Pós-Graduação em Inteligência Artificial Generativa e LLMs

PUC-Rio

Data Mining

Notas de Aula

Vítor Wilher

DM251 · Eletiva · 2025.1

Versão 1.0

Resumo. Mineração de dados: pré-processamento, classificação, agrupamento, regressão e previsão de séries temporais.

Palavras-chave: data mining; classificação; agrupamento; séries temporais

Notas de estudo elaboradas a partir do material das aulas.

Índice

1 Visão Geral da Disciplina **6**

2 Introdução e Análise exploratória de Dados **6**
2.1 O que é Data Mining . . . 6
2.2 Cinco problemas típicos e cinco classes de problemas . . . 7
2.3 A taxonomia de Machine Learning . . . 7
2.4 Análise Exploratória de Dados . . . 8
2.4.1 Tipos de variáveis . . . 8
2.4.2 Medidas-resumo . . . 8
2.5 Fontes e recomendações . . . 8
2.6 Base Mushroom . . . 9

3 Pré-processamento **9**
3.1 Missing values (valores faltantes) . . . 9
3.1.1 Substituição pela média . . . 9
3.1.2 Substituição pela média baseada em outro atributo . . . 10
3.1.3 Substituição pelo valor mais frequente . . . 10
3.2 Normalização . . . 10
3.3 Redução de dimensionalidade . . . 10
3.3.1 A maldição da dimensionalidade . . . 10
3.3.2 Objetivos . . . 11
3.3.3 Duas abordagens . . . 11
3.4 Seleção de atributos . . . 11
3.4.1 Filtros . . . 11
3.4.2 Wrappers . . . 11
3.4.3 Embarcados (embedded) . . . 12
3.5 Agregação de atributos e PCA . . . 12
3.6 Balanceamento de dados . . . 12
3.7 Outliers . . . 13
3.8 Estudos de caso . . . 13
3.8.1 SECOM . . . 13
3.8.2 IBM Analytics Employee Attrition and Performance . . . 14

4 Pré-processamento - Continuação **14**

5 Classificação **15**
5.1 Definição e aplicações . . . 15
5.2 Support Vector Machine (SVM) . . . 15
5.2.1 O separador ótimo . . . 15
5.2.2 Soft margin . . . 16
5.2.3 SVM não linear e o truque do kernel . . . 16
5.2.4 Vantagens e hiperparâmetros . . . 16
5.2.5 Compromisso viés-variância . . . 17
5.2.6 Estudo de caso: análise de crédito bancário . . . 17
5.3 Árvores de Decisão . . . 17
5.3.1 Estrutura . . . 17

5.3.2 O algoritmo ID3 . . . 17
5.3.3 Exemplo trabalhado: a base “Cheat” . . . 18
5.3.4 Vantagens . . . 18
5.4 Comitês (Ensembles) . . . 19
5.4.1 Ideia e motivação . . . 19
5.4.2 Formas de votação . . . 19
5.4.3 Quando usar, vantagens e desvantagens . . . 19
5.4.4 Bagging versus Boosting . . . 20
5.4.5 Stacking e RSM . . . 20
5.5 Random Forest . . . 20
5.6 Estudo de caso: Netflix Prize . . . 21

6 Classificação AD e RF **21**

7 Classificação - Parte 2 **22**
7.1 KNN — K Nearest Neighbors . . . 22
7.1.1 O algoritmo em cinco passos . . . 22
7.1.2 As três pendências . . . 22
7.1.3 Estratégias de desempate . . . 22
7.1.4 Estudo de caso: Câncer de Mama . . . 23
7.2 Regressão Logística . . . 23
7.2.1 Da regressão linear à logística . . . 23
7.2.2 Leitura da curva . . . 24
7.2.3 Inferência . . . 24

8 Associação **24**
8.1 O problema de associação . . . 24
8.2 Métricas fundamentais . . . 25
8.2.1 Interpretando o lift: o exemplo Netflix . . . 25
8.3 Preparação da base . . . 25
8.4 Definindo os parâmetros na prática . . . 26
8.5 Algoritmo Apriori . . . 26
8.6 Estudo de caso: transações de supermercado . . . 26
8.7 Algoritmo FP-Growth . . . 27
8.7.1 Construção da FP-tree . . . 27
8.7.2 Extração dos padrões frequentes . . . 28
8.8 Algoritmo Eclat . . . 28

9 Agrupamento **28**
9.1 Conceito de cluster . . . 28
9.2 O que é uma boa clusterização . . . 29
9.3 Aplicações . . . 29
9.4 Métodos de clusterização . . . 29
9.5 Métodos baseados em particionamento . . . 29
9.5.1 O algoritmo K-means em cinco passos . . . 30
9.5.2 Como determinar o número de clusters: WCSS e Elbow Method . . . 30
9.5.3 Variações do K-means . . . 31

9.6 Clusterização hierárquica . . . 31
9.6.1 Divisivos e aglomerativos . . . 31
9.6.2 Dendrograma e ponto de corte . . . 32
9.6.3 K-means versus Hierárquico . . . 32
9.7 Estudo de caso: clientes de um shopping . . . 32
9.8 Clusterização baseada em densidade: DBSCAN . . . 33
9.8.1 Conceitos . . . 33
9.8.2 O algoritmo . . . 33

10 Regressão **33**
10.1 Correlação e regressão . . . 33
10.2 Regressão linear simples . . . 34
10.2.1 Resíduos e mínimos quadrados . . . 34
10.3 Regressão linear múltipla . . . 34
10.4 Métricas de avaliação . . . 35
10.4.1 Construção do R2 . . . 35
10.4.2 Por que o R2 ajustado . . . 35
10.5 Estudo de caso: 50 Startups . . . 35
10.5.1 Dummy variables e a dummy variable trap . . . 36
10.5.2 Lendo a saída da regressão . . . 36
10.6 Árvores de Regressão . . . 36
10.7 Random Forest para regressão . . . 37
10.8 KNN para regressão . . . 37
10.9 Estudo de caso: aluguel de bicicletas . . . 37

11 Previsão de Séries Temporais **38**
11.1 Definição . . . 38
11.2 Tipos de processamento . . . 38
11.3 Componentes de uma série . . . 38
11.4 Modelos de previsão . . . 38
11.4.1 Naive (ingênuo) . . . 39
11.4.2 Médias móveis . . . 39
11.4.3 Amortecimento exponencial . . . 39
11.4.4 ARMA, ARIMA e SARIMA . . . 39
11.4.5 Modelos auto-regressivos não lineares . . . 39
11.5 Definições de projeto . . . 40
11.6 Arquiteturas NAR, NARX e Nonlinear Input-Output . . . 40
11.7 Estudo de caso: previsão de carga elétrica . . . 40
11.8 Outros estudos de caso . . . 41

12 Deploy (aula extra) **41**
12.1 Machine Learning em Python . . . 41
12.2 Modelo em produção . . . 42
12.3 Estudo de caso: medicamento anti-ansiedade . . . 42
12.4 Estudo de caso: sintomas de dor lombar . . . 42
12.5 Estudo de caso: doenças do coração . . . 43

13 Trabalho **43**

14 Síntese da Disciplina **44**

1 Visão Geral da Disciplina

A disciplina **DM251 — Data Mining** apresenta, de forma sistemática, o processo completo de mineração de dados: desde a compreensão do que é Data Mining e de como ele se insere no ciclo de Descoberta de Conhecimento em Bancos de Dados (KDD), passando pela análise exploratória e pelo tratamento dos dados, até a construção de modelos preditivos e descritivos e a sua colocação em produção. O material foi elaborado pela professora Manoela Kohler (prof.manoela@ica.ele.puc-rio.br).

O encadeamento das aulas segue o **esquema básico de um projeto de Data Mining**: primeiro entende-se o problema e explora-se a base (Aula 1); em seguida trata-se o dado — valores faltantes, normalização, conversão de atributos, redução de dimensionalidade, balanceamento e outliers (Aulas 3 e 4); só então entram os algoritmos de modelagem, organizados pelas **cinco classes de problemas de Data Mining**: previsão, classificação, regressão, agrupamento e associação. Ao final, discute-se como levar um modelo treinado para produção.

Do ponto de vista conceitual, o eixo organizador da disciplina é a taxonomia de **Machine Learning** em aprendizado **supervisionado** (classificação, regressão e previsão de séries temporais), **não supervisionado** (agrupamento e associação) e **por reforço** (aprendizado pela interação de agentes com um ambiente). Essa taxonomia é retomada explicitamente na abertura de quase todas as aulas, servindo de mapa mental para localizar cada novo algoritmo.

A disciplina é fortemente ancorada em **estudos de caso** reais e públicos: churn, consumo de cogumelos (base Mushroom), classificação de plantas (Iris), predição de atrito no ambiente de trabalho (IBM HR Analytics), monitoramento de processos industriais (SECOM), transações de varejo para regras de associação, segmentação de clientes de shopping, modelo para investidores (50 Startups), aluguel de bicicletas, previsão de faturamento e previsão de carga elétrica. A avaliação se dá por um **Trabalho Final**: um problema proposto pela professora via Classroom (ou proposto pelo próprio aluno), resolvido com a ferramenta de preferência e entregue em formato de relatório ou apresentação.

2 Introdução e Análise exploratória de Dados

2.1 O que é Data Mining

A **mineração de dados** é apresentada como parte do processo de **KDD (Knowledge Discovery in Databases)**. Segundo Goebel (1999), citado nos slides, o termo KDD representa o processo de tornar dados de baixo nível em conhecimento de alto nível, enquanto **mineração de dados** pode ser definida, mais estritamente, como a **extração de padrões ou modelos a partir de dados observados**.

Um conceito estruturante apresentado é a pirâmide **DIKW (Data-Information-Knowledge-Wisdom)**:

- **Dados**: resultado das experiências diárias de cada área da firma;
- **Informação**: alinhamento dos dados a um fim na companhia;
- **Conhecimento**: organização dos principais fatos, relações de causalidade e predições;
- **Sabedoria**: prática apoiada sobre uma base sólida de conhecimento.

Data Mining é, acima de tudo, **ir além do dado bruto**: associar para formar nichos, descrever para caracterizar padrões e prever para identificar tendências e limites — sempre com o objetivo de **agregar valor ao empreendimento**.

Três caracterizações complementares são destacadas:

1. É uma **área interdisciplinar**, que combina métodos e ferramentas de aprendizagem de máquina, estatística, banco de dados, sistemas especialistas e visualização de dados;
2. É parte importante de um projeto de **Business Intelligence** bem-sucedido;
3. É um **procedimento de extração de informação e conhecimento** em bases de dados.

2.2 Cinco problemas típicos e cinco classes de problemas

Os slides listam cinco problemas de negócio recorrentes e as perguntas típicas do gestor associadas a cada um:

- **Previsão de faturamento em varejo**: "Qual o faturamento esperado para o próximo trimestre?"; "Qual o pior cenário para o faturamento?";
- **Avaliação de crédito**: "Será que ele vai pagar o empréstimo?"; "Qual o melhor modelo de financiamento para esse cliente (juros, prazo)?";
- **Determinação de localidades promissoras para novas filiais**: "Qual o local mais promissor?"; "Em quanto tempo terei de volta o investimento?";
- **Definição de grupos de consumo para segmentação de mercado**;
- **Oferta de novos serviços e produtos** (Netflix, Amazon, Americanas.com): "Quem observa esse produto tem interesse em ver qual outro?"; "Quem observa esse produto costuma comprar qual outro?".

Esses cinco problemas mapeiam diretamente as **cinco classes de problemas de DM**: previsão, classificação, regressão, agrupamento e associação, respectivamente.

2.3 A taxonomia de Machine Learning

O Machine Learning é dividido em três grandes ramos:

- **Supervisionado**: os registros possuem **atributos** e um **rótulo**. O modelo é um **aproximador**, uma função que mapeia entradas em saída. Subdivide-se em:
  - **Classificação**: o rótulo é **categórico** (por exemplo, aprovado ou reprovado a partir dos dados do estudante);
  - **Regressão**: o rótulo é **contínuo** (por exemplo, a nota do estudante);
  - **Previsão de séries temporais**: o rótulo é contínuo e **dependente do tempo** — a partir de dados históricos, projeta-se a previsão.
- **Não supervisionado**: existem apenas atributos, sem rótulo. Subdivide-se em **agrupamento** (descoberta de semelhanças e grupos entre registros) e **associação** (descoberta de relações entre variáveis, expressa em **regras de associação**).
- **Por reforço**: aprendizado através da interação de agentes com um ambiente.

2.4 Análise Exploratória de Dados

O primeiro passo de qualquer análise de dados é **explorar os dados coletados**. A análise exploratória fornece uma ideia de como os dados se distribuem e qual forma apresentam; além disso, permite verificar se os **pressupostos teóricos** exigidos pela análise escolhida são ou não verificados.

A análise exploratória se organiza em três frentes:

1. **Organização e classificação dos dados**;
2. **Cálculo de medidas-resumo**;
3. **Visualização do conjunto de dados**.

2.4.1 Tipos de variáveis

São reconhecidos três tipos:

- **Variável categórica**: valores em um conjunto finito e **sem ordenação**. Exemplos: sexo, tipo de moradia, estado civil;
- **Variável discreta**: valores em um conjunto finito e **com ordenação**. Exemplos: idade, número de amigos, número de fotos;
- **Variável contínua**: valores em um conjunto infinito e **com ordenação**. Exemplos: peso, altura, tempo online.

O exemplo trabalhado em aula classifica um cadastro de rede social: `ID` e `Nome` como identificadores, `Idade` e `Número de Amigos` como discretas, `Sexo` como categórica e `Tempo Online no Facebook` como contínua.

2.4.2 Medidas-resumo

Medidas-resumo são valores numéricos obtidos a partir de uma amostra que (i) resumem a informação contida nos dados e (ii) exibem o comportamento da distribuição da amostra. Incluem valores centrais, valores extremais, dispersão, assimetria, média, mediana e moda. Os slides agrupam essas medidas em três famílias: **medidas de locação**, **medidas de distribuição de frequências** e **medidas de associação**.

Um esboço mínimo em Python da etapa exploratória:

```
import pandas as pd

iris = pd.read_csv("Iris.csv")
iris.head()
iris.describe()          # medidas-resumo das variaveis continuas
iris["Species"].value_counts()   # distribuicao de frequencias da categorica
iris.corr(numeric_only=True)     # medidas de associacao
```

2.5 Fontes e recomendações

Os slides indicam o **Kaggle** como repositório de bases e competições, e recomendam como leitura corrente: Medium, KDnuggets, arXiv e Papers with Code.

2.6 Base Mushroom

A base **Mushroom** (Kaggle, `uciml/mushroom-classification`) é o estudo de caso introdutório. Ela inclui descrições de amostras hipotéticas correspondentes a **23 espécies de cogumelos**. Cada espécie é identificada como definitivamente **comestível**, definitivamente **venenosa**, ou de consumo desconhecido e não recomendado — esta última classe foi combinada com a venenosa. O ponto didático é forte: o guia de onde as características foram retiradas **afirma claramente que não existe uma regra simples** para determinar a comestibilidade de um cogumelo. É exatamente esse tipo de situação — padrão complexo, não redutível a regra trivial — que justifica a mineração de dados.

3 Pré-processamento

Esta aula abre o bloco de **Tratamento de Dados**, precedido de uma recapitulação dos conceitos, do caráter interdisciplinar de DM, das cinco classes de problemas e do esquema básico de um projeto.

3.1 Missing values (valores faltantes)

Valores faltantes são **muito comuns no mundo real**. As causas apontadas nos slides são:

- atributos novos que surgem com o passar do tempo e com a necessidade de novas informações por parte das empresas;
- atributos não preenchidos por falta de obrigatoriedade.

Uma observação prática importante: **verificar se o número de registros com dados faltantes para um atributo específico não justifica, por si só, a remoção do atributo**.

O espaço de decisões é organizado assim:

- **Deletar**: o **registro** inteiro ou o **atributo** inteiro;
- **Imputar valor**: por **estatística** (um valor, ou uma sequência) ou por **machine learning**.

3.1.1 Substituição pela média

Considere a tabela usada em aula, com os atributos `Idade`, `Estado Civil`, `Nota` e `Atrito`:

Idade	Estado Civil	Nota	Atrito
35	Casado	9	Sim
53	?	7	Sim
68	Casado	10	Não
20	Solteiro	8	Não
29	Casado	?	Sim

A média das notas observadas é $(9 + 7 + 10 + 8) \,/\, 4 = 8.5$, e o valor faltante de `Nota` é substituído por **8.5**.

3.1.2 Substituição pela média baseada em outro atributo

Uma variante mais informativa condiciona a imputação a outro atributo. Restringindo a média às notas dos registros com `Atrito` igual a "Sim" e nota conhecida, obtém-se (9 + 7) / 2 = 8, e o faltante passa a ser **8**. A ideia central é que a imputação condicionada preserva melhor a estrutura dos dados do que a média global.

3.1.3 Substituição pelo valor mais frequente

Para atributos categóricos, imputa-se a **moda**. No exemplo, o `Estado Civil` faltante é preenchido com "Casado", que é o valor mais frequente na coluna.

```
from sklearn.impute import SimpleImputer

imp_num = SimpleImputer(strategy="mean")          # substituicao pela media
imp_cat = SimpleImputer(strategy="most_frequent") # valor mais frequente
```

3.2 Normalização

A **normalização** tem dois objetivos declarados:

- **dar aos atributos pesos iguais**, evitando que atributos com escalas maiores dominem o modelo;
- **diminuir o tempo de convergência dos algoritmos**.

Em seguida, os slides tratam da **conversão de atributos**, isto é, da transformação de atributos categóricos em representações numéricas utilizáveis pelos algoritmos.

3.3 Redução de dimensionalidade

3.3.1 A maldição da dimensionalidade

A **maldição da dimensionalidade** (curse of dimensionality) é o termo que se refere a vários fenômenos que surgem na análise de dados em espaços com muitas dimensões (atributos) — muitas vezes centenas ou milhares. A consequência prática: **adicionar características nem sempre significa melhora no desempenho de um classificador**. De modo geral, o desempenho de um classificador **tende a se degradar a partir de um determinado número de atributos, mesmo que sejam atributos úteis**.

3.3.2 Objetivos

A tarefa é **reduzir o número de atributos**, com três objetivos:

- diminuir o custo do aprendizado;
- aumentar a precisão do algoritmo;
- gerar modelos compactos e mais fáceis de interpretar.

Em geral espera-se que todos os atributos sejam relevantes, mas nem sempre é possível garantir isso; além disso, alguns atributos são **redundantes**. O objetivo é então **definir um conjunto de atributos relevantes e não redundantes**.

3.3.3 Duas abordagens

- **Seleção de atributos**: escolha de um **subconjunto** dos atributos disponíveis. Exemplos: filtros e wrappers;
- **Agregação de atributos**: **criação de novos atributos** a partir da combinação dos existentes. Exemplo: PCA.

3.4 Seleção de atributos

3.4.1 Filtros

Os métodos de **filtro** aplicam uma **medida estatística** para atribuir uma pontuação a cada atributo. Os atributos são então **ranqueados pela pontuação** e podem ser mantidos ou removidos do conjunto de dados. A grande vantagem é o baixo custo computacional, já que não é necessário treinar modelos.

O filtro detalhado em aula é o **ganho de informação**, em dois passos:

1. Calcula-se a **entropia para a classe** — a convenção usada é 0 quando os dados são homogêneos e 1 quando estão igualmente distribuídos;
2. Divide-se a base nos diferentes atributos, calcula-se a entropia para cada um deles e obtém-se o **ganho de informação** como a entropia para a classe menos a entropia para o atributo.

Em outras palavras: **o ganho de informação é baseado na diminuição da entropia depois que uma base de dados é subdividida em um atributo**.

3.4.2 Wrappers

Os métodos **wrapper** tratam a seleção de um conjunto de atributos como um **problema de busca**: diferentes combinações são preparadas, avaliadas e comparadas entre si. Um **modelo preditivo** é usado para avaliar cada combinação e atribuir uma pontuação baseada na precisão do modelo.

O exemplo dos slides usa um **algoritmo genético** sobre uma base com 5 atributos (A1 a A5). O **cromossomo** tem um gene por atributo, e cada gene assume 0 ou 1:

- **0** = sem o respectivo atributo;

- **1** = com o respectivo atributo.

Cada indivíduo criado durante a evolução do algoritmo genético é apresentado ao classificador. Toda a base, restrita aos atributos escolhidos, é usada para treinar o classificador, e a **função de avaliação** pode ser, por exemplo, a acurácia de treinamento. No exemplo, um indivíduo que seleciona três atributos alcança **acurácia de 30%**; o algoritmo genético evolui até chegar à resposta ótima, com um outro subconjunto atingindo **acurácia de 93%**.

3.4.3 Embarcados (embedded)

Nos métodos **embarcados**, o processo de seleção **faz parte do próprio algoritmo de aprendizado**. O exemplo canônico é a **Árvore de Decisão**, que a cada nó escolhe o atributo mais informativo — realizando seleção como subproduto do treinamento.

3.5 Agregação de atributos e PCA

A intuição da **agregação** é ilustrada de forma simples: dois atributos, "massa" e "volume", podem ser agregados em um único atributo, "densidade", com `densidade = massa / volume`. Nesse caso, **não há perda de informação**.

O método formal é o **PCA (Análise de Componentes Principais)**, que consiste em transformar um conjunto de variáveis originais em outro conjunto de variáveis de mesma dimensão, denominadas **componentes principais**. Suas propriedades importantes:

- cada componente principal é uma **combinação linear** de todas as variáveis originais;
- todos os componentes são **ortogonais entre si**, portanto não há informações redundantes;
- são estimados com o propósito de **reter, em ordem de estimação, o máximo de informação** em termos da variação total contida nos dados.

O resultado é a redução da massa de dados com a menor perda possível de informação. A análise de componentes principais é **completamente reversível**, o que a torna versátil para redução de dados e compressão de dados. No exemplo apresentado, com **limiar de variância de 0.95**, obtêm-se **31 componentes principais de um total de 49 atributos**.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.95)   # limiar de variancia explicada
X_reduzido = pca.fit_transform(X)
```

3.6 Balanceamento de dados

A grande maioria das bases de dados **não tem o mesmo número de instâncias em cada classe**; quando a diferença é pequena, não há problema. Em algumas bases esse comportamento é **esperado** — por exemplo, em bases de transações, a maioria será "Normal" e uma pequena minoria "Fraudulenta".

O problema surge quando o desbalanceamento é severo. Com **Classe 1 em 95% dos dados** e **Classe 2 em 5%**, os classificadores ficam **"preguiçosos"**: consegue-se **95% de acurácia** classificando tudo como Classe 1. É o chamado **Paradoxo da Acurácia**: a métrica é alta, mas o modelo não tem valor.

Os slides listam seis linhas de ação:

1. **Coletar mais dados**;
2. **Utilizar diferentes métricas de performance**: matriz de confusão, precisão, recall, Kappa e F1 score;
3. **Reamostrar o conjunto de dados**: **over-sampling** aleatório (replicar a classe rara) ou **under-sampling** aleatório (descartar parte da classe majoritária);
4. **Gerar amostras sintéticas**, por exemplo com **SMOTE (Synthetic Minority Over-Sampling Technique)** — método de over-sampling que gera amostras **sintéticas** da classe rara, ao invés de criar cópias, perturbando um atributo por vez por um valor dentro da diferença entre os exemplos vizinhos;
5. **Testar diferentes algoritmos** — não se deve ter um algoritmo favorito nesse caso;
6. **Penalização**: custo adicional para a classificação errada da classe rara. Com razão entre classes de 5:1, um erro classificado como Class 0 custa 1, enquanto um erro classificado como Class 1 custa 5.

Os slides mencionam ainda as **GANs** como recurso relacionado à geração de dados.

```
from imblearn.over_sampling import SMOTE

X_bal, y_bal = SMOTE(random_state=42).fit_resample(X, y)
```

3.7 Outliers

Algoritmos de aprendizado de máquina são **sensíveis à distribuição e ao intervalo dos dados**. **Outliers** podem enviesar e induzir ao erro, resultando em **maior tempo de treinamento, menor acurácia e modelos piores**.

Três famílias de técnicas de detecção são citadas:

1. **Análise de valores extremos**;
2. **Métodos de proximidade** (por exemplo, k-means);
3. **Métodos de projeção** (por exemplo, mapas de Kohonen).

3.8 Estudos de caso

3.8.1 SECOM

Um complexo processo de fabricação de semicondutores moderno é normalmente feito sob vigilância consistente através do **monitoramento de sinais**, variáveis coletadas de sensores e pontos de medição de processo. Nem todos esses sinais são igualmente valiosos: os sinais medidos contêm uma combinação de **informação útil, informação irrelevante e ruído**. Engenheiros tipicamente têm um número muito maior de sinais do que o realmente necessário. Tratando cada tipo de sinal

como uma característica, a **seleção de atributos** pode ser aplicada para identificar os sinais mais relevantes; os engenheiros de processo podem então usar esses sinais para determinar os fatores-chave da produção, com **redução de tempo e diminuição de custos**. A base SECOM tem **1567 exemplos e 591 atributos** — um caso paradigmático de alta dimensionalidade.

3.8.2 IBM Analytics Employee Attrition and Performance

Conjunto **fictício** criado por cientistas de dados da IBM, cujo objetivo é descobrir os fatores que levam ao desgaste dos funcionários e explorar questões como a influência da distância de casa em função do cargo e do atrito, ou a comparação da renda média mensal por educação e atrito. São **34 atributos**, entre eles escalas ordinais codificadas:

- `Education`: 1 Below College, 2 College, 3 Bachelor, 4 Master, 5 Doctor;
- `EnvironmentSatisfaction`, `JobInvolvement`, `JobSatisfaction`, `RelationshipSatisfaction`: 1 Low, 2 Medium, 3 High, 4 Very High;
- `PerformanceRating`: 1 Low, 2 Good, 3 Excellent, 4 Outstanding;
- `WorkLifeBalance`: 1 Bad, 2 Good, 3 Better, 4 Best;
- além de `Gender`, `MaritalStatus`, `MonthlyIncome`, `Age`, `DistanceFromHome`, `OverTime`, `JobRole`, `YearsWithCurrentManager`, `Over18`, entre outros.

A variável-alvo `Attrition` tem duas classes: Yes e No.

4 Pré-processamento - Continuação

Esta aula dá continuidade ao mesmo conjunto de slides (“Tratamento de Dados”), aprofundando os tópicos já introduzidos: imputação de valores faltantes, normalização e conversão de atributos, redução de dimensionalidade (filtros por ganho de informação, wrappers com algoritmos genéticos, métodos embarcados e PCA), balanceamento com over-sampling, under-sampling, SMOTE e penalização, e tratamento de outliers.

O foco desta continuação está na **aplicação prática** dos conceitos nos estudos de caso SECOM e IBM Attrition, já descritos na seção anterior. Vale destacar aqui a articulação entre as etapas, que é o ponto pedagógico central do bloco:

- **ordem importa**: a imputação de faltantes deve preceder a normalização, e ambas devem preceder a seleção de atributos, sob pena de vazamento de informação ou de resultados inconsistentes;
- **balanceamento é operação de treino**: reamostragem sintética deve ser aplicada apenas ao conjunto de treinamento, para que a avaliação continue refletindo a distribuição real;
- **a escolha da métrica é parte do tratamento**: como visto no paradoxo da acurácia, a decisão sobre precisão, recall, F1 ou Kappa é indissociável da decisão sobre balanceamento.

Um pipeline ilustrativo, encadeando as etapas na ordem discutida:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
```

```
pipe = Pipeline([
    ("imputacao", SimpleImputer(strategy="mean")),
    ("normalizacao", StandardScaler()),
    ("reducao", PCA(n_components=0.95)),
])
X_tratado = pipe.fit_transform(X_treino)
```

5 Classificação

5.1 Definição e aplicações

A **classificação** é um problema de **treinamento supervisionado**: os dados da base **possuem indicação da classe** à qual pertencem. O modelo aprende a mapear atributos em rótulos categóricos.

Aplicações citadas: reconhecimento de padrões (reconhecimento de pessoas, de voz, de doenças vocais, de dígitos), contratos em negociação, detecção de fraude (energia elétrica, cartão de crédito) e análise de crédito.

Os temas desta aula são: **Support Vector Machine**; e, no bloco de métodos baseados em árvores e técnicas de comitê, **Árvores de Decisão**, **Bagging e Boosting** e **Random Forest**.

5.2 Support Vector Machine (SVM)

O **SVM** é um método supervisionado de aprendizado de máquina que **muitas vezes apresenta resultados melhores que muitos métodos populares de classificação**. Foi originalmente concebido para lidar com **classificações binárias**, embora a maior parte dos problemas reais requeira múltiplas classes.

5.2.1 O separador ótimo

A pergunta fundamental é: como separar as classes? Reta, plano ou hiperplano? E qual é o **hiperplano ótimo**? O critério imediato é o **menor erro de classificação**, mas ele não é suficiente, pois em geral existem infinitos separadores que classificam corretamente o conjunto de treinamento.

Para o caso linearmente separável, o separador é uma função da forma $f(x) = ax + b$, com os dois lados definidos por $ax + b < 0$ e $ax + b > 0$.

O critério do SVM é o **hiperplano de margem máxima**: entre todos os separadores válidos, escolhe-se aquele que maximiza a **margem de classificação**, denotada por rho nos slides. Os dois objetivos simultâneos são:

- **classificar corretamente todos os dados de treinamento**;
- **maximizar a margem e minimizar o erro**.

Os pontos que ficam exatamente sobre as bordas da margem são os **vetores de suporte**. A consequência é elegante e prática: **somente os vetores de suporte interessam** — eles definem o hiperplano separador, e os outros dados do conjunto de treinamento **não têm importância depois do modelo criado**.

5.2.2 Soft margin

E se os dados não forem linearmente separáveis? **Variáveis de folga** (xi) podem ser incluídas para **permitir erro de classificação** de exemplos difíceis ou com ruído. Essa formulação é chamada de **soft margin**.

5.2.3 SVM não linear e o truque do kernel

Dados linearmente separáveis, mesmo com algum ruído, funcionam bem. Mas e se a base for mais complexa? A ideia é **mapear os dados para uma dimensão maior**.

A ideia geral: o espaço de atributos original pode — com alta probabilidade, pelo **Teorema de Cover** — ser mapeado para um espaço de atributos maior, onde os dados podem ser separados linearmente. Formalmente, aplica-se um mapeamento Phi que leva x em phi(x).

O **truque do kernel** ("kernel trick") resolve isso sem calcular explicitamente o mapeamento: a forma mais simples de separar grupos de dados é com uma reta ou hiperplano, mas existem situações em que não é possível separar os dados com um separador linear, ou em que uma **região não linear** separa os grupos de forma mais eficiente. O SVM trata esse problema usando uma **função kernel (não linear)** para mapear os dados em um espaço de atributos diferente e maior. Isso significa que **uma função não linear é aprendida por uma máquina de aprendizado linear em um espaço de atributos de dimensão maior**.

5.2.4 Vantagens e hiperparâmetros

Vantagens do SVM:

- consegue lidar bem com grandes conjuntos de exemplos;
- trata bem dados de **alta dimensão**;
- o processo de **classificação é rápido**.

Os dois hiperparâmetros centrais, relacionados à **regularização**, são:

- **C**: o **custo das violações**. Faz o balanceamento entre uma fronteira de decisão suave e uma fronteira que classifique corretamente todos os pontos;
- **Gamma**: o **raio de influência**. Valores baixos significam influência longe, produzindo separação mais suave; valores altos significam influência perto, produzindo um separador que valoriza a classificação correta.

```
from sklearn.svm import SVC

modelo = SVC(kernel="rbf", C=1.0, gamma="scale")
modelo.fit(X_treino, y_treino)
```

5.2.5 Compromisso viés-variância

Um conceito válido **para modelos de Machine Learning em geral**:

- **Viés (bias)**: a incapacidade do modelo de ML de capturar o verdadeiro relacionamento entre os dados;
- **Variância**: a diferença no resultado do modelo de ML para diferentes conjuntos de dados.

Os hiperparâmetros C e Gamma do SVM são, exatamente, os controles desse compromisso.

5.2.6 Estudo de caso: análise de crédito bancário

A base contém **2077 exemplos** de créditos concedidos que foram pagos ou não. Possui **11 atributos de entrada** e **2 classes de saída**: a saída indica se o cliente pagou o empréstimo (=1) ou não pagou (=0).

5.3 Árvores de Decisão

5.3.1 Estrutura

Árvores de decisão criam modelos de classificação na forma de **estruturas**: quebra-se um conjunto de dados em subconjuntos cada vez menores enquanto, simultaneamente, são criadas as árvores de decisão associadas. O resultado final é uma árvore com **nós de decisão** e **nós folhas**.

Características estruturais:

- uma árvore possui um ou mais **ramos**;
- os **nós folhas** representam uma classificação ou decisão;
- o nó mais alto corresponde ao **melhor classificador** e é chamado de **nó raiz**;
- árvores podem lidar tanto com **dados numéricos quanto categóricos**.

A leitura semântica é direta: **cada nó de decisão contém um teste de um atributo**, **cada folha está associada a uma classe** e **cada percurso da raiz à folha corresponde a uma regra de classificação**. É por isso que árvores são **fáceis de implementar e de interpretar**; a estratégia é a de **dividir para conquistar**.

5.3.2 O algoritmo ID3

O **ID3**, de J. R. Quinlan, tem as seguintes características:

- **busca gulosa de cima para baixo** pelo espaço de possíveis ramos;
- **sem backtracking**;
- **pode ficar preso em um ótimo local**;
- **difícil de usar em variáveis contínuas**.

O ID3 utiliza a **entropia** para calcular a homogeneidade dos dados: se os dados são completamente homogêneos, a entropia é **zero**; se estão divididos igualmente, a entropia é **1**.

Os quatro passos do algoritmo:

1. **Calcula-se a entropia para a classe** (0 = dados homogêneos; 1 = dados igualmente distribuídos);
2. **Divide-se a base nos diferentes atributos** e calcula-se a entropia para cada um deles; calcula-se o **ganho de informação** como entropia para a classe menos entropia para o atributo. O ganho de informação é baseado na **diminuição da entropia** depois que a base é subdividida em um atributo;
3. **Escolhe-se o atributo com maior ganho de informação como nó de decisão**. Um ramo que tenha **entropia 0 é uma folha**; um ramo com **entropia maior que 0 precisa ser subdividido**;
4. O algoritmo é **rodado recursivamente** para todos os ramos sem folha até que todos os dados sejam classificados.

O resultado final pode ser lido como um conjunto de **regras de decisão**.

Note-se que este é exatamente o critério de **filtro por ganho de informação** visto no pré-processamento — o que explica por que a árvore de decisão é o exemplo canônico de seleção de atributos **embarcada**.

5.3.3 Exemplo trabalhado: a base "Cheat"

Os slides constroem uma árvore a partir de dez registros com os atributos `Refund`, `Marital Status`, `Taxable Income` e a classe `Cheat`:

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes

(os demais registros seguem o mesmo padrão: 6 No/Married/60K/No; 7 Yes/Divorced/220K/No; 8 No/Single/85K/Yes; 9 No/Married/75K/No; 10 No/Single/90K/Yes).

Os **atributos de divisão** escolhidos produzem a árvore: `Refund` na raiz; se Yes, folha NO; se No, testa-se `MarSt`; se Married, folha NO; se Single ou Divorced, testa-se `TaxInc`, com folha NO abaixo de 80K e folha YES acima de 80K.

A **inferência** de um dado novo — `Refund` = No, `Marital Status` = Married, `Taxable Income` = 80K — percorre a árvore a partir do nó raiz: Refund = No leva ao teste de estado civil; Married leva diretamente à folha NO. O **valor inferido para `Cheat` é `No`**.

5.3.4 Vantagens

Árvores de decisão são **fáceis de entender**, funcionam **mais eficientemente com atributos discretos** e são **extremamente rápidas em classificar dados novos**.

```
from sklearn.tree import DecisionTreeClassifier
```

```
arvore = DecisionTreeClassifier(criterion="entropy")
arvore.fit(X_treino, y_treino)
```

5.4 Comitês (Ensembles)

5.4.1 Ideia e motivação

Comitês consistem em **agregar múltiplos modelos treinados com o objetivo de melhorar o desempenho do modelo conjunto**. A intuição: simula-se o que fazemos quando combinamos o conhecimento de especialistas em um processo de tomada de decisão.

São conhecidos por muitos nomes: comitês especialistas, sistemas múltiplos de classificação, comitê de classificadores, máquina de comitê, mistura de especialistas, aprendizado em conjunto, **ensemble**. Diversos estudos demonstram sua utilização com sucesso em problemas onde um único especialista não funciona bem.

A pergunta que motiva o método: às vezes cada técnica de aprendizado retorna **hipóteses (funções) diferentes**, mas **nenhuma hipótese perfeita**. Poderíamos combinar várias hipóteses imperfeitas para se ter uma hipótese melhor?

As analogias oferecidas: eleições combinam votos de eleitores para escolher um bom candidato; comitês combinam opiniões de especialistas para tomar decisões melhores; estudantes trabalham em conjunto em um projeto. A intuição formal: **indivíduos cometem erros, mas a maioria é menos propensa a erros**; indivíduos em geral têm **conhecimento parcial**, e um comitê pode juntar esse conhecimento.

5.4.2 Formas de votação

As regras de combinação listadas são: maioria dos votos, maioria ponderada dos votos, **Borda count**, média, média ponderada, soma, soma ponderada, produto, máximo, mínimo e mediana.

5.4.3 Quando usar, vantagens e desvantagens

Quando usar: quando se tem um conjunto muito grande de dados; quando a região de domínio do problema é muito complexa; quando se quer melhorar os resultados de classificadores individuais.

Vantagens: a combinação de modelos pode apresentar melhor desempenho que um modelo só; neutraliza ou minimiza fortemente a **instabilidade inerente** aos algoritmos de aprendizagem.

Desvantagens: não há garantia de que as estruturas modulares apresentem os melhores resultados; modelos combinados são **mais difíceis de analisar**; o **custo é alto**.

5.4.4 Bagging versus Boosting

A distinção central:

- No **Boosting**, as observações **classificadas incorretamente são atribuídas maiores pesos**, de modo que os modelos seguintes se concentrem nos casos difíceis;
- Na combinação final, **Bagging** usa **média ou maioria de votos**, enquanto **Boosting** usa **média ponderada** (classificadores com resultados melhores têm pesos maiores).

Ainda sobre Boosting: durante a etapa de treinamento, o **erro de treinamento é guardado**, e existe uma **condição para determinar se um modelo será utilizado ou descartado**. Algoritmos citados: **AdaBoost, LPBoost, XGBoost, GradientBoost, BrownBoost**.

5.4.5 Stacking e RSM

O **Stacking** é apresentado como outra técnica de comitê, na qual as saídas dos modelos-base alimentam um meta-modelo.

O **Random Subspace Method (RSM)** é **similar ao Bagging, mas com aleatorização sobre os atributos**: os classificadores-base aprendem em **subespaços S de mesma dimensão** e a decisão final é por **votação**.

5.5 Random Forest

O **Random Forest** é criado através de **árvores de decisão individuais cujos parâmetros podem variar aleatoriamente**. Ele combina três ingredientes já vistos: **Bagging (bootstrap)**, **Random subspace method** e **maioria de votos**.

O workflow é descrito assim:

- **cada árvore é construída usando uma inicialização (bootstrap) diferente do dataset original**;
- **aproximadamente 1/3 dos casos são deixados de fora** da amostra de inicialização e não são usados na construção daquela árvore;
- cada caso deixado de fora é apresentado a cada uma das árvores da floresta, e cada árvore retorna uma classificação. Para cada caso apresentado à floresta, verifica-se **a classe que obteve o maior número de votos**;
- a **proporção de votos diferentes da classe target em relação ao total de votos é o erro OOB (Out-Of-Bag estimate)**.

O erro OOB é um recurso valioso: fornece uma estimativa de generalização **sem necessidade de um conjunto de validação separado**.

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=500, oob_score=True)
rf.fit(X_treino, y_treino)
print(rf.oob_score_)
```

5.6 Estudo de caso: Netflix Prize

O **Netflix Prize** oferecia um **prêmio de 1 milhão de dólares** por uma **melhora de 10% na acurácia** do sistema de recomendação de filmes da Netflix em relação ao modelo então em uso. O ponto central para a aula: **os melhores times combinaram diversos modelos e algoritmos em um comitê**.

A tarefa era de **aprendizado supervisionado**: os dados de treinamento eram formados por um conjunto de usuários e as avaliações de filmes (1, 2, 3, 4 ou 5 estrelas) feitas por esses usuários; o objetivo era construir um classificador que, dado um usuário e um filme não avaliado, classificasse corretamente aquele filme como 1, 2, 3, 4 ou 5 estrelas.

6 Classificação AD e RF

Esta aula retoma o mesmo conjunto de slides da anterior, com foco aprofundado nas **Árvores de Decisão (AD)** e no **Random Forest (RF)**, incluindo a execução dos estudos de caso.

O ponto de articulação didático entre os dois métodos merece destaque. A **árvore de decisão individual** é interpretável e rápida, mas sofre de **instabilidade**: pequenas variações no conjunto de treinamento podem produzir árvores estruturalmente muito diferentes, porque a busca gulosa do ID3 não faz backtracking e pode ficar presa em ótimos locais. É precisamente essa instabilidade que os comitês **neutralizam ou minimizam fortemente**, como enunciado nas vantagens dos comitês.

O **Random Forest** ataca a instabilidade em duas frentes simultâneas:

1. **Bagging (bootstrap)** — cada árvore vê uma reamostragem diferente dos **registros**, o que descorrelaciona os erros entre árvores;
2. **Random subspace** — cada árvore vê um subconjunto aleatório dos **atributos**, o que impede que um único atributo dominante apareça na raiz de todas as árvores.

A agregação por **maioria de votos** converte árvores individualmente instáveis em um preditor estável. E o **erro OOB**, calculado sobre o terço de casos deixado de fora de cada bootstrap, dá a estimativa de desempenho sem custo adicional de dados.

O preço pago é a **interpretabilidade**: enquanto uma única árvore fornece um conjunto legível de regras de decisão, uma floresta de centenas de árvores é, nas palavras dos slides, um modelo combinado **mais difícil de analisar** e de **custo alto**. Esse é o trade-off que o analista precisa avaliar caso a caso — em contextos regulados, como a **análise de crédito bancário** usada como estudo de caso, a exigência de explicabilidade pode favorecer a árvore única; em contextos de puro desempenho preditivo, a floresta tende a vencer.

Os estudos de caso trabalhados são o de **crédito bancário** (2077 exemplos, 11 atributos de entrada, 2 classes) e o **Netflix Prize**.

7 Classificação - Parte 2

Esta aula amplia o repertório de classificadores com dois métodos adicionais: **KNN** e **Regressão Logística**. Antes disso, recapitula SVM (o mapeamento Phi levando x em phi(x) para um espaço de atributos maior pelo Teorema de Cover), Árvores de Decisão com ID3 e regras de decisão, comitês (Bagging, Boosting, Stacking, RSM) e Random Forest com o workflow de bootstrap e erro OOB.

Uma atualização importante aparece no diagrama de aprendizado supervisionado: na coluna de **Classificação**, a saída passa a ser descrita como **probabilidade de aprovação** a partir dos dados do estudante — o que prepara diretamente a introdução da regressão logística e do pós-processamento por limiar de probabilidade.

7.1 KNN — K Nearest Neighbors

7.1.1 O algoritmo em cinco passos

O **KNN** classifica um novo registro pela **classe majoritária entre seus K vizinhos mais próximos**:

1. **Determinar o valor de K**, ou número de vizinhos;
2. **Calcular a distância** entre cada par de registros;
3. **Determinar quais são os K registros (vizinhos) mais próximos** do novo registro;
4. **Dentre esses K vizinhos, contar o número de vizinhos em cada classe**;
5. **O novo registro vai ser da classe majoritária** entre os vizinhos mais próximos.

O exemplo dos slides usa K = 5 e **distância euclidiana** em um espaço bidimensional (X1, X2). Entre os 5 vizinhos mais próximos, contam-se **3 vizinhos de uma classe e 2 de outra**; o novo registro recebe, portanto, a classe com 3 vizinhos.

7.1.2 As três pendências

O enunciado sintético do método é: **a classe do novo padrão é igual à da K maioria mais próxima**. Isso deixa três pendências práticas:

- **Qual tipo de distância usar?**
- **Qual valor de K?** — a resposta dada é **escolha experimental**;
- **Como desempatar?** — três estratégias são oferecidas: escolha aleatória, escolha aleatória ponderada e classe mais próxima.

7.1.3 Estratégias de desempate

Supondo K = 4 e um empate de 2 contra 2:

- **Escolha aleatória**: "jogue uma moeda honesta" — caso saia cara, escolha a classe vermelha; caso saia coroa, escolha a classe azul;
- **Escolha aleatória ponderada**: "jogue uma moeda desonesta" — dê mais chance à classe que está relacionada à classe que possua mais padrões;

- **Classe mais próxima**: selecione a classe cuja **distância é menor**.

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5, metric="euclidean")
knn.fit(X_treino, y_treino)
proba = knn.predict_proba(X_teste)
```

7.1.4 Estudo de caso: Câncer de Mama

A base vem da **University of Wisconsin, Clinical Sciences Center**, e contém **30 atributos + classe + id**, entre eles:

- **Raio**: distância média do centro a pontos no perímetro do tumor;
- **Textura**: desvio padrão dos valores em escala de cinza;
- **Perímetro**;
- **Área**;
- entre outros.

São **569 instâncias: 357 benignas e 212 malignas**.

O exercício proposto tem três partes:

1. **Criar um modelo KNN** para classificar os tumores em maligno ou benigno, na ferramenta de preferência (Python e/ou RapidMiner), usando o arquivo `breastCancer.csv`;
2. Fazer um **pós-processamento conservador**: **somente inferências com 80% de probabilidade serão classificadas como benigno**;
3. Repetir o exercício para a base `breastCancer_3classes.csv`, que tem uma classe a mais: a classe **Suspeito**.

O item 2 é conceitualmente rico: ele mostra que a saída de um classificador não precisa ser usada com o limiar padrão de 0.5. Em um contexto clínico, o **custo assimétrico dos erros** — declarar benigno um tumor maligno é muito pior que o contrário — justifica exigir alta confiança para a classe benigna. É a mesma lógica da **penalização** vista no balanceamento de dados.

Os arquivos de apoio da aula incluem notebooks para KNN, KNN com **Optuna** (otimização de hiperparâmetros) e Regressão Logística sobre essa base.

7.2 Regressão Logística

7.2.1 Da regressão linear à logística

O ponto de partida é a **regressão linear simples**. Aplicada a um problema de classificação, ela apresenta um defeito evidente: pensando **em termos de probabilidades**, os "pedaços" da reta que caem fora do intervalo entre 0 e 1 **não fazem sentido**. Seria preciso alterar o modelo para representar melhor o problema.

A solução é **aplicar uma função sigmoidal** à saída da regressão linear:

- **Regressão linear**: $y = b_0 + b_1 x_1$;

- **Função sigmoidal**: $f(a) = 1/(1 + e^{-a})$;
- **Regressão logística**: $y' = f(y) = 1/(1 + e^{-(b_0+b_1x_1)})$.

O resultado é uma curva em S que mapeia qualquer valor real no intervalo entre 0 e 1, podendo assim ser lida como **probabilidade**.

7.2.2 Leitura da curva

O exemplo dos slides relaciona **idade** (eixo horizontal) e **probabilidade** (eixo vertical). Ao longo da curva sigmoidal, obtêm-se as seguintes probabilidades para diferentes idades:

Idade	Probabilidade
20	0.7%
30	23%
40	85%
50	99.4%

Repare no comportamento característico: a curva é **muito plana nas extremidades** e **íngreme no meio**. Uma variação de 10 anos entre 20 e 30 muda a probabilidade em cerca de 22 pontos percentuais; a mesma variação entre 40 e 50 muda em pouco mais de 14 pontos. O efeito marginal da variável independente **não é constante**, ao contrário do que ocorre na regressão linear.

7.2.3 Inferência

Para transformar a probabilidade em uma decisão de classe, usa-se um **limiar**. O slide de inferência mostra o corte em **0.5** sobre o eixo Y: acima dele, uma classe; abaixo, a outra. Como visto no exercício de câncer de mama, esse limiar pode e deve ser ajustado conforme o custo dos erros.

```
from sklearn.linear_model import LogisticRegression

logit = LogisticRegression()
logit.fit(X_treino, y_treino)
p = logit.predict_proba(X_teste)[:, 1]
classe = (p >= 0.8).astype(int)   # pos-processamento conservador
```

8 Associação

8.1 O problema de associação

A **associação** pertence ao ramo **não supervisionado** do Machine Learning e visa a **descoberta de relações entre variáveis**, expressas na forma de **regras de associação**. É a técnica por trás da análise de cesta de compras e dos sistemas de recomendação.

8.2 Métricas fundamentais

As três métricas centrais das regras de associação são **suporte**, **confiança** e **lift**. Para uma regra da forma "se A então B", denotada A implica B:

- **Suporte** de um conjunto de itens: a proporção de transações que contém aquele conjunto;
- **Confiança**: a proporção de transações que contêm B **dentre aquelas que contêm A**. Formalmente, Confiança(A implica B) = Suporte(A interseção B) / Suporte(A);
- **Lift**: Lift = Confiança(A implica B) / Suporte(B), que equivale a Suporte(A interseção B) / (Suporte(A) vezes Suporte(B)).

Uma propriedade importante do lift, destacada nos slides: **independe da ordem** — o lift de A implica B é igual ao de B implica A. O que muda entre as duas direções é o **suporte da premissa** e a **confiança da regra**.

8.2.1 Interpretando o lift: o exemplo Netflix

O exemplo é construído com 100 usuários da Netflix:

- **10 gostaram** da série "Breaking Bad" — logo, Suporte(Breaking Bad) = 0.10;
- **40 gostaram** da série "Dexter" — logo, Suporte(Dexter) = 0.40;
- **7 pessoas** que gostaram de "Breaking Bad" também gostaram de "Dexter".

Hipótese: é provável que quem viu "Dexter" goste também de "Breaking Bad".

Para a regra **"Se Dexter então Breaking Bad"**:

- se a Netflix recomendasse "Breaking Bad" para uma pessoa aleatoriamente, a probabilidade dessa pessoa gostar da série seria de **10%**;
- se a Netflix utilizar o conhecimento a priori (pessoas que gostaram de "Dexter" também gostarão de "Breaking Bad"), a probabilidade de uma pessoa gostar de "Breaking Bad" é de **17.5%**;
- **Lift**: melhora de **75%** utilizando o conhecimento a priori.

Para a regra inversa, **"Se Breaking Bad então Dexter"**:

- recomendação aleatória de "Dexter": probabilidade de **40%**, ou seja, Suporte = 0.4;
- com o conhecimento a priori: a probabilidade de gostar de "Dexter" é de 7/10 = **70%**, ou seja, Confiança = 0.7;
- **Lift** = 0.7 / 0.4 = **1.75**, ou seja, **75% de melhora** — confirmando a simetria do lift.

A leitura do lift é, portanto: **medida de melhora na recomendação considerando o conhecimento a priori**.

8.3 Preparação da base

A base deve ser **transformada em uma matriz esparsa** para que possa ser apresentada ao algoritmo de associação. O exemplo dos slides parte de transações em formato de lista:

id	Itens
1	Ovo, Leite, Manteiga
2	Manteiga
3	Chocolate
4	Água
5	Refrigerante, Água
6	Leite, Maçã

E converte para uma matriz binária com uma coluna por produto (Ovo, Leite, Manteiga, Chocolate, Água, Refrigerante, Maçã), na qual a transação 1 recebe 1 nas colunas Ovo, Leite e Manteiga e 0 nas demais; a transação 5 recebe 1 em Água e Refrigerante; e assim por diante.

8.4 Definindo os parâmetros na prática

As dicas práticas dos slides são valiosas:

Support: parte-se de um objetivo de negócio. Para otimizar a venda de produtos comprados **pelo menos 5 vezes ao dia**, em uma base semanal com 7501 transações, o cálculo é 5 vezes 7 dividido por 7501, resultando em **suporte mínimo de aproximadamente 0.005**.

Confidence: um valor **baixo** gera regras que não fazem sentido; um valor **alto** gera regras óbvias. A recomendação é **começar com o valor default (0.8) e ir diminuindo** por tentativa e erro. Observação: um valor de 0.8 significa que todas as regras geradas devem estar corretas em **80% das transações**.

Lift: **ordenar pelo lift em ordem decrescente** para priorizar as regras mais acionáveis.

8.5 Algoritmo Apriori

O **Apriori** é o algoritmo clássico de mineração de regras de associação. Sua ideia é a construção incremental de conjuntos de itens frequentes, eliminando candidatos que não atingem o suporte mínimo antes de expandi-los. Os slides percorrem um exemplo completo, passo a passo, gerando novas regras a cada iteração.

8.6 Estudo de caso: transações de supermercado

A base é uma lista de transações (compras) em um **mercado francês**:

- cada linha da base é uma **transação**;
- cada transação tem de **1 a N itens**;
- existem **119 produtos diferentes** no mercado;
- a base tem **7501 transações** feitas no decorrer de uma semana.

Uma dica é fornecida: **"mineral water", "eggs" e "spaghetti" são os 3 itens mais comprados** no supermercado. O exercício sugere colocar T = 0 no operador Apriori (ordenando por confidence) e observar o resultado, para depois voltar o parâmetro T para 1 (ordenando por lift) — o contraste evidencia por que o lift é a métrica preferida para priorização.

Os benefícios de negócio esperados:

- perceber associações e implementar estratégias de oferta desses produtos;
- perceber **associações nada óbvias**;
- automatizar o sistema de recomendação;
- aumentar vendas;
- oferecer produtos condizentes com o perfil do cliente.

```
from mlxtend.frequent_patterns import apriori, association_rules

itens = apriori(matriz_esparsa, min_support=0.005, use_colnames=True)
regras = association_rules(itens, metric="confidence", min_threshold=0.8)
regras.sort_values("lift", ascending=False).head(10)
```

8.7 Algoritmo FP-Growth

O **FP-Growth (Frequent Pattern)** é um dos algoritmos mais populares para o cálculo de termos frequentes. Suas características:

- usa uma **representação eficiente da base de dados** na forma de uma estrutura em árvore, a **FP-tree**;
- faz **dois scans no banco de dados**: o primeiro para contar itens frequentes, o segundo para construir a FP-tree;
- uma vez construída a FP-tree, usa uma abordagem recursiva de **divide-and-conquer** para obter os conjuntos de itens frequentes.

8.7.1 Construção da FP-tree

Parte-se de uma base com 10 transações sobre os itens A a E. O primeiro scan produz a tabela de frequências e prioridades:

Item	Frequência	Prioridade
A	7	1
B	8	2
C	7	3
D	4	4
E	3	5

Os itens de cada transação são então **reordenados por prioridade**. Assim, a transação 1, originalmente {B, A}, torna-se {A, B}; a transação 2, {D, C, B}, torna-se {B, C, D}; a transação 3, {E, D, C, A}, torna-se {A, C, D, E}; e assim por diante.

A árvore é construída incrementalmente a partir de um nó `null`. Após ler a transação 1, tem-se o caminho A:1 seguido de B:1. Após a transação 2, cria-se um novo ramo B:1, C:1, D:1. Após a transação 3, o nó A passa a A:2 e cria-se abaixo dele o caminho C:1, D:1, E:1.

A árvore final apresenta A:7 e B:3 como filhos da raiz, com os ramos internos B:5, C:3, C:1, D:1, C:3, D:1, E:1 e D:1. **Apontadores** ligam todas as ocorrências de um mesmo item e são usados para facilitar a geração dos termos frequentes; a **header table** registra as frequências (A 7, B 8, C 7, D 4, E 3).

8.7.2 Extração dos padrões frequentes

Com a árvore construída, definem-se os padrões frequentes. Supondo **suporte mínimo igual a 2**, percorrem-se os itens em **ordem inversa de frequência** — portanto, começando por E, depois D, e assim por diante.

Para o item **E**:

- **Conditional pattern base**: P = {(A, C, D : 1), (A, D : 1), (B, C : 1)};
- **Conditional FP-tree**: {(A:2, C:1, D:2), (B:1, C:1)};
- **Frequent Patterns**: {(A, E : 2), (D, E : 2), (A, D, E : 2)}.

Para o item **D**:

- **Conditional pattern base**: P = {(A, B, C : 1), (A, B : 1), (A, C : 1), (A : 1), (B, C : 1)};
- **Conditional FP-tree**: {(A:4, B:2, C:2), (B:1, C:1)};
- **Frequent Patterns**: {(A, D : 4), (B, D : 2), (C, D : 2), (A, B, D : 2), (A, C, D : 2), (B, C, D : 2), (A, B, C, D : 2)}.

O mesmo procedimento é feito para todos os itens e todas as conditional trees, em ordem decrescente de frequência.

8.8 Algoritmo Eclat

Uma observação importante encerra a aula: **a saída do algoritmo Eclat não fornece regras, mas sim a lista dos itens mais frequentemente comprados juntos**. Ou seja, o Eclat entrega **itemsets frequentes**, não implicações com antecedente e consequente — e portanto não produz confiança nem lift no formato de regra.

9 Agrupamento

9.1 Conceito de cluster

Um **cluster** é uma coleção de objetos que são **similares aos objetos do mesmo cluster** e **dissimilares aos objetos de outros clusters**. **Clusterização** é o agrupamento de conjuntos de dados em clusters, e constitui uma **classificação não supervisionada: sem classes predefinidas**.

Os slides advertem que **a noção de um cluster pode ser ambígua** — o mesmo conjunto de pontos pode admitir agrupamentos igualmente defensáveis com diferentes números de grupos.

Na **classificação não supervisionada**, não se conhece o padrão nem o número total de grupos a serem encontrados. O conjunto de dados é particionado em grupos, baseados em características específicas, de tal forma que os pontos dentro de um grupo sejam **mais similares** do que os pontos de outros grupos.

9.2 O que é uma boa clusterização

Uma boa clusterização sempre produz clusters com:

- **alta similaridade dentro das classes**;
- **baixa similaridade entre as classes**.

A qualidade dos resultados depende da **medida de similaridade usada** e do **método e sua implementação**.

9.3 Aplicações

Os slides citam aplicações com referências:

- **Marketing**: identifica grupos distintos de clientes, útil para desenvolver programas de marketing (CHIANG, 2003);
- **Uso da terra**: identifica a possibilidade de alocação de uso da terra para fins agrários e/ou urbanos em uma base de observação via satélite de todo o planeta (LEVIA JR, 2000);
- **Seguro**: identifica grupos de clientes que fazem comunicação de sinistro com alta frequência (YEOH, 2001);
- **Planejamento urbano**: identifica grupos de casas de acordo com tipo, valor e localização geográfica.

9.4 Métodos de clusterização

Três famílias:

- **Particionamento**: constrói várias partições e as avalia usando algum critério;
- **Hierárquico**: cria uma decomposição hierárquica dos objetos usando algum critério;
- **Baseado em densidade**: fundamenta-se em funções de conectividade e de densidade.

9.5 Métodos baseados em particionamento

Dado um valor de k, o objetivo é encontrar k clusters que otimizem um critério de particionamento escolhido. Os principais algoritmos:

- **K-means** (MacQueen, 1967): cada cluster é representado pelo **centro (centroide)** do cluster;
- **K-medoids ou PAM (Partition Around Medoids)** (Kaufman e Rousseeuw, 1987): cada cluster é representado por **um dos objetos** no cluster.

A diferença é sutil mas relevante: o centroide do K-means é um ponto médio que pode não existir na base, enquanto o medoide do PAM é necessariamente um registro real — o que torna o método mais robusto a outliers e mais interpretável.

9.5.1 O algoritmo K-means em cinco passos

1. **Escolher o número de clusters** (no exemplo, K = 2);
2. **Selecionar arbitrariamente K pontos como os centroides iniciais** — importante: **não necessariamente pontos da base de dados**;
3. **Associar cada objeto ao cluster (centroide) mais próximo** (maior similaridade), formando K clusters;
4. **Calcular e realocar o novo centroide de cada cluster** — por exemplo, pela média de cada atributo;
5. **Associar cada objeto ao cluster mais próximo**; **voltar ao passo 4 se algum objeto foi movido de cluster**; **terminar caso contrário**.

Os slides percorrem várias iterações do ciclo entre os passos 4 e 5 até a estabilização, chegando ao modelo final.

```
from sklearn.cluster import KMeans

km = KMeans(n_clusters=5, random_state=42)
rotulos = km.fit_predict(X)
print(km.inertia_)   # WCSS
```

9.5.2 Como determinar o número de clusters: WCSS e Elbow Method

A métrica para avaliar o número de clusters é o **WCSS (Within Cluster Sum of Squares)**: a soma, sobre todos os clusters j e sobre todos os pontos Pi pertencentes ao cluster j, do quadrado da distância entre Pi e o centroide Cj.

O comportamento do WCSS é monotônico e intuitivo:

- com **um único cluster**, o WCSS é muito grande, pois a distância de cada ponto até o centroide é muito grande;
- com **dois clusters**, o WCSS é menor, já que as distâncias entre cada centroide e seus pontos diminuem;
- com **três clusters**, menor ainda.

O limite é claro: o número de clusters pode chegar até o número de dados da base, mas obviamente **essa não é uma abordagem boa**, já que cada ponto seria seu próprio cluster e seu centroide seria igual ao ponto — resultando em **WCSS = 0**.

Como o WCSS sempre cai ao aumentar K, ele não pode ser minimizado diretamente. Daí o **Elbow Method** (método do cotovelo). Os valores tabulados no exemplo:

K	WCSS
2	908.329
3	531.742

K	WCSS
4	368.399
5	222.242
6	186.169

(seguindo com 7: 151.367; 8: 125.059; 9: 113.953; 10: 98.218).

A regra: **usualmente, o número de clusters é definido pela inclinação da reta — quando a melhora no aumento de cluster já não é tão significativa quando comparada com a melhora imediatamente anterior**. Observando a tabela, a queda de K=2 para K=5 é acentuada (de 908 para 222), enquanto de K=5 para K=6 ela cai apenas de 222 para 186. O "cotovelo" está próximo de K = 5.

9.5.3 Variações do K-means

Algumas versões do K-means diferem em: **seleção dos pontos iniciais**, **cálculo da similaridade entre os pontos** e **estratégias para calcular os centroides**.

Para **atributos nominais**, existe o **K-modes** (Huang, 1998), que:

- substitui as **médias** dos clusters por **modas**;
- usa medidas de similaridade apropriadas para atributos nominais;
- usa um método baseado em **frequências** para atualizar as modas dos clusters.

9.6 Clusterização hierárquica

9.6.1 Divisivos e aglomerativos

Duas direções opostas:

- **Métodos divisivos**: partem de todos os registros em um "grande cluster"; esse grande cluster é dividido em dois ou mais clusters menores até que cada cluster tenha somente registros semelhantes;
- **Métodos aglomerativos**: partem de cada registro como um cluster individual; a cada passo, combinam-se clusters com alguma característica comum até que se chegue a um "grande cluster".

O procedimento **aglomerativo (bottom up)** é ilustrado iteração a iteração nos slides: a cada passo, os dois clusters mais próximos são fundidos, com as fusões numeradas de 1 a 11 até restar um único cluster.

9.6.2 Dendrograma e ponto de corte

O resultado é um **dendrograma**: os métodos hierárquicos **decompõem objetos em vários níveis de particionamento aninhados (árvore de clusters)**. Uma clusterização dos objetos é obtida **particionando-se o dendrograma em um nível desejado**; cada componente conectado forma um cluster.

O **AGNES** (aglomerativo) é o exemplo trabalhado: cortando o dendrograma em alturas diferentes, obtêm-se **2 clusters**, **3 clusters** ou **4 clusters** a partir da mesma árvore. Essa é a grande vantagem do método — a decisão sobre K pode ser tomada **depois** do ajuste, inspecionando visualmente a estrutura.

O **DIANA (Divisive Analysis)** é o **inverso do AGNES**: eventualmente cada nó forma um cluster.

9.6.3 K-means versus Hierárquico

Modelo	Prós	Contras
K-means	Simples de entender; trabalha bem em bases pequenas e grandes; rápido e eficiente	É preciso passar o número de clusters como parâmetro
Hierárquico	O número ótimo de clusters pode ser obtido pelo próprio modelo; visualização prática através de dendrograma	Não é apropriado para bases muito grandes

9.7 Estudo de caso: clientes de um shopping

O estudo de caso segmenta clientes de um shopping em função de duas dimensões: **Ganho Anual** e **Traço de Gastos** (spending score). A clusterização produz cinco grupos, e a interpretação de negócio é o ponto alto do exercício:

- **Cluster 1** — ganho alto e baixo gasto: **clientes cuidadosos**;
- **Cluster 2** — ganho médio e médio gasto: **clientes padrão**;
- **Cluster 3** — ganho alto e alto gasto: **clientes alvo**. Deve-se entender melhor os produtos comprados por esses clientes e direcionar melhor as campanhas de marketing;
- **Cluster 4** — ganho baixo e baixo gasto: **clientes sensíveis**;
- **Cluster 5** — ganho baixo e alto gasto: **clientes pouco cuidadosos**.

Esse caso ilustra bem o valor do agrupamento: os rótulos não existiam na base, foram **descobertos** pelo algoritmo e **nomeados** pelo analista com base no conhecimento de domínio.

9.8 Clusterização baseada em densidade: DBSCAN

9.8.1 Conceitos

O **DBSCAN** é um algoritmo baseado em **densidade**, definida como o **número de pontos dentro de um raio específico (Eps)**.

Os três tipos de ponto:

- **core point**: tem um número mínimo de pontos especificado pelo usuário (**MinPts**) dentro do raio **Eps**;
- **border point**: fica localizado na vizinhança de um core point, mas não tem MinPts vizinhos próprios;
- **noise point**: qualquer ponto que não se classifica como core point nem como border point.

A ideia geral: **um cluster é definido como um conjunto máximo de pontos densamente conectados**.

9.8.2 O algoritmo

- Arbitrariamente, **seleciona um ponto p**;
- **Identifica todos os pontos densamente conectados a p** com relação aos parâmetros Eps e MinPts;
- **Se p é um core point, um cluster é formado**;
- **Se p é um border point e não há pontos densamente conectados a p**, o DBSCAN visita o próximo ponto do conjunto de dados;
- **Continua o processo até que todos os pontos tenham sido analisados**.

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.5, min_samples=5)
rotulos = db.fit_predict(X)   # rotulo -1 identifica ruido
```

A grande vantagem do DBSCAN, discutida sob o título "Quando o DBSCAN funciona bem?", é que ele **não exige a especificação do número de clusters**, **encontra clusters de formato arbitrário** (não apenas esféricos, como o K-means) e **identifica ruído explicitamente**. Os slides referenciam Muntaz e Duraiswamy (2010).

10 Regressão

10.1 Correlação e regressão

A **correlação** indica a **força e a direção** do relacionamento entre dois atributos. É uma medida da relação entre dois atributos — mas o alerta é explícito: **correlação não implica causalidade**. Duas variáveis podem estar altamente correlacionadas sem que exista relação de causa e efeito entre elas.

A distinção entre os dois objetivos é precisa:

- na **análise de correlação linear**, o objetivo é determinar o **grau de relacionamento** entre duas variáveis;
- na **análise de regressão linear**, o objetivo é determinar o **modelo que expressa essa relação** (a equação de regressão), ajustada aos dados.

Para que serve a regressão? Pode-se usar o modelo para **predizer o valor de y para um dado valor de x** (ou de um vetor de x), e assim realizar **previsões sobre o comportamento futuro** de algum fenômeno da realidade. Nesse caso, **extrapola-se para o futuro as relações de causa-efeito já observadas no passado** entre as variáveis.

A análise de regressão compreende quatro tipos básicos de modelos: **linear simples**, **linear múltipla**, **não linear simples** e **não linear múltipla**.

10.2 Regressão linear simples

O modelo é $y = b_0 + b_1 x_1$, onde:

- **y** é a **variável dependente**;
- **x1** é a **variável independente**;
- **b1** é o **coeficiente**;
- **b0** é o **intercepto (constante)**.

O exemplo dos slides relaciona experiência e salário: o intercepto está em torno de 30k, e cada **+1 ano de experiência** corresponde a **+10k** no salário previsto. Essa é a leitura direta do coeficiente angular.

10.2.1 Resíduos e mínimos quadrados

Os **resíduos** são as diferenças entre o valor observado Yi e o valor previsto pelo modelo. O critério de ajuste é **minimizar a soma dos quadrados dos resíduos**, isto é, minimizar a soma de $(Y_i - \hat{Y}_i)^2$ sobre as n observações.

Formalmente: uma análise de regressão gera uma equação para descrever a relação entre um ou mais preditores e a variável resposta, e para predizer novas observações com um valor preditor com **precisão maior que o acaso**. A regressão linear geralmente usa o método de estimativa de **mínimos quadrados comum**, que deriva a equação minimizando a soma dos resíduos quadrados.

A regressão fornece a linha que **melhor ajusta** os dados. Essa linha pode ser usada para:

- **examinar como a variável de resposta muda quando o preditor muda**;
- **predizer o valor de uma variável de resposta para qualquer variável preditora**.

10.3 Regressão linear múltipla

A **regressão linear múltipla** examina as relações lineares entre uma **resposta contínua** e **dois ou mais preditores**:

$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_n x_n$

10.4 Métricas de avaliação

Quatro métricas são apresentadas:

- **MAPE**: média, sobre as n observações, do valor absoluto de (real menos previsto) dividido por real. O MAPE **tenta capturar a importância do erro relativo, fornecendo um valor percentual**;
- **RMSE**: raiz da média dos quadrados de (real menos previsto). O RMSE **fornece o erro na dimensão da variável e mede a magnitude média do erro**;
- **R2**: $R^2 = 1 - SS_{res}/SS_{tot}$. Representa a **porcentagem de variação na resposta que é explicada pelo modelo**;
- **R2 ajustado**: $R^2_{aj} = 1 - (1 - R^2) \cdot (n-1)/(n-p-1)$, onde n é o número de amostras e p o número de regressores (variáveis independentes ou atributos).

Quanto mais alto o valor de R2 (ajustado), melhor o modelo ajusta seus dados.

10.4.1 Construção do R2

O R2 é construído a partir de duas somas de quadrados:

- **SSres** — soma dos quadrados dos resíduos: soma de $(Y_i - \hat{Y}_i)^2$;
- **SStot** — soma dos quadrados total: soma de $(Y_i - Y_{avg})^2$, onde Yavg é a média da variável resposta.

E então $R^2 = 1 - SS_{res}/SS_{tot}$. O normal é que esta métrica esteja **entre 0 e 1**. A interpretação: SStot mede a variação total dos dados em torno da média; SSres mede a variação que o modelo **não** conseguiu explicar. O R2 é, portanto, a fração explicada.

10.4.2 Por que o R2 ajustado

Há um problema com o R2 puro. Ao acrescentar um regressor x3 ao modelo, o processo de minimização de SSres nunca piora o ajuste — logo o **R2 pode aumentar mesmo que x3 seja insignificante**. Como o R2 é interpretado como "quão bom é o modelo" (quanto maior, melhor), isso cria um incentivo perverso para inflar o modelo com variáveis inúteis.

O **R2 ajustado** corrige isso penalizando o número de regressores p: o fator (n-1)/(n-p-1) cresce à medida que p aumenta, de modo que uma variável só melhora o R2 ajustado se sua contribuição superar a penalidade.

10.5 Estudo de caso: 50 Startups

O objetivo é criar um **modelo para investidores**. A base contém 50 startups com:

- **4 variáveis independentes (explicativas)**: gastos com P&D, gastos administrativos, gastos com marketing e estado;
- **variável dependente (resposta)**: lucro.

10.5.1 Dummy variables e a dummy variable trap

A variável **Estado** é categórica e precisa ser convertida em **dummy variables** — colunas binárias, uma por categoria.

A armadilha é a **dummy variable trap**: se todas as dummies forem incluídas no modelo, elas se tornam linearmente dependentes. No caso de duas categorias, $D_2 = 1 - D_1$. O resultado é **multicolinearidade**, e **o modelo pode não funcionar de forma apropriada**. A solução prática é omitir uma das dummies (a categoria de referência).

```
import pandas as pd

X = pd.get_dummies(dados, columns=["Estado"], drop_first=True)
```

10.5.2 Lendo a saída da regressão

Os slides detalham a interpretação de cada coluna da tabela de resultados:

- **Coeficiente**: o modelo estima um **aumento esperado de 0.79 no lucro para cada 1 unidade (no caso, 1 dólar) de aumento de gasto com pesquisa e desenvolvimento**, quando as outras variáveis são mantidas constantes;
- **Desvio padrão**: quão precisa foi a estimação do coeficiente da variável em questão. **Quanto menor, mais precisa** é a estimativa;
- **T value**: Estimate dividido por Standard Error — **quantos desvios padrões do zero o coeficiente estimado está**;
- **P-value**: testa a hipótese nula de que o coeficiente é igual a zero (sem efeito). Um **p-value menor que 0.05 indica que você pode rejeitar a hipótese nula**. Em outras palavras, um preditor que tem um p-value pequeno é provavelmente uma boa adição ao seu modelo (**estatisticamente significante**).

Os slides apresentam scripts em R para esse estudo de caso.

10.6 Árvores de Regressão

A referência canônica é **Breiman, Leo, et al. Classification and regression trees. CRC Press, 1984**.

A lógica é análoga à das árvores de classificação, mas o critério de divisão muda: cada split **minimiza a soma do erro quadrático para os lados direito e esquerdo**. Os splits são **escolhidos a partir de valores da base de dados e de forma a minimizar a soma dos erros quadráticos**.

O exemplo dos slides constrói quatro splits sucessivos em um espaço bidimensional — Split 1 em 20, Split 2 em 170, Split 3 em 200, Split 4 em 40 — particionando o plano em regiões retangulares. Em cada folha, a predição é a **média da variável dependente y** dos pontos daquela região. Os valores das folhas no exemplo são 300.5, 65.7, 1023, -64.1 e 0.7.

Isso torna explícita a natureza da árvore de regressão: ela é uma **função constante por partes**, que aproxima superfícies complexas com degraus retangulares.

10.7 Random Forest para regressão

O procedimento é descrito em quatro passos:

1. **Escolhe o número de árvores** que se deseja construir e o espaço de atributos S, e repete os passos 2 e 3 para cada uma das árvores;
2. **Escolhe aleatoriamente K dados e S atributos** do conjunto de treinamento;
3. **Constrói a árvore de decisão associada àqueles K dados**;
4. **Para cada dado novo**, faz com que cada árvore da floresta infira um valor da variável resposta (dependente). A resposta do comitê será, por exemplo, a **média aritmética** da inferência de todas as árvores.

A diferença em relação à versão de classificação está apenas no passo 4: **média aritmética** em vez de **maioria de votos**.

```
from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor(n_estimators=500, random_state=42)
rfr.fit(X_treino, y_treino)
```

10.8 KNN para regressão

O KNN também se adapta à regressão. Enquanto na **classificação** o novo ponto recebe a classe majoritária entre os K vizinhos, na **regressão** ele recebe um valor agregado (tipicamente a média) da variável resposta dos K vizinhos mais próximos. Os slides apresentam a construção visual do método passo a passo.

10.9 Estudo de caso: aluguel de bicicletas

Base de aluguel de bicicletas de 2011 a 2012 (Capital Bikeshare), com **9 variáveis independentes**:

- **Estação do ano** (1: primavera, 2: verão, 3: outono, 4: inverno);
- **Feriado**;
- **Dia de semana**;
- **Dia de trabalho**;
- **Tempo** (1: limpo, 2: nublado, 3: neve ou chuva);
- **Temperatura**;
- **Sensação térmica**;
- **Humidade**;
- **Velocidade do vento**.

A **variável resposta** é a quantidade de horas de uso de bike. Este caso é proposto para prática.

11 Previsão de Séries Temporais

11.1 Definição

Uma **série temporal** é um conjunto de observações **ordenadas no tempo**, **não necessariamente igualmente espaçadas**, que apresentam **dependência serial** — isto é, dependência entre instantes de tempo. É essa dependência serial que distingue o problema de uma regressão comum: as observações não são independentes.

11.2 Tipos de processamento

Três tipos possíveis de processamento sobre uma série temporal:

- **Predição**: futuros valores de x(t);
- **Classificação**: rotular uma série em uma ou algumas classes — por exemplo, "preço vai subir", "preço vai cair", "sem mudanças";
- **Transformação**: de uma série temporal em outra série temporal — por exemplo, do preço do óleo para o consumo de gasolina nos postos.

O exemplo ilustrativo usado é a **cotação EURUSD**.

11.3 Componentes de uma série

Três componentes são identificadas:

Tendência: indica o comportamento **de longo prazo** da série — se ela cresce, decresce ou permanece estável, e qual a **velocidade** dessas mudanças.

Ciclos: são caracterizados pelas oscilações de subida e de queda nas séries, **de forma não periódica**, ao longo da componente de tendência.

Sazonalidade: corresponde às oscilações de subida e de queda que **sempre ocorrem em um determinado período** do ano, do mês, da semana ou do dia.

A **diferença essencial entre as componentes sazonal e cíclica** é que a primeira possui **movimentos facilmente previsíveis, ocorrendo em intervalos regulares de tempo (periódico)**, enquanto os **movimentos cíclicos tendem a ser irregulares**. O exemplo dado de ciclo: uma grande seca seguida de um ano com grande quantidade de chuva.

11.4 Modelos de previsão

Cinco famílias são apresentadas:

- Naive;
- Modelo de média móvel (MA);
- Modelo de amortecimento exponencial;
- Modelo auto-regressivo integrado de média móvel (ARIMA);
- Modelo auto-regressivo não linear.

11.4.1 Naive (ingênuo)

A previsão do valor da série no instante T+1 é **apenas dada pela última observação da série em T**. Uma aplicação clássica é a previsão do **preço de uma ação**. Apesar da simplicidade, o Naive é uma referência de comparação obrigatória: um modelo sofisticado que não supere o Naive não se justifica.

11.4.2 Médias móveis

A previsão é a **média das últimas n observações**. Um dos problemas é a **definição do tamanho da janela**:

- quanto **maior** o valor de n, **mais suave** é a previsão;
- se n é **pequeno**, a previsão **tende a oscilar muito**.

Uma característica importante: **todas as observações têm o mesmo peso**. Mas, na prática, as observações mais recentes tendem a ser mais relevantes — o que leva diretamente aos modelos de amortecimento exponencial.

11.4.3 Amortecimento exponencial

A ideia geral é parecida com a das médias móveis, mas os **pesos das observações decrescem à medida que as observações estão mais longe**. A taxa de decréscimo dos pesos é determinada por **uma ou mais constantes de amortecimento**. A maior dificuldade é justamente a **escolha das constantes de amortecimento**.

11.4.4 ARMA, ARIMA e SARIMA

O **ARMA** trata de processos mistos AR (auto-regressivos) e MA (médias móveis), aplicáveis a **séries estacionárias**. Modelos mais sofisticados incluem séries **não estacionárias (ARIMA)** e **sazonais (SARIMA)**.

11.4.5 Modelos auto-regressivos não lineares

Modelos não lineares são **mais poderosos, mas precisam de mais dados de treinamento e não são tão bem comportados quanto os modelos lineares**. O exemplo é o de **Redes Neurais**. O **pré-processamento dos dados é importante**: pode ser útil **remover a tendência** — referência citada: “Neural network forecasting for seasonal and trend time series”, 2005.

11.5 Definições de projeto

Três definições precisam ser tomadas antes de treinar um modelo de previsão:

- **Janela de entrada**: quantos e quais valores passados da série devem ser utilizados;
- **Horizonte da previsão**: a saída da rede refere-se a quantos passos à frente?
- **Definição de outras variáveis explicativas**: que outras variáveis podem influenciar na previsão? Exemplos: outras séries históricas (índices financeiros, temperatura), dia da semana, hora da previsão, mês da previsão.

11.6 Arquiteturas NAR, NARX e Nonlinear Input-Output

NAR — Nonlinear Autoregressive Model: previsão de séries temporais **com valores da própria série**.

Dentro do NAR, distinguem-se dois regimes de operação:

Processo Multi Step (Closed Loop):

- existe um horizonte da série antes da previsão, e deseja-se prever **n passos à frente**;
- **a própria previsão da rede é utilizada para prever novos valores**;
- **o erro aumenta conforme aumenta-se o número de previsões à frente** — pois o erro se propaga e se acumula.

Processo One Step Ahead (Open Loop):

- existe um horizonte da série antes da previsão, e deseja-se prever **1 passo à frente**;
- cada previsão só é feita quando se tem **todos os valores da série histórica imediatamente anteriores** ao valor a ser previsto;
- **o erro não é propagado** para novas previsões.

NARX — Nonlinear Autoregressive Model with External: previsão de séries temporais **com valores da própria série e valores de outra série**.

Nonlinear Input-Output: previsão de séries temporais **com valores de outra série** apenas. Atenção: **soluções via NARX são mais precisas que essa solução**; só se deve utilizar esta opção quando **não estiverem disponíveis os dados da própria série a ser prevista**.

As **redes recorrentes** são mencionadas e remetidas à disciplina de Redes Neurais.

11.7 Estudo de caso: previsão de carga elétrica

Um sistema preciso de previsão de carga oferece **segurança, confiabilidade e economia** na operação de sistemas de potência.

A arquitetura da solução:

- **4 redes diferentes**, de acordo com o dia da semana;
- **Entradas**: janela de **5 valores passados**; o valor da carga **7 dias antes, no mesmo horário**; e a **codificação binária da hora** a ser prevista;

- **Treinamento**: conjunto de treinamento formado pelos **2 meses anteriores**, com **re-treinamento a cada mês**.

A **topologia da rede**:

- **camada de entrada: 11 atributos** — os 5 últimos valores de carga, a carga no mesmo dia e hora na semana anterior, e a codificação binária do horário;
- **camada escondida: 20 neurônios**;
- **camada de saída: 1 neurônio**.

Note-se como as três definições de projeto (janela de entrada, horizonte e variáveis explicativas adicionais) aparecem materializadas aqui: janela de 5, horizonte de 1 hora, e como variáveis adicionais o dia da semana (via redes separadas), a sazonalidade semanal (carga de 7 dias antes) e a hora do dia (codificação binária).

11.8 Outros estudos de caso

- **Faturamento de varejo**, com os atributos mês, ano e faturamento;
- **Passageiros de companhias aéreas** — série clássica com tendência e sazonalidade evidentes.

```
# esboco de janela deslizante para NAR
import numpy as np

def montar_janelas(serie, janela=5):
    X, y = [], []
    for t in range(janela, len(serie)):
        X.append(serie[t - janela:t])
        y.append(serie[t])
    return np.array(X), np.array(y)
```

12 Deploy (aula extra)

12.1 Machine Learning em Python

Esta aula extra fecha o ciclo do projeto: depois de explorar, tratar e modelar, é preciso **colocar o modelo em produção**. Os slides retomam a taxonomia de Machine Learning e revisam Árvores de Decisão, Comitês e Random Forest antes de tratar do deploy propriamente dito.

A recapitulação reforça os pontos essenciais: árvores de decisão criam modelos de classificação na forma de estruturas, quebrando o conjunto de dados em subconjuntos cada vez menores; são fáceis de entender, funcionam mais eficientemente com atributos discretos e são extremamente rápidas em classificar dados novos. Comitês agregam múltiplos modelos treinados para melhorar a acurácia do modelo conjunto, e o Random Forest combina bootstrap, subespaços aleatórios e votação, com o erro OOB como estimativa de generalização.

12.2 Modelo em produção

As formas de colocar um modelo em produção listadas nos slides são:

- **App desktop ou web**;
- **Executável por linha de comando**;
- **Bat script**;
- **Via código**.

Os arquivos de apoio da aula tornam esse ponto concreto: além dos notebooks de treinamento (`DorLombar.ipynb`) e de inferência (`DorLombar_inference.ipynb`), há `app.py`, `main.py`, `requirements.txt`, `README.md` e `.gitignore` — a estrutura mínima de um projeto empacotado e versionado.

A separação entre **notebook de treinamento** e **notebook de inferência** é o padrão fundamental: o treinamento é executado periodicamente e produz um artefato serializado (o modelo ajustado, junto com os objetos de pré-processamento); a inferência apenas carrega esse artefato e aplica-o a dados novos. Crucialmente, **os mesmos objetos de transformação usados no treino devem ser reutilizados na inferência** — um scaler ou um encoder reajustado sobre os dados de produção produziria resultados inconsistentes.

```
# treinamento: serializa o pipeline completo
import joblib
joblib.dump(pipeline_treinado, "modelo.pkl")

# inferencia: carrega e aplica
modelo = joblib.load("modelo.pkl")
predicoes = modelo.predict(dados_novos)
```

12.3 Estudo de caso: medicamento anti-ansiedade

Dados de experimento sobre os efeitos de medicamentos anti-ansiedade e seu impacto em grupos que têm majoritariamente lembranças felizes ou tristes. As drogas e dosagens:

- **A** — **Alprazolam** (Xanax, longo prazo): 1mg, 3mg, 5mg;
- **T** — **Triazolam** (Halcion, curto prazo): 0,25mg, 0,5mg, 0,75mg;
- **S** — **Sugar Tablet** (placebo): 1 tab, 2 tabs, 3 tabs.

O exercício proposto: carregar a base `Islander_data.csv`; fazer, livremente, **análises gráficas** para entendimento dos dados utilizando a biblioteca **matplotlib**; e treinar um modelo com **árvore de decisão** e **random forest** para prever `Memory_after`.

12.4 Estudo de caso: sintomas de dor lombar

A dor lombar pode ser causada por uma variedade de problemas em qualquer parte da complexa rede interconectada de músculos da coluna vertebral, nervos, ossos, discos ou tendões na coluna lombar. Embora seja extremamente comum, **os sintomas e a gravidade variam muito**: uma simples distensão do músculo lombar pode ser excruciante o suficiente para exigir uma visita à

sala de emergência, enquanto um disco em degeneração pode causar apenas desconforto leve e intermitente.

O conjunto de dados busca identificar se uma pessoa está **anormal ou normal** usando dados coletados da coluna física. São **310 observações e 13 atributos** (12 preditores numéricos e classe binária):

- `Col1` = pelvic incidence;
- `Col2` = pelvic tilt;
- `Col3` = lumbar lordosis angle;
- `Col4` = sacral slope;
- `Col5` = pelvic radius;
- `Col6` = degree spondylolisthesis;
- `Col7` = pelvic slope;
- `Col8` = direct tilt;
- `Col9` = thoracic slope;
- `Col10` = cervical tilt;
- `Col11` = sacrum angle;
- `Col12` = scoliosis slope;
- `Class`: Abnormal ou Normal.

Este é o caso que serve de fio condutor para o deploy, com notebooks separados de treino e inferência e uma aplicação em `app.py`.

12.5 Estudo de caso: doenças do coração

Baseado na **Cleveland Database**, com **13 atributos** (originalmente 76) mais a classe: idade, gênero, máxima frequência cardíaca, tipo de dores no peito, entre outros. São **303 instâncias: 165 doentes e 138 saudáveis**.

13 Trabalho

O **Trabalho Final** é o instrumento de avaliação da disciplina. As regras estabelecidas nos slides de abertura:

1. Será enviado pelo Classroom um **problema proposto pela professora**. Se desejarem, os alunos poderão **propor seus próprios trabalhos**;
2. Cada aluno pode **escolher a ferramenta de sua preferência** para solucionar o problema;
3. O trabalho deverá ser **enviado pelo Classroom em formato de relatório ou apresentação**.

Os arquivos de apoio associados ao trabalho incluem um dicionário de dados (`datadict.pdf`), o enunciado (`Trabalho.pdf`) e as bases `horse.csv` e `horseTest.csv` — a separação entre um conjunto de desenvolvimento e um conjunto de teste indica que a avaliação envolve **generalização para dados não vistos**, e não apenas ajuste sobre a base de treino.

Material de slides não disponível para esta aula.

Ainda assim, o material da disciplina fornece o roteiro completo para a execução do trabalho, que deve seguir o esquema básico de um projeto de Data Mining apresentado desde a primeira aula:

1. **Entendimento do problema** — traduzir a pergunta de negócio em uma das cinco classes de problemas de DM (previsão, classificação, regressão, agrupamento, associação);
2. **Análise exploratória** — classificar as variáveis (categóricas, discretas, contínuas), calcular medidas-resumo e visualizar as distribuições;
3. **Pré-processamento** — tratar valores faltantes (deletar ou imputar), normalizar, converter atributos categóricos, avaliar outliers, verificar o balanceamento das classes e, se necessário, reduzir a dimensionalidade;
4. **Modelagem** — escolher e treinar os algoritmos apropriados, ajustando hiperparâmetros;
5. **Avaliação** — escolher métricas coerentes com o problema e com o balanceamento (matriz de confusão, precisão, recall, F1, Kappa para classificação; MAPE, RMSE, R2 e R2 ajustado para regressão);
6. **Comunicação** — reportar os resultados em relatório ou apresentação, com interpretação de negócio, e não apenas números.

14 Síntese da Disciplina

A disciplina DM251 constrói um percurso completo e coerente. O ponto de partida conceitual é a distinção entre **dado, informação, conhecimento e sabedoria (DIKW)** e a localização da mineração de dados dentro do processo mais amplo de **KDD**. Data Mining não é um algoritmo, mas um **processo interdisciplinar** que combina aprendizagem de máquina, estatística, banco de dados, sistemas especialistas e visualização, com o objetivo declarado de **agregar valor ao empreendimento**.

O eixo organizador é a **taxonomia de Machine Learning**, retomada em praticamente todas as aulas: supervisionado (classificação com rótulo categórico, regressão com rótulo contínuo, previsão de séries com rótulo contínuo dependente do tempo), não supervisionado (agrupamento e associação) e por reforço. Essa taxonomia se conecta diretamente às **cinco classes de problemas de DM** e, através dos estudos de caso, aos **cinco problemas típicos de negócio**.

O bloco de **pré-processamento** é tratado com o peso que merece — duas aulas inteiras. As lições estruturais são: valores faltantes exigem uma decisão consciente entre deletar e imputar; a normalização equaliza pesos e acelera a convergência; a **maldição da dimensionalidade** significa que mais atributos não é melhor, e a redução se dá por **seleção** (filtros por ganho de informação, wrappers por busca com algoritmos genéticos, embarcados como as árvores) ou por **agregação** (PCA); o desbalanceamento produz o **paradoxo da acurácia**, combatido com reamostragem, SMOTE, penalização e, sobretudo, métricas adequadas; outliers distorcem o treinamento.

O bloco de **classificação** percorre SVM (margem máxima, vetores de suporte, soft margin, kernel trick, hiperparâmetros C e Gamma), Árvores de Decisão (ID3, entropia, ganho de informação, regras de decisão), Comitês (bagging, boosting, stacking, RSM), Random Forest (bootstrap, subespaço aleatório, votação, erro OOB), KNN (K vizinhos, distância, estratégias de desempate) e Regressão Logística (sigmoide aplicada à regressão linear, saída como probabilidade, limiar de decisão).

Duas conexões merecem destaque especial. A primeira: o **ganho de informação** aparece duas vezes — como filtro de seleção de atributos no pré-processamento e como critério de divisão do ID3 —, o que explica por que a árvore é o exemplo canônico de seleção **embarcada**. A segunda:

o **compromisso viés-variância** é apresentado como válido "para modelos de Machine Learning em geral", e é ele que dá sentido tanto aos hiperparâmetros do SVM quanto à existência dos comitês, cuja principal vantagem declarada é neutralizar a **instabilidade inerente** aos algoritmos de aprendizagem.

O bloco **não supervisionado** cobre associação (suporte, confiança e lift, com Apriori, FP-Growth e Eclat) e agrupamento (particionamento com K-means e K-medoids, avaliado por WCSS e Elbow Method; hierárquico com AGNES e DIANA, lido em dendrogramas; e densidade com DBSCAN, core, border e noise points). O contraste entre os métodos é instrutivo: o K-means exige K como parâmetro, o hierárquico permite escolher K depois pelo corte do dendrograma, e o DBSCAN dispensa K, encontra formas arbitrárias e identifica ruído explicitamente.

O bloco de **regressão e séries temporais** fecha o repertório supervisionado. A regressão parte da distinção entre correlação e causalidade, formaliza mínimos quadrados, discute o R2 e a necessidade do R2 ajustado, alerta para a dummy variable trap, e estende-se a árvores de regressão, Random Forest (agora com média em vez de votação) e KNN. As séries temporais introduzem a **dependência serial** e as componentes de tendência, ciclo e sazonalidade, percorrendo do Naive ao ARIMA e aos modelos não lineares (NAR, NARX, Nonlinear Input-Output), com a distinção crucial entre **one step ahead** (erro não propagado) e **multi step** (erro acumulado).

Finalmente, a aula de **deploy** lembra que um modelo que não chega à produção não gera valor, e o **Trabalho Final** consolida todo o percurso em um projeto próprio. A lição transversal de toda a disciplina, resumida no slide de encerramento da primeira aula, é: **"Data is Knowledge, and Knowledge is Power."** — mas o caminho do dado ao conhecimento passa por exploração cuidadosa, tratamento rigoroso, modelagem adequada, avaliação honesta e comunicação clara.