Predição do sexo baseado em áudios utilizando árvores de classificação

Fernando Grasser e Pedro Vianna

Introdução

A predição é um recurso estatístico que consiste em alocar, para uma variável específica, um valor baseado nos resultados de outras variáveis.

Com o avanço tecnológico e computacional, foram surgindo novas formas de se criar modelos e realizar estas predições. Um destes métodos é a criação de árvores de decisão, que consiste em utilizar variáveis explicativas e, com separações progressivas baseadas nos valores destas variáveis, predizer o valor da variável resposta.

Caso a variável resposta seja uma variável qualitativa, as árvores de decisão poderão ser utlizadas para classificar as observações em cada uma das categorias desta variável. Assim, essas árvores passam a se chamar árvores de classificação.

Baseado nestes conceitos, os alunos pensaram em um estudo envolvendo variáveis relacionadas ao som, provenientes de áudios das vozes das pessoas, com o objetivo de classificar o sexo da pessoa que está falando no áudio.

Além disso, como dito que existem diversas formas de se modelar as árvores para obter as melhores estimativas, os alunos utilizaram três diferentes algoritmos para modelar este cenário, com o objetivo de se decidir qual possivelmente é o melhor algoritmo.

Dados e pacotes

Os dados e os pacotes utilizados neste trabalho seguem abaixo.

```
setwd("/Users/pedrovianna/Documents/COMPUTACIONAL/Trabalho")
options(scipen = 999)

library(ipred)
library(dplyr)
library(randomForest)
library(gbm)
library(rpart)
library(rattle)
library(tuneR)
library(seewave)
library(ggplot2)
library(tidyr)


voices <- read.csv("voices.csv")
voices <- voices[,-c(seq(13,20,1))]

audios <- read.csv("audios.csv") # Dados coletados pelos alunos
```

Essas variáveis foram excluídas, pois os dados que os alunos coletaram não possuíam estas variáveis.

Bagging

O bootstrap é muito utilizado em situações em que é muito difícil ou até mesmo impossível de se calcular o erro padrão de uma estimativa de interesse. No caso do bagging, este método é usado de uma froma completamente diferente, com o objetivo de melhorar formas de aprendizado estatístico, como as árvores de decisão.

O bootstrap aggregation, também conhecido como bagging, é um procedimento utilizado para reduzir a variância de um processo de aprendizado estatístico. Este método é muito utilizado no contexto de árvores de decisão e por isso está sendo abordado neste trabalho.

Relembrando, dado um conjunto de n observações independentes $Z_1, Z_2, \ldots, Z_n$, cada um com variância σ^2, a variância da média $\bar{Z}$ das observações é dada por σ^2/n. Em outras palavras, utlizar a média de um conjunto de observações reduz a variância. Portanto, uma forma de se diminuir a variância e, assim, aumentar o poder preditivo de um método de aprendizado estatístico é pegar vários conjuntos de dados teste da população, criar modelos preditivos separados, usando em cada um conjunto de dados diferente, e calcular a média das predições resultantes.

Como, na prática, não se tem acesso a múltiplos conjuntos de dados de treinamentos, utilizamos o bootstrap para realizar amostras com repetição dentro do conjunto de dados teste que se possui para o estudo. Assim, considerando que foram realizadas W diferentes estimativas, estas sendo $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^W(x)$, o resultado final do processo de bagging será

$$\hat{f}_{bag}(x) = \frac{1}{W}\sum_{i=1}^{w} \hat{f}^i(x)$$

Essa, descrita acima, é a forma que o bagging é aplicado quando utilizamos árvores de regressão, que são árvores de decisão que possuem uma variável resposta quantitativa. Então, como esse método pode ser aplicado em árvores de classificação?

A forma mais simples de se aplicar o bagging para árvores de classificação é observar a classe predita para cada uma das W árvores criadas em uma observação e, ao final deste processo, classifica-se essa observação na classe que possuiu maior frequência entre as W predições finais.

Random Forest

Como no bagging, no modelo utilizando random forest são construídas um número de árvores de decisão em conjuntos de dados criados com bootstrap. Entretanto, na construção dessas árvores, cada vez que uma divisão é considerada em uma árvore, uma amostra de m dos p preditores é selecionada como candidato para divisão. Uma nova amostra de m preditores é selecionada a cada nova divisão; o valor de m é tipicamente $m \approx \sqrt{p}$.

O fato do método de random forest realizar esta amostragem dos preditores permite que se criem árvores bem diferentes a cada conjunto de dados teste, e com isso mais preditores sendo utilizados ao final do processo, ao contrário do bagging, em que o principal preditor deve ser usado para realizar a primeira divisão em todas as árvores e, assim, são criadas inúmeras árvores parecidas. Assim, as predições das árvores do bagging serão altamente correlacionadas. O fato das predições serem correlacionadas faz com que a variância não seja tão reduzida ao final do processo; portanto, o random forest vem para tentar contornar esse possível problema.

Boosting

Ao contrário do bagging, que cria inúmeras árvores independentes por meio do uso do bootstrap, o boosting cria suas árvores sequencialmente: cada árvore é criada utilizando informações das árvores criadas anteriormente. Além disso, o boosting não utiliza o bootstrap; cada árvore é ajustada utilizando uma versão modificada do banco de dados original.

No caso de um modelo de regressão, assim como o bagging, boosting involve a combinação de um grande número de árvores de decisão, $\hat{f}^1, \ldots, \hat{f}^W$. Esse método tem um aprendizado mais lento. Dado um modelo atual, será ajustado uma árvore de decisão para os resíduos deste modelo; após isso, acrescenta-se essa árvore para a função ajustada com o objetivo de atualizar os resíduos. Essas árvores podem ser pequenas; ajustando-se árvores pequenas para os resíduos, o valor de $\hat{f}$ é melhorado em áreas que não estavam performando bem. O parâmetro λ desascelera ainda mais o processo, permitindo que mais árvores e com diferentes formatos ajam nos resíduos.

Boosting tem três parâmetros de sintonização:

- O número de árvores W. Um valor alto de W pode não deixar o modelo tão adequado.
- O parâmetro λ, $\lambda > 0$. Esse valor controla a velocidade com que o método aprende.Os valores típicos para este parâmetro são 0,01 e 0,001, sendo escolhidos de acordo com o problema.
- O número c de divisões em cada árvore, que controla a complexidade do modelo implementado.

O algoritmo do boosting para árvores de regressão é dado por:

1. Defina $\hat{f}(x) = 0$ e $r_i = y_i$ para todos os i's do banco de dados teste.
2. Para $w = 1, 2, \ldots, W$, repita:
   - Modele uma árove $\hat{f}^w$ com d divisões ($d + 1$ nós teminais) para o conjunto treino (X, r).
   - Atualize $\hat{f}$ adicionando uma versão reduzida da nova árvore:
   $$\hat{f}(x) \leftarrow \hat{f} + \lambda \hat{f}^w(x).$$
   - Atualize os resíduos:
   $$r_i \leftarrow r_i - \lambda \hat{f}^w(x_i).$$
3. Dê a saída do modelo boosting:
$$\hat{f}(x) = \sum_{w=1}^{W} \lambda \hat{f}^w(x).$$

Boosting em árvores de classificação é realizado de forma similar, mas sua teoria é levemente mais complexa e, portanto, não será abordada neste trabalho.

Influência Relativa em Boosting

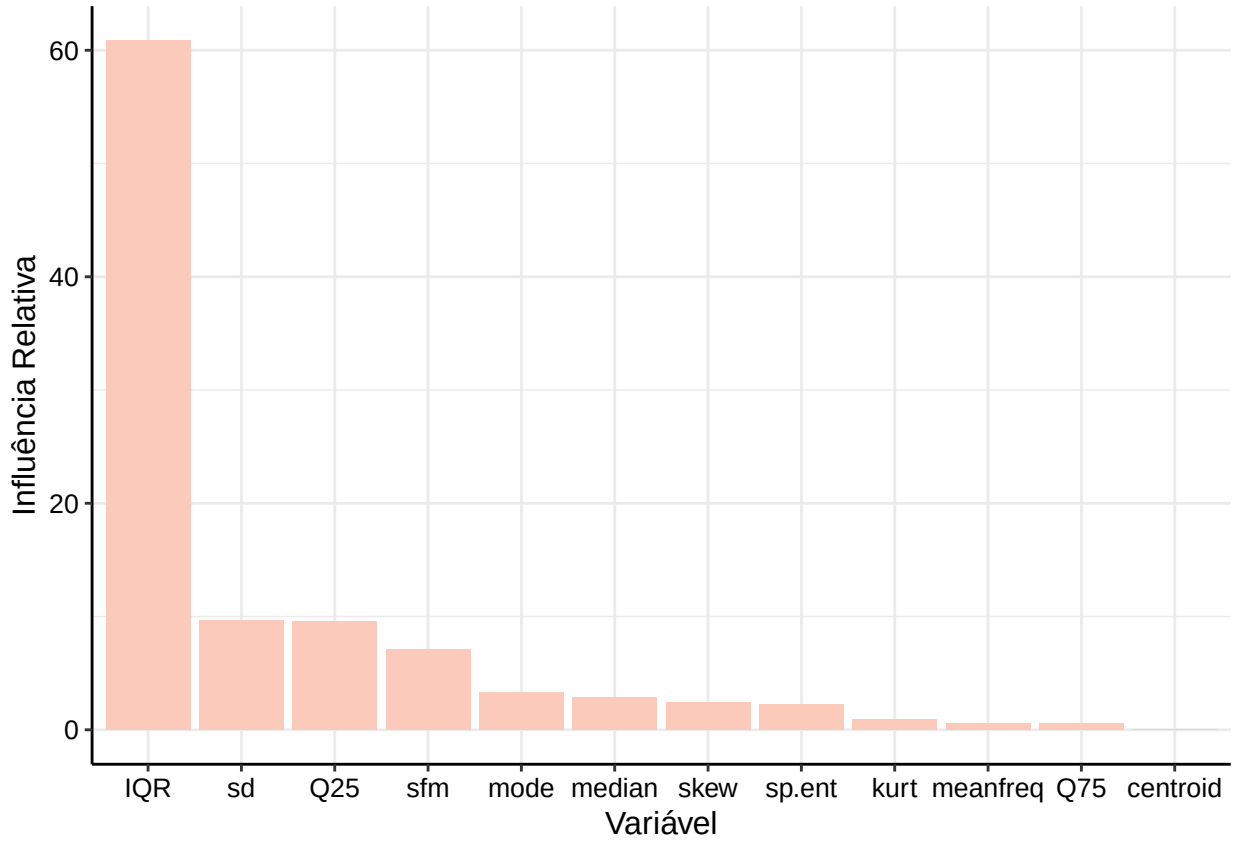
Um método desenvolvido por Friedman(2001) estima a influência relativa das variáveis para o modelo de predição. Esses valores são baseados no número de vezes em que uma variável é selecionada para a construção das árvores e recebem pesos de acordo com a contribuição na melhoria do modelo. Abaixo, pode-se ver a aplicação de inicial da metodologia boosting, utilizando a função `summary()` é possível ver as importâncias relativas que foram atribuídas às variáveis.

```
teste3 <- voices
teste3$label <- ifelse(teste3$label == "male", 1, 0)
treino3.v <- sample(1:nrow(voices), 317)
voices3 <- teste3[-treino3.v,]
treino3 <- teste3[treino3.v,]



boosting <-  gbm(label~., treino3, distribution="bernoulli", n.trees = 1000,
                 interaction.depth = 4)
```

```r
importancia <- data.frame(summary(boosting, plot=F))

reorder(importancia$var, -importancia$rel.inf)
```

```
##  [1] IQR      sd       Q25      sfm      mode     median   skew
##  [8] sp.ent   kurt     meanfreq Q75      centroid
## attr(,"scores")
##    centroid         IQR        kurt    meanfreq      median        mode
##   0.0000000 -60.8547514  -0.9181422  -0.5436590  -2.8639831  -3.2739171
##         Q25         Q75          sd         sfm        skew      sp.ent
##  -9.5434249  -0.5397398  -9.6903333  -7.0809249  -2.4630045  -2.2281198
## 12 Levels: IQR sd Q25 sfm mode median skew sp.ent kurt meanfreq ... centroid
```

```r
ggplot(importancia, aes(x=reorder(importancia$var, -importancia$rel.inf), y=rel.inf)) +
  geom_bar(stat="identity", position="dodge", fill=c("#fbcabb"))+
  labs(x="Variável", y="Influência Relativa")+
  theme_bw()+
  theme(axis.title.y=element_text(colour="black",size=12),
        axis.title.x=element_text(colour="black",size=12),
        axis.text=element_text(colour="black",size=9.5),
        panel.border=element_blank(),
        axis.line=element_line(colour="black"))
```



Em seguida, para testarmos o quanto a o efeito da importância relativa pode causar em modelos preditivos, foi realizado a implemetação de dois modelos bagging. Nota-se que o primeiro é construído a partir das três variáveis com maior importância, enquanto o segundo recebe as três menos importantes.

```
good.model <- gbm(label ~ IQR + Q25 + sd, treino3, distribution="bernoulli",
                  n.trees = 1000, interaction.depth = 4)
sum(diag(prop.table(table(predict(good.model, teste3, type = "response",
                                  n.trees = 1000)>.5, teste3$label))))
```

```
## [1] 0.9188763
```

```
bad.model <- gbm(label ~ centroid + kurt + meanfreq, treino3, distribution="bernoulli",
                 n.trees = 1000, interaction.depth = 4)
sum(diag(prop.table(table(predict(bad.model, teste3, type = "response",
                                  n.trees = 1000)>.5, teste3$label))))
```

```
## [1] 0.7402146
```

As predições mostram a diferença entre a taxa de acerto entre o modelo bom (`good.model`) e o modelo ruim (`bad.model`).

A Aplicação

Como dito um pouco na introdução, o objetivo deste trabalho é classificar vozes como masculinas e femininas, baseadas em certas variáveis retiradas de áudios. Queremos, também, descobrir qual seria o melhor método para se utilizar em estudos deste tipo, baseado na taxa final de predições corretas.

Com isso, é proposto um estudo de simulação, em que será utilizada a validação cruzada para validar os modelos criados, isto é, o banco de dados é separado aleatoriamente em duas partes, uma que treinará os modelos e outra que será usada para predizer o sexo. Além disso, serão utilizados vários tamanhos para esta amostra treino, para observar se este valor influencia na taxa final das predições.

O banco de dados utilizado, disponível no link na referência do estudo, é composto de 3168 observações e, após a exclusão de algumas, 13 variáveis. Este banco foi nomeado na importação como *voices*.

Primeiramente, utilizaremos um valor bem baixo para o tamanho de amostra treino, sendo este de 50 observações. Por ser um estudo de simulação, utilizaremos 200 amostras diferentes, e, após o treinamento dos modelos, será observada a taxa de predições corretas do modelo, sendo estas predições feitas nas observações dos bancos de dados que não estavam presente na amostra.

A implementação desta ideia é dada por:

```
n <- nrow(voices)
t <- 50

taxa_bag4 <- c()
taxa_forest4 <- c()
taxa_boost4 <- c()

for(i in 1:200){
  tt <- sample(1:n, t)
  treino <- voices[tt, ]
  teste <- voices[-tt, ]
  bagging <- bagging(label ~ ., treino)
  forest <- randomForest(label ~ ., treino)


  teste2 <- teste
  treino2 <- treino
  a <- teste2$label
  levels(a) <- c(0,1) #0=female 1=male
```

```
  b <- treino2$label
  levels(b) <- c(0,1)
  teste2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  boosting <-  gbm(label~., treino2, distribution="bernoulli", n.trees = 1000,
                   interaction.depth = 4)

  taxa_bag4[i] <- sum(diag(prop.table(table(predict(bagging, teste,
                                                    type = "prob")[,2]>.5,
                                            teste$label))))

  taxa_forest4[i] <- sum(diag(prop.table(table(predict(forest,
                                                       teste,
                                                       type = "prob")[,2]>.5,
                                               teste$label))))

  taxa_boost4[i] <- sum(diag(prop.table(table(predict(boosting,
                                                      teste2,
                                                      type = "response",
                                                      n.trees = 1000)>.5,
                                              teste2$label))))

}

respostas4 <- data.frame(taxa_forest4,taxa_bag4,taxa_boost4)
```

Como é visto no código, foram utilizadas as funções *bagging*, *randomForest* e *gbm*, dos pacotes *ipred*, *randomForest* e *gbm*, respectivamente, para se modelar este estudo baseado nestes algoritmos. Após a criação do modelo, somamos a diagonal da tabela que mostrava a proporção de predições corretas, já que cada elemento da diagnonal representava os acertos pelo sexo. Com isso, obteve-se uma proporção média de predições corretas, para cada método, iguais a

```
apply(respostas4, 2, mean)
```

```
## taxa_forest4    taxa_bag4  taxa_boost4
##    0.8876427    0.8755821    0.8538037
```

Como pode-se perceber, obtemos proporções em volta de 80% e 90% de predições corretas, utilizando apenas 50 observações para treinar este modelo.

O desvio padrão destas estimativas são iguais a

```
apply(respostas4, 2, sd)
```

```
## taxa_forest4    taxa_bag4  taxa_boost4
##   0.01607491   0.02179626   0.02978847
```

Temos desvio padrões baixos para as estimativas, o que deve indicar que estes modelos são estáveis, já que foram utilizadas diferentes amostras para cada estimativa.

Agora, o mesmo método utilizado acima será utilizado com diferentes tamanhos de amostra, este sendo 317 (que representa 10% do tamanho total do banco), 1000 e 1584 (que representa 50% do total de observações do banco). Após esta implementação, observaremos as taxas médias de cada algoritmo.

```
# Amostra de tamanho 317

t <- round(0.1*n)
```

```r
taxa_bag <- c()
taxa_forest <- c()
taxa_boost <- c()

for(i in 1:200){
  tt <- sample(1:n, t)
  treino <- voices[tt, ]
  teste <- voices[-tt, ]
  bagging <- bagging(label ~ ., treino)
  forest <- randomForest(label ~ ., treino)


  teste2 <- teste
  treino2 <- treino
  a <- teste$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  teste2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  boosting <-  gbm(label~., treino2, distribution="bernoulli", n.trees = 1000,
                   interaction.depth = 4)

  taxa_bag[i] <- sum(diag(prop.table(table(predict(bagging,
                                        teste,
                                        type = "prob")[,2]>.5,
                                      teste$label))))

  taxa_forest[i] <- sum(diag(prop.table(table(predict(forest, teste,
                                           type = "prob")[,2]>.5,
                                     teste$label))))

  taxa_boost[i] <- sum(diag(prop.table(table(predict(boosting,
                                        teste2,
                                        type = "response", n.trees = 1000)>.5,
                                      teste2$label))))
}

respostas1 <- data.frame(taxa_forest,taxa_bag,taxa_boost)



# Amostra de tamanho 1000

t <- 1000


taxa_bag2 <- c()
taxa_forest2 <- c()
taxa_boost2 <- c()
```

```r
for(i in 1:200){
  tt <- sample(1:n, t)
  treino <- voices[tt, ]
  teste <- voices[-tt, ]
  bagging <- bagging(label ~ ., treino)
  forest <- randomForest(label ~ ., treino)


  teste2 <- teste
  treino2 <- treino
  a <- teste2$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  teste2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  boosting <-  gbm(label~., treino2, distribution="bernoulli", n.trees = 1000,
                   interaction.depth = 4)

  taxa_bag2[i] <- sum(diag(prop.table(table(predict(bagging,
                                        teste,
                                        type = "prob")[,2]>.5,
                                  teste$label))))

  taxa_forest2[i] <- sum(diag(prop.table(table(predict(forest,
                                           teste,
                                           type = "prob")[,2]>.5,
                                     teste$label))))

  taxa_boost2[i] <- sum(diag(prop.table(table(predict(boosting,
                                          teste2,
                                          type = "response", n.trees = 1000)>.5,
                                    teste2$label))))

}

respostas2 <- data.frame(taxa_forest2,taxa_bag2,taxa_boost2)


# Amostra de tamanho 1584

t <- round(0.5*n)

taxa_bag3 <- c()
taxa_forest3 <- c()
taxa_boost3 <- c()

for(i in 1:200){
  tt <- sample(1:n, t)
  treino <- voices[tt, ]
  teste <- voices[-tt, ]
  bagging <- bagging(label ~ ., treino)
```

```
  forest <- randomForest(label ~ ., treino)


  teste2 <- teste
  treino2 <- treino
  a <- teste2$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  teste2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  boosting <-  gbm(label~., treino2, distribution="bernoulli", n.trees = 1000,
                   interaction.depth = 4)

  taxa_bag3[i] <- sum(diag(prop.table(table(predict(bagging,
                                                    teste,
                                                    type = "prob")[,2]>.5,
                                            teste$label))))

  taxa_forest3[i] <- sum(diag(prop.table(table(predict(forest,
                                                       teste,
                                                       type = "prob")[,2]>.5,
                                               teste$label))))

  taxa_boost3[i] <- sum(diag(prop.table(table(predict(boosting,
                                                      teste2,
                                                      type = "response",
                                                      n.trees = 1000)>.5,
                                              teste2$label))))

}

respostas3 <- data.frame(taxa_forest3,taxa_bag3,taxa_boost3)
```

Primeiramente, será observada a taxa média dos algoritmos com a amostra de tamanho 317 e o desvio padrão destas estimativas. Estas seguem abaixo.

```
list(Médias = apply(respostas1, 2, mean), DP = apply(respostas1, 2, sd))
```

```
## $Médias
## taxa_forest    taxa_bag  taxa_boost
##   0.9250351   0.9176798   0.9275447
##
## $DP
## taxa_forest    taxa_bag  taxa_boost
## 0.006628651 0.008214195 0.007292535
```

Como é possível perceber, a proporção de classificações corretas está acima de 90% em todos os métodos. Além disso, o desvio padrão reduziu bastante para este tamanho de amostra, o que mostra que os modelos estão ficando mais consistentes.

Agora, os resultados para a amostra de tamanho 1000 seguem abaixo.

```
list(Médias = apply(respostas2, 2, mean), DP = apply(respostas2, 2, sd))
```

```
## $Médias
## taxa_forest2    taxa_bag2  taxa_boost2
##    0.9419649    0.9386900    0.9467689
##
## $DP
## taxa_forest2    taxa_bag2  taxa_boost2
##  0.004914368  0.005435921  0.004844105
```

Com o aumento do tamanho de observações de 317 para 1000 no banco treino, o aumento na proporção de predições corretas não foi tão elevado como no aumento de 50 para 317 observações. Ainda assim, houve um pequeno aumento e os desvio padrões foram ainda mais reduzidos.

Por último, observaremos os resultados utilizando 50% das observações do banco de dados como treino para o modelo, que deu um total de 1584 observações.

```
list(Médias = apply(respostas3, 2, mean), DP = apply(respostas3, 2, sd))
```

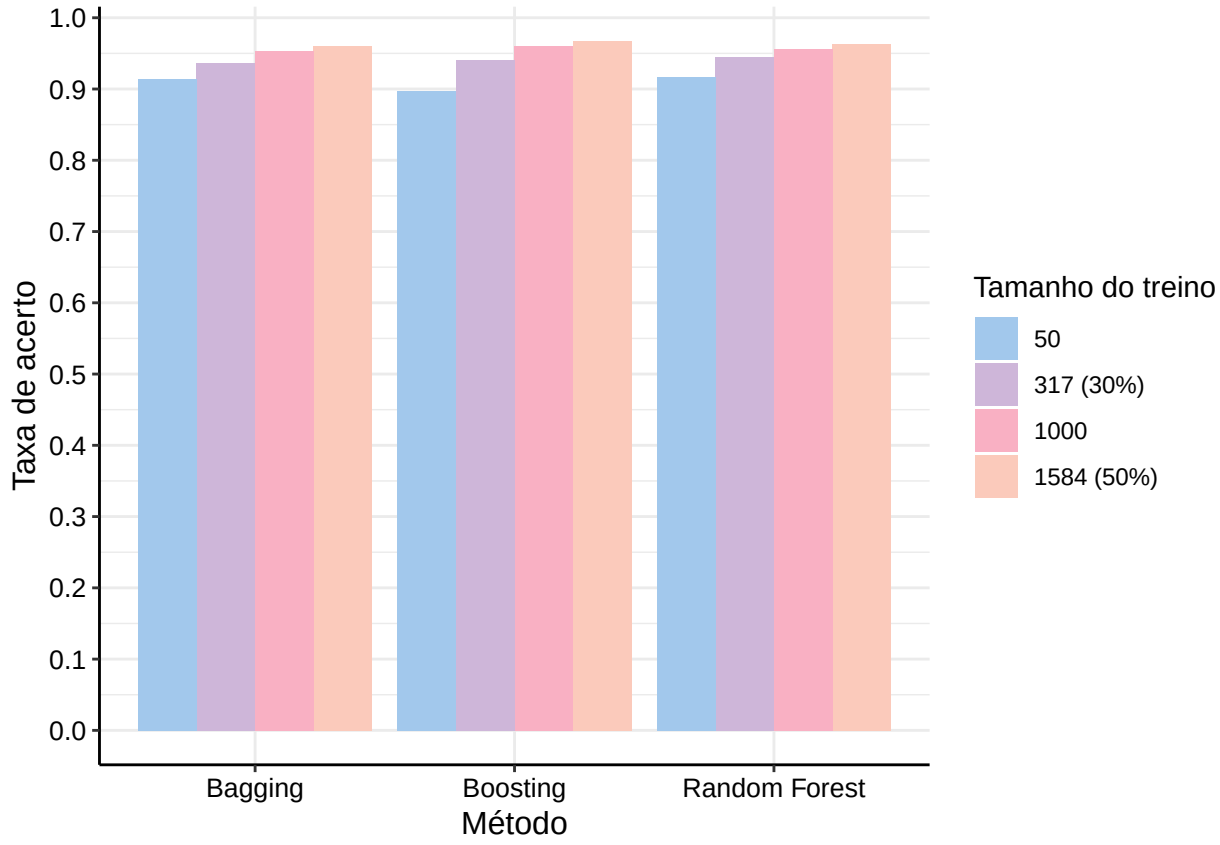
```
## $Médias
## taxa_forest3    taxa_bag3  taxa_boost3
##    0.9473011    0.9454451    0.9527083
##
## $DP
## taxa_forest3    taxa_bag3  taxa_boost3
##  0.005022246  0.005885700  0.004937893
```

Os resultados se manteram estáveis com relação ao tamanho de amostras anterior. Isso pode indicar que um número muito elevado de observações de treino pode não gerar resultados que valham o custo computacional que este método requer, devido ao tempo de processamento destes métodos.

Com isso, observaremos um gráfico de colunas com a predição média de cada médoto por cada tamanho de amostra, para obter uma visualização gráfica destes resultados.

```
respostas1t <- gather(respostas1, "grupo", dados)
respostas1t$teste <- 2
respostas2t <- gather(respostas2, "grupo", dados)
respostas2t$teste <- 3
respostas3t <- gather(respostas3, "grupo", dados)
respostas3t$teste <- 4
respostas4t <- gather(respostas4, "grupo", dados)
respostas4t$teste <- 1


respostas2t$grupo <- gsub("2", "", respostas2t$grupo)
respostas3t$grupo <- gsub("3", "", respostas3t$grupo)
respostas4t$grupo <- gsub("4", "", respostas4t$grupo)


all.table <- rbind(respostas1t,respostas2t,respostas3t, respostas4t)
all.table$teste <- as.factor(all.table$teste)


ggplot(all.table, aes(x=as.character(grupo), y=dados, fill=as.character(teste))) +
  geom_bar(stat="identity", position="dodge")+
  scale_fill_manual(name="Tamanho do treino", values=c("#a2c8ec","#ceb6d9",
                                                       "#f9b0c3", "#fbcabb"),
                                                       labels = c("50", "317 (30%)",
                                                                  "1000","1584 (50%)"))+
```

```
  scale_x_discrete(labels = c("Bagging","Boosting","Random Forest"))+
  labs(x="Método", y="Taxa de acerto")+
  theme_bw()+
  theme(axis.title.y=element_text(colour="black",size=12),
        axis.title.x=element_text(colour="black",size=12),
        axis.text=element_text(colour="black",size=9.5),
        panel.border=element_blank(),
        axis.line=element_line(colour="black"))+
  scale_y_continuous(breaks = seq(0, 1, 0.1))
```



Como podemos observar, há o aumento do número de predições corretas com o aumento do número de observações na amostra treino, mas este aumento é mais perceptível no primeiro salto, de 50 observações para 317.

Predições em Áudios de WhatsApp

Como dito na introdução, além deste banco de dados obtidos da Internet, foi criado, pelos alunos, um banco de dados a partir de áudios de WhatsApp (a forma que este banco foi criada está em anexo, junto com o restantes dos códigos utilizados no trabalho). Com isso, queria-se descobrir o poder preditivo deste tipo de modelo para áudios do dia a dia, para uma implementação mais realista destes métodos. Utilizando as mesmas implementações vistas na seção anterior, temos:

```
predicao.bagging <- c()
predicao.boosting <- c()
predicao.forest <- c()
```

```
for(i in 1:100){
  amostra.treino <- voices[sample(1:n, round(0.3*n)), ]


  aplicacao.bagging <- bagging(label ~ ., amostra.treino)
  aplicacao.forest <- randomForest(label ~ ., amostra.treino)


  audios2 <- audios
  treino2 <- amostra.treino
  a <- audios2$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  audios2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  aplicacao.boosting <-  gbm(label~., treino2, distribution="bernoulli", n.trees = 1000,
                             interaction.depth = 4)



  predicao.bagging[i] <- sum(diag(prop.table(table(predict(aplicacao.bagging, audios),
                                                   audios$label))))
  predicao.forest[i] <- sum(diag(prop.table(table(predict(aplicacao.forest, audios),
                                                  audios$label))))
  predicao.boosting[i] <- sum(diag(prop.table(table(predict(aplicacao.boosting,
                                                            audios2, type = "response",
                                                            n.trees = 1000)>.5,
                                                    audios2$label))))
}


respostas.audios <- data.frame(predicao.bagging, predicao.boosting,
                               predicao.forest)

list(Médias = apply(respostas.audios, 2, mean), DP = apply(respostas.audios, 2, sd))
```

```
## $Médias
##  predicao.bagging predicao.boosting   predicao.forest
##         0.7710714         0.7742857         0.7589286
##
## $DP
##  predicao.bagging predicao.boosting   predicao.forest
##        0.03628512        0.03993140        0.03061547
```

Observando as médias das proporções de predição e o desvio padrão dessas estimativas, observa-se que, em todos os modelos, gira em torno de 76% de predições corretas. Além disso, o desvio padrão é um pouco mais elevado que quando tentando prever para o banco *voices*. Como não sabemos como os áudios do banco *voices* foram coletados, podemos pensar que essa queda na predição pode se dar pela qualidade de áudios de WhatsApp, que pode ser inferior a dos áudios deste banco.

Conclusão

Como foi visto, o estudo da implementação dos métodos de classificação foi realizado para quatro tamanhos de bancos de dados treinos diferentes (50, 317, 1000, 1584) selecionados aleatóriamente 200 vezes cada. Os principais resultados foram retirados do fato de que a quantidade de informação presente nos dados que constituem o treino supervisionado é o fator que mais impactou na taxa de acerto. Enquanto isso, os diferentes métodos implementados não trouxeram grandes diferenças entre a média de acertos. Porém, mesmo levando em consideração a pouca diferença entre a quantidade de acertos, foi possível ver que o desempenho do método de *boosting* teve piores taxas quando o número de treinos foi muito pequeno.

Em relação à aplicação dos modelos construídos em áudios de conversas de WhatsApp, foi possível notar uma taxa de acerto inferior aos testes realizados anteriormente. Porém é importante levar em consideração que não houve um processo de amostragem para adquirir os áudios e, muito menos, foram seguidos os mesmos critérios para a construção do banco de dados *voices*. Por fim, nota-se que para os 28 áudios, os três métodos acertaram o sexo do locutor com mesma frequência.

Referências

TBSHIRANI, ROBERT (2013). *An Introduction to Statistical Learning*

BECKER, KORY (2016). *Identifying the Gender of a Voice using Machine Learning*
http://www.primaryobjects.com/2016/06/22/identifying-the-gender-of-a-voice-using-machine-learning/

FRIEDMAN, JEROME H. (1999). *Greedy Function Approximation: A Gradient Boosting Machine*

Códigos

```
# Códigos para montagem do banco com áudios do WhatsApp

setwd("/Users/pedrovianna/Documents/COMPUTACIONAL/Trabalho/Audios")

library(tuneR)
library(seewave)

xx <- list.files()

Audios <- data.frame()
for(i in 1:length(xx)){
  audio <- readMP3(xx[i])
  a <- spec(audio@right, f=audio@samp.rate)
  b <- specprop(a, f=audio@samp.rate, flim=c(0, 280/1000), plot = F)
  ee <- data.frame() #Criando as mesmas variáveis para o audio a ser testado
  ee[1,1] <- b$mean/1000
  ee[1,2] <- b$sd/1000
  ee[1,3] <- b$median/1000
  ee[1,4] <- b$Q25/1000
  ee[1,5] <- b$Q75/1000
  ee[1,6] <- b$IQR/1000
  ee[1,7] <- b$skewness
  ee[1,8] <- b$kurtosis
  ee[1,9] <- b$sh
  ee[1,10] <- b$sfm
  ee[1,11] <- b$mode/1000
  ee[1,12] <- b$cent/1000
  ee[1,13] <-  NA
  names(ee) <- names(voices)
  Audios <- rbind(ee, Audios)
}



Audios$label <- c("Male", "Male", "Female",
                  "Female", "Male", "Male", "Female", "Female",
                  "Female", "Female", "Female", "Male", "Male",
                  "Male", "Male", "Male", "Female", "Male",
                  "Male", "Male", "Male", "Male", "Male",
                  "Male", "Male", "Female", "Female", "Female")


write.csv(Audios, "/Users/pedrovianna/Documents/COMPUTACIONAL/Trabalho/audios.csv")




##### Códigos da Aplicação das árvores #####


setwd("/Users/pedrovianna/Documents/COMPUTACIONAL/Trabalho")
```

```r
options(scipen = 999)

library(ipred)
library(dplyr)
library(randomForest)
library(gbm)
library(rpart)
library(rattle)
library(tuneR)
library(seewave)
library(ggplot2)
library(tidyr)


voices <- read.csv("voices.csv")
voices <- voices[,-c(seq(13,20,1))]

audios <- read.csv("audios.csv")



# Influência Relativa

teste3 <- voices
teste3$label <- ifelse(teste3$label == "male", 1, 0)
treino3.v <- sample(1:nrow(voices), 317)
voices3 <- teste3[-treino3.v,]
treino3 <- teste3[treino3.v,]


boosting <-  gbm(label~., treino3, distribution="bernoulli", n.trees = 1000,
                 interaction.depth = 4)
importancia <- data.frame(summary(boosting, plot=F))

reorder(importancia$var, -importancia$rel.inf)

ggplot(importancia, aes(x=reorder(importancia$var, -importancia$rel.inf), y=rel.inf)) +
  geom_bar(stat="identity", position="dodge", fill=c("#fbcabb"))+
  labs(x="Variável", y="Influência Relativa")+
  theme_bw()+
  theme(axis.title.y=element_text(colour="black",size=12),
        axis.title.x=element_text(colour="black",size=12),
        axis.text=element_text(colour="black",size=9.5),
        panel.border=element_blank(),
        axis.line=element_line(colour="black"))


good.model <- gbm(label ~ IQR + Q25 + sd, treino3, distribution="bernoulli",
                  n.trees = 1000, interaction.depth = 4)
sum(diag(prop.table(table(predict(good.model, teste3, type = "response",
                                  n.trees = 1000)>.5, teste3$label))))
```

```r
bad.model <- gbm(label ~ centroid + kurt + meanfreq, treino3, distribution="bernoulli",
                     n.trees = 1000, interaction.depth = 4)
sum(diag(prop.table(table(predict(bad.model, voices3, type = "response",
                                  n.trees = 1000)>.5, voices3$label))))
```

```r
## Aplicações ##
```

```r
n <- nrow(voices)
```

```r
# 50

taxa_bag4 <- c()
taxa_forest4 <- c()
taxa_boost4 <- c()
t <- 50

for(i in 1:200){
  tt <- sample(1:n, 50)
  treino <- voices[tt, ]
  teste <- voices[-tt, ]
  bagging <- bagging(label ~ ., treino)
  forest <- randomForest(label ~ ., treino)


  teste2 <- teste
  treino2 <- treino
  a <- teste2$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  teste2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  boosting <-  gbm(label~., treino2, distribution="bernoulli", n.trees = 1000,
                   interaction.depth = 4)

  taxa_bag4[i] <- sum(diag(prop.table(table(predict(bagging,
                                                    teste,
                                                    type = "prob")[,2]>.5,
                                            teste$label))))

  taxa_forest4[i] <- sum(diag(prop.table(table(predict(forest,
                                                       teste,
                                                       type = "prob")[,2]>.5,
                                               teste$label))))
```

```
  taxa_boost4[i] <- sum(diag(prop.table(table(predict(boosting,
                                                      teste2,
                                                      type = "response",
                                                      n.trees = 1000)>.5,
                                              teste2$label))))


}

respostas4 <- data.frame(taxa_forest4,taxa_bag4,taxa_boost4)
apply(respostas4, 2, mean)
apply(respostas4, 2, sd)




# 317

t <- round(0.1*n)

taxa_bag <- c()
taxa_forest <- c()
taxa_boost <- c()



for(i in 1:200){
  tt <- sample(1:n, t)
  treino <- voices[tt, ]
  teste <- voices[-tt, ]
  bagging <- bagging(label ~ ., treino)
  forest <- randomForest(label ~ ., treino)


  teste2 <- teste
  treino2 <- treino
  a <- teste$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  teste2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  boosting <-  gbm(label~., treino2, distribution="bernoulli",
                   n.trees = 1000,
                   interaction.depth = 4)

  taxa_bag[i] <- sum(diag(prop.table(table(predict(bagging,
                                       teste,
                                       type = "prob")[,2]>.5,
                                    teste$label))))

  taxa_forest[i] <- sum(diag(prop.table(table(predict(forest, teste,
```

```
                                          type = "prob")[,2]>.5,
                                   teste$label))))

  taxa_boost[i] <- sum(diag(prop.table(table(predict(boosting,
                                          teste2,
                                          type = "response",
                                          n.trees = 1000)>.5,
                                      teste2$label))))

}

respostas1 <- data.frame(taxa_forest,taxa_bag,taxa_boost)
apply(respostas1, 2, mean)
apply(respostas1, 2, sd)




# 1000

taxa_bag2 <- c()
taxa_forest2 <- c()
taxa_boost2 <- c()


for(i in 1:200){
  tt <- sample(1:n, 1000)
  treino <- voices[tt, ]
  teste <- voices[-tt, ]
  bagging <- bagging(label ~ ., treino)
  forest <- randomForest(label ~ ., treino)


  teste2 <- teste
  treino2 <- treino
  a <- teste2$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  teste2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  boosting <-  gbm(label~., treino2, distribution="bernoulli",
                   n.trees = 1000, interaction.depth = 4)

  taxa_bag2[i] <- sum(diag(prop.table(table(predict(bagging,
                                          teste,
                                          type = "prob")[,2]>.5,
                                   teste$label))))

  taxa_forest2[i] <- sum(diag(prop.table(table(predict(forest,
                                          teste,
                                          type = "prob")[,2]>.5,
```

```
                                    teste$label))))

  taxa_boost2[i] <- sum(diag(prop.table(table(predict(boosting,
                                          teste2,
                                          type = "response",
                                          n.trees = 1000)>.5,
                                    teste2$label))))
}

respostas2 <- data.frame(taxa_forest2,taxa_bag2,taxa_boost2)
apply(respostas2, 2, mean)
apply(respostas2, 2, sd)



# 1584

taxa_bag3 <- c()
taxa_forest3 <- c()
taxa_boost3 <- c()
t <- round(0.5*n)

for(i in 1:200){
  tt <- sample(1:n, t)
  treino <- voices[tt, ]
  teste <- voices[-tt, ]
  bagging <- bagging(label ~ ., treino)
  forest <- randomForest(label ~ ., treino)


  teste2 <- teste
  treino2 <- treino
  a <- teste2$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  teste2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  boosting <-  gbm(label~., treino2, distribution="bernoulli", n.trees = 1000,
                   interaction.depth = 4)

  taxa_bag3[i] <- sum(diag(prop.table(table(predict(bagging,
                                                    teste,
                                                    type = "prob")[,2]>.5,
                                             teste$label))))

  taxa_forest3[i] <- sum(diag(prop.table(table(predict(forest,
                                                       teste,
                                                       type = "prob")[,2]>.5,
                                                teste$label))))

  taxa_boost3[i] <- sum(diag(prop.table(table(predict(boosting,
```

```
                                                    teste2,
                                                    type = "response",
                                                    n.trees = 1000)>.5,
                                            teste2$label))))


}

respostas3 <- data.frame(taxa_forest3,taxa_bag3,taxa_boost3)
apply(respostas3, 2, mean)
apply(respostas3, 2, sd)




respostas1t <- gather(respostas1, "grupo", dados)
respostas1t$teste <- 2
respostas2t <- gather(respostas2, "grupo", dados)
respostas2t$teste <- 3
respostas3t <- gather(respostas3, "grupo", dados)
respostas3t$teste <- 4
respostas4t <- gather(respostas4, "grupo", dados)
respostas4t$teste <- 1


respostas2t$grupo <- gsub("2", "", respostas2t$grupo)
respostas3t$grupo <- gsub("3", "", respostas3t$grupo)
respostas4t$grupo <- gsub("4", "", respostas4t$grupo)



all.table <- rbind(respostas1t,respostas2t,respostas3t, respostas4t)
all.table$teste <- as.factor(all.table$teste)


ggplot(all.table, aes(x=as.character(grupo), y=dados, fill=as.character(teste))) +
  geom_bar(stat="identity", position="dodge")+
  scale_fill_manual(name="Tamanho do treino", values=c("#a2c8ec","#ceb6d9",
                                                       "#f9b0c3", "#fbcabb"),
                                                       labels = c("50", "317 (30%)",
                                                                  "1000","1584 (50%)"))+
  scale_x_discrete(labels = c("Bagging","Boosting","Random Forest"))+
  labs(x="Método", y="Taxa de acerto")+
  theme_bw()+
  theme(axis.title.y=element_text(colour="black",size=12),
        axis.title.x=element_text(colour="black",size=12),
        axis.text=element_text(colour="black",size=9.5),
        panel.border=element_blank(),
        axis.line=element_line(colour="black"))+
  scale_y_continuous(breaks = seq(0, 1, 0.1))
```

```r
### Utilizando áudios de Whatsapp

predicao.bagging <- c()
predicao.boosting <- c()
predicao.forest <- c()


for(i in 1:100){
  amostra.treino <- voices[sample(1:n, round(0.3*n)), ]


  aplicacao.bagging <- bagging(label ~ ., amostra.treino)
  aplicacao.forest <- randomForest(label ~ ., amostra.treino)


  audios2 <- audios
  treino2 <- amostra.treino
  a <- audios2$label
  levels(a) <- c(0,1) #0=female 1=male
  b <- treino2$label
  levels(b) <- c(0,1)
  audios2$label <- as.numeric(levels(a)[a])
  treino2$label <- as.numeric(levels(b)[b])

  aplicacao.boosting <-  gbm(label~., treino2, distribution="bernoulli", n.trees = 1000,
                             interaction.depth = 4)



  predicao.bagging[i] <- sum(diag(prop.table(table(predict(aplicacao.bagging,
                                                           audios), audios$label))))
  predicao.forest[i] <- sum(diag(prop.table(table(predict(aplicacao.forest,
                                                          audios), audios$label))))
  predicao.boosting[i] <- sum(diag(prop.table(table(predict(aplicacao.boosting,
                                                            audios2, type = "response",
                                                            n.trees = 1000)>.5,
                                                    audios2$label))))

}


respostas.audios <- data.frame(predicao.bagging, predicao.boosting,
                               predicao.forest)

list(Médias = apply(respostas.audios, 2, mean), DP = apply(respostas.audios, 2, sd))
```