UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

CAPÍTULO 9:
ÁRVORES DE DECISÃO E APRENDIZAGEM EM CONJUNTO

DISCIPLINA: REDES NEURAIS

DOCENTE: PROF. DR. LUIS GONZAGA SALES VASCONCELOS

DISCENTE: ARTHUR FELIPE PEREIRA DE JESUS

MATRÍCULA: 231004030012

TÓPICOS

1. INTRODUÇÃO
2. ALGUMAS CONSIDERAÇÕES
3. ÁRVORES DE DECISÃO
4. FLORESTAS ALEATÓRIAS
5. SOFT SENSING COM RANDOM FOREST
6. ENSEMBLE LEARNING
7. PREDIÇÃO COM XGBOOST
8. CONCLUSÃO

CONSIDERAÇÕES INICIAIS

CONCEITOS

- Conceito de *Bias* (Viés):
  - Quando um modelo é muito simplificado, com baixa capacidade de aprender padrões complexos.
  - Causa erro sistemático e subajuste (*underfitting*).
- Conceito de *Variance* (Variância):
  - Quando um modelo é altamente sensível aos dados de treinamento, ajustando-se demais aos ruídos.
  - Causa sobreajuste (*overfitting*) e falta de generalização.
- *Trade-off Bias-Variance*:
  - O equilíbrio necessário entre a complexidade do modelo e sua capacidade de generalização.

CONCEITOS

- *Base Model* (Modelo Base):
  - Conceito de modelos simples que servem como unidade fundamental para técnicas mais complexas, como ensembles.
  - Exemplo: árvore de decisão como modelo base para *Random Forests*.
- *Outliers*:
  - Valores que se situam fora do padrão geral dos dados.
  - Representam observações extremamente diferentes das demais.
  - Também chamados de valores discrepantes ou pontos fora da curva.

INTRODUÇÃO

INTRODUÇÃO

- Serão apresentados os principais conceitos sobre Árvores de Decisão e técnicas de *Ensemble Learning*, como *Random Forest*, *AdaBoost* e *XGBoost*.
- Destaca-se a importância dessas técnicas na construção de modelos mais robustos e precisos, com aplicações práticas em *Soft Sensing*, permitindo a estimativa de variáveis difíceis de medir em processos industriais.

ÁRVORES DE DECISÃO

O QUE É *DECISION TREE*?

- Algoritmo de aprendizado de máquina supervisionado.
- É utilizado para classificação e regressão:
  - Prever categorias discretas (sim ou não, por exemplo);
  - Prever valores numéricos (o valor do lucro em reais).
- Estabelece nós (*decision nodes*) que relacionam hierarquia entre si:
  - Nó-raiz (*root node*) é o mais importante:
    - Um dos atributos da base de dados.
  - Nós-folhas (*leaf nodes*) são os resultados finais:
    - Classe ou valor que será gerado como resposta.
  - Ramos (branches) são os caminhos de decisão.

COMO FUNCIONA?

- O modelo é estruturado como uma sequência de declarações condicionais (if-else), o que gera uma estrutura em forma de árvore.
- Cada nó faz uma pergunta clara, facilitando a interpretação de como a árvore chega a uma determinada classe.
- Assim, DTs são ideais para aplicações onde a interpretabilidade do modelo é crucial.

AJUSTE DO MODELO

- O principal desafio no ajuste de uma árvore de decisão está na escolha criteriosa das perguntas em cada nó:
  - Quais atributos utilizar e quais valores de corte empregar.
- O objetivo é dividir o espaço de entrada em regiões menores e mais homogêneas.
- O processo de divisão continua até que não haja mais ganhos ou que algum critério de parada seja satisfeito.
- Caso as divisões sejam mal escolhidas, o modelo resultante terá baixa capacidade de generalização.

ALGORITMO CART

- CART (*Classification and Regression Trees*) é uma variação do algoritmo de DT.
- É um algoritmo preditivo usado em *Machine Learning*.
- Explica como os valores da variável-alvo podem ser previstos.
- É um DT em que cada bifurcação é divindade em uma variável preditora.
- Cada nó tem uma previsão para a variável-alvo no final.
- Ele pode lidar com tarefas de classificação e regressão.
- Basicamente aprende com os dados rotulados para prever dados não vistos.

ESTRUTURA EM ÁRVORE

- O CART constrói uma estrutura semelhante a uma árvore (nós e ramos).
- Nós são os diferentes pontos de decisão.
- Ramos são os caminhos dessas decisões.
- Os nós-folhas contêm um rótulo de classe ou valor previsto.

CRITÉRIOS DE DIVISÃO

- Utiliza o *greedy algorithm* (algoritmo guloso) para dividir os dados em cada nó.
- Avalia todas as divisões possíveis e seleciona a que melhor reduz a impureza dos subconjuntos resultantes.
- Para tarefas de classificação, o CART utiliza impureza de Gini ou entropia.
- Para tarefas de regressão, o CART utiliza o erro quadrático médio (*Mean Squared Error* – MSE).

GREEDY ALGORITHM

- *Greedy Algorithm* ou algoritmo guloso é uma estratégia de solução de problemas que consiste em, a cada passo, escolher a opção que parece ser a melhor no momento, sem considerar as consequências futuras.
- Toma decisões locais ótimas, na esperança de que isso leve a uma solução global ótima.
- Simples de implementar.
- Nem sempre encontra a melhor solução global (pode levar a soluções subótimas).
- Muito eficiente em termos de tempo de execução.
- Nas DTs, ao construir, usa esta abordagem para escolher a melhor divisão em cada nó (máxima redução de impureza).

PODA

- A complexidade da árvore de decisão é definida como o número de divisões na árvore.
- Árvores com menos ramificações são recomendadas, pois são mais fáceis de entender e menos propensas a sofrer *overfitting*.
- Trabalhar em cada nó-folha da árvore e avaliar o efeito de excluí—lo usando um conjunto de testes de retenção é a abordagem de poda mais rápida e simples.
- No Python, veremos que podemos definir a profundidade da árvore através do comando “max_depth”.

ÁRVORES DE DECISÃO PARA CLASSIFICAÇÃO

- Modelo que faz previsões categóricas.
- Atribui uma classe ou categoria a uma nova observação com base em condições sobre as variáveis de entrada.
- A árvore é construída dividindo o conjunto de dados com base em perguntas.
- Cada nó representa uma decisão baseada em uma variável.
- As divisões são feitas de forma gulosa, escolhendo sempre a melhor separação no momento.
- Utiliza o método de impureza de Gini ou entropia como critério de divisão.
- Em Python, podemos utilizar parâmetro de parada como “min_samples_split”.

IMPUREZA DE GINI

- Mede o grau de impureza ou a mistura de classes em um conjunto de dados.
- Quanto menor a impureza de Gini, mais puro é o grupo (ou seja, mais homogêneo em relação à classe).
- Se todas as amostras de um nó pertencem à mesma classe, a impureza de Gini é zero.
- Se as classes estão equilibradas, a impureza é máxima.

IMPUREZA DE GINI

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

- p_i: proporção de elementos da classe i no nó.
- n: número de classes.

ENTROPIA

- Além da impureza de Gini, outra métrica popular é a entropia.
- Quantifica a impureza ou incerteza de um conjunto de dados.
- Mede o nível de desordem:
  - Quanto maior a entropia, mais mistas são as classes no nó.
  - Quanto menor a entropia (próxima de zero), mais puro é o nó.
- Quando todas as amostras pertencem à mesma classe, a entropia é zero (máxima pureza).
- Quando as amostras estão equilibradas entre as classes, a entropia é máxima (máxima impureza).

ENTROPIA

$$I_H = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

- p_i: proporção de elementos da classe i no nó.
- n: número de classes.
- Por convenção, $0 \cdot \log(0) = 0$.

ÁRVORES DE DECISÃO PARA REGRESSÃO

- Modelo que faz previsões contínuas.
- Estima um valor numérico para uma nova observação com base em condições sobre as variáveis.
- Similar à árvore de classificação, mas em vez de prever uma classe, calcula uma média ou outro valor estatístico das amostras que chegam a uma folha da árvore.
- Cada divisão busca minimizar o erro da previsão.
- Utiliza o MSE como critério de divisão.

MEAN SQUARED ERROR (MSE)

- Mede o erro médio ao quadrado entre os valores previstos e os valores reais.
- Quanto menor o MSE, melhor o modelo ou a divisão da árvore.
- Penaliza erros grandes mais severamente, pois eleva o erro ao quadrado.

MEAN SQUARED ERROR (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- y_i: valor real da i-ésima amostra.
- $\hat{y}_i$: valor previsto para a i-ésima amostra.
- n: número de amostras.

CONSIDERAÇÕES FINAIS SOBRE DT

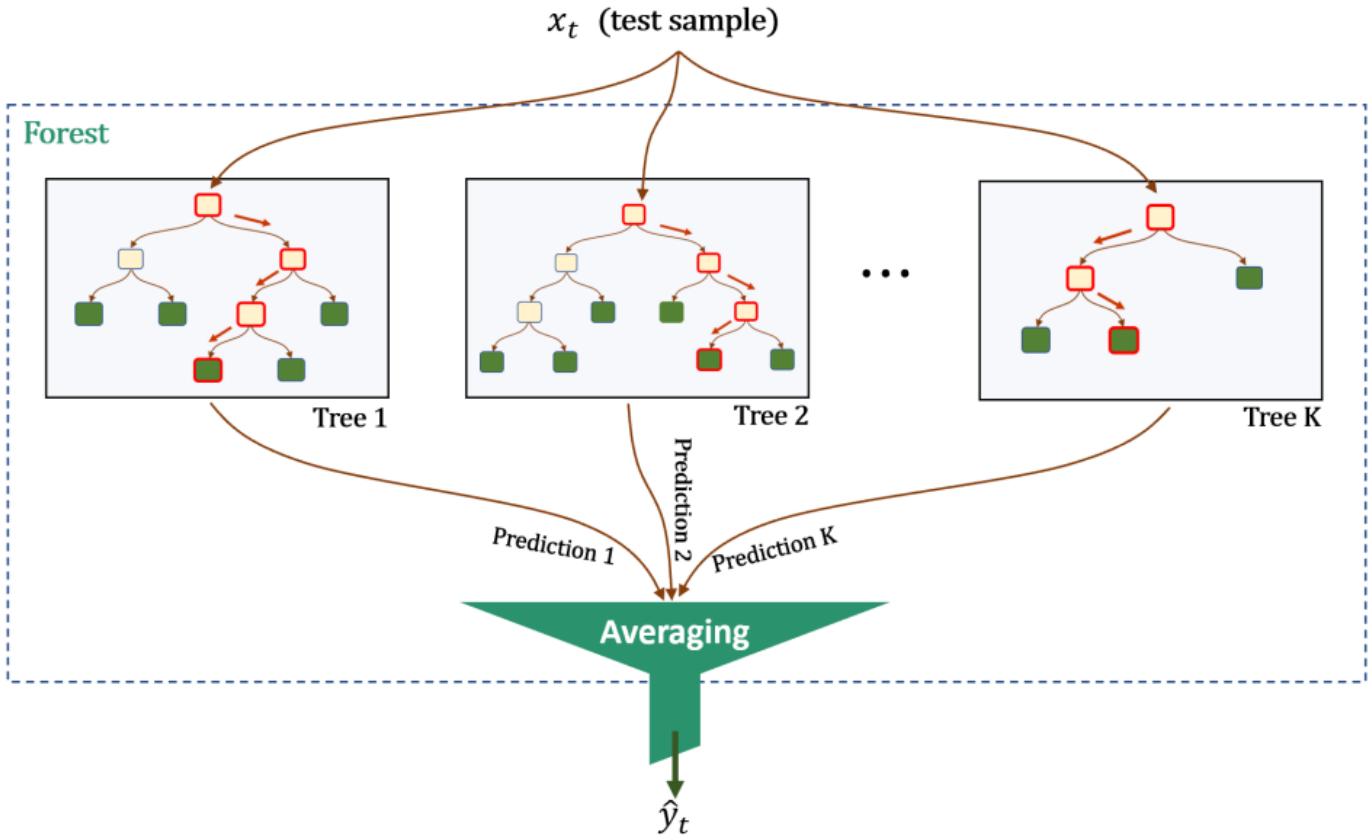
- Apesar da flexibilidade, as árvores de decisão isoladas têm limitações sérias:
  - Tendência ao *overfitting*;
  - Predições pouco suaves (*piecewise*);
  - Instabilidade frente a pequenas variações nos dados de treinamento.
- Contudo, aprender DTs é essencial, pois elas são a base de técnicas mais poderosas como *Random Forests* e *XGBoost*.
- Quando utilizadas em conjunto, árvores fracas podem formar modelos robustos e precisos.

FLORESTAS ALEATÓRIAS

O QUE É *RANDOM FOREST?*

- É uma técnica de *ensemble learning* que combina várias árvores de decisão para criar um modelo mais robusto e preciso.
- Cada árvore contribui com sua predição e, no final:
  - Para classificação: a classe mais votada é escolhida (*majority voting*).
  - Para regressão: é calculada a média das previsões.

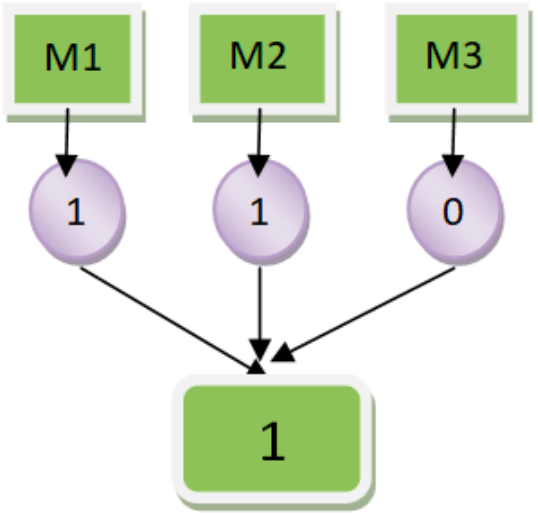
FLORESTA ALEATÓRIA



MAJORITY VOTING

- Técnica de combinação de modelos usada principalmente em *ensemble learning*.
- Cada modelo do *ensemble* faz sua previsão de classe e, no final, a classe mais votada é escolhida como a previsão final.
- Cada modelo individual faz uma previsão categórica.
- Conta-se o número de vezes que cada classe foi prevista.
- A classe com o maior número de votos é selecionada.

MAJORITY VOTING

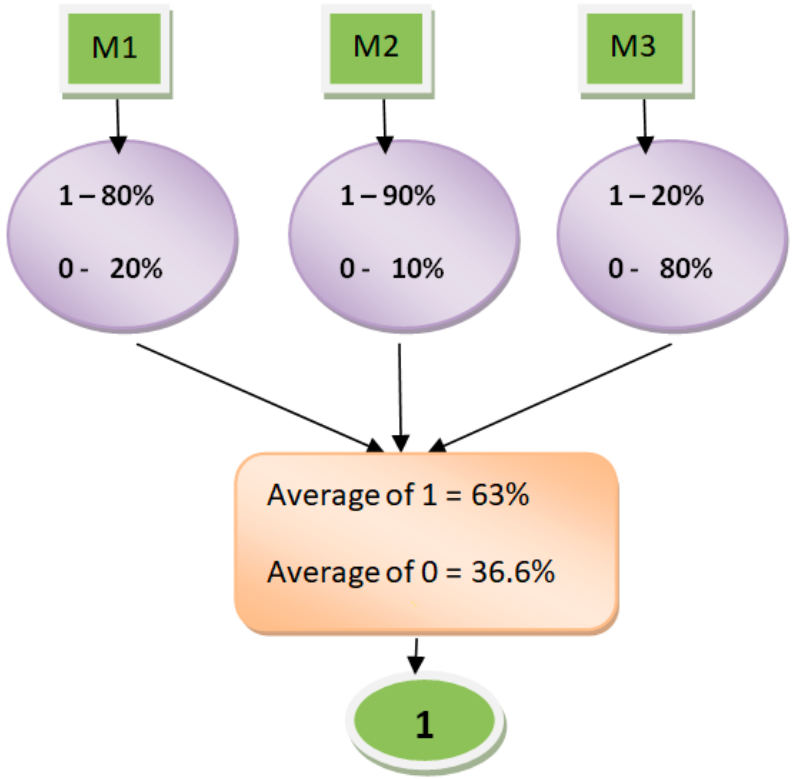


Hard Voting

SOFT VOTING

- Técnica onde cada modelo não fornece apenas a classe prevista, mas também a probabilidade associada a cada classe.
- A previsão final é feita pela soma das probabilidades ponderadas, escolhendo a classe com maior probabilidade agregada.
- Cada modelo gera um vetor de probabilidades para cada classe.
- As probabilidades são somadas (ou ponderadas se for o caso).
- A classe com a maior probabilidade total é escolhida.

SOFT VOTING


M1
M2
M3
1 – 80%
0 - 20%
1 – 90%
0 - 10%
1 – 20%
0 - 80%
Average of 1 = 63%
Average of 0 = 36.6%
1

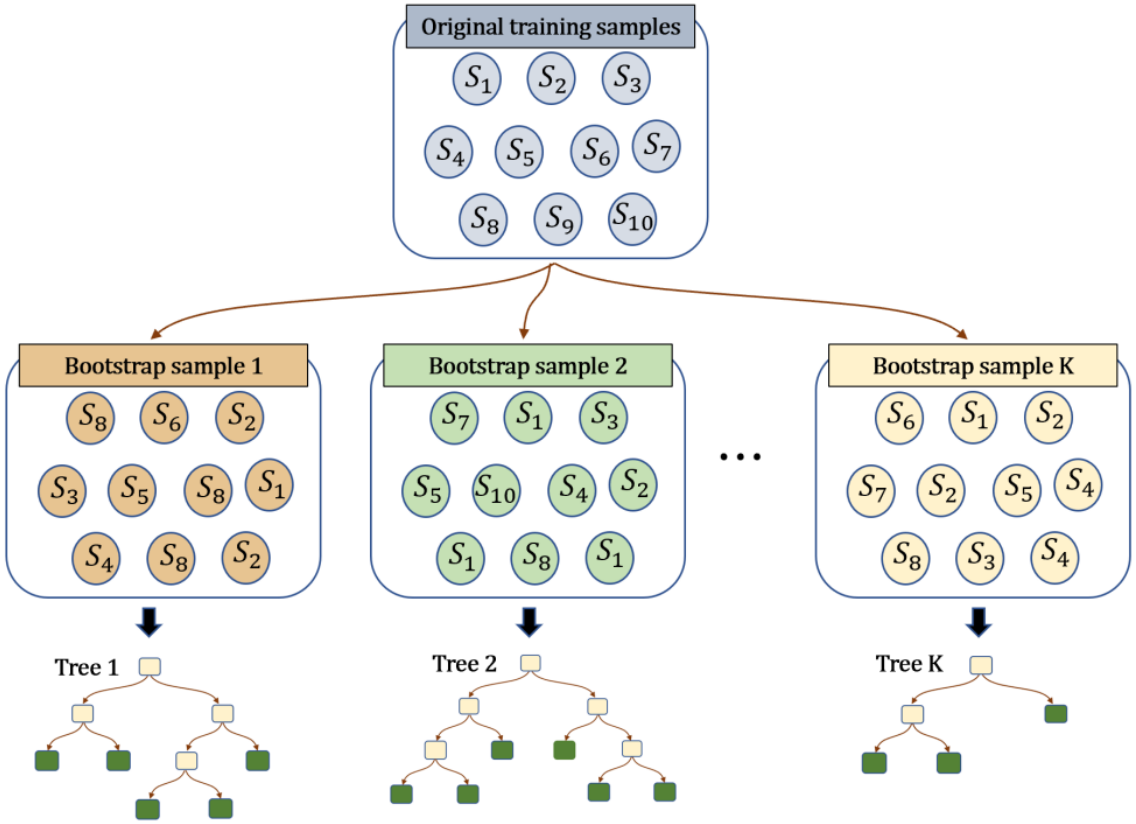
COMO FUNCIONA?

- Gera múltiplos subconjuntos de dados a partir do conjunto original (via *bootstrapping*).
- Para cada subconjunto, treina-se uma árvore de decisão completa, mas a cada divisão, apenas um subconjunto aleatório de variáveis é considerado (*feature bagging*).
- A predição do modelo é obtida pela média das previsões das árvores individuais.
- Essa combinação reduz a variância e gera modelos mais robustos.
- O segredo está em como essas árvores são geradas, de modo que seus erros de predição sejam não correlacionados, o que reduz significativamente o *overfitting*.
- As árvores do RF são treinadas até profundidade máxima, o que dispensa a necessidade de afinar hiperparâmetros como profundidade ou número mínimo de amostras.

BOOTSTRAPPING

- Técnica de reamostragem usada para criar múltiplos subconjuntos aleatórios (com reposições) a partir do conjunto original de dados.
- Gera diversos subconjuntos de dados por amostragem com reposição (ou seja, o mesmo exemplo pode aparecer várias vezes).
- Para cada subconjunto, treina-se um modelo fraco (como uma árvore de decisão).
- As previsões são então combinadas:
  - Votação para classificação.
  - Média para regressão.

BOOTSTRAPPING



FEATURE BAGGING

- *Feature Bagging* é uma técnica de *ensemble learning* que consiste em selecionar aleatoriamente um subconjunto de variáveis (*features*) para treinar cada modelo de um *ensemble*.
- Também conhecido como *Random Subspaces* ou *atribute bagging*.
- Durante a construção de cada árvore, em cada divisão (split), a Random Forest:
  - Considera apenas um subconjunto aleatório de variáveis para escolher a melhor divisão.
  - Não utiliza todas as variáveis disponíveis.
- Reduz a correlação entre os modelos do *ensemble* (maior diversidade).

FUNDAMENTOS MATEMÁTICOS

- Para que o RF funcione de forma eficaz, é essencial que as árvores sejam o mais diferentes possível entre si.
- Isso é obtido por dois mecanismos principais:
  - Uso de conjuntos de treinamento diferentes para cada árvore:
    - Se todas as árvores forem treinadas com os mesmos dados, produzirão os mesmos erros.
    - Para evitar isso, usa-se *bootstrapping*.
    - Cada árvore é treinada com uma amostragem aleatória com reposição de tamanho $N_b \leq N$.
  - Uso de subconjuntos aleatórios de variáveis em cada divisão:
    - Ao invés de usar todas as variáveis para decidir os melhores splits, a RF seleciona aleatoriamente um subconjunto de variáveis em cada nó.
    - Essa técnica, contraintuitiva porém eficaz, aumenta a diversidade entre as árvores.

VANTAGENS E DESVANTAGENS

- Vantagens:
  - Reduz *overfitting* em comparação com uma única árvore de decisão.
  - Altamente robusto a *outliers* e ruídos.
  - Trabalha bem com dados de alta dimensionalidade.
- Desvantagens:
  - Mais computacionalmente custoso.
  - Menor interpretabilidade em relação a uma única árvore.

SOFT SENSING COM *RANDOM FOREST*

SOFT SENSING

- *Soft Sensing* (ou sensor suave) é uma técnica que utiliza modelos matemáticos ou algoritmos de aprendizado de máquina.
- Estima variáveis de processo que:
  - São difíceis, impossíveis, demoradas ou custosas de medir diretamente.
  - Não estão disponíveis em tempo real.
- Coleta-se dados de variáveis de processo que são facilmente medidas (por exemplo, temperatura, pressão, vazão).
- Utiliza-se um modelo preditivo para estimar a variável de interesse, chamada de variável latente (por exemplo, concentração, resistência, qualidade).
- A estimativa serve como um sensor virtual ou *soft sensor*.
- Exemplo: Estimar a resistência do concreto com base nas proporções de materiais (cimento, areia, água) e tempo de cura, ao invés de esperar pelos ensaios destrutivos.

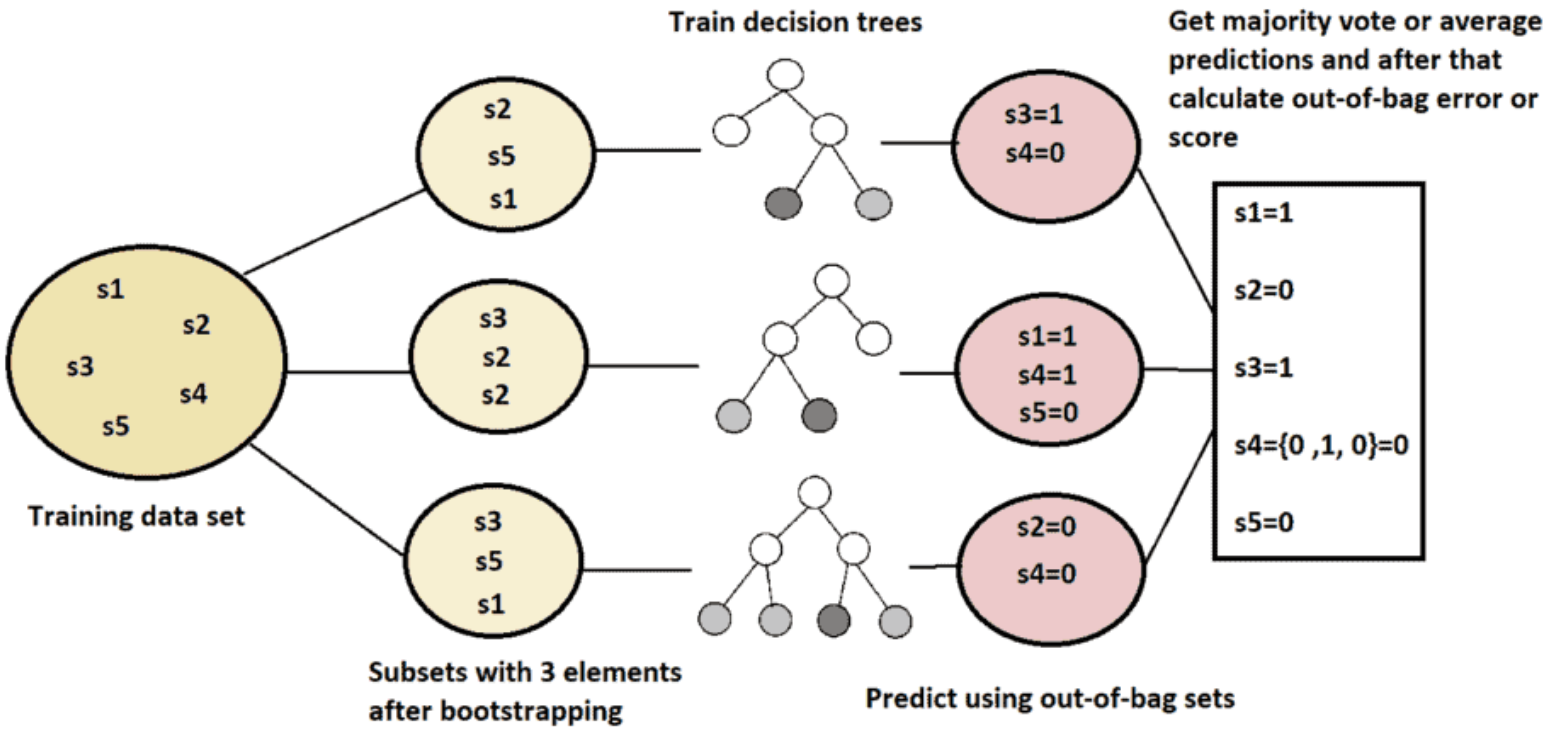
SOFT SENSING COM RANDOM FOREST

- Aplicação da técnica de *Soft Sensing* utilizando o modelo de *Random Forest* como estimador preditivo.
- A *Random Forest* é usado para aprender a relação complexa entre as variáveis de entrada (facilmente mensuráveis) e a variável-alvo (latente).
- Treina-se um *Random Forest Regressor* com dados históricos, relacionando as entradas observadas com as medições da variável.
- Após o treinamento, o modelo pode ser usado para prever a variável em tempo real com base apenas nas variáveis de entrada disponíveis.
- Exemplo: Estimar a concentração de poluentes em uma Estação de Tratamento de Esgoto (ETE) com base em variáveis como vazão, pH e temperatura.

OOB (*OUT-OF-BAG*)

- OOB (*Out-of-Bag*) significa “fora do saco” e é uma técnica utilizada principalmente em modelos baseados em *Bagging*, como a *Random Forest*.
- Refere-se aos dados que não foram selecionados para treinar uma determinada árvore dentro do ensemble.
- Para cada árvore na *Random Forest*, é criado um subconjunto de dados via *bootstrapping* (amostragem com reposição).
- Em média, cerca de 1/3 dos dados não são selecionados para treinar cada árvore → esses são os chamados Out-of-Bag samples.
- Esses dados “deixados de fora” podem ser usados para:
  - Avaliar a performance da árvore de forma natural e não enviesada.
  - Calcular o erro OOB (*Out-of-Bag Error*) sem necessidade de um conjunto de validação separado.

OOB (*OUT-OF-BAG*)



IMPORTÂNCIA DAS VARIÁVEIS

- A RF permite calcular a importância de cada variável com base na redução média de impureza (ou MSE) proporcionada por ela nas divisões.
- Importância em uma árvore:

$$FI_j^{tree} = \frac{\sum_{nodes;\, node\ splitting\ on\ feature\ j} \Delta I_{node}^{tree}}{\sum_{nodes} \Delta I_{node}^{tree}}$$

- Importância média no modelo:

$$FI_j = \frac{\sum_{trees} FI_j^{tree}}{number\ of\ trees}$$

CONSIDERAÇÕES FINAIS SOBRE SS-RF

- *Random Forests* são modelos:
  - Robustos;
  - Pouco sujeitos a *overfitting*;
  - Simples de configurar;
  - Interpretáveis via importância de atributos;
  - Efetivos mesmo sem pré-processamento pesado.

ENSEMBLE LEARNING

O QUE É *ENSEMBLE LEARNING*?

- *Ensemble Learning* (ou aprendizagem em conjunto ou algoritmos de comitê de classificadores) é uma técnica de aprendizado de máquina que combina múltiplos modelos individuais, chamados de “modelos base” ou “modelos fracos”, para produzir um modelo mais forte e preciso.
- O princípio é que um conjunto de modelos diversos pode, em conjunto, superar o desempenho de qualquer modelo individual.
- Pode ser:
  - Heterogêneo: combina modelos diferentes (como SVM, ANN, GMM, RF).
  - Homogêneo: combina instâncias do mesmo tipo de modelo (como múltiplas árvores de decisão).

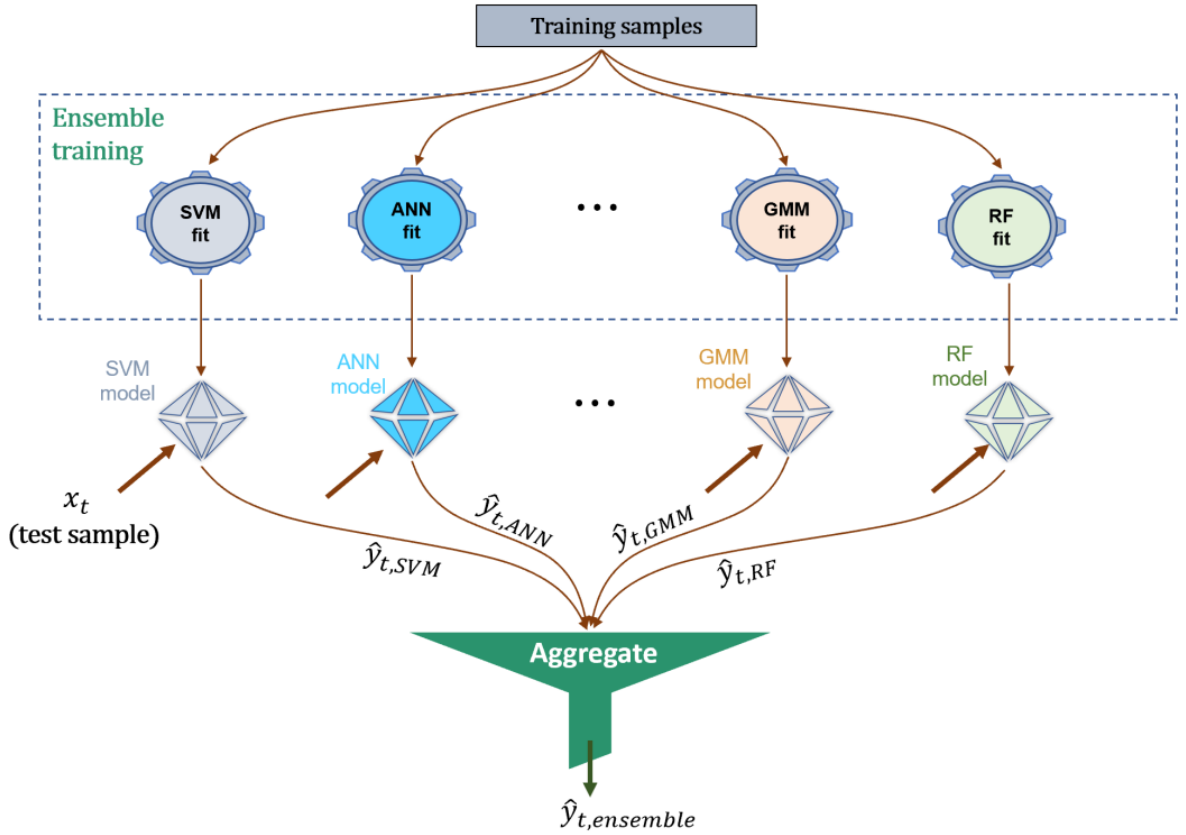
COMO FUNCIONA?

- Treinam-se vários modelos sobre o mesmo problema, mas com alguma forma de variação entre eles:
  - Diferenças nos dados (subconjuntos, pesos).
  - Diferenças nas variáveis usadas.
  - Diferenças nos algoritmos ou parâmetros.
- As previsões de cada modelo são combinadas de uma forma específica:
  - Por votação (classificação).→ Por média (regressão).
  - Por ponderação das previsões.
- Requisitos para que o ensemble funcione bem:
  - Modelos base devem ser melhores que o acaso (precisão > 50%);
  - Erros devem ser não correlacionados (erro que influencia na ocorrência de outro).

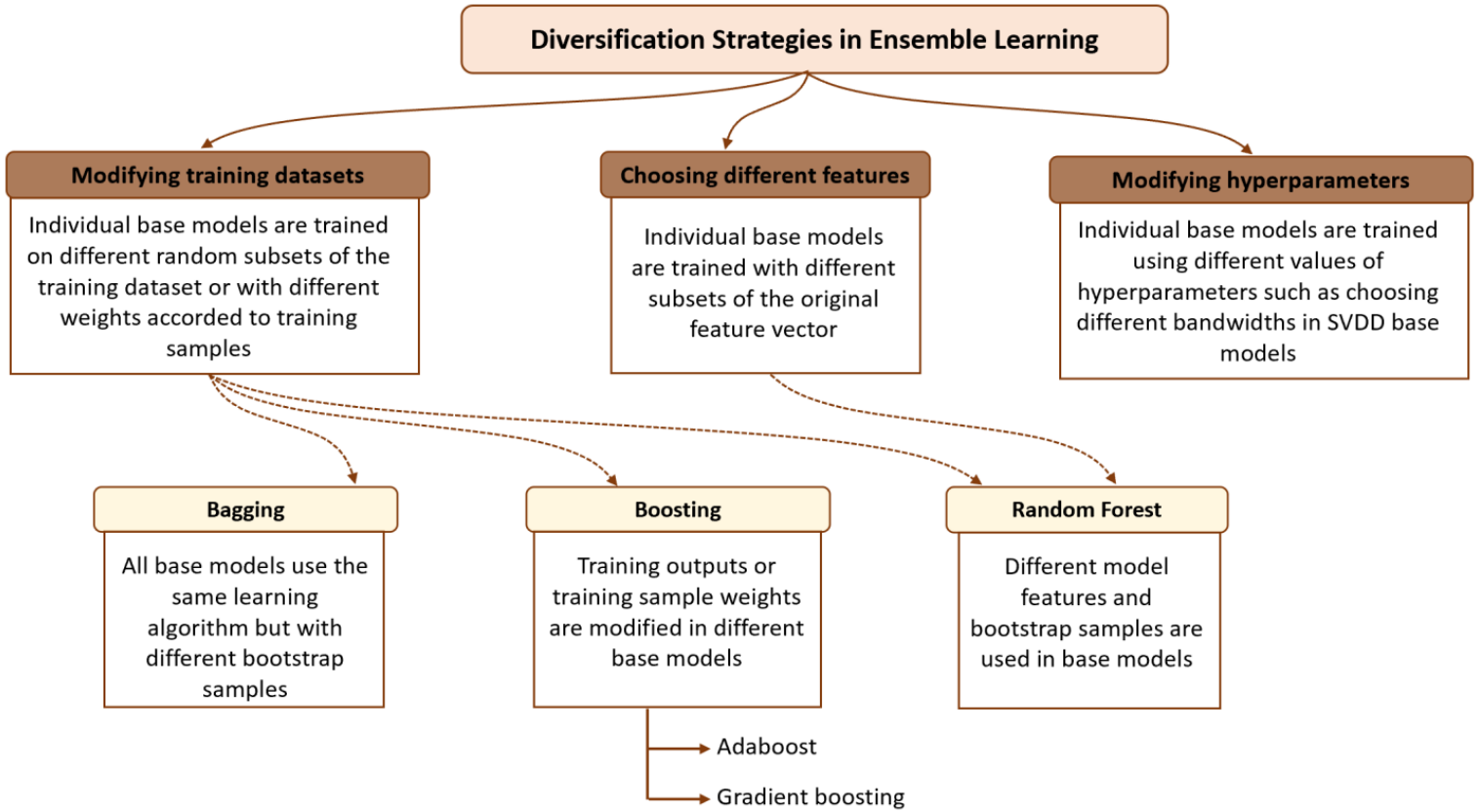
POR QUE USAR?

- Reduz a variância: diminui o risco de *overfitting*.
- Reduz o viés: melhora a capacidade preditiva diminuindo o risco de *underfitting*.
- Aumenta a robustez: menos sensível a *outliers* e ruído nos dados.
- Melhora a generalização: melhor desempenho em dados não vistos.

HETEROGENEOUS ENSEMBLE LEARNING



HOMOGENEOUS BASE MODELS



BAGGING (BOOTSTRAP AGGREGATING)

- O Bagging gera diversos modelos de base a partir de diferentes subconjuntos (com reposição) do conjunto de dados original.
- Diferente do RF, as variáveis de entrada permanecem fixas, e não há aleatoriedade na seleção de atributos.
- O modelo base não precisa ser uma árvore de decisão.
- As predições dos modelos são combinadas (média ou votação).
- Os modelos são treinados de forma independente e paralela.
- Se os dados já forem amostrados de forma independente, *bootstrapping* não é necessário.

BAGGING (BOOTSTRAP AGGREGATING)

- Reduz variância, mas não reduz viés.
- Se todos os modelos estiverem altamente correlacionados ($\bar{\rho} = 1$), o *ensemble* não terá ganho.

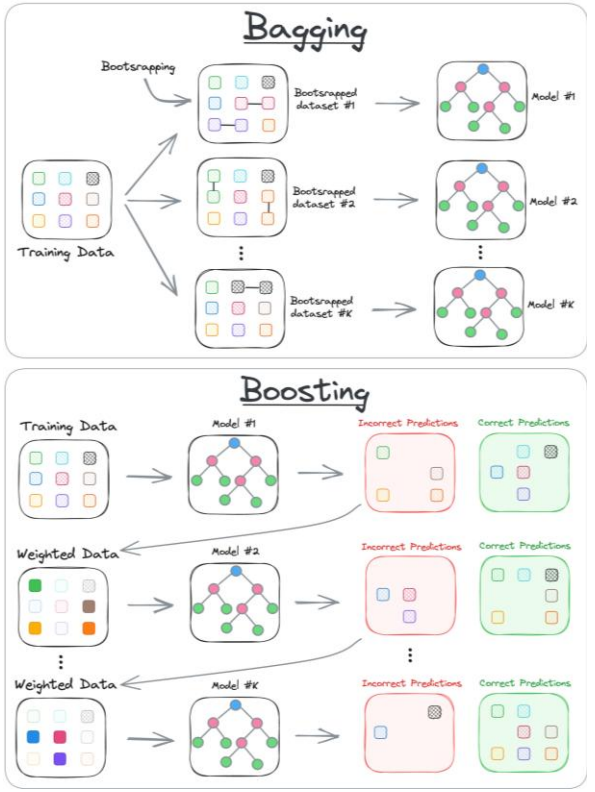
$$\sigma^2_{ensemble} = \sigma^2 \left(\bar{\rho} + \frac{1 - \bar{\rho}}{K} \right)$$

- $\bar{\rho}$: correlação média entre os modelos base;
- K: número de modelos.

BOOSTING

- Enquanto o *bagging* treina os modelos em paralelo, o *boosting* os treina sequencialmente, de modo que cada novo modelo corrige os erros cometidos pelo anterior.
- É uma técnica mais poderosa quando o modelo base sofre de alto viés ou *underfitting*.

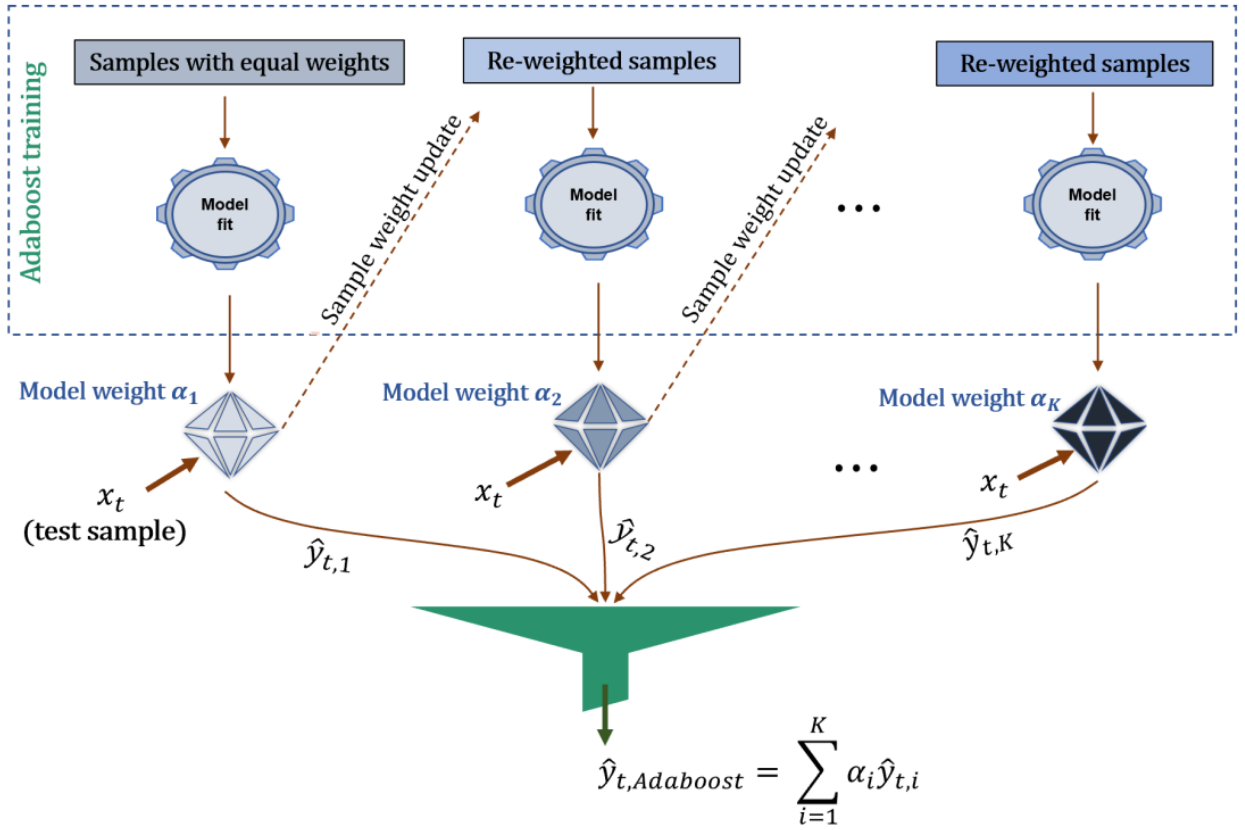
BAGGING X BOOSTING



ADABOOSTING (*ADAPTIVE BOOSTING*)

- *AdaBoost* (*Adaptive Boosting*) é uma técnica de *ensemble learning* baseada em *boosting*, onde vários modelos fracos são combinados de forma sequencial para formar um modelo forte.
- O método adapta-se ao erro dos modelos anteriores, focando nos exemplos que foram classificados incorretamente.
- Inicialmente, todos os exemplos têm o mesmo peso.
- O primeiro modelo é treinado.
- As amostras mal classificadas recebem mais peso.
- O segundo modelo é treinado com foco nessas amostras difíceis.
- O processo continua até um número K de modelos.

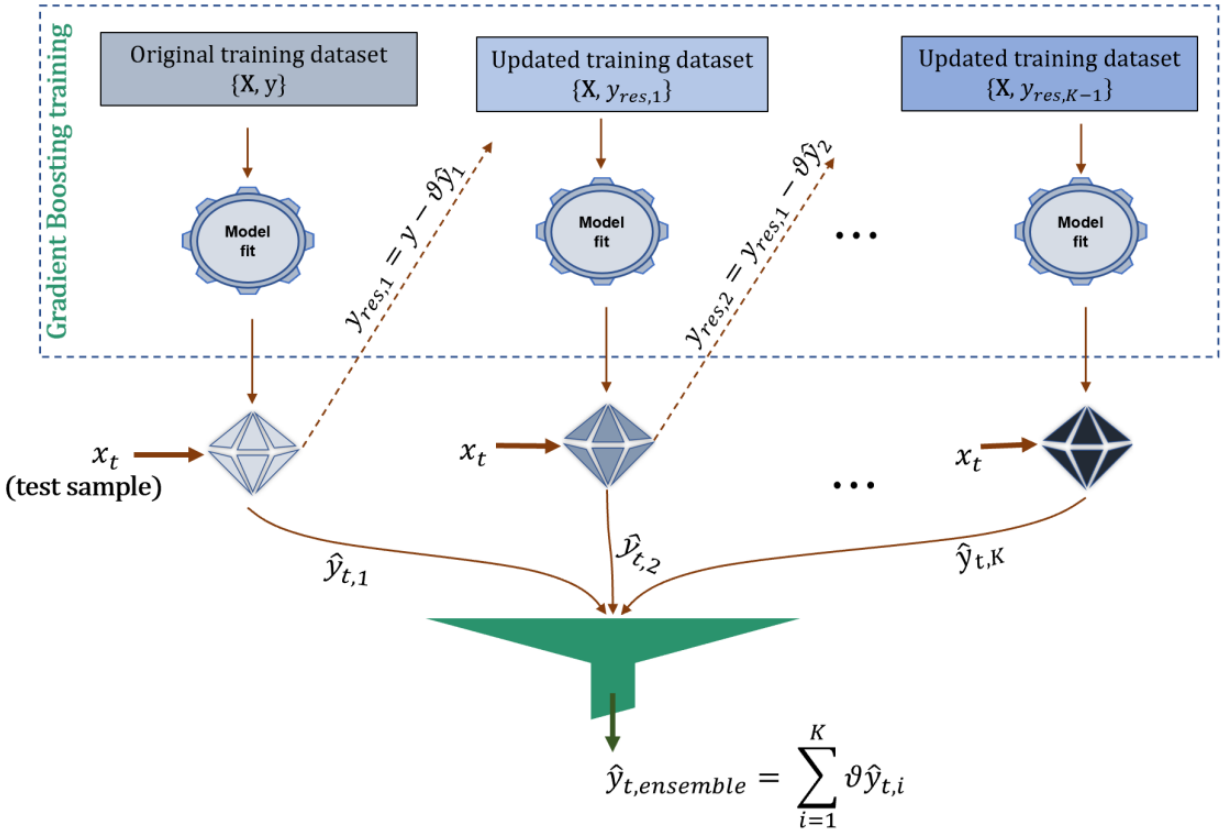
ADABOOSTING (ADAPTIVE BOOSTING)



GRADIENT BOOSTING

- *Gradient Boosting* é uma técnica de *boosting* mais generalizada que também combina modelos fracos sequencialmente, mas baseada na ideia de gradiente descendente para minimizar a função de perda.
- Treina-se o primeiro modelo para prever a variável alvo.
- Calcula-se o erro residual (diferença entre a previsão e o valor real).
- Treina-se o próximo modelo para ajustar esse erro → modela o gradiente da função de perda.
- O processo se repete, com cada modelo corrigindo os erros do modelo anterior.
- A previsão final é a soma ponderada de todas as previsões anteriores.

GRADIENT BOOSTING



PREDIÇÃO COM *XGBOOST*

O QUE É XGBOOST?

- *XGBoost* (*Extreme Gradient Boosting*) é uma biblioteca e algoritmo de aprendizado de máquina baseado em *Gradient Boosting*, altamente otimizado para velocidade e performance.
- Foi desenvolvido para ser eficiente, flexível e portável, sendo atualmente um dos algoritmos mais populares e poderosos para competição e produção.
- Constrói modelos de forma sequencial, onde cada nova árvore tenta corrigir os erros da soma das árvores anteriores, usando o gradiente da função de perda.
- Suporta regressão, classificação, ranking e predição de séries temporais.

COMO FUNCIONA?

- Inicializa o modelo com uma previsão constante (por exemplo, a média dos valores).
- Para cada iteração:
  - Calcula os gradientes (erros) da função de perda.
  - Constrói uma nova árvore de decisão para ajustar esses erros.
  - Atualiza a previsão somando a saída ponderada da nova árvore.
- Repete o processo até atingir o número máximo de árvores ou até que os ganhos sejam mínimos.

POR QUE É TÃO EFICIENTE?

- *Shrinkage* (taxa de aprendizado): reduz o impacto de cada árvore para melhorar a generalização.
- *Subsampling*: amostragem parcial de linhas ou colunas → aumenta a diversidade e evita *overfitting*.
- *Pruning* (poda): elimina automaticamente ramos não produtivos das árvores.
- Paralelização e uso eficiente da memória.

VANTAGENS E DESVANTAGENS

- Vantagens:
  - Altamente preciso.
  - Rápido e eficiente.
  - Suporta paralelização.
  - Lida bem com dados esparsos.
  - Flexível → suporta diferentes funções de perda.
- Desvantagens:
  - Pode ser complexo de ajustar → muitos hiperparâmetros.
  - Menor interpretabilidade que modelos mais simples.
  - Exige cuidado para evitar *overfitting*, principalmente com dados pequenos.

XGBOOST EM SOFT-SENSING

- Em processos industriais, como químicos, petroquímicos ou ambientais, frequentemente há múltiplas variáveis observáveis que são indiretamente correlacionadas com variáveis críticas, que queremos estimar.
- Modela relações não lineares com alta precisão.
- Robusto a ruídos e dados incompletos.
- Pode trabalhar com grande volume de dados.
- Oferece mecanismos de regularização que evitam *overfitting* — problema comum em modelos de *soft-sensing*.
- Melhora o controle de processos.
- Reduz custos operacionais com instrumentação.
- Aumenta a segurança → evita medições perigosas.
- Fornece dados em tempo real para tomada de decisão.

CONCLUSÃO

CONCLUSÃO

- Este capítulo mostrou a evolução da modelagem preditiva, da Árvore de Decisão até métodos avançados como *Ensemble Learning*.
- Técnicas como *Random Forest* e *Boosting* aumentam a robustez e precisão dos modelos, com destaque para o *XGBoost*, amplamente utilizado na indústria e pesquisa.
- O *Soft Sensing* foi ressaltado como aplicação prática, promovendo eficiência, segurança e redução de custos.
- Assim, dominar essas técnicas é fundamental para profissionais que buscam soluções preditivas eficientes e atuais.