

Resumo de ML

1. Aprendizado de Máquina

Aprender é o processo pelo qual um sistema melhora seu desempenho por meio da experiência. O Aprendizado de Máquina (Machine Learning) surge como uma solução para problemas onde é difícil antecipar todas as situações possíveis ou programar uma solução passo a passo. As abordagens para o aprendizado incluem:

- **Aprendizado como busca:** Consiste em enumerar um espaço de conceitos e eliminar aqueles que não condizem com os dados observados.
- **Aprendizado indutivo:** Foca em extrair informações gerais a partir da observação de um conjunto de casos particulares. Um exemplo histórico é como as observações de Tycho Brahe permitiram a Johannes Kepler formular suas leis da mecânica celeste.

O Papel dos Dados

Os dados são a base para o treinamento de programas de aprendizado de máquina. O desenvolvimento de um sistema de ML segue etapas como a definição do problema, coleta e preparação dos dados, treinamento e avaliação do modelo.

Contudo, a coleta de dados apresenta desafios:

- **Desbalanceamento:** Fontes como a Wikipedia possuem uma distribuição de tópicos desigual, com maior concentração em áreas como Esportes, Música e Política.
- **Falta de Diversidade:** Os dados frequentemente sub-representam diversas culturas, geografias e grupos étnicos, privilegiando pontos de vista predominantes. O acesso à internet, por exemplo, é maior entre jovens e em países desenvolvidos.
- **Dados no Brasil:** Embora o acesso à internet atinja 80% dos domicílios, há uma disparidade social: 100% da classe A está conectada, contra 60% das classes D e E.

Relação com Outras Áreas

O documento posiciona o Aprendizado de Máquina dentro de um contexto mais amplo:

- **Inteligência Artificial (IA):** É a área mais ampla, focada na construção de sistemas inteligentes que se comportam como humanos.
- **Aprendizado de Máquina (ML):** É uma subárea da IA que desenvolve algoritmos capazes de aprender.
- **Aprendizado Profundo (Deep Learning):** Uma subárea do ML que utiliza modelos com múltiplas camadas de processamento para aprender representações de dados em vários níveis de

abstração.

- **Ciência de Dados e Mineração de Dados:** A Ciência de Dados estuda a extração de conhecimento a partir de dados, utilizando técnicas de ML , enquanto a Mineração de Dados foca na aplicação de algoritmos para extrair padrões de conjuntos de dados.

2. Perceptron

Ele é o **modelo mais simples de rede neural** e foi o ponto de partida para o desenvolvimento do **aprendizado supervisionado**.

- Objetivo: **Classificar** dados em duas classes (ex: 0 ou 1, -1 ou +1).
- Tipo: **Aprendizado supervisionado (classificação binária)**.
- Baseia-se em **combinar entradas ponderadas por pesos** e aplicar uma **função de ativação**.

O Perceptron representa **um neurônio artificial** com

Componente	Descrição
Entradas (x_1, x_2, \dots, x_n)	Atributos ou variáveis de entrada.
Pesos (w_1, w_2, \dots, w_n)	Parâmetros ajustáveis (importância de cada entrada).
Viés (bias, b)	Constante que desloca o limiar da decisão.
Soma Ponderada (z)	$z = \sum_{i=1}^n (w_i \cdot x_i) + b$
Função de Ativação (f)	Converte (z) em saída binária (0 ou 1).

Saída:

$$y = \begin{cases} 1, & \text{se } (\sum_{i=1}^n w_i x_i + b) > 0 \\ 0, & \text{caso contrário} \end{cases}$$

Pseudocódigo:

```
Entrada: dados de treino  $\{(x_1, y_1), \dots, (x_n, y_n)\}$ 
inicializar pesos  $w_i = 0$  e bias  $b = 0$ 
Definir taxa de aprendizado  $\alpha$  (ex: 0.1)

para época em 1..N:
    para cada amostra  $(x, y_{\text{esperado}})$ :
         $y_{\text{pred}} = \text{ativacao}(\sum(w_i * x_i) + b)$ 
        erro =  $y_{\text{esperado}} - y_{\text{pred}}$ 
        para cada peso  $w_i$ :
```

$$w_i = w_i + \alpha * erro * x_i$$
$$b = b + \alpha * erro$$

Função ativação: retorna 1 se entrada > 0, senão 0.

- $\alpha \rightarrow$ taxa de aprendizado (ex: 0.1)
 - Atualiza pesos apenas quando há erro.
- O Perceptron busca **coeficientes (pesos)** que satisfaçam:

$$y_i(w \cdot x_i + b) > 0$$

para todas as amostras corretamente classificadas.

Se o conjunto for **linearmente separável**, o Perceptron **converge** (ou seja, encontra pesos corretos).

Se **não for separável**, ele **não converge** (fica oscilando).

Representação Geométrica

A fronteira de decisão é dada por:

$$w_1x_1 + w_2x_2 + \dots + w_nx_n + b = 0$$

- Pontos acima \rightarrow classe 1
- Pontos abaixo \rightarrow classe 0

Limitações

Limitação	Explicação
Somente separável linearmente	Não funciona para padrões não lineares (ex: XOR).
Convergência garantida apenas se linearmente separável	Caso contrário, entra em loop.
Função de ativação simples	Apenas degrau; não lida com probabilidades.

Versão Vetorial (Forma Compacta)

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - \hat{y})\mathbf{x}$$
$$b \leftarrow b + \alpha(y - \hat{y})$$

- \mathbf{w} : vetor de pesos
- \mathbf{x} : vetor de entrada
- y : rótulo real
- \hat{y} : previsão

Dica para Prova

Lembre-se:

- O perceptron **não “entende” padrões curvos** — só retas/planos.
- **Erro = esperado - previsto**
- **Atualiza pesos somente se erra**
- **Bias desloca a fronteira de decisão** (não precisa passar pela origem).

3. Tipos de Aprendizado

Supervisionado:

- Dados **rotulados** ($x \rightarrow y$ conhecidos).
- O modelo **aprende mapeamento $f(\mathbf{x}) \rightarrow y$** .
- Exemplo:
 - a. Regressão linear (previsão de valores contínuos)
 - b. Classificação (atribuição de rótulos, ex: spam/não spam)

Como funciona

1. Fornecemos **dados de treino** com pares (entrada, saída):

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

2. O modelo aprende **a relação entre \mathbf{x} e y** ajustando seus parâmetros.
3. Depois é testado em **dados novos** (sem rótulo) para verificar se generalizou bem.

Tipos principais

- **Regressão**: Usada para prever valores contínuos.
- **Classificação**: Utiliza um algoritmo para atribuir dados a categorias específicas.

 Cuidados

- Dividir dataset em **treino e teste**.
 - Evitar **overfitting** (memorizar dados).
 - Avaliar com métricas adequadas (MSE, acurácia, precisão, recall...).
-

◆ Não Supervisionado:

Neste tipo, **os dados não têm rótulos** — o modelo tenta **descobrir padrões, grupos ou estruturas ocultas** nos dados. Não há “resposta correta” para comparar.

- Dados **não rotulados**.
- O algoritmo descobre **padrões ocultos** (grupos, associações).

Objetivo

- Encontrar **relações internas** entre os dados.
- Reduzir dimensionalidade, agrupar exemplos similares ou identificar outliers.
- Exemplo: *Clustering (agrupamento K-Means)*.

⚠ Limitações

- Difícil avaliar se o agrupamento está “correto”.
 - Requer interpretação humana posterior.
-

◆ Por Reforço:

- O agente aprende por **recompensa/punição** ao interagir com o ambiente.
 - Exemplo: jogos, robótica, controle autônomo.
-

4. Modelos Paramétricos vs. Não Paramétricos

- **Modelos Paramétricos:** Resumem os dados com um conjunto de parâmetros de tamanho fixo, independentemente do número de exemplos. Exemplos incluem Regressão Linear e Perceptron.
 - a. **Vantagens:** São mais fáceis de interpretar, mais rápidos e não exigem muitos dados.
 - b. **Desvantagens:** São muito dependentes da função escolhida e mais adequados para problemas simples.
- **Modelos Não Paramétricos:** Não podem ser caracterizados por um conjunto limitado de parâmetros. Exemplos incluem árvores de decisão, redes neurais e SVM.

- a. **Vantagens:** São mais flexíveis, não fazem suposições sobre a função subjacente e possuem alta performance.
 - b. **Desvantagens:** Requerem mais dados, são mais lentos para treinar e têm maior risco de *overfitting*.
-

5. Tipos de Dados

Dados Estruturados

São dados organizados, com informações representadas por atributos (features) e seus valores, geralmente armazenados em bancos de dados ou planilhas. Os tipos de atributos podem ser:

- **Catégoricos/Nominais:** Valores que representam categorias (ex: cor do cabelo).
- **Booleano:** Apenas dois valores (verdadeiro/falso).
- **Ordinal:** Valores que representam uma escala (ex: nível de satisfação).
- **Numéricos:** Valores inteiros ou reais.

Dados Não Estruturados

São compostos por diferentes tipos de dados combinados, como textos e imagens.

- **Preparação de Dados:** Textos podem ser convertidos em representações numéricas (vetores) para capturar relações semânticas, como no exemplo `(rei - homem) + mulher = rainha`.
- **Dados de Imagem:** Uma imagem pode ser vista como uma matriz de pixels. Cada pixel é representado por um valor numérico, seja em escala de cinza (um inteiro de 0 a 255) ou RGB (três inteiros, um para cada cor).

6. Métricas de Avaliação para Modelos

Treinar um modelo (regressão, classificação etc.) não é o suficiente.

Precisamos **quantificar seu desempenho** — ou seja, **medir o erro ou acerto das previsões**.

- **Objetivo:** saber se o modelo **generaliza bem** para dados novos.
- **Problema comum:** *Overfitting* (modelo “decorou” o treino) ou *Underfitting* (modelo muito simples)

Matriz de confusão

		----- REAL -----	
		POSITIVO	NEGATIVO
--- PREVISTO ---	POSITIVO	VP	FP
	NEGATIVO	FN	VN

- **Verdadeiro Positivo (VP):** O modelo previu "positivo" e o valor real era "positivo".
- **Falso Positivo (FP):** O modelo previu "positivo", mas o valor real era "negativo".
- **Falso Negativo (FN):** O modelo previu "negativo", mas o valor real era "positivo".
- **Verdadeiro Negativo (VN):** O modelo previu "negativo" e o valor real era "negativo".

Acurácia

Percentual de acertos totais do modelo.

Boa quando as classes estão **equilibradas**

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN}$$

Revocação (Recall)

De todos os casos que eram realmente "positivos", quantos o modelo conseguiu identificar?

Alta sensibilidade → poucos falsos negativos.

$$\text{Recall} = \frac{VP}{VP + FN}$$

Precisão

Das vezes que o modelo previu "positivo", quantas ele acertou?

Alta precisão → poucos falsos positivos.

$$\text{Precision} = \frac{VP}{VP + FP}$$

F1 - Score

Média harmônica entre precisão e revocação, útil para balancear o impacto de falsos positivos e falsos negativos

Boa quando há **classes desbalanceadas**.

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}}$$

Métrica	Melhor em...	Exemplo
Acurácia	Bases balanceadas	Spam vs. não spam
Precisão	Custo alto de falsos positivos	Biometria bancária
Recall	Custo alto de falsos negativos	Diagnóstico de câncer
F1	Bases desbalanceadas e com erros simétricos	Análise de sentimentos em e-commerce

7. Tipos de Dados e Pré-Processamento

Tipos de atributos:

Tipo	Exemplo	Observação
Nominal	Cor (vermelho, azul)	Sem ordem.
Binário	Fumante (0/1)	Dois estados.
Ordinal	Tamanho (P, M, G)	Ordem, sem distância fixa.
Numérico	Idade, peso	Intervalo ou razão.

Etapas de pré-processamento:

- 1. Limpeza de dados**
 - a. Preencher valores ausentes (média, mediana, modelo preditivo).
 - b. Remover ruído (método de *binning*, regressão).
 - c. Detectar outliers.
- 2. Integração de dados**
 - a. Unir dados de múltiplas fontes, resolvendo duplicatas e conflitos de nomes.
- 3. Redução de dados**
 - a. Reduzir dimensionalidade (PCA, amostragem, seleção de atributos).
- 4. Transformação de dados**
 - a. Normalizar (ex: [0,1]).

b. Discretizar (intervalos, faixas de idade etc.).



Estatística básica:

Medidas de Posição:

medem a localização do meio ou centro de uma distribuição;

- **Média** (tendência central)

$$\frac{\sum x_i}{n}$$

sensível a valores extremos (outliers).

- **Mediana** (valor central)

$$\text{mediana } (x) = \begin{cases} x_{\frac{n+1}{2}} & \text{se } n \text{ for ímpar} \\ \frac{(x_{\frac{n}{2}} + x_{\frac{n+1}{2}})}{2} & \text{se } n \text{ for par} \end{cases}$$

para dados assimétricos, esta é a melhor medida.

- **Moda** (valor mais frequente)

Medidas de dispersão:

são utilizadas para que possamos saber qual o grau de variação dos nossos dados

- **Quartis:**

Dividem os dados ordenados em quatro partes iguais. O primeiro quartil (Q1) corresponde ao percentil 25, o segundo (Q2) é a mediana (percentil 50), e o terceiro (Q3) é o percentil 75

Intervalo Interquartilico (IQR): É a diferença entre o terceiro e o primeiro quartil ($IQR = Q3 - Q1$), representando a dispersão dos 50% centrais dos dados.

Como calcular os quartis:

$$i = (N - 1) \cdot q + 1$$
$$P_p = x_{[i]} + (i - [i]) \cdot (x_{[i]+1} - x_{[i]})$$

onde:

- i = índice
- N = Número de elementos

- q = percentil (0.25, 0.50, 0.75)
- P_p = Valor do percentil (25, 50, 75)
- $[i]$ = valor inteiro de i
- $x_{[i]}$ = Elemento na posição i

Como calcular os limites:

a. $X_i > LS$ (Limite Superior), para $LS = Q_3 + 1.5 \times (Q_3 - Q_1)$

b. $X_i < LI$ (Limite Inferior), para $LI = Q_1 - 1.5 \times (Q_3 - Q_1)$

- **Variância** σ^2 = média dos desvios² da média

$$\frac{1}{n} \sum (x_i - \bar{x})^2$$

- **Desvio padrão** = $\sqrt{\sigma^2}$

Mede o quão distantes os valores estão da média

- **Z-score** = (valor - média) / desvio padrão → mede quão longe está da média.

$$\frac{x - \bar{x}}{\sigma}$$

8. Limpeza de Dados

Limpeza de Dados

Etapa crítica para eliminar **ruído, erros e inconsistências** nos dados.

◆ Problemas comuns:

- **Valores ausentes (missing values)**
- **Outliers (valores atípicos)**
- **Erros de digitação / duplicação**

Método	Descrição
Remoção	Excluir linhas/colunas com muitos valores nulos.
Imputação por média/mediana/moda	Substituir valor ausente por uma medida estatística.
Modelos preditivos	Usar outro modelo para prever o valor ausente.

9. Transformação de Dados

Escalonamento de Dados Numéricos

Normalização

Escala os valores para um **intervalo padrão [0,1]**.

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Padronização

Converte dados para **média = 0 e desvio padrão = 1**.

$$x' = \frac{x - \bar{x}}{\sigma}$$

Classificação de Dados Categóricos

Técnica	Descrição	Exemplo
Label Encoding	Cada categoria recebe um número inteiro.	"Azul"=0, "Verde"=1, "Vermelho"=2
One-Hot Encoding	Cria colunas binárias (0/1) para cada categoria.	"Azul" → [1,0,0]; "Verde" → [0,1,0]
Variável dummy	Semelhante ao One-Hot, mas representa C categorias com C-1 variáveis, evitando redundância.	'verde'=[1,0] 'branco'=[0,1] 'azul'=[0,0]

10. Escolha de Modelos

Para um determinado conjunto de dados, múltiplos algoritmos de aprendizado de máquina podem ser aplicados, e não existe um método universalmente superior a todos os outros. O desempenho de cada algoritmo depende da base de dados utilizada. Portanto, é crucial utilizar critérios de avaliação robustos para selecionar o modelo mais adequado para o problema.

Para avaliar o desempenho de um modelo, especialmente em tarefas de regressão, a medida mais comum é o **Erro Quadrático Médio (MSE - Mean Squared Error)**.

Ele calcula a média dos quadrados das diferenças entre os valores reais (y_i) e os valores previstos pelo modelo $\hat{f}(x_i)$.

$$MSE = \frac{1}{n} \sum (y_i - \hat{f}(x_i))^2$$

para Teste e Treinamento.

onde $\hat{f}(x_i)$ é a previsão que \hat{f} da para i-ésima observação

- MSE será **pequeno** se as respostas previstas forem muito próximas das respostas verdadeiras,
- MSE será **grande** se para algumas das observações, as respostas previstas e verdadeiras diferirem substancialmente.

O **melhor modelo** é aquele que minimiza o erro de teste esperado.

Overfitting

Ocorre quando um modelo se ajusta de forma tão específica aos dados de treinamento que "memoriza" suas particularidades, incluindo ruídos. Isso resulta em um baixo erro de treinamento, mas um alto erro de teste, pois o modelo perde a capacidade de generalizar para novos dados

Trade-off Viés-Variância

O erro de teste esperado pode ser decomposto em três componentes:

1. **Viés (Bias):** O erro introduzido ao usar um modelo simples para aproximar um problema complexo. Modelos mais flexíveis geralmente têm um viés menor.
 2. **Variância (Variance):** A quantidade que o modelo mudaria se fosse treinado com um conjunto de dados diferente. Modelos mais flexíveis tendem a ter uma variância maior.
 3. **Erro Irredutível (ruído):** Um erro inerente aos dados, que não pode ser eliminado por nenhum modelo.
- descreve a relação inversa entre esses dois componentes: ao aumentar a flexibilidade de um modelo, o viés tende a diminuir, mas a variância tende a aumentar. O desafio é encontrar um equilíbrio que minimize o erro total de teste, encontrando um método com baixo viés e baixa variância

11. K Vizinhos Mais Próximos - KNN

Ideia principal: classificar uma nova amostra com base nos rótulos de suas K amostras mais próximas no conjunto de treinamento.

- **Características:**
 - a. Algoritmo supervisionado e não paramétrico.
 - b. Método “preguiçoso”: não há fase de treinamento; a previsão é feita por comparação direta.
- **Pré-processamento:**
 - a. Normalização dos atributos (escala [0,1]).
 - b. Tratamento de dados ausentes (remoção ou imputação).

Funcionamento: Para uma nova instância, o algoritmo calcula a distância (geralmente euclidiana) para todas as instâncias do conjunto de treinamento, identifica os K mais próximos e atribui o rótulo

da classe majoritária entre eles

- **Métricas de distância:** geralmente usa-se a **distância euclidiana**.

$$d(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2}$$

- **Escolha do K:**
 - a. K pequeno → modelo instável, Apenas objetos muito parecidos são considerados
 - b. K Grande → Vizinhos podem ser muito diferentes, Predição tendenciosa para a classe majoritária
 - c. Para classes pares, recomenda-se K ímpar.

12. Conjunto de Treinamento e Teste

Para evitar o *overfitting* e obter uma avaliação de desempenho mais confiável, o conjunto de dados original deve ser dividido. A abordagem mais simples é a **divisão em treino e teste**, usando, por exemplo, 80% dos dados para treinamento e 20% para teste, de forma aleatória.

Quando não há dados de teste suficientes, usamos **técnicas de reamostragem** para estimar o desempenho do modelo

Técnicas de Reamostragem:

K-Fold Cross-Validation

É uma das técnicas mais comuns de validação de modelos.

Como funciona:

1. O conjunto de dados é dividido em **K partes (folds)** aproximadamente do mesmo tamanho.
2. O modelo é treinado em **K-1 folds** e testado no **fold restante**.
3. Repete-se o processo **K vezes**, trocando o fold de teste a cada rodada.
4. O erro final é a **média dos K erros**.

Vantagens:

- Usa todos os dados para treinamento e teste em momentos diferentes.
- Reduz a variância da estimativa do erro.
- K típico: **5 ou 10**.

Desvantagem:

- Pode ser **computacionalmente custoso** para modelos grandes.

Stratified K-Fold Cross-Validation

É uma **variação do K-Fold** usada especialmente em **problemas de classificação**.

Diferença principal:

- Garante que cada fold preserve a **proporção de classes** do conjunto original.
Exemplo: se 30% das amostras são da classe “A” e 70% da classe “B”, cada fold manterá aproximadamente essa proporção.

Vantagem:

- Evita vieses causados por desequilíbrio entre as classes em cada divisão.
- Fornece estimativas de erro mais realistas em conjuntos desbalanceados.

Bootstrap

Gera múltiplos conjuntos de treinamento por amostragem com reposição do conjunto de dados original

Como funciona:

1. Gera-se um **novo conjunto de treinamento** de mesmo tamanho do original, **amostrando com reposição** (alguns exemplos aparecem várias vezes, outros podem não aparecer).
2. O modelo é treinado nesse novo conjunto.
3. As observações **não incluídas** (aproximadamente 36,8% dos dados) formam o **conjunto de teste** (“out-of-bag” samples).
4. O processo é repetido várias vezes (ex.: 1000 vezes) e os erros são **médios**.

Vantagens:

- Boa estimativa de variabilidade (intervalos de confiança).
- Útil quando o conjunto de dados é pequeno.

Desvantagens:

- Pode superestimar o desempenho se o modelo for muito sensível a amostras específicas.

13 . Regressão Linear

A regressão linear é um dos modelos mais fundamentais e amplamente utilizados no aprendizado de máquina supervisionado, especialmente em **problemas de predição contínua**.

Objetivo:

Modelar a relação entre uma **variável dependente (y)** e uma ou mais **variáveis independentes (x₁, x₂, ..., x_n)**, ajustando uma função linear aos dados observados.

Regressão Linear Simples:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

- (β_0): intercepto (valor de y quando x = 0) - coeficiente linear
- (β_1): inclinação (quanto y muda quando x varia 1 unidade) - coeficiente angular
- (ε): erro (diferença entre previsto e real)

A ideia é encontrar os valores de β_0 e β_1 que minimizam o **erro quadrático médio (MSE)** entre as previsões e os valores reais

Mínimos Quadrados

Para encontrar os melhores valores para β_0 e β_1 , o método mais comum é o dos **mínimos quadrados**. Ele busca minimizar a soma dos quadrados dos resíduos (erros), que é a diferença entre os valores reais (y_i) e os valores previstos pela reta ($y_{i_{previsto}}$). Essa soma é conhecida como **Soma dos Quadrados dos Resíduos (RSS - Residual Sum of Squares)**.

$$RSS = \sum (y_i - y_{previsto})^2 = \sum_{i=1}^n (y - (\hat{\beta}_1 x_i + \hat{\beta}_0))^2$$

Derivando RSS em função de $\hat{\beta}_1$ e $\hat{\beta}_0$ e igualando a zero, chegamos nas seguintes fórmulas fechadas:

$$\frac{\partial J(\hat{\beta}_0, \hat{\beta}_1)}{\partial \hat{\beta}_1} = \hat{\beta}_1 \sum x_i + \hat{\beta}_0 \cdot n = \sum y_i$$

$$\frac{\partial J(\hat{\beta}_0, \hat{\beta}_1)}{\partial \hat{\beta}_0} = \hat{\beta}_1 \sum x_i^2 + \hat{\beta}_0 \sum x_i = \sum x_i y_i$$

Regressão Linear Múltipla:

Quando há mais de uma variável independente:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \varepsilon$$

Forma matricial:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

Onde:

- \mathbf{X} é a matriz de entradas (com n amostras e p variáveis);
- $\boldsymbol{\beta}$ é o vetor de coeficientes;

- \mathbf{y} é o vetor de saídas.

Estimação dos parâmetros:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

Esse vetor $\hat{\beta}$ fornece os coeficientes que minimizam o erro de previsão.

Pseudocódigo (ajuste por mínimos quadrados):

```
dados = {(x1, y1), ..., (xn, yn)}
```

```
 $\beta_1 = \text{Cov}(x, y) / \text{Var}(x)$ 
```

```
 $\beta_0 = \text{Média}(y) - \beta_1 * \text{Média}(x)$ 
```

```
para cada novo x:
```

```
    prever  $y_{\text{pred}} = \beta_0 + \beta_1 * x$ 
```

Correlação

Nem todas as variáveis de entrada contribuem igualmente para a previsão da variável de saída. A **seleção de features** é o processo de escolher os atributos mais relevantes, o que pode diminuir o *overfitting*, melhorar o desempenho e reduzir o tempo de treinamento. Uma forma de fazer isso é através da análise de correlação**

Correlação de Pearson:

- Mede **relação linear** entre variáveis.

$$r = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y}$$

$$\rho_P(X, Y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} * \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

- Varia entre **-1 e 1**.
 - a. $r = 1 \rightarrow$ forte positiva
 - i. $r = -1 \rightarrow$ forte negativa

ii. $r \approx 0 \rightarrow$ sem correlação linear

- **Sensível a outliers.**

Interpretação:

- **Correlação alta entre features** \rightarrow multicolinearidade, o que pode distorcer os coeficientes β ;
- **Correlação alta entre uma feature e o alvo (y)** \rightarrow boa candidata para o modelo.





Correlação de Spearman:

- Usa **ordem (ranks)** dos dados (não valores).
- Mede **relação monotônica** (não necessariamente linear).
- **Mais robusta a outliers.**

Valor de ρ	Interpretação
+1	Correlação monotônica positiva perfeita – à medida que X aumenta, Y sempre aumenta.
0	Nenhuma correlação monotônica – não há tendência consistente.
-1	Correlação monotônica negativa perfeita – à medida que X aumenta, Y sempre diminui.

Use Spearman quando:

- Os **dados não seguem distribuição normal** (violam a suposição de normalidade);
- As **relações entre as variáveis são monotônicas**, mas **não lineares**;
Uma relação é **monotônica** se, à medida que uma variável cresce, a outra **sempre cresce** ou **sempre decresce**, mas não necessariamente em linha reta.
Exemplo:
 - a.  Monotônica: à medida que a temperatura aumenta, o consumo de sorvete aumenta (não precisa ser linear).
 - b.  Não monotônica: à medida que o tempo aumenta, a produtividade sobe até certo ponto e depois cai (relação em forma de “U”).
- Os dados são **ordinais** (valores representando posições, classificações ou ranqueamentos);
- Há **outliers**, que podem distorcer a correlação de Pearson.

Validação Cruzada (k-Fold Cross Validation):

1. Divide o dataset em k partes iguais (folds).
2. Treina com $k-1$ partes e testa com a parte restante.
3. Repete k vezes trocando o fold de teste.
4. Faz a **média dos resultados**.

Pseudocódigo:


```
dividir dados em k partes
para i de 1 até k:
    treino = dados - fold_i
    teste = fold_i
    modelo = treinar(treino)
    erro[i] = avaliar(modelo, teste)
mse_final = média(erro)
```

Vantagens:

- Usa todos os dados para treino e teste.
- Reduz viés e variação na avaliação.

7. Overfitting e Underfitting

Conceito	Descrição	Sintoma
Overfitting	Modelo se ajusta demais ao treino	Erro baixo no treino, alto no teste
Underfitting	Modelo muito simples	Erro alto em ambos

 **Objetivo:** encontrar o equilíbrio — bom desempenho em dados **nunca vistos**.

8. Perceptron (Algoritmo de Classificação)

Estrutura:

- Entradas: (x_1, x_2, \dots, x_n)
- Pesos: (w_1, w_2, \dots, w_n)
- Viés (bias): (b)

- Saída:

$$y = \begin{cases} 1, & \text{se } (\sum w_i x_i + b) > 0 \\ 0, & \text{caso contrário} \end{cases}$$

--- ### ✖ Pseudocódigo: `` inicializar pesos $w_i = 0$ e bias $b = 0$ para época em $1..N$: para cada amostra (