

Modelos Preditivos

Definição

Um modelo preditivo é um algoritmo que usa dados históricos para prever resultados desconhecidos.

Exemplos:

Prever vendas futuras.

Classificar e-mails como spam/não spam.

Previsão de preços de imóveis.

Previsão

- Vem do latim *praevidere* (“ver antes”).
- Usada no **sentido amplo** de antecipar algo que vai acontecer no futuro.

Sempre envolve **tempo** → olhar para frente.

Predição

- Mais usada em **estatística, machine learning e IA**.
- Refere-se a **estimar ou classificar um resultado desconhecido**, que pode estar no presente, no passado ou no futuro.

Foco é **estimar um valor ou rótulo desconhecido**, não necessariamente no futuro.

.....

Estrutura geral de um modelo preditivo

Dados de entrada (features).

Pré-processamento (limpeza, normalização, etc.).

Treinamento do modelo com dados históricos.

Predição nos dados de teste.

Avaliação com métricas (acurácia, precisão, recall...).

Predição em novos dados.

Biblioteca Scikit-learn

Uma das bibliotecas mais populares de Machine Learning em Python.

Focada em:

- ✓ Modelos de classificação, regressão e clustering.
 - ✓ Pré-processamento de dados.
 - ✓ Métricas de avaliação.
 - ✓ Pipelines para automação.
-

Exemplo Prático no Colab

Modelos Preditivos no Scikit-learn

1. Modelos de Regressão

Usados quando a variável alvo (**y**) é **numérica contínua**.

Exemplo: preço de casas, previsão de temperatura.

- **LinearRegression** → quando há relação linear simples.
- **Ridge / Lasso** → como a linear, mas com regularização (evita overfitting).
- **DecisionTreeRegressor** → quando a relação não é linear, mas em “regras” (if/else).
- **RandomForestRegressor** → conjunto de várias árvores, melhora precisão.
- **SVR (Support Vector Regressor)** → útil em dados não lineares e complexos.

Quando usar:

Se os dados têm forte correlação linear → LinearRegression.

Muitos atributos e risco de overfitting → Ridge ou Lasso.

Relações não lineares e dados tabulares → Árvores/Florestas.



2. Modelos de Classificação

Usados quando a variável alvo (**y**) é **categórica/discreta**.

Exemplo: prever se um e-mail é spam (sim/não), classificar flores em espécies.

- **LogisticRegression** → simples, interpretável, ótimo para baseline.
- **KNeighborsClassifier (KNN)** → classificação baseada na proximidade dos dados.
- **DecisionTreeClassifier** → fácil de interpretar, divide em regras.
- **RandomForestClassifier** → combina várias árvores, mais robusto.
- **SVC (Support Vector Classifier)** → bom para dados complexos, pode separar com fronteiras não lineares.
- **Naive Bayes** → muito usado em texto e NLP.

Quando usar:

Poucos atributos e relação simples → LogisticRegression.

Dados tabulares com muitas relações não lineares → RandomForest.

Dados de texto → Naive Bayes.

Poucos dados e fácil interpretação → Árvores.

3. Modelos de Clustering (Agrupamento)

Não existe variável alvo. Objetivo é **agrupar dados similares**.

Exemplo: segmentar clientes por perfil de compra.

- **KMeans** → divide os dados em k grupos.
- **DBSCAN** → encontra grupos de diferentes formas/tamanhos, útil com ruído.
- **AgglomerativeClustering** → cria hierarquias de agrupamento.

Quando usar:

Dados bem distribuídos → KMeans.

Dados com ruído e grupos de formas diferentes → DBSCAN.

Quer análise hierárquica → Agglomerative.

4. Modelos de Redução de Dimensionalidade

Usados quando temos **muitos atributos**, para simplificar sem perder tanta informação.

Exemplo: imagens (muitos pixels), textos (muitas palavras).

- **PCA (Principal Component Analysis)** → projeta os dados em menos dimensões.
- **t-SNE / UMAP** → usados para visualizar dados em 2D/3D.

Quando usar:

Pré-processar dados para alimentar outros modelos.

Visualizar padrões escondidos em datasets grandes.

5. Modelos Avançados (Ensemble / Boosting)

Melhoram previsões combinando vários modelos.

- **GradientBoosting** → bom para dados tabulares.
- **XGBoost / LightGBM / CatBoost** → variações mais rápidas e eficientes, usados em competições (Kaggle).

Quando usar:

Dados tabulares com muitas variáveis.

Quando precisa de alta performance (modelos de competição).

.....

Resumo

- **Problema de valores contínuos** → Regressão.
 - **Problema de categorias** → Classificação.
 - **Sem rótulos, quer descobrir grupos** → Clustering.
 - **Dados muito grandes/dimensionais** → Redução de dimensionalidade.
-

Testes e Otimizações de Modelos

Por que testar e otimizar?

Um modelo nunca é perfeito na primeira vez.

Objetivo: **equilibrar viés e variância**.

Testar → garantir que funciona em dados novos.

Otimizar → melhorar performance sem overfitting.

.....

Principais métricas (Regressão)

R^2 (Coeficiente de determinação):

Mede o quanto o modelo explica a variabilidade dos dados.

Valores:

- **1.0** → modelo perfeito (explica tudo).
- **0.0** → modelo não explica nada.
- Pode até ser negativo se o modelo for muito ruim.

Exemplo: Se $R^2 = 0,85$ → 85% da variação do preço das casas é explicada pelo modelo.

Quando o R^2 é considerado bom?

0,0 a 0,2 : Correlação muito fraca; o modelo não explica grande parte da variabilidade.

0,2 a 0,4 : Correlação fraca; o modelo explica alguma variabilidade, mas não é muito forte.

0,4 a 0,6 : Correlação moderada; o modelo explica uma quantidade razoável de variabilidade.

0,6 a 0,8 : Correlação forte; o modelo explica uma parte significativa da variabilidade.

0,8 a 1,0 : Correlação muito forte; o modelo explica a maior parte da variabilidade.

No entanto, um valor de R^2 "bom" pode variar bastante dependendo do contexto do estudo, da natureza dos dados e da área de pesquisa. Por exemplo:

Em áreas como ciências sociais, valores de R^2 em torno de 0,3 a 0,5 podem ser considerados aceitáveis.

Em áreas como física ou engenharia, valores próximos a 0,9 ou acima podem ser esperados.

Principais métricas (Regressão)

RMSE (Root Mean Squared Error – Raiz do Erro Quadrático Médio)

- Mede o erro médio **em unidades da variável alvo**.
- Quanto menor, melhor.
- Exemplo: $RMSE = 15.000$ → em média o modelo erra em R\$ 15 mil ao prever o preço de casas.

.....

Principais métricas (Classificação)

Acurácia

- % de acertos totais.
- Boa para classes balanceadas.
- Exemplo: acurácia 90% → 9 em cada 10 previsões corretas.



O que é a Matriz de Confusão?

É uma **tabela** que mostra o desempenho do modelo comparando as **previsões** com os **valores reais**.

		Predicted		
		Dog	Cat	
Actual	Dog	24	6	30
	Cat	2	18	20

previsto \ real	gato	rato	cachorro
gato	10	2	3
rato	5	14	1
cachorro	1	2	12

Principais métricas (Classificação)

Precisão (Precision)

É a quantidade de vezes que o modelo disse que era positivo e realmente era

Dividido pela quantidade de vezes que ele falou que era positivo

Recall (Sensibilidade)

É a quantidade de vezes que o modelo disse que era positivo e realmente era

Dividido pela quantidade de vezes que era realmente positivo

F1-Score

- Média harmônica entre Precisão e Recall.
- Bom quando há **desbalanceamento de classes**.

$$precisão = \frac{VP}{VP + FP}$$

$$sensibilidade = \frac{VP}{VP + FN}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Principais métricas (Classificação)

Precisão (Precision)

- Dos itens que o modelo disse “sim”, quantos eram realmente “sim”?

Recall (Sensibilidade)

- Dos “sim” reais, quantos o modelo conseguiu identificar?

F1-Score

- Média harmônica entre Precisão e Recall.
- Bom quando há **desbalanceamento de classes**.

$$precisão = \frac{VP}{VP + FP}$$

$$sensibilidade = \frac{VP}{VP + FN}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Opções para average

`average='macro'` → calcula a métrica para **cada classe separadamente** e faz a **média simples**.
(todas as classes têm o mesmo peso).

`average='weighted'` → calcula a métrica por classe e faz a média ponderada pelo número de instâncias de cada classe.
(importante em datasets desbalanceados).

`average='micro'` → usa os **contadores globais** de *true positives*, *false positives* e *false negatives*.
(mais útil quando você quer um resumo geral, não por classe).

`average=None` → retorna um **vetor com a métrica de cada classe** individualmente.

Viés e Overfitting

- **Viés** → modelo simplório que não aprende (Underfitting).
 - **Overfitting** → quando a variância domina e o modelo “decora” os dados de treino.
 - O **objetivo** é achar o meio-termo: **bom no treino e bom no teste**.
-

Viés (Bias)

É quando o modelo **erra por ser simples demais**.


Ele não consegue aprender os padrões reais dos dados.

- **Exemplo da vida real:**

Imagina que você tenta prever o preço de casas **apenas pelo tamanho (m^2)**.

Mas na prática, localização, número de quartos e idade da casa também importam.

Resultado: o modelo erra muito porque “simplificou demais” o problema.

 Isso gera **Underfitting** → o modelo vai mal no **treino** e no **teste**.

.....

Overfitting

Overfitting = **exagero no aprendizado**.

O modelo fica “inteligente demais” para os dados que já viu, mas “burro” para novos.

- **Exemplo com árvore de decisão:**

Uma árvore muito profunda cria regras do tipo:

- “Se a casa tem exatamente 100 m² e 2 banheiros → custa R\$ 250.000”
- “Se tem 101 m² e 2 banheiros → custa R\$ 252.000”

O modelo acaba decorando cada caso, sem aprender a lógica geral.

Tratamento de Dados em Machine Learning

Tratamento de Dados em Machine Learning

Antes de treinar um modelo preditivo, é **fundamental preparar os dados**. Um modelo mal treinado por causa de dados ruins terá **resultados imprecisos** ou enganosos.

O **tratamento de dados** envolve várias etapas:

- Limpeza de dados
 - Transformação e normalização
 - Seleção de variáveis
 - Análise de correlação
-

Limpeza de dados

Objetivo: **garantir que os dados estejam consistentes e completos.**

Principais estratégias:

- Remover ou preencher valores faltantes (NaN).
- Tratar dados duplicados.
- Corrigir inconsistências (ex.: datas no formato errado).
- Remover outliers extremos, se necessário.

💡 Exemplo: Se uma coluna de idade tem valor negativo ou 9999, devemos corrigir ou remover.

Transformação de variáveis

Alguns algoritmos exigem que os dados estejam **na mesma escala**.

- **Normalização (Min-Max Scaling)**: coloca todos os valores entre 0 e 1.
- **Padronização (Z-score)**: transforma os dados para média = 0 e desvio padrão = 1.

Por que é importante?

- Evita que variáveis com números grandes dominem o modelo.
 - Facilita o treino de modelos que usam **distâncias ou gradientes**, como regressão, redes neurais, KNN e SVM.
-

Normalização (Min-Max Scaling):

Transforma os valores para um intervalo fixo, geralmente entre **0 e 1**.

Normalizar é como ajustar todos os números para uma mesma régua de medida. Assim, o modelo não dá importância maior para uma variável só porque ela tem números grandes.

$$x^i = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Exemplo: Idade 50 $\rightarrow (50 - 0) / (100 - 0) = \mathbf{0.5}$

Padronização (Z-score)

Transforma os dados para terem **média = 0** e **desvio padrão = 1**

$$z = \frac{x - \mu}{\sigma}$$

Aqui, **x** é o valor observado,
μ (mu) é a média da população,
σ (sigma) é o desvio padrão da população.

Exemplo: Se a média de salários é 25.000 e o desvio padrão é 10.000, um salário de 30.000 vira:

$$(30000 - 25000) / 10000 = 0.5 \quad (30000 - 25000) / 10000 = 0.5 \quad (30000 - 25000) / 10000 = 0.5$$

Diferença entre Normalização e Padronização

Normalização (Min-Max Scaling)

Transforma os dados para um intervalo fixo, geralmente entre **0 e 1**.

Útil quando queremos preservar a **proporção relativa**.

Padronização (Z-score Scaling)

Transforma para média 0 e desvio padrão 1.

Útil quando os dados têm **distribuições diferentes** ou quando vamos aplicar **modelos que assumem normalidade** (ex:

Regressão Linear, PCA).

Seleção de variáveis

Nem todas as variáveis influenciam a variável alvo.

- **Remover colunas irrelevantes** ajuda a reduzir ruído e melhorar performance.
- **Análise de correlação:** ajuda a identificar variáveis importantes.

Exemplo:

- Variáveis com correlação muito fraca com a variável alvo podem ser descartadas.
- Porém, **atenção:** correlação fraca não significa irrelevância absoluta. Pode haver relações não lineares que a correlação simples não captura.

Estudos mostram que selecionar variáveis com base em correlação e experiência do domínio aumenta a acurácia dos modelos (O'Neil, 2016).
