Risk assessment of credit grants from financial institutions and development of predictive model to support the decision to grant credit or not to new clients

Mario Peres

October 20, 2021

This project was part of my studies in the course "Big Data Analytics with R and Azure Machine Learning" at Data Science Academy, and another solution for this problem was also provided by the instructors. My motivation to do this work was to practice all the new knowledge and skills acquired (R language) during my training, including data munging, exploratory analysis, data visualization, feature selection, creation and evaluation of machine learning models. The main purpose of this project was to improve the previous versions of the model and find my own solution for this problem, using different data manipulation techniques and three algorithms, including Random Forest, Support Vector Machines and Naive Bayes.

Problem definition: Financial institutions want a predictive model to support their decision on providing credit grants or not to new clients based on their historical data.

Data: A csv file used to run this project is provided in the same folder as this project. The variable names are self-explanatory, however there is no dictionary for the categories of all variables, as it was not supplied and do not interfere in this analysis. Moreover, it is also available an attached R script, named fun_utils.R, which is necessary as it holds a few functions used in this project.

Number of features: **21 (20 predictors)**

Target variable: "**credit.rating**"

Let's get started!!!

Firstly, the packages and the fun_utils.R script were loades. Moreover, due to randomness during data manipulation, a seed was set to allow later reproducibility of part of this work, as the this seed will not be able to control all the randomness generated in many parts of this project. Therefore, executing this code again, may result in slightly different outcomes.

```
# Loading packages
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(plyr)
```

```
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
library(ggplot2)
library(ROSE)
```

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

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(e1071)
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```
# Loading the source script 'plot_utils.R'
source('fun_utils.R')

# Setting seed
set.seed(123)
```

Following that, the csv file 'credit_dataset.csv', which is also in the project folder at gitHub, was also loaded. Then, the data type of variables were checked and the data was visualized.

```
# Loading dataset
df <- read.csv('credit_dataset.csv')
```

```
# Checking data types
glimpse(df)
```

```
## Rows: 1,000
## Columns: 21
## $ credit.rating                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ account.balance               <int> 1, 1, 2, 1, 1, 1, 1, 1, 3, 2, 1, 1, 1, ~
## $ credit.duration.months        <int> 18, 9, 12, 12, 12, 10, 8, 6, 18, 24, 11~
## $ previous.credit.payment.status <int> 3, 3, 2, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, ~
## $ credit.purpose                <int> 2, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 1, 3, ~
## $ credit.amount                 <int> 1049, 2799, 841, 2122, 2171, 2241, 3398~
## $ savings                       <int> 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 1, 2, 1, ~
## $ employment.duration           <int> 1, 2, 3, 2, 2, 1, 3, 1, 1, 1, 2, 3, 3, ~
## $ installment.rate              <int> 4, 2, 2, 3, 4, 1, 1, 2, 4, 1, 2, 1, 1, ~
## $ marital.status                <int> 1, 3, 1, 3, 3, 3, 3, 3, 1, 1, 3, 4, 1, ~
## $ guarantor                     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ residence.duration            <int> 4, 2, 4, 2, 4, 3, 4, 4, 4, 4, 2, 4, 4, ~
## $ current.assets                <int> 2, 1, 1, 1, 2, 1, 1, 1, 3, 4, 1, 3, 3, ~
## $ age                           <int> 21, 36, 23, 39, 38, 48, 39, 40, 65, 23,~
## $ other.credits                 <int> 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ apartment.type                <int> 1, 1, 1, 1, 2, 1, 2, 2, 2, 1, 1, 1, 2, ~
## $ bank.credits                  <int> 1, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 1, ~
## $ occupation                    <int> 3, 3, 2, 2, 2, 2, 2, 2, 1, 1, 3, 3, 3, ~
## $ dependents                    <int> 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 1, ~
## $ telephone                     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ foreign.worker                <int> 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, ~
```

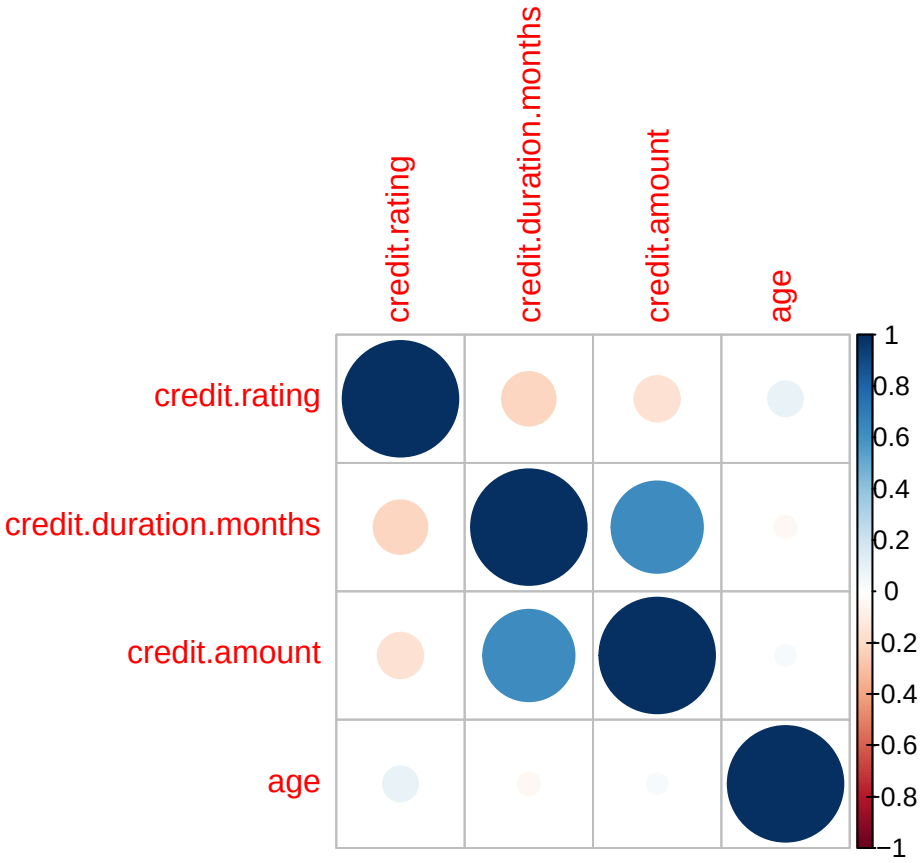
```
# Visualizing the dataset
View(df)
```

From this first glance at the data type of variables, it was observed that only three variables should be of numeric type, including credit.duration.months, credit.amount and age. All the other variables should be set to factor. But before that, the correlations between target and numeric variables were calculated.

```
# Calculating correlations between numeric variables and target variable
df_cor <- cor(df[,c(1,3,6,14)])

# Visualizing correlations
df_cor
```

```
##                         credit.rating credit.duration.months credit.amount
## credit.rating             1.00000000            -0.21492667   -0.15474015
## credit.duration.months   -0.21492667             1.00000000    0.62498846
## credit.amount            -0.15474015             0.62498846    1.00000000
## age                       0.09127195            -0.03754986    0.03227268
##                                age
## credit.rating           0.09127195
## credit.duration.months -0.03754986
## credit.amount           0.03227268
## age                     1.00000000
```

```
# Plotting correlations
corrplot(df_cor)
```

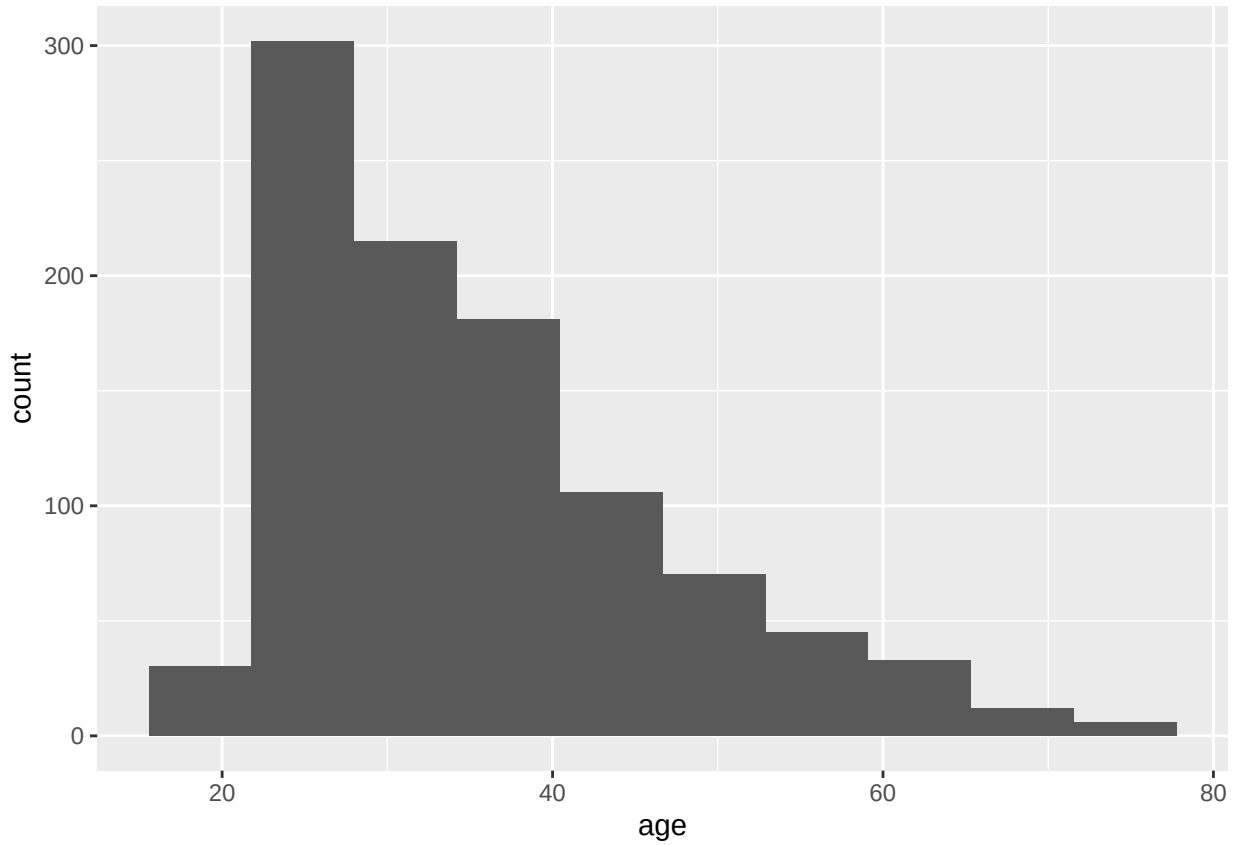


No strong linear correlations were observed considering the evaluated variables. Correlations of -0.21, -0.15 and +0.09 were observed between credit.rating X credit.duration.months, credit.amount and age, respectively.

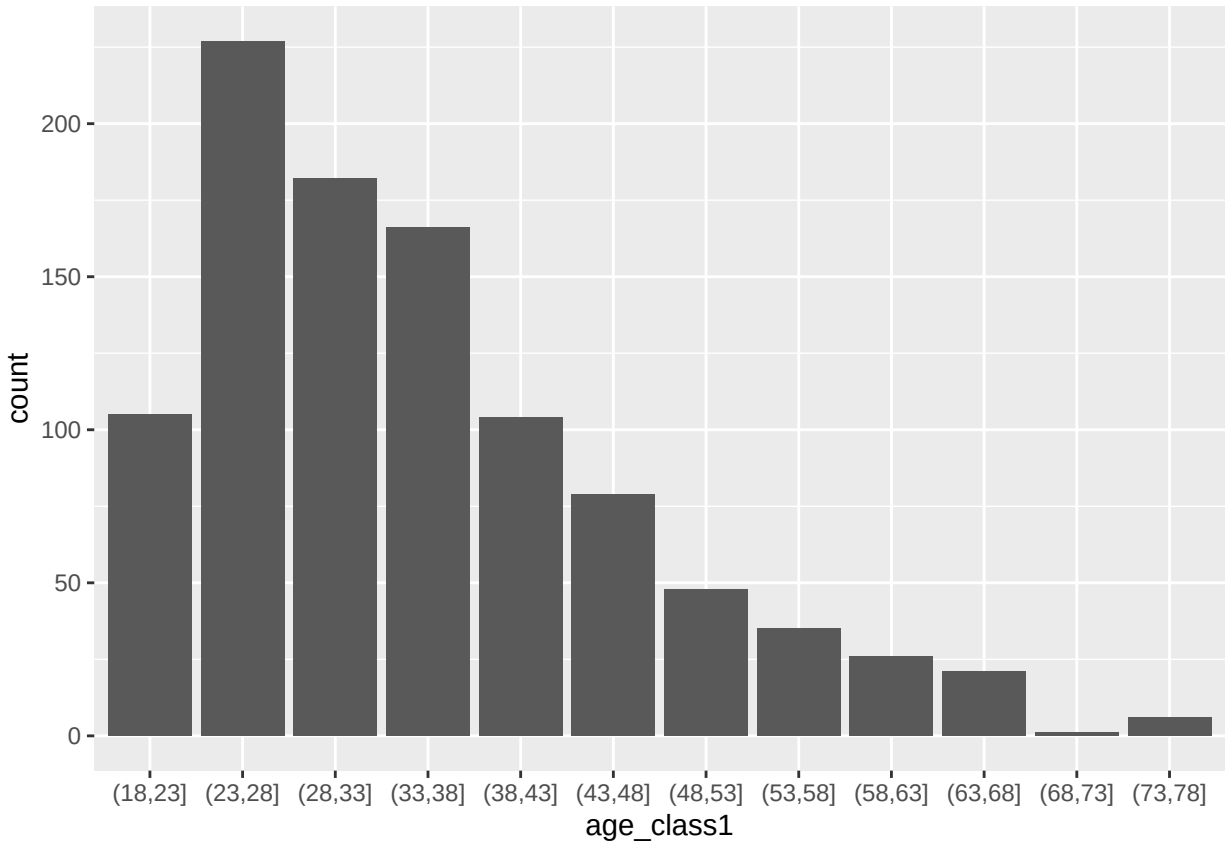
After that, a few more categorical variables were created (feature engineering) based on the information of numeric variables. To achieve that, the numerical variables were separated into classes, using at least two different sets of intervals (smaller and larger). Performing this task allows the selection of the best intervals to be used for each variable, and therefore, the process of feature selection.

In this step, new variables containing some class intervals, categorizing the age variable, were created. They considered intervals of 5 (class1) and 10 (class2) years. The selected starting point was 18 years old, as this should be the minimum age required to open a bank account. Furthermore, the minimum age of an individual in this data set was 19. Some visualizations were generated to check the distribution of the new variables age_class1 and age_class2, considering both classes separately.

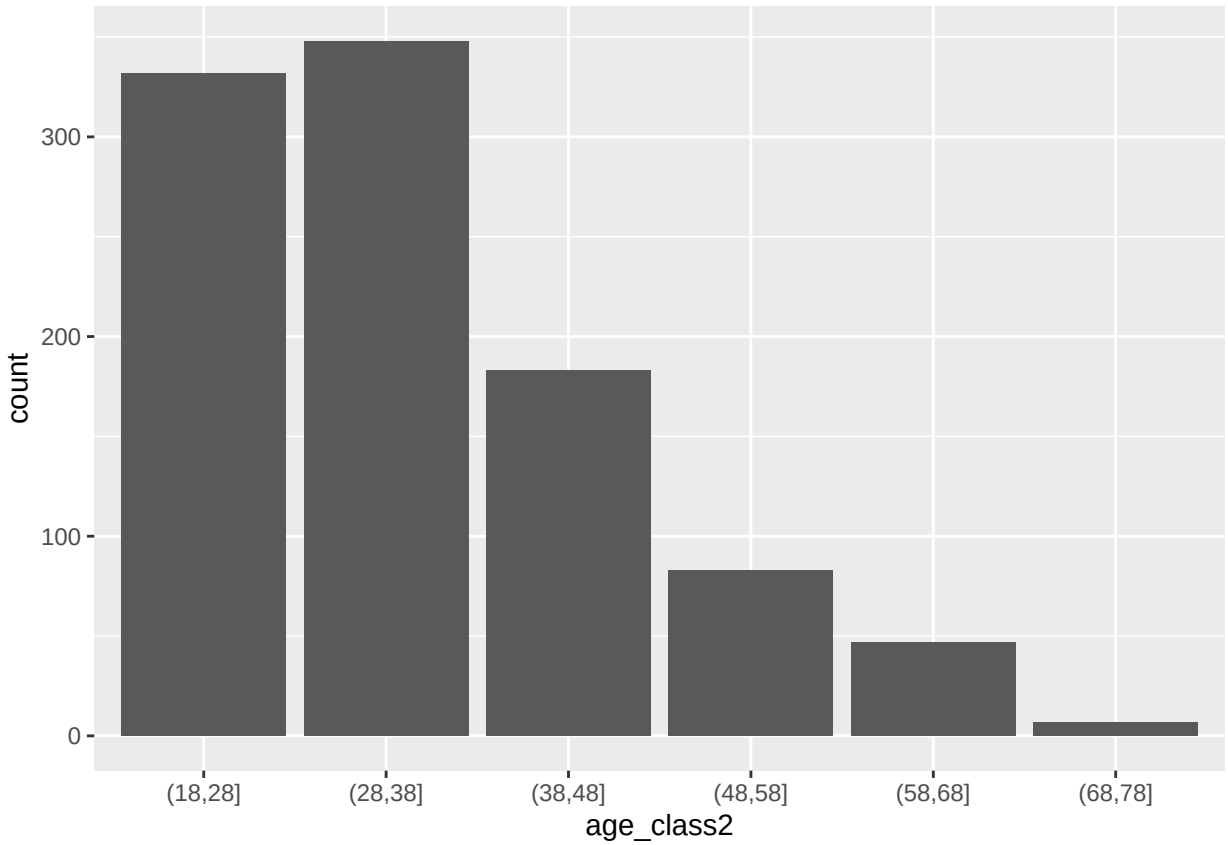
```
# Looking at the distribution of the variable age
ggplot(data = df) +
  geom_histogram(aes(x = age), bins = 10)
```




```
# Creating vectors containing the class for age breaks (1 and 2)
age_div_class1 <- round(seq(18, max(df$age) + 3, 5),0)
age_div_class2 <- round(seq(18, max(df$age) + 3, 10),0)

# Creating two new categorical variables, named age_class1 and age_class2,
# based on the vectors containing the class for age breaks (1 and 2)
df <- dplyr::mutate(df,
             age_class1 = cut(age, breaks = age_div_class1),
             age_class2 = cut(age, breaks = age_div_class2))

# Plotting the number of clients by age_class1
ggplot(df) +
geom_bar(aes(x = age_class1))
```

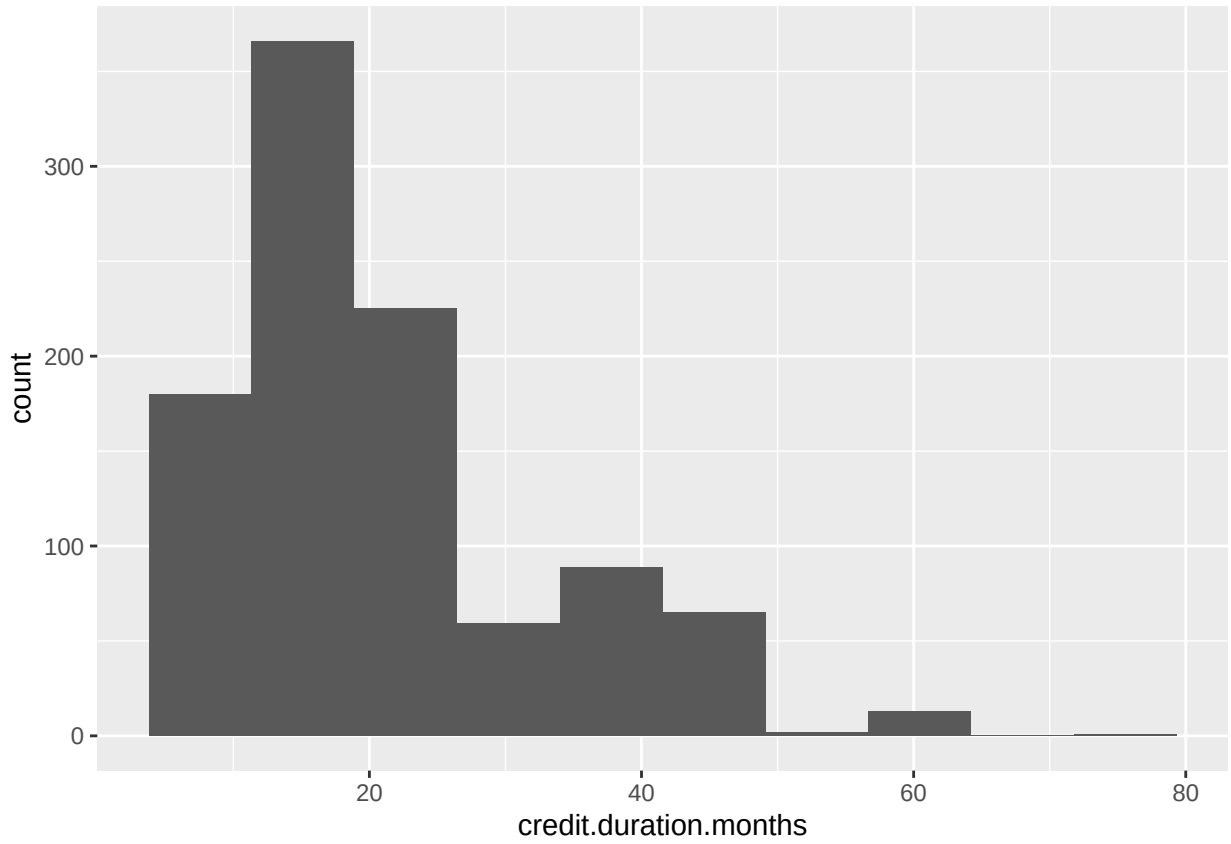



```
# Plotting the number of clients by age_class2
ggplot(df) +
  geom_bar(aes(x = age_class2))
```

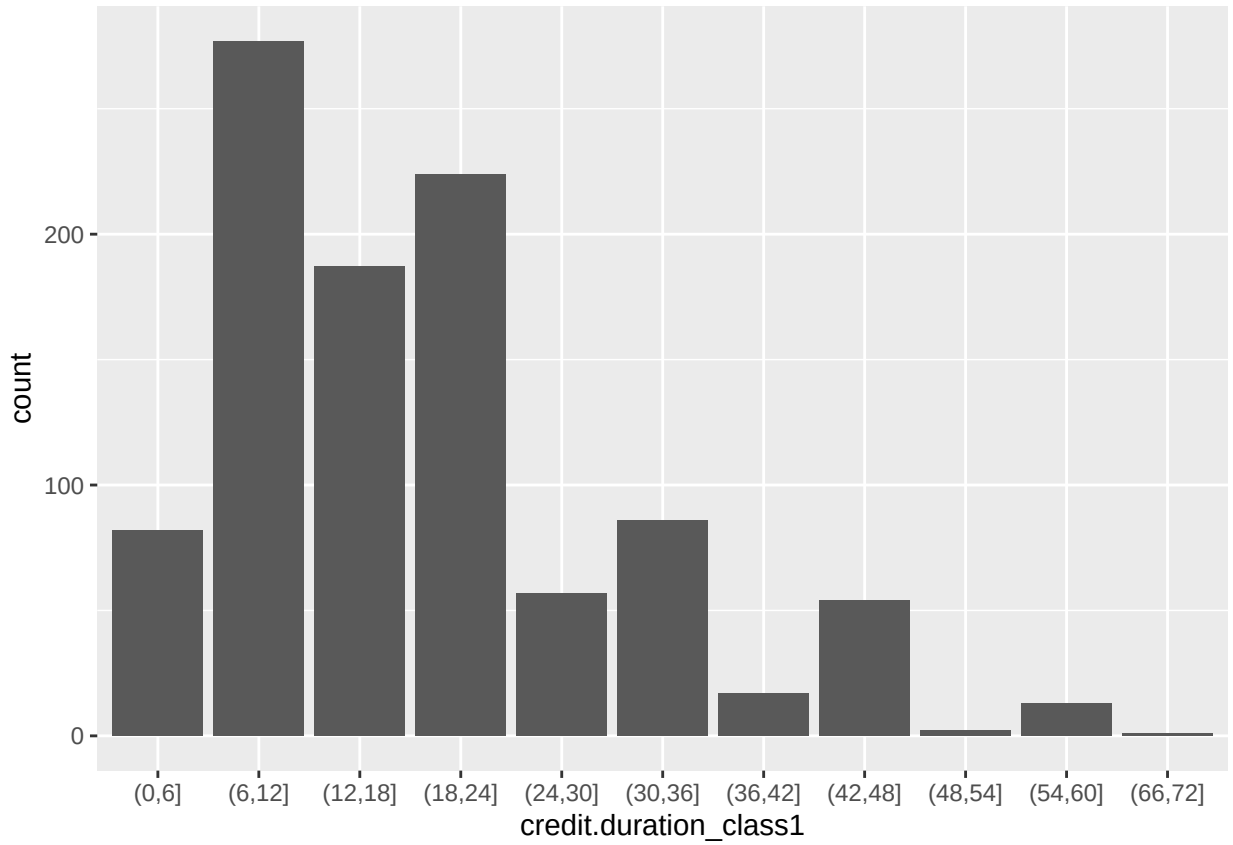


Following that, new variables containing some class intervals, categorizing the credit.duration.months variable, were created. These ones considered intervals of 6-month (class1) and 12-month (class2) periods. Some visualizations were also generated to check the distribution of the new variables credit.duration_class1 and credit.duration_class2, considering both classes separately.

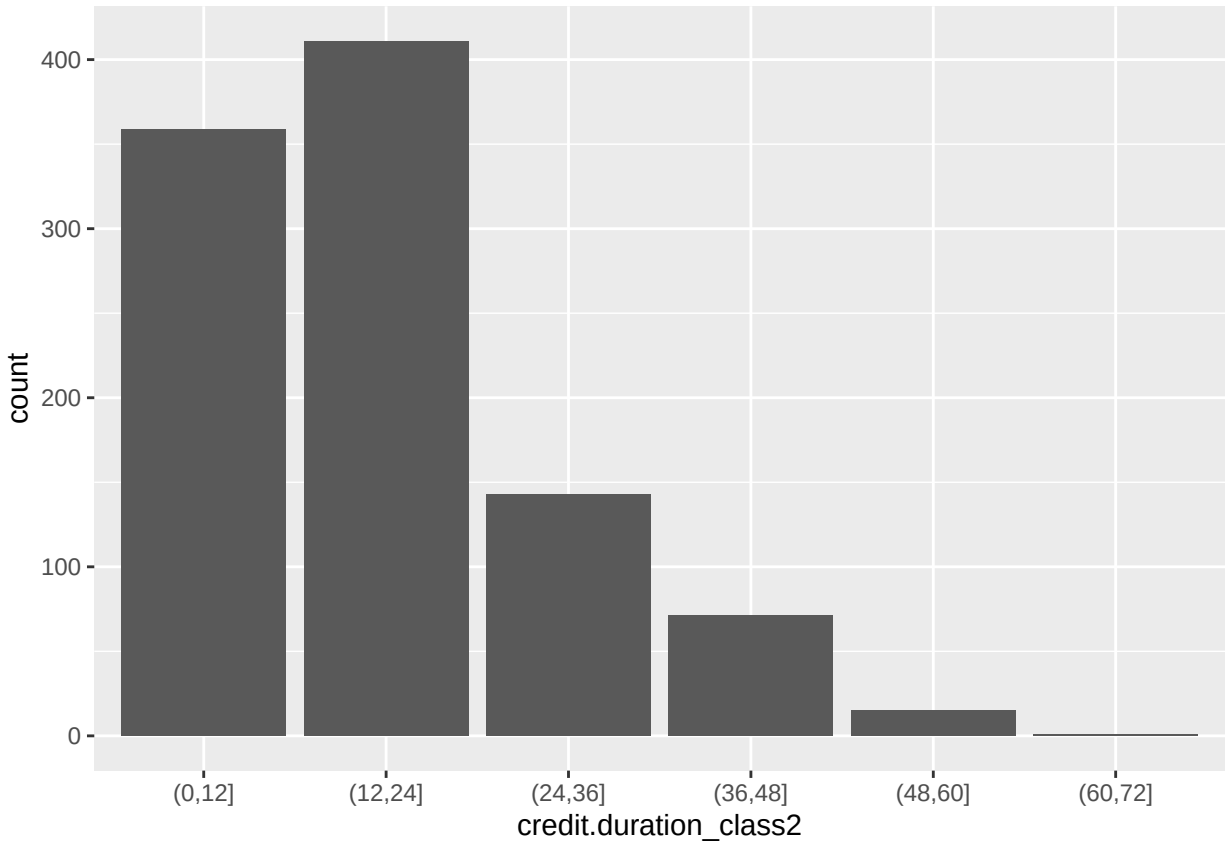
```
# Looking at the distribution of the variable credit.duration.months
ggplot(data = df) +
  geom_histogram(aes(x = credit.duration.months), bins = 10)
```




```
# Creating vectors containing the class breaks for credit.duration.months (1 and 2)
credit.duration_div_class1 <- round(seq(0, max(df$credit.duration.months), 6),0)
credit.duration_div_class2 <- round(seq(0, max(df$credit.duration.months), 12),0)

# Creating two new categorical variables, named credit.duration_class1 and
# credit.duration_class2, based on the vectors containing the class for
# credit.duration.months breaks (1 and 2)
df <- dplyr::mutate(df,
             credit.duration_class1 = cut(credit.duration.months, breaks = credit.duration_div_class1),
             credit.duration_class2 = cut(credit.duration.months, breaks = credit.duration_div_class2))

# Plotting the number of clients by credit.duration_class1
ggplot(df) +
  geom_bar(aes(x = credit.duration_class1))
```

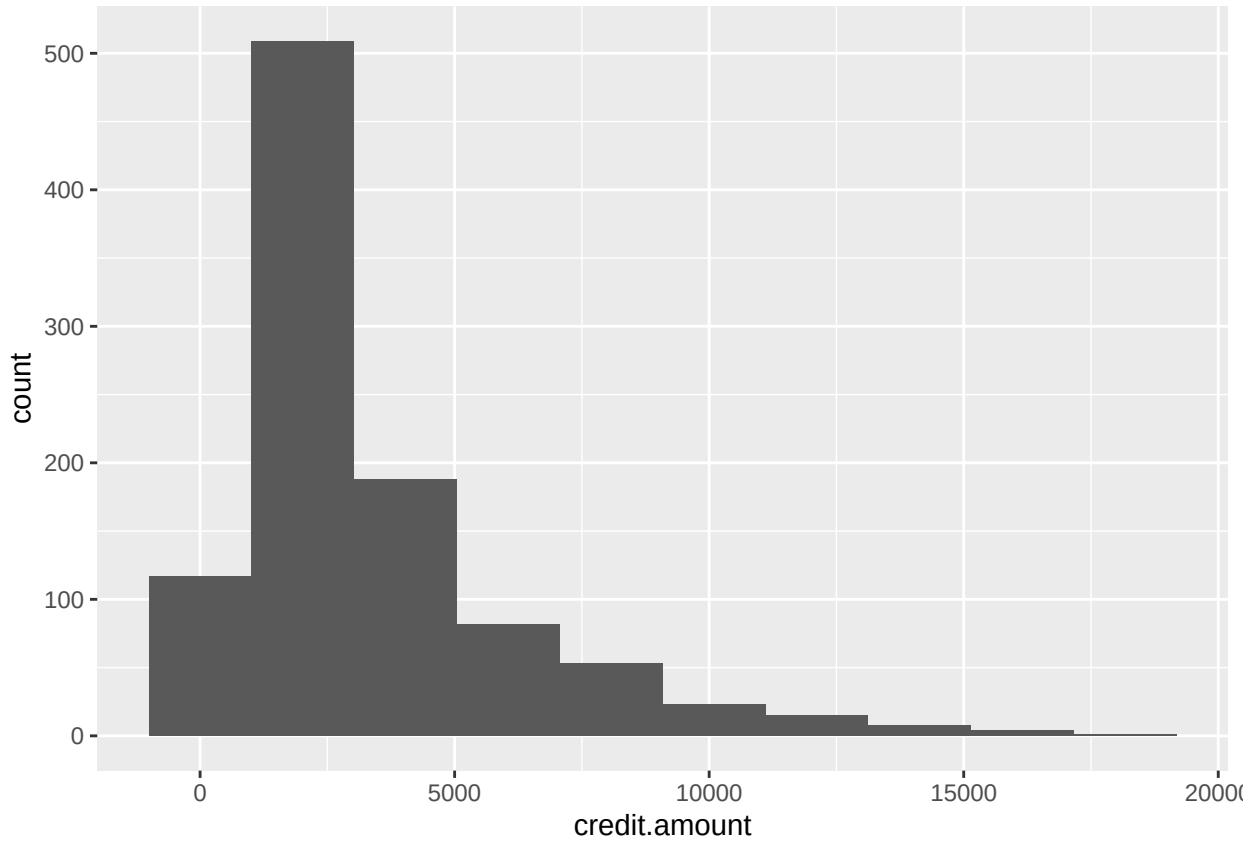



```
# Plotting the number of clients by credit.duration_class1
ggplot(df) +
  geom_bar(aes(x = credit.duration_class2))
```

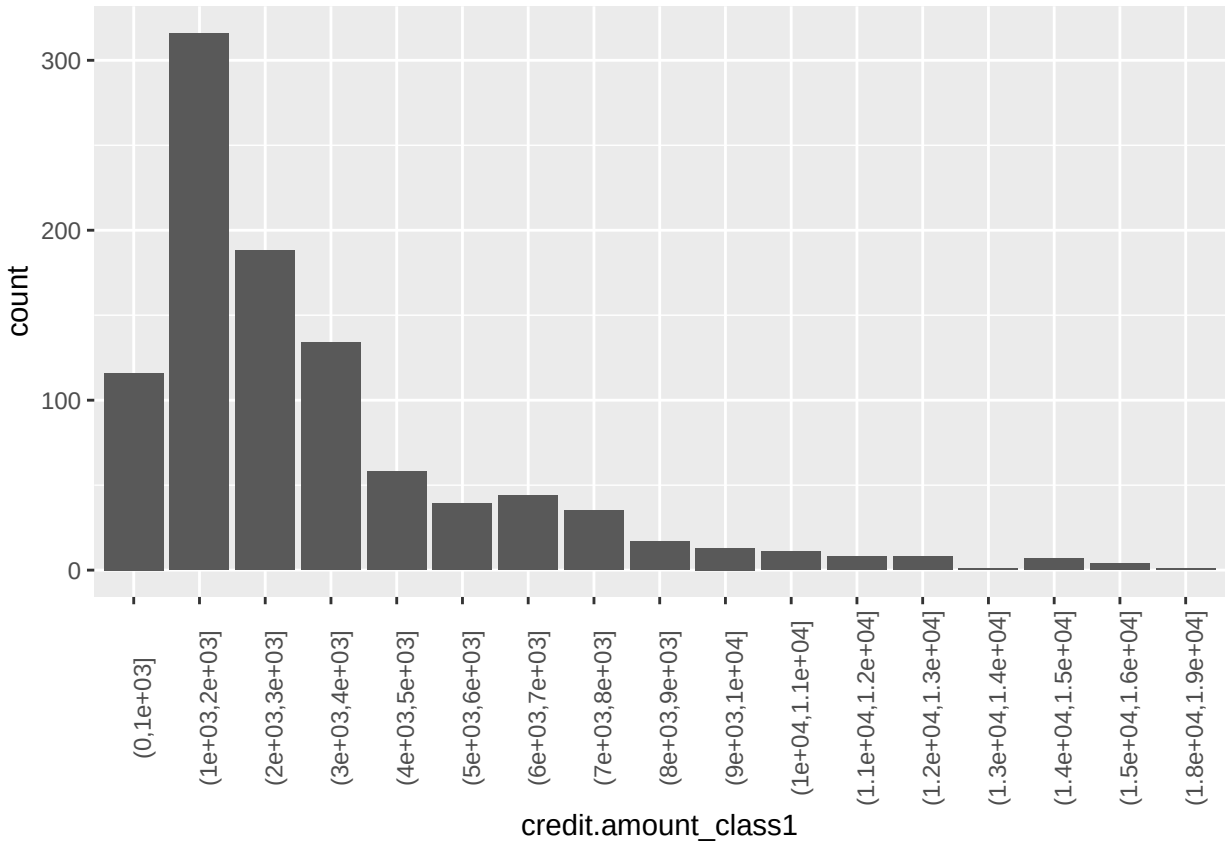


After that, a new variables containing some class intervals, categorizing the amount variable, were created. These ones considered intervals of 1000 (class1), 3000 (class2) and 5000 (class3) of dollars (let's suppose that these values were supplied in dollars). More visualizations were generated to check the distribution of the new variables credit.amount_class1, credit.amount_class2 and credit.amount_class3, considering all classes separately.

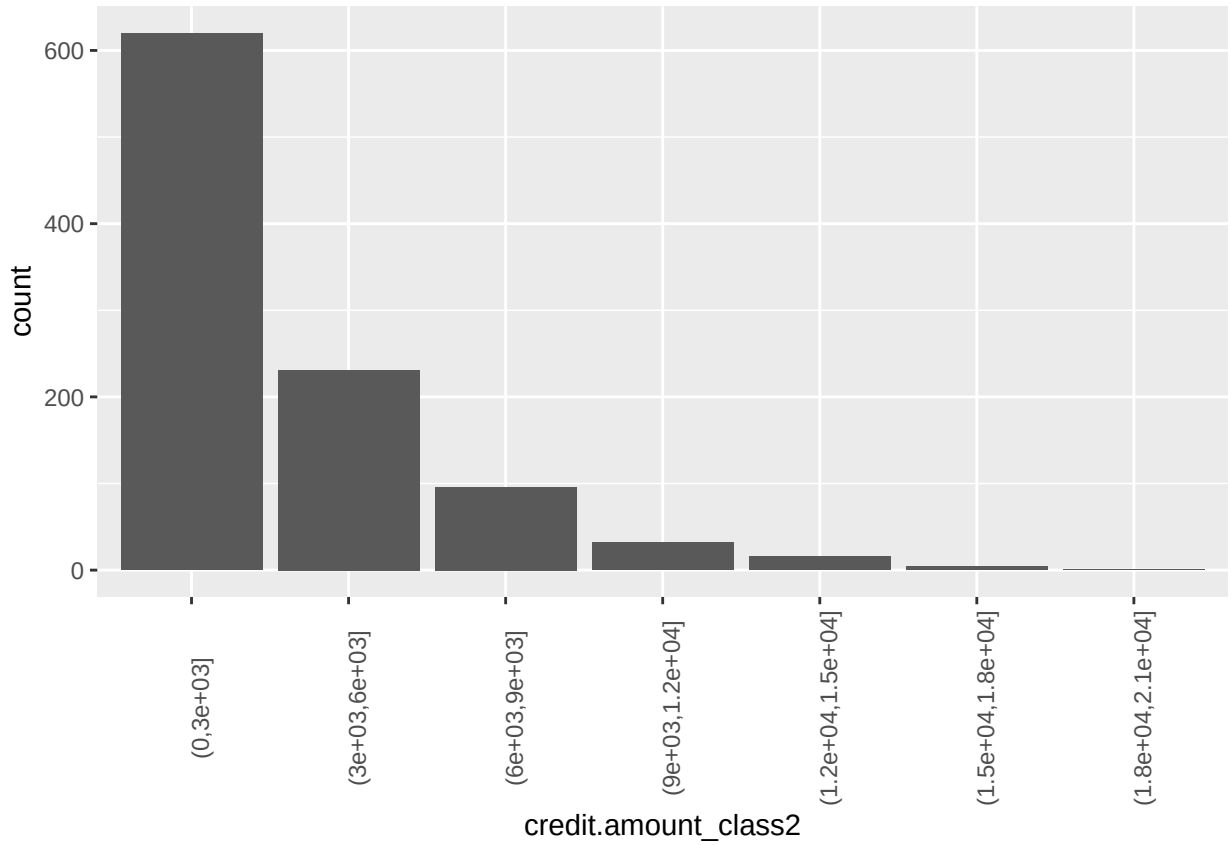
```
# Looking at the distribution of the variable credit.amount
ggplot(data = df) +
  geom_histogram(aes(x = credit.amount), bins = 10)
```



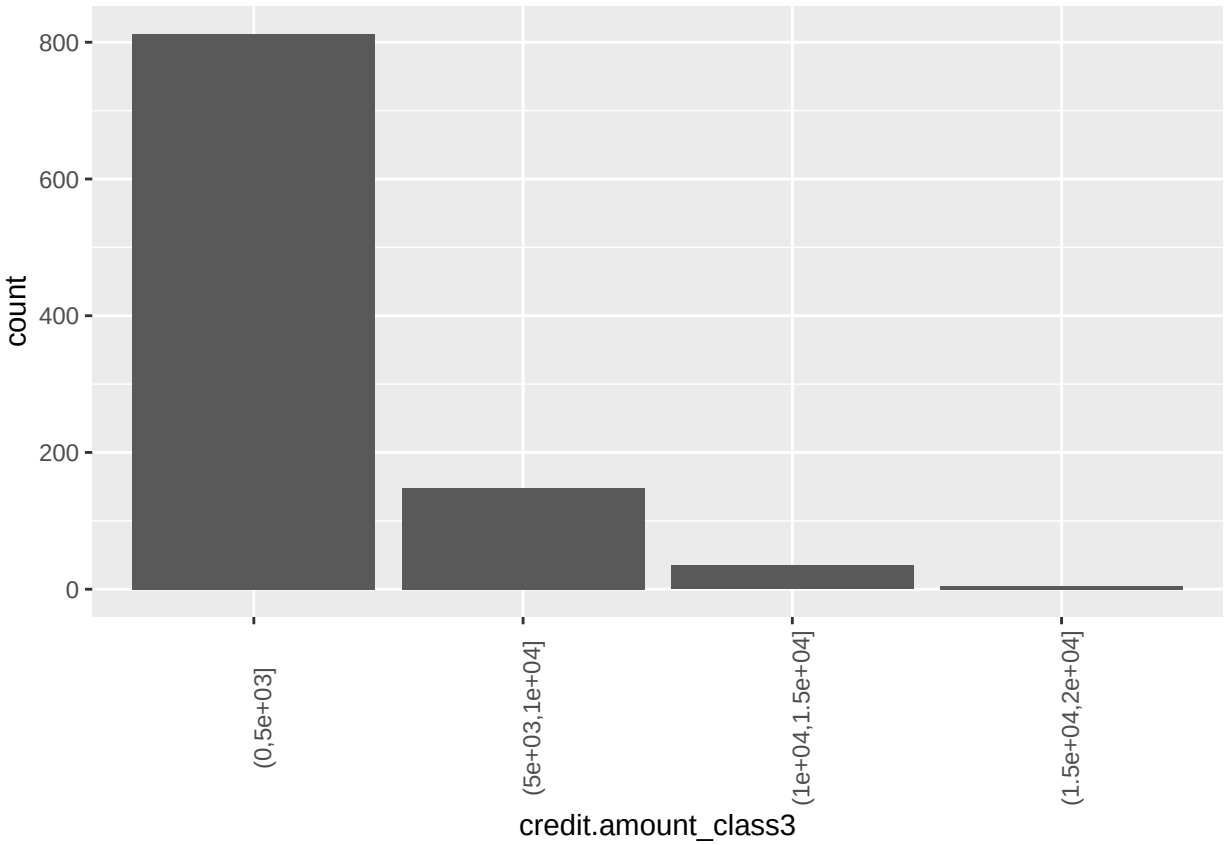

```
# Creating vectors containing the class breaks for credit.amount (1, 2 and 3)
credit.amount_div_class1 <- round(seq(0, round_any(max(df$credit.amount), 1000, f = ceiling), 1000), 0)

credit.amount_div_class2 <- round(seq(0, round_any(max(df$credit.amount), 3000, f = ceiling), 3000), 0)

credit.amount_div_class3 <- round(seq(0, round_any(max(df$credit.amount), 5000, f = ceiling), 5000), 0)

# Creating three new categorical variables, named credit.amount_class1,
# credit.amount_class2 and credit.amount_class3, based on the vectors
# containing the class for credit.amount breaks (1 and 2)
df <- dplyr::mutate(df,
             credit.amount_class1 = cut(credit.amount, breaks = credit.amount_div_class1),
             credit.amount_class2 = cut(credit.amount, breaks = credit.amount_div_class2),
             credit.amount_class3 = cut(credit.amount, breaks = credit.amount_div_class3))

# Plotting the number of clients by credit.amount_class1
ggplot(df) +
  geom_bar(aes(x = credit.amount_class1)) +
  theme(
    axis.text.x = element_text(angle = 90)
  )
```




```
# Plotting the number of clients by credit.amount_class2
ggplot(df) +
  geom_bar(aes(x = credit.amount_class2)) +
  theme(
    axis.text.x = element_text(angle = 90)
  )
```




```
# Plotting the number of clients by credit.amount_class3
ggplot(df) +
  geom_bar(aes(x = credit.amount_class3)) +
  theme(
    axis.text.x = element_text(angle = 90)
  )
```



Visualizing the modified data set to check if the categorical variables were created as expected.

```
# Let's check our new variables in the modified dataset
View(df)
```

As everything looked good with the created categorical variables, the next step was to check if the classes of the target variable was balanced.

```
# Checking if the target variable (credit.rating) is balanced
prop.table(table(df$credit.rating))
```

```
##
##   0   1
## 0.3 0.7
```

As shown in the previous analysis, there were only 30% of data in the class 0. Therefore, in the following steps the modified data set was balanced, considering the target variable. But before that, the data type of variables were checked again.

```
# Checking the data types again
glimpse(df)
```

```
## Rows: 1,000
## Columns: 28
## $ credit.rating                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ account.balance               <int> 1, 1, 2, 1, 1, 1, 1, 1, 3, 2, 1, 1, 1, ~
## $ credit.duration.months        <int> 18, 9, 12, 12, 12, 10, 8, 6, 18, 24, 11~
## $ previous.credit.payment.status <int> 3, 3, 2, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, ~
## $ credit.purpose                <int> 2, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 1, 3, ~
## $ credit.amount                 <int> 1049, 2799, 841, 2122, 2171, 2241, 3398~
```

```
## $ savings                          <int> 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 1, 2, 1, ~
## $ employment.duration              <int> 1, 2, 3, 2, 2, 1, 3, 1, 1, 1, 2, 3, 3, ~
## $ installment.rate                 <int> 4, 2, 2, 3, 4, 1, 1, 2, 4, 1, 2, 1, 1, ~
## $ marital.status                   <int> 1, 3, 1, 3, 3, 3, 3, 3, 1, 1, 3, 4, 1, ~
## $ guarantor                        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ residence.duration               <int> 4, 2, 4, 2, 4, 3, 4, 4, 4, 4, 2, 4, 4, ~
## $ current.assets                   <int> 2, 1, 1, 1, 2, 1, 1, 1, 3, 4, 1, 3, 3, ~
## $ age                              <int> 21, 36, 23, 39, 38, 48, 39, 40, 65, 23,~
## $ other.credits                    <int> 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ apartment.type                   <int> 1, 1, 1, 1, 2, 1, 2, 2, 2, 1, 1, 1, 2, ~
## $ bank.credits                     <int> 1, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 1, ~
## $ occupation                       <int> 3, 3, 2, 2, 2, 2, 2, 2, 1, 1, 3, 3, 3, ~
## $ dependents                       <int> 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 1, ~
## $ telephone                        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ foreign.worker                   <int> 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, ~
## $ age_class1                       <fct> "(18,23]", "(33,38]", "(18,23]", "(38,4~
## $ age_class2                       <fct> "(18,28]", "(28,38]", "(18,28]", "(38,4~
## $ credit.duration_class1           <fct> "(12,18]", "(6,12]", "(6,12]", "(6,12]"~
## $ credit.duration_class2           <fct> "(12,24]", "(0,12]", "(0,12]", "(0,12]"~
## $ credit.amount_class1             <fct> "(1e+03,2e+03]", "(2e+03,3e+03]", "(0,1~
## $ credit.amount_class2             <fct> "(0,3e+03]", "(0,3e+03]", "(0,3e+03]", ~
## $ credit.amount_class3             <fct> "(0,5e+03]", "(0,5e+03]", "(0,5e+03]", ~
```

All the variables were transformed to factor type, including the numeric variables and the balance of the data was performed using the method ROSE from package ROSE, and considering the target variable. This was performed because the method used to balance the data used produced a more adequate data set when using the variables in this way. In this case, "more adequate" was related to the fact that the balanced data set maintained similar characteristics compared to the original data set, such as the distributions format and mean values. Finally the results of this step are visualized.

```
# Transforming all variables (excepted the ones that are already factors) to factor
df <- to_factors(df, colnames(df)[-c(22:28)])

# Creating the df.bal dataset with ROSE, which should be a balanced now
df.bal <- ROSE(credit.rating ~ ., data=df, N = 1000, p = 0.5)$data

# Checking if the target variable (credit.rating) is now balanced!
prop.table(table(df.bal$credit.rating))
```

```
##
##     1     0
## 0.507 0.493
```

A vector containing all the names of numerical variables was created. Then, the function 'to_numeric' was applied to transform the numerical variables of the balanced data set (**df.bal**) to numeric type of data.

```
# Creating a vector containing all the names of numerical variables
numericVars <- c('credit.duration.months',
                 'credit.amount',
                 'age')

# Executing the function (to_numeric) to transform the numerical variables to numeric type
df.bal <- to_numeric(df.bal, numericVars)
```

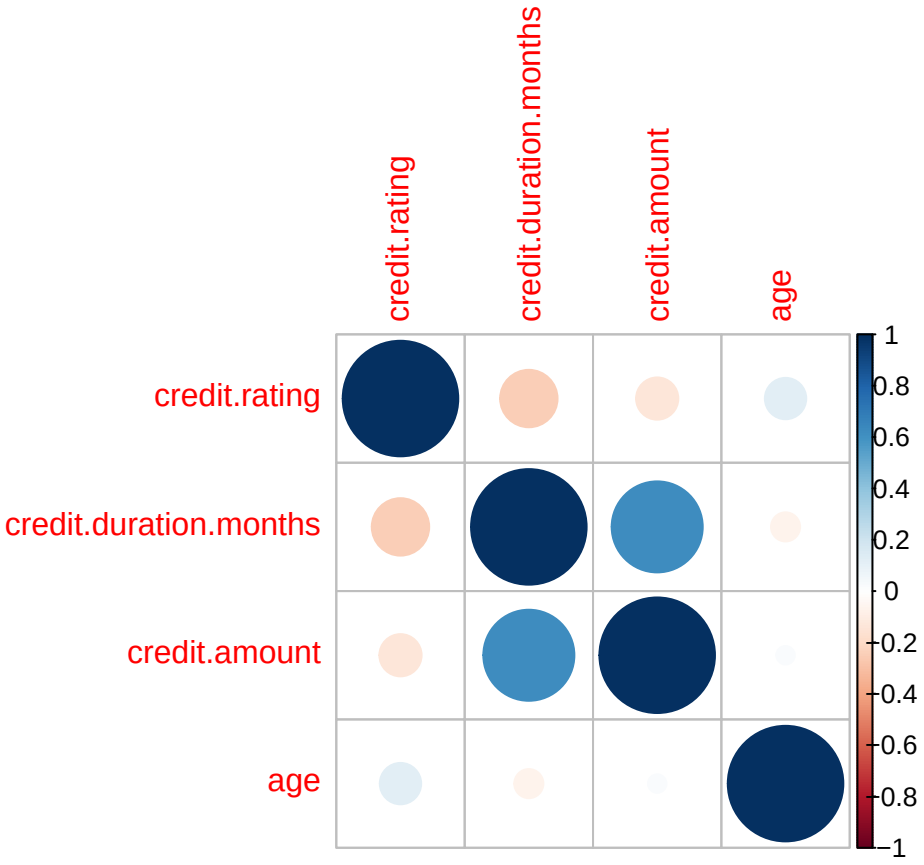
In order to facilitate the correlation analysis with the numerical variables, the factor variable credit.rating was also transformed to integer type. The correlation between the target variable (credit.rating) and all

the other numerical variables were calculated and visualized. Finally at the end of this block of code, the credit.rating variable was transformed back to factor type as it should be for further analysis.

```
# Transforming credit.rating to integer
df.bal$credit.rating <- as.integer(as.character(df.bal$credit.rating))

# Calculating correlations
df.bal_cor <- cor(df.bal[,c(1,3,6,14)])

# Visualizing correlations
df.bal_cor
```

```
##                         credit.rating credit.duration.months credit.amount
## credit.rating               1.0000000             -0.24692149   -0.13340939
## credit.duration.months     -0.2469215              1.00000000    0.62064288
## credit.amount              -0.1334094              0.62064288    1.00000000
## age                         0.1280624             -0.06330453    0.02549036
##                                 age
## credit.rating            0.12806241
## credit.duration.months -0.06330453
## credit.amount           0.02549036
## age                     1.00000000
```

```
# Plotting correlations
corrplot(df.bal_cor)
```




```
# Transforming credit.rating back to factor
df.bal$credit.rating <- as.factor(df.bal$credit.rating)
```

From this analysis, the correlations observed in the balanced data set seems to have varied slightly, but the observed variations do not seem to have changed significantly and are not expected to cause problems with the model development. The means and the data distributions of the numeric variables from the original data set (**df**) were also similar to the means and distributions of the balanced data set (**df.bal**). The classes of the target variable were considered separately for this analysis.

age

```
# Preparing plotting area to receive four plots in one area
par(mfrow=c(2,2))

# Calculating the means age for both datasets (df and df.bal), considering both target labels
mean(as.numeric(df[df$credit.rating == '0',]$age))
```

```
## [1] 15.94667
```

```
mean(df.bal[df.bal$credit.rating == '0',]$age)
```

```
## [1] 15.75254
```

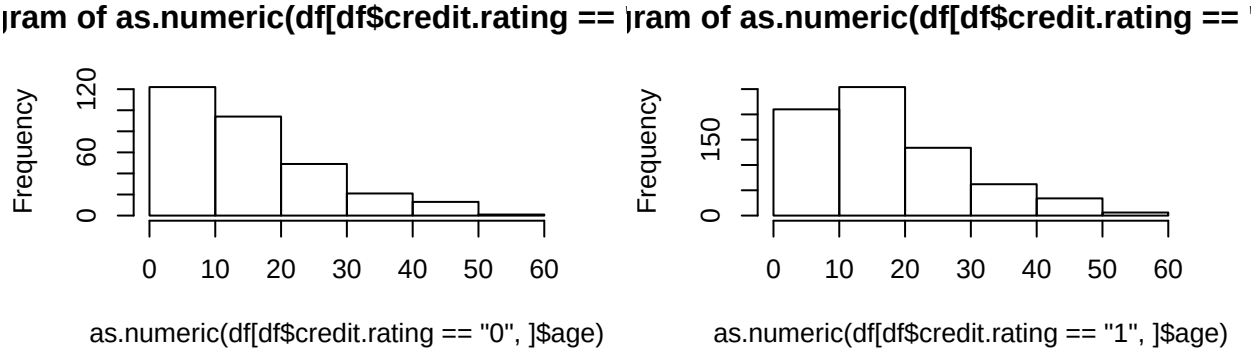
```
mean(as.numeric(df[df$credit.rating == '1',]$age))
```

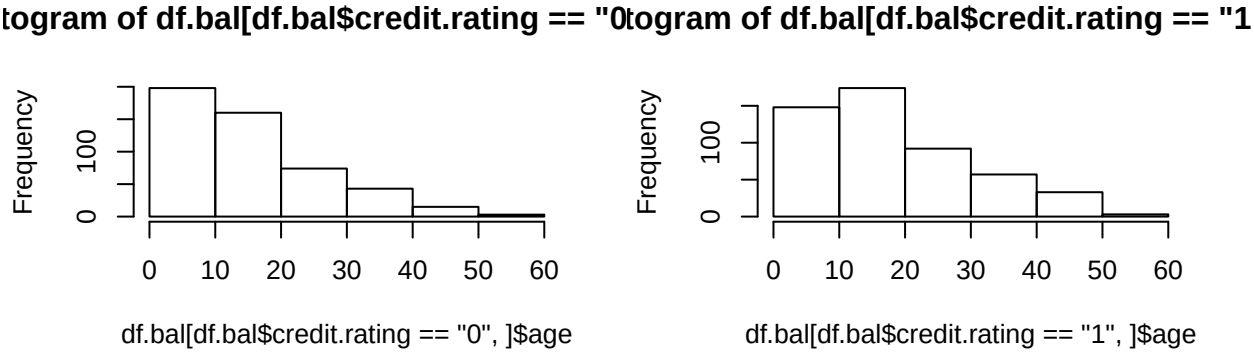
```
## [1] 18.19
```

```
mean(df.bal[df.bal$credit.rating == '1',]$age)
```

```
## [1] 18.714
```

```
# Plotting histograms for age, consideting both datasets and target labels
hist(as.numeric(df[df$credit.rating == '0',]$age), breaks = 5)
hist(as.numeric(df[df$credit.rating == '1',]$age), breaks = 5)
hist(df.bal[df.bal$credit.rating == '0',]$age, breaks = 5)
hist(df.bal[df.bal$credit.rating == '1',]$age, breaks = 5)
```





credit.duration.months

```
# Preparing plotting area to receive four plots in one area
par(mfrow=c(2,2))

# Calculating the means age for both datasets (df and df.bal), considering both target labels
mean(as.numeric(df[df$credit.rating == '0',]$credit.duration.months))
```

```
## [1] 17.19667
```

```
mean(df.bal[df.bal$credit.rating == '0',]$credit.duration.months)
```

```
## [1] 17.71602
```

```
mean(as.numeric(df[df$credit.rating == '1',]$credit.duration.months))
```

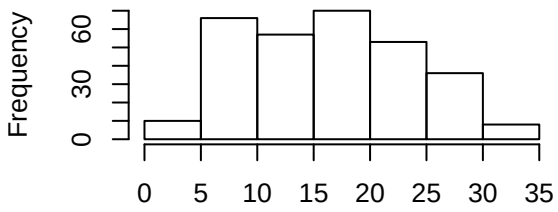
```
## [1] 13.65571
```

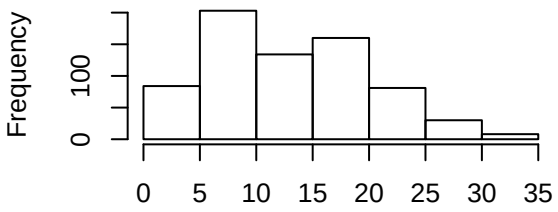
```
mean(df.bal[df.bal$credit.rating == '1',]$credit.duration.months)
```

```
## [1] 14.01578
```

```
# Plotting histograms for age, consideting both datasets and target labels
hist(as.numeric(df[df$credit.rating == '0',]$credit.duration.months), breaks = 5)
hist(as.numeric(df[df$credit.rating == '1',]$credit.duration.months), breaks = 5)
hist(df.bal[df.bal$credit.rating == '0',]$credit.duration.months, breaks = 5)
hist(df.bal[df.bal$credit.rating == '1',]$credit.duration.months, breaks = 5)
```

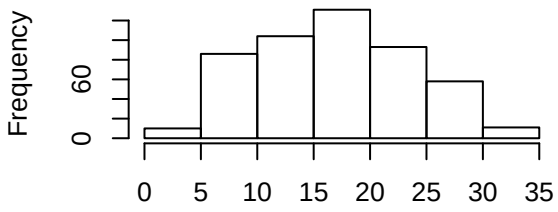
.numeric(df[df$credit.rating == "0",]$cre .numeric(df[df$credit.rating == "1",]$cre

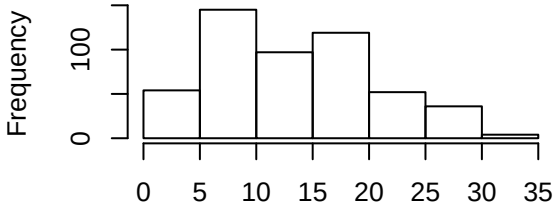




s.numeric(df[df$credit.rating == "0",]$credit.duration.n s.numeric(df[df$credit.rating == "1",]$credit.duration.r

df.bal[df.bal$credit.rating == "0",]$credit df.bal[df.bal$credit.rating == "1",]$credit





df.bal[df.bal$credit.rating == "0",]$credit.duration.mc df.bal[df.bal$credit.rating == "1",]$credit.duration.mc

credit.amount

```
# Preparing plotting area to receive four plots in one area
par(mfrow=c(2,2))
```

```
# Calculating the means age for both datasets (df and df.bal), considering both target labels
mean(as.numeric(df[df$credit.rating == '0',]$credit.amount))
```

```
## [1] 486.8867
```

```
mean(df.bal[df.bal$credit.rating == '0',]$credit.amount)
```

```
## [1] 509.5355
```

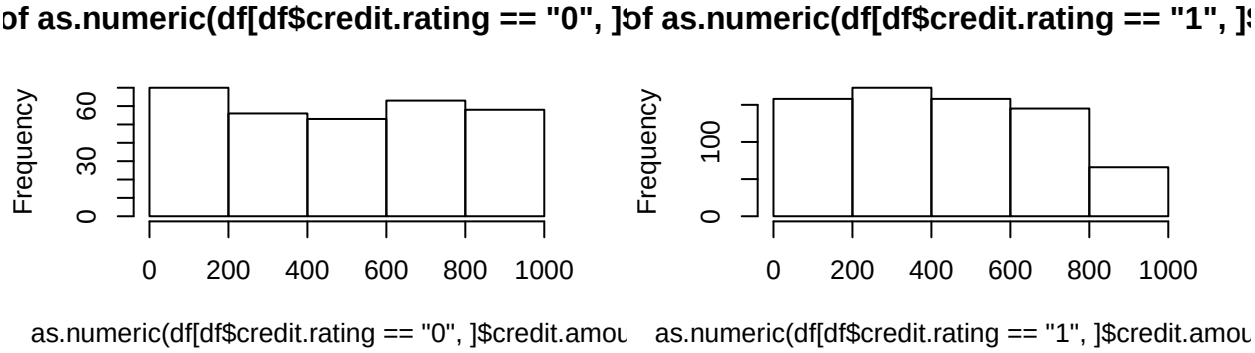
```
mean(as.numeric(df[df$credit.rating == '1',]$credit.amount))
```

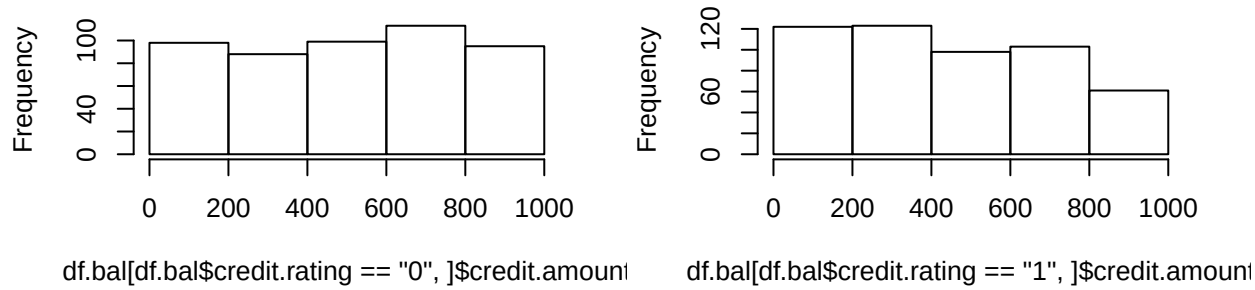
```
## [1] 434.3129
```

```
mean(df.bal[df.bal$credit.rating == '1',]$credit.amount)
```

```
## [1] 436.925
```

```
# Plotting histograms for age, consideting both datasets and target labels
hist(as.numeric(df[df$credit.rating == '0',]$credit.amount), breaks = 5)
hist(as.numeric(df[df$credit.rating == '1',]$credit.amount), breaks = 5)
hist(df.bal[df.bal$credit.rating == '0',]$credit.amount, breaks = 5)
hist(df.bal[df.bal$credit.rating == '1',]$credit.amount, breaks = 5)
```






```
# Setting plotting area back to one plot
par(mfrow=c(1,1))
```

It seems that the balanced data set (**df.bal**) is alright with respect to the numerical variables. Overall, the means and the distributions did not present much difference, considering the original data set and the balanced one. The balanced data set was checked for missing values (NA) and scale was applied to the numerical variables. Finally, the data type of modified variables was presented.

```
# Checking df.bal for NA values
sum(is.na(df.bal))
```

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

```
# Scalling numeric variables
df.bal <- to_scale(df.bal, numericVars)

# Transforming scaled variables (type matrix) back to numeric again
df.bal <- to_numeric(df.bal, numericVars)

# Checking the changes in data types
glimpse(df.bal)
```

```
## Rows: 1,000
## Columns: 28
## $ credit.rating                  <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ account.balance                <fct> 3, 1, 3, 2, 3, 1, 3, 1, 2, 3, 3, 3, 3, ~
## $ credit.duration.months         <dbl> -1.17933161, -1.71296583, 0.15475392, -~
## $ previous.credit.payment.status <fct> 3, 3, 2, 3, 1, 3, 3, 1, 2, 2, 2, 2, 3, ~
## $ credit.purpose                 <fct> 3, 4, 3, 3, 3, 1, 3, 2, 4, 4, 3, 4, 3, ~
## $ credit.amount                  <dbl> -0.950333336, -0.931967419, 0.118563011~
## $ savings                        <fct> 4, 1, 3, 1, 2, 1, 1, 3, 1, 4, 1, 1, 1, ~
## $ employment.duration            <fct> 3, 1, 4, 4, 4, 4, 3, 2, 4, 1, 4, 4, 3, ~
## $ installment.rate               <fct> 4, 2, 3, 3, 4, 4, 2, 4, 1, 4, 4, 2, 2, ~
## $ marital.status                 <fct> 3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, ~
## $ guarantor                      <fct> 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, ~
## $ residence.duration             <fct> 2, 4, 4, 4, 4, 4, 3, 4, 1, 4, 2, 4, 3, ~
## $ current.assets                 <fct> 2, 1, 3, 1, 3, 2, 1, 1, 1, 2, 3, 2, 1, ~
## $ age                            <dbl> -0.71356628, 0.41029629, 0.41029629, 1.~
## $ other.credits                  <fct> 2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 2, 2, ~
## $ apartment.type                 <fct> 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ bank.credits                   <fct> 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, ~
## $ occupation                     <fct> 3, 2, 3, 2, 2, 3, 2, 3, 4, 4, 3, 4, 2, ~
## $ dependents                     <fct> 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 2, ~
## $ telephone                      <fct> 2, 1, 1, 2, 1, 2, 1, 2, 2, 2, 1, 2, 1, ~
## $ foreign.worker                 <fct> 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ age_class1                     <fct> "(23,28]", "(38,43]", "(38,43]", "(43,4~
## $ age_class2                     <fct> "(18,28]", "(38,48]", "(38,48]", "(38,4~
## $ credit.duration_class1         <fct> "(6,12]", "(0,6]", "(18,24]", "(12,18]"~
## $ credit.duration_class2         <fct> "(0,12]", "(0,12]", "(12,24]", "(12,24]~
## $ credit.amount_class1           <fct> "(1e+03,2e+03]", "(1e+03,2e+03]", "(2e+~
## $ credit.amount_class2           <fct> "(0,3e+03]", "(0,3e+03]", "(0,3e+03]", ~
## $ credit.amount_class3           <fct> "(0,5e+03]", "(0,5e+03]", "(0,5e+03]", ~
```

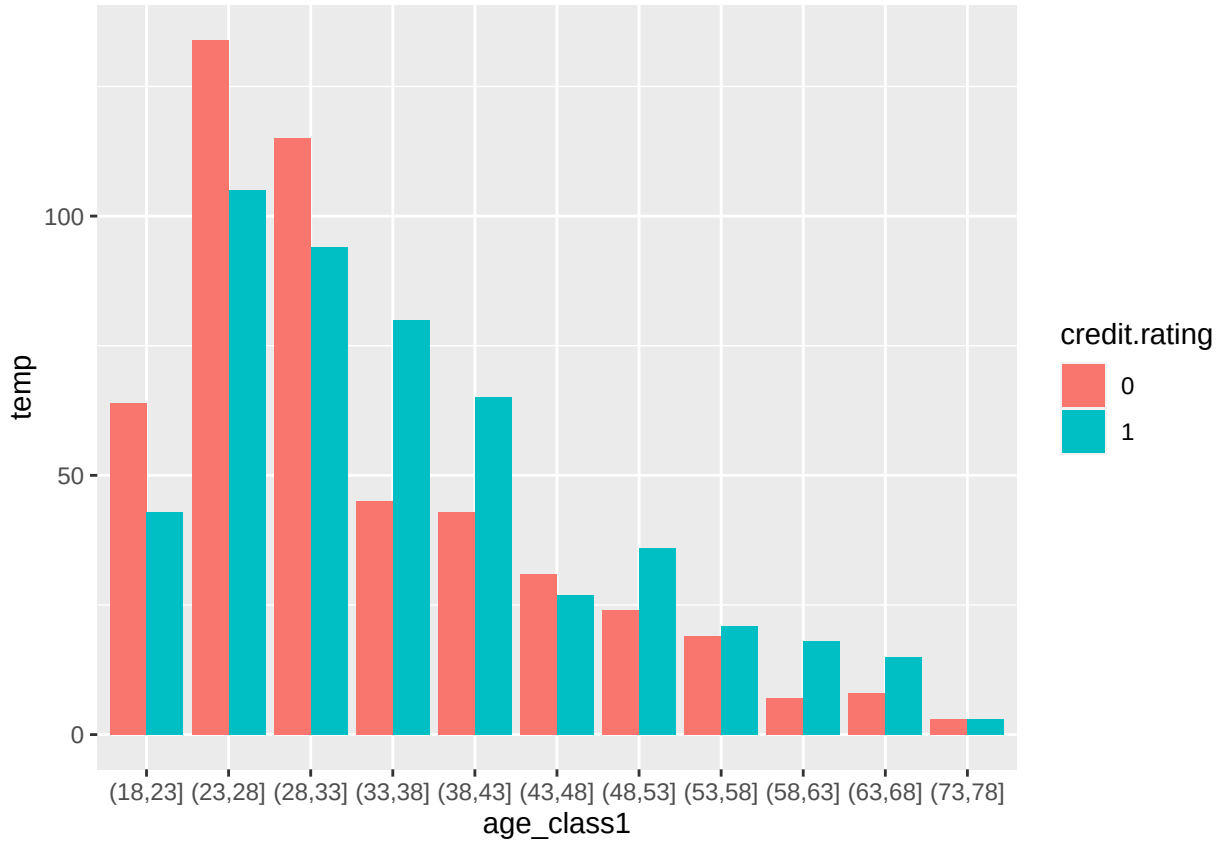
In order to assess the relationship of the created categorical variables and the target variable, a graphical analysis was performed. The distribution of the created variables (age_class1, age_class2, credit.duration_class1, credit.duration_class2, credit.amount_class1, credit.amount_class2 adn credit.amount_class3) were plotted considering the classes of the target variable separately.

age

```
# class1
df.bal %>%
  group_by(credit.rating, age_class1) %>%
  dplyr::summarise(temp = n()) %>%
```

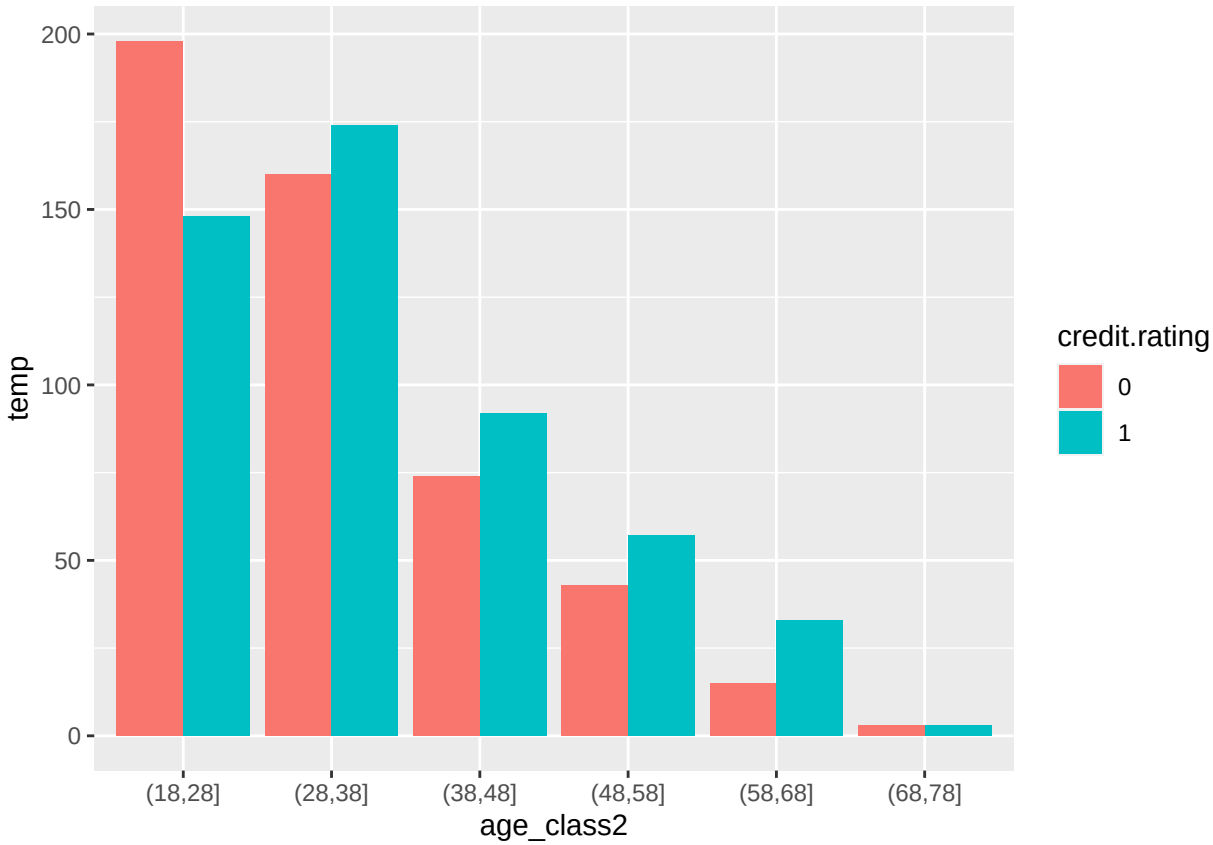
```
  dplyr::arrange(desc(age_class1)) %>%
  ggplot() +
  geom_bar(aes(x=age_class1, y=temp, fill=credit.rating),
           stat = 'identity',
           position = position_dodge())
```

```
## `summarise()` has grouped output by 'credit.rating'. You can override using the `.groups` argument.
```




```
# class2
df.bal %>%
  group_by(credit.rating, age_class2) %>%
  dplyr::summarise(temp = n()) %>%
  dplyr::arrange(desc(age_class2)) %>%
  ggplot() +
  geom_bar(aes(x=age_class2, y=temp, fill=credit.rating),
           stat = 'identity',
           position = position_dodge())
```

```
## `summarise()` has grouped output by 'credit.rating'. You can override using the `.groups` argument.
```



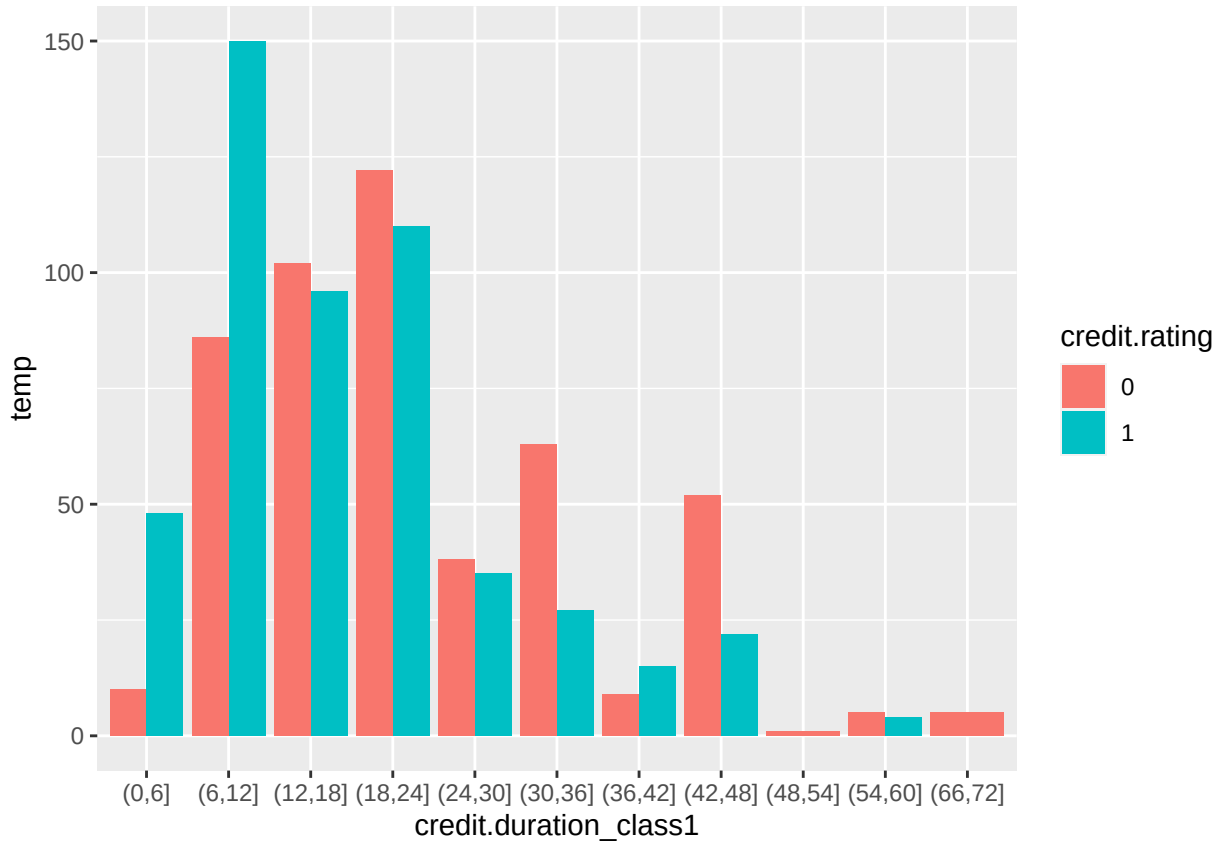
From the analysis of the created categorical variables age_class1 and age_class2, it was observed that the proportion of "bad payers" are much higher among younger clients, presenting ages between 18 to 33, compared to older clients. It seems that there was a relationship between this new variable and the target variable.

P.S.: *It is relevant to mention that when we use the term 'clients' in this project, we are referring to the clients presented i*

credit.duration

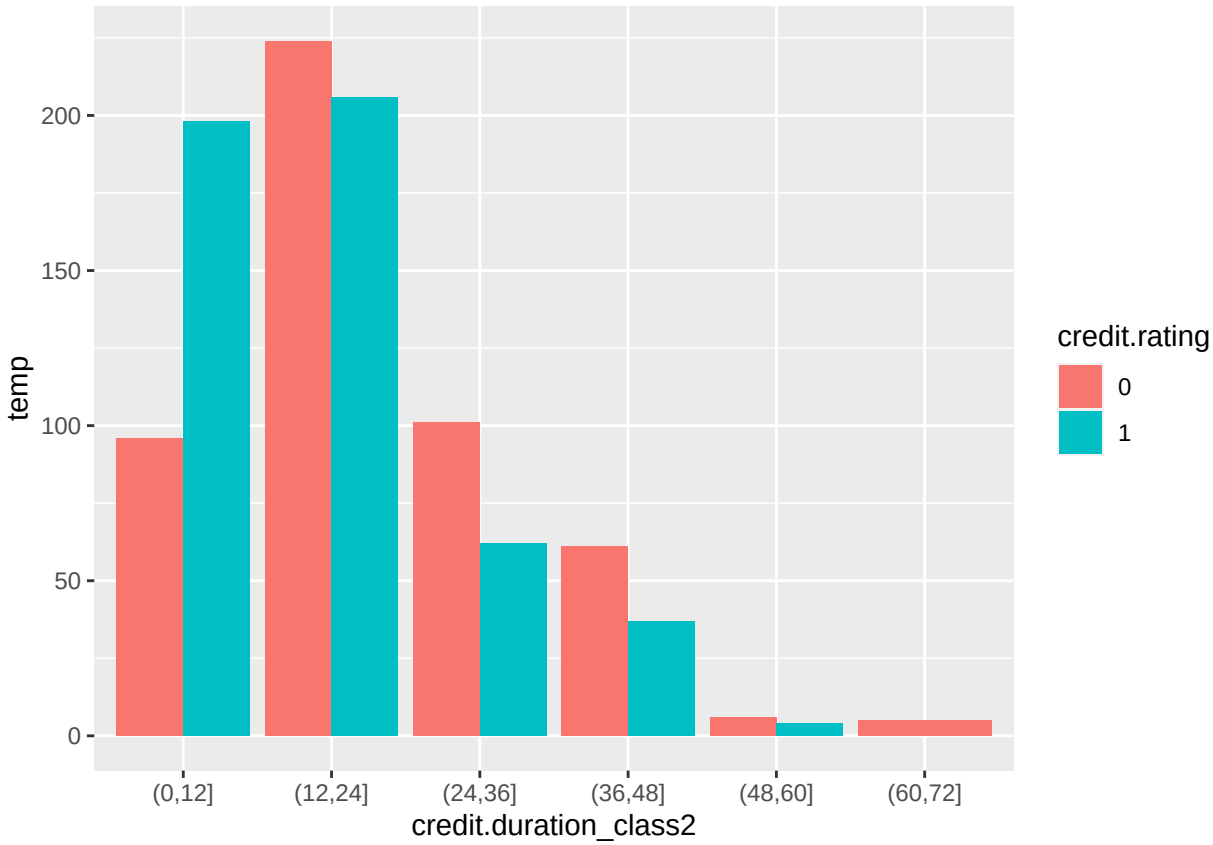
```
# class1
df.bal %>%
  group_by(credit.rating, credit.duration_class1) %>%
  dplyr::summarise(temp = n()) %>%
  dplyr::arrange(desc(credit.duration_class1)) %>%
  ggplot() +
  geom_bar(aes(x=credit.duration_class1, y=temp, fill=credit.rating),
           stat = 'identity',
           position = position_dodge())
```

```
## `summarise()` has grouped output by 'credit.rating'. You can override using the `.groups` argument.
```




```
# class2
df.bal %>%
  group_by(credit.rating, credit.duration_class2) %>%
  dplyr::summarise(temp = n()) %>%
  dplyr::arrange(desc(credit.duration_class2)) %>%
  ggplot() +
  geom_bar(aes(x=credit.duration_class2, y=temp, fill=credit.rating),
           stat = 'identity',
           position = position_dodge())
```

```
## `summarise()` has grouped output by 'credit.rating'. You can override using the `.groups` argument.
```

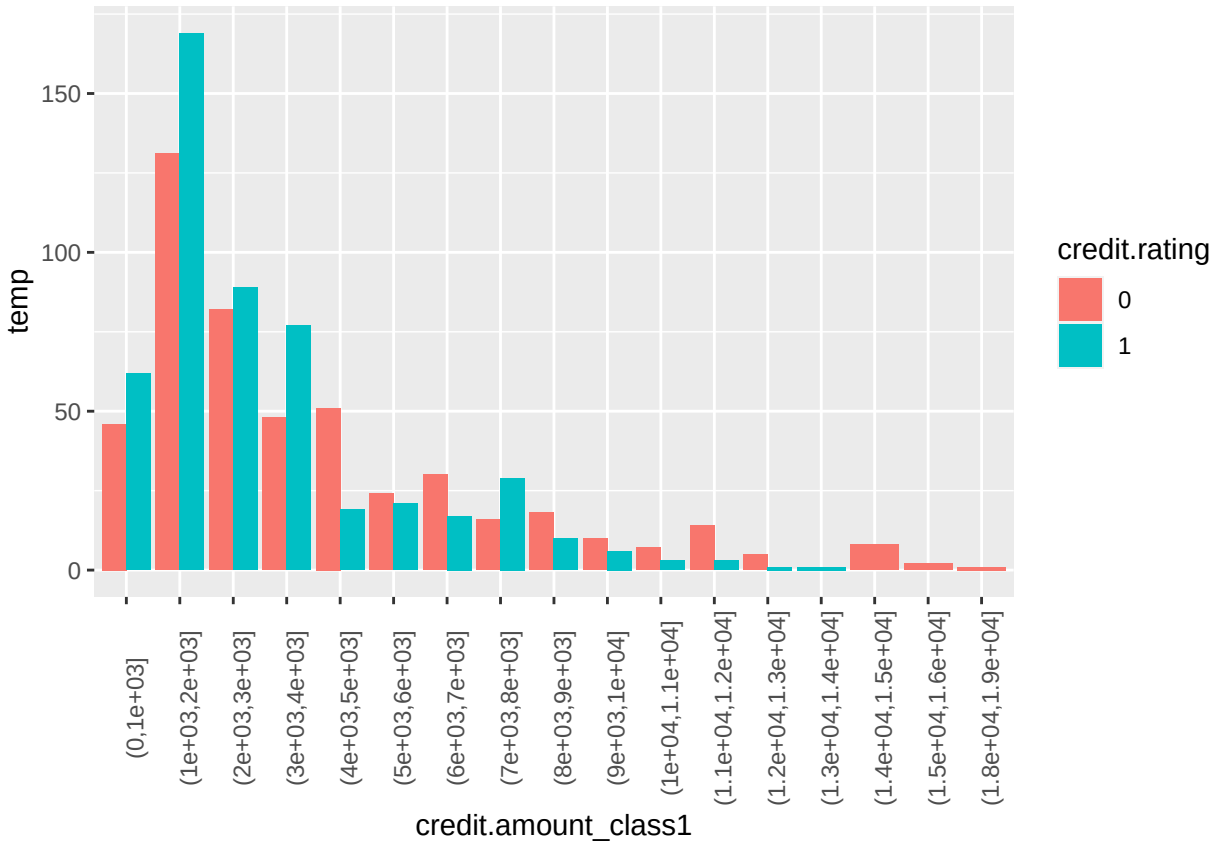


In this analysis, it seems that clients who are granted credit for shorter periods tended to be 'better payers' compared to credit grants for long periods, especially for periods over 2 years. Again, these variables seemed to be also related to the target variable.

credit.amount

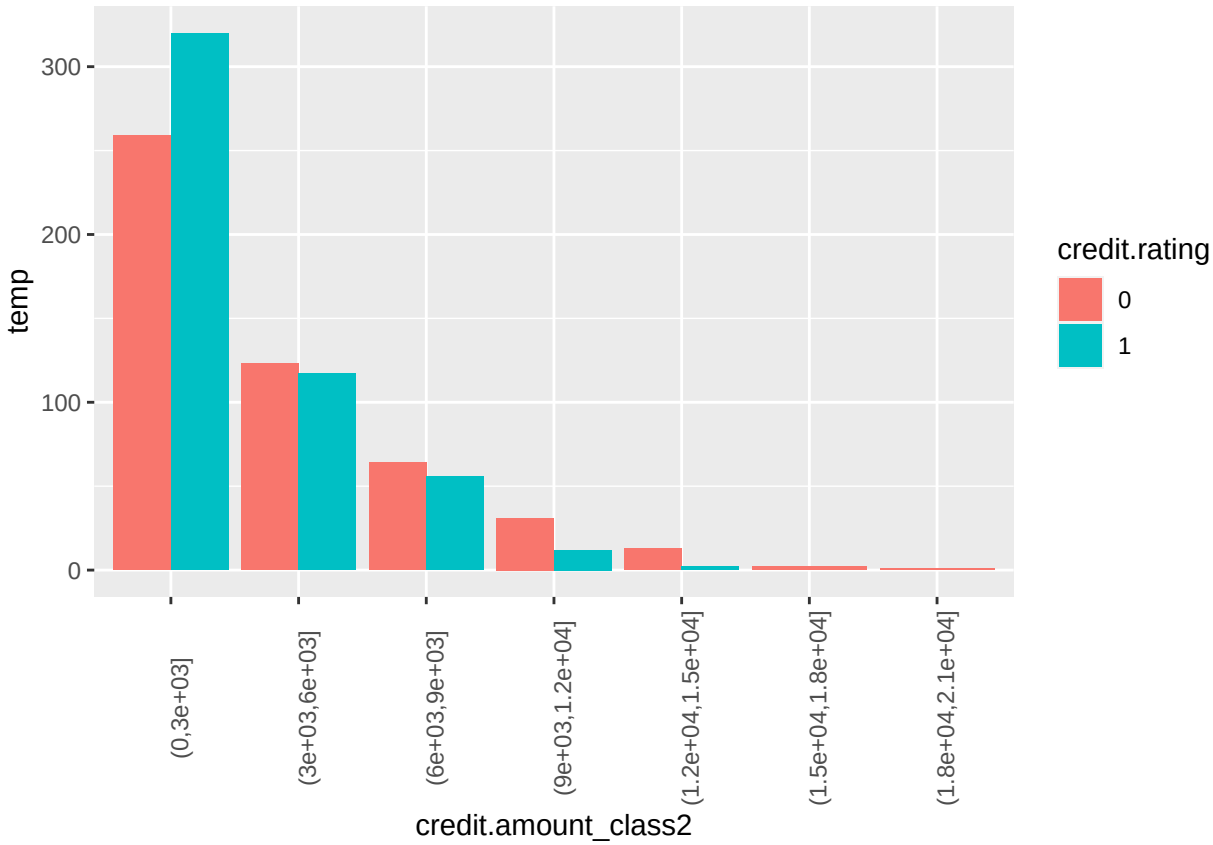
```
# class1
df.bal %>%
  group_by(credit.rating, credit.amount_class1) %>%
  dplyr::summarise(temp = n()) %>%
  dplyr::arrange(desc(credit.amount_class1)) %>%
  ggplot() +
  geom_bar(aes(x=credit.amount_class1, y=temp, fill=credit.rating),
           stat = 'identity',
           position = position_dodge()) +
  theme(
    axis.text.x = element_text(angle = 90)
    )
```

```
## `summarise()` has grouped output by 'credit.rating'. You can override using the `.groups` argument.
```



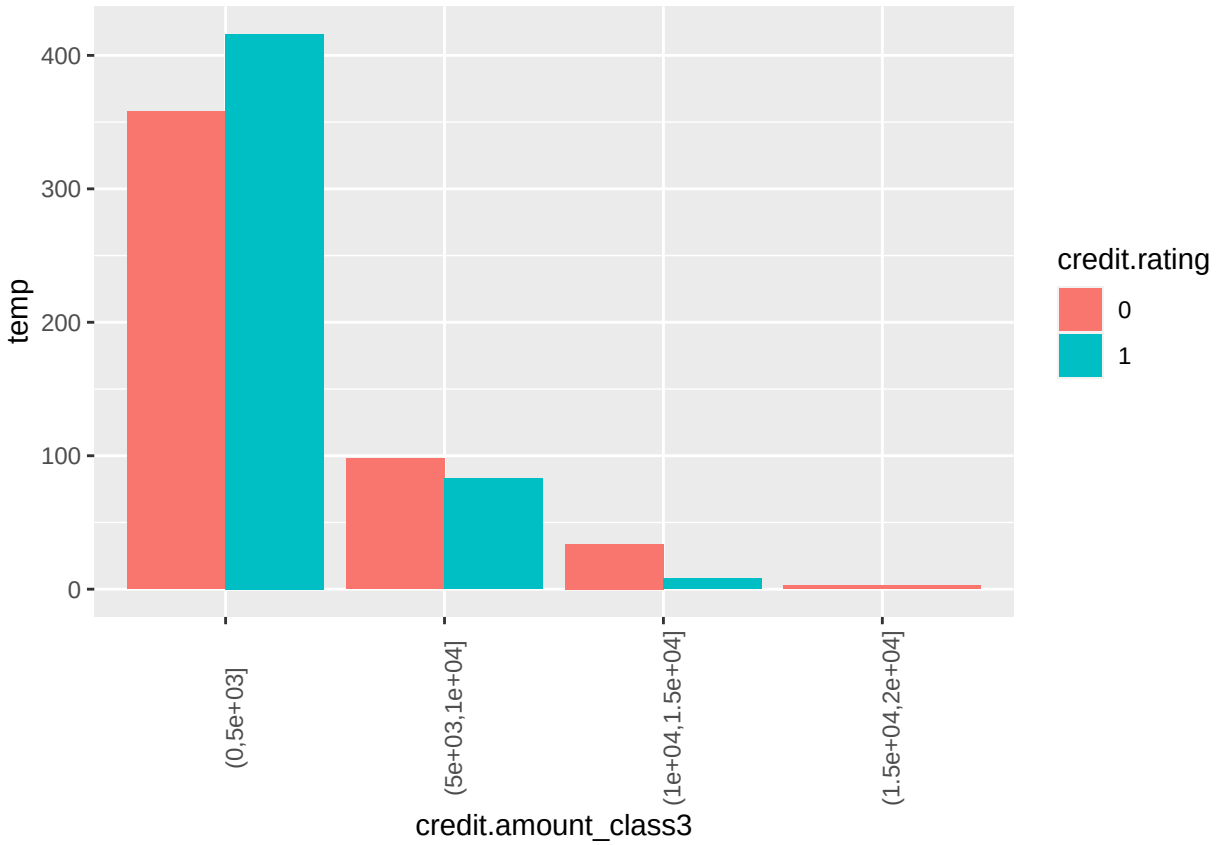

```
# class2
df.bal %>%
  group_by(credit.rating, credit.amount_class2) %>%
  dplyr::summarise(temp = n()) %>%
  dplyr::arrange(desc(credit.amount_class2)) %>%
  ggplot() +
  geom_bar(aes(x=credit.amount_class2, y=temp, fill=credit.rating),
           stat = 'identity',
           position = position_dodge()) +
  theme(
    axis.text.x = element_text(angle = 90)
    )
```

```
## `summarise()` has grouped output by 'credit.rating'. You can override using the `.groups` argument.
```




```
# class3
df.bal %>%
  group_by(credit.rating, credit.amount_class3) %>%
  dplyr::summarise(temp = n()) %>%
  dplyr::arrange(desc(credit.amount_class3)) %>%
  ggplot() +
  geom_bar(aes(x=credit.amount_class3, y=temp, fill=credit.rating),
           stat = 'identity',
           position = position_dodge()) +
  theme(
    axis.text.x = element_text(angle = 90)
    )
```

```
## `summarise()` has grouped output by 'credit.rating'. You can override using the `.groups` argument.
```



Considering the categorical credit.amount variables, it was observed that clients who borrow larger amounts of money are less prone to pay back (bad payers) compared to the ones that borrow smaller amounts.

Following that, the Chi-squared test was applied to all categorical variables and the Crammer's V index was calculated. This index is a measure of association between two nominal variables. To do so, two lists to hold the Chi-squared and the V results of the test were created. The test was performed and the lists were filled with the outcomes of the test.

```
defaultW <- getOption("warn")
options(warn = -1)

# Creating the lists to hold the results
vList <- list()
chiList <- list()

# Performing the test for all categorical variables combined with the target variable
for (i in colnames(df.bal)[-c(1,3,6,14)]) {
  chi <- chisq.test(df.bal$credit.rating, df.bal[,i])
  tbl <- table(df.bal$credit.rating, df.bal[,i])

  v <- sqrt(chi$statistic / sum(tbl))
  vList[[i]] <- v
  chiList[[i]] <- chi
  }
options(warn = defaultW)
```

Next, a new data.frame 'chisq_test_results' was created and and fed with the results from the Chi-squared and V lists. Then the data.frame was sorted considering the descent order of variable V.

```
# Creating data.frame to hold all results from chi-squared test
chisq_test_results <- data.frame('X_squared' = rep(0,length(colnames(df.bal)[-c(1,3,6,14)])),
                                 'p-value' = rep(0,length(colnames(df.bal)[-c(1,3,6,14)])),
                                 "Crammer_index_V" = rep(0,length(colnames(df.bal)[-c(1,3,6,14)])),
                                 row.names = colnames(df.bal)[-c(1,3,6,14)])

# Filling in the new data.frame with the results of the previous test
for (i in 1:length(chisq_test_results$X_squared)) {
  chisq_test_results[i,1] <- chiList[[i]]$statistic
  chisq_test_results[i,2] <- chiList[[i]]$p.value
  chisq_test_results[i,3] <- vList[[i]]
  }

# Sorting the dataframe acording to Crammer's index in a descent order
chisq_test_results <- chisq_test_results %>%
  dplyr::arrange(desc(Crammer_index_V))

# Visualizing the test results in a table
chisq_test_results
```

```
##                                  X_squared      p.value Crammer_index_V
## account.balance                185.89731578 4.294482e-41     0.431158110
## credit.duration_class1          77.17067816 1.795740e-12     0.277796109
## savings                         72.87282262 1.034980e-15     0.269949667
## credit.amount_class1            62.86447452 1.710611e-07     0.250727889
## previous.credit.payment.status  62.02693738 3.396422e-14     0.249052078
## credit.duration_class2          56.56516961 6.215774e-11     0.237834332
## credit.purpose                  37.44329261 3.707538e-08     0.193502694
## age_class1                      33.58873335 2.167772e-04     0.183272293
## current.assets                  28.49034368 2.865504e-06     0.168790828
## employment.duration             26.72728887 6.715565e-06     0.163484828
## credit.amount_class2            26.38111712 1.890312e-04     0.162422650
## credit.amount_class3            24.49338595 1.970369e-05     0.156503629
## marital.status                  23.60096554 7.500936e-06     0.153626057
## other.credits                   20.51761648 5.908495e-06     0.143239717
## age_class2                      18.28165031 2.613363e-03     0.135209653
## apartment.type                  14.74480977 6.283552e-04     0.121428208
## foreign.worker                   9.85964388 1.689438e-03     0.099295739
## installment.rate                 8.57033850 3.558407e-02     0.092576123
## bank.credits                     5.15327671 2.320255e-02     0.071786327
## telephone                        4.74627519 2.936178e-02     0.068893216
## residence.duration               3.83996542 2.792716e-01     0.061967454
## occupation                       3.76615031 2.878480e-01     0.061368969
## guarantor                        0.06107719 8.048014e-01     0.007815190
## dependents                       0.03873813 8.439684e-01     0.006223996
```

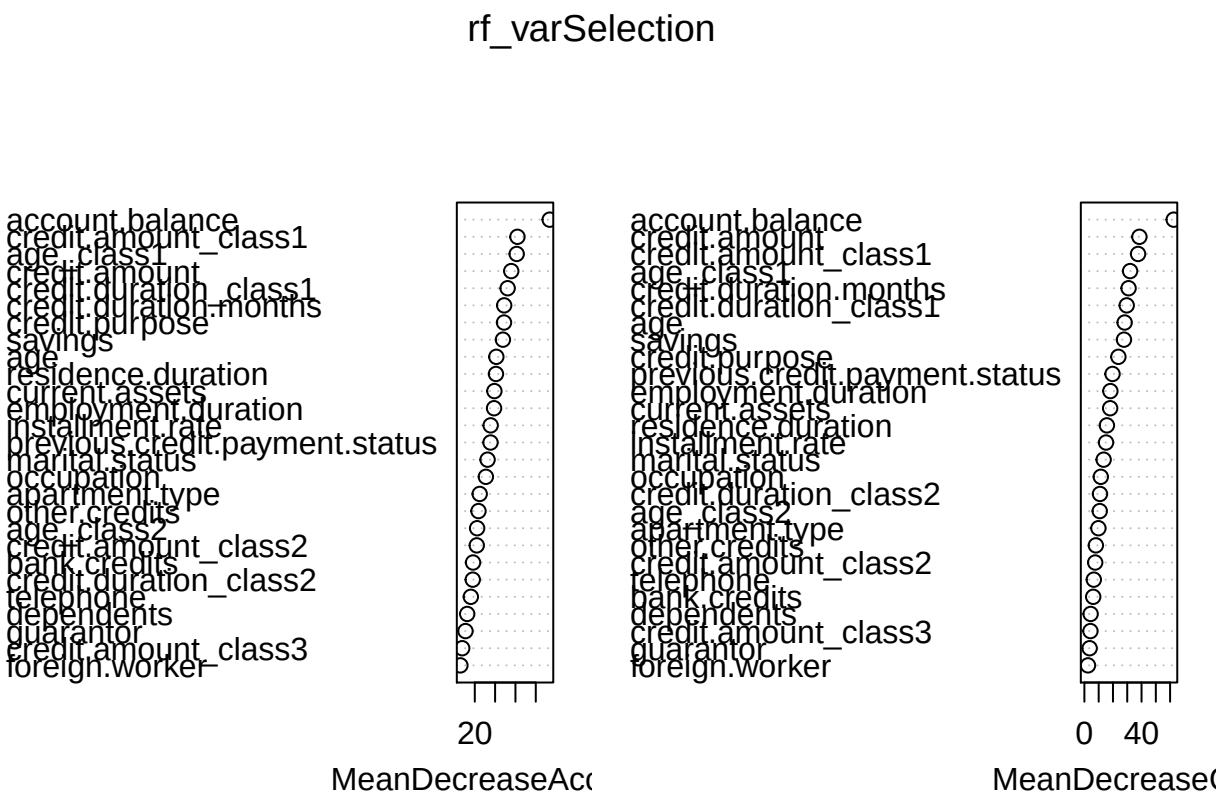
Considering a significance level of 0.05, it was observed that a few categorical variables do not have a significant association with the target variable, while others seems to hold a stronger relationship with the target. The results of this analysis is intended to support feature selection later on.

But before selecting the variables that will be part o the model, the Random Forest algorithm was used as another technique for feature selection, and applied, in this case, to check the most important variables in our data set. Check it out.

```r
# Creating a model with random forest
rf_varSelection <- randomForest(credit.rating ~ ., data = df.bal, importance = TRUE)

# Visualizing the results
rf_varSelection
```

```
##
## Call:
##  randomForest(formula = credit.rating ~ ., data = df.bal, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 9%
## Confusion matrix:
##     0   1 class.error
## 0 453  40  0.08113590
## 1  50 457  0.09861933
```

```r
# Plotting variables importance
varImpPlot(rf_varSelection)
```



Well, it is important to remember that a few variables in the data set presented the same information. This happened because we have created a few variables from other variables presented in the data set, and these variables are certainly related.

The previous analysis showed that among the created categorical variables, the variables considering shorter intervals (class1) were more relevant to the model compared to the variables with longer intervals. Therefore the variables age_class2, credit.duration_class2, credit.amount_class2 and credit.amount_class3 were removed from the data set.

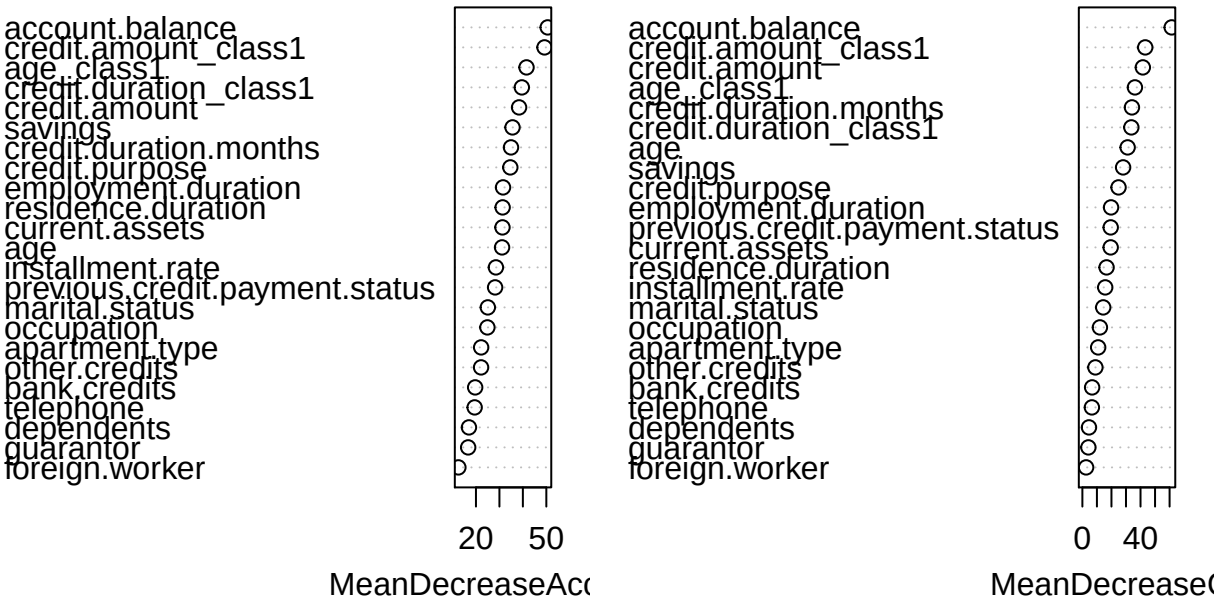
```r
# Removing class2 and class3 variables
df.bal <- df.bal %>%
  select(-c(23, 25, 27, 28))
```

Following the removal of less important created variables from the data set, there are still, at least, three variables with duplicate information, which are the numerical variables age, credit.duration.months and credit.amount and their corresponding categorical variables age_class1, credit.duration_class1 and credit.amount_class1, respectively. However before selecting which ones should be kept for the following steps, the importance of variables was checked again with the Random Forest method for feature selection.

```r
# Creating another model with random forest
rf_varSelection2 <- randomForest(credit.rating ~ ., data = df.bal, importance = TRUE)

# Visualizing the results
rf_varSelection2
```

```
##
## Call:
##  randomForest(formula = credit.rating ~ ., data = df.bal, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 8.6%
## Confusion matrix:
##     0   1 class.error
## 0 452  41   0.0831643
## 1  45 462   0.0887574
```

```r
# Plotting variables importance
varImpPlot(rf_varSelection2)
```



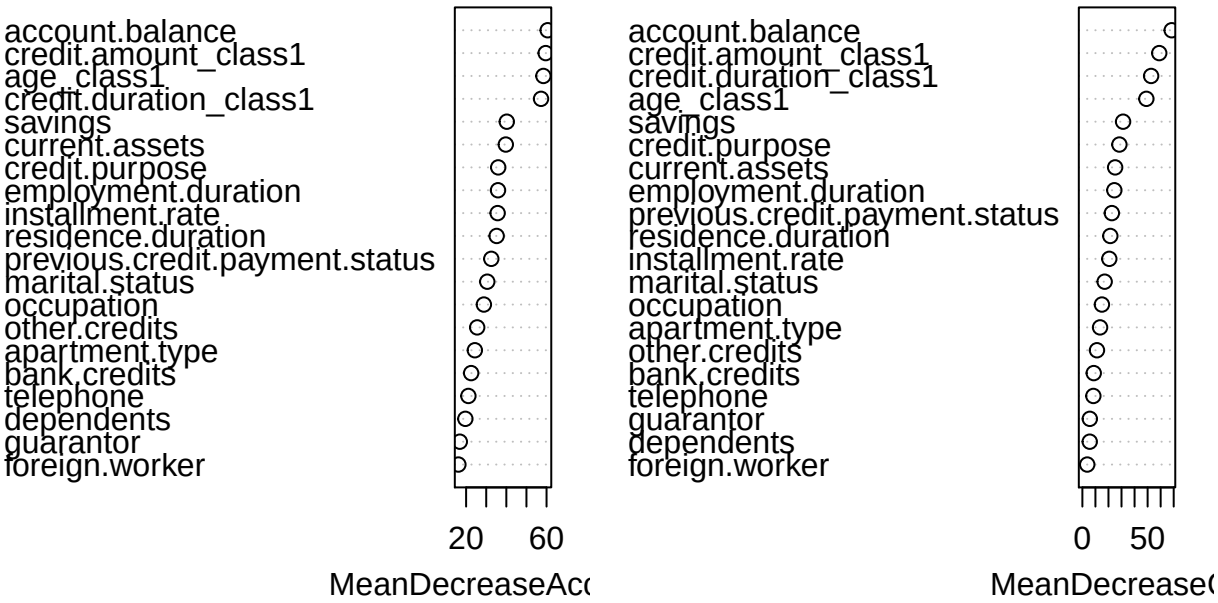
Two other test were performed during this step of feature selection. Firstly, the Random Forest algorithm was used to evaluate the importance of predictors considering the removal of numerical variables, whereas in the second test, the removal of the corresponding created categorical variables were considered. In both tests we have eliminated the duplicate problem caused by the insertion of the new variables in the data set.

Without the numerical variables:

```
# Creating another model with random forest
rf_varSelection3 <- randomForest(credit.rating ~ ., data = df.bal[,-c(3,6,14)], importance = TRUE)

# Visualizing the results
rf_varSelection3
```

```
##
## Call:
##  randomForest(formula = credit.rating ~ ., data = df.bal[, -c(3,      6, 14)], importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 8.7%
## Confusion matrix:
##     0   1 class.error
## 0 451  42   0.0851927
## 1  45 462   0.0887574
```

```
# Plotting variables importance
varImpPlot(rf_varSelection3)
```

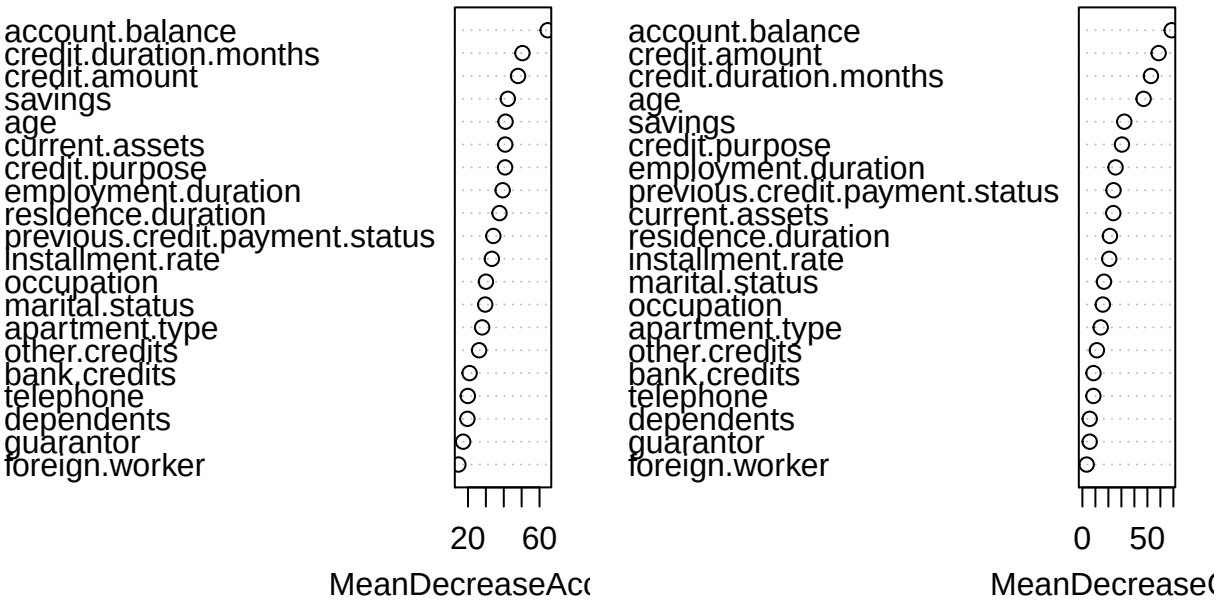


Without the created categorical variables corresponding to the numeric variables

```
# Creating another model with random forest
rf_varSelection4 <- randomForest(credit.rating ~ ., data = df.bal[,-c(22,23,24)], importance = TRUE)

# Visualizing the results
rf_varSelection4
```

```
##
## Call:
##  randomForest(formula = credit.rating ~ ., data = df.bal[, -c(22,      23, 24)], importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 9.8%
## Confusion matrix:
##     0   1 class.error
## 0 459  34  0.06896552
## 1  64 443  0.12623274
```

```
# Plotting variables importance
varImpPlot(rf_varSelection4)
```



As the Random Forest model created without the numerical variables presented a performance slightly better (lower error rate). Moreover, when the created categorical variables were used, their importance were higher compared to the importance of the corresponding variables the numeric ones were used. Therefore, the numerical variables were definitely removed from the data set, which was used for the following analysis.

From now on, only categorical variables are part of the data set, as it can be seen below.

```
# Removing the numerical variables
df.bal <- df.bal[,-c(3,6,14)]

# Checking data types again
glimpse(df.bal)
```

```
## Rows: 1,000
## Columns: 21
## $ credit.rating                  <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ account.balance                <fct> 3, 1, 3, 2, 3, 1, 3, 1, 2, 3, 3, 3, 3, ~
## $ previous.credit.payment.status <fct> 3, 3, 2, 3, 1, 3, 3, 1, 2, 2, 2, 2, 3, ~
## $ credit.purpose                 <fct> 3, 4, 3, 3, 3, 1, 3, 2, 4, 4, 3, 4, 3, ~
## $ savings                        <fct> 4, 1, 3, 1, 2, 1, 1, 3, 1, 4, 1, 1, 1, ~
## $ employment.duration            <fct> 3, 1, 4, 4, 4, 4, 3, 2, 4, 1, 4, 4, 3, ~
## $ installment.rate               <fct> 4, 2, 3, 3, 4, 4, 2, 4, 1, 4, 4, 2, 2, ~
## $ marital.status                 <fct> 3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, ~
## $ guarantor                      <fct> 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, ~
## $ residence.duration             <fct> 2, 4, 4, 4, 4, 4, 3, 4, 1, 4, 2, 4, 3, ~
## $ current.assets                 <fct> 2, 1, 3, 1, 3, 2, 1, 1, 1, 2, 3, 2, 1, ~
## $ other.credits                  <fct> 2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 2, 2, ~
## $ apartment.type                 <fct> 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ bank.credits                   <fct> 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, ~
## $ occupation                     <fct> 3, 2, 3, 2, 2, 3, 2, 3, 4, 4, 3, 4, 2, ~
```

```
## $ dependents                    <fct> 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 2, ~
## $ telephone                     <fct> 2, 1, 1, 2, 1, 2, 1, 2, 2, 2, 1, 2, 1, ~
## $ foreign.worker                <fct> 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ age_class1                    <fct> "(23,28]", "(38,43]", "(38,43]", "(43,4~
## $ credit.duration_class1        <fct> "(6,12]", "(0,6]", "(18,24]", "(12,18]"~
## $ credit.amount_class1          <fct> "(1e+03,2e+03]", "(1e+03,2e+03]", "(2e+~
```

The Chi-squared test was applied to all categorical variables of the new data set again, which considered the removal of some duplicate variables, and the Crammer's V index was calculated. To do so, another two lists to hold the Chi-squared and the V results of this test were created. The test was performed and the lists were filled in.

```
defaultW <- getOption("warn")
options(warn = -1)
# Creating the lists to hold the results
vList2 <- list()
chiList2 <- list()

# Performing the test for all categorical variables combined with the target variable
for (i in colnames(df.bal)[-1]) {
  chi <- chisq.test(df.bal$credit.rating, df.bal[,i])
  tbl <- table(df.bal$credit.rating, df.bal[,i])

  v <- sqrt(chi$statistic / sum(tbl))
  vList2[[i]] <- v
  chiList2[[i]] <- chi
  }
options(warn = defaultW)
```

A new data.frame 'chisq_test_results2' was created and fed with the results from the Chi-squared and V lists. Then the data.frame was sorted considering the descent order of variable V.

```
# Creating data.frame to hold all results from chi-squared test
chisq_test_results2 <- data.frame('X_squared' = rep(0,length(colnames(df.bal)[-1])),
                                  'p-value' = rep(0,length(colnames(df.bal)[-1])),
                                  "Crammer_index_V" = rep(0,length(colnames(df.bal)[-1])),
                                  row.names = colnames(df.bal)[-1])

# Filling in the new data.frame with the results of the previous test
for (i in 1:length(chisq_test_results2$X_squared)) {
  chisq_test_results2[i,1] <- chiList2[[i]]$statistic
  chisq_test_results2[i,2] <- chiList2[[i]]$p.value
  chisq_test_results2[i,3] <- vList2[[i]]
  }

# Sorting the dataframe acording to Crammer's index in a descent order
chisq_test_results2 <- chisq_test_results2 %>%
  dplyr::arrange(desc(Crammer_index_V))

# Visualizing the test results in a table
chisq_test_results2
```

```
##                                X_squared      p.value Crammer_index_V
## account.balance               185.89731578 4.294482e-41     0.431158110
## credit.duration_class1         77.17067816 1.795740e-12     0.277796109
## savings                        72.87282262 1.034980e-15     0.269949667
```

```
## credit.amount_class1            62.86447452 1.710611e-07     0.250727889
## previous.credit.payment.status  62.02693738 3.396422e-14     0.249052078
## credit.purpose                  37.44329261 3.707538e-08     0.193502694
## age_class1                      33.58873335 2.167772e-04     0.183272293
## current.assets                  28.49034368 2.865504e-06     0.168790828
## employment.duration             26.72728887 6.715565e-06     0.163484828
## marital.status                  23.60096554 7.500936e-06     0.153626057
## other.credits                   20.51761648 5.908495e-06     0.143239717
## apartment.type                  14.74480977 6.283552e-04     0.121428208
## foreign.worker                   9.85964388 1.689438e-03     0.099295739
## installment.rate                 8.57033850 3.558407e-02     0.092576123
## bank.credits                     5.15327671 2.320255e-02     0.071786327
## telephone                        4.74627519 2.936178e-02     0.068893216
## residence.duration               3.83996542 2.792716e-01     0.061967454
## occupation                       3.76615031 2.878480e-01     0.061368969
## guarantor                        0.06107719 8.048014e-01     0.007815190
## dependents                       0.03873813 8.439684e-01     0.006223996
```

In the following steps, a few sets of predictor variables were created for testing. The selected variables for each set of predictors are printed right after its creation.

The first set comprised all the predictor variables contained in the updated data set (**df.bal**). After that, the predictor variables that did not present a significant association with target variable, considering a significance level of 0.05 were removed from the data set.

```
# First set of predictors - entire dataset
testVars1 <- colnames(df.bal)
writeLines(c('Predictors:', testVars1))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## installment.rate
## marital.status
## guarantor
## residence.duration
## current.assets
## other.credits
## apartment.type
## bank.credits
## occupation
## dependents
## telephone
## foreign.worker
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
# Second set of predictors - removing a few variables that presented p-value > 0.05 during the chi-squa
testVars2 <- testVars1[-c(9,10,15,16)]
writeLines(c('Predictors:', testVars2))
```

```
## Predictors:
```

```
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## installment.rate
## marital.status
## current.assets
## other.credits
## apartment.type
## bank.credits
## telephone
## foreign.worker
## age_class1
## credit.duration_class1
## credit.amount_class1
```

Other 11 sets of predictors were created based on the second set. In a crescent order, the sets 3 to 13 were created by removing one variable from the previous set. This variable was the less important one, based on the Crammer's V index. The set 13, which presents the lower number of variables, contains 5 predictors, while set 3 presents 13 predictors, as shown below.

```
testVars3 <- testVars2[-13]
writeLines(c('Predictors:', testVars3))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## installment.rate
## marital.status
## current.assets
## other.credits
## apartment.type
## bank.credits
## foreign.worker
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars4 <- testVars3[-12]
writeLines(c('Predictors:', testVars4))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## installment.rate
## marital.status
```

```
## current.assets
## other.credits
## apartment.type
## foreign.worker
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars5 <- testVars4[-7]
writeLines(c('Predictors:', testVars5))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## marital.status
## current.assets
## other.credits
## apartment.type
## foreign.worker
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars6 <- testVars5[-11]
writeLines(c('Predictors:', testVars6))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## marital.status
## current.assets
## other.credits
## apartment.type
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars7 <- testVars6[-10]
writeLines(c('Predictors:', testVars7))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## marital.status
```

```
## current.assets
## other.credits
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars8 <- testVars7[-9]
writeLines(c('Predictors:', testVars8))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## marital.status
## current.assets
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars9 <- testVars8[-7]
writeLines(c('Predictors:', testVars9))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## employment.duration
## current.assets
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars10 <- testVars9[-6]
writeLines(c('Predictors:', testVars10))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## current.assets
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars11 <- testVars10[-6]
writeLines(c('Predictors:', testVars11))
```

```
## Predictors:
## credit.rating
## account.balance
```

```
## previous.credit.payment.status
## credit.purpose
## savings
## age_class1
## credit.duration_class1
## credit.amount_class1
```

```
testVars12 <- testVars11[-6]
writeLines(c('Predictors:', testVars12))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## credit.purpose
## savings
## credit.duration_class1
## credit.amount_class1
```

```
testVars13 <- testVars12[-4]
writeLines(c('Predictors:', testVars13))
```

```
## Predictors:
## credit.rating
## account.balance
## previous.credit.payment.status
## savings
## credit.duration_class1
## credit.amount_class1
```

Visualizing the results from the importance test of variables using the Random Forest model 3 to support the selection of some more sets of predictors to be tested.

```
# Visualizing the importance of variables
data.frame(feature = rownames(varImp(rf_varSelection3)),
           importance_value = data.frame(varImp(rf_varSelection3))$X0) %>%
           dplyr::arrange(desc(importance_value))
```

```
##                          feature importance_value
## 1                 account.balance         52.44008
## 2          credit.duration_class1         46.34781
## 3            credit.amount_class1         44.94758
## 4                      age_class1         43.54656
## 5                         savings         33.41348
## 6                 current.assets         31.49635
## 7                 credit.purpose         29.94228
## 8             employment.duration         27.66183
## 9                installment.rate         27.64454
## 10 previous.credit.payment.status         26.68843
## 11             residence.duration         26.19004
## 12                 marital.status         25.26688
## 13                  other.credits         22.26788
## 14                     occupation         21.90258
## 15                 apartment.type         19.97690
## 16                   bank.credits         17.91045
## 17                      telephone         16.44441
## 18                     dependents         14.30817
```

```
## 19                      guarantor         13.06471
## 20                foreign.worker         12.73453
```

From the results shown above, it was decided to create more 13 sets of predictors to be part of the following tests. In this case, the set 14 (first of this step) was filled with the 5 most important variables according to the mean Gini index. From this set to the last one, predictors were added one by one, following the descent order of the mean Gini index, resulting in the set 26 (17 predictors).

```
testVars14 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,7)])
writeLines(c('Predictors:', testVars14))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## age_class1
```

```
testVars15 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,7,8)])
writeLines(c('Predictors:', testVars15))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## age_class1
## current.assets
```

```
testVars16 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,6,7,8)])
writeLines(c('Predictors:', testVars16))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## credit.purpose
## age_class1
## current.assets
```

```
testVars17 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,6,7,8,9)])
writeLines(c('Predictors:', testVars17))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## credit.purpose
## age_class1
## current.assets
## employment.duration
```

```
testVars18 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,6,7,8,9,14)])
writeLines(c('Predictors:', testVars18))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## credit.purpose
## age_class1
## current.assets
## employment.duration
## installment.rate
```

```
testVars19 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,5,6,7,8,9,14)])
writeLines(c('Predictors:', testVars19))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## previous.credit.payment.status
## credit.purpose
## age_class1
## current.assets
## employment.duration
## installment.rate
```

```
testVars20 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,5,6,7,8,9,14,17)])
writeLines(c('Predictors:', testVars20))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## previous.credit.payment.status
## credit.purpose
## age_class1
## current.assets
## employment.duration
## installment.rate
## residence.duration
```

```
testVars21 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,5,6,7,8,9,10,14,17)])
writeLines(c('Predictors:', testVars21))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
```

```
## credit.amount_class1
## previous.credit.payment.status
## credit.purpose
## age_class1
## current.assets
## employment.duration
## marital.status
## installment.rate
## residence.duration
```

```
testVars22 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,5,6,7,8,9,10,11,14,17)])
writeLines(c('Predictors:', testVars22))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## previous.credit.payment.status
## credit.purpose
## age_class1
## current.assets
## employment.duration
## marital.status
## other.credits
## installment.rate
## residence.duration
```

```
testVars23 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,5,6,7,8,9,10,11,14,17,18)])
writeLines(c('Predictors:', testVars23))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## previous.credit.payment.status
## credit.purpose
## age_class1
## current.assets
## employment.duration
## marital.status
## other.credits
## installment.rate
## residence.duration
## occupation
```

```
testVars24 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,5,6,7,8,9,10,11,12,14,17,18)])
writeLines(c('Predictors:', testVars24))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
```

```
## savings
## credit.amount_class1
## previous.credit.payment.status
## credit.purpose
## age_class1
## current.assets
## employment.duration
## marital.status
## other.credits
## apartment.type
## installment.rate
## residence.duration
## occupation
```

```
testVars25 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,5,6,7,8,9,10,11,12,14,15,17,18
writeLines(c('Predictors:', testVars25))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## previous.credit.payment.status
## credit.purpose
## age_class1
## current.assets
## employment.duration
## marital.status
## other.credits
## apartment.type
## installment.rate
## bank.credits
## residence.duration
## occupation
```

```
testVars26 <- c('credit.rating', rownames(chisq_test_results2)[c(1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,17
writeLines(c('Predictors:', testVars26))
```

```
## Predictors:
## credit.rating
## account.balance
## credit.duration_class1
## savings
## credit.amount_class1
## previous.credit.payment.status
## credit.purpose
## age_class1
## current.assets
## employment.duration
## marital.status
## other.credits
## apartment.type
## installment.rate
## bank.credits
```

```
## telephone
## residence.duration
## occupation
```

Following the selection of sets of predictor variables (26 in total) to be used and tested during the creation of machine learning models, in order to facilitate the next step of this project, a list to hold all these sets that was created.

```
# Preparing a list with all the set of variables that will be tested
testList <- list(testVars1, testVars2, testVars3, testVars4, testVars5, testVars6, testVars7, testVars8
testVars24, testVars25, testVars26)
```

At this point, the data munging, exploratory analysis and feature selection tasks were concluded. Therefore, from now on, the following steps take care of the machine learning models creation and evaluation. In the end, the best model was selected and a summary of its parameters as well as information on the model evaluation are presented.

First, the data set **df.bal** was splitted into training data and testing data. Then their dimension were checked just to guarantee that the split was performed as intended.

```
# Creating an index list to separate the train and test datasets
indexes_t <- as.integer(sample(rownames(df.bal), size = 0.7*length(rownames(df.bal)), replace = F))

# Dividing the dataset into train_data and test_data
train_data <- df.bal[indexes_t,]
test_data <- df.bal[-indexes_t,]

# Checking the dimensions of the train_data and test_data
dim(train_data)
```

```
## [1] 700  21
```

```
dim(test_data)
```

```
## [1] 300  21
```

As the data was correctly split and the 26 sets of predictors prepared for the tests, three algorithms were selected to train the different versions of machine learning models that were created in the following steps. The selected algorithms are presented below:

- *Random Forest*
- *Support Vector Machines*
- *Naive Bayes*

In total there were created 104 models, which were evaluated based on many factors, but especially on its accuracy. Using the algorithms Random Forest and Naibe Bayes, 52 models were created, being 26 for each algorithm. In the case of Support Vector Machines, 52 models were created, but 26 used the kernel configured to 'Linear', whereas the other 26 models used the kernel configured to 'Radial'.

```
# Creating two lists to hold the results and the accuracy of the RF models

rfModelList <- list(rep(0,length(testList)))
rfTestAccuracyList <- list(rep(0,length(testList)))

# Iterating through the testing variables list to create the strings that
# will represent the formula for the training
for (i in 1:length(testList)) {
```

```
  temp1 <- paste(testList[[i]][1:2], collapse = ' ~ ')
  temp2 <- paste(testList[[i]][3:length(testList[[i]])], collapse = ' + ')
  temp3 <- paste(temp1, temp2, sep = ' + ')

  # Transforming the string version of a formula to a proper formula
  formula1 <- as.formula(object = temp3)

  # Performing the training with RF
  rf <- randomForest(formula1, data = train_data)

  # Appending new results of the created models into the RF model list
  rfModelList[[i]] <- rf

  # Making the predictions for the test data
  pred <- predict(object = rf, newdata = test_data[,testList[[i]][-1]])

  # Calculating the accuracy of the model and appending it to the RF accuracy model list
  rfTestAccuracyList[[i]] <- round(sum(pred == test_data[,1]) / length(test_data[,1]) * 100, 2)
}
```

Random Forest

```
# Creating two lists to hold the results and the accuracy of the SVM Linear models

svmLinearModelList <- list(rep(0,length(testList)))
svmLinearTestAccuracyList <- list(rep(0,length(testList)))

# Iterating through the testing variables list to create the strings that

# will represent the formula for the training

for (i in 1:length(testList)) {

  temp1 <- paste(testList[[i]][1:2], collapse = ' ~ ')

  temp2 <- paste(testList[[i]][3:length(testList[[i]])], collapse = ' + ')

  temp3 <- paste(temp1, temp2, sep = ' + ')

# Transforming the string version of a formula to a proper formula
  formula1 <- as.formula(object = temp3)

# Performing the training with SVM linear
  svm1 <- svm(formula = formula1, data = train_data, type = 'C-classification', kernel = 'linear')

# Appending new results of the created models into the SVM linear model list
  svmLinearModelList[[i]] <- svm1

# Making the predictions for the test data
  pred <- predict(object = svm1, newdata = test_data[,testList[[i]][-1]])

# Calculating the accuracy of the model and appending it to the SVM linear accuracy model list
  svmLinearTestAccuracyList[[i]] <- round(sum(pred == test_data[,1]) / length(test_data[,1]) * 100, 2)
```

```
}
```

Support Vector Machine (SVM) - Linear kernel

```
# Creating two lists to hold the results and the accuracy of the SVM Radial models
svmRadialModelList <- list(rep(0,length(testList)))
svmRadialTestAccuracyList <- list(rep(0,length(testList)))

# Iterating through the testing variables list to create the strings that
# will represent the formula for the training
for (i in 1:length(testList)) {
  temp1 <- paste(testList[[i]][1:2], collapse = ' ~ ')
  temp2 <- paste(testList[[i]][3:length(testList[[i]])], collapse = ' + ')
  temp3 <- paste(temp1, temp2, sep = ' + ')

  # Transforming the string version of a formula to a proper formula
  formula1 <- as.formula(object = temp3)

  # Performing the training with SVM radial
  svm1 <- svm(formula = formula1,
              data = train_data,
              type = 'C-classification',
              kernel = 'radial')

  # Appending new results of the created models into the SVM radial model list
  svmRadialModelList[[i]] <- svm1

  # Making the predictions for the test data
  pred <- predict(object = svm1, newdata = test_data[,testList[[i]][-1]])

  # Calculating the accuracy of the model and appending it to the SVM radial accuracy model list
  svmRadialTestAccuracyList[[i]] <- round(sum(pred == test_data[,1]) / length(test_data[,1]) * 100, 2)
}
```

Support Vector Machine (SVM) - Radial kernel

```
# Creating two lists to hold the results and the accuracy of the NB models
nbModelList <- list(rep(0,length(testList)))
nbTestAccuracyList <- list(rep(0,length(testList)))

# Iterating through the testing variables list to create the strings that
# will represent the formula for the training
for (i in 1:length(testList)) {
  temp1 <- paste(testList[[i]][1:2], collapse = ' ~ ')
  temp2 <- paste(testList[[i]][3:length(testList[[i]])], collapse = ' + ')
  temp3 <- paste(temp1, temp2, sep = ' + ')

  # Transforming the string version of a formula to a proper formula
  formula1 <- as.formula(object = temp3)

  # Performing the training with NB
  nb <- naive_bayes(formula = formula1,
```

```
                    data = train_data,
                    usekernel = T,
                    laplace = 1)

  # Appending new results of the created models into the NB model list
  nbModelList[[i]] <- nb

  # Making the predictions for the test data
  pred <- predict(object = nb, newdata = test_data[,testList[[i]][-1]])

  # Calculating the accuracy of the model and appending it to the NB accuracy model list
  nbTestAccuracyList[[i]] <- round(sum(pred == test_data[,1]) / length(test_data[,1]) * 100, 2)
}
```

Naive Bayes The highest accuracy for each set of tests (26) can be visualized below:

```
# Visualizing the best accuracy for each machine learning algorithm tested
max(data.frame(nbTestAccuracyList))
```

```
## [1] 80.67
```

```
max(data.frame(svmRadialTestAccuracyList))
```

```
## [1] 76
```

```
max(data.frame(svmLinearTestAccuracyList))
```

```
## [1] 79.67
```

```
max(data.frame(rfTestAccuracyList))
```

```
## [1] 88.67
```

```
rfModelList[[3]]
```

```
##
## Call:
##  randomForest(formula = formula1, data = train_data)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.86%
## Confusion matrix:
##     0   1 class.error
## 0 302  50   0.1420455
## 1  47 301   0.1350575
```

Checking which model presented the highest accuracy.

```
rfTestAccuracyList
```

```
## [[1]]
## [1] 88
##
## [[2]]
## [1] 87.67
##
## [[3]]
```

```
## [1] 87.67
##
## [[4]]
## [1] 88.33
##
## [[5]]
## [1] 88
##
## [[6]]
## [1] 88.33
##
## [[7]]
## [1] 88
##
## [[8]]
## [1] 87.67
##
## [[9]]
## [1] 88
##
## [[10]]
## [1] 85.67
##
## [[11]]
## [1] 84.33
##
## [[12]]
## [1] 83
##
## [[13]]
## [1] 79.67
##
## [[14]]
## [1] 81.33
##
## [[15]]
## [1] 85
##
## [[16]]
## [1] 84
##
## [[17]]
## [1] 87
##
## [[18]]
## [1] 87.33
##
## [[19]]
## [1] 88
##
## [[20]]
## [1] 88.67
##
## [[21]]
```

```
## [1] 88.33
##
## [[22]]
## [1] 88
##
## [[23]]
## [1] 87.33
##
## [[24]]
## [1] 87.33
##
## [[25]]
## [1] 88.67
##
## [[26]]
## [1] 86
```

As shown above, two models created with Random Forest presented the highest accuracy. These are the models 20 and 25 of the list. The model 25 uses 16 predictors, while model 20 uses only 11, therefore, in this case we selected model 20.

So, let's check and print out the predictor variables used in this version of the model created with the Random Forest algorithm (testVars20).

```
testVars20
```

```
##  [1] "credit.rating"                 "account.balance"
##  [3] "credit.duration_class1"        "savings"
##  [5] "credit.amount_class1"          "previous.credit.payment.status"
##  [7] "credit.purpose"                "age_class1"
##  [9] "current.assets"                "employment.duration"
## [11] "installment.rate"              "residence.duration"
```

Therefore, the model with highest accuracy (88.7%) used 11 predictors and was trained by the Random Forest algorithm. The predictors are:

account.balance, previous.credit.payment.status, credit.purpose, savings, employment.duration, current.assets, residence.duration, installment.rate, age_class1, credit.duration_class1 and credit.amount_class1

In the next step, a summary of the main measures to evaluate the selected model was prepared.

```
# Making the predictions again using the test data
pred <- predict(rfModelList[[20]], test_data[,-1])

# Creating a Confusion Matrix for the Prediction X Observed
confMat <- confusionMatrix(data = pred, reference = test_data[,1])

# Calculating error_rate of model ( (FN + FP) / (TP + TN + FP + FN) )
error_rate <- (confMat$table[1,2] + confMat$table[2,1]) / sum(confMat$table)

# Calculating the accuracy of the model ( 1 - error_rate )
accuracy <- 1 - error_rate

# Calculating the precision of the model (sensitivity) ( TP / (FP + TP) )
precision <- confMat$table[2,2] / (confMat$table[2,1] + confMat$table[2,2])

# Calculating the recall of the model ( TP / (FN + TP) )
recall <- confMat$table[2,2] / (confMat$table[1,2] + confMat$table[2,2])
```

```
# Calculating the f-measure of model ( (2 \* Precision \* recall) / (Precision \* recall) )
f <- (2 * precision * recall) / (precision + recall)
```

Visualizing the summary of the evaluation parameters for the selected model.

```
# Visualizing the results on the screen
writeLines(c('Error rate: ', paste0(round(error_rate * 100,1), '%'),
             'Accuracy: ', paste0(round(accuracy * 100,1), '%'),
             'Precision: ', paste0(round(precision * 100,1), '%'),
             'Recall: ', paste0(round(recall * 100,1), '%'),
             'f-measure: ', paste0(round(f * 100,1), '%'),
             '\n',
             'Sensitivity: ', round(precision, 3),
             'Specificity: ', round(confMat$table[1,1] / (confMat$table[1,2] + confMat$table[1,1]), 3)))
```

```
## Error rate:
## 11.3%
## Accuracy:
## 88.7%
## Precision:
## 89.8%
## Recall:
## 88.7%
## f-measure:
## 89.2%
##
##
## Sensitivity:
## 0.898
## Specificity:
## 0.874
```

```
print('Confusion Matrix')
```

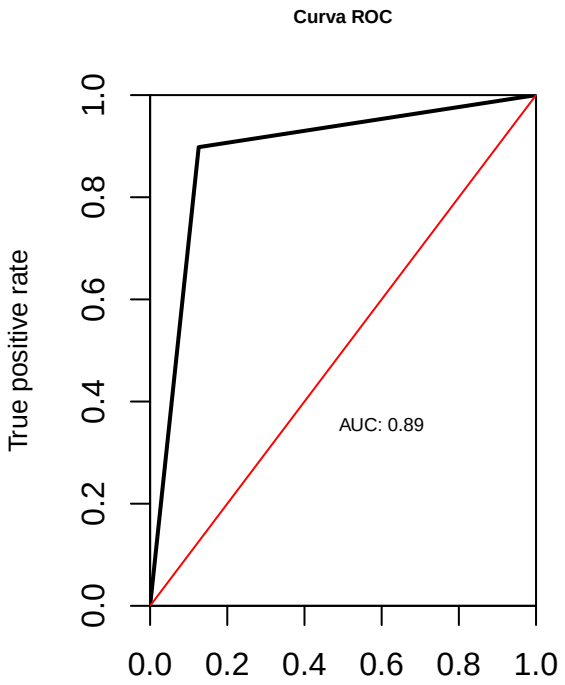
```
## [1] "Confusion Matrix"
```

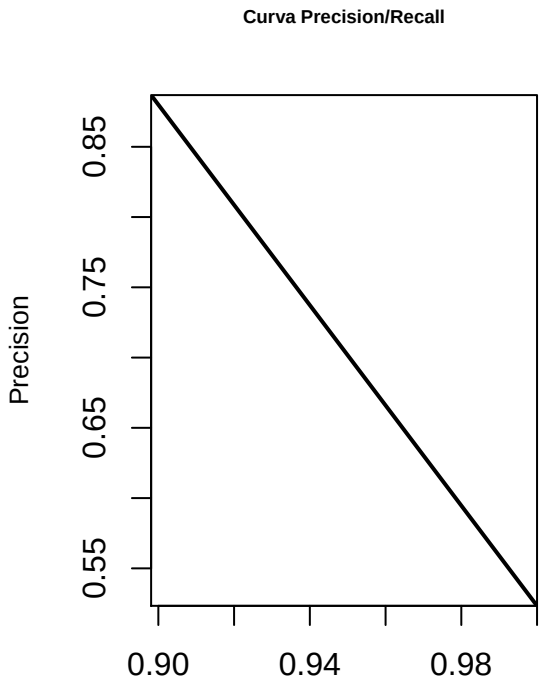
```
print(confMat$table)
```

```
##            Reference
## Prediction   0   1
##          0 125  18
##          1  16 141
```

Finally, two plots were created to present the ROC and PR lines for the selected model.

```
# Creating a predictions object to plot the ROC and the PR graphics
predictions <- prediction(as.integer(as.character(test_data[,1])), as.integer(as.character(pred)))

# Preparing the plotting area to receive two plots
par(mfrow = c(1,2))

# Plotting the ROC curve using the functions from plot_utils.R
plot.roc.curve(predictions, title.text = "Curva ROC")

# Plotting the PR curve using the functions from plot_utils.R
plot.pr.curve(predictions, title.text = "Curva Precision/Recall")
```





As shown in the summary, the selected model presented a good performance. This model presented a sensitivity of 89.8% and a specificity of 87.4%. This means that, on average, this model would wrongly classify a client as "good payer" about 10.2% of the times and it would also wrongly classify a "bad payer" about 12.6% of the times. In the first case the institution would loose money as it would grant credit to a "bad payer" and in the second, the institution would loose a "good payer" client by refusing the credit grant.

Both problems are not good to the Institution, but for this specific case, this model is doing a good job because it is better to have a model that make higher rates of wrong predictions to avoid granting credits to "good payer" than to have a model with higher rates of wrong predictions to grant credit to "bad payer" clients.

THE END
