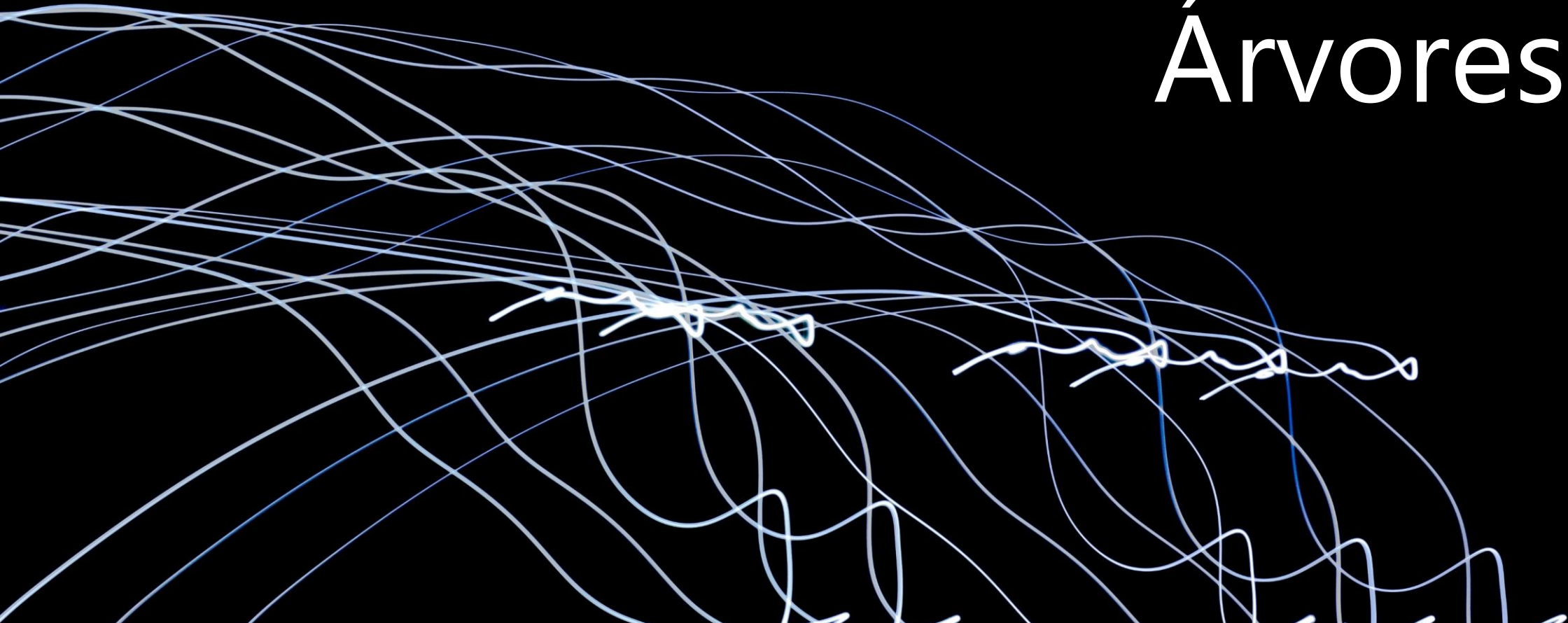


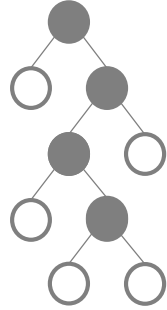
# Modelos de Árvores





# O que vamos aprender

Árvore de Decisão  
(Decision Trees)



Floresta Aleatória  
(Random Forest)

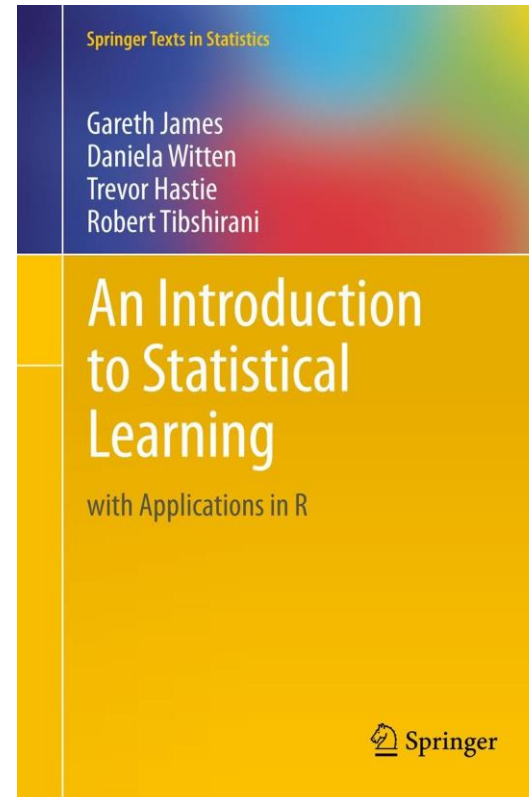
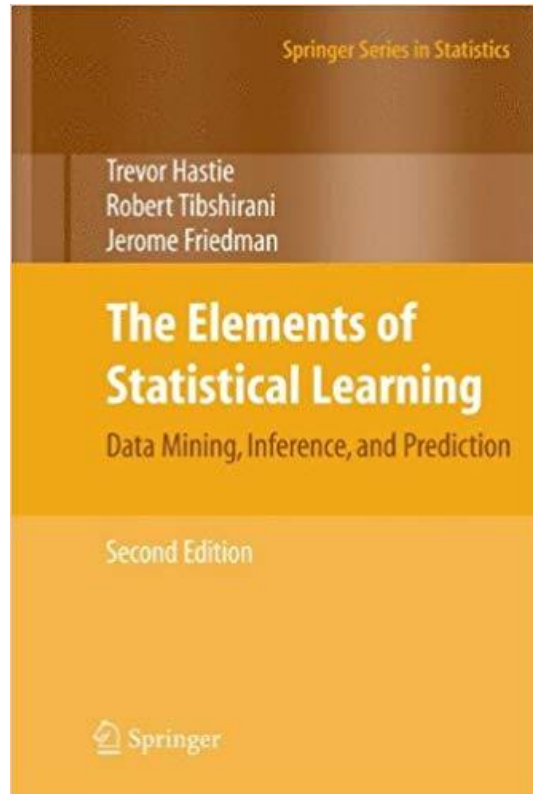


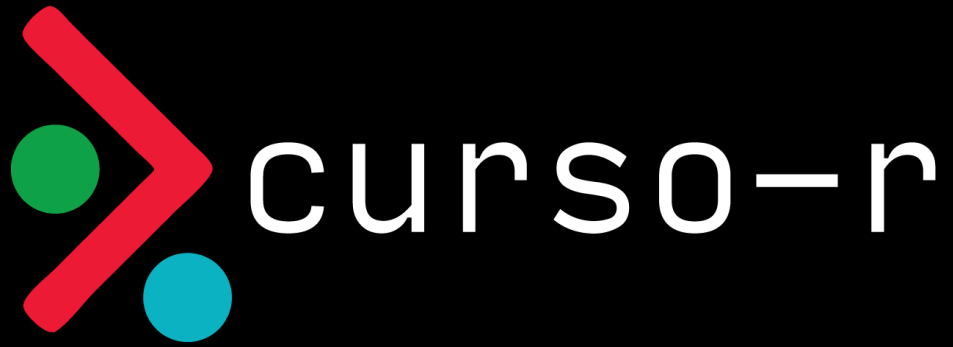
Gradient Boosting



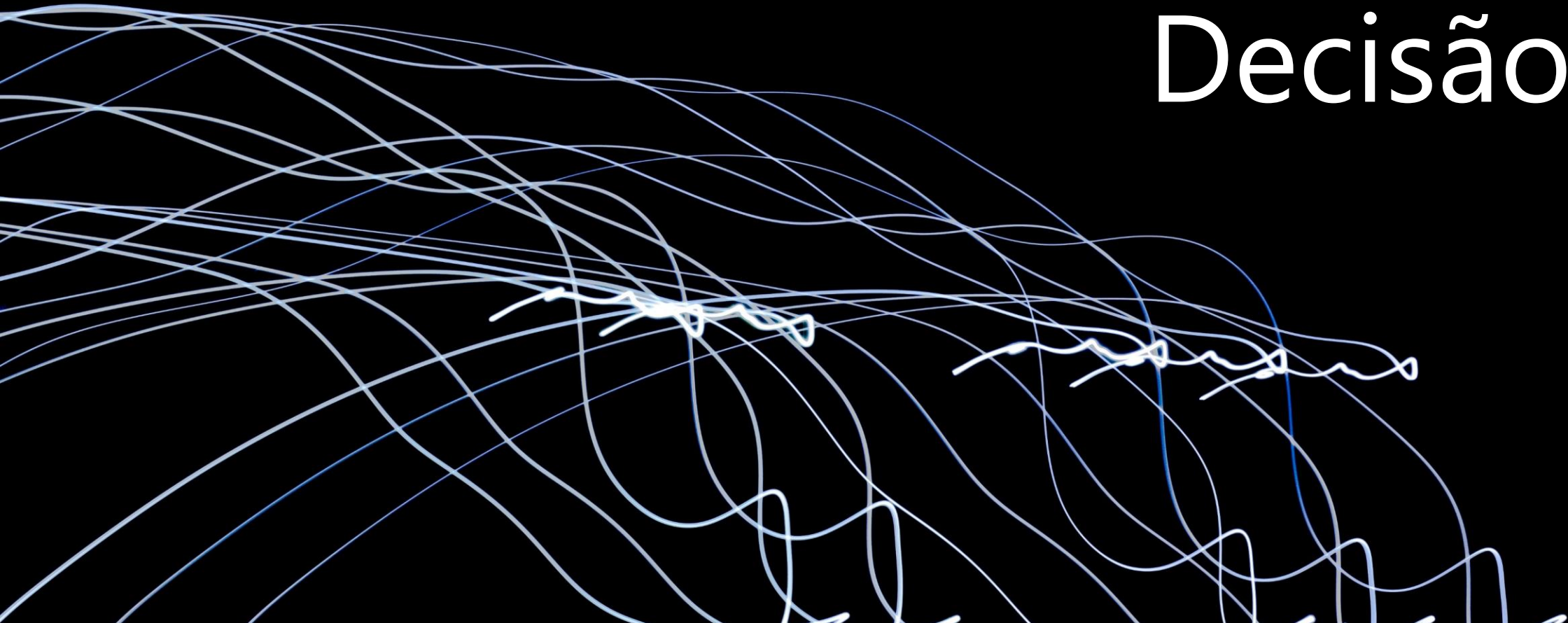


# Referências



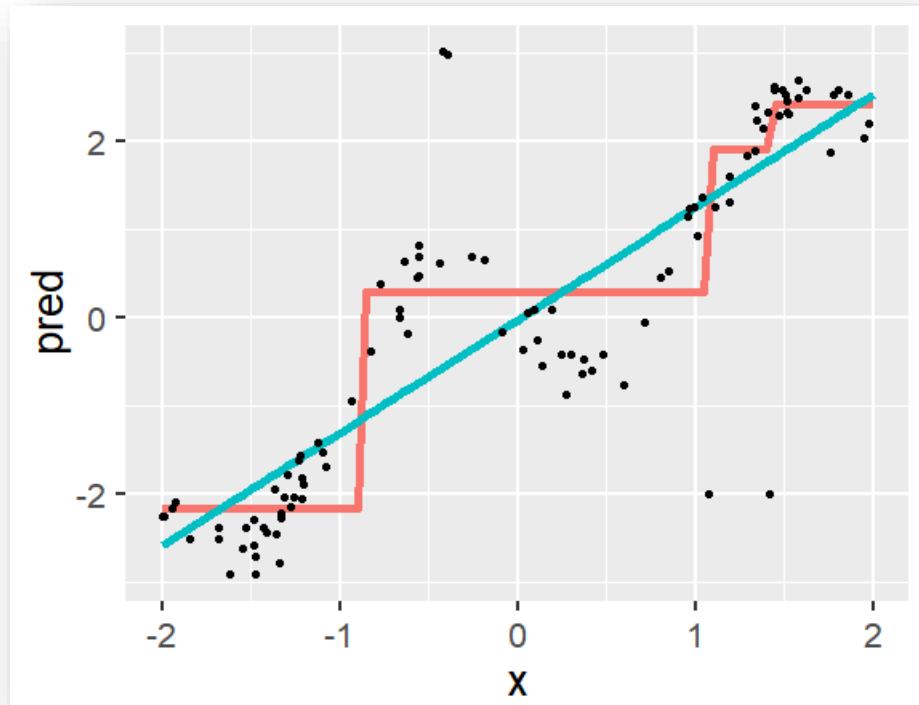


# Árvore de Decisão

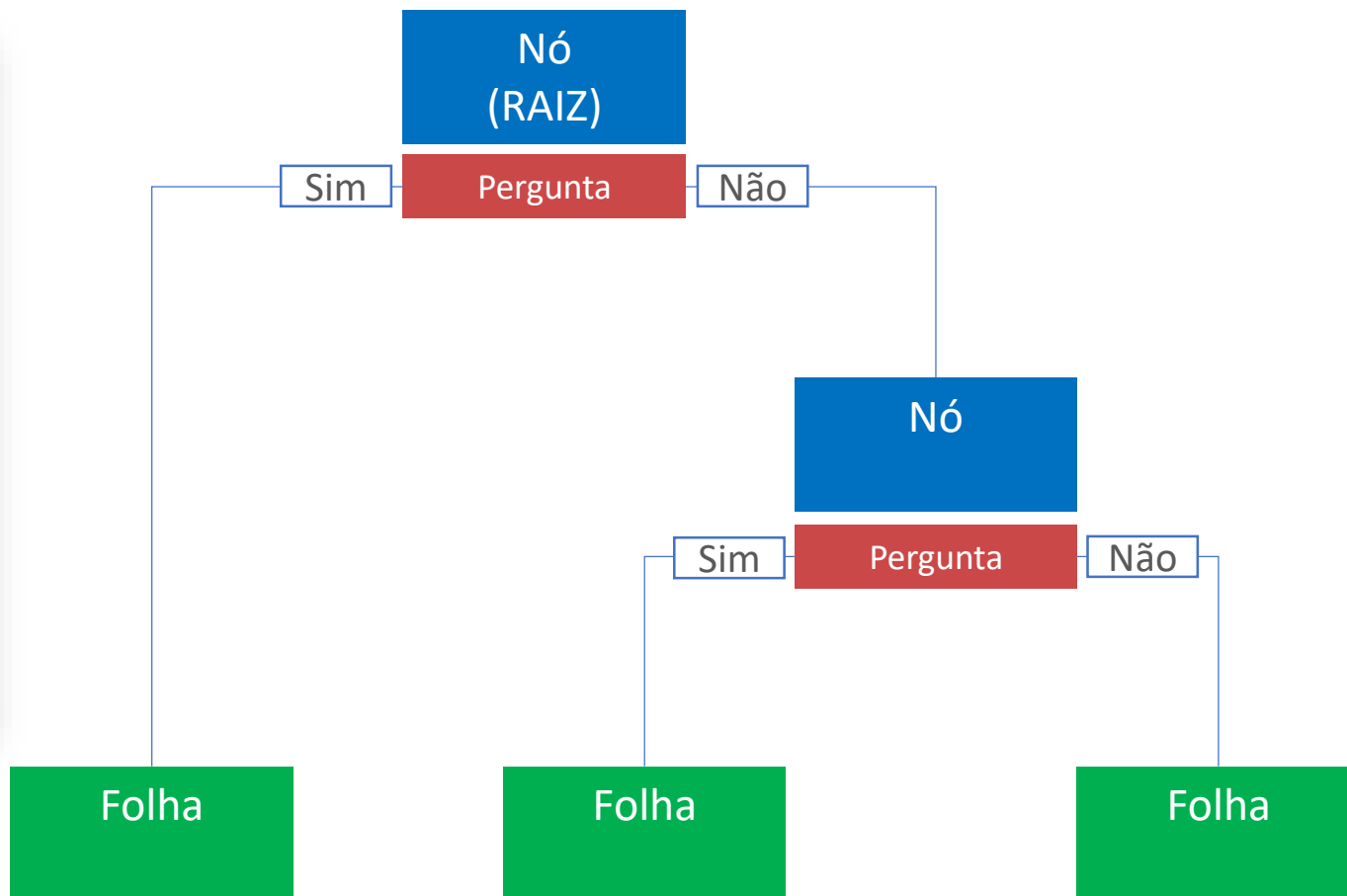




# Anatomia



$$Y \approx f(X)$$

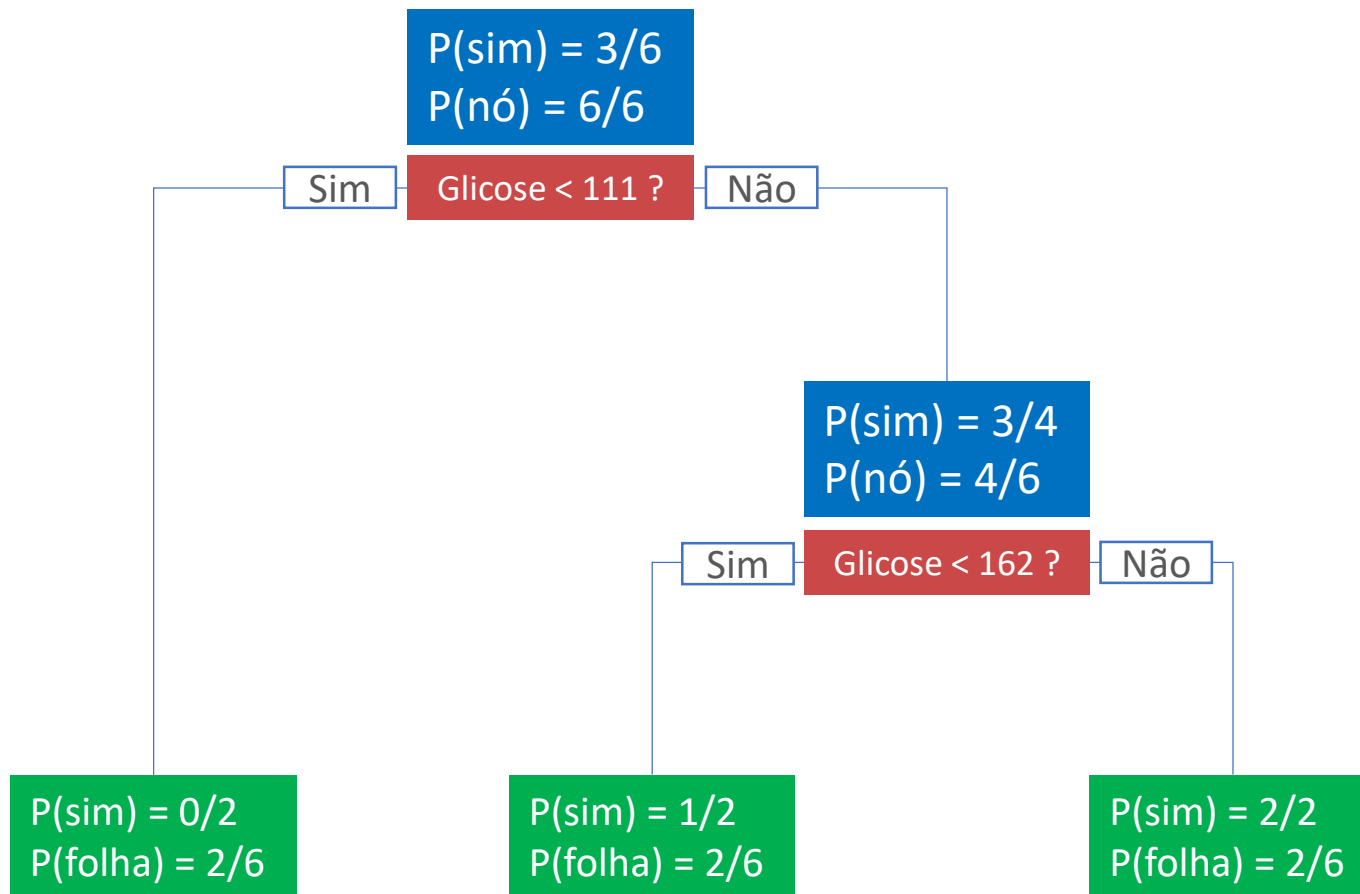




# Anatomia

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$

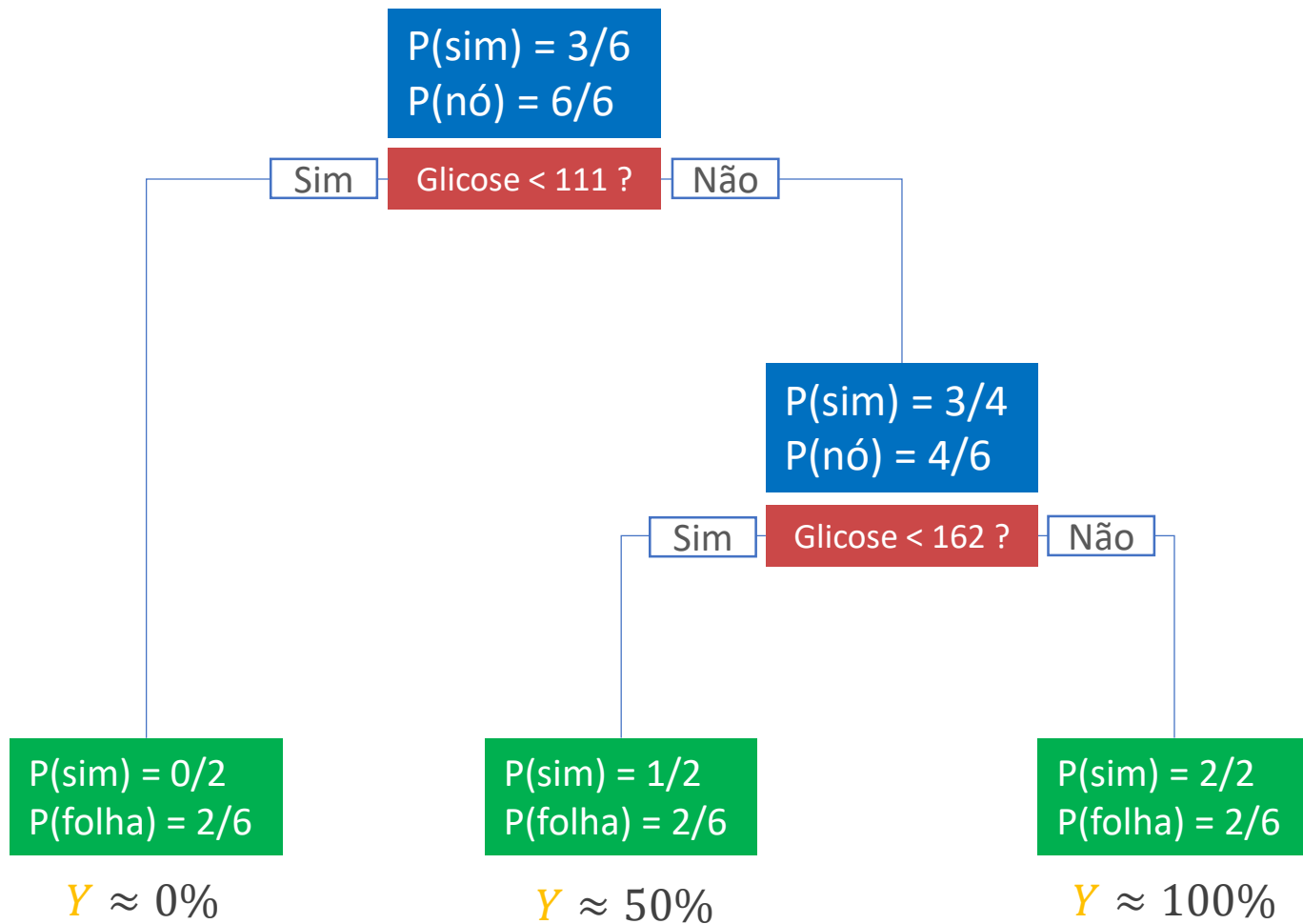




# Anatomia

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$

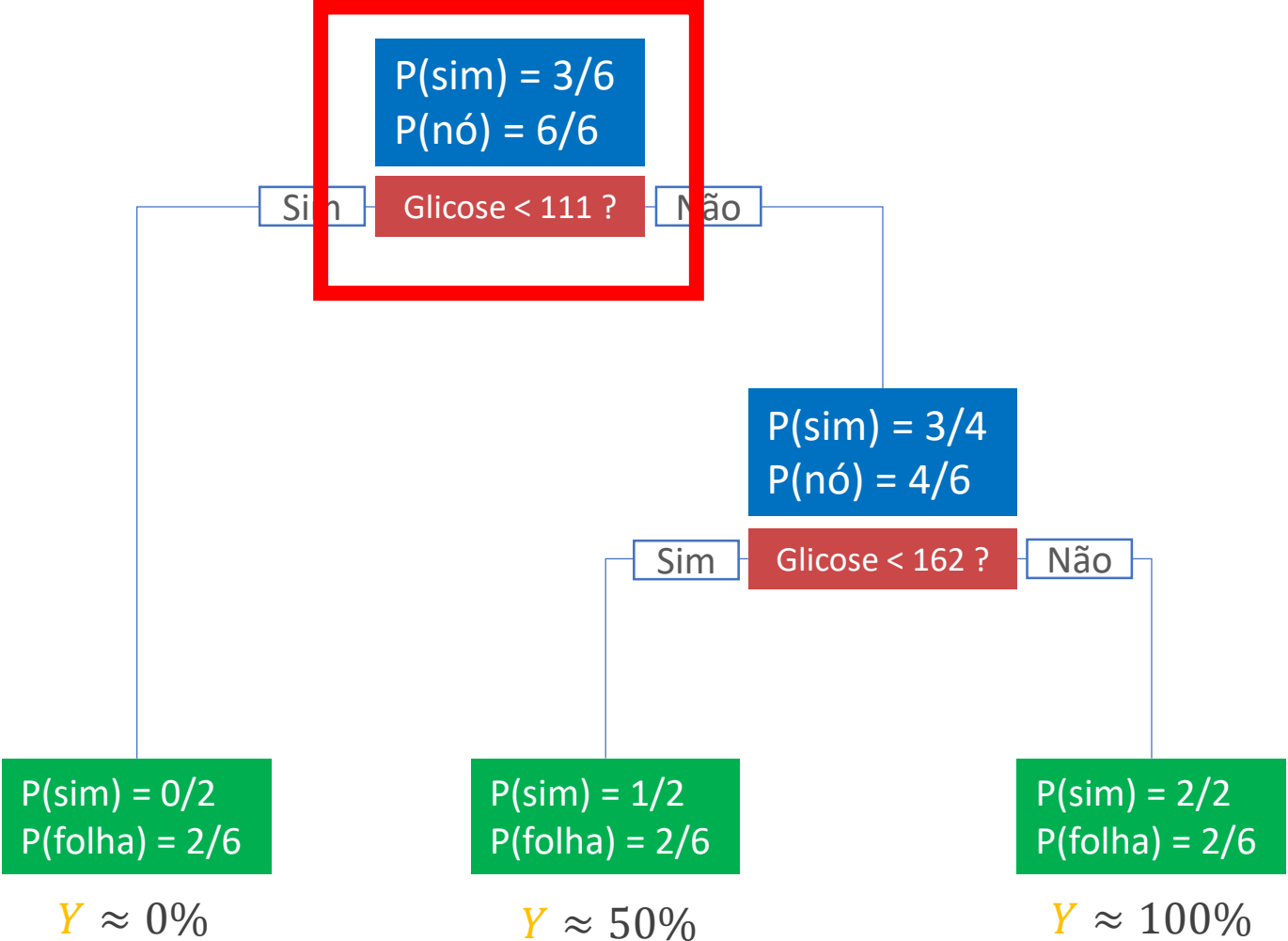




# Anatomia

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$



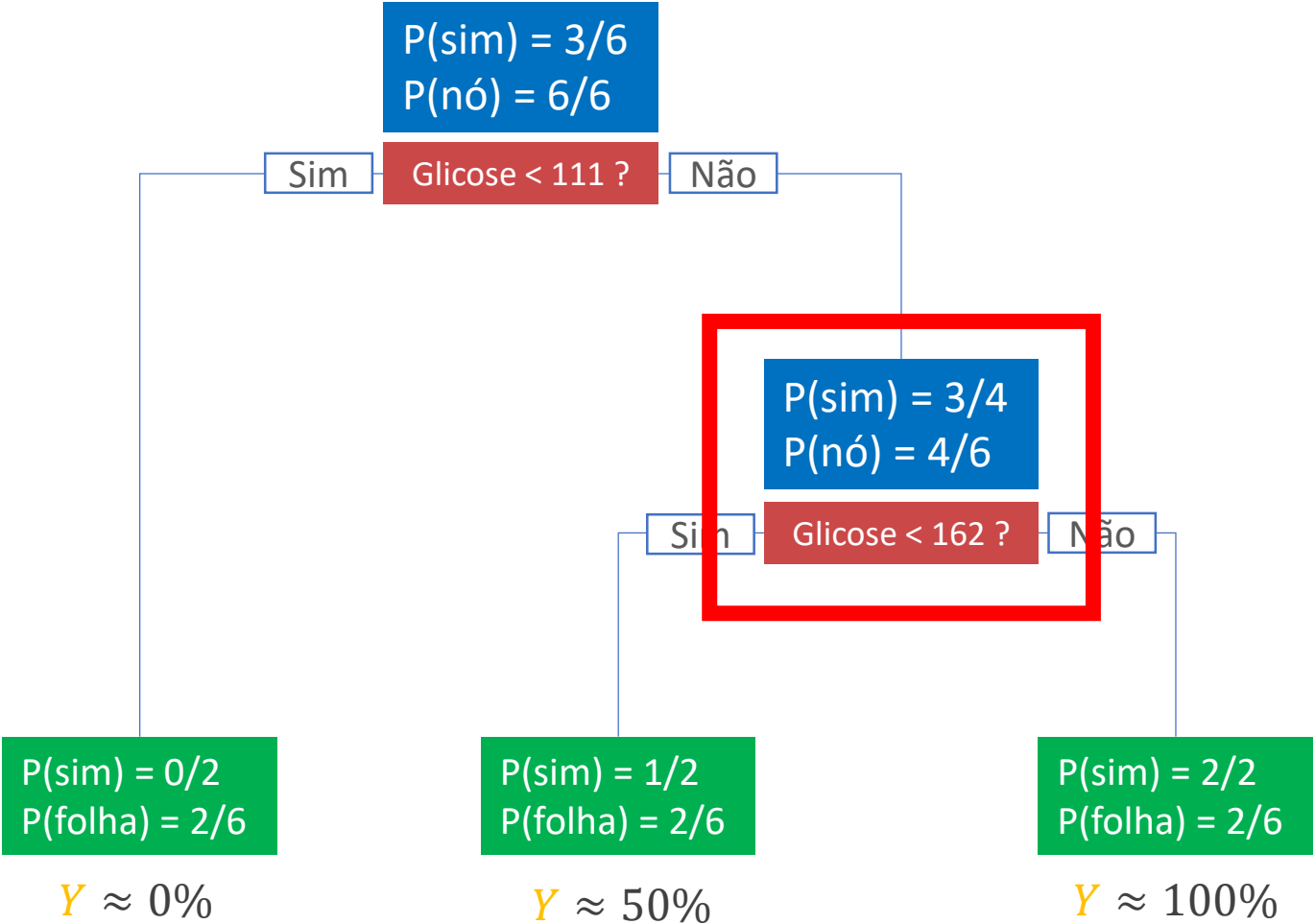




# Anatomia

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$

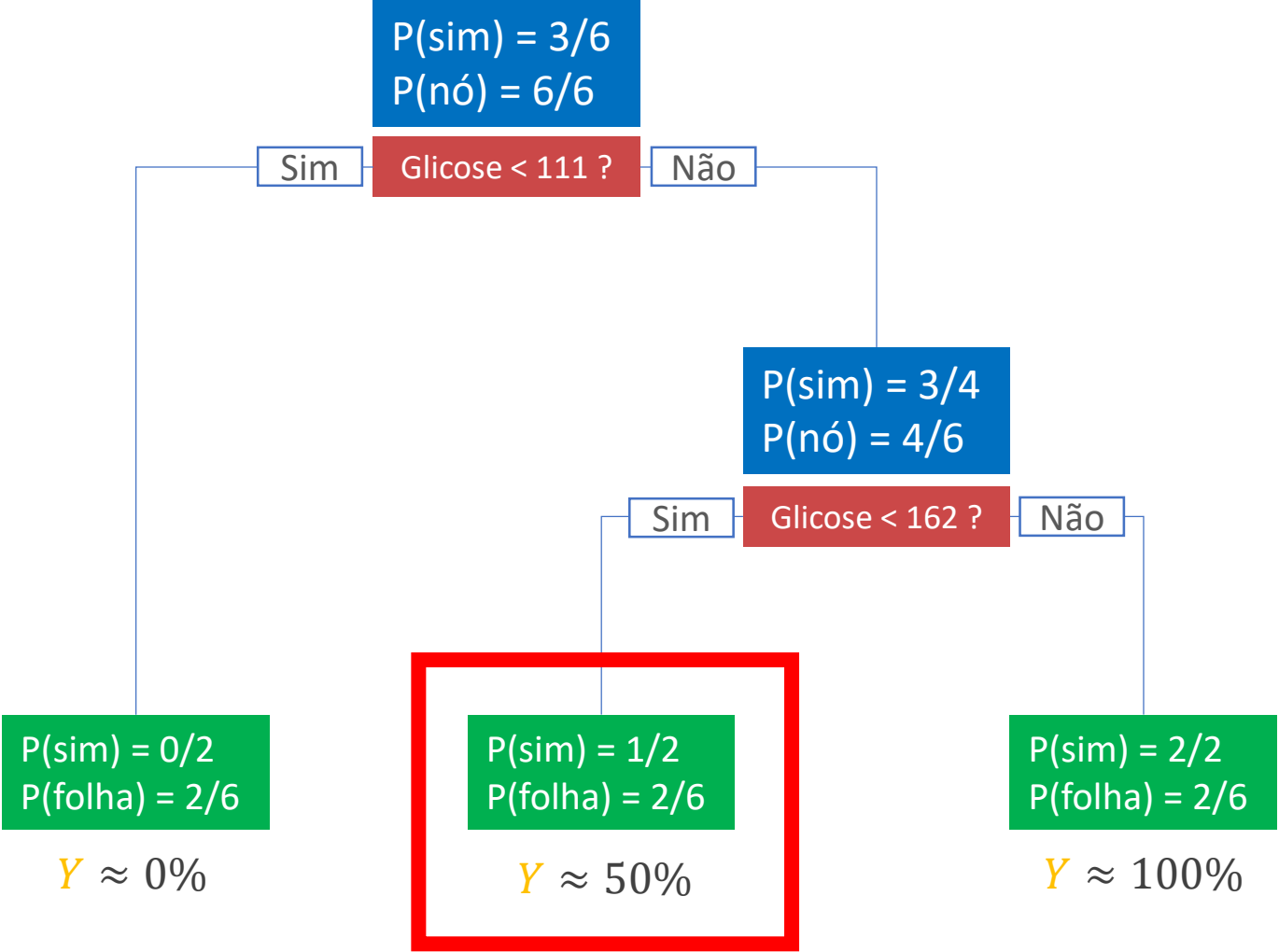




# Anatomia

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$

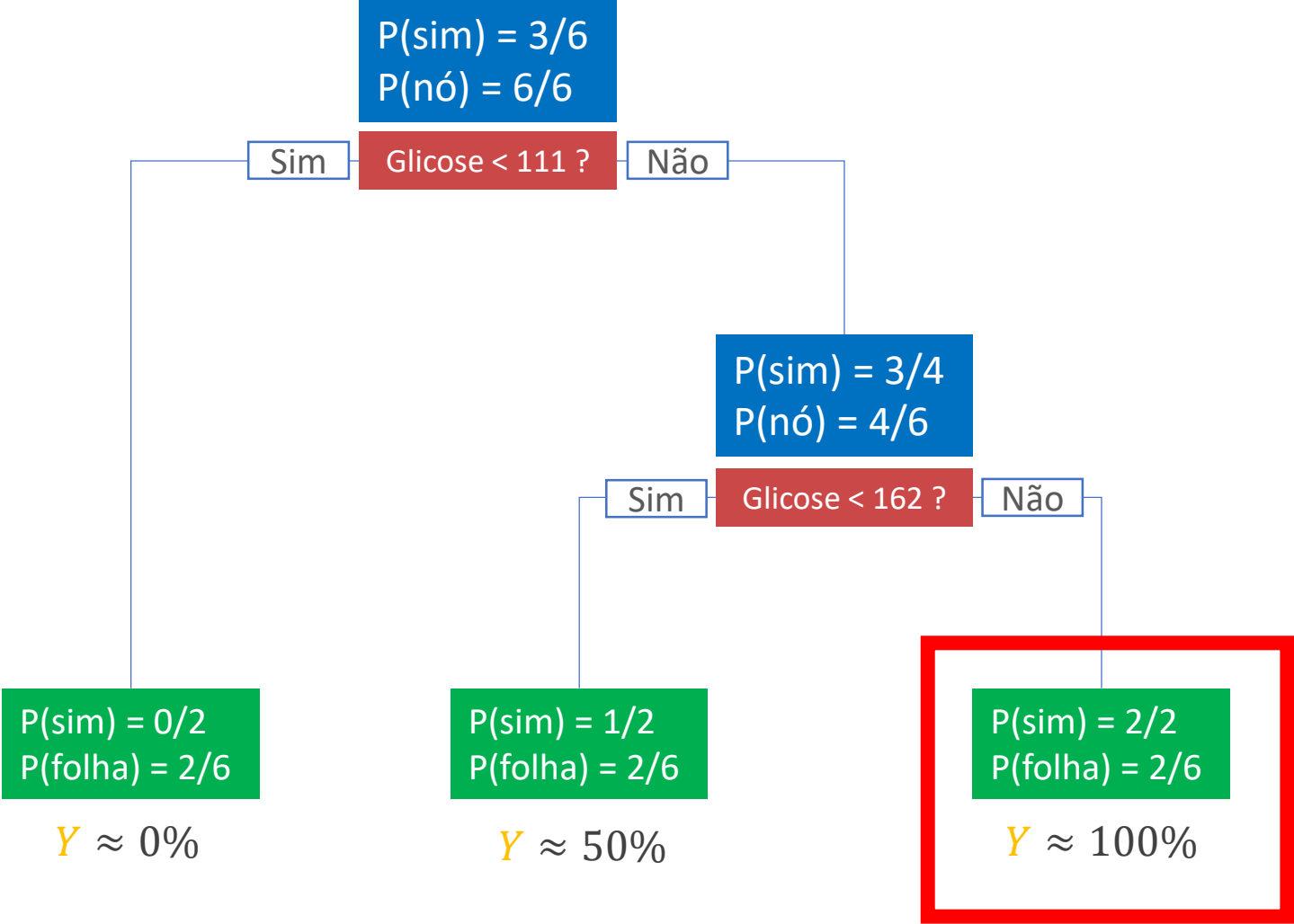




# Anatomia

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$

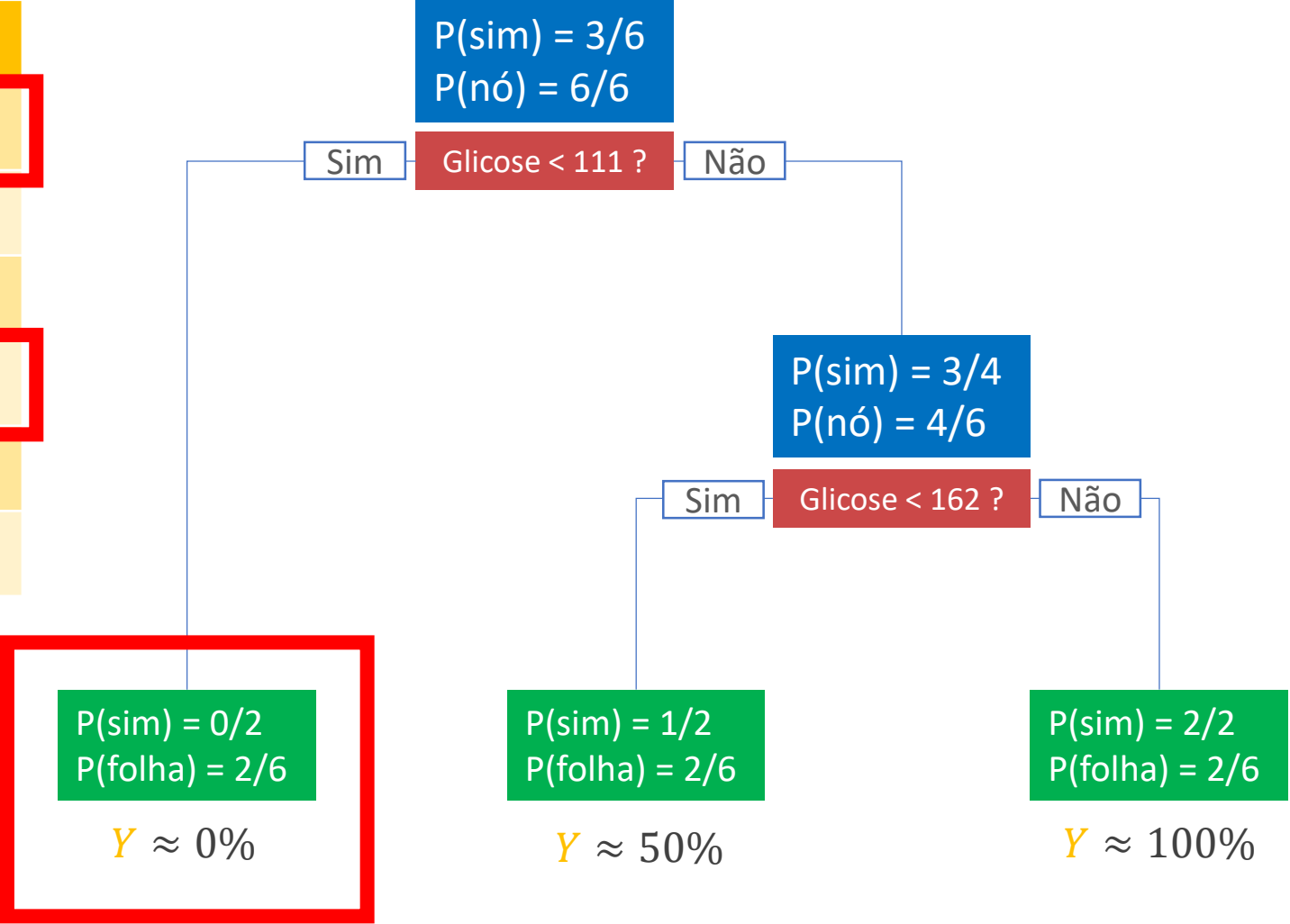




# Anatomia

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$

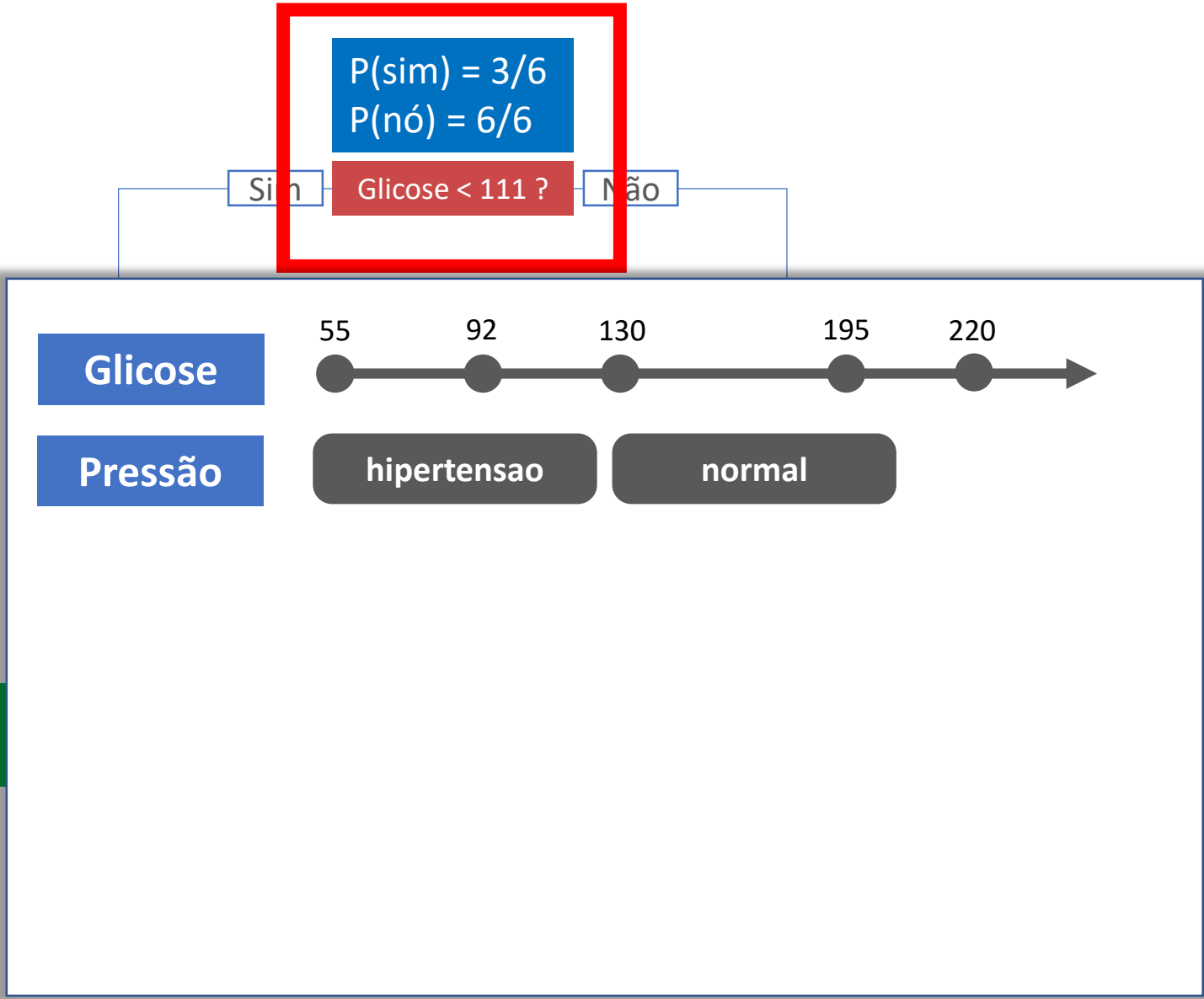




# Perguntas

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$

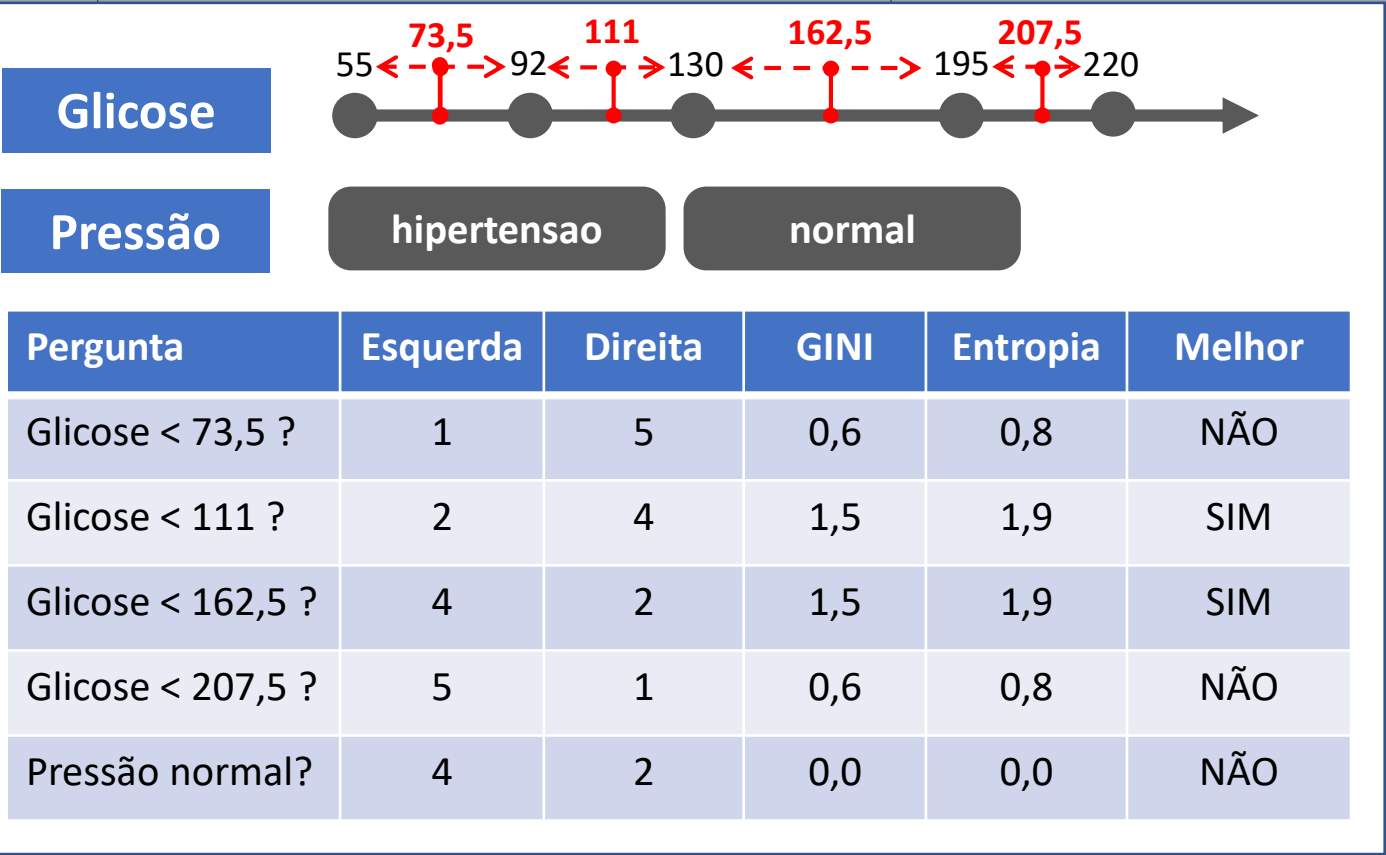
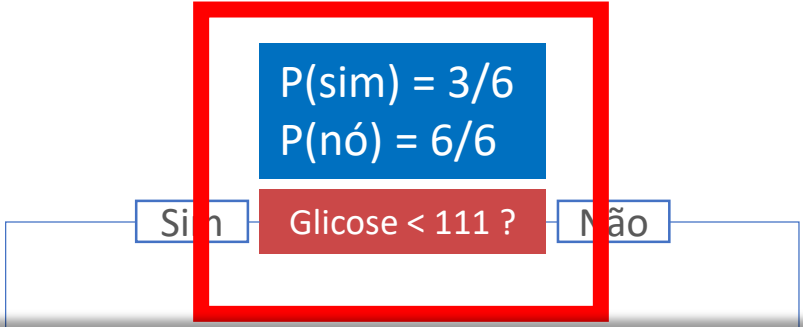




# Perguntas

Paciente	Pressão	Glicose	Diabetes
Alfredo	hipertensao	92	nao
Beatriz	normal	130	sim
Carla	normal	130	nao
Daniela	normal	55	nao
Ernesto	hipertensao	220	sim
Flavia	normal	195	sim

$$Y \approx f(X)$$





# Impureza e Ganho de Informação

## Ganho de informação: (*Information Gain*)

$$IG(P, lado) = N \cdot \text{Impureza}(P) - N_{esq} \cdot \text{Impureza}(P, esq) - N_{dir} \cdot \text{Impureza}(P, dir)$$

## Medidas de impureza mais comuns:

Impureza	Tarefa	Fórmula	Descrição
GINI	Classificação	$1 - \sum p_i^2$	$p_i$ é a proporção do rótulo $i$ , $i = 1, \dots, C$ .
Entropia	Classificação	$-\sum p_i \log(p_i)$	$p_i$ é a proporção do rótulo $i$ , $i = 1, \dots, C$ .
Variância	Regressão	$\frac{1}{N} \sum (y_k - \hat{y}_k)^2$	$y_k$ é o observado e $\hat{y}_k$ é a média da folha.



# Impureza e Ganho de Informação

Qual "pergunta" é a melhor? I ou II?



$P(\text{sim}) = 3/6$   
 $P(\text{nó}) = 6/6$

Sim

Glicose < 162 ?

Não

$P(\text{sim}) = 2/2$   
 $P(\text{folha}) = 2/6$

$Y \approx 100\%$

$P(\text{sim}) = 1/4$   
 $P(\text{folha}) = 4/6$

$Y \approx 25\%$



$P(\text{sim}) = 3/6$   
 $P(\text{nó}) = 6/6$

Sim

Glicose < 207 ?

Não

$P(\text{sim}) = 1/1$   
 $P(\text{folha}) = 1/6$

$Y \approx 100\%$

$P(\text{sim}) = 2/5$   
 $P(\text{folha}) = 5/6$

$Y \approx 40\%$

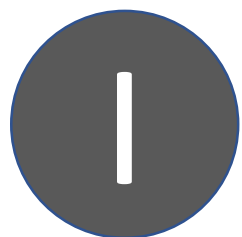




# Impureza e Ganho de Informação

## Qual quebra é a melhor? I ou II?

Exemplo usando GINI como medida de IMPUREZA



$P(\text{sim}) = 3/6$   
 $P(\text{nó}) = 6/6$

$$1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0,50$$

Sim

Glicose < 162 ?

Não

$$1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0,00$$

$P(\text{sim}) = 2/2$   
 $P(\text{folha}) = 2/6$

$Y \approx 100\%$

$$1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0,38$$

$P(\text{sim}) = 1/4$   
 $P(\text{folha}) = 4/6$

$Y \approx 25\%$

$$GINI(I) = 6 * 0,50 - 2 * 0,00 - 4 * 0,38$$

$$GINI(I) = 1,5$$



$P(\text{sim}) = 3/6$   
 $P(\text{nó}) = 6/6$

$$1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0,50$$

Sim

Glicose < 207 ?

Não

$$1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2 = 0,00$$

$P(\text{sim}) = 1/1$   
 $P(\text{folha}) = 1/6$

$Y \approx 100\%$

$$1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0,48$$

$P(\text{sim}) = 2/5$   
 $P(\text{folha}) = 5/6$

$Y \approx 40\%$

$$GINI(II) = 6 * 0,50 - 1 * 0,00 - 5 * 0,48$$

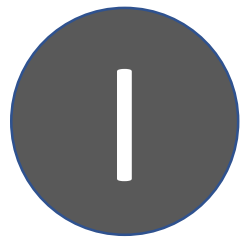
$$GINI(II) = 0,6$$



# Impureza e Ganho de Informação

## Exercício

Exemplo usando GINI como medida de IMPUREZA



$$P(\text{sim}) = 3/6$$
$$P(\text{nó}) = 6/6$$

$$1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0,50$$

Sim

Glicose < 162 ?

Não

$$1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0,00$$

$$P(\text{sim}) = 2/2$$
$$P(\text{folha}) = 2/6$$

$$Y \approx 100\%$$

$$1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0,38$$

$$P(\text{sim}) = 1/4$$
$$P(\text{folha}) = 4/6$$

$$Y \approx 25\%$$

$$GINI(I) = 6 * 0,50 - 2 * 0,00 - 4 * 0,38$$
$$GINI(I) = 1,5$$



$$P(\text{sim}) = 3/6$$
$$P(\text{nó}) = 6/6$$

Sim

Glicose < 111 ?

Não

$$P(\text{sim}) = 0/2$$
$$P(\text{folha}) = 2/6$$

$$Y \approx 0\%$$

$$P(\text{sim}) = 3/4$$
$$P(\text{folha}) = 4/6$$

$$Y \approx 75\%$$

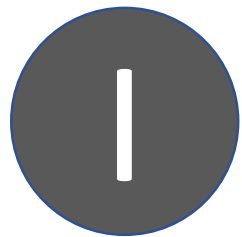
$$GINI(II) =$$
$$GINI(II) =$$



# Impureza e Ganho de Informação

## Exercício (resposta)

Exemplo usando GINI como medida de IMPUREZA



$$P(\text{sim}) = 3/6$$
$$P(\text{nó}) = 6/6$$

$$1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0,50$$

Sim

Glicose < 162 ?

Não

$$1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0,00$$

$$P(\text{sim}) = 2/2$$
$$P(\text{folha}) = 2/6$$

$$Y \approx 100\%$$

$$1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0,38$$

$$P(\text{sim}) = 1/4$$
$$P(\text{folha}) = 4/6$$

$$Y \approx 25\%$$

$$GINI(I) = 6 * 0,50 - 2 * 0,00 - 4 * 0,38$$

$$GINI(I) = 1,5$$



$$P(\text{sim}) = 3/6$$
$$P(\text{nó}) = 6/6$$

$$1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0,50$$

Sim

Glicose < 111 ?

Não

$$1 - \left(\frac{0}{2}\right)^2 - \left(\frac{2}{2}\right)^2 = 0,00$$

$$P(\text{sim}) = 0/2$$
$$P(\text{folha}) = 2/6$$

$$Y \approx 0\%$$

$$1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0,38$$

$$P(\text{sim}) = 3/4$$
$$P(\text{folha}) = 4/6$$

$$Y \approx 75\%$$

$$GINI(II) = 6 * 0,50 - 2 * 0,00 - 4 * 0,38$$

$$GINI(II) = 1,5$$

# Hiperparâmetros e Overfitting

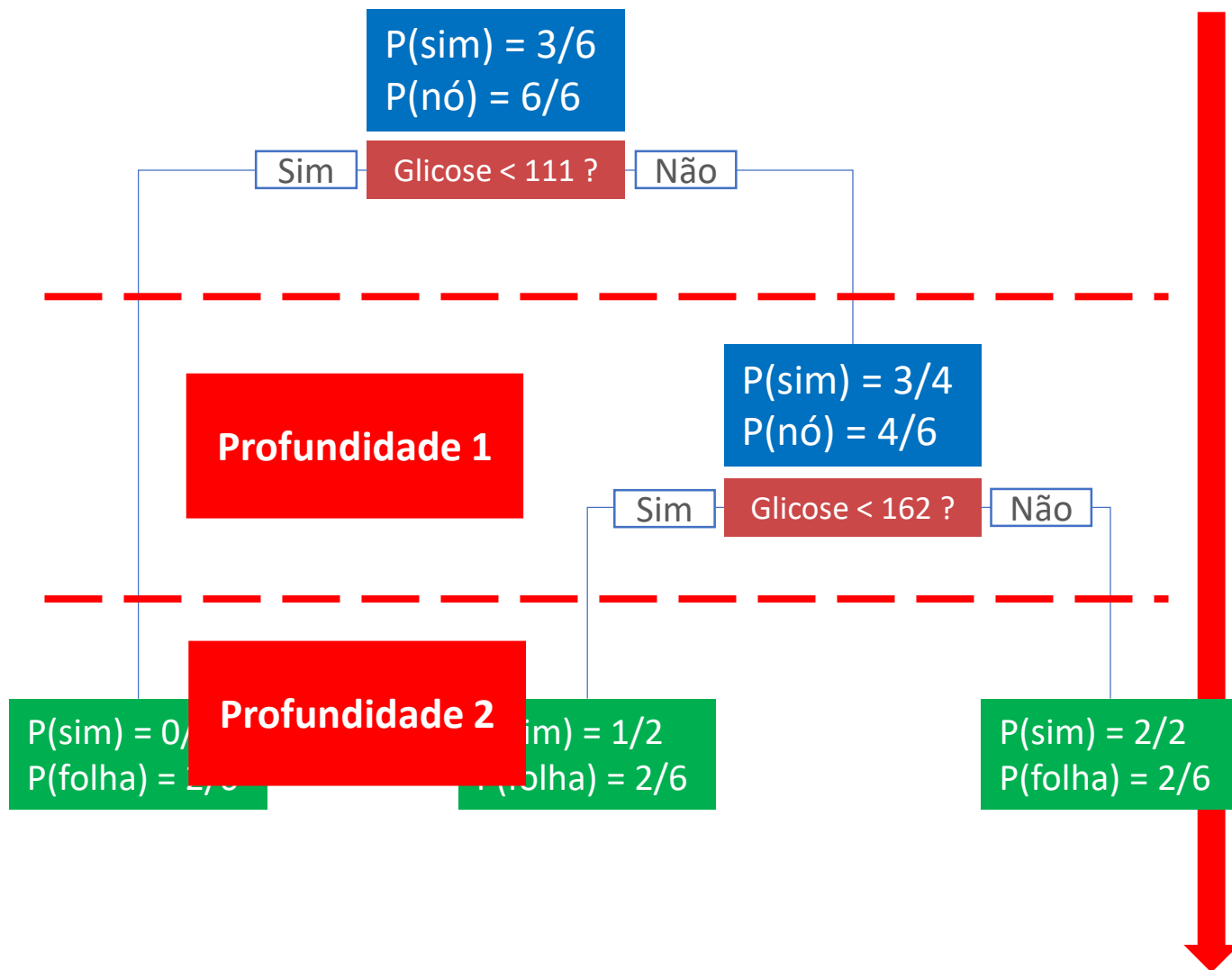
Hiperparâmetros do pacote `rpart` do R

**maxdepth** – Profundidade: quanto mais profunda a árvore for, maior risco de overfitting.

**minsplit** – Quantidade mínima de observações dentro de um nó para se considerar dividir em duas folhas novas. Quanto menor, maior risco de overfitting.

**minbucket** – Quantidade mínima de observações dentro das novas folhas para se considerar dividir. Quanto menor, maior risco de overfitting.

**cp** – Parâmetro de complexidade: limite mínimo de ganho de informação que a divisão tem que fornecer para concretizar a criação das folhas.



# Hiperparâmetros e Overfitting

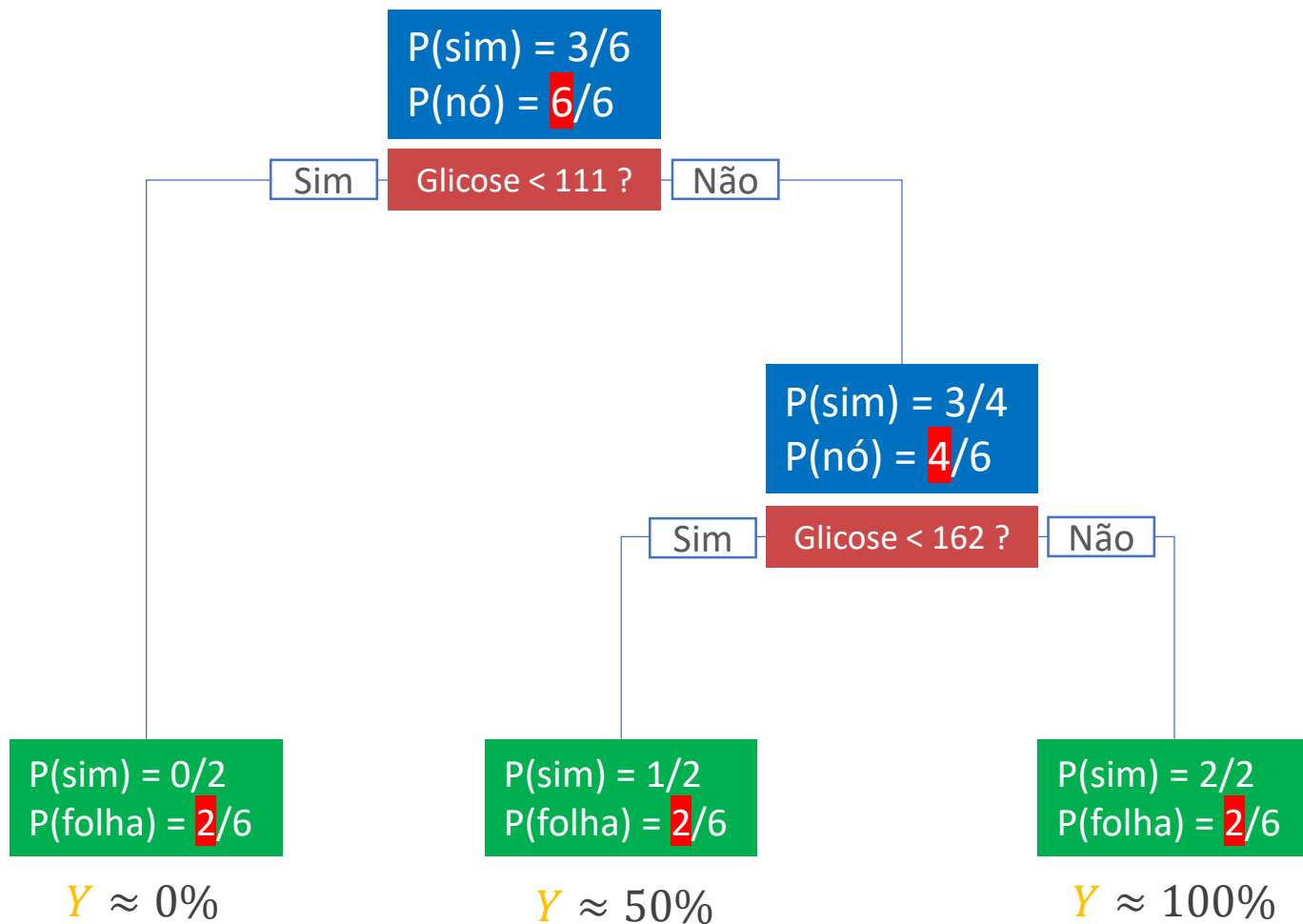
## Hiperparâmetros do pacote `rpart` do R

**maxdepth** – Profundidade: quanto mais profunda a árvore for, maior risco de overfitting.

**minsplit** – Quantidade mínima de observações dentro de um nó para se considerar dividir em duas folhas novas. Quanto menor, maior risco de overfitting.

**minbucket** – Quantidade mínima de observações dentro das novas folhas para se considerar dividir. Quanto menor, maior risco de overfitting.

**cp** – Parâmetro de complexidade: limite mínimo de ganho de informação que a divisão tem que fornecer para concretizar a criação das folhas.



# Hiperparâmetros e Overfitting

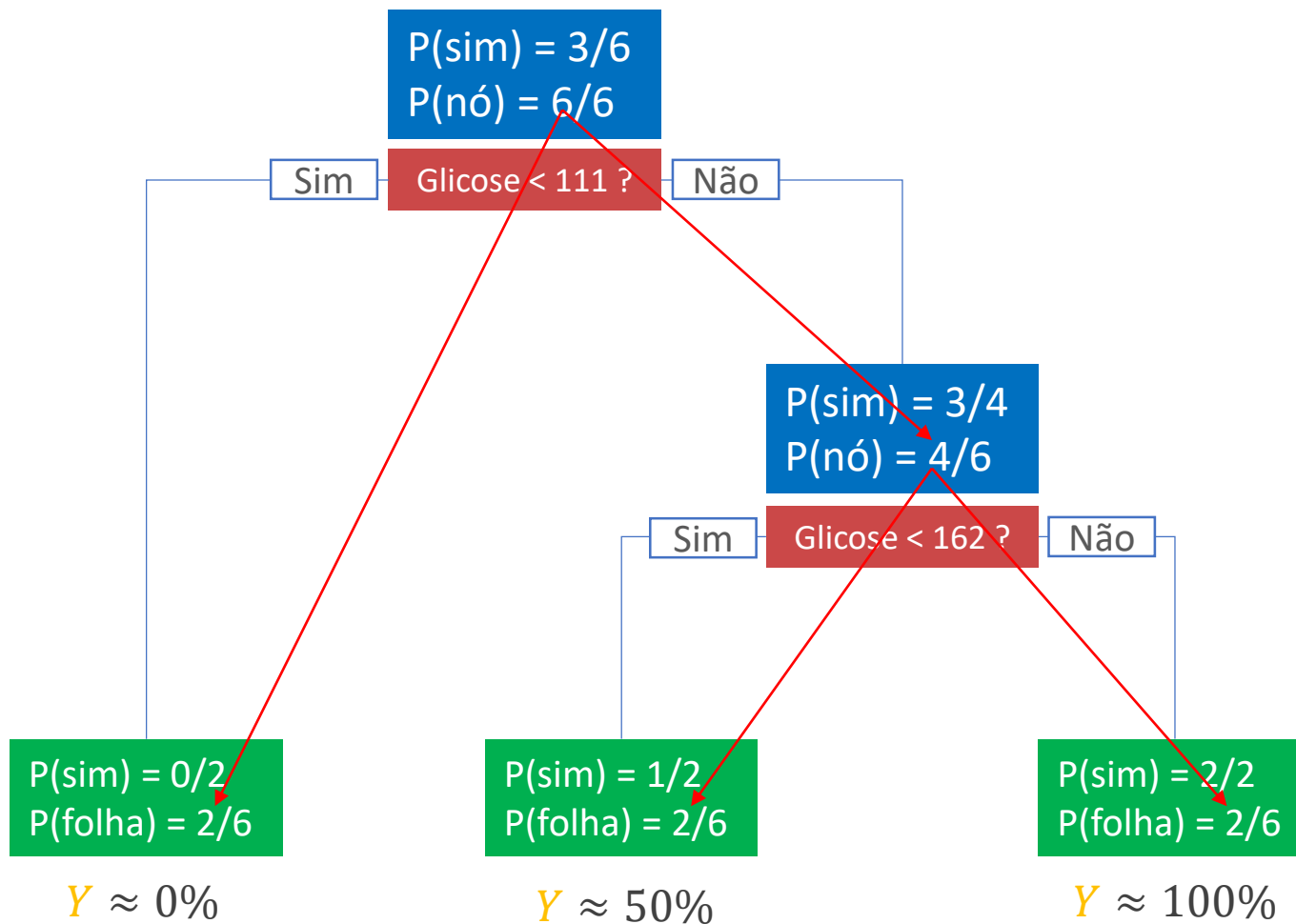
## Hiperparâmetros do pacote `rpart` do R

**maxdepth** – Profundidade: quanto mais profunda a árvore for, maior risco de overfitting.

**minsplit** – Quantidade mínima de observações dentro de um nó para se considerar dividir em duas folhas novas. Quanto menor, maior risco de overfitting.

**minbucket** – Quantidade mínima de observações dentro das novas folhas para se considerar dividir. Quanto menor, maior risco de overfitting.

**cp** – Parâmetro de complexidade: limite mínimo de ganho de informação que a divisão tem que fornecer para concretizar a criação das folhas.



# Hiperparâmetros e Overfitting

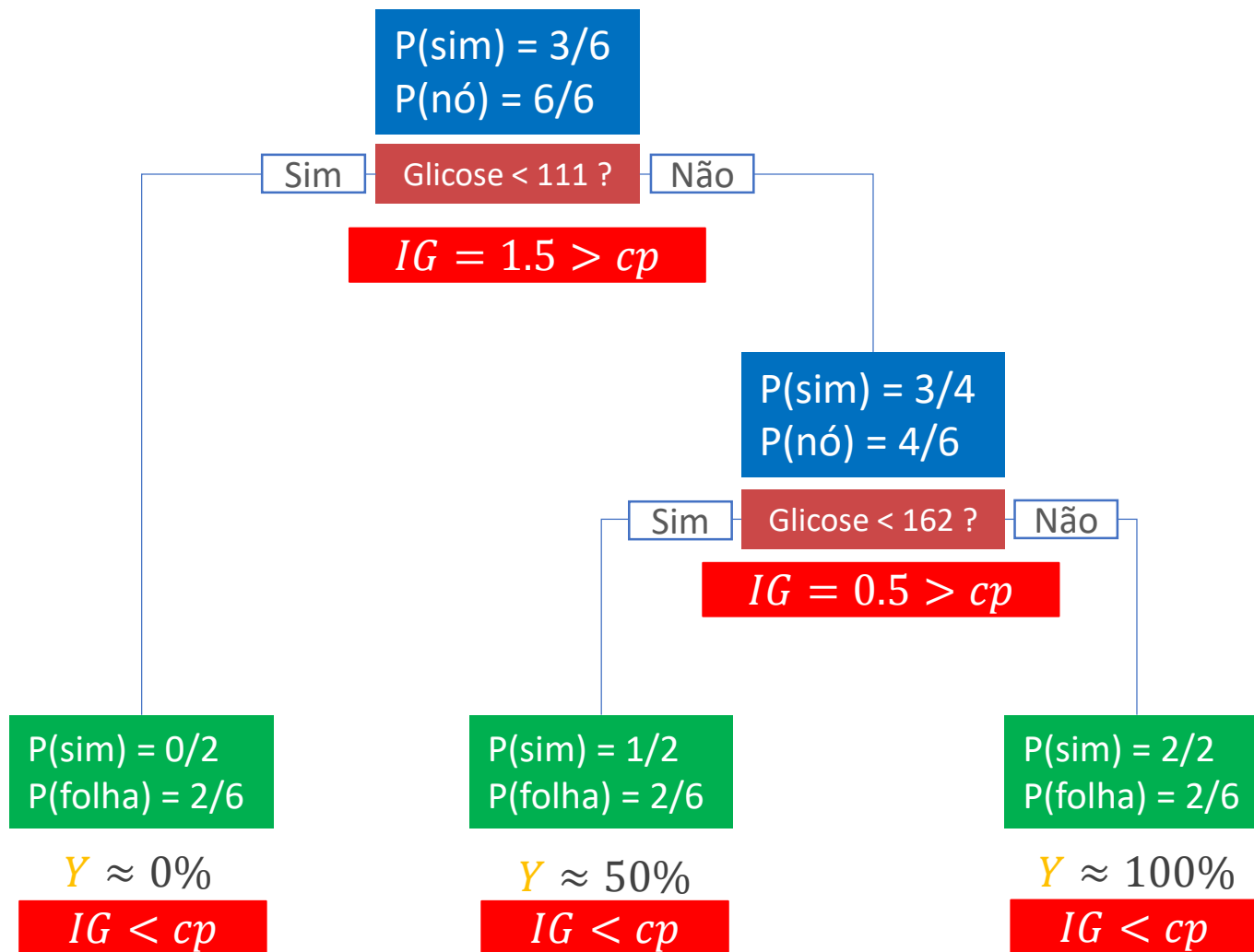
## Hiperparâmetros do pacote `rpart` do R

**maxdepth** – Profundidade: quanto mais profunda a árvore for, maior risco de overfitting.

**minsplit** – Quantidade mínima de observações dentro de um nó para se considerar dividir em duas folhas novas. Quanto menor, maior risco de overfitting.

**minbucket** – Quantidade mínima de observações dentro das novas folhas para se considerar dividir. Quanto menor, maior risco de overfitting.

**cp** – Parâmetro de complexidade: limite mínimo de ganho de informação que a divisão tem que fornecer para concretizar a criação das folhas.

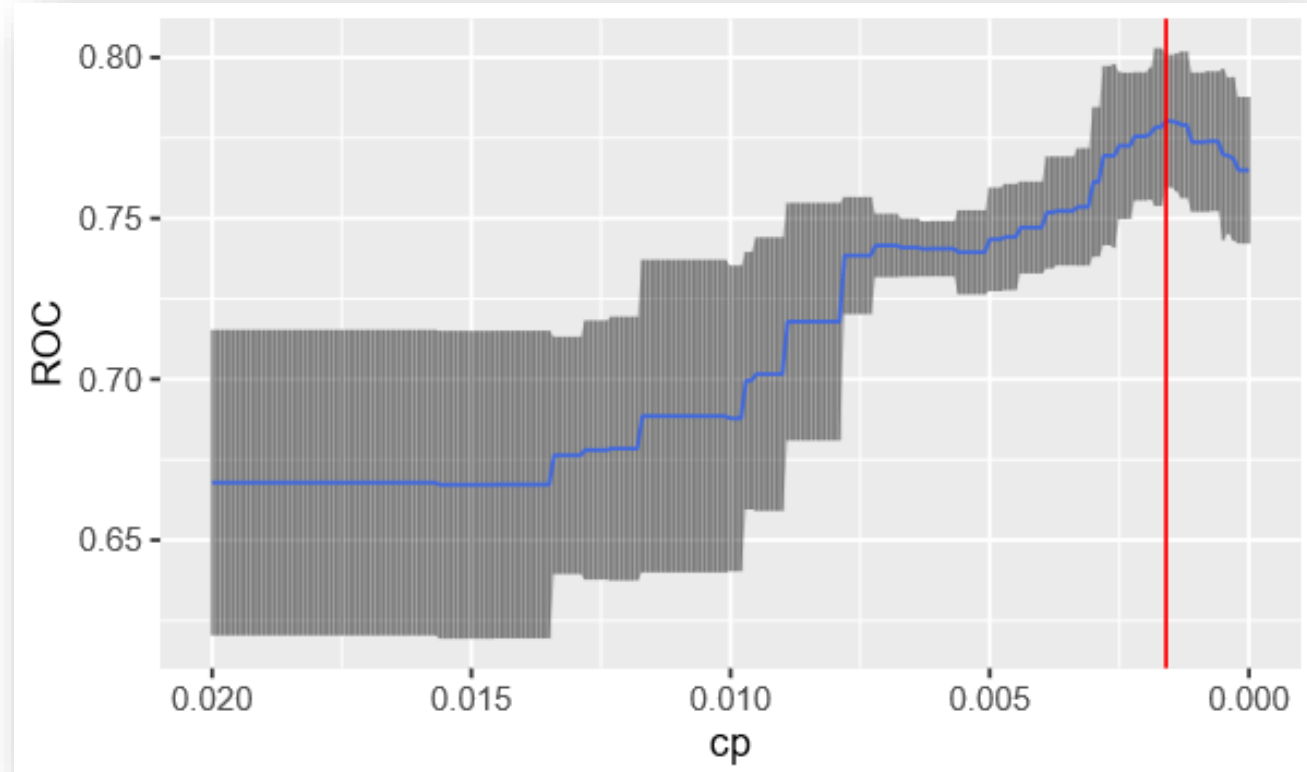




# CP – Complexity Parameter

$$R_{cp}(T) = R(T) + cp * |T| * R(T_1)$$

- Quanto maior o CP menos quebras a árvore vai ter.
- Selecionamos o tamanho de árvore ideal variando o CP (por meio de cross-validation!).







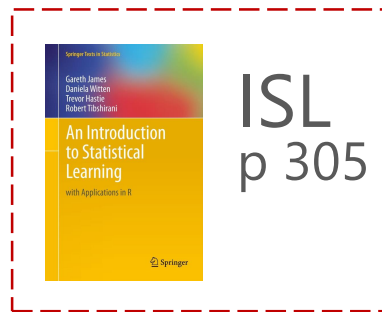
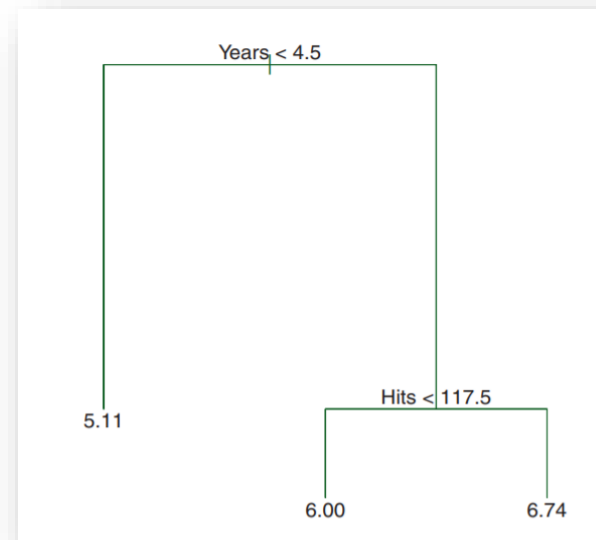
```
library(rpart)
```

Ao R...



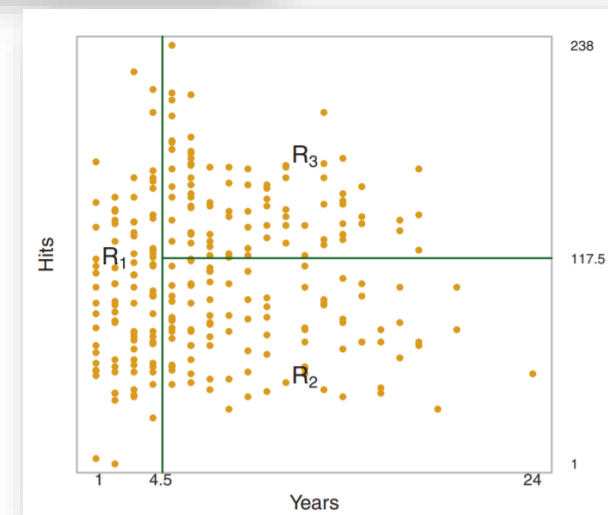
# Comentários e mais um pouco de intuição

- O exemplo foi dado com variável resposta (diabetes) de apenas duas classes: SIM e NÃO. Poderia ter mais, por exemplo: o dígito escrito em uma imagem. A variável resposta teria 10 classes possíveis que seriam os algarismos de 0 a 9.
- A variável explicativa hipertensão apresentava apenas duas classes também, mas poderia apresentar mais. Nesse caso, os algoritmos de árvores têm de decidir como fazer as PERGUNTAS. [Esse link da Freakonometrics](#) apresenta a heurística mais utilizada nesse caso.
- As figuras são duas representações diferentes para um mesmo modelo de árvore. Nesse caso as variáveis explicativas Years e Hits são ambas contínuas e poderiam ser visualizadas num gráfico de dispersão. As regiões R1, R2 e R3 correspondem às folhas da árvore apresentada no gráfico de cima.



Algoritmos mais conhecidos:

- CART (rpart)
- ID3
- C4.5
- C5.0





# Relação Viés-Variância (*Bias-Variance tradeoff*)

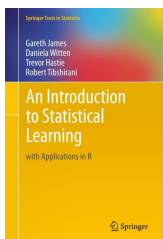
Erro de predição esperado =

$$E \left( Y - \hat{f}(x_0) \right)^2 =$$

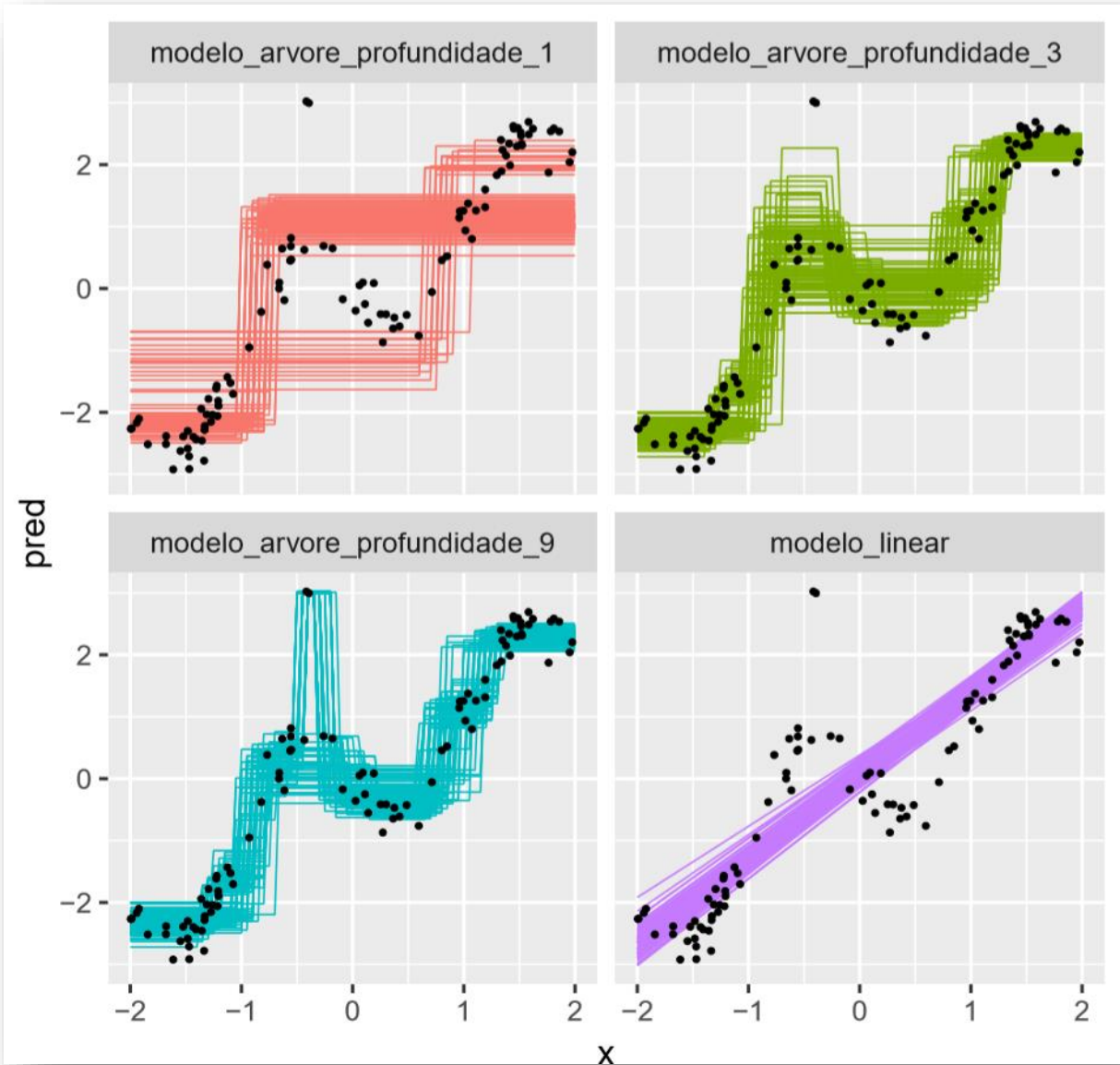
$$E \left( f(x_0) + \epsilon - \hat{f}(x_0) \right)^2 =$$

$$\left[ E\hat{f}(x_0) - f(x_0) \right]^2 + E\left[ \hat{f}(x_0) - E\hat{f}(x_0) \right]^2 + \text{Var}(\epsilon) =$$

$$\text{Viés}^2 + \text{Variância} + \text{Erro Irredutível}$$



ISL  
p 33





# Relação Viés-Variância (*Bias-Variance tradeoff*)

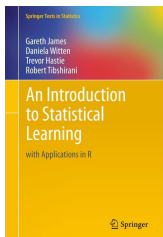
*Erro de predição esperado =*

$$E \left( Y - \hat{f}(x_0) \right)^2 =$$

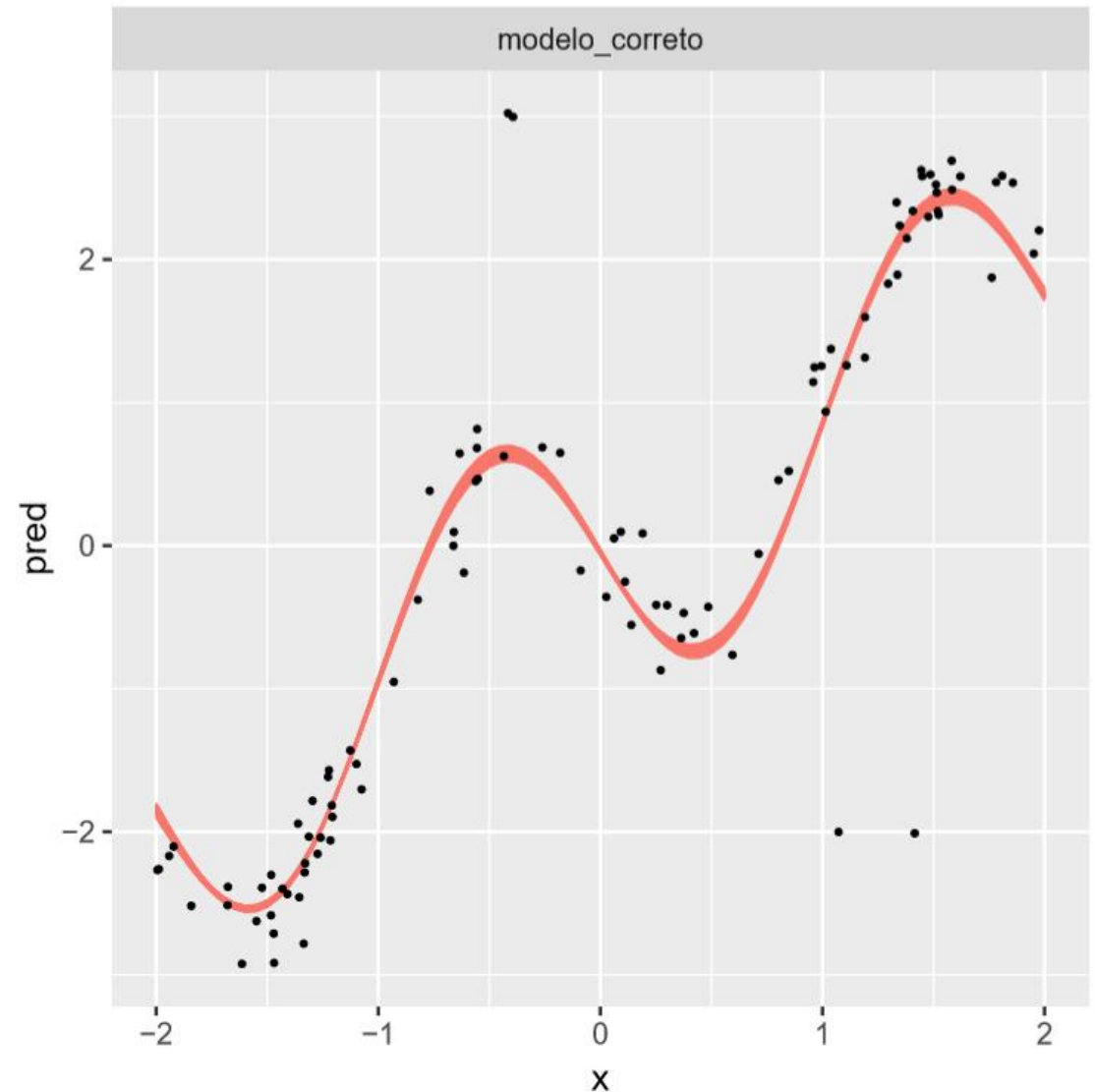
$$E \left( f(x_0) + \epsilon - \hat{f}(x_0) \right)^2 =$$

$$\left[ E\hat{f}(x_0) - f(x_0) \right]^2 + E\left[ \hat{f}(x_0) - E\hat{f}(x_0) \right]^2 + \text{Var}(\epsilon) =$$

$$\text{Viés}^2 + \text{Variância} + \text{Erro Irredutível}$$

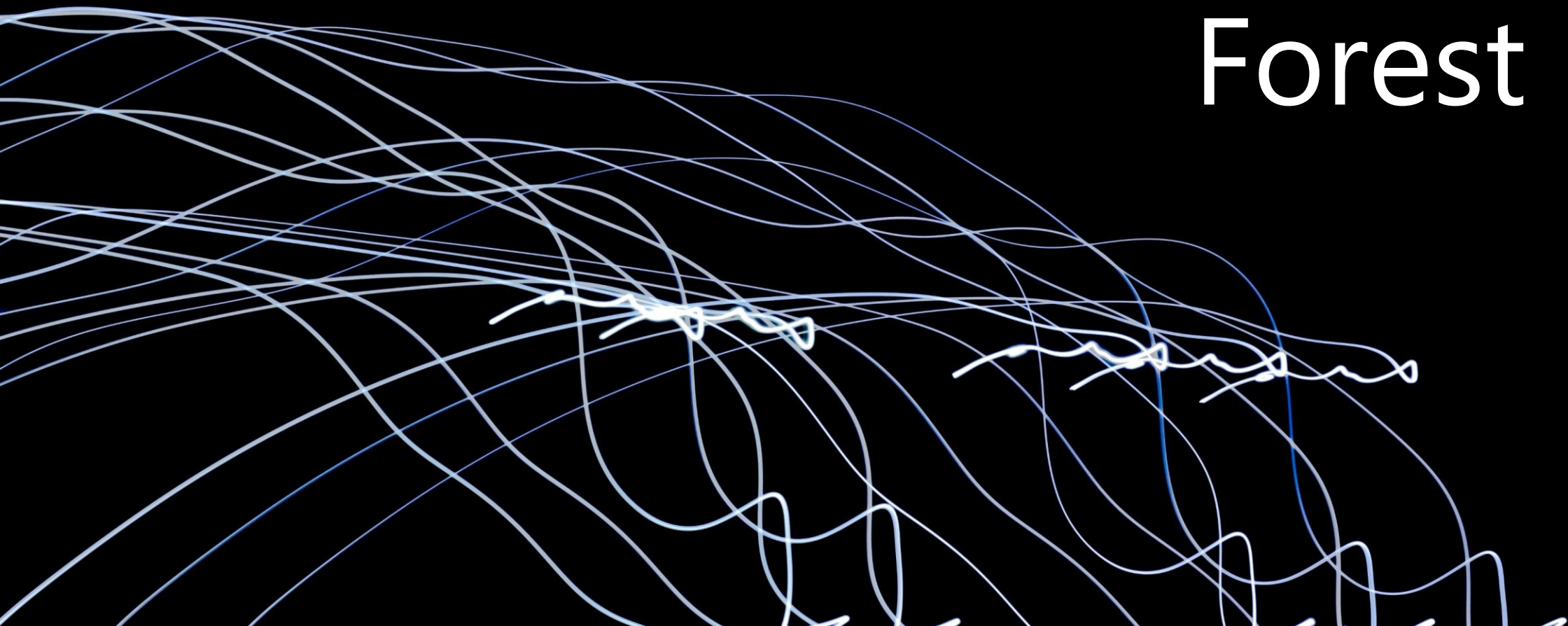


ISL  
p 33







# Random Forest



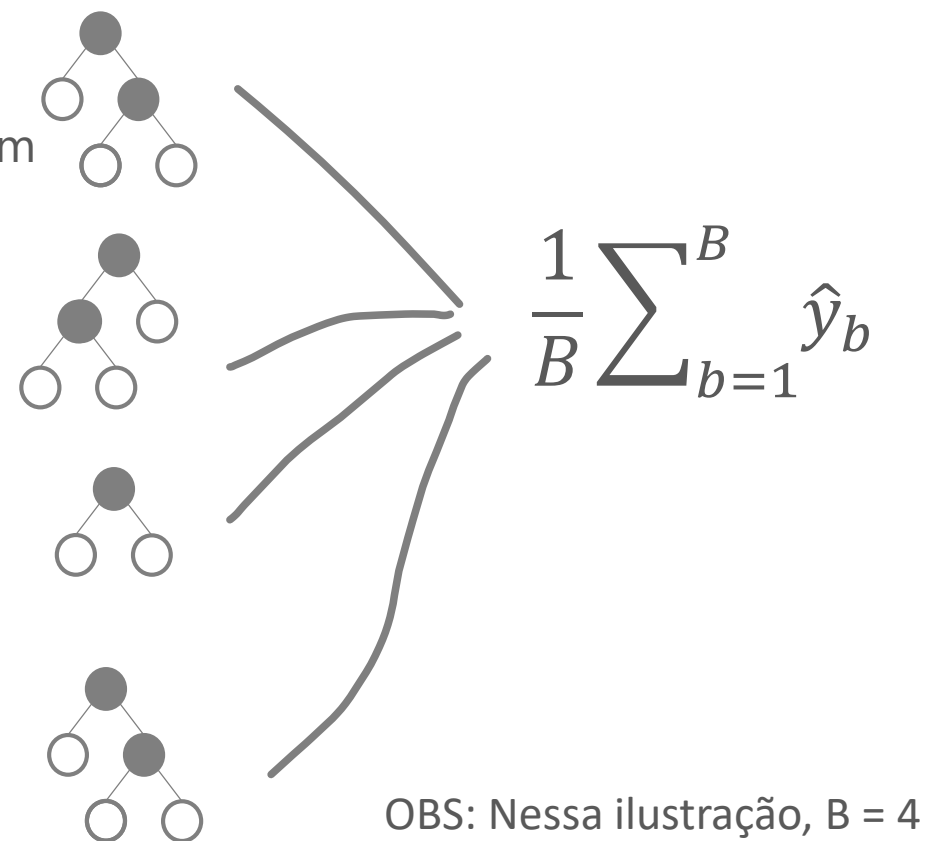


# Random Forest

- Random Forest é a combinação de “palpites” de um monte de árvores de decisão.
- É um algoritmo de uma classe especial de ENSEMBLE: BAGGING.
- ENSEMBLE: mistura de 2 ou mais modelos.  ESL p 605
- BAGGING: Bootstrap AGGregation  ESL p 282
- Diferencial para os BAGGINs tradicionais: Sorteia as colunas também

## Algoritmo:

- 1) Sorteie B conjuntos de observações da base D
- 2) Para cada conjunto b de B, sorteie m variáveis de D
- 3) Para cada uma das B sub-bases geradas por (b, m) construa uma árvore de decisão
- 4) Para previsão final, agregue as previsões individuais de cada uma das B árvores.





# Hiperparâmetros e Overfitting

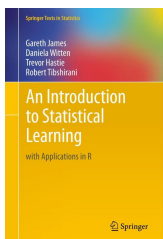
Hiperparâmetros do pacote `randomForest` do R

**ntree** – Número de árvores (amostras bootstrap) para treinar. Não afeta muito o overfitting.

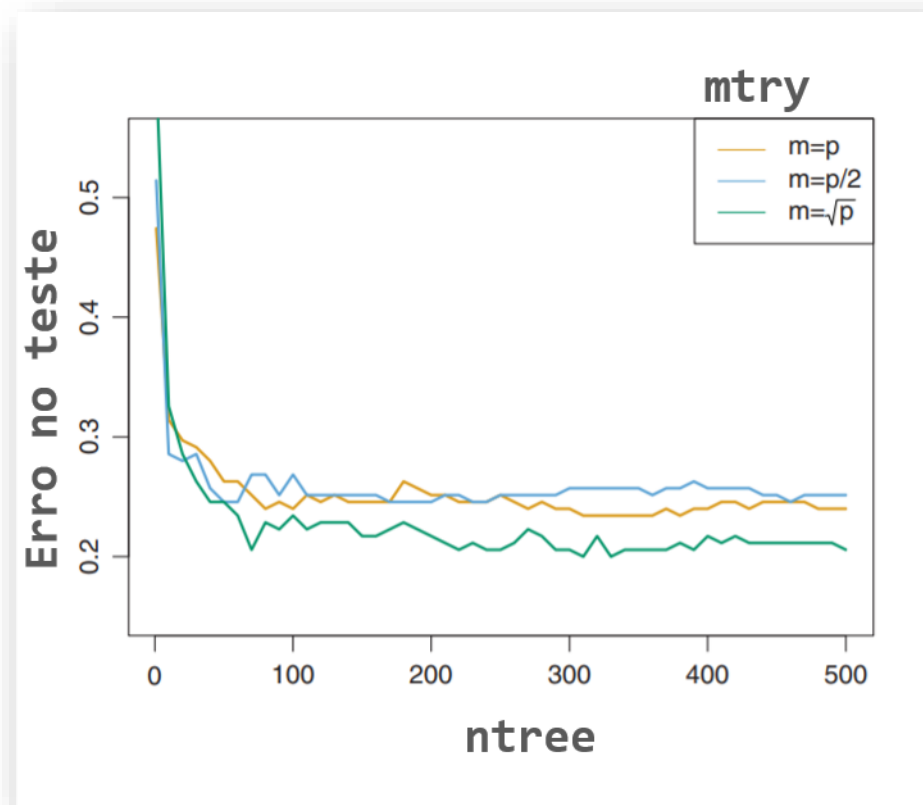
**mtry** – Quantidade de variáveis (colunas) sorteadas por árvore. Tem que testar via cross-validation, pois é afetado pela razão entre #variáveis boas e #ruído.

**nodesize** – Análogo ao **minsplit** do `rpart`. Quantidade mínima de observações no nó para poder dividir.

OBS: random forest não usa CP. Ele permite que as árvores cresçam indeterminadamente, condicionadas apenas pelo **nodesize**.



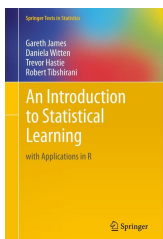
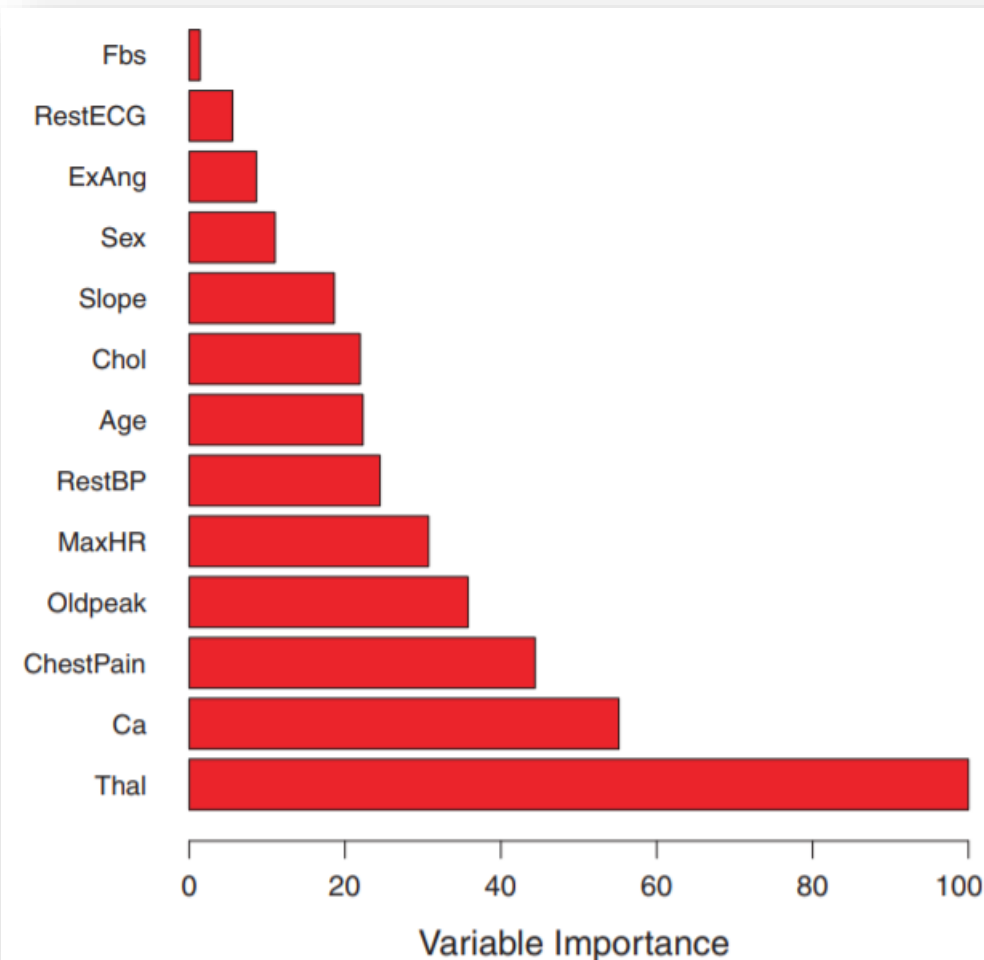
ISL  
p 322





# Importância das Variáveis (*variable importance*)

- A importância de uma certa variável X é calculada (na maioria das vezes) pela média dos ganhos de informação das quebras feitas por aquela variável.
- O gráfico ao lado mostra uma escala de 0 a 100. É a maneira como se costuma apresentar a importância da variável uma vez que a média do ganho de informação é difícil de interpretar.
- No R: `varImp(modelo)`



ISL  
p 319



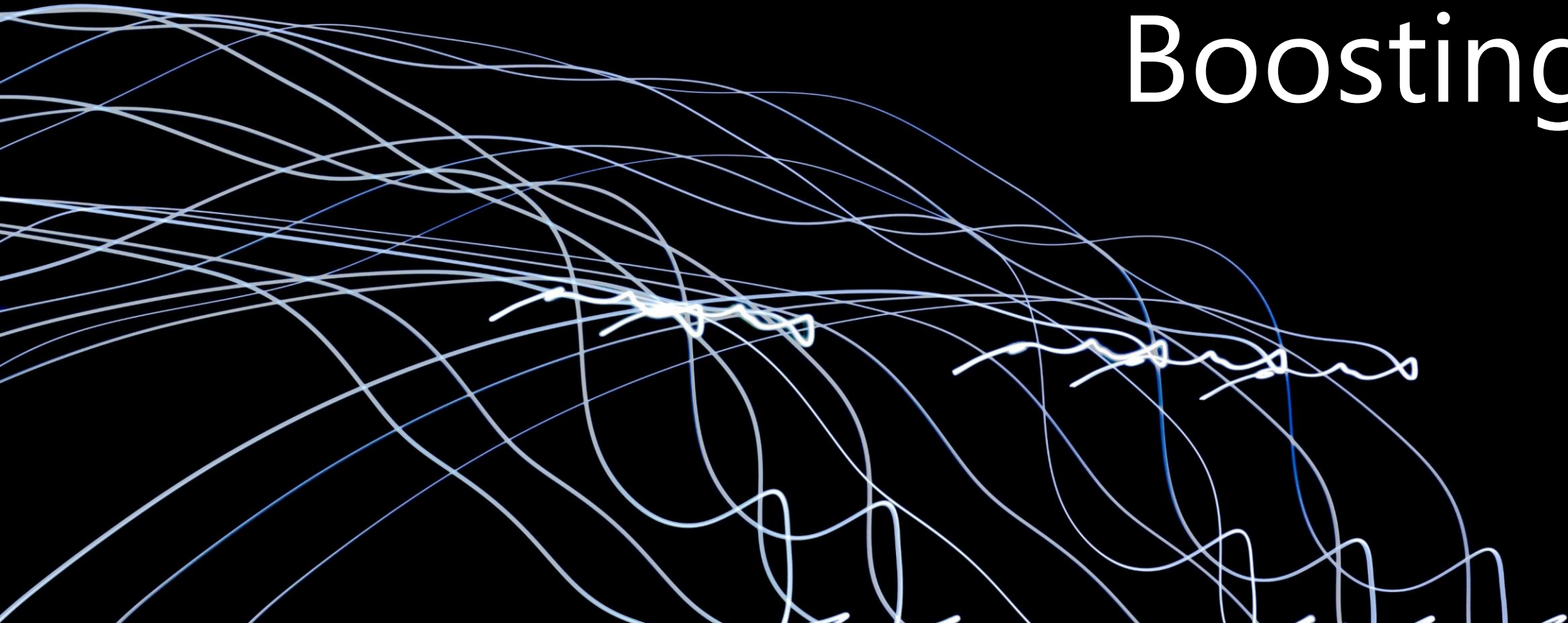


```
library(randomForest)
```

Ao R...



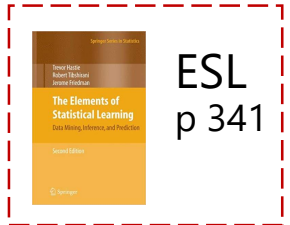
# Gradient Boosting





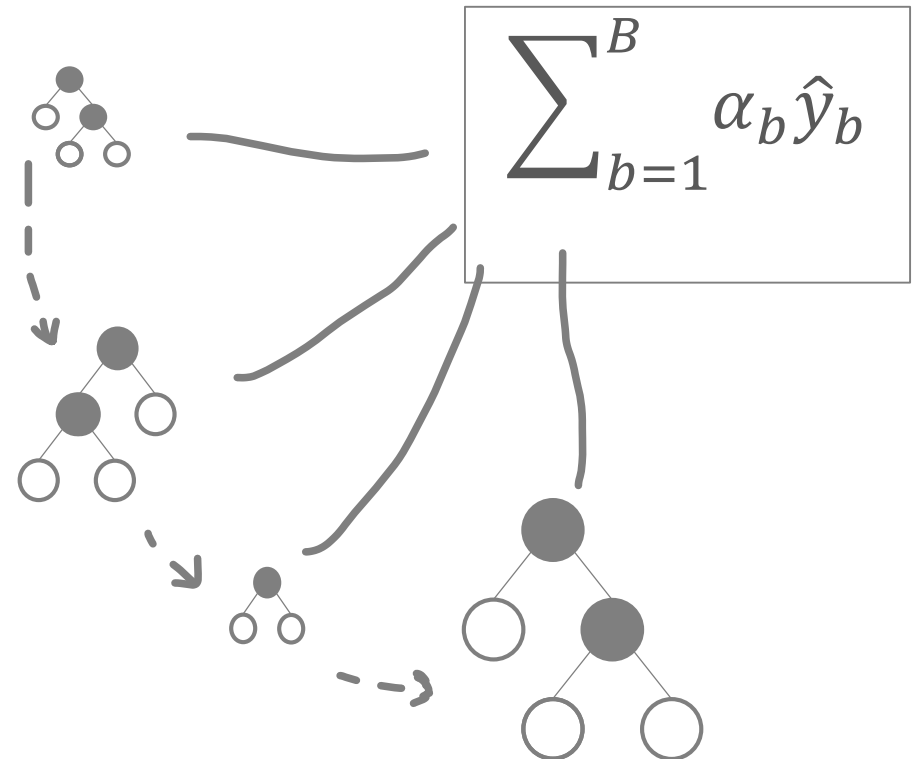
# Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.



Modelo proposto: Expansão de bases de funções

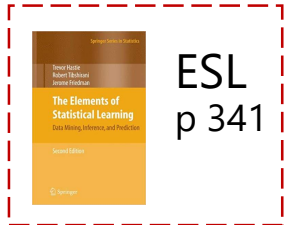
$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$$





# Boosting

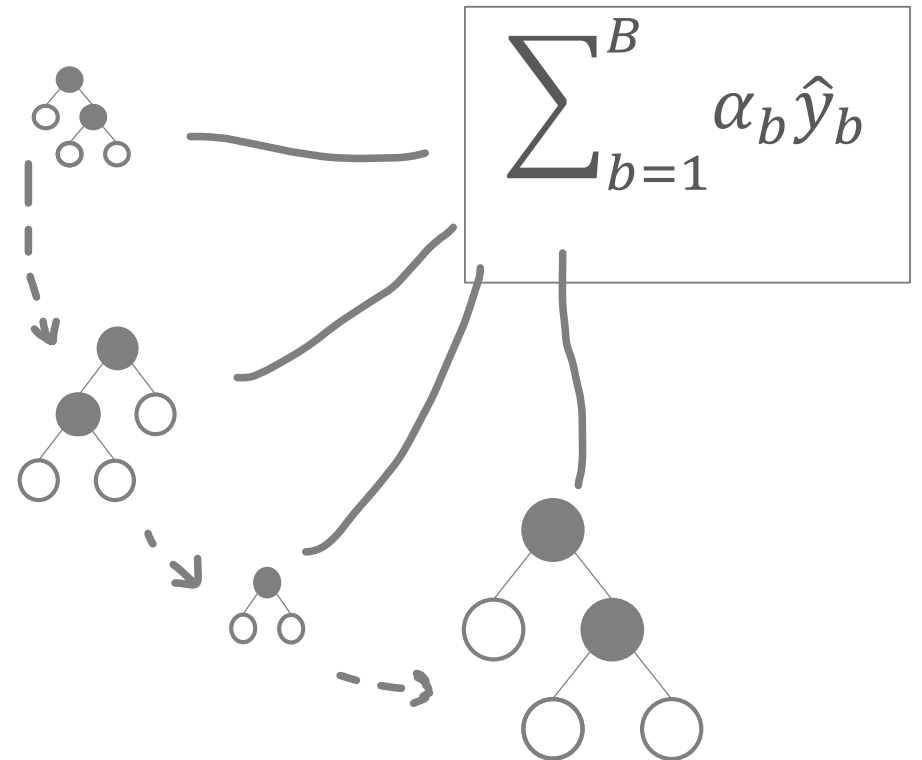
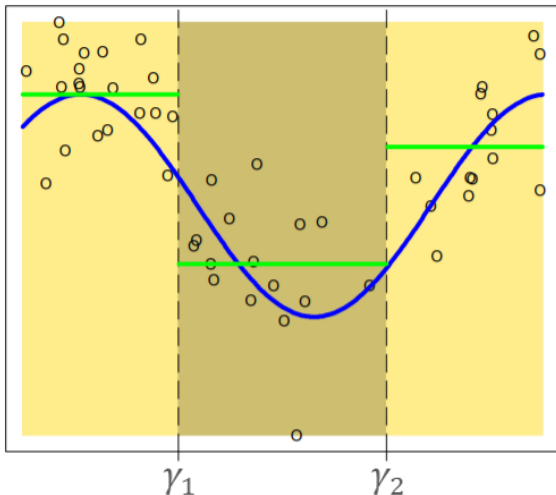
- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.



Modelo proposto: Expansão de bases de funções

$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$$

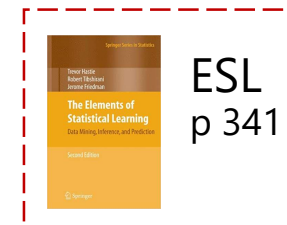
EXEMPLO:  $b(x, \gamma_1) = I(x < \gamma_1)$ ,  $b(x, \gamma_2) = I(x < \gamma_2)$ ,  $b(x, \gamma_3) = I(x < \gamma_3)$





# Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.



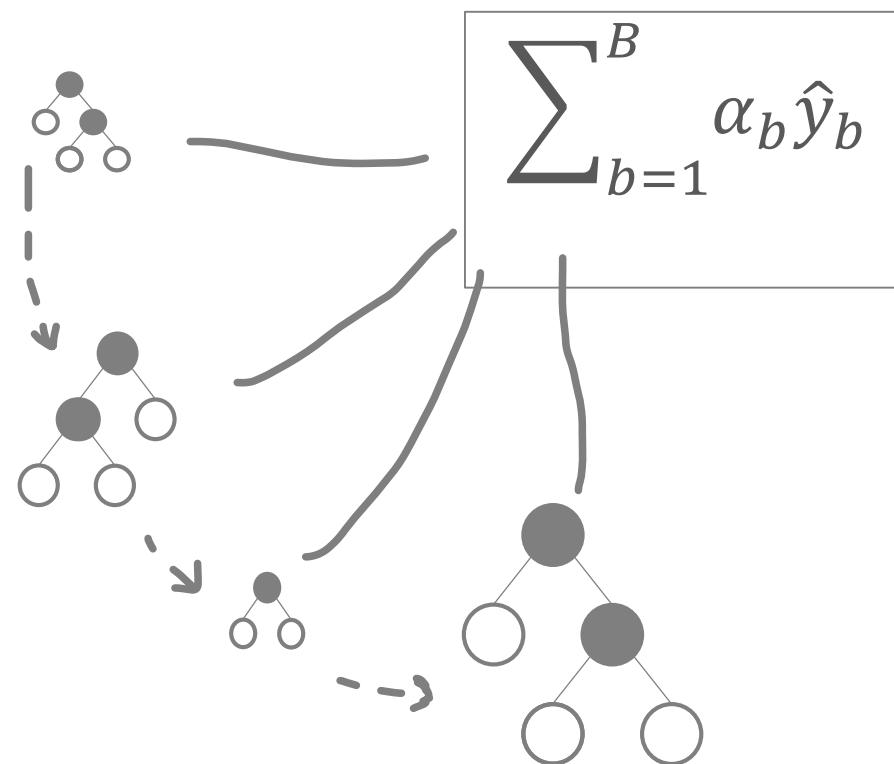
## Modelo proposto: Expansão de bases de funções

$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$$

EXEMPLO:  $b(x, \gamma_1) = I(x < \gamma_1)$ ,  $b(x, \gamma_2) = I(x < \gamma_2)$ ,  $b(x, \gamma_3) = I(x < \gamma_3)$

### PROBLEMAS:

- No exemplo  $\gamma_1, \gamma_2$  e  $\gamma_3$  foram fixados. Gostaríamos de encontrar conjuntamente  $(\beta_m, \gamma_m)$  que melhor se ajustasse, mas é virtualmente impossível na maioria das vezes.





# Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.

## Forward Stage-wise Modeling (coração do Boosting):

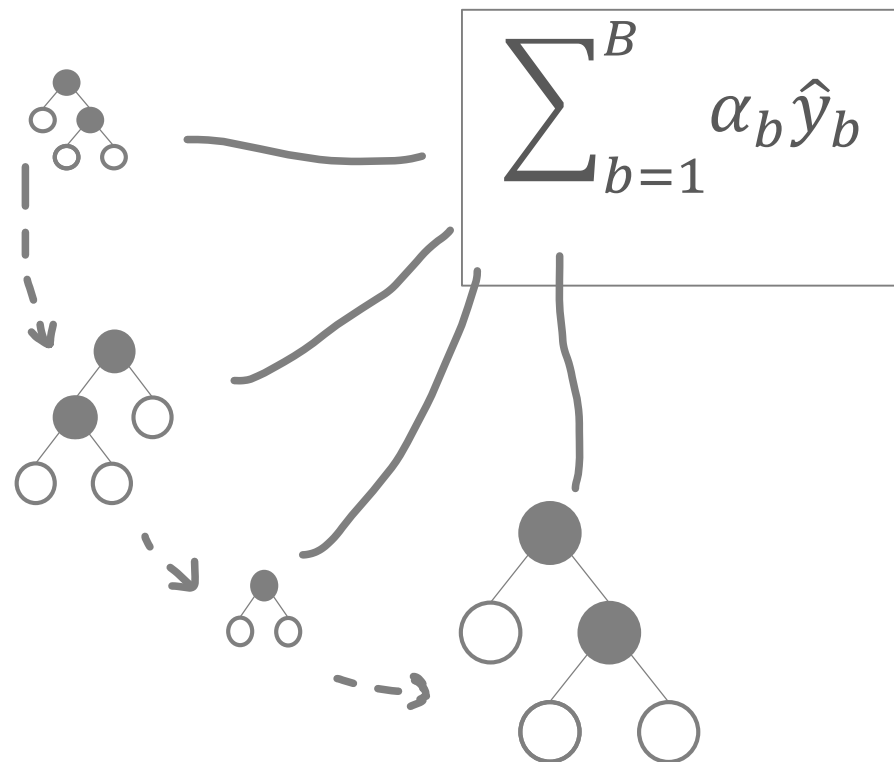
1) Initialize  $f_0(x) = 0$

2) Para  $m = 1$  a  $M$ :

a) Calcule:

$$(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$$

b) Atribua  $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$





# Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.

## Forward Stage-wise Modeling (coração do Boosting):

1) Inicialize  $f_0(x) = 0$

2) Para  $m = 1$  a  $M$ :

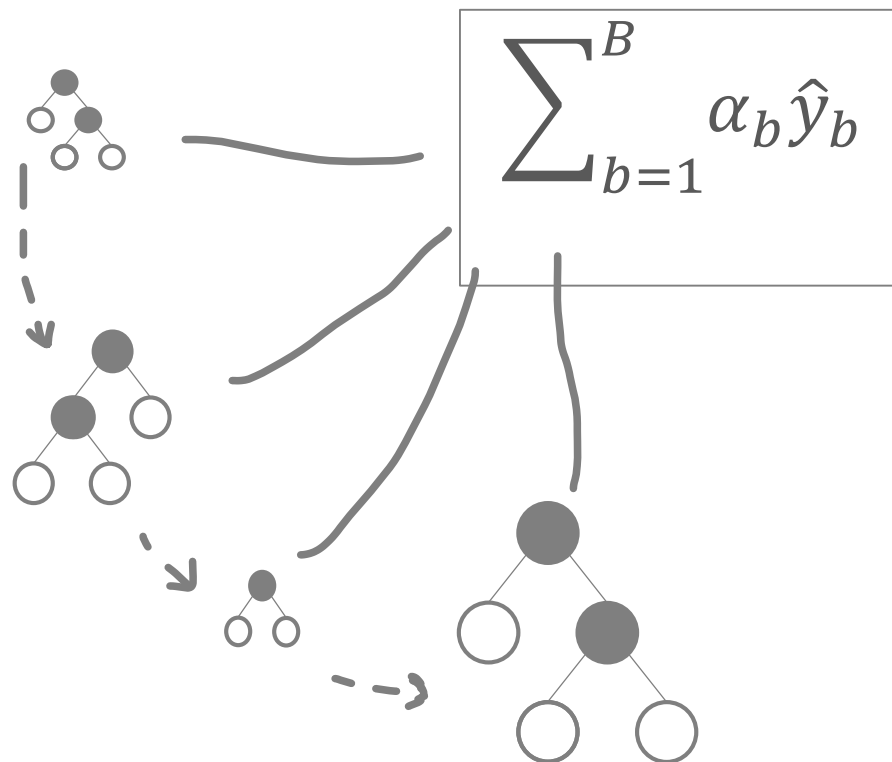
a) Calcule:

$$(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$$

b) Atribua  $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$

EXEMPLO DE LOSS FUNCTION:  $L(y_i, f(x)) = (y_i - f(x))^2$

$$\begin{aligned} L(y_i, f(x)) &= (y_i - f_{m-1}(x_i) - \beta b(x_i; \gamma))^2 \\ &= (r_i - \beta b(x_i; \gamma))^2 \end{aligned}$$





# Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.

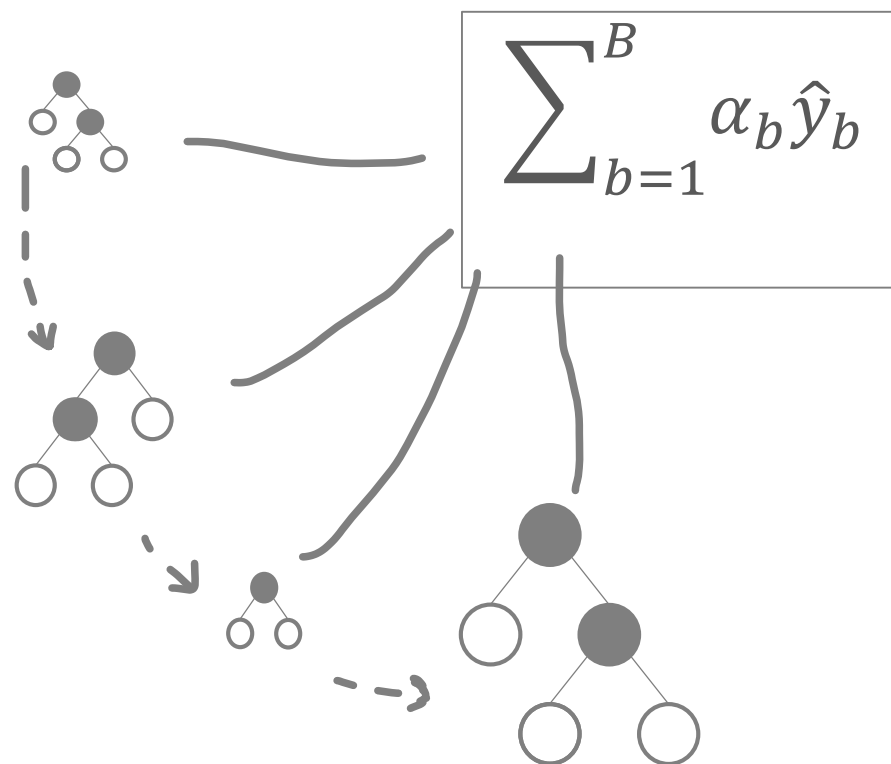
## AdaBoost (versão de classificação binária):

- 1) LOSS utilizada:  $L(y, f(x)) = \exp(-y * f(x))$ . Em que  $Y \in \{-1, 1\}$ .
- 2) As funções  $b(x, \gamma)$  agora serão árvores  $T_m(x)$  que irão retornar ou 1 ou -1.
- 3) Temos que minimizar, então (para ser adicionado em cada passo m):

$$(\beta_m, T_m) = \arg \min_{\beta, T} \sum_{i=1}^N \exp(-y_i (f_{m-1}(x_i) + \beta T(x_i)))$$

- 4) Solução:  $T_m = \arg \min_T \sum_{i=1}^N w_i^{(m)} I(y_i \neq T(x_i))$        $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$

$$\beta_m = \frac{1}{2} \log \left( \frac{1 - \text{err}_m}{\text{err}_m} \right)$$

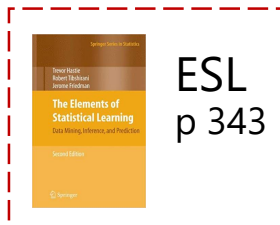






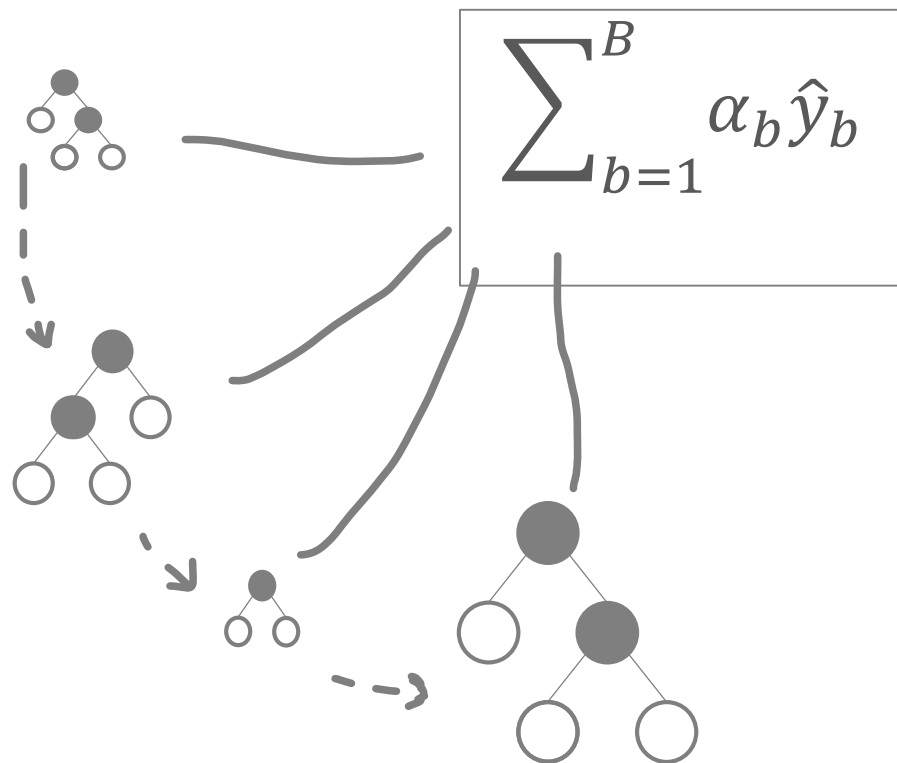
# Boosting

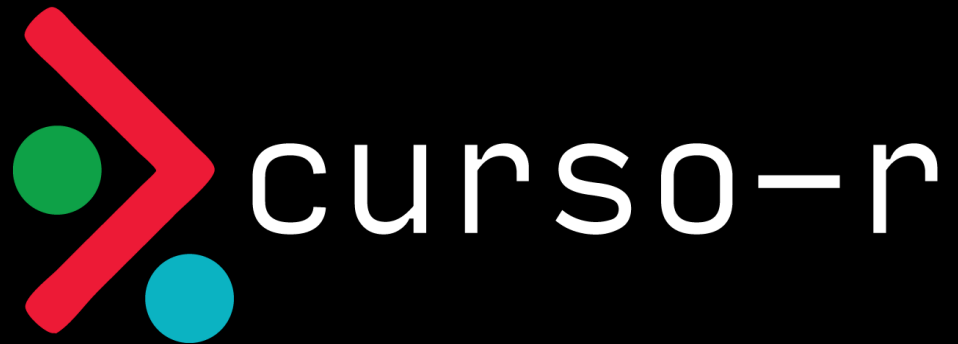
- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.



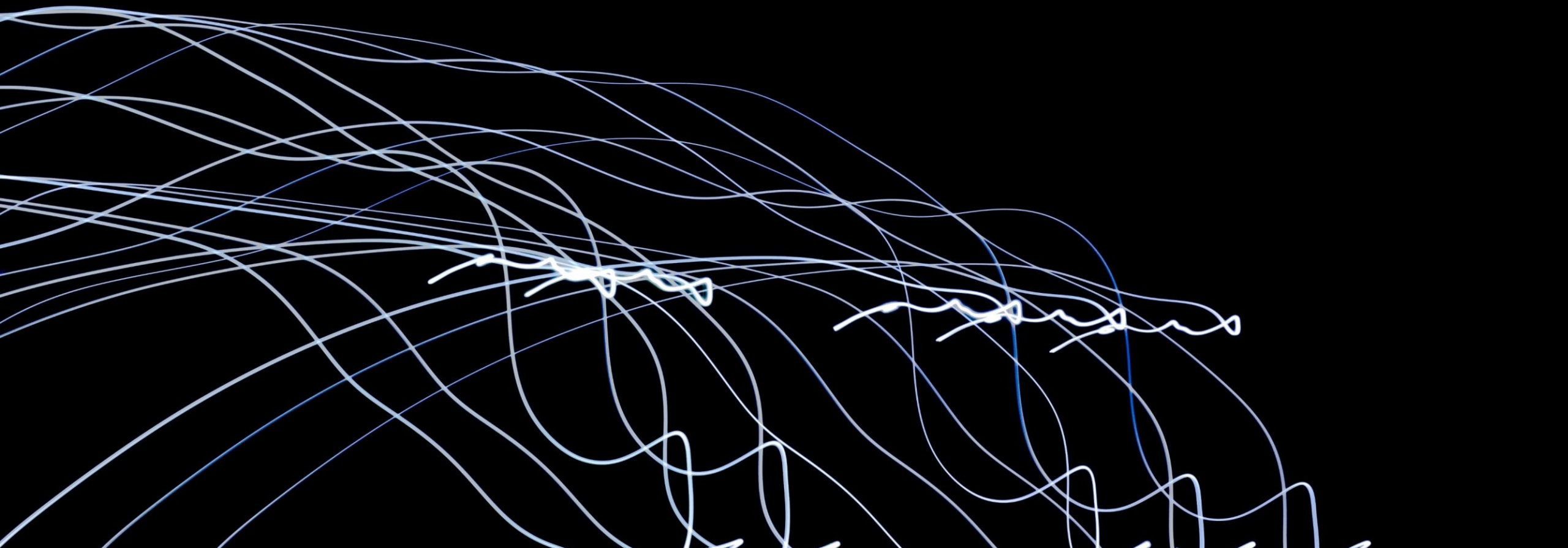
## Algoritmo AdaBoost (versão de classificação binária):

- 1) Codifique  $Y \in \{-1, 1\}$ .
- 2) Inicializa-se pesos  $w_i = \frac{1}{N}, i = 1, 2, \dots, N$
- 3) Para  $m = 1$  a  $M$ :
  - a) Ajuste uma árvore  $T_m(x)$  usando os pesos  $w_i$ .
  - b) Calcule o erro de  $m$ :  $err_m = \frac{\sum_{i=1}^N w_i I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i}$
  - c) Compute  $\alpha_m = \log\left(\frac{1-err_m}{err_m}\right)$
  - d) Atualize os pesos:  $w_i \leftarrow w_i \cdot \exp\left(\alpha_m \cdot I(y_i \neq T_m(x_i))\right), i = 1, \dots, N$
- 4) Previsão:  $sign[\sum_{m=1}^M \alpha_m T(x)]$





# XGBoost





# XGBoost

- XGBoost é uma implementação eficiente do Gradient Boost.
- Ele também é uma sofisticação. O XGBoost traz de volta um monte de hiperparâmetros de regularização.
- Top 2 no Ranking de Algoritmos que mais ganharam Kaggle.
- No R: library(xgboost)

$$\begin{aligned}\text{obj}^{(t)} &\approx \sum_{i=1}^n [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \\ &= \sum_{j=1}^T [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T\end{aligned}$$

## Hiperparâmetros do pacote `xgboost` do R

**nrounds** – Número de árvores.

**max\_depth** – Profundidade máxima da árvore.

**eta** – Tamanho do passo em busca do mínimo da função de custo. Quanto menor, mais devagar. Aconselha-se aumentar o número de árvores junto!

**gamma** – Parâmetro regularizador. Análogo ao CP do `rpart`.

**colsample\_bytree** – Qtd de variáveis sorteadas por árvore.

**subsample** – Quantidade de observações por árvore.

**min\_child\_weight** – Observações mínimas nas folhas.

**lambda** - Regularizador L2. Controla o tamanho dos parâmetros das folhas como se fosse o RIDGE.

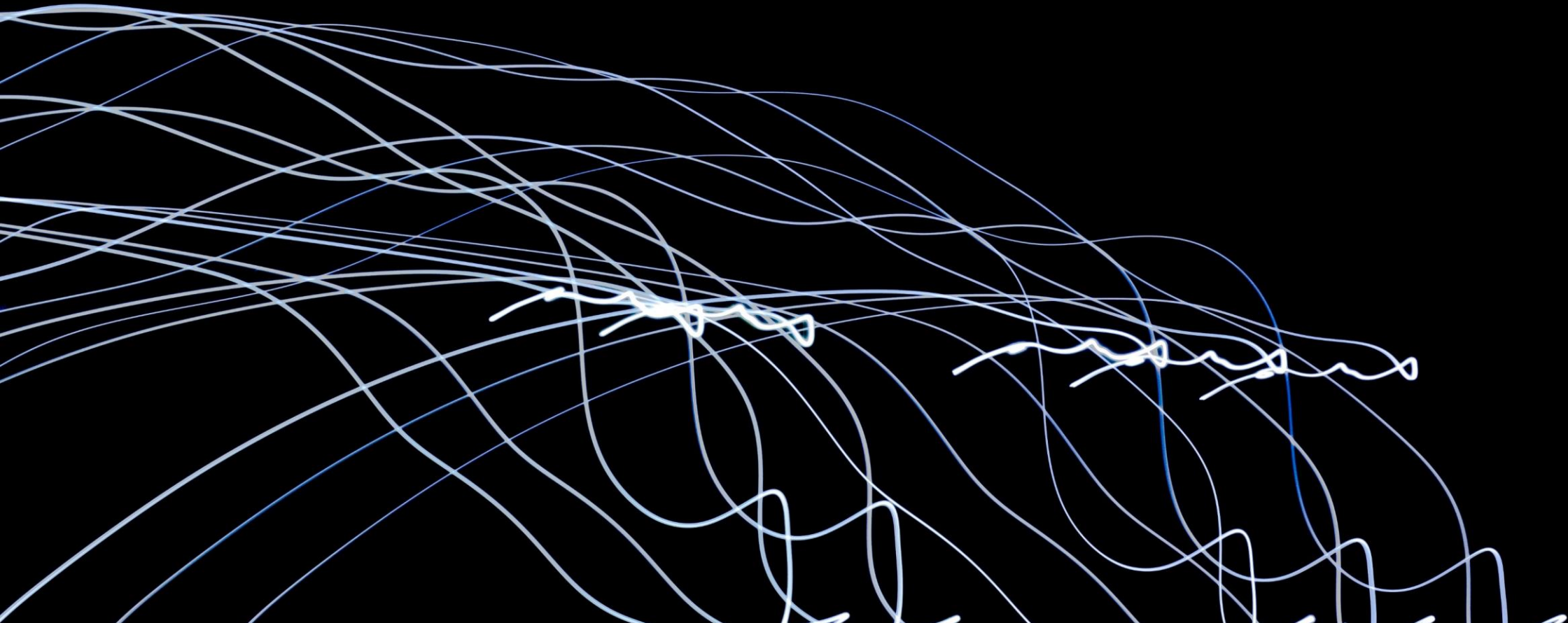


```
library(xgboost)
```

Ao R...



Misc





# Resumão de R

```
train_control <- trainControl(  
  method = "cv"  
  ,number = 5  
  ,verboseIter = TRUE  
  ,classProbs = TRUE  
  ,summaryFunction = myTwoClassSummary  
  ,allowParallel = FALSE  
)
```

```
library(rpart)  
  
tune_grid_tree <- expand.grid(  
  cp = c(0.1, 0.5, 1, 2, 5, 10)  
)  
  
modelo_tree <- train(  
  Survived ~ .  
  ,data = titanic_train  
  ,method = "rpart"  
  ,trControl = train_control  
  ,tuneGrid = tune_grid_tree  
)
```

```
library(randomForest)  
  
tune_grid_rf <- expand.grid(  
  mtry = c(5, 15, 30, 60, 120, 200, 500)  
)  
  
modelo_rf <- train(  
  Survived ~ .  
  ,data = titanic_train  
  ,method = "rf"  
  ,trControl = train_control  
  ,tuneGrid = tune_grid_rf  
)
```

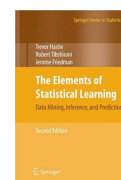
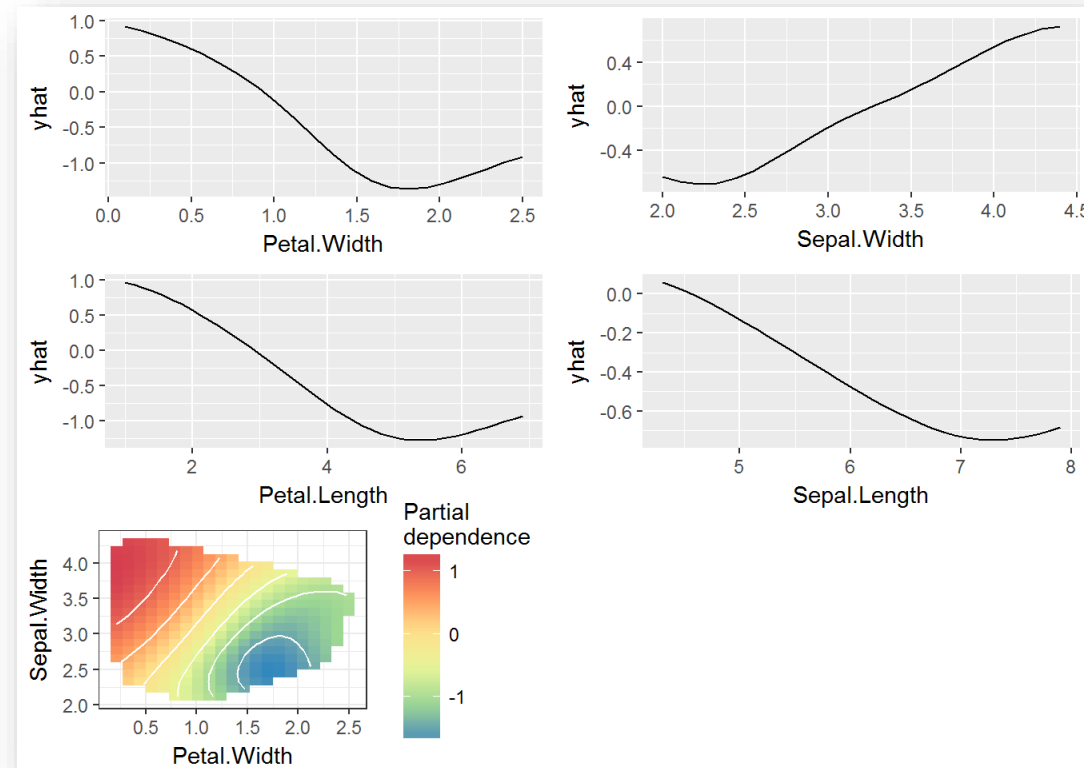
```
library(xgboost)  
  
tune_grid_xgb <- expand.grid(  
  nrounds = c(500, 1000, 1500, 2000),  
  max_depth = c(3, 6, 9, 12),  
  eta = c(0.01, 0.05, 0.1),  
  gamma = c(0.01, 0.1, 0.5, 1),  
  colsample_bytree = c(0.6, 0.7, 0.8, 0.9),  
  min_child_weight = c(1, 10, 100, 1000),  
  subsample = c(0.6, 0.7, 0.8, 0.9)  
)  
  
modelo_xgb <- train(  
  Survived ~ .  
  ,data = titanic_train  
  ,method = "xgbTree"  
  ,trControl = train_control  
  ,tuneGrid = tune_grid_xgb  
)
```



# Gráfico de Dependência Parcial (*partial dependence plot*)

**pdp** – Partial Dependence Plot: Serve para mostrar o efeito (marginal) de uma variável explicativa na estimativa do modelo.

- É possível fazer o efeito conjunto de duas ou mais variáveis.
- **Motivação:** Random Forest, Boosting, Redes Neurais, SVM e tantos outros modelos são difíceis de serem interpretados diretamente pelos seus parâmetros.
- **Receita:** para cada observação da sua base, crie N linhas a mais alterando os valores de uma variável enquanto mantém as demais características fixas. Então, calcule as respectivas estimativas.
- No R: pacotes **pdp** ou **plotmo**



ESL  
p 369



# Do R para o SQL

<https://tidypredict.netlify.com/>

<https://rviews.rstudio.com/2018/11/07/in-database-xgboost-predictions-with-r/>