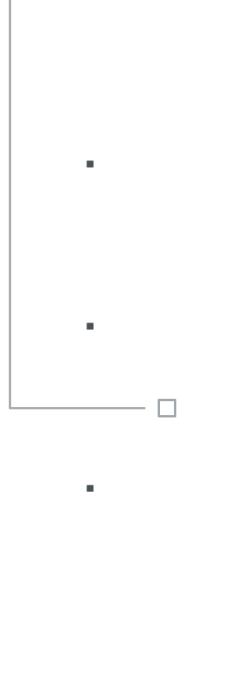


FIAP

MBA



# Métodos CART

Dheny R. Fernandes

## 1. Regression Trees

1. Intuição
2. Exemplo
3. Como construir uma árvore
  1. RSS
  2. MSE
  3. Overfitting
  4. Múltiplas características
  5. Implementação
4. Pruning
  1. Implementação

## 2. Decision Trees

1. Como construir uma árvore
2. Gini
3. Exemplo

# Regression Trees

*Classification and Regression Trees*, ou CART, é um termo usado para se referir aos algoritmos de Árvore de Decisão que podem ser usados para modelar problemas tanto de classificação quanto de regressão.

O Algoritmo CART é o fundamento para importantes algoritmos como *Bagged Decision Trees*, *Boosted Decision Trees* e *Random Forest*.

Na aula de hoje, focaremos em Regression Trees

Uma Árvore de Decisão usa uma estrutura de árvore para representar um número de possíveis caminhos de decisão e um resultado para cada caminho.

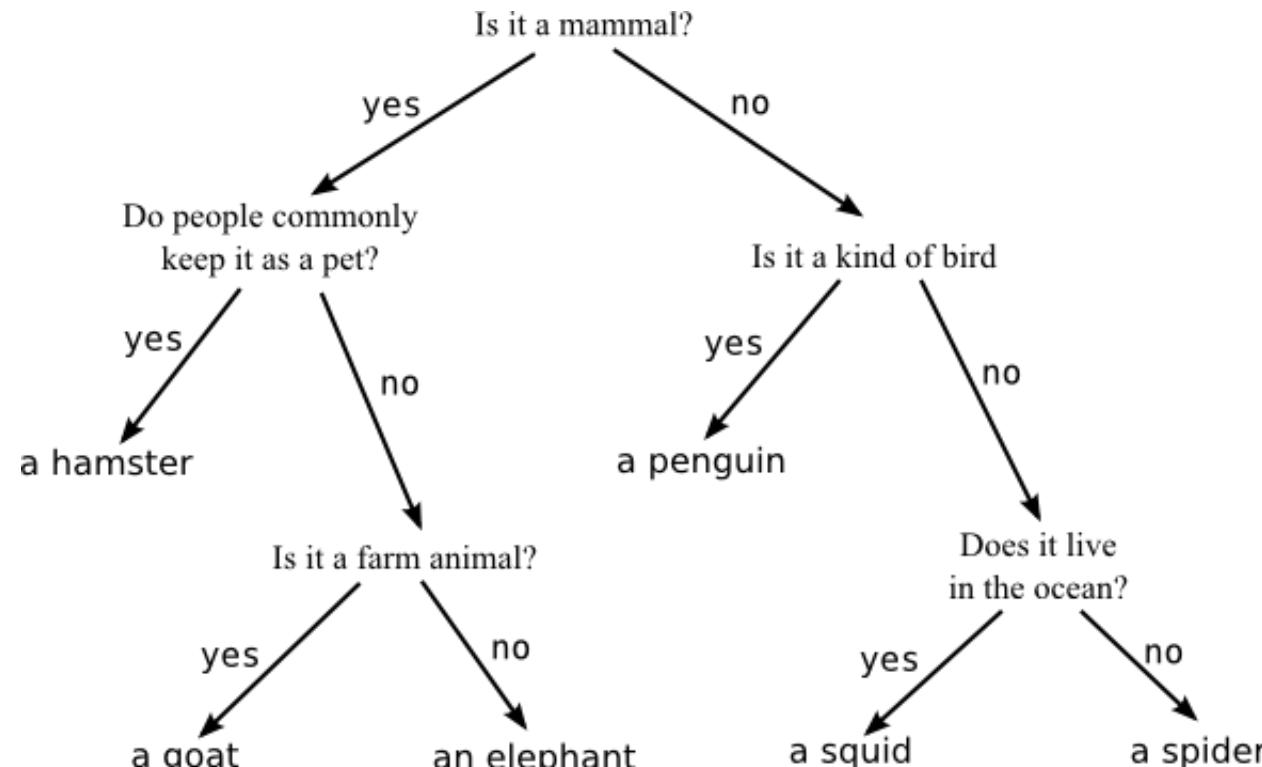
É similar a um jogo de advinha baseado em perguntas:

- “Estou pensando em um animal.”
- “É um mamífero?”
- “Sim.”
- “As pessoas costumam tê-lo como um animal de estimação?”
- “Não.”
- “Ele vive numa fazenda?”
- “Não.”
- “É um elefante”
- “Sim”

As escolhas correspondem ao seguinte caminho:

- “É mamífero” -> “Não o tem como animal de estimação” -> “Não vive numa fazenda” -> “Elefante”

E podemos representar o jogo (de maneira simplória) usando uma Árvore de Decisão:



## Prós:

1. Fácil de entender e interpretar
2. A maneira em que se chega a uma predição é transparente
3. Podem lidar tanto com atributos numéricos quanto categóricos

## Contras:

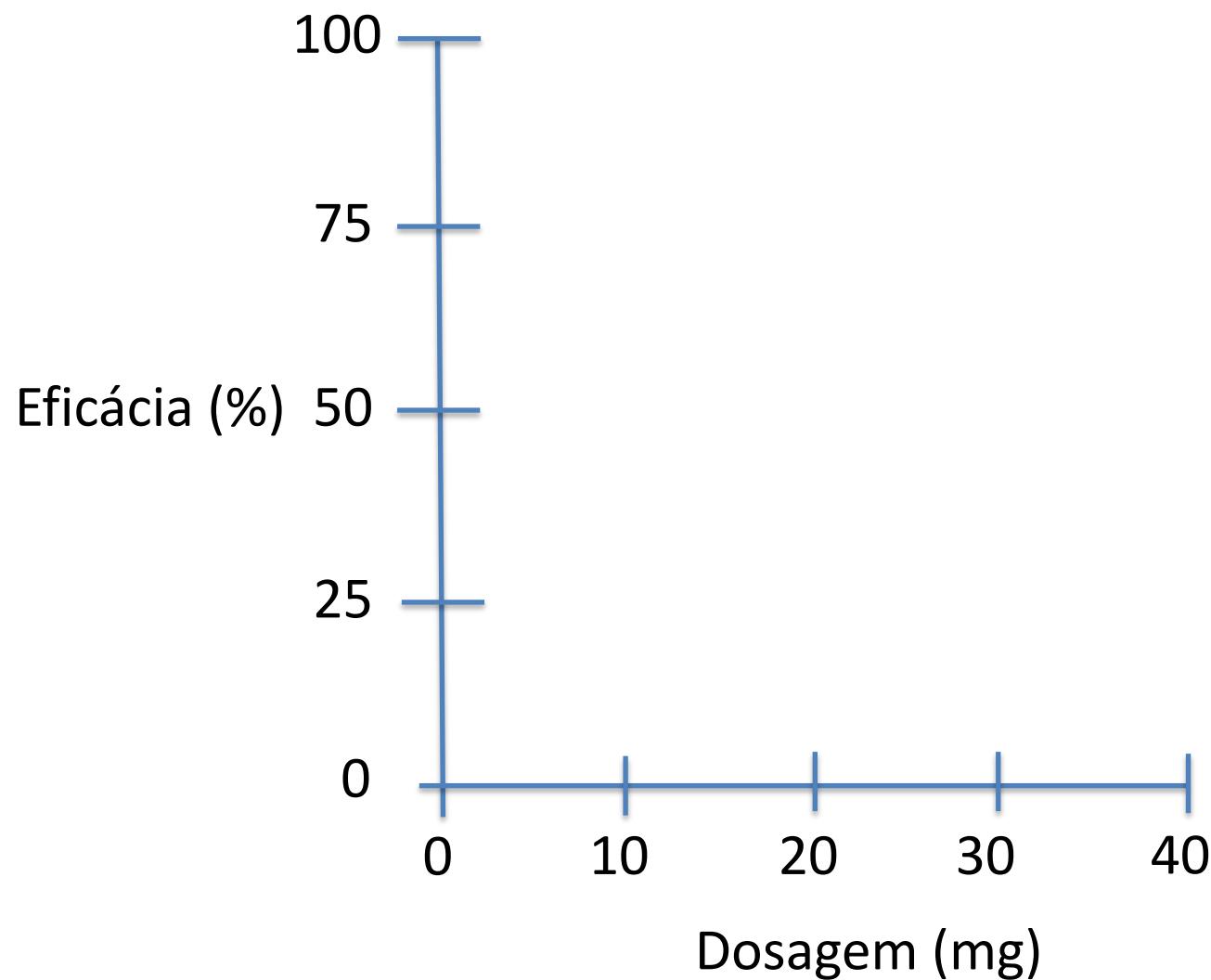
1. Alto custo computacional para encontrar a Árvore ótima
2. Overfitting

Vamos entender a construção de uma *Regression Tree* a partir de um exemplo:

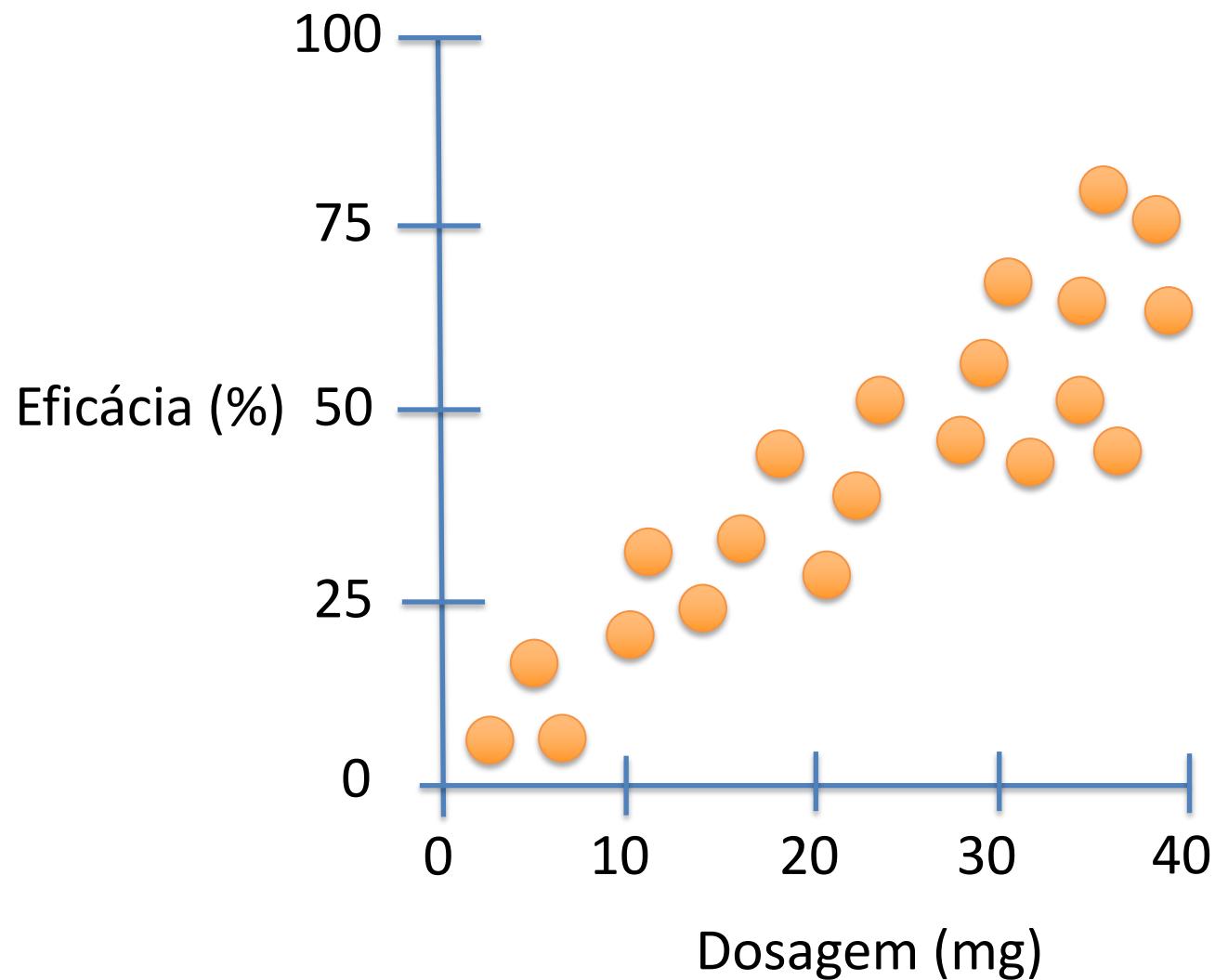
Estamos desenvolvendo um novo medicamento para combater uma gripe comum. Entretanto, ainda **não sabemos a dosagem ótima**.

Uma maneira de determinar seu valor é dar diferentes dosagens e mensurar o quanto eficaz cada uma delas foi.

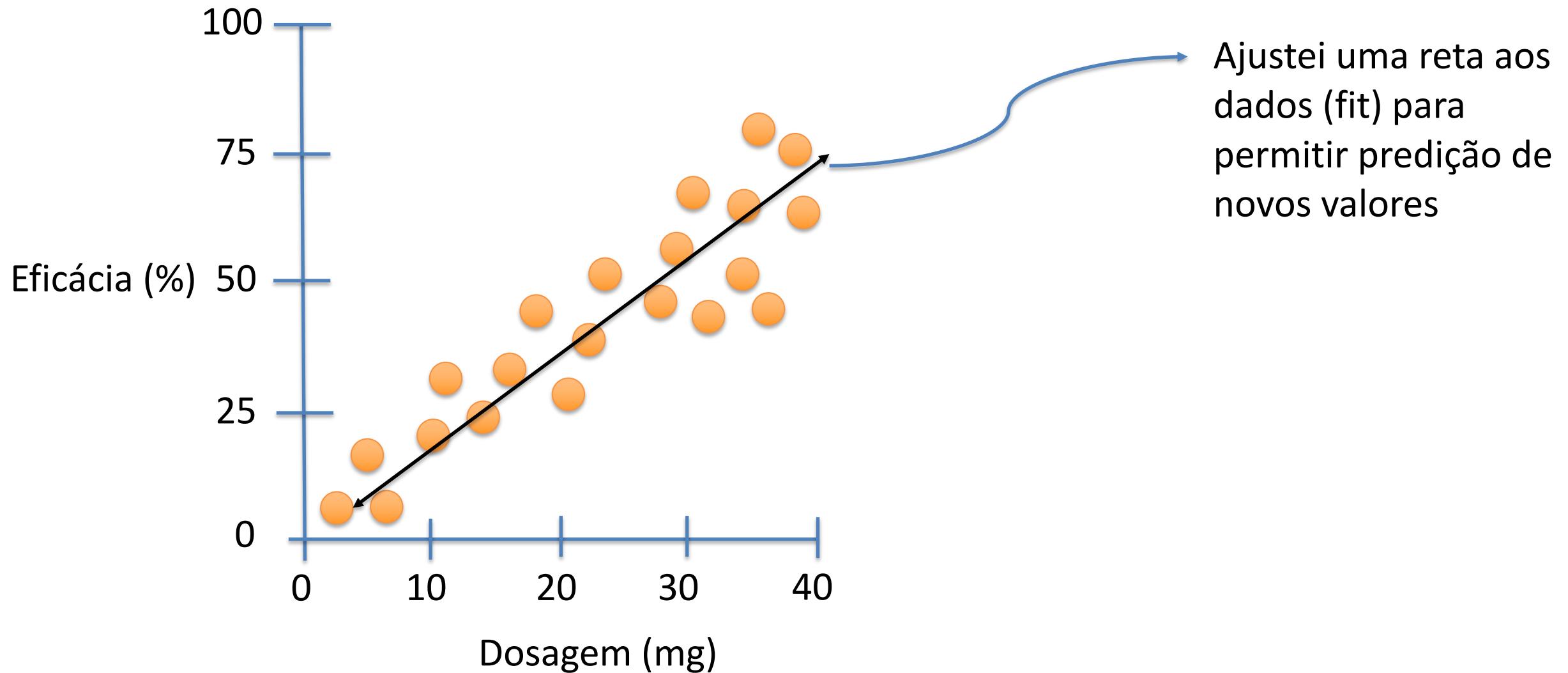
# Regression Trees - Exemplo



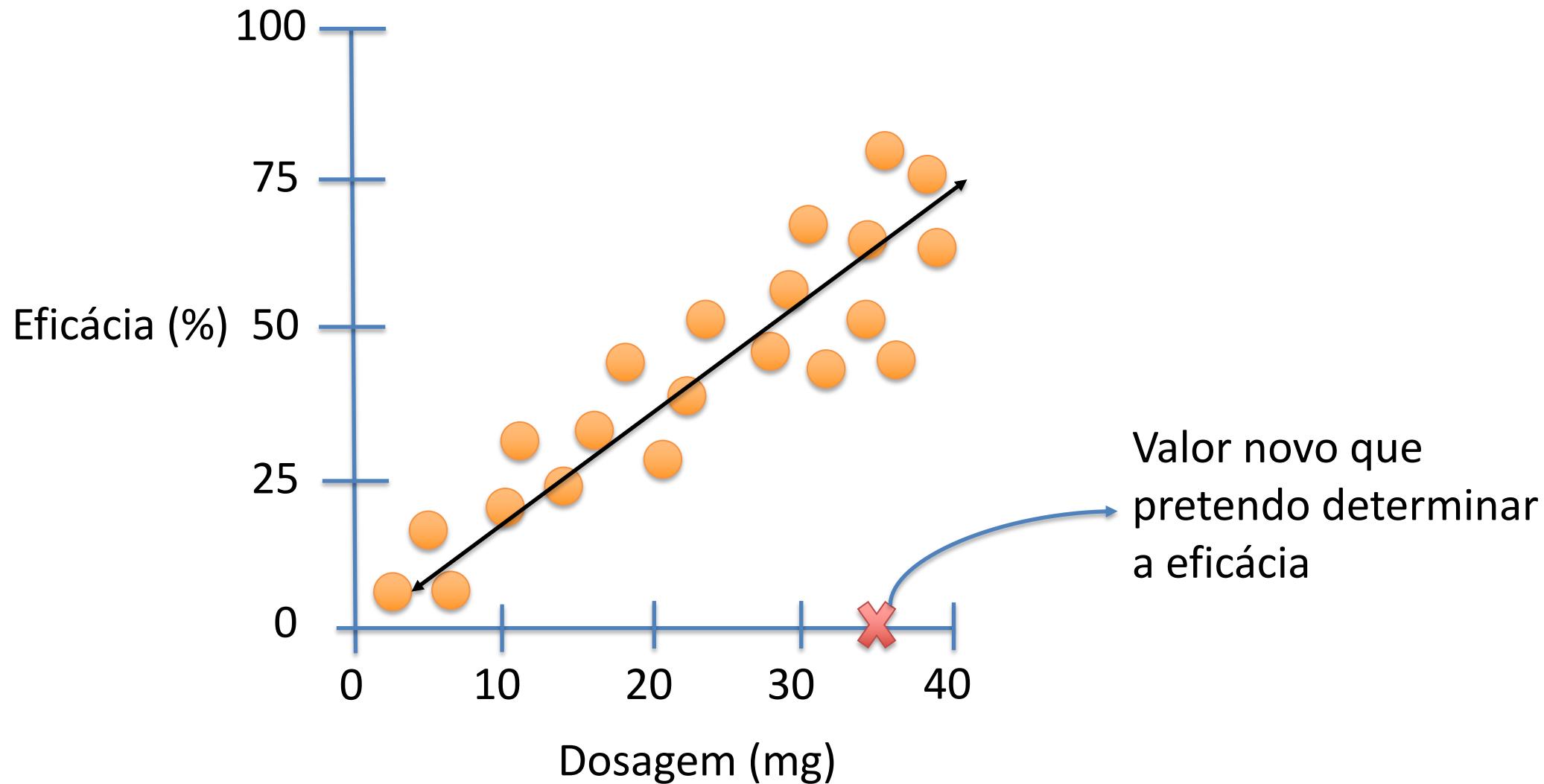
# Regression Trees - Exemplo



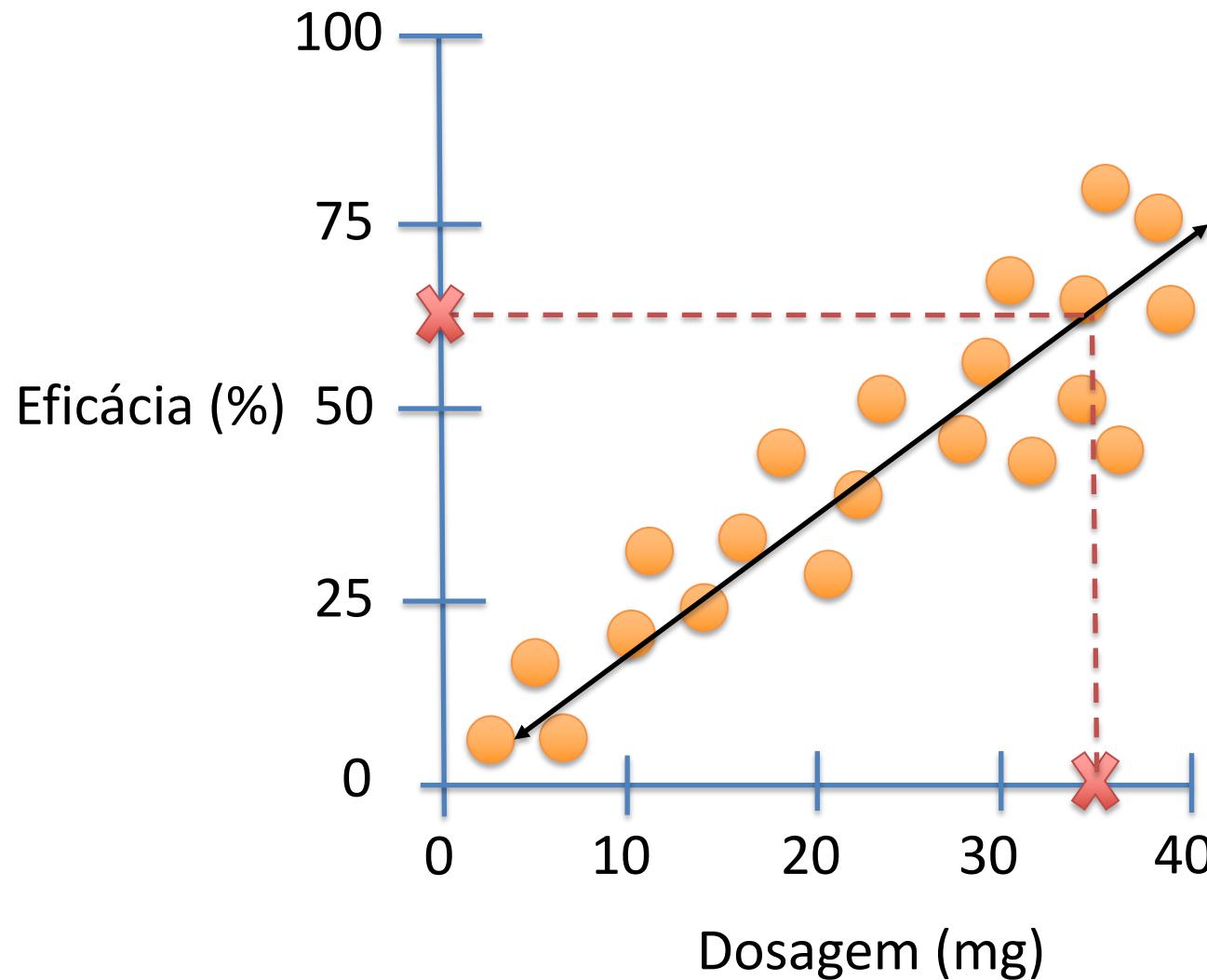
# Regression Trees - Exemplo



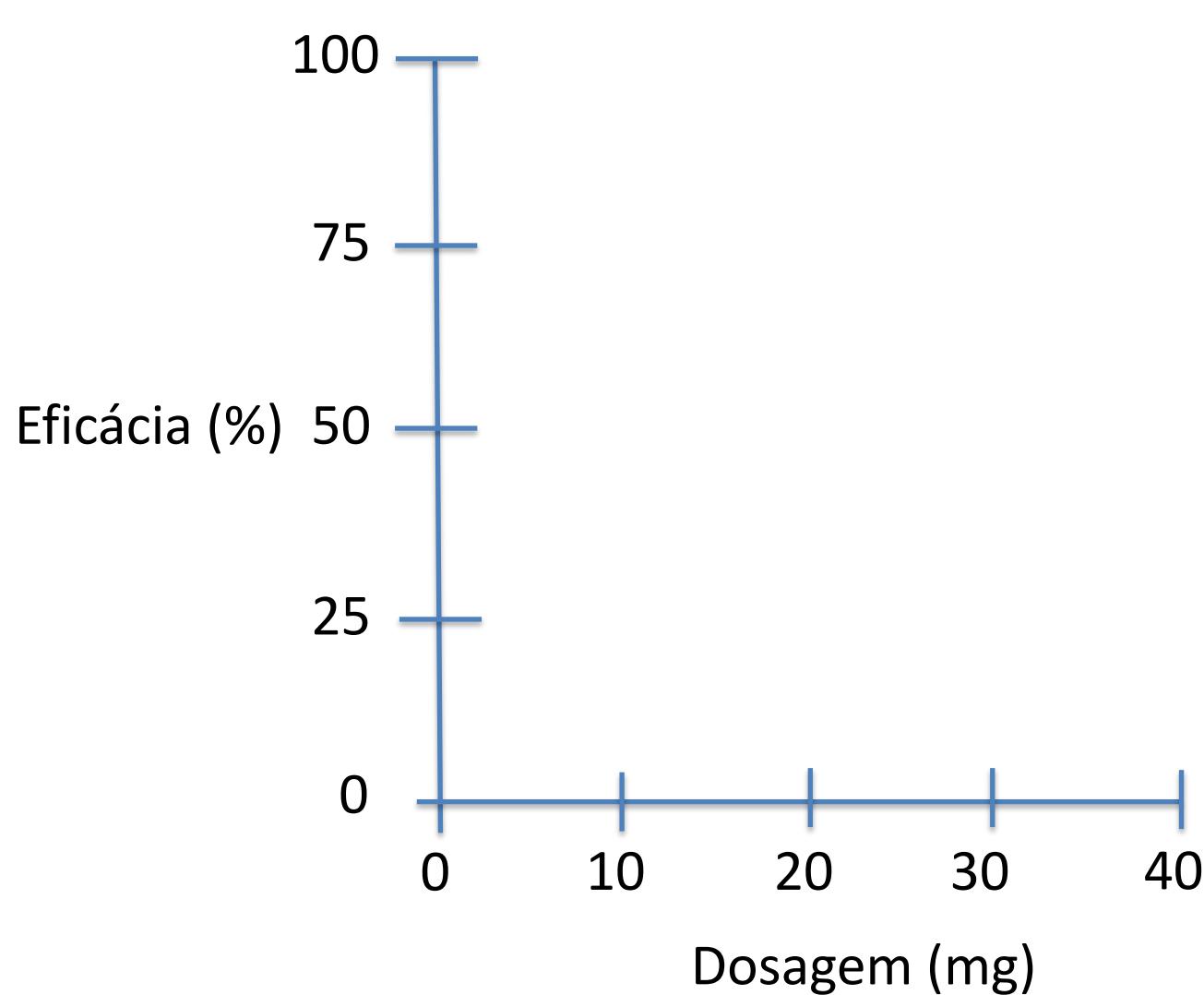
# Regression Trees - Exemplo



# Regression Trees - Exemplo

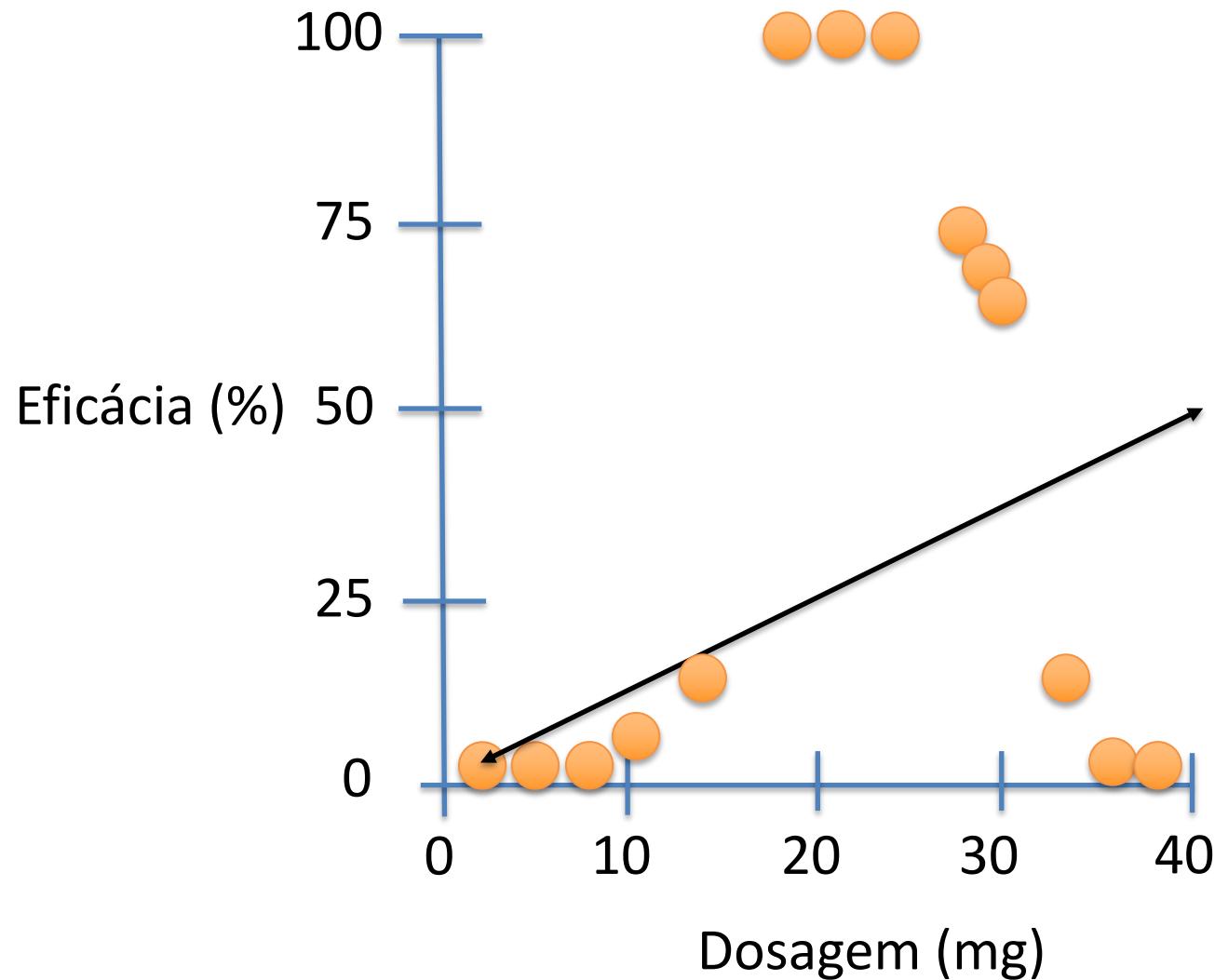


Uma dosagem de 35mg produz um resultado de 68% de eficácia



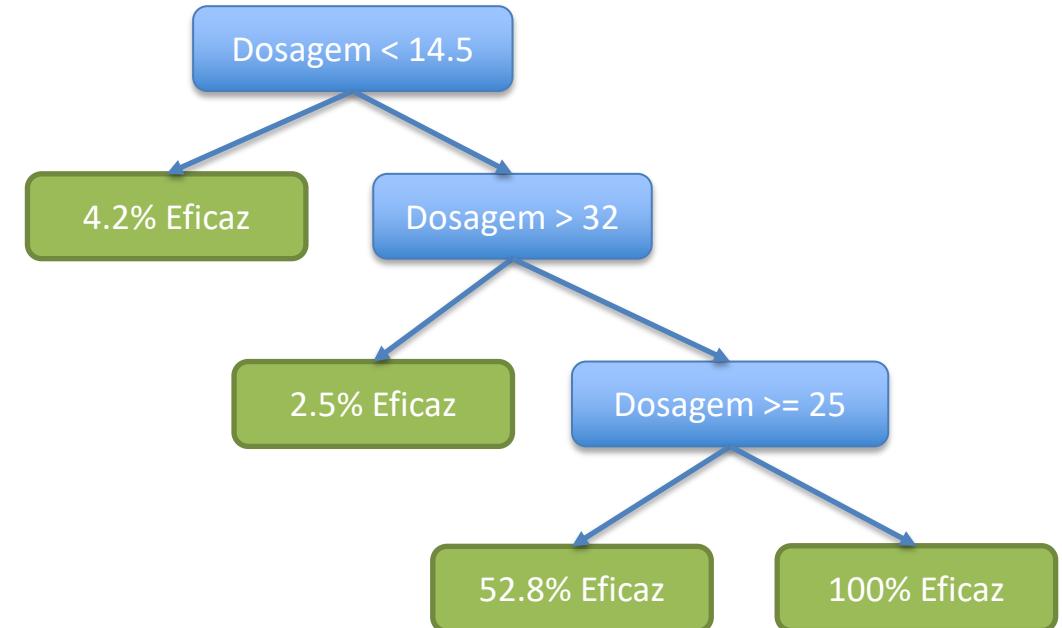
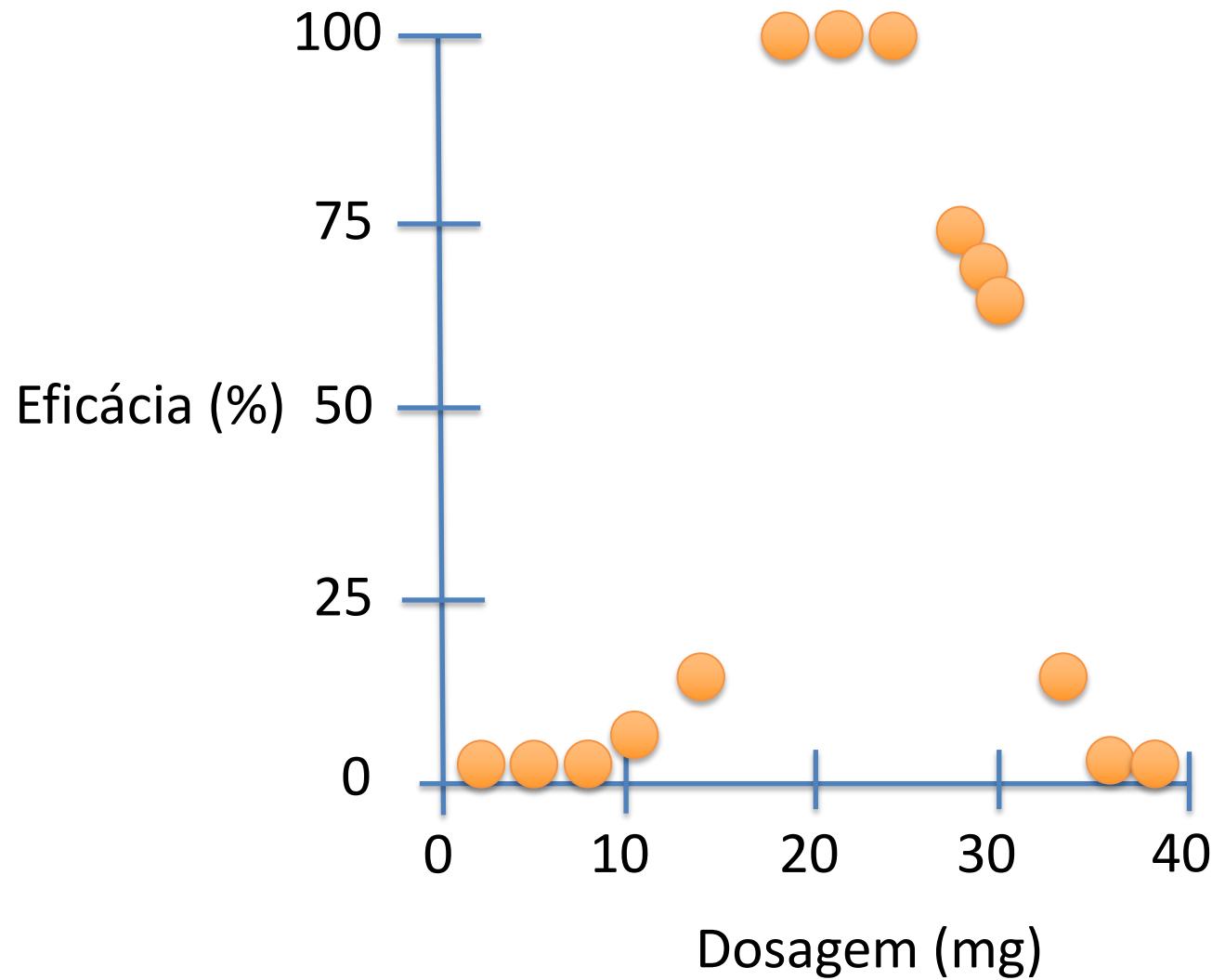
O que acontece quando  
meus dados possuem esse  
comportamento?

## Regression Trees - Exemplo

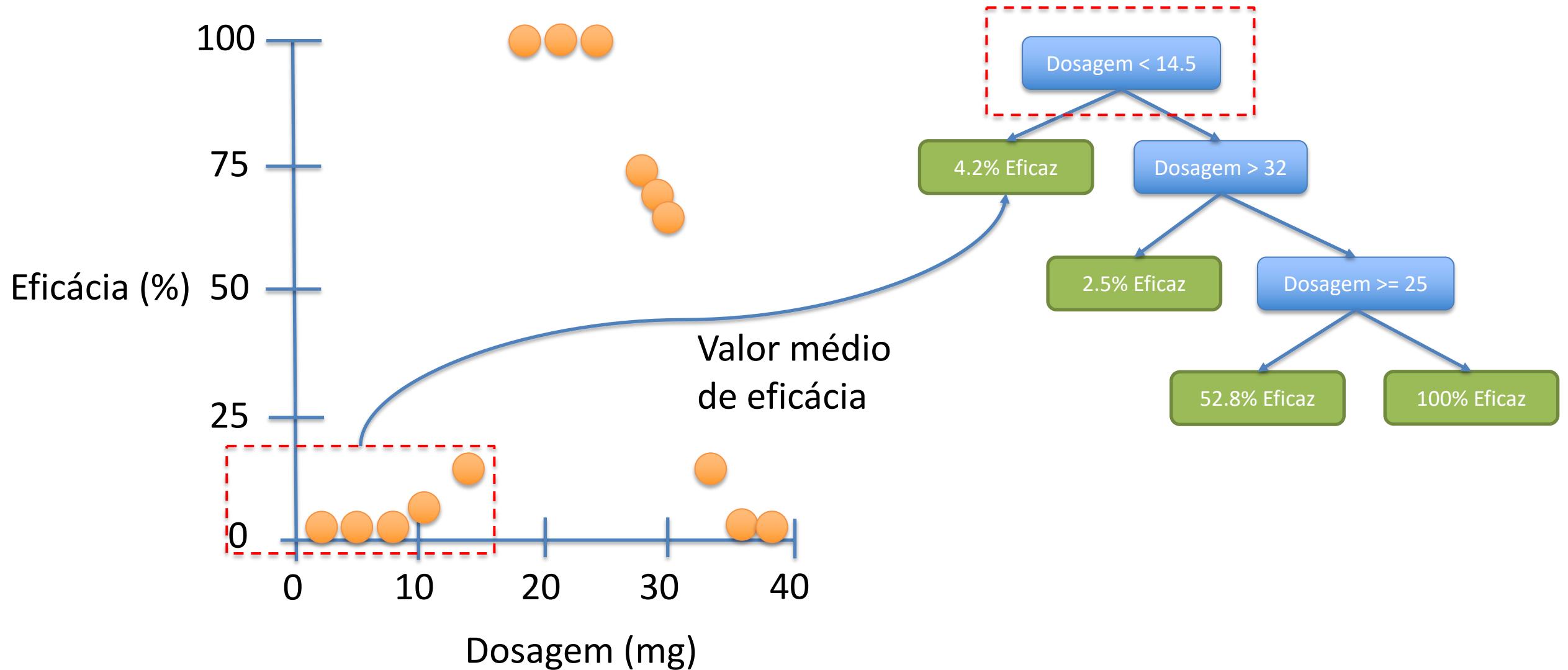


Claramente, uma simples regressão linear não resolve o problema

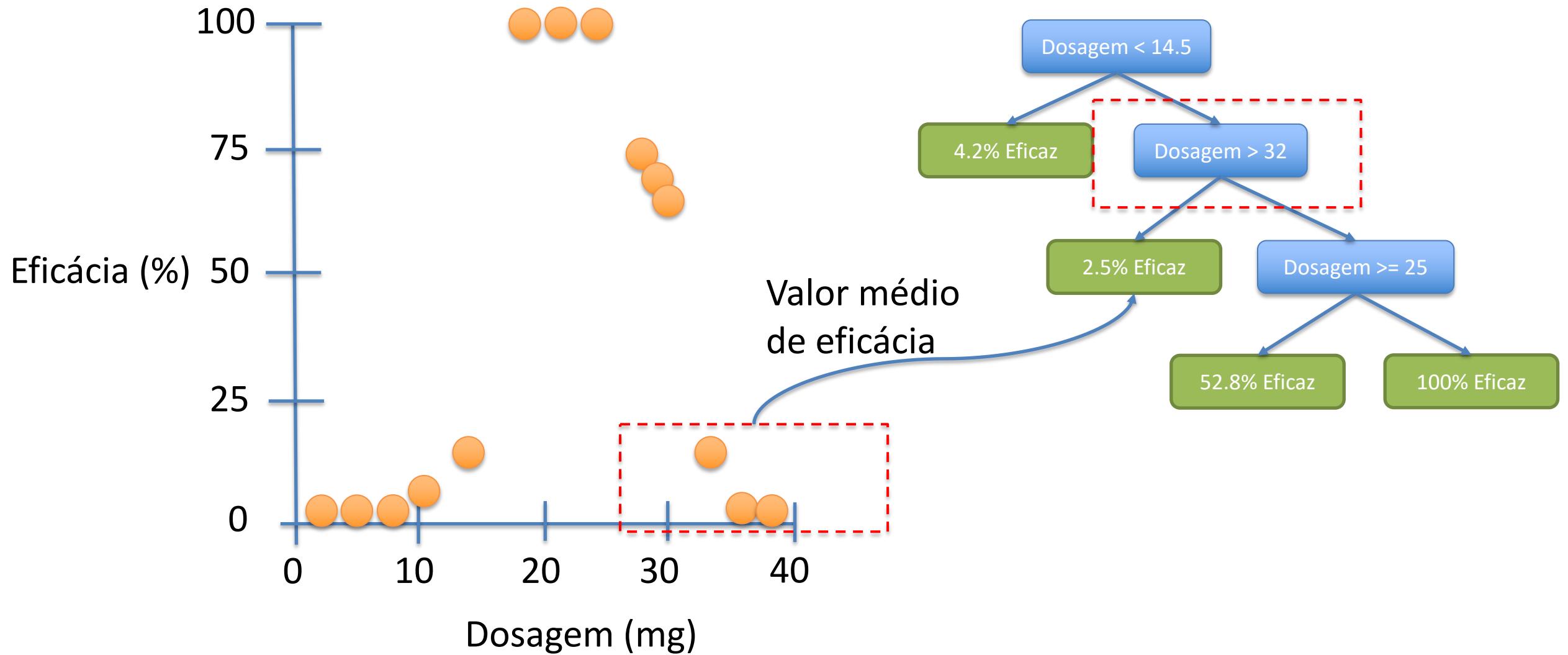
# Regression Trees - Exemplo



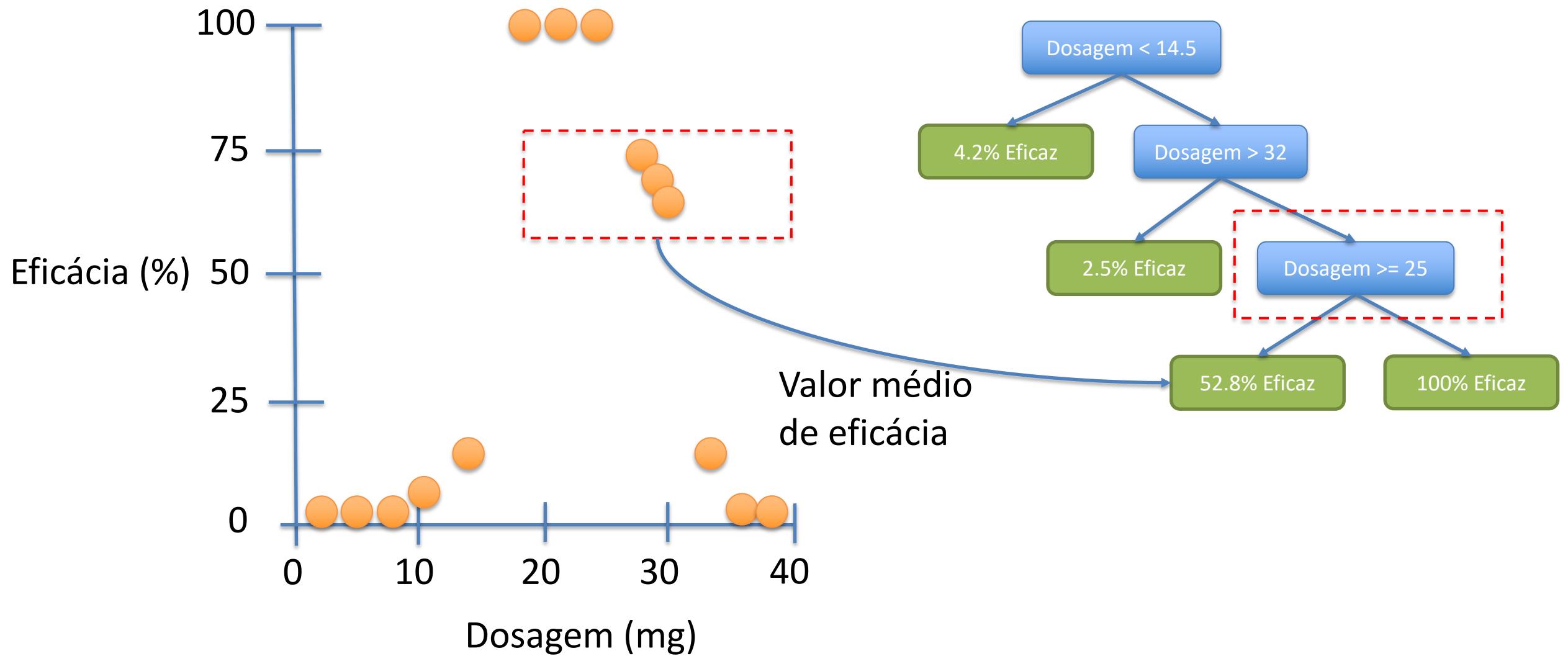
# Regression Trees - Exemplo



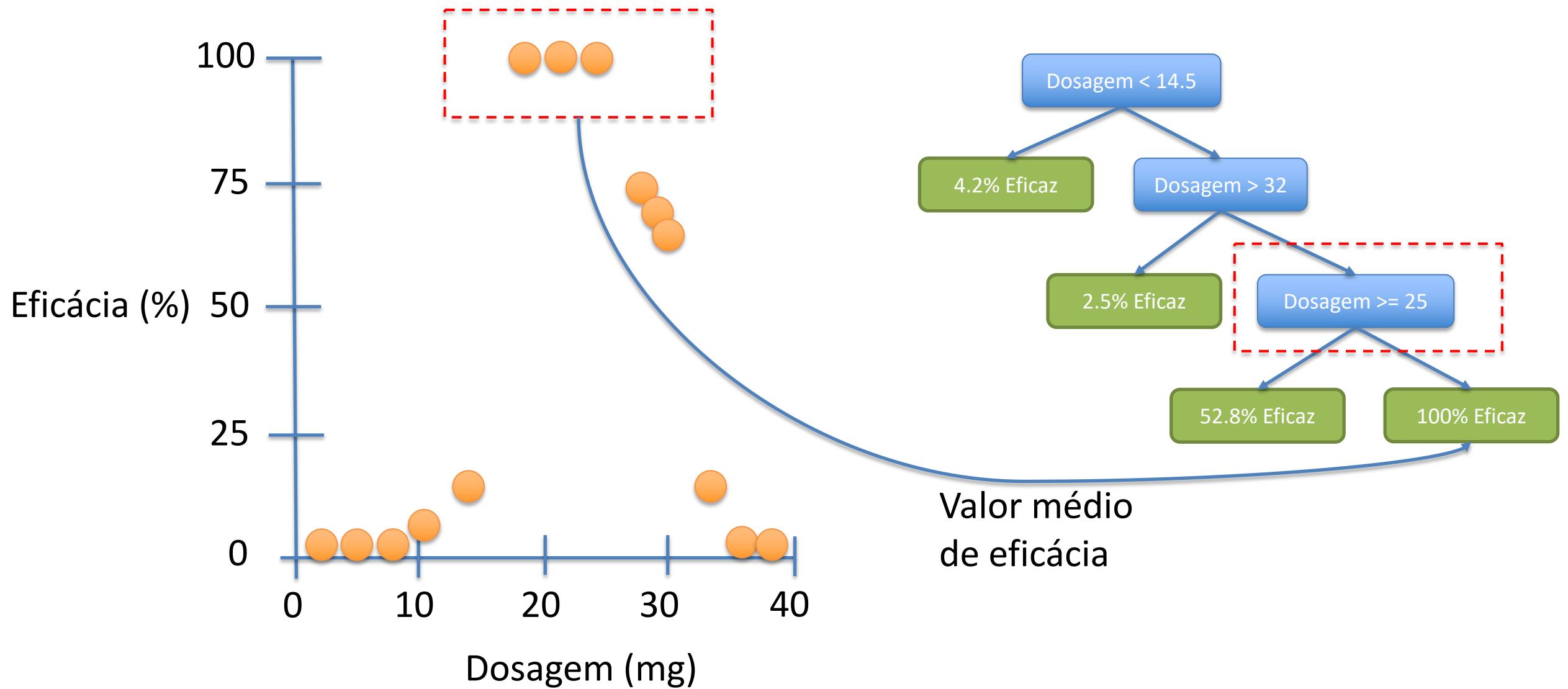
# Regression Trees - Exemplo



# Regression Trees - Exemplo



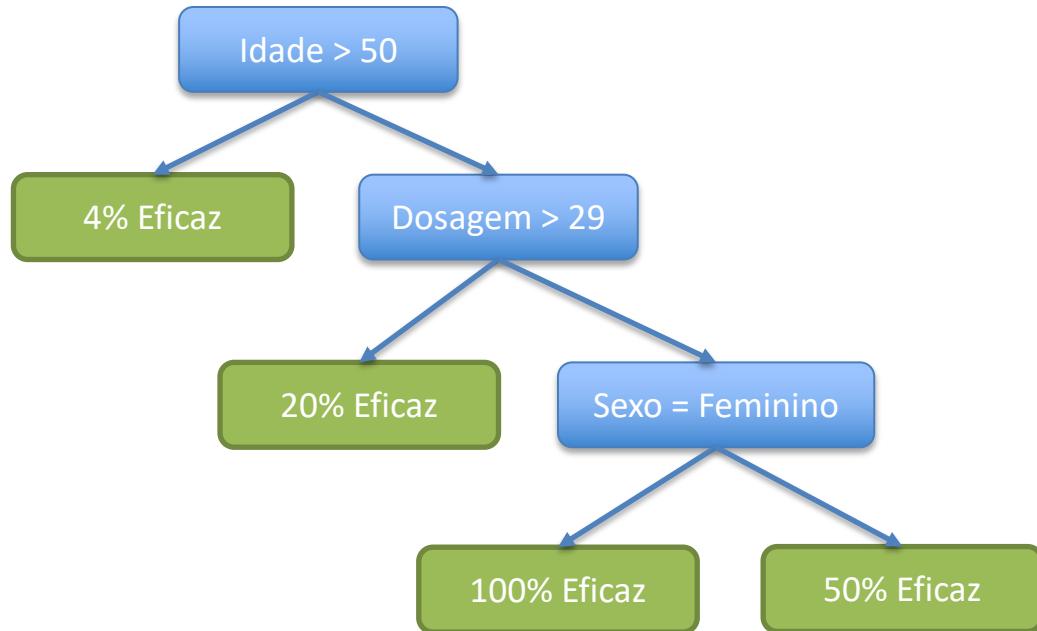
# Regression Trees - Exemplo



Entretanto, o ganho de usar *Regression Trees* não está em tentar prever uma variável a partir de apenas uma *feature*. É quando usamos mais de uma *feature* que seu valor se torna real.

Dosagem	Idade	Sexo	Eficácia
10	25	Feminino	98
20	73	Masculino	0
35	54	Feminino	100
5	12	Masculino	44
...	...	...	...

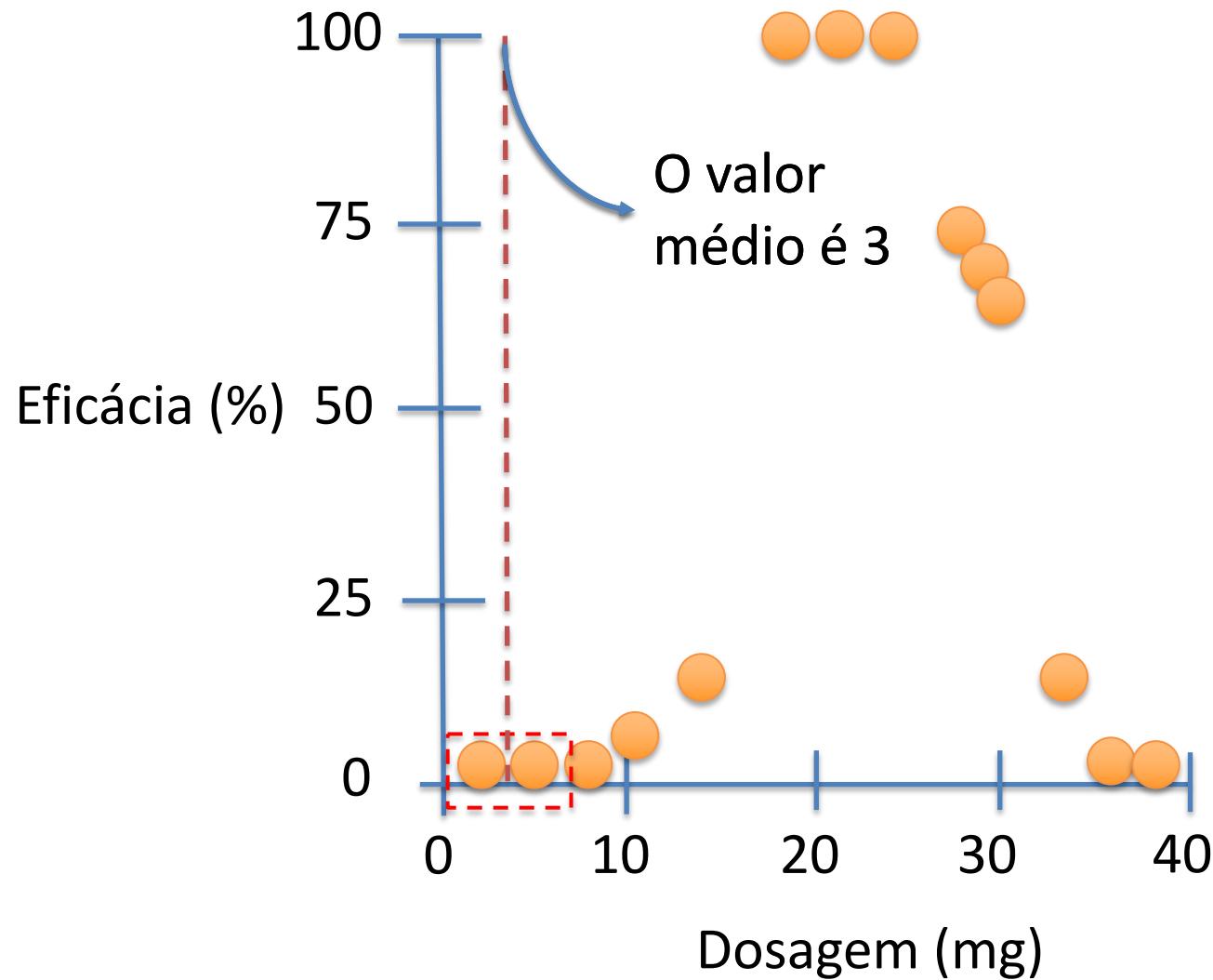
E as Regression Trees lidam muito bem com regressão multivariada.



Dosage m	Idade	Sexo	Eficácia
10	25	Feminin o	98
20	73	Masculi no	0
35	54	Feminin o	100
5	12	Masculi no	44

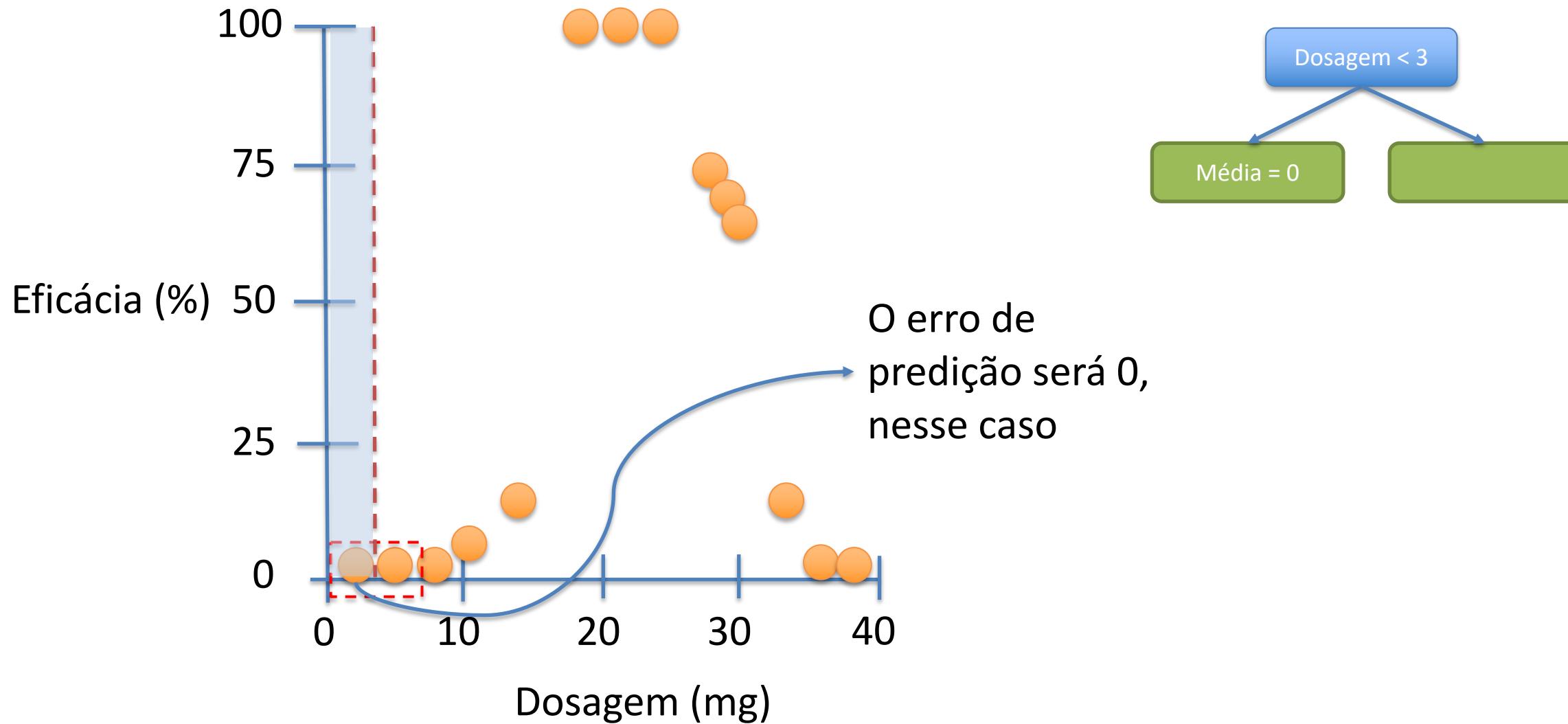
Ok, entendemos a ideia geral por trás das Regression Trees, mas como elas são construídas?

# Regression Trees – Como construir uma Regression Tree?

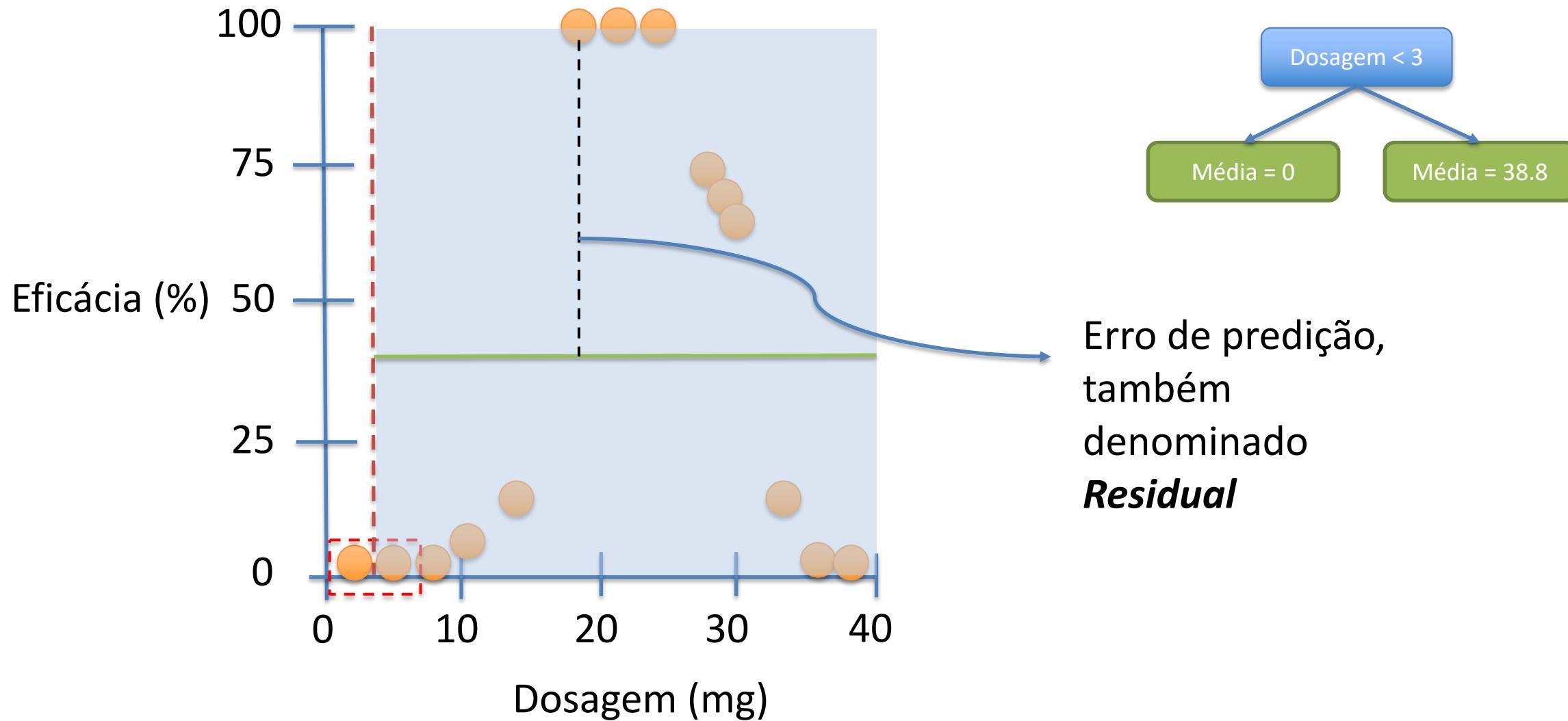


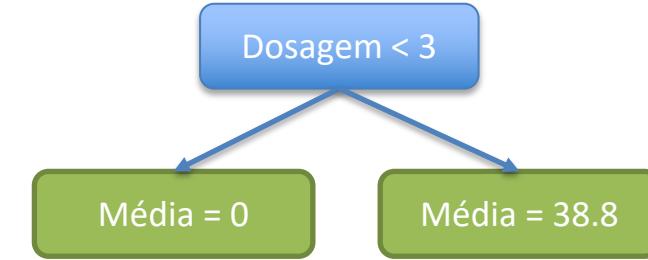
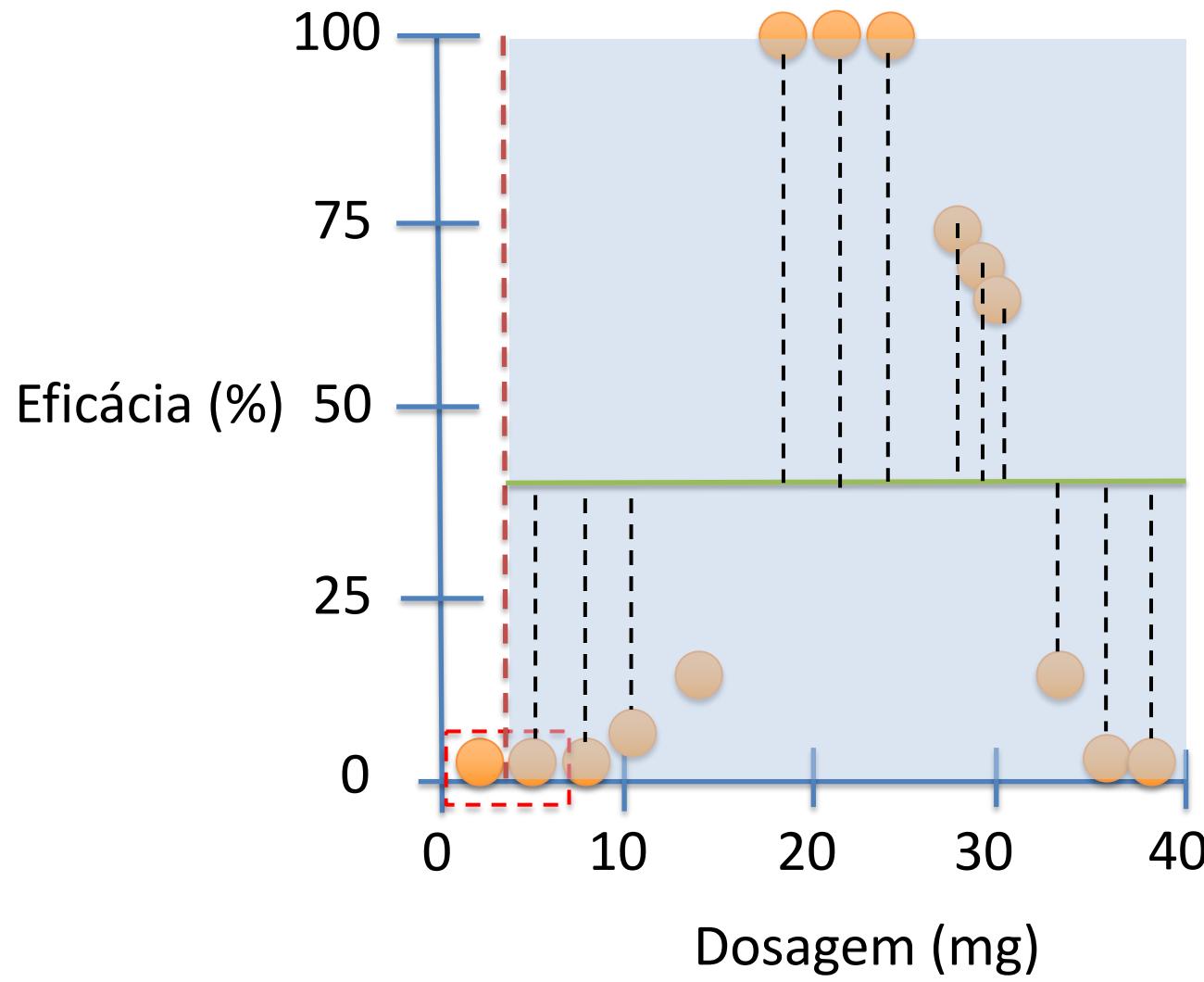
Voltando aos dados,  
vamos focar nas duas  
primeiras observações

# Regression Trees – Como construir uma Regression Tree?



# Regression Trees – Como construir uma Regression Tree?

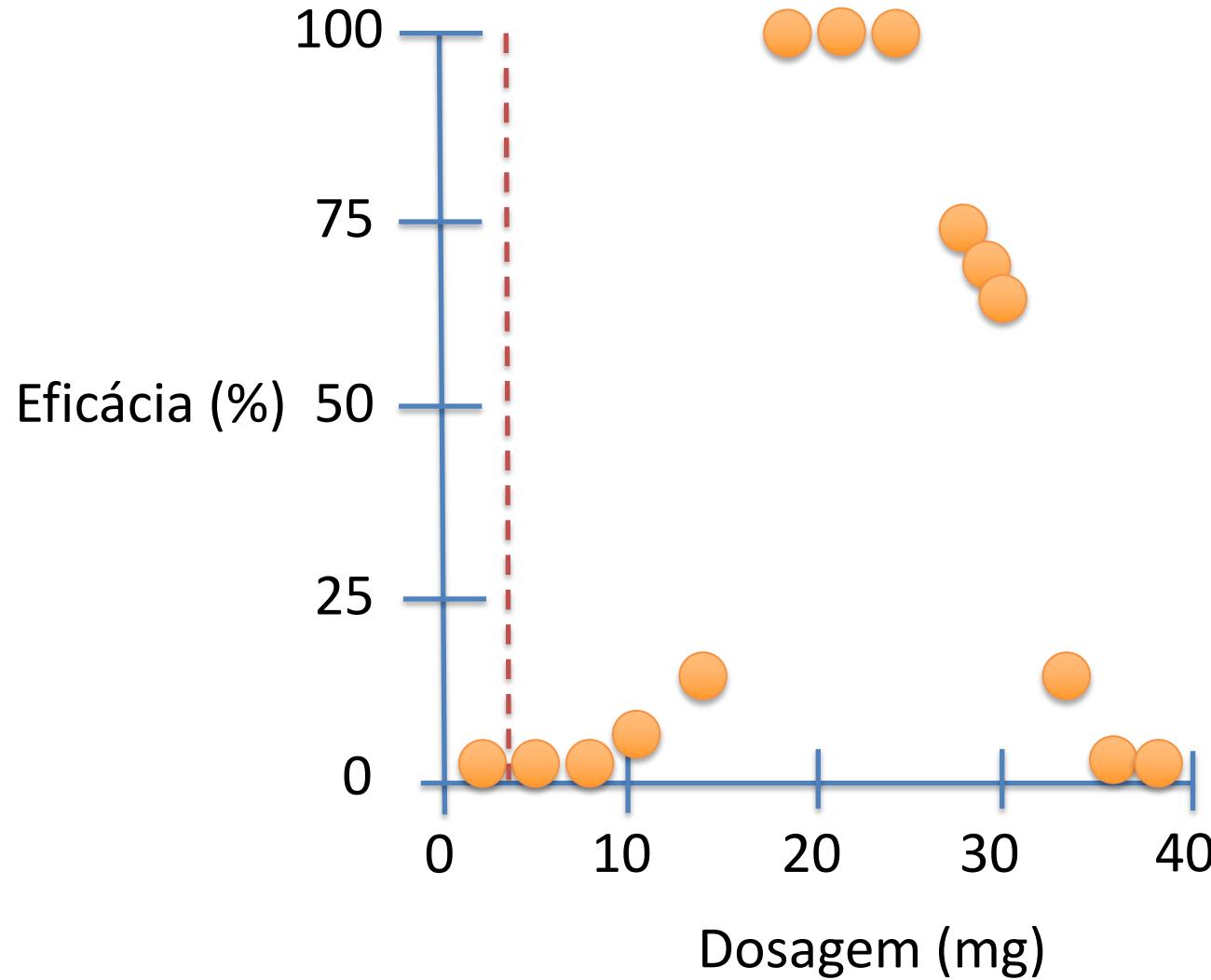




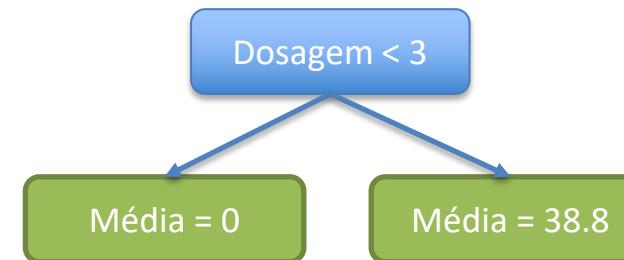
Para cada observação, podemos desenhar os Residuals, que nada mais é que a diferença entre o valor predito e observado

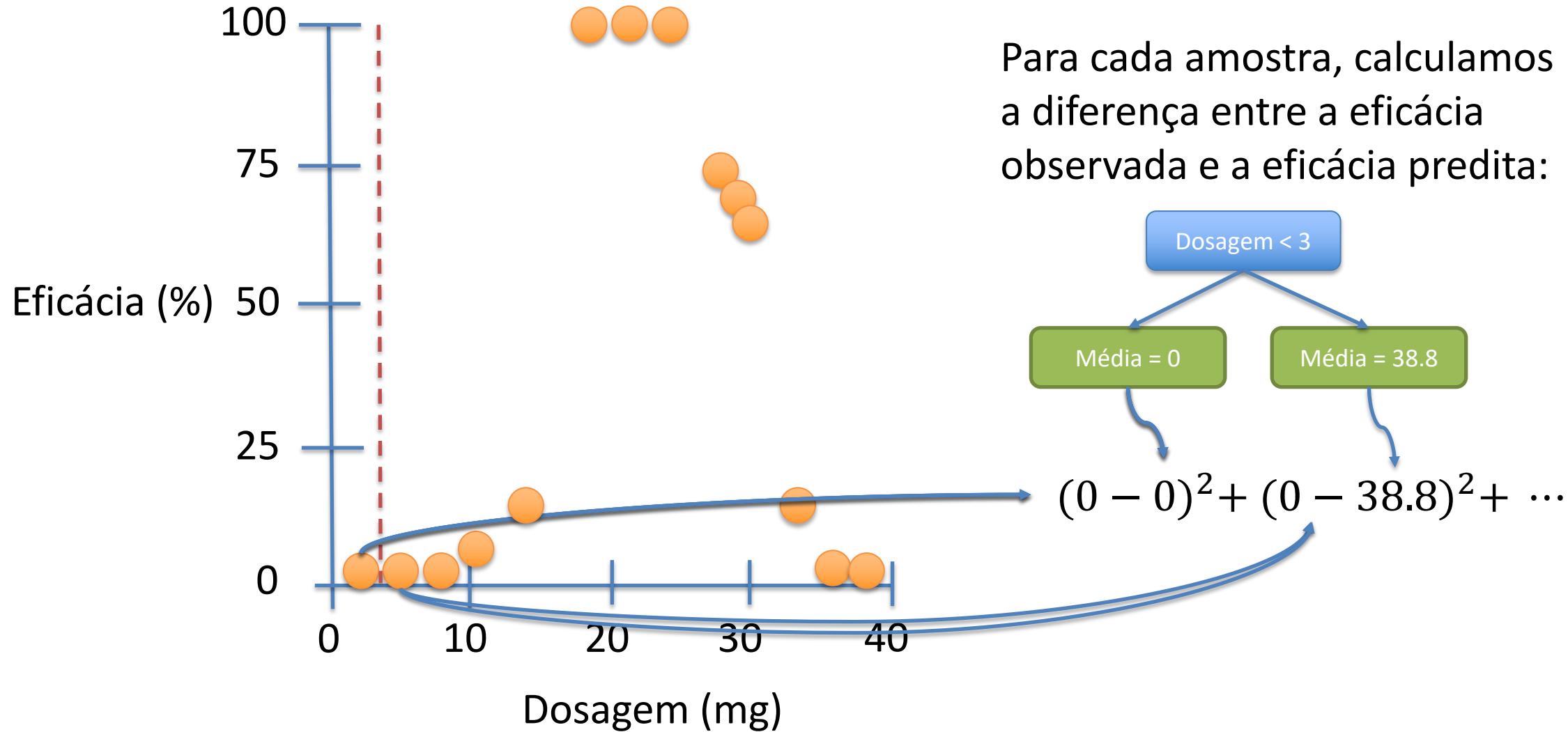
E podemos usar os Residuals para quantificar a qualidade das previsões.

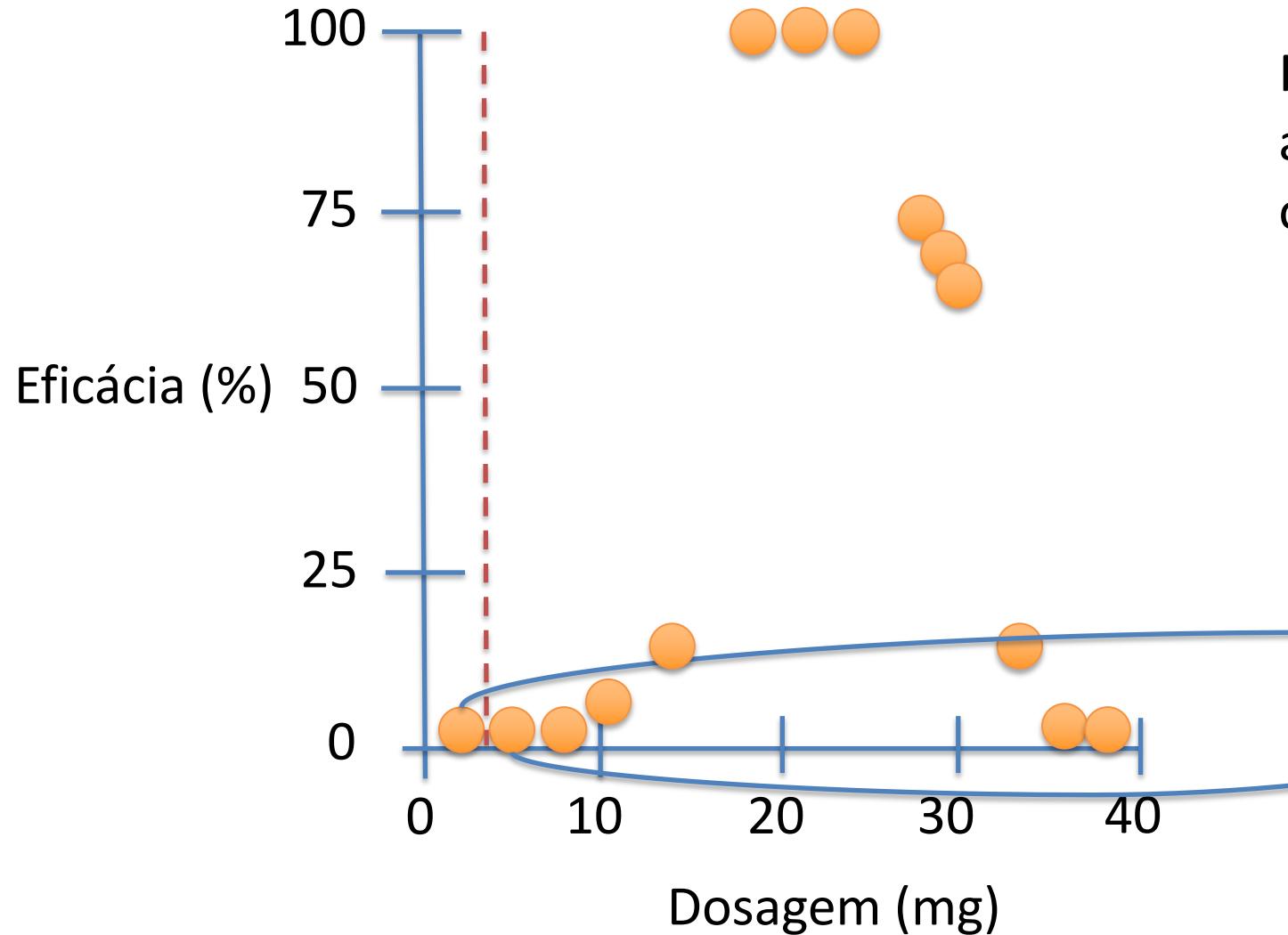
# Regression Trees – Como construir uma Regression Tree?



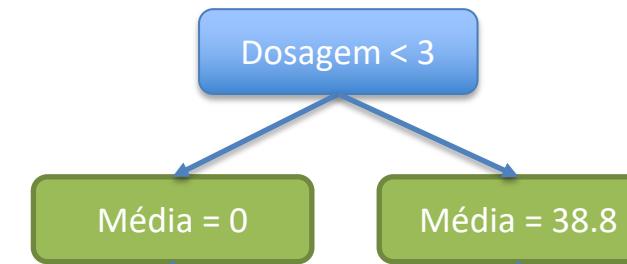
Para cada amostra, calculamos a diferença entre a eficácia observada e a eficácia predita:







Para cada amostra, calculamos a diferença entre a eficácia observada e a eficácia predita:

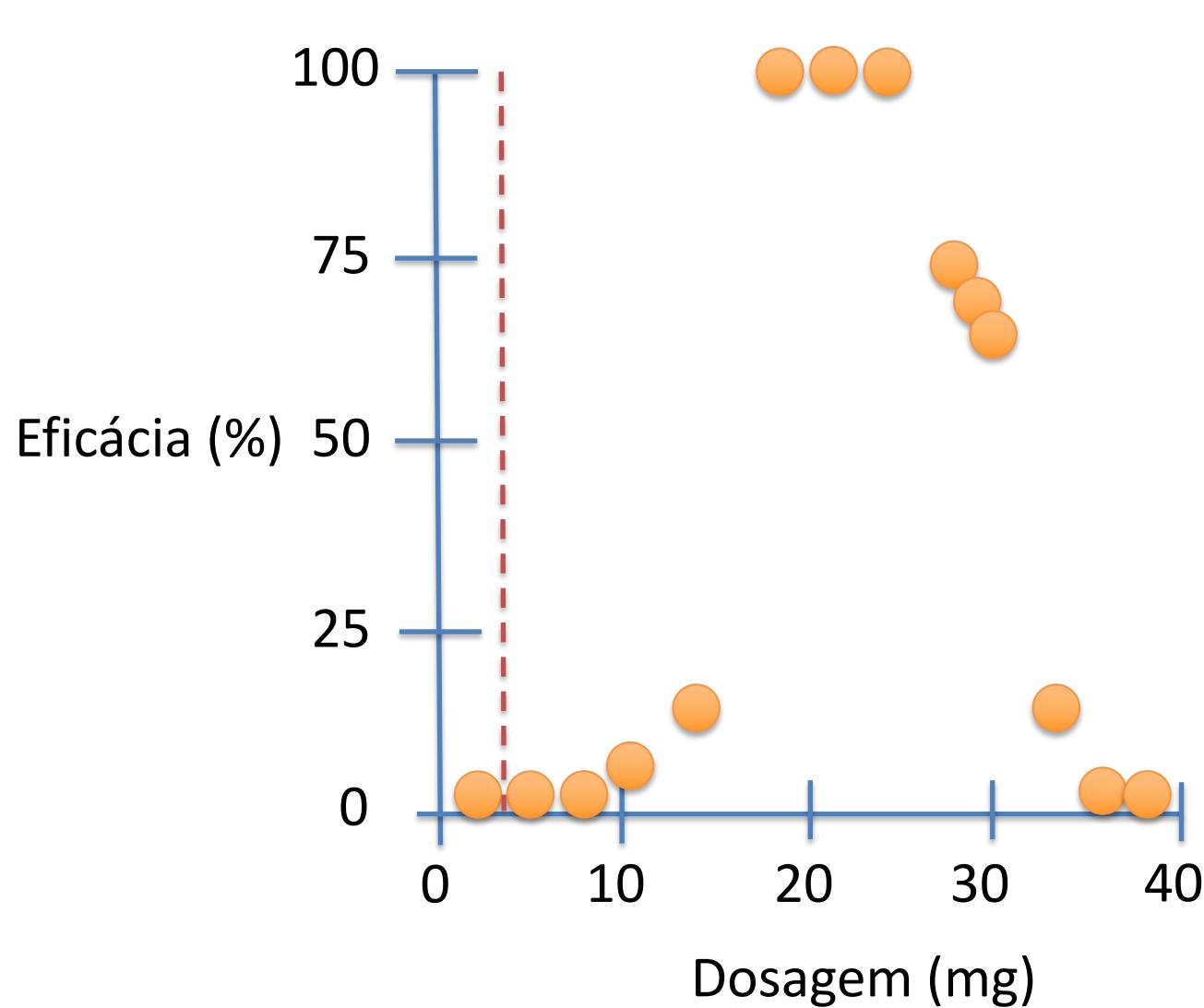


$$(0 - 0)^2 + (0 - 38.8)^2 + \dots = 27468.5$$

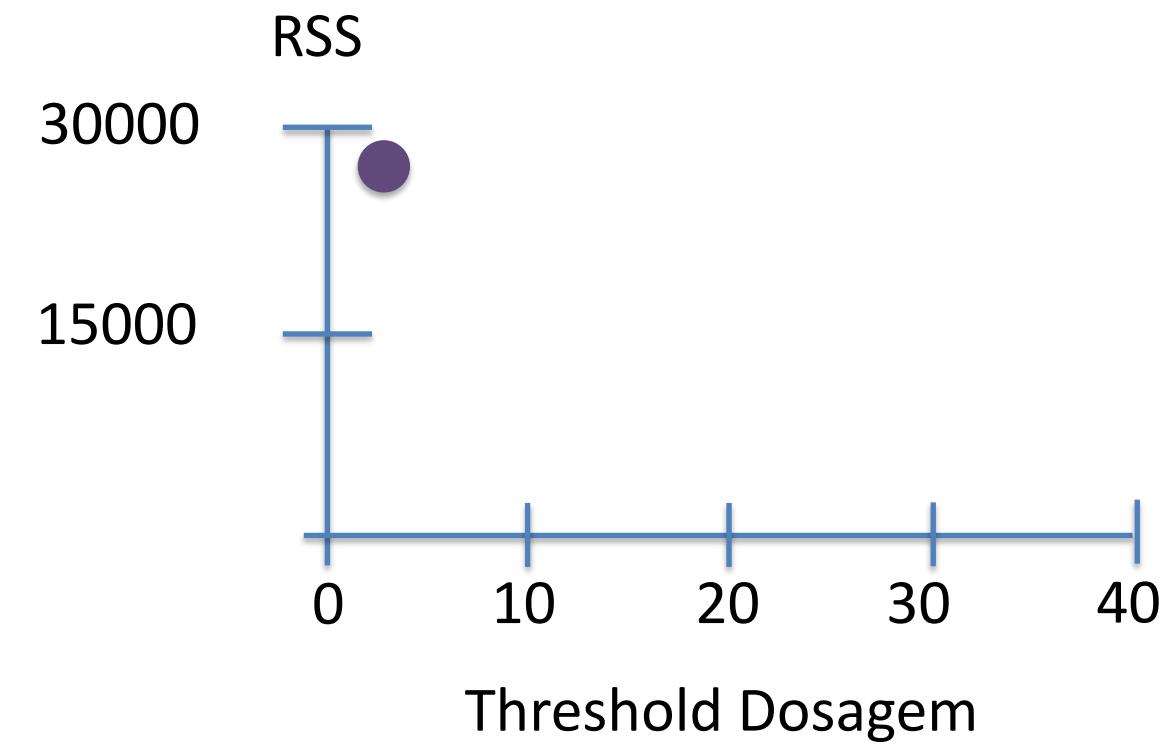
RSS = Residual Sum of Squares

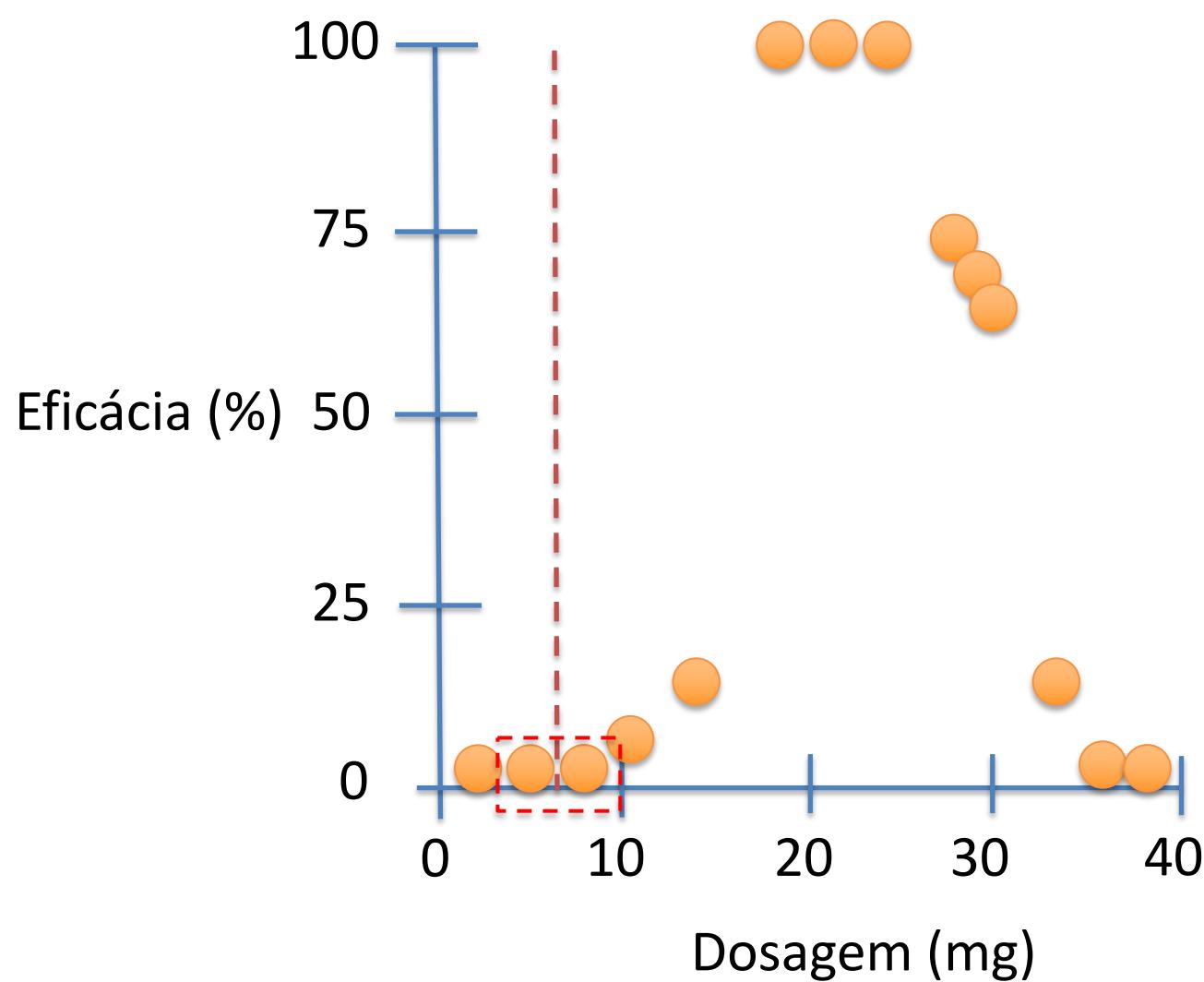
- É uma técnica estatística usada para mensurar a quantidade de variância num conjunto de dados.
- Quanto menor o valor de RSS, melhor o modelo se ajusta aos dados (fit).
- $RSS = \sum_{i=1}^n (y^i - f(x)^i)^2$ 
  - $y^i$  = valor observado para a amostra  $i$
  - $f(x)^i$  = valor predito para a amostra  $i$

- Uma outra métrica que podemos usar para determinar a qualidade de uma árvore de regressão é o Mean Squared Error (MSE)
- A diferença em relação ao RSS é que ele divide o erro pela quantidade de amostras, oferecendo um valor médio.
- $MSE = \frac{1}{n} \sum_{i=1}^n (y^i - f(x)^i)^2$ 
  - $y^i$  = valor observado para a amostra  $i$
  - $f(x)^i$  = valor predito para a amostra  $i$

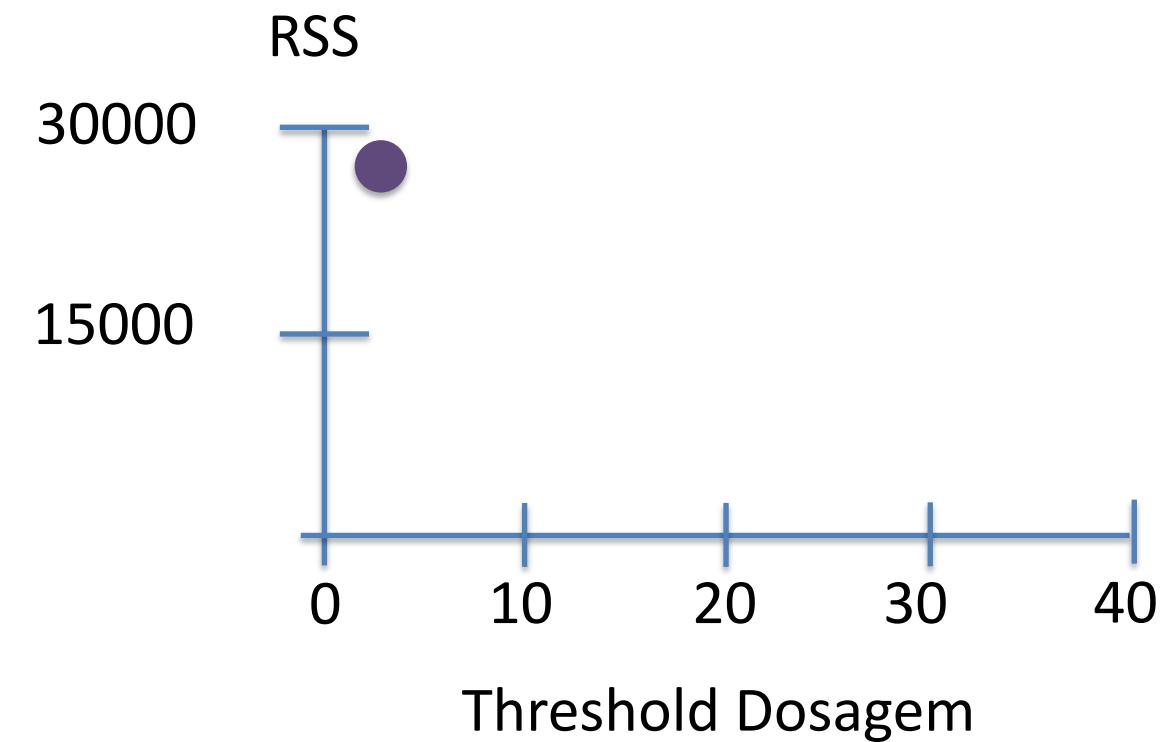


Podemos plotar o valor de RSS de acordo com cada threshold de dosagem para escolher o melhor ponto raiz para construir a árvore

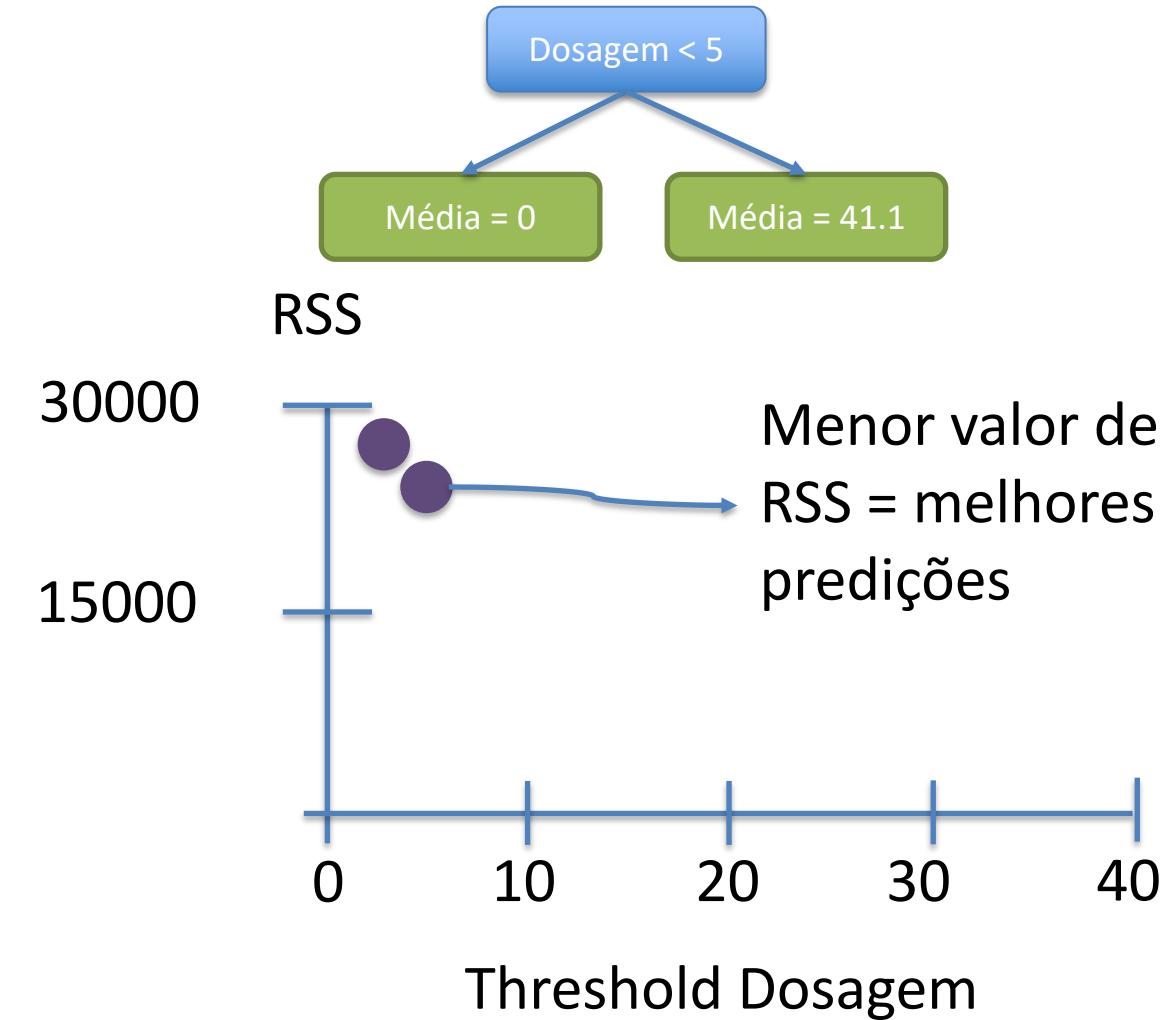
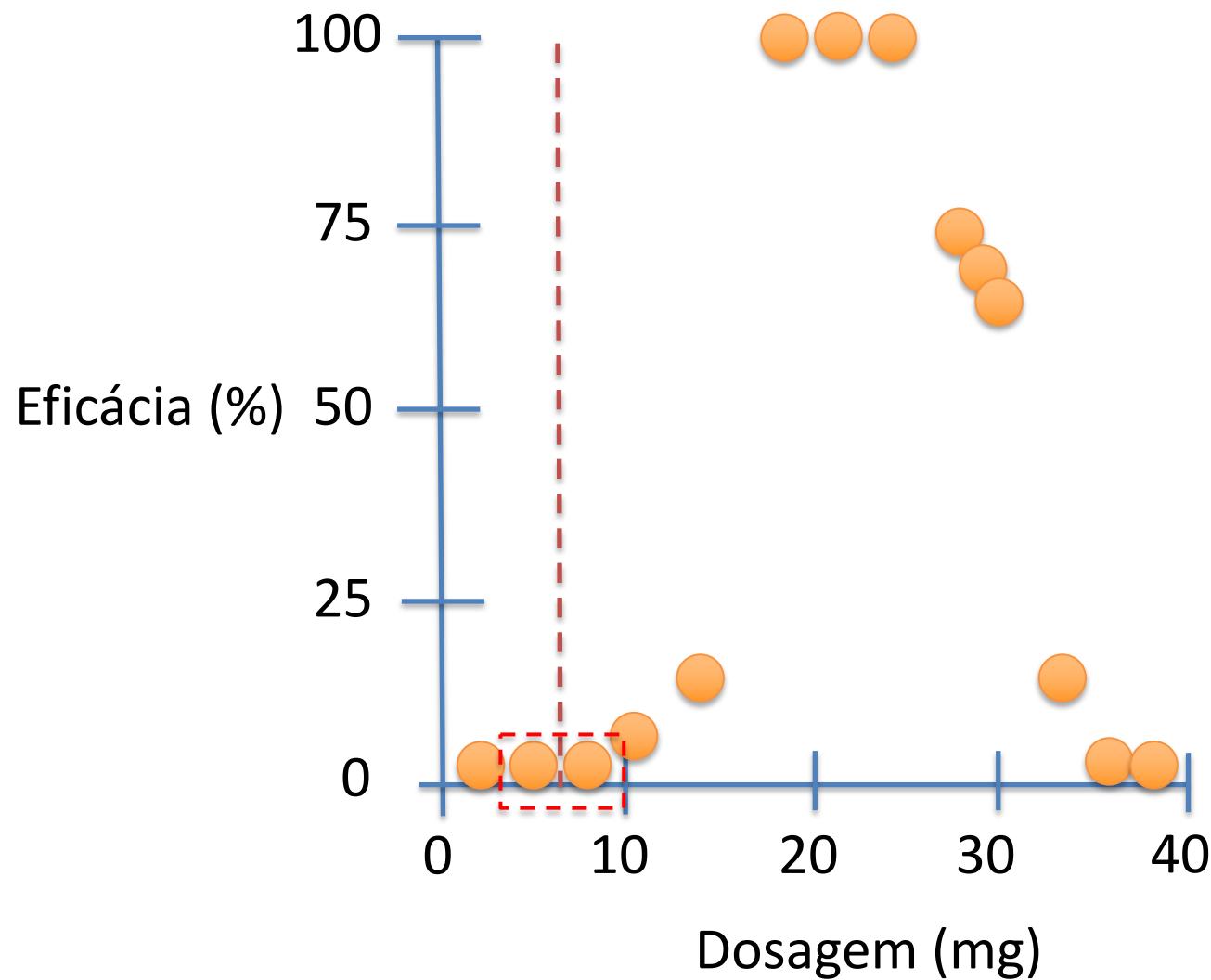


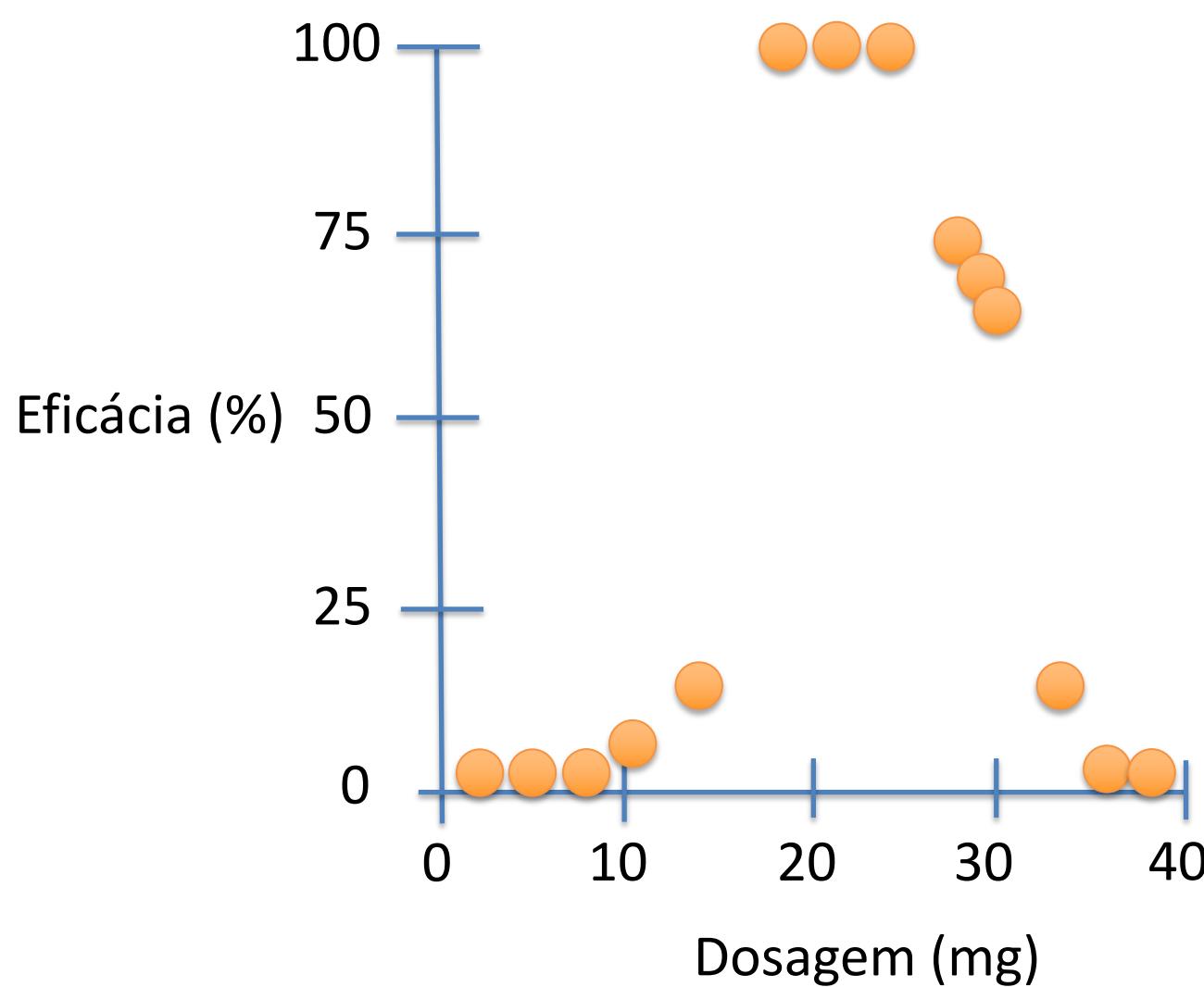


Agora, pegamos o valor médio das duas amostras subsequentes a primeira e criamos um novo threshold

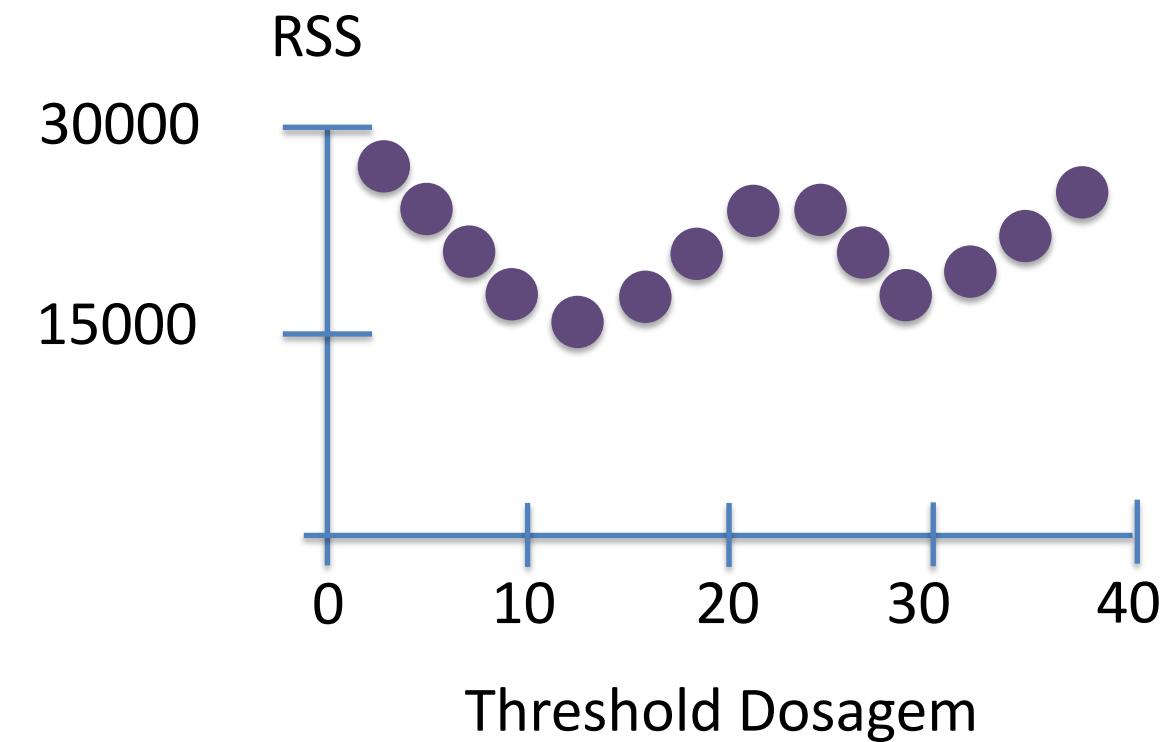


# Regression Trees – Como construir uma Regression Tree?

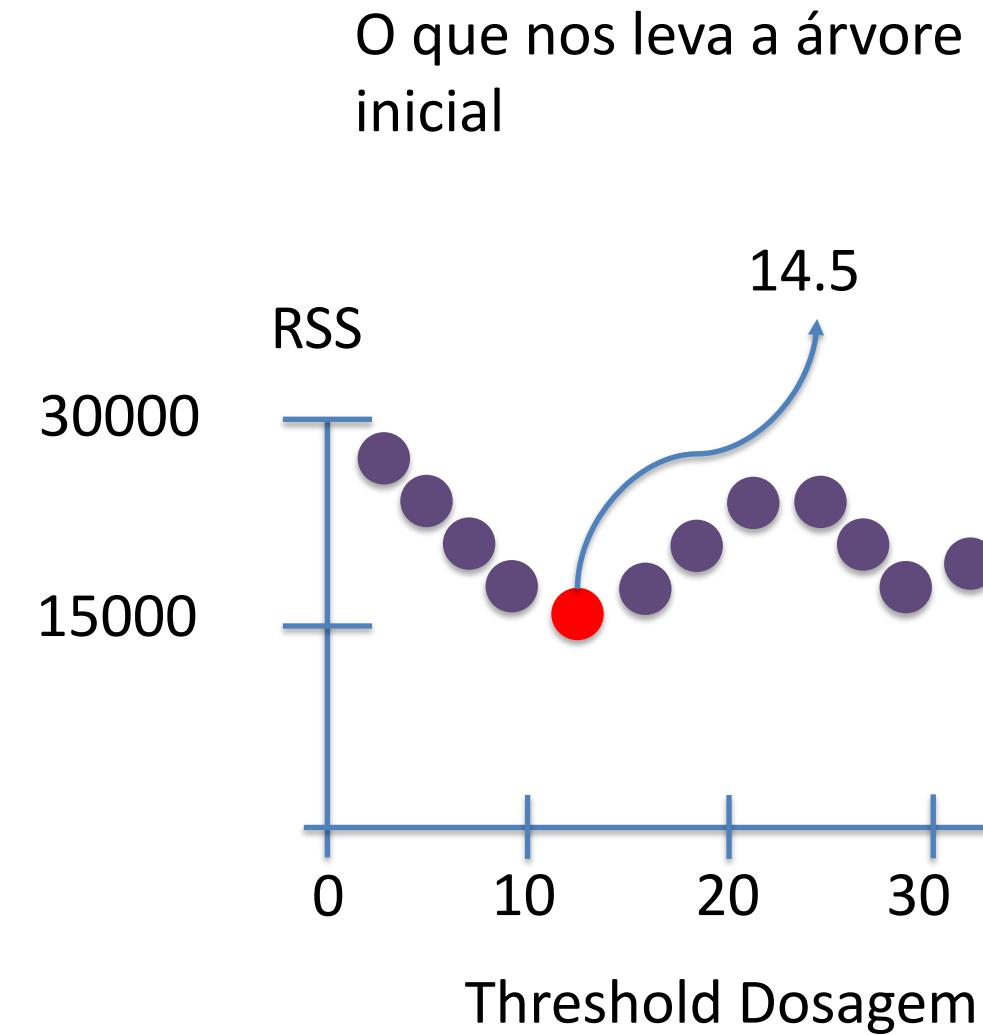
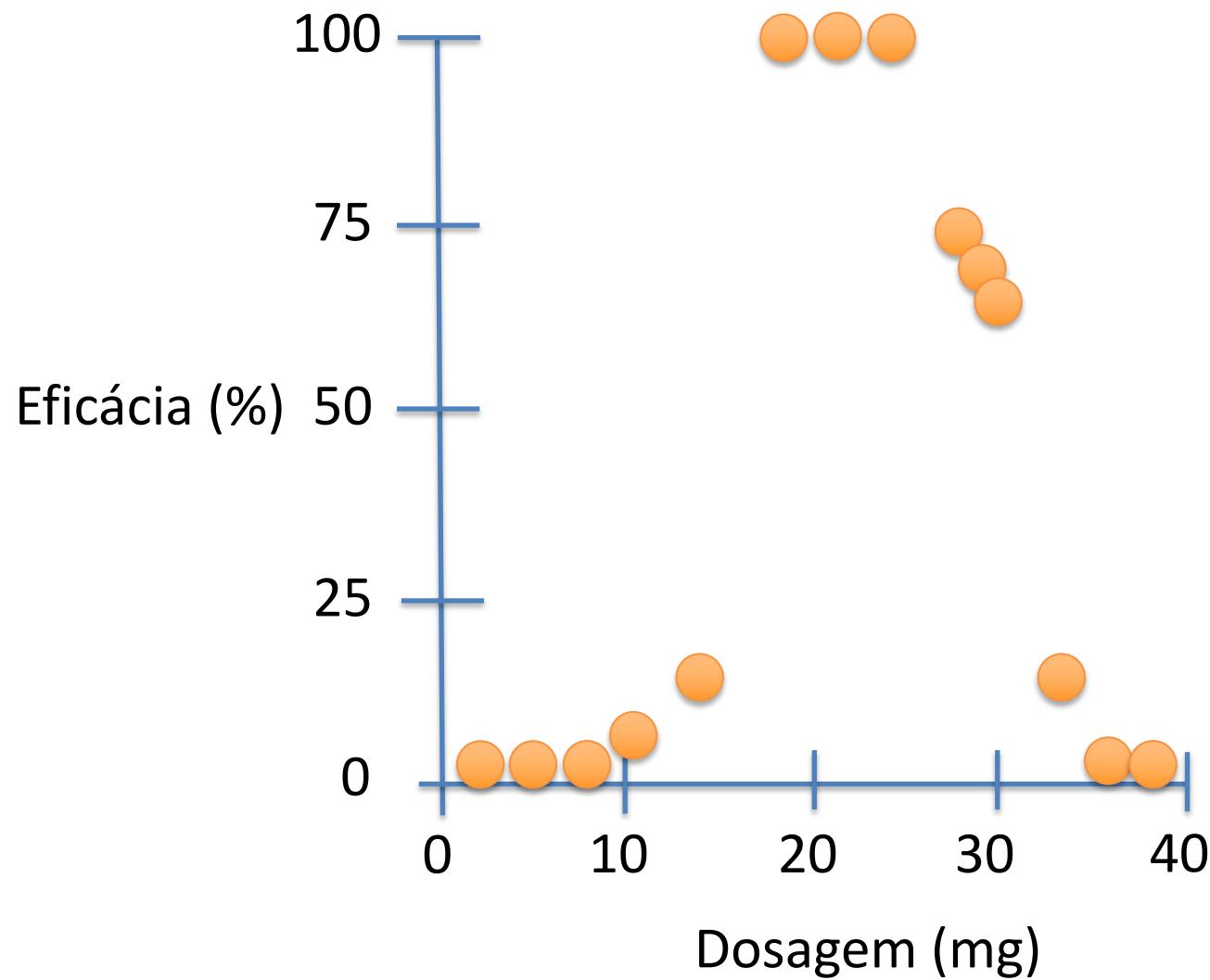




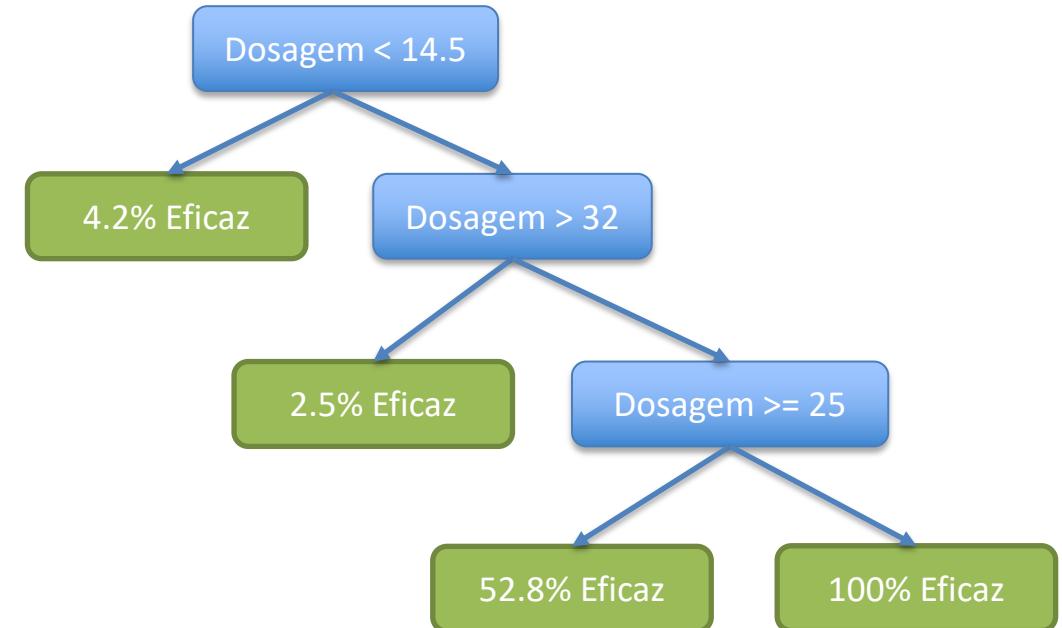
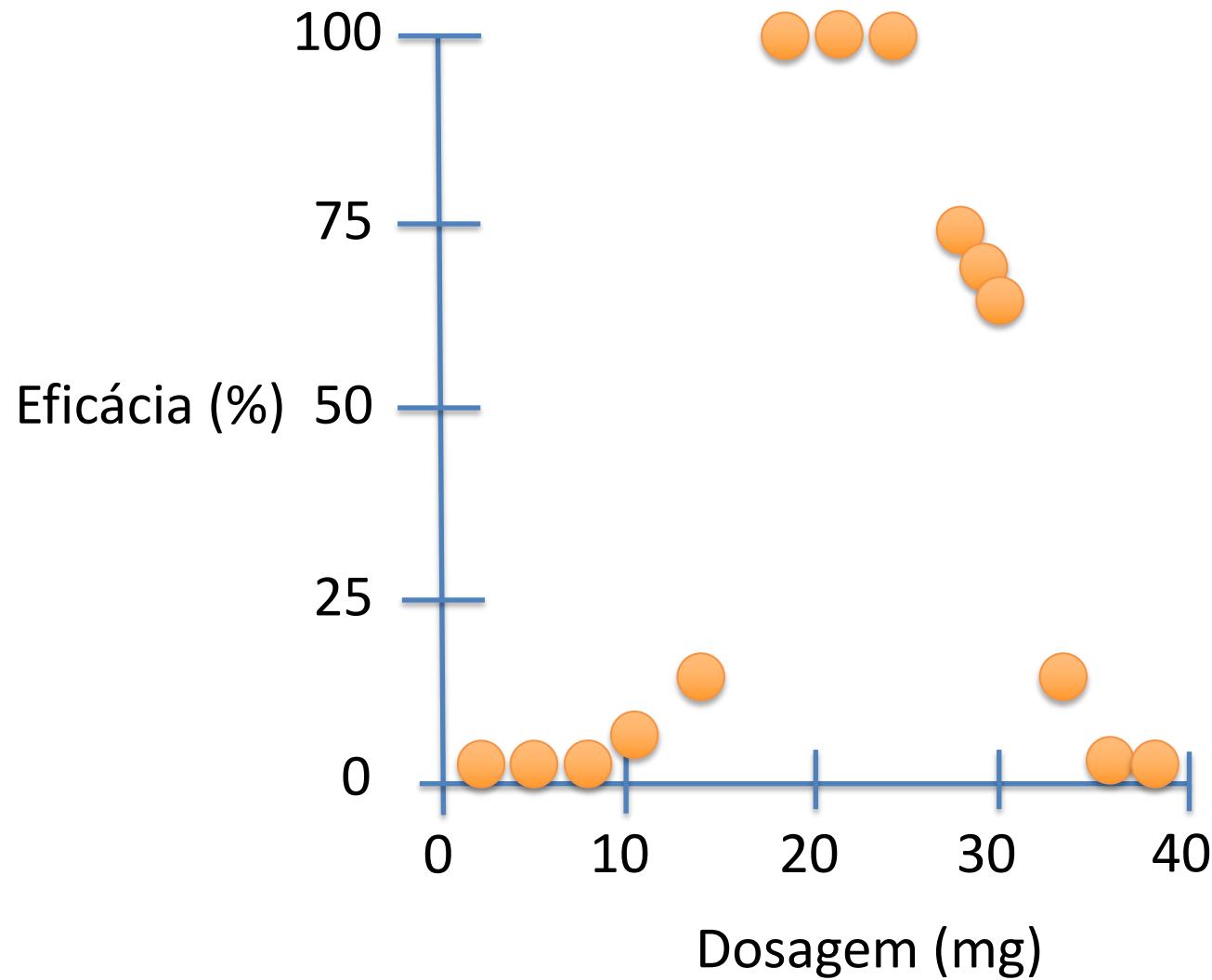
Repetindo o processo para todos os pontos, podemos encontrar o threshold que retorna o menor valor de RSS



# Regression Trees – Como construir uma Regression Tree?



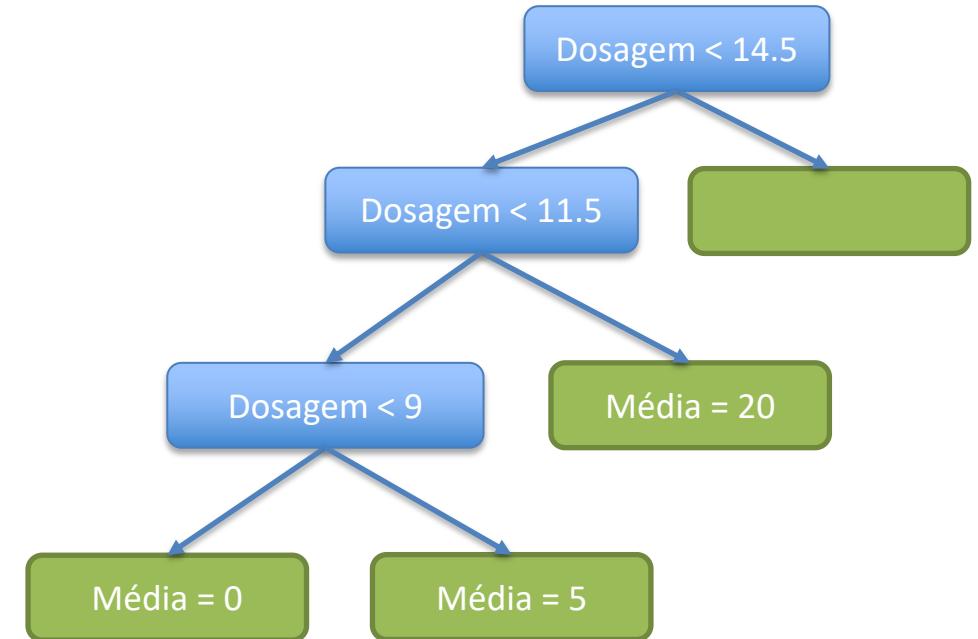
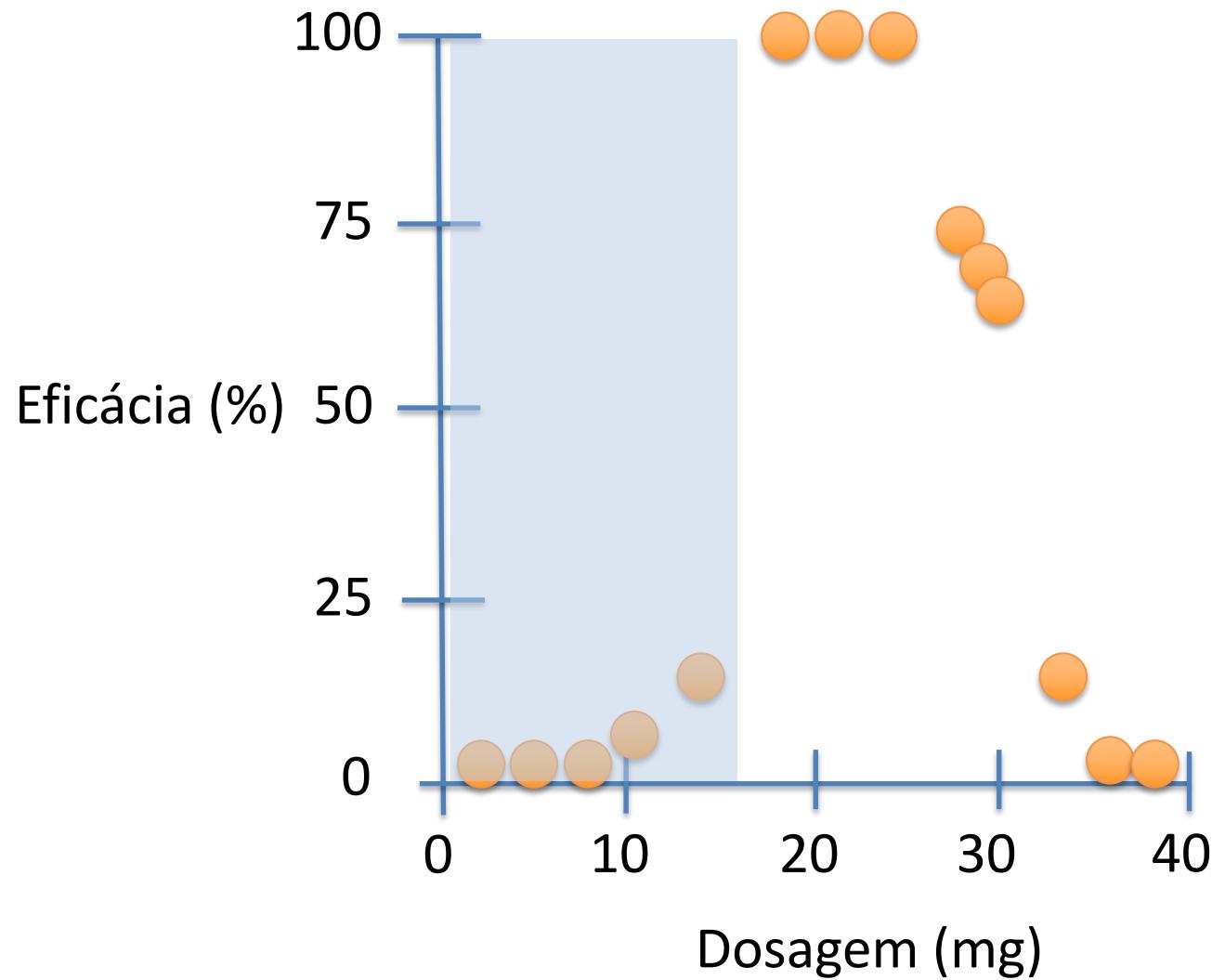
# Regression Trees - Exemplo



- Em resumo, dividimos os dados em dois grupos encontrando o valor de threshold que nos retorna o menor valor de RSS

- Um ponto importante a se observar ao realizar o split é se há um número mínimo de amostras em cada divisão.
- Focando nas amostras demarcadas, poderíamos criar a seguinte árvore:

# Regression Trees - Overfitting

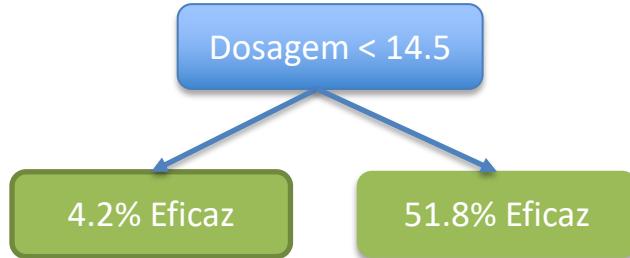


Importante notar que todas as previsões são perfeitas.  
Isso é bom?

- Geralmente, quando existe um fit perfeito no conjunto de dados, há overfitting, isto é, o modelo torna-se especialista no conjunto de treino, mas não sabe generalizar para amostras ainda não vistas
- Bias/Variance tradeoff

- Assim, uma maneira de prevenir esse erro é dividir apenas quando existe mais observações do que um número pré-determinado.
- 20, geralmente, é um bom número para se iniciar, mas é importante testar outros números.

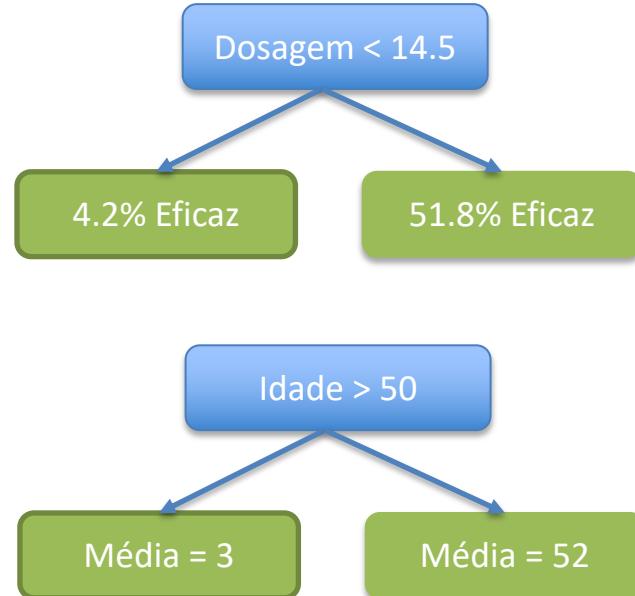
- Até agora, usamos apenas um preditor (característica) para prever a dosagem.
- Vamos agora entender como construir uma árvore de regressão usando mais de 1 preditor.



Escolhemos, para esse preditor, o valor que retorna o menor RSS

Dosagem	Idade	Sexo	Eficácia
10	25	Feminino	98
20	73	Masculino	0
35	54	Feminino	100
5	12	Masculino	44
...	...	...	...

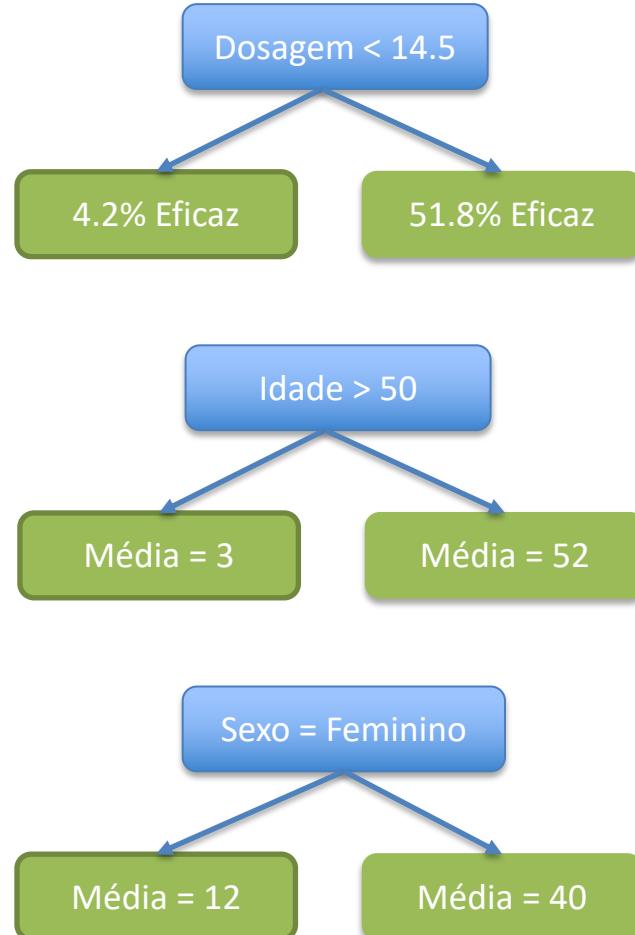
Iniciamos avaliando cada preditor como um candidato a raiz de acordo com o valor de RSS obtido. Repetimos o processo que vimos para construir a árvore para cada preditor.



Dosagem	Idade	Sexo	Eficácia
10	25	Feminino	98
20	73	Masculino	0
35	54	Feminino	100
5	12	Masculino	44
...	...	...	...

Repetimos o processo para a variável Idade e escolhemos o threshold que retorna o menor valor de RSS

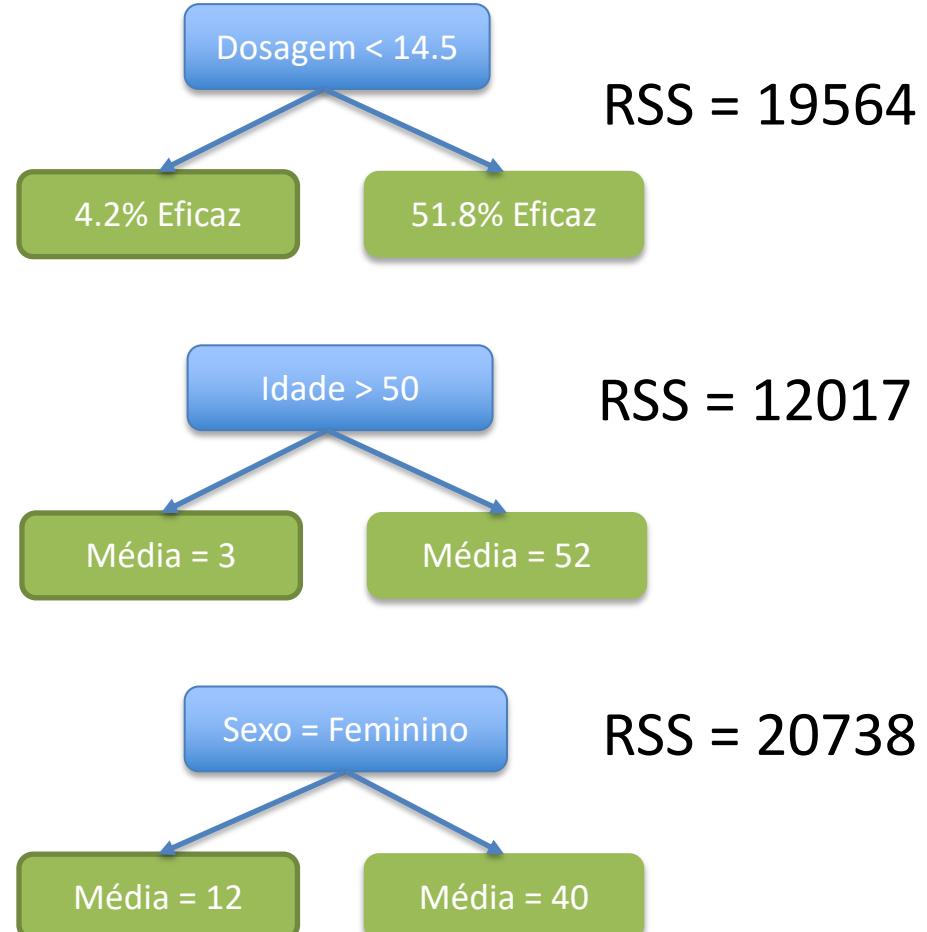
# Regression Trees - 2 ou mais Predictors



Dosagem	Idade	Sexo	Eficácia
10	25	Feminino	98
20	73	Masculino	0
35	54	Feminino	100
5	12	Masculino	44
...	...	...	...

Fazemos o mesmo para a variável Sexo

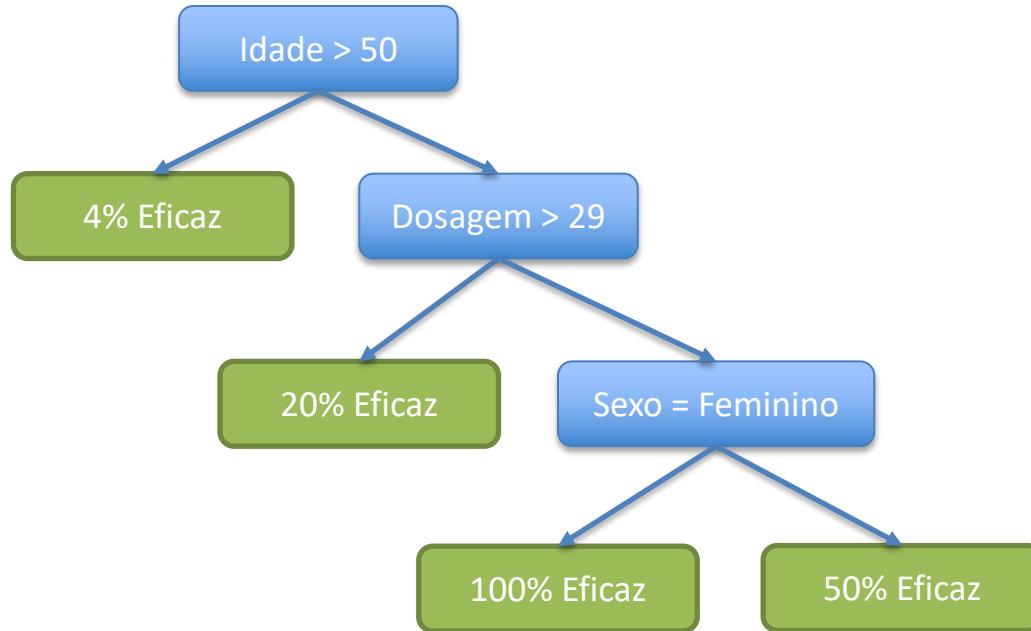
# Regression Trees - 2 ou mais Predictors



Dosagem	Idade	Sexo	Eficácia
10	25	Feminino	98
20	73	Masculino	0
35	54	Feminino	100
5	12	Masculino	44
...	...	...	...

Agora, comparamos o valor de RSS de cada candidato e escolhemos o menor

# Regression Trees - Exemplo



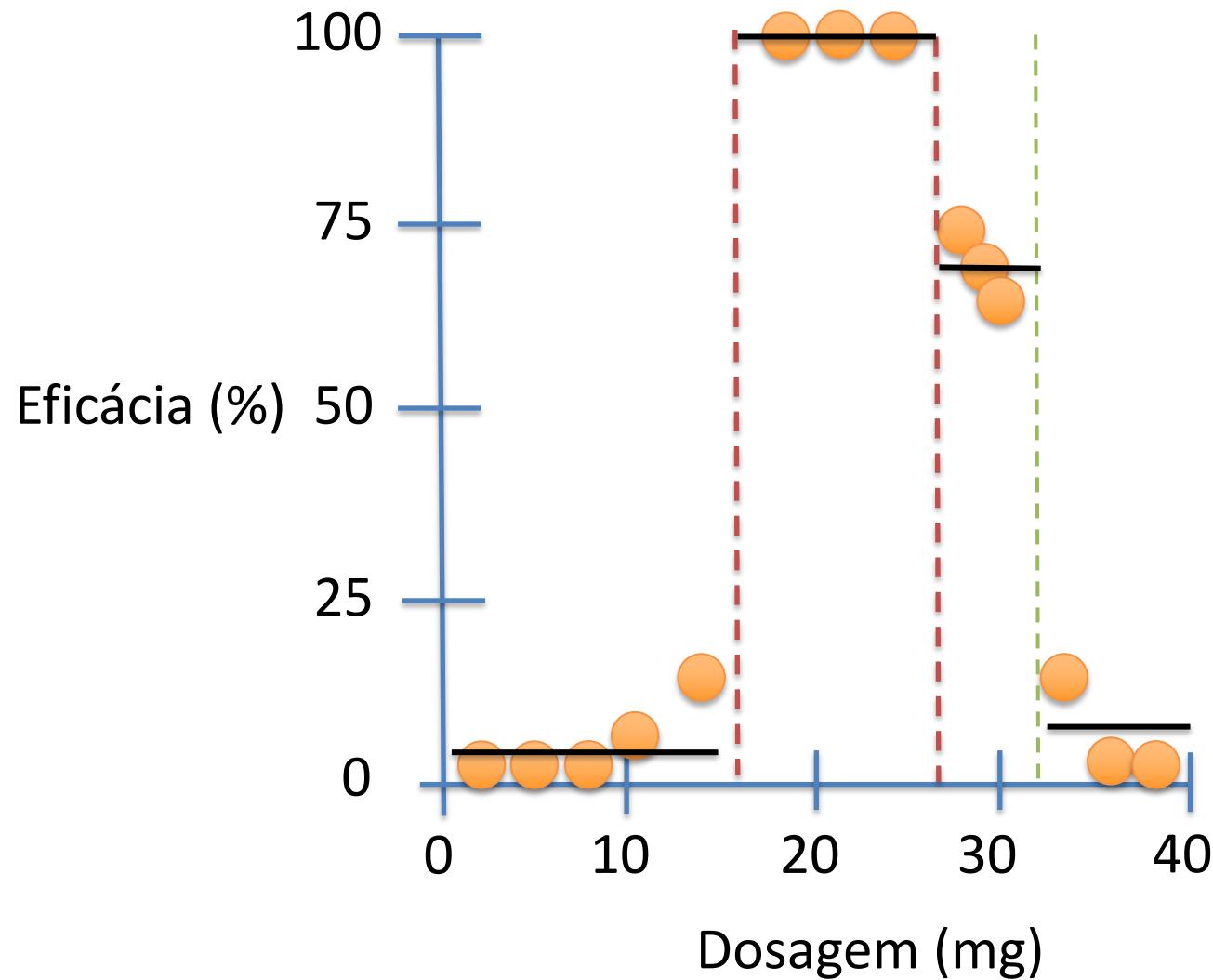
Dosagem	Idade	Sexo	Eficácia
10	25	Feminino	98
20	73	Masculino	0
35	54	Feminino	100
5	12	Masculino	44
...	...	...	...

Agora, tomamos o candidato com menor valor de RSS e usamos como raiz e construímos a árvore levando em conta os valores de RSS, em ordem crescente.

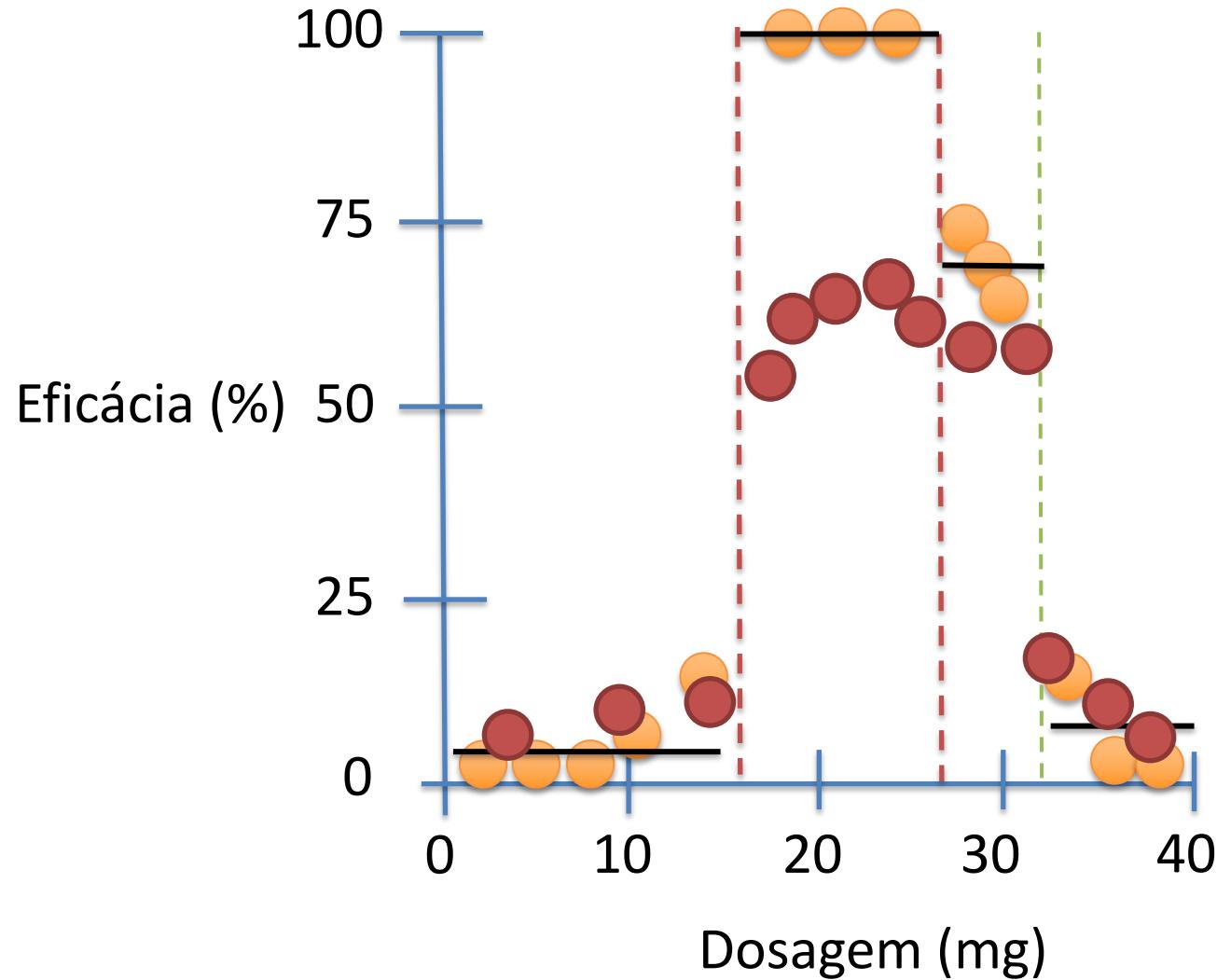
# Pruning

Overfitting pode ser novamente um problema quando vamos predizer um conjunto de teste. Isso significa que existe um erro muito pequeno na etapa de treinamento e um erro muito grande na etapa de teste.

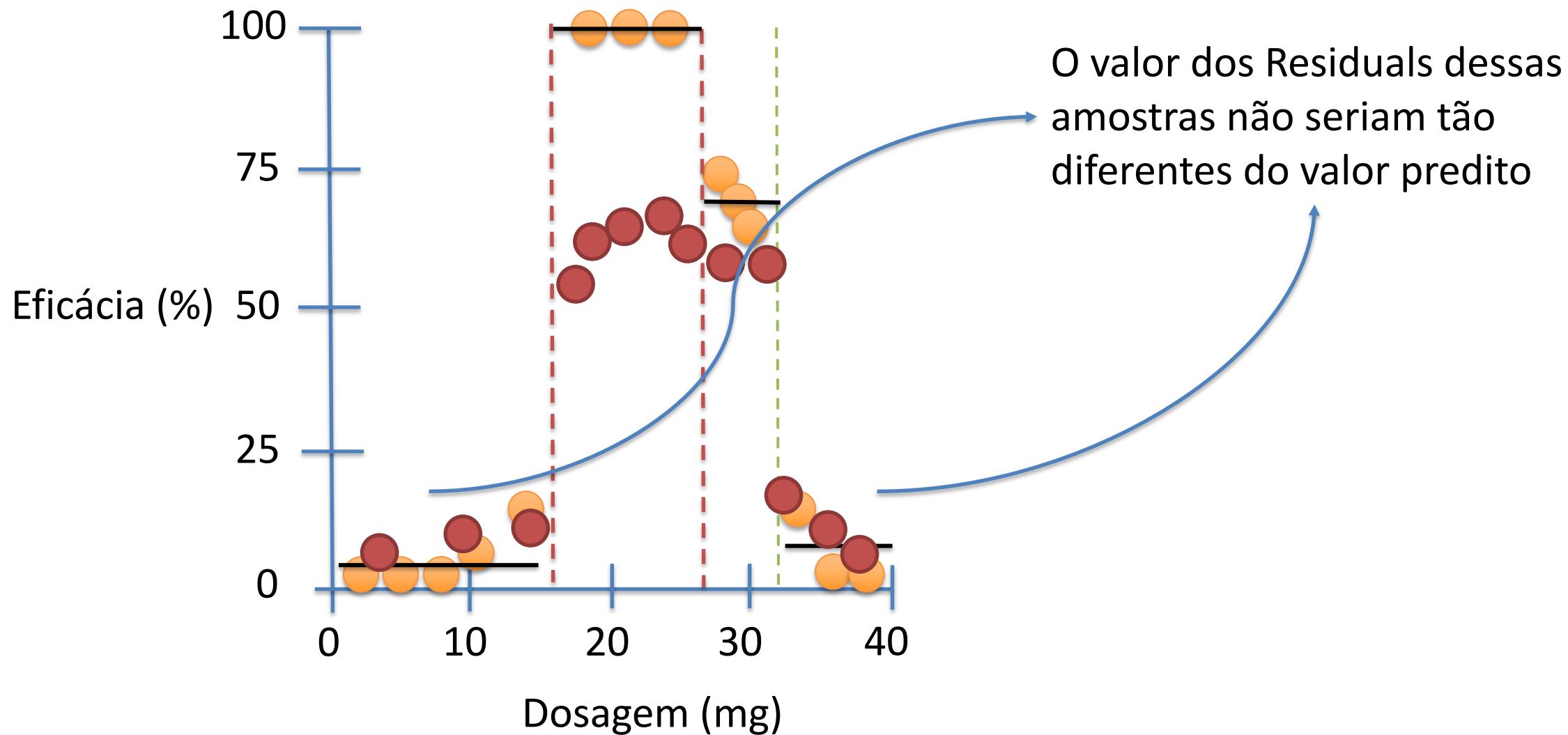
Vamos analisar o conjunto de treino novamente:

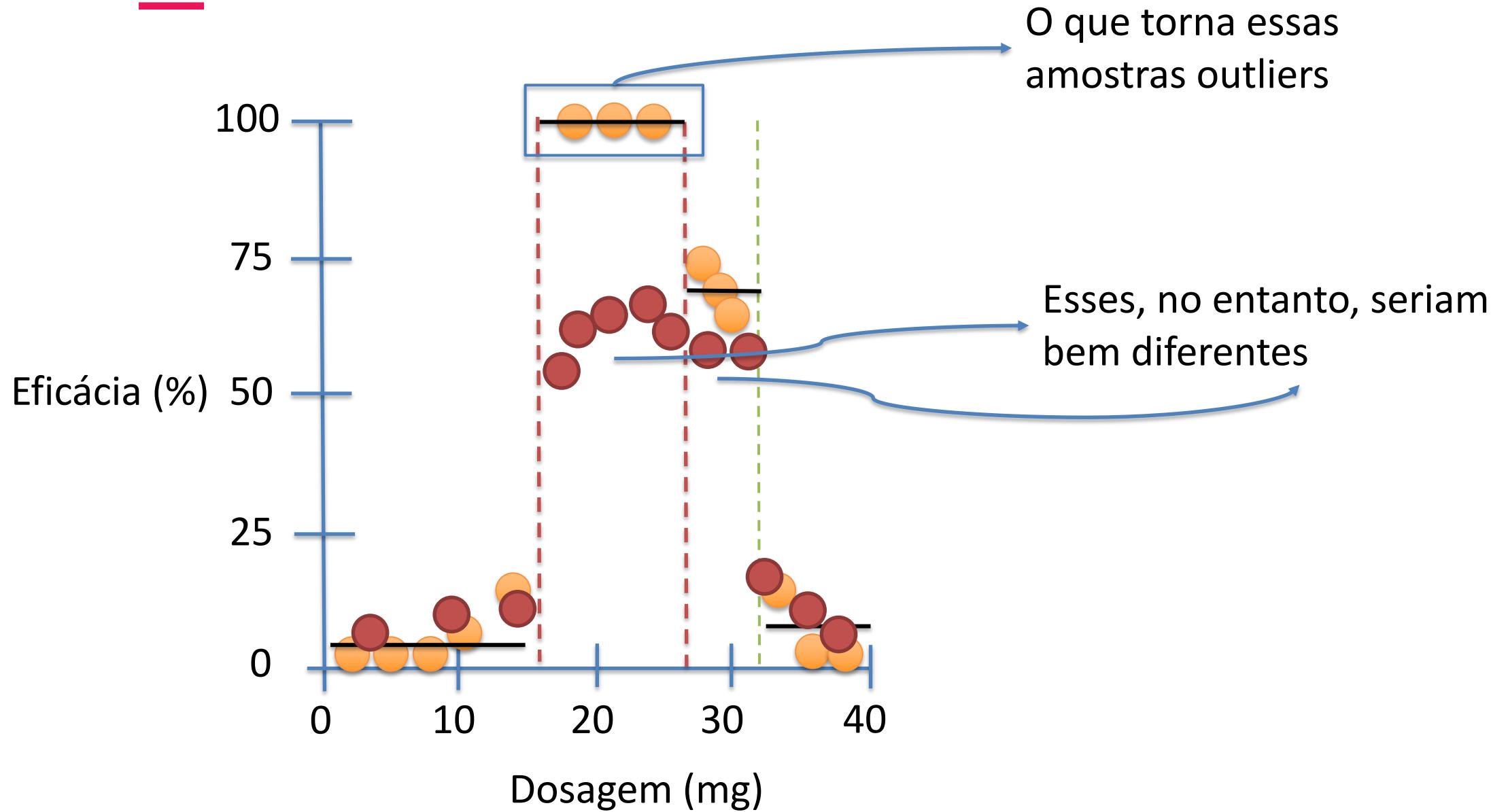


Dados de treinamento com a região que representa a árvore e seus respectivos valores médios de eficácia.

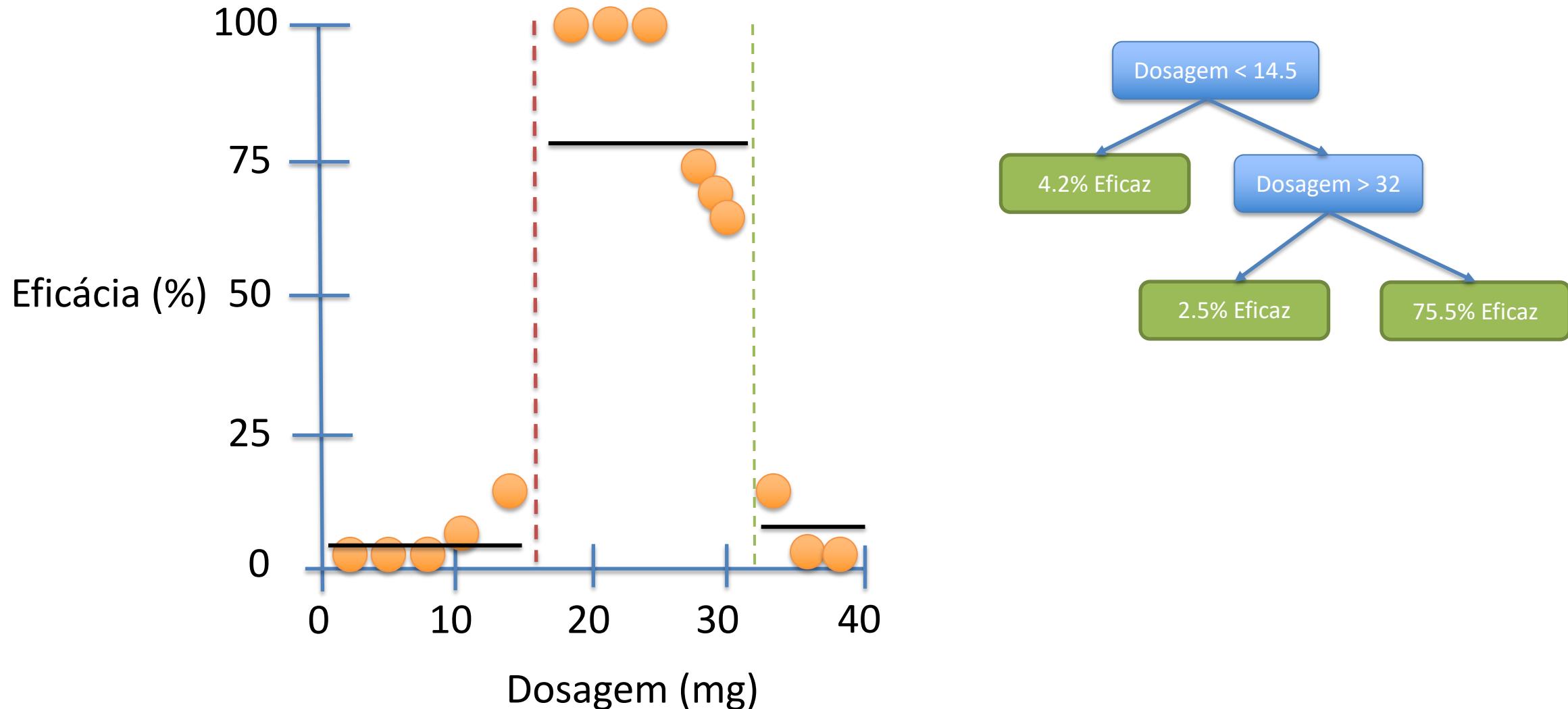


O que aconteceria se esses fossem o conjunto de teste?

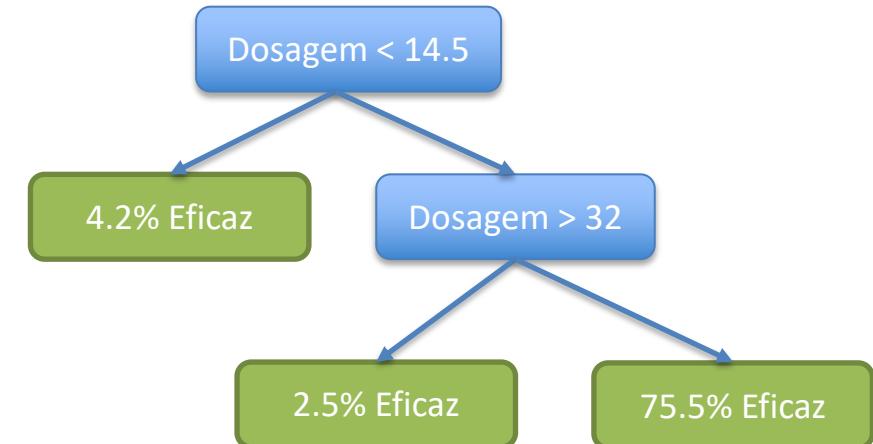
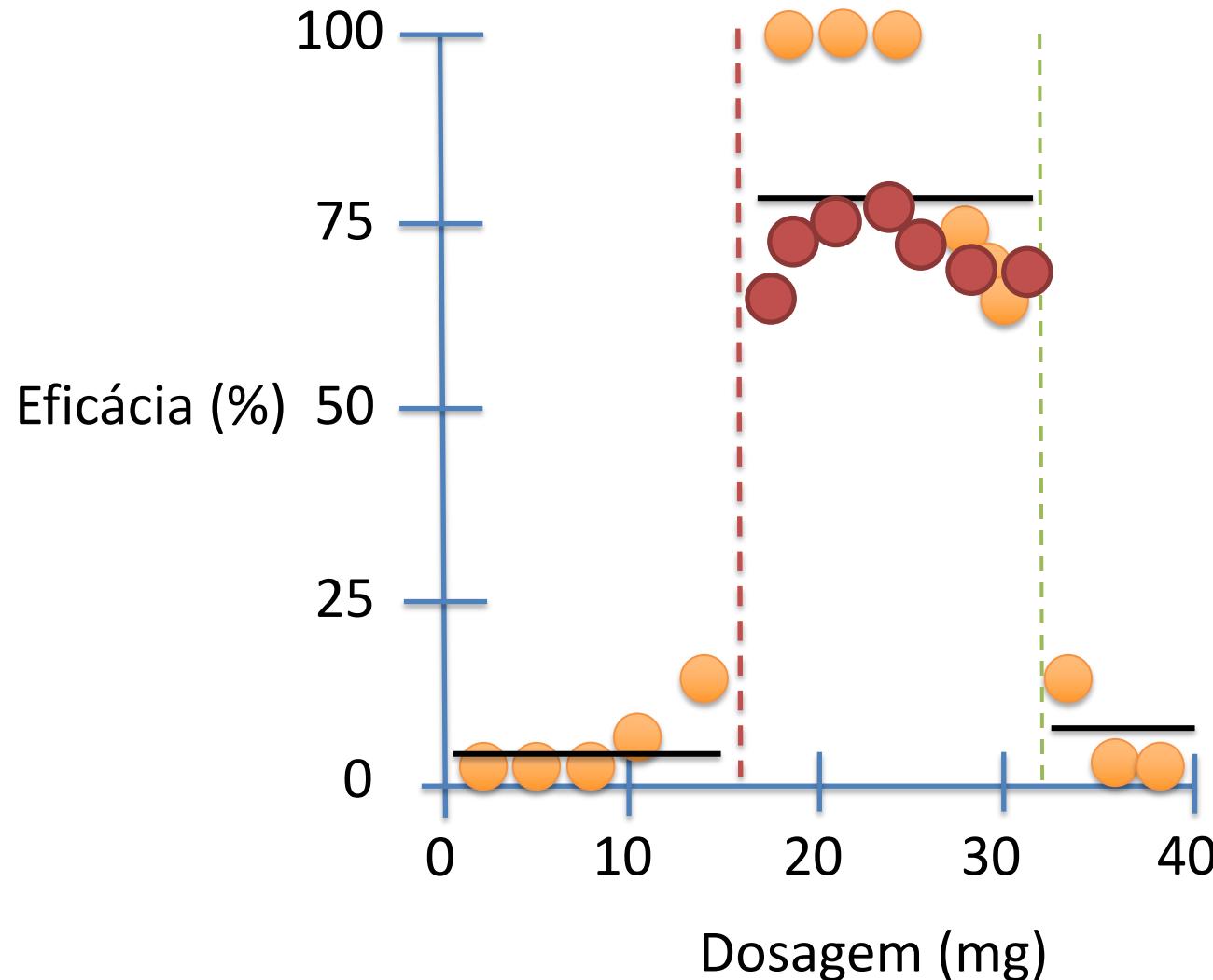




Uma maneira de eliminar overfitting é remover algumas das folhas

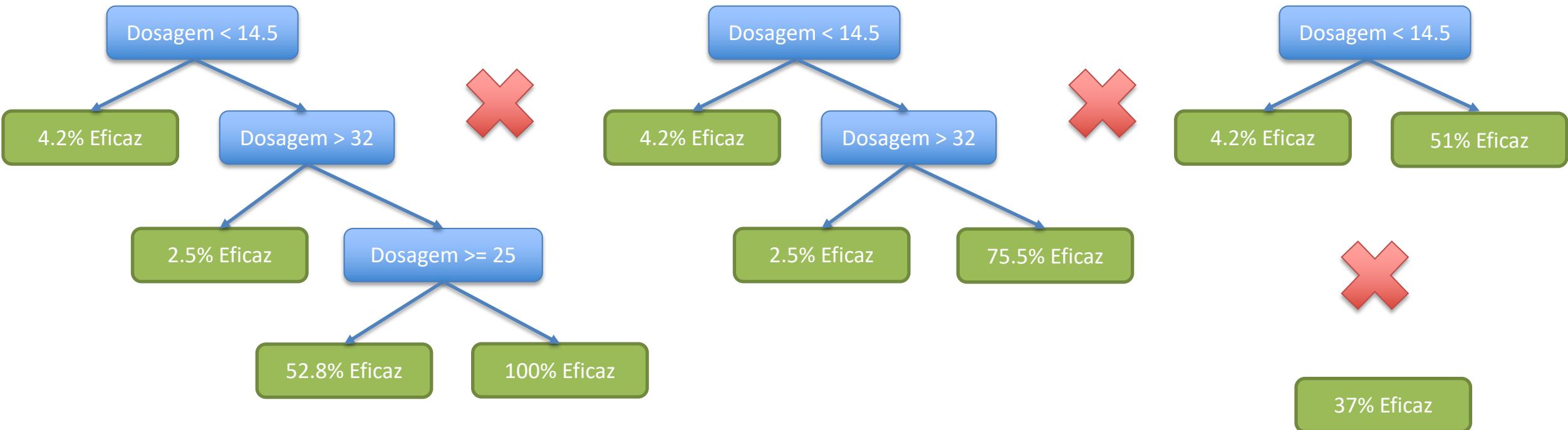


Uma maneira de eliminar overfitting é remover algumas das folhas



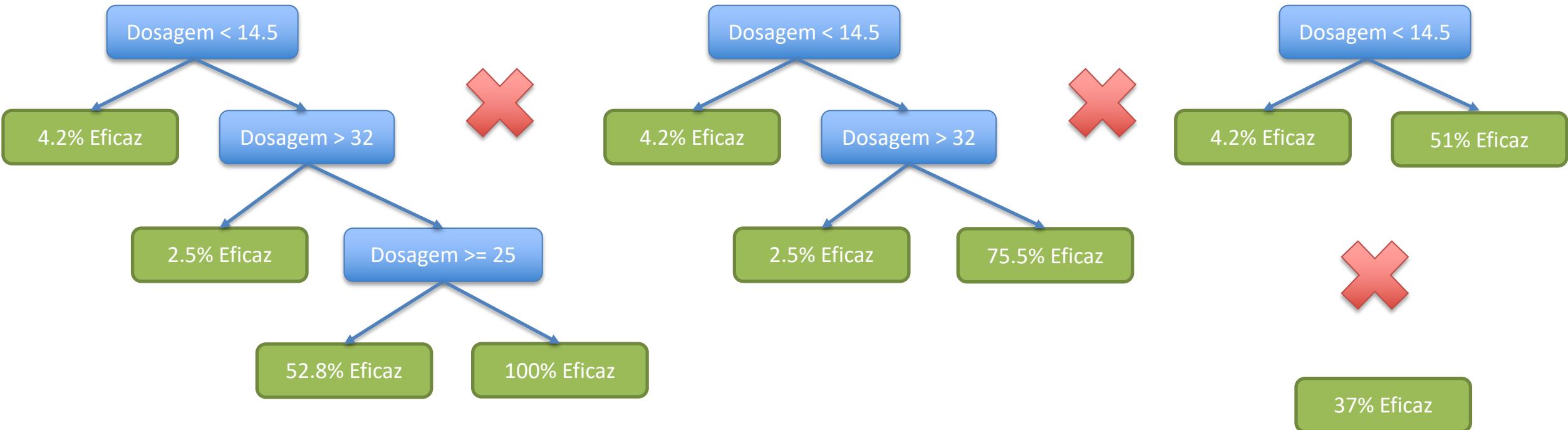
Isso aumenta um pouco os Residuals na etapa de treino, mas essa nova sub-árvore oferece um resultado melhor para os dados de teste

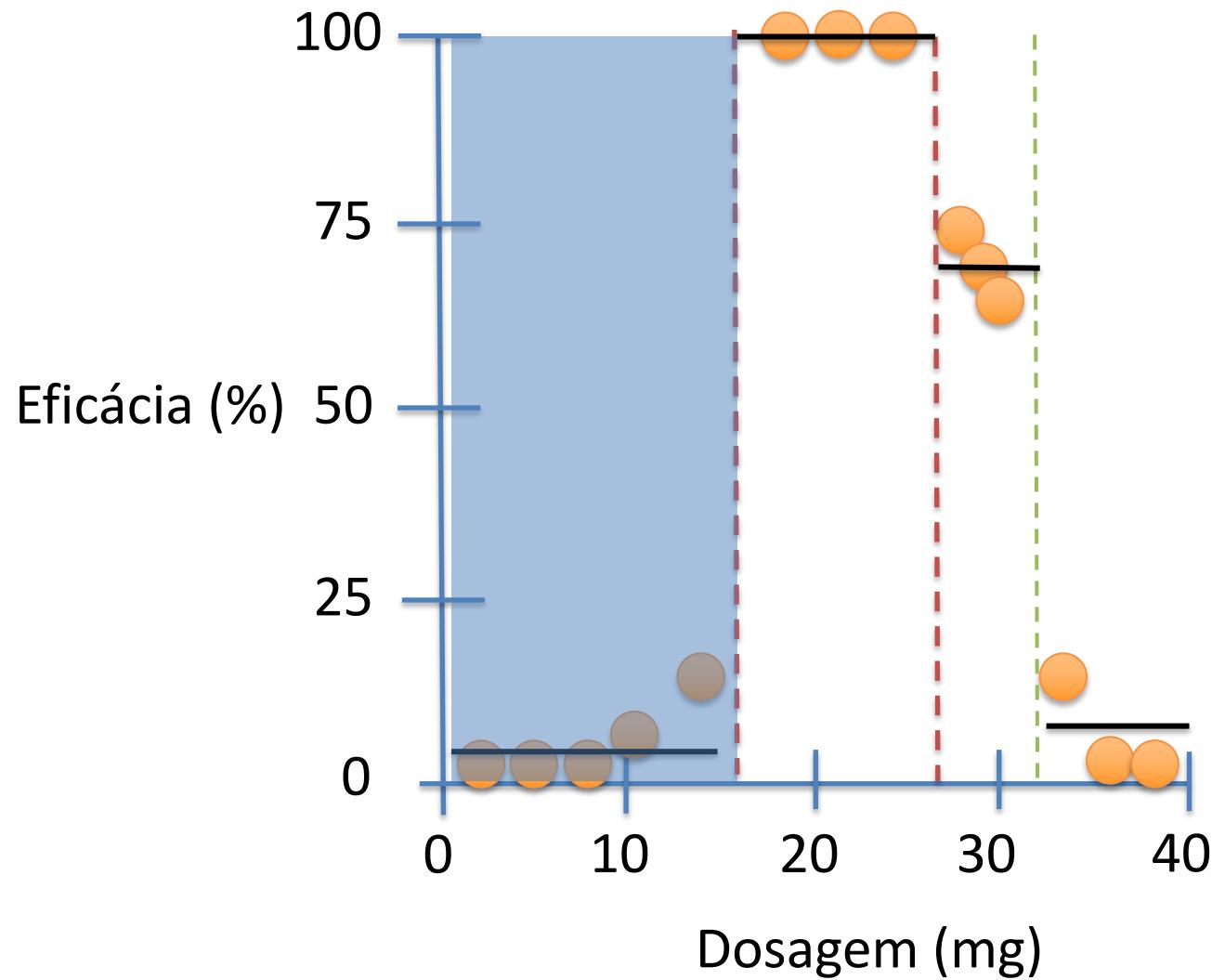
Podemos continuar removendo folhas até permanecer apenas a raiz. Assim, a decisão consiste em determinar qual a melhor árvore a ser utilizada



- Para responder essa questão, usaremos Cost Complexity Pruning, um algoritmo parametrizado por  $\alpha \geq 0$ , conhecido como parâmetro de complexidade, que é usado para mensurar  $R_\alpha(T)$  (cost complexity) para uma dada árvore  $T$ .
- $$R_\alpha(T) = R(T) + \alpha|\tilde{T}|$$
- Em que,  $|\tilde{T}|$  é o número de nós terminais (folhas) e  $R(T)$  é a taxa de erro de predição total dos nós terminais (RSS, em nosso caso)

O primeiro passo é calcular o RSS de cada árvore:





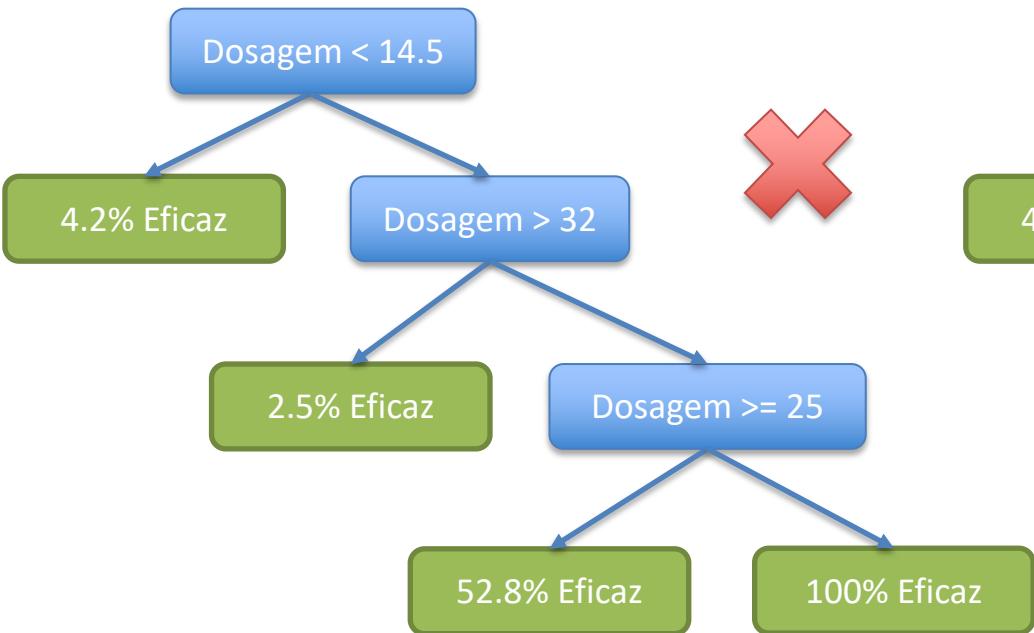
Dosagem < 14.5  
4.2% Eficaz

$$(0 - 4.2)^2 + (0 - 4.2)^2 + (0 - 4.2)^2 + (5 - 4.2)^2 = 303.2$$

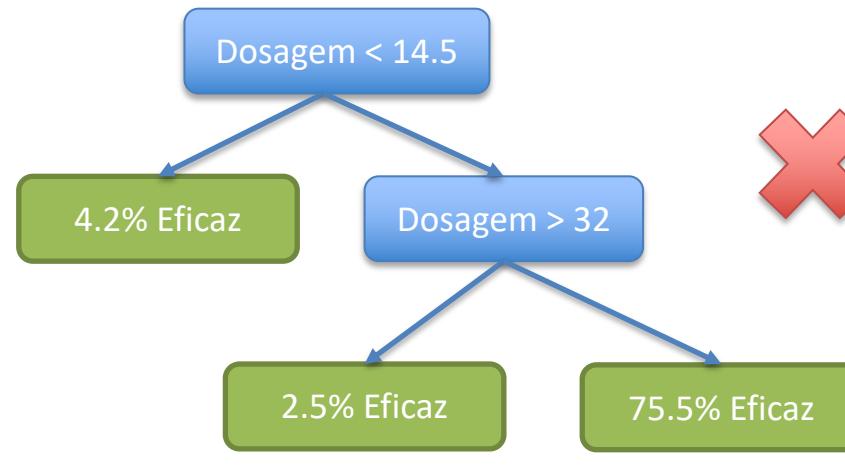
Depois, eu repito o processo para cada parte e somo o resultado. Em seguida, o processo é feito para cada árvore.

O primeiro passo é calcular o RSS de cada árvore:

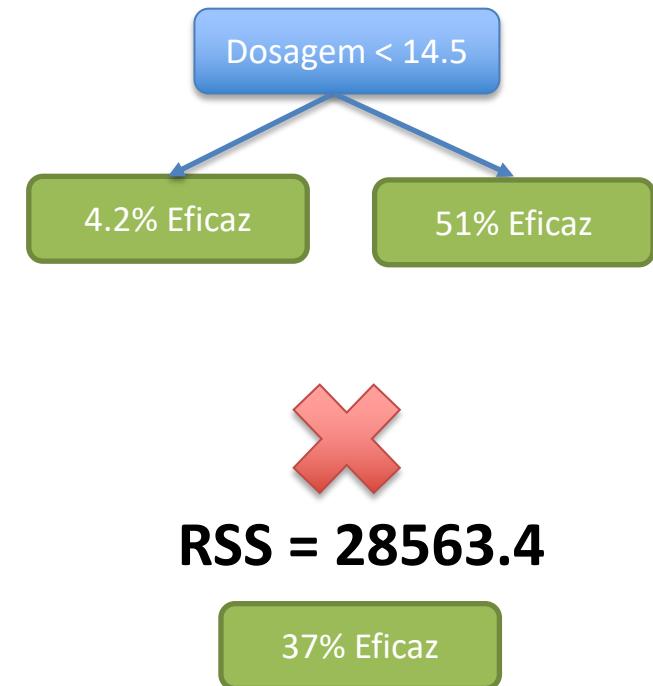
**RSS = 504.6**



**RSS = 5215.8**

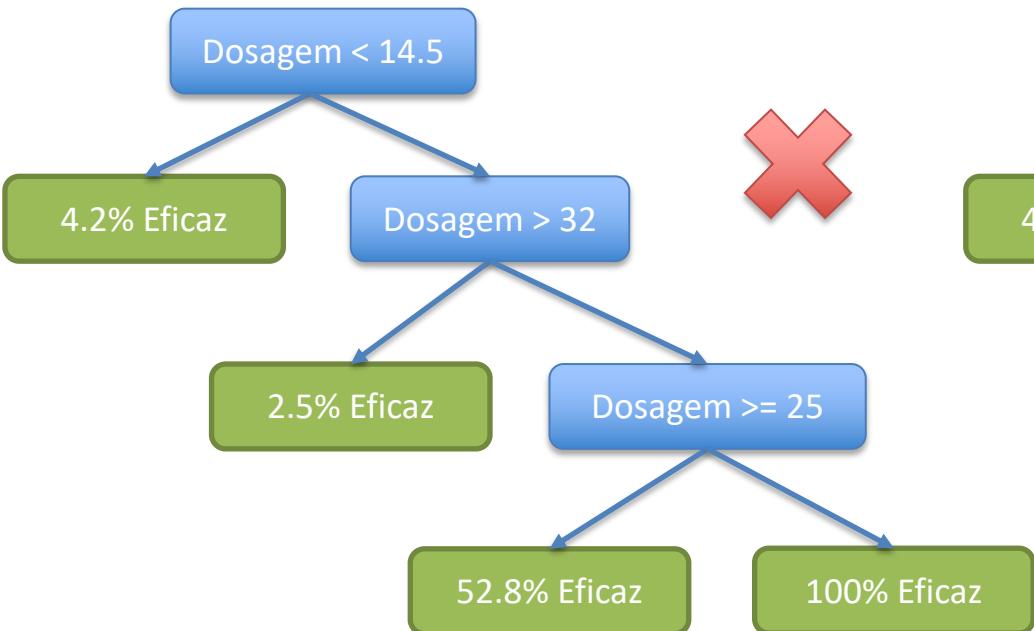


**RSS = 19156.7**

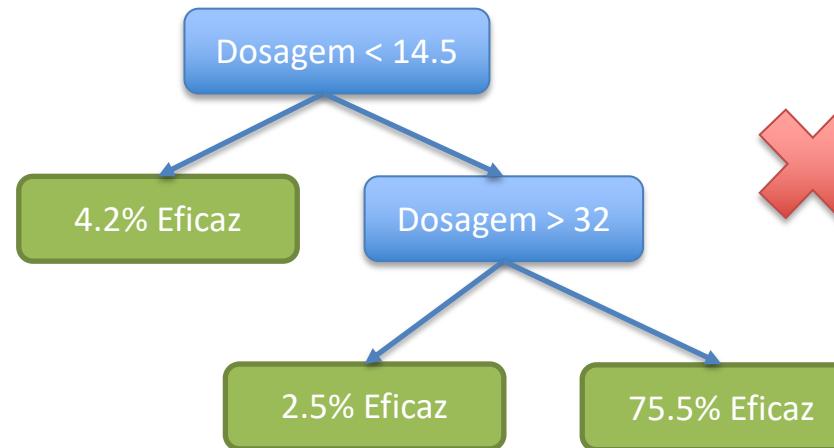


Agora, calculamos o parâmetro  $\alpha|\tilde{T}|$  para determinar o Score de cada árvore ( $R_\alpha(T)$ )

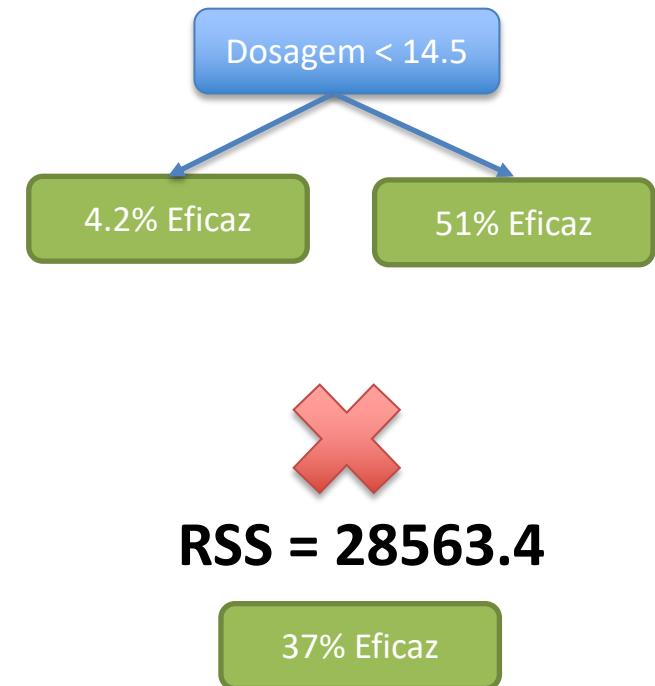
**RSS = 504.6**



**RSS = 5215.8**

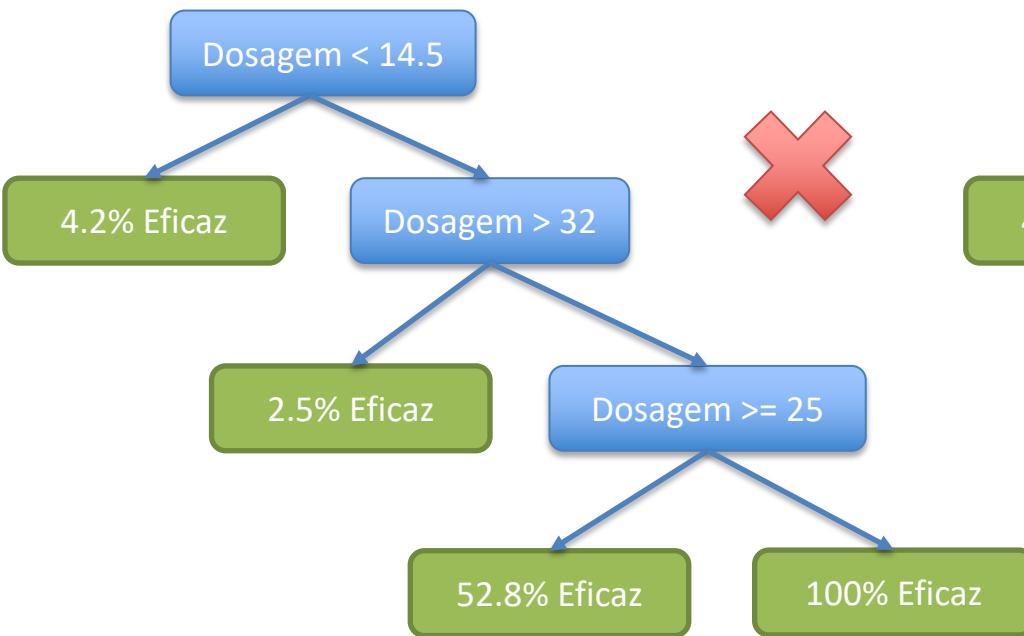


**RSS = 19156.7**

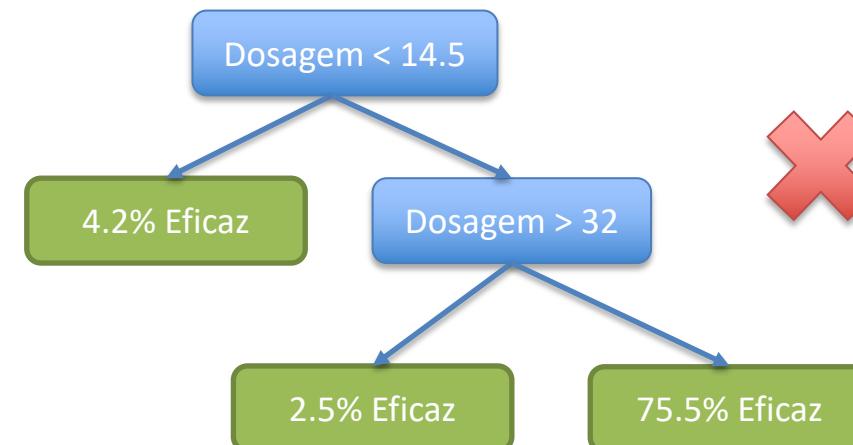


O parâmetro  $\alpha$  serve para ponderar a diferença no número de folhas entre as árvores. Podemos usar cross-validation ou separar o dataset em conjunto de treino e teste para determinar o valor de  $\alpha$  que maximiza a acurácia no conjunto de teste

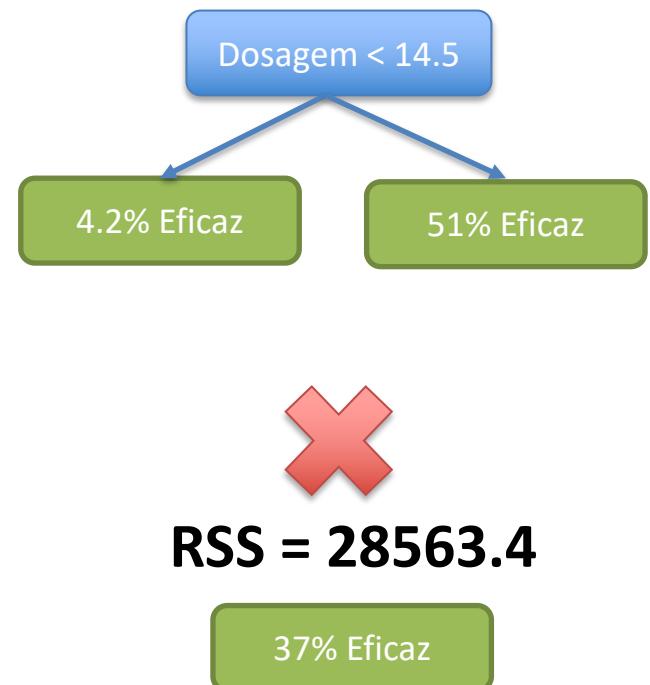
**RSS = 504.6**



**RSS = 5215.8**

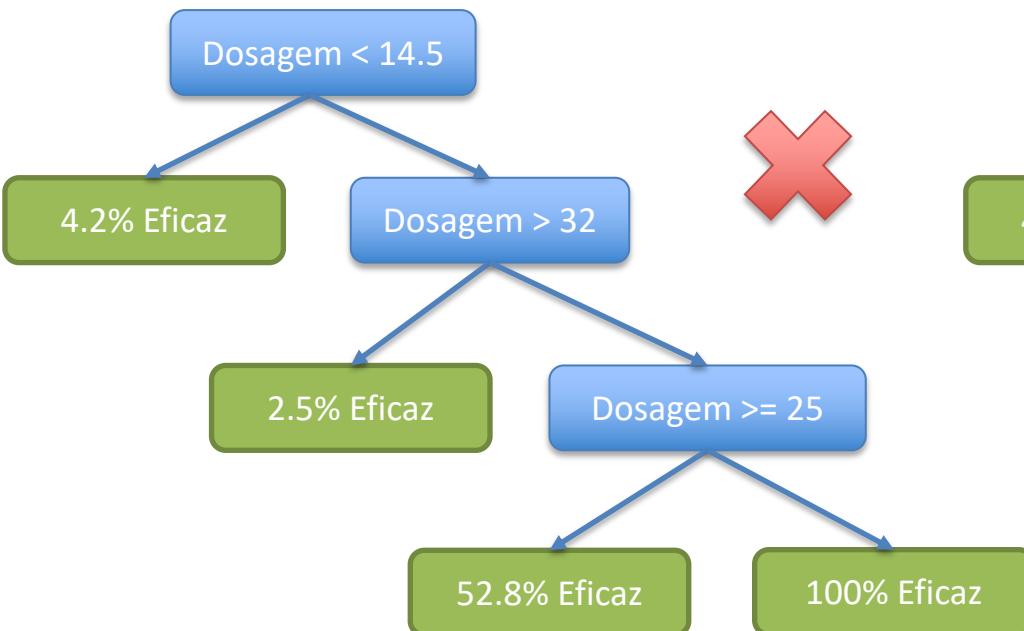


**RSS = 19156.7**



A título de exemplo, tomemos  $\alpha = 10000$

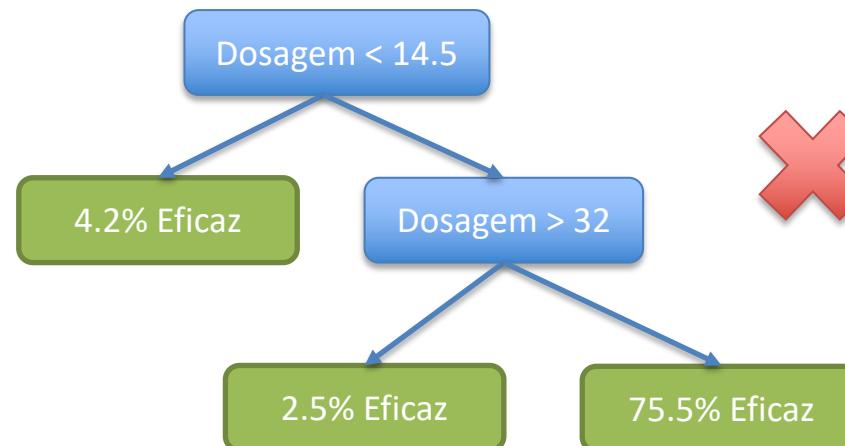
**RSS = 504.6**



$$R_\alpha(T) = R(T) + \alpha |\tilde{T}| =$$

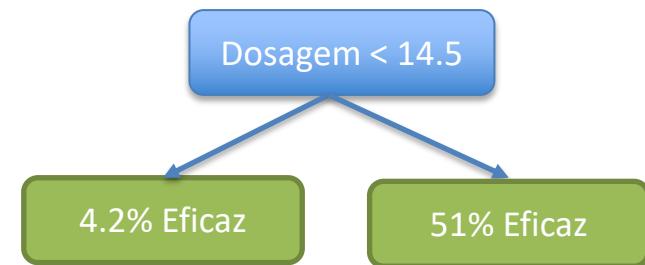
$$504.6 + 10000 \times 4 = 40504.6$$

**RSS = 5215.8**



$$R_\alpha(T) = 34898.2$$

**RSS = 19156.7**



$$R_\alpha(T) = 39052.2$$



$$RSS = 28563.4$$

37% Eficaz

$$R_\alpha(T) = 37443.9$$

# Decision Trees

*Classification and Regression Trees*, ou CART, é um termo usado para se referir aos algoritmos de Árvore de Decisão que podem ser usados para modelar problemas tanto de classificação quanto de regressão.

O Algoritmo CART é o fundamento para importantes algoritmos como *Bagged Decision Trees*, *Boosted Decision Trees* e *Random Forest*.

Aqui, vamos entender de modo geral as Decision Trees, visto que já estudamos profundamente as Regression Trees

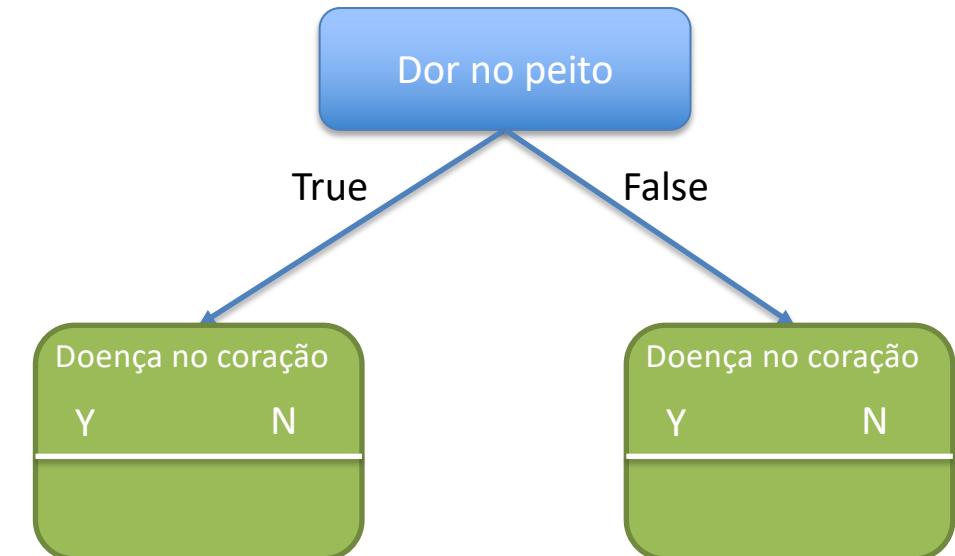
Para começar, vamos entender como construir uma árvore de decisão. Para tanto, suponha que eu possua os seguintes dados:

Dor no peito	Boa circulação Sanguínea	Artérias bloqueadas	Doença cardíaca
N	N	N	N
Y	Y	Y	Y
Y	Y	N	N
Y	N	?	Y
...	...	...	...

Assim como nas regression trees, a primeira coisa a ser feita é avaliar cada feature como uma possível candidata a ser o nó raiz.

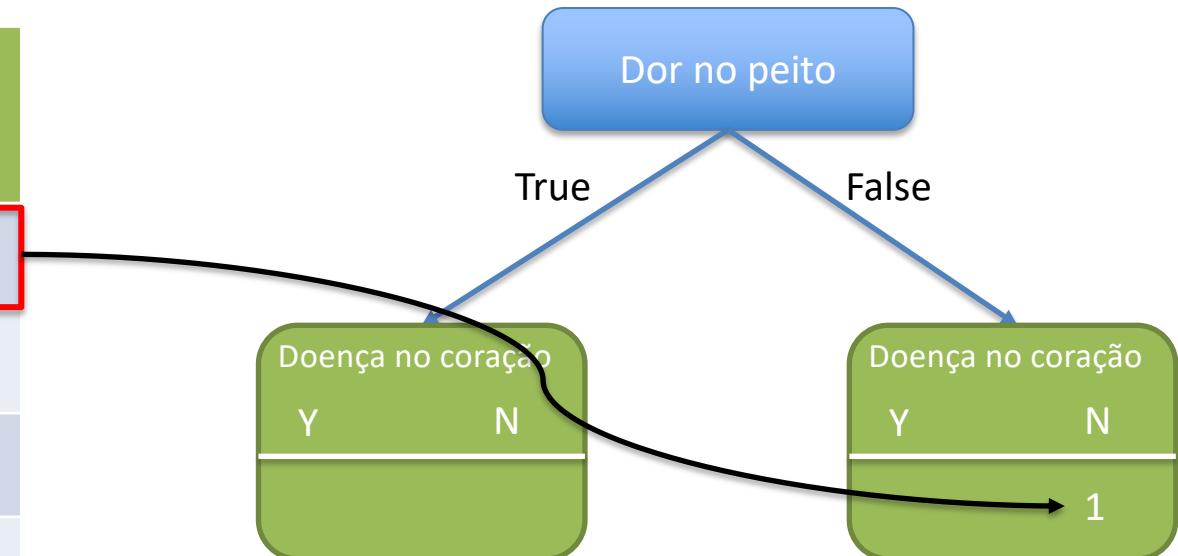
Vamos verificar como Dor no peito prediz doença cardíaca:

Dor no peito	Boa circulação Sanguínea	Artérias bloqueadas	Doença cardíaca
N	N	N	N
Y	Y	Y	Y
Y	Y	N	N
Y	N	?	Y
...	...	...	...



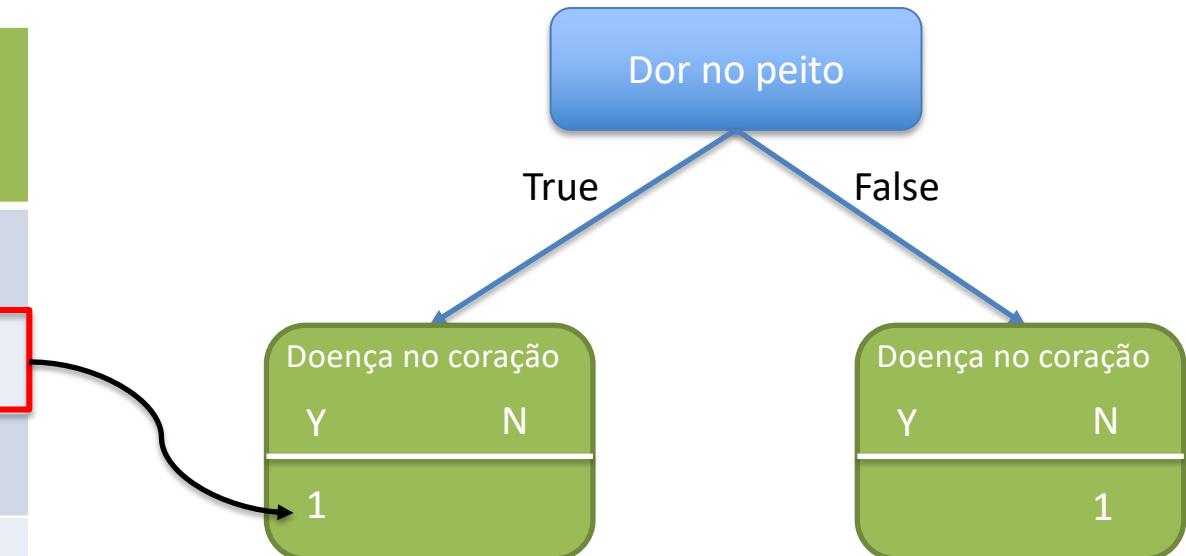
Agora contamos os valores para cada combinação:

Dor no peito	Boa circulação Sanguínea	Artérias bloqueadas	Doença cardíaca
N	N	N	N
Y	Y	Y	Y
Y	Y	N	N
Y	N	?	Y
...	...	...	...



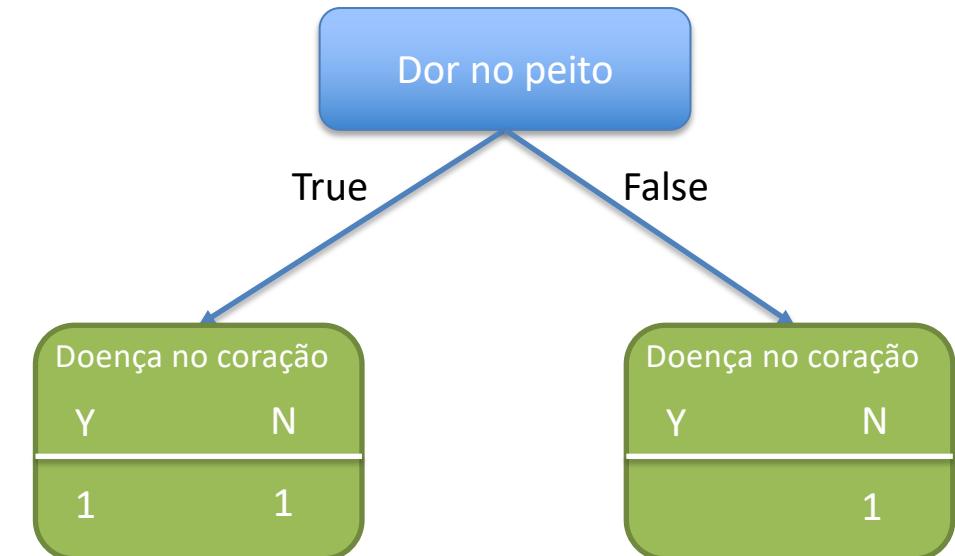
Agora contamos os valores para cada combinação:

Dor no peito	Boa circulação Sanguínea	Artérias bloqueadas	Doença cardíaca
N	N	N	N
Y	Y	Y	Y
Y	Y	N	N
Y	N	?	Y
...	...	...	...



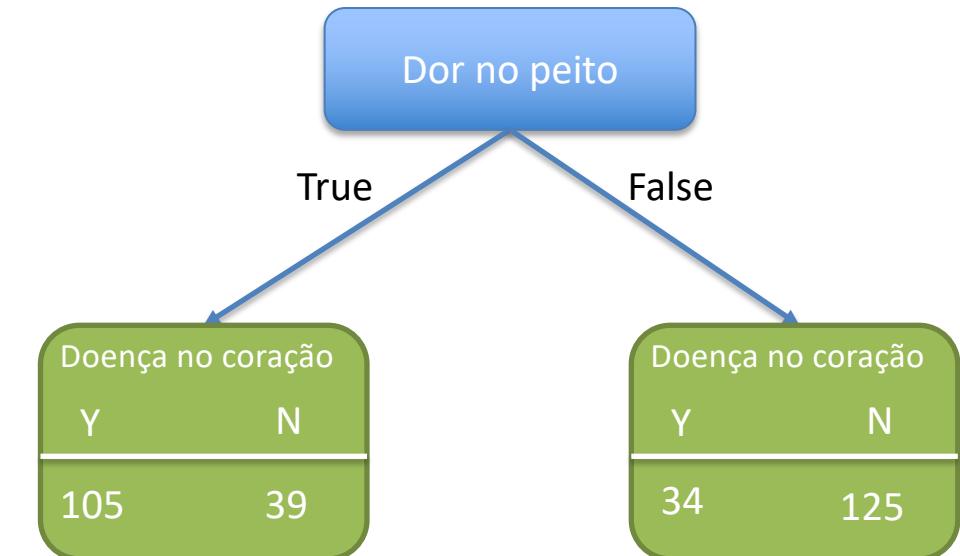
Agora contamos os valores para cada combinação:

Dor no peito	Boa circulação Sanguínea	Artérias bloqueadas	Doença cardíaca
N	N	N	N
Y	Y	Y	Y
Y	Y	N	N
Y	N	?	Y
...	...	...	...



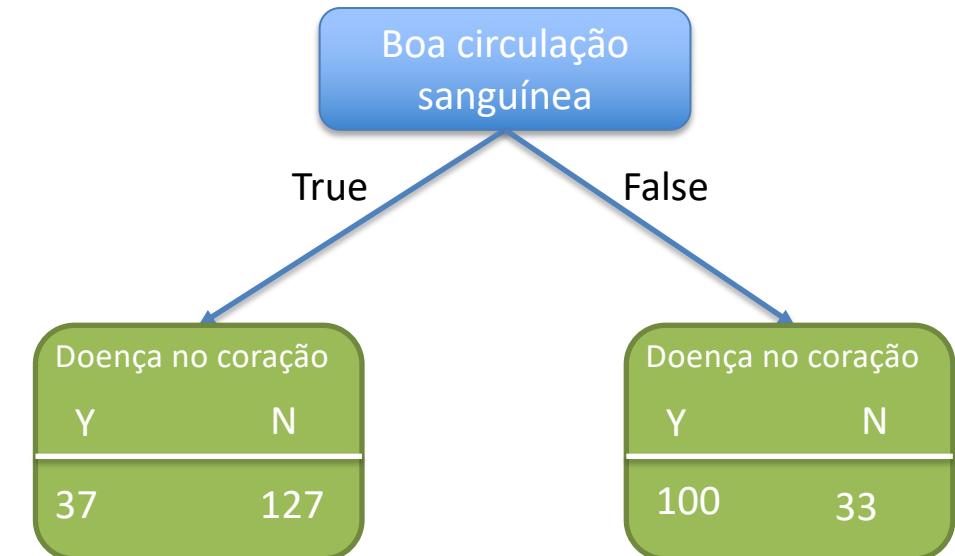
E calculamos até obter o valor relativo a todas as linhas do data set:

Dor no peito	Boa circulação Sanguínea	Artérias bloqueadas	Doença cardíaca
N	N	N	N
Y	Y	Y	Y
Y	Y	N	N
Y	N	?	Y
...	...	...	...



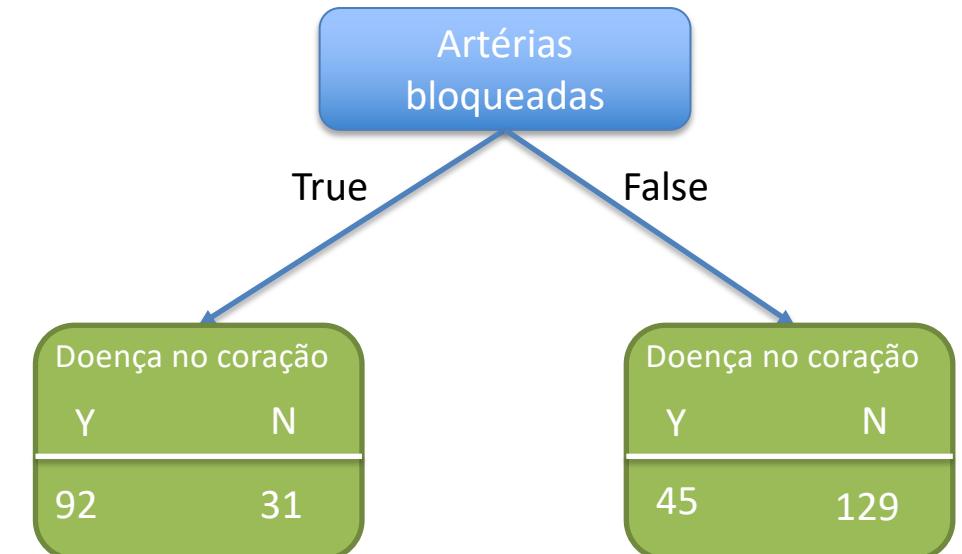
Agora repetimos o processo para Boa circulação sanguínea

Dor no peito	Boa circulação Sanguínea	Artérias bloqueadas	Doença cardíaca
N	N	N	N
Y	Y	Y	Y
Y	Y	N	N
Y	N	?	Y
...	...	...	...



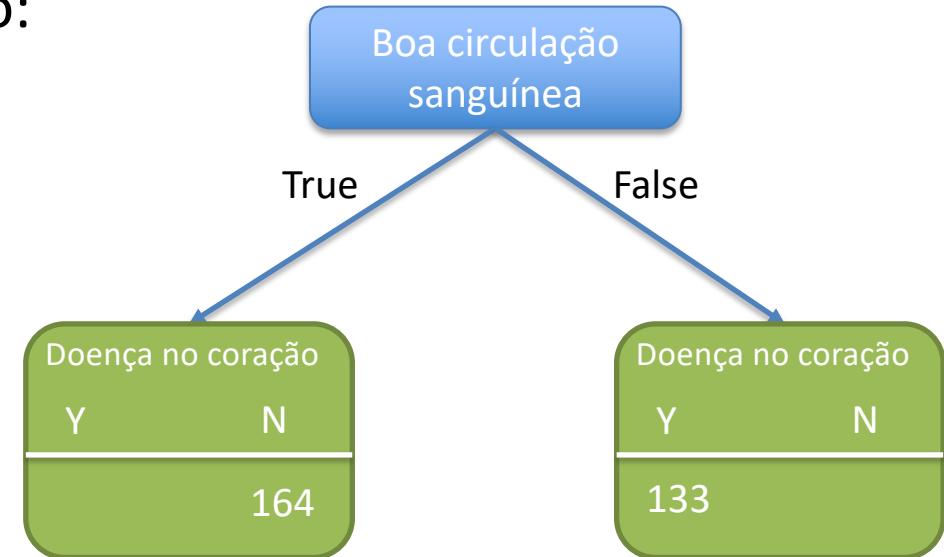
E para Artérias bloqueadas também:

Dor no peito	Boa circulação Sanguínea	Artérias bloqueadas	Doença cardíaca
N	N	N	N
Y	Y	Y	Y
Y	Y	N	N
Y	N	?	Y
...	...	...	...



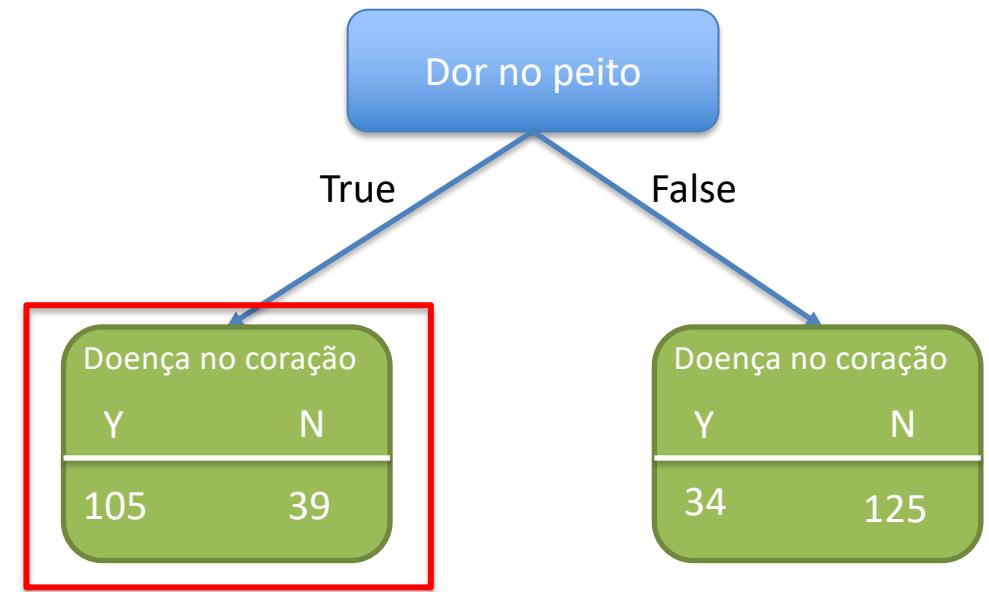
Importante notar que, como não sabemos o valor para esse paciente, não o levamos em consideração

Agora, precisamos determinar quanto bem cada feature conseguiu separar os pacientes com e sem doença cardíaca. Olhando os dados, vimos que nenhuma feature conseguiu uma separação perfeita. Exemplo:

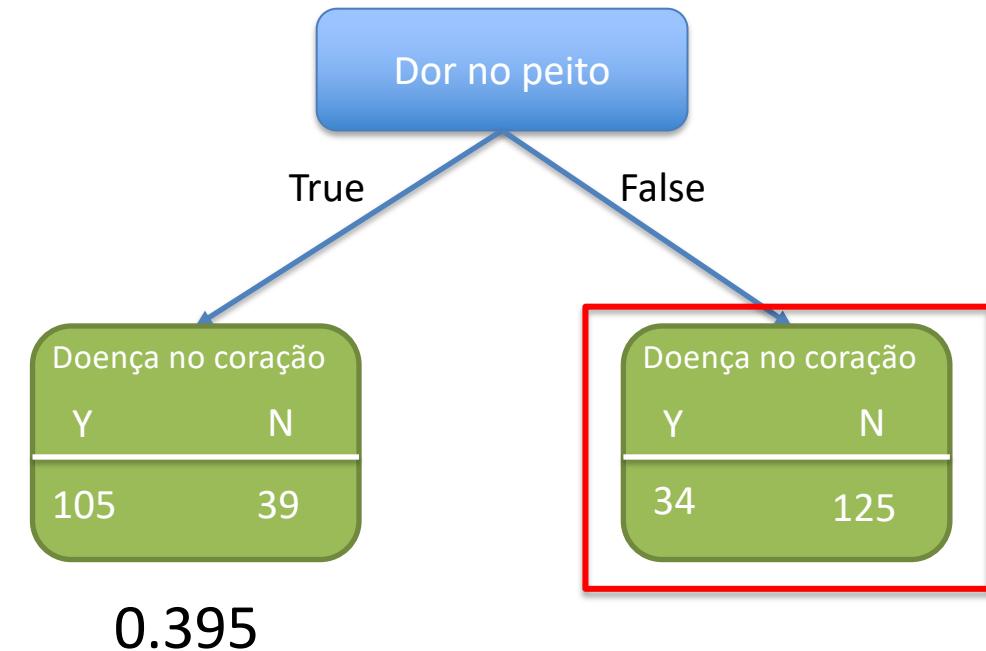


Isso significa que todas as features são **impuras**. Assim, para determinar qual feature deve ser o nó raiz, precisamos de uma maneira de mensurar e comparar **impureza**.

Aqui, vamos usar a métrica **Gini**:  $Gini = 1 - \sum_{i=1}^k p_i^2$



$$Gini = 1 - (p(Y)^2 - p(N)^2) = 1 - \left(\frac{105}{105 + 39}\right)^2 - \left(\frac{39}{105 + 39}\right)^2 = 0.395$$

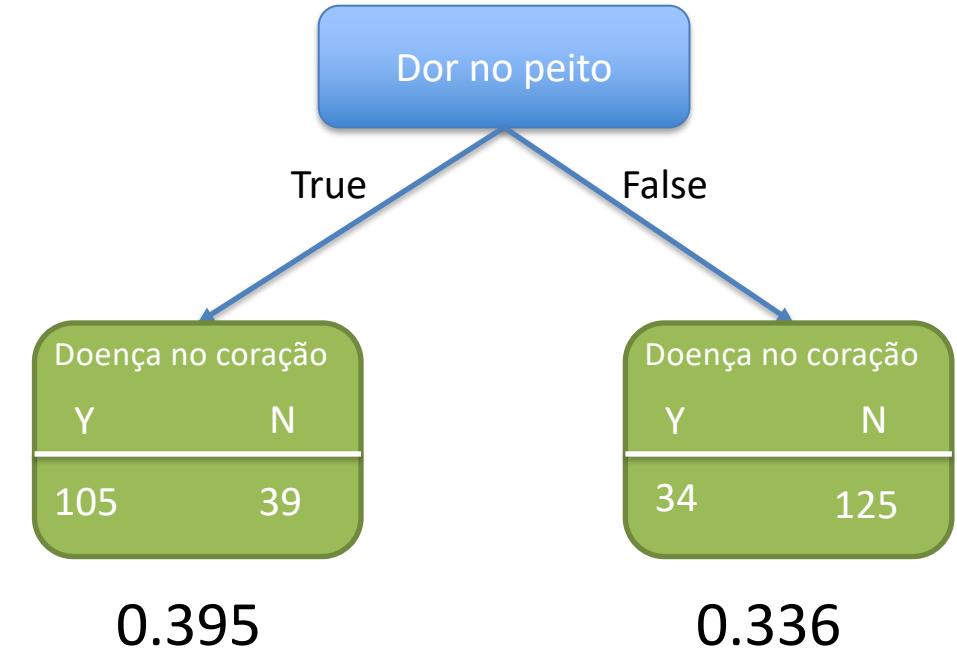


0.395

$$Gini = 1 - (p(Y)^2 - p(N)^2) = 1 - \left(\frac{34}{34 + 125}\right)^2 - \left(\frac{125}{34 + 125}\right)^2 = 0.336$$

Agora que calculamos o Gini para ambas as folhas, podemos calcular o Gini total do nó Dor no peito para separar pacientes com ou sem doença cardíaca.

Vamos usar a média ponderada pela quantidade de pacientes para determinar o Gini total



0.395

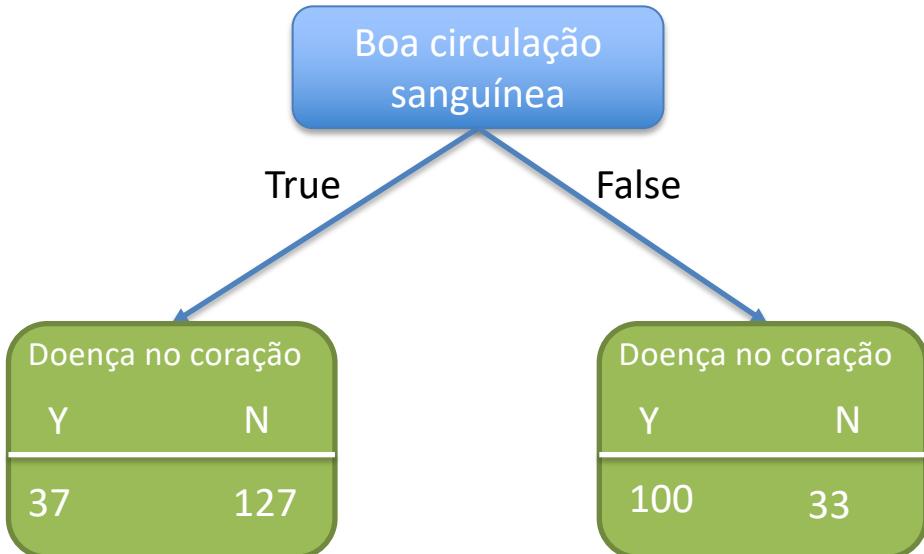
0.336

$$GiniTotal = \left( \frac{144}{144 + 159} \right) \times 0.395 + \left( \frac{159}{144 + 159} \right) \times 0.336 = 0.364$$

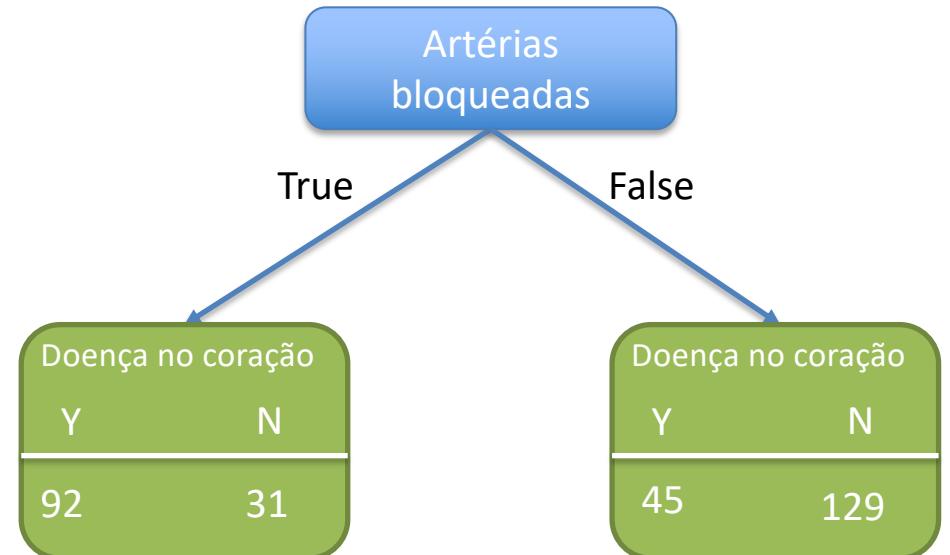
# Decision Trees

---

Repetimos o processo para as outras features:



$$Gini = 0.360$$



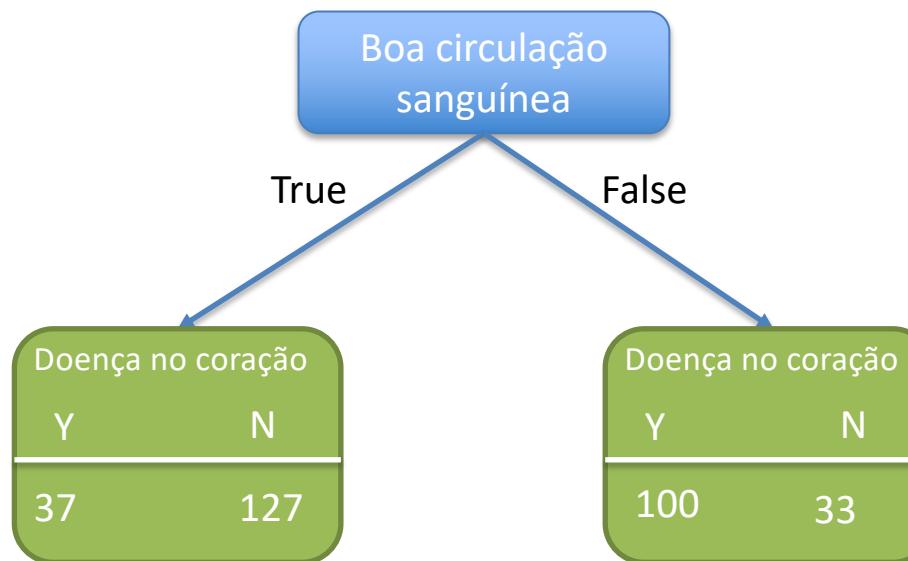
$$Gini = 0.381$$

# Decision Trees

---

Das métricas, conseguimos observar que Boa circulação sanguínea possui o menor valor de Gini, isto é, é a feature que melhor separa os pacientes entre aqueles que possuem doença cardíaca e aqueles que não possuem.

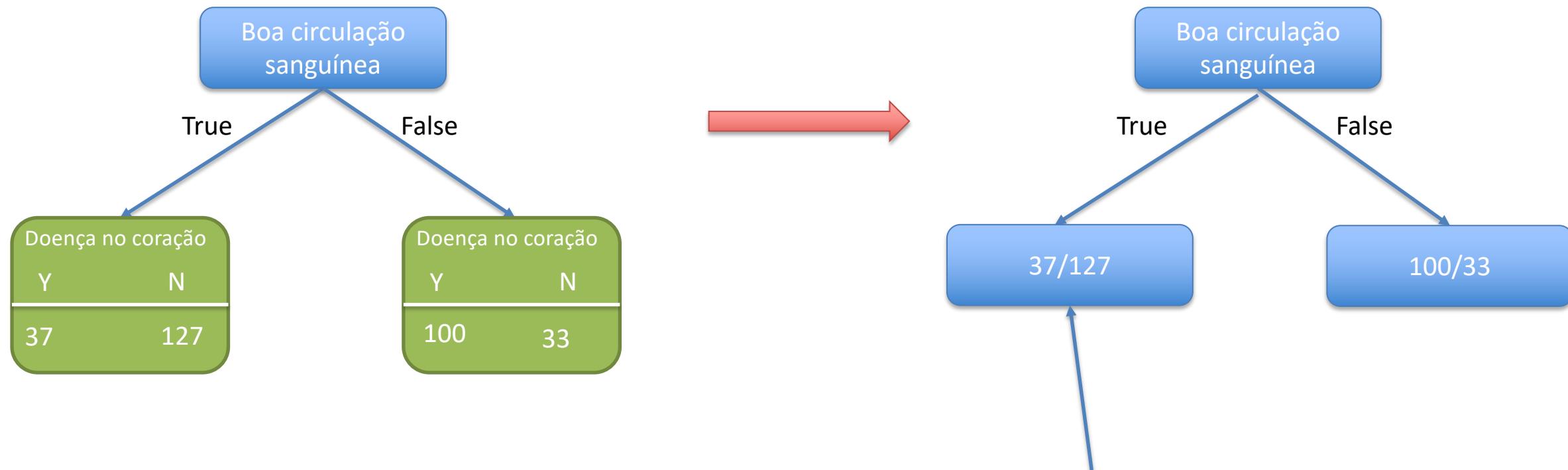
Portanto, ela sera usada como nó raiz:



# Decision Trees

---

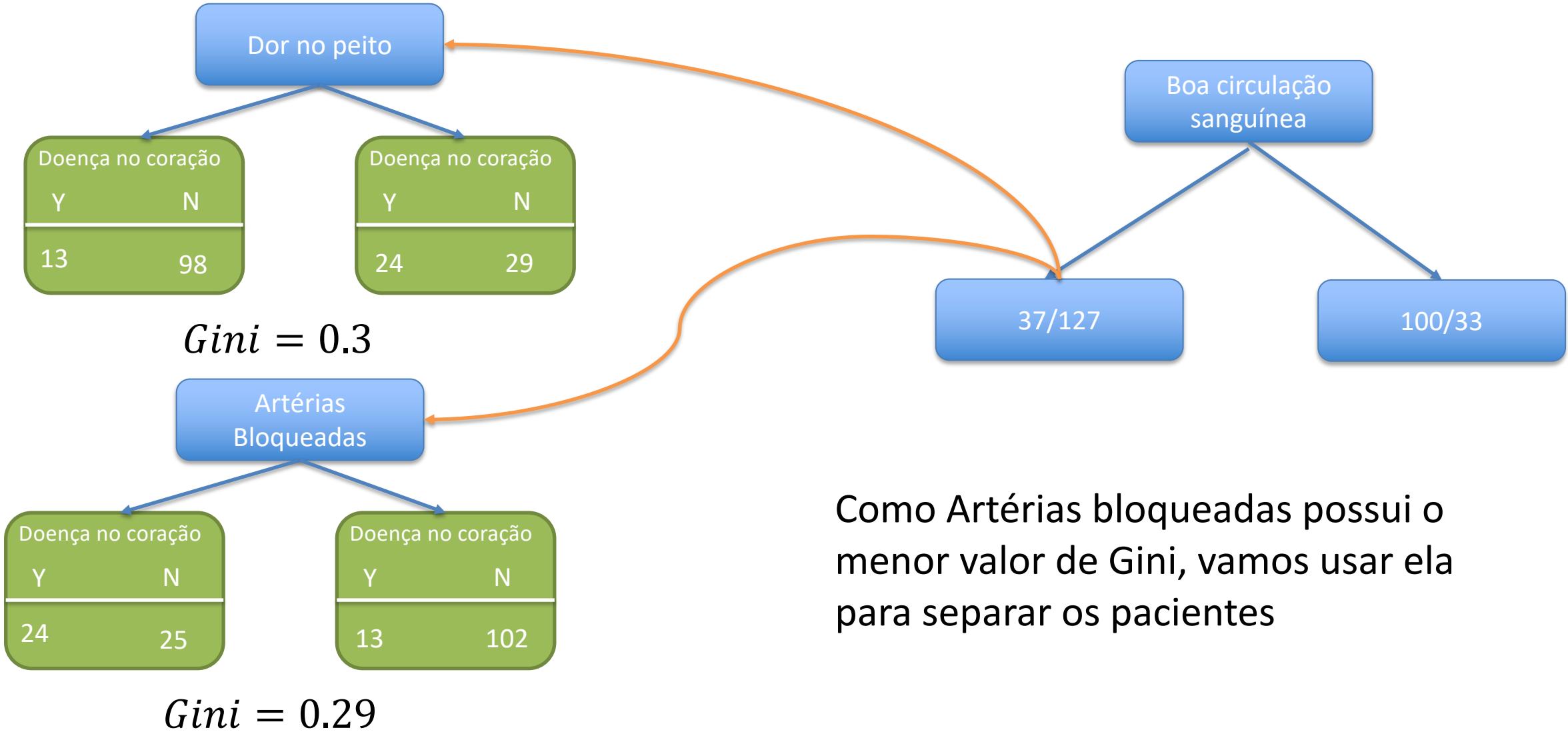
Importante notar que os nós folhas de Boa circulação sanguínea agora se comportarão como nós da árvore. Assim, temos que:



E agora precisamos descobrir como Dor no peito e Artérias bloqueadas separam esse nó.

# Decision Trees

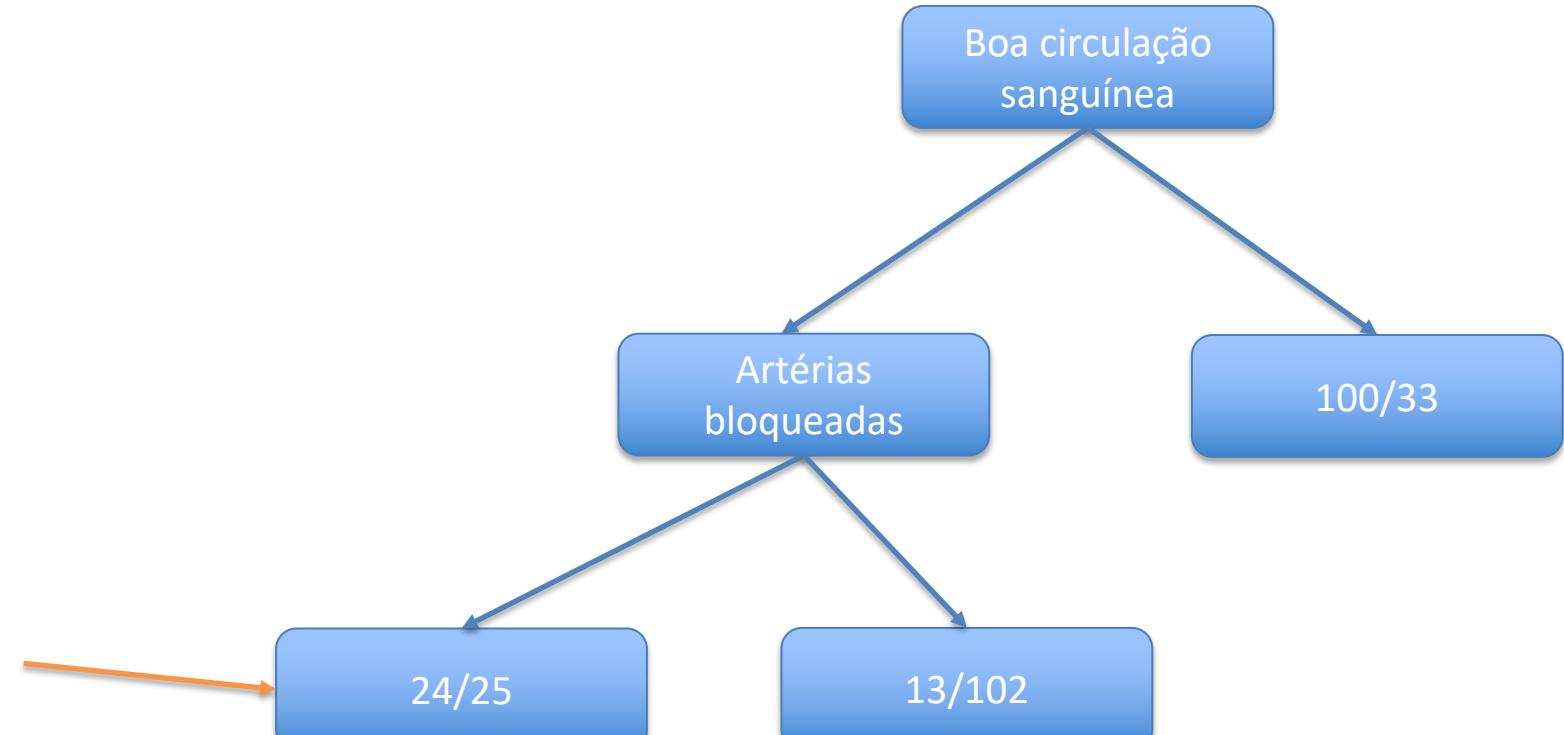
Da mesma forma que fizemos antes, separamos os pacientes baseados em Dor no peito e Artérias bloqueadas e calculamos o valor da impureza



## Decision Trees

---

Esta é a árvore que temos até agora:



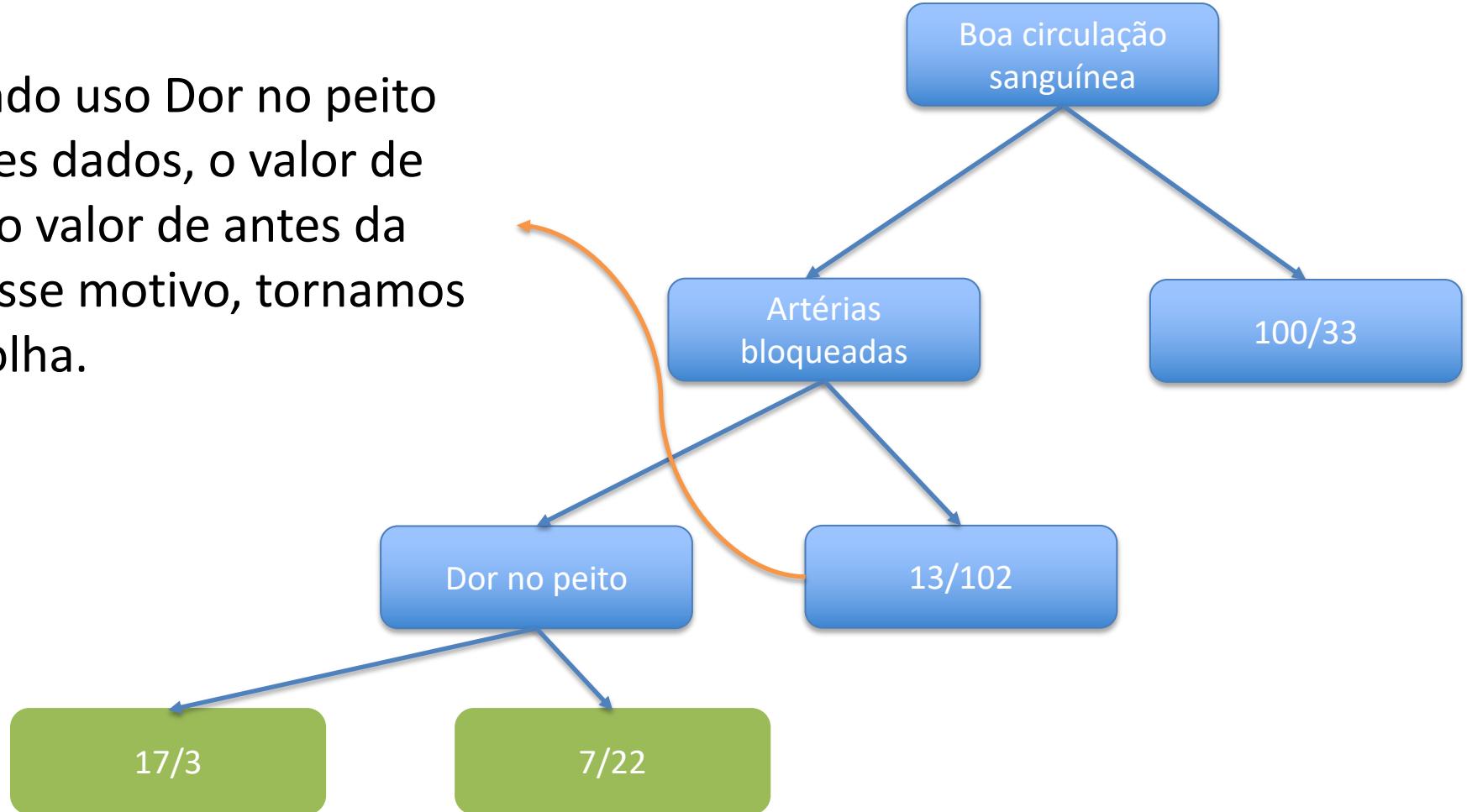
Tudo que restou é a feature Dor no peito, então vamos ver quão bem ela separa essas amostras.

# Decision Trees

---

A separação forneceu bons resultados.

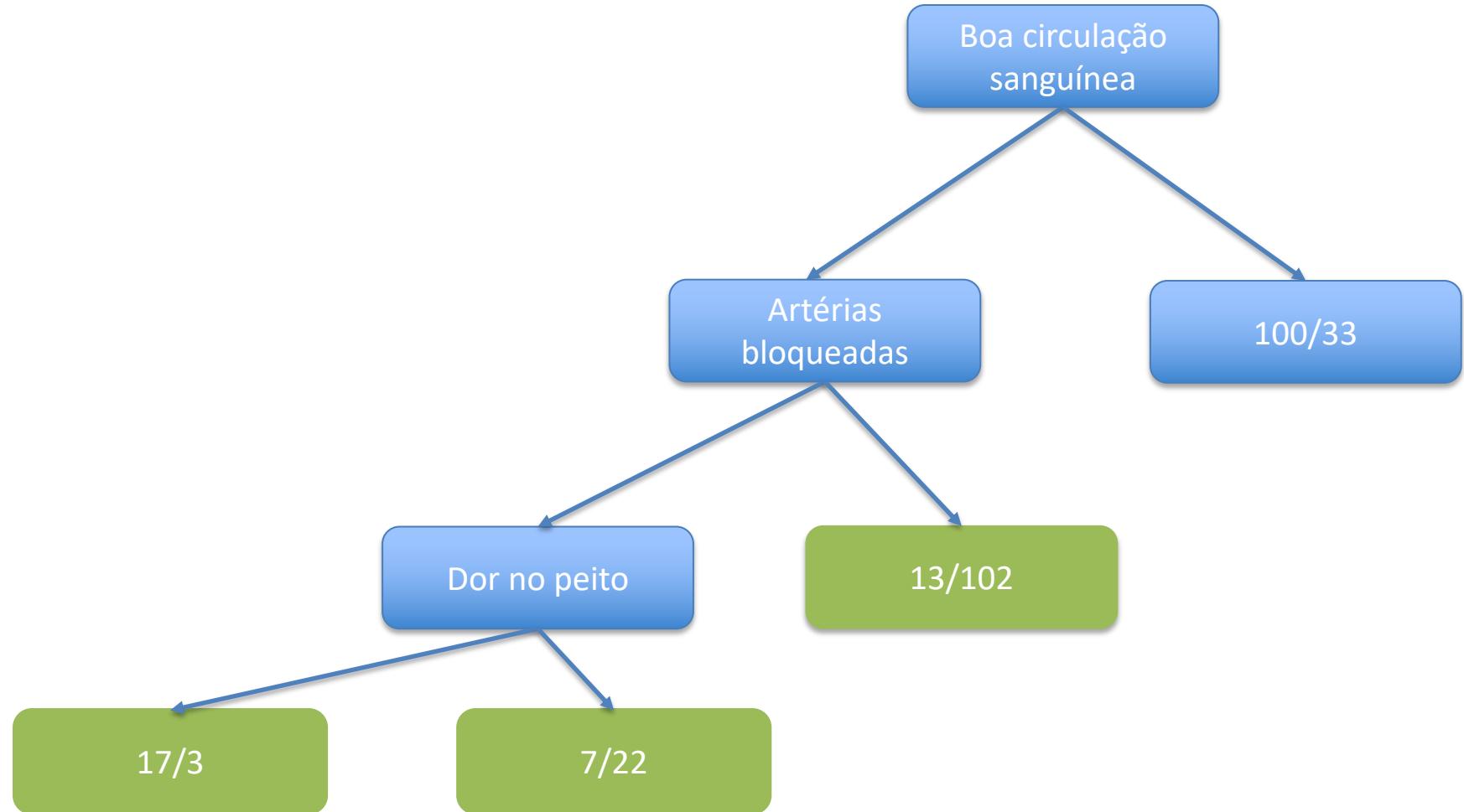
Entretanto, quando uso Dor no peito para separar esses dados, o valor de Gini é superior ao valor de antes da separação. Por esse motivo, tornamos esse nó um nó folha.



# Decision Trees

---

A separação forneceu bons resultados.

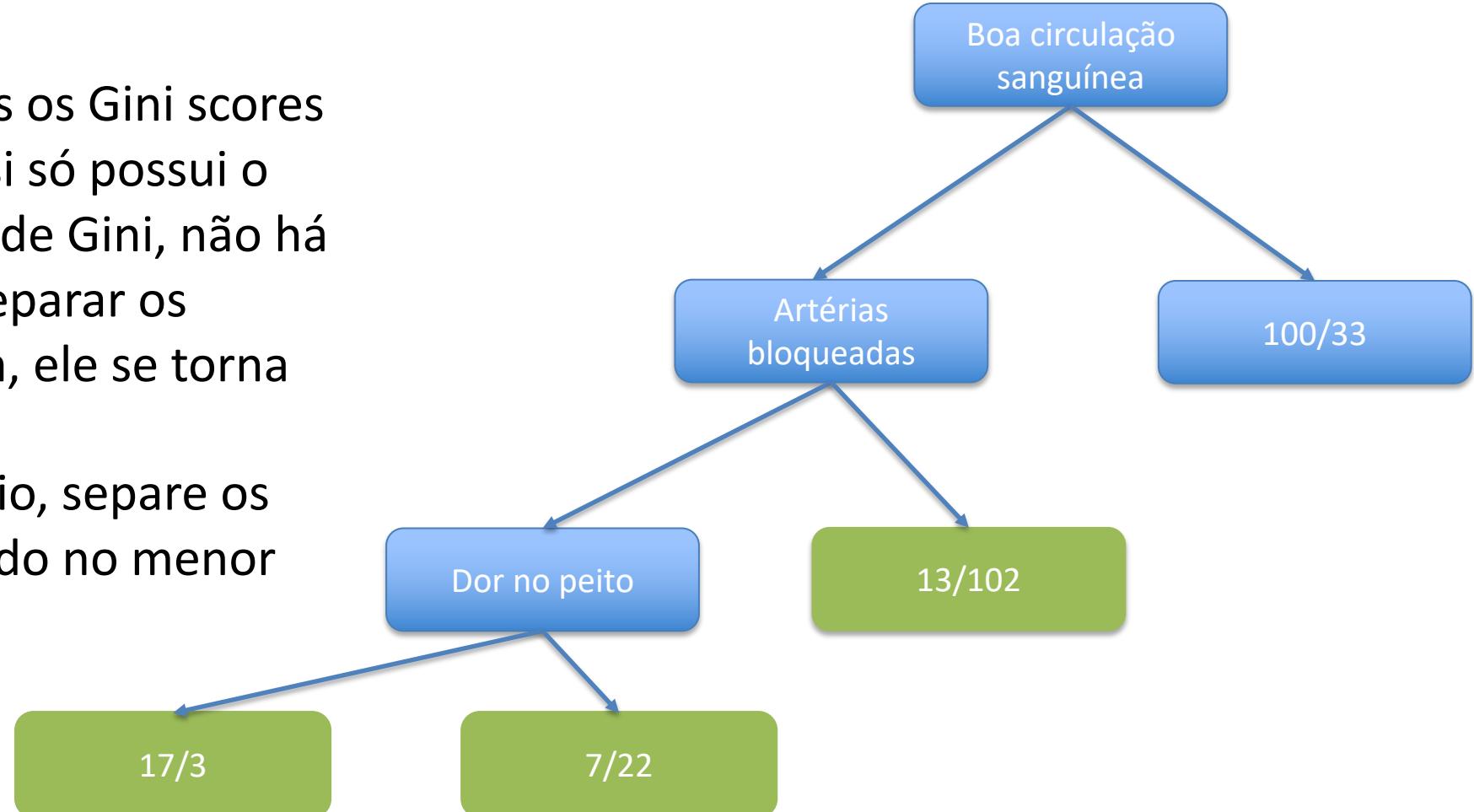


## Decision Trees

---

Agora, precisamos trabalhar no lado direito da árvore. Para isso, só repetimos os passos vistos até aqui:

1. Calcule todos os Gini scores
2. Se o nó por si só possui o menor valor de Gini, não há razão para separar os dados. Assim, ele se torna um nó folha
3. Caso contrário, separe os dados baseado no menor valor de Gini



# Decision Trees

---

E obteremos essa árvore



## Decision Trees

---

Uma importante etapa nas árvores de decisão é o ajuste nos hiperparâmetros.

Antes, porém, é importante fazer uma distinção entre parâmetros e hiperparâmetros:

1. **Parâmetros**: são ajustados em tempo de execução do modelo, mais especificamente [na etapa de treino](#).
2. **Hiperparâmetros**: são ajustados [antes da etapa de treino](#).  
Obviamente, se não valor for selecionado, os valores padrões serão usados.

Aqui, vamos focar em encontrar os melhores hiperparametros.

## Decision Trees

---

O primeiro hiperparâmetros que vamos entender é o *max\_depth*, que define a profundidade de uma árvore. Por padrão, não há limite para *max\_depth*, então podem existir milhões de splits numa árvore, resultando em overfitting. Limitando *max\_depth* a pequenos números, a variância é reduzida e o modelo generaliza melhor para novos dados.

O segundo hiperparâmetro é o *min\_samples\_leaf*, que provê uma restrição aumentando o número de amostras que uma folha pode ter. Quando não há, restrição, *min\_samples\_leaf=1*, o que significa que os nós folhas podem conter apenas uma amostra (propenso a overfitting). Quando aumentamos *min\_samples\_leaf*, reduzimos variância.

Existem vários outros hiperparâmetros a serem analisados e a documentação da scikit-learn provê detalhes de cada um deles.