Bagging e Floresta Aleatória

Tiago Mendonça dos Santos

tiagoms.com
tiagomendonca
tiagoms1@insper.edu.br

Introdução

Árvores

Aspectos Positivos

- Fácil de explicar (muito mais que regressão linear)
- Podem ser apresentadas graficamente e facilmente interpretadas por pessoas que não são especialistas no assunto
- Tratam facilmente preditores qualitativos, sem a necessidade da criação de variáveis indicadoras / *dummies*
- Não é sensível a escala como outros métodos

Aspectos Negativos

- Uma pequena alteração nos dados pode causar uma grande alteração na árvore estimada (variância alta)
- Previsões baseadas em regiões retangulares
- Não apresentam desempenho preditivo tão bom quanto outros métodos

Árvores

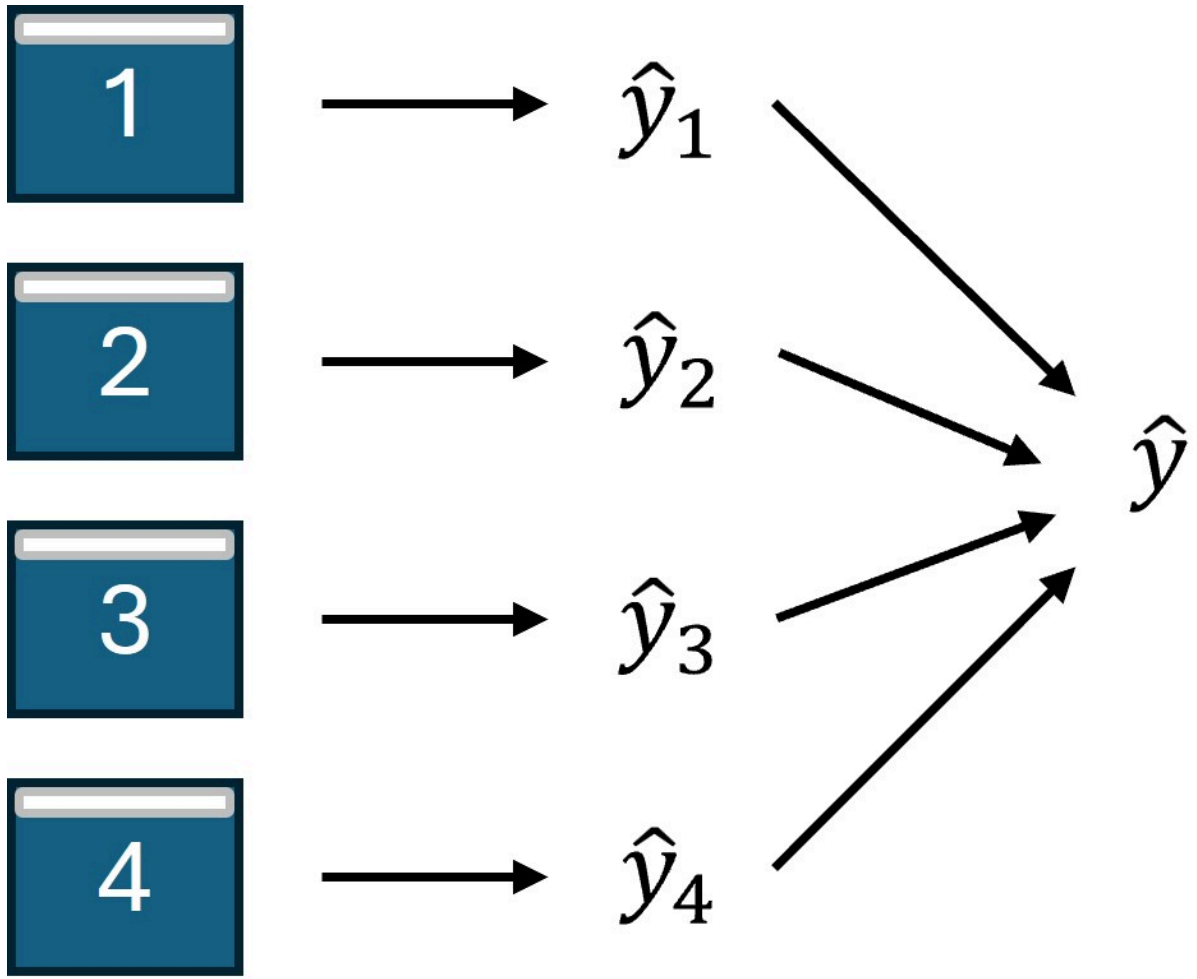
Aspectos Positivos

- Fácil de explicar (muito mais que regressão linear)
- Podem ser apresentadas graficamente e facilmente interpretadas por pessoas que não são especialistas no assunto
- Tratam facilmente preditores qualitativos, sem a necessidade da criação de variáveis indicadoras / *dummies*
- Não é sensível a escala como outros métodos

Aspectos Negativos

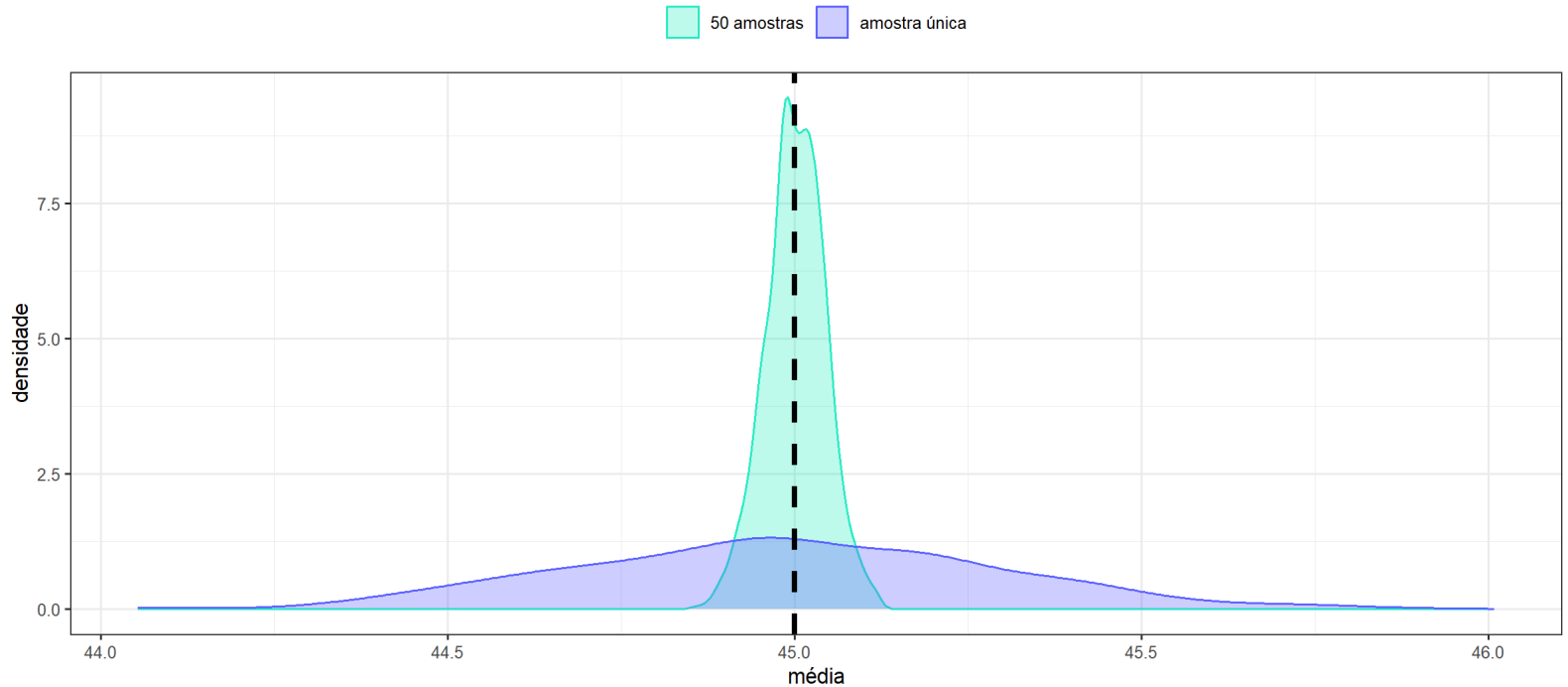
- **Uma pequena alteração nos dados pode causar uma grande alteração na árvore estimada (variância alta)**
- Previsões baseadas em regiões retangulares
- **Não apresentam desempenho preditivo tão bom quanto outros métodos**

Árvores



Árvores

Utilizar muitos conjuntos de dados e treinar o regressor com cada conjunto poderia reduzir a variância do método. As respostas seriam tomadas em média como sendo a resposta resultante do conjunto (*ensemble*) de regressores (daí o termo "método de *ensemble*").



Resultado de 1.000 simulações com $X \sim N(45, 3^2)$ e $n = 100$.

Árvores

Qual seria o problema para aplicar esse método em uma situação típica?

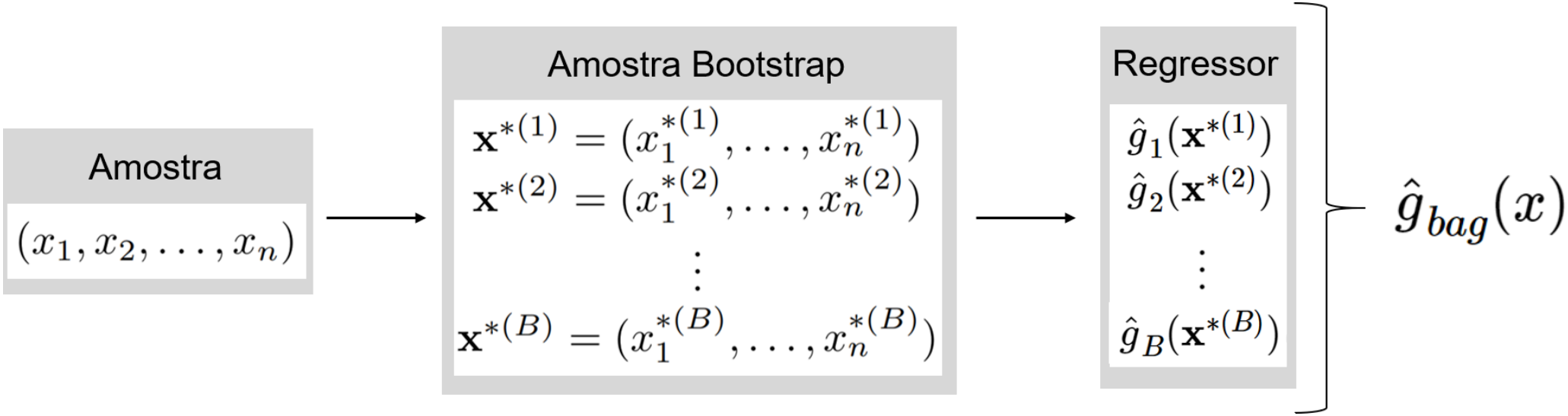
Em geral não temos acesso a múltiplos conjuntos de dados de treinamento. A forma de contornar esse problema é trabalhar com amostras *bootstrap*.

Agregando-se muitas árvores de decisão com métodos como o *bagging* (agregação *bootstrap*), a performance preditiva das árvores pode ser melhorada substancialmente (método proposto por Leo Breiman em 1996).

Leo Breiman, 1973

obs: *bagging* é um procedimento geral, não restrito apenas a árvores de classificação e regressão, cujo propósito é reduzir a variância de um método de aprendizagem.

Bagging



A partir de B amostras *bootstrap*, temos que

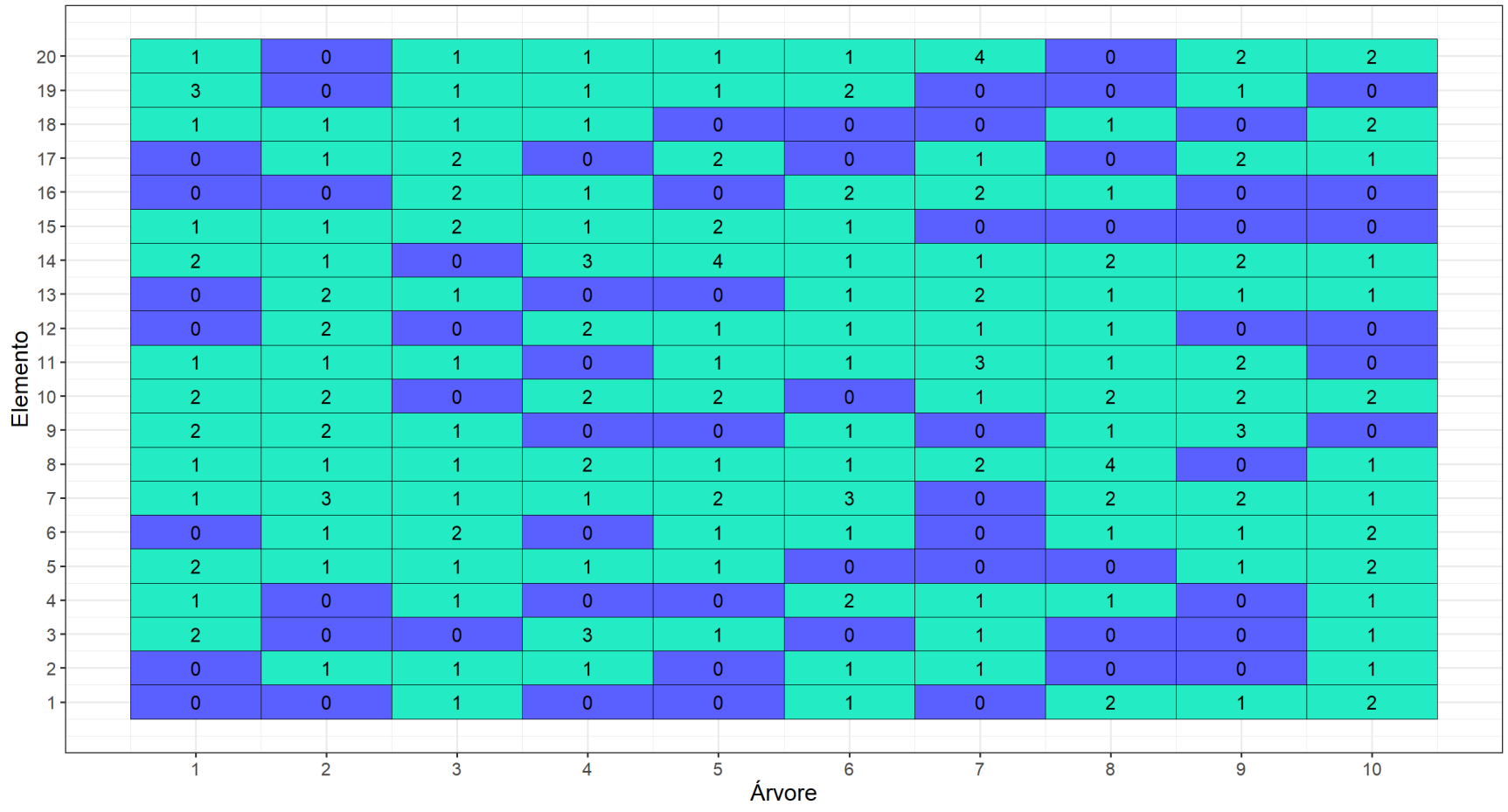
$$\hat{g}_{\text{bag}}(x) = \frac{1}{B}\sum_{i=1}^{B} \hat{g}_i(x).$$

Crescemos árvores bem altas, que, individualmente, terão grande variância e viés baixo. **O processo de *bagging* cuida da redução da variância**.

Para classificação, $\hat{g}_{\text{bag}}(x)$ seria dado pelo "voto da maioria" dos $\hat{g}_i(x)$.

Out-of-Bag

A seguir são apresentadas as frequências com que cada observação foi sorteada para criar as respectivas árvores.

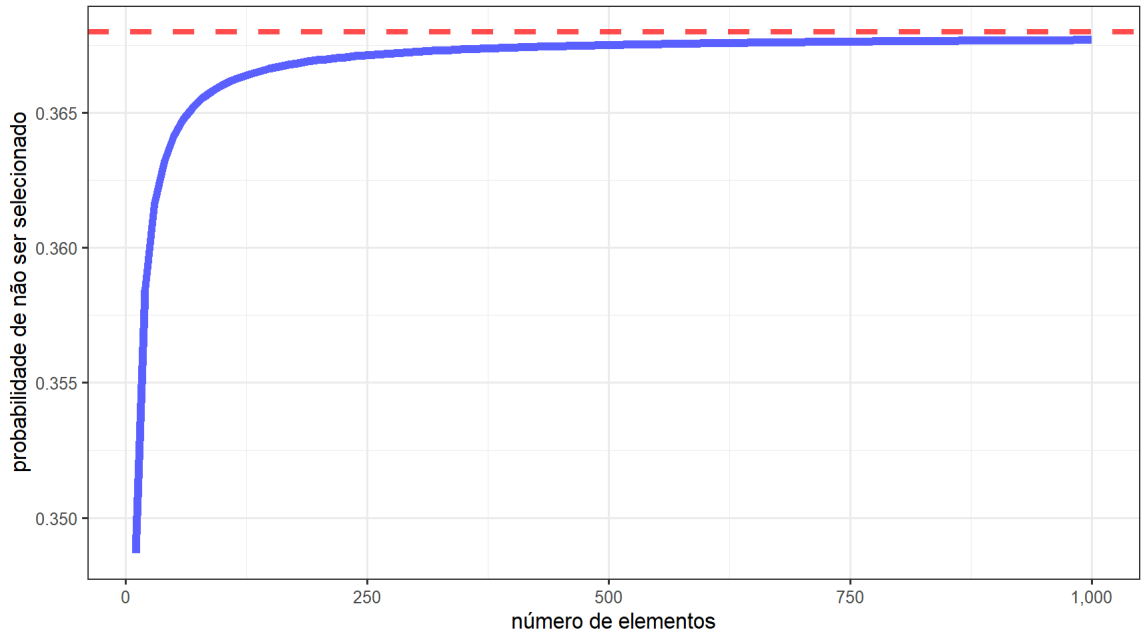


Erro *Out-of-Bag*

Forma simples de estimar o erro de teste sem precisarmos recorrer a procedimentos de validação cruzada.

Cada árvore crescida durante o procedimento do *bagging*, em média, utiliza aproximadamente dois terços[1] dos dados originais de treinamento.

Os dados do um terço remanescente de uma determinada árvore do *ensemble* são denominados observações *out-of-bag* (fora-da-sacola) da árvore em questão.



[1] retiradas do livro *An Introduction to Statistical Learning with Applications in R*.

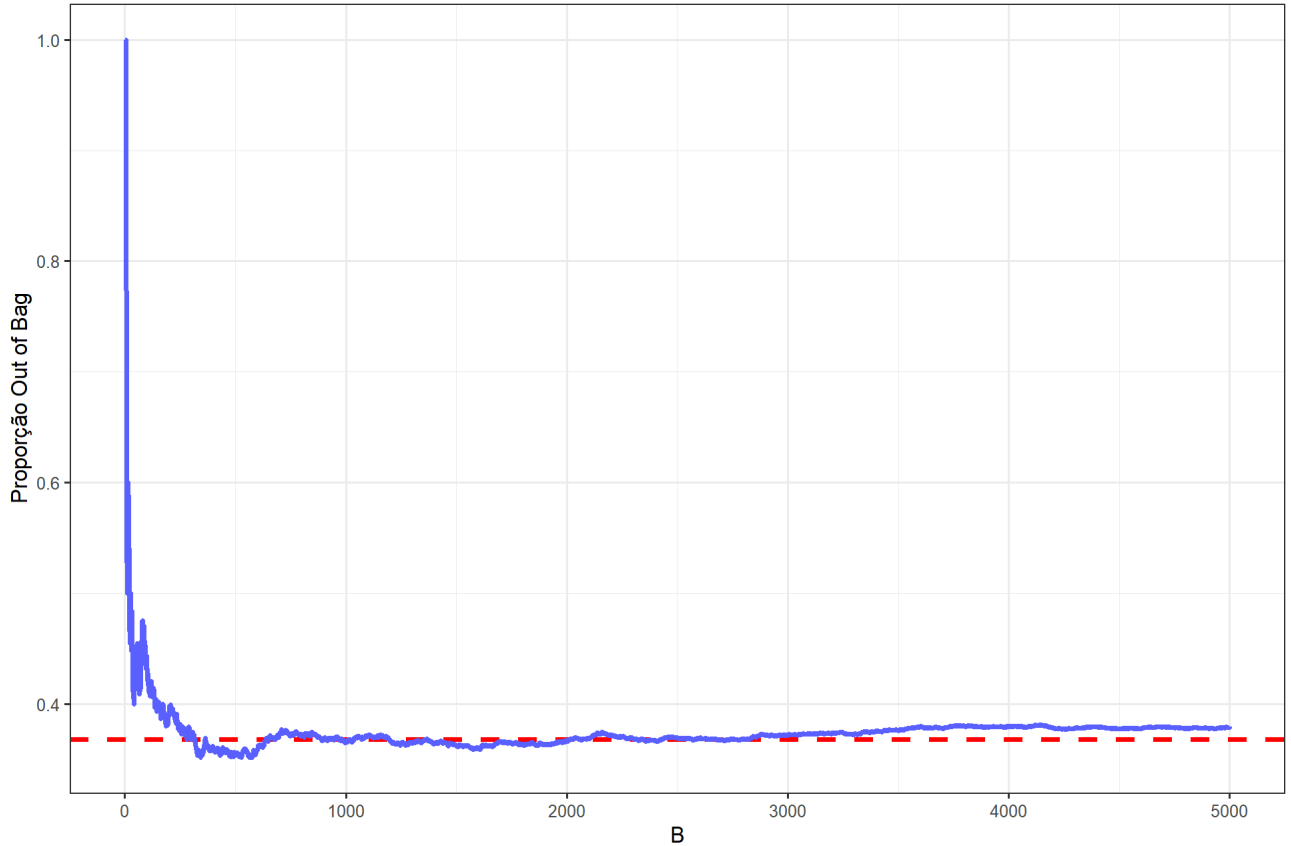
Erro *Out-of-Bag*

```
library(tidyverse)

dados <- tibble(B = 1:5000, out = NA)

pop <- 1:500

indicadora <- function(x, populacao){

  amostra <- sample(populacao, length(populacao), replace = TRUE)

  return(!any(amostra == x))
}


for(i in 1:nrow(dados)) dados$out[i] <- indicadora(15, pop)

dados %>%
  mutate(prop_out = cumsum(out)/B) %>%
  ggplot(aes(B, prop_out)) +
    geom_hline(yintercept = 1/exp(1), linetype = "dashed", color = "red", size = 1.2) +
    geom_line(size = 1.2, color = "#5B5FFF") +
    labs(x = "B", y = "Proporção Out of Bag") +
    theme_bw()
```

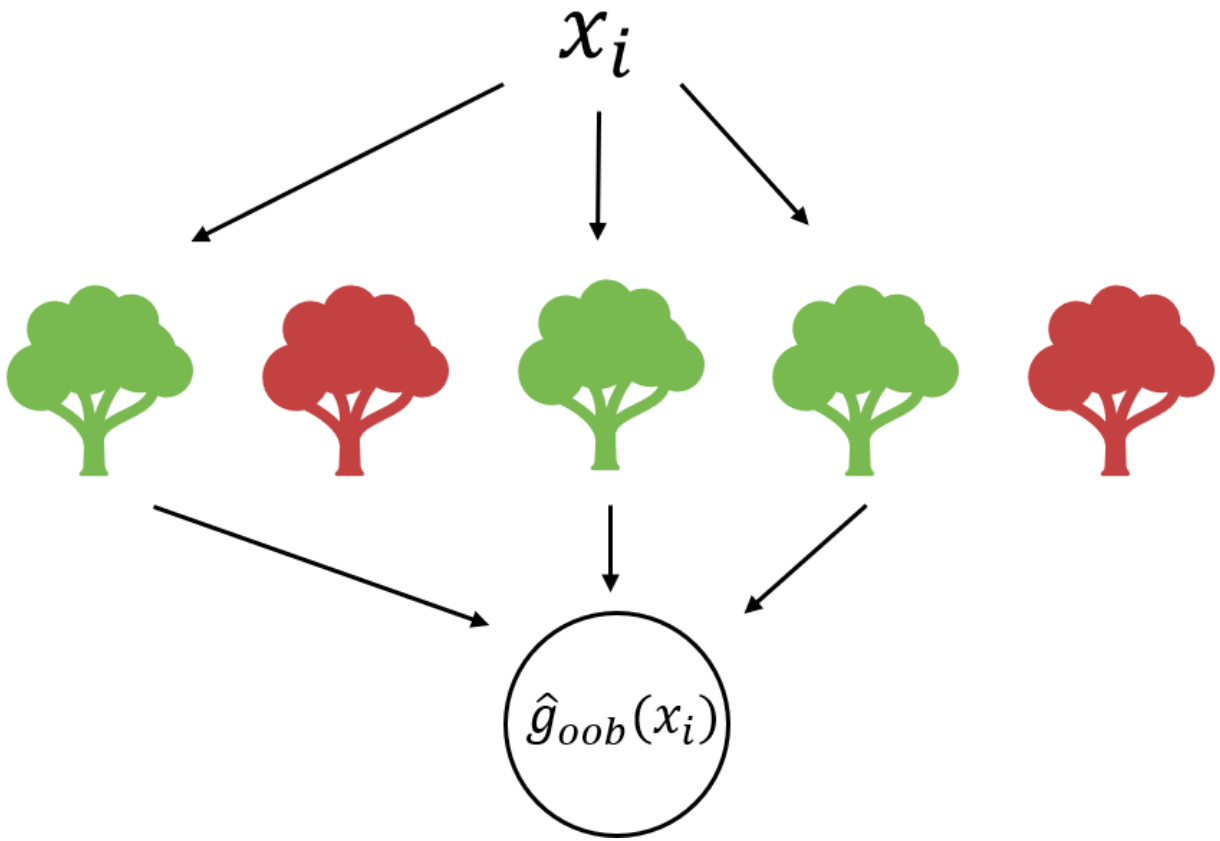
Erro *Out-of-Bag* (outra forma)

```
pop <- 1:500

dados <- tibble(B = 1:5000, out = NA) %>%
  mutate(out = map_lgl(B, ~ !(15 %in% sample(pop, 500, replace = TRUE))))

dados %>%
  mutate(prop_out = cumsum(out)/B) %>%
  ggplot(aes(B, prop_out)) +
    geom_hline(yintercept = 1/exp(1), linetype = "dashed", color = "red", size = 1.2) +
    geom_line(size = 1.2, color = "#5B5FFF") +
    labs(x = "B", y = "Proporção Out of Bag") +
    theme_bw()
```

Erro *Out-of-Bag*



Erro *Out-of-Bag*



Podemos prever a resposta do i-ésimo dado de treinamento utilizando todas as árvores do *ensemble* nas quais este dado pertence às observações *out-of-bag*.

Fazendo isto para cada um dos dados de treinamento obtemos uma estimativa válida do erro de teste esperado.

Housing Values in Suburbs of Boston[1]

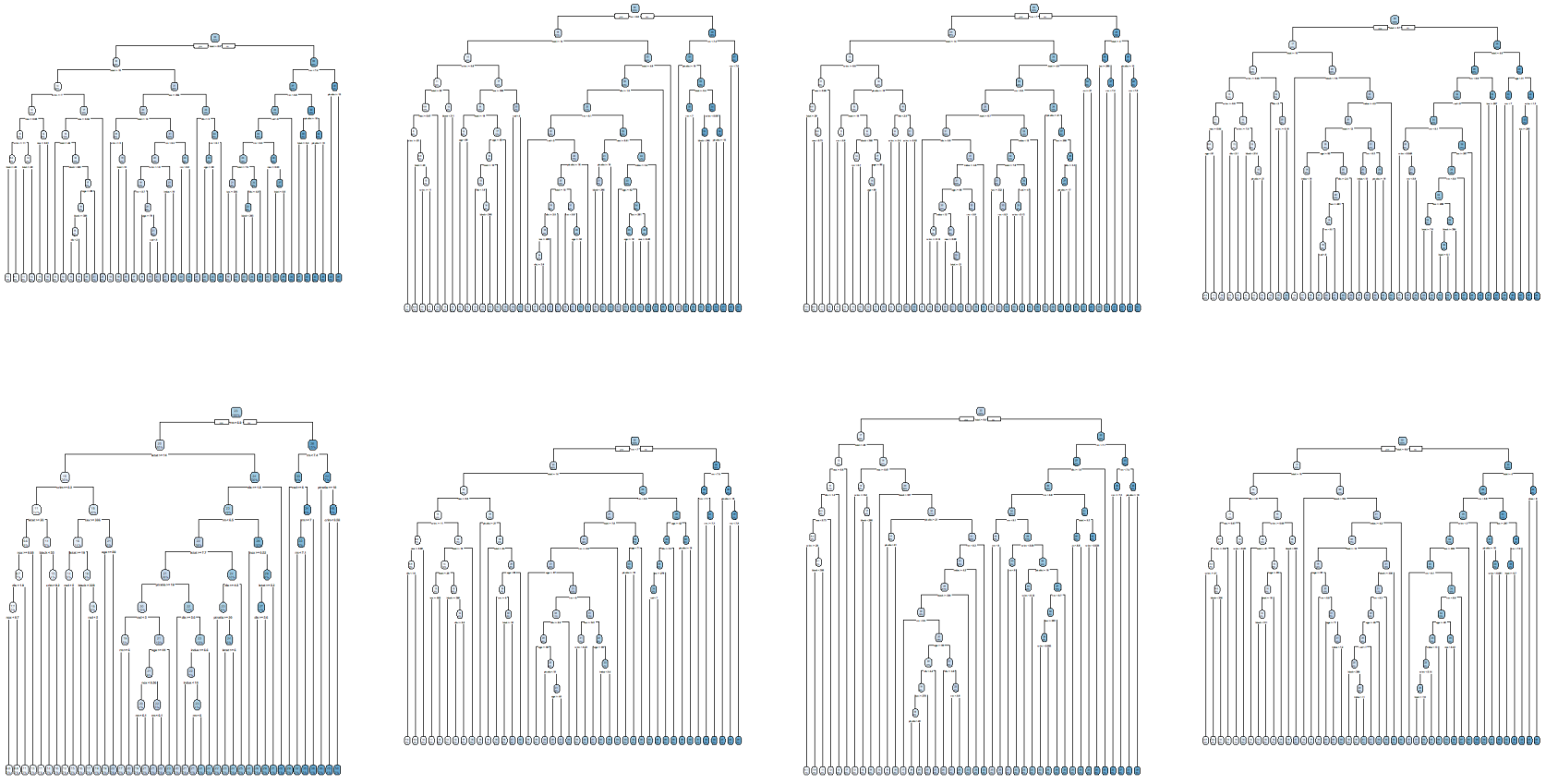
- **crim**: per capita crime rate by town.
- **zn**: proportion of residential land zoned for lots over 25,000 sq.ft.
- **indus**: proportion of non-retail business acres per town.
- **chas**: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- **nox**: nitrogen oxides concentration (parts per 10 million).
- **rm**: average number of rooms per dwelling.
- **age**: proportion of owner-occupied units built prior to 1940.
- **dis**: weighted mean of distances to five Boston employment centres.
- **rad**: index of accessibility to radial highways.
- **tax**: full-value property-tax rate per $10,000.
- **ptratio**: pupil-teacher ratio by town.
- **black**: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
- **lstat**: lower status of the population (percent).
- **medv**: median value of owner-occupied homes in $1000s.

[1] esses dados podem ser encontrados no pacote MASS.

Bagging

```
library(MASS)
library(rpart.plot)

data(Boston)

set.seed(123)

par(mfrow = c(2, 4))

for(i in 1:8){

  amostra <- sample(nrow(Boston), size = nrow(Boston), replace = TRUE)

  arvore <- rpart(medv ~ ., data = Boston[amostra, ], control = rpart.control(cp = 0))

  rpart.plot(arvore)
}
```

Bagging



Floresta Aleatória

Bagging

Split 1	Split 2	...	Split S
crim	crim	...	crim
zo	zo	...	zo
indus	indus	...	indus
chas	chas	...	chas
nox	nox	...	nox
rm	rm	...	rm
age	age	...	age
Dis	Dis	...	Dis
rad	rad	...	rad
tax	tax	...	tax
pratio	pratio	...	pratio
black	black	...	black
lstat	lstat	...	lstat

Floresta Aleatória

Split 1	Split 2	...	Split S
crim	crim	...	crim
zo	zo	...	zo
indus	indus	...	indus
chas	chas	...	chas
nox	nox	...	nox
rm	rm	...	rm
age	age	...	age
Dis	Dis	...	Dis
rad	rad	...	rad
tax	tax	...	tax
pratio	pratio	...	pratio
black	black	...	black
lstat	lstat	...	lstat

Floresta Aleatória[1]

1. Para $b = 1, \ldots, B$:
    1. Retire uma amostra *bootstrap* $\mathbf{Z}^*$ de tamanho N do conjunto de treinamento.
    2. Cresça uma árvore T_b a partir da amostra *bootstrap*, repetindo recursivamente os seguintes passos para cada nó terminal da árvore até alcançar o número mínimo de observações n_{min} de cada nó.
        1. Selecione m variáveis aleatoriamente entre as p variáveis.
        2. Escolha a melhor variável para a divisão/*split* entre as m.
        3. Divida o nó em dois nós descendentes.
2. Retorne o comitê de árvores $\{T_b\}_{b=1}^{B}$

Para fazer uma previsão para uma dada observação $\mathbf{x}$:

Regressão: $\hat{f}^B = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x})$

Classificação: Seja $\hat{C}_b(\mathbf{x})$ a classe predita pela b-ésima árvore. Então, $\hat{C}^B(\mathbf{x}) =$ voto da maioria $\left\{\hat{C}_b(\mathbf{x})\right\}_{b=1}^{B}$.

[1] descrição apresentada no livro *An Introduction to Statistical Learning with Applications in R*.

Hiperparâmetros

Número de preditoras a ser considerado em cada divisão/*split*

Para **classificação** recomenda-se utilizar $\sqrt{p}$ e número mínimo de observações por nó igual a um. Já para **regressão** recomenda-se utilizar $p/3$ e número mínimo de observações por nó igual a 5.

Número de árvores da floresta

Tipicamente se utiliza de 500 a 1.000 árvores. É importante notar que, geralmente, aumentar B não causa sobreajuste dos dados.

Árvore x Floresta Aleatória

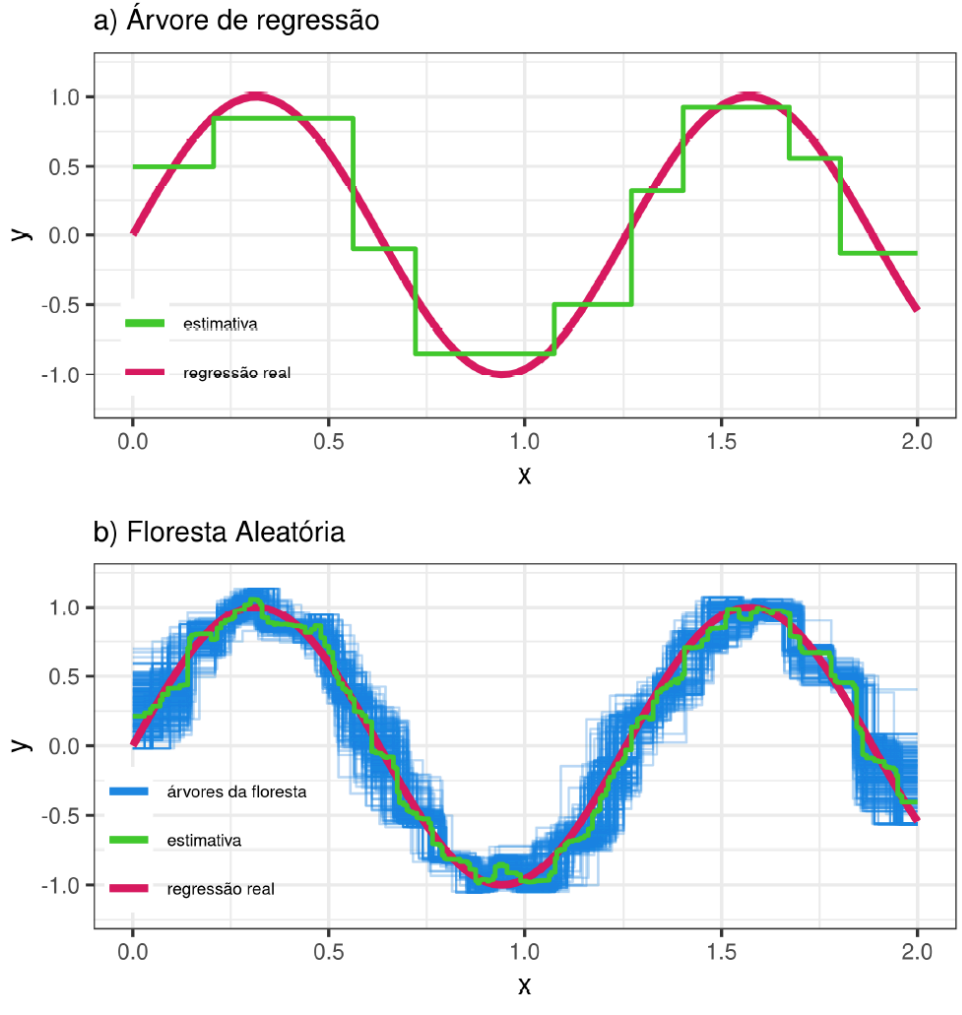


Figura retirada do livro Aprendizado de máquina: uma abordagem estatística.

Árvore x Floresta Aleatória

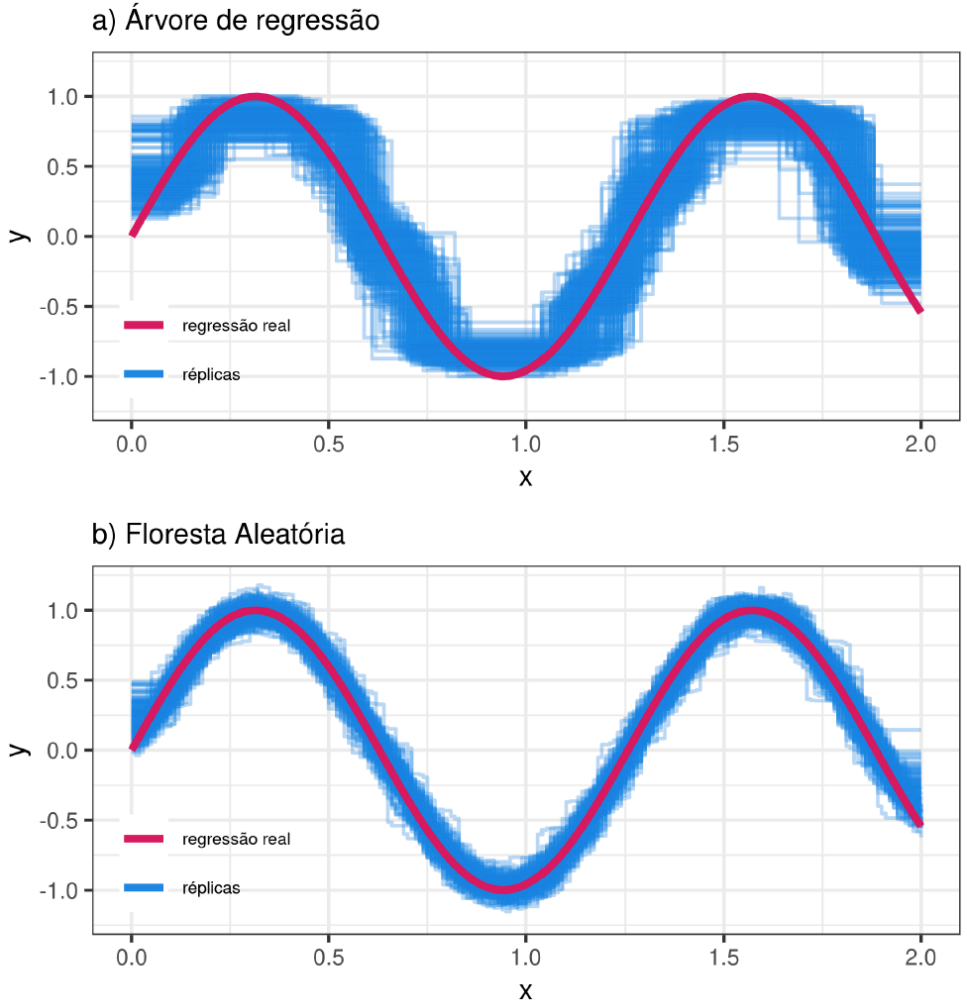


Figura retirada do livro Aprendizado de máquina: uma abordagem estatística.

Housing Values in Suburbs of Boston

```
library(MASS); library(ranger); data(Boston)

set.seed(123)

(rf <- ranger(medv ~ ., data = Boston))
```

Copy Code

```
## Ranger result
##
## Call:
##  ranger(medv ~ ., data = Boston)
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      506
## Number of independent variables:  13
## Mtry:                             3
## Target node size:                 5
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       10.73947
## R squared (OOB):                  0.873036
```

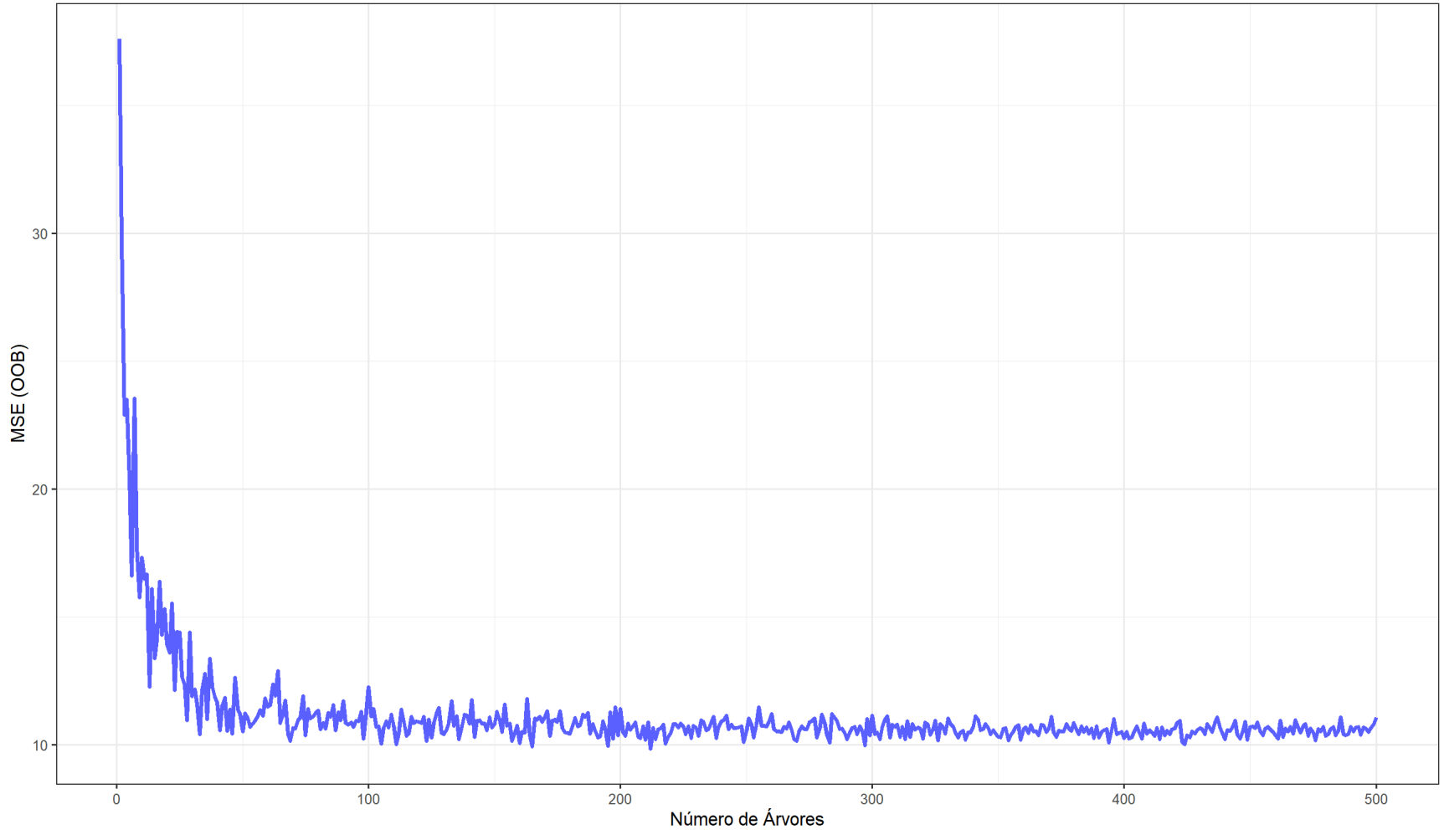
Mean of squared residuals (baseado em oob): `mean((Boston$medv - rf$predictions)^2)`

% Var explained: 1 - MSE/Var(y)

Housing Values in Suburbs of Boston

```
resultados <- tibble(n_arvores = 1:500,
                     mse = NA)


for (i in 1:nrow(resultados)) {

   rf <- ranger(medv ~ ., num.trees = resultados$n_arvores[i], data = Boston)

   resultados$mse[i] <- rf$prediction.error

}

resultados %>%
  ggplot(aes(n_arvores, mse)) +
    geom_line(color = "#5B5FFF", size = 1.2) +
    labs(x = "Número de Árvores", y = "MSE (OOB)") +
    theme_bw()
```

Housing Values in Suburbs of Boston



Housing Values in Suburbs of Boston

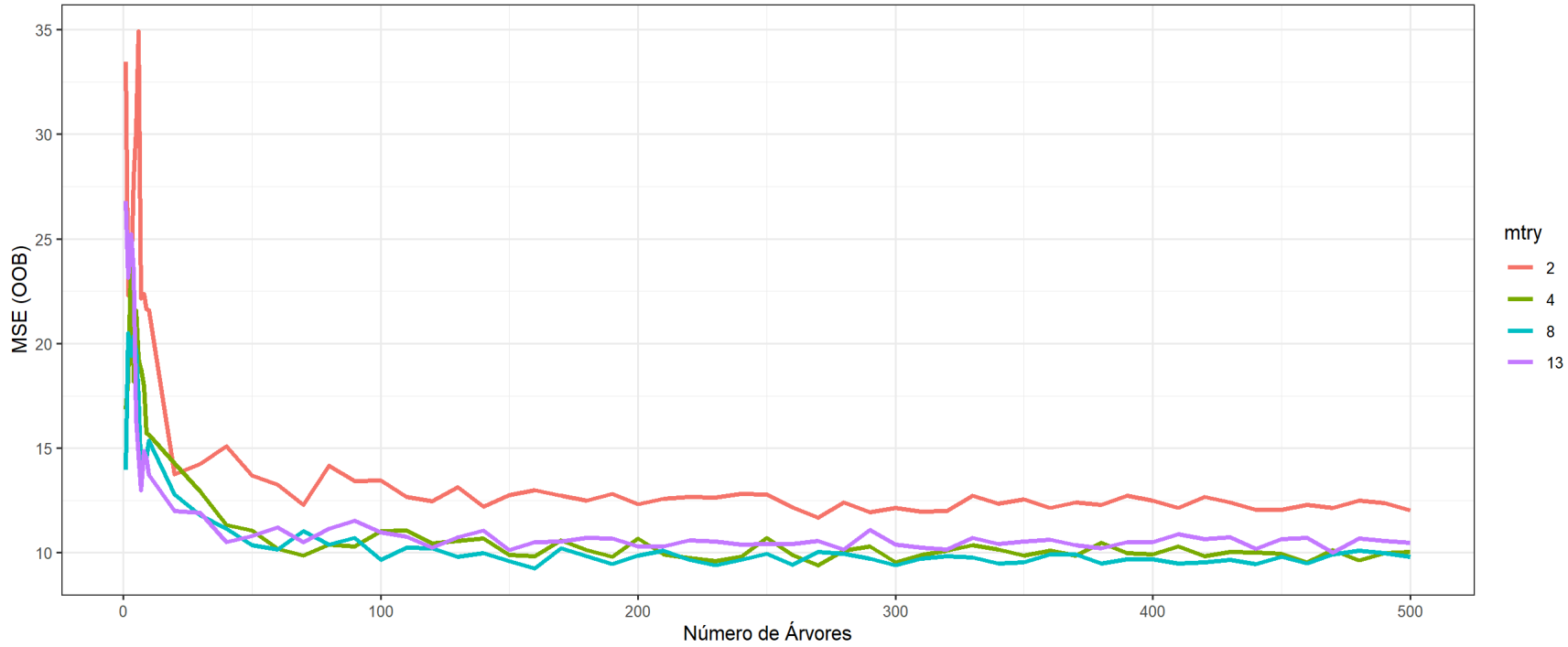
Insper

```
resultados <- crossing(mtry = c(2, 4, 8, 13),
                       n_arvores = c(1:10, seq(10, 500, 10)))


ajusta <- function(mtry, n_arvores) {

   rf <- ranger(medv ~ ., num.trees = n_arvores, mtry = mtry, data = Boston)

   return(rf$prediction.error)

}


resultados <- resultados %>%
  mutate(mse = map2_dbl(mtry, n_arvores, ajusta))

head(resultados)
```

```
## # A tibble: 6 × 3
##    mtry n_arvores   mse
##   <dbl>     <dbl> <dbl>
## 1     2         1  33.5
## 2     2         2  22.3
## 3     2         3  23.1
## 4     2         4  27.9
## 5     2         5  30.2
```

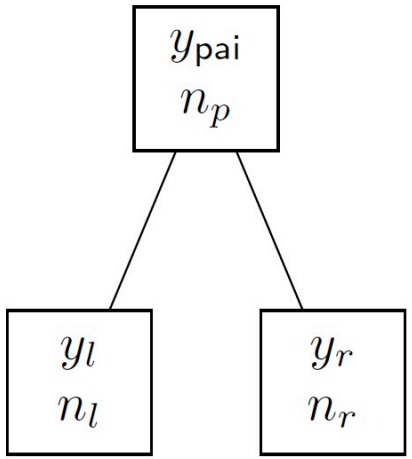
Housing Values in Suburbs of Boston

```
resultados %>%
  mutate(mtry = factor(mtry)) %>%
  ggplot(aes(n_arvores, mse, group = mtry, color = mtry)) +
    geom_line( size = 1.2) +
    labs(x = "Número de Árvores", y = "MSE (OOB)") +
    theme_bw()
```



Variable Importance

Verifica-se a redução total no RSS (*residual sum of squares*) devido a divisão/*split* relativa a uma dada preditora para cada árvore e calcula-se a média dessas reduções de acordo com número de árvores na floresta. Portanto, um valor alto indica que a preditora/variável é importante.

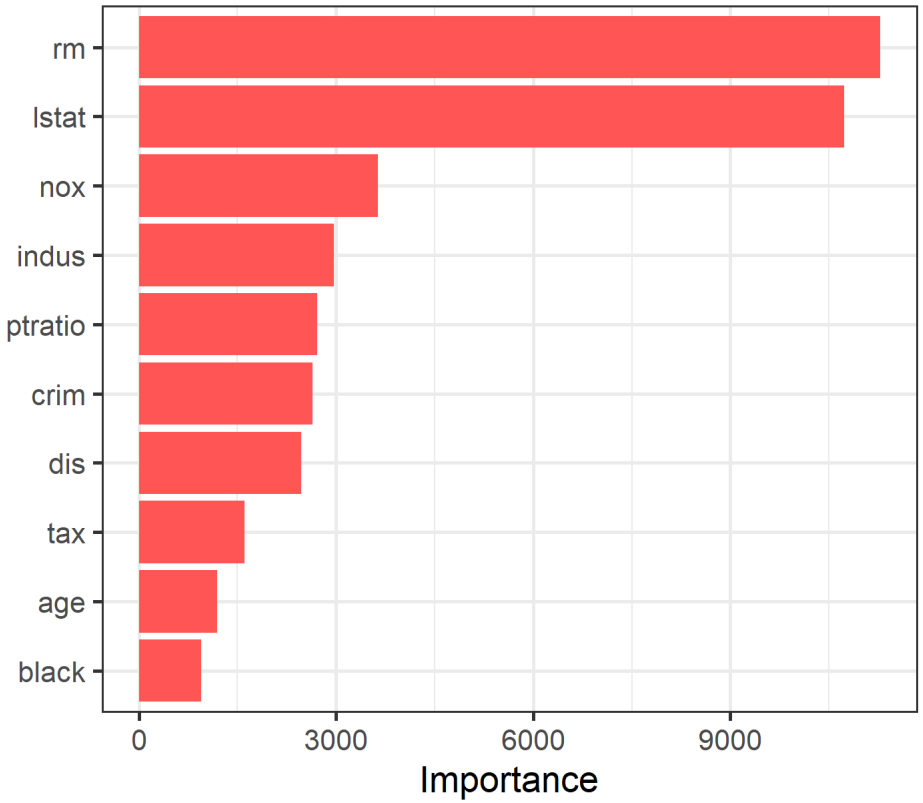


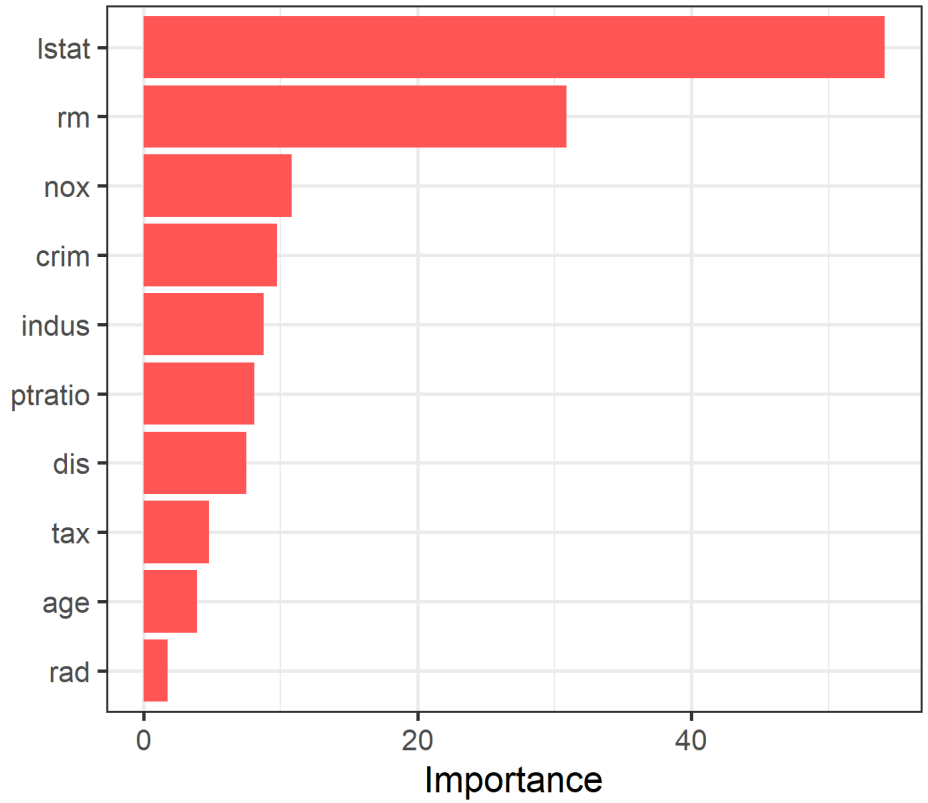
$$\text{RSS}_{\text{pai}} - \text{RSS}_{\text{l}} - \text{RSS}_{\text{r}}$$

$$\sum_{i=1}^{n_p}(y_{\text{pai, i}} - \bar{y}_{\text{pai, i}})^2 - \sum_{i=1}^{n_l}(y_{l,i} - \bar{y}_{l,i})^2 - \sum_{i=1}^{n_r}(y_{r,i} - \bar{y}_{r,i})^2$$

Housing Values in Suburbs of Boston

```
rf1 <- ranger(medv ~ ., importance = "impurity", data = Boston)
vip::vip(rf1, aesthetics = list(fill = "#FF5757"))

rf2 <- ranger(medv ~ ., importance = "permutation", data = Boston)
vip::vip(rf2, aesthetics = list(fill = "#FF5757"))
```





Churn[1]

Conjunto de 5.000 observações e 19 preditoras para churn.

- state
- account_length
- area_code
- international_plan
- voice_mail_plan
- number_vmail_messages
- total_day_minutes
- total_day_calls
- total_day_charge
- total_eve_minutes
- total_eve_calls
- total_eve_charge
- total_night_minutes
- total_night_calls
- total_night_charge
- total_intl_minutes
- total_intl_calls
- total_intl_charge
- number_customer_service_calls
- **churn**

[1] dados no pacote modeldata.

Churn

total_day_calls	total_day_charge	total_eve_minutes	total_eve_calls	total_eve_charge	total_night_minu
110	45.07	197.4	99	16.78	244.7
123	27.47	195.5	103	16.62	254.4
114	41.38	121.2	110	10.3	162.6
71	50.9	61.9	88	5.26	196.9
113	28.34	148.3	122	12.61	186.9
98	37.98	220.6	101	18.75	203.9
88	37.09	348.5	108	29.62	212.6
79	26.69	103.1	94	8.76	211.8
97	31.37	351.6	80	29.89	215.8
84	43.96	222	111	18.87	326.4

Churn

Para utilizar floresta aleatória com esses dados, considere o seguinte código

```
library(modeldata)
library(ranger)
library(rsample)

data(mlc_churn)

mlc_churn$churn <- factor(mlc_churn$churn, levels = c("no", "yes"))

set.seed(15)

splits <- initial_split(mlc_churn, prop = .9, strata = "churn")

treino <- training(splits)
teste <- testing(splits)

(rf <- ranger(churn ~ ., data = treino))
```

Churn

```
## Ranger result
##
## Call:
##  ranger(churn ~ ., data = treino)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      4499
## Number of independent variables:  19
## Mtry:                             4
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             4.07 %
```

```
rf$confusion.matrix
```

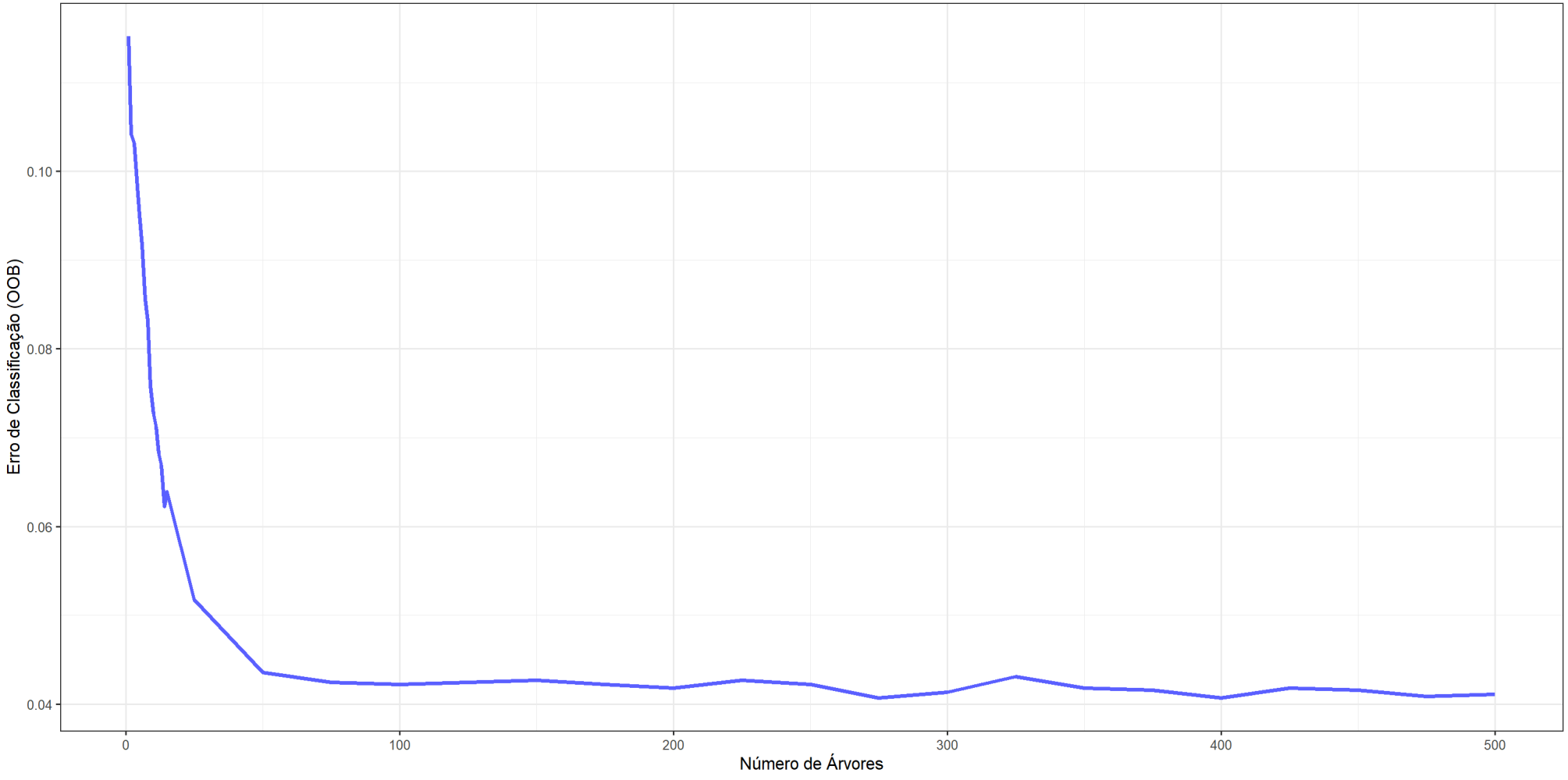
```
##      predicted
## true    no  yes
##   no  3838   25
##   yes  158  478
```

A matriz de confusão é baseada nos dados OOB. As linhas da matriz indicam a categoria da resposta observada e as colunas indicam as categorias classificadas.

Churn

```
resultados <- tibble(n_arvores = c(1:15, seq(25, 500, 25)),
                     erro = NA)

resultados <- resultados %>%
  mutate(erro = map_dbl(n_arvores, ~ranger(churn ~ ., num.trees = .x,
                                           data = treino)$prediction.error))

resultados %>%
  ggplot(aes(n_arvores, erro)) +
    geom_line(color = "#5B5FFF", size = 1.2) +
    labs(x = "Número de Árvores", y = "Erro de Classificação (OOB)") +
    theme_bw()
```

Churn


0.10
0.08
0.06
0.04
Erro de Classificação (OOB)
0
100
200
300
400
500
Número de Árvores

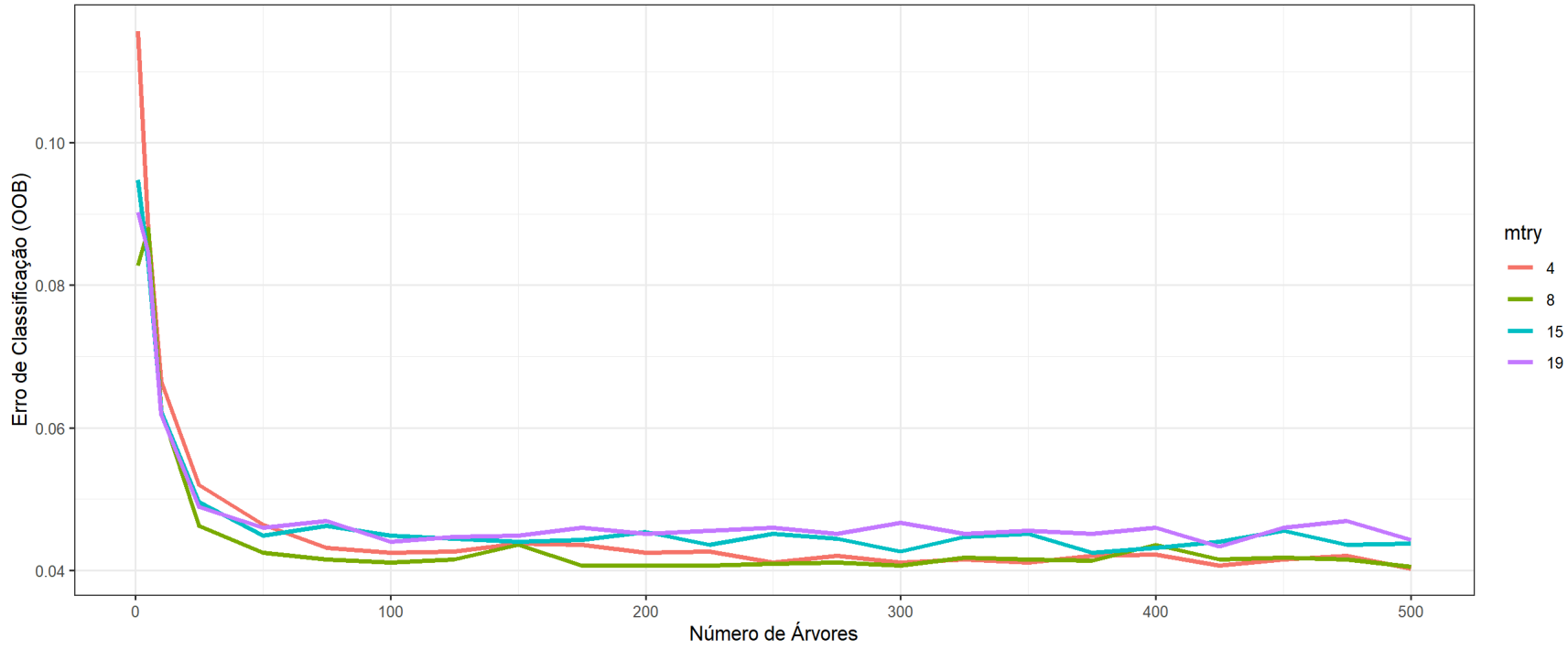
Churn

```
resultados <- crossing(mtry = c(4, 8, 15, 19),
                       n_arvores = c(1, 5, 10, seq(25, 500, 25)))


ajusta <- function(mtry, n_arvores) {

   rf <- ranger(churn ~ ., num.trees = n_arvores, mtry = mtry, data = treino)

   return(rf$prediction.error)

}


resultados <- resultados %>%
  mutate(erro = map2_dbl(mtry, n_arvores, ajusta))

head(resultados)
```

```
## # A tibble: 6 × 3
##    mtry n_arvores   erro
##   <dbl>     <dbl>  <dbl>
## 1     4         1 0.116
## 2     4         5 0.0877
## 3     4        10 0.0665
## 4     4        25 0.0520
## 5     4        50 0.0465
```

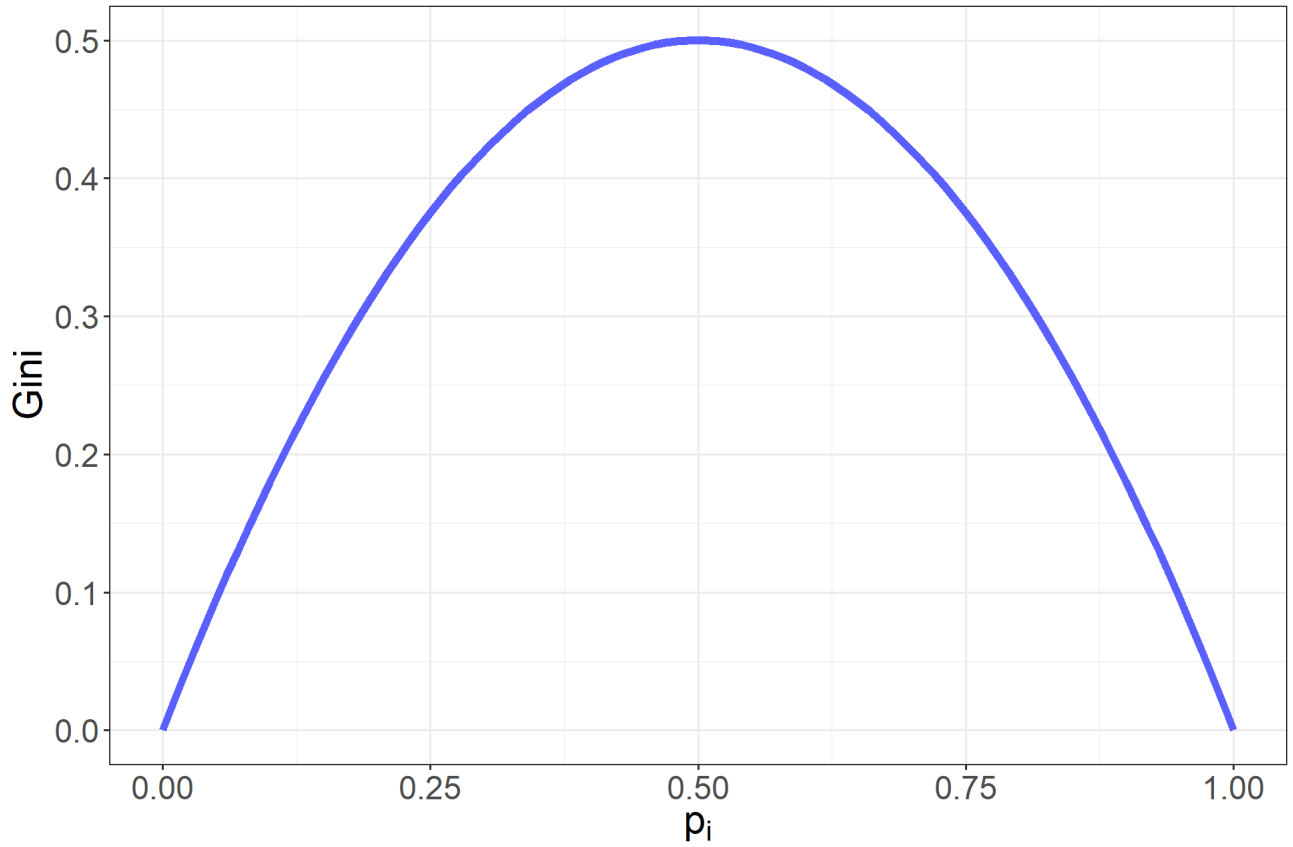
Churn

```
resultados %>%
  mutate(mtry = factor(mtry)) %>%
  ggplot(aes(n_arvores, erro, group = mtry, color = mtry)) +
    geom_line( size = 1.2) +
    labs(x = "Número de Árvores", y = "Erro de Classificação (OOB)") +
    theme_bw()
```



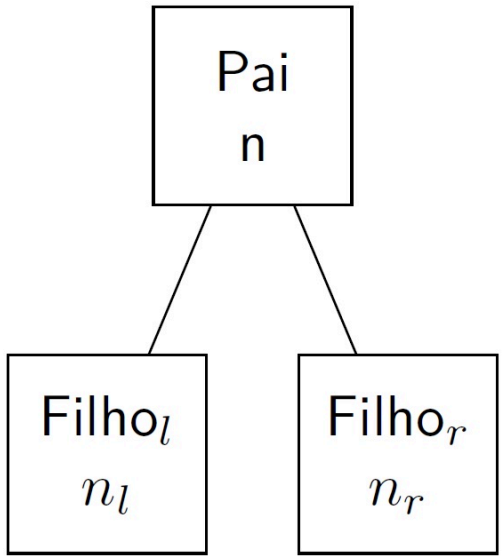
Gini Index

$$\text{Gini(node)} = \sum_{i=1}^{\text{classes}} p_i(1 - p_i)$$



Variable Importance

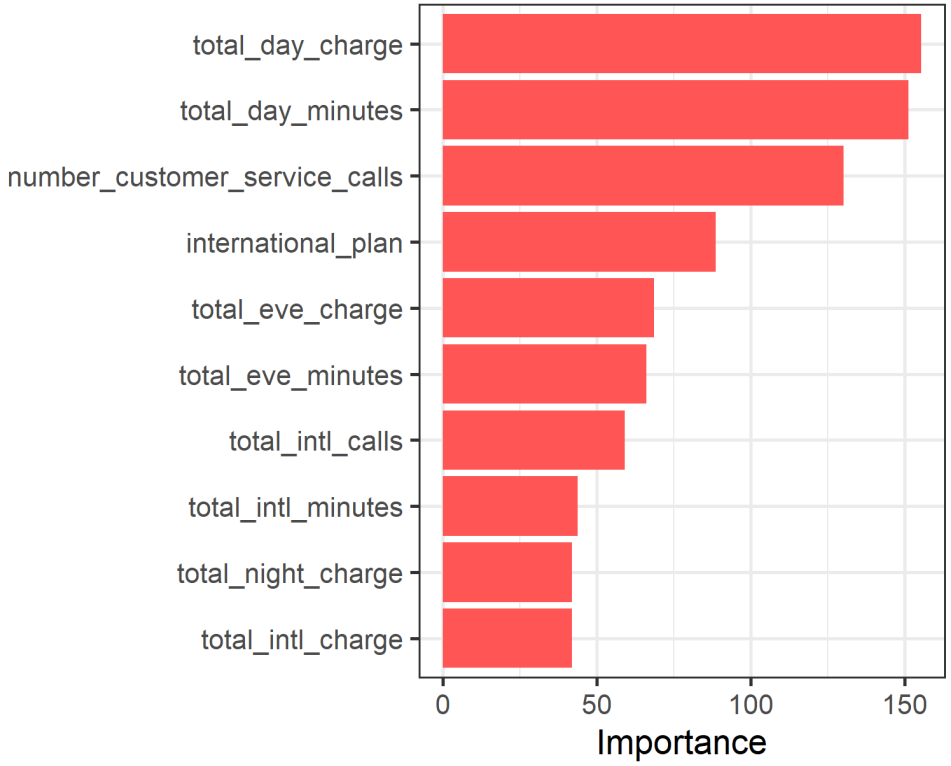
Verifica-se a redução total no índice de Gini para cada preditora com as observações utilizadas na construção da árvore (*in bag*) e calcula-se a média de acordo com o número de árvores na floresta. A redução é ponderada pelo número de observações no nó ancestral e descendentes.

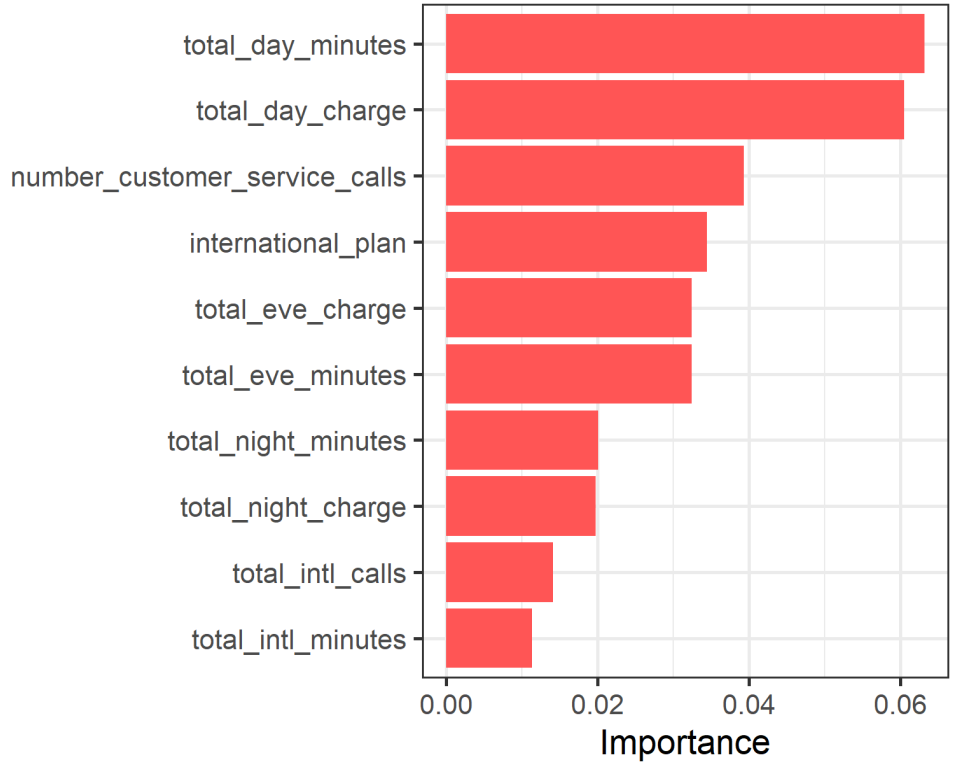


$$n \times \text{Gini}(\text{Pai}) - n_l \times \text{Gini}(\text{Filho}_l) - n_r \times \text{Gini}(\text{Filho}_r)$$

Churn

```
rf1 <- ranger(churn ~ ., importance = "impurity", data = treino)
vip::vip(rf1, aesthetics = list(fill = "#FF5757"))

rf2 <- ranger(churn ~ ., importance = "permutation", data = treino)
vip::vip(rf2, aesthetics = list(fill = "#FF5757"))
```





Churn

```
rf <- ranger(churn ~ ., data = treino)

head(predict(rf, teste, type = "response")$predictions)
```

```
## [1] no  yes no  no  no  no
## Levels: no yes
```

```
rf <- ranger(churn ~ ., probability = TRUE, data = treino)

head(predict(rf, teste, type = "response")$predictions)
```

```
##             no        yes
## [1,] 0.7724500 0.22755000
## [2,] 0.4360992 0.56390079
## [3,] 0.9534175 0.04658254
## [4,] 0.9431460 0.05685397
## [5,] 0.8970754 0.10292460
## [6,] 0.5963706 0.40362937
```

Churn

```
library(pROC)

prob_churn <- predict(rf, teste, type = "response")$predictions[,2]

table(observado = teste$churn,
      predito = ifelse(prob_churn >= .5, "yes", "no"))
```

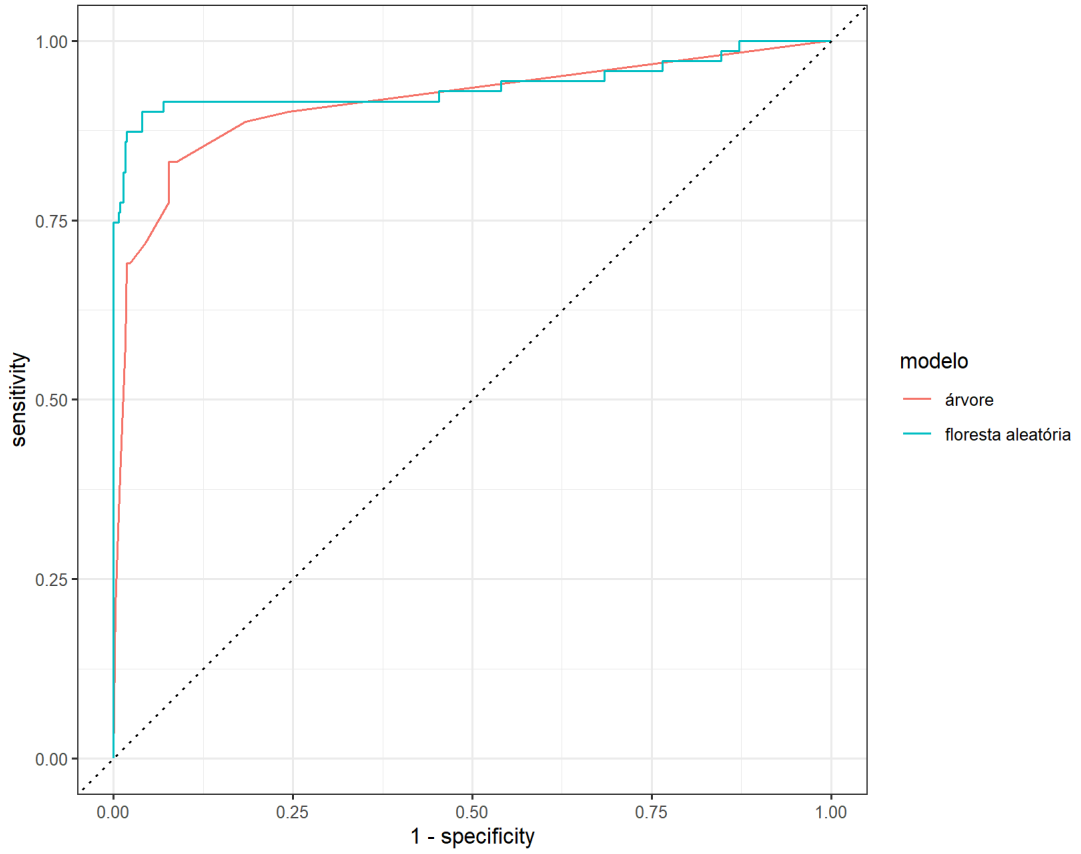
```
##          predito
## observado  no yes
##       no  425   5
##       yes  16  55
```

```
desempenho <- roc(teste$churn, prob_churn)

coords(desempenho, .5, ret = c("1-accuracy", "sensitivity", "specificity", "ppv", "npv"))
```

```
##           1-accuracy sensitivity specificity       ppv       npv
## threshold 0.04191617   0.7746479   0.9883721 0.9166667 0.9637188
```

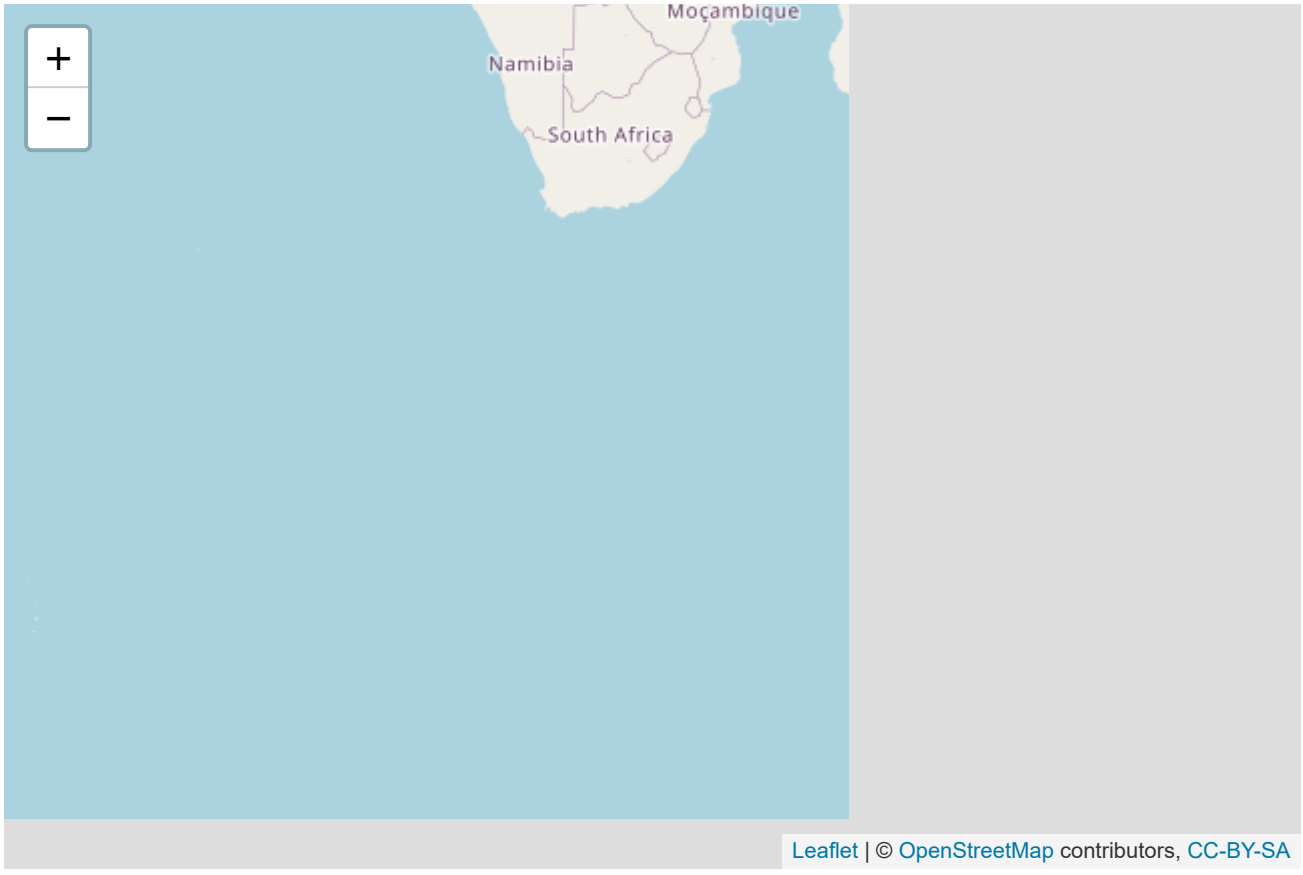
Churn

Curva ROC Código



São Paulo - properties

Mapa Código



Obrigado!

tiagoms.com

tiagomendonca

tiagoms1@insper.edu.br