Risco de Crédito: Como uma Melhor Seleção de Variáveis Pode Ampliar a Eficácia das Previsões

Rafael Bicudo Rosa

24 de julho de 2018

Como a seleção de variáveis pode aumentar a eficácia de um modelo

O objetivo deste trabalho é usar dados sobre análises de crédito realizadas na Alemanha, executar uma análise exploratória, construir uma série de modelos de classificação baseados nessa, observar como a melhor seleção de variaveis afeta a performance e escolher o mais preciso.

Modelos de aprendizado de máquina sobre condições de pagamento são de importância fundamental a qualquer negócio do setor financeiro. Embora nunca sejam perfeitos, podem poupar horas de trabalho, sob custo muito baixo, ao pré-selecionar possíveis maus pagadores para uma análise mais rebuscada de forma muito eficiente.

Os dados incluem 1000 observações de concessão de crédito, cada uma com 21 variáveis, sendo a última a classificação do solicitante (bom ou mau pagador), e as restantes características qualitativas e quantitativas sobre esses mesmos. Todas as informacões foram retiradas do repositório online da Universidade de Irvine, Califórnia (https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data), assim como uma melhor explicação do significado das variáveis.

Etapa 1 - Coleta dos Dados

Assim como descrito acima, os dados serão retirados de um repositório online contendo a base em si no formato table, e as informações de cada uma das características. Em seguida, as variáveis serão nomeadas e, por fim, ter-se-á a primeira visão do dataframe.

```
## Obtencao dos dados

# Carrega o dataset antes da transformacao
german_credit_l <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data
Credit <- read.table(german_credit_l)

# Nome das variaveis
names(Credit) <- c('CheckingAcctStat', 'Duration', 'CreditHistory', 'Purpose', 'CreditAmount',
                   'SavingsBonds', 'Employment', 'InstallmentRatePecnt', 'SexAndStatus',
                   'OtherDetorsGuarantors', 'PresentResidenceTime', 'Property', 'Age',
                   'OtherInstallments', 'Housing', 'ExistingCreditsAtBank', 'Job', 'NumberDependents',
                   'Telephone', 'ForeignWorker', 'CreditStatus')

# Analise do dataframe
str(Credit)
```

```
## 'data.frame':    1000 obs. of  21 variables:
##  $ CheckingAcctStat     : Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1 4 4 2 4 2 ...
##  $ Duration             : int  6 48 12 42 24 36 24 36 12 30 ...
##  $ CreditHistory        : Factor w/ 5 levels "A30","A31","A32",..: 5 3 5 3 4 3 3 3 3 5 ...
##  $ Purpose              : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4 1 8 4 2 5 1 ...
##  $ CreditAmount         : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ SavingsBonds         : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1 5 3 1 4 1 ...
##  $ Employment           : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3 3 5 3 4 1 ...
```

```
##  $ InstallmentRatePecnt : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ SexAndStatus         : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3 3 3 3 1 4 ...
##  $ OtherDetorsGuarantors: Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1 1 1 1 ...
##  $ PresentResidenceTime : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ Property             : Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2 3 1 3 ...
##  $ Age                  : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ OtherInstallments    : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ Housing              : Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2 1 2 2 ...
##  $ ExistingCreditsAtBank: int  2 1 1 1 2 1 1 1 1 2 ...
##  $ Job                  : Factor w/ 4 levels "A171","A172",..: 3 3 2 3 3 2 3 4 2 4 ...
##  $ NumberDependents     : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ Telephone            : Factor w/ 2 levels "A191","A192": 2 1 1 1 1 2 1 2 1 1 ...
##  $ ForeignWorker        : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CreditStatus         : int  1 2 1 1 2 1 1 1 1 2 ...
```

```
summary(Credit)
```

```
##  CheckingAcctStat    Duration    CreditHistory    Purpose
##  A11:274          Min.   : 4.0   A30: 40       A43    :280
##  A12:269          1st Qu.:12.0   A31: 49       A40    :234
##  A13: 63          Median :18.0   A32:530       A42    :181
##  A14:394          Mean   :20.9   A33: 88       A41    :103
##                   3rd Qu.:24.0   A34:293       A49    : 97
##                   Max.   :72.0                 A46    : 50
##                                                (Other): 55
##   CreditAmount   SavingsBonds Employment InstallmentRatePecnt SexAndStatus
##  Min.   :  250   A61:603      A71: 62    Min.   :1.000        A91: 50
##  1st Qu.: 1366   A62:103      A72:172    1st Qu.:2.000        A92:310
##  Median : 2320   A63: 63      A73:339    Median :3.000        A93:548
##  Mean   : 3271   A64: 48      A74:174    Mean   :2.973        A94: 92
##  3rd Qu.: 3972   A65:183      A75:253    3rd Qu.:4.000
##  Max.   :18424                           Max.   :4.000
##
##  OtherDetorsGuarantors PresentResidenceTime Property        Age
##  A101:907              Min.   :1.000        A121:282   Min.   :19.00
##  A102: 41              1st Qu.:2.000        A122:232   1st Qu.:27.00
##  A103: 52              Median :3.000        A123:332   Median :33.00
##                        Mean   :2.845        A124:154   Mean   :35.55
##                        3rd Qu.:4.000                   3rd Qu.:42.00
##                        Max.   :4.000                   Max.   :75.00
##
##  OtherInstallments Housing    ExistingCreditsAtBank   Job
##  A141:139          A151:179   Min.   :1.000         A171: 22
##  A142: 47          A152:713   1st Qu.:1.000         A172:200
##  A143:814          A153:108   Median :1.000         A173:630
##                               Mean   :1.407         A174:148
##                               3rd Qu.:2.000
##                               Max.   :4.000
##
##  NumberDependents Telephone  ForeignWorker  CreditStatus
##  Min.   :1.000    A191:596   A201:963      Min.   :1.0
##  1st Qu.:1.000    A192:404   A202: 37      1st Qu.:1.0
##  Median :1.000                             Median :1.0
##  Mean   :1.155                             Mean   :1.3
##  3rd Qu.:1.000                             3rd Qu.:2.0
```

```
##  Max.   :2.000                              Max.   :2.0
##
```

Etapa 2 - Limpeza e Preparação dos dados

A partir do demonstrado acima, vê-se a existência de alguns pontos de atenção: a diferenca de grandezas entre as variáveis quantitativas, e algumas variáveis qualitativas como numéricas. Ambos podem levar a viéses ruins ou inconsistências em modelos preditivos, portanto se segue a uma etapa de ajustamento dos dados.

```
## Data Cleaning

# Definicao variavel de interesse
Credit[, 'CreditStatus'] <- factor(Credit[, 'CreditStatus'], labels = c('Good', 'Bad'))

# Funcao para automatizar "fatorizacao" das variaveis
to.factor <- function(df, features) {
  for (feature in features) {
    df[[feature]] <- as.factor(df[[feature]])
  }
  return(df)
}

# Criacao do string vector das variaveis a  serem fatorizadas e sua fatorizacao
categorical_vars <- c('CheckingAcctStat', 'CreditHistory', 'Purpose',
                      'SavingsBonds', 'Employment', 'InstallmentRatePecnt', 'SexAndStatus',
                      'OtherDetorsGuarantors', 'PresentResidenceTime', 'Property',
                      'OtherInstallments', 'Housing', 'ExistingCreditsAtBank', 'Job', 'NumberDependents'
                      'Telephone', 'ForeignWorker', 'CreditStatus')

Credit <- to.factor(Credit, categorical_vars)

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

# Normalizacao das variaveis
numeric_vars <- c("Duration", "Age", "CreditAmount")
scale_Credit <- scale.features(Credit, numeric_vars)
```

Etapa 3 - Dividindo os dados em treino e teste

Com a preparacao dos dados concluída, pode-se prosseguir a separação dos dados entre treino, para modelagem e exploração, e teste, para validação da aprendizagem.

```
# Carregando pacotes necessarios
library(caret)
```

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

```
## Loading required package: ggplot2
```

```
# Separacao dos Sets de Treino e Teste
set.seed(666)
sample <- createDataPartition(scale_Credit$CreditStatus, times = 1, list = F, p = .6)
train_sample <- scale_Credit[sample, ]
test_sample <- scale_Credit[-sample, ]
```

Etapa 4 - Seleção das variáveis

Com todas as transformações concluídas, segue-se para a seleção das variáveis mais explicativas. Para executar a tarefa, foi criada uma função para aplicar o método de seleção recursiva usando "Randon Forests", através do uso do pacote de Machine Learning Caret (usado ao longo de todo trabalho inclusive).

O método consiste em iterações de vários modelos, a começar pelo pleno, testando várias combinações retirando-se algumas variáveis, e comparando seus poderes explicativos. Por sua vez, o algoritmo de referência para o processo é um dos modelos de classificação mais eficientes da atualidade. (Para mais informacoes sobre os processos disponiveis, checar a documentacao do pacote Caret: http://topepo.github.io/caret/recursive-feature-elimination.html#backwards-selection)

```
## Feature Selection

# Carregando pacotes necessarios
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
```

```
library(ggplot2)

# Funcao para selecao das variaveis
rfe.feature.selection <- function(num_iters=20, features, target){
  variable_sizes <- 1:10
  control <- rfeControl(functions = rfFuncs, method = "cv",
                        verbose = FALSE, returnResamp = "all",
                        number = num_iters)
  rfe_results <- rfe(x = features, y = target,
                     sizes = variable_sizes,
                     rfeControl = control)
  return(rfe_results)
}

# Executando a funcao e para obter features mais explicativas
rfe_results <- rfe.feature.selection(features = train_sample[,-21],
                                     target = train_sample[,21])

# Selecao das Features mais significates e visualizacao da significancia
rfe_results
```

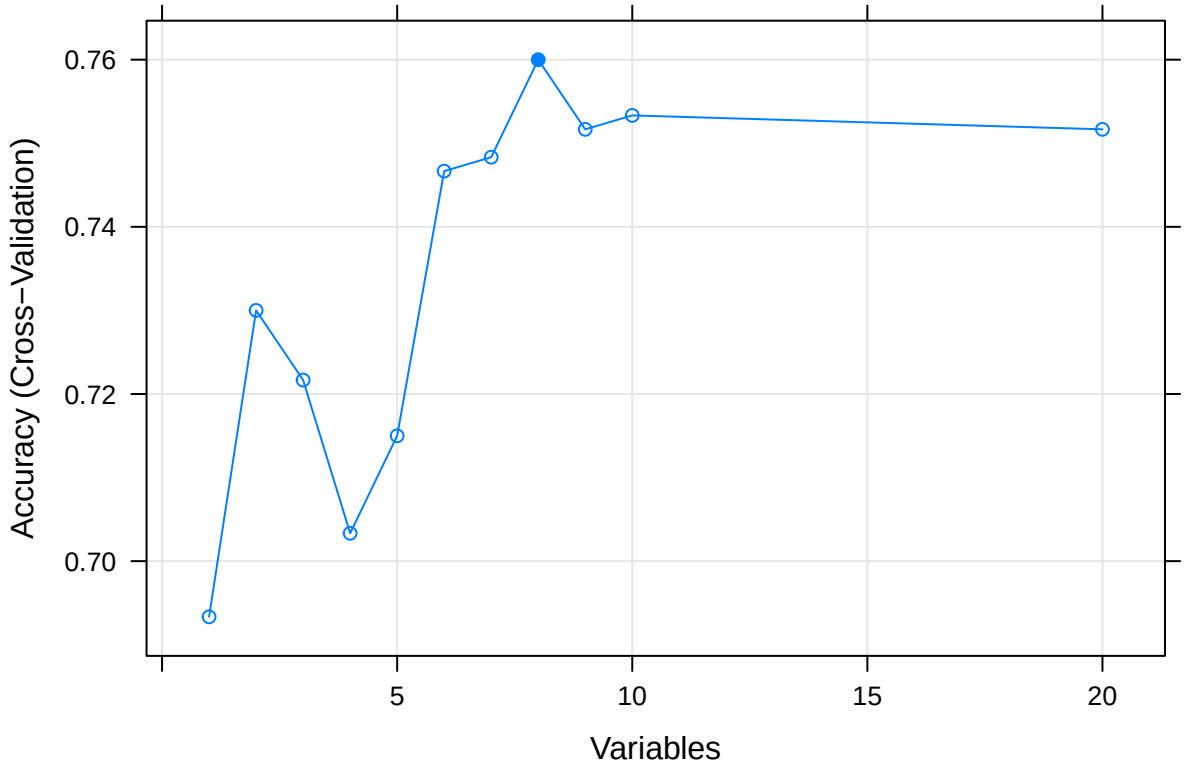
```
##
```

```
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (20 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.6933 0.2085    0.06719  0.2025
##          2   0.7300 0.2851    0.06389  0.2099
##          3   0.7217 0.2487    0.05437  0.1783
##          4   0.7033 0.2458    0.06831  0.1745
##          5   0.7150 0.2617    0.06163  0.1606
##          6   0.7467 0.3292    0.07446  0.2144
##          7   0.7483 0.3281    0.07452  0.2029
##          8   0.7600 0.3570    0.07383  0.2161        *
##          9   0.7517 0.3576    0.07452  0.1992
##         10   0.7533 0.3546    0.07829  0.2145
##         20   0.7517 0.3218    0.06965  0.1956
##
## The top 5 variables (out of 8):
##    CheckingAcctStat, Duration, CreditHistory, CreditAmount, SavingsBonds
```

```
varImp((rfe_results), scale = F)
```

```
##                        Overall
## CheckingAcctStat      20.486732
## Duration              10.622675
## CreditHistory          6.822450
## CreditAmount           6.802129
## SavingsBonds           6.477065
## OtherDetorsGuarantors  5.746643
## Purpose                5.509115
## OtherInstallments      4.806770
## Property               4.770802
## Employment             4.769671
```

```
optVariables <- rfe_results[["optVariables"]]
plot(rfe_results, type=c("g", "o"))
```

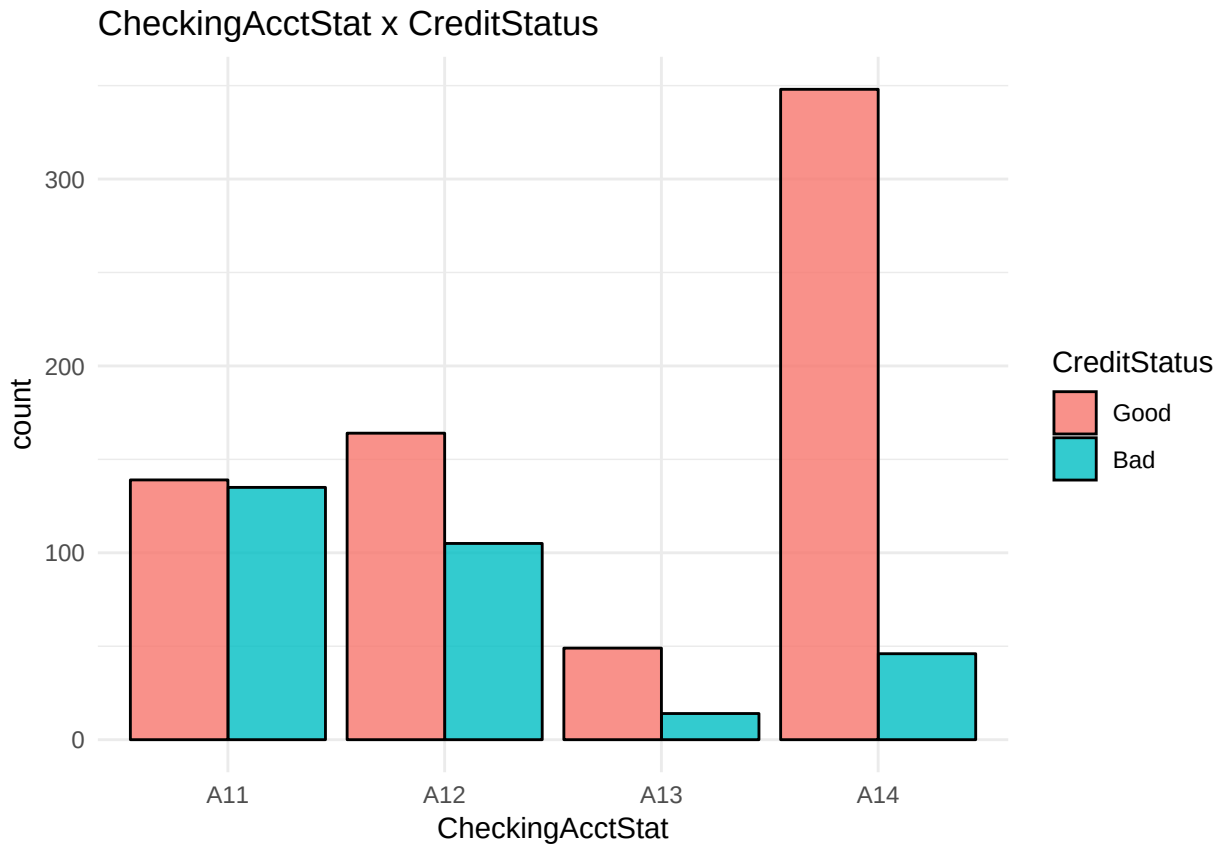



```
optVariables
```

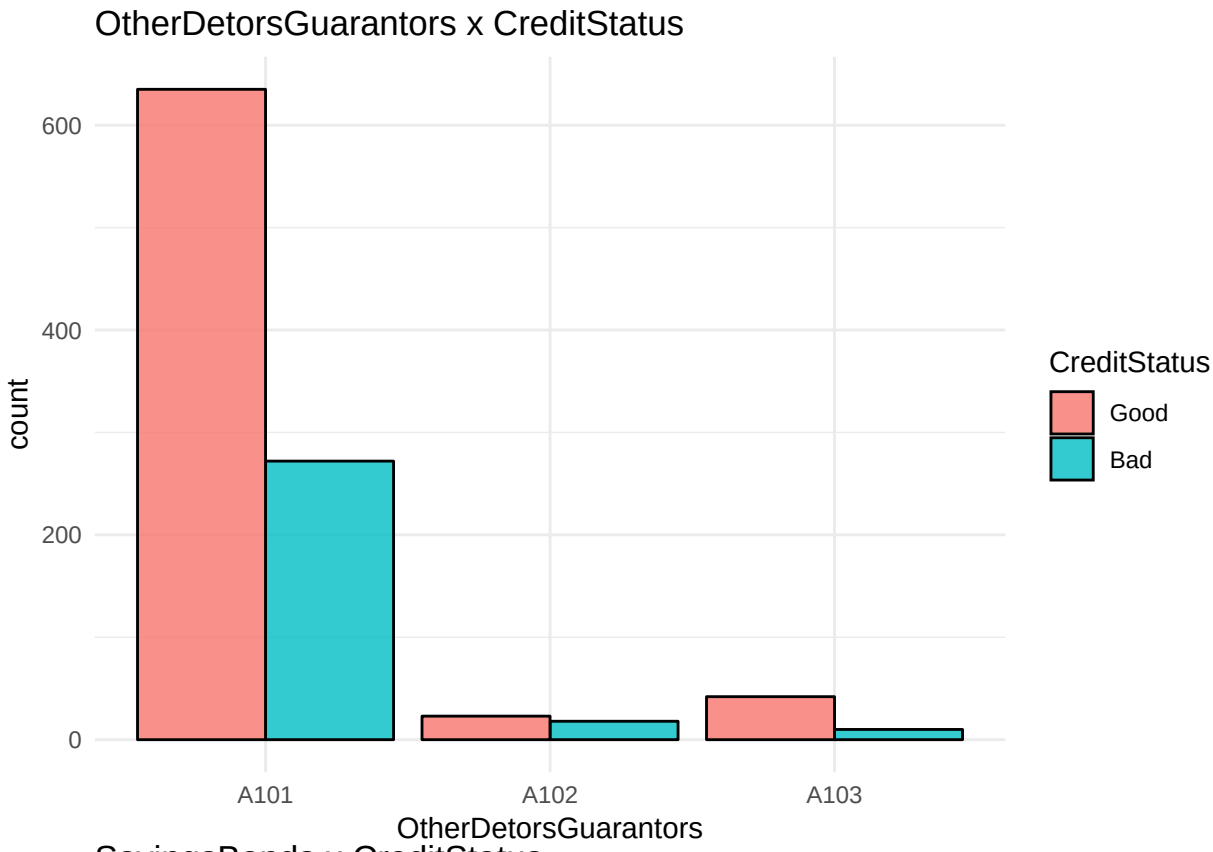
```
## [1] "CheckingAcctStat"      "Duration"              "CreditHistory"
## [4] "CreditAmount"          "SavingsBonds"          "OtherDetorsGuarantors"
## [7] "Purpose"               "Employment"
```

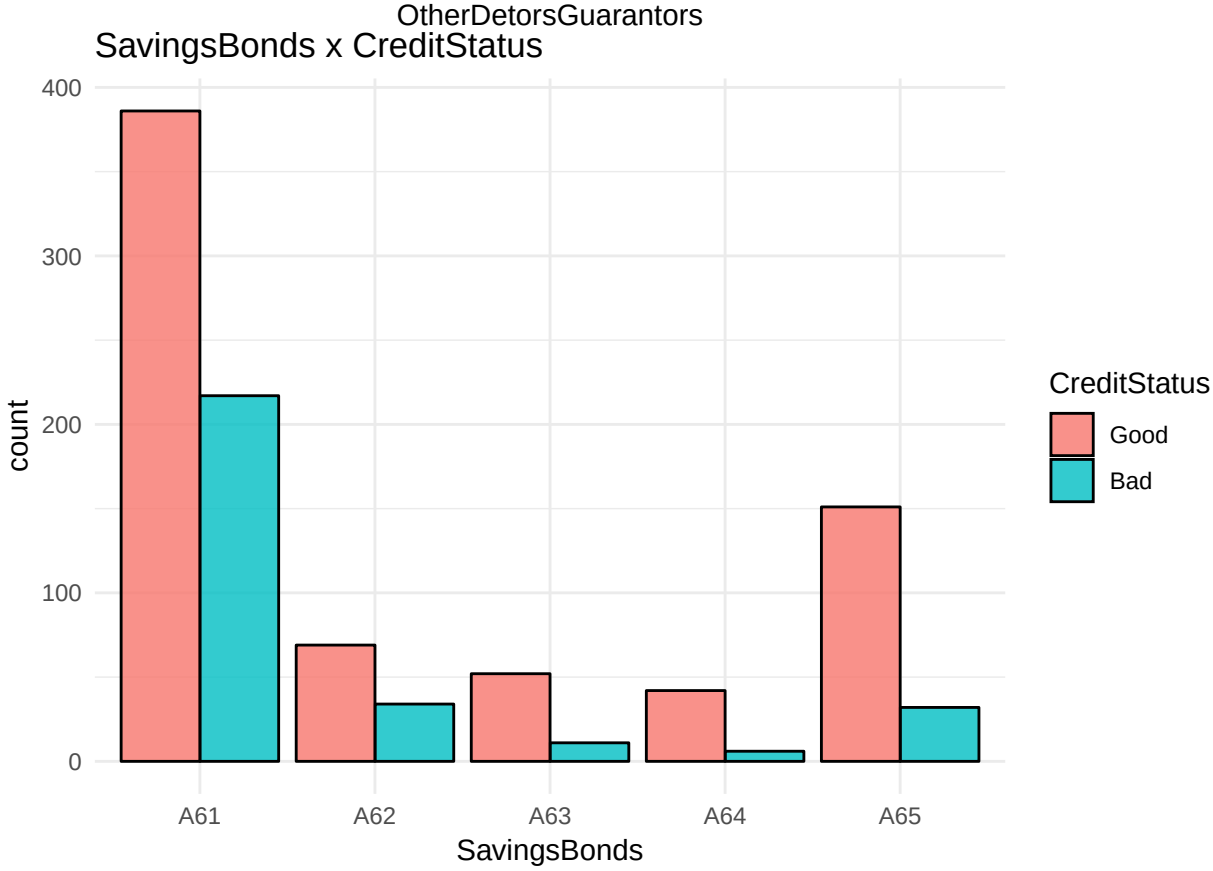
O gráfico acima demonstra como o poder explicativo do modelo varia através da inclusão de mais características até chegar ao seu número ótimo. Para ilustrar melhor seu poder explicativo na prática, seguem, abaixo, gráficos entre as váriveis explicativas citadas e nosso alvo.

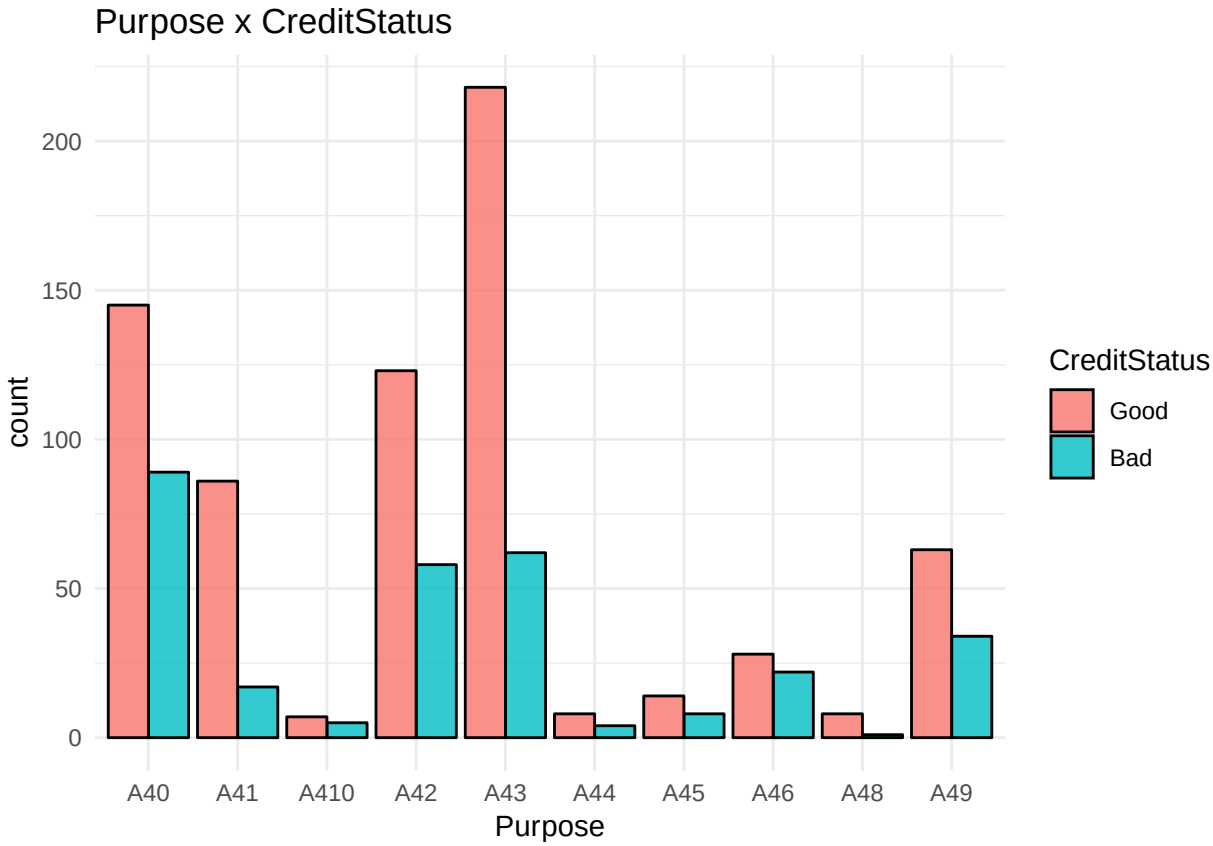
```
# Plot das variaveis otimas

# Categoricas
plots_cat<- list()
for (i in c('CheckingAcctStat', 'CreditHistory','OtherDetorsGuarantors', 'SavingsBonds', 'Purpose',
            'Employment')) {
  plots_cat[[i]] <- ggplot(Credit, aes_string(x = i, fill = 'CreditStatus')) +
  geom_bar(alpha=0.8, colour='black', position = 'dodge') + ggtitle(paste(i, 'x CreditStatus')) +
  theme_minimal()
  print(plots_cat[[i]])
}
```

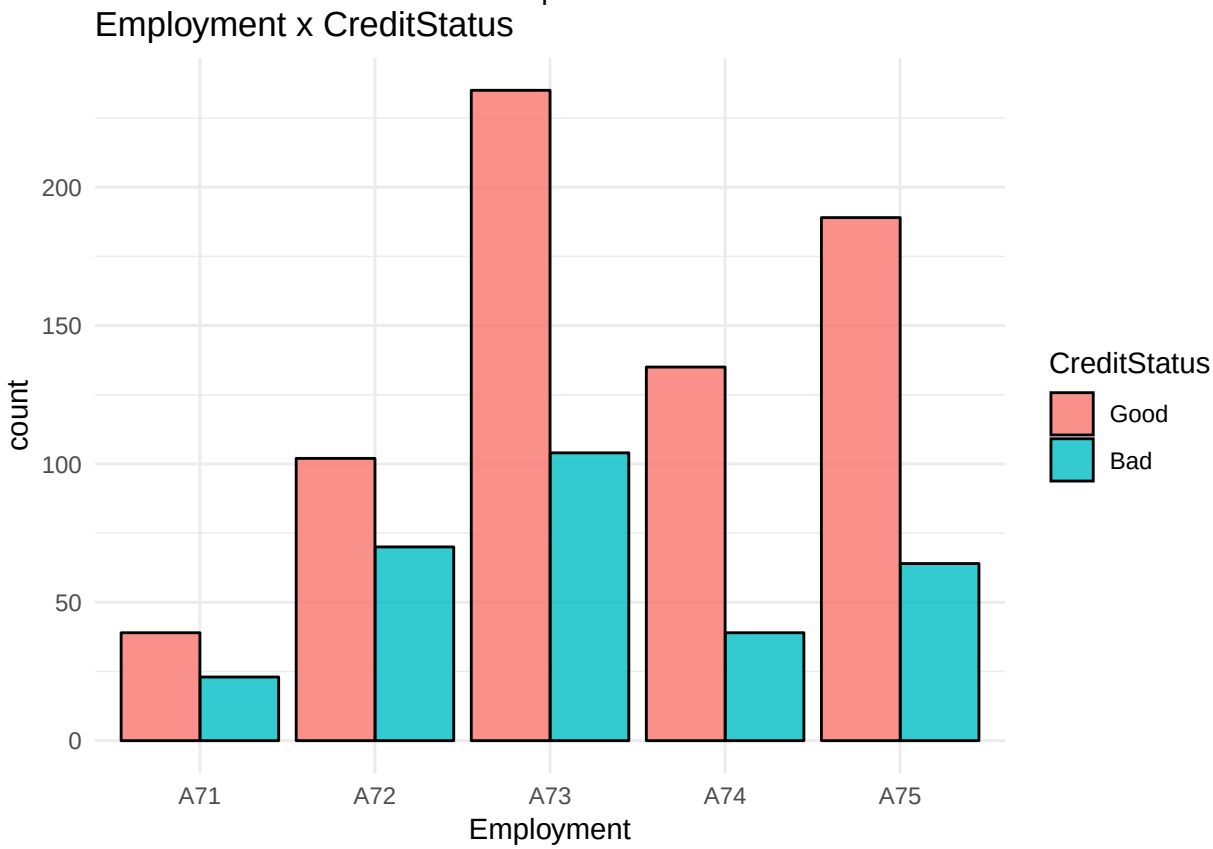

CheckingAcctStat x CreditStatus
count
300
200
100
0
A11
A12
A13
A14
CheckingAcctStat
CreditStatus
Good
Bad


CreditHistory x CreditStatus
count
300
200
100
0
A30
A31
A32
A33
A34
CreditHistory
CreditStatus
Good
Bad


OtherDetorsGuarantors x CreditStatus
600
400
200
0
count
A101
A102
A103
OtherDetorsGuarantors
CreditStatus
Good
Bad


SavingsBonds x CreditStatus
400
300
200
100
0
count
A61
A62
A63
A64
A65
SavingsBonds
CreditStatus
Good
Bad


Purpose x CreditStatus
count
200
150
100
50
0
A40
A41
A410
A42
A43
A44
A45
A46
A48
A49
Purpose
CreditStatus
Good
Bad


Employment x CreditStatus
count
200
150
100
50
0
A71
A72
A73
A74
A75
Employment
CreditStatus
Good
Bad

```r
# Continua (Duration, CreditAmount, Age)
plots_cont<- list()
for (i in c('Duration', 'CreditAmount')) {
  plots_cont[[i]] <- ggplot(Credit, aes_string(x = 'CreditStatus', y = i, fill = 'CreditStatus')) +
    geom_boxplot(alpha=0.8, colour='black', position = 'dodge') + ggtitle(paste(i, 'x CreditStatus')) +
    theme_classic()
  print(plots_cont[[i]])
}
```

Etapa 5 - Criação do Modelo de Referência

Após terminar a análise exploratória, prossegue-se à criação do modelo base: contendo todas as variáveis e sem nenhuma transformação, será usado como referência para os próximos.

```
## Regressao Classica
SimpleModel <- train(CreditStatus ~ ., data = train_sample,
                     family = binomial(),
                     method = 'glm',
                     trControl = trainControl(method = 'none'))

# Avaliando o modelo
SimpleModel_pred <-predict(SimpleModel, test_sample)
confusionMatrix(SimpleModel_pred, test_sample$CreditStatus)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Good Bad
##       Good  230  64
##       Bad    50  56
##
##                Accuracy : 0.715
##                  95% CI : (0.668, 0.7588)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.2758
##
```

```
##                   Kappa : 0.298
##  Mcnemar's Test P-Value : 0.2234
##
##             Sensitivity : 0.8214
##             Specificity : 0.4667
##          Pos Pred Value : 0.7823
##          Neg Pred Value : 0.5283
##              Prevalence : 0.7000
##          Detection Rate : 0.5750
##    Detection Prevalence : 0.7350
##       Balanced Accuracy : 0.6440
##
##        'Positive' Class : Good
##
```

Etapa 6 - Execução dos Modelos

Por fim, chega-se à motivação do trabalho: selecionar o modelo com o melhor poder de classificar bons e maus pagadores. Como a amostra de treino não é tão grande, usam-se validações cruzadas repetidas (geradas a partir de bootstrap) para aumentá-la artificialmente. Só serão consideradas as características com maior poder explicativo selecionadas anteriormente para garantir a máxima eficiência. A métrica escolhida foi 'ROC', pois, como o problema é de classificação, ela é mais indicada por usar a taxa entre falsos e verdadeiros positivos.

```
# Melhores Variaveis
FS_formula = as.formula(CreditStatus ~ CheckingAcctStat + Duration + OtherDetorsGuarantors +CreditAmoun
  CreditHistory + SavingsBonds + Purpose + Employment)

# Controle dos modelos de treino
train_control <- trainControl(method="repeatedcv",
                              number=10,
                              repeats=3,
                              returnResamp = "all",
                              classProbs = TRUE,
                              summaryFunction = twoClassSummary,
                              verboseIter = TRUE,
                              savePredictions = TRUE)

## Modelos com tratamento

# Glm com melhores variaveis
FSModel <- train(FS_formula,
                                data = train_sample,
                                method = 'glm',
                                metric = "ROC",
                                trControl = train_control)

## Glmnet

# Grid para escolher melhores hiper parametros
tuning_grid_glm <-  expand.grid(alpha =  c(0, 0.25, 0.75, 1), # indica o peso (0 <- 1) entre L1 e L2
                                lambda = seq(0.0001, 1, length = 5)) # indica o tamanho da penalidade

# Modelo
```

```
GlmnetModel <- train(FS_formula,
                     data = train_sample,
                     metric = "ROC",
                     method = "glmnet",
                     trControl = train_control,
                     tuneGrid = tuning_grid_glm)

## Random Forest

# Grid para escolher melhores hiper parametros (no caso, so mtry - resto e default)
tuning_grid_rf <- expand.grid(.mtry = c(2, 4, 6, 8),.splitrule = "gini", .min.node.size = 1)

# Modelo
RfModel <- train(FS_formula,
                 data = train_sample,
                 metric = "ROC",
                 method = "ranger",
                 trControl = train_control,
                 tuneGrid = tuning_grid_rf)

## KNN

# Grid para escolher melhores hiper parametros (no caso, apenas número de vizinhos - k)
tuning_grid_knn <- expand.grid(.k = 1:25)

# Modelo
KnnModel <- train(FS_formula,
                  data = train_sample,
                  metric = "ROC",
                  method = "knn",
                  trControl = train_control,
                  tuneGrid = tuning_grid_knn)
```

Assim como visto acima, foram selecionados 4 algorítmos de classificação muito utilizados: GLM, GLMNET, RF e KNN. O primeiro é o mais simples, sendo o mesmo utilizado para controle; o segundo é uma versão mais sofisticada deste que utiliza as penalidade conhecidas como L1 e L2; o terceiro é o já mencionado "Random Forests"; e o quarto é um "K-Nearest Neighbors", que se baseia em observações similares para se chegar a uma classificação. Nos 3 últimos, por serem mais complexos, pode-se executar a escolha de hiper-parâmetros com o intuito de melhorar ainda mais o desempenho. Para otimizar essa opção, usa-se o recurso "expand.grid" que compara diferentes combinações deles.

Etapa 7 - Análise dos Resultados

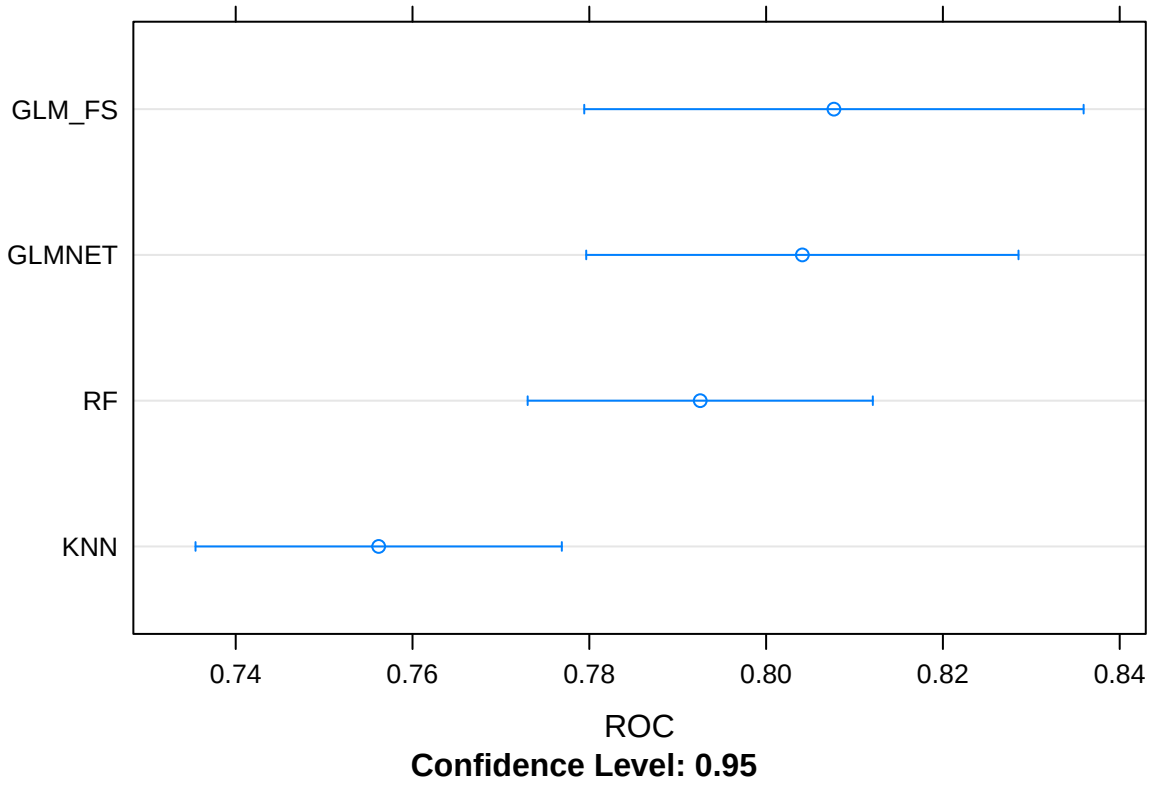
Após o ajustamento dos modelos, faz-se uma comparação da sua métrica utilizada para descobrir qual possui a melhor performance no treino.

```
# Lista com os modelos
model_list <- list(GLM_FS = FSModel, GLMNET = GlmnetModel, RF = RfModel, KNN = KnnModel)

# Função resamples para compará-los
resamples <- resamples(model_list)
```

```
## Warning in resamples.default(model_list): 'GLM_FS' did not have
## 'returnResamp="final"; the optimal tuning parameters are used
```

```
## Warning in resamples.default(model_list): 'GLMNET' did not have
## 'returnResamp="final"; the optimal tuning parameters are used

## Warning in resamples.default(model_list): 'RF' did not have
## 'returnResamp="final"; the optimal tuning parameters are used

## Warning in resamples.default(model_list): 'KNN' did not have
## 'returnResamp="final"; the optimal tuning parameters are used
```

```
# dotplot
dotplot(resamples, metric = 'ROC')
```




```
# Previsao dos melhores
GlmnetModel_pred <-predict(GlmnetModel, test_sample)
FSModel_pred <- predict(FSModel, test_sample)
confusionMatrix(GlmnetModel_pred, test_sample$CreditStatus)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Good Bad
##       Good  240  67
##       Bad    40  53
##
##                Accuracy : 0.7325
##                  95% CI : (0.6863, 0.7753)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.08524
##
##                   Kappa : 0.3193
##  Mcnemar's Test P-Value : 0.01195
```

```
##
##             Sensitivity : 0.8571
##             Specificity : 0.4417
##          Pos Pred Value : 0.7818
##          Neg Pred Value : 0.5699
##              Prevalence : 0.7000
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.7675
##       Balanced Accuracy : 0.6494
##
##        'Positive' Class : Good
##
```

```
confusionMatrix(FSModel_pred, test_sample$CreditStatus)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Good Bad
##       Good  234  64
##       Bad    46  56
##
##                Accuracy : 0.725
##                  95% CI : (0.6784, 0.7682)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.1498
##
##                   Kappa : 0.3159
##  Mcnemar's Test P-Value : 0.1050
##
##             Sensitivity : 0.8357
##             Specificity : 0.4667
##          Pos Pred Value : 0.7852
##          Neg Pred Value : 0.5490
##              Prevalence : 0.7000
##          Detection Rate : 0.5850
##    Detection Prevalence : 0.7450
##       Balanced Accuracy : 0.6512
##
##        'Positive' Class : Good
##
```

Como se pode observar acima, os modelos GLM's obtiveram melhor desempenho dentre os outros. Resultado também observado nas suas tabelas de confusão: houve uma melhora na acurácia de 2.48 %, no entanto o mais incrível é constatar como o modelo mais simples possui tanta capacidade.

Vale ressaltar como há espaço para muitas outras melhorias, como a criação de novas variáveis a partir das originais ou outros algorítmos de classificação, no entanto o objetivo principal do trabalho foi demonstrar como somente uma melhor escolha de variáveis pode, por vezes, ser um fator muito mais relevante para performance do que o uso de modelos super complexos.

Fim