Risco de Credito

Carlos E. Carvalho

3/13/2021

Este documento foi criado como exercício no curso Formação Cientista de Dados da DataScience Academy (https://www.datascienceacademy.com.br/bundles?bundle_id=formacao-cientista-de-dados) O objetivo é analisar dados históricos e encontrar um modelo para prever o risco de dar crédito em uma instituição bancária.

Primeiro, carregar os pacotes necessários.

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
## corrplot 0.84 loaded
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.4     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v purrr   0.3.4     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x psych::%+%()            masks ggplot2::%+%()
## x psych::alpha()          masks ggplot2::alpha()
## x dplyr::filter()         masks stats::filter()
## x rvest::guess_encoding() masks readr::guess_encoding()
## x dplyr::lag()            masks stats::lag()
## x purrr::pluck()          masks rvest::pluck()
```

```
##
## Attaching package: 'lattice'

## The following object is masked from 'package:corrgram':
##
##     panel.fill

## Loading required package: grid

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

##
## Attaching package: 'DMwR'

## The following object is masked from 'package:psych':
##
##     crossValidation

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:psych':
##
##     outlier

## The following object is masked from 'package:ggplot2':
##
##     margin

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:ROCR':
##
##     prediction

## The following object is masked from 'package:dplyr':
##
##     compute

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:psych':
##
##     alpha

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

Agora, carregar o dataset

```
dataframe1 <- read.csv("credit_dataset.csv")
```

Visualizando o dataframe completo. Aparentemente a primeira coluna (credit.rating) é a coluna que indica se o crédito foi concedido ou não:

```
head(dataframe1)
```

```
##   credit.rating account.balance credit.duration.months
## 1             1               1                     18
## 2             1               1                      9
## 3             1               2                     12
## 4             1               1                     12
## 5             1               1                     12
## 6             1               1                     10
##   previous.credit.payment.status credit.purpose credit.amount savings
## 1                              3              2          1049       1
## 2                              3              4          2799       1
## 3                              2              4           841       2
## 4                              3              4          2122       1
## 5                              3              4          2171       1
## 6                              3              4          2241       1
##   employment.duration installment.rate marital.status guarantor
## 1                   1                4              1         1
## 2                   2                2              3         1
## 3                   3                2              1         1
## 4                   2                3              3         1
```

```
## 5                   2               4             3         1
## 6                   1               1             3         1
##   residence.duration current.assets age other.credits apartment.type
## 1                  4              2  21             2              1
## 2                  2              1  36             2              1
## 3                  4              1  23             2              1
## 4                  2              1  39             2              1
## 5                  4              2  38             1              2
## 6                  3              1  48             2              1
##   bank.credits occupation dependents telephone foreign.worker
## 1            1          3          1         1              1
## 2            2          3          2         1              1
## 3            1          2          1         1              1
## 4            2          2          2         1              2
## 5            2          2          1         1              2
## 6            2          2          2         1              2
```

Dimensões do dataframe

```
dim(dataframe1)
```

```
## [1] 1000   21
```

Verificando o tipo das colunas.

```
glimpse(dataframe1)
```

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

Aparentemente todas as colunas foram consideradas numéricas.

Criando funções para converter variáveis categóricas para tipo fator.

```
to.factors <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- as.factor(df[[variable]])
  }
  return(df)
}
```

Como as variáveis numéricas estão em escalas diferentes, deve-se fazer uma escala e normalização

```
scale.features <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- scale(df[[variable]], center=T, scale=T)
  }
  return(df)
}
```

Normalizando as variáveis

```
numeric.vars <- c("credit.duration.months", "age", "credit.amount")
dataframe2 <- scale.features(dataframe1, numeric.vars)
```

Variáveis categóricas que serão transformadas para fator

```
categorical.vars <- c('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')
dataframe2 <- to.factors(df = dataframe2, variables = categorical.vars)
head(dataframe2)
```

```
##   credit.rating account.balance credit.duration.months
## 1             1               1             -0.2407368
## 2             1               1             -0.9870788
## 3             1               2             -0.7382981
## 4             1               1             -0.7382981
## 5             1               1             -0.7382981
## 6             1               1             -0.9041519
##   previous.credit.payment.status credit.purpose credit.amount savings
## 1                              3              2    -0.7872630       1
## 2                              3              4    -0.1673006       1
## 3                              2              4    -0.8609500       2
## 4                              3              4    -0.4071375       1
## 5                              3              4    -0.3897785       1
## 6                              3              4    -0.3649800       1
##   employment.duration installment.rate marital.status guarantor
## 1                   1                4              1         1
## 2                   2                2              3         1
## 3                   3                2              1         1
## 4                   2                3              3         1
## 5                   2                4              3         1
## 6                   1                1              3         1
```

```
##   residence.duration current.assets         age other.credits apartment.type
## 1                  4              2 -1.28093214             2              1
## 2                  2              1  0.04034293             2              1
## 3                  4              1 -1.10476213             2              1
## 4                  2              1  0.30459795             2              1
## 5                  4              2  0.21651294             1              2
## 6                  3              1  1.09736299             2              1
##   bank.credits occupation dependents telephone foreign.worker
## 1            1          3          1         1              1
## 2            2          3          2         1              1
## 3            1          2          1         1              1
## 4            2          2          2         1              2
## 5            2          2          1         1              2
## 6            2          2          2         1              2
```

Criando um dataframe apenas com as colunas numéricas do dataset original:

```
dataframe3 <- dataframe1[,c(3,6,14)]
head(dataframe3)
```

```
##   credit.duration.months credit.amount age
## 1                     18          1049  21
## 2                      9          2799  36
## 3                     12           841  23
## 4                     12          2122  39
## 5                     12          2171  38
## 6                     10          2241  48
```

Observando algumas medidas de tendência central das variáveis numéricas:

```
summary(dataframe3)
```

```
##  credit.duration.months credit.amount        age
##  Min.   : 4.0           Min.   :  250   Min.   :19.00
##  1st Qu.:12.0           1st Qu.: 1366   1st Qu.:27.00
##  Median :18.0           Median : 2320   Median :33.00
##  Mean   :20.9           Mean   : 3271   Mean   :35.54
##  3rd Qu.:24.0           3rd Qu.: 3972   3rd Qu.:42.00
##  Max.   :72.0           Max.   :18424   Max.   :75.00
```

Observando a correlação entre as colunas numéricas:

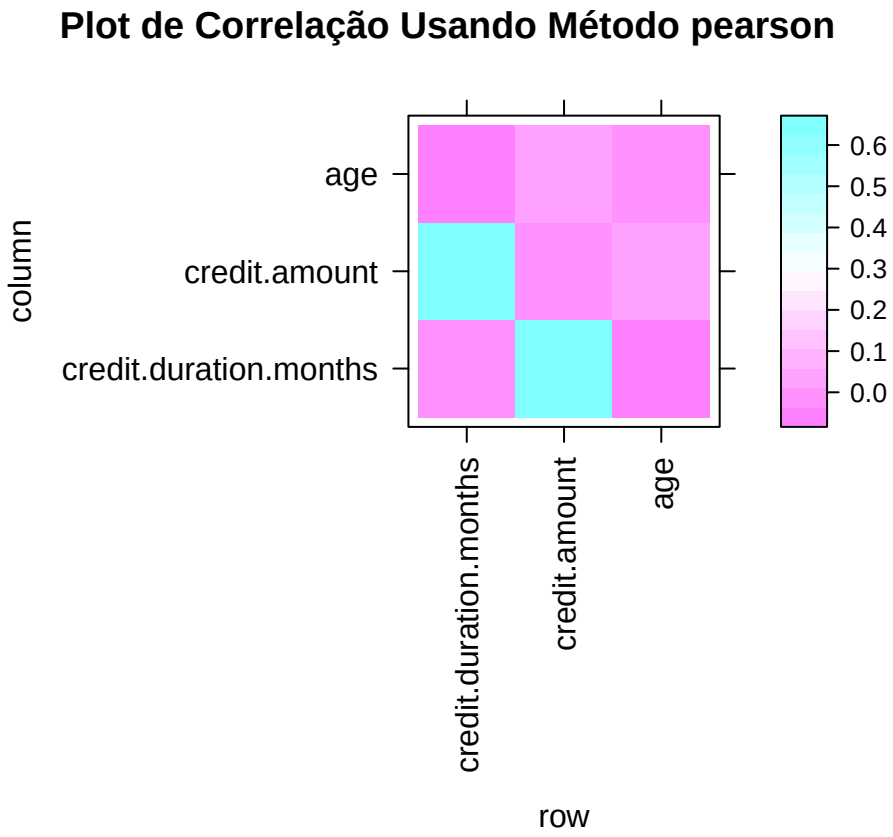
```
cols <- c("credit.duration.months", "credit.amount", "age")
```

Vetor com os métodos de correlação:

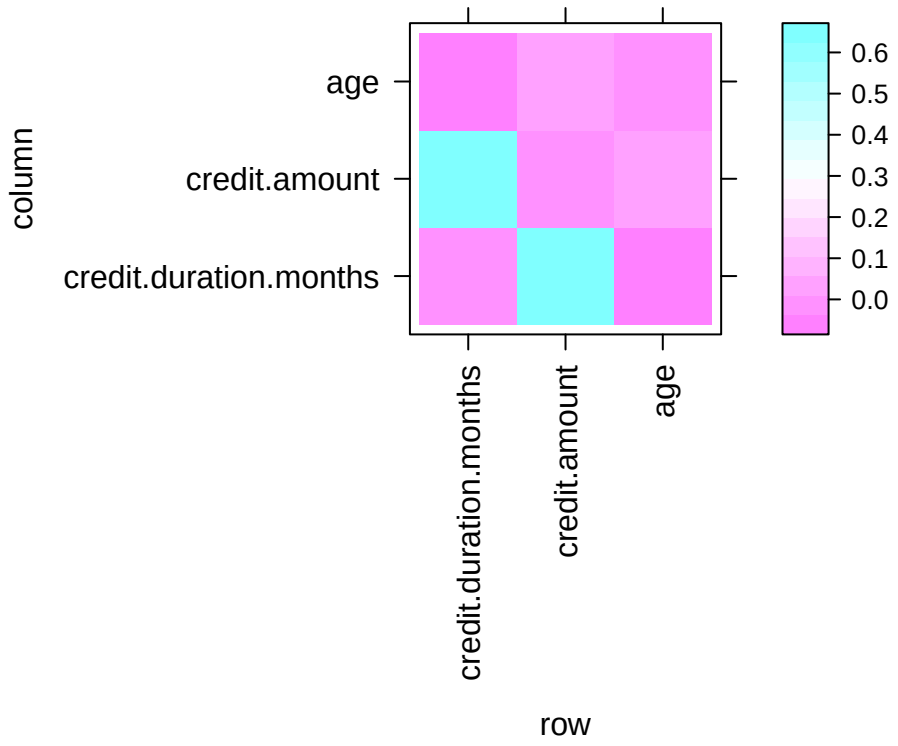
```
metodos <- c("pearson", "spearman")

cors <- lapply(metodos, function(method)
  (cor(dataframe2[,cols], method = method)))
```

```r
plot.cors <- function(x, labs){
  diag(x) <- 0.0
  plot(levelplot(x,
                 main = paste("Plot de Correlação Usando Método", labs),
                 scales = list(x = list(rot = 90), cex = 1.0)))
}
```



Mapa de correlação:




```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

Aparentemente existe uma forte correlação entre as variáveis credit.amount e credit.duration.months. Verificando a quantidade de crédito bom e ruim

```
table(dataframe2$credit.rating)
```

```
##
##   0   1
## 300 700
```

É possível verificar que existem muito mais casos com crédito bom (1) do que com crédito ruim (0). Então é necessário balancear para que haja quantidades parecidas e o modelo não fique tendencioso.

Passando a coluna alvo para a última posição do dataframe:

```
dataframe4 <- dataframe2$credit.rating
dataframe2$credit.rating <- NULL
dataframe2 <- cbind(dataframe2, dataframe4)
dataframe2 <- dataframe2 %>%
  rename(
```

```
    credit.rating = dataframe4
  )
```

Fazendo o balanceamento:

```
dataframe2 <- SMOTE(credit.rating ~ ., data = dataframe2, perc.over = 100)
```

Agora vamos ver como ficou a proporção:

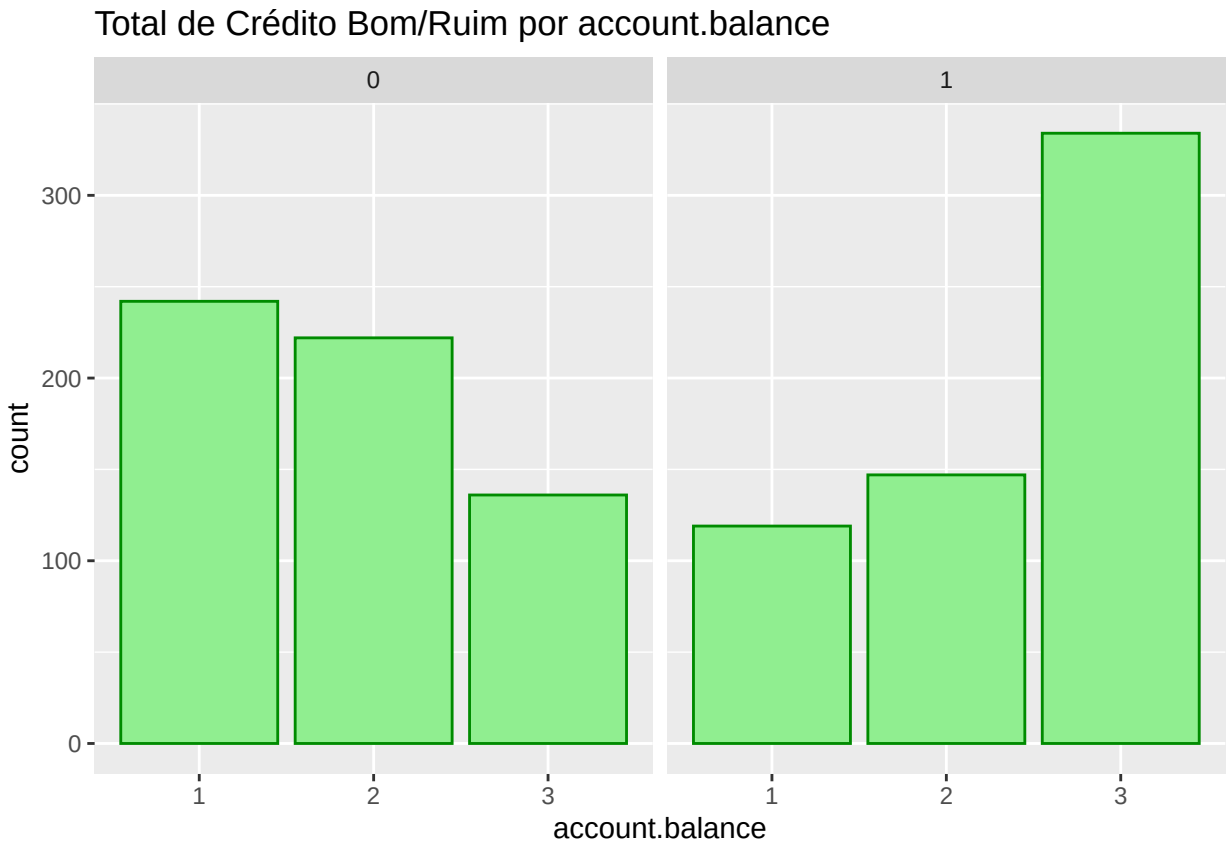
```
table(dataframe2$credit.rating)
```

```
##
##   0   1
## 600 600
```

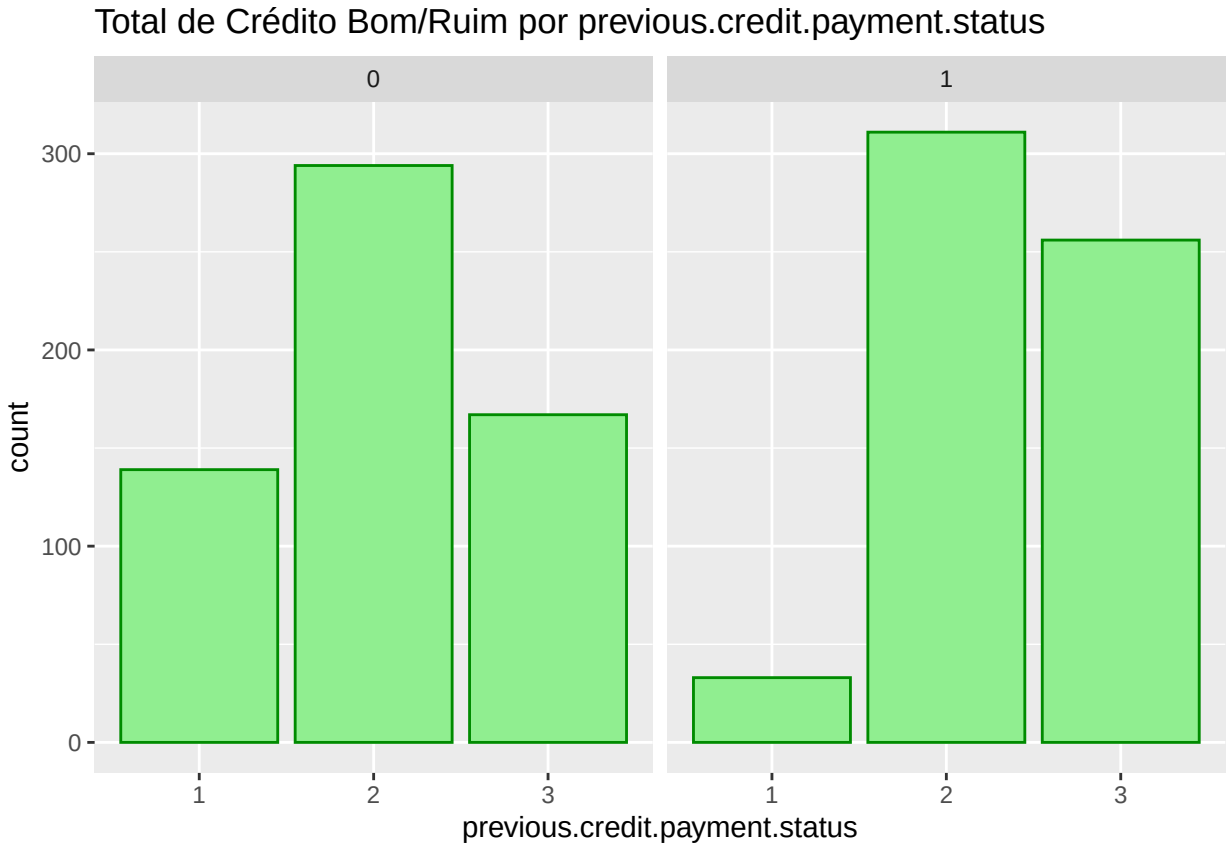
Agora o dataset tem a mesma quantidade de observações com crédito bom e ruim.

Criando gráfico de barras para observar as variáveis:

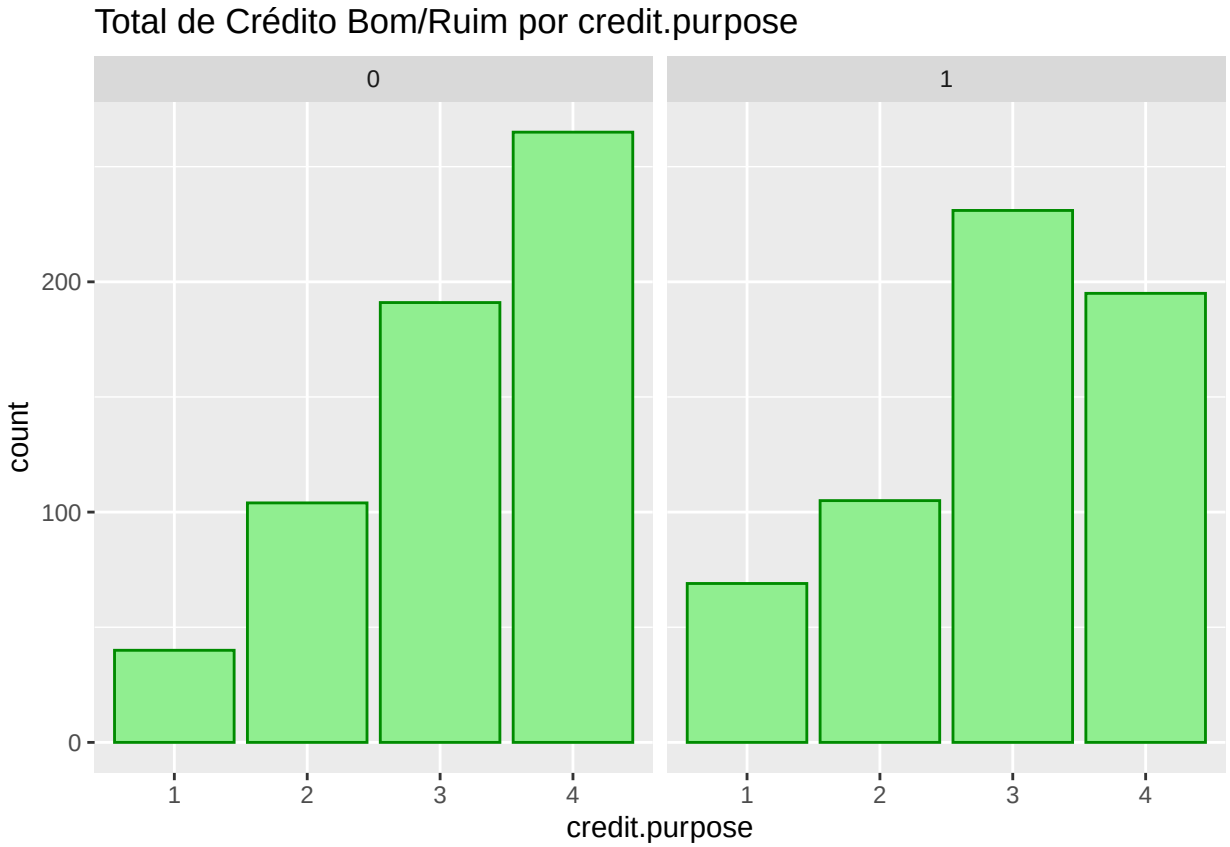
```
## [[1]]
```



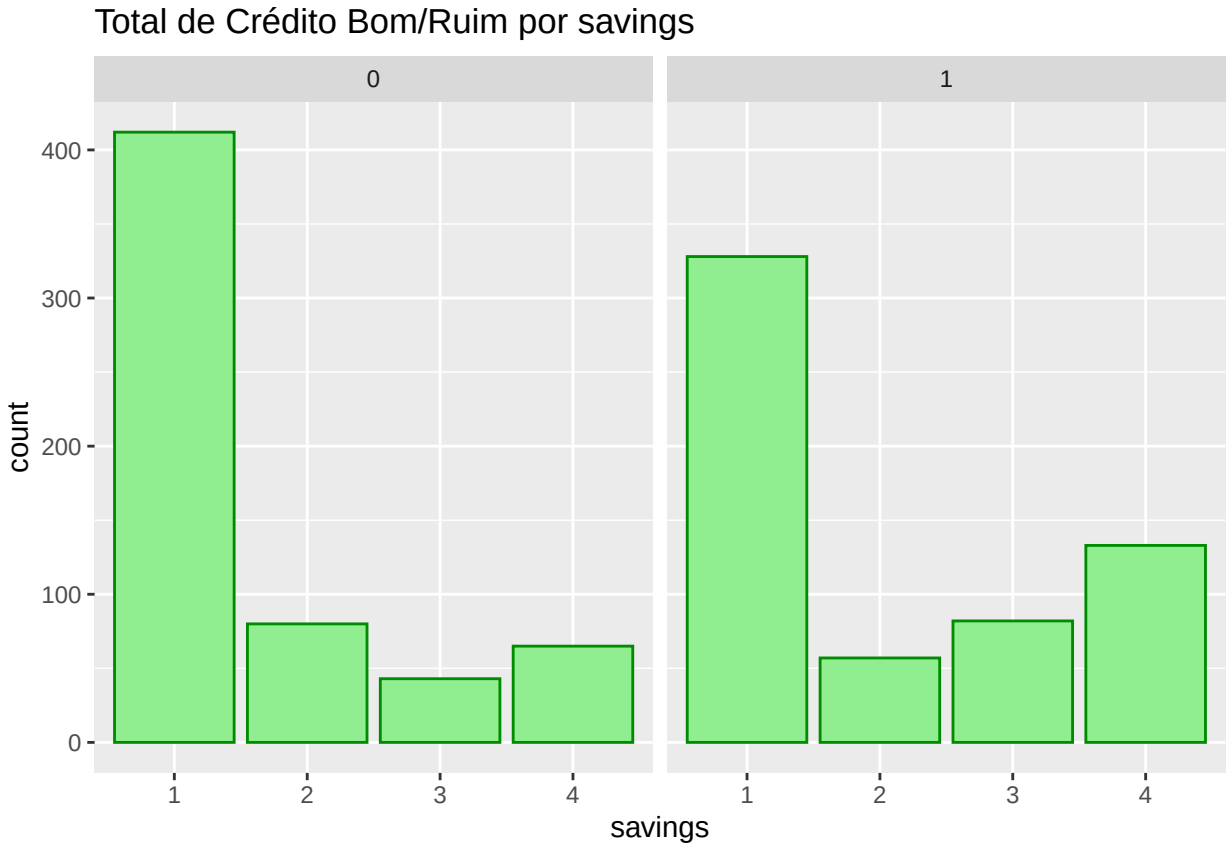

```
##
## [[2]]
## NULL
##
## [[3]]
```


Total de Crédito Bom/Ruim por previous.credit.payment.status
0
1
300
200
100
0
count
1
2
3
1
2
3
previous.credit.payment.status

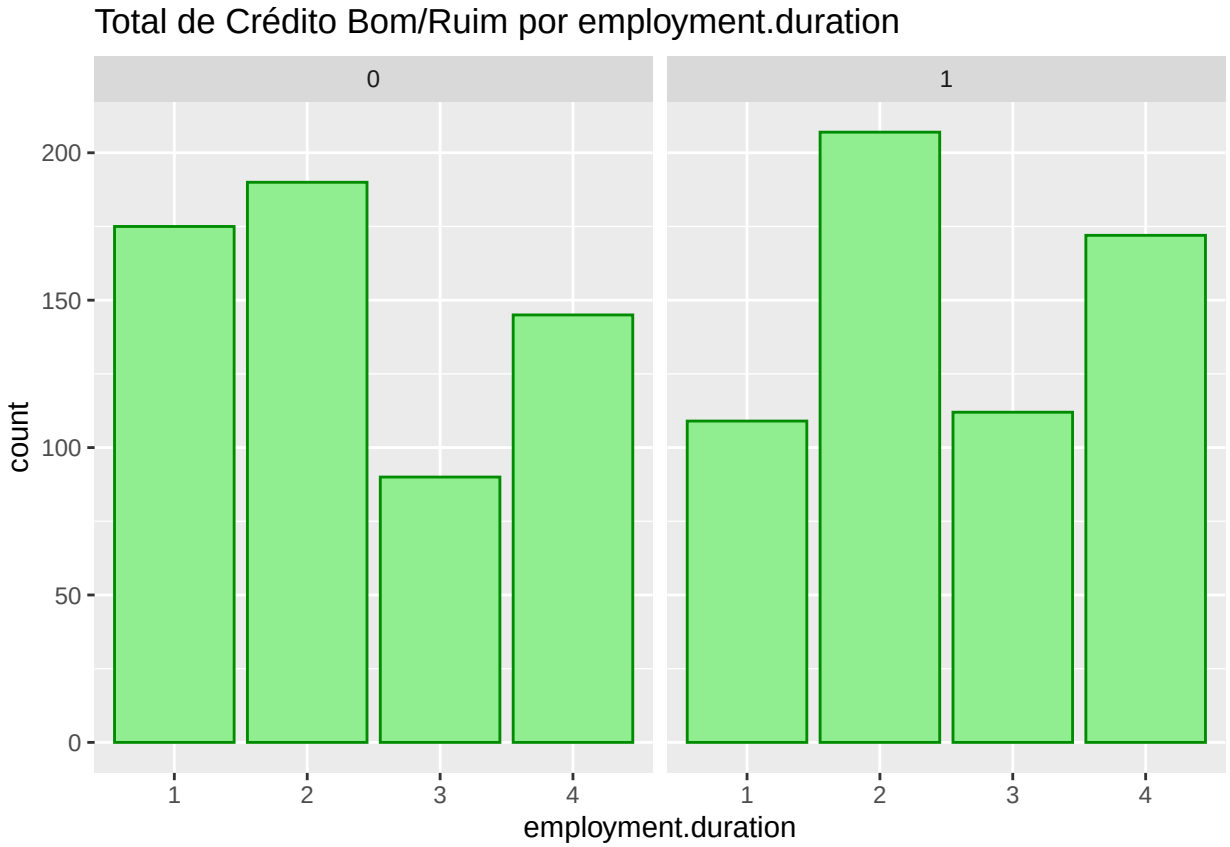
```
##
## [[4]]
```


Total de Crédito Bom/Ruim por credit.purpose
0
1
200
100
0
count
1
2
3
4
1
2
3
4
credit.purpose

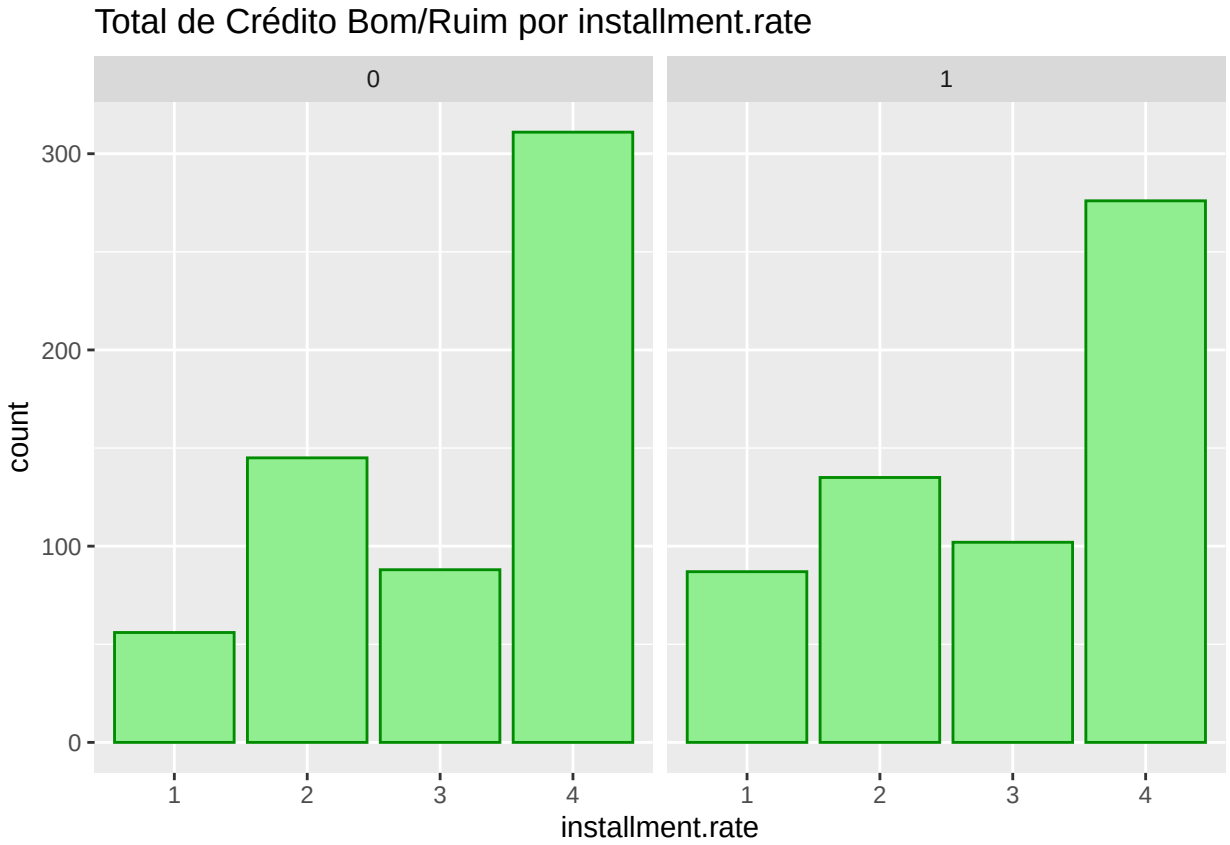
```
##
## [[5]]
## NULL
##
## [[6]]
```


Total de Crédito Bom/Ruim por savings
0
1
400
300
200
100
0
count
1
2
3
4
1
2
3
4
savings

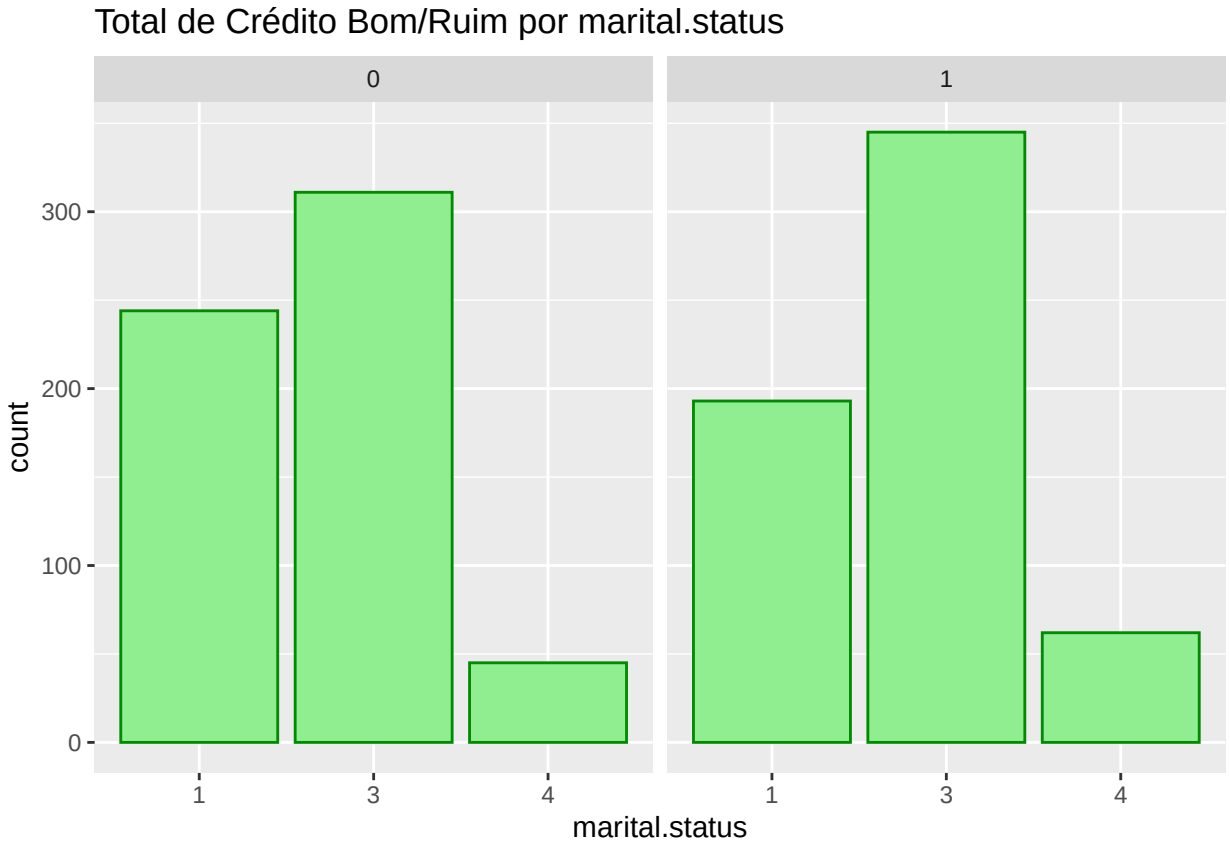
```
##
## [[7]]
```


Total de Crédito Bom/Ruim por employment.duration
0
1
200
150
100
50
0
count
1
2
3
4
employment.duration

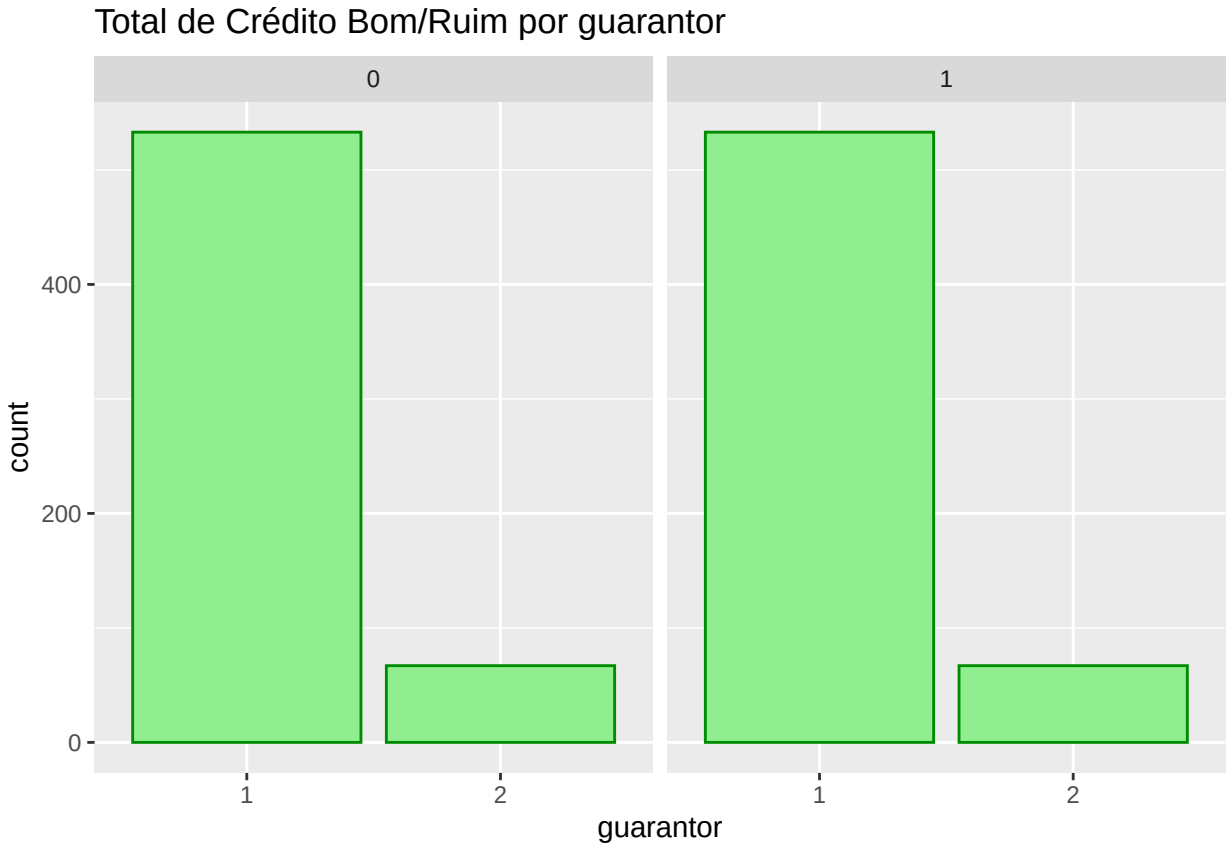
```
##
## [[8]]
```


Total de Crédito Bom/Ruim por installment.rate
0
1
300
200
100
0
count
1
2
3
4
1
2
3
4
installment.rate

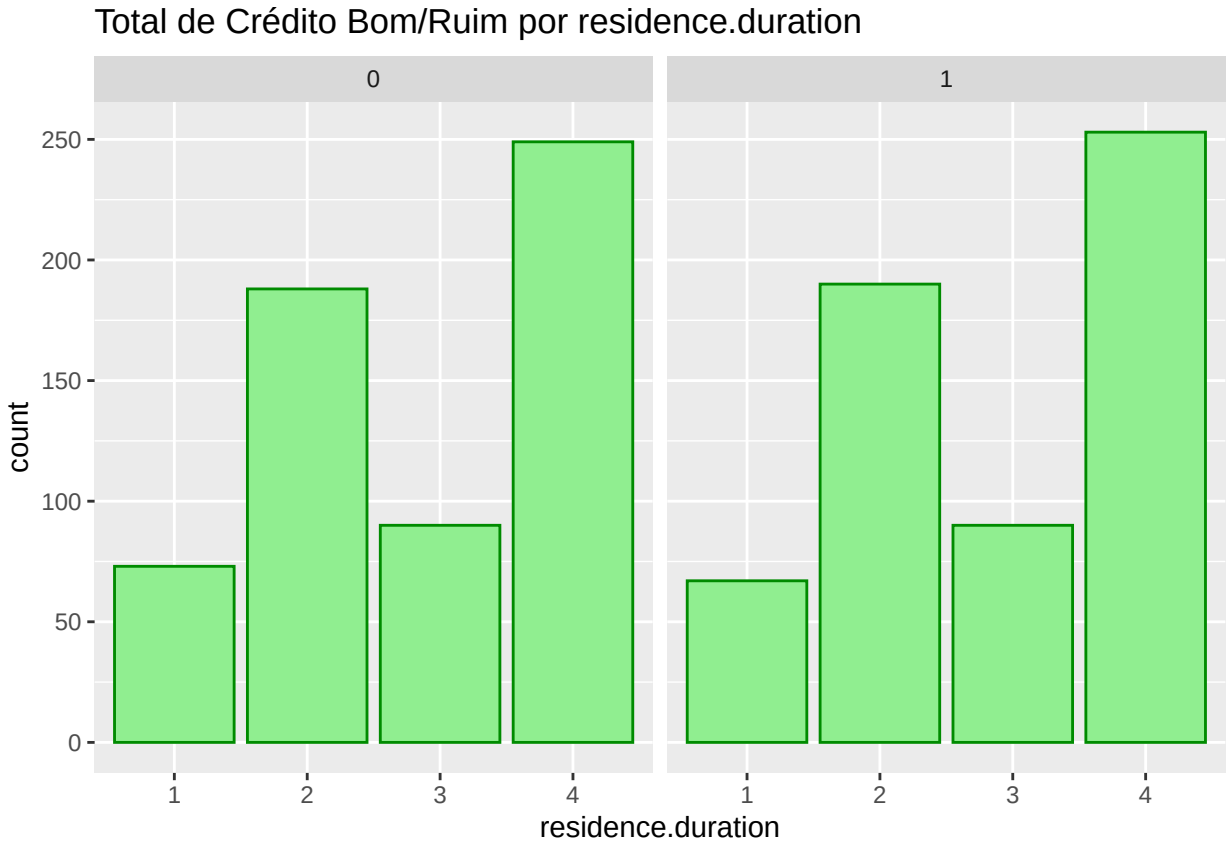
```
##
## [[9]]
```


Total de Crédito Bom/Ruim por marital.status
0
1
300
200
100
0
count
1
3
4
1
3
4
marital.status

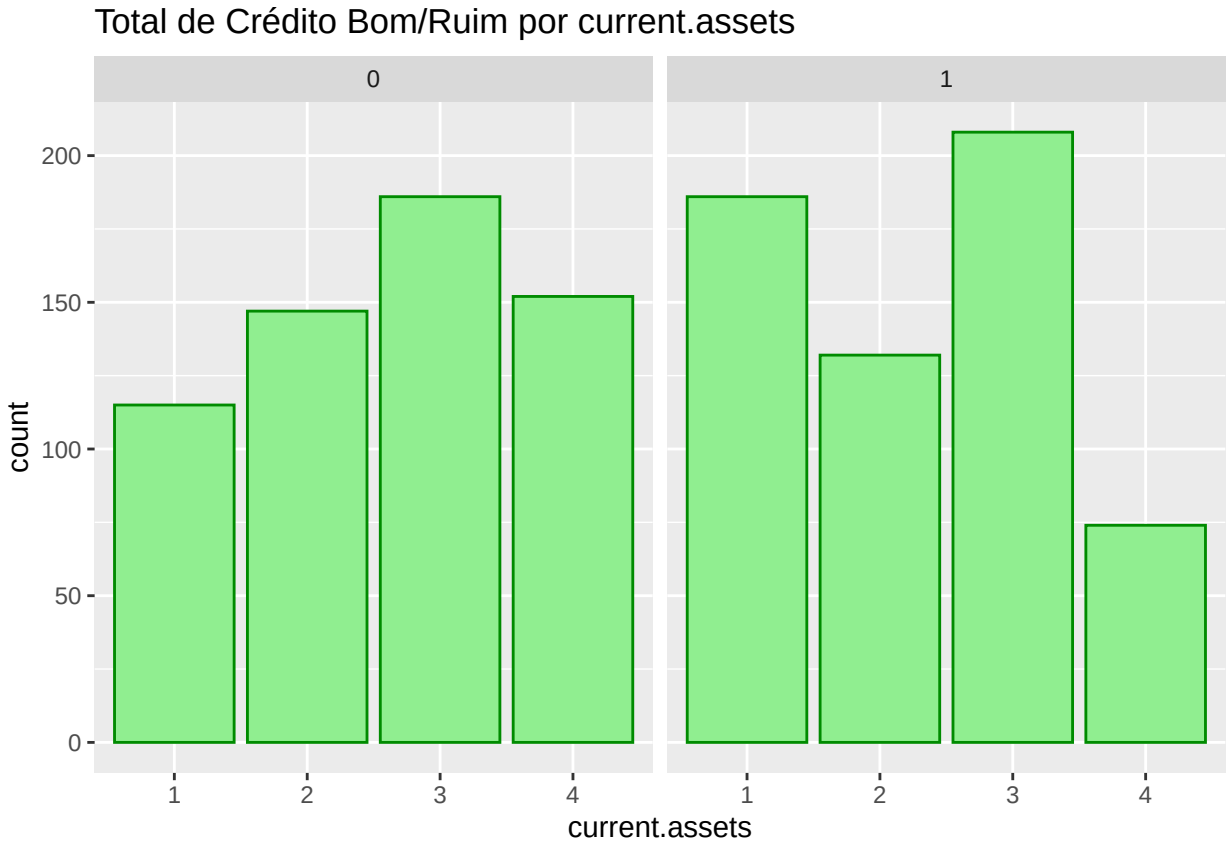
```
##
## [[10]]
```


Total de Crédito Bom/Ruim por guarantor
0
1
400
200
0
count
1
2
1
2
guarantor

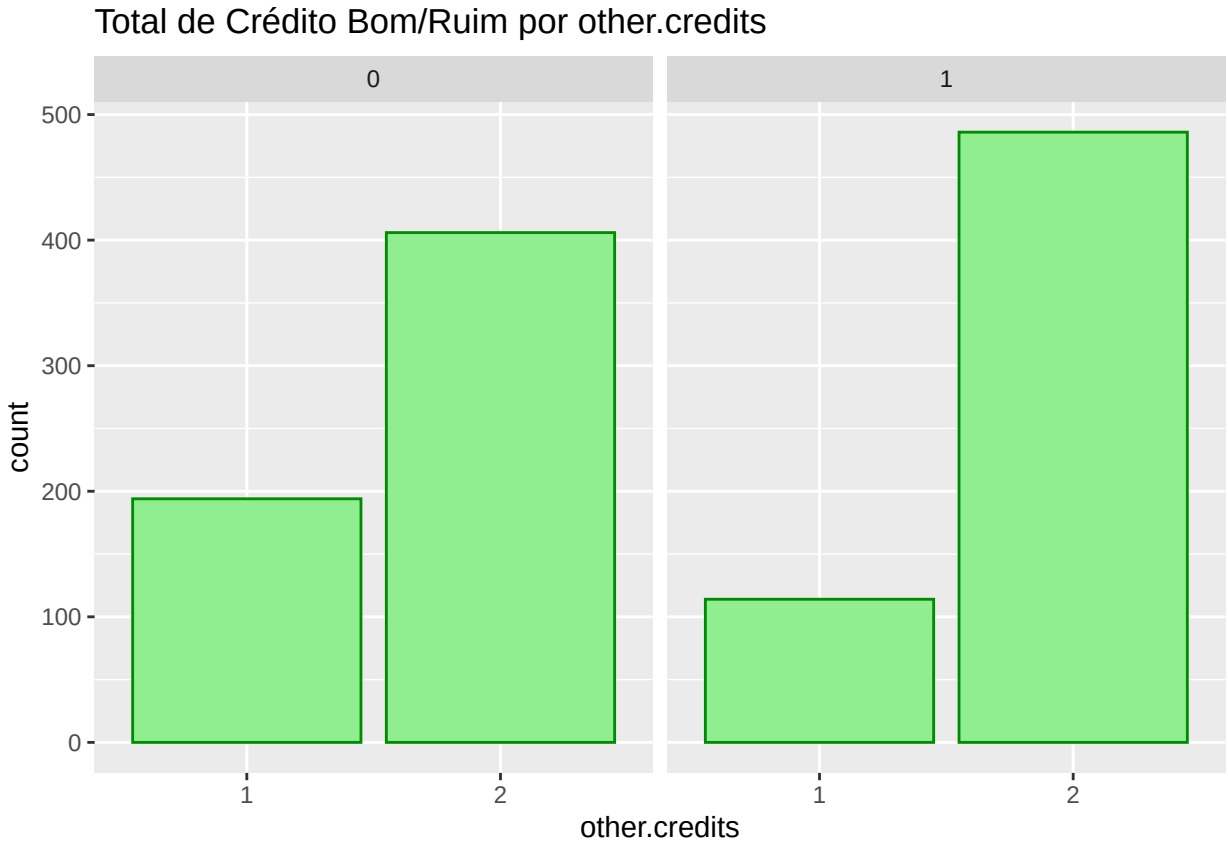
```
##
## [[11]]
```


Total de Crédito Bom/Ruim por residence.duration
0
1
250
200
150
100
50
0
count
1
2
3
4
residence.duration

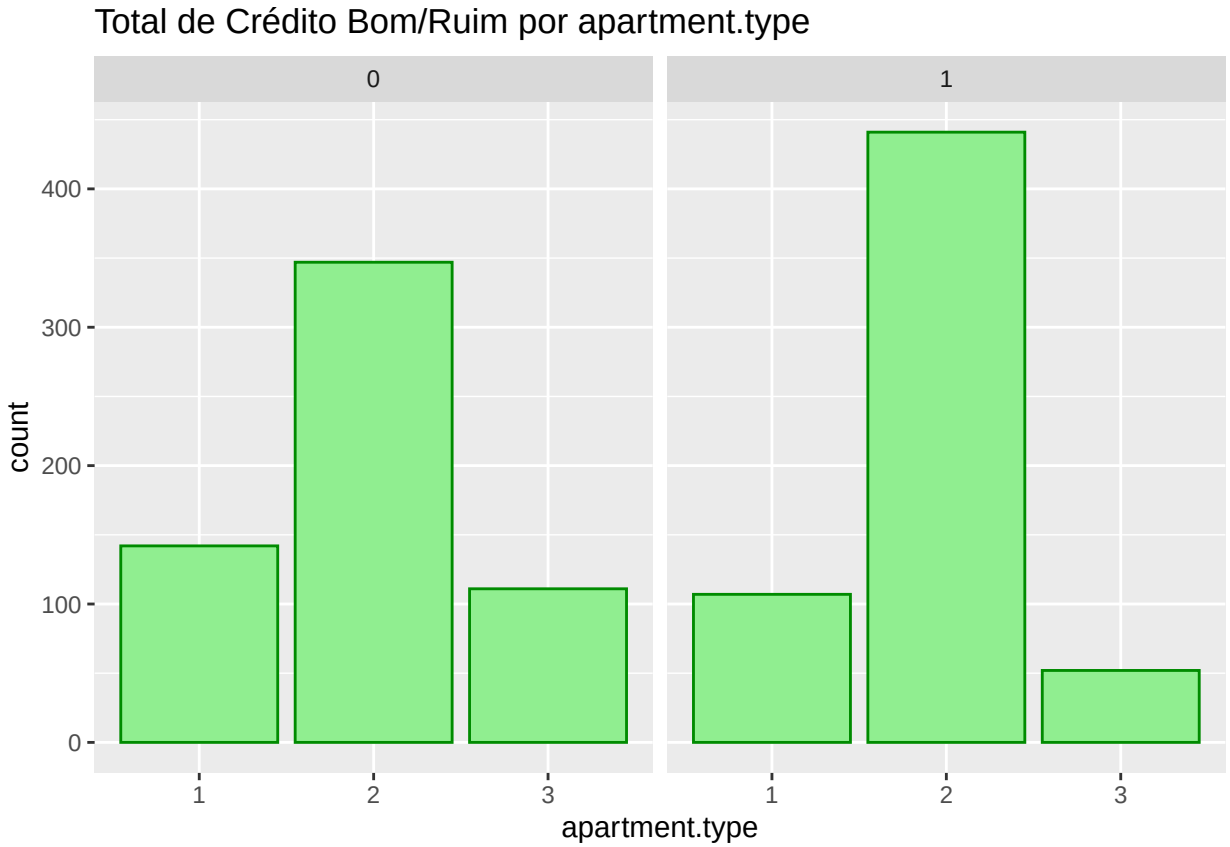
```
##
## [[12]]
```


Total de Crédito Bom/Ruim por current.assets
0
1
200
150
100
50
0
count
1
2
3
4
current.assets

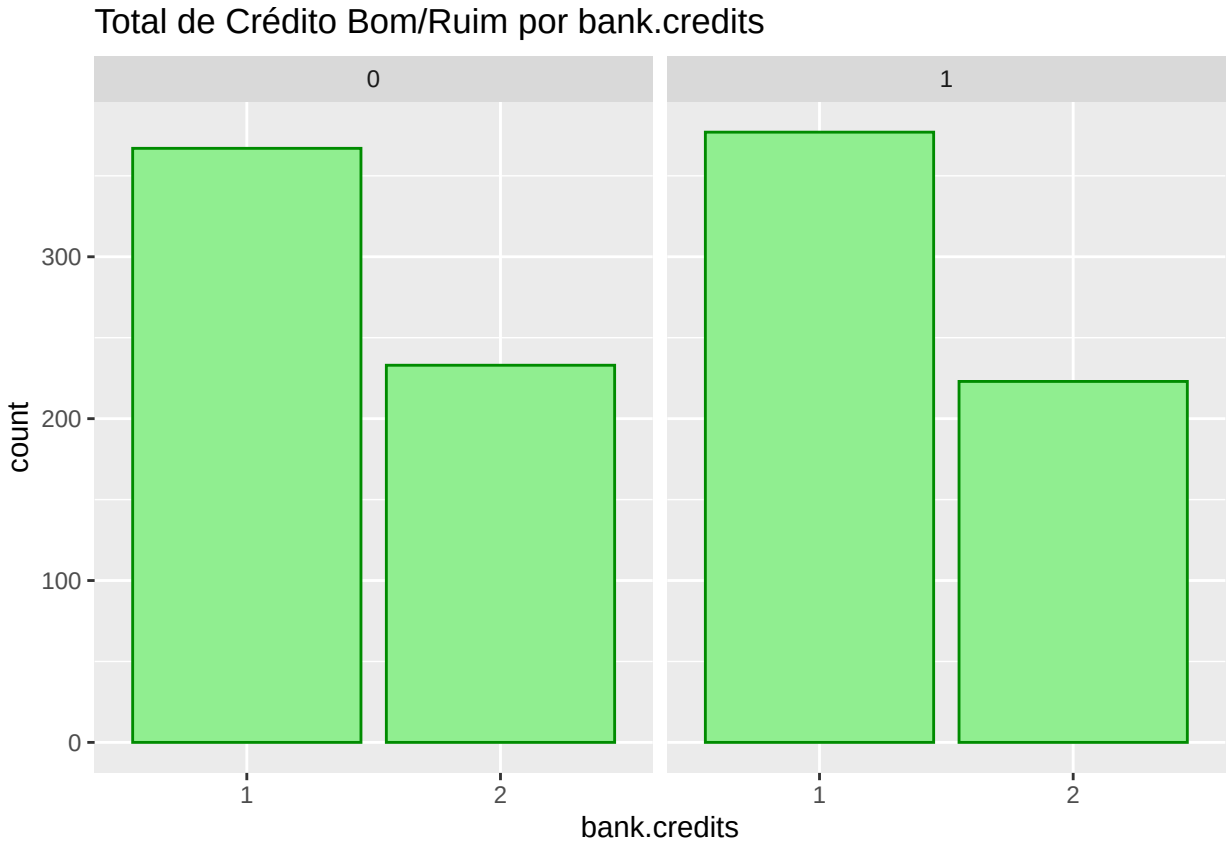
```
##
## [[13]]
## NULL
##
## [[14]]
```


Total de Crédito Bom/Ruim por other.credits
0
1
500
400
300
200
100
0
count
1
2
1
2
other.credits

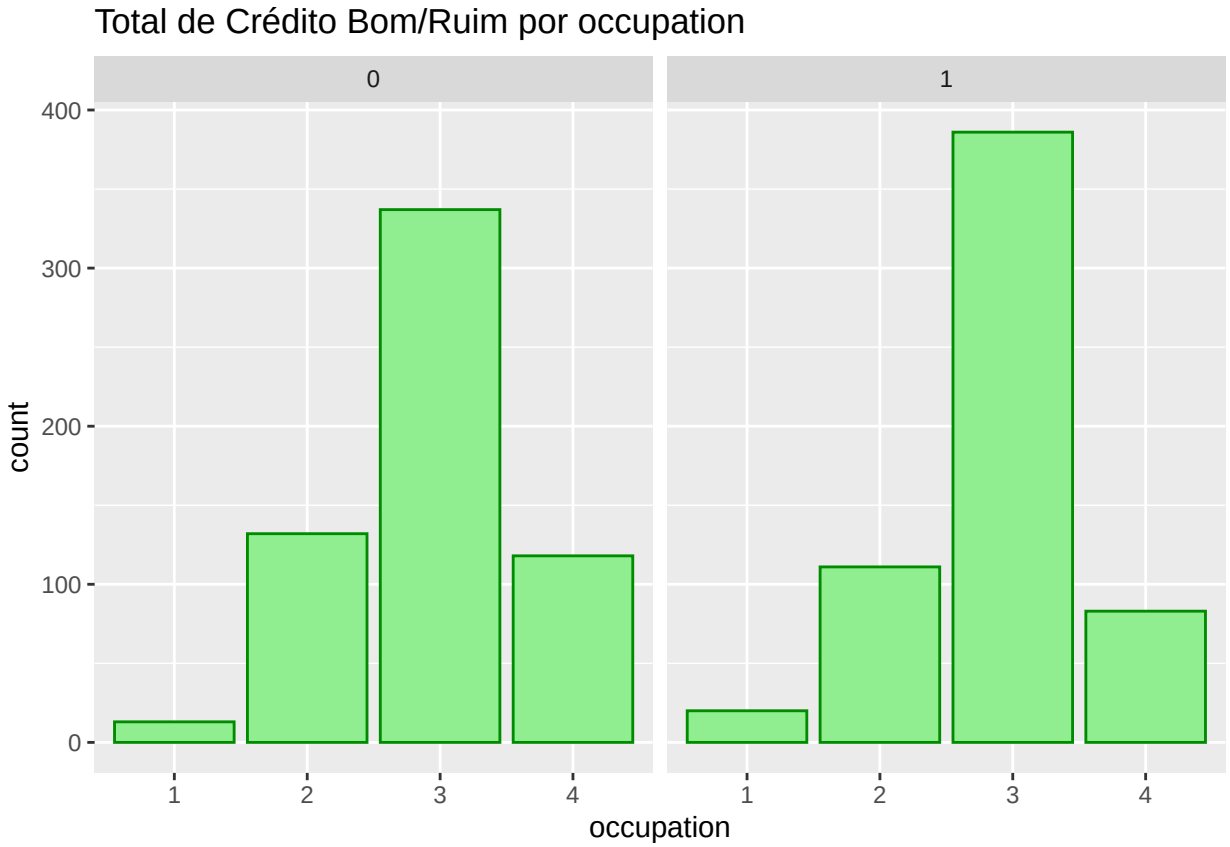
```
##
## [[15]]
```


Total de Crédito Bom/Ruim por apartment.type
0
1
400
300
200
100
0
count
1
2
3
1
2
3
apartment.type

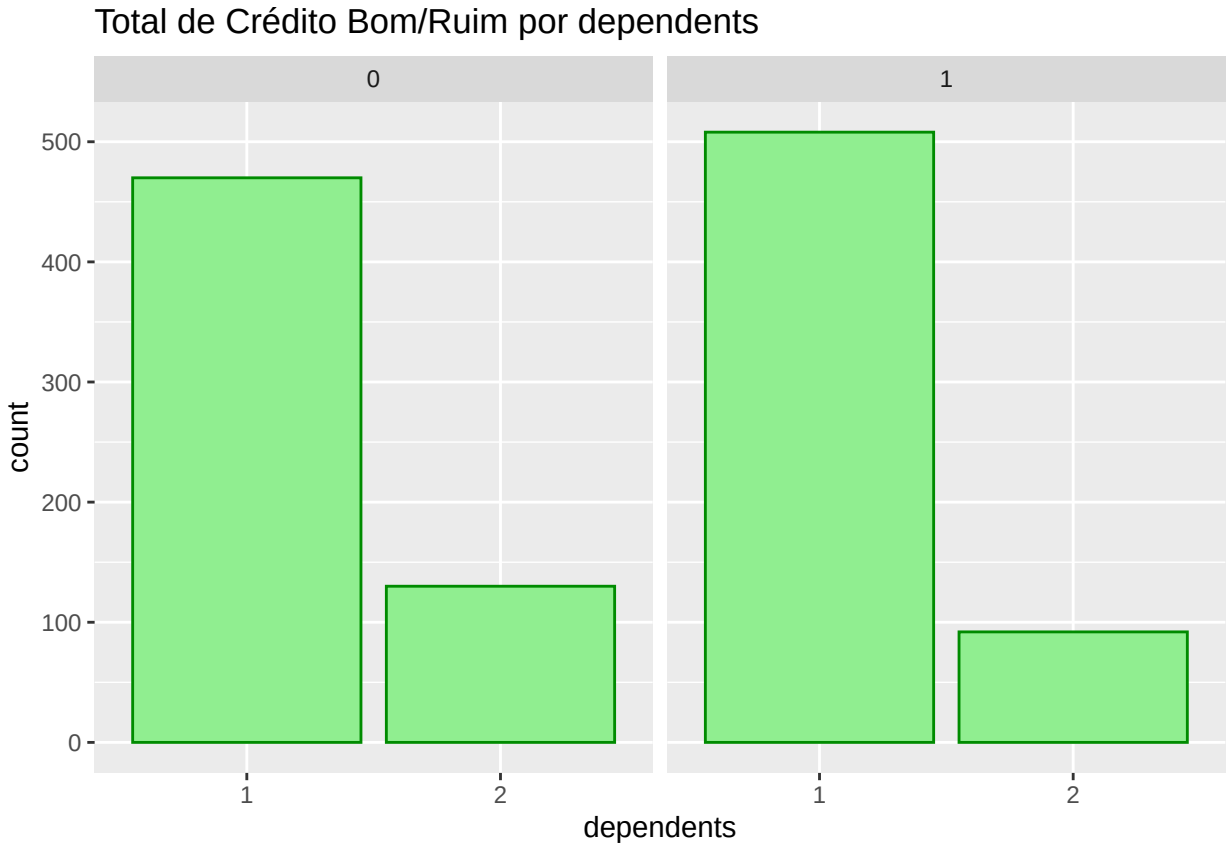
```
##
## [[16]]
```


Total de Crédito Bom/Ruim por bank.credits
0
1
300
200
100
0
count
1
2
1
2
bank.credits

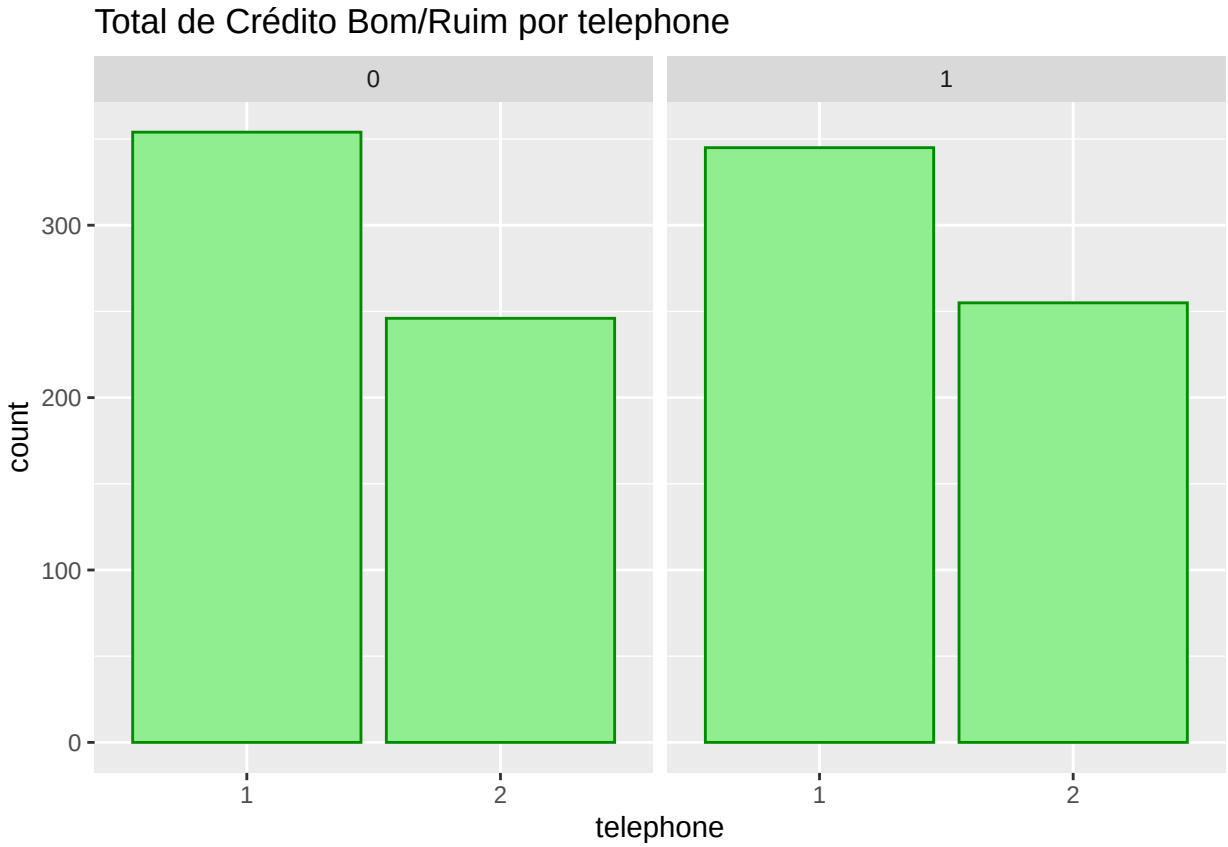
```
##
## [[17]]
```


Total de Crédito Bom/Ruim por occupation
0
1
400
300
200
100
0
count
1
2
3
4
occupation

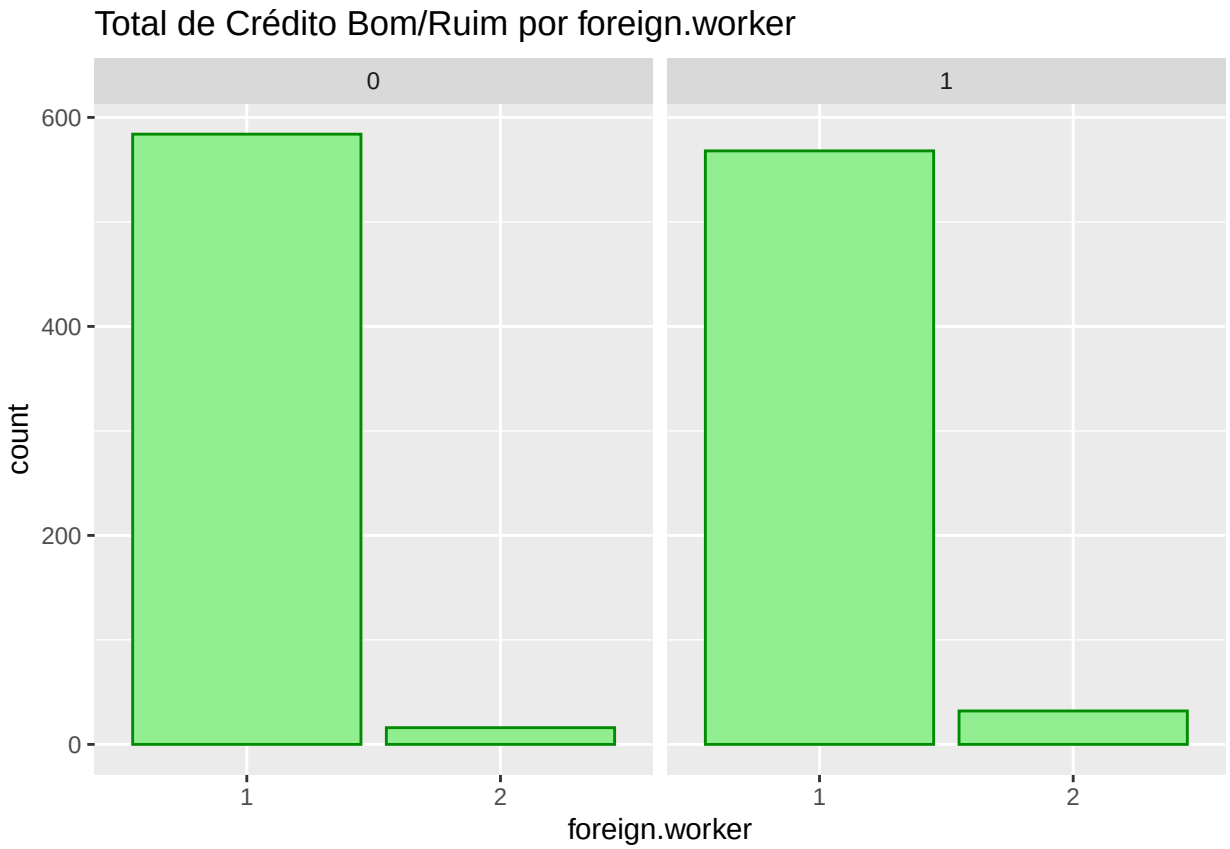
```
##
## [[18]]
```


Total de Crédito Bom/Ruim por dependents
0
1
500
400
300
200
100
0
count
1
2
1
2
dependents

```
##
## [[19]]
```


Total de Crédito Bom/Ruim por telephone
0
1
300
200
100
0
count
1
2
1
2
telephone

```
##
## [[20]]
```



Verificando a importância de cada variável para os modelos. Criando uma função para seleção de variáveis:

```
run.feature.selection <- function(num.iters=20, feature.vars, class.var){
  set.seed(10)
  variable.sizes <- 1:10
  control <- rfeControl(functions = rfFuncs, method = "cv",
                        verbose = FALSE, returnResamp = "all",
                        number = num.iters)
  results.rfe <- rfe(x = feature.vars, y = class.var,
                     sizes = variable.sizes,
                     rfeControl = control)
  return(results.rfe)
}
```

Executando a função para poder escolher as variáveis mais importantes para o modelo:

```
rfe.results <- run.feature.selection(feature.vars = dataframe2[,-21],
                                     class.var = dataframe2[,21])
```

Visualizando os resultados

```
rfe.results
```

```
##
## Recursive feature selection
##
```

```
## Outer resampling method: Cross-Validated (20 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.6650 0.3300    0.07028 0.14055
##          2   0.6675 0.3350    0.06409 0.12818
##          3   0.7225 0.4450    0.05442 0.10883
##          4   0.7667 0.5333    0.04426 0.08852
##          5   0.7992 0.5983    0.05115 0.10230
##          6   0.8192 0.6383    0.04402 0.08804
##          7   0.8333 0.6667    0.05187 0.10373
##          8   0.8283 0.6567    0.04959 0.09918
##          9   0.8325 0.6650    0.05256 0.10512
##         10   0.8342 0.6683    0.05115 0.10230
##         20   0.8425 0.6850    0.04475 0.08949        *
##
## The top 5 variables (out of 20):
##    account.balance, credit.duration.months, previous.credit.payment.status, credit.amount, savings
```

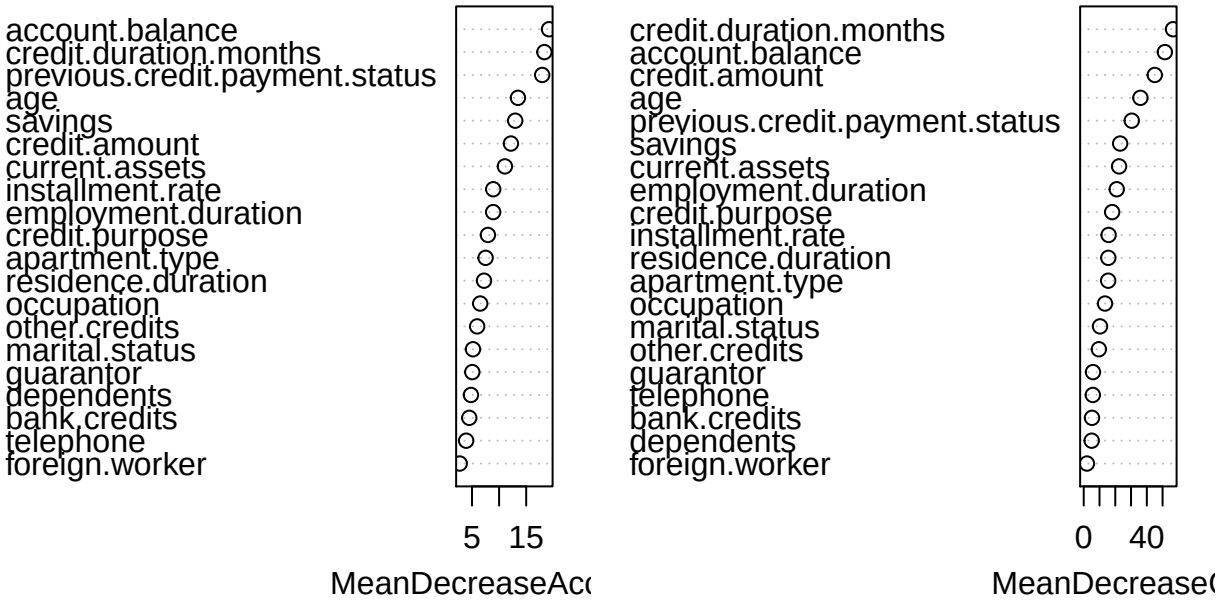
```
varImp((rfe.results))
```

```
##                                 Overall
## account.balance                40.267095
## credit.duration.months         33.209097
## previous.credit.payment.status 26.309938
## credit.amount                  24.790839
## savings                        23.728994
## age                            23.454627
## employment.duration            20.338702
## current.assets                 18.672133
## credit.purpose                 17.597194
## installment.rate               16.819531
## residence.duration             15.163763
## occupation                     13.750345
## apartment.type                 13.611361
## marital.status                 12.814906
## other.credits                  12.028133
## telephone                      10.551408
## guarantor                      10.193481
## dependents                      9.313668
## bank.credits                    9.286134
## foreign.worker                  5.531636
```

Utilização do modelo random forest para criação de um plot de importância das variáveis preditoras

```
modelo <- randomForest(credit.rating ~ .,
                       data = dataframe2,
                       ntree = 100, nodesize = 10, importance = T)

varImpPlot(modelo)
```



A princípio será montado um modelo com todas as variáveis e outro apenas com as 6 variáveis mais importantes indicadas pelo random forest. Isso será feito para comparação entre os modelos e como forma de validação para a retirada de algumas variáveis.

Para iniciar a construção do modelo é necessário dividir os dados em treino e teste, de forma aleatória. Essa divisão será 70% do dataset para dados de treino e 30% para dados de teste.

```
amostra <- sample.split(dataframe2$credit.rating, SplitRatio = 0.70)

# Criando dados de treino - 70% dos dados
treino = subset(dataframe2, amostra == TRUE)

# Criando dados de teste - 30% dos dados
teste = subset(dataframe2, amostra == FALSE)
```

Os modelos estudados serão: - Regressão Logística; - Random Forest; - Support Vector Machine; - Naive Bayes

Construindo um modelo de regressão logística com todas as variáveis:

```
formula.init <- "credit.rating ~ ."
formula.init <- as.formula(formula.init)
modelo_RL_1 <- glm(formula = formula.init, data = treino, family = "binomial")
```

Visualizando o modelo:

```r
summary(modelo_RL_1)
```

```
##
## Call:
## glm(formula = formula.init, family = "binomial", data = treino)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.27847  -0.85532   0.07559   0.84219   2.56427
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     0.031582   0.797138   0.040 0.968397
## account.balance2                0.390547   0.219089   1.783 0.074653 .
## account.balance3                1.554545   0.219989   7.066 1.59e-12 ***
## credit.duration.months         -0.426891   0.124205  -3.437 0.000588 ***
## previous.credit.payment.status2 1.054649   0.286334   3.683 0.000230 ***
## previous.credit.payment.status3 1.361920   0.300223   4.536 5.72e-06 ***
## credit.purpose2                -1.076818   0.387015  -2.782 0.005396 **
## credit.purpose3                -1.098674   0.362181  -3.033 0.002417 **
## credit.purpose4                -1.454038   0.352407  -4.126 3.69e-05 ***
## credit.amount                  -0.270174   0.135526  -1.994 0.046205 *
## savings2                        0.115384   0.277771   0.415 0.677854
## savings3                        0.519036   0.302890   1.714 0.086600 .
## savings4                        0.820537   0.248561   3.301 0.000963 ***
## employment.duration2            0.512413   0.243096   2.108 0.035043 *
## employment.duration3            0.894564   0.296001   3.022 0.002510 **
## employment.duration4            0.475976   0.271068   1.756 0.079100 .
## installment.rate2              -0.530066   0.310142  -1.709 0.087431 .
## installment.rate3              -0.394401   0.343137  -1.149 0.250392
## installment.rate4              -0.566309   0.292782  -1.934 0.053084 .
## marital.status3                 0.271426   0.190212   1.427 0.153590
## marital.status4                 0.428126   0.337477   1.269 0.204581
## guarantor2                      0.483938   0.270624   1.788 0.073739 .
## residence.duration2            -0.493641   0.304385  -1.622 0.104854
## residence.duration3            -0.123171   0.334694  -0.368 0.712865
## residence.duration4             0.007963   0.296743   0.027 0.978590
## current.assets2                -0.537425   0.246927  -2.176 0.029521 *
## current.assets3                -0.126031   0.230688  -0.546 0.584841
## current.assets4                -0.643056   0.328005  -1.961 0.049937 *
## age                             0.086944   0.103514   0.840 0.400951
## other.credits2                  0.266223   0.202172   1.317 0.187901
## apartment.type2                 0.622655   0.219922   2.831 0.004637 **
## apartment.type3                 0.055270   0.378264   0.146 0.883830
## bank.credits2                  -0.124596   0.201039  -0.620 0.535416
## occupation2                    -1.220482   0.574869  -2.123 0.033749 *
## occupation3                    -1.109486   0.555737  -1.996 0.045888 *
## occupation4                    -1.213383   0.581942  -2.085 0.037064 *
## dependents2                    -0.546920   0.233404  -2.343 0.019118 *
## telephone2                      0.106976   0.189962   0.563 0.573336
## foreign.worker2                 0.616293   0.487069   1.265 0.205761
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1164.49  on 839  degrees of freedom
## Residual deviance:  872.31  on 801  degrees of freedom
## AIC: 950.31
##
## Number of Fisher Scoring iterations: 4
```

Testando o modelo nos dados de teste

```
prevendo_RL_1 <- predict(modelo_RL_1, teste, type="response")
prevendo_RL_1 <- round(prevendo_RL_1)

test.feature.vars <- teste[,-21]
test.class.var <- teste[,21]
```

Criando uma confusion matrix para avaliar os resultados dos testes:

```
CF_1 <- confusionMatrix(table(data = prevendo_RL_1, reference = test.class.var), positive = '1')
CF_1$table
```

```
##     reference
## data   0   1
##    0 130  42
##    1  50 138
```

Acurácia do modelo de regressão logística com todas as variáveis do dataset:

```
CF_1$overall["Accuracy"]
```

```
##  Accuracy
## 0.7444444
```

Visualizando os valores previstos e observados

```
resultados_RL_1 <- cbind(prevendo_RL_1, teste$credit.rating)
colnames(resultados_RL_1) <- c('Previsto','Real')
resultados_RL_1 <- as.data.frame(resultados_RL_1)
head(resultados_RL_1)
```

```
##     Previsto Real
## 639        0    2
## 106        1    2
## 309        0    2
## 31         1    2
## 140        1    2
## 74         1    2
```

Feature selection - Observando o gráfico com as variáveis mais importantes para os modelos.

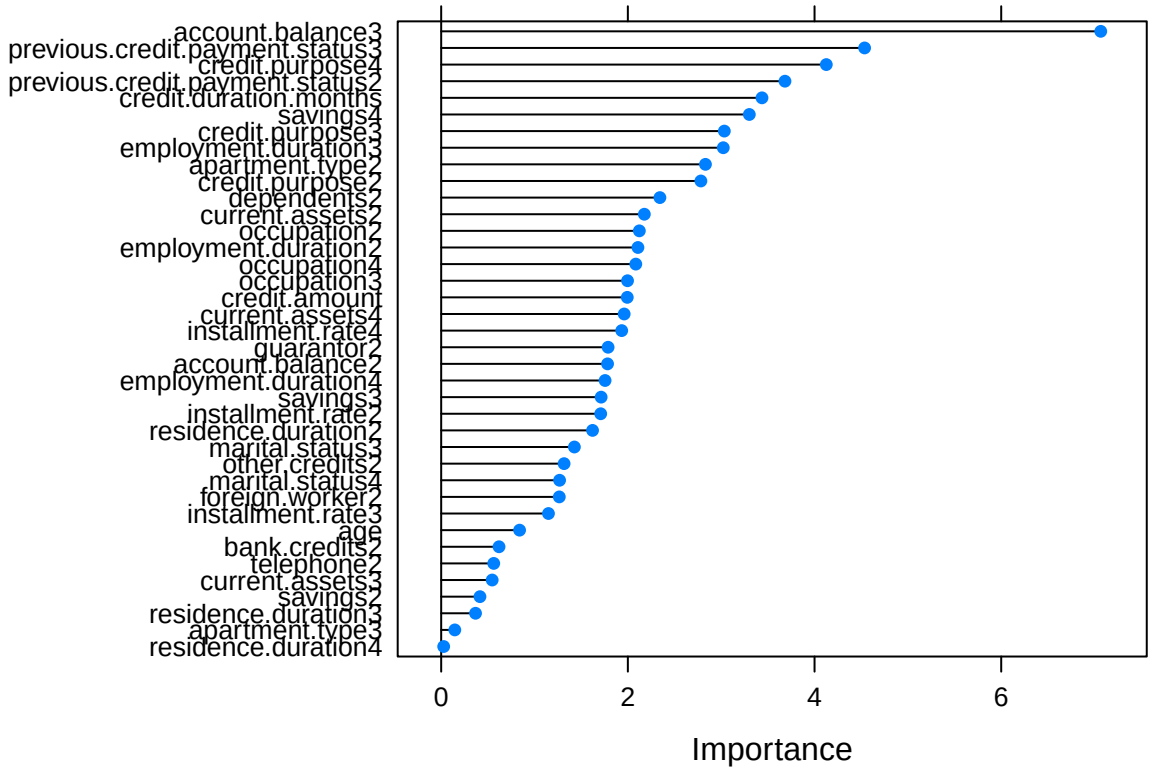
```r
formula <- "credit.rating ~ ."
formula <- as.formula(formula)
control <- trainControl(method = "repeatedcv", number = 10, repeats = 2)
model <- train(formula, data = treino, method = "glm", trControl = control)
print(model)
```

```
## Generalized Linear Model
##
## 840 samples
##  20 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 2 times)
## Summary of sample sizes: 756, 756, 756, 756, 756, 756, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7053571  0.4107143
```

```r
importance <- varImp(model, scale = FALSE)
importance
```

```
## glm variable importance
##
##   only 20 most important variables shown (out of 38)
##
##                                 Overall
## account.balance3                  7.066
## previous.credit.payment.status3   4.536
## credit.purpose4                   4.126
## previous.credit.payment.status2   3.683
## credit.duration.months            3.437
## savings4                          3.301
## credit.purpose3                   3.033
## employment.duration3              3.022
## apartment.type2                   2.831
## credit.purpose2                   2.782
## dependents2                       2.343
## current.assets2                   2.176
## occupation2                       2.123
## employment.duration2              2.108
## occupation4                       2.085
## occupation3                       1.996
## credit.amount                     1.994
## current.assets4                   1.961
## installment.rate4                 1.934
## guarantor2                        1.788
```

```r
plot(importance)
```



Pode se ver que as 6 varáveis mais importantes são: - account.balance - credit.purpose - credit.amount - previous.credit.payment.status - savings - current.assets

Construindo o modelo com as variáveis selecionadas

```
formula.new <- "credit.rating ~ account.balance + credit.purpose + previous.credit.payment.status + savi
formula.new <- as.formula(formula.new)
modelo_RL_2 <- glm(formula = formula.new, data = treino, family = "binomial")
```

Visualizando o modelo

```
summary(modelo_RL_2)
```

```
##
## Call:
## glm(formula = formula.new, family = "binomial", data = treino)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.15422  -0.93900   0.08157   0.93194   2.54622
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -0.52821    0.40922  -1.291 0.196790
## account.balance2                0.39401    0.20394   1.932 0.053358 .
## account.balance3                1.46236    0.20181   7.246 4.29e-13 ***
```

```
## credit.purpose2                 -1.03849    0.35941  -2.889 0.003860 **
## credit.purpose3                 -1.08525    0.34096  -3.183 0.001458 **
## credit.purpose4                 -1.36478    0.33023  -4.133 3.58e-05 ***
## previous.credit.payment.status2  1.17777    0.26234   4.489 7.14e-06 ***
## previous.credit.payment.status3  1.42688    0.27072   5.271 1.36e-07 ***
## savings2                         0.09638    0.25652   0.376 0.707119
## savings3                         0.51455    0.28277   1.820 0.068807 .
## savings4                         0.85283    0.22713   3.755 0.000173 ***
## current.assets2                 -0.52797    0.22962  -2.299 0.021485 *
## current.assets3                 -0.20098    0.21009  -0.957 0.338745
## current.assets4                 -0.97271    0.26357  -3.690 0.000224 ***
## credit.amount                   -0.50215    0.10253  -4.898 9.69e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1164.49  on 839  degrees of freedom
## Residual deviance:  940.63  on 825  degrees of freedom
## AIC: 970.63
##
## Number of Fisher Scoring iterations: 4
```

Testando o modelo nos dados de teste

```
prevendo_RL_2 <- predict(modelo_RL_2, teste, type = "response")
prevendo_RL_2 <- round(prevendo_RL_2)
```

Avaliando o modelo

```
CF_2 <- confusionMatrix(table(data = prevendo_RL_2, reference = test.class.var), positive = '1')
CF_2$table
```

```
##     reference
## data   0   1
##    0 120  48
##    1  60 132
```

Acurácia do modelo de regressão logística com as 6 variáveis mais importantes, indicadas pelo modelo random forest:

```
CF_2$overall["Accuracy"]
```

```
## Accuracy
##      0.7
```

Criação de um vetor para salvar a acurácia de todos os modelos estudados

```
accuracyVector <- c(CF_1$overall["Accuracy"],CF_2$overall["Accuracy"])
```

Criação do modelo Random Forest com todas as variáveis:

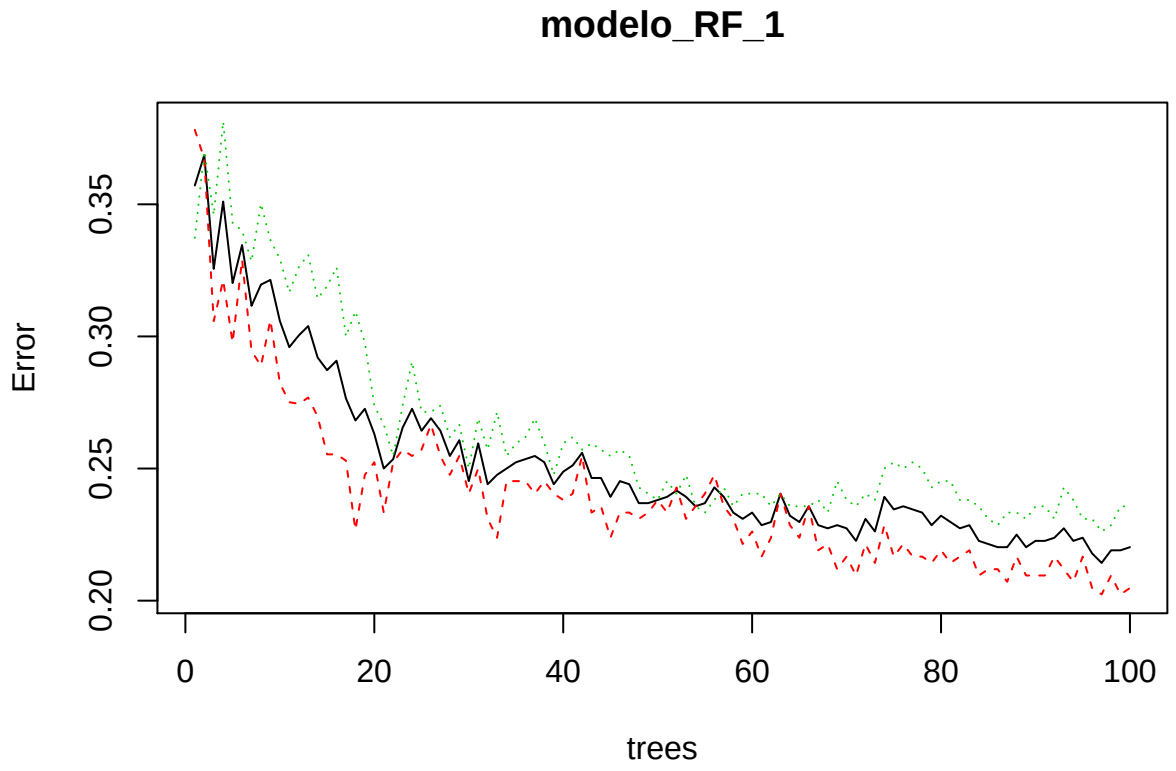
```
modelo_RF_1 <- randomForest(credit.rating ~ .,
                        data = treino,
                        ntree = 100,
                        nodesize = 10)


print(modelo_RF_1)
```

```
##
## Call:
##  randomForest(formula = credit.rating ~ ., data = treino, ntree = 100,      nodesize = 10)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 22.02%
## Confusion matrix:
##     0   1 class.error
## 0 334  86   0.2047619
## 1  99 321   0.2357143
```

Visualizando o comportamento do erro do modelo:

```
plot(modelo_RF_1)
```



Fazendo as previsões com o modelo random forest:

```
prevendo_RF_1 <- predict(modelo_RF_1, newdata = teste)
head(prevendo_RF_1)
```

```
## 639 106 309  31 140  74
##   1   1   0   1   1   1
## Levels: 0 1
```

Visualizando os valores previstos e observados:

```
resultados_RF_1 <- cbind(prevendo_RF_1, teste$credit.rating)
colnames(resultados_RF_1) <- c('Previsto','Real')
resultados_RF_1 <- as.data.frame(resultados_RF_1)
head(resultados_RF_1)
```

```
##     Previsto Real
## 639        2    2
## 106        2    2
## 309        1    2
## 31         2    2
## 140        2    2
## 74         2    2
```

Gerando a confusion matrix do modelo random forest com todas as variáveis:

```
resultados_RF_1$Previsto <- factor(resultados_RF_1$Previsto)
resultados_RF_1$Real <- factor(resultados_RF_1$Real)

CF_3 <- confusionMatrix(resultados_RF_1$Real, resultados_RF_1$Previsto)
CF_3$table
```

```
##           Reference
## Prediction   1   2
##          1 146  34
##          2  35 145
```

Acurácia do modelo random forest com todas as variáveis:

```
CF_3$overall["Accuracy"]
```

```
##  Accuracy
## 0.8083333
```

Salvando a acurácia no vetor:

```
accuracyVector <- c(accuracyVector,CF_3$overall["Accuracy"])
```

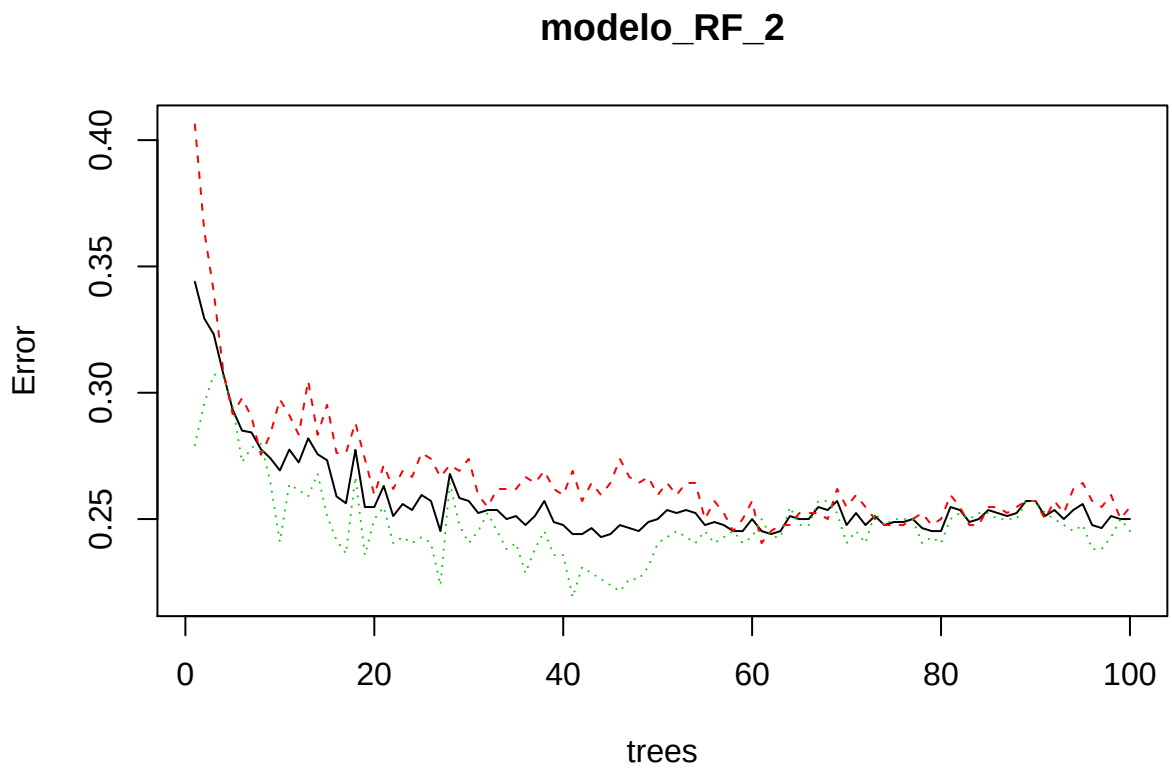
Criação do modelo Random Forest com seis variáveis:

```
modelo_RF_2 <- randomForest(credit.rating ~ account.balance
                            + previous.credit.payment.status
                            + credit.duration.months
                            + credit.amount
                            + current.assets
                            + savings,
                            data = treino,
                            ntree = 100,
                            nodesize = 10)
print(modelo_RF_2)
```

```
##
## Call:
##  randomForest(formula = credit.rating ~ account.balance + previous.credit.payment.status +      credi
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 25%
## Confusion matrix:
##     0   1 class.error
## 0 313 107   0.2547619
## 1 103 317   0.2452381
```

Visualizando o comportamento do erro do modelo:

```
plot(modelo_RF_2)
```



Fazendo as previsões:

```
prevendo_RF_2 <- predict(modelo_RF_2, newdata = teste)
head(prevendo_RF_2)
```

```
## 639 106 309  31 140  74
##   1   1   0   1   1   1
## Levels: 0 1
```

Visualizando os valores previstos e observados:

```
resultados_RF_2 <- cbind(prevendo_RF_2, teste$credit.rating)
colnames(resultados_RF_2) <- c('Previsto','Real')
resultados_RF_2 <- as.data.frame(resultados_RF_2)
head(resultados_RF_2)
```

```
##     Previsto Real
## 639        2    2
## 106        2    2
## 309        1    2
## 31         2    2
## 140        2    2
## 74         2    2
```

Gerando a confusion matrix do modelo random forest com 6 variáveis:

```
resultados_RF_2$Previsto <- factor(resultados_RF_2$Previsto)
resultados_RF_2$Real <- factor(resultados_RF_2$Real)

CF_4 <- confusionMatrix(resultados_RF_2$Real, resultados_RF_2$Previsto)
CF_4$table
```

```
##           Reference
## Prediction   1   2
##          1 132  48
##          2  39 141
```

Acurácia do modelo:

```
CF_4$overall["Accuracy"]
```

```
##  Accuracy
## 0.7583333
```

Salvando a acurácia no vetor:

```
accuracyVector <- c(accuracyVector,CF_4$overall["Accuracy"])
```

Modelo Naive Bayes com todas as variáveis:

```
modelo_NB_1 <- naiveBayes(credit.rating ~ .,treino)
print(modelo_NB_1)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   0   1
## 0.5 0.5
##
## Conditional probabilities:
##    account.balance
## Y           1         2         3
##   0 0.3880952 0.3904762 0.2214286
##   1 0.1904762 0.2595238 0.5500000
##
##    credit.duration.months
## Y         [,1]      [,2]
##   0  0.3003034 1.0238460
##   1 -0.2213872 0.8429229
##
##    previous.credit.payment.status
## Y            1          2          3
```

```
##   0 0.23333333 0.48809524 0.27857143
##   1 0.06428571 0.50714286 0.42857143
##
##    credit.purpose
## Y           1          2          3          4
##   0 0.06428571 0.16428571 0.30000000 0.47142857
##   1 0.10476190 0.17857143 0.37619048 0.34047619
##
##    credit.amount
## Y         [,1]      [,2]
##   0  0.2608186 1.1452485
##   1 -0.1757464 0.7692527
##
##    savings
## Y           1          2          3          4
##   0 0.67619048 0.13571429 0.06904762 0.11904762
##   1 0.54047619 0.10952381 0.13571429 0.21428571
##
##    employment.duration
## Y           1         2         3         4
##   0 0.2952381 0.3380952 0.1285714 0.2380952
##   1 0.1761905 0.3452381 0.1833333 0.2952381
##
##    installment.rate
## Y           1         2         3         4
##   0 0.1047619 0.2428571 0.1428571 0.5095238
##   1 0.1380952 0.2095238 0.1666667 0.4857143
##
##    marital.status
## Y           1          3          4
##   0 0.39761905 0.54285714 0.05952381
##   1 0.32380952 0.56428571 0.11190476
##
##    guarantor
## Y           1         2
##   0 0.8928571 0.1071429
##   1 0.8785714 0.1214286
##
##    residence.duration
## Y           1         2         3         4
##   0 0.1166667 0.3071429 0.1547619 0.4214286
##   1 0.1166667 0.2976190 0.1547619 0.4309524
##
##    current.assets
## Y           1         2         3         4
##   0 0.1904762 0.2452381 0.3071429 0.2571429
##   1 0.3190476 0.2119048 0.3500000 0.1190476
##
##    age
## Y          [,1]     [,2]
##   0 -0.06629672 0.900965
##   1  0.04768335 1.033655
##
##    other.credits
```

```
## Y           1         2
##   0 0.3142857 0.6857143
##   1 0.1809524 0.8190476
##
##    apartment.type
## Y           1          2          3
##   0 0.24285714 0.58333333 0.17380952
##   1 0.19047619 0.72857143 0.08095238
##
##    bank.credits
## Y           1         2
##   0 0.6309524 0.3690476
##   1 0.6047619 0.3952381
##
##    occupation
## Y           1          2          3          4
##   0 0.02142857 0.21666667 0.56190476 0.20000000
##   1 0.03095238 0.20000000 0.64285714 0.12619048
##
##    dependents
## Y           1         2
##   0 0.7809524 0.2190476
##   1 0.8500000 0.1500000
##
##    telephone
## Y           1         2
##   0 0.5714286 0.4285714
##   1 0.5833333 0.4166667
##
##    foreign.worker
## Y           1          2
##   0 0.97619048 0.02380952
##   1 0.95238095 0.04761905
```

```
prevendo_NB_1 <- predict(modelo_NB_1, teste[,-21])
head(prevendo_NB_1)
```

```
## [1] 1 1 0 1 1 1
## Levels: 0 1
```

```
table(prevendo_NB_1, true = teste$credit.rating)
```

```
##              true
## prevendo_NB_1   0   1
##             0 115  39
##             1  65 141
```

Visualizando os valores previstos e observados

```
resultados_NB_1 <- cbind(prevendo_NB_1, teste$credit.rating)
colnames(resultados_NB_1) <- c('Previsto','Real')
resultados_NB_1 <- as.data.frame(resultados_NB_1)
head(resultados_NB_1)
```

```
##   Previsto Real
## 1        2    2
## 2        2    2
## 3        1    2
## 4        2    2
## 5        2    2
## 6        2    2
```

Gerando a confusion matrix:

```
resultados_NB_1$Previsto <- factor(resultados_NB_1$Previsto)
resultados_NB_1$Real <- factor(resultados_NB_1$Real)

CF_5 <- confusionMatrix(resultados_NB_1$Real, resultados_NB_1$Previsto)
CF_5$table
```

```
##           Reference
## Prediction   1   2
##          1 115  65
##          2  39 141
```

Acurácia do modelo Naive Bayes com todas as variáveis do dataset:

```
CF_5$overall["Accuracy"]
```

```
##  Accuracy
## 0.7111111
```

Salvando no vetor de acurácia:

```
accuracyVector <- c(accuracyVector, CF_5$overall["Accuracy"])
```

Modelo Naive Bayes com 6 variáveis:

```
modelo_NB_2 <- naiveBayes(credit.rating ~ account.balance
                          + previous.credit.payment.status
                          + credit.duration.months
                          + credit.amount
                          + current.assets
                          + savings,
                          treino)
print(modelo_NB_2)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
```

```
##   0   1
## 0.5 0.5
##
## Conditional probabilities:
##    account.balance
## Y           1         2         3
##   0 0.3880952 0.3904762 0.2214286
##   1 0.1904762 0.2595238 0.5500000
##
##    previous.credit.payment.status
## Y            1          2          3
##   0 0.23333333 0.48809524 0.27857143
##   1 0.06428571 0.50714286 0.42857143
##
##    credit.duration.months
## Y         [,1]      [,2]
##   0  0.3003034 1.0238460
##   1 -0.2213872 0.8429229
##
##    credit.amount
## Y         [,1]      [,2]
##   0  0.2608186 1.1452485
##   1 -0.1757464 0.7692527
##
##    current.assets
## Y           1         2         3         4
##   0 0.1904762 0.2452381 0.3071429 0.2571429
##   1 0.3190476 0.2119048 0.3500000 0.1190476
##
##    savings
## Y            1          2          3          4
##   0 0.67619048 0.13571429 0.06904762 0.11904762
##   1 0.54047619 0.10952381 0.13571429 0.21428571
```

```
prevendo_NB_2 <- predict(modelo_NB_2, teste[,-21])
head(prevendo_NB_2)
```

```
## [1] 0 1 0 1 1 1
## Levels: 0 1
```

```
table(prevendo_NB_2, true = teste$credit.rating)
```

```
##              true
## prevendo_NB_2   0   1
##             0 102  36
##             1  78 144
```

Visualizando os valores previstos e observados

```
resultados_NB_2 <- cbind(prevendo_NB_2, teste$credit.rating)
colnames(resultados_NB_2) <- c('Previsto','Real')
resultados_NB_2 <- as.data.frame(resultados_NB_2)
head(resultados_NB_2)
```

```
##   Previsto Real
## 1        1    2
## 2        2    2
## 3        1    2
## 4        2    2
## 5        2    2
## 6        2    2
```

Geraando confusion matrix:

```
resultados_NB_2$Previsto <- factor(resultados_NB_2$Previsto)
resultados_NB_2$Real <- factor(resultados_NB_2$Real)

CF_6 <- confusionMatrix(resultados_NB_2$Real, resultados_NB_2$Previsto)
CF_6$table
```

```
##           Reference
## Prediction   1   2
##          1 102  78
##          2  36 144
```

Acurácia do modelo Naive Bayes com 6 variáveis:

```
CF_6$overall["Accuracy"]
```

```
##  Accuracy
## 0.6833333
```

```
accuracyVector <- c(accuracyVector, CF_6$overall["Accuracy"])
```

Modelo Support Vector Machine com todas as variáveis- SVM

```
modelo_SVM_1 <- ksvm(credit.rating ~ .,data = treino, kernel="vanilladot" )
```

```
##  Setting default kernel parameters
```

Fazendo previsões com os dados de teste:

```
prevendo_SVM_1 <- predict(modelo_SVM_1, teste)
head(prevendo_SVM_1)
```

```
## [1] 0 1 0 1 1 1
## Levels: 0 1
```

Visualizando os valores previstos e observados

```
resultados_SVM_1 <- cbind(prevendo_SVM_1, teste$credit.rating)
colnames(resultados_SVM_1) <- c('Previsto','Real')
resultados_SVM_1 <- as.data.frame(resultados_SVM_1)
head(resultados_SVM_1)
```

```
##   Previsto Real
## 1        1    2
## 2        2    2
## 3        1    2
## 4        2    2
## 5        2    2
## 6        2    2
```

Gerando a confusion matrix para o modelo support vector machine:

```
resultados_SVM_1$Previsto <- factor(resultados_SVM_1$Previsto)
resultados_SVM_1$Real <- factor(resultados_SVM_1$Real)
CF_7 <- confusionMatrix(resultados_SVM_1$Real, resultados_SVM_1$Previsto)
CF_7$table
```

```
##           Reference
## Prediction   1   2
##          1 135  45
##          2  43 137
```

Acurácia do modelo SVM com todas as variáveis:

```
CF_7$overall["Accuracy"]
```

```
##  Accuracy
## 0.7555556
```

Salvando a acurácia no vetor:

```
accuracyVector <- c(accuracyVector, CF_7$overall["Accuracy"])
```

Modelo Suport Vector Machine com 6 variáveis- SVM:

```
modelo_SVM_2 <- ksvm(credit.rating ~ account.balance
                     + previous.credit.payment.status
                     + credit.duration.months
                     + credit.amount
                     + current.assets
                     + savings,
                     data = treino, kernel="vanilladot" )
```

```
##  Setting default kernel parameters
```

Fazendo previsões com os dados de teste:

```
prevendo_SVM_2 <- predict(modelo_SVM_2, teste)
head(prevendo_SVM_2)
```

```
## [1] 0 1 0 1 1 1
## Levels: 0 1
```

Visualizando os valores previstos e observados:

```
resultados_SVM_2 <- cbind(prevendo_SVM_2, teste$credit.rating)
colnames(resultados_SVM_2) <- c('Previsto','Real')
resultados_SVM_2 <- as.data.frame(resultados_SVM_2)
head(resultados_SVM_2)
```

```
##   Previsto Real
## 1        1    2
## 2        2    2
## 3        1    2
## 4        2    2
## 5        2    2
## 6        2    2
```

Gerando a confusion matrix:

```
resultados_SVM_2$Previsto <- factor(resultados_SVM_2$Previsto)
resultados_SVM_2$Real <- factor(resultados_SVM_2$Real)

CF_8 <- confusionMatrix(resultados_SVM_2$Real, resultados_SVM_2$Previsto)
CF_8$table
```

```
##           Reference
## Prediction   1   2
##          1 132  48
##          2  51 129
```

Acurácia do modelo SVM com 6 variáveis:

```
CF_8$overall["Accuracy"]
```

```
## Accuracy
##    0.725
```

Salvando no vetor de acurácias:

```
accuracyVector <- c(accuracyVector, CF_8$overall["Accuracy"])
```

Criando um dataframe com todas as acurácias conseguidas nos 6 modelos testados.

```
Modelos <- c("Regressao Logistica Todas", "Regressao Logistica 6","RandomForest Todas", "RandomForest 6
             "Naive Bayes 6", "SVM Todas", "SVM 6")
accuracyDataFrame <- data.frame(Modelos, accuracyVector)
colnames(accuracyDataFrame) <- c("Modelos", "Acurácia")
head(accuracyDataFrame)
```

```
##                     Modelos  Acurácia
## 1 Regressao Logistica Todas 0.7444444
## 2     Regressao Logistica 6 0.7000000
## 3        RandomForest Todas 0.8083333
## 4            RandomForest 6 0.7583333
## 5         Naive Bayes Todas 0.7111111
## 6             Naive Bayes 6 0.6833333
```

Pode se ver que o modelo que alcançou a maior acurácia foi o random forest com todas as variáveis. O próximo passo seria otimizar o modelo de forma a aumentar a acurácia.