Mineração de Dados

Aula 8: Árvores

Rafael Izbicki

Revisão

Vimos que a função de risco é dada por

$$R(g) := \mathbb{E}[\mathbb{I}(Y \neq g(\mathbf{X}))] = \mathbb{P}\left(Y \neq g(\mathbf{X})\right),$$

Nem sempre tal função nos traz toda informação sobre g. É comum considerar matrizes de confusão.

Examplo de matriz de confusão

	Valor verdadeiro	
Valor Predito	Y=0	Y=1
Y=0	VN	FN
Y=1	FP	VP

V: Verdadeiro / F: Falso
P: Positivo / N: Negativo

Com base nessa tabela, define-se

- Sensibilidade: VP/(VP+FN) (dos pacientes doentes, quantos foram corretamente identificados?)
- Especificidade: VN/(VN+FP) (dos pacientes não doentes, quantos foram corretamente identificados?)

Um outro problema relacionado a isso: se $Y = 1$ é raro, em geral teremos $\mathbb{P}(Y = 1|x)$ baixo.

Assim, usar um corte de $1/2$ pode não ajudar: a regra $g(x) = \mathbb{I}(\mathbb{P}(Y = 1|x) \geq 1/2)$ nos levará a sempre decidir por $Y = 0$, mesmo se essas probabilidades estiverem bem estimadas.

Na prática, para evitar isso é comum buscar cortes diferentes de $1/2$, i.e., buscam-se regras do tipo

$$g(x) = \mathbb{I}(\mathbb{P}(Y = 1|x) \geq K)$$

para diferentes cortes K

Curva ROC

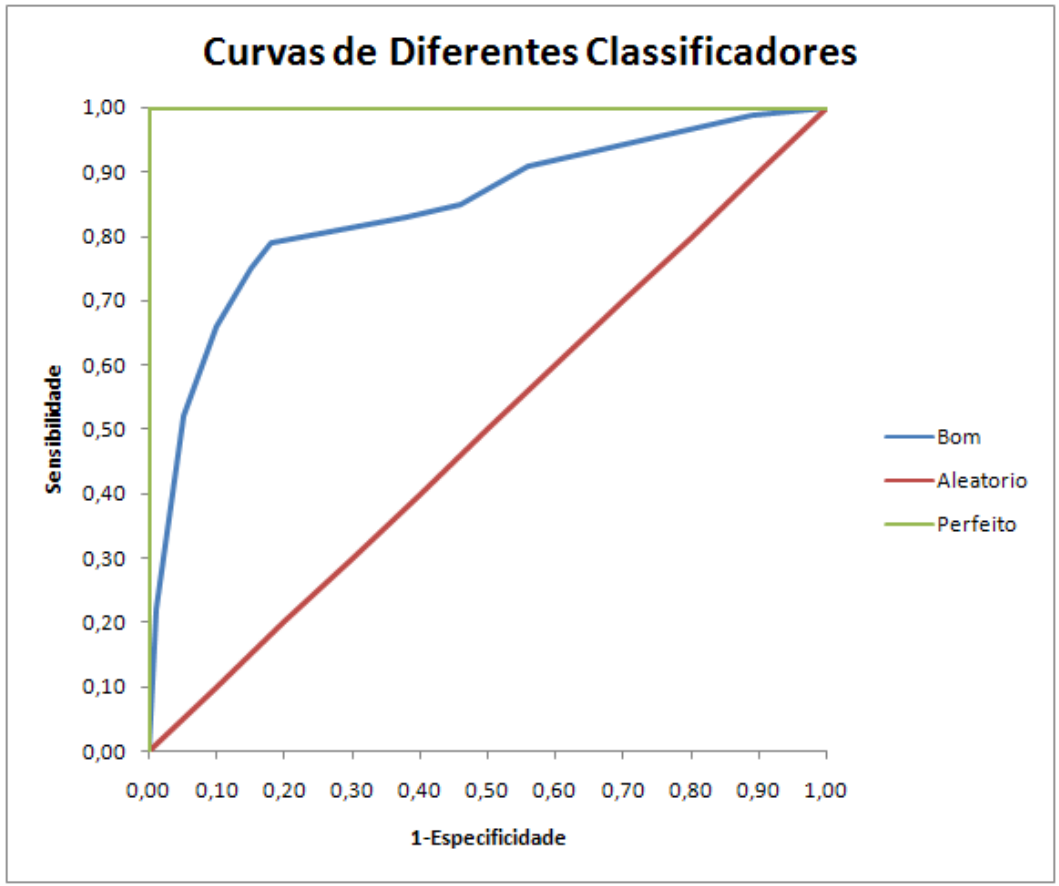


Gráfico da Sensibilidade vs 1-Especificidade para diferentes K's.

É comum escolher K que maximize "Sensibilidade+Especificidade"

Análise Discriminante

Vamos lembrar o Teorema de Bayes mais uma vez:

$$\mathbb{P}(Y = c|\mathbf{x}) = \frac{f(\mathbf{x}|Y = c)\mathbb{P}(Y = c)}{\sum_{s\in\mathcal{X}} f(\mathbf{x}|Y = s)\mathbb{P}(Y = s)}$$

Na análise discriminante, supomos que o vetor $\mathbf{X}$, dado Y, tem distribuição normal multivariada.

Existem duas formas de análise discriminante mais comuns:

- Análise Discriminante Linear
- Análise Discriminante Quadrática

Análise Discriminante Linear

Assumimos que

$$\mathbf{X} = (X_1, \ldots, X_d) | Y = c \sim Normal(\mu_c, \Sigma),$$

i.e.,

$$f(\mathbf{x}|Y = c) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-(\mathbf{x}-\mu_c)'\Sigma^{-1}(\mathbf{x}-\mu_c)}$$

Regra de decisão:

$$2\mathbf{x}'\widehat{\Sigma}^{-1}\widehat{\mu}_1 - 2\mathbf{x}'\widehat{\Sigma}^{-1}\widehat{\mu}_0 \geq K''$$

Análise Discriminante Quadrática

Assumimos que

$$\mathbf{X} = (X_1, \ldots, X_d) | Y = c \sim \mathit{Normal}(\mu_c, \Sigma_c),$$

i.e.,

$$f(\mathbf{x}|Y = c) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_c|}} e^{-(\mathbf{x}-\mu_c)' \Sigma_c^{-1} (\mathbf{x}-\mu_c)}$$

Mesma suposição que a Linear, mas com variâncias diferentes em cada grupo.

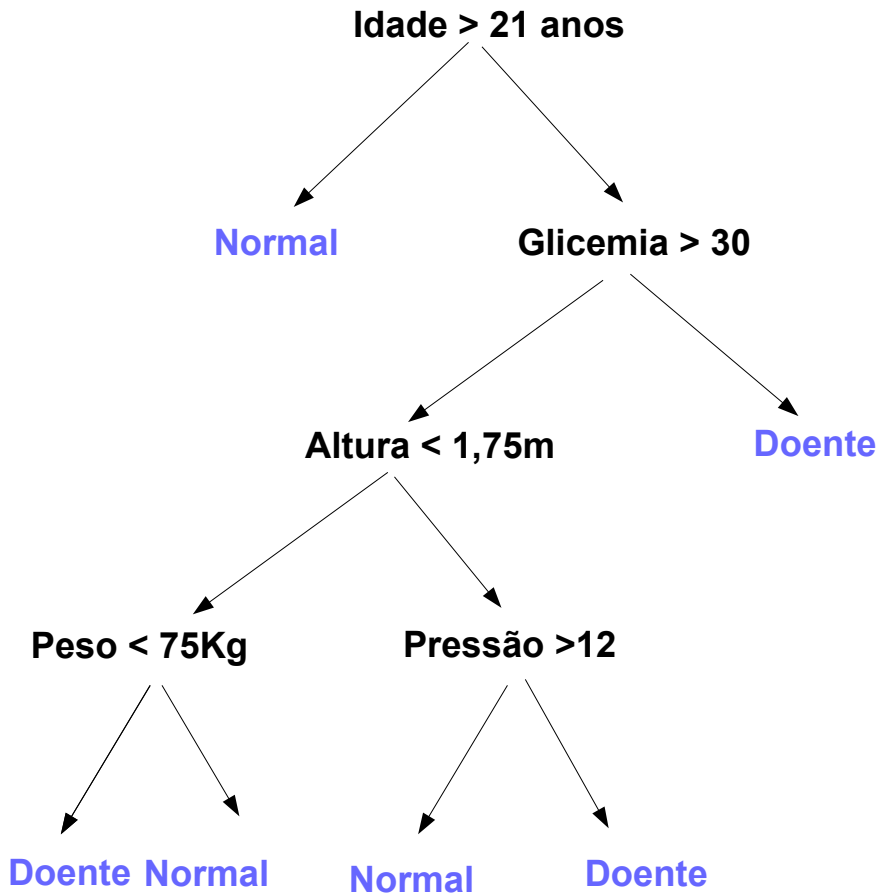
Regra de decisão:

$$-(\mathbf{x} - \widehat{\mu}_1)' \widehat{\Sigma_1}^{-1} (\mathbf{x} - \widehat{\mu}_1) + (\mathbf{x} - \widehat{\mu}_0)' \widehat{\Sigma_0}^{-1} (\mathbf{x} - \widehat{\mu}_0) \geq K'$$

www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/

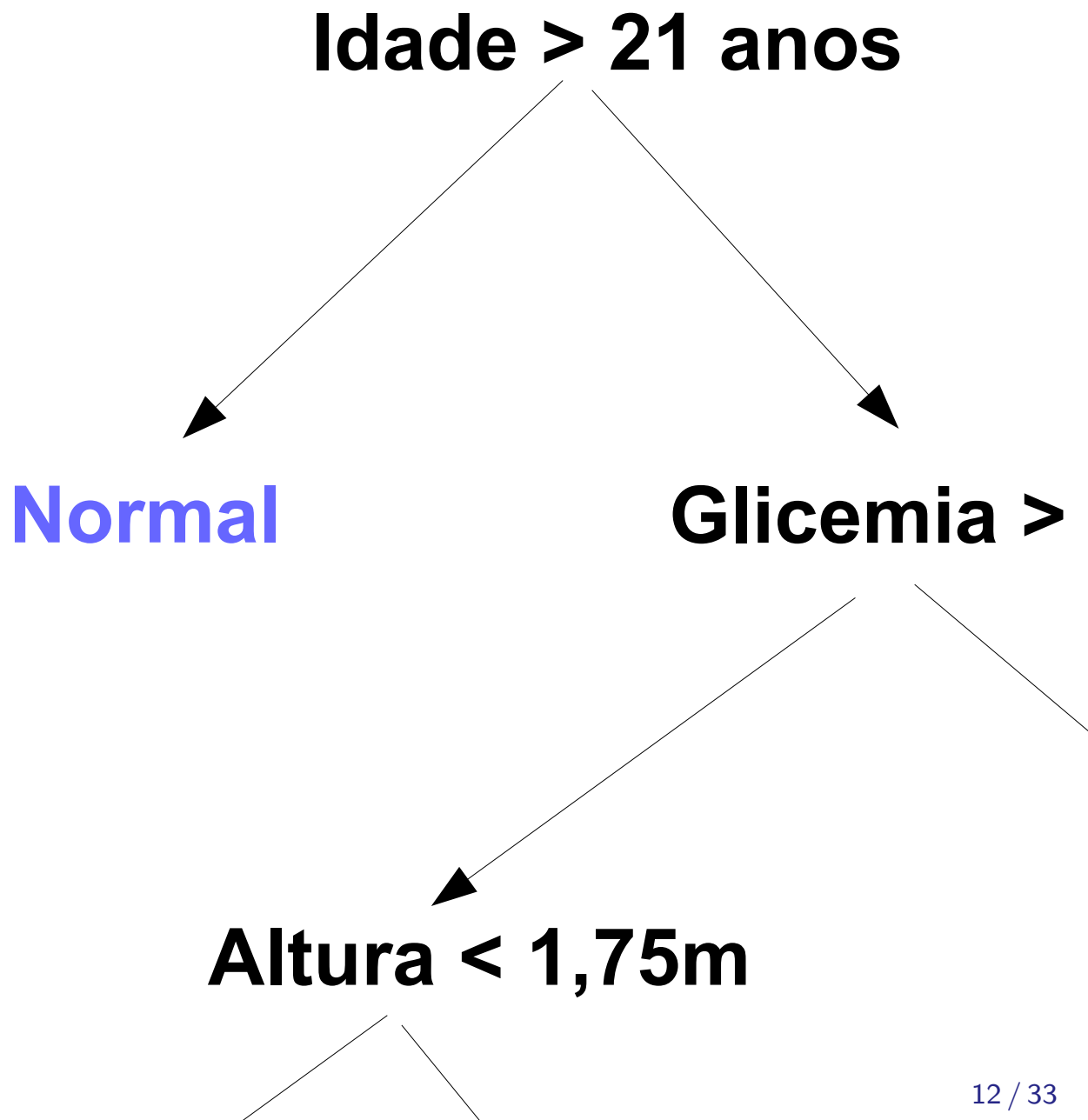
Árvores de Classificação

Não: direita, Sim: esqueda



O que é uma árvore?

Nós, folhas


Idade > 21 anos
Normal
Glicemia >
Altura < 1,75m

Classification Trees – Árvores de Classificação

Árvores são um método de predição simples e útil para interpretação.

A ideia é dividir o espaço das covariáveis em J regiões distintas sem intersecção, $R_1, \ldots, R_J$

Para prever um novo x, vemos a qual região x pertence. Digamos que $x \in R_4$. Nossa predição para Y é então dada por

$$\widehat{Y} = moda(y_i : x_i \in R_4).$$

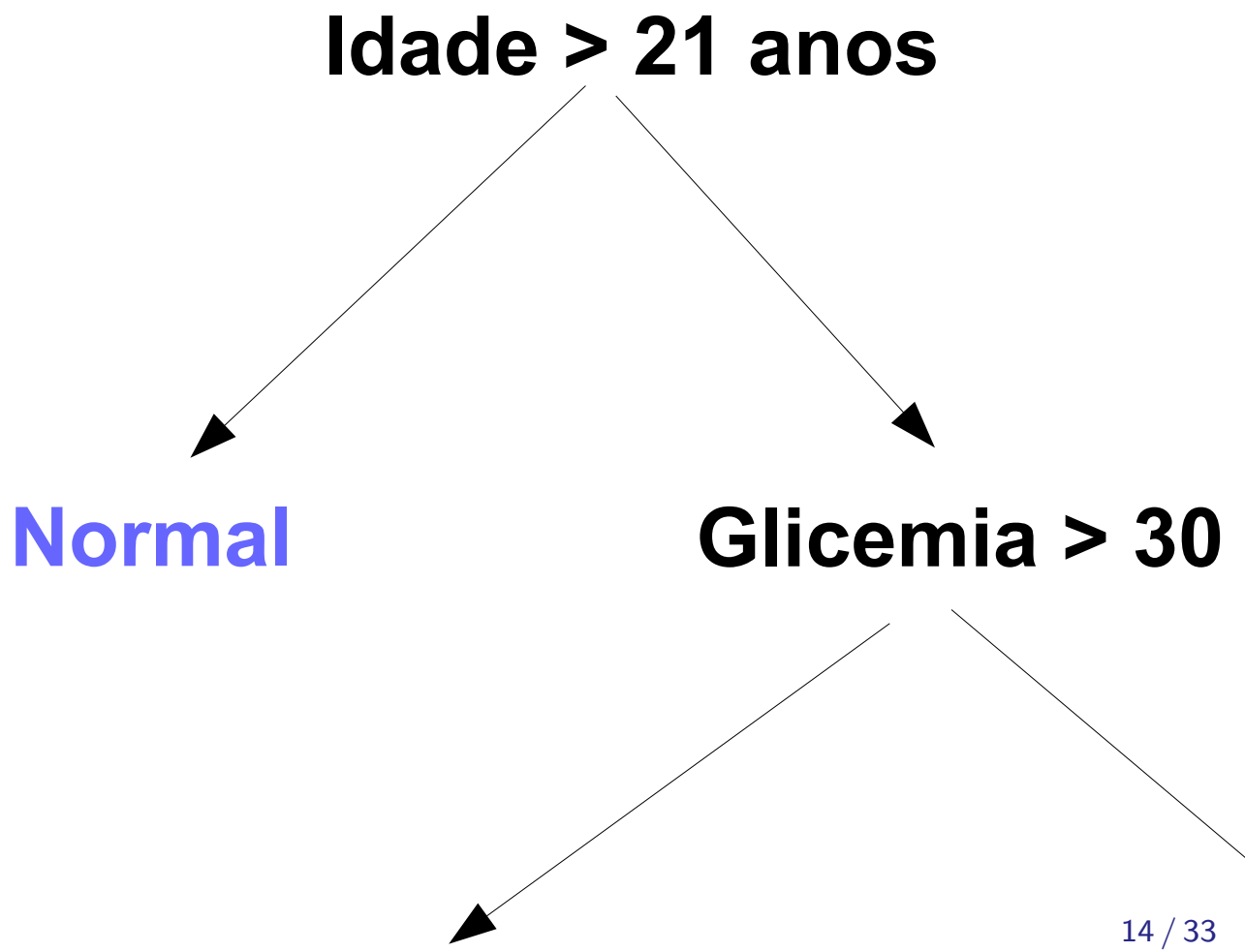
Em geral,

$$g(x) = moda(y_i : x_i \in R^x),$$

onde R^x é a região na qual x cai.

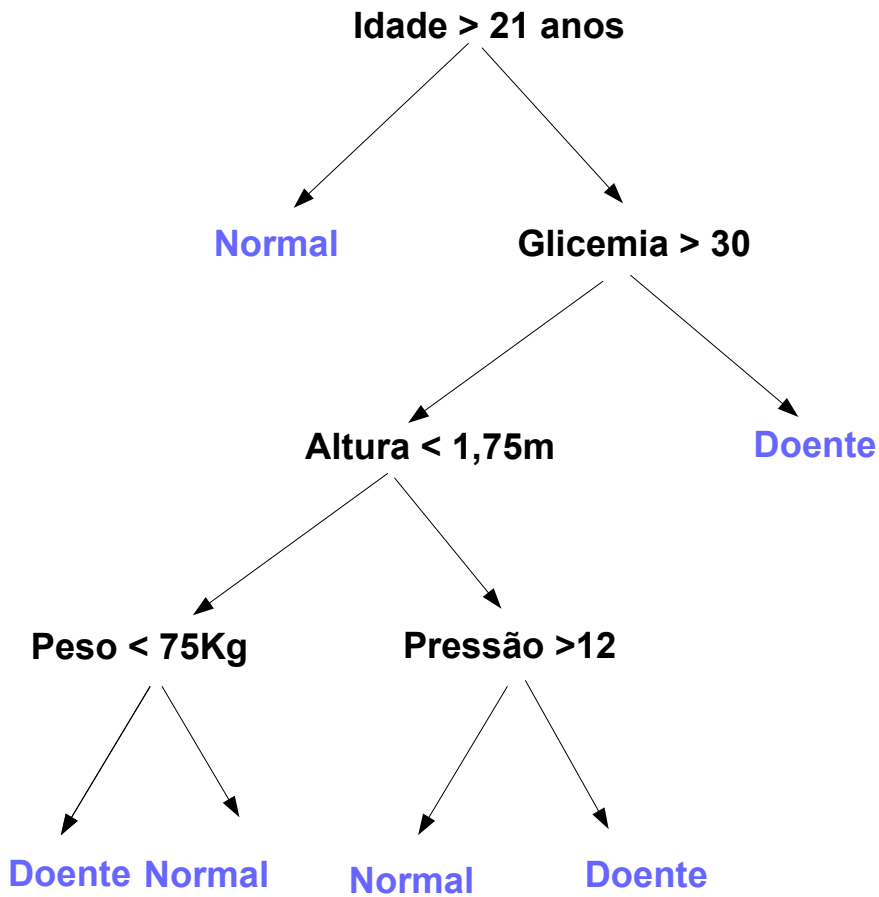
Como determinar as regiões $R_1, \ldots, R_J$?

Utilizamos divisões binárias de cada covariável. Regiões representam retângulos.



Como determinar as regiões $R_1, \ldots, R_J$?

Não: direita, Sim: esqueda



Para criar tal árvore, usamos duas etapas:

1. Criamos uma árvore "grande"
2. Podamos esta árvore

Porque podar uma árvore? Evitar overfitting! Mais detalhes em breve

Etapa 1: Criamos uma árvore "grande"

Antes de começar a criar a árvore, definimos uma medida de quão pura uma dada árvore é. Aqui, usamos o índice de Gini:

$$P(T) = \sum_{R} \sum_{c \in \mathcal{C}} \widehat{p}_{R,c}(1 - \widehat{p}_{R,c})$$

Aqui, R representa uma das regiões induzidas pela árvore, e $\widehat{p}_{R,c}$ é a proporção de elementos que caem na região R e são classificados como sendo da categoria c. T (tree) representa a árvore em questão.

Note que quanto mais "puras" as folhas são, mais próximo de zero essa medida é.

Uma árvore T "pura" tem um valor de $P(T)$ baixo.

Etapa 1: Criamos uma árvore "grande"

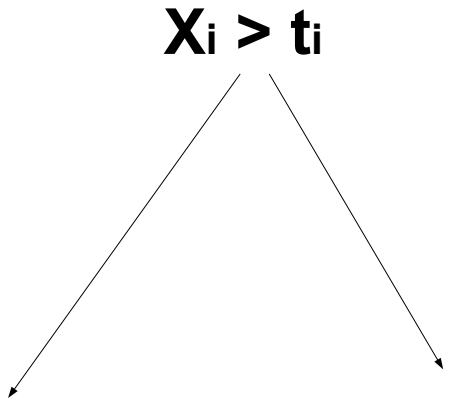
Queremos encontrar a árvore T que tem o menos valor de $P(T)$. Isso é muito difícil computacionalmente, então usaremos uma heurística.

Primeiro, vamos determinar qual variável estará no topo da árvore, e qual será o corte usado.

Para isto, buscamos

$$\arg\min_{x_i, t_i} P(T_{(x_i, t_i)}),$$

onde $T_{(x_i,t_i)}$ é a árvore com apenas um nó (x_i) e com corte t_i:

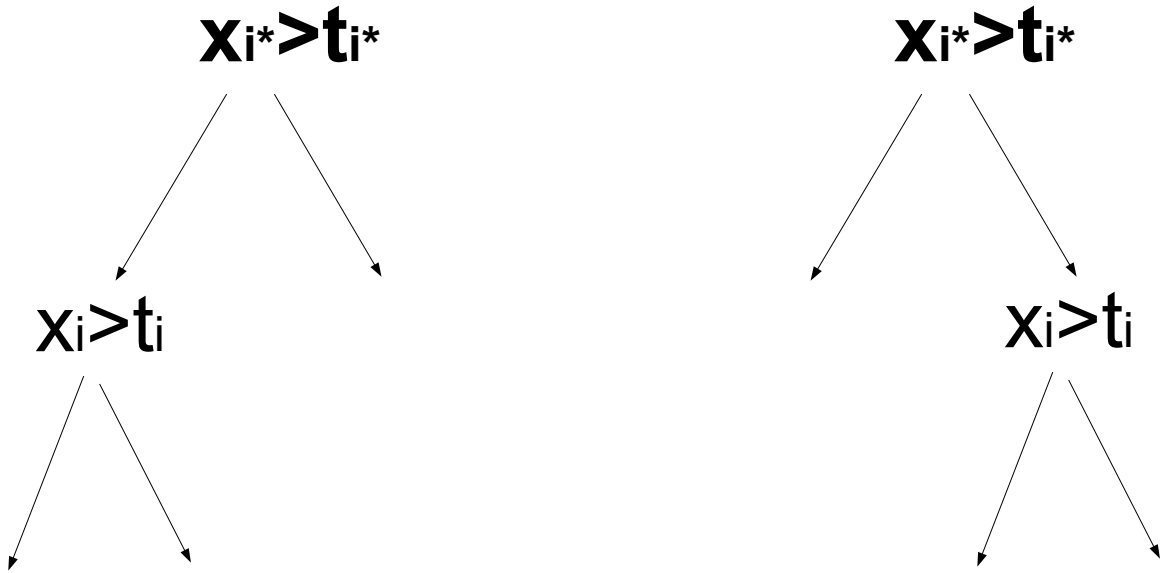


Etapa 1: Criamos uma árvore “grande”

Uma vez que determinamos que variável será usada no primeiro nó e seu corte (x_{i*} e t_{i*}), fazemos o mesmo para o segundo nó (o segundo nó pode ser quaisquer um dos vazios).

Para isto, buscamos

$$\arg\min_{x_i, t_i} P(T_{(x_{i*}, t_{i*}),(x_i, t_i)})$$



Etapa 1: Criamos uma árvore “grande”

Prosseguimos com essa estratégia até criar uma árvore suficientemente grande.

Problema: a árvore criada pode se ajustar demais aos dados (overfitting).

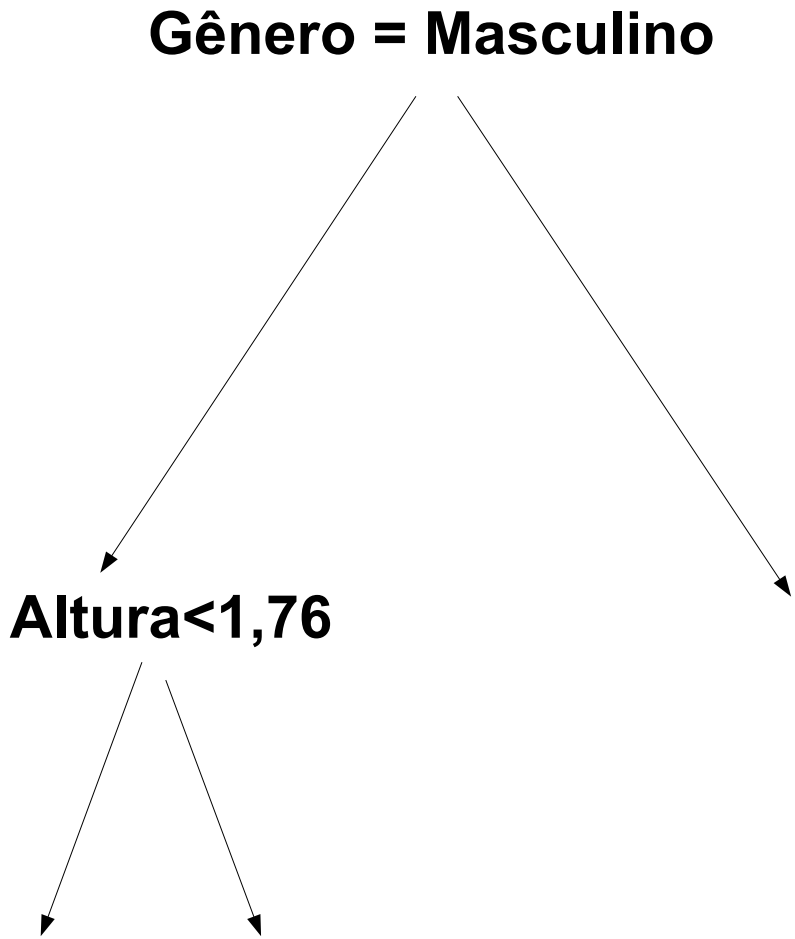
Para isso existe a etapa 2...

Etapa 2: Podamos a árvore

Para isso, retiramos cada nó da árvore, um por vez. Vemos então como o erro preditivo muda no conjunto de validação.

Note que o tamanho da árvore é uma medida da complexidade de g. Assim, este é um tuning parameter.

É fácil adicionar um variável discreta X_i!



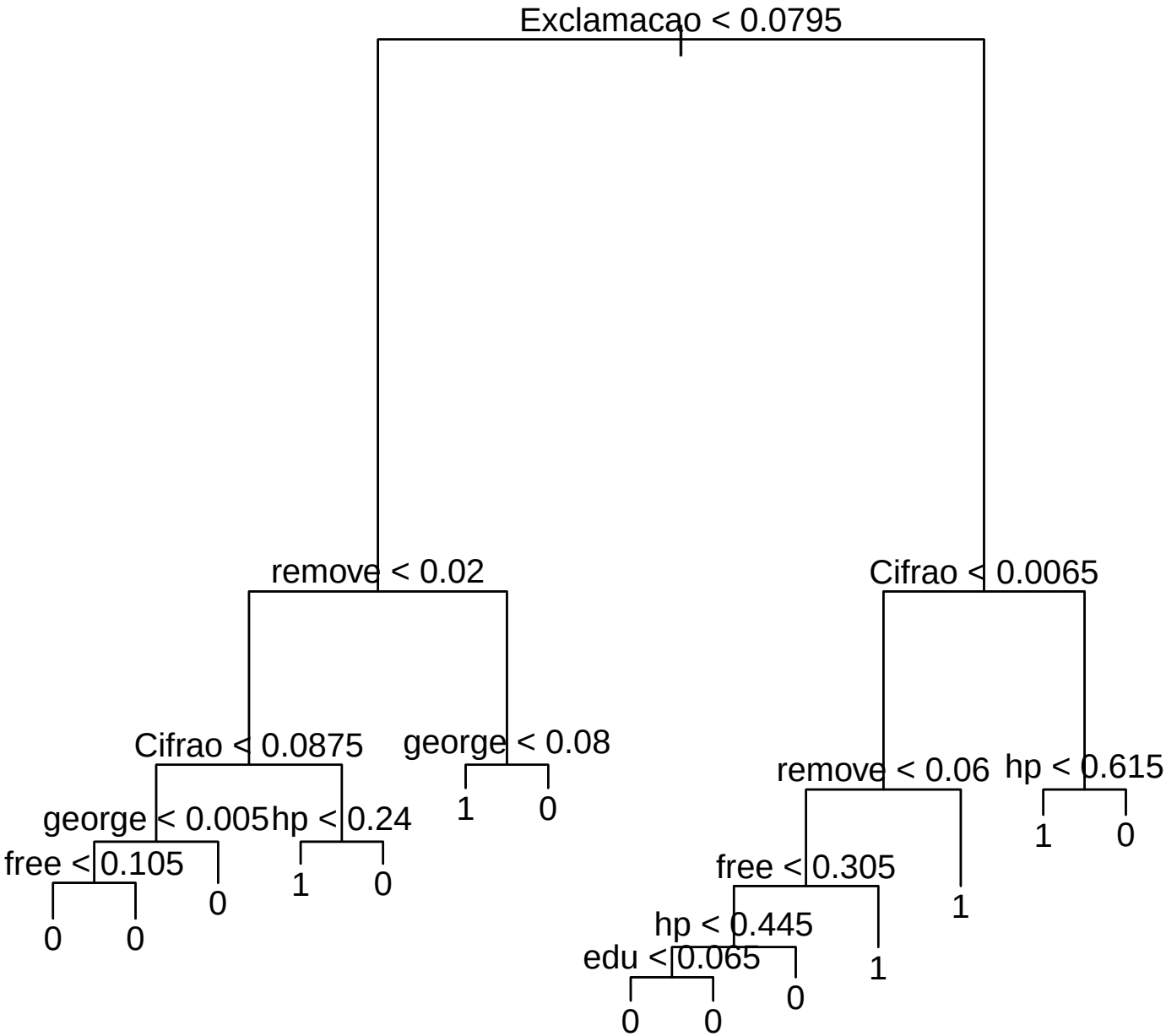
No R

Atenção: você deve converter a resposta para um fator para indicar que ela é qualitativa (use as.factor).

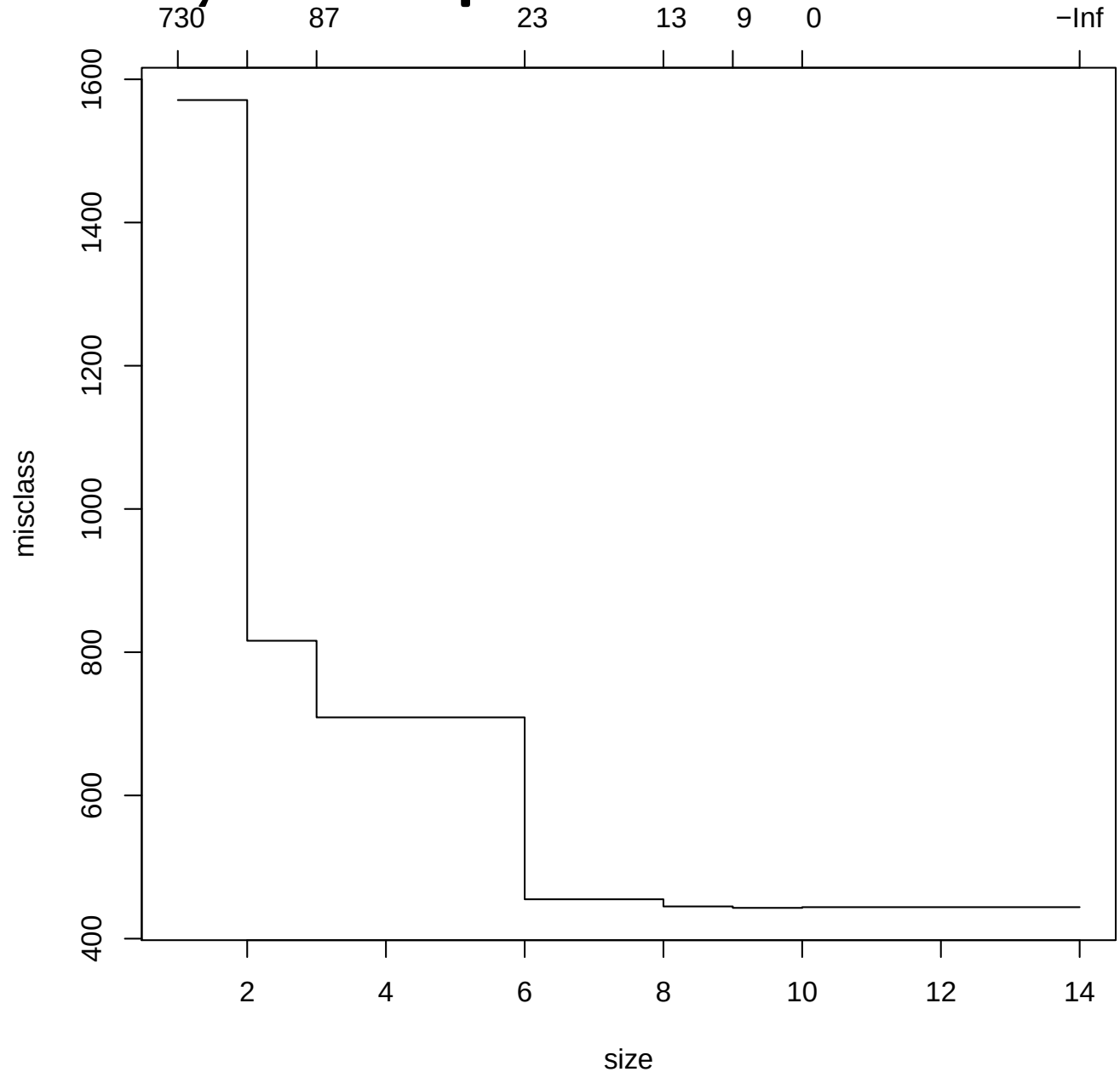
```
library(tree)
# arvore grande:
ajuste=tree(formula,data=dados,subset=treinamento)
plot(ajuste) # arvore grande
text(ajuste, pretty =0) # arvore grande
vcErro = cv.tree(ajuste, FUN = prune.misclass) # podar
                                               # a arvore
                                               # (valid.
                                               # cruzada)
plot(vcErro) # erro da validacao cruzada como
             # funcao do numero de folhas
```

```
bestSize=vcErro$size[which.min(vcErro$dev)] # melhor
                                            # tamanho

# poda para o melhor tamanho:
ajusteVC = prune.misclass(ajuste, best =bestSize)
plot(ajusteVC) # melhor arvore
text(ajusteVC, pretty =0)

# predizer conjunto teste:
tree.pred = predict(ajusteVC,dados[teste,],type ="class")
```

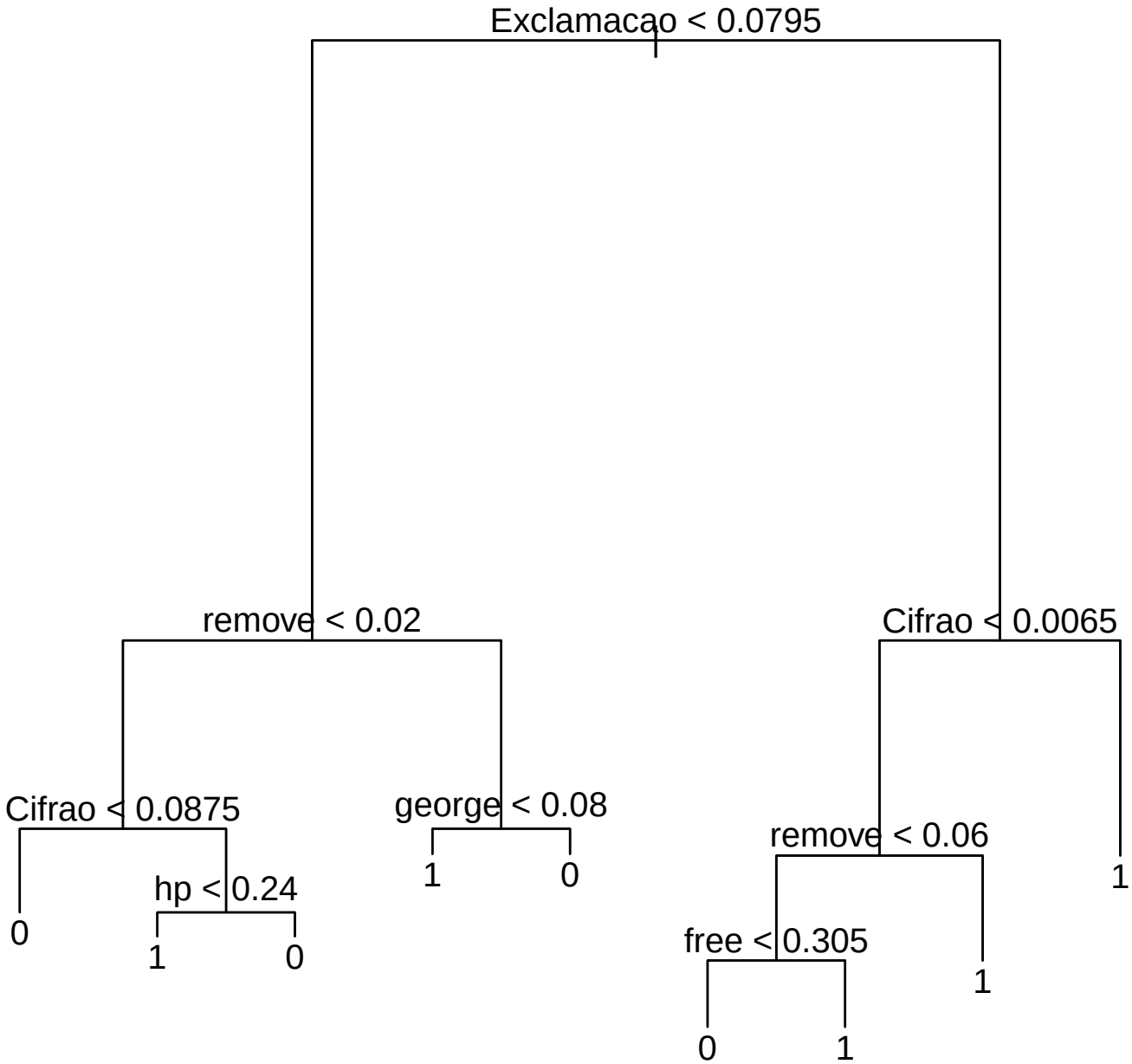
Ex: Detecção de Spams. Árvore "Grande"



Ex: Detecção de Spams. Podando



Ex: Detecção de Spams. Árvore Podada



Ex: Detecção de Spams.

Porcentagem de erros: 10% (regressão logística: 8%)

Árvore de Regressão

Podemos também usar árvores em problemas de regressão.

Neste caso, são chamadas de Árvore de Regressão

Contruimos uma árvore de regressão da mesma forma que uma árvore de classificação, com a seguintes diferenças:

- O valor predito por uma árvore de classificação é dado por

$$g(x) = \sum_{k \in R^x} y_k,$$

  onde R^x é a região na qual x cai.
- Ao invés de usar o coeficiente Gini para medir o quão puro uma partição é, usamos o EQM:

$$P(T) = \sum_{R} \sum_{k \in R} (y_k - \widehat{y}_R)^2$$

Para ajustar uma árvore de regressão no R, usamos os mesmos comandos que para ajustar uma árvore de classificação, com exceção que para podar a árvore, fazemos:

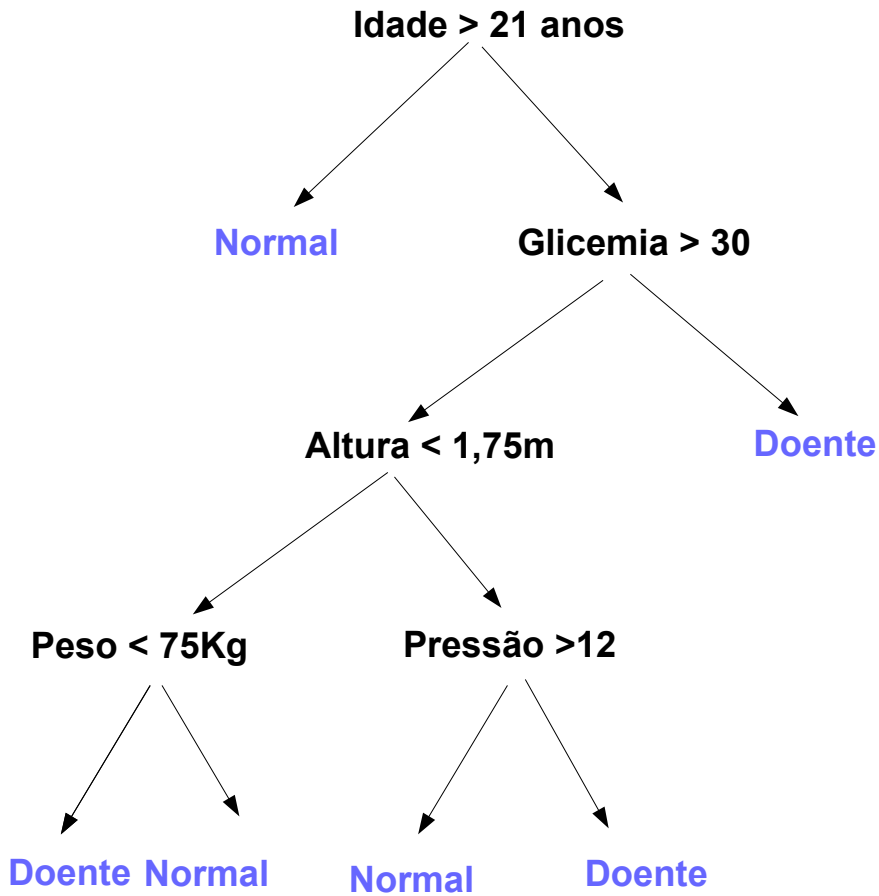
```
vcErro = cv.tree(ajuste)
```

e, para predizer novas observações,

```
tree.pred = predict(ajusteVC,dados[teste,])
```

Resumo

Árvores de predição consistem em um método simples, intuitivo e de fácil entendimento e intepretação. Em particular, elas não necessitam de equações para serem entendidas (mais fácil para nao estatísticos intepretarem e aplicarem).



Para criar tal árvore, usamos duas etapas:

1. Criamos uma árvore "grande"
2. Podamos esta árvore

Criamos uma árvore grande procurando, a cada passo, a melhor variável para predizer Y, dada a árvore obtida no passo anterior.

A poda é feita para evitar o overfitting.

Apesar de serem fácil de interpretar, frequentemente árvores podem possuir poder preditivo não tão bom quanto outros métodos.

Para isso, vamos ver como combinar árvores na próxima aula.