Sensores espectroscópicos e modelos de regressão aplicados na análise de solos

Aula 3 – Métodos baseados em Ensemble

Me. José Vinícius Ribeiro

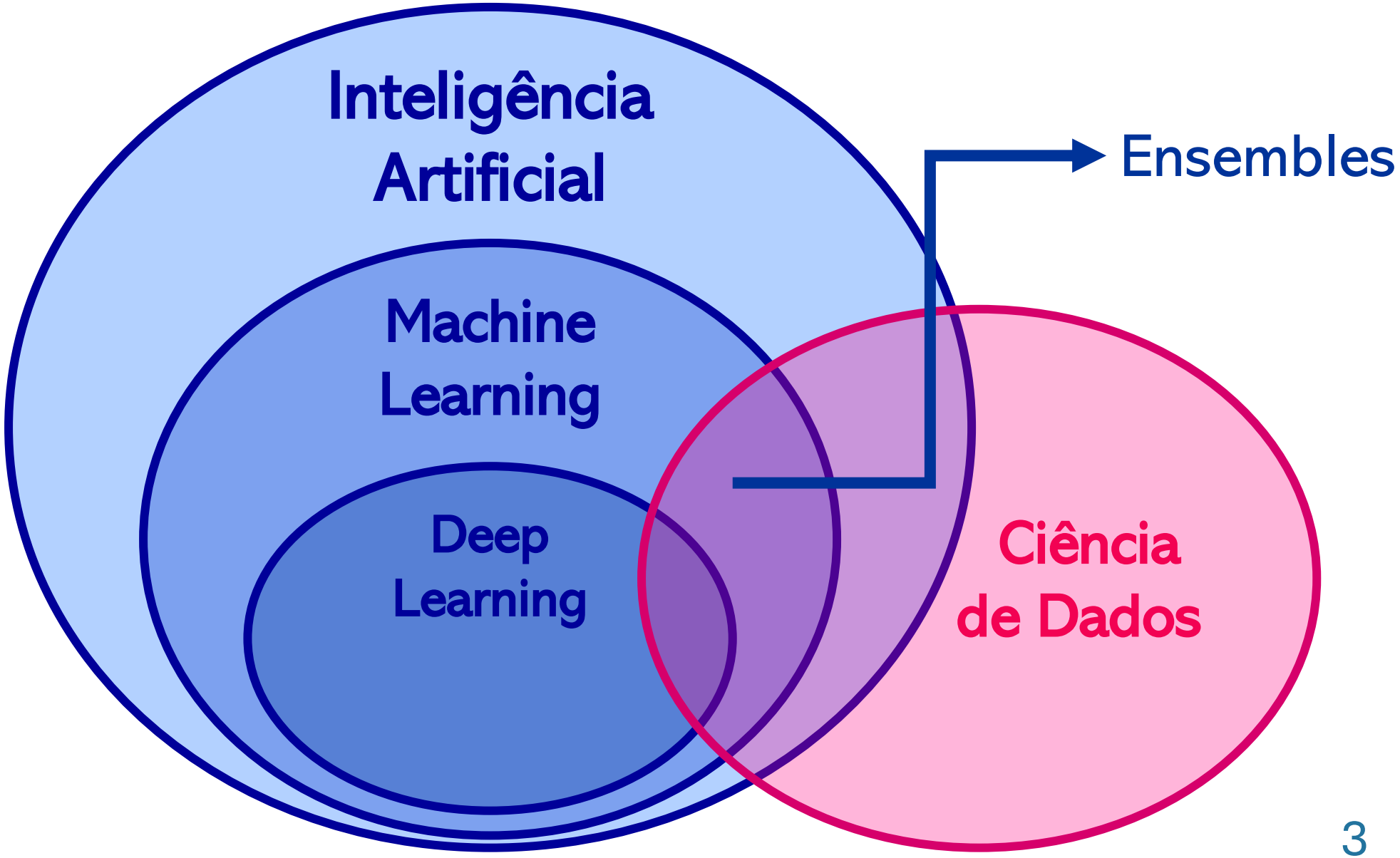
Universidade Estadual de Londrina

PÓS GRADUAÇÃO FÍSICA UEL

SUMÁRIO

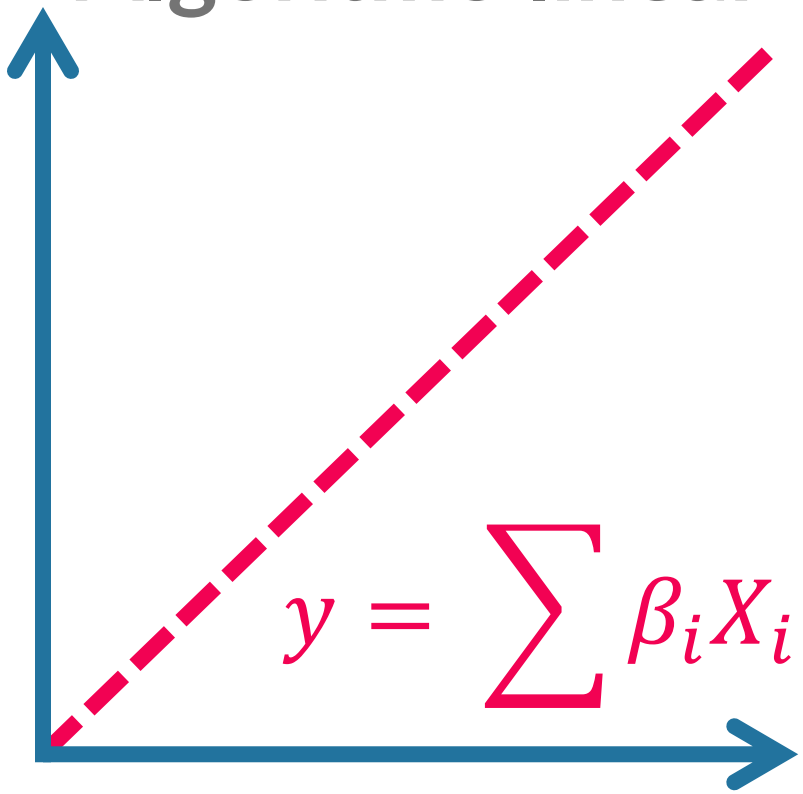
- Linearidade x Não-linearidade
- Conceito de ensemble
- Regressão por árvore de decisão
- Random Forest
- *Stacked generalization*
- Explicabilidade
- Prática no python (vscode)

Vamos nos situar...



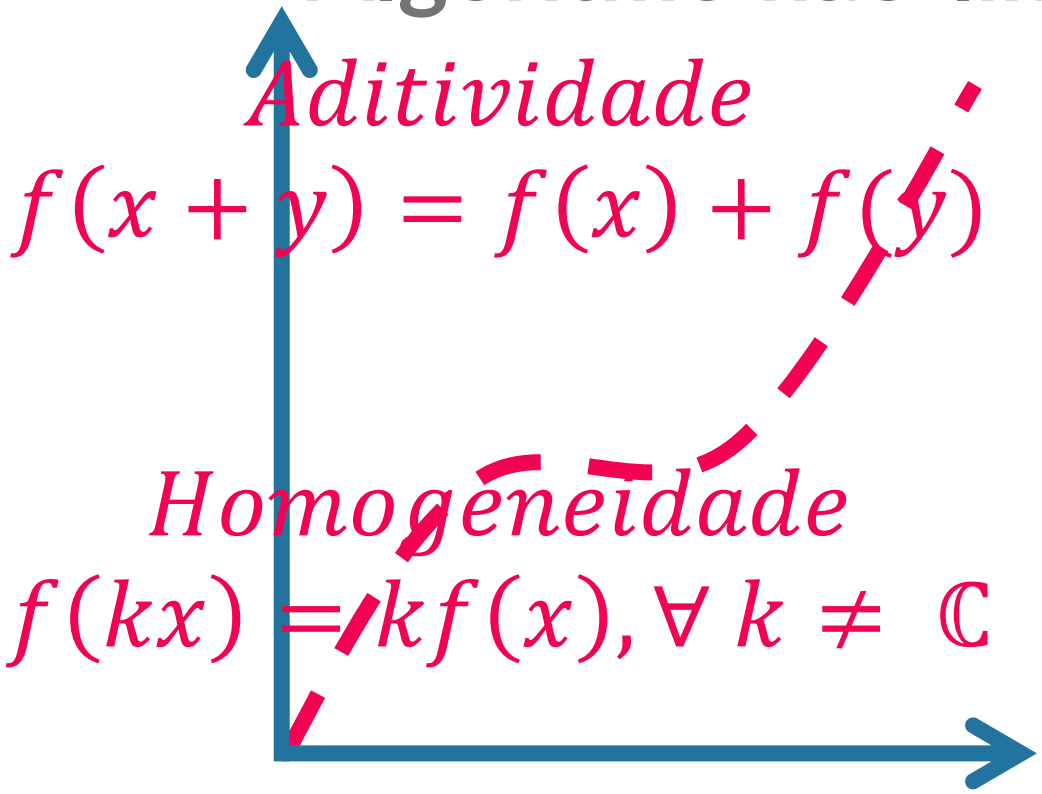
LINEARIDADE X NÃO-LINEARIDADE

Algoritmo linear



A relação entre as variáveis (matriz X) e o target (vetor y) pode ser expressa como uma combinação linear de variáveis. Apenas operações lineares.

Algoritmo não-linear



Pressuposto de que a relação entre o target e as variáveis é mais complexa. Muitas possibilidades de novas operações

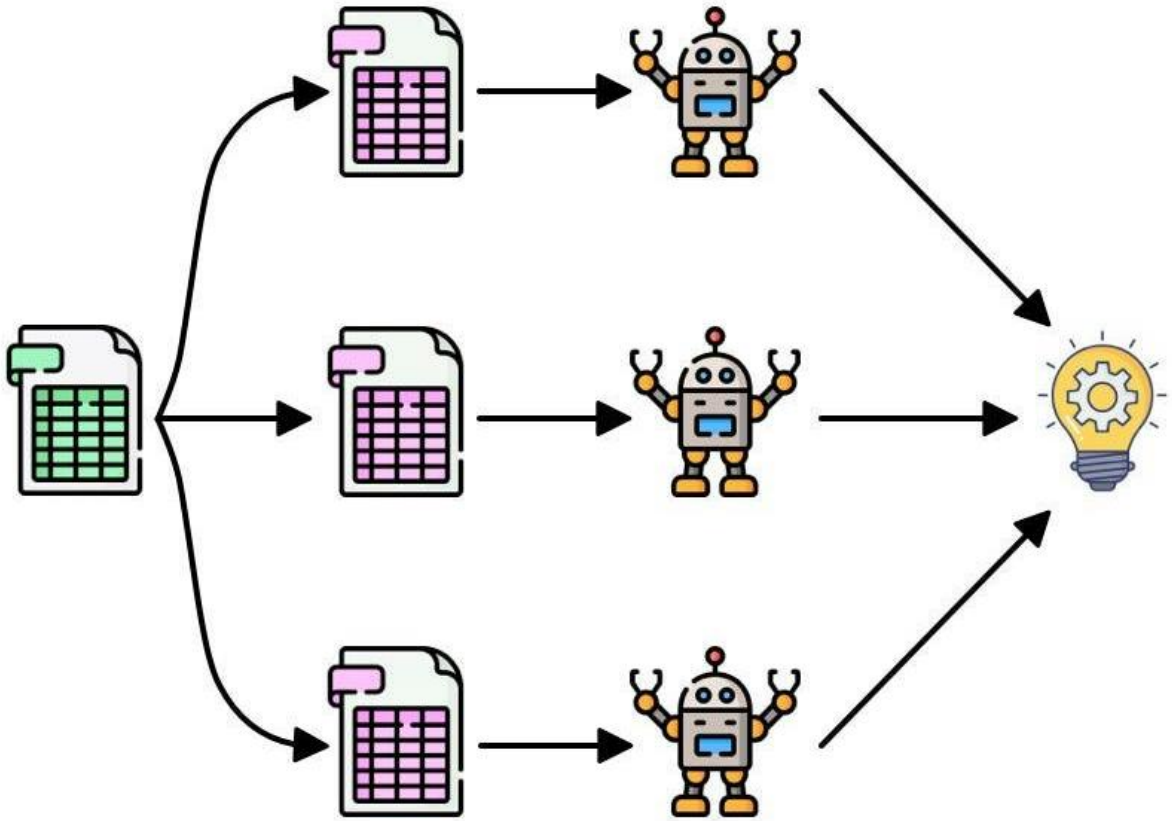
Ensembles

Ensemble Learning, também chamado de aprendizado por agrupamento, se baseia na ideia de combinar diversos modelos mais simples (*weak learner*), treiná-los para uma mesma tarefa, e produzir a partir desses um modelo agrupado mais complexo (*strong learner*)

Essa ideia produz modelos mais robustos e menos suscetíveis ao *bias*

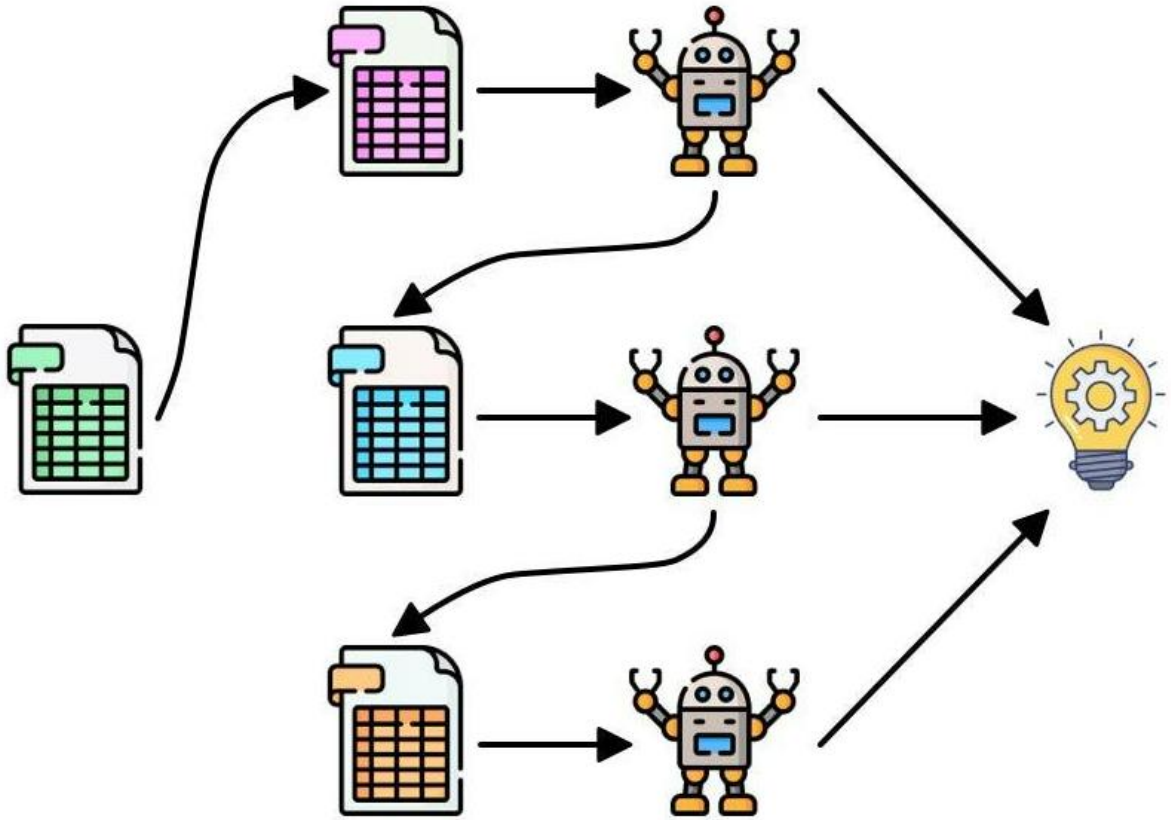
Tipos de ensemble

Métodos paralelos: treinam cada modelo base separadamente/paralelamente dos outros. Os modelos não compartilham informações



Tipos de ensemble

Métodos sequenciais: processo iterativo onde cada novo modelo busca minimizar os erros do modelo etapa anterior. Vão sempre na direção de maior complexidade dos dados.



PRINCIPAIS ALGORITMOS ATUALMENTE

Ensemble paralelo

- Árvore de decisão
- Random Forest
- Cubist
- Stacked-modeling
- Bagged k-Nearest Neighbors (Bagged KNN)

Ensemble sequencial

- *XGBoost*
- *Gradient-boosting model (GBM)*
- *AdaBoost*
- LightGBM

Ensemble paralelo

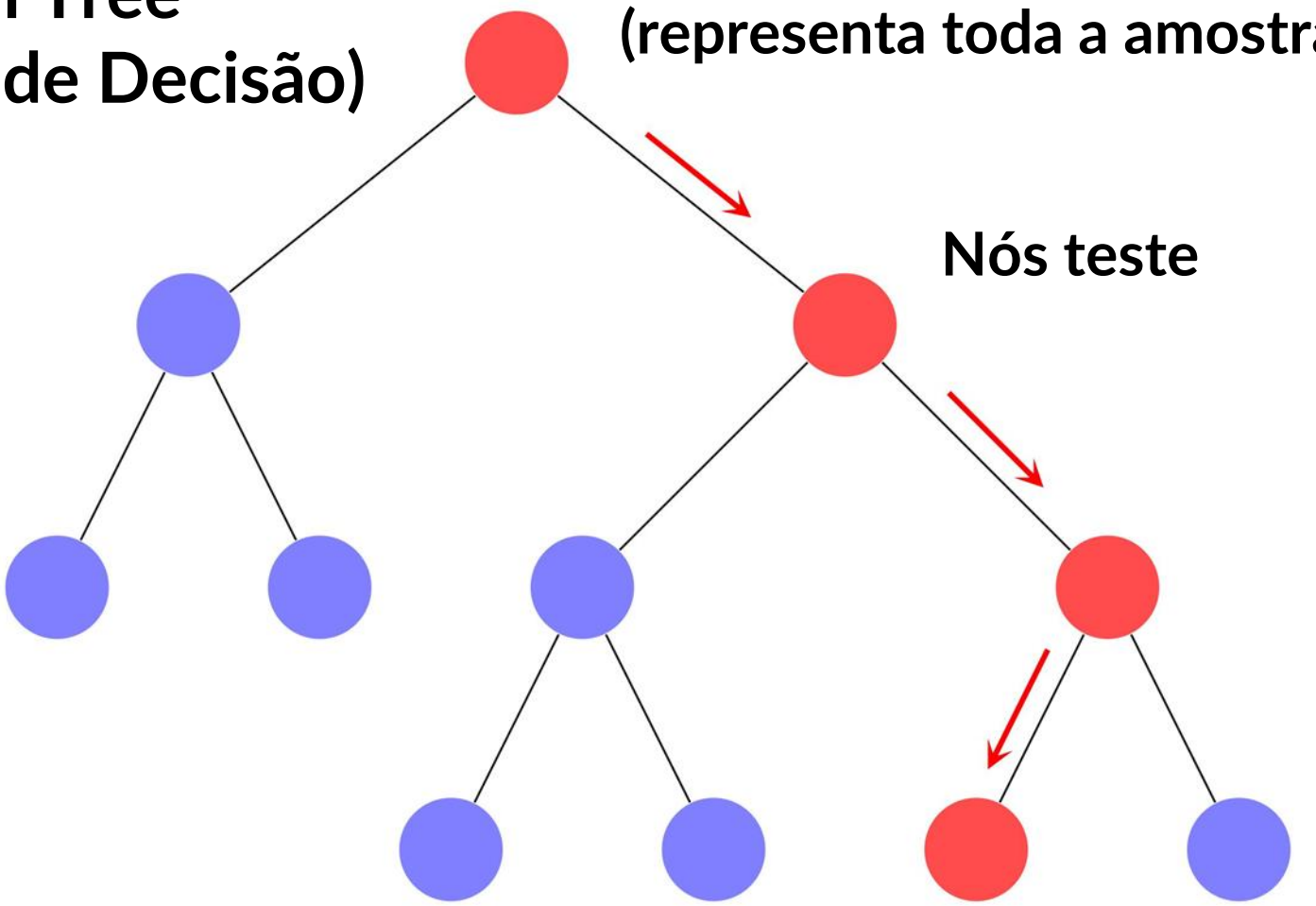
Vamos focar nos modelos treinados paralelamente e baseados em *bagging* **(também chamado de *bootstrap*)**

O *Bagging* é uma estratégia de re-amostragem que gera sub-conjuntos diversos aleatoriamente escolhidos provenientes de um dado conjunto inicial (nesse caso os dados de treinamento)

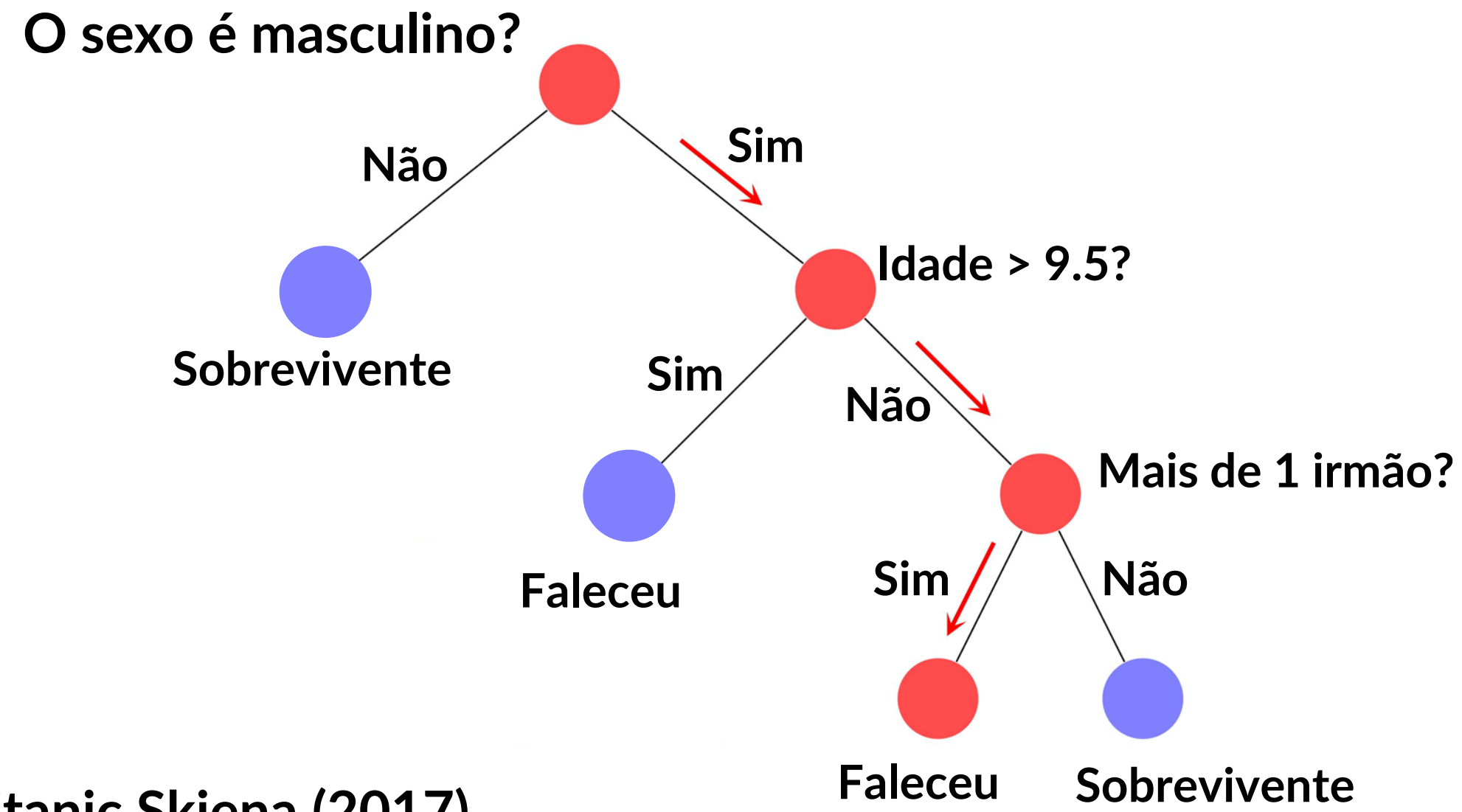
ÁRVORES DE DECISÃO

Árvores de decisão

Decision Tree
(Árvore de Decisão)



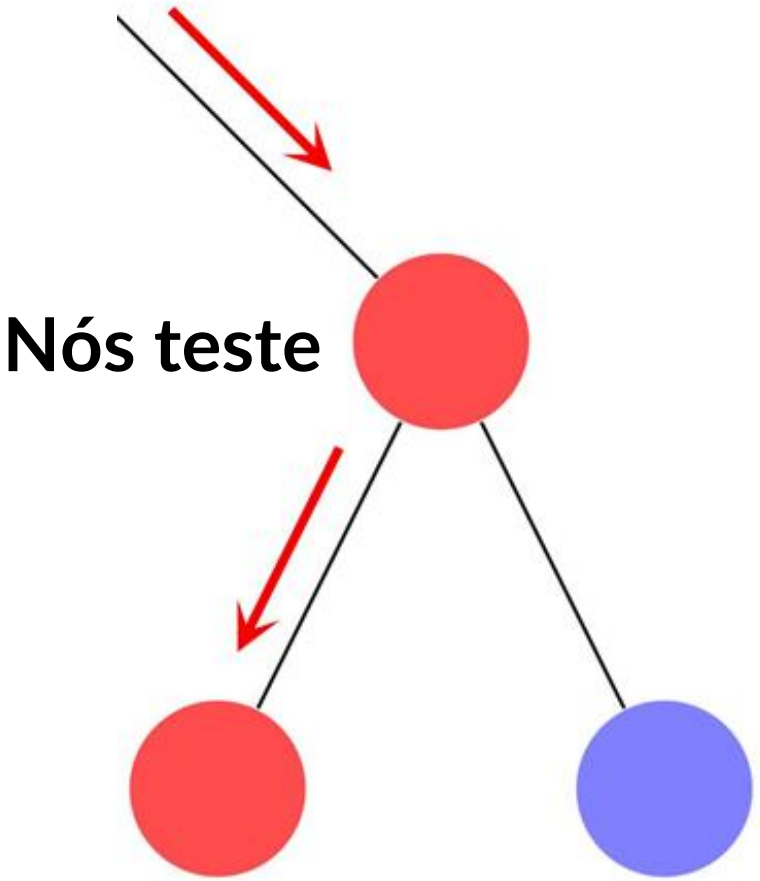
Árvores de decisão



Exemplo Titanic Skiena (2017)

SKIENA, Steven S. The data science design manual. Springer, 2017

Árvores de decisão



Portanto, existem *variaveis* que carregam mais informação (facilitam a divisão de classes) do que outras

Parâmetros de pureza para identificá-las

- **Entropia, Índice de Gini, Erros Quadráticos Médios, Erros Absolutos Médios**

Exemplo numérico: modelo de regressão

$X_{ij} = (X_{11}, X_{12}, \ldots, X_{1n}, X_{21}, X_{22}, \ldots, X_{2n}, \ldots X_{nm})$

n= amostras

$y_i = (y_1, y_2, y_3, \ldots, y_n)$

m = variáveis

Suponha a configuração para o modelo: n=5 e m=1

Uma possivel *decision tree* seria

X=[1, 2, 3, 4, 5] e y=[1.2, 1.9, 3.1, 4.2, 5.0]

Árvores de decisão

X=[1, 2, 3, 4, 5] e y=[1.2, 1.9, 3.1, 4.2, 5.0]

Ponto de divisão X=2.5

Métrica de impureza → $MSE = \sum_{i=1}^{p} \frac{(y_i - \bar{y}_i)^2}{p}$

- $X < 2.5$

y=[1.2, 1.9] $MSE = \frac{(1.2 - 1.55)^2 + (1.9 - 1.55)^2}{2}$

$\bar{y}_i$=1.55

$= 0.1225$

Árvores de decisão

X=[1, 2, 3, 4, 5] e y=[1.2, 1.9, 3.1, 4.2, 5.0]

- X >2.5 y=[3.1, 4.2, 5.0] $\bar{y}_i$=4.1

$$MSE = \frac{(3.1 - 4.1)^2 + (4.2 - 4.1)^2 + (5.0 - 4.1)^2}{3}$$

$$= 0.6067$$

Árvores de decisão

X=[1, 2, 3, 4, 5] e y=[1.2, 1.9, 3.1, 4.2, 5.0]

- MSE geral para a divisão em X=2.5

$$MSE_{total} = \frac{2}{5}\,0.1225 + \frac{3}{5}\,0.6067$$

$$= 0.41302$$

Árvores de decisão

X=[1, 2, 3, 4, 5] e y=[1.2, 1.9, 3.1, 4.2, 5.0]

Novo ponto de divisão X= 3.5

X<3.5 y=[1.2, 1.9, 3.1] $\bar{y}_i$=2.067

$$MSE = 0.616$$

X>3.5 y=[4.2, 5.0] $\bar{y}_i$=4.6

$$MSE = 0.16$$

Árvores de decisão

X=[1, 2, 3, 4, 5] e y=[1.2, 1.9, 3.1, 4.2, 5.0]

Novo ponto de divisão X= 3.5

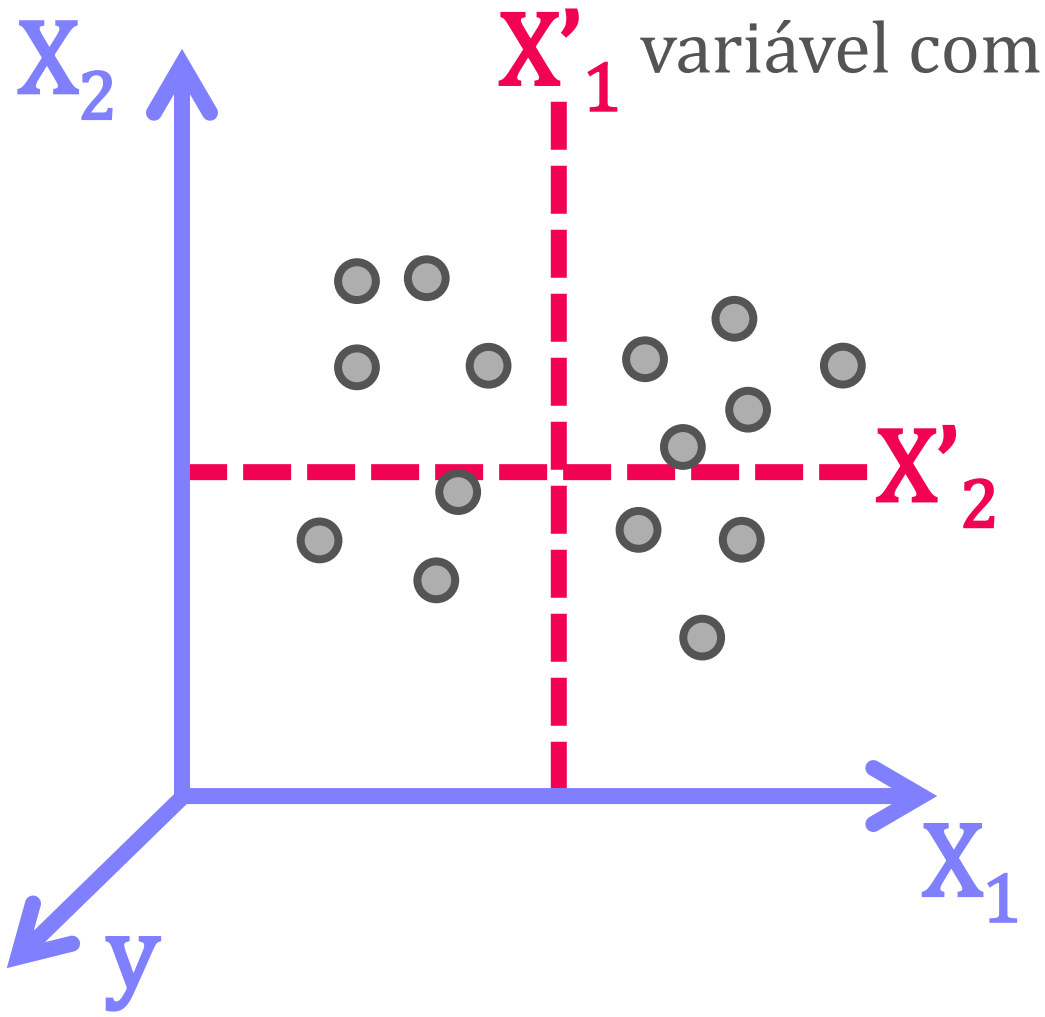
$$MSE_{total} = \frac{3}{5}\,0.616 + \frac{2}{5}\,0.16$$

$$= 0.4336 > 0.4130$$

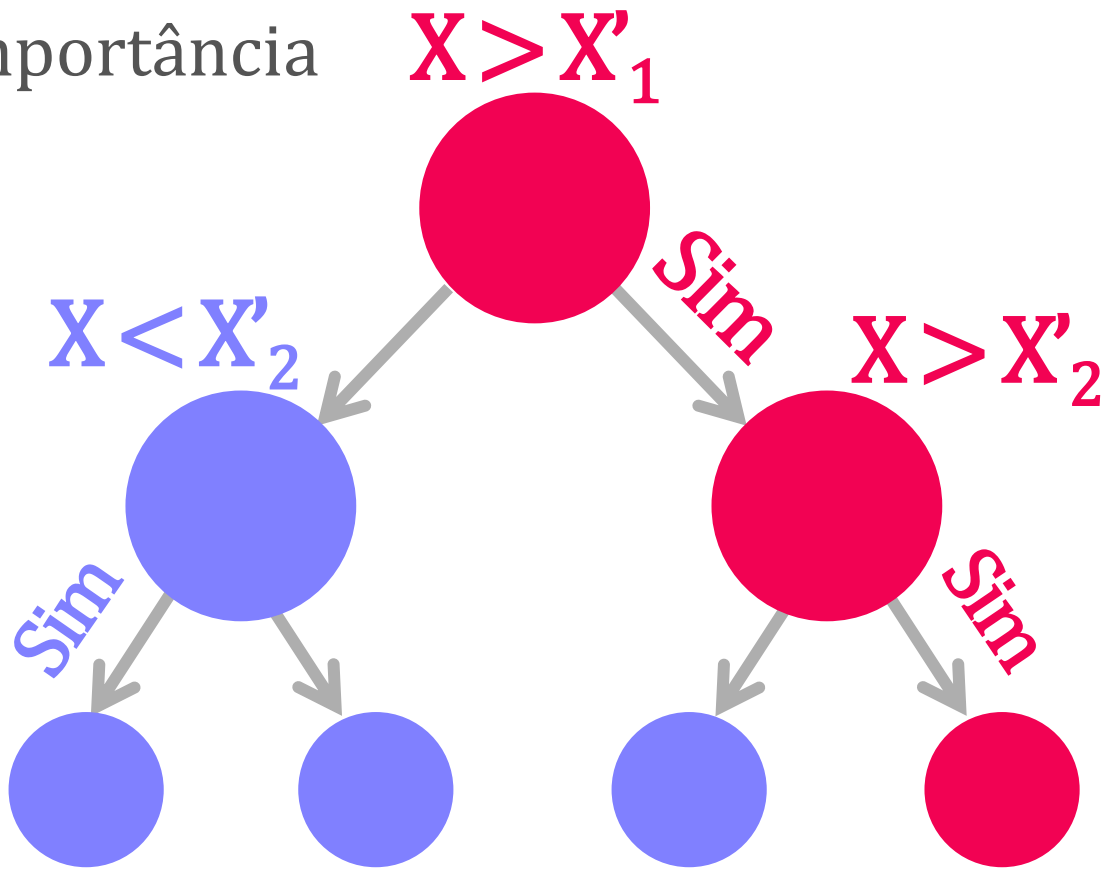
Qual o melhor ponto para dividir os dados? Ou em outras palavras, ser utilizado como o primeiro nó da arvore?

Árvores de decisão

Qual o melhor ponto para dividir os dados? Ou em outras palavras, ser utilizado como o primeiro nó da arvore?



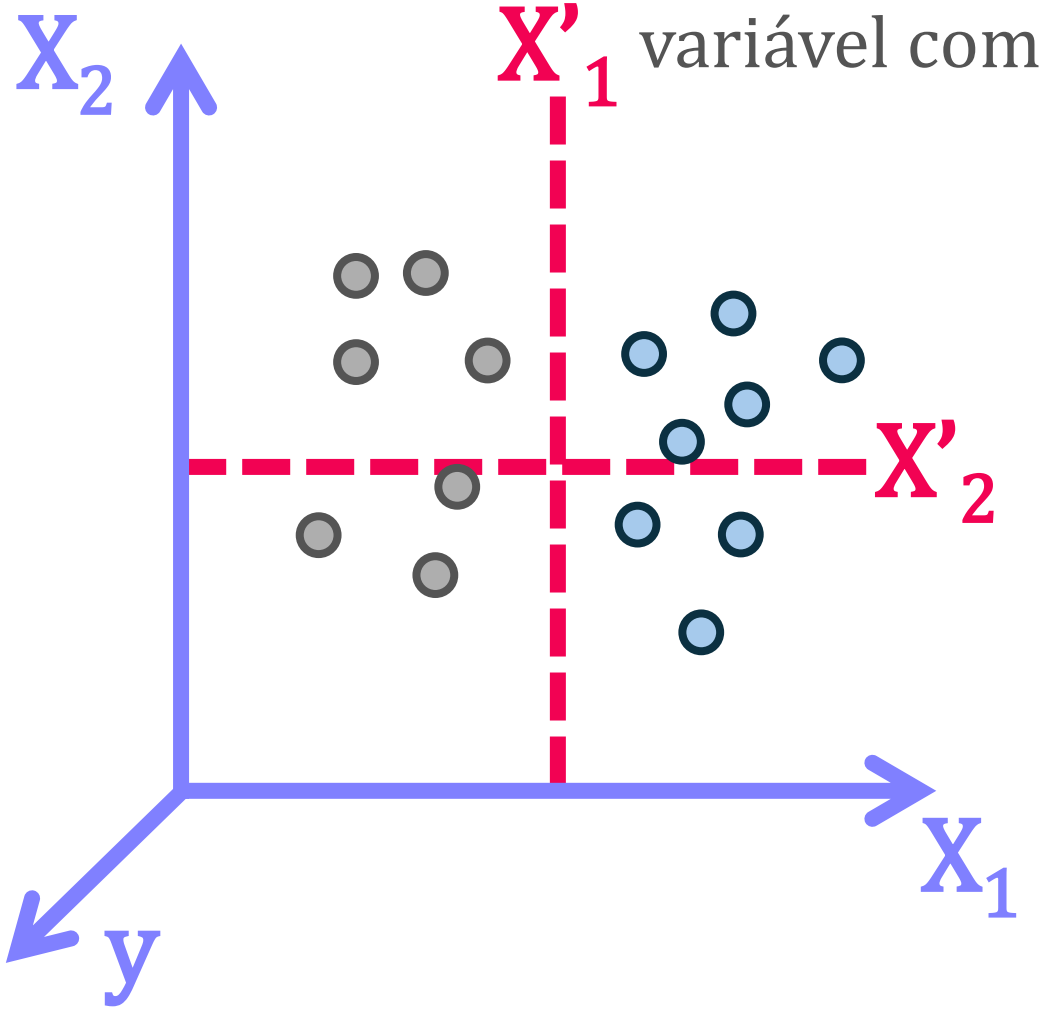
variável com alta importância



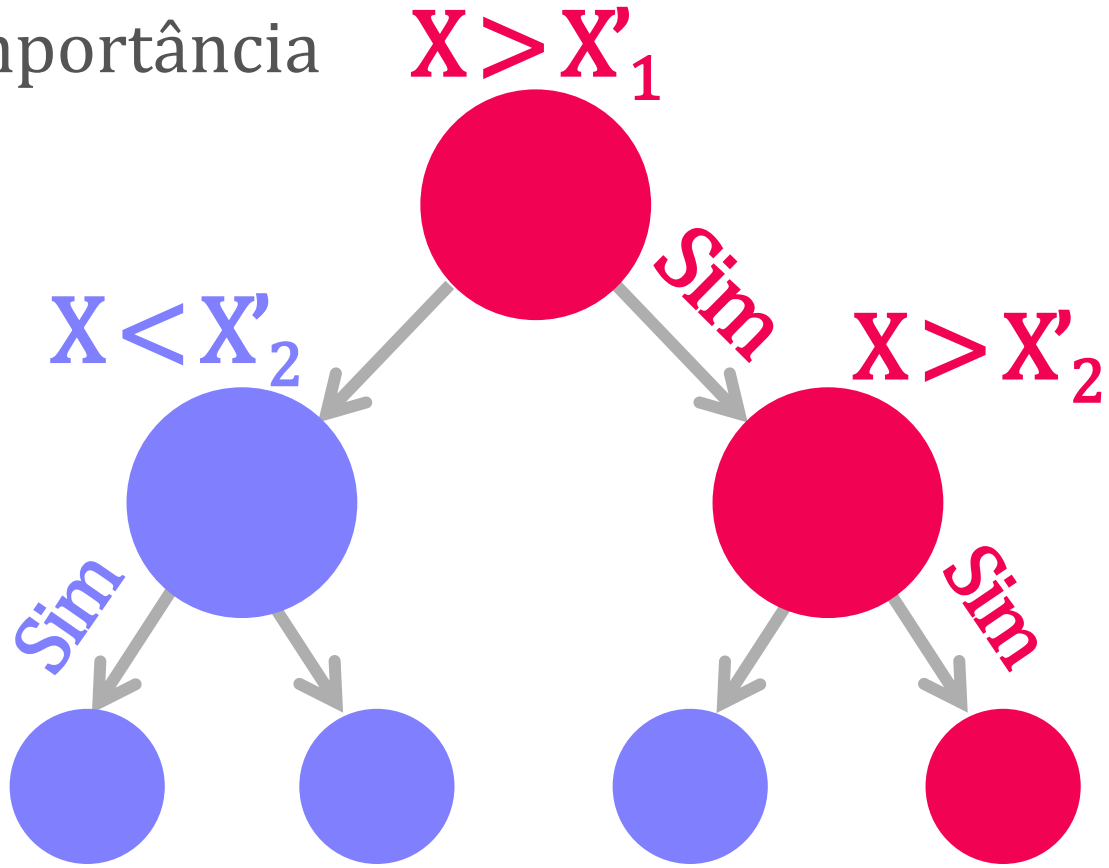
As predições finais são os y médios delimitados pelos hiperplanos (regressão)

Árvores de decisão

Para classificação



variável com alta importância

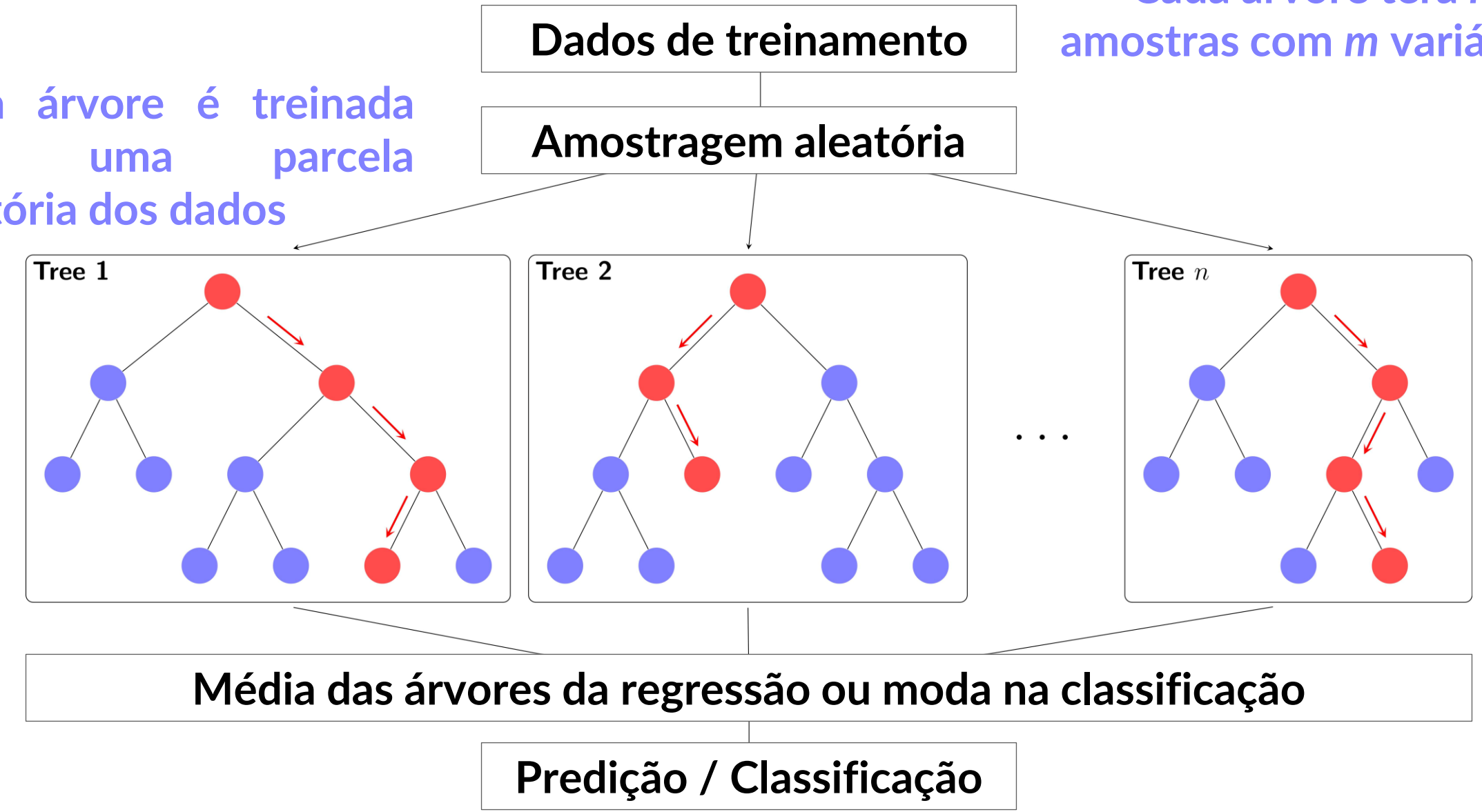


As predições finais são os y mais votados delimitados pelos hiperplanos

Random Forest - Bagging

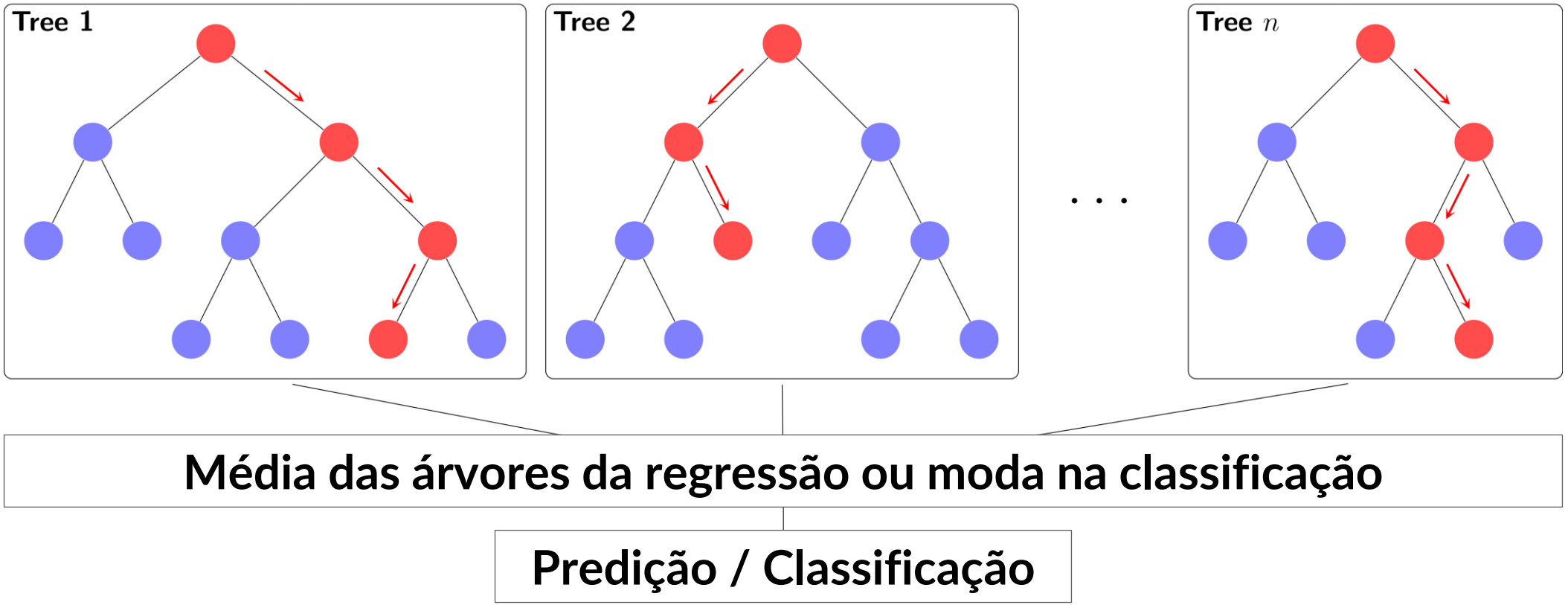
Cada arvore terá *n* amostras com *m* variáveis

Cada árvore é treinada com uma parcela aleatória dos dados



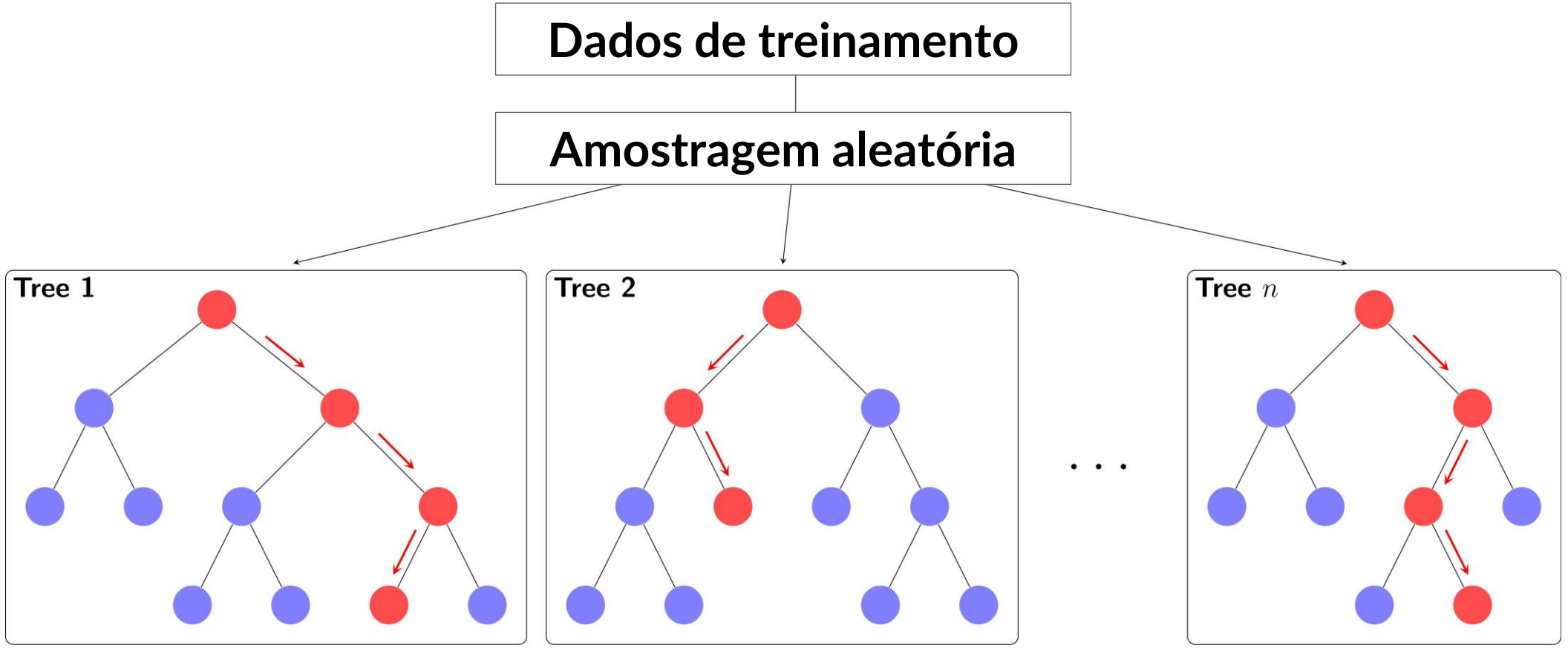
Random Forest - Bagging

Essa aleatoriedade garante que as árvores sejam descorrelacionadas, o que tende a reduzir o risco de *overfitting* e melhora a robustez do modelo



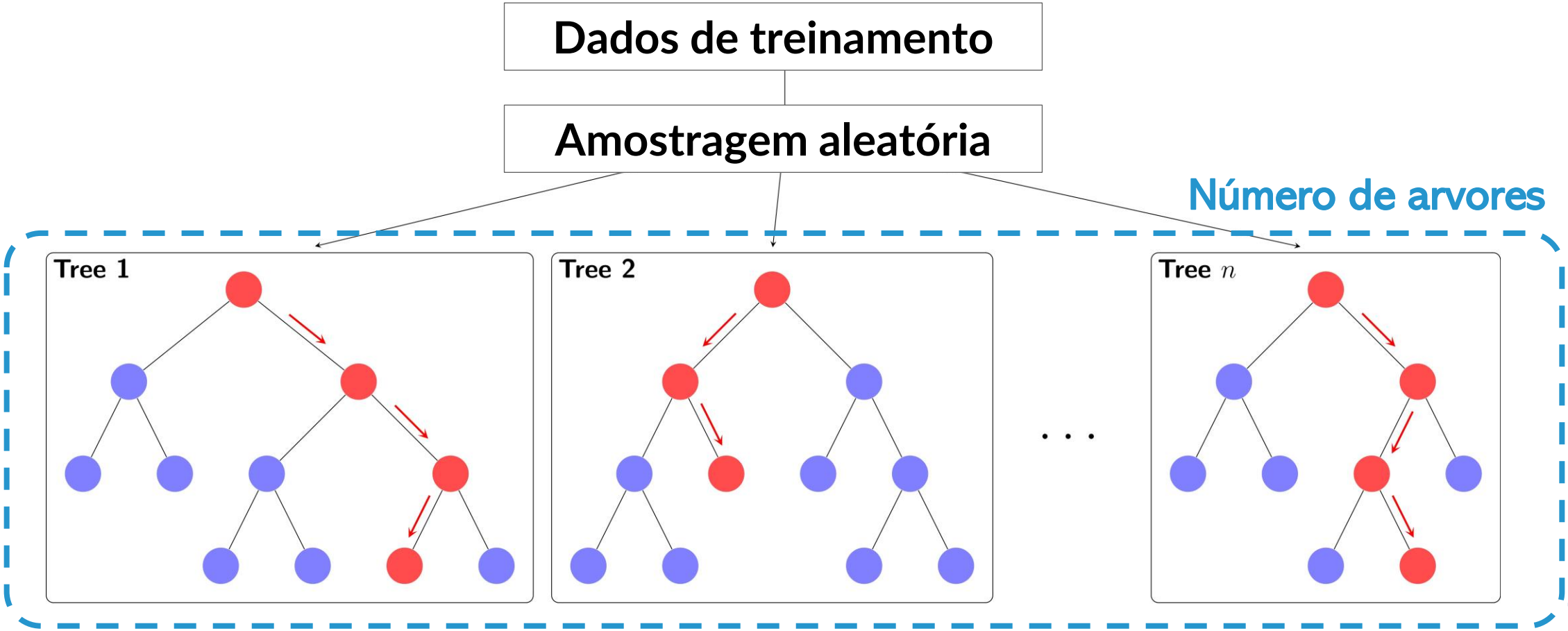
Random Forest - Bagging

Hiperparâmetros: são propriedades internas especificas de cada modelo que precisam ser ajustadas ao longo do treinamento. São variáveis que controlam diferentes aspectos de funcionamento do modelo. Essenciais para uma boa acurácia



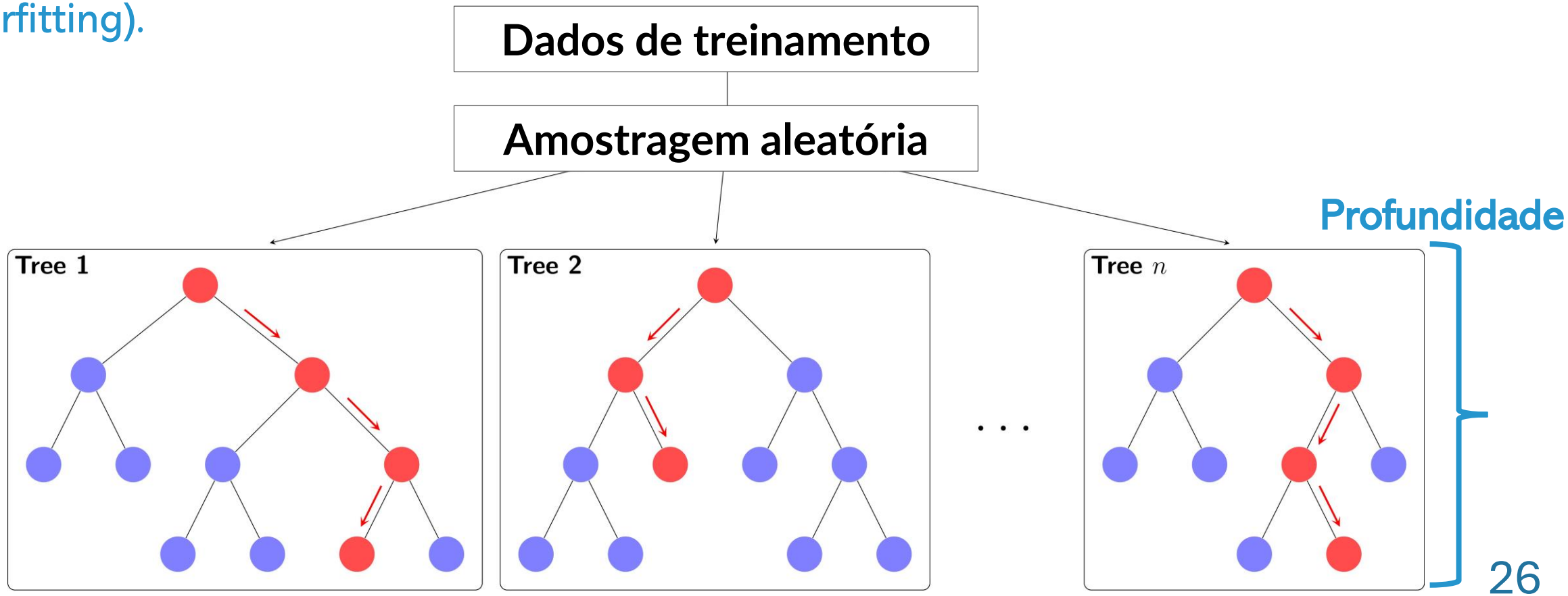
Random Forest - Bagging

***n_estimators* (numero de árvores):** quanto mais árvores, menor a variância das previsões, pois o efeito de decisões extremas de uma única árvore é suavizado pela média das demais. Aumentar muito esse valor eleva o tempo de treinamento e predição pois aumenta a complexidade do modelo.



Random Forest - Bagging

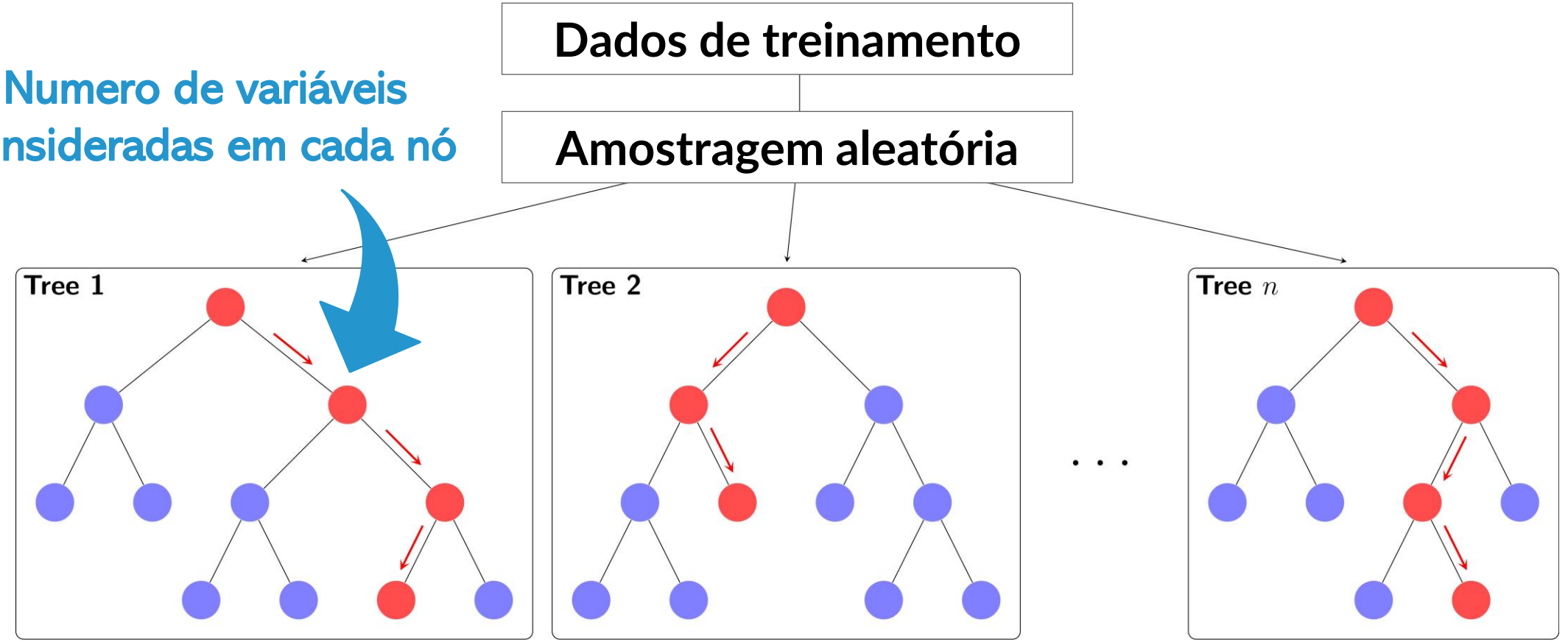
max_depth (profundidade máxima): A profundidade máxima de cada árvore (quantas variáveis são consideradas individualmente). Valores muito baixos resultam em árvores rasas que não capturam padrões complexos (underfitting) enquanto valores muito altos permitem árvores complexas que decoram ruídos dos dados (overfitting).



Random Forest – Principais Hiperparâmetros

Max_features: Define o número de variáveis consideradas em cada divisão de nó. Menor gera árvores mais diversas e diminui correlação entre elas. Entretanto, valores muito baixos podem levar a *underfitting* (árvores fracas), enquanto valores próximos ao total de variaveis podem causar *overfitting* similar a um single *decision tree*

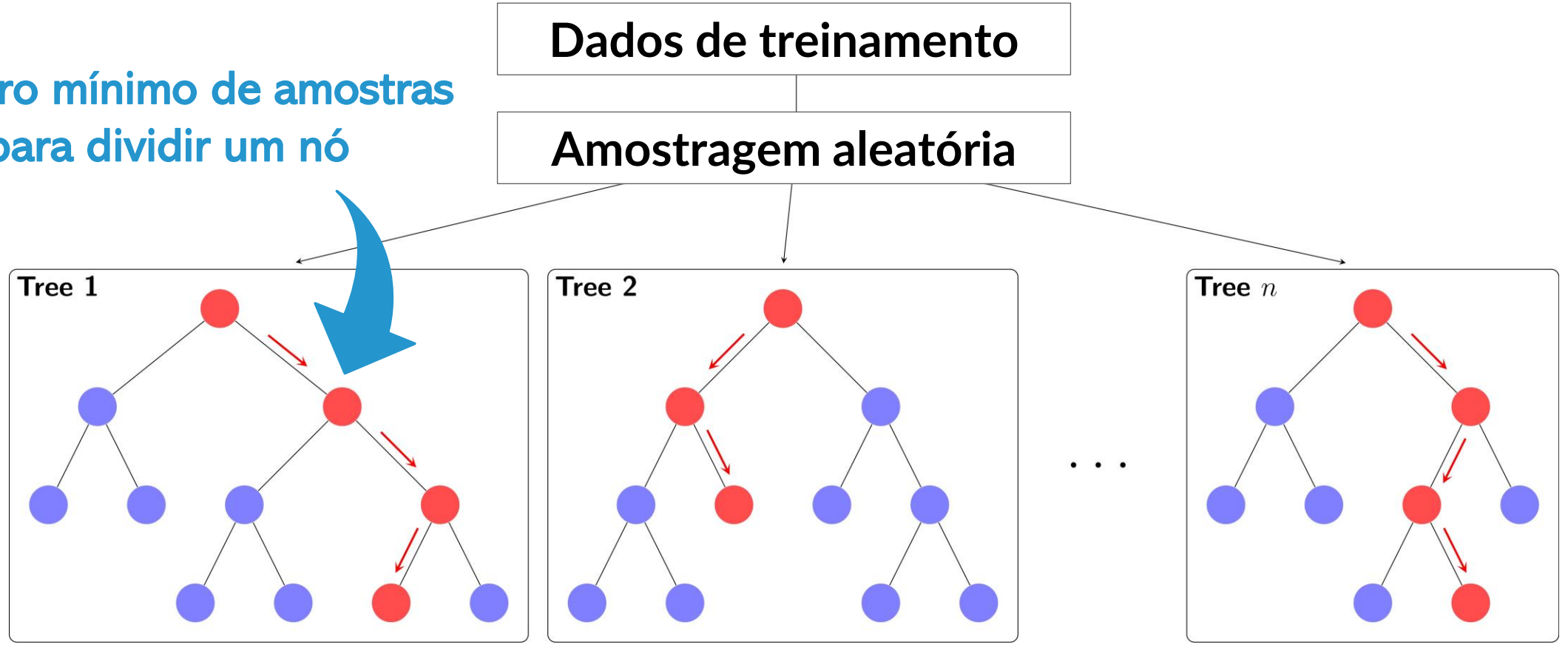
Numero de variáveis consideradas em cada nó



Random Forest – Principais Hiperparâmetros

min_samples_split: O mínimo de amostras que um nó deve ter para ser dividido sub-nós. Quando é muito baixo, a árvore tende a continuar se dividindo até que cada folha seja pura, capturando ruídos e levando a *overfitting*. Com altos valores, poucas divisões são permitidas, resultando em árvores rasas que não conseguem modelar relações complexas

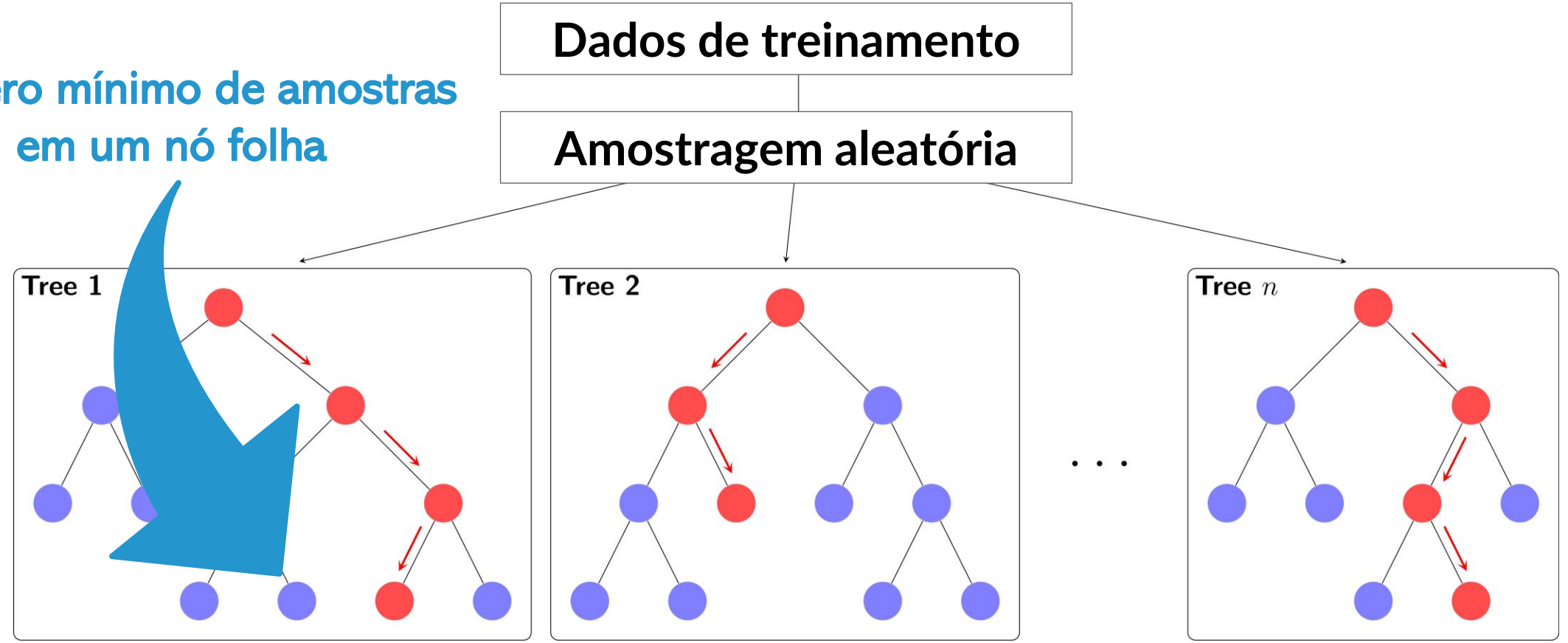
Numero mínimo de amostras para dividir um nó



Random Forest – Principais Hiperparâmetros

min_samples_leaf: Controla o número mínimo de amostras que cada folha (nó terminal) deve conter. Menores geram folhas contendo poucos exemplos, o que favorece curvas de previsão muito irregulares e *overfitting*, pois "cada exceção dos dados pode virar uma regra". Ao aumentar, cada folha agrupa mais exemplos (observações), suavizando a função preditiva.

Numero mínimo de amostras em um nó folha



Ajustando os hiperparametros

Como ajustar os hiperparâmetros? **Validação cruzada!**

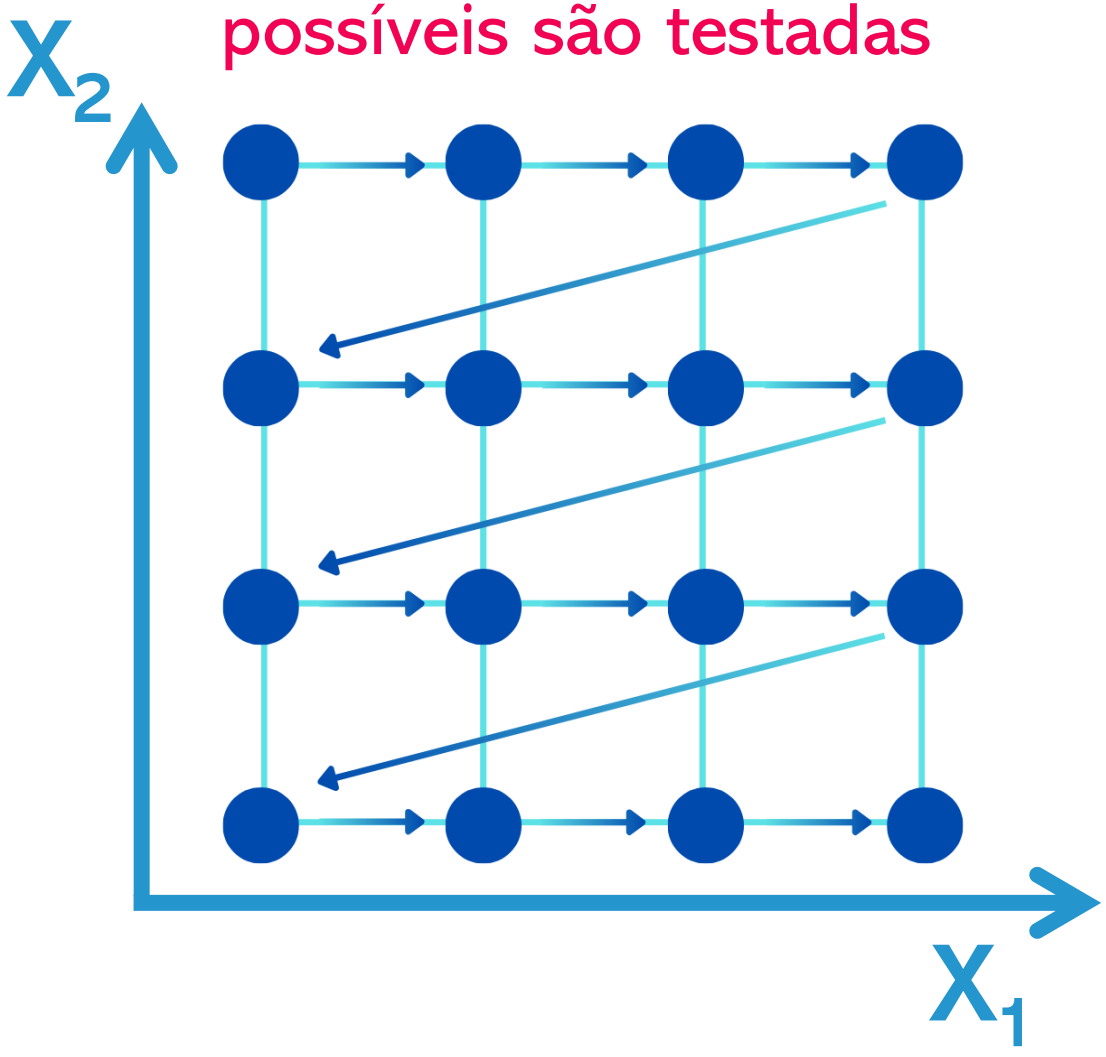
O processo como um todo pode ser visto como um **problema de otimização**, no qual se busca uma combinação ótima de hiperparâmetros.

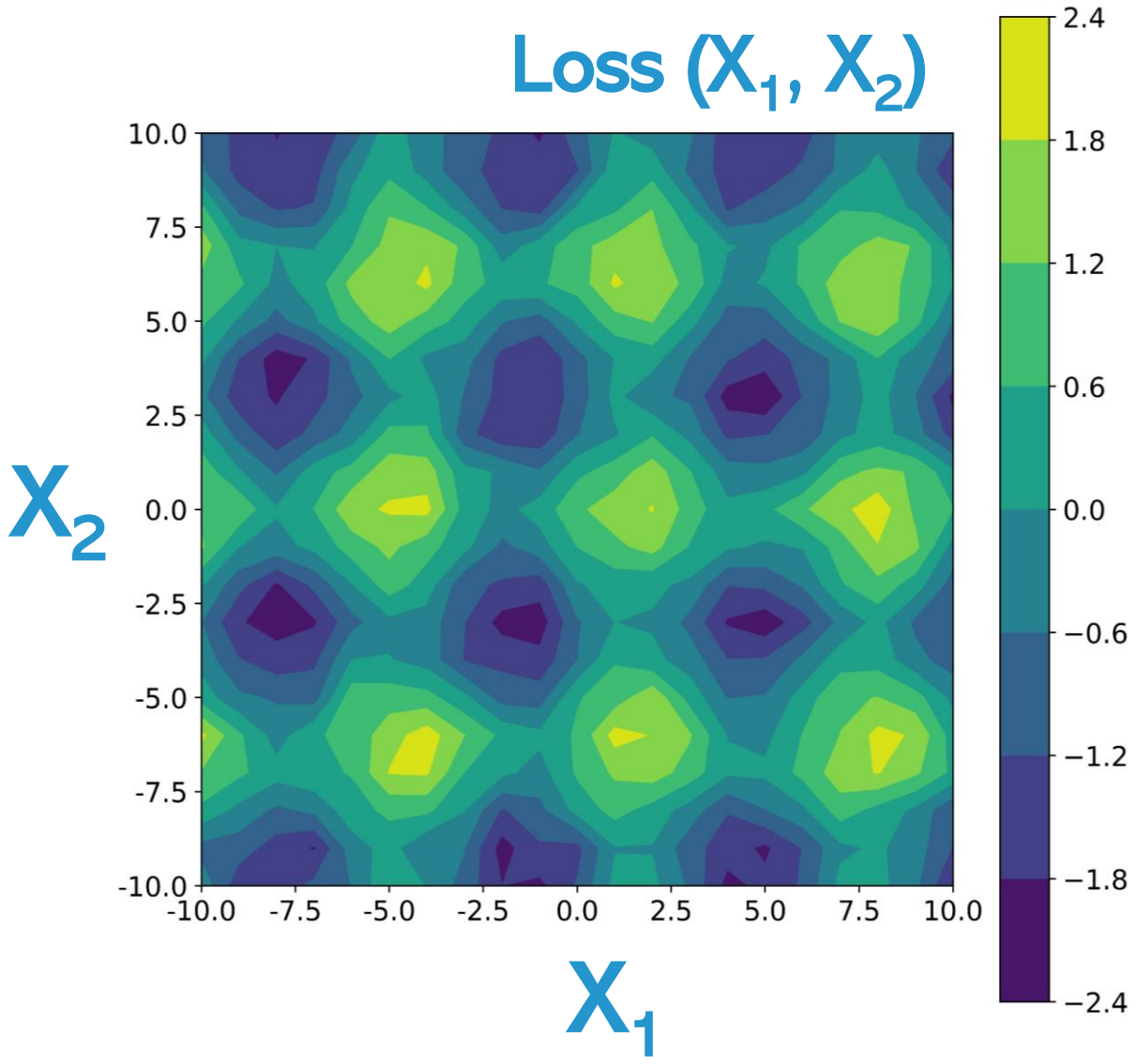
Uma **função de perda** é ajustada e busca-se minimizá-la!

Diversas metodologias podem ser utilizadas para esse proposito: Algoritmo genético, Otimização bayesiana, **Grades de pesquisa exaustiva**, **Grades de pesquisa aleatória**, Otimização por Enxame de Partículas (Particle Swarm Optimization), Recozimento Simulado (Simulated Annealing), etc.

Grades de pesquisa exaustiva

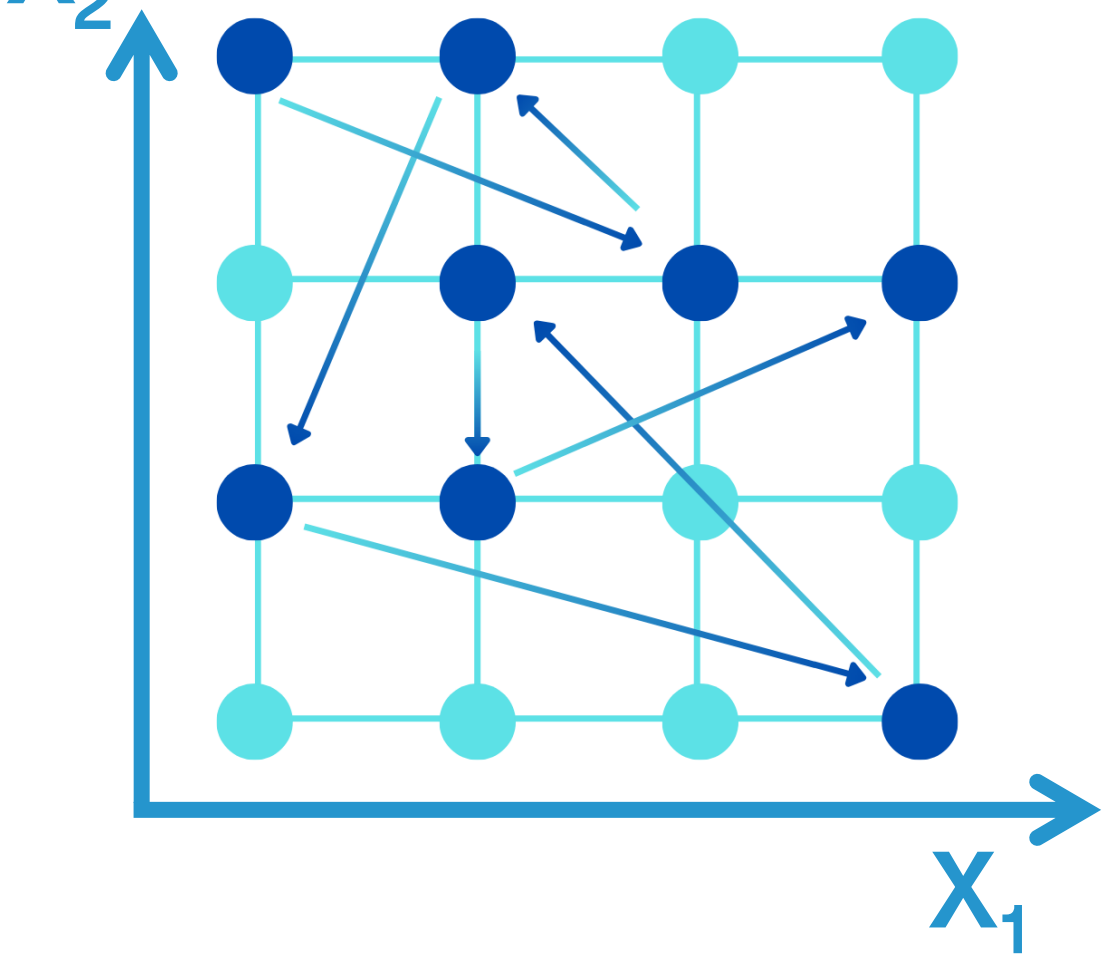
Todas as combinações possíveis são testadas



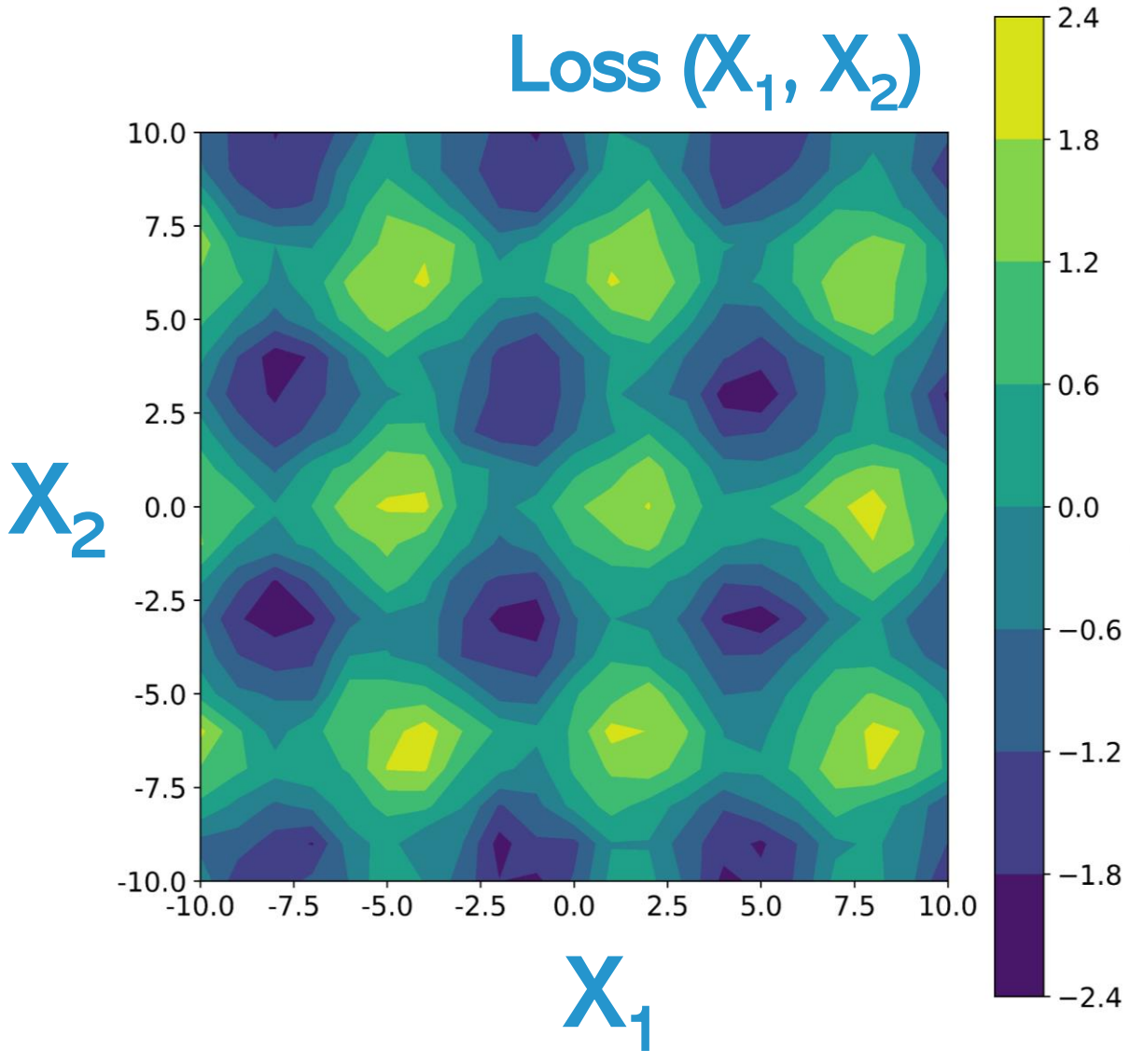


Grades de pesquisa aleatória

Número de combinações aleatórias pré-definido



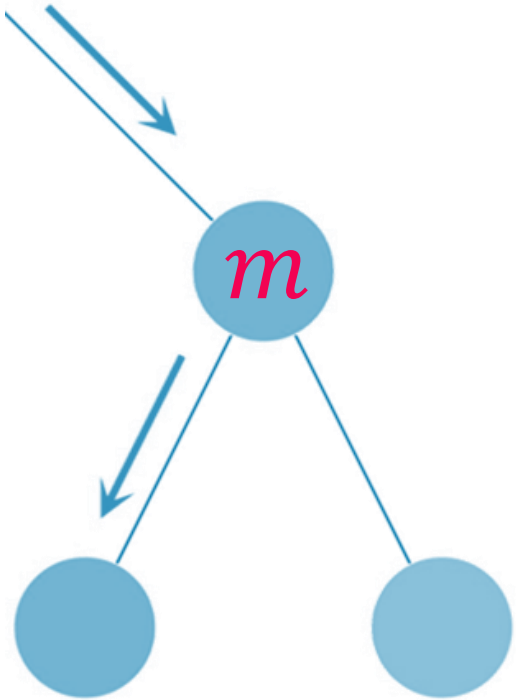
Loss (X_1, X_2)



Interpretabilidade

Interpretabilidade

Cada árvore de decisão é construída de forma recursiva, selecionando em cada nó a *feature* que **maximiza** a redução de um critério de **impureza**



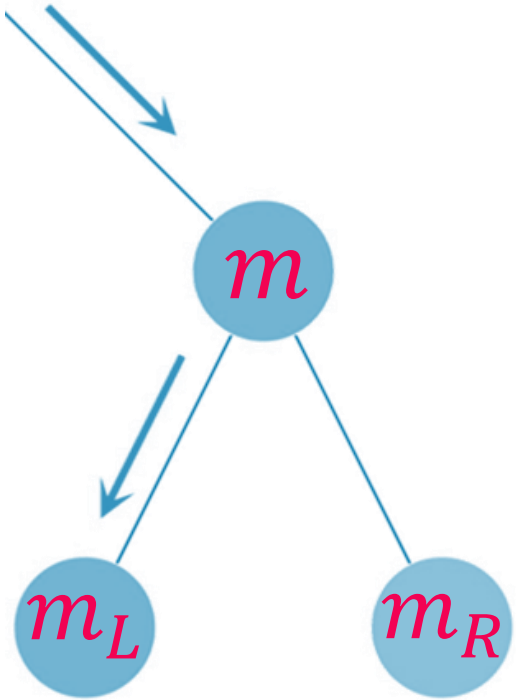
Ex **MSE:** $$i(m) = \frac{1}{N(m)} \sum_{i \in m} (y_i - \bar{y}_m)^2$$

- $i(m)$: impureza (ou variância) no nó m;
- $N(m)$: número de amostras no nó m;
- y_i: valor real observado;
- $\bar{y}_m$: média dos valores y_i no nó m.

Interpretabilidade

Impureza: $\Delta i(m) = i(m) - (p_L \cdot i(m_L) + p_R \cdot i(m_R))$

- $i(m)$: impureza do nó pai;
- $i(m_L)$ e $i(m_R)$: impurezas dos nós filhos (esquerdo e direito, respectivamente);
- p_L e p_R: proporções de amostras que caem nos nós esquerdo e direito, ou seja,



$$p_L = \frac{N(m_L)}{N(m)} \quad \text{e} \quad p_R = \frac{N(m_R)}{N(m)}$$

Interpretabilidade

Feature importance em cada árvore:

$$\mathrm{FI}^{(tree)}(j) = \sum_{m \in M_j} \frac{N(m)}{N_{\text{total}}} \Delta i(m)$$

- M_j: conjunto de nós onde a feature j foi utilizada para realizar o split;
- $N(m)$: número de amostras no nó m;
- N_{total}: número total de amostras utilizadas para construir a árvore;
- $\Delta i(m)$: redução de impureza alcançada no nó m com a divisão.

Interpretabilidade

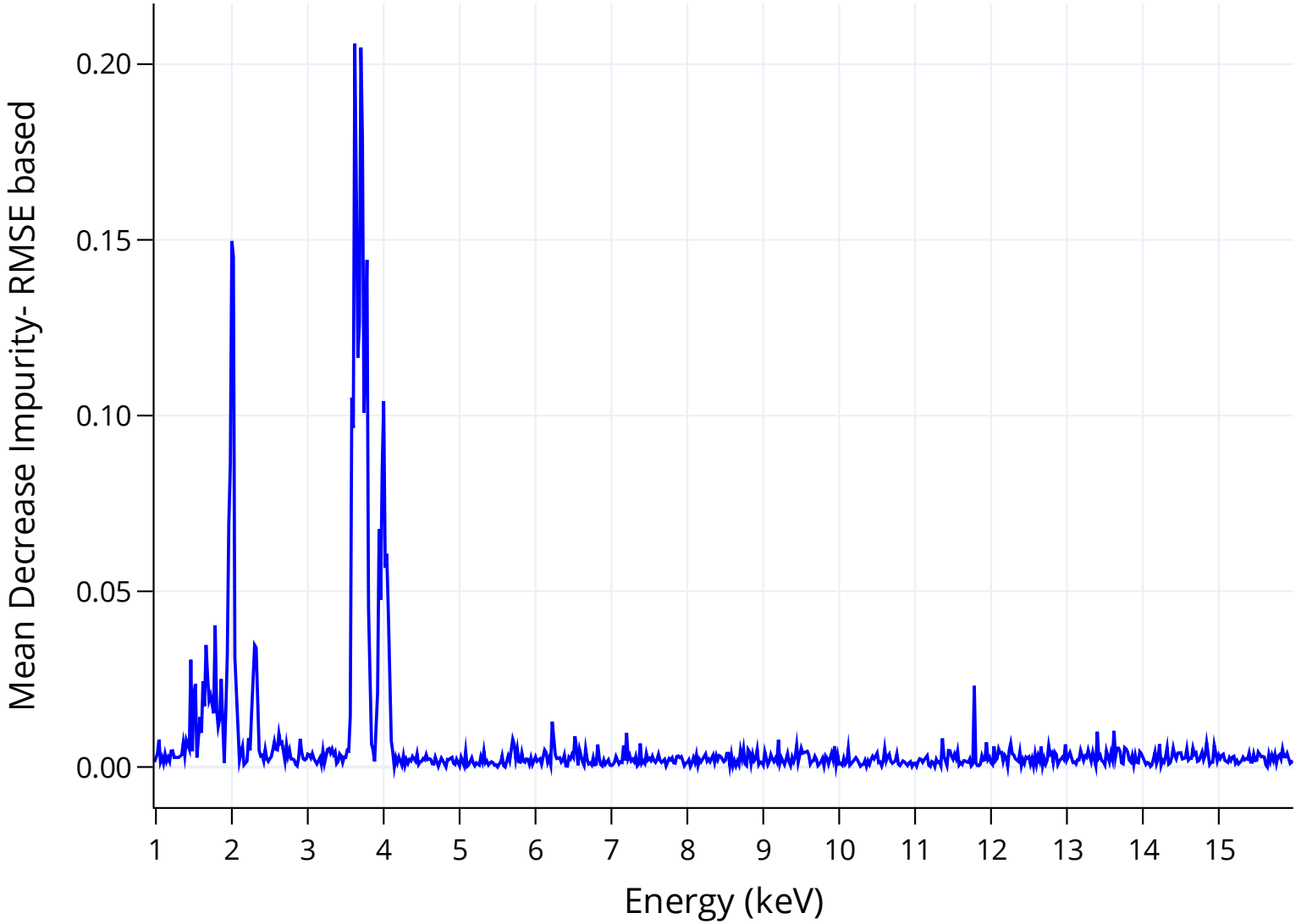
Feature importance no modelo geral:

$$\mathrm{FI}_{RF}(j) = \frac{1}{N_{\mathrm{trees}}} \sum_{t=1}^{N_{\mathrm{trees}}} \mathrm{FI}_t(j)$$

- N_{trees}: número de árvores na floresta;
- $\mathrm{FI}_t(j)$: importância da feature j na árvore t.

Interpretabilidade

Exemplo de modelo RF espectral



Random Forest

Vantagens

- Alta capacidade preditiva
- Realiza *feature importance*
- Lida bem com *datasets* grandes
- Gera métricas internas de erro
- Aleatoriedade combate o *overfitting*

Desvantagens

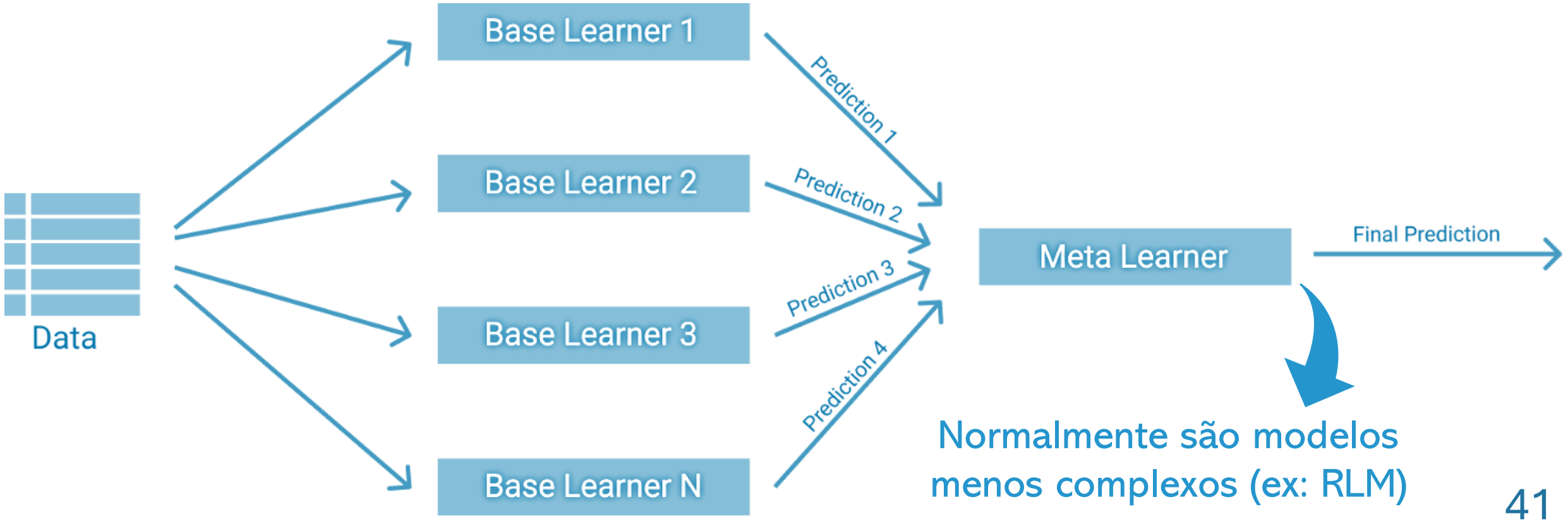
- Complexidade computacional
- Dependência dos hiperparametros
- Aleatoriedade
- Não lida bem imagem/som/texto

Stacked Generalization

Stacked Generalization

Considerando vários modelos podem ser eficazes em um problema, mas de maneiras diferentes, esse método combina todos através de um **meta-modelo**

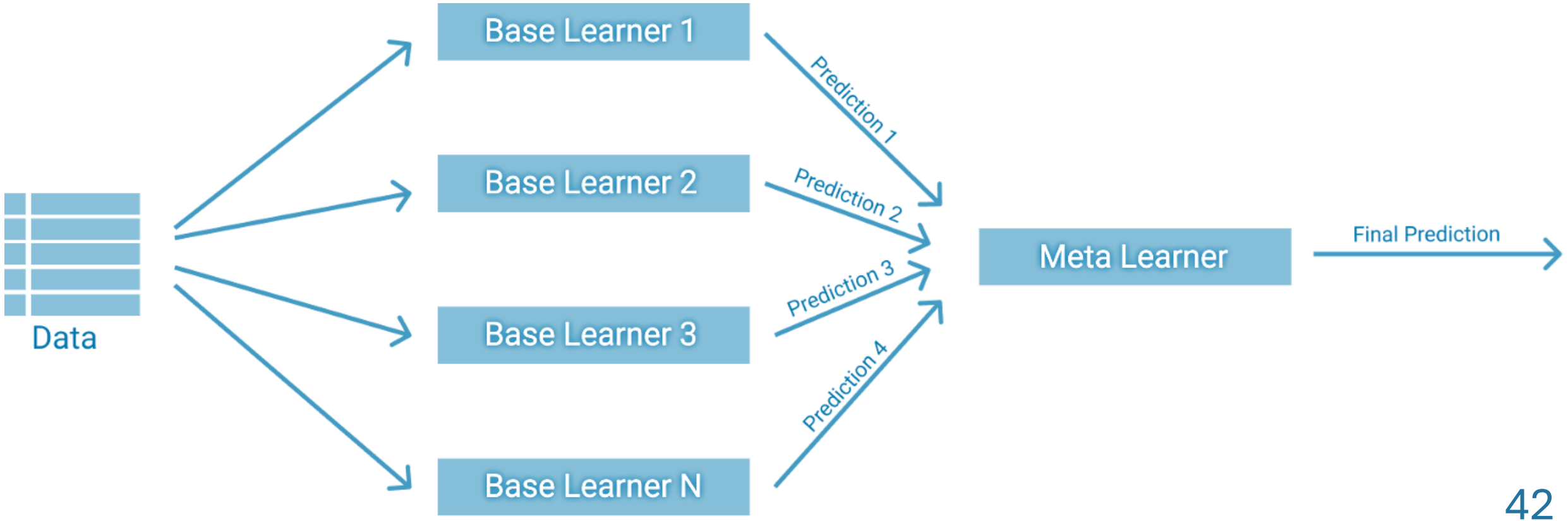
Técnica de "**empilhar**" modelos de base, que treinam no mesmo *dataset*



Normalmente são modelos menos complexos (ex: RLM)

Stacked Generalization

Embora o método ofereça benefícios como **possível** maior precisão e robustez, ele também tem algumas desvantagens, incluindo **maior complexidade** e a necessidade de **mais recursos computacionais**.



Prática no VS Code