Classificação de Texto Musical

Antonio Mendes M. Jr

26/02/2021

Classificação textual

Neste documento é descrita a criação de classificadores binários para NLP com o objetivo de classificar letras musicais como sendo pertencentes a duas artistas distintas: Beyoncé e Rihanna.

Pacotes utilizados

Primeiramente foram carregados os pacotes utilizados. O pacote `readxl` para leitura de arquivos xls, `dplyr` para operações com os dataframes e afins, `tm` para trabalhar com Bag of Words/Matrizes de Termos, `RSNNS` para trabalhar com Redes Neurais Artificiais, `naivebayes` para trabalhar com o claussificador Naive Bays, `e1071` para classificador SVM, `randomForest` para classificador Floresta Aleatória, `pROC` para plotar curvas ROC e `caret` para calcular métricas de avaliação.

```
library(readxl)
library(dplyr)
library(tm)
library(RSNNS)
library(pROC)
library(caret)
library(naivebayes)
library(e1071)
library(randomForest)
```

Dataset

O dataset é composto por duas colunas contendo as letras das músicas e o nome da artista à qual pertence.

Primeiramente os dados são carrados a partir de um arquivo `.xls` e são então embaralhados para não beneficiar alguma classe no momento da divisão do dataset em dados de treinamento e teste. Para fins de ilustração, são mostrados os dez primeiros elmentos do dataset.

```
dados <- read_excel("teste_smarkio_lbs.xls", sheet=2, col_names=TRUE)
dados <- dados[sample(1:nrow(dados),
                      length(1:nrow(dados))), 1:ncol(dados)]
head(dados, 10)
```

```
## # A tibble: 10 x 2
##    letra                                                          artista
##    <chr>                                                          <chr>
##  1 "Mama, I understand your many sleepless nights When you're seat and ~ Beyoncé
##  2 "Ladies and gentleman Welcome to Beyoncé Homecoming 2018"              Beyoncé
##  3 "I be on the hotline, like err'day Making sure the DJ know what I wa~ Beyoncé
##  4 "Oh, angels sent from up above You know you make my world light up W~ Beyoncé
##  5 "[Kanye West:] You a bad girl and your friends bad too, oh We got th~ Beyoncé
```

```
##  6 "[jay-z] Yeah, b. talk yo shhhhhh (partna let me upgrade u) How you ~ Beyoncé
##  7 "It's over and done, But the heartache lives on inside And who is th~ Beyoncé
##  8 "You're holding me And I close my eyes, You're whispering And I star~ Beyoncé
##  9 "Hit me!  [Hook: Beyoncé, Kelly Rowland &amp;amp; Michelle Williams] Can~ Beyoncé
## 10 "Jay-z Uh-uh-uh You ready b? Let's go get 'em.  Look for me, young b~ Beyoncé
```

Extração de características

Visando a extração de características (features extraction), foi criado o Bag of Words dos textos das letras musicais. Tal ação resulta em uma relação da contagem dos termos (palavras) em cada um dos documento (letras). Durante a criação da Matriz de Termos, todas as palavras são passadas para escrita em minúsculo, são removidos os sinais de pontuação, as stopwords e os espaços em branco. Além disso, foi realizada a stemização (stemming) e são removidos os termos mais esparsos da matriz. A função `head()` mostra parte dos dados após processamento.

```
gerar_matriz_termos = function(res){

  corpus = VCorpus(VectorSource(res$letra))
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removeNumbers)
  corpus<- tm_map(corpus,removePunctuation)
  corpus <- tm_map(corpus, removeWords, stopwords("english"))
  corpus <- tm_map(corpus, stemDocument)
  corpus <- tm_map(corpus, stripWhitespace)
  matriz_termos <- DocumentTermMatrix(corpus)
  matriz_termos <- removeSparseTerms(matriz_termos, .95)
  return(matriz_termos)
}

#Matriz de termos
matriz <- gerar_matriz_termos(dados)


dados <- data.frame(as.factor(dados$artista),
                    as.matrix(matriz))
names(dados)[1]<- paste("artista")

head(dados, c(10L, 13L))
```

```
##    artista act aint air alon alright alway anoth anyth arm around ask ass
## 1  Beyoncé   0    0   0    0       0     2     0     0   0      0   0   0
## 2  Beyoncé   0    0   0    0       0     0     0     0   0      0   0   0
## 3  Beyoncé   0    0   0    1       0     0     0     0   1      0   0   0
## 4  Beyoncé   0    0   0    0       0     0     0     0   0      0   0   0
## 5  Beyoncé   0    0   0    0       0     0     0     0   0      0   0   0
## 6  Beyoncé   0    3   0    2       0     0     0     2   0      0   0   0
## 7  Beyoncé   0    0   0    0       0     0     0     0   0      0   0   0
## 8  Beyoncé   0    0   0    0       0     0     1     0   0      0   0   0
## 9  Beyoncé   0    0   0    0       0     0     0     0   0      0   1   0
## 10 Beyoncé   0    2   1    0       0     0     0     2   0      0   0   0
```

Classificador 1) Redes Neurais Artificiais (RNAs)

O primeiro classificador utilizado foi RNA do tipo Multilayer Perceptron (MLP). Os dados foram dividios em variáveis de entrada (dataValues) em valores alvo (dataTargets).

```
dataValues <- dados[-1]
dataTargets <- t(dados[1])
```

Os valores alvo foram codificados para o formato one-hot.

```
dataTargets <- decodeClassLabels(dataTargets)
head(dataTargets)
```

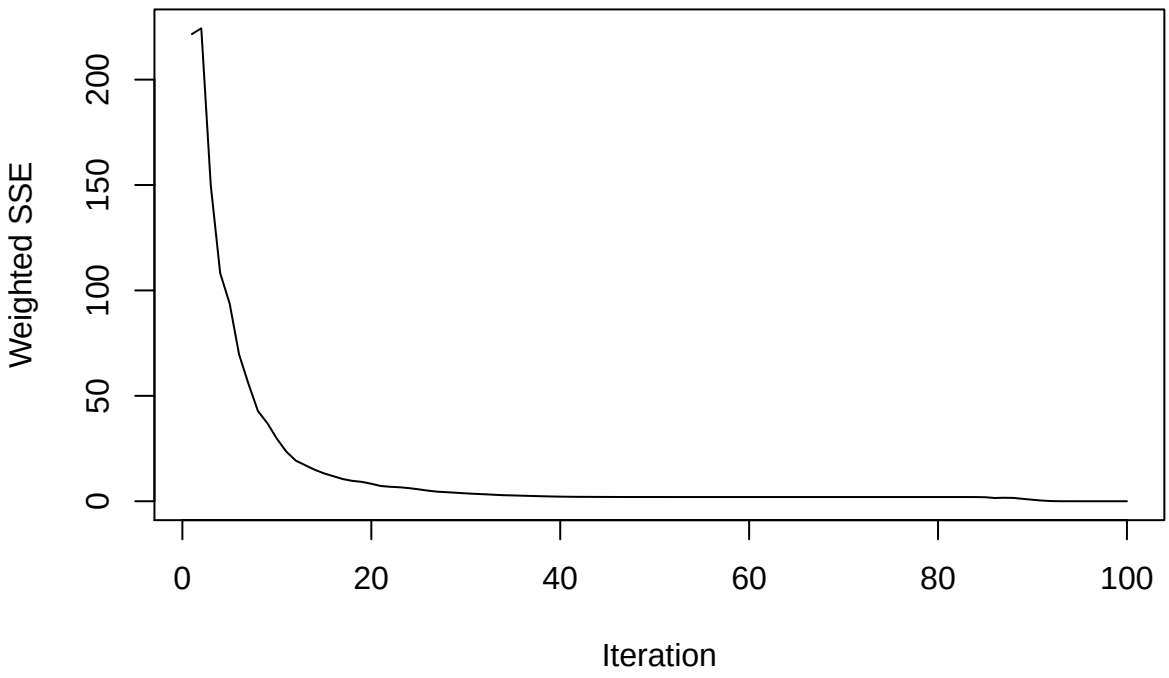
```
##      Beyoncé Rihanna
## [1,]       1       0
## [2,]       1       0
## [3,]       1       0
## [4,]       1       0
## [5,]       1       0
## [6,]       1       0
```

Os dados foram divididos em treino e teste, considerando a proporção 85/15. Além disso, foram normalizados para valores entre 0 e 1.

```
dataset <- splitForTrainingAndTest(dataValues, dataTargets, ratio = 0.15)
dataset <- normTrainingAndTestSet(dataset, type = "0_1")
```

A RNA formada possui 10 neurônios na camada escondida e dois na camada de saída (número de classes), utiliza função de ativação tangente hiperbólica nos neurônios da camada escondida e logística nos neurônios da camada de saída. O algoritmo de aprendizado escolhido foi o Resilient Backpropagation. O erro interativo foi plotado.

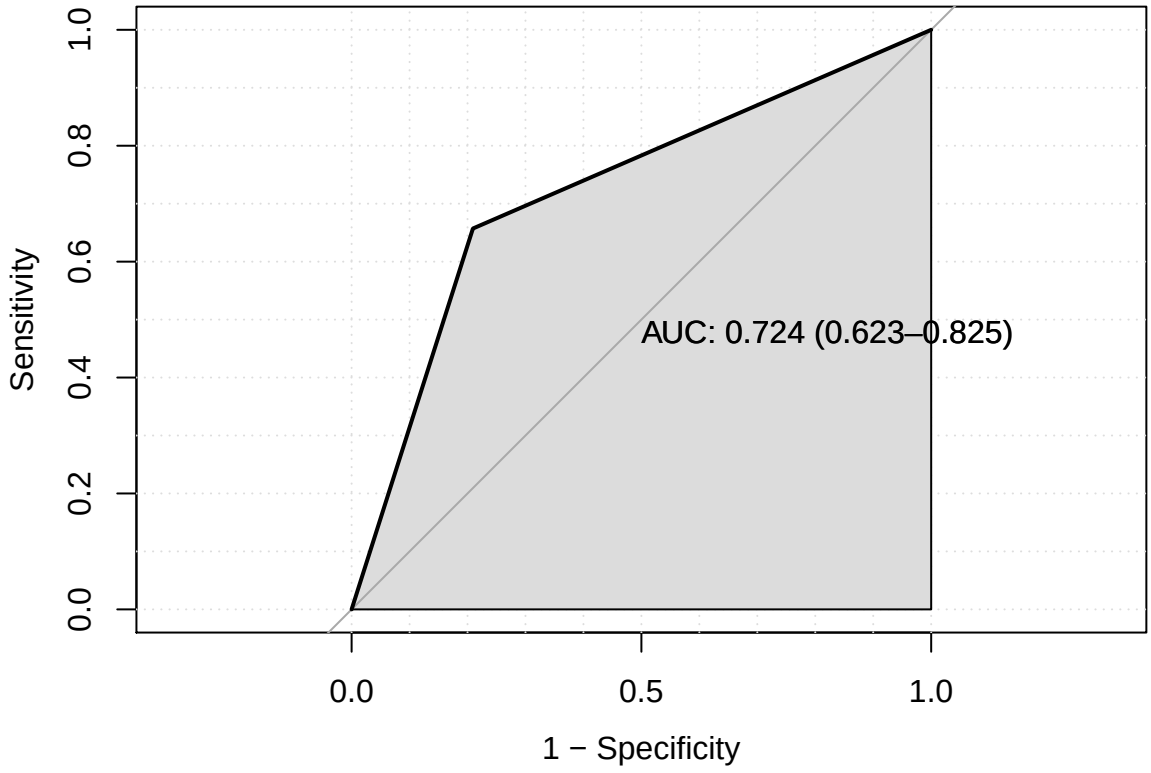
```
model_rna <- mlp(dataset$inputsTrain, dataset$targetsTrain, size=c(10),
                 hiddenActFunc='Act_TanH', learnFunc = "Rprop",
                 outputActFunc = "Act_Logistic", maxit=100, linOut = FALSE)
plotIterativeError(model_rna)
```



Em seguida o modelo é utilizado para realizar predições em relação aos dados de teste. É plotada a curva ROC do modelo e calculada métricas a partir da matriz de confusão (acurácia, índice Kappa)

```
pred_rna <- predict(model_rna, dataset$inputsTest)
data_test_code <- encodeClassLabels(dataset$targetsTest)
resp_test_code <- encodeClassLabels(pred_rna)
```

```
roc <- pROC::roc(data_test_code ~ resp_test_code,
                 plot=T, print.auc=TRUE,
                 auc.polygon=T, grid=TRUE, legacy.axes=T, ci = T)
```




```
metrics <- caret::confusionMatrix(as.factor(encodeClassLabels(
  dataset$targetsTest)),
  as.factor(encodeClassLabels(pred_rna)))
(cm <- metrics$table)
```

```
##           Reference
## Prediction  1  2
##          1 34  9
##          2 12 23
```

```
(acc_rna <- metrics$overall[1])
```

```
##  Accuracy
## 0.7307692
```

```
(kappa_rna <- metrics$overall[2])
```

```
##     Kappa
## 0.4514401
```

```
(auc_rna <- roc$auc)
```

```
## Area under the curve: 0.7239
```

Classificador 2) Naive Bayes

O segundo classificador utilizado foi Naive Bayes. Os dados novamente foram divididos em 80/20 para treinamento e teste. As métricas de avaliação foram calculadas.

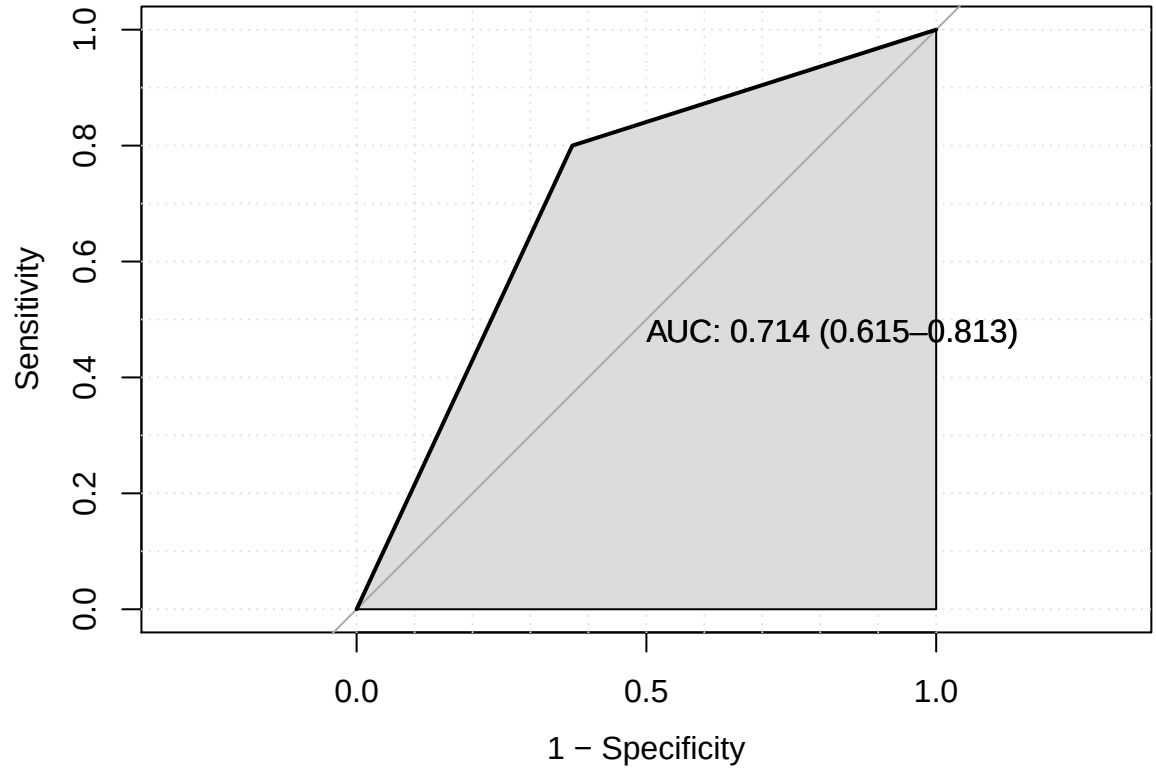
```
# Divisão em dados de treino e teste
data_train <- dados[1:round(0.85*nrow(dados)),]
data_test <- anti_join(dados,data_train)

model_nb <- naivebayes::naive_bayes(artista ~ ., data = data_train)
pred_nb <- predict(model_nb, data_test, type = "class")
```

```
## Warning: predict.naive_bayes(): more features in the newdata are provided as
## there are probability tables in the object. Calculation is performed based on
## features to be found in the tables.
```

```
data_test_code <- as.numeric(data_test$artista)
resp_test_code <- as.numeric(pred_nb)
```

```
roc <- pROC::roc(data_test_code ~ resp_test_code,
                 plot=T, print.auc=TRUE,
                 auc.polygon=T, grid=TRUE, legacy.axes=T, ci = T)
```




```
metrics <- caret::confusionMatrix(as.factor(data_test$artista),
                                  as.factor(pred_nb))
(cm <- metrics$table)
```

```
##           Reference
## Prediction Beyoncé Rihanna
```

```
##    Beyoncé      27      16
##    Rihanna       7      28
```

```
(acc_nb <- metrics$overall[1])
```

```
##  Accuracy
## 0.7051282
```

```
(kappa_nb <- metrics$overall[2])
```

```
##     Kappa
## 0.4179104
```

```
(auc_nb <- roc$auc)
```

```
## Area under the curve: 0.714
```

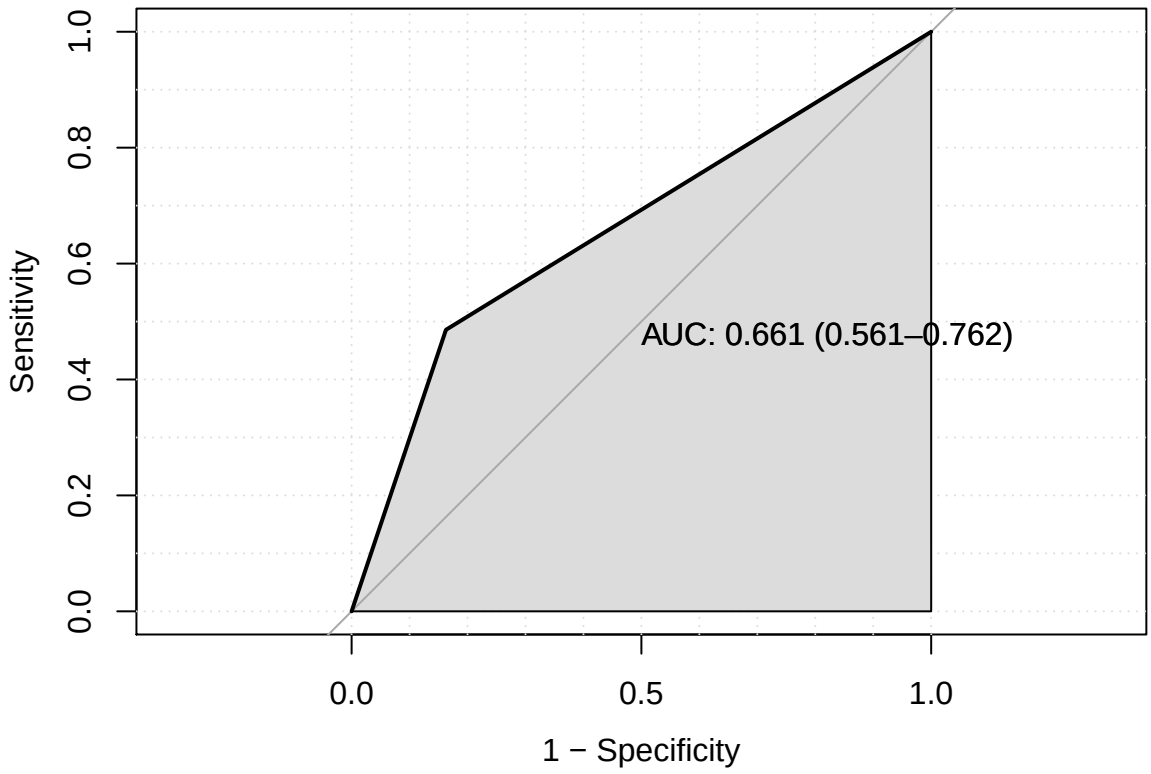
Classificador 3) SVM

O terceiro classificador utilizado foi SVM. Os dados novamente foram divididos em 80/20 para treinamento e teste. As métricas de avaliação foram calculadas.

```
# Divisão em dados de treino e teste
data_train <- dados[1:round(0.85*nrow(dados)),]
data_test <- anti_join(dados,data_train)

model_svm <- e1071::svm(artista ~ ., data = data_train)
pred_svm <- predict(model_svm, data_test, type = "class")
data_test_code <- as.numeric(data_test$artista)
resp_test_code <- as.numeric(pred_svm)
```

```
roc <- pROC::roc(data_test_code ~ resp_test_code,
                 plot=T, print.auc=TRUE,
                 auc.polygon=T, grid=TRUE, legacy.axes=T, ci = T)
```




```
metrics <- caret::confusionMatrix(as.factor(data_test$artista),
                                  as.factor(pred_svm))
(cm <- metrics$table)
```

```
##           Reference
## Prediction Beyoncé Rihanna
##    Beyoncé      36       7
##    Rihanna      18      17
```

```
(acc_svm <- metrics$overall[1])
```

```
##  Accuracy
## 0.6794872
```

```
(kappa_svm <- metrics$overall[2])
```

```
##     Kappa
## 0.3326489
```

```
(auc_svm <- roc$auc)
```

```
## Area under the curve: 0.6615
```

Classificador 4) Random Forest

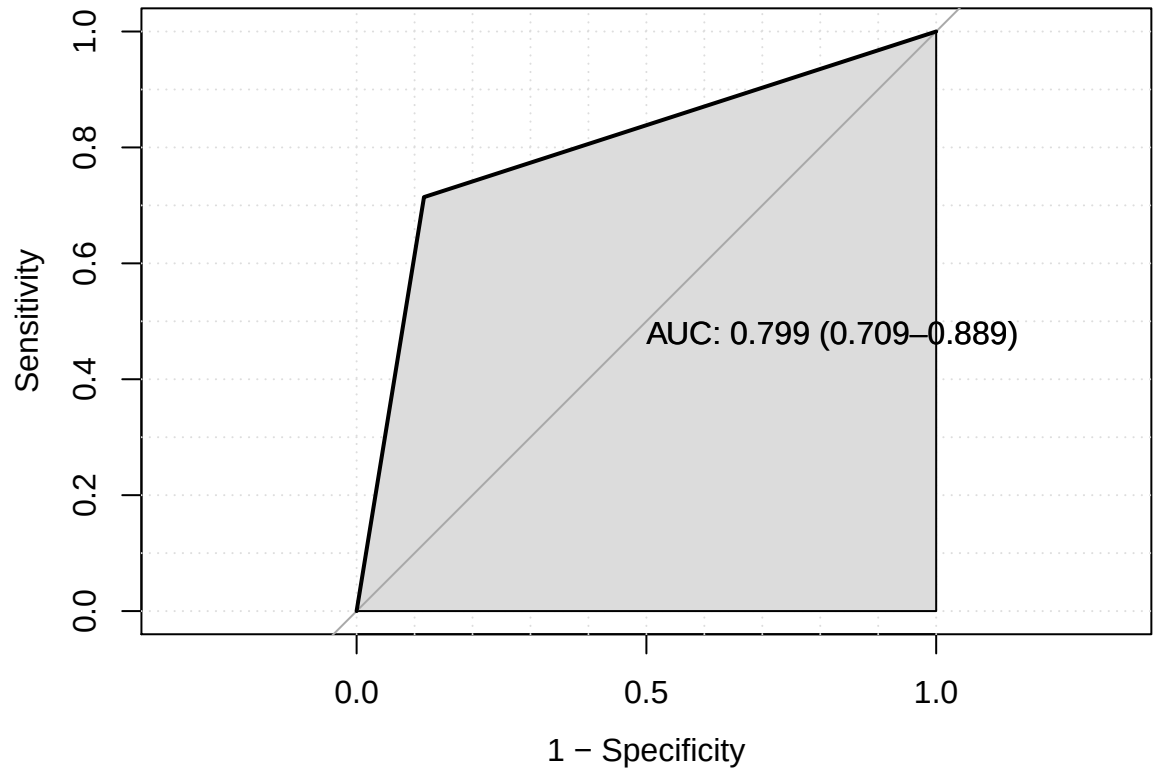
O terceiro classificador utilizado foi Random Forest. Os dados novamente foram divididos em 80/20 para treinamento e teste. Foram definidas 150 árvores. As métricas de avaliação foram calculadas.

```
# Divisão em dados de treino e teste
data_train <- dados[1:round(0.85*nrow(dados)),]
data_test <- anti_join(dados,data_train)

model_rf <- randomForest(artista ~ ., data = data_train,
                         ntree=150, proximity=TRUE)

pred_rf <- predict(model_rf, data_test, type = "class")
data_test_code <- as.numeric(data_test$artista)
resp_test_code <- as.numeric(pred_rf)
```

```
roc <- pROC::roc(data_test_code ~ resp_test_code,
                 plot=T, print.auc=TRUE,
                 auc.polygon=T, grid=TRUE, legacy.axes=T, ci = T)
```




```
metrics <- caret::confusionMatrix(as.factor(data_test$artista),
                                  as.factor(pred_rf))
(cm <- metrics$table)
```

```
##           Reference
## Prediction Beyoncé Rihanna
##    Beyoncé      38       5
##    Rihanna      10      25
```

```r
(acc_rf <- metrics$overall[1])
```

```
##  Accuracy
## 0.8076923
```

```r
(kappa_rf <- metrics$overall[2])
```

```
##     Kappa
## 0.6060606
```

```r
(auc_rf <- roc$auc)
```

```
## Area under the curve: 0.799
```

Classificador 5) Comparação dos modelos

A fim de facilitar a comparação entre os modelores, as métricas de avaliação foram sumarizadas na tabela abaixo.

```
df_metrics <- data.frame(c(acc_rna, acc_nb, acc_svm, acc_rf),
                         c(kappa_rna, kappa_nb, kappa_svm, kappa_rf),
                         c(auc_rna, auc_nb, auc_svm, auc_rf))
rownames(df_metrics) <- c('Redes Neurais', 'Naive Bayes',
                          'SVM', 'Random Forest')
colnames(df_metrics) <- c('Acurácia', 'Kappa', 'AUC')
df_metrics
```

```
##                Acurácia     Kappa       AUC
## Redes Neurais 0.7307692 0.4514401 0.7239203
## Naive Bayes   0.7051282 0.4179104 0.7139535
## SVM           0.6794872 0.3326489 0.6614618
## Random Forest 0.8076923 0.6060606 0.7990033
```