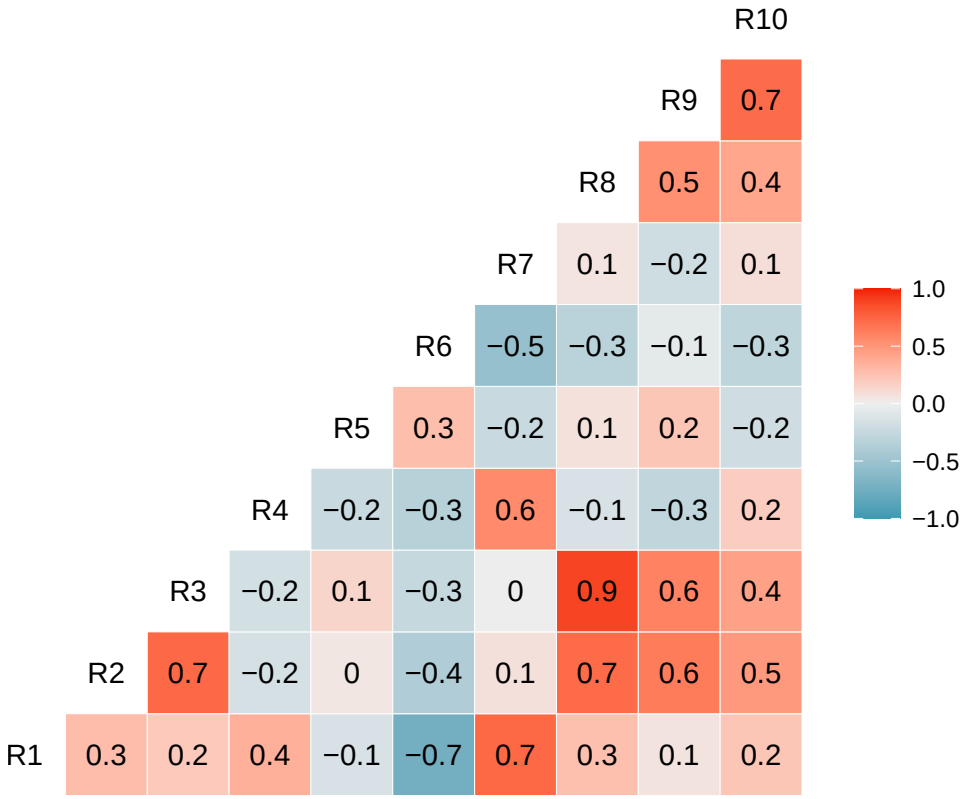


As also seen from the boxplot above, we observe outliers in the variable – R1, R4, R5, R6, R7. We can notice also that all the rates are centered over 0.

Bivariate Analysis

```
#Bank_clean$DLRSN <- as.numeric(Bank_clean$DLRSN)
ggcorr(Bank_clean, label=T)
```

```
## Warning in ggcorr(Bank_clean, label = T): data in column(s) 'DLRSN' are not
## numeric and were ignored
```



- Response Variable DLRSN is highly correlated with variable R10 followed by R8, R1, R6 and R3.

The variables R10 (Market capitalization), R8 (NI/TA), R1 (WC/TA), R3 (EBIT/TA), and R6 (TL/TA) are correlated with the likelihood of bankruptcy for a company. Market capitalization is an important indicator of a company's value, and a significant decrease in market capitalization is often associated with financial distress and a higher probability of bankruptcy. Additionally, financial ratios such as NI/TA, WC/TA, EBIT/TA, and TL/TA provide insights into profitability, liquidity, and leverage. A low NI/TA ratio indicates lower profitability relative to total assets and a potential inability to cover expenses, increasing the risk of bankruptcy. A negative WC/TA ratio suggests insufficient working capital to meet short-term obligations, elevating the likelihood of bankruptcy. A low EBIT/TA ratio signifies poor operational performance and reduced profitability, contributing to the risk of financial distress and bankruptcy. Lastly, a high TL/TA ratio reflects higher financial risk and potential challenges in servicing debt, increasing the risk of bankruptcy.

- R2 Retained Earnings/Total Asset and R3 Earnings Before Interest & Tax/Total Asset (correlation: 0.7):

  The positive correlation between these two ratios can be attributed to the fact that companies with higher earnings before interest and taxes (EBIT) relative to their total assets tend to have a larger amount of profits available to be retained within the company. This results in higher retained earnings relative to their total assets, as retained earnings represent the portion of profits that a company reinvests instead of distributing to shareholders.

- R3 Earnings Before Interest & Tax/Total Asset and R8 Net Income/Total Asset (correlation: 0.9):

  The strong positive correlation between EBIT/Total Asset and Net Income/Total Asset can be explained by the fact that both ratios are related to a company's profitability. EBIT represents the earnings generated by a company before accounting for interest and taxes, and net income represents the earnings after accounting for all expenses, including interest and taxes. Companies that perform well in terms of EBIT relative to their total assets are likely to continue performing well after considering all expenses, leading to a strong correlation between these two ratios.

- R1 Working Capital/Total Asset and R7 Current Asset/Current Liability (correlation: 0.7):

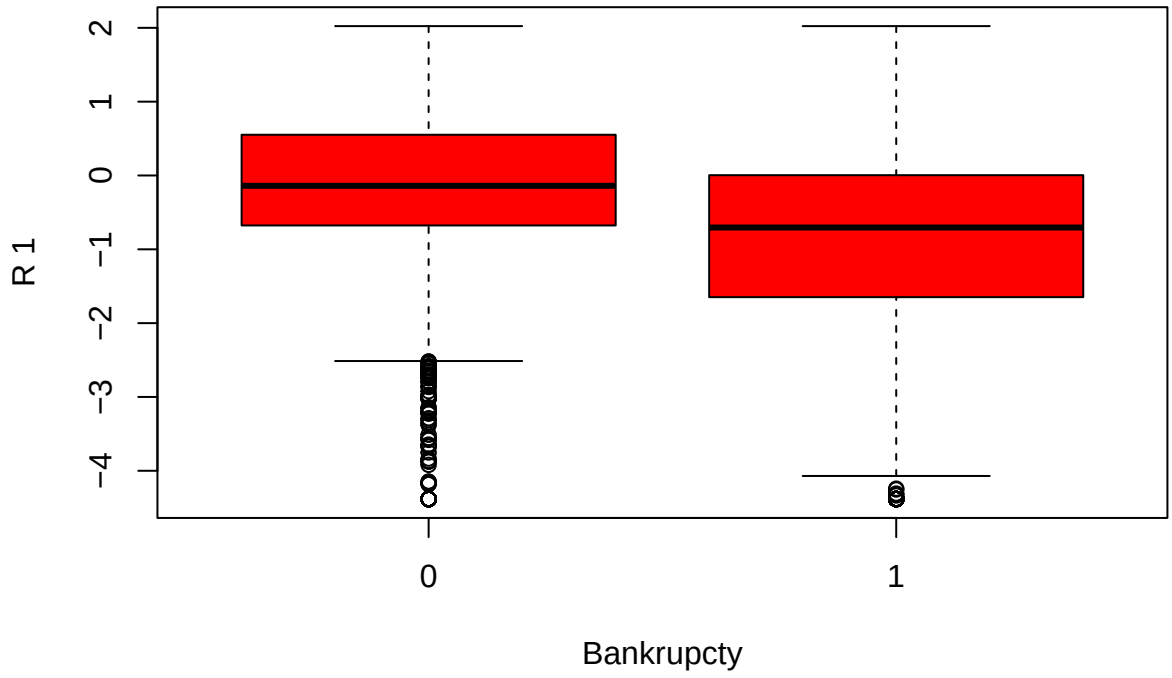
  The positive correlation between these two ratios is due to the nature of their components. Working capital is the difference between current assets and current liabilities. When a company has a higher ratio of current assets to current liabilities, it indicates better short-term liquidity and a stronger ability to cover short-term obligations. This often translates to a higher working capital relative to their total assets, leading to a positive correlation.

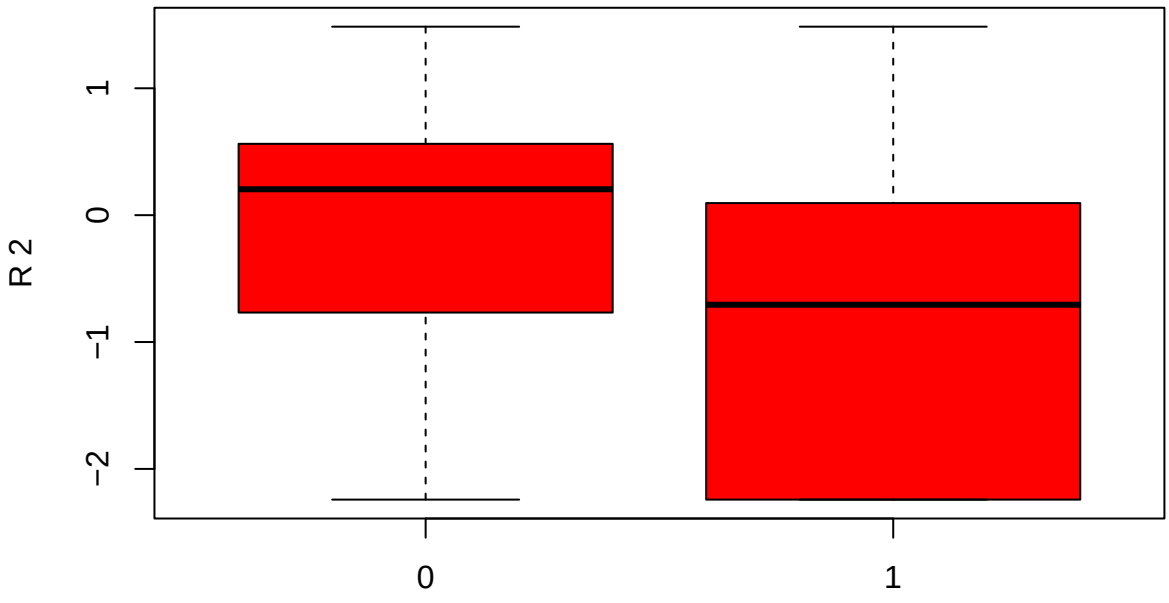
- R1 Working Capital/Total Asset and R6 Total Liability/Total Asset (correlation: -0.7):

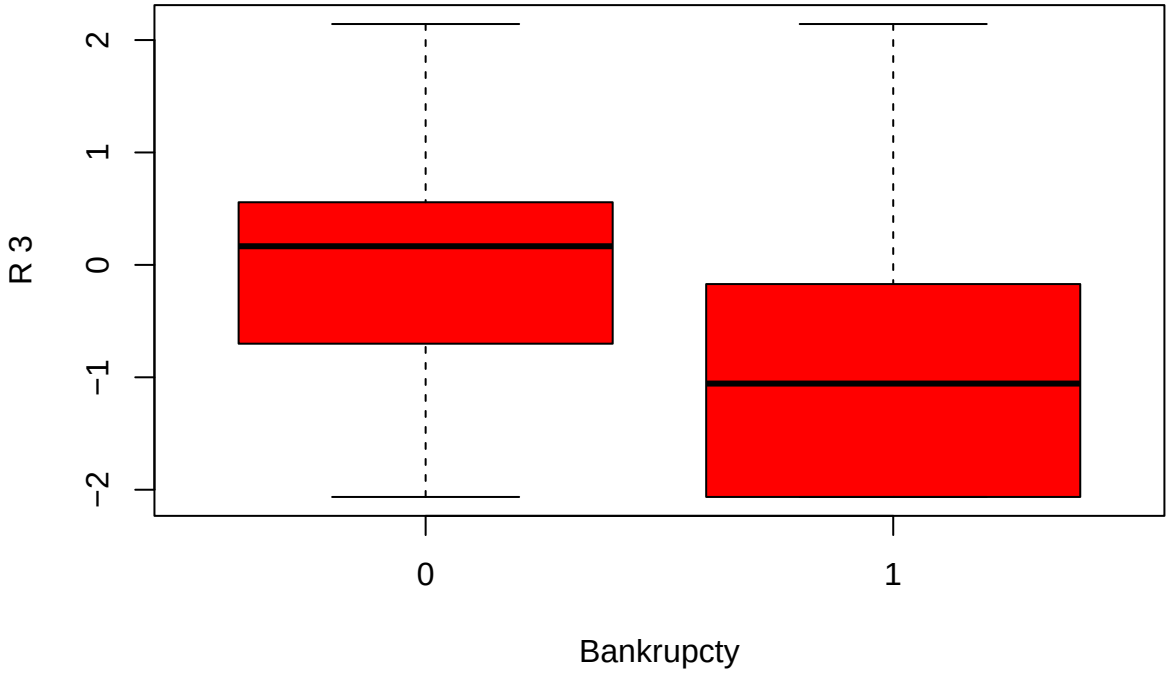
  The negative correlation between working capital/total assets and total liability/total asset can be explained by the impact of debt on a company's working capital. Working capital, as mentioned earlier, is the difference between current assets and current liabilities. When a company has higher total liabilities relative to its total assets, it typically indicates higher debt levels. Higher debt can reduce a company's working capital, leading to a negative correlation between these two ratios. Conversely, as a company's reliance on debt decreases, its working capital may increase, resulting in the observed negative correlation.

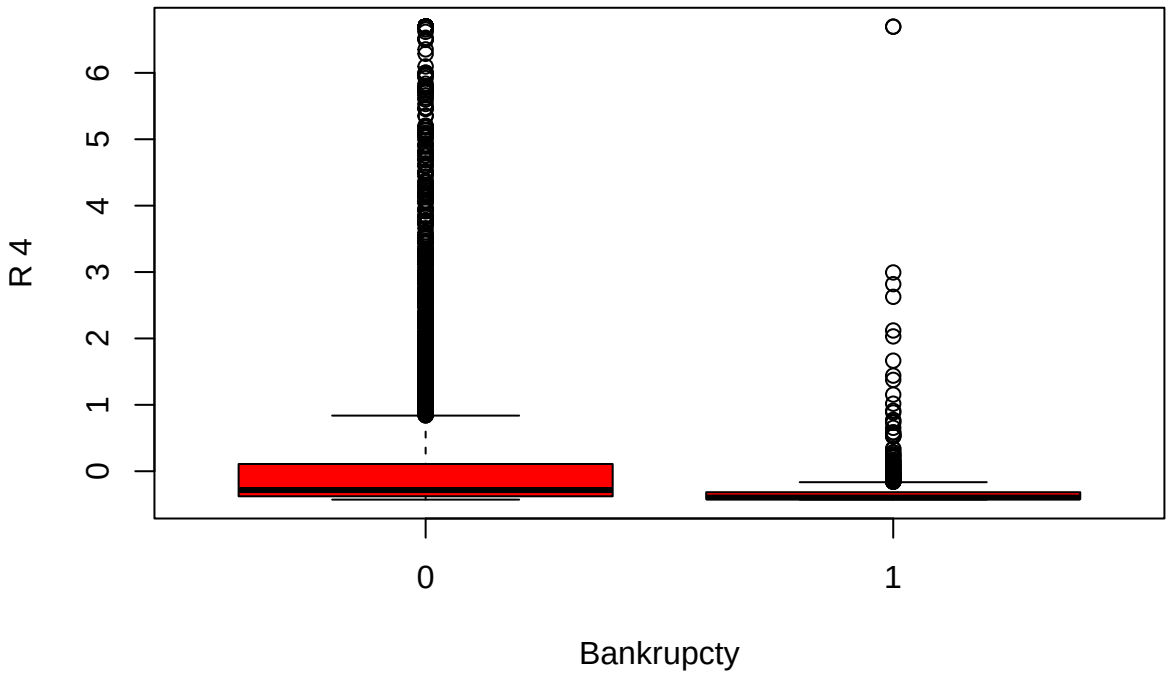
  We now observe through boxplots how the modes of each independent variable are distributed within each class of the dependent variable.

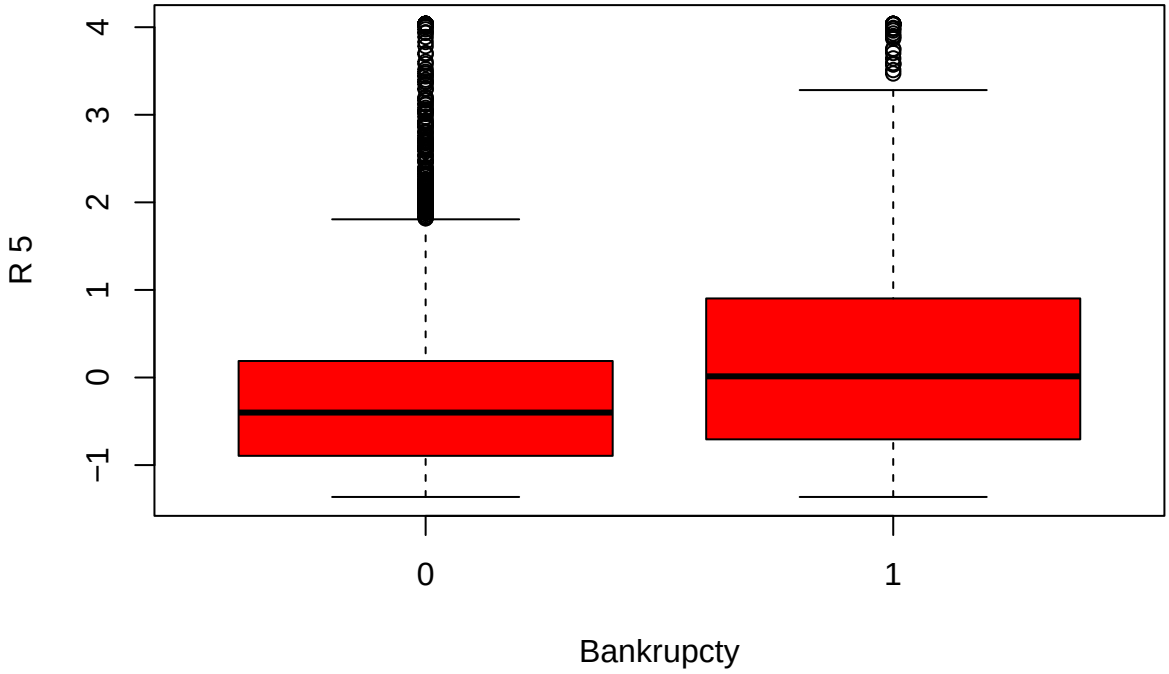
```
for (i in 2:11) {
  boxplot(Bank_clean[,i] ~ DLRSN, data = Bank_clean, xlab = "Bankrupcty", ylab = paste("R", i-1), col="
}
```

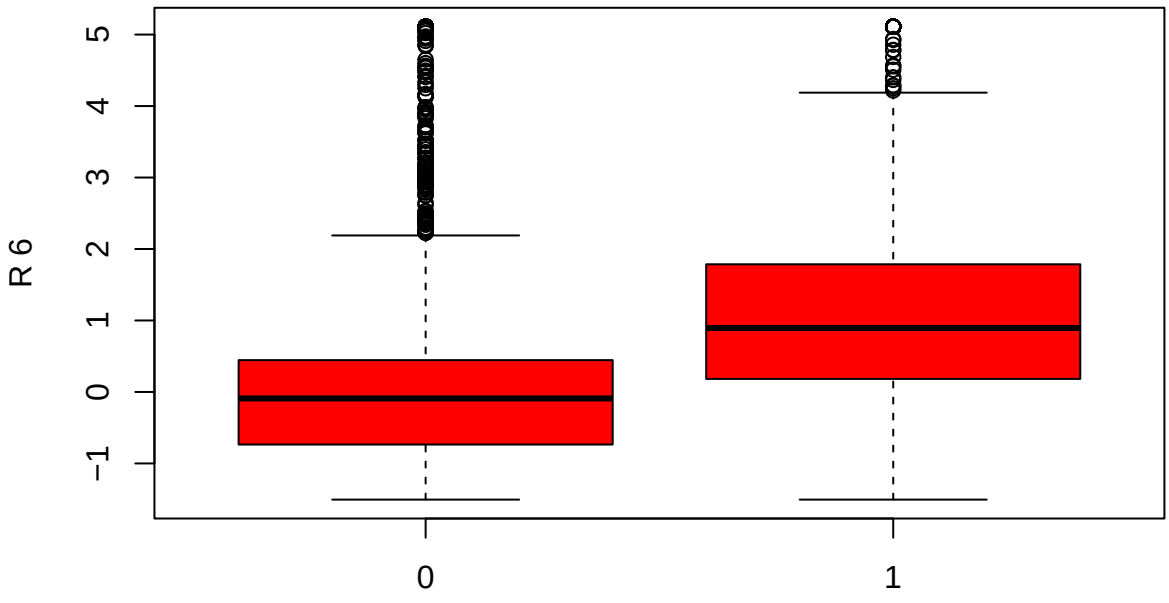


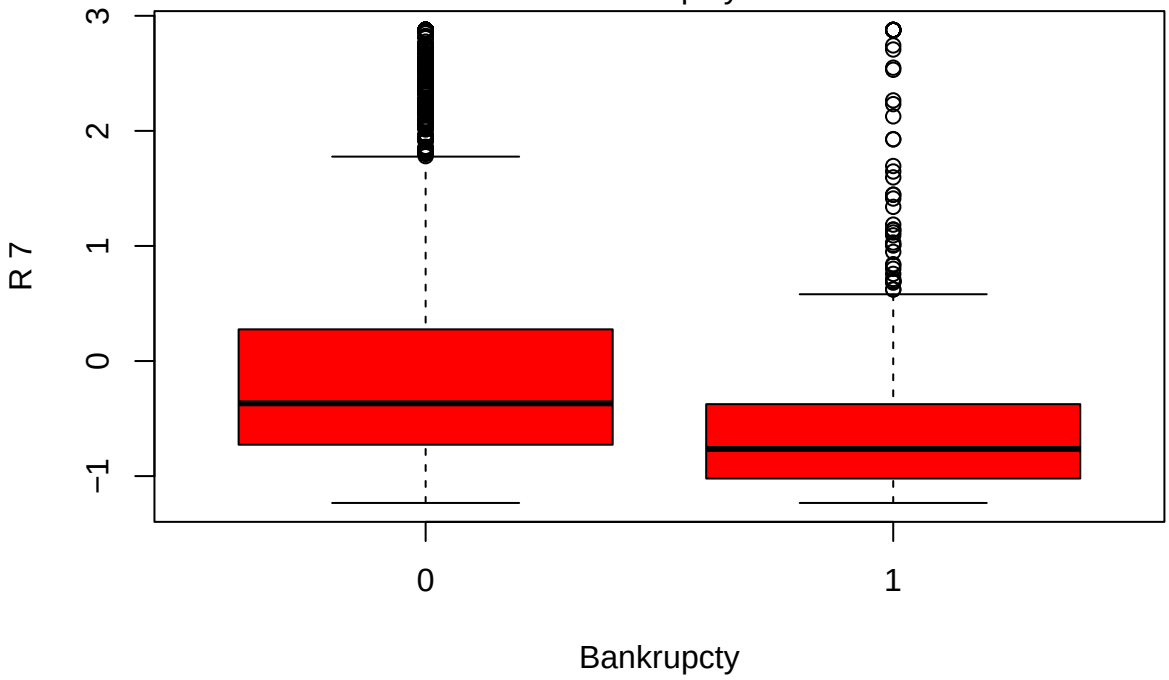

R 2
1
0
−1
−2
0
1
Bankrupcty

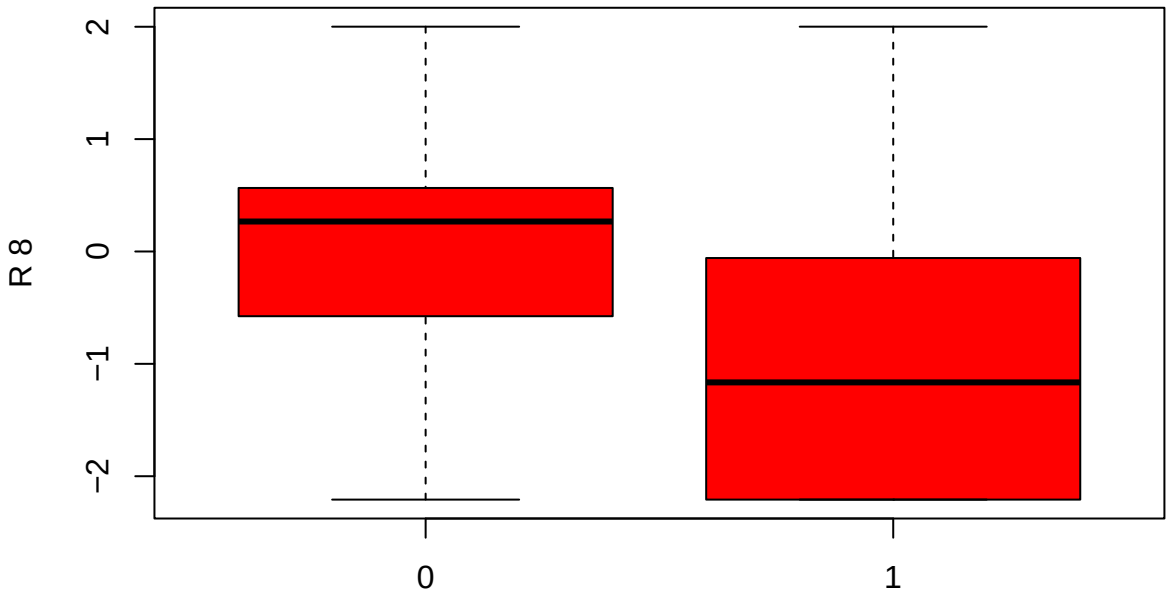

R 3
2
1
0
−1
−2
0
1
Bankrupcty

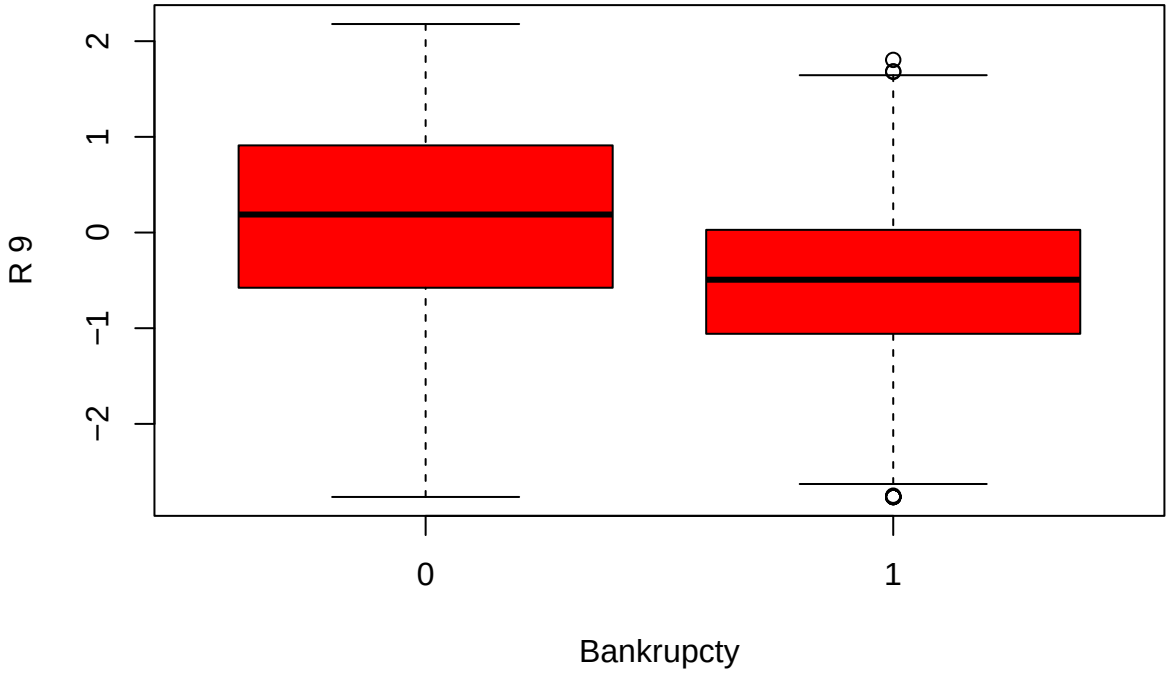

R 4
0
1
2
3
4
5
6
0
1
Bankrupcty

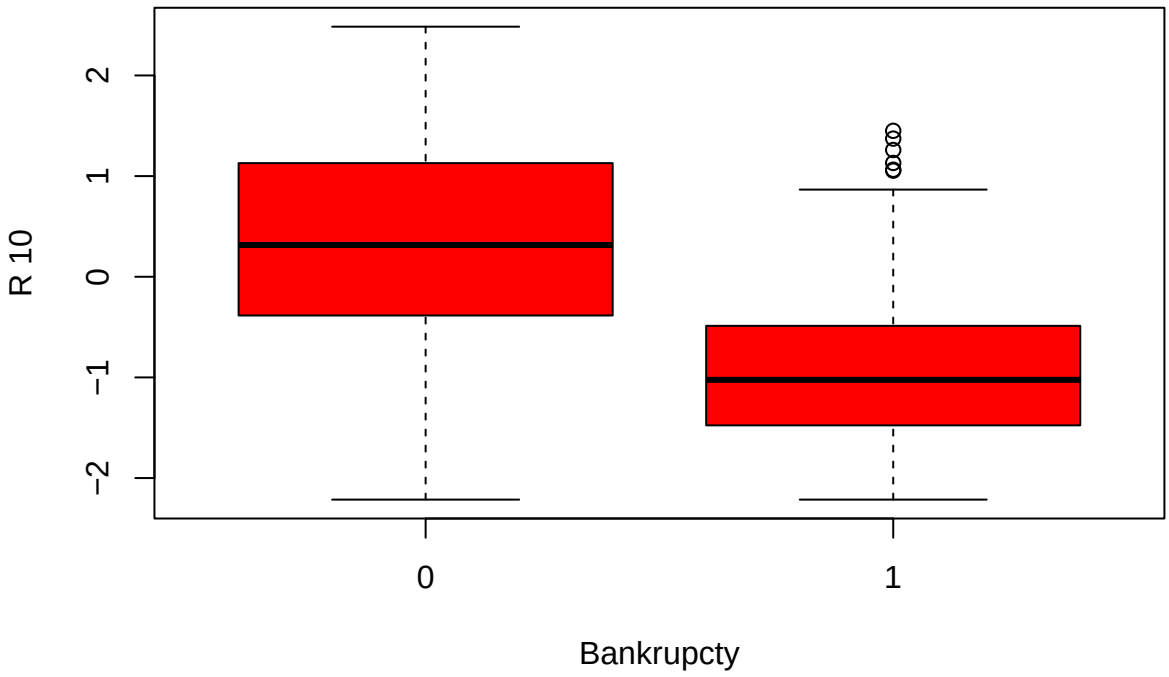

R 5
−1
0
1
2
3
4
0
1
Bankrupcty


R 6
5
4
3
2
1
0
−1
0
1
Bankrupcty


R 7
3
2
1
0
−1
0
1
Bankrupcty


R 8
2
1
0
−1
−2
0
1
Bankrupcty


R 9
2
1
0
−1
−2
0
1
Bankrupcty



- **R1 boxplot**:

It is normal to observe higher Working Capital/Total Assets (WC/TA) ratios for non-bankrupt companies. This trend is expected due to their focus on financial health, risk management, creditor confidence, and operational efficiency. Non-bankrupt companies often maintain higher working capital levels to ensure stability, sustain growth, and withstand economic challenges, making them better prepared to meet short-term obligations.

- **R2 boxplot**:

It is normal to observe higher Retained Earnings/Total Assets (RE/TA) ratios for non-bankrupt companies. This pattern indicates that non-bankrupt companies tend to retain a larger portion of their profits within the business, which contributes to their financial stability, supports sustainable growth, and enhances their ability to weather economic challenges.

- **R3 boxplot**:

It is normal to observe higher Earnings Before Interest and Tax/Total Assets (EBIT/TA) ratios for non-bankrupt companies. This trend indicates that non-bankrupt companies generate higher earnings relative to their total assets, reflecting their stronger profitability, effective cost management, and potential resilience in challenging economic conditions.

- **R4 boxplot**:

It is normal to observe higher Market Value of Equity/Total Liability (ME/TL) ratios for non-bankrupt companies. This pattern suggests that non-bankrupt companies have a higher market valuation relative to their total liabilities, indicating investor confidence, positive market perception, and lower perceived financial risk, which are all favorable indicators for the company's financial health and stability.

- **R5 boxplot**:

It is noteworthy to observe higher Sales/Total Assets (S/TA) ratios for bankrupt companies. This trend suggests that bankrupt companies generate higher sales relative to their total assets, which might indicate aggressive revenue recognition practices or inefficient asset utilization. High S/TA ratios without corresponding profitability and financial stability can pose a risk, contributing to financial distress and potential bankruptcy.

- **R6 boxplot**:

It is noteworthy to observe higher Total Liability/Total Assets (TL/TA) ratios for bankrupt companies. This trend indicates that bankrupt companies have a higher proportion of total liabilities relative to their total assets, which may imply excessive debt burden and financial leverage. High TL/TA ratios can increase financial risk, making it challenging for the company to meet its debt obligations and potentially leading to financial distress and bankruptcy.

- **R7 boxplot**:

It is normal to observe higher Current Assets/Current Liability (CA/CL) ratios for non-bankrupt companies. This pattern suggests that non-bankrupt companies have a higher proportion of current assets relative to their current liabilities, indicating stronger short-term liquidity and a better ability to cover short-term obligations. Higher CA/CL ratios are favorable indicators of financial health and reduce the risk of insolvency or bankruptcy for these companies.

- **R8 boxplot**:

It is normal to observe higher Net Income/Total Assets (NI/TA) ratios for non-bankrupt companies. This trend indicates that non-bankrupt companies generate higher net income relative to their total assets, reflecting their profitability, efficient operations, and potential financial stability. Higher NI/TA ratios are positive signals of a company's financial health and performance, reducing the likelihood of bankruptcy.

- **R9 boxplot**:

It is normal to observe higher Bankruptcy Cost (Log(Sales)) for non-bankrupt companies. This pattern suggests that non-bankrupt companies face lower bankruptcy costs, which may be attributed to their stronger financial position, better risk management, and ability to handle adverse events. Lower bankruptcy costs are indicative of reduced financial distress and a lower probability of bankruptcy for these companies.

- **R10 boxplot**:

It is normal to observe higher Market Capitalization (Log(Abs(Price) * Number of Shares Outstanding / 1000)) for non-bankrupt companies. This trend indicates that non-bankrupt companies have a higher market valuation, reflecting investor confidence, positive market sentiment, and better prospects for growth and financial stability. Higher market capitalization is a favorable indicator of the company's perceived value and reduced financial risk, contributing to a lower likelihood of bankruptcy.

Model Selection

We will now employ different methods to find the best model.

Data-set splitting

It's necessary now to split data into:

- training set, that will be used to train the models;
- test set, used to evaluate the performance of these models.

Training set is made up by 75% of the original dataset.

```
set.seed(12957846)
random_indx = sample(1:nrow(Bank_clean),
                     size = round(0.75 * nrow(Bank_clean)),replace = FALSE)
#Creation of training set:
train = Bank_clean[random_indx, ]
#Creation of test set:
test = Bank_clean[-random_indx, ]
```

Thanks to this split we can evaluate the models' performance over some unseen data.

Logistic regression models

We have created several logistic regression models to predict the likelihood of a company experiencing failure and classify whether it has failed or not. The logistic regression model is used to estimate the probability of an observation belonging to class 1 (failure) of the binary response variable, based on the values of its independent variables. This model is based on the logit function, which is the logarithm of the ratio between the probability of a company falling into bankruptcy and the probability of it not falling into bankruptcy.

The different models we have defined are as follows:

- **Complete Logistic Regression model**: This model uses all available independent variables to predict the probability of company failure.
- **Logistic with Stepsize Selection**: This model uses stepwise selection to determine which independent variables to include in the model, controlling the step size between variables.
  - **Backwards elimination**: In this model, starting with a full set of independent variables, gradually less significant variables are eliminated based on statistical tests or selection criteria.
  - **Forward selection**: In this model, starting with an empty set of independent variables, gradually more significant variables are added based on statistical tests or selection criteria.
- **Balanced logistic model**: This model addresses class imbalance using equal proportions of both positive and negative samples. This approach helps mitigate the impact of class imbalance and improves the model's ability to learn from both classes.
- **Balanced logistic model + Stepwise Selection**: This model combines class imbalance handling with stepwise selection by selecting the most relevant variables.
  - **Backward Balanced**: In this model, a combination of backward elimination and class balancing is used to select the most significant variables and address class imbalance.
  - **Forward Balanced**: In this model, a combination of forward selection and class balancing is used to select the most significant variables and address class imbalance.

By employing these various modeling techniques, we aim to identify the most influential independent variables and build reliable logistic regression models for predicting company failure.

Complete Logistic Regression model

```
Bank_clean$DLRSN=as.factor(Bank_clean$DLRSN)
model_1 = glm(data = train, train$DLRSN ~ .,family = "binomial")
s_1=summary(model_1)
s_1
```

```
##
## Call:
## glm(formula = train$DLRSN ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2840  -0.4639  -0.2391  -0.1112   3.3682
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.60352    0.08253 -31.548  < 2e-16 ***
## R1           0.19881    0.07731   2.571  0.01013 *
## R2           0.55800    0.08156   6.842 7.82e-12 ***
## R3          -0.46245    0.10726  -4.312 1.62e-05 ***
```

```
## R4          -0.10886    0.11281  -0.965  0.33456
## R5           0.01537    0.05530   0.278  0.78101
## R6           0.23295    0.05748   4.053 5.07e-05 ***
## R7          -0.55684    0.11476  -4.852 1.22e-06 ***
## R8          -0.29034    0.09268  -3.133  0.00173 **
## R9           0.31994    0.10817   2.958  0.00310 **
## R10         -1.50116    0.10111 -14.847  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3292.0  on 4076  degrees of freedom
## Residual deviance: 2260.1  on 4066  degrees of freedom
## AIC: 2282.1
##
## Number of Fisher Scoring iterations: 7
```

In this model, most of the covariates are found to be highly significant, suggesting that they have a significant impact on predicting a firm's bankruptcy. However, two variables, R4 (ME/TL - Market Value of Equity/Total Liability) and R5 (S/TA - Sales/Total Assets) coefficients are found to be close to zeros and do not contribute significantly to the determination of bankruptcy. These variables are not only statistically not very significant but also from the economic point of view they're limitated in predicting bankruptcy. This could be due to various factors, such as market volatility that may not always reflect the financial strength or risk of a company and due to a certain degree of discretion granted in the context of accounting pratices.

Specifically, the coefficient R1 has an estimated value of 0.19881. To obtain the odds value associated with R1, we can calculate the exponential of the coefficient. Therefore, the odds ratio (OR) for R1 can be calculated as: OR = exp(0.19881) ≈ 1.219 This means that for every one-unit increase in the R1 variable, the odds of success increase by approximately 21,9 %. Instead, the coefficient R10 has an estimated value of -1.50116. To interpret the coefficient R10, we can consider the exponential of the estimated value to obtain the odds ratio (OR): OR = exp(-1.50116) ≈ 0.222 This means that for every one-unit increase in the R10 variable, the odds of success decrease by approximately 77.8% compared to the reference category.

The deviances of the logistic regression model for bankruptcy prediction indicate the discrepancy between the observed data and the values predicted by the model. In the provided output, the null deviance is 3292.0 on 4076 degrees of freedom, while the residual deviance is 2260.1 on 4066 degrees of freedom.

A high null deviance suggests that an empty model without any predictors fails to explain the variability in the data. Conversely, a lower residual deviance indicates that the proposed logistic model has a better fit to the data compared to the empty model. However, it is important to consider additional metrics and the specific context to assess the adequacy of the model in predicting bankruptcy.

It is advisable to compare these deviances with other goodness-of-fit measures such as AIC and BIC to obtain a more comprehensive evaluation of the logistic model's performance in bankruptcy prediction. Additionally, conducting residual analysis and model validation tests can provide further insights into the model's effectiveness. We safely ascertain the difference between the two deviations by performing a chi-squared test.

```
p_value= 1 - pchisq(s_1$null.deviance-s_1$deviance, 7)
p_value
```

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

To assess the presence of multicollinearity among the independent variables, the Variance Inflation Factor (VIF) is introduced for all the covariates. The VIF indicates if an independent variable is more strongly related to another independent variable than to the response variable. A variable is affected by multicollinearity if its VIF value exceeds 5 or 10.

Please note that the mentioned VIF threshold (5 or 10) is commonly used as a guideline, but the specific threshold may vary depending on the context and the particular analysis.

```
# Compute the VIF
vif(model_1)
```

```
##       R1       R2       R3       R4       R5       R6       R7       R8
## 4.111213 3.085077 4.203248 1.498436 1.519480 2.936702 2.360513 3.984901
##       R9      R10
## 3.638721 2.334999
```

As it's showed by the results, this model is not affected by multicollinearity.

We can therefore proceed with the predictions. As the aim of the study is predicting bankruptcy, we'd like to have a model that doesn't missclassify an example if labelled as affected by the bankruptcy.

```
#Prediction
pred_model_1<- predict(model_1, test, type = "response")
pred_model_1r<- ifelse(pred_model_1 > 0.5, 1, 0)
```

Additionally, we calculate the confusion matrix to assess various evaluation metrics, notably Accuracy and Sensitivity.

Accuracy quantifies the correctness of predictions across the entire dataset:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}}$$

Sensitivity, also known as True Positive Rate, evaluates the model's capability to correctly identify positive instances.

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

```
#Confusion matrix
c_1=table(test$DLRSN, pred_model_1r)
c_1
```

```
##    pred_model_1r
##        0    1
##   0 1122   29
##   1  131   77
```

Accuracy:

```
accuracy_1 = (c_1[1,1]+c_1[2,2])/nrow(test)
accuracy_1
```

```
## [1] 0.8822664
```

Even if we have an high accuracy it does not mean that our model is performing well. In fact we are using unbalanced dataset and according to the theory we know that the accuracy is not very significant in this case.

Sensitivity:

```
sensitivity_1 = c_1[2,2]/(c_1[2,2]+c_1[2,1])
sensitivity_1
```

```
## [1] 0.3701923
```

Now we can see that the sensitivity is quite low, but this is reasonable given that we have fewer observations of bankruptcy rather than of non-bankruptcy observations and the model may not have sufficient observation to understand characteristics of a bankrupted company.

Balanced logistic model

To solve the problem of the unbalance of the response variable, it's defined a balanced model, implemented through oversampling and undersampling methods. These methods are used to solve the issue regarding unbalanced data, because it can negatively impact the performance of classifiers, as they tend to favor the majority class due to the imbalance in the training data.

Firstly, we implement this method through the `ovun.sample` function of the "*ROSE*" library. In our study, we implement both oversampling and undersampling by setting `method` of `ovun sample` to `both`; this means that new synthetic samples will be generated for the minority class and random samples will be removed from the majority class. However, we must pay attention to the possible loss of information.

```
data_balanced<- ovun.sample(DLRSN ~ ., data = Bank_clean, method = "both", p = 0.5, seed = 1)$data
prop.table(table(data_balanced$DLRSN))
```

```
##
##         0         1
## 0.5112215 0.4887785
```

After that the variable "***DLRSN***" is balanced, it is necessary to define a balanced training set over train our model:

```
set.seed(123)
random_indices <- sample(1:nrow(Bank_clean), size = round(0.75 * nrow(Bank_clean)), replace = FALSE)
train_balanced<- data_balanced[random_indices, ]
```

```
model_2 <- glm(data = train_balanced, train_balanced$DLRSN ~ ., family = binomial())
s_2<- summary(model_2)
s_2
```

```
##
## Call:
## glm(formula = train_balanced$DLRSN ~ ., family = binomial(),
##     data = train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0432  -0.6654  -0.1137   0.6817   2.7193
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.80528    0.05203 -15.477  < 2e-16 ***
## R1           0.18576    0.06100   3.045 0.002323 **
## R2           0.51445    0.06450   7.976 1.51e-15 ***
## R3          -0.35974    0.07908  -4.549 5.39e-06 ***
## R4          -0.31622    0.07705  -4.104 4.06e-05 ***
## R5           0.09481    0.04980   1.904 0.056909 .
## R6           0.39678    0.05200   7.630 2.34e-14 ***
## R7          -0.17752    0.07467  -2.377 0.017439 *
## R8          -0.25487    0.06803  -3.747 0.000179 ***
## R9          -0.11115    0.09162  -1.213 0.225045
## R10         -1.27695    0.08215 -15.544  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5649.7  on 4076  degrees of freedom
## Residual deviance: 3588.7  on 4066  degrees of freedom
## AIC: 3610.7
##
## Number of Fisher Scoring iterations: 6
```

In the second model, each variable is highly significant and has a positive contribution on the probability of having diabetes, except for R9 Log(SALE) which exhibits no significant impact on bankruptcy classification (p-value= 0.22). Visually inspecting the distribution of Log(SALE) using histograms and Q-Q plots indicates that it follows an approximately normal distribution. Despite its normally, forma an economic standpoint, this variable's limited significance suggests that fluctuations in revenue may not be a reliable indicator for predicting bankruptcy risk. Other crucial financial factors such as debt levels, cash flow, and profitability are likely playing more substantial roles in determining a company's financial health and likelihood of bankruptcy. Consequently, removing Log(SALE) from the model is justified, as it alings with the variable's limited economic significance and allows a focus on more influential predictors.

The difference between the two deviances explains that the model is well fitted on the data; while the AIC value is bigger than the one of the model_1, that may be due to the fact that changhing the distribution of balanced data may affect the fit of the model. Redistribution of data may result in a reduction in the model fit to the data , which could be reflected in an increase in AIC.

```
vif(model_2)
```

```
##       R1       R2       R3       R4       R5       R6       R7       R8
## 3.451273 3.012555 4.139814 1.606045 1.684777 2.600375 2.459634 3.867552
##       R9      R10
## 4.192459 2.570544
```

Also for *model_2*, there isn't the multicollinearity problem, so we can keep all the covariates.

```
pred_model_2<- predict(model_2, test, type = "response")
pred_model_2<- ifelse(pred_model_2 > 0.5, 1, 0)

table(test$DLRSN, pred_model_2)
```

```
##    pred_model_2
##       0   1
##   0 942 209
##   1  34 174
```

```
cm_m3=table(test$DLRSN, pred_model_2)
```

The predictions are computed on the original test set, to verify the prediction abilities after applying data balancing on the balanced training set. It produces a number of False Negatives lower than *model_1* and a bigger number of False Positives. As a result of balancing, the model becomes more sensitive to detecting instances of the minority class, leading to a reduction in False Negatives. The model is now more capable of correctly identifying positive instances from the minority class, resulting in fewer instances being falsely classified as negatives. However, this increased sensitivity to the minority class may come at the cost of increased False Positives. The model might become more prone to incorrectly classifying instances from the majority class as positives, resulting in a higher number of False Positives. This trade-off between False Negatives and False Positives is a common challenge when dealing with imbalanced datasets.

```
accuracy_2 = ( cm_m3[1,1]+cm_m3[2,2] ) / nrow(test)
accuracy_2
```

```
## [1] 0.8211921
```

```
sensitivity_2 = cm_m3[2,2] / ( cm_m3[2,2]+ cm_m3[2,1] )

sensitivity_2
```

```
## [1] 0.8365385
```

The metrics values are satisfying: the model has an ability to correctly classify examples equal to 81.1% and has the highest value of True Positive Rate(83.65 %), so far.

Logistic with Stepsize Selection

Stepwise Selection, a variable selection method, is used to determine the optimal predictor variables for our logistic regression model. The process involves iteratively adding or removing predictors based on the Akaike Information Criteria (AIC). This criteria balances model complexity (number of parameters) and data fit (log-likelihood) with the formula AIC = 2k - 2*log(L). The goal is to minimize AIC, indicating a better-fitting model. We apply both Forward selection, starting with the Null model, and Backwards elimination, starting with the Complete model, considering the previous VIF test results for model improvement.

Backwards elimination

```
model_opt <- glm(data = train, train$DLRSN ~ ., family = binomial())
model_step_back <- stepAIC(model_opt, direction = "back" ,
trace = FALSE)
```

```
s_back=summary(model_step_back)
s_back
```

```
##
## Call:
## glm(formula = train$DLRSN ~ R1 + R2 + R3 + R6 + R7 + R8 + R9 +
##     R10, family = binomial(), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2922  -0.4616  -0.2401  -0.1149   3.3978
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.60040    0.08174 -31.811  < 2e-16 ***
## R1           0.20448    0.07709   2.653  0.00799 **
## R2           0.56679    0.08103   6.995 2.66e-12 ***
## R3          -0.46530    0.10729  -4.337 1.44e-05 ***
## R6           0.24334    0.05615   4.334 1.47e-05 ***
## R7          -0.58467    0.11105  -5.265 1.40e-07 ***
## R8          -0.28543    0.09267  -3.080  0.00207 **
## R9           0.36553    0.08800   4.154 3.27e-05 ***
## R10         -1.54677    0.08519 -18.157  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3292.0  on 4076  degrees of freedom
## Residual deviance: 2261.2  on 4068  degrees of freedom
```

```
## AIC: 2279.2
##
## Number of Fisher Scoring iterations: 6
```

The estimated model excludes the coefficients of R4 and R5, as they were found to be statistically insignificant in the complete model.

With this refined model, we can proceed to make predictions.

```
#Prediction
pred_model_back<- predict(model_step_back, test, type = "response")
pred_model_backr<- ifelse(pred_model_back > 0.5, 1, 0)
```

We compute the confusion matrix and the previous metrics:

```
#Confusion matrix
c_back=table(test$DLRSN, pred_model_backr)
c_back
```

```
##    pred_model_backr
##        0    1
##   0 1122   29
##   1  131   77
```

Accuracy:

```
accuracy_back = (c_back[1,1]+c_back[2,2])/nrow(test)
accuracy_back
```

```
## [1] 0.8822664
```

Sensitivity:

```
sensitivity_back = c_back[2,2]/(c_back[2,2]+c_back[2,1])
sensitivity_back
```

```
## [1] 0.3701923
```

Forward selection

```
model_null=glm(data = train, train$DLRSN ~ 1, family = binomial())
model_step_for <- step(model_null, direction = "forward" ,
scope = list(lower = model_null, upper = model_opt) ,
trace = FALSE)
```

```
s_for=summary(model_step_for)
s_for
```

```
##
## Call:
## glm(formula = train$DLRSN ~ R10 + R7 + R9 + R3 + R2 + R6 + R8 +
##     R1, family = binomial(), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2922  -0.4616  -0.2401  -0.1149   3.3978
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -2.60040    0.08174 -31.811  < 2e-16 ***
## R10         -1.54677    0.08519 -18.157  < 2e-16 ***
## R7          -0.58467    0.11105  -5.265 1.40e-07 ***
## R9           0.36553    0.08800   4.154 3.27e-05 ***
## R3          -0.46530    0.10729  -4.337 1.44e-05 ***
## R2           0.56679    0.08103   6.995 2.66e-12 ***
## R6           0.24334    0.05615   4.334 1.47e-05 ***
## R8          -0.28543    0.09267  -3.080  0.00207 **
## R1           0.20448    0.07709   2.653  0.00799 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3292.0  on 4076  degrees of freedom
## Residual deviance: 2261.2  on 4068  degrees of freedom
## AIC: 2279.2
##
## Number of Fisher Scoring iterations: 6
```

```
p_value= 1 - pchisq(s_for$null.deviance-s_for$deviance, 6)
p_value
```

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

It is clear from the difference between the two deviances that the model is well fitted on the data.

We can therefore proceed with the predictions:

```
#Prediction
pred_model_for<- predict(model_step_for, test, type = "response")
pred_model_forr<- ifelse(pred_model_for > 0.5, 1, 0)
```

```
#Confusion matrix
c_for=table(test$DLRSN, pred_model_forr)
c_for
```

```
##    pred_model_forr
##        0    1
##   0 1122   29
##   1  131   77
```

Accuracy:

```
accuracy_for = (c_for[1,1]+c_for[2,2])/nrow(test)

accuracy_for
```

```
## [1] 0.8822664
```

Sensitivity:

```
sensitivity_for = c_for[2,2]/(c_for[2,2]+c_for[2,1])

sensitivity_for
```

```
## [1] 0.3701923
```

The performance of these predictive models are the same as the logistic regression both in terms of accuracy and sensitivity, these models are performing similarly in terms of predicting and classifying bankruptcy. The

confusion matrix outcome corresponds to the one obtained by the complete logistic rergession model. The reason could lie in the fact that R4 and R5 are not very significant for the purposes of prediction, we can observe this by comparing the AICs obtained from the different models. in fact we can see that there is a slight decrease but since this difference is very small it implies that the two variables are not very significant

Balanced logistic model + Stepwise Selection

We apply the Stepwise Selection method to the balanced regression model:

Backward Balanced

```
#Definition of the model

model_2b <- step(model_2 , direction="backward", trace=F)
s_2b<- summary(model_2b, criteria = "aic")
s_2b
```

```
##
## Call:
## glm(formula = train_balanced$DLRSN ~ R1 + R2 + R3 + R4 + R5 +
##     R6 + R7 + R8 + R10, family = binomial(), data = train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0026  -0.6680  -0.1212   0.6773   2.6584
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.80665    0.05193 -15.533  < 2e-16 ***
## R1           0.17472    0.06035   2.895 0.003793 **
## R2           0.49019    0.06126   8.002 1.22e-15 ***
## R3          -0.37527    0.07813  -4.803 1.56e-06 ***
## R4          -0.27886    0.06883  -4.052 5.09e-05 ***
## R5           0.06186    0.04164   1.486 0.137371
## R6           0.38301    0.05065   7.562 3.96e-14 ***
## R7          -0.16862    0.07427  -2.270 0.023188 *
## R8          -0.24412    0.06741  -3.621 0.000293 ***
## R10         -1.34785    0.05835 -23.101  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5649.7  on 4076  degrees of freedom
## Residual deviance: 3590.2  on 4067  degrees of freedom
## AIC: 3610.2
##
## Number of Fisher Scoring iterations: 5
```

As we expectd, this method remove R9, but the aic remains basically the same. The covariates are significant, except R5, which is slightly unremarkable, but the method decide to keep it anyway

```
vif(model_2b)
```

```
##       R1       R2       R3       R4       R5       R6       R7       R8
## 3.397445 2.724131 4.039316 1.346864 1.188126 2.487946 2.445335 3.797288
```

```
##      R10
## 1.297279
```

```
pred_model_2b<- predict(model_2b, test, type = "response")
pred_model_2b<- ifelse(pred_model_2b > 0.5, 1, 0)

table(test$DLRSN, pred_model_2b)
```

```
##    pred_model_2b
##       0   1
##   0 943 208
##   1  34 174
```

```
cm_m5=table(test$DLRSN, pred_model_2b)
```

```
accuracy_5 = ( cm_m5[1,1]+cm_m5[2,2] ) / nrow(test)
accuracy_5
```

```
## [1] 0.8219279
```

```
sensitivity_5 = cm_m5[2,2] / ( cm_m5[2,2]+ cm_m5[2,1] )
sensitivity_5
```

```
## [1] 0.8365385
```

Looking at the accuracy and sensitivity results, we note that the results are the same as for the full model trained on balanced data. So removing the nonsignificant covariate R9 did not lead to much improvement in performance.

Forward Balanced

```
model_null_b=glm(data = train_balanced, train_balanced$DLRSN ~ 1, family = binomial())
model_2f <- step(model_null_b, direction = "forward" ,

scope = list(lower = model_null_b, upper = model_2) ,
trace = FALSE)
s_2f<- summary(model_2f, criteria = "aic")
s_2f
```

```
##
## Call:
## glm(formula = train_balanced$DLRSN ~ R10 + R6 + R4 + R3 + R2 +
##     R8 + R1 + R7 + R5, family = binomial(), data = train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0026  -0.6680  -0.1212   0.6773   2.6584
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.80665    0.05193 -15.533  < 2e-16 ***
## R10         -1.34785    0.05835 -23.101  < 2e-16 ***
## R6           0.38301    0.05065   7.562 3.96e-14 ***
## R4          -0.27886    0.06883  -4.052 5.09e-05 ***
## R3          -0.37527    0.07813  -4.803 1.56e-06 ***
## R2           0.49019    0.06126   8.002 1.22e-15 ***
## R8          -0.24412    0.06741  -3.621 0.000293 ***
```

```
## R1           0.17472    0.06035   2.895 0.003793 **
## R7          -0.16862    0.07427  -2.270 0.023188 *
## R5           0.06186    0.04164   1.486 0.137371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5649.7  on 4076  degrees of freedom
## Residual deviance: 3590.2  on 4067  degrees of freedom
## AIC: 3610.2
##
## Number of Fisher Scoring iterations: 5
```

```
vif(model_2f)
```

```
##      R10       R6       R4       R3       R2       R8       R1       R7
## 1.297279 2.487946 1.346864 4.039316 2.724131 3.797288 3.397445 2.445335
##       R5
## 1.188126
```

```
pred_model_2f<- predict(model_2f, test, type = "response")
pred_model_2f<- ifelse(pred_model_2f > 0.5, 1, 0)

table(test$DLRSN, pred_model_2f)
```

```
##    pred_model_2f
##       0   1
##   0 943 208
##   1  34 174
```

```
cm_m6=table(test$DLRSN, pred_model_2f)
```

```
accuracy_2f = ( cm_m6[1,1]+cm_m6[2,2] ) / nrow(test)
accuracy_2f
```

```
## [1] 0.8219279
```

```
sensitivity_2f = cm_m6[2,2] / ( cm_m6[2,2]+ cm_m6[2,1] )
sensitivity_2f
```

```
## [1] 0.8365385
```

Shrinkage methods

In the context of bankruptcy classification, Shrinkage methods are a group of statistical techniques used to improve the performance of predictive models by reducing the impact of noisy or less informative features (i.e., financial ratios). The term "shrinkage" refers to the process of moderating the estimates of model coefficients towards a central value or towards zero. Shrinkage methods are particularly valuable when dealing with high-dimensional data, where the number of features (financial ratios) is large relative to the number of observations (companies).

Shrinkage methods are a regularization technique that adds a penalty term to the standard loss function used in the model training process. This penalty discourages the model from relying too heavily on any single feature, thus avoiding overfitting and improving generalization to new, unseen data. The two most commonly used Shrinkage methods are Ridge Regression and Lasso Regression.

Another way to select the best independent variables that allows forecasting the diabetes status is represented

by shrinkage methods. Moreover they are useful to avoid Overfitting. In these algorithms, to compute the coefficients of the regression, we minimize a function made up by squared errors and a penalty term that makes coefficients estimates shrinking towards 0. In particular we are fitting:

- Ridge Regression model, that uses quadratic shrinkage : in the loss we consider the sum of squared coefficients multiplied by the tuning parameter lambda.
- Lasso Regression model, that uses absolute value shrinkage: in the loss we consider the sum of absolute value coefficients multiplied by the tuning parameter lambda.

The tuning parameter plays an important role in shrinkage methods, since higher the value, bigger will be the penalization. In both methods we use cross validation to choose the lambda which minimizes the misclassification error (maximizes the accuracy). It's important to note that, for their different loss formulation, Lasso regression is able to shrink parameter to exactly zero (and so it performs an automatic predictors selection), while Ridge is not capable of this.

Ridge Regression

Before applying Ridge and Lasso regression using the glmnet library, we must create two separate datasets: a training set and a test set. These datasets should not include the response variable (target variable) we are trying to predict, as the goal is to use the remaining features (financial ratios) to build the predictive models.

Once we have the training and test sets ready, we need to convert them into matrix objects. The glmnet library expects the data in matrix format for efficient computation. By converting the datasets into matrices, we ensure that the data is represented in a suitable format for the regression models.

```
train_for_shrinkage=as.matrix(train[,-1])
test_for_shrinkage=as.matrix(test[,-1])
```

```
model_ridge=cv.glmnet(train_for_shrinkage, train$DLRSN,
alpha = 0, family = "binomial", type.measure = "class")
model_ridge
```

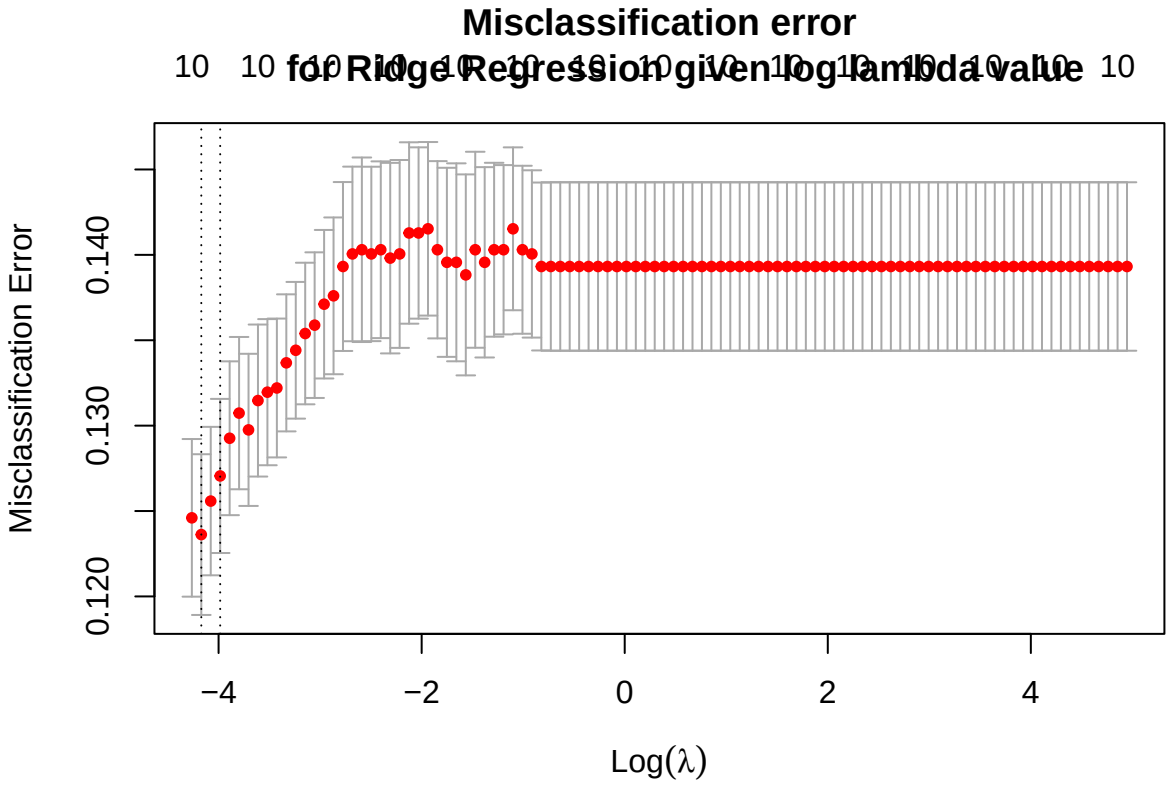
```
##
## Call:  cv.glmnet(x = train_for_shrinkage, y = train$DLRSN, type.measure = "class",      alpha = 0, fa
##
## Measure: Misclassification Error
##
##      Lambda Index Measure       SE Nonzero
## min 0.01545    99  0.1236 0.004705      10
## 1se 0.01861    97  0.1270 0.004512      10
```

```
coef(model_ridge)
```

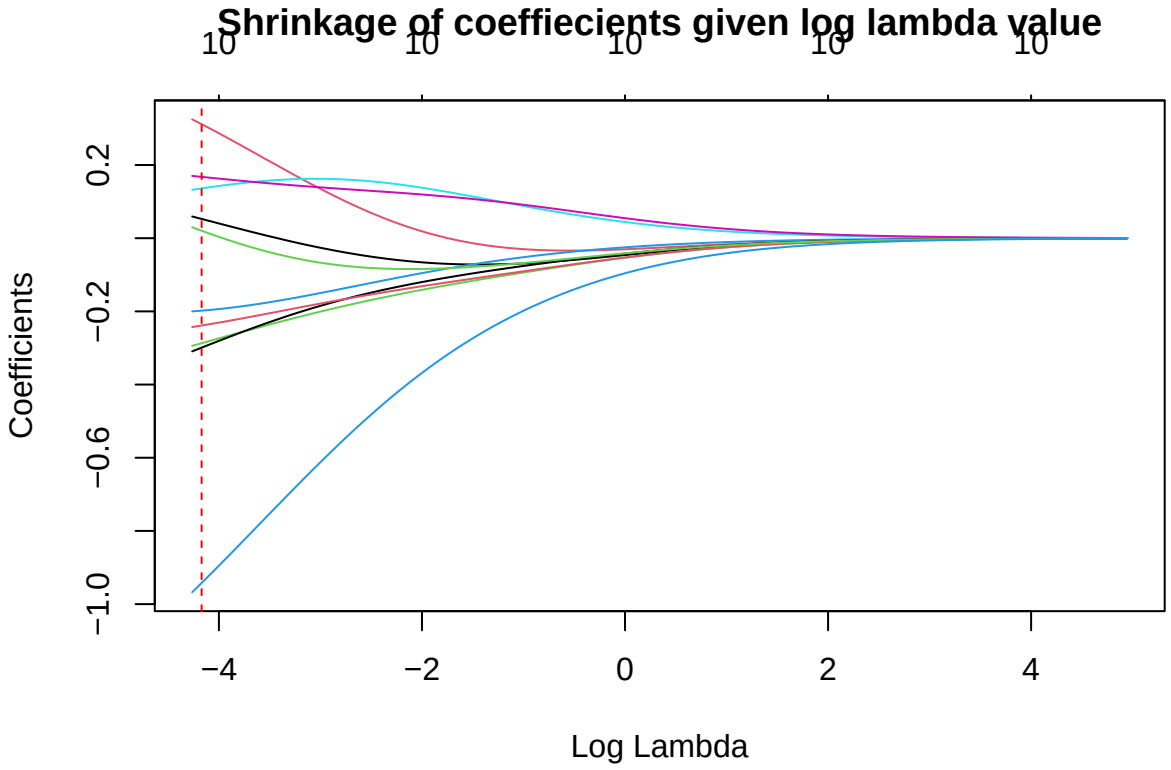
```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -2.315539443
## R1           0.039188818
## R2           0.284372281
## R3          -0.272575027
## R4          -0.193213599
## R5           0.143369172
## R6           0.162414022
## R7          -0.278881964
## R8          -0.229904146
## R9           0.002104827
## R10         -0.890564985
```

Ridge regression successfully prevents overfitting by shrinking the coefficient estimates towards zero, but they are not forced to reach exactly zero. This allows us to maintain all the financial ratios (R1 to R10) in the model while reducing their impact to some extent. It is interesting to note that the "Intercept" plays a significant role in the classification, and some ratios, like R2, R6, and R9, have a positive impact, while others, like R3, R4, R7, and R8, have a negative impact. The ratio R10, in particular, appears to have a strong negative effect on the classification outcome.

```
plot(model_ridge, main="Misclassification error
     for Ridge Regression given log lambda value")
```




```
z=glmnet(train_for_shrinkage, train$DLRSN,

alpha = 0, family = "binomial", type.measure = "class")
plot(z, xvar="lambda",
     main="Shrinkage of coeffiecients given log lambda value")
abline(v = log(model_ridge$lambda.min), col = "red",lty=2)
```



We can use the model to make predictions and compute our metrics of interest:

```
pred_ridge<- predict(model_ridge,test_for_shrinkage,
                     type = "response", s = model_ridge$lambda.min)
pred_ridger<-ifelse(pred_ridge > 0.5, 1, 0)
```

```
#Confusion matrix
c_ridge=table(test$DLRSN, pred_ridger)
c_ridge
```

```
##    pred_ridger
##        0    1
##   0 1126   25
##   1  159   49
```

Accuracy:

```
accuracy_ridge = (c_ridge[1,1]+c_ridge[2,2])/nrow(test)

accuracy_ridge
```

```
## [1] 0.8646063
```

Sensitivity:

```
sensitivity_ridge = c_ridge[2,2]/(c_ridge[2,2]+c_ridge[2,1])
sensitivity_ridge
```

```
## [1] 0.2355769
```

Lasso Regression

Lasso Regression, also known as L1 regularization, shares similarities with Ridge Regression but employs a different penalty term. Instead of using the L2 norm of the coefficients vector, Lasso uses the L1 norm. This has the effect of driving some coefficients to exactly zero, effectively performing feature selection.

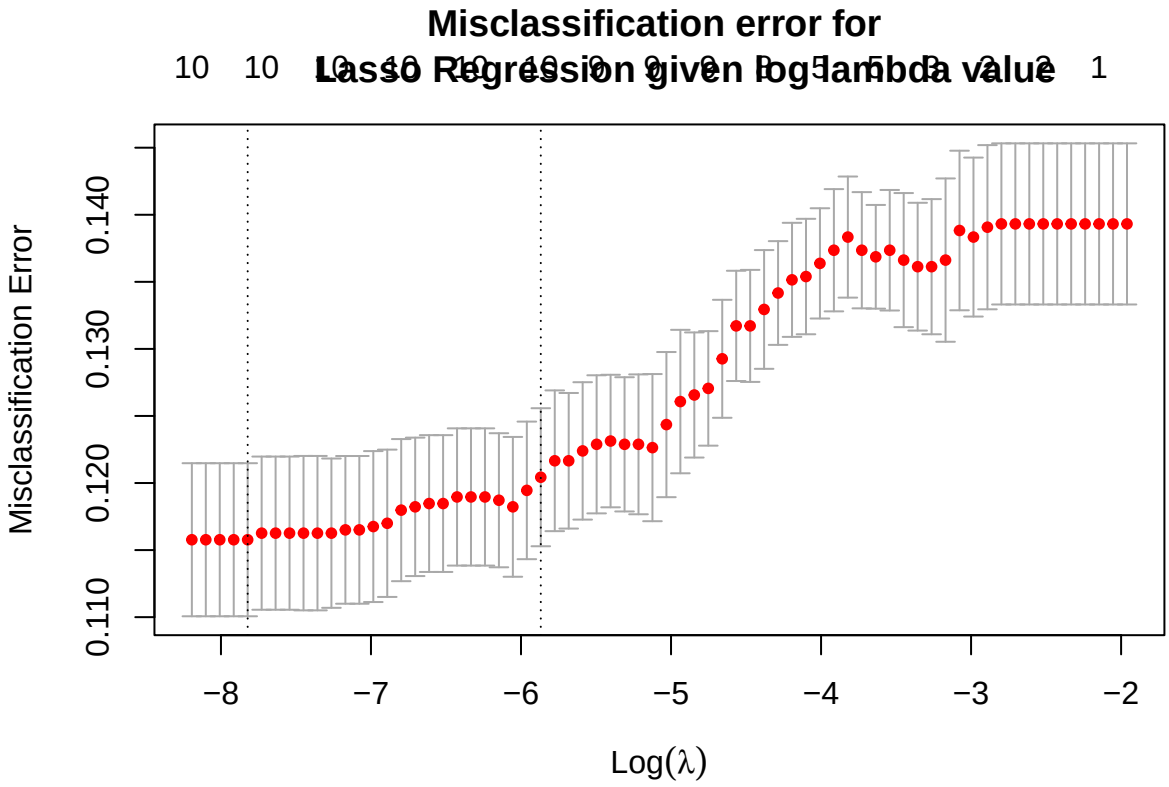
```
model_lasso=cv.glmnet(train_for_shrinkage, train$DLRSN,

alpha = 1, family = "binomial", type.measure = "class")
model_lasso
```

```
##
## Call:  cv.glmnet(x = train_for_shrinkage, y = train$DLRSN, type.measure = "class",      alpha = 1, fa
##
## Measure: Misclassification Error
##
##        Lambda Index Measure       SE Nonzero
## min 0.0004009    64  0.1158 0.005706      10
## 1se 0.0028283    43  0.1204 0.005146      10
```

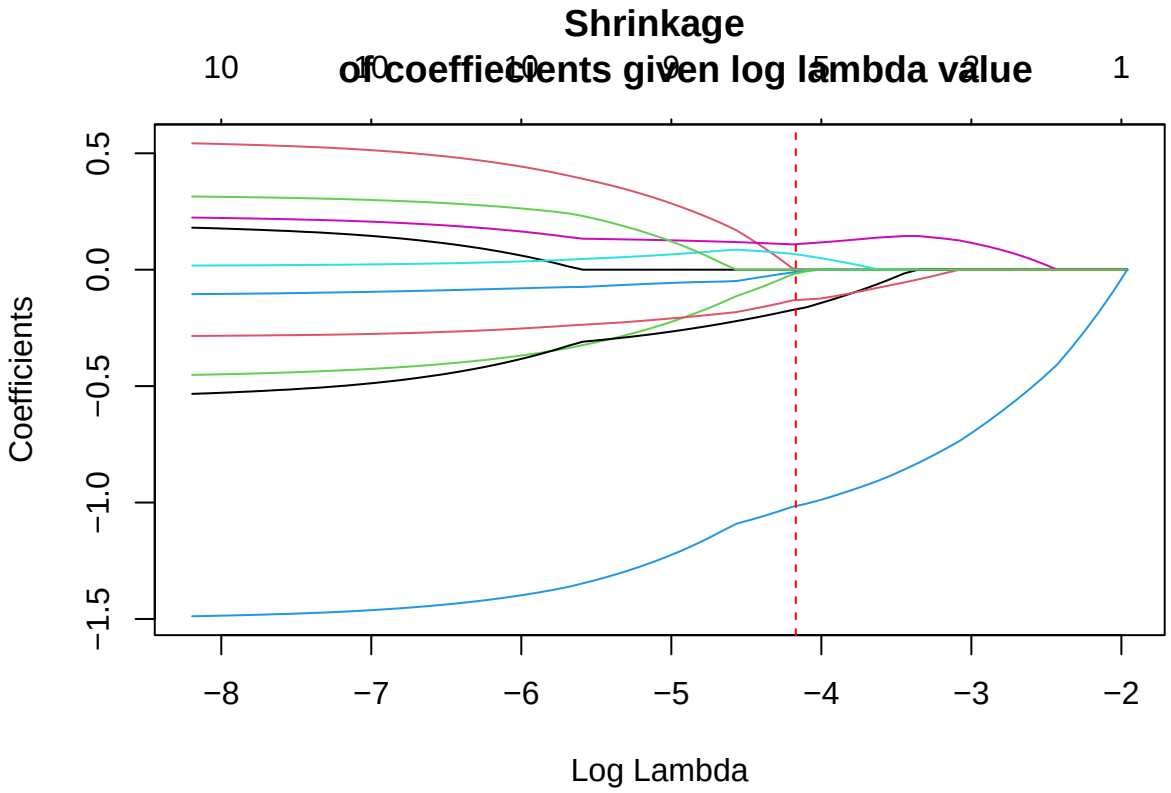
```
coef(model_lasso)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -2.48976903
## R1           0.04269894
## R2           0.42781324
## R3          -0.35636224
## R4          -0.07755878
## R5           0.03821946
## R6           0.15485284
## R7          -0.36123575
## R8          -0.24757103
## R9           0.25504478
## R10         -1.38434189
```

```
plot(model_lasso,main="Misclassification error for
     Lasso Regression given log lambda value")
```




```
z=glmnet(train_for_shrinkage, train$DLRSN,

alpha = 1, family = "binomial", type.measure = "class")
plot(z, xvar="lambda",main="Shrinkage
     of coeffiecients given log lambda value")
abline(v = log(model_ridge$lambda.min), col = "red",lty=2)
```




```
pred_lasso<- predict(model_lasso,test_for_shrinkage,
                     type = "response", s = model_lasso$lambda.min)
pred_lassor<-ifelse(pred_lasso > 0.5, 1, 0)
```

```
#Confusion matrix
c_lasso=table(test$DLRSN, pred_lassor)
c_lasso
```

```
##    pred_lassor
##        0    1
##   0 1123   28
##   1  132   76
```

Accuracy:

```
accuracy_lasso = (c_lasso[1,1]+c_lasso[2,2])/nrow(test)
accuracy_lasso
```

```
## [1] 0.8822664
```

Sensitivity:

```
sensitivity_lasso = c_lasso[2,2]/(c_lasso[2,2]+c_lasso[2,1])
sensitivity_lasso
```

```
## [1] 0.3653846
```

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a statistical technique used for classification, assuming that the predictor variables have a linear relationship with the class labels. It relies on Bayesian inference principles to

find a linear combination of predictors that maximizes class separation, known as the discriminant function. This function is then used to classify new observations based on their discriminant scores.

LDA has several assumptions for reliable results:

1. Linearly separated classes: LDA requires that the classes can be separated by linear boundaries.

2. Multivariate normality: LDA performs better than logistic regression when the sample size is small and predictor distributions in each class are approximately normal.

3. Homoscedasticity: It assumes that all covariates have the same variance.

4. Independence of predictors: LDA works best with independent predictor variables, which is not an issue in our case as the predictors show no multicollinearity.

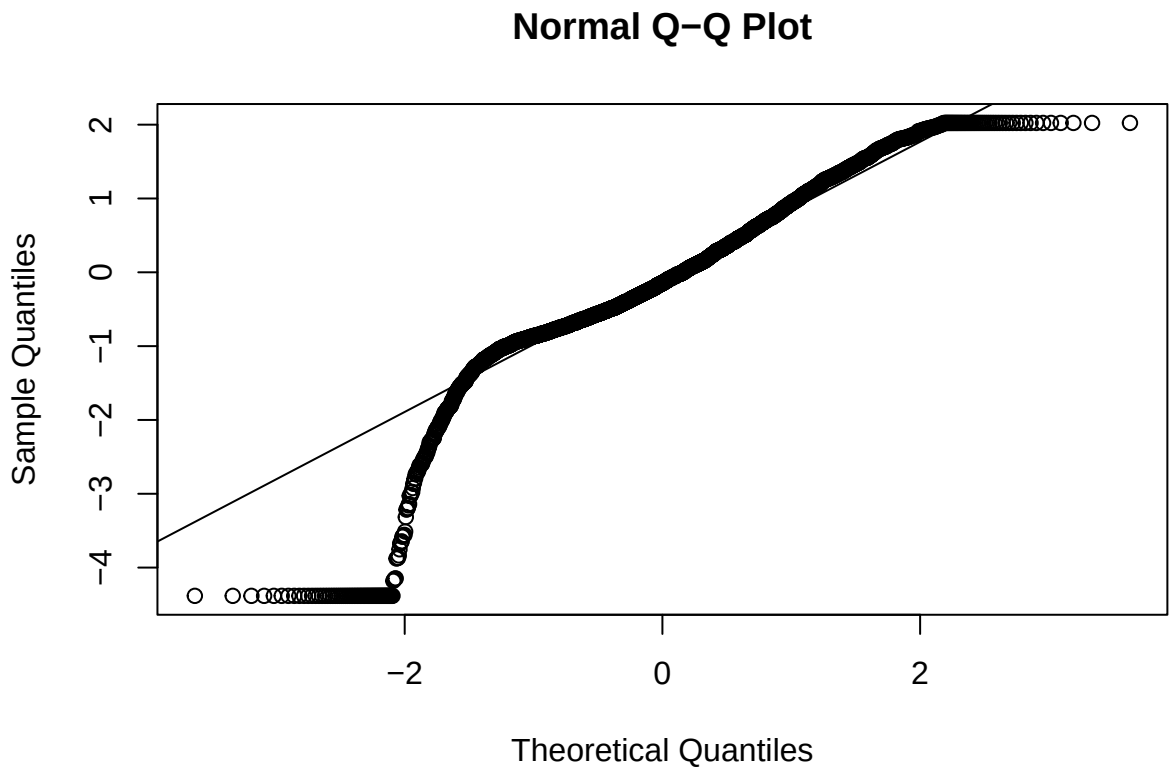
However, LDA is sensitive to outliers, so it's essential to address any potential outliers during the analysis process.

Normality check

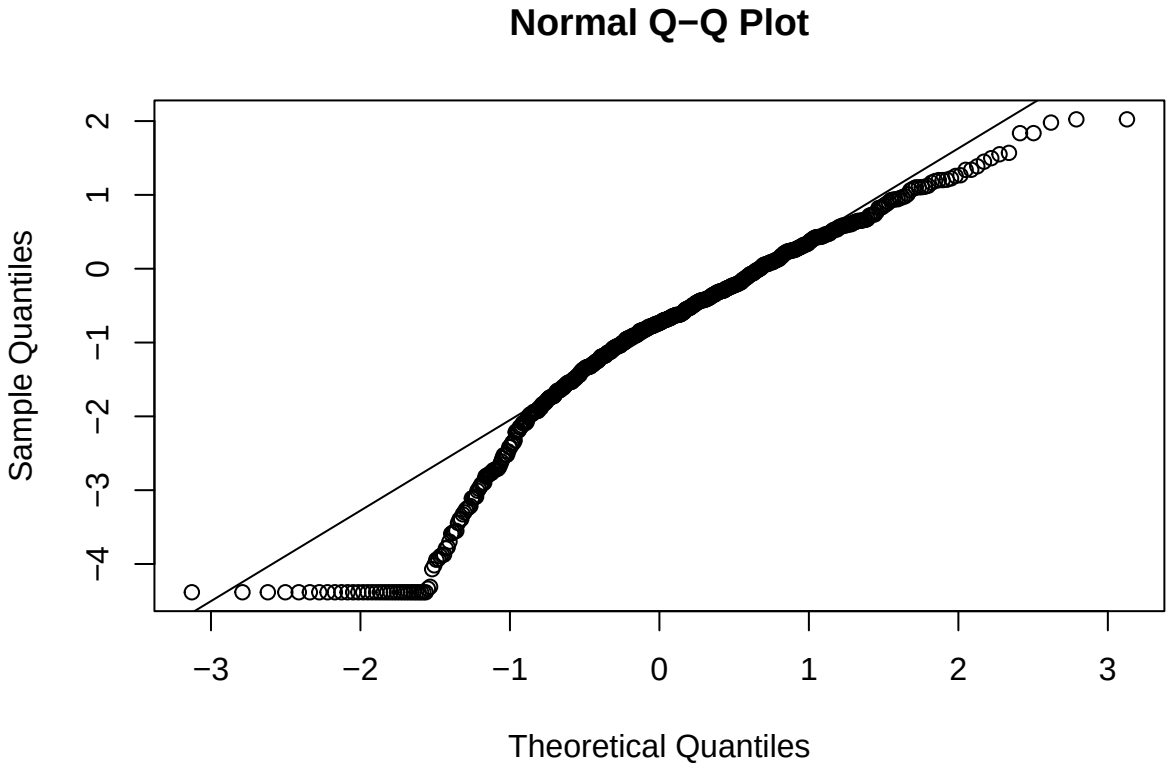
By employing Q-Q plots, we assess the normality of predictor distributions within each "DLRSN" class. This graphical technique allows us to visually compare the quantiles of the data against the theoretical quantiles of a normal distribution. A roughly straight line on the Q-Q plot suggests approximate normality, aiding in the evaluation of the underlying data distribution's similarity to a normal distribution for each class.

- **"R1"**:

```
qqnorm(train$R1[train$DLRSN==0])
qqline(train$R1[train$DLRSN==0])
```



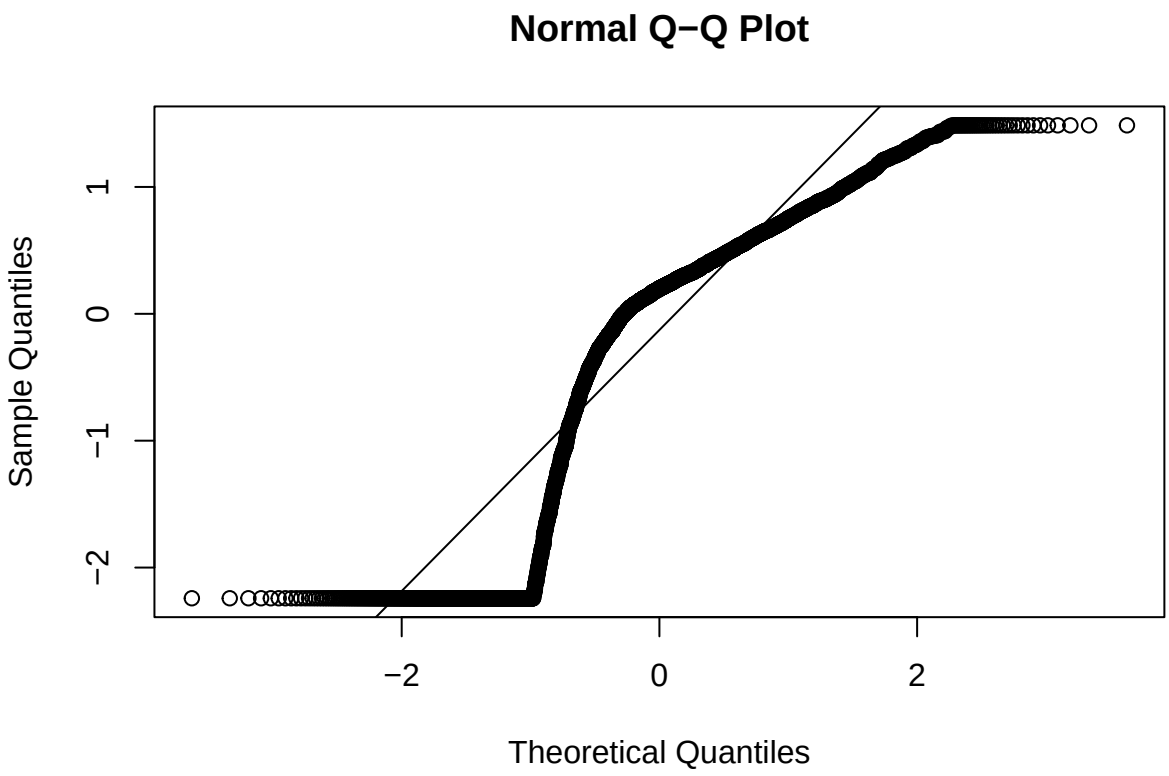

```
qqnorm(train$R1[train$DLRSN==1])
qqline(train$R1[train$DLRSN==1])
```



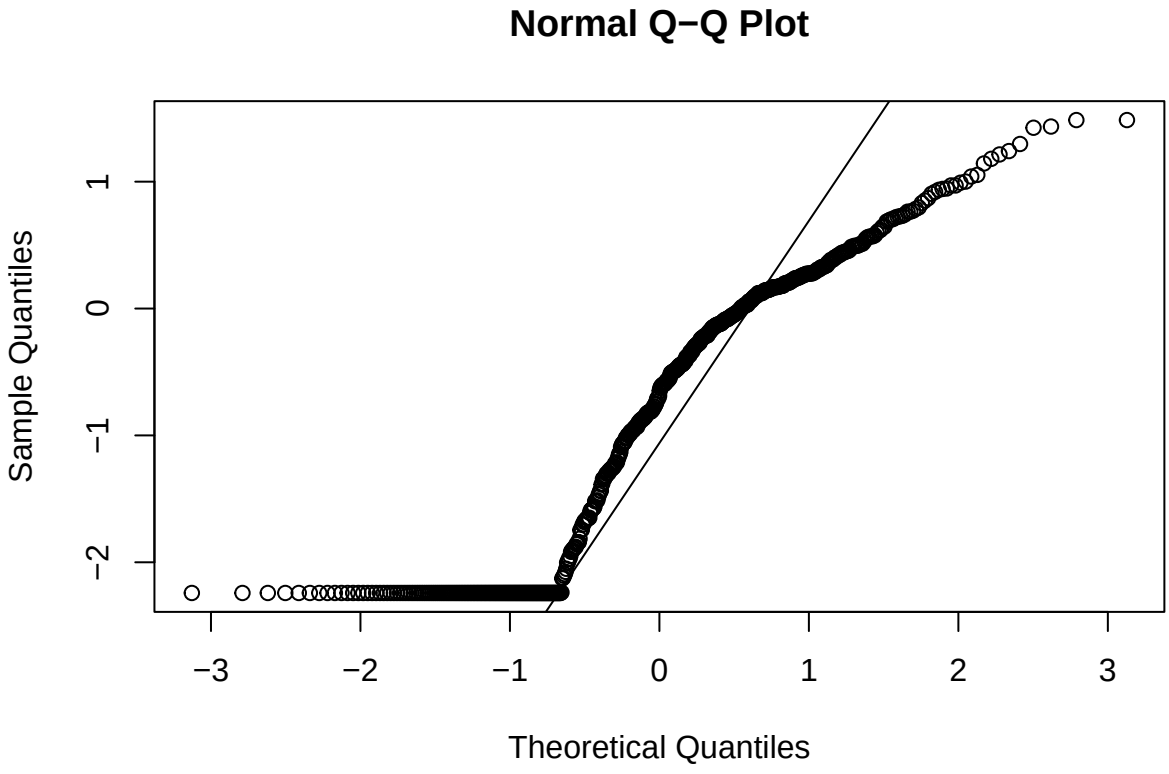
Regarding the variable "R1", it seems to follow approximately the Normal distribution, except for the values on the left we can roughly regard the distribution as normal.

- **"R2"**:

```
qqnorm(train$R2[train$DLRSN==0])
qqline(train$R2[train$DLRSN==0])
```



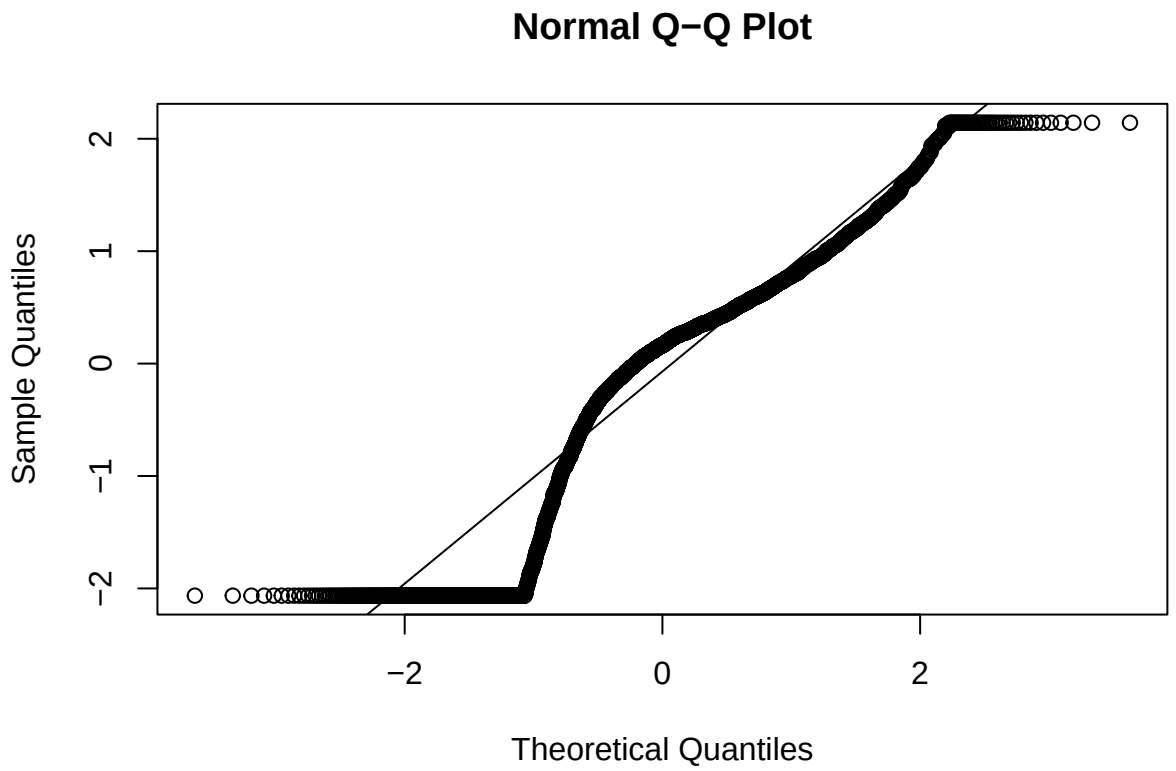

```
qqnorm(train$R2[train$DLRSN==1])
qqline(train$R2[train$DLRSN==1])
```



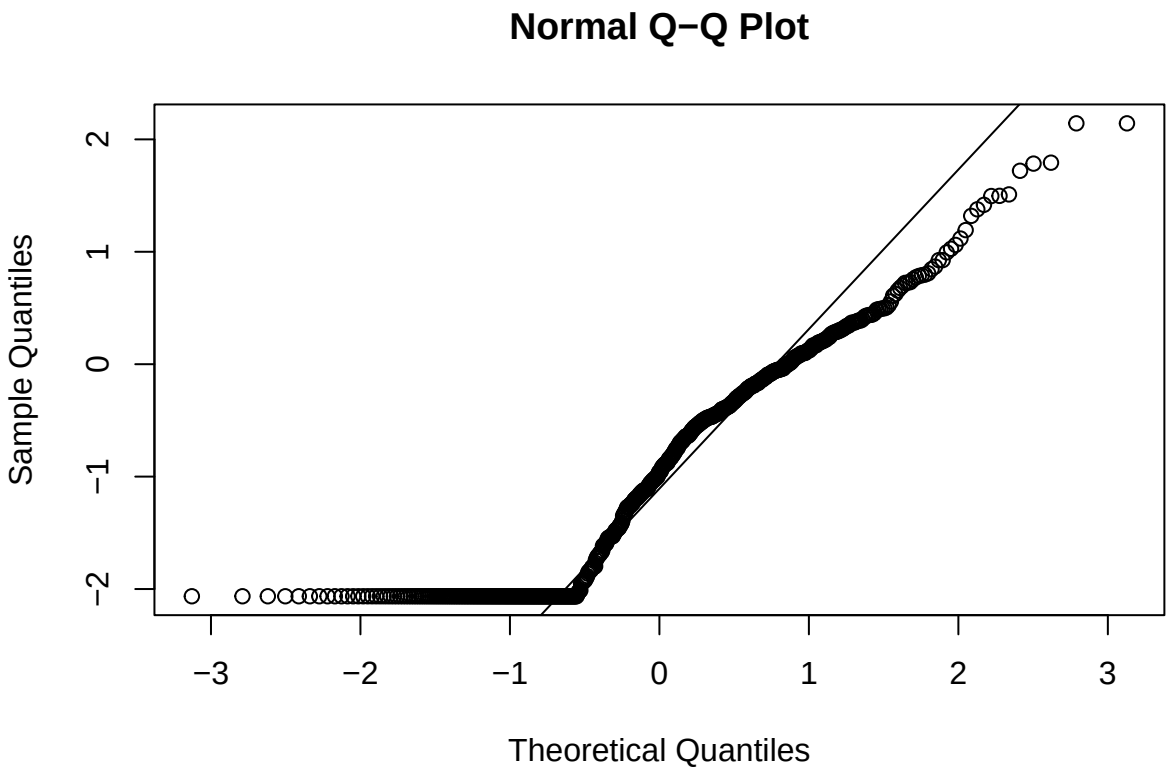
For "R2"'s distribution the same argument made for R1 applies.

- **"R3**":

```
qqnorm(train$R3[train$DLRSN==0])
qqline(train$R3[train$DLRSN==0])
```



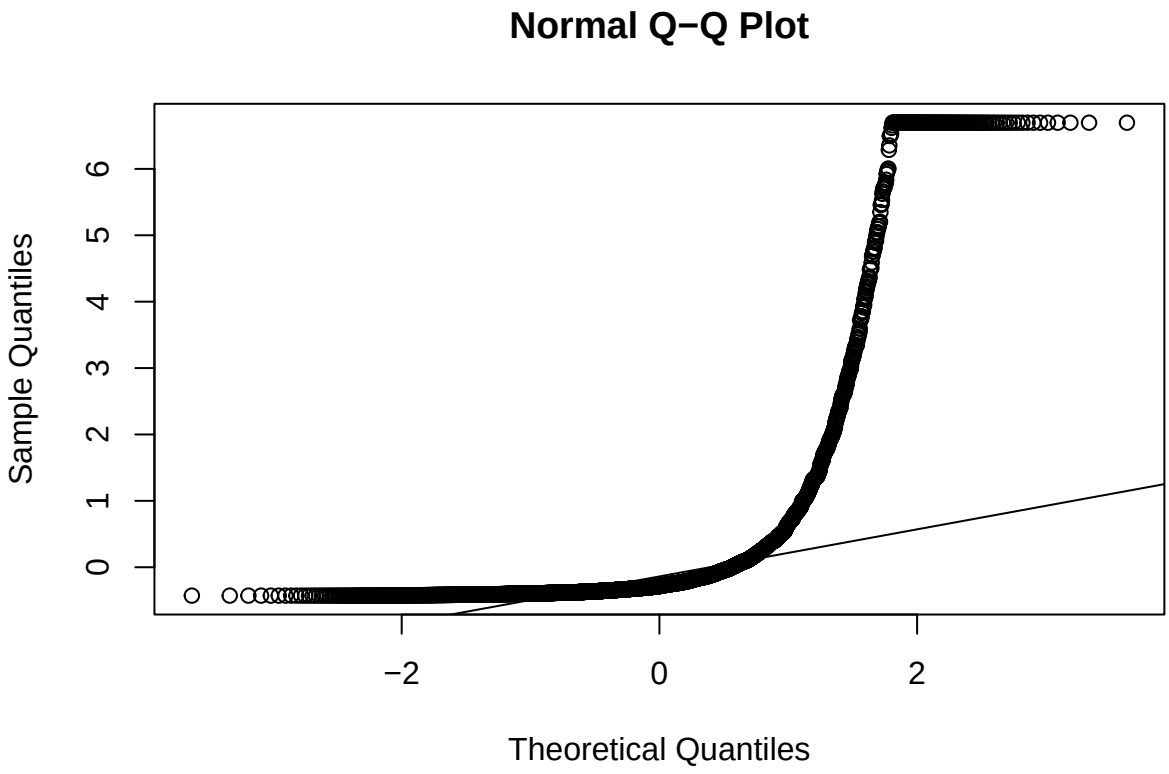

```
qqnorm(train$R3[train$DLRSN==1])
qqline(train$R3[train$DLRSN==1])
```



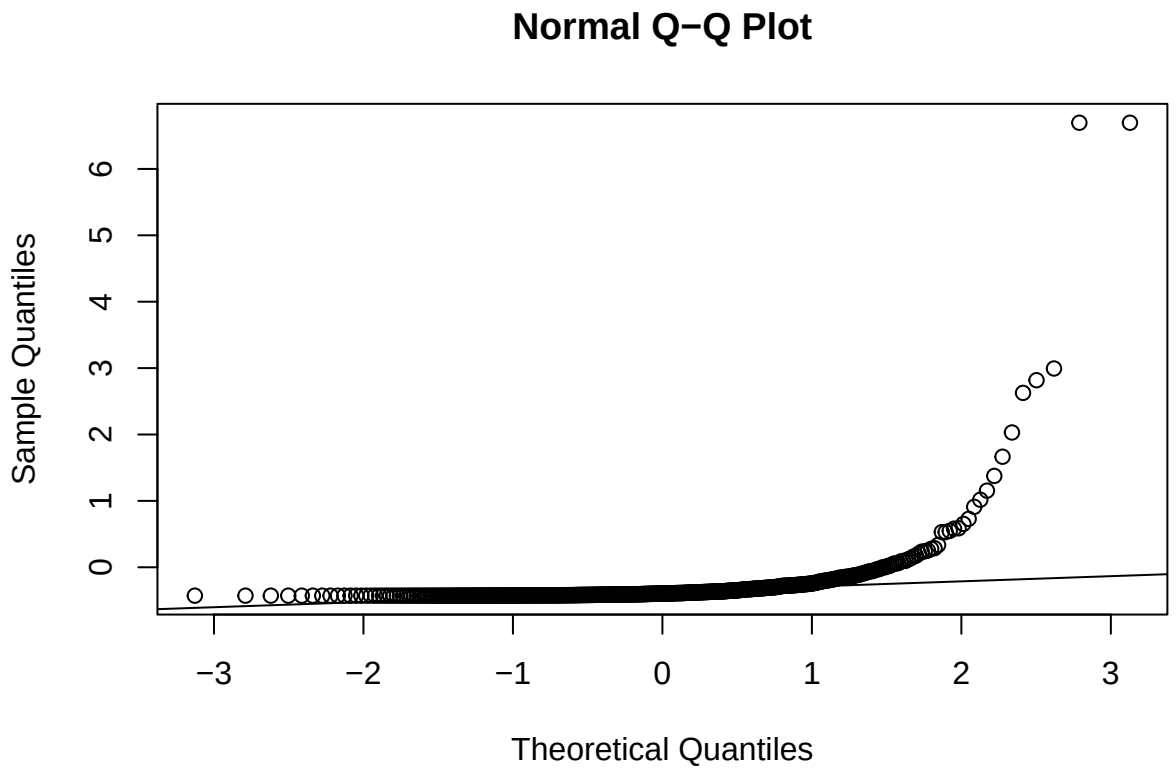
The variable "R3" also has a distribution approximately close to the Normal.

- **"R4"**:

```
qqnorm(train$R4[train$DLRSN==0])
qqline(train$R4[train$DLRSN==0])
```



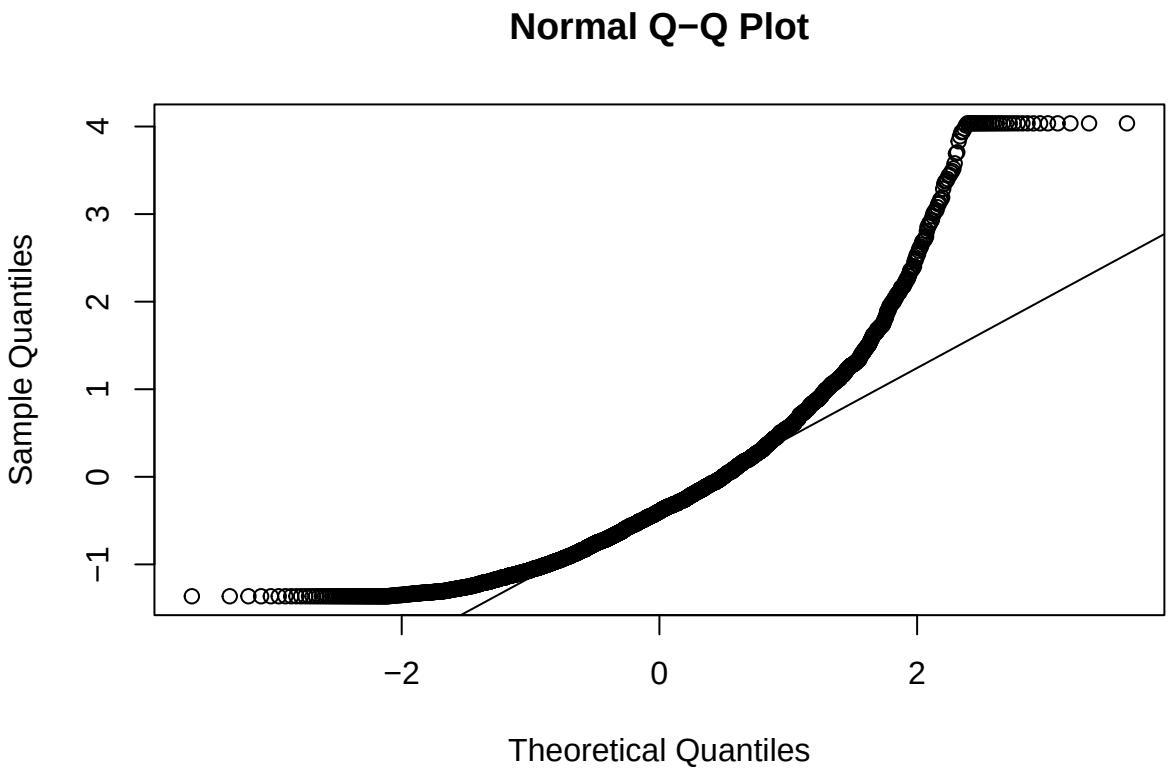
```
qqnorm(train$R4[train$DLRSN==1])
qqline(train$R4[train$DLRSN==1])
```



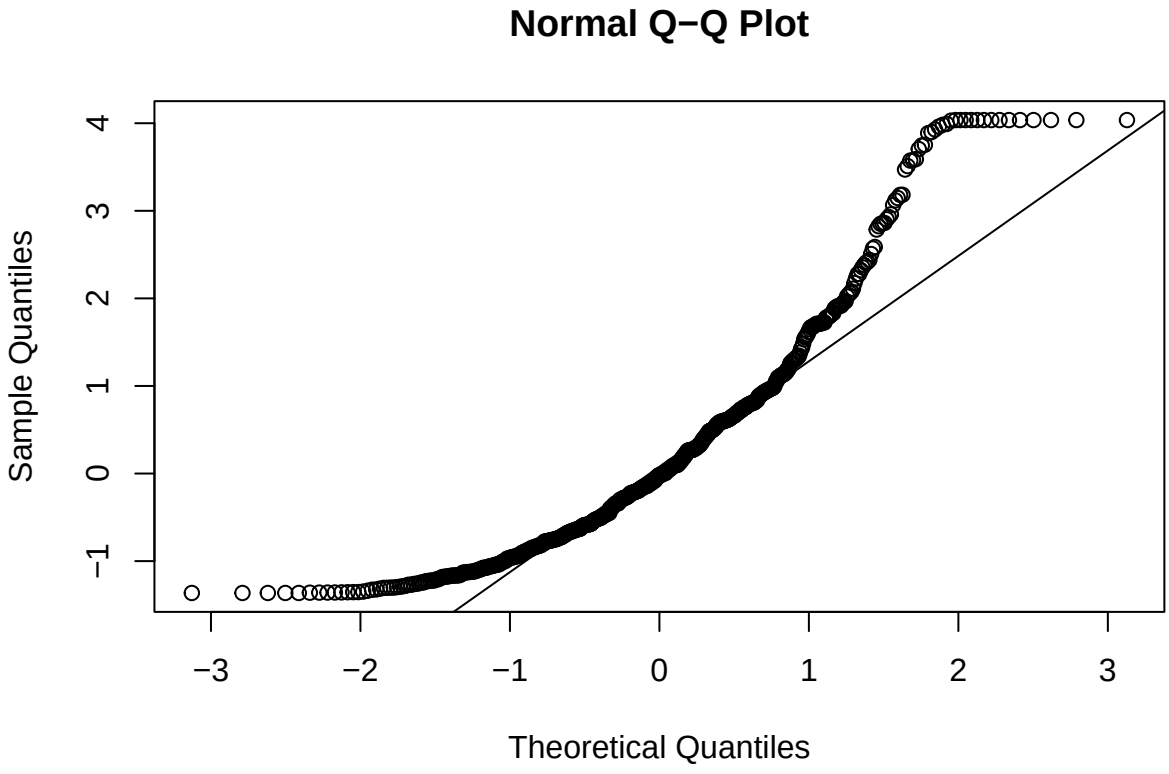
The variable "R4" does not fit the Normal distribution very well, but it seems to be the only one so far.

- **"R5"**:

```
qqnorm(train$R5[train$DLRSN==0])
qqline(train$R5[train$DLRSN==0])
```



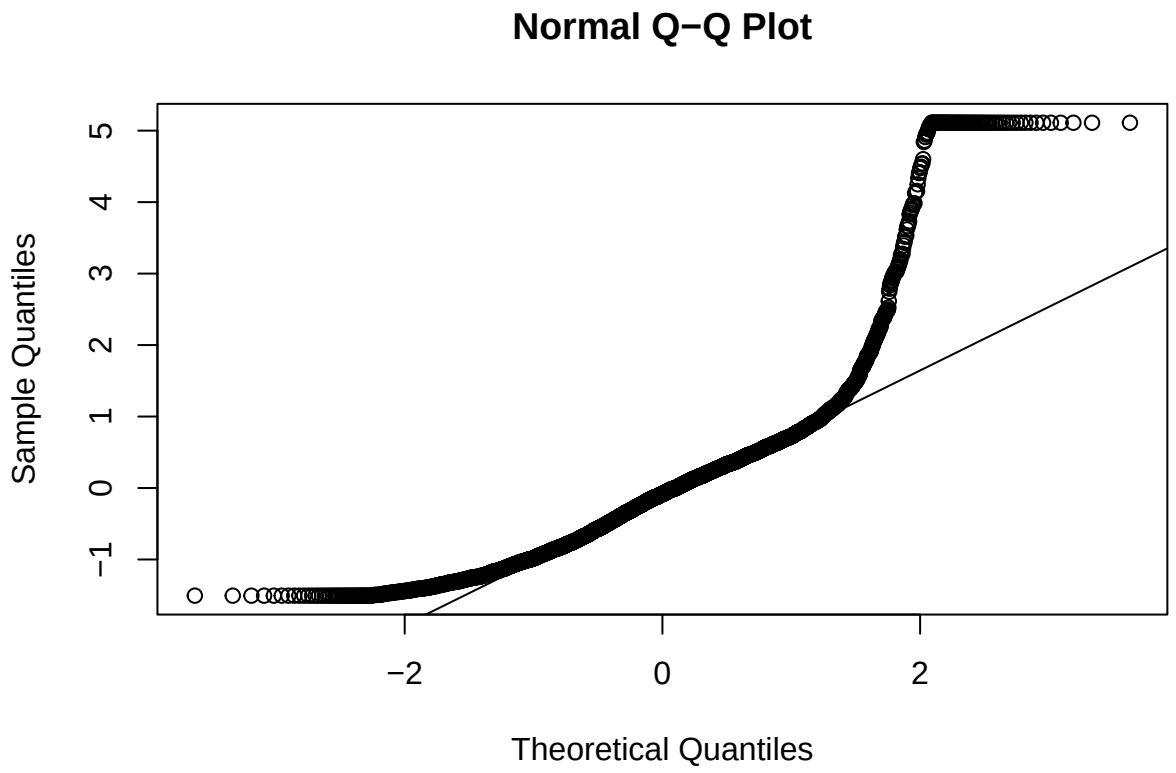

```
qqnorm(train$R5[train$DLRSN==1])
qqline(train$R5[train$DLRSN==1])
```



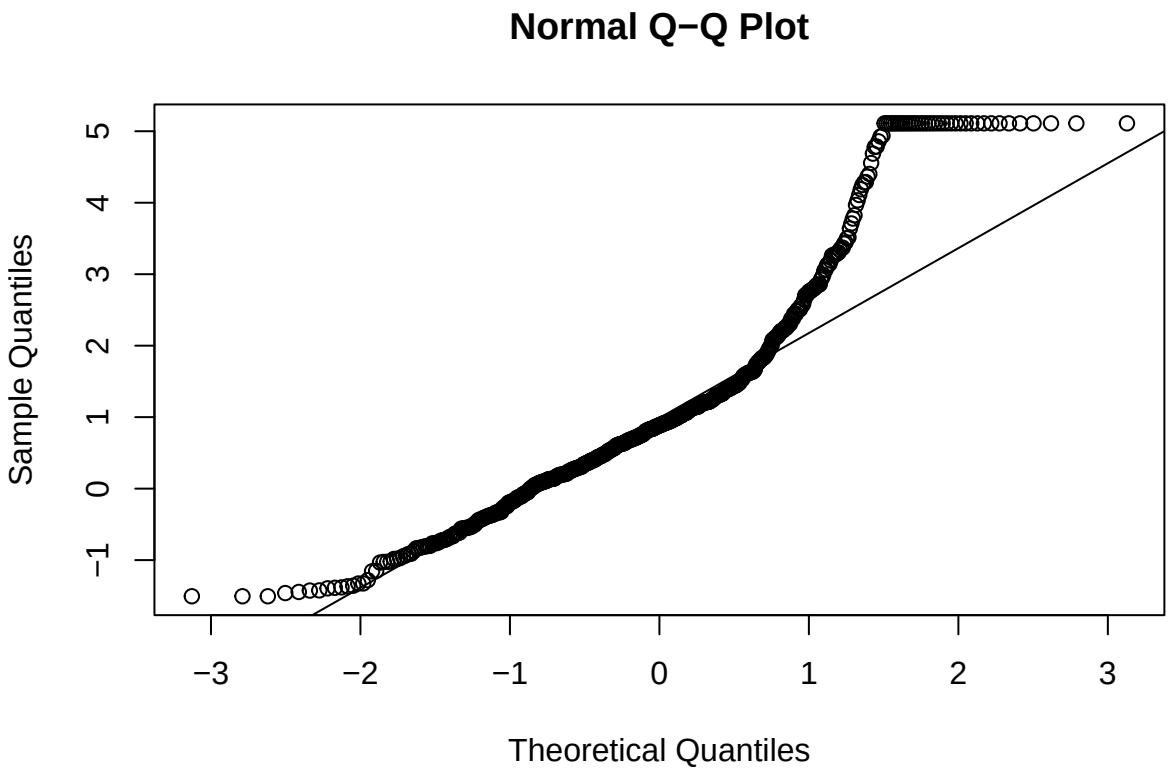
The variable"R5" has a distribution that differs somewhat from the normal distribution.

- **"R6"**:

```
qqnorm(train$R6[train$DLRSN==0])
qqline(train$R6[train$DLRSN==0])
```



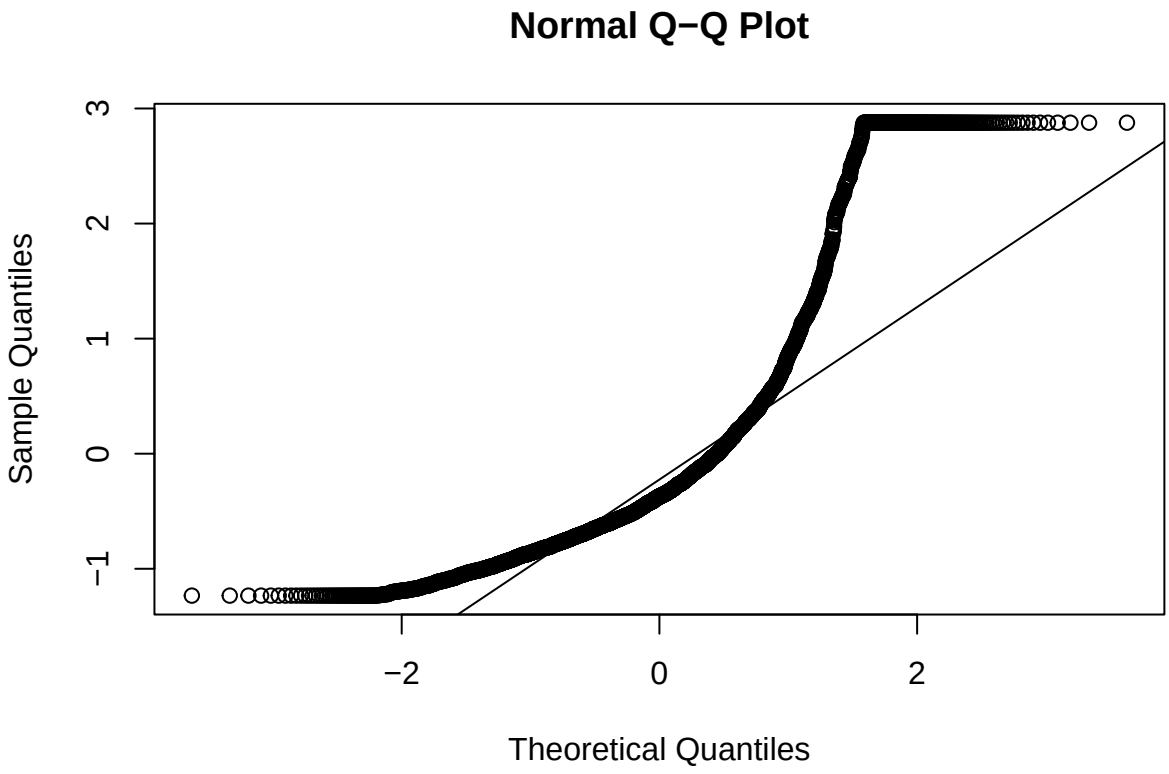

```
qqnorm(train$R6[train$DLRSN==1])
qqline(train$R6[train$DLRSN==1])
```



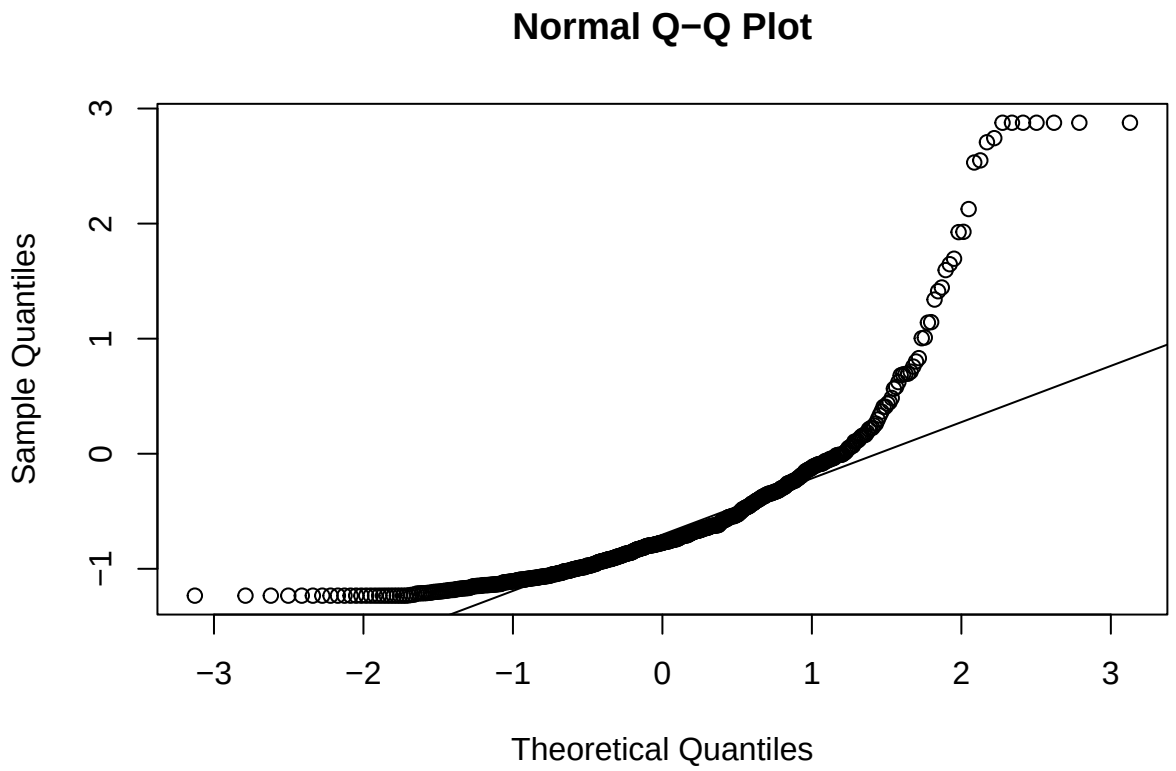
The variable "R6" differs from a normal distribution

- **"R7"**:

```
qqnorm(train$R7[train$DLRSN==0])
qqline(train$R7[train$DLRSN==0])
```



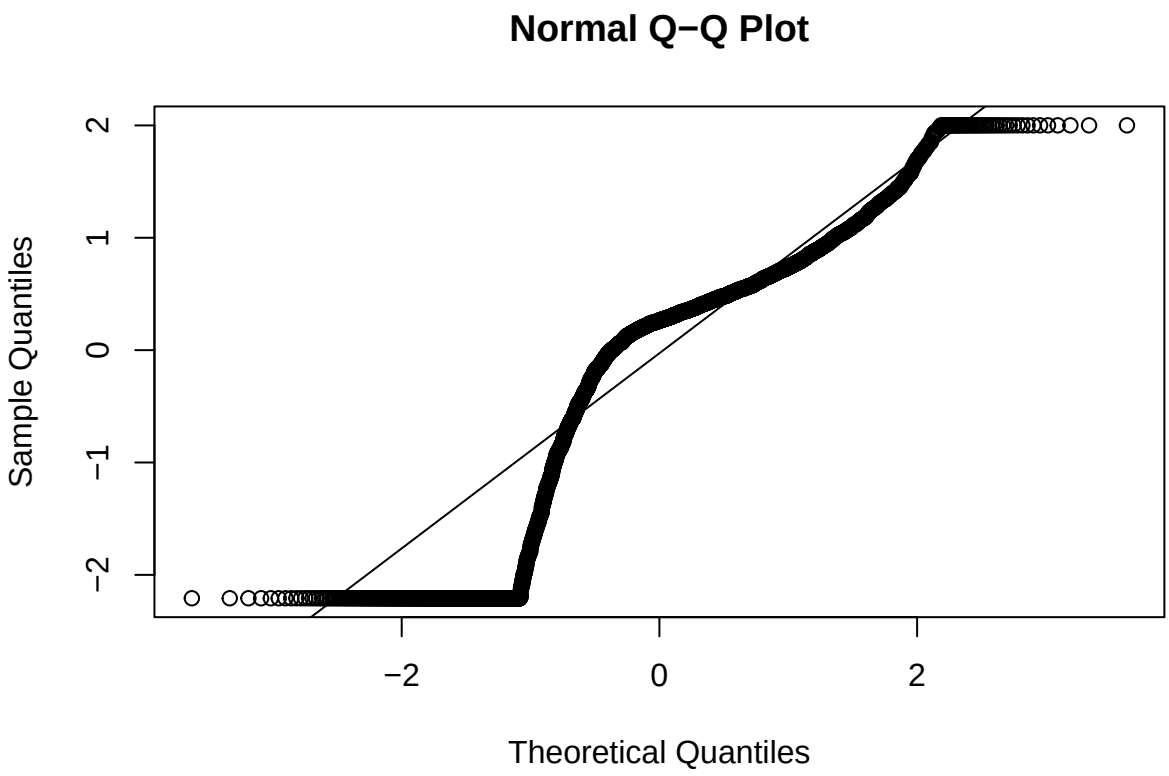
```
qqnorm(train$R7[train$DLRSN==1])
qqline(train$R7[train$DLRSN==1])
```



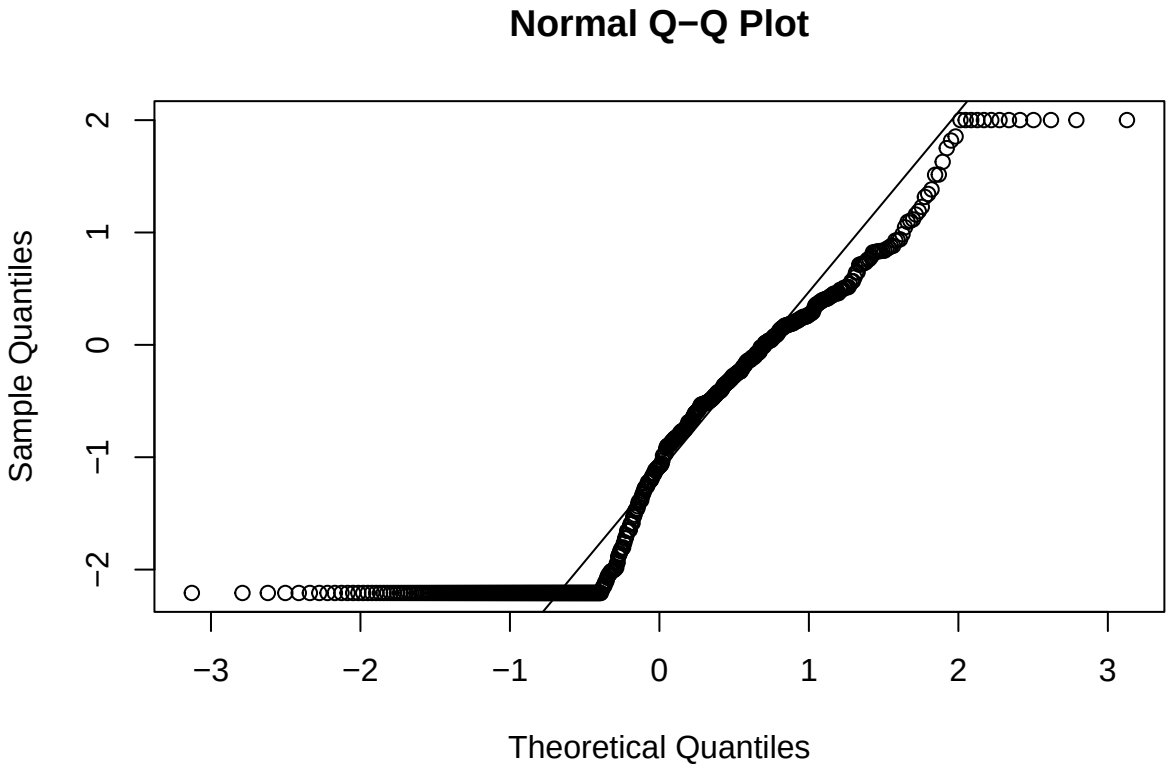
Same as "R6" distribution.

- **"R8"**:

```
qqnorm(train$R8[train$DLRSN==0])
qqline(train$R8[train$DLRSN==0])
```



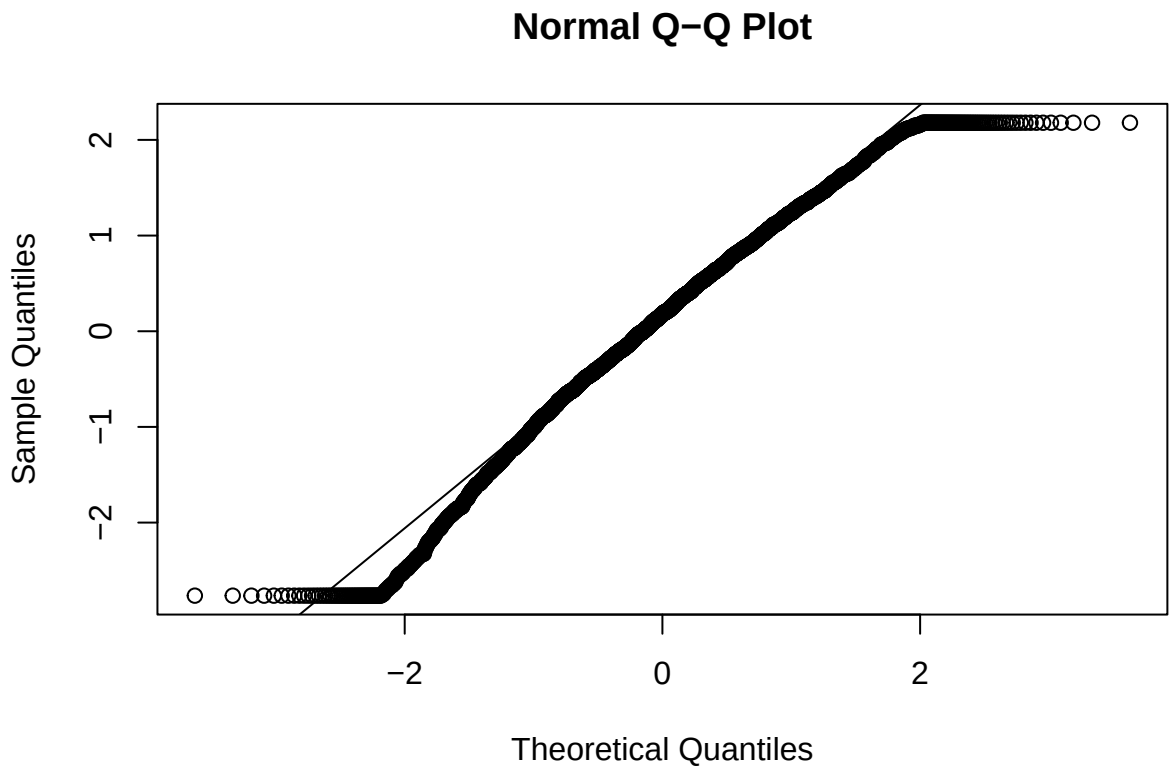

```
qqnorm(train$R8[train$DLRSN==1])
qqline(train$R8[train$DLRSN==1])
```



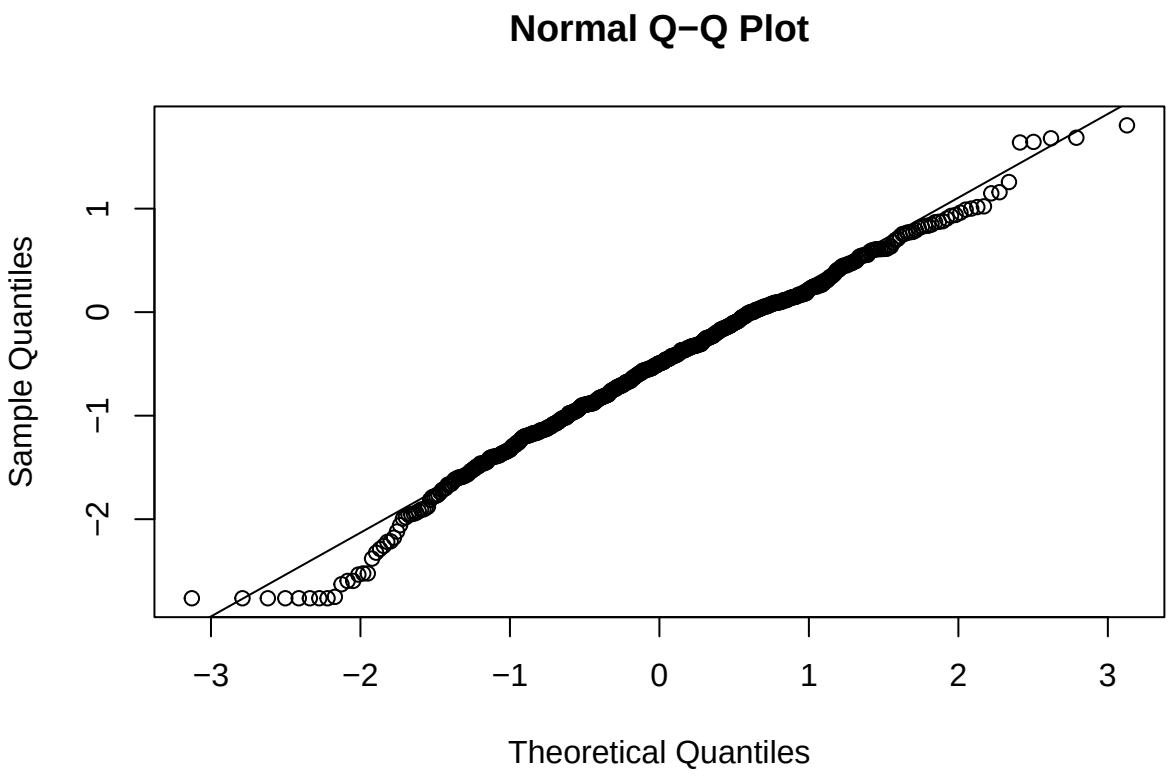
The distribution of "R8" roughly follows a normal distribution.

- **"R9"**:

```
qqnorm(train$R9[train$DLRSN==0])
qqline(train$R9[train$DLRSN==0])
```

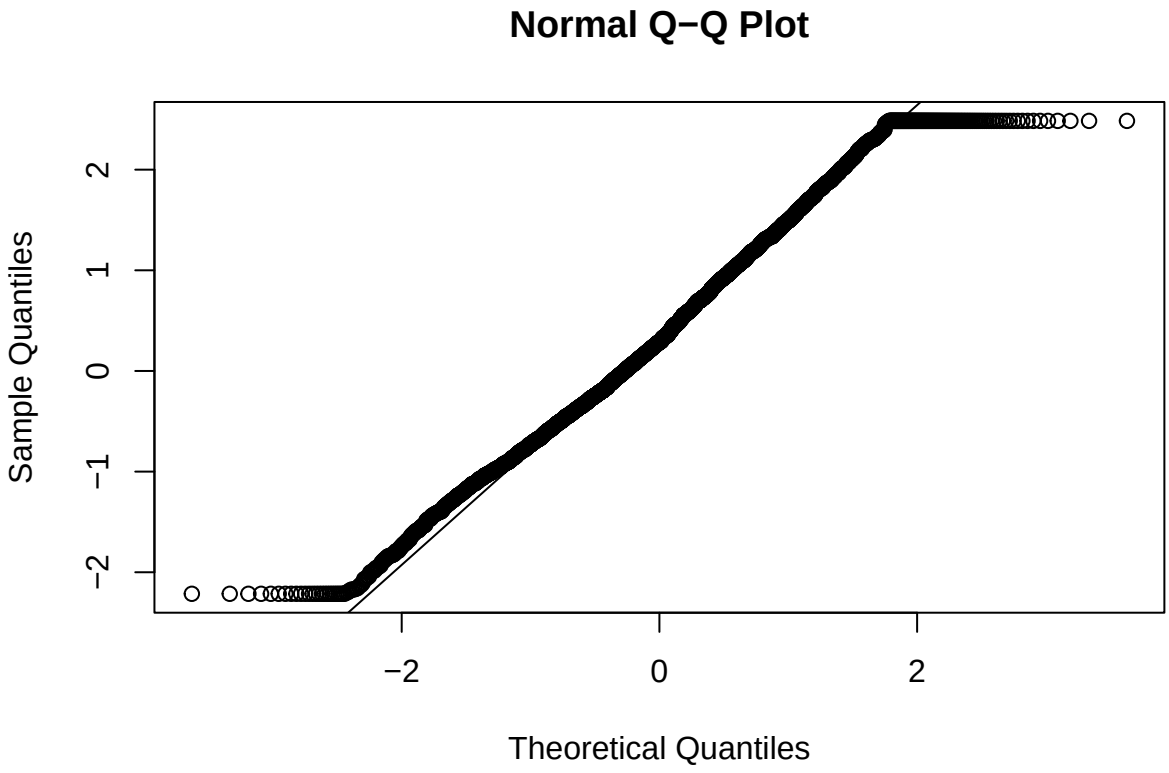



```
qqnorm(train$R9[train$DLRSN==1])
qqline(train$R9[train$DLRSN==1])
```

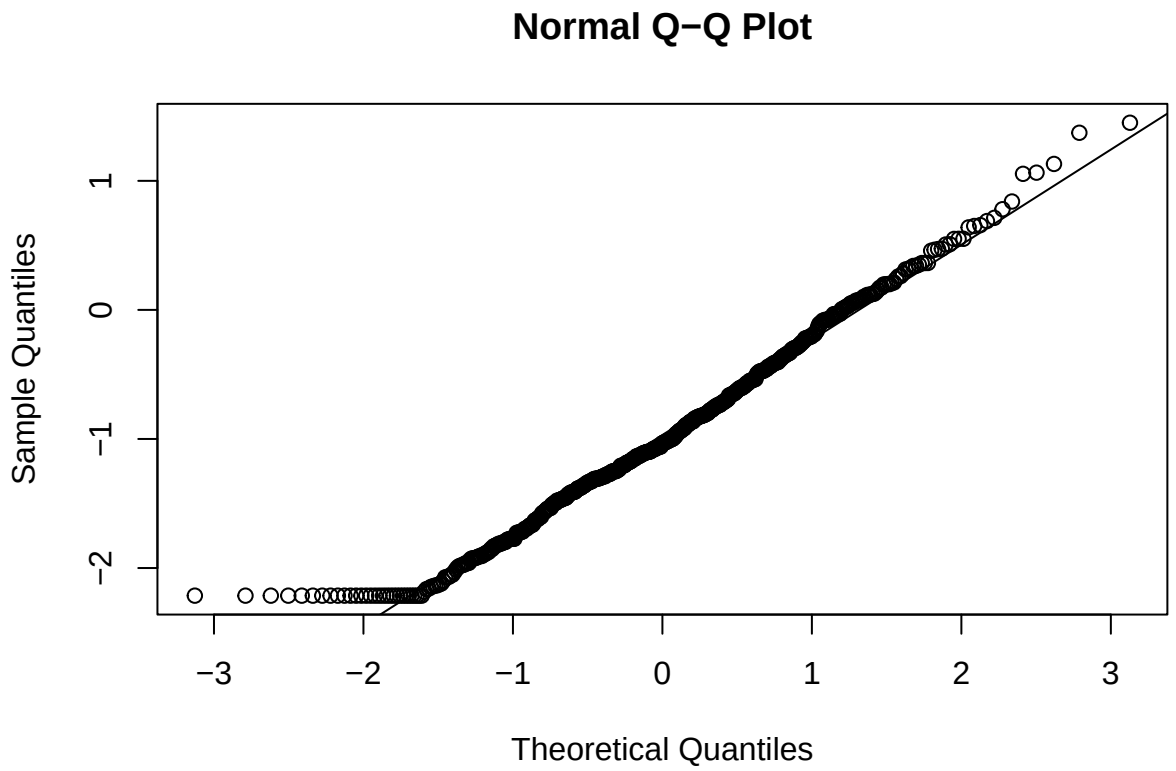


- **"R10"**:

```
qqnorm(train$R10[train$DLRSN==0])
qqline(train$R10[train$DLRSN==0])
```



```
qqnorm(train$R10[train$DLRSN==1])
qqline(train$R10[train$DLRSN==1])
```



The distributions of R9 and R10 fit perfectly with the normal distribution.

Like "R4", "R6" and "R7" also does not fit the Gaussian curve optimally; but, overall, we can be satisfied with the multivariate normality concerning our variables.

Outliers Detection and Elimination

As mentioned earlier, it is crucial to examine the presence of outliers as they can impact the results of LDA. To detect outliers, we employ the interquartile rule (IQR), which is based on the interquartile range: the difference between the third quartile (Q3) and the first quartile (Q1), representing a measure of data dispersion. Outliers are identified as data points that fall above or below a specific range relative to the IQR.

To address outliers, we first create a new training set as a duplicate of the original. This new training set will be modified whenever outliers are detected.

```
#Training set without outliers
train_wo_out = train
```

Next, IQR is applied to identify any outliers contained within our predictors. Once detected, they are excluded from "*train_wo_out*".

```
Q1_R1 <- quantile(train_wo_out$R1, 0.25)
Q3_R1 <- quantile(train_wo_out$R1, 0.75)

IQR_R1 = Q3_R1-Q1_R1
inf_lim_R1 <- Q1_R1 - 1.5 * IQR_R1
sup_lim_R1 <- Q3_R1 + 1.5 * IQR_R1

outliers_R1 <- train_wo_out$DLRSN[train_wo_out$R1 <
```

```
                                          inf_lim_R1 |
                                          train_wo_out$R1 > sup_lim_R1]
length(outliers_R1)
```

```
## [1] 190
```

```
train_wo_out = train_wo_out[train_wo_out$R1
                            >= inf_lim_R1 &
                              train_wo_out$R1
                            <= sup_lim_R1,]
```

```
Q1_R2 <- quantile(train_wo_out$R2, 0.25)
Q3_R2 <- quantile(train_wo_out$R2, 0.75)

IQR_R2 = Q3_R2-Q1_R2
inf_lim_R2 <- Q1_R2 - 1.5 * IQR_R2
sup_lim_R2 <- Q3_R2 + 1.5 * IQR_R2

outliers_R2 <- train_wo_out$DLRSN[train_wo_out$R2 <
                                          inf_lim_R2 |
                                          train_wo_out$R2 > sup_lim_R2]
length(outliers_R2)
```

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

```
train_wo_out = train_wo_out[train_wo_out$R2
                            >= inf_lim_R2 &
                              train_wo_out$R2
                            <= sup_lim_R2,]
```

```
Q1_bp <- quantile(train_wo_out$R3, 0.25)
Q3_bp <- quantile(train_wo_out$R3, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$R3[train_wo_out$R3 <
                                          inf_lim_bp |
                                          train_wo_out$R3 > sup_lim_bp]
length(outliers_bp)
```

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

```
train_wo_out = train_wo_out[train_wo_out$R3
                            >= inf_lim_bp &
                              train_wo_out$R3
                            <= sup_lim_bp,]
```

```
Q1_bp <- quantile(train_wo_out$R4, 0.25)
Q3_bp <- quantile(train_wo_out$R4, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$R4[train_wo_out$R4 <
                                          inf_lim_bp |
```

```
                                          train_wo_out$R4 > sup_lim_bp]
length(outliers_bp)
```

```
## [1] 553
```

```
train_wo_out = train_wo_out[train_wo_out$R4
                            >= inf_lim_bp &
                              train_wo_out$R4
                            <= sup_lim_bp,]
```

```
Q1_bp <- quantile(train_wo_out$R5, 0.25)
Q3_bp <- quantile(train_wo_out$R5, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$R5[train_wo_out$R5 <
                                          inf_lim_bp |
                                          train_wo_out$R5 > sup_lim_bp]
length(outliers_bp)
```

```
## [1] 140
```

```
train_wo_out = train_wo_out[train_wo_out$R5
                            >= inf_lim_bp &
                              train_wo_out$R5
                            <= sup_lim_bp,]
```

```
Q1_bp <- quantile(train_wo_out$R6, 0.25)
Q3_bp <- quantile(train_wo_out$R6, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$R6[train_wo_out$R6 <
                                          inf_lim_bp |
                                          train_wo_out$R6 > sup_lim_bp]
length(outliers_bp)
```

```
## [1] 117
```

```
train_wo_out = train_wo_out[train_wo_out$R6
                            >= inf_lim_bp &
                              train_wo_out$R6
                            <= sup_lim_bp,]
```

```
Q1_bp <- quantile(train_wo_out$R7, 0.25)
Q3_bp <- quantile(train_wo_out$R7, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$R3[train_wo_out$R7 <
                                          inf_lim_bp |
                                          train_wo_out$R7 > sup_lim_bp]
length(outliers_bp)
```

```
## [1] 197
```

```
train_wo_out = train_wo_out[train_wo_out$R7
                            >= inf_lim_bp &
                              train_wo_out$R7
                            <= sup_lim_bp,]
```

```
Q1_bp <- quantile(train_wo_out$R8, 0.25)
Q3_bp <- quantile(train_wo_out$R8, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$R8[train_wo_out$R8 <
                                          inf_lim_bp |
                                          train_wo_out$R8 > sup_lim_bp]
length(outliers_bp)
```

```
## [1] 383
```

```
train_wo_out = train_wo_out[train_wo_out$R8
                            >= inf_lim_bp &
                              train_wo_out$R8
                            <= sup_lim_bp,]
```

```
Q1_bp <- quantile(train_wo_out$R9, 0.25)
Q3_bp <- quantile(train_wo_out$R9, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$R9[train_wo_out$R9 <
                                          inf_lim_bp |
                                          train_wo_out$R9 > sup_lim_bp]
length(outliers_bp)
```

```
## [1] 11
```

```
train_wo_out = train_wo_out[train_wo_out$R9
                            >= inf_lim_bp &
                              train_wo_out$R9
                            <= sup_lim_bp,]
```

```
Q1_bp <- quantile(train_wo_out$R10, 0.25)
Q3_bp <- quantile(train_wo_out$R10, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$R10[train_wo_out$R10 <
                                          inf_lim_bp |
                                          train_wo_out$R10 > sup_lim_bp]
length(outliers_bp)
```

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

```
train_wo_out = train_wo_out[train_wo_out$R10
                            >= inf_lim_bp &
                              train_wo_out$R10
                            <= sup_lim_bp,]
```

In this way we identified outliers for variables R1, R4, R5, R6 ,R7 and R8, as we saw earlier during the univariate analysis of the variables through boxplots

```
nrow(train)
```

```
## [1] 4077
```

```
nrow(train_wo_out)
```

```
## [1] 2486
```

"train_wo_out" turns out to be less than the initial training set of 1591 observations after the outliers are excluded.

LDA Procedure

To perform Linear Discriminant Analysis, we use the `lda` function built into the `MASS` package:

```
library(MASS)
```

```
lda_model <- lda(train_wo_out$DLRSN~ . ,data=train_wo_out, family="binomial")
lda_model
```

```
## Call:
## lda(train_wo_out$DLRSN ~ ., data = train_wo_out, family = "binomial")
##
## Prior probabilities of groups:
##          0          1
## 0.90144811 0.09855189
##
## Group means:
##           R1         R2         R3         R4         R5         R6         R7
## 0 -0.2200323  0.1985141  0.2685574 -0.2526546 -0.1811433 0.03655333 -0.3717949
## 1 -0.4176232 -0.1451270 -0.3133331 -0.3436484 -0.0252586 0.47429074 -0.5484844
##           R8         R9        R10
## 0  0.2908955  0.5400001  0.5263459
## 1 -0.2067870 -0.2961650 -0.7589940
##
## Coefficients of linear discriminants:
##             LD1
## R1  -0.04032582
## R2   0.63755778
## R3  -0.59711462
## R4   1.61387395
## R5   0.14291014
## R6   1.04002201
## R7  -0.28352263
## R8  -0.15044479
## R9  -0.22789214
## R10 -0.72640355
```

The output provides a single coefficient for the LD1 discriminant direction for each predictor variable. These

coefficients indicate the relative importance of each variable in separating the classes. Interpreting the coefficients:

R1: The variable R1 has a coefficient of -0.0403. A negative coefficient suggests that higher values of R1 are associated with a lower probability of belonging to the positive class (Group 1) in the LDA model.

R2: The variable R2 has a coefficient of 0.6376. A positive coefficient indicates that higher values of R2 are associated with a higher probability of belonging to the positive class in the LDA model.

R3: The variable R3 has a coefficient of -0.5971. A negative coefficient suggests that higher values of R3 are associated with a lower probability of belonging to the positive class.

R4: The variable R4 has a coefficient of 1.6139. A positive coefficient indicates that higher values of R4 are associated with a higher probability of belonging to the positive class.

R5: The variable R5 has a coefficient of 0.1429. A positive coefficient suggests that higher values of R5 are associated with a slightly higher probability of belonging to the positive class.

R6: The variable R6 has a coefficient of 1.0400. A positive coefficient indicates that higher values of R6 are associated with a higher probability of belonging to the positive class.

R7: The variable R7 has a coefficient of -0.2835. A negative coefficient suggests that higher values of R7 are associated with a lower probability of belonging to the positive class.

R8: The variable R8 has a coefficient of -0.1504. A negative coefficient indicates that higher values of R8 are associated with a lower probability of belonging to the positive class.R9: Th e variable R9 has a coefficient of -0.2279. A negative coefficient suggests that higher values of R9 are associated with a lower probability of belonging to the positive class.

R9: The variable R9 has a coefficient of -0.2279. A negative coefficient suggests that higher values of R9 are associated with a lower probability of belonging to the positive class.

R10: The variable R10 has a coefficient of -0.7264. A negative coefficient indicates that higher values of R10 are associated with a lower probability of belonging to the positive class.

Predictions are obtained using the "predict" command, and we focus on the posterior probabilities. For new observations, these probabilities are calculated to determine their most likely class assignment. The posterior probabilities represent the model's level of "confidence" in classifying an observation into a specific group. Higher posterior probabilities indicate a more certain classification, while lower probabilities suggest some uncertainty in the prediction.

```
# Computing predictions:
pred_lda_model<- predict(lda_model, test, type = "response")
post_lda_model<- pred_lda_model$posterior

pred_lda_modelr <- ifelse(post_lda_model[,2] > 0.5, 1, 0)
```

Finally, we calculate the metrics:

```
c_lda = table(test$DLRSN,pred_lda_modelr)
c_lda
```

```
##    pred_lda_modelr
##       0   1
##   0 959 192
##   1  80 128
```

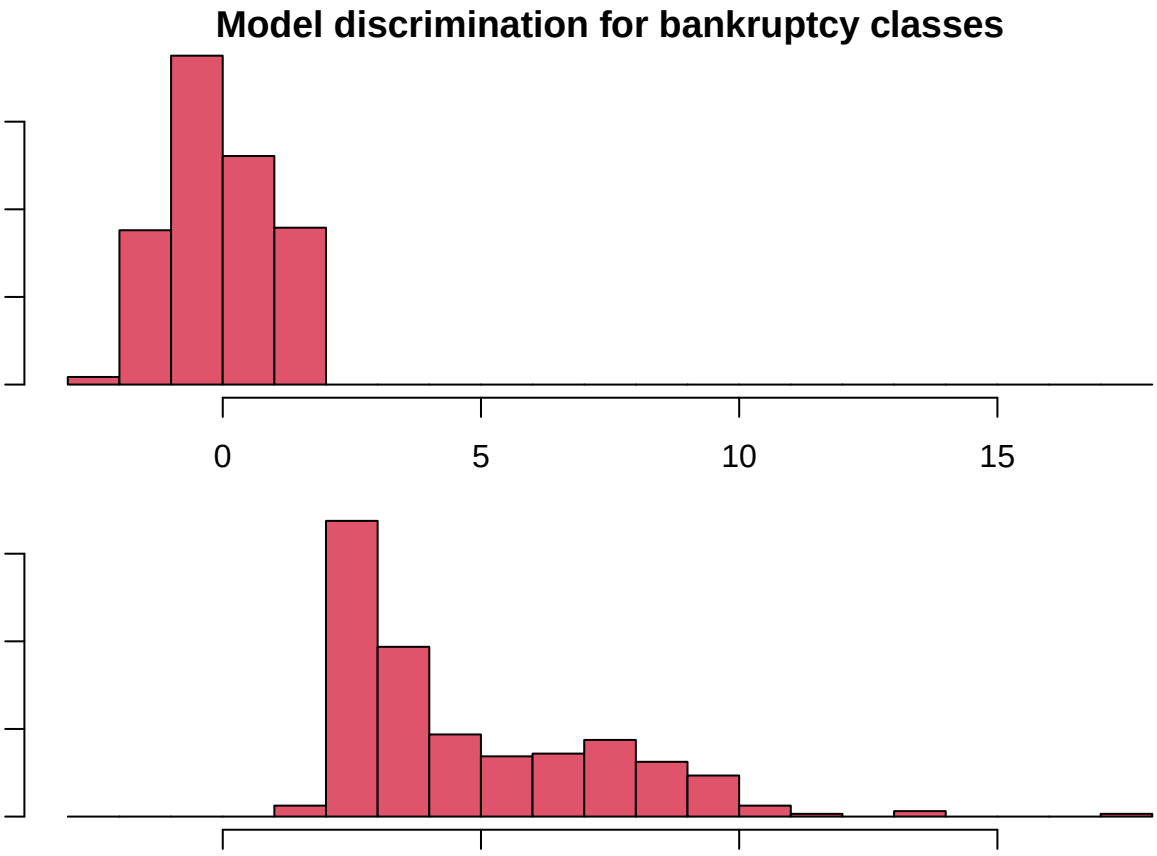
```
# Accuracy:
accuracy_lda = (c_lda[1,1]+c_lda[2,2])/nrow(test)
accuracy_lda
```

```
## [1] 0.7998528
```

```r
# Sensitivity:
sensitivity_lda = c_lda[2,2]/(c_lda[2,2]+c_lda[2,1])

sensitivity_lda
```

```
## [1] 0.6153846
```

```r
par(mar=c(1, 1, 1, 1))
ldahist(pred_lda_model$x[,1], pred_lda_model$class, col=2)
title("Model discrimination for bankruptcy classes")
```



Quadratic Discriminant Analysis

The homoscedasticity assumption of LDA is a significant assumption, and it can be relaxed by using Quadratic Discriminant Analysis (QDA). QDA is an extension of LDA that estimates separate mean and variance matrices for each class. Unlike LDA, QDA calculates a variance matrix for each class, making it more robust to situations with varying variances within classes.

However, QDA is more sensitive to outliers and requires a larger training set. In our analysis, we trained the QDA model on "train_wo_out," which addresses the issue of outliers. QDA provides an alternative when the homoscedasticity assumption is not met and can yield better results when the variance structures among classes differ significantly.

```r
qda_model <- qda(train_wo_out$DLRSN~. ,data=train_wo_out, family="binomial")

qda_model
```

```
## Call:
```

```
## qda(train_wo_out$DLRSN ~ ., data = train_wo_out, family = "binomial")
##
## Prior probabilities of groups:
##          0          1
## 0.90144811 0.09855189
##
## Group means:
##           R1         R2         R3         R4         R5         R6         R7
## 0 -0.2200323  0.1985141  0.2685574 -0.2526546 -0.1811433 0.03655333 -0.3717949
## 1 -0.4176232 -0.1451270 -0.3133331 -0.3436484 -0.0252586 0.47429074 -0.5484844
##           R8         R9        R10
## 0  0.2908955  0.5400001  0.5263459
## 1 -0.2067870 -0.2961650 -0.7589940
```

```
# Computing predictions:
pred_qda_model<- predict(qda_model, test, type = "response")
post_qda_model<- pred_qda_model$posterior

pred_qda_modelr <- ifelse(post_qda_model[,2] > 0.5, 1, 0)
```

```
# Confusion matrix with threshold = 0.3
c_qda = table(test$DLRSN,pred_qda_modelr)
c_qda
```

```
##    pred_qda_modelr
##       0   1
##   0 992 159
##   1  62 146
```

```
# Accuracy:
accuracy_qda = (c_qda[1,1]+c_qda[2,2])/nrow(test)
accuracy_qda
```

```
## [1] 0.8373804
```

```
# Sensitivity:
sensitivity_qda = c_qda[2,2]/(c_qda[2,2]+c_qda[2,1])
sensitivity_qda
```

```
## [1] 0.7019231
```

The metrics computed for QDA result are better than LDA metrics; both for accuracy and sensitivity. QDA assumes that classes have different covariances, while LDA assumes homogeneous covariances across classes. Probably our data are complex and exhibit a nonlinear relationships between predictor variables and the response variable, so QDA may perform better because it considers quadratic relationships between variables

K-NN

K-Nearest Neighbors (KNN) is a non-parametric classification technique that assigns an individual to a specific class based on the majority class among its k-nearest neighbors in the feature space. To ensure optimal performance, it's essential to select an appropriate value for k. Given the data's imbalance, we experiment with different k values up to 100.

If we choose a very high k, it may result in underfitting the data, as it would classify all examples as the majority class. Therefore, we need to strike a balance and find the k value that maximizes the chosen evaluation metric while considering the trade-off between bias and variance. The goal is to achieve a well-generalized model that accurately classifies both positive and negative instances without being overly

influenced by the majority class.

```
library(class)
accuracy_all_knn=vector()
sensitivity_all_knn=vector()
for (k in 1:100) {
    knn_=knn(train[,-1], test[,-1], cl = train$DLRSN, k = k)
    c=table(knn_,test$DLRSN)
    accuracy_iter=(c[1,1]+c[2,2])/nrow(test)
    sensitivity_iter=c[2,2]/(c[2,2]+c[2,1])
    accuracy_all_knn=c(accuracy_all_knn,accuracy_iter)
    sensitivity_all_knn=c(sensitivity_all_knn,sensitivity_iter)

    }
print("ACCURACY:")
```

```
## [1] "ACCURACY:"
```

```
print(max(accuracy_all_knn))
```

```
## [1] 0.8844739
```

```
print("K=")
```

```
## [1] "K="
```

```
print(which.max(accuracy_all_knn))
```

```
## [1] 46
```

```
print("############################")
```

```
## [1] "############################"
```

```
print("SENSITIVITY")
```

```
## [1] "SENSITIVITY"
```

```
print(max(sensitivity_all_knn))
```

```
## [1] 0.8
```

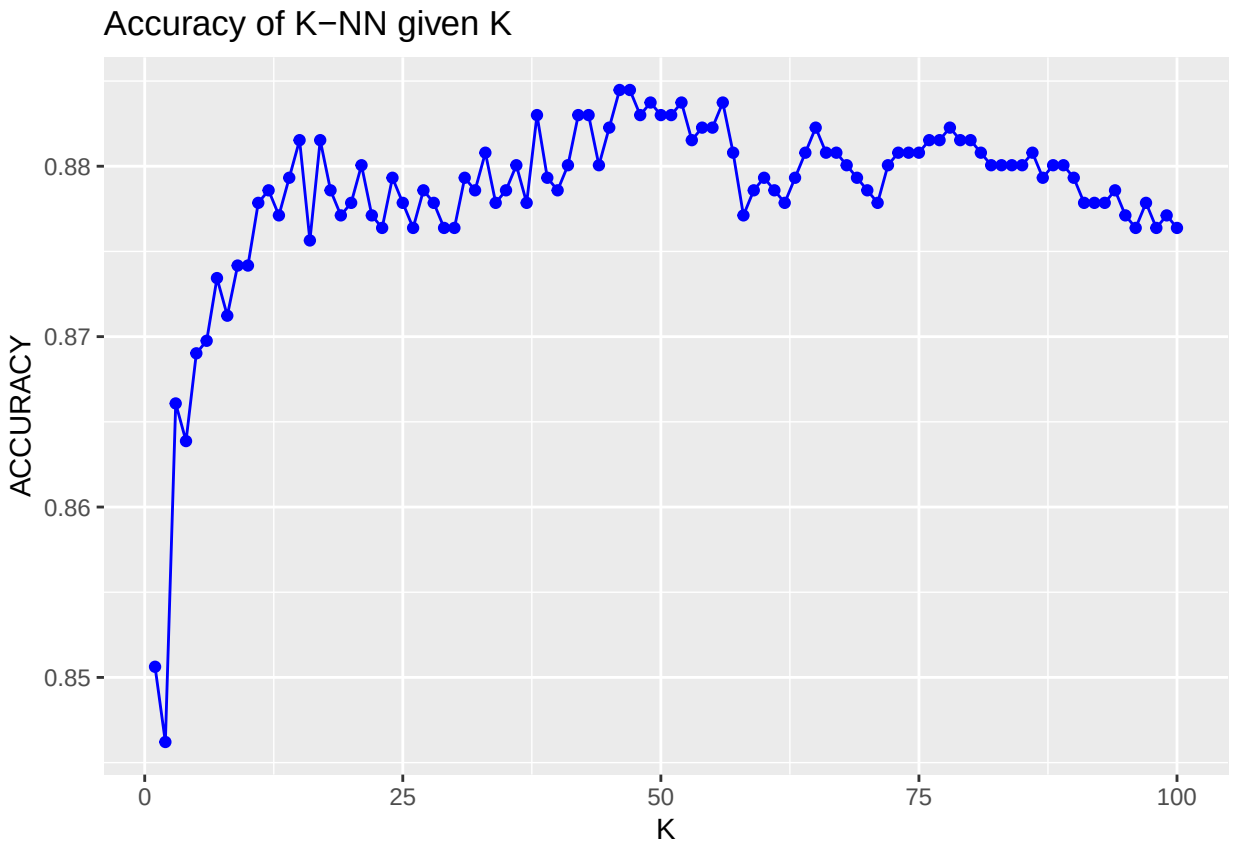
```
print("K=")
```

```
## [1] "K="
```

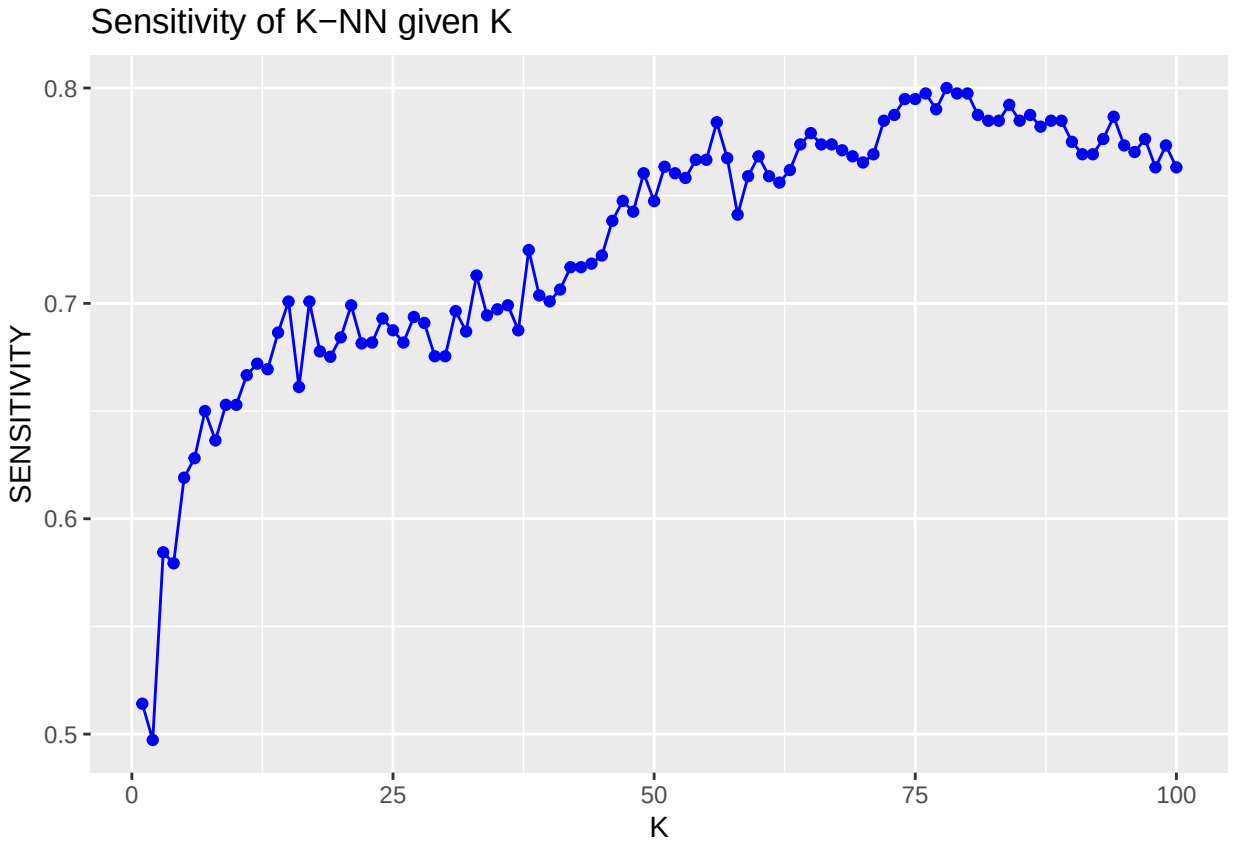
```
print(which.max(sensitivity_all_knn))
```

```
## [1] 78
```

```
accuracy_k=data.frame(K=1:100,ACCURACY=accuracy_all_knn)
ggplot(accuracy_k, aes(x = K, y = ACCURACY)) +
geom_line(colour="blue") +
geom_point(colour="blue") + ggtitle("Accuracy of K-NN given K")
```




```
sensitivity_k=data.frame(K=1:100, SENSITIVITY=sensitivity_all_knn)
ggplot(sensitivity_k, aes(x = K, y = SENSITIVITY)) +
geom_line(colour="blue") +
geom_point(colour="blue") + ggtitle("Sensitivity of K-NN given K")
```



We can observe a really nice result, since k=55 produces both best accuracy and best sensitivity.

Bayes Classifier

The Naive Bayes model is a probabilistic classifier that applies Bayes' theorem to estimate the probability of an instance belonging to a class given the presence of certain attributes. It assumes the simplifying assumption that the attributes are conditionally independent of each other, given the class value. The model calculates the conditional probability using Bayes' theorem and makes predictions based on the class with the highest probability. It assumes that the attributes contribute independently to the likelihood of the class, which simplifies the calculation.

```
library(e1071)
bayes_cl <- naiveBayes(DLRSN ~., data = Bank_clean)

# predict

predictions <- predict(bayes_cl, newdat = test)
cm_n <- table(test$DLRSN, predictions)
cm_n
```

```
##    predictions
##       0   1
##   0 994 157
##   1  57 151
```

```
accuracy_n = (cm_n[1,1]+cm_n[2,2])/nrow(test)
accuracy_n
```

```
## [1] 0.8425313
```

```
sensitivity_n = cm_n[2,2]/(cm_n[2,2]+cm_n[2,1])
sensitivity_n
```

```
## [1] 0.7259615
```

Model Comparison

ROC Curve

The ROC (Receiver Operating Characteristic) curve is introduced as a valuable tool to assess and visualize the classification model's performance. It plots the relationship between sensitivity (True Positive Rate) on the y-axis and 1 - specificity (True Negative Rate) on the x-axis. The ROC curve aids in selecting the best model by maximizing the area under the curve (AUC).

The AUC ranges from 0 to 1, with a value closer to 1 indicating better predictive ability of the model, while an AUC of 0.5 implies that the model performs no better than random chance. In our specific scenario, we prioritize models with high Sensitivity values, alongside high AUC values, as it is crucial to correctly identify positive instances (True Positives) and minimize False Negatives in the prediction process.

```
roc_logistic_b = roc(test$DLRSN,
                     as.numeric(predictions), levels=c(0, 1))
```

```
## Setting direction: controls < cases
```

```
plot(roc_logistic_b, legacy.axes=TRUE,
     main = "Curva ROC", print.auc = FALSE, xlim=c(1.2, -0.2))

roc_logistic_c = roc(test$DLRSN,
                     as.numeric(pred_model_1), levels=c(0, 1))
```

```
## Setting direction: controls < cases
```

```
lines(roc_logistic_c, col = "purple", lty = 1)

roc_logistic_f = roc(test$DLRSN, pred_model_2, levels=c(0, 1))
```

```
## Setting direction: controls < cases
```

```
lines(roc_logistic_f, col = "red", lty = 1)


roc_lda = roc(test$DLRSN,
              as.numeric(post_lda_model[,2]), levels=c(0, 1))
```

```
## Setting direction: controls < cases
```

```
lines(roc_lda, col = "green", lty = 1)


roc_qda = roc(test$DLRSN,
              as.numeric(post_qda_model[,2]), levels=c(0, 1))
```

```
## Setting direction: controls < cases
```

```
lines(roc_qda, col = "pink", lty = 1)
```

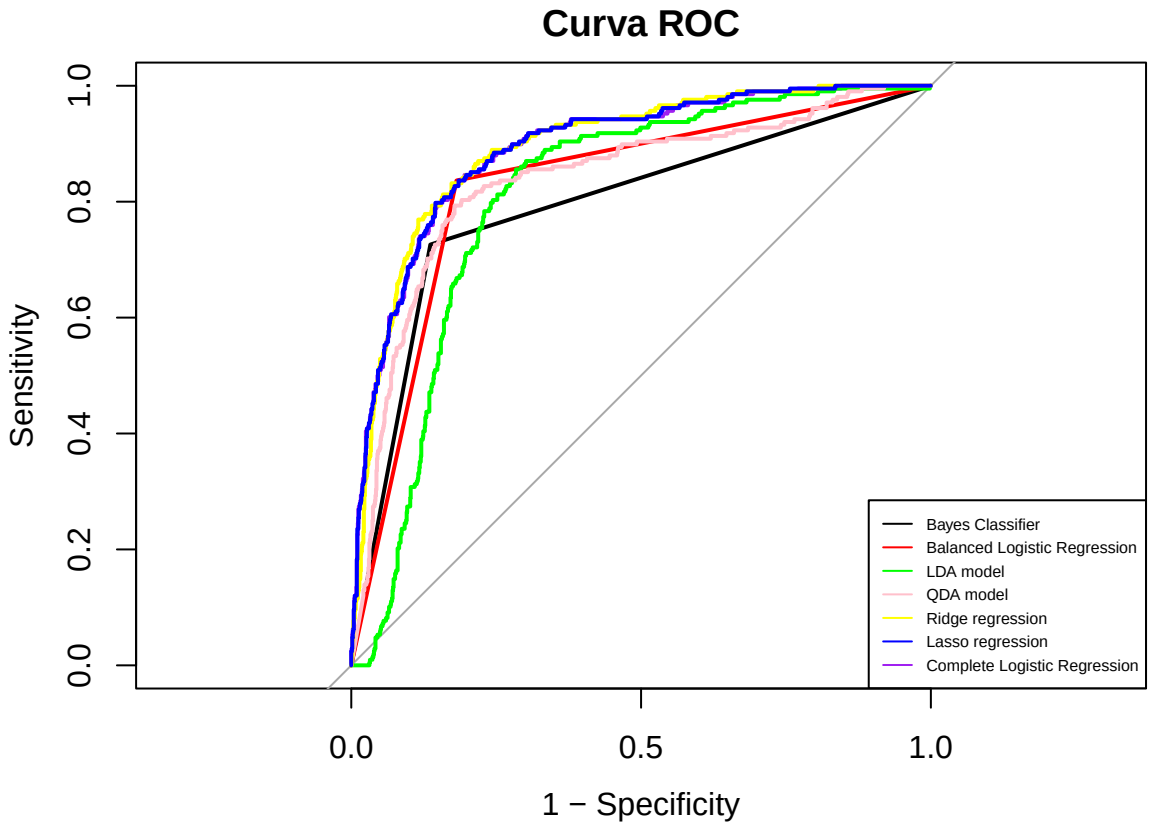
```
roc_ridge = roc(test$DLRSN,
                as.numeric(pred_ridge), levels=c(0, 1))
```

```
## Setting direction: controls < cases
```

```
lines(roc_ridge, col = "yellow", lty = 1)


roc_lasso = roc(test$DLRSN,
                as.numeric(pred_lasso), levels=c(0, 1))
```

```
## Setting direction: controls < cases
```

```
lines(roc_lasso, col = "blue", lty = 1)

legend("bottomright", legend = c("Bayes Classifier",
                                 "Balanced Logistic Regression",
                                 "LDA model", "QDA model",
                                 "Ridge regression",
                                 "Lasso regression",
                                 "Complete Logistic Regression"),
       col = c("black", "red", "green", "pink",
               "yellow",  "blue", "purple"), lty = 1, cex = 0.5)
```




```
print("AUC LOGISTIC BALANCED")
```

```
## [1] "AUC LOGISTIC BALANCED"
```

```
auc(test$DLRSN,pred_model_2)[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## [1] 0.8274786
```

```
print("AUC LOGISTIC WITH ALL FEATURES")
```

```
## [1] "AUC LOGISTIC WITH ALL FEATURES"
```

```
auc(test$DLRSN,pred_model_1)[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## [1] 0.8918624
```

```
print("AUC Byes_classifier")
```

```
## [1] "AUC Byes_classifier"
```

```
auc(test$DLRSN,as.numeric(predictions))[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## [1] 0.7947792
```

```
print("AUC LDA")
```

```
## [1] "AUC LDA"
```

```
auc(test$DLRSN,post_lda_model[,2])[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## [1] 0.8078469
```

```
print("AUC QDA")
```

```
## [1] "AUC QDA"
```

```
auc(test$DLRSN,post_qda_model[,2])[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## [1] 0.8376328
```

```
print("AUC RIDGE REGRESSION")
```

```
## [1] "AUC RIDGE REGRESSION"
```

```
auc(test$DLRSN,as.numeric(pred_ridge))[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## [1] 0.8937254
```

```
print("AUC LASSO REGRESSION")
```

```
## [1] "AUC LASSO REGRESSION"
```

```
auc(test$DLRSN,as.numeric(pred_lasso))[1]
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## [1] 0.892347
```

We can observe how all models have overall the same AUC, with the ridge, lasso and completed logistic that are the best.

Metrics Comparison

Moreover, given the purpose of our analysis, we would prefer a model with the higher True Positive Rate (Sensitivity) without caring much about Sensibility (True Negative Rate).

The following table resumes the accuracy and sensitivity reached by each model we estimated:

Model	Accuracy	Sensitivity	AUC
Complete Logistic Model	0.8822	0.3701	0.8901
Balanced Logistic Model	0.8211	0.8361	0.8274
Bayes Classifier Model	0.8447	0.7307	0.8012
Ridge Regression Model	0.8646	0.2355	0.8937
Lasso Regression Model	0.8785	0.3365	0.8930
LDA Model	0.8005	0.6153	0.8074
QDA Model	0.8292	0.7403	0.8379
55-NN	0.8802	0.8006	

We can observe how the accuracy is good in all models, while what changes significantly is the sensitivity. In light of this, it would be suitable to group the models into two groups:

- first group, high accuracy, low sensitivity: complete logistic model, ridge regression model, lasso regression model.
- second group, high accuracy, low sensitivity: balanced logistic model, Bayes classifier, lda model, qda model, knn.

Consider the models with the highest aucs, and we can see that we are on an almost equal footing between the balanced model, qda, and Bayes. Among these we decide not to consider balanced since it is trained on artificially modified data, thus preferring models trained on the original data. Among qda and Bayes we finally decide to select the model with the highest sensitivity, albeit by only 1 percentage point. So as the final model for our predictions we will use the qda model.

Conclusions

At the beginning of our analysis we already knew that since bankruptcy is a complex and articulated phenomenon, it would have been difficult to find a single cause that generates it, therefore behind the development of bankruptcy there is not a single particularly responsible index but rather a concomitance of a set of multiple factors. Having said that, the result of our project was able to identify some ratios among the 10 ratios present, in particular R4 (ME/TL - Market Value of Equity/Total Liability) and R5 (S/TA - Sales/Total Assets) , which affect the determination of bankruptcy to a lesser extent than the other variables considered.

- R4 (ME/TL) represents the ratio between the market value of a company's equity and its total liability. It indicates how much the market value of equity covers the company's total liabilities. The lower impact of this variable suggests that the market value of equity may not be a significant factor in predicting bankruptcy for the specific dataset under analysis. Other variables may be more relevant in explaining this phenomenon.
- R5 (S/TA) represents the ratio between the sales value of a company and its total assets. This ratio measures the company's ability to generate sales relative to its assets. Its lower impact may indicate that the sales-to-assets ratio is not a significant predictive factor for bankruptcy in the specific context of the analysis. Other variables may play a more important role in identifying companies that are prone to bankruptcy.