

Foto de master1305,
disponível no FreePik.
Adaptada pelo autor.

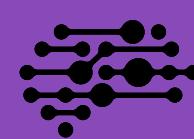
Introdução a Machine Learning

**Módulo 2 - Métodos de Aprendizado
de Máquina e suas aplicações**

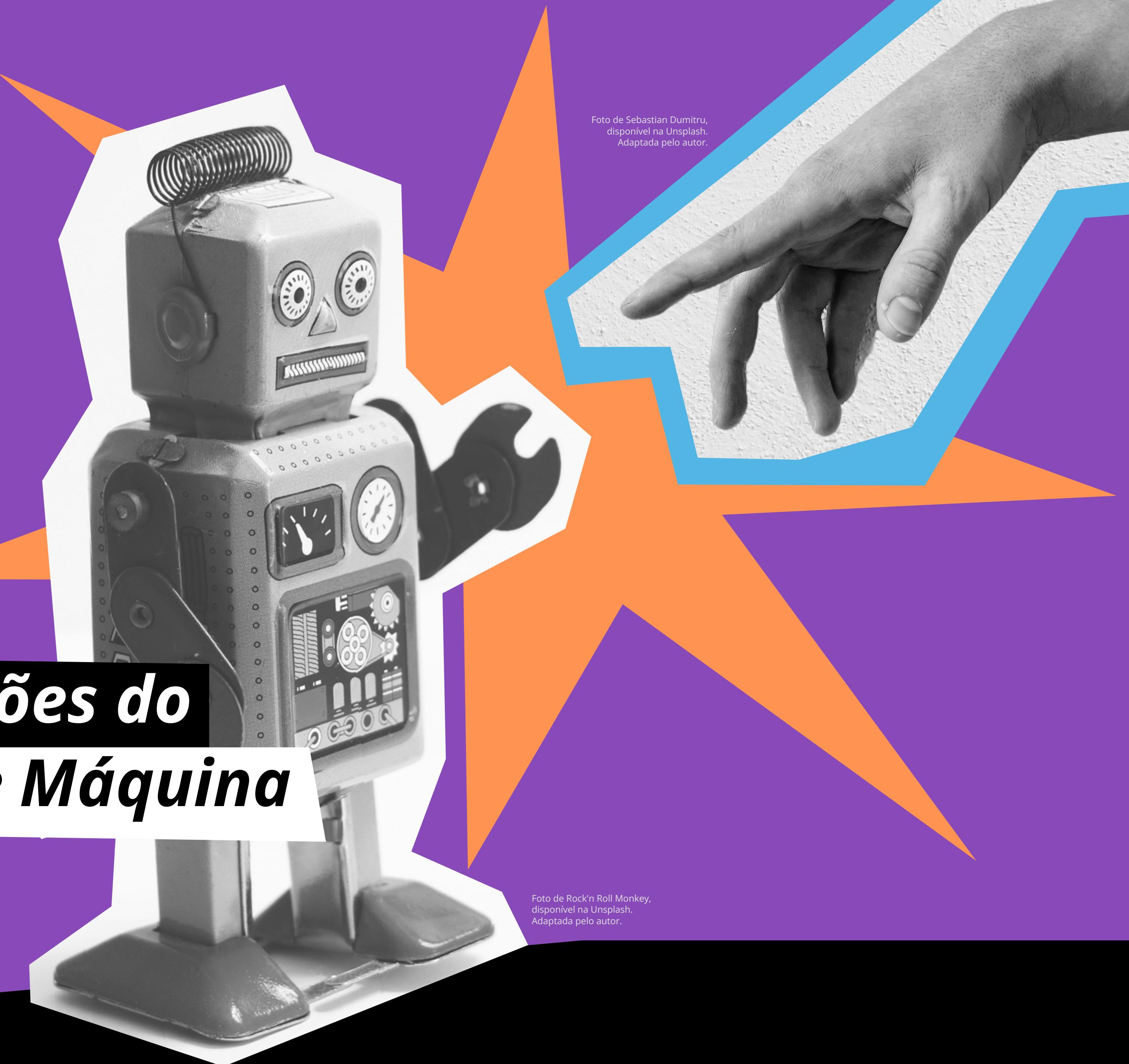


Sumário

| | |
|---|-----------|
| Aula 1 - Aplicações do Aprendizado de Máquina | 1 |
| Aula 2 - Modelos | 3 |
| 2.1 Treinando modelos..... | 5 |
| Aula 3 - Problemas de Aprendizado de Máquina | 7 |
| 3.1 Classificação..... | 9 |
| 3.2 Regressão Linear | 10 |
| 3.3 Agrupamento..... | 12 |
| Aula 4 - Outros modelos | 13 |
| 4.1 Máquina de Vetores de Suporte (<i>Support-Vector Machine</i>) | 14 |
| 4.2 Árvore de Decisão | 16 |
| 4.3 <i>Naive Bayes</i> | 19 |
| 4.4 <i>K-Nearest Neighbors</i> | 20 |
| 4.5 <i>K-means</i> | 22 |
| Explore mais! | 24 |
| Referências Bibliográficas..... | 25 |



Aula 1 - Aplicações do Aprendizado de Máquina



Aprendizado de Máquina é uma área da inteligência artificial que estuda como os computadores podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana, isto é, **sem serem explicitamente programados para isso**. Essa capacidade de aprender a partir de exemplos é o que diferencia o Aprendizado de Máquina de outras técnicas de programação tradicionais.

O Aprendizado de Máquina é uma forma de programação que permite aos computadores também se adaptar às novas situações. É diferente, por exemplo, dos algoritmos tradicionais de computação, que seguem instruções fixas e pré-definidas para resolver problemas específicos. Os algoritmos de Aprendizado de Máquina podem ser usados para tarefas complexas, dinâmicas ou que envolvem grandes quantidades de dados, como reconhecimento de voz, detecção de fraudes, recomendação de produtos, etc.

Eles também podem melhorar seu desempenho com o tempo, à medida que recebem mais dados e *feedback*. Por essas razões, os algoritmos de Aprendizado de Máquina podem oferecer vantagens sobre os algoritmos tradicionais de computação em muitos domínios e aplicações.



Ilustração feita por upklyak,
disponível no Freepik.
Adaptada pelo autor.

Alguns exemplos de empresas que utilizam Aprendizado de Máquina são:

- **Google Fotos:** reconhecer rostos, objetos e lugares nas fotos e organizar as imagens por categorias;
- **Netflix:** recomendar filmes e séries aos usuários com base em seus gostos e preferências;
- **Spotify:** criar *playlists* personalizadas para os usuários com base em suas músicas favoritas;
- **OpenAI:** responder perguntas em linguagem natural e auxiliar na tomada de decisões através do ChatGPT;
- **DeepMind:** jogar o jogo de tabuleiro *Go*, atingindo resultados melhores que jogadores profissionais usando a inteligência artificial AlphaGo.

Aula 2 - Modelos

Foto de Robina Weermeijer,
disponível na Unsplash.
Adaptada pelo autor.

Foto de La-Rel Easter,
disponível na Unsplash.
Adaptada pelo autor.

Um dos conceitos fundamentais do Aprendizado de Máquina é o de modelo. Um modelo nada mais é do que uma representação matemática de um fenômeno ou processo real, que pode ser usado para fazer previsões, também chamado de **inferências**, sobre dados novos ou desconhecidos.

Um modelo de Aprendizado de Máquina pode ser visto como uma função que mapeia entradas (características) e saídas (rótulos ou valores).

Por exemplo, se quisermos prever o preço de uma casa a partir de suas características, como:

- Área;
- Número de quartos;
- Localização.



Podemos usar um modelo de Aprendizado de Máquina que relaciona essas variáveis com o preço. Um modelo pode ter diferentes formas e complexidades, dependendo do problema e dos dados disponíveis.

De modo prático, podemos definir um modelo de Aprendizado de Máquina como um **algoritmo que aprende a executar uma tarefa a partir de um conjunto de dados**. O conceito de modelo está relacionado com o conceito de treino, que é o processo pelo qual o modelo ajusta os seus parâmetros para tomar boas decisões quando for fazer previsões utilizando dados novos.

2.1 Treinando modelos

Treino, em *Machine Learning*, é o processo de ensinar um algoritmo, ou modelo, a aprender a partir de dados. O algoritmo recebe um conjunto de dados de entrada, chamado de **dados de treino**, e usa esses dados para ajustar seus parâmetros internos, chamados de pesos.

O objetivo é que o algoritmo consiga generalizar o que aprendeu para novos dados que não foram usados no treino, chamados de **dados de teste**. Existem diferentes formas de treinar um modelo de Aprendizado de Máquina, dependendo do tipo de tarefa e dos dados disponíveis.

Algumas das formas mais comuns são:

- **Aprendizado supervisionado:** o conjunto de dados contém exemplos de entradas e saídas esperadas, e o modelo tenta aprender a mapear as entradas nas saídas. Por exemplo, um modelo que classifica comentários como positivos, neutros ou negativos a partir do texto;
- **Aprendizado não supervisionado:** o conjunto de dados contém apenas exemplos de entradas, e o modelo tenta encontrar padrões ou estruturas nos dados sem ter uma saída pré-definida. Por exemplo, um modelo que agrupa clientes com base no seu histórico de compras;

- **Aprendizado semi-supervisionado:** o conjunto de dados contém alguns exemplos de entradas e saídas, e outros exemplos apenas de entradas. O modelo tenta aproveitar as informações dos dois tipos de exemplos para aprender a tarefa. Por exemplo, um modelo que classifica imagens em categorias usando algumas imagens rotuladas e outras não rotuladas;

- **Aprendizado por reforço:** o modelo interage com um ambiente e recebe recompensas ou punições pelas suas ações. O modelo tenta aprender a maximizar a recompensa total ao longo do tempo. Por exemplo, um modelo que aprende a jogar xadrez contra um adversário.

Para utilizarmos conjuntos de dados no treino de algoritmos de Aprendizado de Máquina, devemos representá-los matematicamente de forma adequada. Uma forma comum de fazer isso é transformar os dados em vetores numéricos, que podem ser manipulados por operações matemáticas.

Por exemplo: se temos um conjunto de dados sobre flores, podemos representar cada flor por um vetor que contém os valores de seus atributos, como comprimento e largura das pétalas e da sépala. Assim, podemos usar esses vetores para calcular distâncias, ângulos, médias e outras medidas que nos ajudam a entender as características e os padrões dos dados. Além disso, podemos aplicar técnicas de pré-processamento, como normalização, padronização, redução de dimensionalidade e seleção de atributos, para melhorar a qualidade e a eficiência dos dados para o treino dos algoritmos.

Alguns dos principais conceitos envolvidos no treinamento de um algoritmo de Aprendizado de Máquina são:

- **Função loss:** a função *loss*, ou função custo, é um dos conceitos mais importantes durante o treino de um modelo, é a função que mede quanto bem o algoritmo se ajusta aos dados. Quanto menor o valor da função custo, melhor o modelo (mais próximo dos valores corretos ele está). A função *loss* utilizada varia e depende do tipo de problema que se quer resolver, como classificação, regressão ou clusterização, sendo calculada de acordo com o cenário do treino;
- **Gradiente descendente:** o gradiente descendente é um algoritmo iterativo que busca o mínimo de uma função custo, que mede quanto bem o modelo se ajusta aos dados. O algoritmo funciona atualizando os parâmetros do modelo em pequenos passos na direção oposta ao gradiente da função custo, que indica a direção de maior aumento da função. O algoritmo termina quando o gradiente é próximo de zero ou quando um número máximo de iterações é atingido;
- **Overfitting:** é quando o algoritmo se ajusta demais aos dados de treinamento e perde a capacidade de generalizar para novos dados. Isso pode acontecer quando o modelo é muito complexo ou quando os dados de treinamento são insuficientes ou ruidosos;
- **Underfitting:** é quando o algoritmo pouco se ajusta aos dados de treinamento e não consegue capturar as relações entre as variáveis. Ao contrário do overfitting, o underfitting pode acontecer quando o modelo é muito simples ou quando os parâmetros do modelo não são otimizados adequadamente;

- **Ajuste de parâmetros:** é o processo de encontrar os melhores valores para os parâmetros do modelo, que minimizam a função *loss* e maximizam a acurácia das previsões. Existem diferentes métodos para fazer o ajuste de parâmetros, como busca em grade, busca aleatória, validação cruzada e gradiente descendente.

Por exemplo, se usarmos um modelo linear para prever o preço da casa, como $y = ax + b$, onde y é o preço, x é a área e a e b são os parâmetros a serem ajustados, podemos usar um método chamado mínimos quadrados para encontrar os valores de a e b que minimizam a soma dos quadrados dos erros entre os preços reais e os preços previstos pelo modelo. Veremos mais adiante como modelos lineares são treinados e utilizados para realizar inferências.

Geralmente, treinamos nossos modelos utilizando apenas 70% a 80% do total dos dados disponíveis. O restante deve ser utilizado para testar se o novo modelo que foi treinado consegue se sair bem ao lidar com dados não vistos anteriormente. Essa estratégia de validação é extremamente importante para garantir que o modelo não sofreu *under* — ou *overfitting*, e se está atingindo as métricas de performance desejadas, generalizando bem para novos dados.

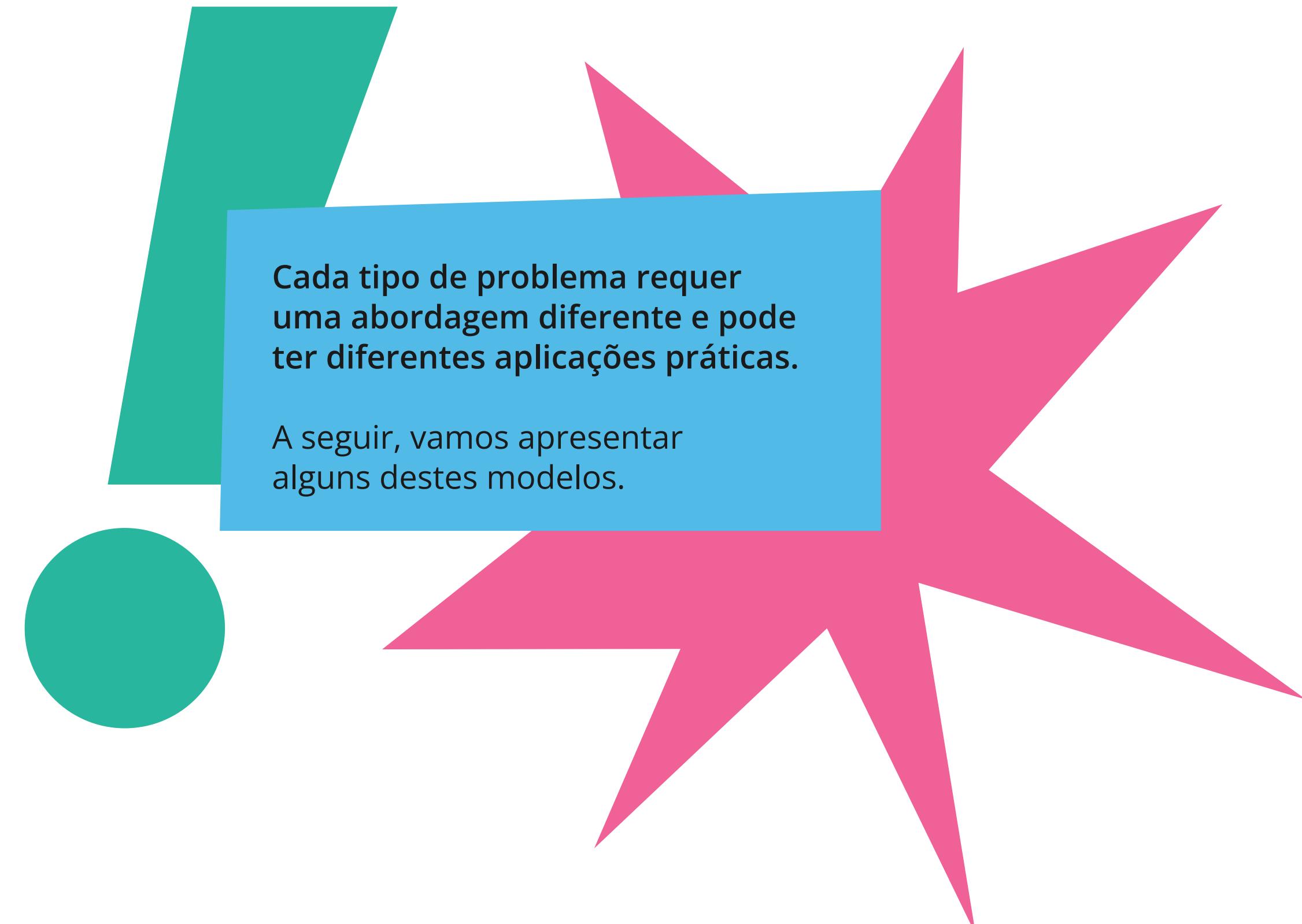
Aula 3 - Problemas de Aprendizado de Máquina



Os problemas de Aprendizado de Máquina podem ser classificados em diferentes tipos, de acordo com a natureza dos dados, das tarefas e das técnicas envolvidas. Dois tipos comuns de problemas de aprendizado de máquina são:

- **Classificação:** é o problema de atribuir um rótulo ou uma categoria a uma observação, com base em um conjunto de características. Por exemplo, classificar um *e-mail* como *spam* ou não *spam*, ou classificar uma imagem como um gato ou um cachorro. A classificação pode ser binária, quando há apenas dois rótulos possíveis, ou multiclasse, quando há mais de dois rótulos possíveis;
- **Regressão:** é o problema de estimar uma variável contínua ou numérica a partir de um conjunto de características. Por exemplo, estimar o preço de uma casa, o salário de uma pessoa ou a demanda por um produto. A regressão pode ser linear, quando há uma relação linear entre as características e a variável alvo, ou não linear, quando essa relação é mais complexa;

Além da classificação e da regressão, existem outros tipos de problemas de aprendizado de máquina, como: Agrupamento, Associação, Detecção de Anomalias e Aprendizado por reforço.



Cada tipo de problema requer uma abordagem diferente e pode ter diferentes aplicações práticas.

A seguir, vamos apresentar alguns destes modelos.

3.1 Classificação

Classificação é uma tarefa de Aprendizado de Máquina supervisionado que consiste em prever a qual de duas ou mais classes (categorias) uma instância de dados pertence. Por exemplo, dado um conjunto de dados sobre pacientes com sintomas de uma doença, um algoritmo de classificação pode aprender a identificar quais pacientes estão doentes e quais não estão, com base em características como idade, peso, temperatura, etc.

Essas características são chamadas de **variáveis preditoras ou atributos**, e a classe que se quer prever é chamada de variável resposta ou rótulo. O algoritmo de classificação usa um conjunto de dados de treinamento, que são exemplos rotulados com a classe correta, para construir um modelo que possa generalizar para novos dados não rotulados.

Existem vários tipos de algoritmos de classificação, como, por exemplo:

- Regressão linear e logística;
- Máquinas de vetores de suporte;
- Árvores de decisão;
- Redes neurais.

Cada um tem suas vantagens e desvantagens, e a escolha do melhor algoritmo depende do problema e dos dados disponíveis.

3.2 Regressão Linear

A Regressão Linear é um método estatístico que permite estudar a relação entre uma variável dependente (ou resposta) e uma ou mais variáveis independentes (ou preditoras). O objetivo da Regressão Linear é encontrar uma equação linear que descreva o melhor possível essa relação, ou seja, que minimize a diferença entre os valores observados e os valores estimados pela equação. A equação da Regressão Linear tem a seguinte forma (figura 1):

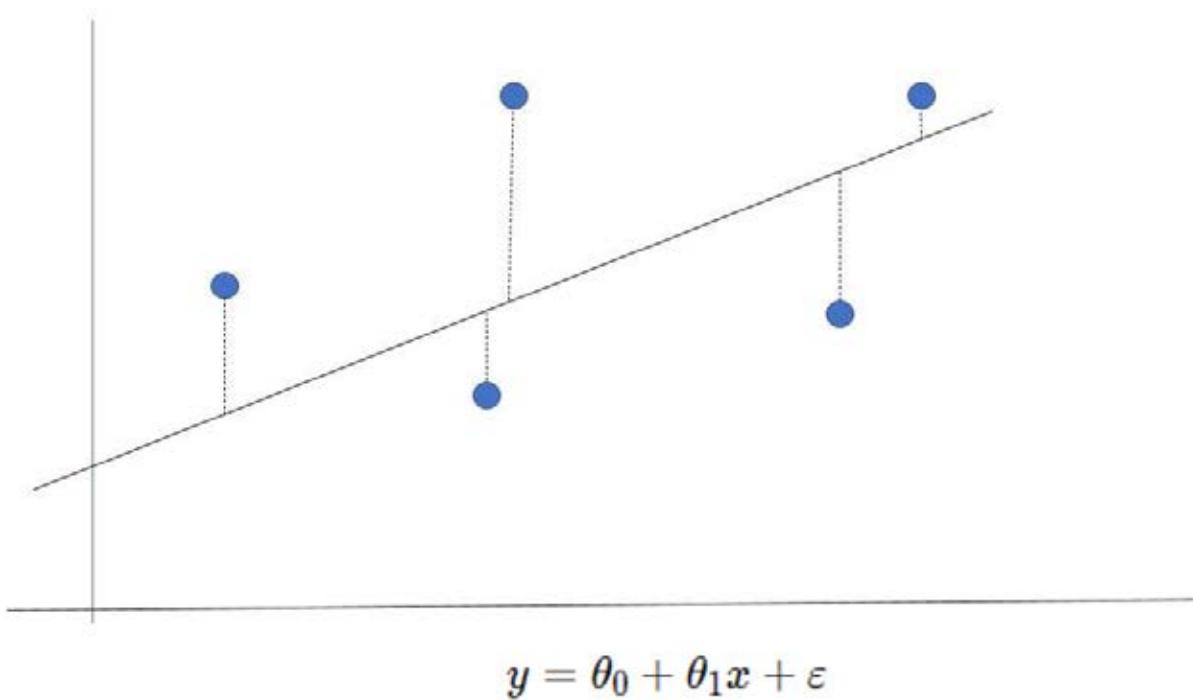


Figura 1 – Forma e equação de Regressão Linear. Fonte: feita pelo autor.

Como visto no gráfico anterior (figura 1), o objetivo da Regressão Linear é **obter uma reta que melhor se ajusta aos dados de treinamento**, minimizando o erro (distância entre os pontos e a reta, representada pelas linhas tracejadas) por meio da redução da função *loss*.

Existem vários métodos para estimar os coeficientes da regressão, sendo o mais comum o método dos mínimos quadrados ordinários (MQO), que consiste em minimizar a soma dos quadrados dos erros. Podemos utilizar também o gradiente descendente para encontrar os pesos que levam ao melhor "fit" do modelo aos dados, isto é, minimiza o erro.

Por ser um modelo de baixa complexidade, a Regressão Linear costuma não obter bons resultados quando estamos lidando com conjuntos de dados nos quais as variáveis não apresentam uma certa relação linear. É possível perceber a seguir (figura 2) que o modelo apresenta um alto erro entre o valor predito (reta) e o correto (pontos) em diversos casos, principalmente devido à dispersão dos pontos e a falta de correlação linear entre eles.

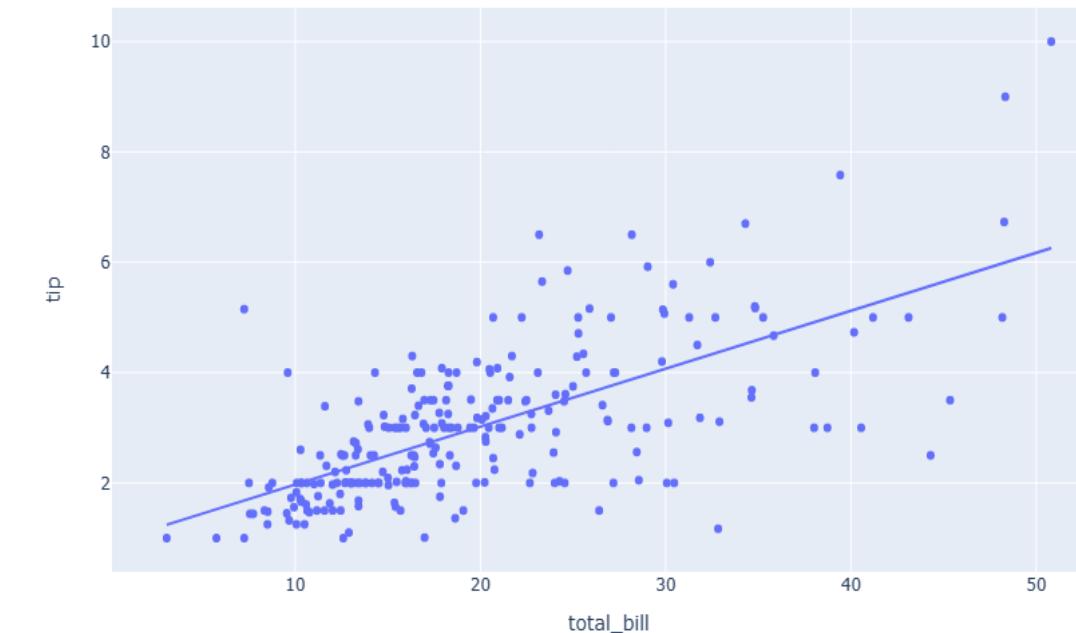


Figura 2 – Gráfico variáveis sem Relação Linear. Fonte: feita pelo autor.

Da mesma forma, se os dados seguem uma distribuição exponencial, logarítmica ou polinomial, a Regressão Linear (reta vermelha) não será capaz de capturar adequadamente o padrão dos dados e produzir uma boa estimativa da variável dependente. Nesses casos, é preferível utilizar outros modelos de regressão que se ajustem melhor à natureza dos dados, como a regressão não linear ou a regressão logística, como apresentado na linha azul (figura 3).

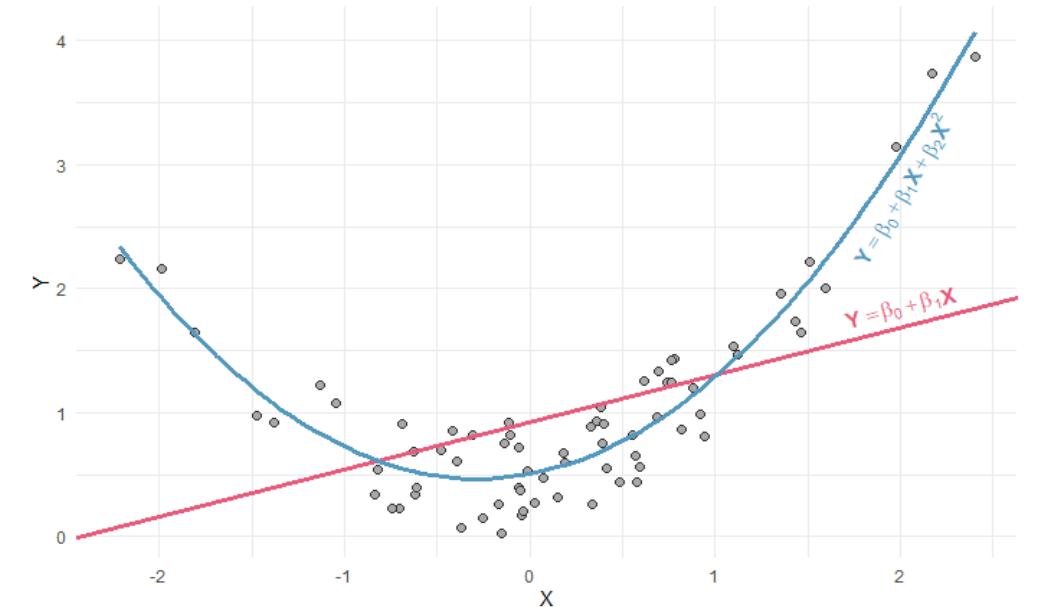


Figura 3 – Gráfico variáveis não linear ou logística. Fonte: feita pelo autor.

3.3 Agrupamento

Agrupamento (ou *clustering*) é uma técnica de Aprendizado de Máquina que visa encontrar grupos de objetos similares em um conjunto de dados.

O objetivo é que as instâncias dentro de um mesmo grupo sejam mais parecidas entre si do que com os objetos de outros grupos.

Por exemplo, se temos um conjunto de dados sobre clientes de uma loja, podemos usar Agrupamento para identificar segmentos de mercado baseados em características como idade, renda, preferências de compra, etc. Desse modo, a clusterização nos possibilita direcionar melhor nossas estratégias de *marketing* e atendimento para cada segmento.



Aula 4 - Outros modelos



Foto de Jason Leung,
disponível na Unsplash.
Adaptada pelo autor.



Foto disponível no
FreePik. Adaptada
pelo autor.

4.1 Máquina de Vetores de Suporte (Support-Vector Machine)

Uma Máquina de Vetores de Suporte (SVM, do inglês *Support Vector Machine*) é um Modelo de Aprendizado supervisionado que pode ser aplicado para resolver tarefas de classificação e regressão. Para classificação, a SVM analisa os dados e encontra um hiperplano que separa as classes-alvo com a maior margem possível. Um hiperplano é uma superfície de dimensão **n-1** que divide um espaço de dimensão **n** em duas partes. Por exemplo, em um espaço bidimensional, um hiperplano é uma reta; em um espaço tridimensional, o hiperplano definido pela SVM é um plano.

Assumimos que o número de dimensões é igual ao número de **features** do nosso conjunto de dados que estamos levando em consideração durante o treinamento. Por exemplo, um *dataset* que contém quatro características de cada casa armazenadas (número de quartos, área e número de banheiros e valor, por exemplo) é mapeado em um espaço de quatro dimensões, considerando que todas as features serão utilizadas no treino. Logo, para treinarmos uma SVM nesse conjunto de dados teremos um hiperplano de ordem **n-1**, no caso, ordem três.

Vemos a seguir (figura 4) a representação de um conjunto de dados qualquer em um espaço de duas dimensões. Logo, nossa SVM será a reta tracejada (uma dimensão) que consegue melhor dividir as duas classes representadas por pontos verdes e laranjas. Veremos a seguir como esse algoritmo define o hiperplano que melhor realiza essa separação.

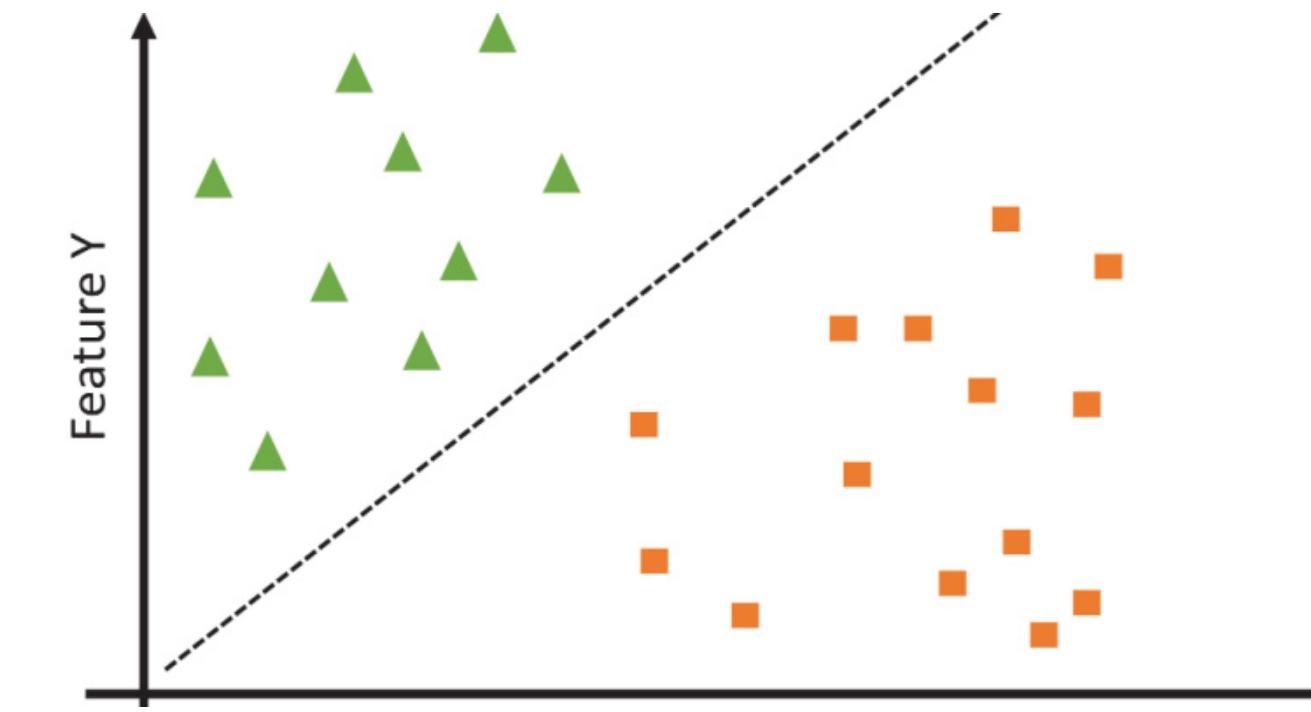


Figura 4 – Conjunto de dados em duas dimensões⁴.

Nos gráficos a seguir (figuras 5 e 6), vemos diversos hiperplanos que tentam separar as classes de quadrados e bolas em diversas formas. Para encontrar o hiperplano ideal, precisamos utilizar outra definição importante para SVMs: **os vetores de suporte**. No gráfico à direita (figura 6) vemos o hiperplano que maximiza a distância entre as classes, com base nos vetores de suporte (quadrados e bolas preenchidas).

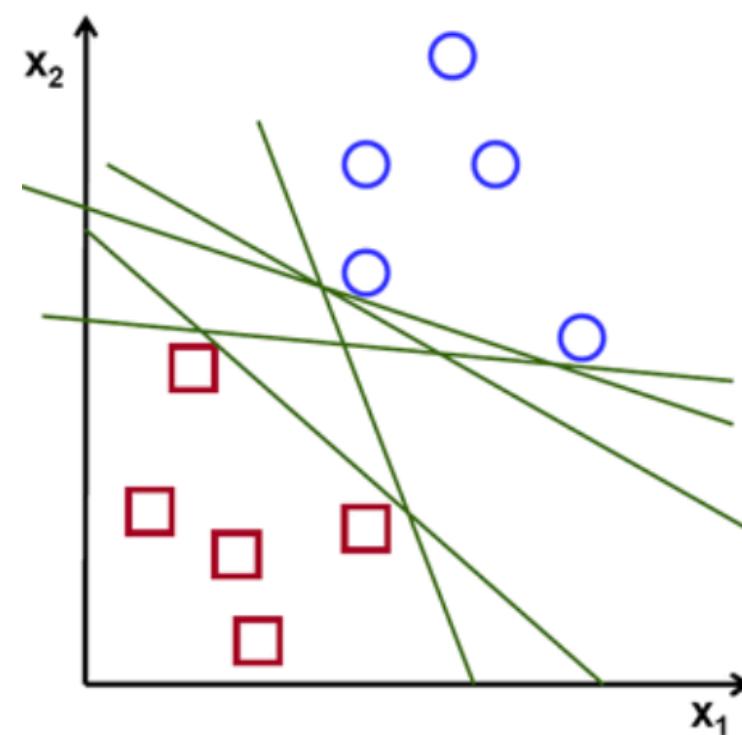


Figura 5 – Conjunto de dados em duas dimensões⁵.

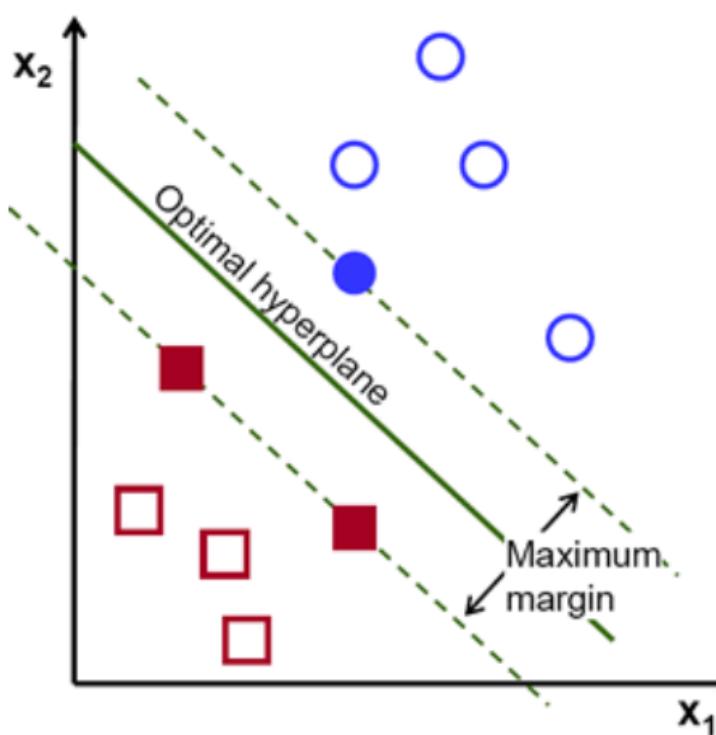


Figura 6 – Conjunto de dados em duas dimensões⁵.

dimensão e usa apenas um subconjunto dos pontos de treinamento na Função de Decisão (chamados vetores de suporte), o que a torna também eficiente em termos de memória. Um ponto negativo da SVM é que ela pode ser sensível a ruídos e *outliers*, ou seja, dados que estão fora do padrão esperado. Em conjuntos de dados mais robustos, a SVM também exige um alto custo computacional para treinar e ajustar os parâmetros adequados.

O hiperplano ótimo é encontrado pela solução de um problema de otimização que minimiza a norma do vetor normal ao plano, sujeito à restrição de que todos os pontos sejam classificados corretamente. Os pontos mais próximos do hiperplano são chamados de vetores de suporte, pois eles suportam o hiperplano e determinam sua posição e orientação. Os vetores de suporte são os dados mais relevantes para o treinamento da SVM, pois são os que determinam a posição e a orientação da fronteira de decisão. Os demais dados não influenciam na construção do modelo, apenas servem para validar sua precisão.

Uma SVM pode ser aplicada em diversos domínios, como reconhecimento de imagens, detecção de *spam*, diagnóstico médico, análise de sentimentos, etc. O principal benefício de uma SVM é que ela é eficaz em espaços de alta

⁵Fonte: Máquinas de vetores de suportes SVM: introdução aos algoritmos de aprendizado de máquina. Alvarez Soluções Digitais, 2020. Disponível em: <https://alvarezsolucoesdigitais.com/aprendizado-de-maquina/maquinas-de-vetores-de-suportes-svm-introducao-aos-algoritmos-de-aprendizado-de-maquina/>. Acesso em: maio de 2023.

4.2 Árvore de Decisão

Outro algoritmo clássico em Aprendizado de Máquina é a Árvore de Decisão, que também pode ser usada para resolver problemas de classificação. Uma Árvore de Decisão é composta por nós que representam perguntas ou condições sobre os dados, e ramos que representam as possíveis respostas ou resultados. O objetivo é construir uma Árvore que consiga separar os dados em grupos homogêneos, de acordo com a variável alvo (a classe ou o valor que se quer prever). Dessa forma, a Árvore de Decisão é um modelo de aprendizado de máquina que usa uma estrutura hierárquica de regras para classificar e prever dados.

Para construir uma Árvore de Decisão, é preciso escolher qual atributo dos dados será o nó raiz, ou seja, a primeira pergunta ou condição que será aplicada aos dados. Em seguida, é preciso escolher quais atributos serão os nós filhos, ou seja, as perguntas ou condições subsequentes que serão aplicadas aos subconjuntos de dados resultantes do nó raiz. Esse processo se repete até que se chegue aos nós-folha, que são os resultados finais da Árvore.

O esquema a seguir (figura 7) apresenta um exemplo simples de Árvore de Decisão que classifica espécies de animais. Vemos que a cada camada da árvore, mais discriminado os subconjuntos ficam através de perguntas binárias (verdadeiro ou falso), até que o resultado final (definição da espécie) seja alcançado.

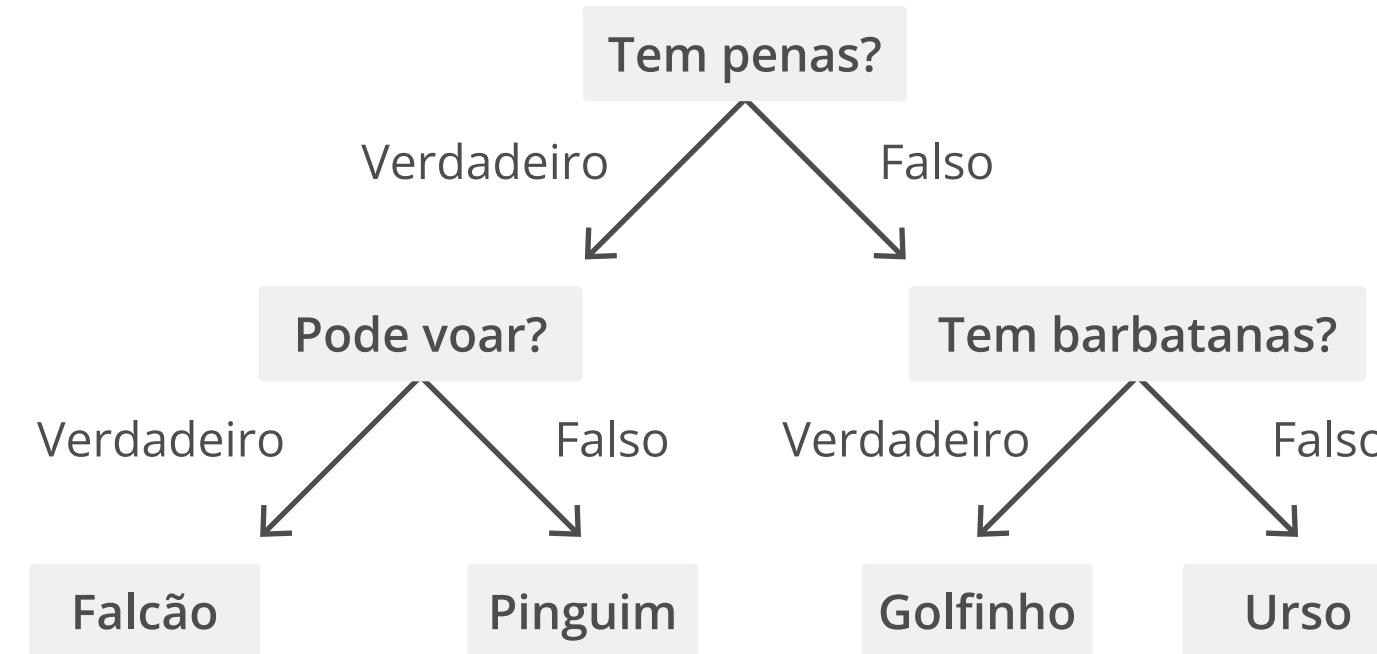


Figura 7 – Árvore de Decisão. Fonte: feita pelo autor.

Para escolher os atributos que serão os nós da Árvore, existem diferentes algoritmos que usam medidas como o ganho de informação e a entropia, que servem para indicar o quanto um atributo consegue reduzir a incerteza ou a desordem dos dados. Quanto maior o ganho de informação e menor a entropia, melhor é o atributo para ser um nó da Árvore. Em outras palavras, gostaríamos sempre de escolher os nós para a Árvore que melhor discriminem os dados analisados, de modo a gerar uma Árvore com menor profundidade.

Por fim, a função de custo mais comum usada para avaliar a qualidade de uma Árvore de Decisão é o erro de classificação, que é a proporção de instâncias que foram classificadas incorretamente pela Árvore. Quanto menor o erro de classificação, melhor é a Árvore.

As Árvores de Decisão são altamente interpretáveis, pois tem uma estrutura visual e lógica simples, são capazes de ligar com dados numéricos e categóricos, além de dados incompletos e/ou ausentes. Além disso, como o

modelo de Árvore de Decisão subdivide o conjunto em diversos subconjuntos menores, ruídos e outliers também são levados em consideração no momento do treinamento.

Por outro lado, Árvores de Decisão tendem a ser instáveis, ou seja, mudar muito sua estrutura com pequenas variações nos dados de treinamento. Isso pode afetar a confiabilidade e a consistência do algoritmo quando inserimos novos exemplos. Existem algumas técnicas avançadas como *bagging* e *boosting* que servem para mitigar a instabilidade inerente dos modelos de Árvore de Decisão.

4.2.1 Random Forest

Random Forest é um método de aprendizado de máquina que combina várias árvores de decisão para fazer previsões ou classificações. Para isso, cada árvore de decisão é treinada com um subconjunto aleatório dos dados e das variáveis, o que aumenta a diversidade e reduz o risco de *overfitting*. A previsão ou classificação final é feita pela média ou pela votação das previsões ou classificações de todas as árvores.

Veja a representação, de forma simplificada, como uma *Random Forest* seleciona a classe final da fruta (figura 8). Em uma *Random Forest*, várias árvores de decisão são criadas a partir do conjunto de dados de treinamento,

cada uma com uma seleção aleatória de variáveis e subconjuntos de dados. Isso cria um conjunto diversificado de modelos, cada um com sua própria tendência e erro.

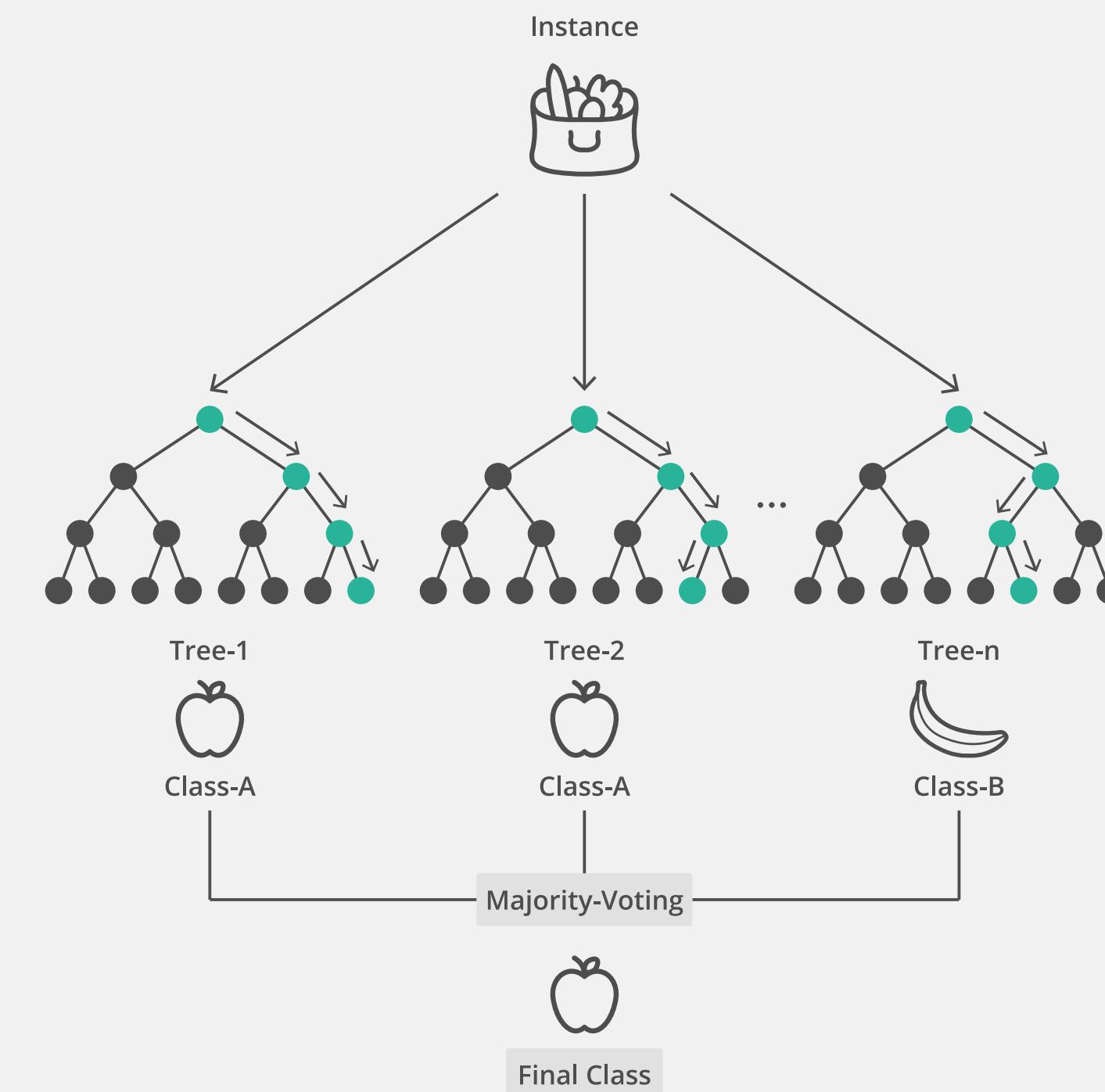


Figura 8 – Random Forest⁶.

⁶Fonte: Understand Random Forest Algorithms With Examples. Analytics Vidhya, 2023. Disponível em: <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>. Acesso em: maio de 2023. Adaptada pelo autor.

O processo de votação no *Random Forest* é usado para fazer uma previsão para um novo ponto de dados com base nas previsões feitas por cada árvore individual no modelo. Cada árvore produz uma previsão para o ponto de dados, que pode ser uma classe (no caso de um problema de classificação) ou um valor numérico (no caso de um problema de regressão).

No processo de votação por maioria, as previsões de cada árvore são combinadas e a classe ou valor numérico que ocorre com mais frequência nas previsões é escolhido como a previsão final. No processo de votação por média, as previsões numéricas de cada árvore são combinadas e a média das previsões é escolhida como a previsão final.

Em alguns casos, as previsões individuais das árvores podem ser ponderadas para dar mais peso a certas árvores ou previsões. Por exemplo, uma árvore com um alto desempenho pode ser ponderada mais fortemente do que uma árvore com um baixo desempenho.

A principal vantagem da *Random Forest* é sua capacidade de lidar com dados de alta dimensionalidade, lidar bem com variáveis categóricas e numéricas, e evitar o problema de *overfitting* (ajuste excessivo) comum em modelos de árvores de decisão individuais.

***Random Forest* é um método robusto, flexível e fácil de usar, que pode lidar com dados numéricos e categóricos, bem como com dados faltantes e ruidosos.**

4.3 Naive Bayes

O *Naive Bayes* é um algoritmo de classificação supervisionado que utiliza o teorema de *Bayes* para estimar a probabilidade de uma classe para uma dada observação. Ele é baseado na suposição ingênua (do inglês, *naive*) de que as características de uma observação são independentes entre si, ou seja, a presença ou ausência de uma característica não está relacionada à presença ou ausência de outra. Essa suposição simplifica os cálculos necessários para estimar as probabilidades e torna o modelo mais eficiente.

Dessa forma, para calcular a probabilidade condicional das características dada a classe, o *Naive Bayes* multiplica as probabilidades das características individuais. Por exemplo, se tivermos duas características **A** e **B**, a probabilidade condicional das características dada a classe seria:

$$P(A, B | \text{classe}) = P(A | \text{classe}) * P(B | \text{classe})$$

Onde:

- $P(A)$ é a probabilidade da característica A ocorrer;
- $P(B)$ é a probabilidade da característica B ocorrer;
- $P(\text{características} | \text{classe})$ é a probabilidade condicional das características dadas a classe.

Por fim, o algoritmo escolhe a classe com a maior probabilidade condicional como a classe prevista para a observação.

O *Naive Bayes* é um algoritmo simples e rápido que funciona bem em grandes conjuntos de dados com muitas características. Frequentemente, utilizamos ele em tarefas de classificação de texto, como categorização de *spam* ou análise de sentimentos em redes sociais. No entanto, a suposição de independência das características pode ser muito simplificadora para alguns problemas e afeta a precisão do algoritmo em certos casos.

4.4 K-Nearest Neighbors

KNN (*K*-Nearest Neighbors, ou **K** vizinhos mais próximos) é um algoritmo de Aprendizado de Máquina supervisionado utilizado para classificação e regressão que se baseia na ideia de que objetos similares estão próximos uns aos outros. Quando se deseja classificar um objeto, o algoritmo procura os **K** objetos mais próximos (de acordo com alguma medida de distância, como distância euclidiana) no conjunto de treinamento e determina a classe do objeto a partir da classe dos **K** vizinhos mais próximos.

Para a classificação, o KNN utiliza uma votação entre as classes dos **K** vizinhos mais próximos para determinar a classe do objeto de teste. Veja a seguir (figura 9) uma ilustração do KNN em um conjunto de dados que possui duas classes. O objetivo é **classificar a instância representada pela estrela**.

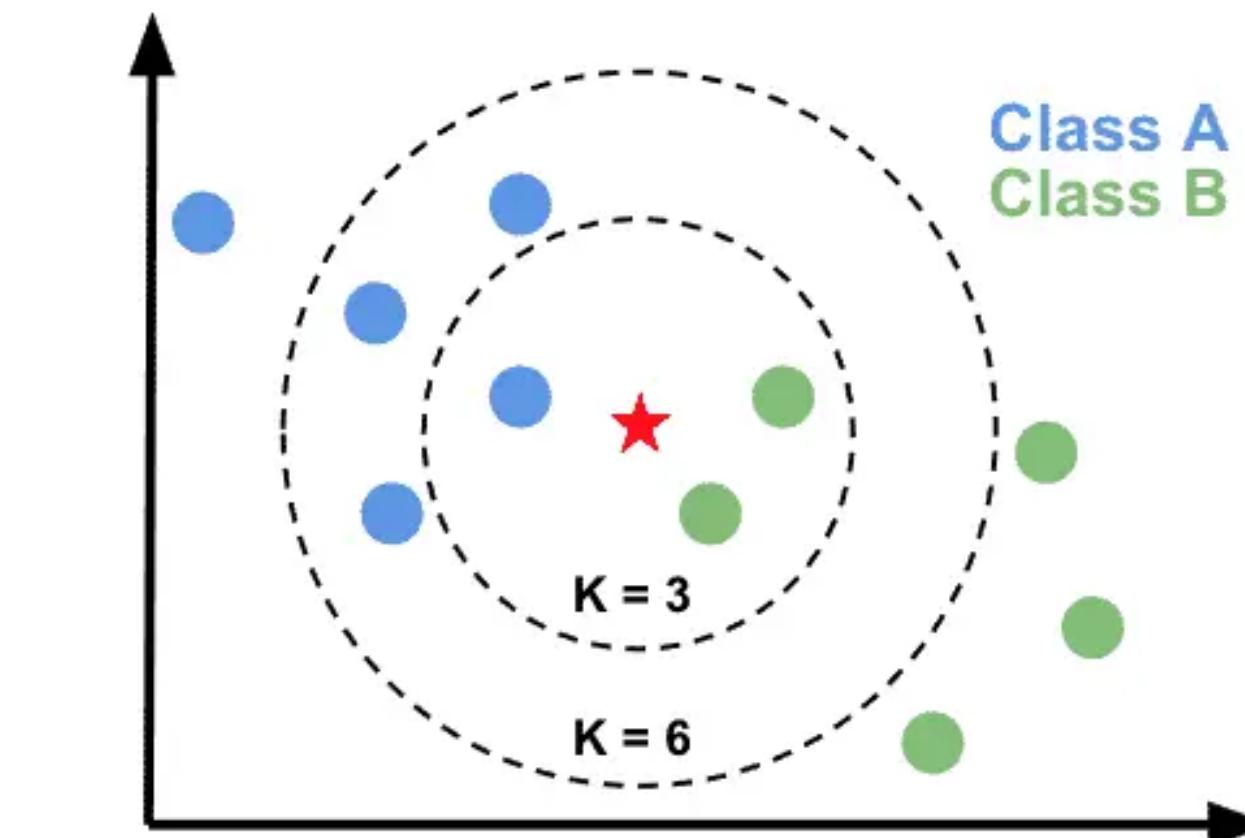


Figura 9 – KNN em conjunto de dados⁷.

Caso seja escolhido um valor de **K=3**, ou seja, os três vizinhos mais próximos participam da votação para definir a classe da instância estrela, ela será classificada como classe B. Entretanto, caso tivéssemos escolhido **K** como 6, o contrário teria acontecido visto que agora a maioria dos vizinhos pertencem à classe A.

O valor de **K** é um parâmetro importante no algoritmo e deve ser escolhido cuidadosamente.

⁷Fonte: CHOUINARD, Jean-Christophe. k-Nearest Neighbors (KNN) in Python. JC Choinard, 2022. Disponível em: <https://www.jcchouinard.com/k-nearest-neighbors/>. Acesso em: maio de 2023.

Um valor muito baixo pode levar a uma classificação instável, enquanto um valor muito alto pode reduzir a precisão da classificação. É comum realizar uma busca em um conjunto de valores de **K** para encontrar o melhor valor para o conjunto de dados em questão.

O KNN é um algoritmo simples, mas pode ser bastante efetivo para determinados problemas de classificação ou regressão, especialmente quando a dimensionalidade do conjunto de dados é baixa e há uma grande quantidade de dados rotulados disponíveis.

4.5 K-means

O algoritmo *K-means* é um método de Aprendizado de Máquina não-supervisionado que agrupa dados em **K** grupos (ou *clusters*) distintos. O objetivo deste método é **minimizar a distância entre cada ponto de dados e o centróide do grupo ao qual pertence**. Um centróide é o ponto médio de todos os pontos de dados em um grupo. A seguir é apresentado um *dataset* agrupado em três *clusters* distintos e bem discriminados (figura 10).

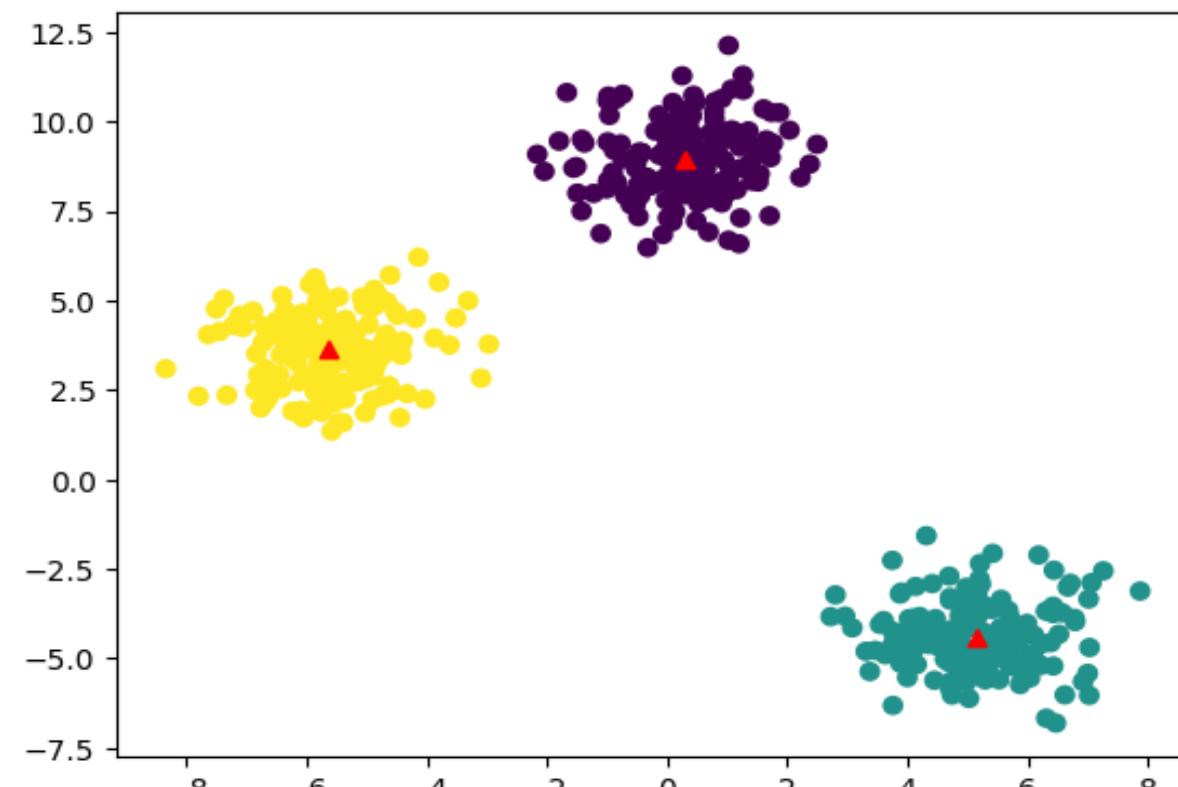


Figura 10 – Dataset de três clusters distintos⁸

O algoritmo *K-means* segue os seguintes passos:

- 1. Inicialização:** o algoritmo inicia escolhendo **K** centróides aleatórios (um para cada *cluster*) a partir dos dados;
- 2. Atribuição:** cada ponto de dados é atribuído ao centróide mais próximo. Isso é feito calculando a distância entre o ponto de dados e cada centróide. Para isso podemos usar;
- 3. Atualização:** uma vez que cada ponto de dados foi atribuído a um *cluster*, o centróide de cada cluster é atualizado para ser a média dos pontos de dados atribuídos a ele;
- 4. Repetição:** os passos 2 e 3 são repetidos até que não haja mais mudanças na atribuição dos pontos de dados aos *clusters*, ou um limite pré-estabelecido de iterações seja atingido.

Apesar de sua simplicidade, o algoritmo *K-means* é sensível à escolha inicial dos centróides, o que pode levar a resultados diferentes para novas execuções do algoritmo. Além disso, ele assume que os grupos são esféricos e têm tamanhos semelhantes, o que pode não ser verdade para alguns conjuntos de dados.

Vale notar também que o algoritmo *K-means* pode não funcionar bem em alguns casos, como em conjuntos de dados com distribuição irregular ou com dados com variação ampla. Além disso, a escolha de um número adequado de *clusters* é uma etapa crítica do algoritmo e pode exigir uma análise cuidadosa dos dados e do problema em questão.

⁸Fonte: CHOUINARD, Jean-Christophe. k-Nearest Neighbors (KNN) in Python. JC Choinard, 2022. Disponível em: <https://www.jcchouinard.com/k-nearest-neighbors/>. Acesso em: maio de 2023.

É mostrado no exemplo a seguir, um agrupamento de um *dataset* em três *clusters* (figura 11). Podemos perceber que os grupos não estão visivelmente separados, tendo muitos pontos próximos que também poderiam pertencer ao *cluster* vizinho, devido à natureza dos dados.

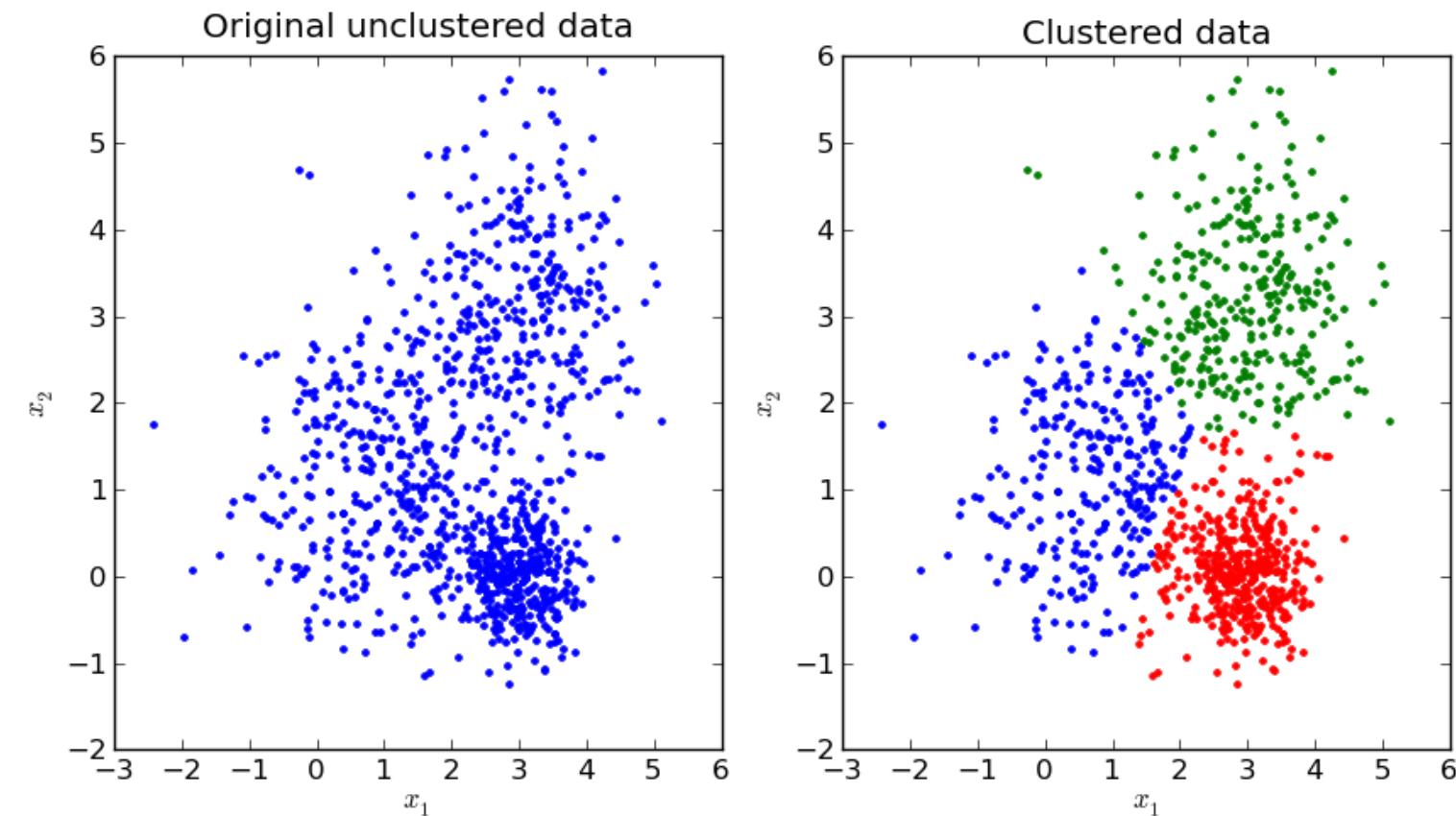


Figura 11 – Dataset de três clusters distintos⁹.

O *k-means* requer que o número **K** de grupos seja especificado previamente, o que pode ser difícil de determinar na prática. Para mitigar esse problema, métodos foram desenvolvidos ao longo dos anos, sendo o mais conhecido entre eles o **método elbow (do inglês, cotovelo)**.

Tal método envolve executar o *K-means* para diferentes valores de **K** e plotar a soma dos quadrados das distâncias dos pontos de cada grupo em relação ao número de *clusters*.

O número ideal de clusters é geralmente o ponto no gráfico em que a curva começa a se nivelar, formando um "cotovelo" (*elbow*).

Este método é simples e intuitivo, e pode ser facilmente implementado para ajudar a determinar o número de clusters mais apropriado para um conjunto de dados.

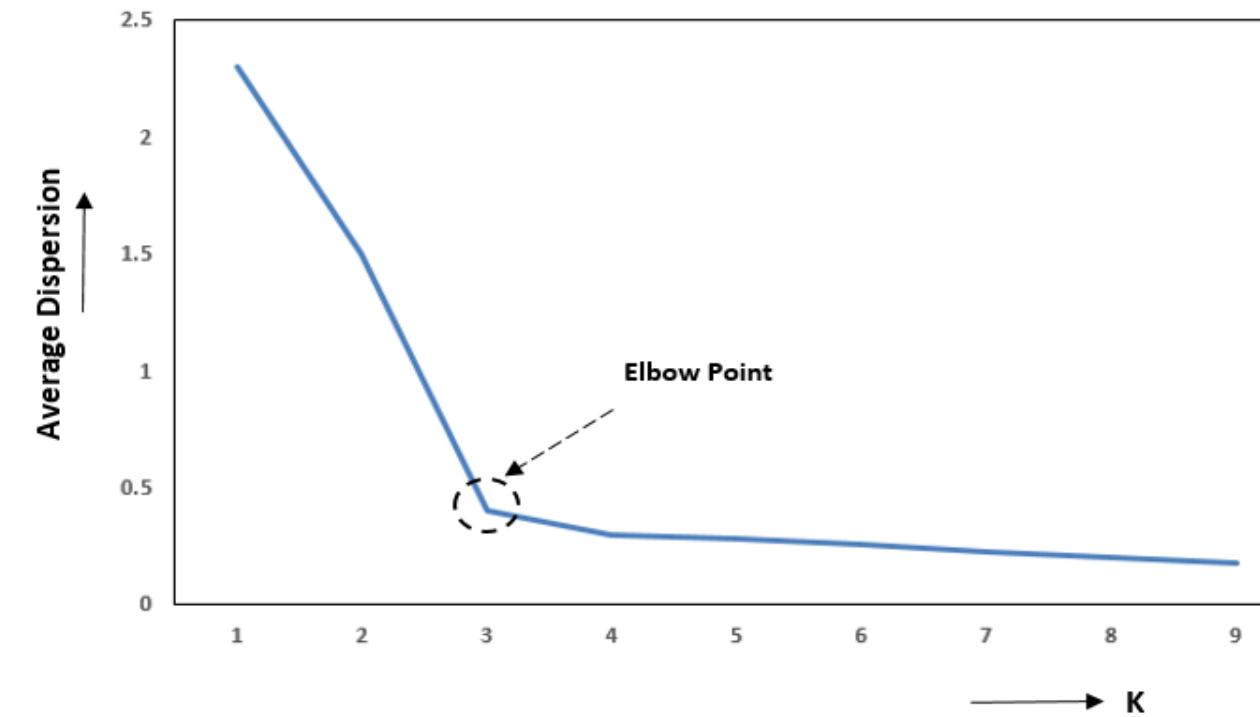


Figura 12 – Gráfico método elbow¹⁰.

⁹Fonte: LANDMAN, Nathan; et. al. k-Means Clustering. Brilliant. Disponível em: <https://brilliant.org/wiki/k-means-clustering/>. Acesso em: maio de 2023.

¹⁰Fonte: DANGETI, Pratap. Statistics for Machine Learning. O'Reilly. Disponível em: <https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/c71ea970-0f3c-4973-8d3a-b09a7a6553c1.xhtml>. Acesso em: maio de 2023.

Explore mais!

No canal "[StatQuest with Josh Starmer](#)" você poderá conhecer a base de diversos modelos de aprendizado de máquina.

No fórum " Where good ideas find you" você encontrará artigos sobre diversos assuntos relacionados a ML, com tutorias, explicações, entre outras coisas.

O vídeo "[Inteligência Artificial brincando de Pique-Esconde](#)" mostra as IAs nos dias de hoje e como elas percebem o mundo.



Ilustração feita por upklyak, disponível no Freepik. Adaptada pelo autor.

Referências Bibliográficas

TensorFlow Core. Disponível em: <https://www.tensorflow.org/tutorials/?hl=pt-br>. Acesso em: maio de 2023.

Intro to Machine Learning. Disponível em: <https://www.kaggle.com/learn/intro-to-machine-learning>. Acesso em: maio de 2023.

AURÉLIEN GÉRON. Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligent systems. 2. ed. [s.l.] O'Reilly Media, Inc., 2019. Acesso em: abril de 2023.

Machine Learning Models: What They Are and How to Build Them. Disponível em: <https://www.coursera.org/articles/machine-learning-models>. Acesso em: maio de 2023.