Modelo Preditivo para Análise de Risco

Renato Jorge Dyszy

09/05/2020

Mini-Projeto - Modelo Preditivo para Análise de Risco

Para esta análise, vamos usar um conjunto de dados German Credit Data, já devidamente limpo e organizado para a criação do modelo preditivo.

Todo o projeto será descrito de acordo com suas etapas.

Etapa 1 - Coletando os Dados

Aqui está a coleta de dados, neste caso um arquivo csv.

```
# Coletando dados
  dataset.df <- read.csv("credit_dataset.csv", header = TRUE, sep = ",")
  str(dataset.df)
```

```
## 'data.frame':    1000 obs. of  21 variables:
##  $ credit.rating                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ account.balance               : int  1 1 2 1 1 1 1 1 3 2 ...
##  $ 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 ...
##  $ credit.purpose                : int  2 4 4 4 4 4 4 4 3 3 ...
##  $ credit.amount                 : int  1049 2799 841 2122 2171 2241 3398 1361 1098 3758 ...
##  $ savings                       : int  1 1 2 1 1 1 1 1 1 3 ...
##  $ employment.duration           : int  1 2 3 2 2 1 3 1 1 1 ...
##  $ installment.rate              : int  4 2 2 3 4 1 1 2 4 1 ...
##  $ marital.status                : int  1 3 1 3 3 3 3 3 1 1 ...
##  $ guarantor                     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ residence.duration            : int  4 2 4 2 4 3 4 4 4 4 ...
##  $ current.assets                : int  2 1 1 1 2 1 1 1 3 4 ...
##  $ 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 ...
##  $ apartment.type                : int  1 1 1 1 2 1 2 2 2 1 ...
##  $ bank.credits                  : int  1 2 1 2 2 2 2 1 2 1 ...
##  $ occupation                    : int  3 3 2 2 2 2 2 2 1 1 ...
##  $ dependents                    : int  1 2 1 2 1 2 1 2 1 1 ...
##  $ telephone                     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ foreign.worker                : int  1 1 1 2 2 2 2 2 1 1 ...
```

Etapa 2 - Funções utilizadas

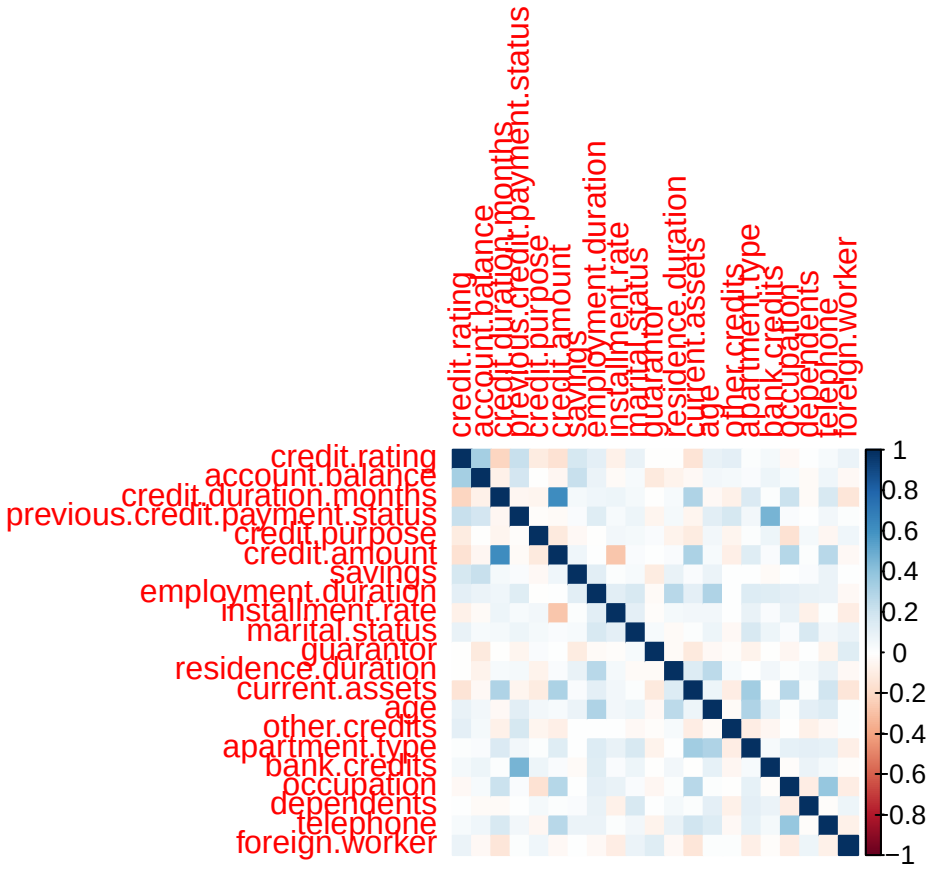
```
# Função para converter variáveis numéricas para fator
toFactors <- function ( df, vars) {

  for (variable in vars ){
    df[[variable]] <- as.factor(df[[variable]])
  }

  return(df)
}
# ----------

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

# Geração de curvas ROC
  library(ROCR)
  plot.roc.curve <- function(predictions, title.text){
    perf <- performance(predictions, "tpr", "fpr")
    plot(perf, col = "black", lty = 1, lwd = 2,
         main = title.text, cex.main = 0.6,
         cex.lab = 0.8, xaxs="i", yaxs="i")
    abline(0,1, col = "red")
    auc <- performance(predictions, "auc")
    auc <- unlist(slot(auc, "y.values"))
    auc <- round(auc,2)
    legend(0.4,0.4, legend = c(paste0("AUC: ",auc)), cex = 0.6, bty = "n", box.col = "white")

  }

  plot.pr.curve <- function(predictions, title.text){
    perf <- performance(predictions, "prec", "rec")
    plot(perf, col = "black", lty = 1, lwd = 2,
         main = title.text, cex.main = 0.6, cex.lab = 0.8, xaxs = "i", yaxs = "i")
  }
# -----------
```

Etapa 3 - Correlação entre as váriaveis

```
df <- as.data.frame(dataset.df)
data_cor <- cor(df)

library(corrplot)
```

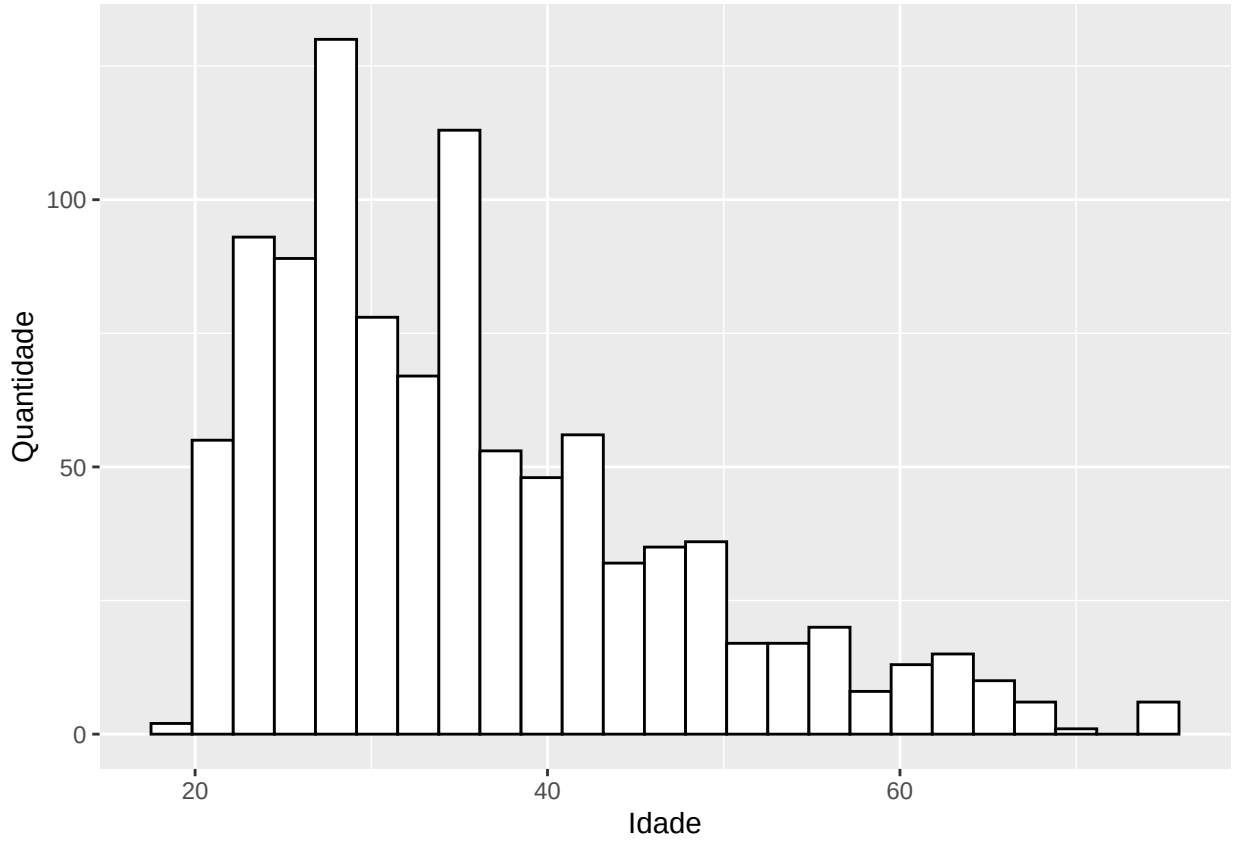
```
## corrplot 0.84 loaded
```

```
corrplot(data_cor, method = 'color')
```

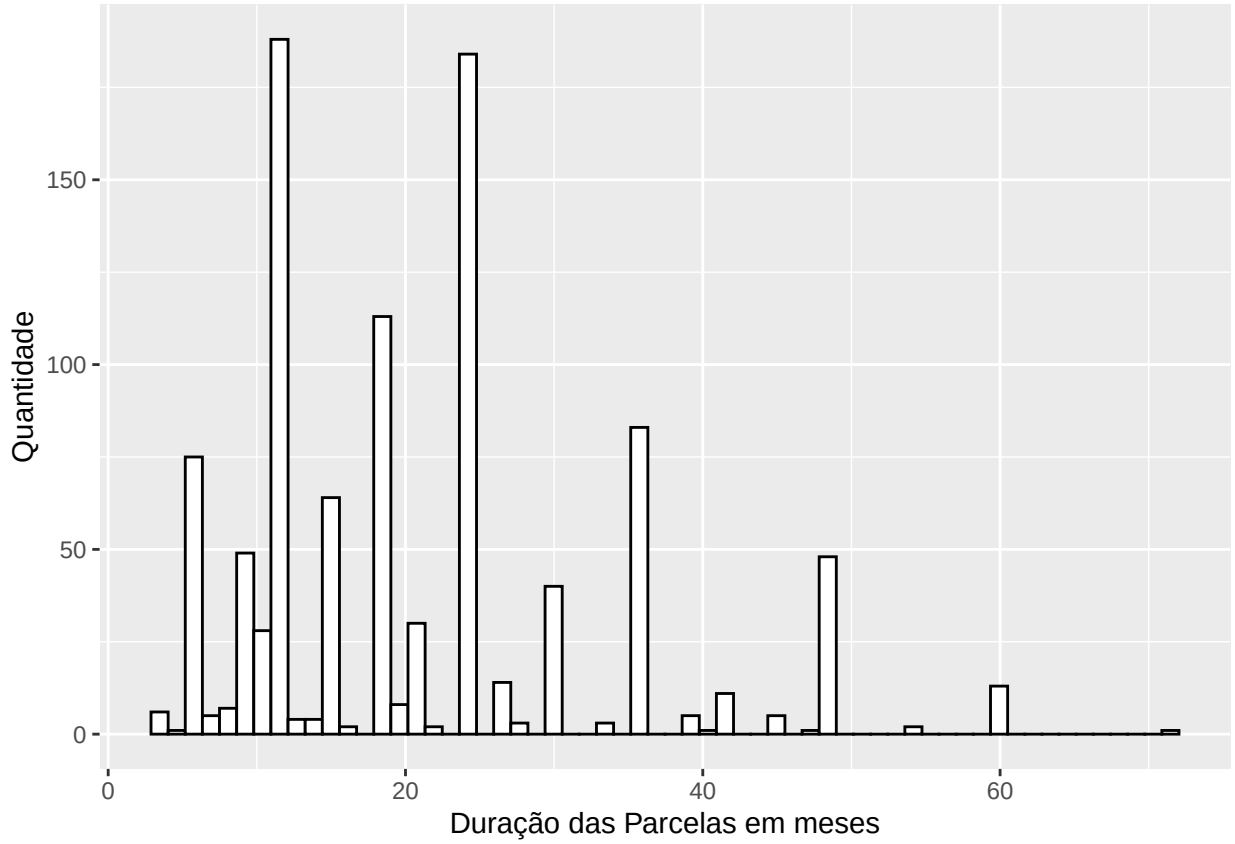


Etapa 4 - Gráficos

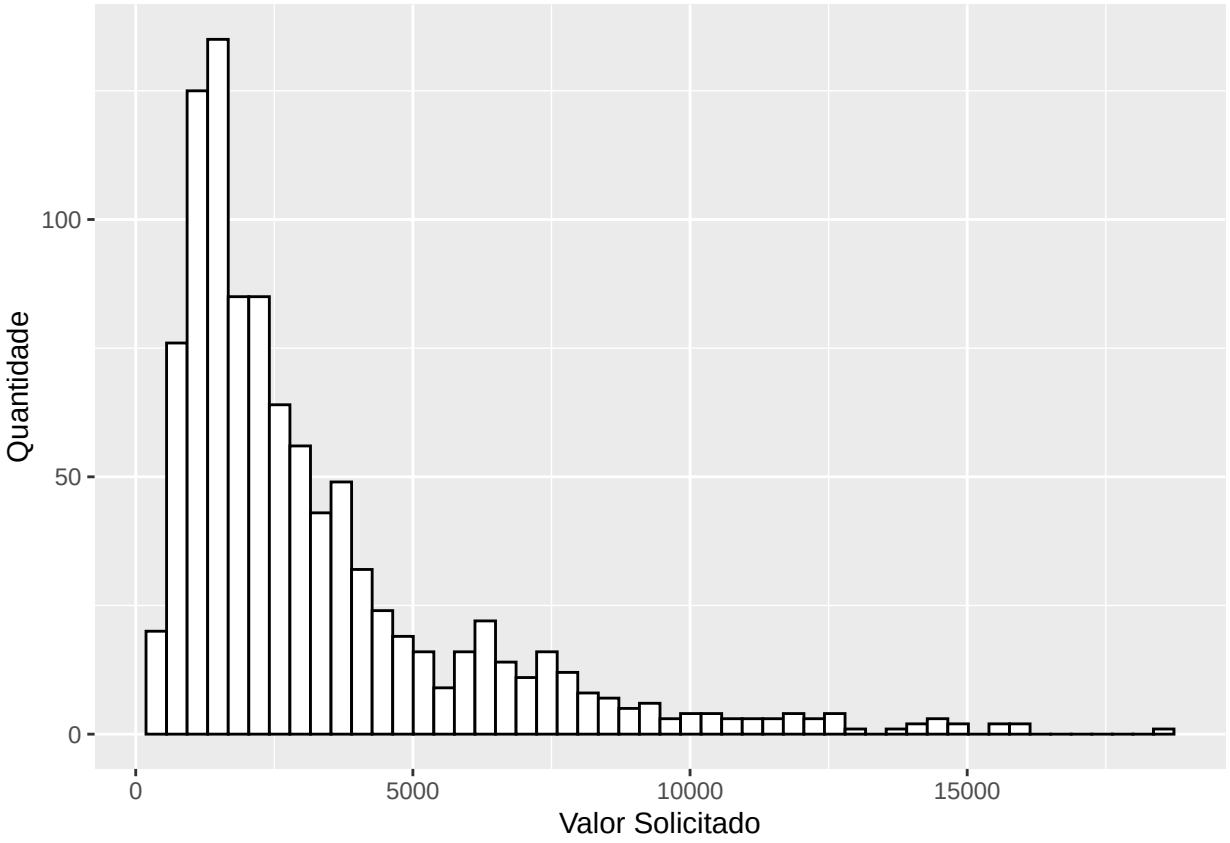
```
library(ggplot2)

ggplot(dataset.df) +
  geom_histogram(aes(x=age), bins = 25, color = "black", fill = "white") +
    xlab("Idade") + ylab("Quantidade")
```




```
ggplot(dataset.df) +
  geom_histogram(aes(x=credit.duration.months), color = "black", bins = 60, fill = "white") +
    xlab("Duração das Parcelas em meses") + ylab("Quantidade")
```




```
ggplot(dataset.df) +
  geom_histogram(aes(x=credit.amount), bins = 50, color = "black", fill = "white") +
    xlab("Valor Solicitado") + ylab("Quantidade")
```



Etapa 5 - Normalizando os Dados

```
# Normalização
numeric.vars <- c('credit.duration.months','credit.amount','age')
dataset.df <- scale.features(dataset.df, numeric.vars)

# Convertendo as variáveis para o tipo fator (categórica)
varFactors <- 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')
  dataset.df <- toFactors(dataset.df, varFactors)
```

Etapa 6 - Dividindo os dados em dados de treino e de teste

```
  library(caTools)

  amostra <- sample.split(dataset.df$credit.rating, SplitRatio = 0.60)
  dataset.df.treino <- subset(dataset.df, amostra == TRUE)
  dataset.df.teste <- subset(dataset.df, amostra == FALSE)
```

Etapa 7 - Criando e Avaliando a Primeira Versão do Modelo

```
  # install.packages("e1071")
  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
```

```
  # Modelo com todas as váriaveis usando Random Forest
    rf.model <- randomForest(credit.rating ~ . , dataset.df.treino, ntree = 100, nodesize = 10)
    rf.predictions <- predict(rf.model, dataset.df.teste, type="response")

    # Gerando Confusion Matrix
    confusionMatrix(table(data = rf.predictions, reference = dataset.df.teste[,1]), positive = '1')
```

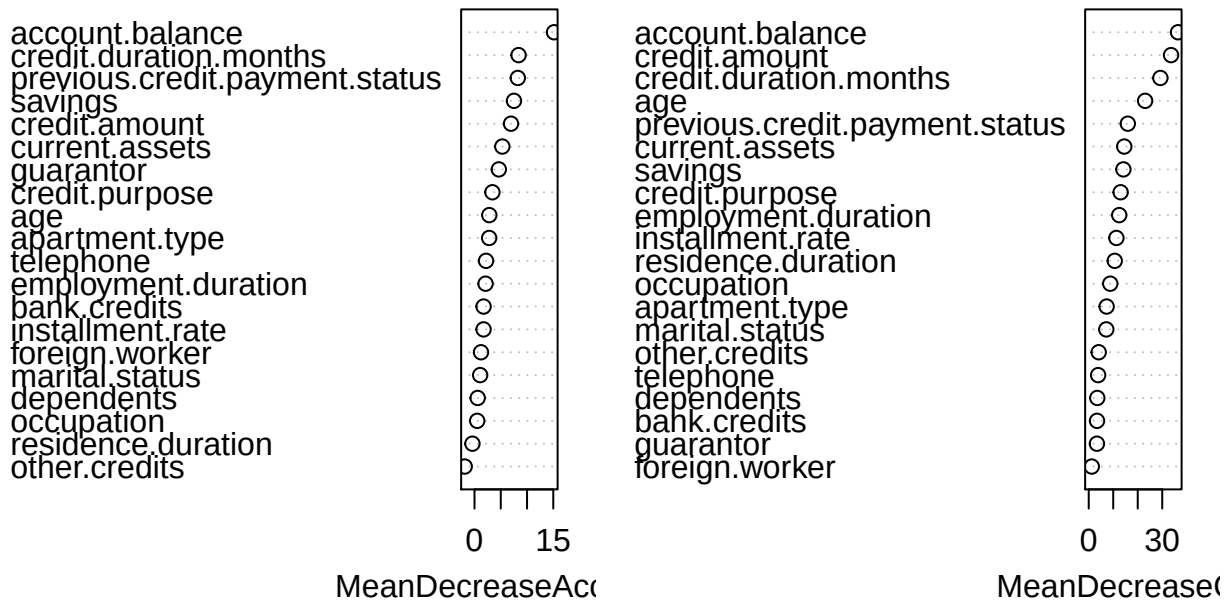
```
## Confusion Matrix and Statistics
##
##     reference
## data   0   1
##    0  39  19
##    1  81 261
##
##                Accuracy : 0.75
```

```
##                  95% CI : (0.7046, 0.7917)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.01553
##
##                   Kappa : 0.3017
##
##  Mcnemar's Test P-Value : 1.061e-09
##
##             Sensitivity : 0.9321
##             Specificity : 0.3250
##          Pos Pred Value : 0.7632
##          Neg Pred Value : 0.6724
##              Prevalence : 0.7000
##          Detection Rate : 0.6525
##    Detection Prevalence : 0.8550
##       Balanced Accuracy : 0.6286
##
##        'Positive' Class : 1
##
```

```
# -----------
```

Etapa 8 - Feature Selection

```
library(randomForest)

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



Etapa 9 - Otimizando o Modelo

```
formula <- ("credit.rating ~ account.balance +
                             credit.duration.months +
                             previous.credit.payment.status +
                             credit.amount +
                             savings +
                             apartment.type +
                             credit.purpose")
formula <- as.formula(formula)

rf.model <- randomForest(formula, dataset.df.treino, ntree = 100, nodesize = 10)
rf.predictions <- predict(rf.model, dataset.df.teste, type="response")

# Imprimindo o resultado
print(rf.model)
```

```
##
## Call:
##  randomForest(formula = formula, data = dataset.df.treino, ntree = 100,      nodesize = 10)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
```

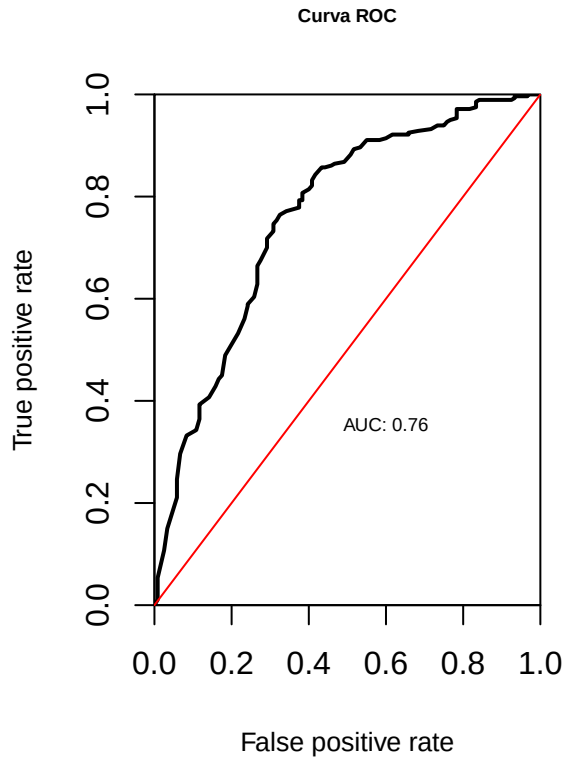
```
##
##         OOB estimate of  error rate: 25.83%
## Confusion matrix:
##    0   1 class.error
## 0 73 107   0.5944444
## 1 48 372   0.1142857
```

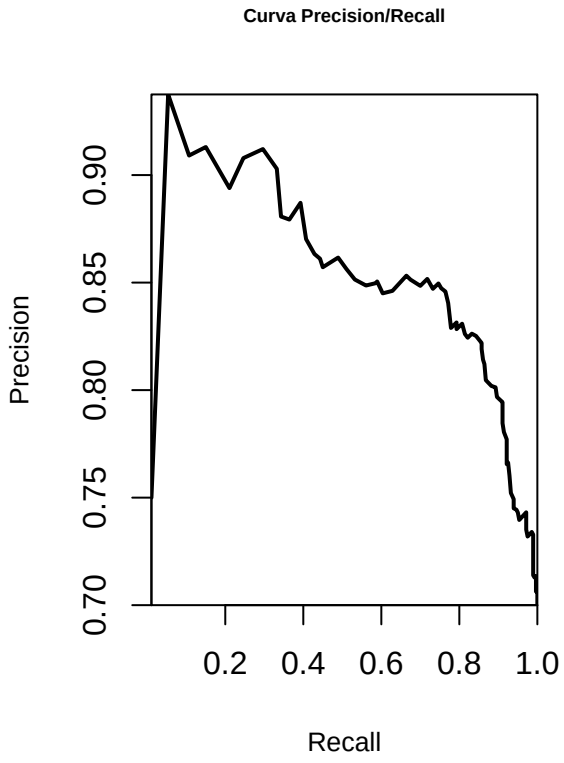
```
  # Gerando Confusino Matrix
  confusionMatrix(table(data = rf.predictions, reference = dataset.df.teste[,1]), positive = '1')
```

```
## Confusion Matrix and Statistics
##
##     reference
## data   0   1
##    0  58  30
##    1  62 250
##
##                Accuracy : 0.77
##                  95% CI : (0.7256, 0.8104)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.001077
##
##                   Kappa : 0.4072
##
##  Mcnemar's Test P-Value : 0.001229
##
##             Sensitivity : 0.8929
##             Specificity : 0.4833
##          Pos Pred Value : 0.8013
##          Neg Pred Value : 0.6591
##              Prevalence : 0.7000
##          Detection Rate : 0.6250
##    Detection Prevalence : 0.7800
##       Balanced Accuracy : 0.6881
##
##        'Positive' Class : 1
##
```

Etapa 10 - Curva ROC e Avaliação Final do Modelo

```
  class1 <- predict(rf.model, newdata = dataset.df.teste, type = 'prob')
  class2 <- dataset.df.teste$credit.rating

  pred <- prediction(class1[,2], class2)
  perf <- performance(pred, "tpr","fpr")

  par(mfrow = c(1,2))
  plot.roc.curve(pred, title.text = "Curva ROC")
  plot.pr.curve(pred, title.text = "Curva Precision/Recall")
```


Curva ROC
True positive rate
False positive rate
AUC: 0.76
0.0
0.2
0.4
0.6
0.8
1.0


Curva Precision/Recall
Precision
Recall
0.70
0.75
0.80
0.85
0.90
0.2
0.4
0.6
0.8
1.0

Fim