

Introdução a Machine Learning

Módulo 3 - Métricas e Análises

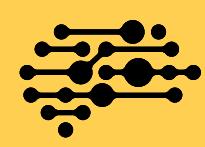
Foto por Robina Weermeijer, disponível na Unsplash. Adaptada pelo autor.



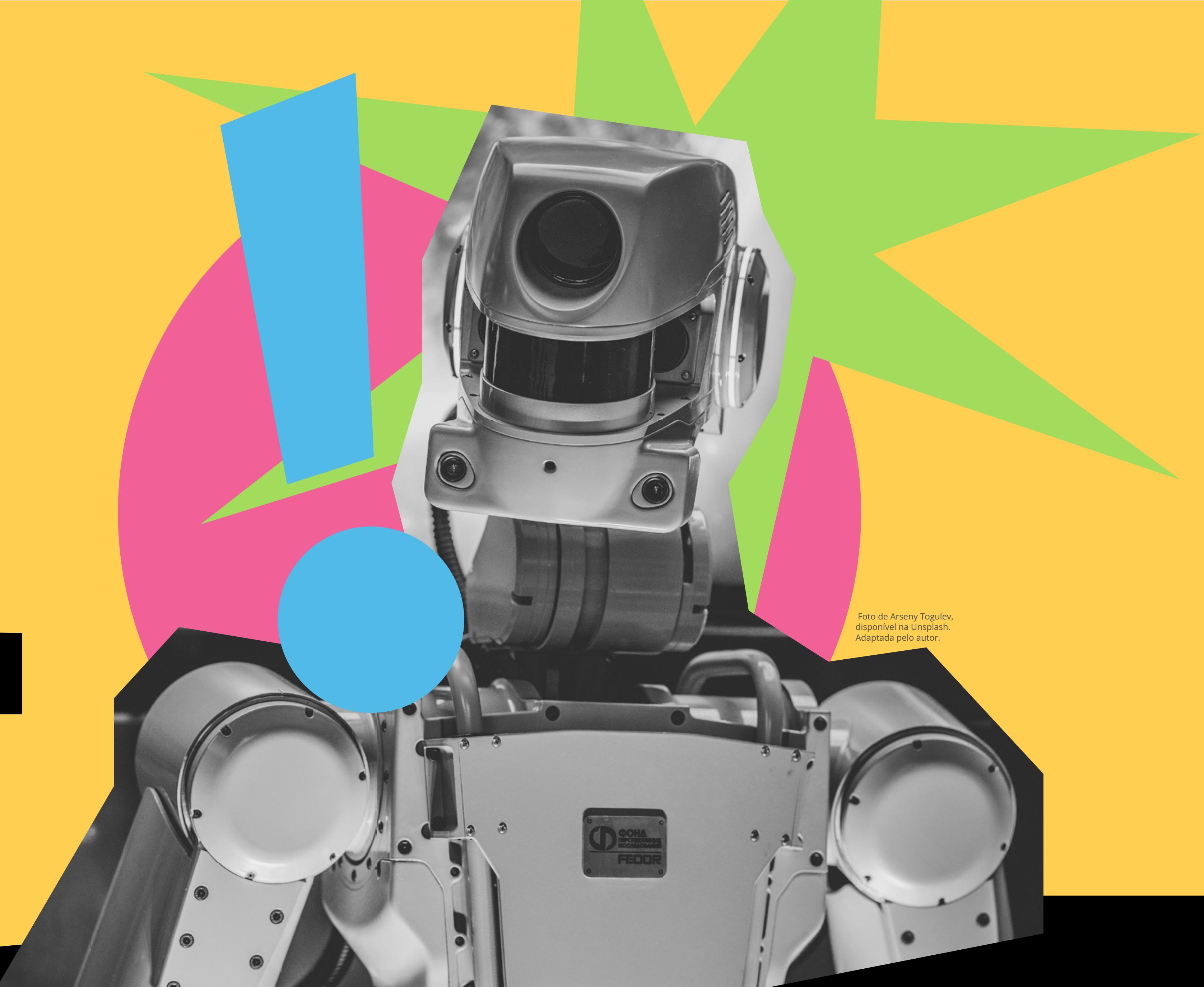
Foto de Georges Malher, disponível na Unsplash. Adaptada pelo autor.

Sumário

Aula 1 - Matriz de Confusão	1
Aula 2 - Métricas	4
2.1 Classificação.....	5
2.2 Regressão	6
Aula 3 - Métodos de Aprimoramento.....	8
3.1 ROC e AUC.....	9
3.2 Validação Cruzada.....	10
3.3 Tunagem de Hiperparâmetros.....	12
Explore mais!	14
Referências Bibliográficas.....	15



Aula 1 - Matriz de Confusão



Uma Matriz de Confusão é uma ferramenta que permite avaliar o desempenho de um modelo de classificação. Ela mostra o número de acertos e erros do modelo para cada classe prevista, comparando com os valores reais. Por exemplo, se você tem um modelo que classifica imagens de gatos e cachorros, a matriz de confusão vai mostrar quantas imagens de gatos foram corretamente classificadas como gatos, quantas foram incorretamente classificadas como cachorros, e vice-versa. A matriz de confusão também pode ser usada para calcular métricas importantes, como precisão, *recall*, acurácia e *f1-score*, utilizadas para medir a performance do modelo em diferentes cenários.

A Matriz de Confusão tem duas dimensões: a **classe real (ground truth)** e a **classe prevista (predicted)**. Cada célula da matriz representa o número de casos que pertencem a uma determinada combinação de classe real e classe prevista. Por exemplo, se a classe real é gato e a classe prevista é cachorro, isso significa que o algoritmo errou ao classificar um gato como cachorro. As células da diagonal principal da matriz representam os casos em que o algoritmo acertou ao classificar a foto como gato ou cachorro.

A partir da Matriz de Confusão, podemos calcular várias métricas que nos ajudam a entender melhor o desempenho do algoritmo. Essas métricas dependem dos conceitos de verdadeiro positivo (TP), verdadeiro negativo (TN), falso positivo (FP) e falso negativo (FN), definidos da seguinte forma (figura 1):

		<i>Ground truth</i>		
		+	-	
<i>Predicted</i>	+	<i>True Positive (TP)</i>	<i>False Positive (FP)</i>	$Precision = TP / (TP + FP)$
	-	<i>False Negative (FN)</i>	<i>True Negative (TN)</i>	$Recall = TP / (TP + FN)$

$Accuracy = TP + TN / (TP + FP + FN)$

Figura 1 – Matriz de Confusão.

Podemos concluir da matriz anterior (figura 1) as seguintes informações:

- **Verdadeiro positivo (TP):** o algoritmo previu corretamente a classe positiva (gato);
- **Verdadeiro negativo (TN):** o algoritmo previu corretamente a classe negativa (cachorro);
- **Falso positivo (FP):** o algoritmo previu incorretamente a classe positiva (gato), quando na verdade era negativa (cachorro);
- **Falso negativo (FN):** o algoritmo previu incorretamente a classe negativa (cachorro), quando na verdade era positiva (gato).

Um exemplo de matriz de confusão para o problema de classificação de fotos de gatos e cachorros é:

Classe Real/Classe Prevista	Gato	Cachorro
Gato	TP	FP
Cachorro	FN	TN

Tabela 1 – Matriz para problema de classificação. Fonte: feita pelo autor.



Aula 2 - Métricas

2.1 Classificação

Para ilustrar melhor o conceito de Matriz de Confusão, vamos supor que temos um conjunto de 13 fotos, sendo 8 de gatos e 5 de cachorros. O algoritmo de classificação acertou 8 fotos e errou 5, sendo que 3 gatos foram classificados como cachorros e 2 cachorros foram classificados como gatos. Nesse caso, a matriz de confusão seria (tabela 2):

Classe Real/Classe Prevista	Gato	Cachorro
Gato	5	3
Cachorro	2	3

Tabela 2 – Matriz de confusão de algoritmos de classificação. Fonte: feita pelo autor.

As métricas que podemos calcular a partir dessa matriz são:

- **Acurácia:** a proporção de casos classificados corretamente pelo algoritmo. É calculada pela fórmula: $(TP + TN) / (TP + TN + FP + FN)$. No nosso exemplo, a acurácia seria: $(5 + 3) / (5 + 3 + 3 + 2) = 0.615$;

- **Precisão:** a proporção de casos positivos previstos pelo algoritmo que são realmente positivos. É calculada pela fórmula: $TP / (TP + FP)$. No exemplo, a precisão seria: $5 / (5 + 3) = 0.625$;
- **Recall (ou sensibilidade):** a proporção de casos positivos reais que são previstos corretamente pelo algoritmo. É calculada pela fórmula: $TP / (TP + FN)$. No nosso exemplo, o recall seria: $5 / (5 + 2) = 0.71$;
- **F1 score:** muito útil quando se tem um conjunto de dados desequilibrado, ou seja, quando há uma grande disparidade entre o número de exemplos positivos e negativos. Ele fornece uma medida balanceada de desempenho que leva em consideração tanto a precisão quanto o recall. É calculado pela fórmula: $F1 \text{ score} = 2 \times (\text{Precisão} \times \text{Recall}) / (\text{Precisão} + \text{Recall})$. No exemplo, o F1 score seria $2 \times (0.625 \times 0.71) / (0.625 + 0.71) = 0.665$.

A Matriz de Confusão é uma ferramenta importante para avaliar a qualidade do modelo de classificação, identificar possíveis erros e tomar medidas para melhorar o desempenho dos modelos avaliados.

2.2 Regressão

Relembrando, as tarefas de Regressão em Aprendizado de Máquina: são aquelas em que o objetivo é prever um valor contínuo ou quantitativo, em vez de uma classe ou categoria.

Ao contrário das tarefas de classificação, em que o objetivo é atribuir um rótulo a uma instância com base em suas características, as tarefas de Regressão envolvem a previsão de um valor numérico, como a previsão do preço de uma casa com base em suas características (tamanho, número de quartos, localização, etc.) ou a previsão da temperatura com base em dados meteorológicos (hora do dia, umidade, velocidade do vento, etc.).

As métricas para modelos de Aprendizado de Máquina de Regressão são medidas que avaliam o desempenho e a precisão de um modelo que tenta prever uma variável contínua.

Algumas das métricas mais comuns são:

- **Erro Médio Absoluto (MAE)**: é a média das diferenças absolutas entre as previsões e os valores reais. Ele mede a magnitude média dos erros em um conjunto de previsões e é calculado pela fórmula:

$$\text{MAE} = (1/n) * \sum_{i=1}^n |y_i - \hat{y}_i|$$

Onde **n** é o número de observações, **y_i** é o valor real da i-ésima observação e **ŷ_i** é o valor previsto da i-ésima observação.

- **Erro médio quadrático (MSE)**: é a média dos erros ao quadrado entre as previsões e os valores reais. Ele mede a média da magnitude do erro ao quadrado em um conjunto de previsões e é calculado pela fórmula:

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

- **Raiz do Erro Médio Quadrático (RMSE)**: é a raiz quadrada do MSE e é útil porque é medida na mesma escala que a variável de destino. É calculado pela fórmula:

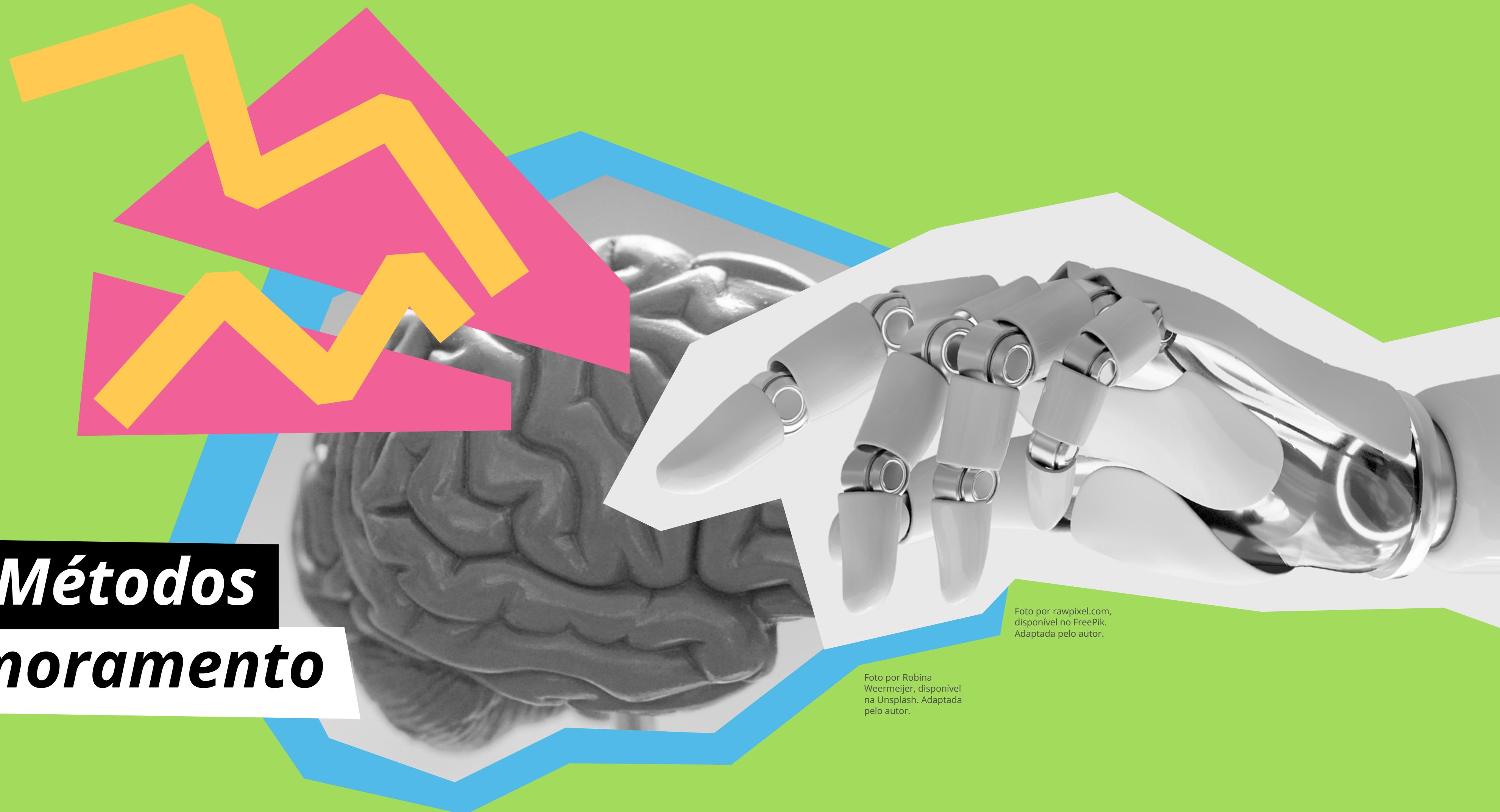
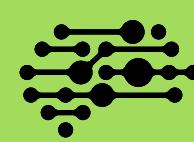
$$\text{RMSE} = \sqrt{(\text{MSE})}$$

- **Coeficiente de determinação (R^2):** é uma medida que indica a proporção da variância na variável de destino que é explicada pelas variáveis explicativas do modelo. Ele varia de 0 a 1, onde 0 indica que o modelo não explica a variância na variável de destino e 1 indica que o modelo explica toda a variância. É calculado pela fórmula:

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

Onde y_i é o valor real da i-ésima observação, \hat{y}_i é o valor previsto da i-ésima observação, \bar{y} é a média dos valores reais e n é o número de observações.

Essas métricas são úteis para avaliar a qualidade do modelo em prever valores contínuos. Cada métrica tem suas próprias vantagens e desvantagens, e a escolha da métrica depende da tarefa específica e dos requisitos do modelo.



Aula 3 - Métodos de Aprimoramento

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

Foto por rawpixel.com,
disponível no FreePik.
Adaptada pelo autor.

3.1 ROC e AUC

Os métodos *Receiver Operating Characteristic (ROC)* e *Area Under the Curve (AUC)* são métricas utilizadas para avaliar o desempenho de modelos de classificação binária.

A curva ROC é um gráfico que representa a taxa de verdadeiros positivos (TPR) em função da taxa de falsos positivos (FPR) para diferentes pontos de corte ou limiares de classificação. A TPR, também conhecida como sensibilidade ou *recall*, é a proporção de instâncias positivas corretamente classificadas em relação ao total de instâncias positivas. A FPR, por sua vez, é a proporção de instâncias negativas erroneamente classificadas como positivas em relação ao total de instâncias negativas.

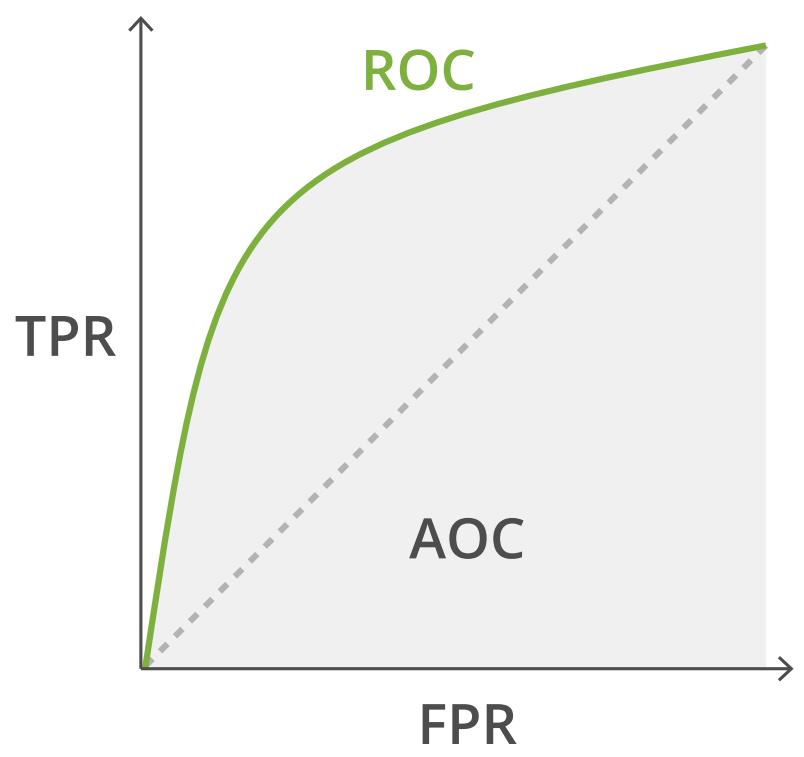


Figura 2 – Curva ROC².

A curva ROC é construída variando o limiar de classificação do modelo e calculando a TPR e FPR correspondentes em cada ponto. **Quanto mais próximo a curva ROC**

estiver do canto superior esquerdo do gráfico, melhor será o desempenho do modelo, indicando uma alta TPR e uma baixa FPR em diferentes limiares de classificação.

A AUC, por sua vez, é a área sob a curva ROC. Ela representa a capacidade de classificação do modelo em um intervalo completo de classificação. A AUC varia entre 0 e 1, onde um valor de 1 indica um modelo “perfeito” que é capaz de distinguir perfeitamente entre instâncias positivas e negativas, e um valor de 0,5 indica um modelo que realiza classificações aleatórias.

Em resumo, a curva ROC e a AUC fornecem uma medida abrangente do desempenho de um modelo de classificação binária, levando em consideração a taxa de verdadeiros positivos e falsos positivos em diferentes limites de classificação.

Quanto maior a AUC, melhor é o desempenho do modelo na classificação de instâncias positivas e negativas.

3.2 Validação Cruzada

Validação Cruzada, também conhecida como *Cross-Validation* em inglês, é uma técnica utilizada para **avaliar o desempenho de um modelo de Aprendizado de Máquina e estimar sua capacidade de generalização** para dados não vistos.

Quando treinamos um modelo é comum dividirmos nosso conjunto de dados em um conjunto de treinamento e um conjunto de teste. O conjunto de treinamento é usado para treinar o modelo e o conjunto de teste é usado para avaliar o desempenho do modelo em dados não vistos. No entanto, essa abordagem pode resultar em uma avaliação enviesada, pois estamos avaliando o modelo apenas em um conjunto específico de dados de teste.

A Validação Cruzada ajuda a mitigar esse viés ao dividir o conjunto de dados em várias partes, ou "dobras" (*folds*), e realizar o treinamento e a avaliação do modelo em diferentes combinações dessas dobras. A forma mais comum de Validação Cruzada é a Validação Cruzada *K-fold*.

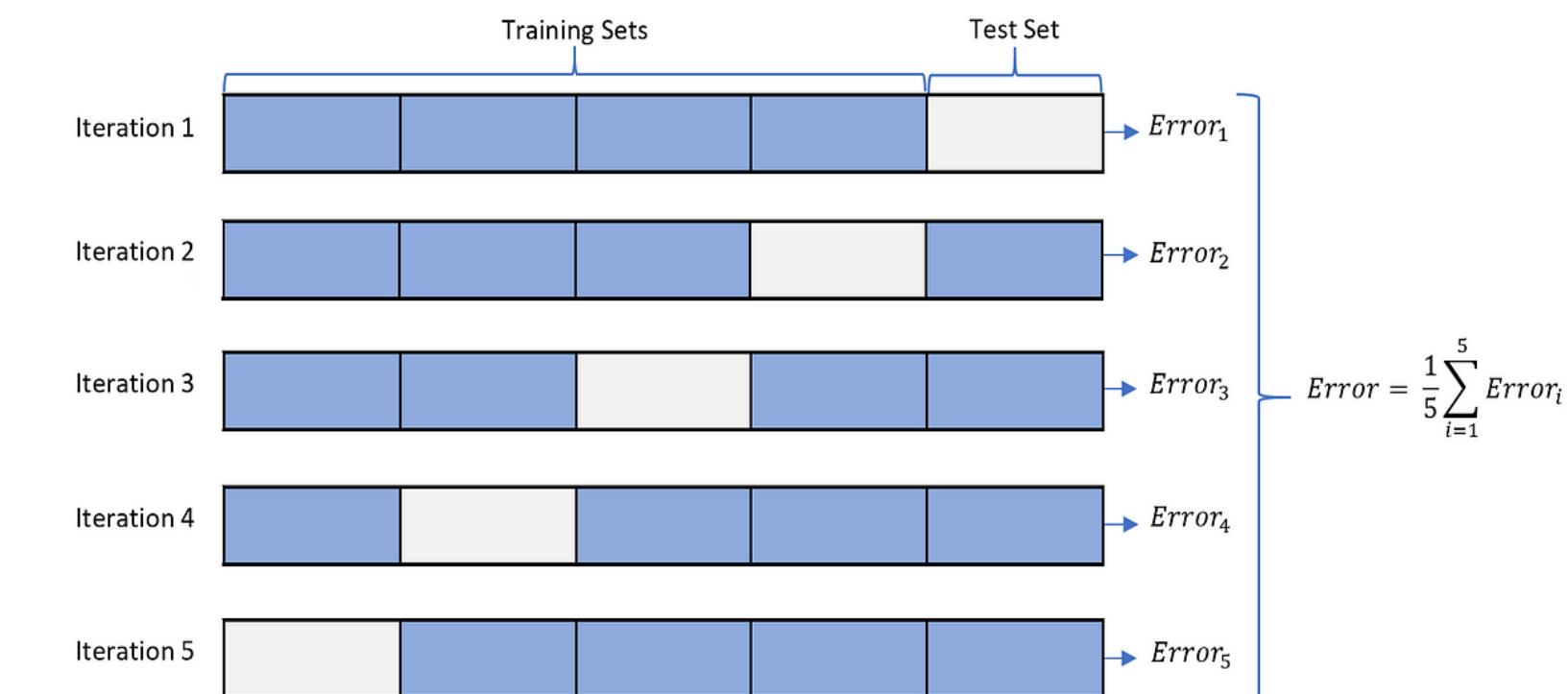


Figura 3 – Modelo de Validação Cruzada

No método de Validação Cruzada *K-fold*, o conjunto de dados é dividido em **K** dobras de tamanhos aproximadamente iguais. Em seguida, o modelo é treinado **K** vezes, sendo que em cada iteração, uma das dobras é usada como conjunto de teste e as outras dobras são usadas como conjunto de treinamento.

O desempenho do modelo é registrado em cada iteração usando uma métrica de avaliação relevante, como acurácia, precisão, *recall*, *F1-score* ou erro médio absoluto. Esses resultados podem ser usados posteriormente para calcular a média, desvio padrão ou outras estatísticas relevantes do desempenho do modelo.

Ao final do processo de Validação Cruzada *K-fold*, temos **K** medidas de desempenho que podem ser usadas para avaliar a capacidade de generalização do modelo. Podemos calcular a média dos resultados para obter uma estimativa geral do desempenho do modelo ou examinar a variabilidade dos resultados para avaliar a consistência do modelo em diferentes dobras.

Existem variações da Validação Cruzada além do *K-fold* tradicional, como a *Validação Cruzada Leave-One-Out*, em que cada exemplo de dados é usado como conjunto de teste uma vez, enquanto os demais exemplos são usados para treinamento. Essa abordagem é computacionalmente mais intensiva, mas pode ser útil quando se tem um conjunto de dados muito pequeno.

Em resumo, existem diferentes formas de fazer essa divisão, sendo as mais comuns o método *holdout*, o método *K-fold* e o método *Leave-One-Out*:

- **O Método *holdout*:** consiste em separar uma parte dos dados para teste e usar o restante para treinamento. Por exemplo, podemos usar 80% dos dados para treinar o modelo e 20% para testá-lo. Essa é uma forma simples e rápida de avaliar o modelo, mas pode ser sensível à forma como os dados são divididos. Se os dados de teste forem muito diferentes dos dados de treinamento, o resultado pode não ser confiável;
- **O Método *K-fold*:** como vimos, consiste em dividir os dados em **K** partes iguais e usar cada uma delas como teste uma vez, enquanto as outras ($K - 1$) partes são usadas para treinamento. O resultado final é a média das **K** validações obtidas. Esse método é mais robusto que o *holdout*, pois usa todos os dados para treinamento e teste, mas também é mais demorado;
- **O Método *Leave-One-Out*:** consiste em usar apenas um dado como teste e todos os outros como treinamento. Esse processo é repetido para cada dado do conjunto, ou seja, se temos **n** dados, fazemos **n** avaliações. Esse método é o mais rigoroso, pois usa quase todos os dados para treinamento e evita a dependência da divisão dos dados, mas também é o mais custoso computacionalmente.

3.3 Tunagem de Hiperparâmetros

A Tunagem de Hiperparâmetros, também conhecida como **Ajuste de Hiperparâmetros**, é o processo de encontrar a combinação ideal de hiperparâmetros para um algoritmo de Aprendizado de Máquina. Os hiperparâmetros são parâmetros que não são aprendidos pelo modelo durante o treinamento, mas que precisam ser definidos antes do treinamento e podem ter um impacto significativo no desempenho do modelo.

Os hiperparâmetros controlam diferentes aspectos do modelo, como a complexidade, a regularização e o critério de treinamento. Exemplos comuns de hiperparâmetros, que foram apresentados anteriormente, incluem:

- O número de estimadores em uma *Random Forest*;
- A taxa de aprendizado em um algoritmo de gradiente descendente;
- O valor de C em uma *Support Vector Machine*.

A Tunagem de Hiperparâmetros envolve a busca por uma combinação ótima de valores de hiperparâmetros que maximize o desempenho do modelo em uma determinada métrica de avaliação.

O objetivo é encontrar a configuração de hiperparâmetros que resulte no melhor desempenho preditivo ou na melhor capacidade de generalização do modelo.

Para isso, existem várias abordagens para a Tunagem de Hiperparâmetros, veja a seguir:

- **Grid Search:** nessa abordagem, uma grade de valores possíveis para cada hiperparâmetro é definida, e todas as combinações são avaliadas. O desempenho do modelo é calculado para cada combinação e o conjunto de hiperparâmetros com o melhor desempenho é selecionado;
- **Random Search:** nessa abordagem, um conjunto aleatório de combinações de hiperparâmetros é selecionado para avaliação. Essa abordagem pode ser mais eficiente em termos de tempo de computação quando o espaço de busca de hiperparâmetros é grande;
- **Otimização Bayesiana (Bayesian Optimization):** é uma abordagem mais avançada que utiliza a teoria de probabilidade para modelar a relação entre hiperparâmetros e desempenho do modelo. Através de uma combinação de inferência estatística e métodos de otimização, a busca pelos melhores hiperparâmetros é mais eficiente;
- **Técnicas automatizadas de Tunagem de Hiperparâmetros:** existem também técnicas automatizadas que buscam encontrar a melhor combinação de hiperparâmetros de forma mais eficiente, como a otimização baseada em algoritmos genéticos.

A Tunagem de Hiperparâmetros é uma etapa crítica no desenvolvimento de modelos de Aprendizado de Máquina, pois a escolha adequada dos hiperparâmetros pode impactar significativamente o desempenho e a capacidade de generalização do modelo. Por isso, é importante realizar a Tunagem de Hiperparâmetros de forma cuidadosa e sistemática para obter os melhores resultados.

Explore mais!

A ferramenta *Business Intelligence and Analytics Software* facilita a criação de dashboards, confira [aqui](#).

O artigo "Dashboard Design Patterns" demonstra boas práticas na hora de apresentar os dados, leia [aqui](#).

A biblioteca *Manim Community* é uma biblioteca python para criação de vídeos animados, acesse [aqui](#).



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

Referências Bibliográficas

Machine Learning Fundamentals: The Confusion Matrix. Disponível em: <https://www.youtube.com/watch?v=Kdsp6soqA7o&pp=ygUcM2JsdWUxYnJvd24gY29uZnVzaW9uIG1hdHJpeA%3D%3D>. Acesso em: maio de 2023.

Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.22.1 documentation. Disponível em: <https://scikit-learn.org/stable/modules/model_evaluation.html>. Acesso em: maio de 2023.

Model selection and evaluation. Disponível em: https://scikit-learn.org/stable/model_selection.html#model-selection. Acesso em: maio de 2023.