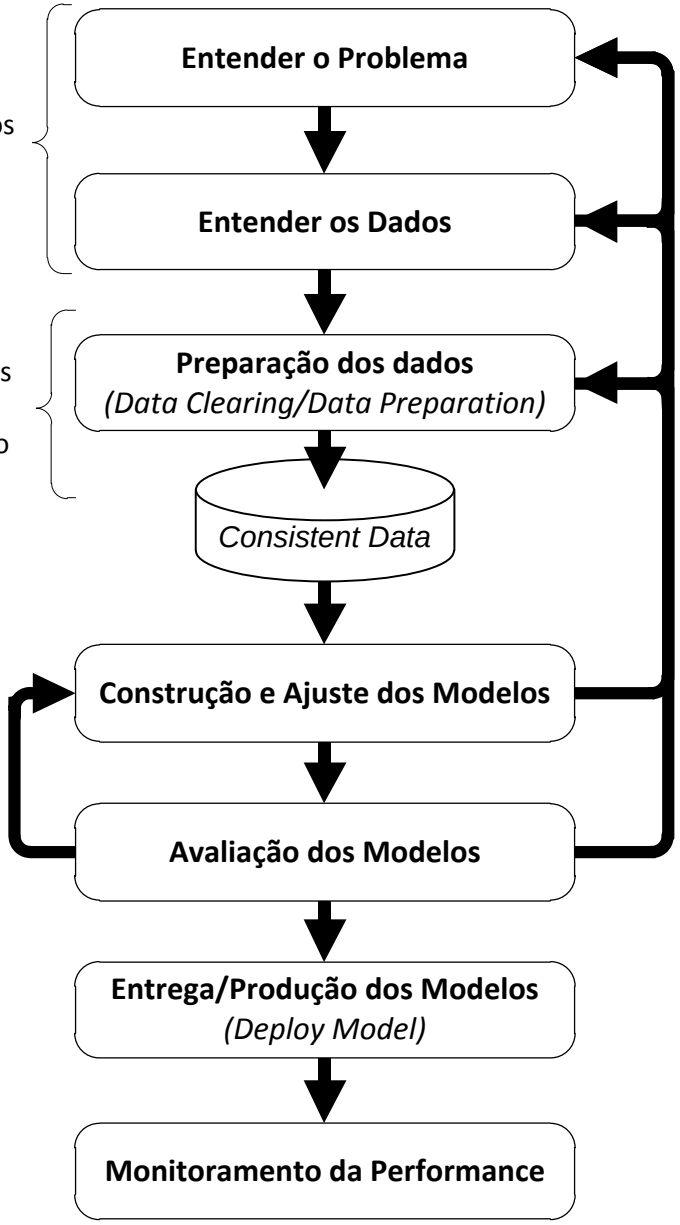
Decision Tree e Credit Scoring

Decision Tree and Credit Scoring

Considere o problema de avaliar o crédito de um tomador de empréstimo com base em informações pessoais e financeiras, como por exemplo

- Informações Pessoais
  - Nível de Escolaridade
  - Idade
  - Estado Civil, etc...
- Informações do Empréstimo
  - Duração
  - Montante, etc...
- Informações Financeiras
  - Montante de poupança
  - Patrimônio, etc...

Decision Tree and Credit Scoring



- Feito uma única vez
- Utilizar aplicativos interativos
- Normalmente não há necessidade de automatizar

Automatizar a execução no caso de Projetos ou Modelos em que este procedimento será recorrente ou o modelo esteja sujeito a alteração

- Brain Storn, experiência própria e conversar com as partes interessadas
- Definir objetivo (OUTPUT):
  Classificação do crédito como em duas categorias: Bom ou Ruim

- Análise Exploratória dos dados
- Regressões, tabelas de contingência, histogramas
- Medidas de resumos das variáveis
- Classificação das variáveis

- Padronização dos dados
- Categorização dos dados
- Operações aritméticas
- Inclusão de variáveis/exclusão

Modelos possíveis:
- **Decision Trees / Random Forests**
- Redes Bayesianas
- Regressões Logit

- Confusion Matrix
- ROC *Receiver Operating Characteristic Curve*
- *Cross-Folder validadtion*
- Erro Quadrado Médio
- Etc..

Base de Dados: *German Credit Database*

Referência:

- Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository http://www.ics.uci.edu/~mlearn/MLRepository.html . Irvine, CA: University of California, School of Information and Computer Science. Source of German Credit Data.
- Gayler, R (2008) Credit Risks Analystics Occasional newsletter. Retrieved from http://r.gayler.googlepages.com/CRAON01.pdf
- Guide to Credit Scoring in R – CRAN https://www.google.com.br/search?q=Guide+to+Credit+Scoring+in+R&ie=utf-8&oe=utf-8&client=firefox-b-ab&gfe_rd=cr&ei=jVFYV8z6J--p8weckLsQ
- Analysis of German Credit Data - The Pennsylvania State University - STAT 897D - Applied Data Mining and Statistical Learning

Exercício 01

Objetivo:

Construir uma árvore de decisão para classificação nominal de crédito dos registros da base *German Credit Data* e avaliar a performance do modelo utilizando *confusion matrix*

Packages: `rpart,caret,e1071`

1. Carregar os dados: `R/database/german_credit_2.csv`

2. Fazer a análise exploratória dos dados : tipos das variáveis, histogramas, medidas decritivas, tabelas de contingencia, etc…

3. Preparar a base para a classificação nominal . Criar as categorias **nominais**:

    Creditability: 1 = “good” e 0 = “bad”

    CreditAmount: '0-2500‘; '2500-5000‘; '5000+‘

4. Separar **aleatoriamente** a base em 2 conjuntos:

    `trainData`: Base de Treinamento: 60% observações

    `testData`: Base de Teste: 40% observações

Exercício 01 (continuação)

4. Utilizar a Base de Treinamento para construir uma árvore de decisão para classificação nominal da variável `Creditability` utilizando todas as demais variáveis

```
## Monta a árvore de treinamento por classificação
trainTree = rpart(Creditability~.,data=trainData, method="class");
```

5. Utilizando a árvore construida no item anterior e a função `predict(.)` para prever a classificação nominal os registros da base de test: `testData`

6. Contrastruir a confusion matrix utilizando a função `table(.)` e em seguida com a função `confusionMatrix(.)` do pacote `caret`.

7. Utilizando a árvore construida no item anterior e a função `predict(.)` para prever a classificação probabilistica dos registros da base de test: `testData` e construir a curva `ROC` para avaliar a performance

Exercício 01

```
set.seed(0);
library(rpart);
library(ROCR);
library(caret);

## Carrega a base de dados
data = read.table("../database/german_credit_2.csv",sep=",",header = TRUE, stringsAsFactors = FALSE);

## -- Tratamento dos dados

## Criação da classe nominal
data$Creditability = ifelse(data$Creditability==1,"good","bad");

## Categorização do 'Credit Amount'
data$CreditAmount = (ifelse(data$CreditAmount<=2500,'0-2500',ifelse(data$CreditAmount<=5000,'2600-5000','5000+')))

## -- Separação da base aleatória em base de treinamento e test

## Base de treinamento e base de teste
ids = sort(sample(nrow(data),nrow(data)*0.6));

## Train dataset
trainData = data[ids,];

## Test dataset
testData = data[-ids,]
```

Atenção: No data-frame cada coluna comporta um único tipo de dado. Erro ao comparar se fizer conversão parcial!

Exercício 01

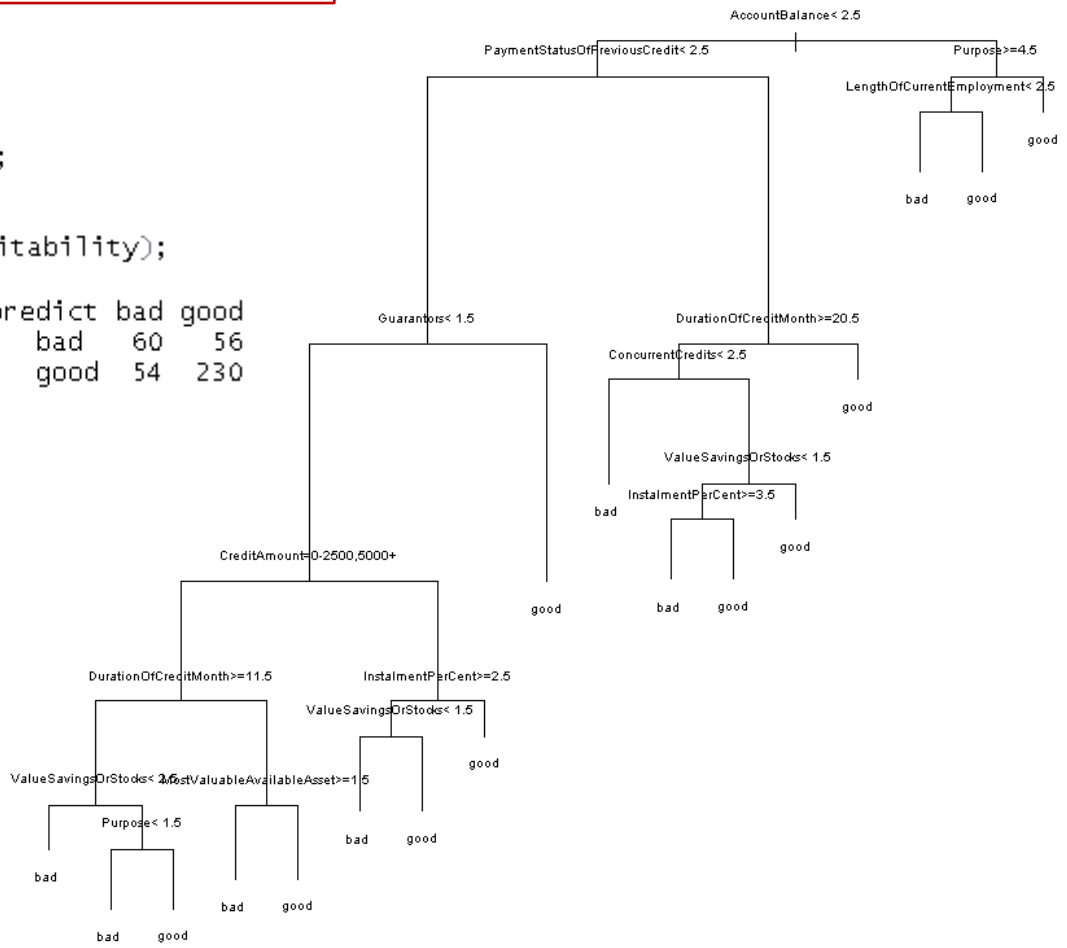
```
## Contrução do modelo

## Monta a árvore de treinamento por classificação
trainTree = rpart(Creditability~., data=trainData, method="class");

## Plota a árvore de classificação
summary(trainTree);
plot(trainTree);
text(trainTree, pretty=0 ,cex=0.6)

## Faz a previsão
testTree.predict = predict(trainTree,testD

## Confusion Matrix a função table (.)
table(testTree.predict, testData$Creditability);

## Confusion Matrix a função confusionMatrix(.)
confusionMatrix(testTree.predict, testData$Creditability);
```

Classificação nominal

Regressão contra todas as variáveis: Creditability ~.

```
Confusion Matrix and Statistics

          Reference
Prediction bad good
      bad   60   56
      good  54  230

               Accuracy : 0.725
                 95% CI : (0.6784, 0.7682)
    No Information Rate : 0.715
    P-Value [Acc > NIR] : 0.3517

                  Kappa : 0.3288
 Mcnemar's Test P-Value : 0.9240

            Sensitivity : 0.5263
            Specificity : 0.8042
         Pos Pred Value : 0.5172
         Neg Pred Value : 0.8099
             Prevalence : 0.2850
         Detection Rate : 0.1500
   Detection Prevalence : 0.2900
      Balanced Accuracy : 0.6653

       'Positive' Class : bad
```

```
testTree.predict bad good
            bad   60   56
            good  54  230
```

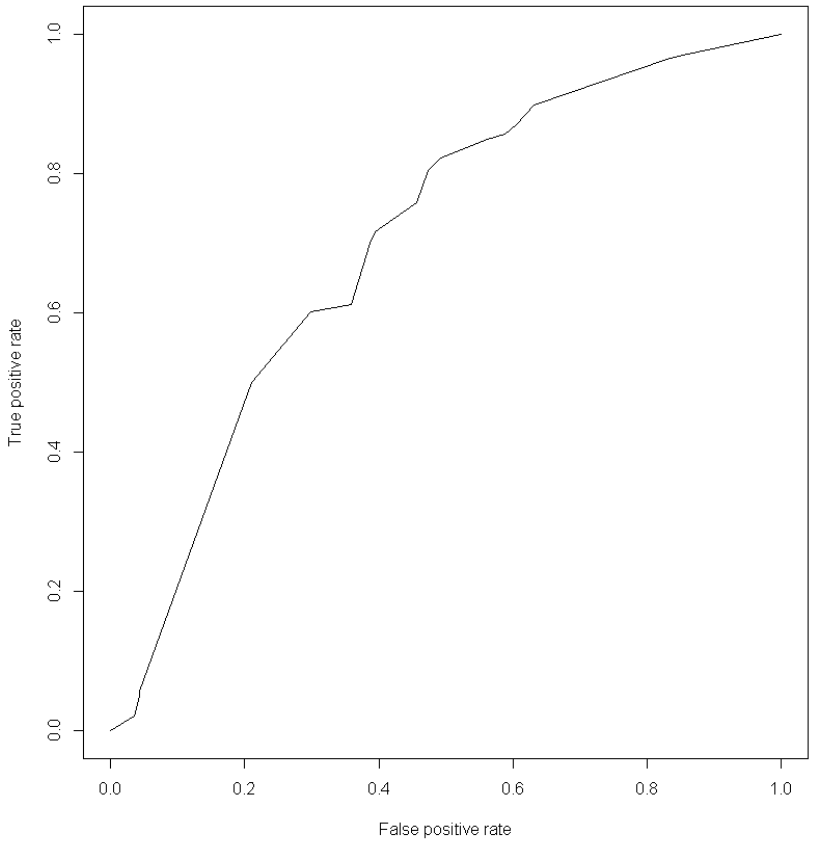


Exercício 01

```
## Previsão com probabilidades
testTree.predict.prob = predict(trainTree,testData,type='prob');

## Confusion Matrix
table(ifelse(testTree.predict.prob[,2]>0.5,"good","bad"), testData$Creditability);

## Faz a previsão por florestas
testTree.prediction = prediction(testTree.predict.prob[,2], testData$Creditability);
testTree.performance = performance(testTree.prediction,"tpr","fpr");
plot(testTree.performance)
```

```
> testTree.predict.prob
           bad       good
3   0.76119403 0.23880597
6   0.17307692 0.82692308
8   0.17307692 0.82692308
9   0.09482759 0.90517241
10  0.20000000 0.80000000
11  0.17307692 0.82692308
13  0.17307692 0.82692308
16  0.31818182 0.68181818
17  0.17307692 0.82692308
18  0.31818182 0.68181818
19  0.66666667 0.33333333
25  0.17307692 0.82692308
26  0.17307692 0.82692308
27  0.76119403 0.23880597
29  0.09482759 0.90517241
33  0.09482759 0.90517241
34  0.09482759 0.90517241
37  0.09482759 0.90517241
38  0.09482759 0.90517241
```

```
      bad good
bad    60   56
good   54  230
```



Confusion Matrix

		Predicted condition		
	Total population	Predicted Condition positive	Predicted Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$
True condition	condition positive	**True positive**	**False Negative** (Type II error)	True positive rate (TPR), Sensitivity, Recall = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$
	condition negative	**False Positive** (Type I error)	**True negative**	False positive rate (FPR), Fall-out = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$
	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	

Exemplo:

```
Confusion Matrix and Statistics

          Reference
Prediction bad good
      bad   60   56
      good  54  230

               Accuracy : 0.725
                 95% CI : (0.6784, 0.7682)
    No Information Rate : 0.715
    P-Value [Acc > NIR] : 0.3517

                  Kappa : 0.3288
 Mcnemar's Test P-Value : 0.9240

            Sensitivity : 0.5263
            Specificity : 0.8042
         Pos Pred Value : 0.5172
         Neg Pred Value : 0.8099
             Prevalence : 0.2850
         Detection Rate : 0.1500
   Detection Prevalence : 0.2900
      Balanced Accuracy : 0.6653

       'Positive' Class : bad
```

Ref: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Receiver Operating Characteristic (ROC)

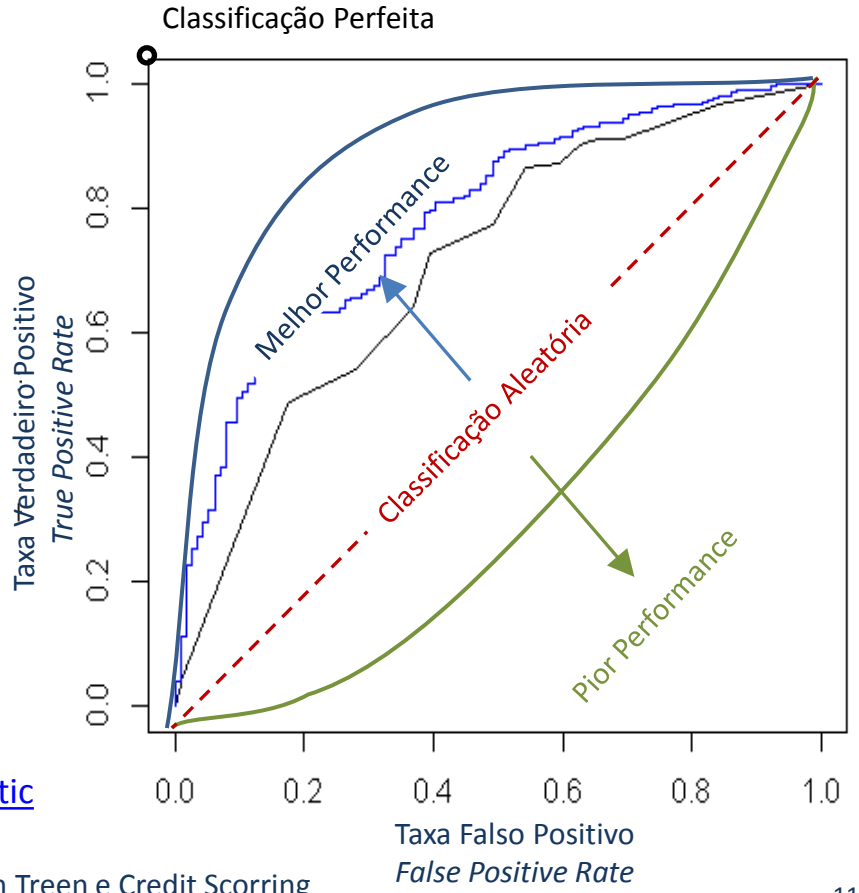
- A Receiver Operation Characteristic (ROC) é uma representação gráfica da performance de classificadores binários construido a partir da variação do critério de aceitação ou corte (*threshold*).

- Para se construir uma ROC marcamos no plano cartesiano X x Y os pares (x,y) que representam para cada cada nível do critério de aceite a Taxa de Falso Positivo (eixo x) e a Taxa de Verdadeiro Positivo (eixo y)

- A área do gráfico é 1.0 *x* 1.0 = 1.0.

A área abaixo da curva é uma medida da performace o classificador:

Área	Avaliação
Igual 1.0	Perfeito
mais próximo a 1.0	Melhor Performance
Igual 0.5	Aleatório
mais distante de 1.0	Pior Performance



Ref: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Exercício 01 - Observação

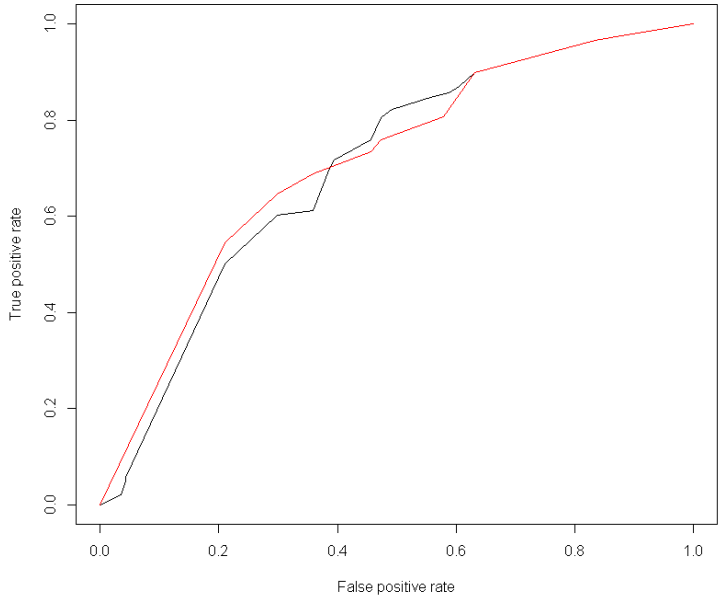
- A inclusão de mais variáves ao modelo não garante que serão obtidos melhores resultados.

- Normalmente quando as variáveis do modelo estão muito correlacionadas os modelos obtidos são sub-ótimos.

- No exercício a construção da árvore de decisão poderia ser refeito incluindo apenas um subconjunto das variáveis.

```
## Monta a árvore de treinamento por classificação
trainTree = rpart(Creditability~AccountBalance+DurationOfCreditMonth+PaymentStatusOfPreviousCredit+
                                CreditAmount+ValueSavingsOrStocks+Guarantors+DurationInCurrentAddress+
                                Age+ConcurrentCredits+Occupation+NoOfDependents,
                                data=trainData, method="class");
```

```
Confusion Matrix and Statistics

          Reference
Prediction bad good
      bad   42   29
      good  72  257

               Accuracy : 0.7475
                 95% CI : (0.7019, 0.7894)
    No Information Rate : 0.715
    P-Value [Acc > NIR] : 0.08189
```



Exercício 02 – *Decison Tree and Random Forest*

Objetivo:

Construir 2 classificadores crédito **modelo de regressão** :

1) Árvore de decisão

2) Random Forest (500 árvores aleatórias)

Comparar a performance dos classificadores com a base *German Credit Data.*

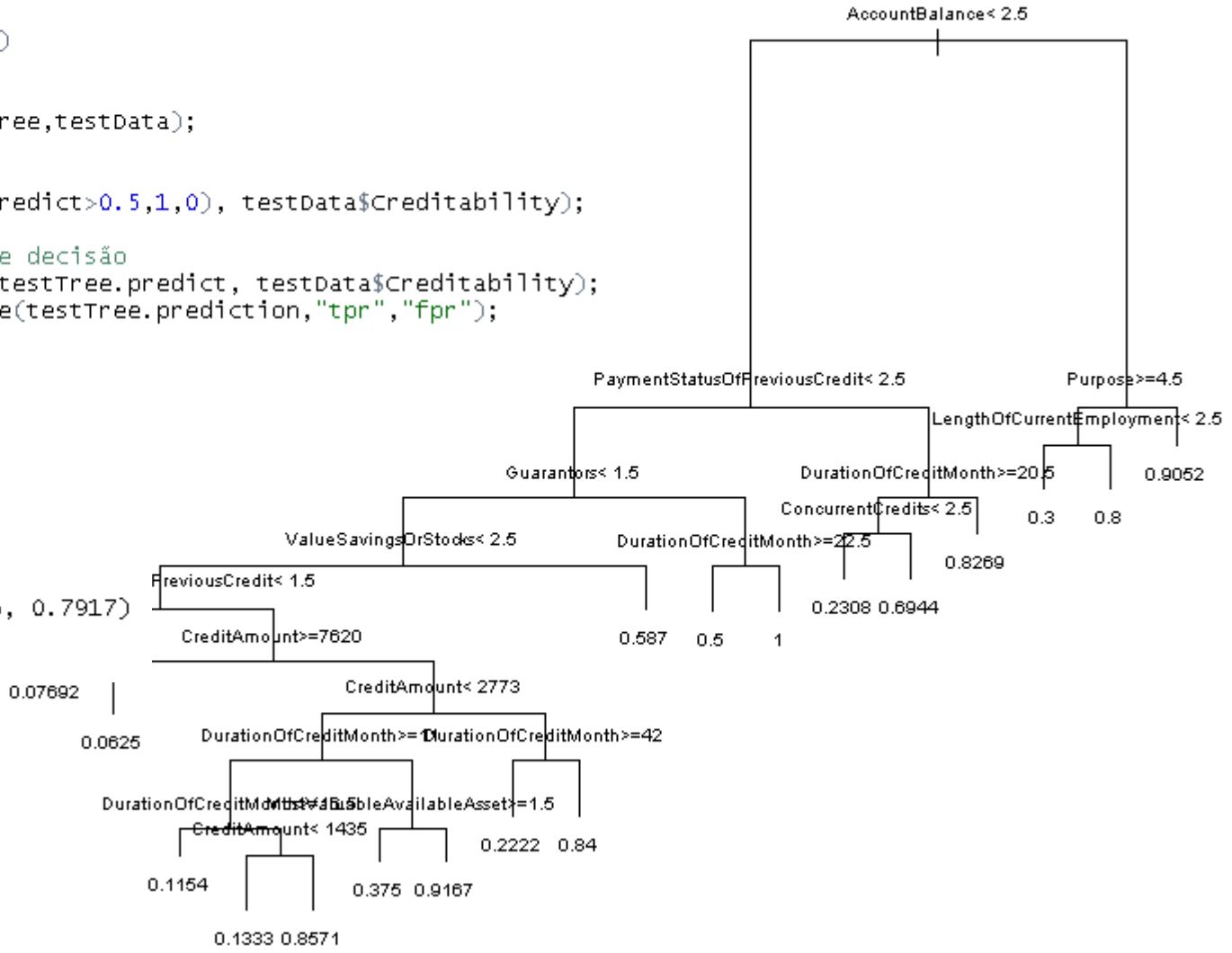
Package: `rpart,ROCR,randomForest`

1. Carregar os dados: `R/database/german_credit_2.csv`
2. Separar **aleatoriamente** a base em 2 conjuntos:
   - `trainData`: Base de Treinamento: 60% observações
   - `testData`: Base de Teste: 40% observações
3. Utilizar a Base de Treinamento para construir os classificadores da variável `Creditability` utilizando todas as demais variáveis
4. Fazer a previsão para a base de teste e comparar a performance dos modelos utilizando a curva ROC.

Exercício 02

```
## Contrução do modelo de árvore de descisão

## Monta a árvore de treinamento por classificação
trainTree = rpart(Creditability~., data=trainData, method = "anova");

## Plota a árvore de classificação
plot(trainTree);
text(trainTree, pretty=0 ,cex=0.6)

## Faz a previsão
testTree.predict = predict(trainTree,testData);

## Confusion Matrix
confusionMatrix(ifelse(testTree.predict>0.5,1,0), testData$Creditability);

## Faz a previsão para a árvore de decisão
testTree.prediction = prediction(testTree.predict, testData$Creditability);
testTree.performance = performance(testTree.prediction,"tpr","fpr");
plot(testTree.performance)
```

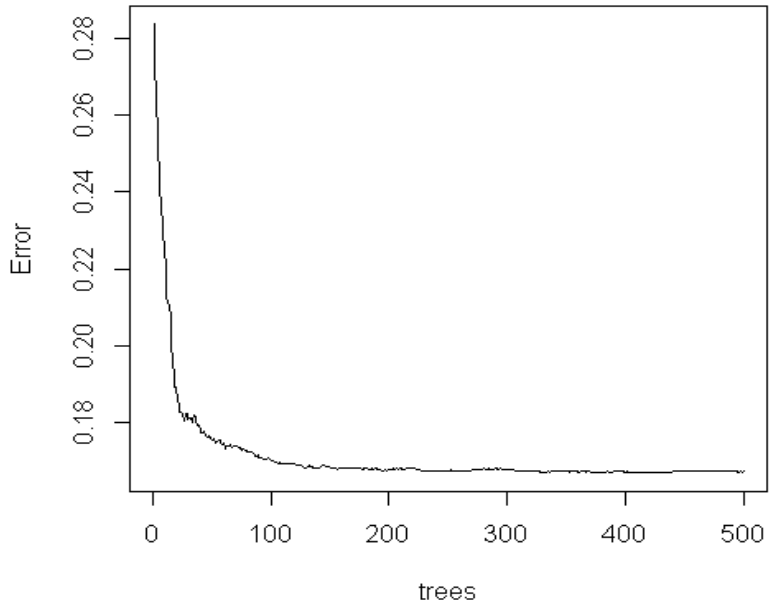
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  52  38
         1  62 248

               Accuracy : 0.75
                 95% CI : (0.7046, 0.7917)
    No Information Rate : 0.715
    P-Value [Acc > NIR] : 0.06604
```



Exercício 02

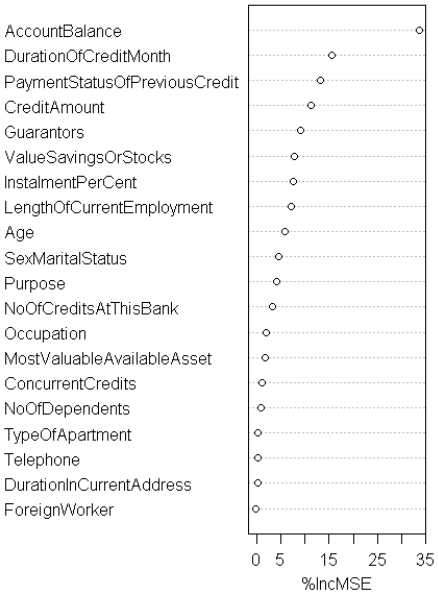
```
## Monta o Modelo de RandomForest

## Monta Floresta a partir da base de treinamento
trainForest = randomForest(Creditability~., data=trainData, importance=TRUE, proximity=TRUE,
                           ntree=500, keep.forest=TRUE);

## Plota: num.árvores vs error
plot(trainForest);

## Gráfico de importância das variávesi
varImpPlot(trainForest);

## Faz a previsão e indica a probabilidade
testForest.predict = predict(trainForest,testData);

## Confusion Matrix
confusionMatrix(ifelse(testForest.predict>0.5,1,0), testData$Creditability);

## Faz a previsão por florestas
testForest.prediction = prediction(testForest.predict, testData$Creditability);
testForest.performance = performance(testForest.prediction,"tpr","fpr");
plot(testTree.performance)
plot(testForest.performance,col="Blue",add=TRUE)
```

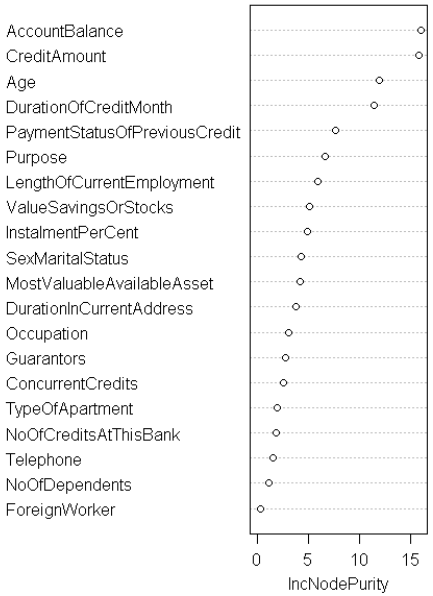
```
      Confusion Matrix and Statistics

                Reference
      Prediction   0   1
               0  57  35
               1  57 251

                     Accuracy : 0.77
                       95% CI : (0.7256, 0.8104)
          No Information Rate : 0.715
          P-Value [Acc > NIR] : 0.007691
```



Exercício 02

Gráfico que expressa a importância de cada variável





Comparativo de Performance