COLHEITA DE DADOS E INSIGHTS - DADOS VALIOSOS E MADUROS

A PRIMEIRA TÉCNICA DE APRENDIZADO DE MÁQUINA

14

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

1 A PRIMEIRA TÉCNICA DE APRENDIZADO DE MÁQUINA

1.1 Introdução

Os algoritmos preditivos supervisionados são a base para muitas aplicações modernas de Inteligência Artificial (IA) e Machine Learning (ML). Esses algoritmos são projetados para **aprender a partir de dados rotulados**, ou seja, dados que vêm acompanhados de uma "resposta correta" ou "*label*". No caso específico da classificação, o objetivo é ensinar o modelo a **identificar a categoria** à qual um novo dado pertence, com base nos padrões que aprendeu durante o treinamento.

O conceito de aprendizado supervisionado gira em torno da ideia de treinar um modelo com exemplos já conhecidos, onde os dados de entrada (features) estão associados a um rótulo ou resultado específico (*label*). Por exemplo, em um problema de classificação de e-mails, cada e-mail pode ser marcado como "spam" ou "não spam". O algoritmo aprende a distinguir essas categorias ao analisar os padrões presentes nos e-mails rotulados, e aplica esse conhecimento para classificar novos e-mails que encontra (ilustrado na figura "Exemplo clássico de um modelo preditivo do tipo classificação").

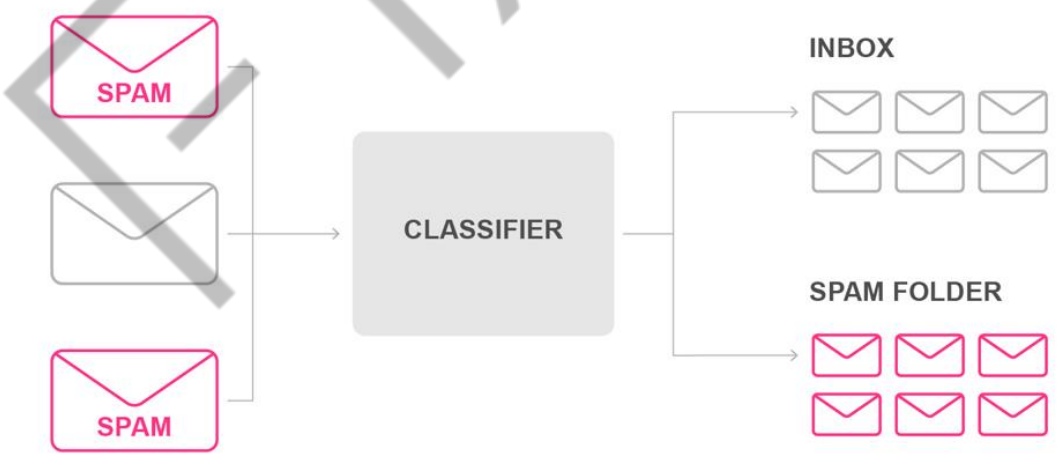

Figura 1 - Exemplo clássico de um modelo preditivo do tipo classificação
Fonte: Decom (2021)

A importância dos algoritmos de classificação se estende a uma ampla gama de aplicações. Eles são utilizados para **prever se um cliente vai comprar um produto**, **diagnosticar doenças** em exames médicos, ou até mesmo para **reconhecimento de voz e imagens**. O que todos esses casos têm em comum é a

necessidade de um modelo capaz de tomar decisões baseadas em padrões aprendidos a partir de dados já coletados (exemplos).

A classificação não precisa ser binária, como exemplificado na figura anterior. Podemos também distinguir exemplos sob múltiplas classes, desde que elas sejam finitas e conhecidas (ou seja, que haja exemplos de cada classe no conjunto de treinamento). Observe a figura “Tipos de classificação”.

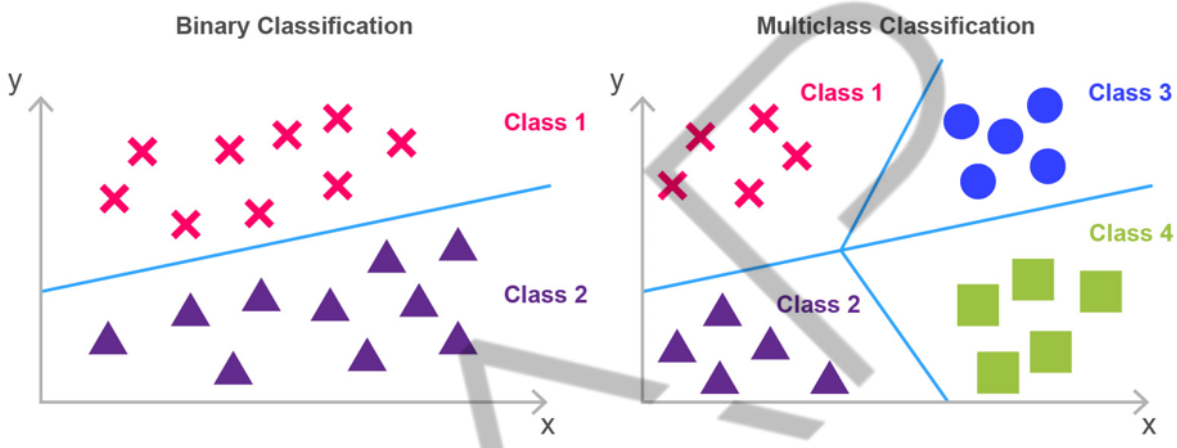

Figura 2 - Tipos de classificação
Fonte: X (2022)

Aprender com os dados significa que o algoritmo identifica as relações entre as variáveis de entrada e o rótulo, criando um **modelo que pode generalizar para novos dados**. A qualidade do aprendizado **depende da quantidade e qualidade dos dados** disponíveis, assim como da escolha do algoritmo adequado para o problema. Esse processo de aprendizagem permite aos sistemas de Machine Learning automatizar a tomada de decisões em uma variedade de contextos.

Ao explorar os algoritmos de classificação, você descobrirá como **diferentes abordagens podem ser aplicadas para resolver problemas práticos**. Desde técnicas simples, como K-Nearest Neighbors (KNN), até métodos mais complexos, como Support Vector Machines (SVM), cada algoritmo oferece suas próprias vantagens e desafios. Entender como cada um aprende com os dados e como realiza previsões é essencial para se tornar proficiente em ML e em sua aplicação prática.

2 APLICAÇÕES

O aprendizado supervisionado de classificação tem se mostrado uma ferramenta poderosa em diversos segmentos, permitindo a automação e a otimização de processos que anteriormente dependiam de decisões humanas. Mesmo em dados estruturados ou em dados complexos, como imagens e textos, esses modelos preditivos são capazes de transformar grandes volumes de dados em *insights* valiosos, auxiliando na tomada de decisões em tempo real.

2.1 Na educação

No setor educacional, por exemplo, o uso de aprendizado supervisionado pode ser visto na **personalização do ensino**. Instituições de ensino podem utilizar dados estruturados, como histórico acadêmico, frequência e participação em atividades, para prever o desempenho futuro dos alunos. O cenário envolve escolas ou universidades que desejam **identificar alunos em risco de reprovação ou evasão**. O problema está na necessidade de intervenções antecipadas para melhorar esses indicadores. Portanto, o objetivo é utilizar um modelo preditivo para classificar os alunos em diferentes categorias de risco. A tomada de decisão, nesse caso, envolve a implementação de ações personalizadas, como tutorias ou aconselhamento, com base nas previsões do modelo.

2.2 No varejo

No varejo, a classificação supervisionada é amplamente utilizada para **segmentação de clientes e campanhas de marketing direcionadas**. Os varejistas podem analisar dados estruturados, como histórico de compras, comportamento de navegação e interações em redes sociais, para prever quais clientes são mais propensos a responder positivamente a uma promoção específica. O cenário típico é o de uma empresa que deseja aumentar as taxas de conversão de suas campanhas de marketing. O problema é **identificar quais clientes têm maior probabilidade de realizar uma compra** com base em um desconto oferecido. O objetivo é classificar os clientes em grupos, como “altamente propensos a comprar” ou “menos propensos

a comprar", para que as campanhas sejam mais eficazes. A decisão que se segue ao uso do modelo preditivo é a de direcionar esforços de marketing apenas para aqueles clientes com maior probabilidade de conversão, otimizando o orçamento e aumentando as vendas.

2.3 Na saúde

Na área da saúde, a classificação supervisionada desempenha um papel crucial no **diagnóstico precoce de doenças**. Um exemplo relevante é a utilização de dados complexos, como imagens médicas, para prever a presença de condições como câncer. O cenário envolve um hospital que busca melhorar a precisão e a rapidez no diagnóstico de câncer a partir de mamografias. O problema está na grande quantidade de exames a serem analisados, podendo sobrecarregar os profissionais de saúde. O objetivo, então, é treinar um modelo de classificação que possa **distinguir entre imagens normais e aquelas que apresentam sinais de câncer**. Adotando esse modelo de operação, os médicos podem priorizar os casos que necessitam de atenção imediata, agilizando o processo de diagnóstico e aumentando as chances de tratamento bem-sucedido.

2.4 No agronegócio

No agronegócio, o aprendizado supervisionado de classificação é aplicado para **otimizar a produção e a gestão de recursos**. Um exemplo é o uso de dados estruturados combinados com imagens de satélite para **classificar áreas agrícolas com base no risco de pragas ou doenças nas plantações**. O cenário envolve uma fazenda de grande escala que precisa monitorar milhares de hectares de cultivo. O problema está na detecção precoce de áreas afetadas por pragas, que podem comprometer a produção. O objetivo é utilizar um modelo de classificação que, ao analisar as imagens de satélite e dados climáticos, possa identificar e classificar as áreas de risco. A decisão tomada com base no modelo preditivo pode incluir a aplicação direcionada de defensivos agrícolas ou a adoção de medidas preventivas em áreas específicas, otimizando o uso de insumos e preservando a produtividade da safra.

Os exemplos citados mostram como o aprendizado supervisionado de classificação está transformando diversos setores, oferecendo soluções precisas e eficientes para problemas complexos. Ao incorporar modelos preditivos no processo de tomada de decisão, as organizações conseguem não apenas melhorar suas operações, mas também fornecer resultados de maior impacto.

Em qualquer caso, **o processo de construir e avaliar um modelo preditivo é sempre o mesmo**, conforme ilustrado na figura “Etapas do processo de treinamento e avaliação de um modelo preditivo de classificação”. A partir da base de dados, podemos dividir em: TREINO, VALIDAÇÃO e TESTE. O TREINO e a VALIDAÇÃO são utilizados em conjunto para gerar um modelo preditivo por meio de um algoritmo de ML. O melhor modelo tem sua performance final avaliada (e reportada) sobre os dados do TESTE.

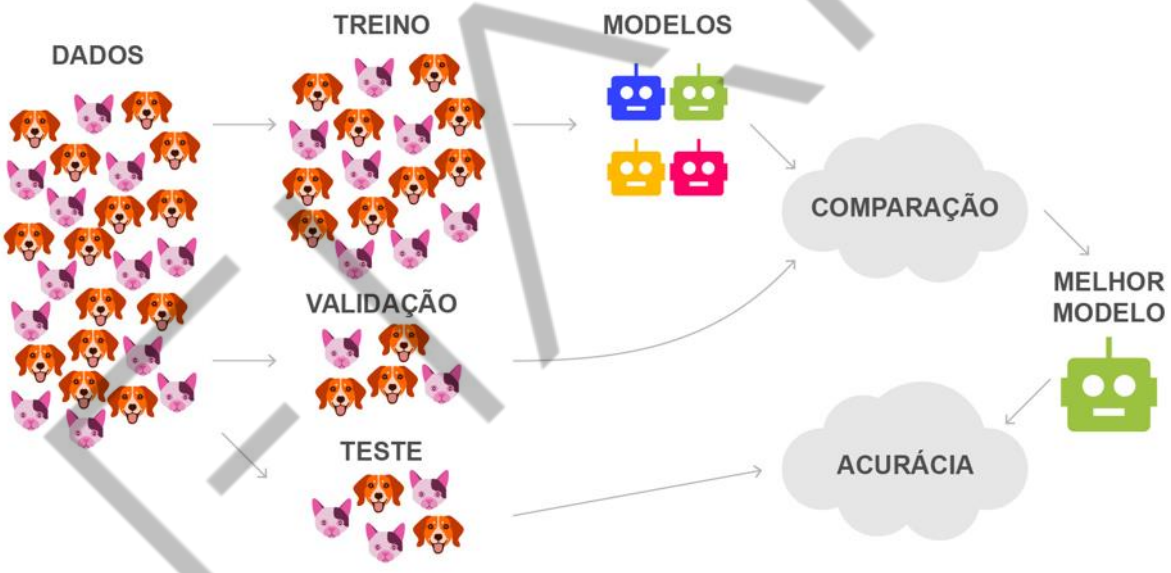

Figura 3 - Etapas do processo de treinamento e avaliação de um modelo preditivo de classificação
Fonte: Análise macro (2023)

Isso nos leva para um novo assunto: como avaliar a performance de modelos preditivos de classificação?

A primeira técnica de aprendizado de máquina

3 AVALIAÇÃO DE DESEMPENHO

Avaliar a performance de um modelo preditivo é essencial para garantir que ele realmente cumpra seu propósito de maneira eficaz e confiável. Sem essa avaliação, corremos o risco de implementar um modelo que pode estar gerando previsões imprecisas, podendo levar a decisões equivocadas com consequências potencialmente significativas.

Calcular a performance permite **identificar pontos fortes e fracos do modelo**, compará-lo com alternativas, ajustar parâmetros e, se necessário, melhorar o modelo para que ele se alinhe melhor com os objetivos do negócio. Além disso, **uma avaliação rigorosa da performance é fundamental para assegurar que o modelo generalize bem para novos dados**, e isso é fundamental para seu sucesso em aplicações práticas.

3.1 Métricas de avaliação

As métricas de avaliação fornecem uma base sólida para **entender quão bem o modelo está realizando a tarefa de classificação** e onde ele pode ser melhorado. Neste momento, iremos discutir as principais métricas que são utilizadas para avaliar modelos de classificação, abordando como são calculadas, interpretadas e onde são mais adequadas.

3.1.1 Acurácia

A acurácia é uma das métricas mais simples e amplamente utilizada para realizar as avaliações de modelos de classificação. Ela é calculada como a proporção de previsões corretas (tanto positivas quanto negativas) sobre o total de previsões realizadas pelo modelo. Em termos matemáticos, a acurácia é dada por:

$$Acurácia\ (ACC) = \frac{Número\ de\ previsões\ corretas}{Total\ de\ previsões}$$

A acurácia é fácil de interpretar: ela indica a porcentagem de exemplos corretamente classificados pelo modelo. No entanto, essa métrica tem uma limitação

significativa quando os dados são desbalanceados, ou seja, quando uma das classes ocorre muito mais frequentemente do que a outra. Neste caso, um modelo pode alcançar alta acurácia simplesmente por prever sempre a classe majoritária, sem realmente aprender os padrões dos dados. Portanto, a acurácia é mais útil em cenários onde as classes estão balanceadas.

3.1.2 Precisão e revocação

A **precisão (ou *precision*)** é uma métrica que **foca na qualidade das previsões positivas do modelo**. Ela é calculada como a razão entre o número de verdadeiros positivos (exemplos corretamente classificados como positivos) e o total de previsões positivas feitas pelo modelo (incluindo os falsos positivos):

$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos}$$

A precisão é interpretada como **a proporção de previsões positivas que são realmente corretas**. Essa métrica é particularmente importante em situações nas quais **o custo de um falso positivo é alto**. Por exemplo, em um sistema de detecção de fraudes, se o modelo identificar uma transação legítima como fraudulenta (falso positivo), isso pode gerar inconvenientes ao usuário e custos para a empresa. No entanto, um ponto fraco da precisão é que ela não leva em consideração os falsos negativos, causando problemas em certos contextos.

A **revocação**, também conhecida como sensibilidade ou *recall*, **mede a capacidade do modelo de identificar todos os exemplos positivos**. Ela é calculada como a razão entre o número de verdadeiros positivos e o total de exemplos que realmente são positivos (verdadeiros positivos mais falsos negativos):

$$Revocação\ (Recall) = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos}$$

A revocação é interpretada como **a proporção de exemplos positivos que foram corretamente identificados pelo modelo**. Ela é especialmente útil em situações onde é crucial detectar todos os casos positivos, como em diagnósticos médicos, que caso não seja identificada uma doença (falso negativo) pode resultar em consequências graves. Entretanto, a revocação pode ser maximizada às custas

da precisão, por exemplo, classificando tudo como positivo, o que aumenta o número de verdadeiros positivos, mas também de falsos positivos.

3.1.3 F1-Score

O F1-Score combina a precisão e a revocação em uma única medida, sendo a média harmônica dessas duas métricas. É calculado como:

$$F1\ Score = \frac{2 \times Precisão \times Revocação}{Precisão + Revocação}$$

O F1-Score oferece equilíbrio entre precisão e revocação, sendo **especialmente útil quando se precisa considerar ambos os tipos de erros (falsos positivos e falsos negativos)**. Essa métrica é vantajosa em cenários com classes desbalanceadas, onde uma alta acurácia pode não refletir o desempenho real do modelo. No entanto, uma limitação do F1-Score é que ele não dá peso extra para casos nos quais um dos erros é mais grave que o outro, como quando os falsos negativos são mais prejudiciais que os falsos positivos.

3.1.4 Curva ROC e AUC

A Curva ROC (Receiver Operating Characteristic) é um gráfico que mostra a relação entre a taxa de verdadeiros positivos (revocação) e a taxa de falsos positivos para diferentes limiares de decisão do modelo. A figura “Exemplo de Curva ROC” ilustra suas curvas características.

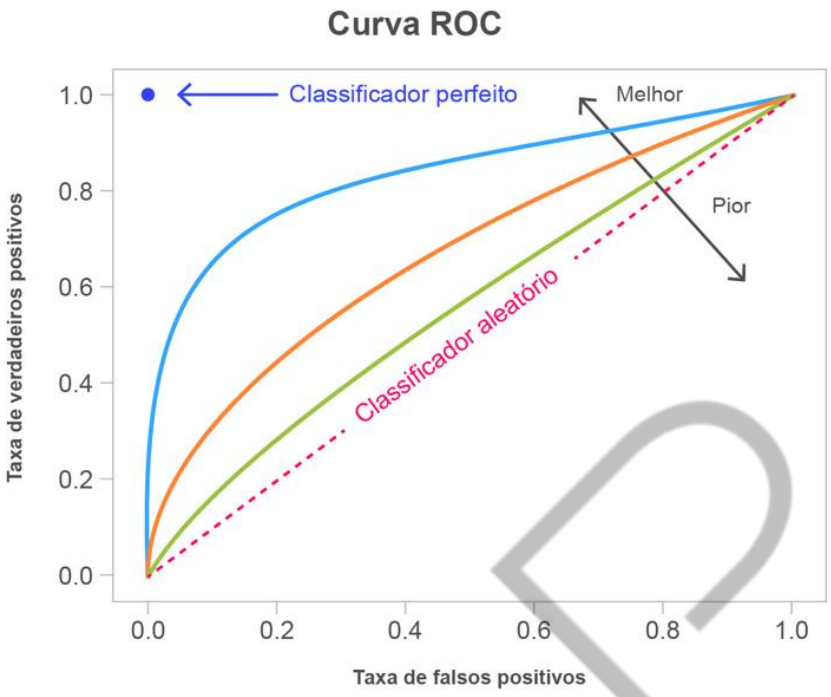

Figura 4 - Exemplo de Curva ROC
Fonte: Wikipedia (2024)

A Curva ROC é utilizada para avaliar modelos de classificação binária. Um classificador aleatório (por exemplo, um verificador de spam que apenas tenta adivinhar se um e-mail é spam ou não, sem aprender nada) tem a curva representada pelo pontilhado rosa. Já o classificador perfeito (que sempre acerta se um e-mail é spam ou não) possui o ponto operacional no ponto em azul no canto superior esquerdo da figura anterior (representado pela coordenada (x, y) = (0.0, 1,0). Basicamente, sua curva sobe do ponto (0.0, 0.0) ao ponto (0.0, 1.0) e deste ao ponto (1.0, 1.0). Quanto mais a curva se distancia do classificador aleatório em direção ao classificador perfeito, melhor é aquele modelo. Portanto, na figura “Exemplo de Curva ROC”, o classificador laranja é **melhor** comparado ao classificador verde, porém ainda é **pior** que o classificador azul.

O AUC (Area Under the Curve) é uma métrica que quantifica a área sob a Curva ROC, fornecendo uma medida única do desempenho do modelo em todos os limiares possíveis.

A Curva ROC e o AUC são úteis para **avaliar a capacidade discriminatória de um modelo, especialmente em problemas de classificação binária**. Um AUC próximo de 1 indica que o modelo tem um bom desempenho em separar as classes, enquanto um AUC de 0,5 sugere que o modelo não está fazendo melhor do que uma escolha aleatória. A desvantagem do AUC é ser menos informativo ao tratar de

problemas onde a relação entre classes é altamente desbalanceada, ou em contextos em que um tipo de erro é mais crítico comparado a outro.

3.1.5 Matriz de confusão

A matriz de confusão é uma ferramenta visual e numérica que resume o desempenho de um modelo de classificação. Ela mostra o número de **verdadeiros positivos** (**TP**: *True Positive*), **verdadeiros negativos** (**TN**: *True Negative*), **falsos positivos** (**FP**: *False Positive*) e **falsos negativos** (**FN**: *False Negative*) em uma tabela, facilitando a análise detalhada de como o modelo está errando. Cada célula da matriz representa uma combinação de previsões e resultados reais. Observe uma representação de matriz de confusão na figura “Exemplo de matriz de confusão binária”.

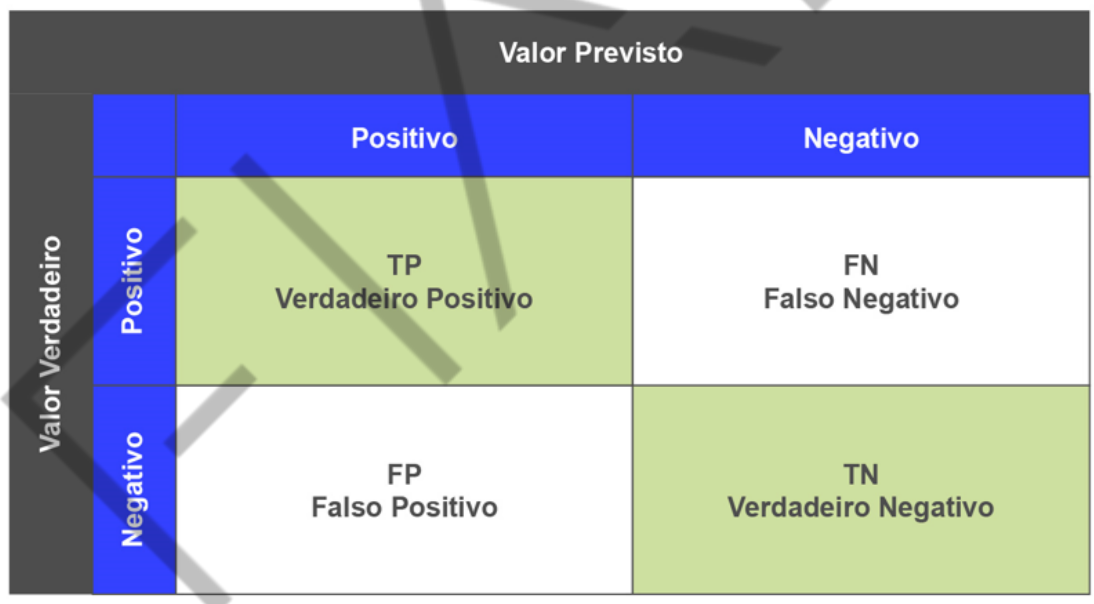

Figura 5 - Exemplo de matriz de confusão binária
Fonte: LinkedIn (2018)

A interpretação da matriz de confusão permite identificar os tipos de erros que o modelo está cometendo e se há uma tendência em errar mais em uma classe do que em outra. A vantagem da matriz de confusão é proporcionar uma visão completa do desempenho do modelo, ao contrário de métricas isoladas. No entanto, analisar a matriz de confusão pode ser menos intuitivo em modelos que lidam com múltiplas classes, onde a matriz se torna maior e mais complexa.

Por outro lado, ao contrário da Curva ROC que apenas trata modelos binários, podemos traçar a matriz de confusão para problemas multiclasse também. Na diagonal principal sempre terá os acertos, e fora delas estarão os erros. A figura “Exemplo de confusão multiclasse” ilustra um exemplo hipotético.

		Predicted			
		On time	Late	Very Late	
Actual	On time	40	7	3	50
	Late	5	25	5	35
	Vary Late	3	5	8	16

Figura 6 - Exemplo de matriz de confusão multiclasse
Fonte: Microsoft (2023)

Nesta figura, as colunas indicam os números de exemplos preditos naquela classe. Por exemplo, temos $40 + 5 + 3 = 48$ exemplos do teste que foram classificados como “*on time*”. As linhas indicam a distribuição da soma real de exemplos do teste em cada *label*. Por exemplo, na base apontada temos $40 + 7 + 3 = 50$ exemplos verdadeiramente indicados como “*on time*”. A diagonal principal indica $40 + 25 + 8 = 73$ exemplos corretamente classificados, de um total de $40 + 7 + 3 + 5 + 25 + 5 + 3 + 5 + 8 = 100$ exemplos.

Uma boa matriz de confusão possui os exemplos do teste distribuídos na diagonal principal (destacado em verde na figura “Exemplo de matriz de confusão multiclasse”) e poucos exemplos nas demais células, pois elas representam predições incorretas.

3.1.6 Por que existem tantas métricas e qual o benefício de conhecê-las?

Cada uma dessas métricas oferece uma perspectiva diferente sobre o desempenho de um modelo de classificação. Vale mencionar que, além destas, existem muitas outras. Entender os pontos fortes e fracos de cada métrica é fundamental para selecionar a abordagem mais adequada e, assim, garantir que o modelo escolhido atenda às expectativas e objetivos do projeto.

3.2 Escolhendo a métrica ideal para o problema

Escolher a métrica ideal para avaliar um modelo de classificação é uma etapa fundamental, além disso, depende das características do problema em questão e dos objetivos do projeto. Diferentes métricas podem fornecer *insights* variados sobre o desempenho do modelo, e a seleção da métrica adequada garantirá que o modelo esteja otimizado para os resultados mais relevantes.

A figura “Dicas para escolher métricas de avaliação de modelos preditivos de classificação” mostra sete passos a serem considerados na escolha da métrica ideal para problemas de classificação.

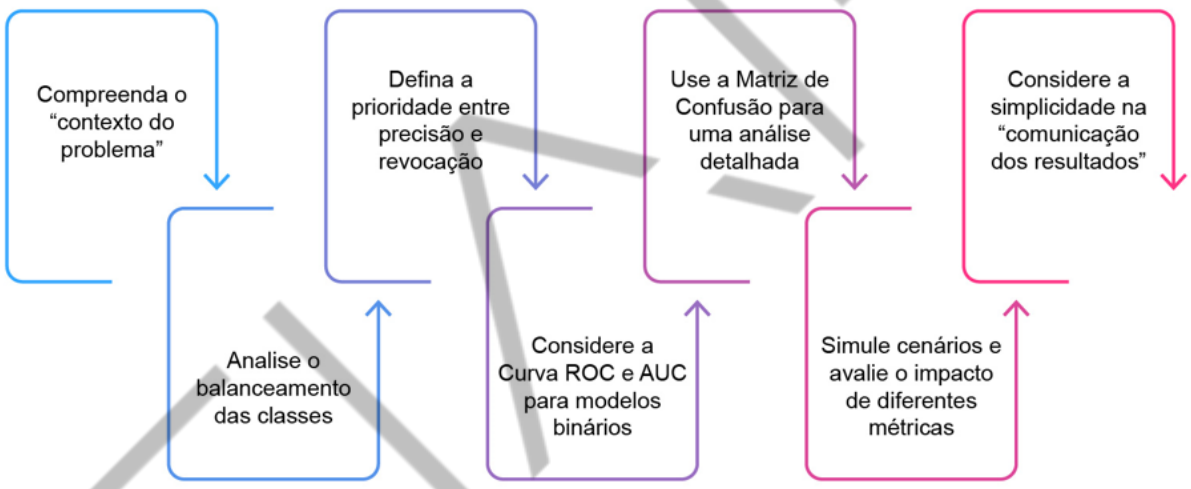

Figura 7 - Dicas para escolher métricas de avaliação de modelos preditivos de classificação
Fonte: Elaborado pelo autor (2024)

As dicas são:

1. **Compreenda o “contexto do problema”:** o primeiro passo é entender profundamente o contexto do problema de classificação. Pergunte-se qual é o impacto de um erro de classificação. Um falso positivo ou um falso negativo tem consequências mais graves? Por exemplo, em um sistema de diagnóstico médico, um falso negativo (não identificar uma doença) pode ser muito mais prejudicial do que um falso positivo. Compreender esses impactos ajuda a determinar quais tipos de erros devem ser minimizados.
2. **Analise o balanceamento das classes:** verifique se as classes estão balanceadas. Em muitos problemas de classificação, como detecção de fraudes ou diagnóstico de doenças raras, as classes são desbalanceadas,

com uma classe ocorrendo muito mais frequentemente que a outra. Se as classes estão desbalanceadas, métricas como acurácia podem ser enganosas, já que um modelo pode atingir alta acurácia simplesmente prevendo a classe majoritária. Nesse caso, métricas como precisão, revocação ou F1-Score podem ser mais adequadas.

3. **Defina a prioridade entre Precisão e Revocação:** com base no contexto, determine se o foco deve estar em maximizar a precisão ou a revocação, ou em encontrar um equilíbrio entre as duas. Se o custo de um falso positivo é alto (por exemplo, em um sistema de triagem de candidatos a um emprego), a precisão deve ser priorizada. Se o custo de um falso negativo é mais alto (como na detecção de fraudes, onde deixar de detectar uma fraude pode ser muito caro), a revocação pode ser mais importante. Se ambos os erros são críticos, o F1-Score, que equilibra precisão e revocação, pode ser a melhor escolha.
4. **Considere a Curva ROC e AUC para modelos binários:** se o problema envolve classificação binária e há interesse em entender o desempenho do modelo em vários limiares de decisão, a Curva ROC e o AUC são ferramentas úteis pois elas permitem avaliar a capacidade do modelo de separar as classes positivas e negativas ao longo de diferentes pontos de corte. Um AUC alto indica um bom desempenho geral do modelo, mas se a tarefa envolver classes desbalanceadas, é importante interpretar o AUC com cuidado e em conjunto com outras métricas.
5. **Usar a matriz de confusão para uma análise detalhada:** se for necessário entender exatamente onde o modelo está errando, a matriz de confusão oferece uma visão detalhada. Ela permite analisar as previsões corretas e incorretas para cada classe, o que pode ser especialmente útil em problemas de múltiplas classes. A matriz de confusão ajuda a identificar padrões de erro e a entender se há uma tendência do modelo em confundir classes específicas.
6. **Simular cenários e avaliar o impacto de diferentes métricas:** antes de tomar uma decisão definitiva, simule diferentes cenários utilizando várias métricas. Por exemplo, avalie como mudanças no limiar de decisão afetam a precisão, a revocação e o F1-Score. Isso pode ser feito ajustando o limiar

de decisão em um modelo de classificação e observando como as métricas respondem. Esse processo ajuda a visualizar o *trade-off* entre diferentes métricas e a escolher a que melhor atende às necessidades do problema.

7. **Considerar a simplicidade na comunicação dos resultados:** por fim, considere como os resultados serão comunicados para os *stakeholders*. Métricas simples e intuitivas, como acurácia, são mais fáceis de explicar, mas podem não capturar toda a complexidade do problema. Em situações onde é necessário comunicar de forma clara e direta, combine métricas mais complexas com métricas simples para que os resultados sejam compreendidos por todos os envolvidos.

4 PRINCIPAIS ALGORITMOS

Agora que sabemos o que é ML, do que se trata o aprendizado supervisionado, quais são os problemas típicos modelados pela classificação e como avaliar os modelos treinados, está na hora de conhecermos os principais algoritmos que aprendem conceitos categóricos a partir dos dados.

A lista de algoritmos é vasta e nossa intenção não é passar todos eles, mas dar destaques aos principais. Assim, a partir destes você poderá conhecer outros e contrastar seus benefícios e limitações. Vamos lá?

4.1 K-Nearest Neighbors

O K-Nearest Neighbors (KNN) é um algoritmo de aprendizado supervisionado simples, mas poderoso, utilizado tanto para classificação quanto para regressão. A intuição básica por trás do KNN é que objetos similares tendem a estar próximos uns dos outros em um espaço de características. Em outras palavras, a classe ou valor de um dado ponto é determinado com base nas classes ou valores dos seus vizinhos mais próximos. O princípio fundamental é “você é o que seus vizinhos são”.

A figura “Intuição do KNN” ilustra a intuição do KNN: imagine que possuímos vários pontos representantes de duas classes (azul e rosa) distribuídos por um espaço com duas variáveis (X e Y). Agora, imagine que um novo ponto será representado pela bolinha verde, e seja apresentado a este espaço. Pela região onde ele se encontra, qual deveria ser seu *label*? Claramente, **como muitos vizinhos pertencem à classe rosa, esta deve ser a classe do ponto em questão.**

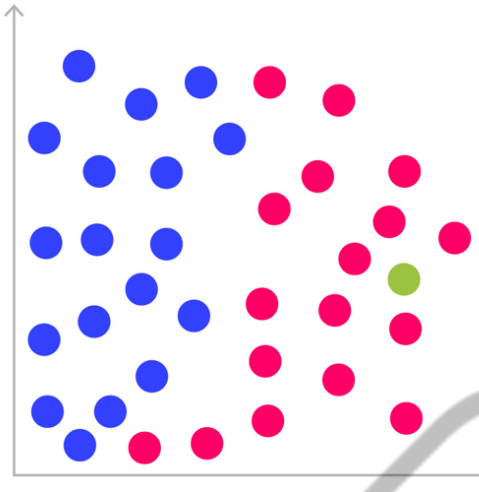

Figura 8 - Intuição do KNN
Fonte: Didática Tech (2024)

O racional ilustrado nesta figura pode ser estendido para outro número de dimensões (número de variáveis). Para nós, seres humanos, a interpretação do espaço dimensional maior que três é abstrata, mas a máquina consegue estabelecer relações entre os dados em um número maior de dimensões.

4.1.1 Funcionamento do algoritmo

O KNN funciona de maneira bastante direta:

- **Definição do valor de K:** primeiramente, escolhe-se o número de vizinhos mais próximos, denotado por K.
- **Cálculo das distâncias:** para um novo ponto de dados, o algoritmo calcula a distância entre esse ponto e todos os outros pontos do conjunto de dados de treinamento. A métrica de distância mais comum é a distância Euclidiana, mas outras métricas podem ser usadas (veremos adiante).
- **Identificação dos vizinhos mais próximos:** com base nas distâncias calculadas, o algoritmo identifica os K pontos mais próximos do novo ponto.
- **Decisão da classe (ou valor) do novo ponto:** na classificação, o KNN atribui ao novo ponto a classe mais comum entre seus K vizinhos mais próximos (maioria simples). Na regressão, o KNN atribui ao novo ponto a média dos valores dos K vizinhos mais próximos.

4.1.2 Vantagens e desvantagens do método

Como todo algoritmo preditivo, o KNN possui vantagens e desvantagens, como apresentadas no quadro “Vantagens e desvantagens do KNN”. Não há como saber se o KNN funcionará bem ou mal no seu conjunto de dados, portanto, é sempre recomendado experimentar.

Vantagens	Desvantagens
Simplicidade: o KNN é fácil de entender e implementar, sem a necessidade de suposições complexas sobre os dados.	Custo computacional: para cada nova previsão, o KNN precisa calcular a distância entre o ponto e todos os pontos do conjunto de treinamento, o que pode ser computacionalmente caro.
Flexibilidade: pode ser usado para problemas de classificação e regressão.	Sensível ao valor de K: a escolha do valor de K pode influenciar significativamente o desempenho do modelo.
Não paramétrico: não faz suposições sobre a distribuição dos dados, o que o torna útil em muitos cenários.	Sensível a dados desbalanceados: se uma classe for majoritária, ela pode dominar as previsões simplesmente por ser mais frequente entre os vizinhos.

Quadro 1 - Vantagens e desvantagens do KNN
Fonte: Elaborado pelo autor (2024)

4.1.3 Parametrização e influência no resultado

Em relação aos hiperparâmetros do KNN, ou seja, as fontes de configuração do algoritmo, temos as seguintes alternativas:

- **Valor de K:** o valor de K é o parâmetro mais crítico do KNN. Um valor pequeno de K faz o modelo mais sensível ao ruído, resultando em overfitting. Por outro lado, um valor grande de K suaviza a decisão, mas pode levar a underfitting, onde o modelo se torna muito genérico e perde a capacidade de capturar as particularidades dos dados. Por exemplo, na figura “Intuição do KNN II” temos duas classes, a amarela e a roxa. Para o ponto rosa, seu *label* pode ser roxo se escolhermos k=3 ou amarelo se escolhermos k=6.

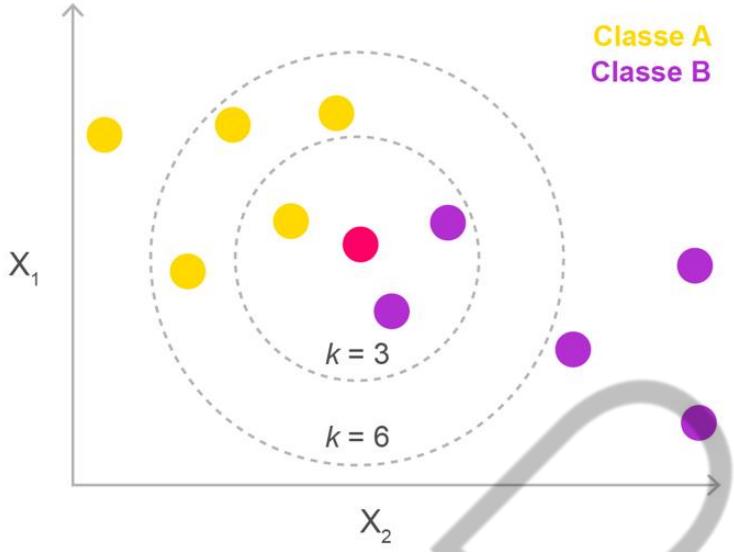

Figura 9 - Intuição do KNN II
Fonte: Didática Tech (2024)

- **Métrica de distância:** a escolha da métrica de distância (Euclidiana, Manhattan, Cosseno) pode influenciar a performance do modelo. A distância Euclidiana, por exemplo, pode ser mais adequada para dados onde a magnitude das diferenças é importante, enquanto a distância Manhattan pode ser preferível em espaços de características de alta dimensão.

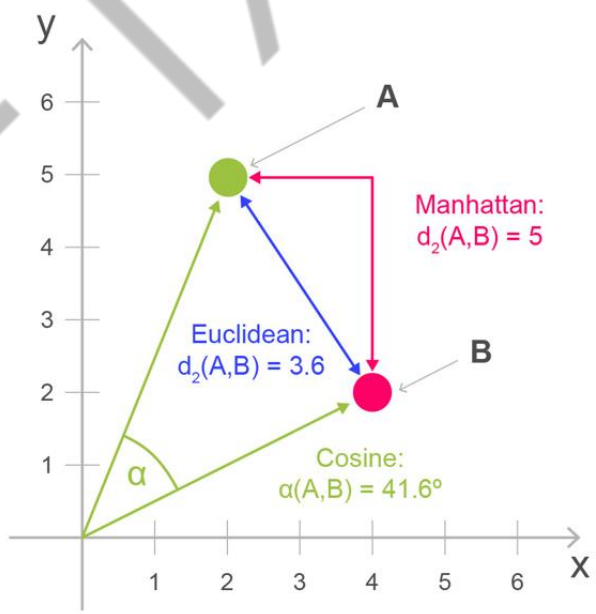

Figura 10 - Fórmulas de distância
Fonte: Digital Humanities (2016)

- **Ponderação dos vizinhos:** em alguns casos, os vizinhos mais próximos podem ser ponderados de forma diferente (por exemplo, ponderando

inversamente à distância) para que pontos mais próximos tenham mais influência na decisão final. Alterar qualquer configuração acima pode mudar significativamente os resultados do modelo. Assim, é essencial realizar experimentos com diferentes configurações de K, distância e ponderação de vizinhos, para encontrar a configuração ideal para o problema em questão. Além disso, por ser um algoritmo baseado em distâncias, é recomendado normalizar/padronizar os dados antes de iniciar o treinamento.

4.2 Regressão Logística

A Regressão Logística é um algoritmo de aprendizado supervisionado **utilizado principalmente para problemas de classificação binária**, onde o objetivo é **prever a probabilidade** de uma observação pertencer a uma das duas classes possíveis. Ao contrário da regressão linear, que prevê valores contínuos, a regressão logística prevê a probabilidade de um evento ocorrer (resultado entre 0 e 1) e, em seguida, mapeia essa probabilidade para uma classe categórica.

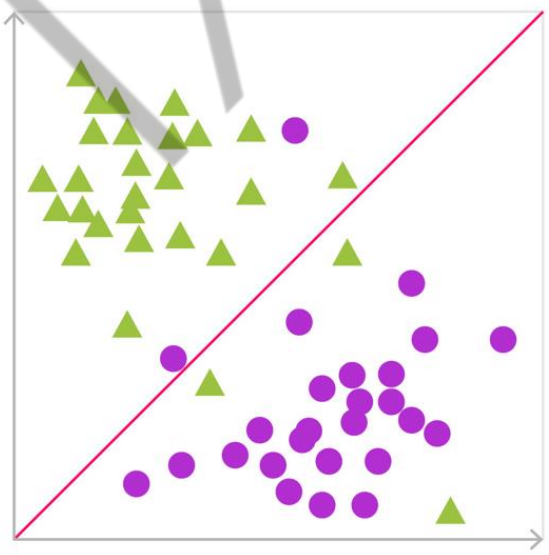

Figura 11 - Intuição da regressão logística
Fonte: Lean saúde (2024)

A figura “Intuição da regressão logística” mostra a intuição, que é encontrar uma separação linear entre duas classes. Assim, transformamos o resultado linear de uma combinação das características (features) em uma probabilidade usando a

função logística (também conhecida como função sigmoide, como ilustrado na figura “Representação da regressão logística sobre a função sigmoide”).

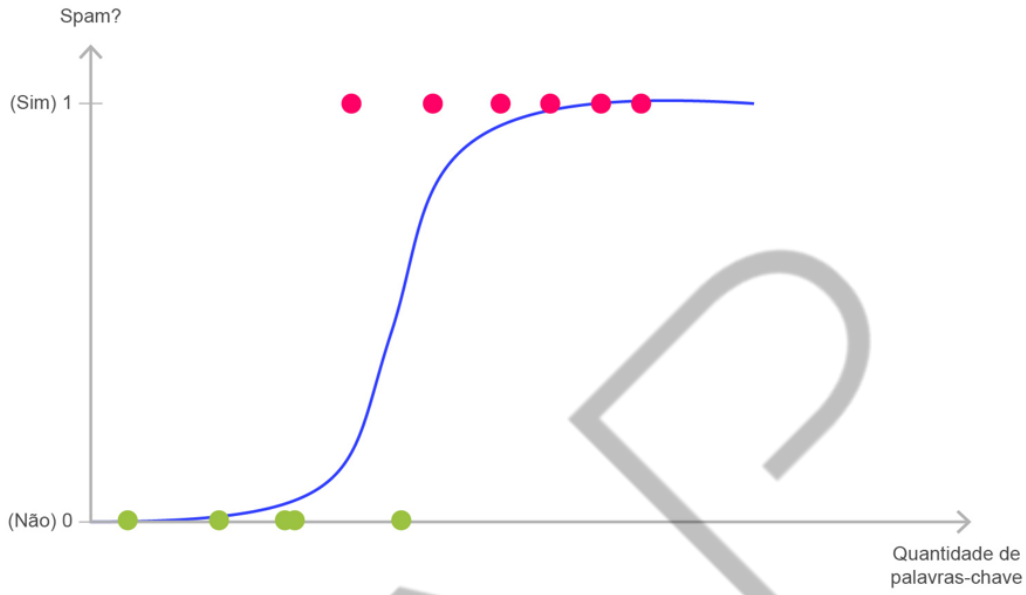

Figura 12 - Representação da regressão logística sobre a função sigmoide
Fonte: Brains (2023)

4.2.1 Funcionamento do algoritmo

A regressão logística passa por três etapas bem definidas:

1. **Modelo Linear:** a regressão logística começa como uma regressão linear, onde uma combinação linear das variáveis de entrada (features) é calculada. Isso pode ser expresso matematicamente como:

$$Z = \beta 0 + \beta 1x1 + \beta 2x2 + \ldots + \beta nxn$$

Onde “$\beta 0$” é o intercepto e “$\beta 1, \beta 2, \ldots, \beta n$” são os coeficientes das variáveis de entrada “$x1, x2, \ldots xn$”.

2. **Função sigmoide:** a saída do modelo linear “Z” é passada por uma função sigmoide, que mapeia qualquer valor real em um intervalo de 0 a 1, representando a probabilidade de a observação pertencer a uma classe específica:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

O resultado da função sigmoide é a probabilidade de que a observação pertença à classe “positiva” (geralmente codificada como 1).

3. **Classificação:** finalmente, a probabilidade gerada pela função sigmoide é comparada a um limiar (geralmente 0,5). Se a probabilidade for maior que o limiar, o modelo classifica a observação como pertencente à classe positiva, caso contrário, à classe negativa.

4.2.2 Vantagens e desvantagens do método

Como todo algoritmo preditivo, a Regressão Logística possui vantagens e desvantagens, como apresentadas no quadro a seguir.

Vantagens	Desvantagens
Simplicidade e interpretabilidade: a regressão logística é fácil de implementar e interpretar, com os coeficientes β representando a influência de cada variável na probabilidade de um evento.	Assume linearidade: a regressão logística assume uma relação linear entre as variáveis de entrada e a variável dependente, o que pode não ser apropriado para todos os problemas.
Probabilidade direta: ao prever probabilidades, a regressão logística permite uma interpretação clara do resultado em termos de risco ou probabilidade de ocorrência de um evento.	Limitação para classes múltiplas: embora existam extensões para a classificação multiclasse (como regressão logística multinomial), a regressão logística simples é restrita a problemas binários.
Rápida e eficiente: funciona bem em problemas de classificação binária, especialmente com conjuntos de dados relativamente pequenos e lineares.	Sensível a outliers: a presença de outliers pode distorcer a função de decisão, influenciando negativamente o modelo.

Quadro 2 - Vantagens e desvantagens da Regressão Logística
Fonte: Elaborado pelo autor (2024)

4.2.3 Parametrização e influência no resultado

Em relação aos hiperparâmetros de Regressão Logística, ou seja, as fontes de configuração do algoritmo, temos as seguintes alternativas:

- **Coeficientes β:** os coeficientes são os parâmetros que o modelo ajusta durante o treinamento. Eles indicam a força e a direção da relação entre cada variável de entrada e a probabilidade de ocorrência da classe positiva.
- **Regularização (L1 e L2):** a regularização é uma técnica usada para evitar overfitting, adicionando uma penalidade à função de custo. O L1 (Lasso)

regulariza somando o valor absoluto dos coeficientes, podendo levar à seleção de variáveis. O L2 (Ridge) regulariza somando o quadrado dos coeficientes, o que tende a diminuir a magnitude dos coeficientes sem zerá-los. A regularização influencia diretamente a complexidade do modelo e sua capacidade de generalizar para novos dados.

- **Limiar de decisão:** o limiar padrão é 0,5, mas pode ser ajustado dependendo do problema. Alterar o limiar pode aumentar a sensibilidade (revocação) ou a especificidade (precisão), dependendo de qual tipo de erro (falsos positivos ou falsos negativos) é mais crítico para o problema.

A escolha do tipo e da intensidade da regularização (L1 ou L2) influencia a complexidade do modelo. Uma regularização forte pode simplificar o modelo e prevenir overfitting, mas também pode levar a um subajuste (underfitting) se for muito intensa. Por outro lado, modificar o limiar de decisão permite ao modelo ser mais conservador ou mais liberal nas previsões. Por exemplo, em problemas onde a minimização de falsos negativos é crucial (como em diagnósticos médicos), um limiar menor pode ser preferível.

A Regressão Logística é uma ferramenta versátil e poderosa para problemas de classificação binária. Apesar da sua simplicidade e interpretabilidade, sua eficácia depende da compreensão e ajuste cuidadoso dos parâmetros para garantir que o modelo seja bem ajustado às características do problema.

4.3 Árvore de Decisão

A Árvore de Decisão é um algoritmo de aprendizado supervisionado usado para problemas de classificação e regressão. O objetivo por trás da Árvore de Decisão é dividir repetidamente o conjunto de dados em subconjuntos mais homogêneos, com base nos valores das variáveis de entrada, até que cada subconjunto pertença a uma única classe ou contenha um número mínimo de exemplos. O algoritmo funciona como um processo de tomada de decisões, onde cada “ramificação” da árvore representa uma decisão com base em uma característica, e cada “folha” representa uma classe ou valor final.

Considere por exemplo a árvore ilustrada na figura “Exemplo de uma Árvore de Decisão”, que decide se uma pessoa deve ir à praia ou não (*label*) baseado nas variáveis “dia com sol” (sim ou não), “dia com vento” (sim ou não).

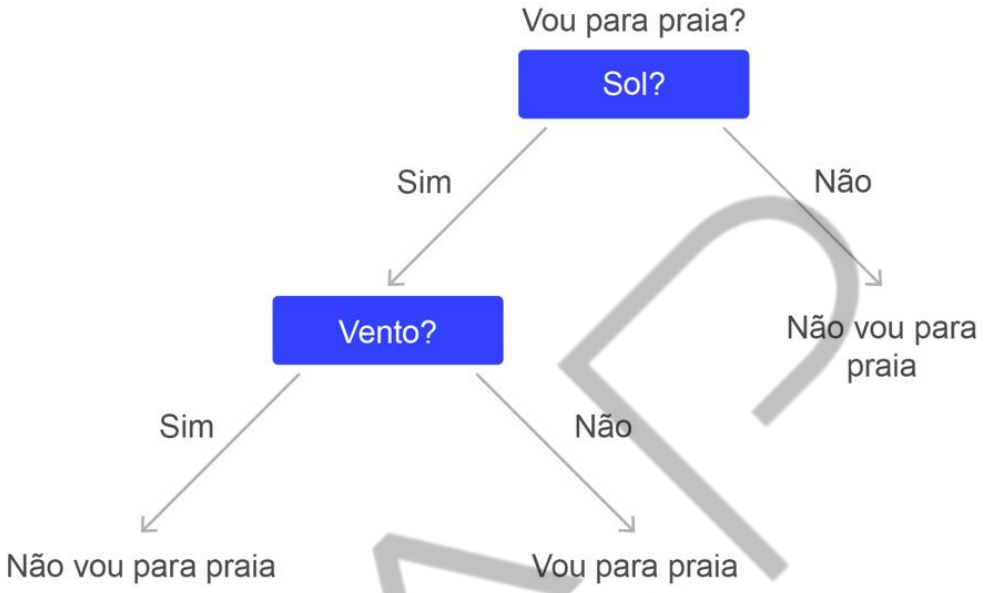

Figura 13 - Exemplo de uma Árvore de Decisão
Fonte: Didática Tech (2024)

Conforme percorremos pela árvore podemos ver a resposta dada pelo algoritmo, por exemplo, se “**há** sol” e “**não há** vento”, então irei para a praia. Nós chamamos cada “seta” de ramo, cada variável de “nó”, sendo que o primeiro é **o nó raiz** (o mais relevante) e os últimos são **os nós de decisão** (ou folhas). Os demais são os nós intermediários. No exemplo dado pela figura, temos dois níveis de decisão, portanto a altura da árvore é dois.

A construção da árvore (treinamento do modelo) consiste em identificar quais variáveis são mais relevantes para serem escolhidas como os nós dos primeiros níveis, de modo que cada partição (ou seja, cada ramo) fique com um conjunto de exemplos cada vez mais puro.

4.3.1 Funcionamento do algoritmo

A árvores de decisão é construída em três etapas:

1. **Seleção da Melhor Divisão:** tudo começa com a escolha da variável de entrada (nó raiz) que melhor divide os dados em termos de homogeneidade das classes. O critério de seleção pode variar, mas os mais comuns são o

Gini Index, Entropia (para classificação) e o Mean Squared Error (MSE) para regressão. O objetivo é escolher a divisão que resulta nos subconjuntos mais puros possíveis.

2. **Divisão Recursiva:** a partir do nó raiz, o conjunto de dados é dividido em dois ou mais subconjuntos com base na variável escolhida. Esse processo é repetido recursivamente para cada subconjunto, formando novos nós, até que uma condição de parada seja atingida. As condições de parada podem ser um número mínimo de exemplos em um nó, uma profundidade máxima da árvore ou a pureza completa dos nós.
3. **Classificação ou Predição:** uma vez que a árvore é construída, um novo exemplo é classificado percorrendo a árvore desde a raiz até uma folha, seguindo as decisões (condições) estabelecidas em cada nó. A classe ou valor da folha onde o exemplo termina é a previsão do modelo.

4.3.2 Vantagens e desvantagens do método

Assim como todo algoritmo preditivo, a Árvore de Decisão possui vantagens e desvantagens, como apresentadas no quadro a seguir:

Vantagens	Desvantagens
Interpretação simples: as Árvores de Decisão são fáceis de entender e interpretar, pois seguem uma lógica de tomada de decisão semelhante à humana, com decisões baseadas em perguntas sequenciais.	Propensão ao overfitting: as Árvores de Decisão podem se tornar excessivamente complexas e se ajustar demais ao conjunto de dados de treinamento, resultando em um modelo que não generaliza bem para novos dados.
Flexibilidade: podem ser usadas para problemas de classificação e regressão e não exigem suposições sobre a distribuição dos dados.	Instabilidade: pequenas variações nos dados de treinamento podem resultar em árvores completamente diferentes, pois as decisões são dependentes dos dados.
Manuseio de dados categóricos e numéricos: a Árvore de Decisão pode lidar com ambos os tipos de dados, categóricos e numéricos, sem necessidade de conversão prévia.	Tendência a preferir variáveis com mais níveis: o algoritmo tende a preferir variáveis com mais níveis, o que pode distorcer a interpretação dos resultados.

Quadro 3 - Vantagens e desvantagens da Árvore de Decisão
Fonte: Elaborado pelo autor (2024)

4.3.3 Parametrização e influência no resultado

Existem diversos hiperparâmetros que influenciam a construção da Árvore de Decisão. Os mais comuns são:

- **Critério de divisão:** o critério escolhido para avaliar a qualidade da divisão, como Gini Index, Entropia ou MSE, influencia como a árvore será construída. Cada critério tem suas próprias vantagens, como o Gini Index, que tende a ser mais rápido, ou a Entropia, que considera a pureza das divisões mais detalhadamente.
- **Profundidade máxima (*Max Depth*):** controla o número máximo de níveis na árvore. Limitar a profundidade pode ajudar a evitar o overfitting, criando um modelo mais simples e generalizável.
- **Número mínimo de exemplos por nó (*Min Samples Split*):** define o número mínimo de exemplos necessários para que um nó possa ser dividido. Um valor maior pode resultar em árvores menos complexas e mais generalizáveis.
- **Número mínimo de exemplos em um nó folha (*Min Samples Leaf*):** define o número mínimo de exemplos necessários para formar um nó folha. Isso ajuda a suavizar a árvore e pode prevenir overfitting em problemas de regressão.

De modo geral, uma árvore mais profunda permite capturar mais detalhes dos dados, mas também aumenta o risco de overfitting. Reduzir a profundidade força a árvore a generalizar mais, o que pode melhorar o desempenho em dados de teste.

Em relação ao número mínimo de exemplos por nó, um valor alto pode resultar em uma árvore menos complexa, mas também pode deixar de capturar padrões importantes se o valor for excessivamente restritivo (underfitting). Um valor baixo permite divisões mais frequentes, mas pode levar ao overfitting.

Portanto, as Árvores de Decisão são uma ferramenta versátil e poderosa para uma ampla variedade de problemas de aprendizado supervisionado. Entretanto, seus ganhos de simplicidade e interpretabilidade (extremamente vantajosos em aplicações

que exigem explicabilidade), precisam ser balanceados com boas configurações de hiperparâmetros, evitando as armadilhas de underfitting ou overfitting.

4.4 Floresta Aleatória

A Floresta Aleatória (*Random Forest*) é um poderoso algoritmo de aprendizado supervisionado que **combina várias Árvores de Decisão para melhorar a precisão e a robustez do modelo**. A intuição por trás da Floresta Aleatória é que a combinação de múltiplas Árvores de Decisão, construídas de forma aleatória e independente, resulta em um modelo mais estável e preciso do que uma única árvore (ilustrado na figura “Processo de decisão em uma Floresta Aleatória”).

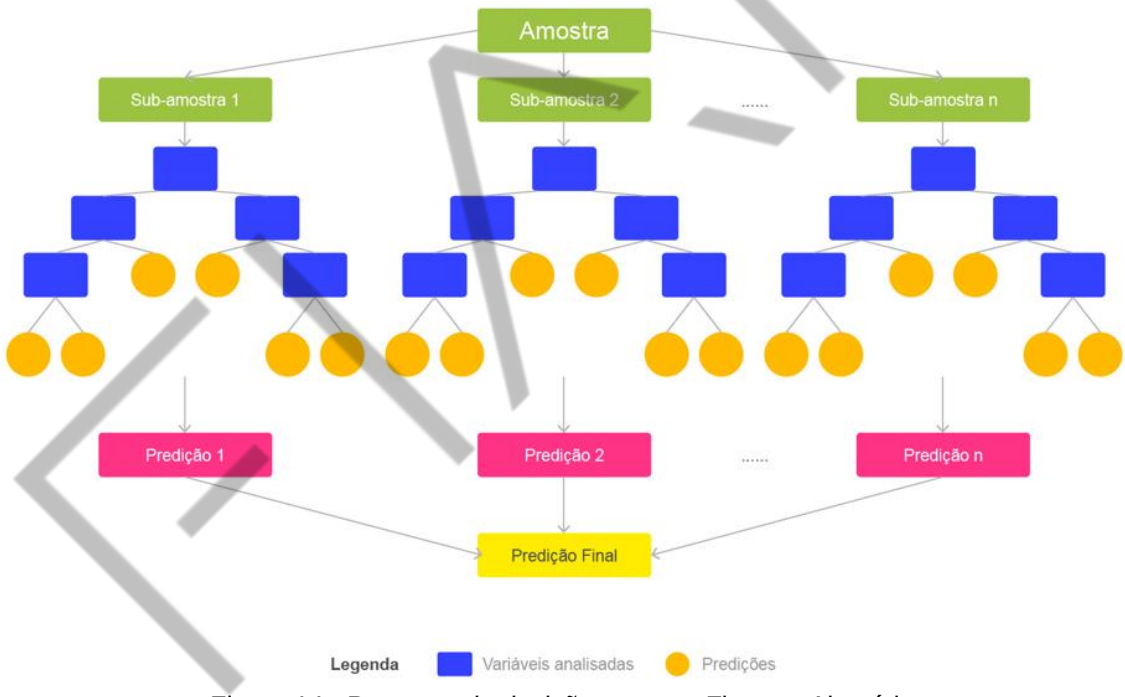

Figura 14 - Processo de decisão em uma Floresta Aleatória
Fonte: Researchgate (2020)

Esse processo de combinação de múltiplos modelos é conhecido como *“ensemble learning”* (aprendizado em conjunto), onde cada árvore contribui com uma “votação” para a decisão final, tornando o modelo menos propenso a overfitting.

Ao combinar os resultados de várias Árvores de Decisão, a Floresta Aleatória reduz o risco de overfitting, pois erros individuais das árvores tendem a se cancelar. O modelo final é mais robusto e generaliza melhor para dados desconhecidos.

4.4.1 Funcionamento do algoritmo

A Floresta Aleatória é construída em dois grandes passos:

1. **Construção de Múltiplas Árvores de Decisão:** a Floresta Aleatória cria várias Árvores de Decisão independentes. Cada árvore é treinada utilizando um subconjunto aleatório dos dados de treinamento, selecionado com reposição (técnica conhecida como "*bootstrap sampling*"). Além disso, em cada nó de uma árvore, um subconjunto aleatório de características (features) é escolhido para a divisão, o que introduz mais diversidade entre as árvores.
2. **Votação ou Agregação:** para uma nova previsão, o modelo consulta todas as árvores da floresta. No caso de classificação, cada árvore "vota" em uma classe, e a classe com o maior número de votos é a previsão final do modelo. No caso de regressão, as previsões de todas as árvores são agregadas (por exemplo, fazendo a média) para produzir a previsão final.

4.4.2 Vantagens e desvantagens do método

Quando comparada com a Árvore de Decisão, a Floresta Aleatória possui vantagens e desvantagens, resumidas no quadro a seguir:

Vantagens	Desvantagens
Melhor generalização: a Floresta Aleatória tende a generalizar melhor do que uma única Árvore de Decisão, pois combina várias árvores, mitigando o risco de overfitting.	Complexidade e custo computacional: o processo pode ser computacionalmente caro, especialmente para grandes conjuntos de dados, devido à necessidade de construir e consultar múltiplas árvores.
Robustez: é menos sensível a outliers e variações nos dados do que uma única Árvore de Decisão.	Interpretação mais difícil: enquanto uma única Árvore de Decisão é interpretável, uma Floresta Aleatória não é.
Trabalha bem com dados desbalanceados: a Floresta Aleatória pode lidar melhor com classes desbalanceadas, especialmente quando o número de árvores é grande.	Risco de overfitting em casos específicos: embora a Floresta Aleatória reduza o risco de overfitting, pode ainda ocorrer em casos específicos, especialmente se não for parametrizada corretamente.

Quadro 4 - Vantagens e desvantagens da Floresta Aleatória
Fonte: Elaborado pelo autor (2024)

4.4.3 Parametrização e influência no resultado

Os hiperparâmetros das Florestas Aleatórias são essencialmente os mesmos que os da Árvore de Decisão. Em relação à floresta, existem dois hiperparâmetros exclusivos que valem a pena ser mencionados:

- **Número de árvores (n_estimators):** é o número de árvores a serem construídas na floresta. Um número maior de árvores geralmente melhora a performance do modelo, mas aumenta o custo computacional.
- **Número de características a serem consideradas em cada divisão (max_features):** especifica quantas características são consideradas para divisão em cada nó. Valores comuns incluem a raiz quadrada do número total de características ou um valor fixo pequeno. Com valores menores, as árvores são mais diversificadas, enquanto valores maiores podem aumentar a precisão individual das árvores.

O custo computacional de treinar uma Floresta Aleatória aumenta linearmente com o número de árvores. Ainda que mais árvores geralmente melhore a estabilidade e a precisão do modelo, o retorno tende a diminuir após um certo ponto.

A Floresta Aleatória é uma escolha popular para muitos problemas de classificação e regressão devido à sua robustez e capacidade de generalizar bem para dados desconhecidos. Com o ajuste certo, a Floresta Aleatória pode ser uma ferramenta poderosa em qualquer pipeline de aprendizado supervisionado.

4.5 Support Vector Machine

As máquinas de vetores de suporte (SVM, do inglês Support Vector Machine) são um poderoso algoritmo de aprendizado supervisionado utilizado tanto para classificação quanto para regressão.

A intuição por trás do SVM é encontrar o hiperplano que melhor separa as classes no espaço de características, de modo que a margem entre as duas classes seja maximizada. A figura “Intuição do SVM” mostra a intuição da classificação binária

via SVM usando duas variáveis (X, Y). Neste caso, o hiperplano de separação é uma reta e o mecanismo é estendível para problemas multiclasse em mais dimensões.

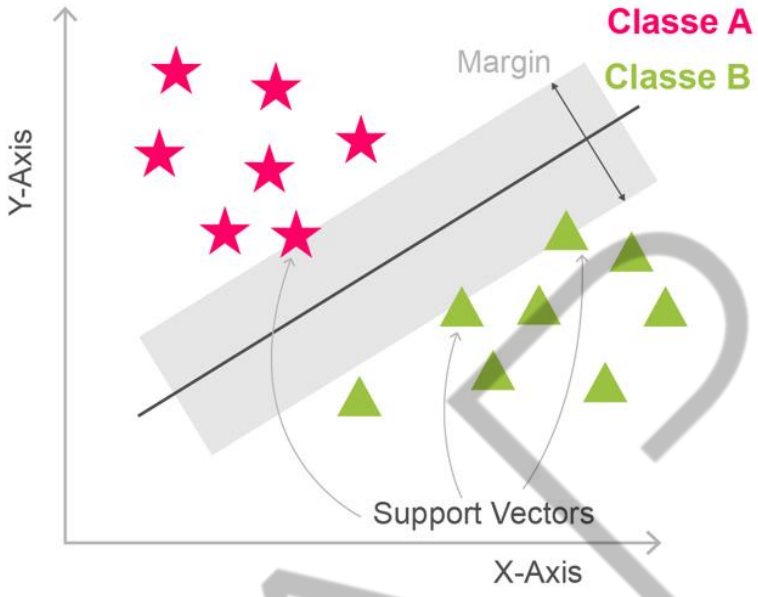

Figura 15 - Intuição do SVM
Fonte: Medium (2023)

A figura também destaca o que são os “vetores de suporte”. Eles são os pontos de cada classe do conjunto de treinamento mais próximos entre si, e que justamente “suportam” (apoiam) o posicionamento do hiperplano (a reta, no caso). Quanto mais distantes os vetores de suporte, maior é a margem, portanto mais robusta é a classificação — ou seja, haverá uma boa separação entre as classes sem margem para predições incorretas.

O SVM busca um hiperplano ótimo que não apenas separa as classes, mas que também maximiza a distância mínima entre qualquer ponto de dado e o hiperplano, garantindo assim que o modelo seja robusto e generalize bem para novos dados. Por ser um algoritmo que também depende do conceito de distância de pontos, **é imprescindível normalizar ou padronizar os dados** antes de iniciar o treinamento.

4.5.1 Funcionamento do algoritmo

O SVM passa por quatro etapas durante o treinamento do algoritmo. São elas:

1. **Definição do hiperplano:** em um espaço de características *{N}*-dimensional, um hiperplano é uma superfície *{N-1}*-dimensional que separa

os pontos de dados em diferentes classes. Em um problema de classificação binária, o SVM tenta encontrar o hiperplano que maximiza a margem entre as duas classes. A margem é definida como a distância entre o hiperplano e os pontos de dados mais próximos de qualquer classe, conhecidos como vetores de suporte.

2. **Maximização da Margem:** o objetivo do SVM é maximizar essa margem, pois um hiperplano com uma margem maior tende a ser mais robusto a novos dados. Os vetores de suporte são os pontos de dados que estão mais próximos do hiperplano e são essenciais para definir a posição do hiperplano ótimo.
3. **Kernel Trick:** se os dados não são linearmente separáveis no espaço original, o SVM pode utilizar uma técnica chamada 'kernel trick'. Essa técnica mapeia os dados para um espaço de maior dimensionalidade onde eles podem ser linearmente separáveis. Existem vários tipos de kernels, como o kernel linear, polinomial, e o mais comum, o kernel *RBF* (Radial Basis Function), que permite ao SVM encontrar limites de decisão não lineares.
4. **Classificação:** depois que o hiperplano é definido, novos pontos de dados são classificados com base em qual lado do hiperplano eles caem: se estiverem de um lado, são classificados como pertencentes a uma classe; se estiverem do outro lado, pertencem à outra classe.

4.5.2 Vantagens e desvantagens do método

Por ser um algoritmo mais robusto e complexo que os anteriores, o SVM pode levar a resultados mais precisos. No entanto, não há garantias que ele sempre se adequará bem ao conjunto de dados e/ou ao problema modelado. Assim, também é importante conhecer suas vantagens e desvantagens. Observe o quadro:

Vantagens	Desvantagens
Eficácia em espaços de alta dimensão: SVM é eficaz em espaços de alta dimensionalidade (muitas variáveis), e ainda funciona bem mesmo quando o número de dimensões é maior que o número de amostras.	Complexidade computacional: SVM pode ser computacionalmente intenso, especialmente em grandes conjuntos de dados, tanto em termos de tempo de treinamento quanto de memória.
Uso do kernel trick: a capacidade de usar kernels permite que o SVM lide com problemas não lineares de forma eficaz, mapeando os dados para um espaço onde eles se tornam linearmente separáveis.	Escolha do kernel: a escolha do kernel e dos parâmetros associados é crítica e pode ser complicada, exigindo muita experimentação para encontrar o melhor desempenho.
Margem máxima: a maximização da margem entre classes aumenta a robustez do modelo e melhora sua capacidade de generalização.	Interpretabilidade: o modelo resultante, especialmente com kernels não lineares, pode ser difícil de interpretar em termos de como as decisões são tomadas.

Quadro 5 - Vantagens e desvantagens do SVM
Fonte: Elaborado pelo autor (2024)

4.5.3 Parametrização e influência no resultado

Existem essencialmente três hiperparâmetros no SVM. Sua definição e conceito pode ser um pouco confusa devido à influência inversamente proporcional no resultado. Portanto, recomendamos ler com atenção e observar as imagens seguintes para melhor compreensão:

- **Escolha do kernel:** a seleção do kernel é um dos aspectos mais críticos do SVM. O kernel linear é adequado para dados linearmente separáveis, enquanto kernels mais complexos, como o RBF, são utilizados para dados não lineares. A escolha do kernel pode afetar significativamente a performance do modelo. O kernel essencialmente indica a “forma” do hiperplano de separação, sendo mais ou menos linear, conforme ilustrado na figura “Tipos de kernel do SVM”:

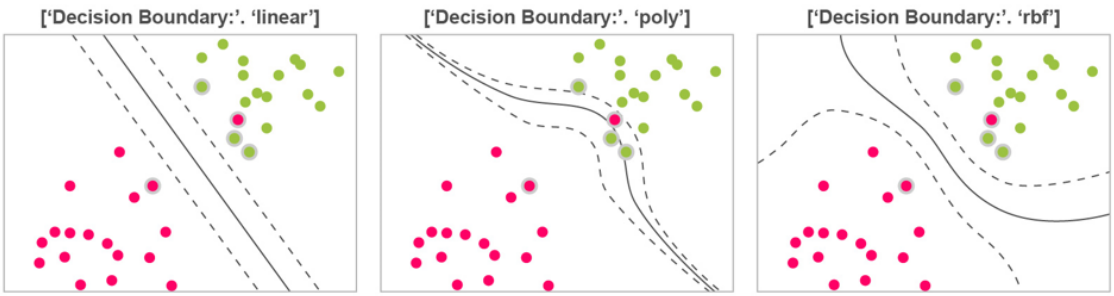

Figura 16 - Tipos de kernel do SVM
Fonte: Medium (2018)

- **Parâmetro C:** o parâmetro **C** controla a *trade-off* entre maximizar a margem e minimizar o erro de classificação. Um valor alto de **C** penaliza mais os erros de classificação, resultando em uma margem menor, mas com menos violações; um valor baixo de **C** permite uma margem maior, mas com mais violações. A figura “Influência do hiperparâmetro C na margem do SVM” ilustra o impacto do valor de **C** na grossura da margem.

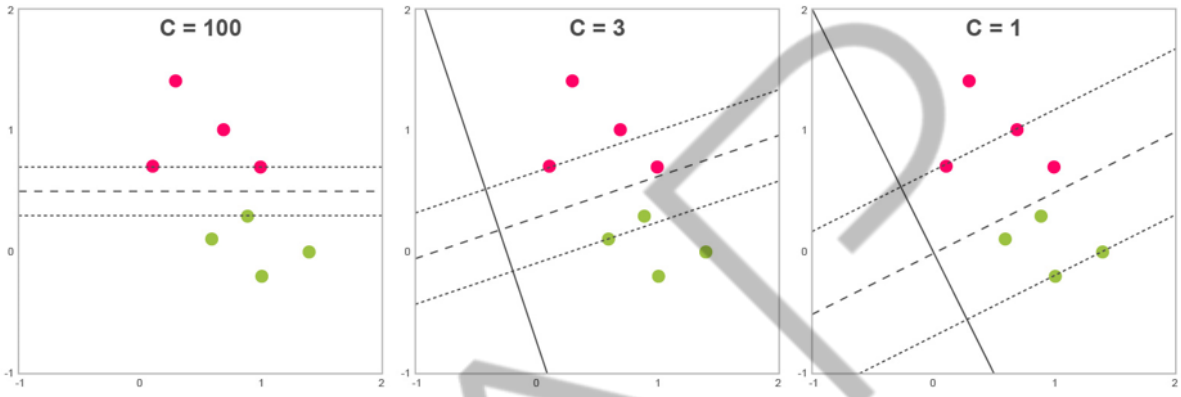

Figura 17 - Influência do hiperparâmetros C na margem do SVM
Fonte: Medium (2017)

- **Parâmetro γ (gamma) (para kernels não lineares):** o parâmetro γ no kernel RBF controla a influência de um único exemplo de treinamento. Valores **altos** de γ significam que cada ponto de dado tem uma esfera de influência menor, o que pode levar ao overfitting devido a um hiperplano mais rígido; valores **baixos** de γ levam a uma esfera de influência maior, resultando em um hiperplano mais suave. Observe a figura:

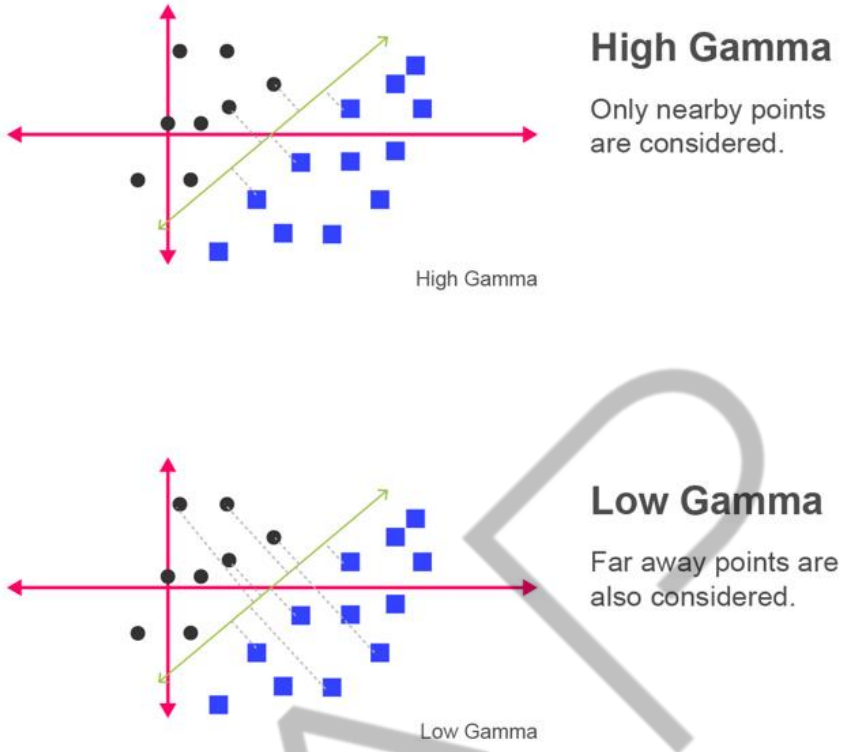

Figura 18 - Influência do hiperparâmetros γ (gamma) na margem do SVM
Fonte: Medium (2020)

O SVM é uma ferramenta poderosa e flexível, especialmente eficaz em problemas de alta dimensionalidade e onde a separação não é linear. No entanto, seu desempenho depende fortemente da escolha apropriada do kernel e da configuração correta dos hiperparâmetros, exigindo uma compreensão detalhada do problema e dos dados em questão.

Um kernel inadequado pode resultar em um modelo que não captura corretamente as relações entre as variáveis de entrada e a classe de saída, ocasionando um desempenho insatisfatório. Por esse motivo, é essencial testar diferentes kernels para encontrar o kernel mais apropriado.

O hiperparâmetros **C controla a grossura do hiperplano**, enquanto o **gamma controla sua flexibilidade** (mais ou menos rígido, ou seja, com mais ou menos curvas).

5 HANDS ON: ANÁLISES PREDITIVAS DE CLASSIFICAÇÃO

Agora que aprendemos os principais algoritmos de aprendizado supervisionado de classificação, está na hora implementarmos sobre um dataset real.

Toda nossa prática será feita em um jupyter notebook no ambiente do Google Colaboratory (Colab).

Dica: veja o material HOW TO para acessar o ambiente do Colab.

Uma vez no Colab, crie um novo notebook, dê um nome como preferir (sugestão: Aula 2), e faça cada bloco de código fonte em uma célula. Após inserir o código na célula do notebook, execute-a e veja a saída gerada.

Dica: utilize o arquivo de apoio “fertilizer_prediction.csv”.

5.1 Sobre a base de dados

Faremos o estudo de caso do dataset “**Fertilizer Prediction**”, disponível no Kaggle. Este dataset contém informações sobre o tipo de solo, temperatura, umidade, pH, produto cultivado e outros fatores, com o objetivo de prever o tipo adequado de fertilizante a ser utilizado.

No Agronegócio, a utilização do fertilizante adequado é crucial para maximizar o rendimento das culturas, garantindo que as plantas recebam os nutrientes necessários para um crescimento saudável. Por isso, é necessário escolher o fertilizante certo com base nas condições do solo e ambientais para garantir uma nutrição equilibrada, reduzindo o desperdício de recursos e o impacto ambiental. Isso também contribui para a sustentabilidade agrícola, preservando a fertilidade do solo a longo prazo e melhorando a qualidade das colheitas.

5.2 Sobre a biblioteca Sklearn

A biblioteca **Scikit-Learn (Sklearn)** é uma das ferramentas mais populares e amplamente utilizadas para a implementação de modelos de aprendizado de máquina

em Python. Desenvolvida inicialmente como um projeto de código aberto, a biblioteca fornece uma vasta gama de algoritmos de aprendizado de máquina supervisionados e não supervisionados, incluindo classificação, regressão, clusterização e redução de dimensionalidade. Além disso, sua documentação é impecável.

Uma das principais vantagens do Sklearn é sua simplicidade e consistência na interface de usuário, o que facilita o uso até mesmo para iniciantes em Machine Learning. A biblioteca é construída sobre outras bibliotecas científicas robustas como NumPy, SciPy e Matplotlib, permitindo um desempenho eficiente e uma integração fluida com outras ferramentas do ecossistema Python.

Além de algoritmos, o Scikit-Learn também oferece ferramentas para pré-processamento de dados, seleção de modelos, validação cruzada e pipelines, que ajudam a criar fluxos de trabalho de Machine Learning completos, desde a limpeza de dados até a implementação de modelos preditivos. Com uma comunidade ativa de desenvolvedores e uma documentação abrangente, o Sklearn é uma escolha excelente tanto para pesquisadores quanto para profissionais que buscam aplicar aprendizado de máquina em problemas reais.

A primeira técnica de aprendizado de máquina

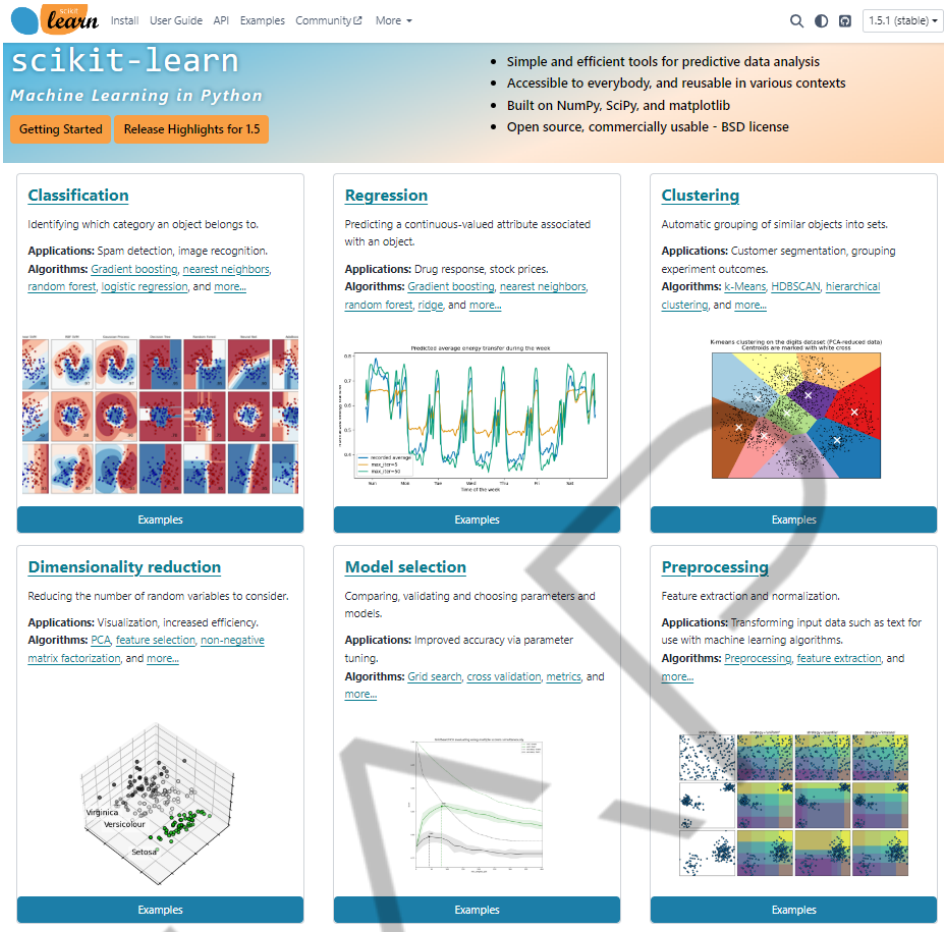

Figura 19 - Página inicial da documentação do Skicit-Learn (Sklearn)
Fonte: Elaborado pelo autor (2024)

A figura anterior mostra a página inicial da documentação do Sklearn. Podemos ver a implementação de diversas famílias de algoritmos de Machine Learning. Além disso, a documentação é rica em exemplos e tutoriais. A configuração dos hiperparâmetros de cada modelo é detalhada na documentação. Não deixe de consultá-la!

Observação: a documentação do Sklearn é excelente! Não deixe consultá-la no site: <https://scikit-learn.org/stable/>

5.3 Implementando nossa análise

O primeiro passo é a importação do código necessário, leitura e exibição dos dados (ou de uma parte deles).

A primeira técnica de aprendizado de máquina

```
# Imports
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder,
OneHotEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,
classification_report

# Carregar o dataset
df = pd.read_csv("fertilizer_prediction.csv")

# Exibir as primeiras linhas do DataFrame
df.head()
```

Código-fonte 1 - Imports e leitura do dataset
Fonte: Elaborado pelo autor (2024)

Este código exibe as primeiras cinco linhas do dataset, ilustrado na figura “Resultado do código-fonte 1”, nas quais vemos a presença de nove colunas:

	Temperature	Humidity	Moisture	Soil Type	Crop Type	Nitrogen	Potassium	Phosphorous	Fertilizer Name
0	26	52	38	Sandy	Maize	37	0	0	Urea
1	29	52	45	Loamy	Sugarcane	12	0	36	DAP
2	34	65	62	Black	Cotton	7	9	30	14-35-14
3	32	62	34	Red	Tobacco	22	0	20	28-28
4	28	54	46	Clayey	Paddy	35	0	0	Urea

Figura 20 - Resultado do código-fonte 1
Fonte: Elaborada pelo autor (2024)

Agora, podemos começar a exploração de dados que guiará a limpeza da base! No código a seguir veremos os aspectos gerais da base:

```
# Mostra informações gerais sobre a base
df.info()
```

Código-fonte 2 - Exibindo informações gerais da base
Fonte: Elaborado pelo autor (2024)

A figura “Resultado do código-fonte 2” mostra que a base é composta por 99 linhas (exemplos) e nove colunas (variáveis). Todas são numéricas, exceto o tipo de solo, cultura e nome do fertilizante. Teremos que tratá-las adiante. Além disso, não há dados faltantes, portanto aqui não precisamos nos preocupar com *missings*.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Temperature      99 non-null     int64
 1   Humidity         99 non-null     int64
 2   Moisture         99 non-null     int64
 3   Soil Type        99 non-null     object
 4   Crop Type        99 non-null     object
 5   Nitrogen         99 non-null     int64
 6   Potassium        99 non-null     int64
 7   Phosphorous      99 non-null     int64
 8   Fertilizer Name  99 non-null     object
dtypes: int64(6), object(3)
memory usage: 7.1+ KB
```

Figura 21 - Resultado do código-fonte 2
Fonte: Elaborada pelo autor (2024)

Mesmo não tendo valores ausentes, outros problemas podem aparecer, como outliers e dados duplicados. Vejamos como a base se apresenta nestes quesitos:

```
# Imprime informações relevantes à limpeza

# Verificar a presença de dados duplicados
duplicates = df.duplicated().sum()
print("Número de dados duplicados:", duplicates)

# Verificar a presença de outliers
plt.figure(figsize=(12, 6))
sns.boxplot(data=df)
plt.title("Boxplot para detectar outliers")
plt.xticks(rotation=45)
plt.show()
```

Código-fonte 3 - Buscando por dados duplicados e outliers
Fonte: Elaborado pelo autor (2024)

O código-fonte “Buscando por dados duplicados e outliers” aponta que não há dados duplicados, e a figura “Resultado do código-fonte 3” apresenta um *boxplot* de cada variável numérica. Ainda que as distribuições dos valores variem, há apenas um outlier para *Potassium*. Vamos ignorar este outlier e não nos preocupar em retirá-lo, pois está pouco discrepante em relação ao valor máximo.

A primeira técnica de aprendizado de máquina

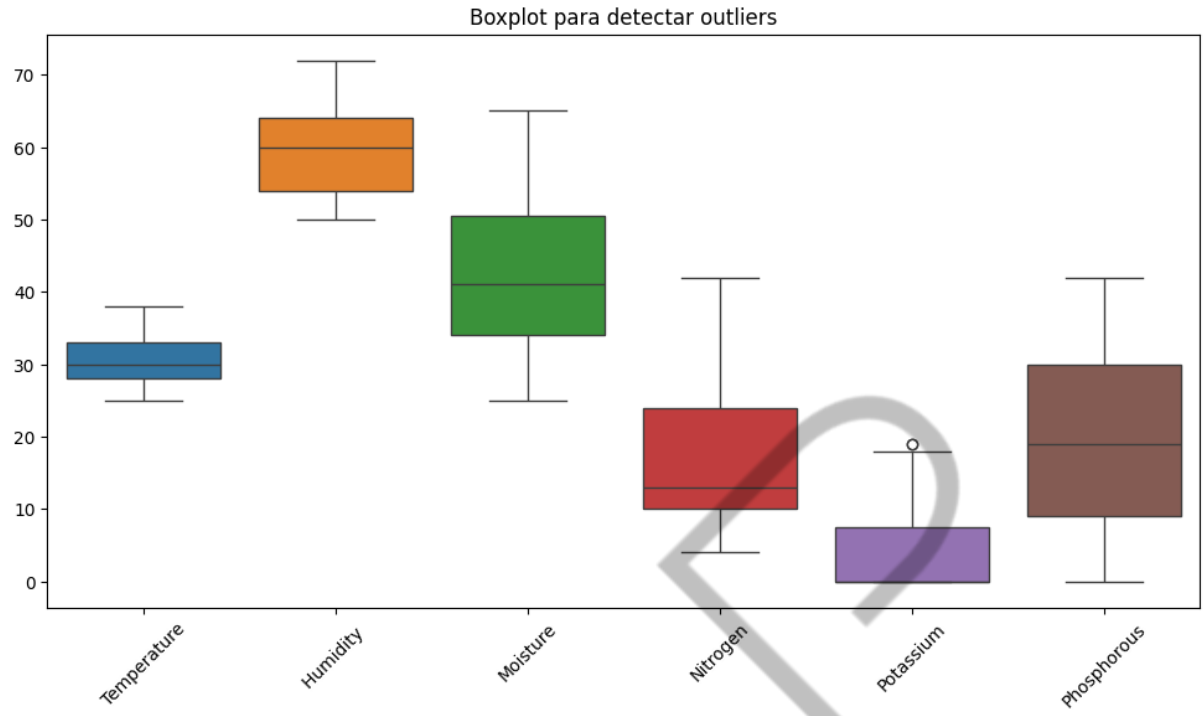

Figura 22 - Resultado do código-fonte 3
Fonte: Elaborada pelo autor (2024)

Também podemos explorar a distribuição dos *labels*:

```
# Exploração da distribuição dos labels
plt.figure(figsize=(10, 6))
sns.countplot(x='Fertilizer Name', data=df)
plt.title("Distribuição dos Tipos de Fertilizantes")
plt.xticks(rotation=45)
plt.show()
```

Código-fonte 4 - Explorando a distribuição dos *labels*
Fonte: Elaborado pelo autor (2024)

A figura “Resultado do código-fonte 4” mostra que temos sete classes possíveis, portanto, é um problema de **classificação multiclasse**. Ainda que exista uma variação na quantidade de exemplos por classe há uma certa distribuição. As classes minoritárias são 17-17-17 e 10-26-26. Assim, a acurácia pode não ser a melhor métrica, mas podemos avaliá-la em conjunto com outras:

A primeira técnica de aprendizado de máquina

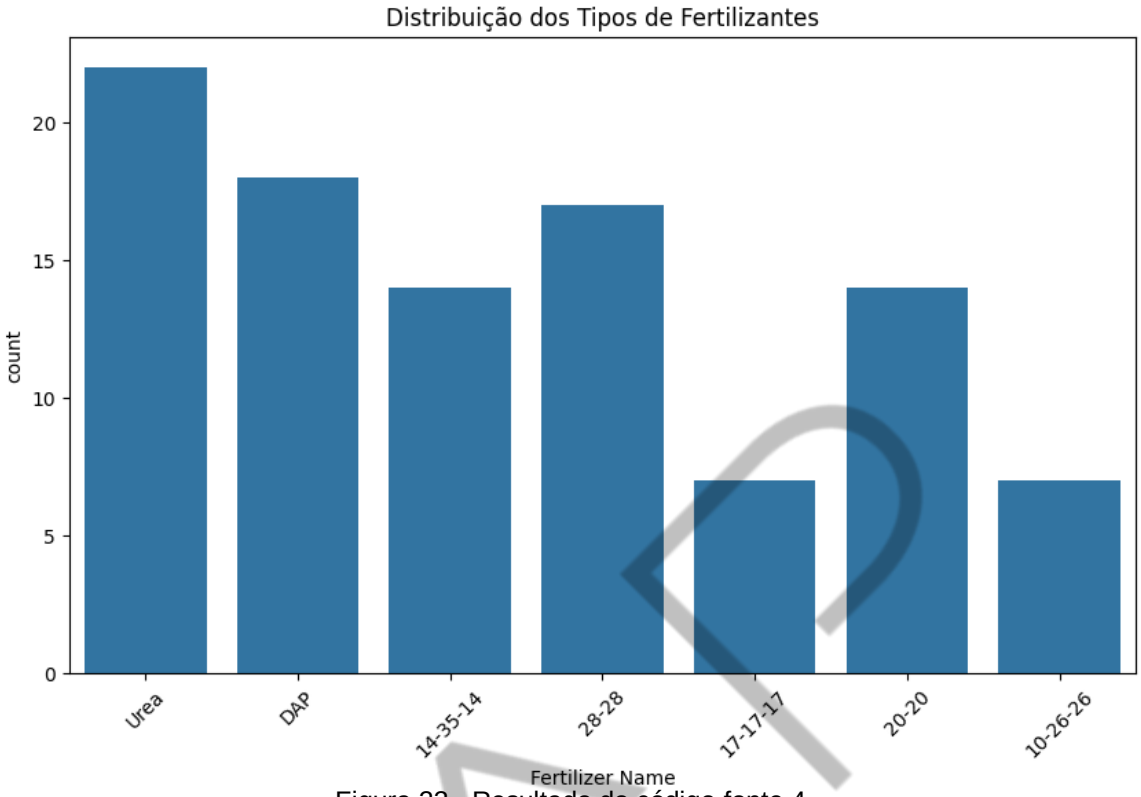

Figura 23 - Resultado do código-fonte 4
Fonte: Elaborada pelo autor (2024)

Outra boa prática em Machine Learning é **investigar a correlação entre as variáveis**. Se duas ou mais colunas estão altamente correlacionadas, pode ser interessante remover algumas delas, para facilitar a convergência do algoritmo preditivo em menor dimensão. No nosso caso, a verificação de correlação é:

```
    # Correlação entre as features numéricas
    numerics = ['int16', 'int32', 'int64', 'float16',
'float32', 'float64']
    plt.figure(figsize=(12, 8))
    sns.heatmap(df.select_dtypes(include=numerics).corr(),
annot=True, cmap='coolwarm')
    plt.title("Matriz de Correlação")
    plt.show()
```

Código-fonte 5 - Explorando a correlação das features numéricas
Fonte: Elaborado pelo autor (2024)

A figura “Resultado do código-fonte 5” mostra que de fato algumas variáveis estão fortemente correlacionadas (direta ou indiretamente). Por exemplo: “temperatura e umidade” ou “fósforo e nitrogênio”:

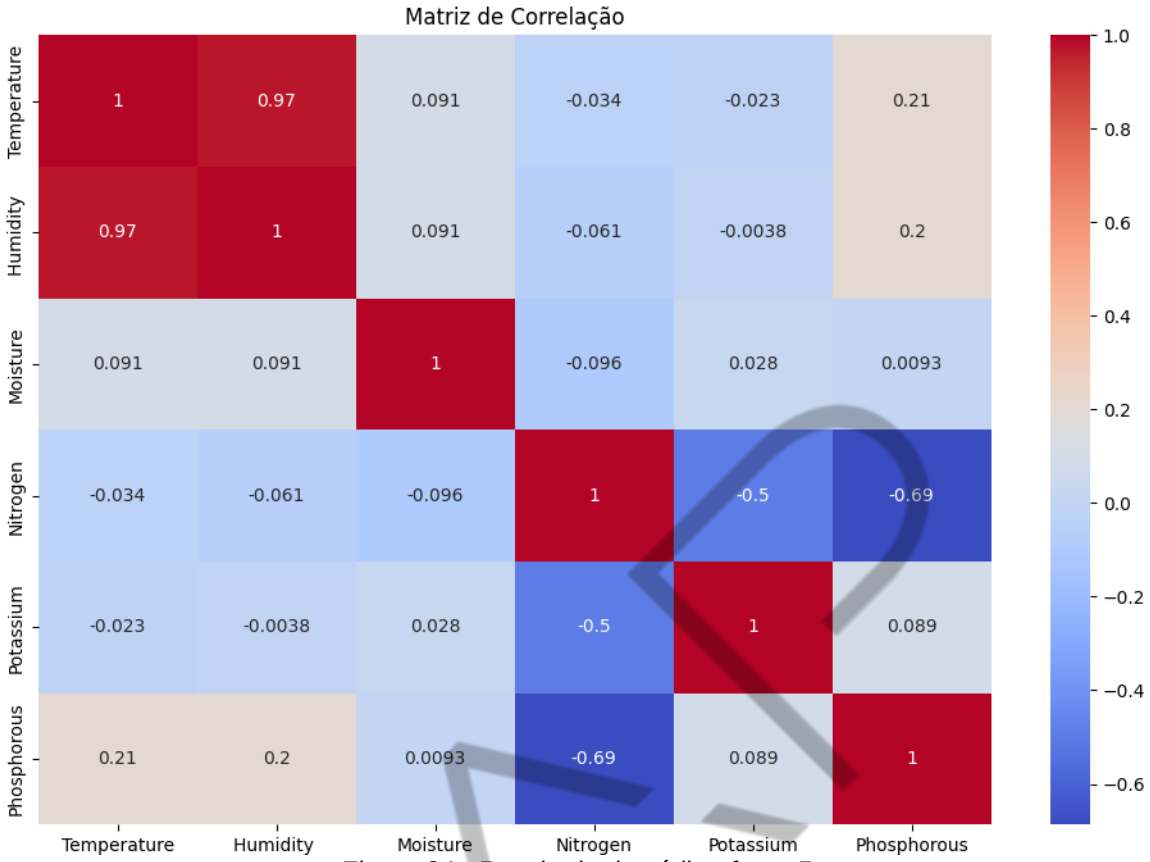

Figura 24 - Resultado do código-fonte 5
Fonte: Elaborada pelo autor (2024)

No momento não vamos nos preocupar com isso, e vamos trabalhar com todas as variáveis sendo features para o modelo preditivo.

Aprendemos que a base não apresenta problemas nos dados, mas se houvessem dados duplicados ou faltantes, poderíamos ajustá-la com os seguintes comandos:

```
    # Remover dados duplicados, se houver
    df = df.drop_duplicates()

    # Tratar outliers, se necessário (substituir por média,
mediana, etc.)
    # Exemplo: substituir outliers da coluna 'Temperature'
pela mediana
    median_temperature = df['Temperature'].median()
    df['Temperature']  =  df['Temperature'].apply(lambda  x:
median_temperature if x > df['Temperature'].quantile(0.975) or
x < df['Temperature'].quantile(0.025) else x)

    df.shape
```

Código-fonte 6 - Exemplo de limpeza de dados
Fonte: Elaborado pelo autor (2024)

O código-fonte anterior não tem efeito no dataset em questão, e apenas mostra que o dataframe resultante continua com 99 linhas e nove colunas.

Neste momento, iremos migrar para engenharia de features! Esta é uma etapa muito importante. Aqui realizaremos as seguintes ações:

1. Separamos as features **X** do *label* **y**;
2. Criamos uma codificação para os dados em string, representando-os em números via `LabelEncoder` (para dados ordinais ou *labels*) ou `OneHotEncoder` (para dados nominais);
3. Removemos as colunas originais antes dos tratamentos;
4. Separamos os dados em treino (80%) e teste (20%);
5. Normalizamos as features numéricas, sempre “aprendendo” a fórmula no treino e a aplicando no treino e no teste.

```
    # Separando features e labels
    X = df.drop('Fertilizer Name', axis=1)
    y = df['Fertilizer Name']

    # Label Encoder para a variável alvo
    le = LabelEncoder()
    y = le.fit_transform(y)

    # Lista de colunas categóricas e aplicação de One-Hot
Encoding
    categorical_cols = ['Soil Type', 'Crop Type']
    ohe = OneHotEncoder(handle_unknown='ignore')
    X_encoded                                                =
pd.DataFrame(ohe.fit_transform(X[categorical_cols]).toarray()
)
    X_encoded = X_encoded.add_prefix('OHE_')

    # Removendo colunas categóricas originais
    X = X.drop(categorical_cols, axis=1)

    # Concatenando as features codificadas com as numéricas
    X = pd.concat([X, X_encoded], axis=1)

    # Dividindo os dados em conjuntos de treino e teste (80%
treino, 20% teste)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
    # Normalização das features numéricas
    scaler = MinMaxScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_teste_scaled = scaler.transform(X_test)
```

Código-fonte 7 - Engenharia de features
Fonte: Elaborado pelo autor (2024)

O código-fonte anterior não possui saídas, mas se você quiser, pode explorar o conteúdo atual de cada variável obtida:

- `X_train`: contém os dados de treino;
- `X_test`: contém os dados de teste;
- `X_train_scaled`: contém os dados de treino já normalizados;
- `X_test_scaled`: contém os dados de teste já normalizados;
- `y_train`: contém os *labels* do treino;
- `y_test`: contém os *labels* do teste.

Agora estamos prontos para **construir os modelos preditivos**! Iniciaremos pelo baseline, que será a Regressão Logística e o KNN usando K=9:

```
    # Regressão Logística
    logreg = LogisticRegression()
    logreg.fit(X_train_scaled, y_train)
    y_pred_logreg = logreg.predict(X_teste_scaled)
    print("Acurácia Regressão Logística:",
accuracy_score(y_test, y_pred_logreg))
    print(classification_report(y_test, y_pred_logreg))

    # KNN
    knn = KNeighborsClassifier(n_neighbors=9)
    knn.fit(X_train_scaled, y_train)
    y_pred_knn = knn.predict(X_teste_scaled)
    print("Acurácia KNN:", accuracy_score(y_test,
y_pred_knn))
    print(classification_report(y_test, y_pred_knn))
```

Código-fonte 8 - Modelos preditivos iniciais
Fonte: Elaborado pelo autor (2024)

A figura “Resultado do código-fonte 8” mostra os resultados. Comparando ambos os modelos, podemos ver que a Regressão Logística performou melhor que o KNN. Isso indica que este problema pode ser mais bem identificado de forma linear.

A primeira técnica de aprendizado de máquina

```
Acurácia Regressão Logística: 0.7
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.00      0.00      0.00         1
           3       0.00      0.00      0.00         1
           4       0.60      0.60      0.60         5
           5       0.57      0.80      0.67         5
           6       1.00      1.00      1.00         6

    accuracy                           0.70        20
   macro avg       0.53      0.48      0.49        20
weighted avg       0.69      0.70      0.68        20

Acurácia KNN: 0.3
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.00      0.00      0.00         1
           3       0.00      0.00      0.00         1
           4       0.33      0.20      0.25         5
           5       0.33      0.20      0.25         5
           6       0.80      0.67      0.73         6

    accuracy                           0.30        20
   macro avg       0.24      0.18      0.20        20
weighted avg       0.41      0.30      0.34        20
```

Figura 25 - Resultado do código-fonte 8
Fonte: Elaborada pelo autor (2024)

Podemos criar outros modelos baseados em SVM. Ainda que o modelo baseline indicou uma certa linearidade nos dados, será que modelos mais complexos levariam a melhores resultados?

```
    # SVM com kernel RBF -> mais complexo
    svm_rbf = SVC(kernel='rbf')
    svm_rbf.fit(X_train_scaled, y_train)
    y_pred_svm_rbf = svm_rbf.predict(X_teste_scaled)
    print("Acurácia SVM (RBF):", accuracy_score(y_test,
y_pred_svm_rbf))
    print(classification_report(y_test, y_pred_svm_rbf))

    # SVM com kernel polinomial -> intermediário
    svm_poly = SVC(kernel='poly')
    svm_poly.fit(X_train_scaled, y_train)
    y_pred_svm_poly = svm_poly.predict(X_teste_scaled)
    print("Acurácia SVM (Polinomial):",
accuracy_score(y_test, y_pred_svm_poly))
    print(classification_report(y_test, y_pred_svm_poly))

    # SVM com kernel linear -> o mais simples
    svm_linear = SVC(kernel='linear')
    svm_linear.fit(X_train_scaled, y_train)
    y_pred_svm_linear = svm_linear.predict(X_teste_scaled)
    print("Acurácia SVM (Linear):", accuracy_score(y_test,
y_pred_svm_linear))
    print(classification_report(y_test, y_pred_svm_linear))
```

Código-fonte 9 - Modelos preditivos baseados em SVM
Fonte: Elaborado pelo autor (2024)

A figura “Resultado do código-fonte 9” mostra os resultados. Entre os modelos baseados em SVM, o kernel linear performou melhor, o que está em linha com o baseline.

A primeira técnica de aprendizado de máquina

```
Acurácia SVM (RBF): 0.35
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.00      0.00      0.00         1
           3       0.00      0.00      0.00         1
           4       0.43      0.60      0.50         5
           5       0.00      0.00      0.00         5
           6       1.00      0.50      0.67         6

    accuracy                           0.35        20
   macro avg       0.40      0.27      0.31        20
weighted avg       0.51      0.35      0.39        20

Acurácia SVM (Polinomial): 0.45
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         1
           4       0.43      0.60      0.50         5
           5       0.60      0.60      0.60         5
           6       1.00      0.33      0.50         6

    accuracy                           0.45        20
   macro avg       0.43      0.29      0.32        20
weighted avg       0.66      0.45      0.49        20

Acurácia SVM (Linear): 0.85
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         1
           4       0.83      1.00      0.91         5
           5       1.00      1.00      1.00         5
           6       1.00      1.00      1.00         6

    accuracy                           0.85        20
   macro avg       0.55      0.50      0.51        20
weighted avg       0.86      0.85      0.84        20
```

Figura 26 - Resultado do código-fonte 9
Fonte: Elaborada pelo autor (2024)

Agora, vamos criar modelos baseados em árvores! Será que estes modelos podem ser mais robustos que uma divisão puramente linear (Regressão Logística) e não tão complexo quanto um SVM?

```
    # Decision Tree
    dt = DecisionTreeClassifier()
    dt.fit(X_train_scaled, y_train)
    y_pred_dt = dt.predict(X_teste_scaled)
    print("Acurácia Decision Tree:", accuracy_score(y_test,
y_pred_dt))
    print(classification_report(y_test, y_pred_dt))

    # Random Forest
    rf = RandomForestClassifier(n_estimators=25)
    rf.fit(X_train_scaled, y_train)
    y_pred_rf = rf.predict(X_teste_scaled)
    print("Acurácia Random Forest:", accuracy_score(y_test,
y_pred_rf))
    print(classification_report(y_test, y_pred_rf))
```

Código-fonte 10 - Modelos preditivos baseados em SVM
Fonte: Elaborado pelo autor (2024)

A seguinte figura mostra os resultados. Entre os modelos baseados em árvore, a Árvore de Decisão performou melhor. Ou seja, a robustez da floresta foi desnecessária neste caso (mesmo com apenas 25 árvores na floresta).

```
Acurácia Decision Tree: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         1
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         5
           5       1.00      1.00      1.00         5
           6       1.00      1.00      1.00         6

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Acurácia Random Forest: 0.95
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       1.00      1.00      1.00         1
           2       0.00      0.00      0.00         0
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         5
           5       1.00      1.00      1.00         5
           6       1.00      1.00      1.00         6

    accuracy                           0.95        20
   macro avg       0.86      0.79      0.81        20
weighted avg       1.00      0.95      0.97        20
```

Figura 27 - Resultado do código-fonte 10
Fonte: Elaborada pelo autor (2024)

Portanto, de todos os modelos apresentados, o melhor foi a Árvore de Decisão. Inclusive ela foi perfeita! Com isso, podemos destacar alguns pontos:

- Um modelo com acurácia de 100% pode ser indício de contaminação de treino e teste. Mas neste caso não aconteceu.
- Nossa base é muito pequena (possui apenas ~100 exemplos), então o conjunto de teste não possui uma robustez e representação significativa para podermos afirmar que isso se replicará na realidade.
- Nem sempre conseguiremos modelos com performance tão alta. Cada caso é um caso.

A figura a seguir mostra o código e a distribuição dos *labels* do conjunto de teste. Observe que das 20 amostras que compõem esse conjunto, há apenas um exemplo do tipo “14-35-14” e 1 exemplo de “20-20”. Será que o modelo acertou porque realmente aprendeu a diferenciá-lo ou foi sorte de selecionar um exemplo “fácil” de ser predito?

```
pd.DataFrame(le.inverse_transform(y_test)) \
  .value_counts() \
  .reset_index() \
  .rename(columns={0: "Fertilizante", "count": "Quantidade"}) \
  .sort_values(by="Quantidade", ascending=False)
```

	Fertilizante	Quantidade
0	Urea	6
1	28-28	5
2	DAP	5
3	10-26-26	2
4	14-35-14	1
5	20-20	1

Figura 28 - Resultado do código-fonte 10
Fonte: Elaborada pelo autor (2024)

É justamente essas perguntas de análise da efetividade do modelo que fazem um cientista de dados ser um bom cientista de dados! Tais perguntas, alinhadas ao conhecimento de negócio, permitirão um trabalho mais profundo com os dados gerando maior impacto nas organizações.

CONCLUSÃO

Neste capítulo, exploramos os fundamentos do aprendizado supervisionado, com foco nos algoritmos de classificação, como KNN, Regressão Logística, Árvores de Decisão, Florestas Aleatórias e SVM. Além disso, discutimos como esses algoritmos funcionam, suas aplicações práticas, e a importância de escolher a métrica de avaliação correta para garantir que os modelos atendam aos objetivos do negócio, especialmente em cenários de classes desbalanceadas. Também enfatizamos a importância da preparação adequada dos dados, destacando como a limpeza, a exploração e a análise inicial influenciam diretamente o desempenho dos algoritmos de classificação.

Com este capítulo, você deve ter uma compreensão sólida de como selecionar e aplicar esses algoritmos a problemas reais, ajustando parâmetros e métricas para obter modelos robustos e precisos, capazes de generalizar bem para novos dados.

REFERÊNCIAS

ANÁLISE Supervisionado vs. Aprendizado Não-Supervisionado em machine learning. **Lean Saúde**, 2024. Disponível em: <https://www.leansaude.com.br/aprendizado-supervisionado-vs-aprendizado-nao-supervisionado-em-machine-learning/>. Acesso em: 29 ago. 2024.

ARTIGO. Resultados de modelos de machine learning. **Microsoft**, 07 de março de 2023. Disponível em: <https://learn.microsoft.com/pt-br/dynamics365/finance/finance-insights/confusion-matrix>. Acesso em: 27 ago. 2024.

CARACTERÍSTICA de Operação do Receptor. **Wikipedia**, 23 de fevereiro de 2024. Disponível em: <https://pt.wikipedia.org/wiki/Caracter%C3%ADstica_de_Opera%C3%A7%C3%A3o_do_Receptor>. Acesso em: 27 ago. 2024.

CIELEN, Davy; MEYSMAN, Arno. **Introducing data science:** big data, machine learning, and more, using Python tools. Simon and Schuster, 2016.

COMO funciona o algoritmo Árvore de Decisão. **Didática Tech**, 2024. Disponível em: <https://didatica.tech/como-funciona-o-algoritmo-arvore-de-decisao/>. Acesso em: 27 ago. 2024.

IZBICKI, Rafael; DOS SANTOS, Tiago Mendonça. **Aprendizado de máquina: uma abordagem estatística**. Rafael Izbicki, 2020.

KARPINSKI, Leonardo. Machine Learning pt 3: Descomplicando a Matriz de Confusão e Calculando Acurácia, Precisão e Recall. **LinkedIn**, 01 de outubro de 2018. Disponível em: <https://www.linkedin.com/pulse/machine-learning-pt-3-descomplicando-matriz-de-e-recall-karpinski/>. Acesso em: 27 ago. 2024.

KHUNTIA, Omkar Subhasish. Support Vector Machine A-Z. **Medium**, 04 de março de 2020. Disponível em: <https://medium.com/@omkarsubhasishkhuntia/support-vector-machine-a-z-c3421ecd2ad7>. Acesso em: 27 ago. 2024.

LANA, Julio Cesar. Modelo Floresta Aleatória composto por múltiplas árvores de decisão. A predição final resulta da predição mais frequente em cada árvore. **Researchgate**, 2020. Disponível em: <https://www.researchgate.net/figure/Figura-39-Modelo-Floresta-Aleatoria-composto-por-multiplas-arvores-de-decisao-A_fig7_361609244>. Acesso em: 27 ago. 2024.

LOPES, André. Modelos de Classificação: Regressão Logística. **Brains**, 07 de janeiro de 2023. Disponível em: <https://brains.dev/2023/modelos-de-classificacao-regressao-logistica/>. Acesso em: 27 ago. 2024.

MANDOT, Pushkar. What is the Significance of C value in Support Vector Machine. **Medium**, 09 de setembro de 2017. Disponível em: <https://medium.com/@pushkarmandot/what-is-the-significance-of-c-value-in-support-vector-machine-28224e852c5a>. Acesso em: 27 ago. 2024.

O que é e como funciona o algoritmo KNN. **Didática Tech**, 2024. Disponível em: <https://didatica.tech/o-que-e-e-como-funciona-o-algoritmo-knn/>. Acesso em: 27 ago. 2024.

PROVOST, Foster. **Data Science para Negócios by Tom Fawcett Foster Provost** [Fawcett, Tom](z-lib. org).

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligencia Artificial**: un enfoque moderno. 2004.

SILVA, Fernando. Reamostragem em modelos preditivos: separação treino e teste. **Análise Macro**, 31 de julho de 2023. Disponível em: <https://analisemacro.com.br/econometria-e-machine-learning/reamostragem-em-modelos-preditivos-separacao-treino-e-teste/>. Acesso em: 27 ago. 2024.

TIPOS de aprendizado de máquina e algumas aplicações. **Decom**, 2021. Disponível em: <https://www2.decom.ufop.br/terralab/tipos-de-aprendizado-de-maquina-e-algumas-aplicacoes/>. Acesso em: 27 ago. 2024.

VINICIUSSEGATTO. SVM, ou Support Vector Machine. **Medium**, 11 de outubro de 2023. Disponível em: <https://medium.com/liga-mackenzie-de-ia-ci%C3%AAncia-de-dados/svm-ou-support-vector-machine-7efcabdcc7be>. Acesso em: 27 ago. 2024.

VUKOV, Bojana (BojanaVukov). “Multiclass classification and softmax function”. 23 de agosto de 2022, 03:00. **X**. Disponível em: <https://x.com/BojanaVukov/status/1561956403266330624> . Acesso em: 27 ago. 2024.

OUTLIERS or Key Profiles? Understanding Distance Measures for Authorship Attribution. **Digital Humanities**, 16 de julho de 2016. Disponível em: <https://dh2016.adho.org/abstracts/253>. Acesso em: 27 ago. 2024.

GLOSSÁRIO

Termo	Explicação
AUC	Area Under the Curve.
FN	False Negative (falso negativo).
FP	False Positive (falso positivo).
IA	Inteligência Artificial.
KNN	K-Nearest Neighbors (K Vizinhos Mais Próximos).
ML	Machine Learning (aprendizado de máquina).
MSE	Mean Squared Error (erro quadrático médio).
RBF	Radial Basis Function (função de base radial).
ROC	Receiver Operating Characteristic (curva característica de operação do receptor).
SVM	Support Vector Machine (máquina de vetores de suporte).
TN	True Negative (verdadeiro negativo).
TP	True Positive (verdadeiro positivo).