Pós-Graduação em Inteligência Artificial Generativa e LLMs

PUC-Rio

Data Mining

Notas de Aula

Vítor Wilher

DM243 · Eletiva · 2024.3

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

A disciplina **Data Mining (DM)** apresenta, de forma estruturada e progressiva, o processo completo de extração de conhecimento a partir de bases de dados. O ponto de partida é conceitual: entender o que é mineração de dados, como ela se insere no processo mais amplo de **KDD (Knowledge Discovery in Databases)** e quais são as classes típicas de problemas que ela resolve no mundo corporativo. A partir daí, o curso percorre cada etapa de um projeto real: análise exploratória, pré-processamento e tratamento dos dados, modelagem (supervisionada e não supervisionada), avaliação e, por fim, colocação do modelo em produção.

O encadeamento das aulas segue exatamente a lógica de um projeto de mineração. As primeiras aulas fixam vocabulário e ferramentas de exploração — tipos de variáveis, medidas resumo e visualização. As aulas seguintes tratam do que costuma consumir a maior parte do tempo em projetos reais: **valores faltantes, normalização, redução de dimensionalidade, seleção de atributos, balanceamento de classes e outliers**. Só então o curso avança para os algoritmos propriamente ditos.

O bloco de modelagem é o mais extenso. Ele começa por **classificação** (SVM, árvores de decisão, comitês, Random Forest, KNN e regressão logística), passa por **associação** (Apriori, FP-Growth e Eclat), por **agrupamento** (K-means, hierárquico e DBSCAN), por **regressão** (linear simples e múltipla, árvores de regressão, Random Forest e KNN para regressão) e termina em **previsão de séries temporais** (Naive, médias móveis, amortecimento exponencial, ARIMA e modelos auto-regressivos não lineares). Cada família de técnicas é ancorada em estudos de caso concretos: crédito bancário, câncer de mama, cesta de compras de supermercado, clientes de shopping, startups, aluguel de bicicletas e previsão de carga elétrica.

A aula final fecha o ciclo tratando de **deploy**: como um modelo treinado sai do notebook e passa a responder a novas observações em um aplicativo, script ou serviço. A avaliação da disciplina é feita por um **trabalho final**, com problema proposto pela professora (ou proposto pelo próprio aluno), com liberdade de ferramenta e entrega em formato de relatório ou apresentação. O material foi elaborado pela professora Manoela Kohler.

2 Introdução e Análise exploratória de Dados

2.1 O que é mineração de dados

A mineração de dados é apresentada como **parte do processo de Descoberta de Conhecimento em Banco de Dados (KDD)**. Seguindo Goebel (1999), o termo KDD designa o processo de transformar dados de baixo nível em conhecimento de alto nível, enquanto a mineração de dados propriamente dita corresponde à **extração de padrões ou modelos a partir de dados observados**.

Um conceito organizador útil é a hierarquia **DIKW (Data-Information-Knowledge-Wisdom)**:

- **Dados**: resultado das experiências diárias de cada área da firma;
- **Informação**: alinhar os 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, portanto, é ir além do simples armazenamento: **associar** para formar nichos, **descrever** para caracterizar padrões e **prever** para identificar tendências e limites. Acima de tudo, o objetivo é **agregar valor ao empreendimento**.

Trata-se de uma **área interdisciplinar**, que combina métodos de aprendizado de máquina, estatística, banco de dados, sistemas especialistas e visualização de dados. É também parte importante de um projeto de Business Intelligence bem sucedido.

2.2 Cinco problemas típicos e cinco classes de problemas

Os slides listam cinco problemas de negócio recorrentes e associam cada um a uma classe de tarefa de mineração:

- **Previsão de faturamento em varejo** — problema de **previsão** (séries temporais). Perguntas do gestor: qual o faturamento esperado para o próximo trimestre? Qual o pior cenário?
- **Avaliação de crédito** — problema de **classificação**. Perguntas: será que o cliente vai pagar o empréstimo? Qual o melhor modelo de financiamento (juros, prazo)?
- **Determinação de localidades promissoras para novas filiais** — problema de **regressão**. Perguntas: qual o local mais promissor? Em quanto tempo o investimento retorna?
- **Definição de grupos de consumo para segmentação de mercado** — problema de **agrupamento**.
- **Oferta de novos serviços e produtos** (Netflix, Amazon e similares) — problema de **associação**. Perguntas: quem observa esse produto tem interesse em ver qual outro? Costuma comprar qual outro?

2.3 O mapa do aprendizado de máquina

O curso adota um mapa que se repete em praticamente todas as aulas e que vale memorizar:

- **Aprendizado supervisionado**: os dados possuem atributos e um **rótulo**. Divide-se em **classificação** (rótulo categórico), **regressão** (rótulo contínuo) e **previsão de séries temporais** (rótulo contínuo e dependente do tempo). O modelo é um **aproximador**: uma função que mapeia entradas em saída.
- **Aprendizado não supervisionado**: há apenas atributos, sem rótulo. Divide-se em **agrupamento** (descoberta de semelhanças e grupos entre registros) e **associação** (descoberta de relações entre variáveis).
- **Aprendizado por reforço**: aprendizado através da interação de agentes com um ambiente.

O exemplo didático usado é o de dados de estudantes: classificar como "Aprovado/Reprovado" é classificação; estimar a nota é regressão; estimar a nota ao longo do tempo, a partir de dados históricos, é previsão de séries.

2.4 Análise exploratória de dados

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

Ela se organiza em três frentes: **organização e classificação dos dados**, **cálculo de medidas resumo** e **visualização do conjunto de dados**.

2.4.1 Tipos de variáveis

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

No exemplo trabalhado em aula, um cadastro com ID, nome, idade, sexo, número de amigos e tempo online é decomposto atributo a atributo nessas categorias.

2.4.2 Medidas resumo

São valores numéricos obtidos a partir de uma amostra que **resumem a informação contida nos dados** e **exibem o comportamento da distribuição da amostra**: valores centrais, valores extremos, dispersão, assimetria, média, mediana, moda. Os slides organizam esse conjunto em **medidas de locação**, **medidas de distribuição de frequências** e **medidas de associação**.

2.5 Bases e recursos citados

A aula introduz a base **Mushroom** (disponível no Kaggle), com descrições de amostras hipotéticas correspondentes a 23 espécies de cogumelos, cada uma identificada como comestível, venenosa ou de consumo desconhecido e não recomendado — a última classe foi combinada com a venenosa. O ponto pedagógico é explícito no próprio guia de origem: **não existe uma regra simples para determinar a comestibilidade de um cogumelo**, o que justifica a abordagem por mineração de dados.

Como fontes de acompanhamento contínuo, são recomendados Kaggle, Medium, KDnuggets, arXiv e Papers with Code.

3 Pré-processamento

3.1 Missing values

Valores faltantes são **muito comuns no mundo real**. As causas típicas incluem atributos novos que surgem com o tempo, conforme a empresa passa a precisar de novas informações, e atributos não preenchidos por falta de obrigatoriedade. Uma observação importante: **verificar se o número de registros com dados faltantes para um atributo específico não justifica, por si só, a remoção do atributo**.

As estratégias apresentadas se organizam em duas grandes decisões: **deletar** (o registro ou o atributo) ou **imputar valor**. A imputação pode ser feita por **estatística** (valor único ou sequência) ou por **machine learning**.

Três exemplos numéricos ilustram a imputação estatística sobre uma pequena tabela com os atributos idade, estado civil, nota e atrito:

- **Substituição pela média**: com as notas 9, 7, 10 e 8 observadas, a média é 8,5, valor usado para preencher a nota ausente.
- **Substituição pela média baseada em outro atributo**: restringindo o cálculo aos registros com o mesmo valor de atrito ("Sim"), a média passa a ser $(9 + 7) / 2 = 8$.
- **Substituição pelo valor mais frequente**: para o atributo categórico estado civil, o valor ausente é preenchido com "Casado", que é a moda da coluna.

3.2 Normalização

Os objetivos da normalização são dois e devem ser lembrados sempre que houver atributos em escalas distintas: **dar aos atributos pesos iguais** e **diminuir o tempo de convergência dos algoritmos**. Em conjunto com a conversão de atributos (por exemplo, transformar categorias em representações numéricas), a normalização compõe a etapa de preparação que antecede a modelagem.

3.3 Redução de dimensionalidade

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

Os objetivos da redução são **diminuir o custo do aprendizado**, **aumentar a precisão do algoritmo** e **gerar modelos compactos e mais fáceis de interpretar**. O alvo é definir um conjunto de atributos **relevantes e não redundantes**.

Existem duas abordagens:

- **Seleção de atributos**: escolha de um subconjunto de atributos relevantes entre os disponíveis. Exemplos: filtros e wrappers.

- **Agregação de atributos**: criação de novos atributos a partir da combinação dos existentes. Exemplo: PCA.

3.3.1 Seleção por filtros e o ganho de informação

Os **métodos de filtro** aplicam uma medida estatística para atribuir uma pontuação a cada atributo. Os atributos são então ranqueados pela pontuação e selecionados para permanecer ou ser removidos.

O **ganho de informação** é o filtro detalhado em aula. O procedimento tem dois passos:

1. Calcula-se a **entropia da classe** — o valor 0 indica dados homogêneos e o valor 1 indica dados igualmente distribuídos.
2. Divide-se a base pelos diferentes atributos, calcula-se a entropia para cada um deles e obtém-se o ganho de informação como a diferença entre a entropia da classe e a entropia após a divisão pelo atributo.

Em outras palavras, **o ganho de informação é baseado na diminuição da entropia depois que a base é subdividida por um atributo**.

3.3.2 Seleção por wrappers

Os **métodos wrapper** tratam a seleção 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 dado é a busca por **algoritmo genético**. Suponha uma base com 5 atributos, A1 a A5. Cada indivíduo é um **cromossomo** de 5 genes, e cada gene assume 0 (sem o respectivo atributo) ou 1 (com o respectivo atributo). Cada indivíduo criado durante a evolução é apresentado ao classificador; a **função de avaliação** pode ser, por exemplo, a acurácia de treinamento. No exemplo dos slides, uma combinação inicial produz acurácia de 30% e o algoritmo genético evolui até uma combinação com acurácia de 93%.

3.3.3 Métodos embarcados

Nos métodos **embarcados (embedded)**, o processo de seleção **faz parte do próprio algoritmo de aprendizado**. O exemplo canônico é a árvore de decisão, que escolhe atributos ao construir os nós.

3.3.4 Agregação de atributos e PCA

A agregação é ilustrada por um caso sem perda de informação: os atributos “massa” e “volume” podem ser combinados no atributo “densidade”, com densidade igual a massa dividida por volume.

O **PCA (Análise de Componentes Principais)** consiste em transformar um conjunto de variáveis originais em outro conjunto de mesma dimensão, denominado **componentes principais**, com três propriedades importantes:

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

O resultado é redução da massa de dados com a menor perda possível de informação. O PCA é **completamente reversível**, o que o torna versátil também para compressão de dados. No exemplo apresentado, um **limiar de variância de 0,95** resulta em **31 componentes principais** a partir de um total de 49 atributos.

4 Pré-processamento - Continuação

4.1 Balanceamento de dados

A grande maioria das bases não tem o mesmo número de instâncias em cada classe; quando a diferença é pequena, não há problema. Em alguns domínios o desbalanceamento é esperado — em bases de transações, a maioria será "Normal" e uma pequena minoria será "Fraudulenta".

O risco é o **paradoxo da acurácia**. Se a classe 1 representa 95% dos dados e a classe 2 apenas 5%, os classificadores ficam "preguiçosos": basta classificar tudo como classe 1 para obter 95% de acurácia — um modelo inútil com métrica excelente.

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 exemplos da classe rara) ou **under-sampling aleatório** (descartar exemplos da classe majoritária).
4. **Gerar amostras sintéticas**. O exemplo é o **SMOTE (Synthetic Minority Over-Sampling Technique)**, método de over-sampling que gera amostras sintéticas da classe rara — em vez de criar cópias — perturbando um atributo por vez por um valor dentro da diferença entre os exemplos vizinhos. Os slides também mencionam o uso de **GANs** nessa linha.
5. **Testar diferentes algoritmos**. Não se deve ter um algoritmo favorito nesse caso.
6. **Penalização**: atribuir custo adicional à classificação errada da classe rara. Com razão entre classes de 5:1, um erro na classe 0 custa 1 e um erro na classe 1 custa 5.

4.2 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 tratamento são citadas:

1. **Análise de valores extremos**;
2. **Métodos de proximidade**, como k-means;
3. **Métodos de projeção**, como mapas de Kohonen.

4.3 Estudos de caso

Dois casos ilustram a etapa de pré-processamento em escala realista:

- **SECOM** (repositório UCI): processo moderno de fabricação de semicondutores monitorado por sensores. Nem todos os sinais são igualmente valiosos — os sinais medidos contêm uma combinação de informação útil, informação irrelevante e ruído. Como engenheiros normalmente coletam muito mais sinais do que o necessário, a seleção de atributos permite identificar os sinais mais relevantes e determinar os fatores-chave do processo, com potencial de redução de tempo e custo de produção. A base tem **1567 exemplos e 591 atributos**.
- **IBM Analytics Employee Attrition and Performance**: conjunto de dados fictício criado por cientistas de dados da IBM para descobrir os fatores que levam ao desgaste dos funcionários. Possui **34 atributos** — entre eles `Education`, `EnvironmentSatisfaction`, `JobInvolvement`, `JobSatisfaction`, `PerformanceRating`, `RelationshipSatisfaction`, `WorkLifeBalance`, além de gênero, estado civil, renda mensal, idade, distância de casa, horas extras e cargo — e **2 classes** na variável `Attrition` (Yes / No). Vários atributos são escalas ordinais codificadas numericamente, por exemplo `Education` de 1 ("Below College") a 5 ("Doctor").

5 Classificação

5.1 Definição e aplicações

Classificação é **treinamento supervisionado**: os dados da base possuem indicação da classe a que pertencem. As aplicações citadas incluem **reconhecimento de padrões** (pessoas, voz, doenças vocais, dígitos), contratos em negociação, **detecção de fraude** (energia elétrica, cartão de crédito) e **análise de crédito**.

5.2 Support Vector Machine (SVM)

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

A pergunta central é: dado um conjunto de pontos de duas classes, **como separá-los?** Reta, plano ou hiperplano — e **qual o hiperplano ótimo?** A resposta intuitiva seria "o de menor erro de classificação", mas o SVM vai além: busca o **hiperplano de margem máxima**, isto é, aquele que maximiza a distância entre a fronteira e os pontos mais próximos de cada classe. Os dois objetivos combinados são **classificar corretamente todos os dados de treinamento** e **maximizar a margem, minimizando o erro**.

Os pontos que tocam a margem são os **vetores de suporte**. A consequência prática é notável: **somente os vetores de suporte importam** — eles definem o hiperplano separador, e os demais dados do conjunto de treinamento não têm importância depois que o modelo é criado.

5.2.1 Soft margin e o truque do kernel

E se os dados não forem linearmente separáveis? Duas respostas complementares:

- **Variáveis de folga**: podem ser incluídas para permitir erro de classificação de exemplos difíceis ou com ruído. É a chamada **soft margin**.
- **Mapeamento para dimensão maior**: pelo **Teorema de Cover**, o espaço de atributos original pode — com alta probabilidade — ser mapeado para um espaço de atributos maior, no qual os dados podem ser separados linearmente. Formalmente, aplica-se uma transformação que leva x em $\varphi(x)$.

O **truque do kernel (kernel trick)** implementa essa ideia. A forma mais simples de separar grupos é com uma reta ou hiperplano, mas há situações em que isso não é possível ou em que uma região não linear separa os grupos de forma mais eficiente. O SVM usa uma **função kernel não linear** para mapear os dados em um espaço de atributos maior, de modo que **uma função não linear é aprendida por uma máquina de aprendizado linear** nesse espaço ampliado.

5.2.2 Hiperparâmetros e o compromisso viés-variância

Dois hiperparâmetros de regularização são destacados:

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

Isso conecta o SVM ao **compromisso viés-variância**, válido para modelos de machine learning em geral: **viés (bias)** é a incapacidade do modelo de capturar o verdadeiro relacionamento entre os dados; **variância** é a diferença no resultado do modelo para diferentes conjuntos de dados.

As vantagens do SVM listadas em aula: lida bem com grandes conjuntos de exemplos, trata bem dados de alta dimensão e o **processo de classificação é rápido**.

O estudo de caso é a **análise de crédito bancário**: base com **2077 exemplos** de créditos concedidos que foram pagos ou não, com **11 atributos de entrada** e **2 classes de saída** — a saída indica se o cliente pagou o empréstimo (valor 1) ou não pagou (valor 0).

5.3 Árvores de decisão

Árvores de decisão criam modelos de classificação na forma de estruturas hierárquicas. Quebra-se o conjunto de dados em subconjuntos cada vez menores enquanto se constroem, simultaneamente, as árvores associadas. O resultado final é uma árvore com **nós de decisão** e **nós folha**.

Elementos estruturais:

- uma árvore possui um ou mais **ramos**;
- os **nós folha** 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**;
- **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**.

A filosofia é "dividir para conquistar", e as árvores são **fáceis de implementar e interpretar**.

5.3.1 O algoritmo ID3

O **ID3**, de J. R. Quinlan, realiza uma **busca gulosa de cima para baixo** pelo espaço de possíveis ramos, **sem backtracking**. Como consequência, **pode ficar preso em um ótimo local**, e é **difícil de usar em variáveis contínuas**.

O critério de escolha é a **entropia**, usada para medir a homogeneidade dos dados: se os dados são completamente homogêneos, a entropia é zero; se estão divididos igualmente, a entropia é 1. O procedimento é:

1. Calcula-se a entropia para a classe.
2. Divide-se a base pelos diferentes atributos, calcula-se a entropia de cada um e o **ganho de informação** correspondente (entropia da classe menos entropia após a divisão pelo atributo).
3. Escolhe-se como nó de decisão o atributo de **maior ganho de informação**. Um ramo com entropia 0 vira folha; um ramo com entropia maior que 0 precisa ser subdividido.
4. O algoritmo roda **recursivamente** para todos os ramos sem folha, até que todos os dados sejam classificados.

O produto final são **regras de decisão** legíveis.

5.3.2 Exemplo trabalhado: fraude fiscal

Os slides trabalham uma base clássica com dez registros e os atributos `Refund` (Yes/No), `Marital Status` (Single, Married, Divorced), `Taxable Income` e a classe `Cheat` (Yes/No). A árvore resultante tem `Refund` como raiz; no ramo "No" testa-se `MarSt`; no ramo "Single, Divorced" testa-se `TaxInc` com corte em 80K.

A inferência de um novo registro (`Refund` = No, `Marital Status` = Married, `Taxable Income` = 80K) percorre a árvore a partir do nó raiz: `Refund` igual a "No" leva ao nó `MarSt`; "Married" leva diretamente a uma folha. O **valor inferido para `Cheat` é "No"**.

Resumo das características: árvores de decisão são fáceis de entender, funcionam **mais eficientemente com atributos discretos** e são **extremamente rápidas para classificar dados novos**.

6 Classificação - Continuação

6.1 Comitês (ensembles)

Um **comitê** agrega múltiplos modelos treinados com o objetivo de melhorar o desempenho do modelo conjunto. A intuição é direta: simula o que fazemos quando combinamos o conhecimento de vários especialistas em um processo de tomada de decisão.

A técnica aparece na literatura sob muitos nomes: comitês especialistas, sistemas múltiplos de classificação, comitê de classificadores, máquina de comitê, mistura de especialistas, aprendizado em conjunto e **ensemble**. Diversos estudos demonstram seu uso bem-sucedido em problemas nos quais um único especialista não funciona bem.

6.1.1 Regras de combinação

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

6.1.2 Motivação e quando usar

A questão que motiva o aprendizado de comitês é: às vezes cada técnica de aprendizado retorna hipóteses (funções) diferentes, mas nenhuma hipótese perfeita — **poderíamos combinar várias hipóteses imperfeitas para obter uma hipótese melhor?**

As analogias usadas em aula: eleições combinam votos de eleitores para escolher um bom candidato; comitês combinam opiniões de especialistas; estudantes trabalham em conjunto em um projeto. A intuição subjacente: **indivíduos cometem erros, mas a maioria é menos propensa a errar**, e indivíduos em geral têm conhecimento parcial, de modo que o comitê pode juntar conhecimento.

Quando usar: quando há um **conjunto muito grande de dados**, quando a **região de domínio do problema é muito complexa**, e quando se deseja **melhorar os resultados de classificadores individuais**.

Vantagens: a combinação de modelos pode apresentar melhor desempenho que um modelo isolado, e 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**; e o **custo é alto**.

6.1.3 Bagging, Boosting, Stacking e RSM

- **Bagging**: combina os modelos por **média ou maioria de votos**. Cada modelo é treinado sobre uma reamostragem (bootstrap) da base.
- **Boosting**: **observações classificadas incorretamente recebem maiores pesos**, de modo que os modelos seguintes se concentram nos casos difíceis. A combinação é por **média ponderada**, na qual classificadores com resultados melhores têm pesos maiores. Durante o treinamento, o erro é guardado, e existe uma condição para determinar se um modelo será utilizado ou descartado. Exemplos de algoritmos: **AdaBoost, LPBoost, XGBoost, GradientBoost, BrownBoost**.
- **Stacking**: empilhamento de modelos, no qual a saída de um conjunto de modelos-base alimenta um modelo agregador.
- **RSM (Random Subspace Method)**: 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.

6.2 Random Forest

A **Random Forest** é criada a partir de árvores de decisão individuais cujos parâmetros podem variar aleatoriamente. Ela combina, num único algoritmo, **bagging (bootstrap)**, **random subspace method** e **maioria de votos**.

O workflow descrito em aula:

- cada árvore é construída usando uma inicialização (**bootstrap**) diferente do dataset original;
- aproximadamente **um terço dos casos** fica de fora da amostra de inicialização e não é usado na construção daquela árvore;
- cada caso deixado de fora é apresentado a cada árvore 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 alvo em relação ao total de votos é o **erro OOB (Out-Of-Bag estimate)**.

O OOB é, portanto, uma estimativa de erro obtida "de graça", sem necessidade de um conjunto de validação separado.

6.3 Estudo de caso: Netflix Prize

O **Netflix Prize** ofereceu 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. A tarefa era de aprendizado supervisionado: os dados de treinamento eram formados por um conjunto de usuários e as avaliações dos filmes (1 a 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.

A lição relevante para esta aula: **os melhores times combinaram diversos modelos e algoritmos em um comitê**.

7 Classificação - Parte 2

7.1 K Nearest Neighbors (KNN)

O KNN classifica um novo padrão pela **classe majoritária entre os K vizinhos mais próximos**. O algoritmo tem cinco passos:

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 mais próximos** do novo registro.
4. Entre esses K vizinhos, **contar o número de vizinhos em cada classe**.
5. Atribuir ao novo registro a **classe majoritária** entre os vizinhos mais próximos.

No exemplo dos slides, com K = 5 e distância euclidiana, os cinco vizinhos mais próximos se distribuem em 3 de uma classe e 2 de outra, e o novo registro recebe a classe com 3 vizinhos.

7.1.1 Três pendências práticas

O método deixa três decisões em aberto:

- **Qual tipo de distância usar?** Diferentes métricas de distância produzem diferentes vizinhanças.
- **Qual valor de K?** A resposta dada em aula é direta: **escolha experimental**.
- **Como desempatar?** Três estratégias são apresentadas, ilustradas com K = 4 e empate de 2 a 2:
  - **Escolha aleatória**: "jogue uma moeda honesta" — cara escolhe a classe vermelha, coroa escolhe a azul.
  - **Escolha aleatória ponderada**: "jogue uma moeda desonesta" — dê mais chance à classe que possui mais padrões.
  - **Classe mais próxima**: selecione a classe cuja distância é menor.

Como o KNN depende de distâncias, ele é particularmente sensível à escala dos atributos, o que reforça a importância da normalização vista no bloco de pré-processamento.

7.2 Regressão logística

A construção conceitual parte da **regressão linear simples**, dada por $y = b_0 + b_1 x_1$. O problema é que, ao modelar um fenômeno binário, a reta produz valores fora do intervalo interpretável como probabilidade — "esses pedaços não fazem mais sentido" quando pensamos em probabilidades.

A solução é **aplicar a função sigmoidal** ao resultado linear:

$$f(a) = \frac{1}{1 + e^{-a}}$$

de modo que o modelo final se torna

$$y' = f(y) = \frac{1}{1 + e^{-(b_0 + b_1 x_1)}}$$

O resultado é uma curva em S que mapeia qualquer valor real no intervalo entre 0 e 1, interpretável como **probabilidade**. No exemplo apresentado, com idade no eixo horizontal, o modelo estima probabilidades de **0,7% aos 20 anos, 23% aos 30, 85% aos 40 e 99,4% aos 50**.

Para a **inferência**, aplica-se um limiar — tipicamente **0,5** — sobre a probabilidade estimada para decidir a classe.

7.3 Estudo de caso: câncer de mama

A base vem da **University of Wisconsin, Clinical Sciences Center**, e contém **30 atributos mais classe e 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**.

São **569 instâncias**, sendo **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), com o arquivo `breastCancer.csv`.
2. Fazer um **pós-processamento conservador**: somente inferências com **80% de probabilidade** serão classificadas como benigno. Este é um ponto importante — em domínios com custo assimétrico de erro, o limiar de decisão não deve ser 0,5 por padrão.
3. Repetir o exercício para a base `breastCancer_3classes.csv`, que acrescenta a classe "Suspeito".

Em Python, o esqueleto do modelo KNN seria:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

modelo = KNeighborsClassifier(n_neighbors=5)
modelo.fit(X_train, y_train)

# pos-processamento conservador: so classifica como benigno com prob >= 0.80
probs = modelo.predict_proba(X_test)
```

Os arquivos de apoio da aula incluem notebooks de KNN e de regressão logística para essa base, além de uma versão com otimização de hiperparâmetros via Optuna.

8 Associação

8.1 O problema de associação

A associação pertence ao **aprendizado não supervisionado** e busca a **descoberta de relações entre variáveis**. O caso emblemático é a análise de cesta de compras: quais itens tendem a ser comprados juntos.

8.2 Conceitos e métricas

Uma **regra de associação** tem a forma "se A então B", com A chamado de **antecedente (premissa)** e B de **consequente**. Três métricas governam a avaliação de regras:

- **Suporte**: frequência relativa com que o conjunto de itens aparece nas transações.
- **Confiança**: dada por

$$\text{Confiança}(A \to B) = \frac{\text{Suporte}(A \cap B)}{\text{Suporte}(A)}$$

- **Lift**: dado por

$$\text{Lift}(A \to B) = \frac{\text{Confiança}(A \to B)}{\text{Suporte}(B)} = \frac{\text{Suporte}(A \cap B)}{\text{Suporte}(A) \cdot \text{Suporte}(B)}$$

O lift é uma **medida de melhora na recomendação considerando conhecimento a priori**. Note que a fórmula do lift **independe da ordem** — o lift de A para B é igual ao de B para A. O que muda entre as duas direções é **o suporte da premissa e a confiança da regra**.

8.2.1 O exemplo da Netflix

Um exemplo numérico esclarece o significado do lift. Considere 100 usuários da Netflix: **10 gostaram de "Breaking Bad"**, **40 gostaram de "Dexter"** e **7 pessoas que gostaram de "Breaking Bad" também gostaram de "Dexter"**.

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

- Se a Netflix recomendasse "Breaking Bad" aleatoriamente, a probabilidade da pessoa gostar seria de **10%** (o suporte do consequente).
- Usando o conhecimento a priori de que quem gostou de "Dexter" tende a gostar de "Breaking Bad", a probabilidade sobe para **17,5%**.
- O lift indica uma **melhora de 75%** ao usar o conhecimento a priori.

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

- Recomendação aleatória de "Dexter": probabilidade de **40%**, ou seja, suporte de 0,4.
- Com conhecimento a priori: 7 de 10, ou seja, **confiança de 0,7**.
- Lift igual a 0,7 dividido por 0,4, isto é, **1,75** — os mesmos **75% de melhora**.

8.2.2 Dicas de calibração dos parâmetros

- **Support**: parta do objetivo de negócio. Para otimizar a venda de produtos comprados pelo menos 5 vezes ao dia, em uma semana isso dá 5 vezes $7 = 35$ ocorrências; com 7501 transações, o suporte mínimo fica em aproximadamente **0,005**.
- **Confidence**: confiança **baixa** gera regras que não fazem sentido; confiança **alta** gera regras óbvias. A recomendação é **começar com o valor padrão de 0,8 e ir diminuindo por tentativa e erro**. Um valor de 0,8 significa que todas as regras geradas devem estar corretas em 80% das transações.
- **Lift**: **ordenar as regras pelo lift em ordem decrescente**.

8.3 Preparação da base

A base deve ser **transformada em uma matriz esparsa** para que possa ser apresentada ao algoritmo de associação. Uma lista de transações no formato "id, itens" — por exemplo, transação 1 com Ovo, Leite e Manteiga; transação 2 apenas com Manteiga — vira uma tabela em que cada produto é uma coluna binária (1 se presente na transação, 0 caso contrário).

8.4 Algoritmo Apriori

O Apriori é o algoritmo clássico de descoberta de itens frequentes, trabalhado em aula por meio de um exemplo passo a passo em que conjuntos candidatos são gerados e podados de acordo com o suporte mínimo, e novas regras são derivadas dos conjuntos frequentes sobreviventes a cada nível.

8.5 Algoritmo FP-Growth

O **FP-Growth (Frequent Pattern Growth)** 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**;
- exige **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 do tipo divide-and-conquer** para obter os conjuntos de itens frequentes.

8.5.1 Construção da FP-tree

A construção começa por **ordenar os itens por prioridade**, que é dada pela frequência. No exemplo dos slides, com dez transações, as frequências são A com 7, B com 8, C com 7, D com 4 e E com 3, resultando nas prioridades 1 a 5 para A, B, C, D e E respectivamente. Cada transação é então reescrita com os itens nessa ordem — por exemplo, a transação {B,A} vira {A,B} e a transação {E,D,C,A} vira {A,C,D,E}.

A árvore é construída incrementalmente a partir de um nó `null`. Após ler a transação 1 ({A,B}), tem-se o caminho A:1 seguido de B:1. Após a transação 2 ({B,C,D}), acrescenta-se um novo ramo.

Ao final das dez transações, o nó A acumula contagem 7, com ramos B:5 e C:1, e assim por diante. **Apontadores** (representados em vermelho nos slides) ligam ocorrências do mesmo item em ramos distintos e são usados para facilitar a geração dos termos frequentes; uma **header table** registra item e contagem total.

8.5.2 Extração dos padrões frequentes

Com a árvore construída, definem-se os padrões frequentes assumindo **suporte mínimo igual a 2**. O procedimento percorre os itens em **ordem inversa da frequência** — começa por E, depois D, e assim por diante.

Para **E**, a **conditional pattern base** é P = {(A,C,D:1), (A,D:1), (B,C:1)}. A **conditional FP-tree** correspondente é {(A:2, C:1, D:2), (B:1, C:1)} e os **padrões frequentes** extraídos são {(A,E:2), (D,E:2), (A,D,E:2)}.

Para **D**, a conditional pattern base é P = {(A,B,C:1), (A,B:1), (A,C:1), (A:1), (B,C:1)}; a conditional FP-tree é {(A:4, B:2, C:2), (B:1, C:1)}; e os padrões frequentes são {(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 repetido para todos os itens e todas as conditional trees, em ordem decrescente de frequência.

8.6 Eclat

O **Eclat** é o terceiro algoritmo do bloco. Uma observação importante o distingue dos anteriores: **a saída do Eclat não fornece regras, mas sim a lista dos itens mais frequentemente comprados juntos**. Ou seja, entrega conjuntos frequentes, não implicações com antecedente e consequente.

8.7 Estudo de caso: transações de supermercado

O caso trabalhado é uma lista de transações de 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 ao longo de uma semana.

Uma dica de leitura dos resultados: **"mineral water", "eggs" e "spaghetti" são os três itens mais comprados** no supermercado — o que ajuda a distinguir regras triviais (impulsionadas apenas pela popularidade do consequente) de descobertas genuínas. Justamente por isso, ordenar por confiança tende a devolver regras óbvias, ao passo que ordenar por lift revela **associações nada óbvias**.

Os benefícios de negócio esperados: perceber associações e implementar estratégias de oferta desses produtos, perceber associações não óbvias, **automatizar um sistema de recomendação**, aumentar vendas e oferecer produtos condizentes com o perfil do cliente.

9 Agrupamento

9.1 Conceitos fundamentais

Um **cluster** é uma coleção de objetos **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**.

Um alerta conceitual atravessa toda a aula: **a noção de cluster pode ser ambígua** — o mesmo conjunto de pontos admite mais de um agrupamento razoável, dependendo do critério adotado.

Na classificação não supervisionada, **não conhecemos o padrão nem o número total de grupos** a serem encontrados. O conjunto de dados é particionado em grupos com base em características específicas, de modo que os pontos dentro de um grupo sejam mais similares entre si do que em relação a pontos de outros grupos.

9.1.1 O que é uma boa clusterização?

Uma boa clusterização sempre produz clusters com **alta similaridade intraclasse** e **baixa similaridade entre classes**. A qualidade dos resultados depende de dois fatores: a **medida de similaridade usada** e o **método e sua implementação**.

9.1.2 Aplicações

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

9.1.3 Métodos de clusterização

- **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.

9.2 Métodos baseados em particionamento

Dado um valor de k, o objetivo é encontrar k clusters que otimizem um critério de particionamento escolhido. Os dois 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** do cluster.

9.2.1 O algoritmo K-means passo a passo

1. **Escolher o número de clusters** K. No exemplo dos slides, K = 2.
2. **Selecionar arbitrariamente K pontos como centroides iniciais** — não necessariamente pontos da base de dados.
3. **Associar cada objeto ao cluster (centroide) mais próximo**, isto é, de maior similaridade, formando K clusters.
4. **Calcular e realocar o novo centroide de cada cluster** — por exemplo, a média de cada atributo dos pontos do cluster.
5. **Associar cada objeto ao cluster mais próximo**. Se algum objeto mudou de cluster, voltar ao passo 4; caso contrário, terminar.

O modelo final é o conjunto de centroides e a atribuição de cada ponto ao seu cluster.

9.2.2 Escolha do número de clusters: WCSS e Elbow Method

A métrica usada para avaliar o número de clusters é o **WCSS (Within Cluster Sum of Squares)**: a soma, sobre todos os clusters e sobre todos os pontos de cada cluster, do quadrado da distância entre o ponto e o centroide do seu cluster.

Intuitivamente, com um único cluster o WCSS é muito grande, pois a distância de cada ponto ao centroide é grande. Com dois clusters o WCSS diminui, com três diminui ainda mais, e assim sucessivamente. O limite é evidente: **o número de clusters pode chegar ao número de registros da base**, situação em que cada ponto é seu próprio cluster, o centroide coincide com o ponto e **o WCSS é igual a zero** — o que obviamente não é uma boa abordagem.

O **Elbow Method** resolve esse impasse. Traça-se o WCSS em função de K e procura-se o "cotovelo" da curva. Os valores apresentados em aula:

K	WCSS	K	WCSS
2	908,329	7	151,367
3	531,742	8	125,059
4	368,399	9	113,953
5	222,242	10	98,218
6	186,169		

Usualmente, o número de clusters é definido pela **inclinação da reta**: escolhe-se o ponto a partir do qual a melhora obtida ao aumentar o número de clusters já não é tão significativa quando comparada à melhora imediatamente anterior.

9.2.3 Variações do K-means

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

Para **atributos nominais**, usa-se o **K-modes** (Huang, 1998), que **substitui as médias dos clusters por modas**, usa medidas de similaridade próprias para atributos nominais e emprega um método baseado em frequências para atualizar as modas.

9.3 Clusterização hierárquica

Há duas famílias:

- **Métodos divisivos**: todos os registros formam inicialmente um único "grande cluster", que é dividido em dois ou mais clusters menores até que cada cluster contenha somente registros semelhantes.
- **Métodos aglomerativos**: cada registro começa como um cluster e, a cada passo, combinam-se clusters com alguma característica comum até se chegar a um único grande cluster.

O processo aglomerativo (**bottom up**) é ilustrado iteração a iteração: na primeira iteração os dois pontos mais próximos se unem; a cada nova iteração, mais uma fusão ocorre, até que todos os pontos estejam em um único grupo.

9.3.1 Dendrograma, AGNES e DIANA

O **AGNES** decompõe os objetos em vários níveis de particionamento aninhados, formando uma árvore de clusters conhecida como **dendrograma**. Uma clusterização dos objetos é obtida **particionando-se o dendrograma em um nível desejado**: cada componente conectado forma um cluster. Cortando o dendrograma em alturas diferentes obtêm-se 2, 3 ou 4 clusters, conforme o **ponto de corte** escolhido.

O **DIANA (Divisive Analysis)** faz o procedimento inverso ao do AGNES, partindo do cluster único e dividindo sucessivamente até que, eventualmente, cada nó forme um cluster.

9.3.2 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 pelo dendrograma	Não é apropriado para bases muito grandes

9.4 Estudo de caso: clientes de um shopping

O caso usa dois atributos — **ganho anual** e **traço de gastos** — e produz cinco clusters com leitura de negócio direta:

- **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**.

Este é o exemplo mais claro de como o agrupamento, sem qualquer rótulo prévio, gera segmentação acionável.

9.5 Clusterização baseada em densidade: DBSCAN

O **DBSCAN** define **densidade como o número de pontos dentro de um raio específico (Eps)**. A ideia central: **um cluster é definido como um conjunto máximo de pontos densamente conectados**.

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 nem como core point nem como border point.

O algoritmo procede assim:

- seleciona arbitrariamente um ponto p;
- identifica todos os pontos densamente conectados a p em relação a 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;
- continua até que todos os pontos tenham sido analisados.

A grande vantagem do DBSCAN é que ele **não exige o número de clusters como entrada** e identifica naturalmente **ruído** — o que o K-means não faz, já que atribui todo ponto a algum cluster.

10 DM MEAD - Regressão

10.1 Correlação e regressão

A **correlação** indica a **força e a direção do relacionamento entre dois atributos**. Um alerta essencial: **correlação não implica causalidade** — duas variáveis podem estar altamente correlacionadas sem que exista relação de causa e efeito entre elas.

A distinção entre as duas análises 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? Para **predizer o valor de y para um dado valor de x** e para realizar previsões sobre o comportamento futuro de um fenômeno. Nesse caso, **extrapola-se para o futuro as relações de causa e efeito já observadas no passado**.

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

$$y = b_0 + b_1 x_1$$

no qual y é a **variável dependente**, x_1 é a **variável independente**, b_1 é o **coeficiente** e b_0 é o **intercepto (constante)**.

A leitura do coeficiente é direta e vale para todos os modelos lineares. No exemplo de salário em função de experiência, o intercepto é 30k e o coeficiente indica que **um ano adicional de experiência corresponde a mais 10k**.

10.2.1 Resíduos e mínimos quadrados

Os **resíduos** são as diferenças entre o valor observado e o valor previsto. O ajuste da reta consiste em **minimizar a soma dos quadrados dos resíduos**.

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

A reta ajustada pode ser usada para **examinar como a variável resposta muda quando o preditor muda** e para **predizer o valor da resposta para qualquer valor do preditor**.

10.3 Regressão linear múltipla

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

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$$

10.4 Métricas de avaliação

Quatro métricas são apresentadas:

- **MAPE**: média do erro absoluto relativo, isto é, a média de |real − previsto| dividido por real. **Tenta capturar a importância do erro relativo, fornecendo um valor percentual**.
- **RMSE**: raiz da média dos quadrados de (real − previsto). **Fornece o erro na dimensão da variável e mede a magnitude média do erro**.
- **R quadrado**: dado por

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

onde SS_{res} é a **soma dos quadrados dos resíduos** e SS_{tot} é a **soma dos quadrados totais**, calculada em relação à média de y. O R quadrado **representa a porcentagem de variação na resposta que é explicada pelo modelo**; o normal é que essa métrica esteja entre 0 e 1, e quanto mais alta, melhor o ajuste.

- **R quadrado ajustado**: dado por

$$R^2_{aj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

com n igual ao número de amostras e p igual ao número de regressores. Ele existe para resolver um problema concreto: ao acrescentar uma variável x_3 ao modelo, mesmo que ela seja **insignificante**, o R^2 pode aumentar. O ajuste penaliza o número de regressores e evita que a métrica premie modelos inflados.

10.5 Dummy variables e a armadilha das dummies

Variáveis categóricas precisam ser convertidas em **variáveis dummy** (indicadoras binárias) para entrar no modelo. Existe, porém, a **dummy variable trap**: se uma categoria com dois níveis for representada por duas dummies, tem-se $D_2 = 1 - D_1$, ou seja, uma é combinação exata da outra. O resultado é **multicolinearidade**, e **o modelo pode não funcionar de forma apropriada**. A solução prática é omitir uma das categorias, usando-a como referência.

10.6 Interpretação da saída de uma regressão

Os slides detalham como ler cada coluna da tabela de resultados, usando como exemplo o gasto com pesquisa e desenvolvimento:

- **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 P&D, **quando as outras variáveis são mantidas constantes**.
- **Desvio padrão**: mede **quão precisa foi a estimação do coeficiente**. Quanto menor, mais precisa é a estimativa.
- **T value**: razão entre a estimativa e o erro padrão. Indica **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 se pode rejeitar a hipótese nula. Em outras palavras, um preditor com p-value pequeno é provavelmente uma boa adição ao modelo, sendo estatisticamente significante.

10.7 Árvores de regressão

As árvores de regressão têm como referência clássica Breiman, Leo, et al., *Classification and regression trees*, CRC Press, 1984.

O princípio de construção é análogo ao das árvores de classificação, mas o critério muda: os **splits são escolhidos a partir de valores da base de dados e de forma a minimizar a soma dos erros quadráticos** dos lados direito e esquerdo da divisão.

O exemplo dos slides constrói quatro splits sucessivos em um espaço bidimensional — o primeiro em 20, o segundo em 170, o terceiro em 200 e o quarto em 40 — particionando o plano em regiões retangulares. **Em cada folha, o valor predito é a média da variável dependente y** dos pontos daquela região: no exemplo, os valores 300,5; 65,7; 1023; $-64{,}1$ e 0,7.

Essa é a diferença essencial em relação à árvore de classificação: a folha guarda uma **média**, não uma classe majoritária.

10.8 Random Forest para regressão

O procedimento tem quatro passos:

1. Escolher o **número de árvores** a construir e o espaço de atributos S, repetindo os passos 2 e 3 para cada árvore.
2. Escolher **aleatoriamente K dados e S atributos** do conjunto de treinamento.
3. **Construir a árvore de decisão** associada àqueles K dados.
4. Para cada dado novo, fazer com que **cada árvore da floresta infira um valor** da variável resposta. A resposta do comitê será, por exemplo, a **média aritmética** das inferências de todas as árvores.

A diferença em relação ao caso de classificação está no passo 4: em vez de maioria de votos, usa-se a média das predições.

10.9 KNN para regressão

O KNN também se adapta à regressão. Em vez de atribuir a classe majoritária entre os K vizinhos mais próximos, o modelo devolve um **valor numérico agregado** a partir dos valores da variável resposta desses vizinhos.

10.10 Estudos de caso

Startups — regressão com o objetivo de criar um **modelo para investidores**. A base tem 50 startups, com **4 variáveis independentes**: gastos com P&D, gastos administrativos, gastos com marketing e estado (variável categórica, que exige tratamento por dummies). A **variável dependente** é o lucro.

Aluguel de bicicletas (2011-2012) — base do serviço Capital Bikeshare, com **9 variáveis independentes**: estação do ano (1 primavera, 2 verão, 3 outono, 4 inverno), feriado, dia da semana, dia de trabalho, tempo (1 limpo, 2 nublado, 3 neve ou chuva), temperatura, sensação térmica, umidade e velocidade do vento. A **variável resposta** são as horas de uso de bicicleta.

11 Previsão de Séries Temporais

11.1 Definição e componentes

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 que impede tratar séries temporais como uma tabela comum de registros independentes.

Os tipos possíveis de processamento de uma série são três:

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

11.1.1 Tendência, ciclos e sazonalidade

- **Tendência**: indica o comportamento **de longo prazo** da série, isto é, se ela cresce, decresce ou permanece estável, e qual a velocidade dessas mudanças.
- **Ciclos**: oscilações de subida e de queda **de forma não periódica**, ao longo da componente de tendência.
- **Sazonalidade**: 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: **a sazonal possui movimentos facilmente previsíveis, ocorrendo em intervalos regulares de tempo (periódicos), enquanto movimentos cíclicos tendem a ser irregulares**. O exemplo dado para o comportamento cíclico é uma grande seca seguida de um ano com grande quantidade de chuva.

11.2 Modelos de previsão

Cinco famílias são apresentadas, em ordem crescente de sofisticação.

11.2.1 Naive (ingênuo)

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

11.2.2 Médias móveis (MA)

A previsão é a **média das últimas n observações**. O problema 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 limita o método: **todas as observações têm o mesmo peso**. Mas, na prática, as observações mais recentes tendem a ser mais relevantes — o que conduz diretamente ao próximo modelo.

11.2.3 Amortecimento exponencial

A ideia geral é parecida com a das médias móveis, mas os **pesos das observações decrescem à medida que as observações ficam mais distantes** no tempo. A taxa de decréscimo é determinada por uma ou mais **constantes de amortecimento**, e a maior dificuldade do método é justamente a **escolha dessas constantes**.

11.2.4 ARMA, ARIMA e SARIMA

O **ARMA** representa processos mistos auto-regressivos e de médias móveis, aplicáveis a **séries estacionárias**. Modelos mais sofisticados estendem essa família: o **ARIMA** inclui séries **não estacionárias** e o **SARIMA** inclui séries **sazonais**.

11.2.5 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 típico são **redes neurais**. O **pré-processamento é importante** nesses casos: pode ser útil **remover a tendência** antes do treinamento.

11.3 Definições operacionais

Antes de treinar qualquer modelo de previsão, três decisões precisam ser tomadas:

- **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 a previsão — por exemplo, outras séries históricas (índices financeiros, temperatura), dia da semana, hora da previsão, mês da previsão.

11.4 Arquiteturas NAR, NARX e Nonlinear Input-Output

- **NAR (Nonlinear Autoregressive Model)**: previsão de séries temporais **com valores da própria série**.
- **NARX (Nonlinear Autoregressive Model with External Input)**: previsão **com valores da própria série e valores de outra série**.
- **Nonlinear Input-Output**: previsão **apenas com valores de outra série**. Atenção: **soluções via NARX são mais precisas que essa** — utilize esta opção somente quando não estiverem disponíveis os dados da própria série a ser prevista.

Os slides também mencionam **redes recorrentes**, remetendo o aprofundamento à disciplina de Redes Neurais.

11.4.1 One Step Ahead versus Multi Step

Duas formas de operar a previsão:

- **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 têm todos os valores da série histórica imediatamente anteriores ao valor a ser previsto. **O erro não é propagado para novas previsões**.
- **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**, de modo que **o erro aumenta conforme se aumenta o número de previsões à frente**.

Essa distinção é fundamental na avaliação de modelos: uma acurácia excelente em open loop não garante desempenho aceitável em closed loop.

11.5 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 solução apresentada usa **4 redes diferentes de acordo com o dia da semana**. As entradas são:

- **janela de 5 valores passados** da carga;

- **valor da carga 7 dias antes, no mesmo horário**;
- **codificação binária da hora a ser prevista**.

O treinamento usa como conjunto os **2 meses anteriores**, com **re-treinamento a cada mês**.

A topologia da rede tem:

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

Esse caso ilustra bem as decisões operacionais listadas antes: a janela de entrada, o horizonte e as variáveis explicativas adicionais foram escolhidos a partir do conhecimento do domínio — o padrão de carga elétrica é fortemente sazonal por hora do dia e por dia da semana.

Outros dois casos são citados para prática: **faturamento de varejo** (com os atributos mês, ano e faturamento) e **passageiros de companhias aéreas**.

12 Deploy

12.1 Do notebook para a produção

A aula final trata da etapa que fecha o ciclo do projeto: colocar o modelo treinado em uso. As formas de entrega citadas são:

- **App desktop ou web**;
- **Executável por linha de comando**;
- **Bat script**;
- **Via código**, isto é, integração direta em outro sistema.

Os arquivos de apoio da aula reforçam essa lista: além dos notebooks de treinamento, há um notebook separado de **inferência**, um `app.py`, um `main.py`, um `requirements.txt` e um `README.md` — a estrutura mínima de um projeto de machine learning versionado e reprodutível. Vale destacar a separação entre o notebook de treinamento e o de inferência: **o artefato que vai para produção não é o notebook de exploração, mas um script enxuto que carrega o modelo já treinado e responde a novas observações**.

Um esqueleto ilustrativo dessa separação em Python:

```
# treinamento: salva o modelo em disco
import joblib
from sklearn.ensemble import RandomForestClassifier

modelo = RandomForestClassifier()
modelo.fit(X_train, y_train)
joblib.dump(modelo, "modelo.pkl")
```

```
# inferencia: carrega o modelo e responde a novos dados
import joblib
```

```
import pandas as pd

modelo = joblib.load("modelo.pkl")
novos = pd.read_csv("Dataset_spine_unknown.csv")
predicoes = modelo.predict(novos)
```

Note que a etapa de pré-processamento aplicada no treinamento — normalização, codificação de categorias, tratamento de faltantes — precisa ser **exatamente reproduzida** na inferência. Esse é o motivo pelo qual o pipeline de transformação também deve ser serializado junto com o modelo.

12.2 Recapitulação de árvores e comitês

A aula retoma os conceitos centrais de modelagem antes dos exercícios práticos: árvores de decisão criam modelos na forma de estruturas hierárquicas, 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 conjunto, e a Random Forest combina bootstrap, aleatorização de atributos e votação, com estimativa de erro **OOB**.

12.3 Casos práticos

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, 2 ou 3 comprimidos.

O exercício pede carregar a base `Islander_data.csv`, fazer livremente análises gráficas para entendimento dos dados usando a biblioteca **matplotlib** e treinar modelos de **árvore de decisão** e **random forest** para prever `Memory_after`.

Sintomas de dor lombar — a dor lombar pode ser causada por uma variedade de problemas em qualquer parte da rede interconectada de músculos, nervos, ossos, discos ou tendões da coluna lombar. Embora 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 objetivo é **identificar se uma pessoa está anormal ou normal** a partir de dados coletados da coluna física. A base tem **310 observações e 13 atributos** (12 preditores numéricos e uma classe binária):

- `Col1` incidência pélvica; `Col2` inclinação pélvica; `Col3` ângulo de lordose lombar; `Col4` inclinação sacral;
- `Col5` raio pélvico; `Col6` grau de espondilolistese; `Col7` inclinação pélvica (pelvic slope); `Col8` inclinação direta;
- `Col9` inclinação torácica; `Col10` inclinação cervical; `Col11` ângulo do sacro; `Col12` inclinação de escoliose;

- **Class**: Abnormal ou Normal.

Este é o caso conduzido de ponta a ponta na aula, com notebook de treinamento (`DorLombar.ipynb`), notebook de inferência (`DorLombar_inference.ipynb`), base de treino (`Dataset_spine.csv`) e base sem rótulo para simular produção (`Dataset_spine_unknown.csv`).

Doenças do coração — base Cleveland, com **13 atributos** (originalmente 76) mais a classe: idade, gênero, máxima frequência cardíaca, tipo de dores no peito, entre outros. São **303 instâncias**, sendo **165 doentes e 138 saudáveis**.

13 Trabalho

A avaliação da disciplina é feita por meio de um **Trabalho Final**, com as seguintes regras:

1. **Será enviado pelo Classroom um problema proposto pela professora**. Os alunos que desejarem podem **propor seus próprios trabalhos**.
2. Os alunos podem **escolher a ferramenta de preferência** para solucionar o problema.
3. O trabalho deve ser **enviado pelo Classroom em formato de relatório ou apresentação**.

Os arquivos de apoio disponibilizados para o trabalho incluem um dicionário de dados (`datadict.pdf`), as bases `horse.csv` e `horseTest.csv`, e o enunciado (`Trabalho.pdf`).

O escopo do curso oferece o roteiro natural para a execução: começar pela **análise exploratória** (tipos de variáveis, medidas resumo e visualização), passar pelo **pré-processamento** (missing values, outliers, normalização, seleção de atributos e balanceamento), aplicar e comparar **múltiplos algoritmos** — sem "algoritmo favorito", como recomendado na aula de balanceamento —, avaliar com **métricas adequadas ao problema** (não apenas acurácia, especialmente em bases desbalanceadas) e, quando pertinente, discutir a **colocação do modelo em produção**.

Os estudos de caso listados na ementa dão a medida da variedade de problemas cobertos: churn rate, consumo de cogumelos, classificação de plantas, predição de atrito e insatisfações no ambiente de trabalho, monitoramento de processos, avaliação de transações para gerenciamento de vendas e marketing, extração de conhecimento de consumo em shoppings, modelo para investidores, predição do uso de bicicletas para empresas de aluguel, previsão de faturamento e previsão do índice de preços de imóveis.

14 Síntese da Disciplina

A disciplina constrói, do início ao fim, uma competência prática: conduzir um projeto de mineração de dados completo, do entendimento do problema de negócio até a entrega de um modelo em operação.

O primeiro fio condutor é o **mapa das cinco classes de problemas** — previsão, classificação, regressão, agrupamento e associação — associadas a cinco problemas de negócio concretos. Esse mapa reaparece na abertura de praticamente todas as aulas e é o que permite ao aluno, diante de uma pergunta de negócio nova, identificar rapidamente que tipo de tarefa de mineração ela representa e, portanto, que família de algoritmos considerar.

O segundo fio condutor é a **primazia do pré-processamento**. Duas aulas inteiras são dedicadas a missing values, normalização, redução de dimensionalidade, seleção de atributos, balanceamento e outliers — e as lições são recorrentes ao longo de todo o curso. A maldição da dimensionalidade explica por que mais atributos não significam melhor modelo. O paradoxo da acurácia explica por que 95% de acerto pode representar um modelo inútil. A sensibilidade a outliers e a escalas explica por que KNN, K-means e SVM exigem normalização. Nada disso é detalhe de implementação: é o que separa um projeto que funciona de um que não funciona.

O terceiro fio condutor é a **progressão de simples para complexo dentro de cada família**. Em classificação, parte-se de um separador linear (SVM linear) para soft margin e kernel trick; de uma árvore isolada para comitês, bagging, boosting e Random Forest. Em associação, do Apriori ao FP-Growth, mais eficiente por usar a FP-tree, e ao Eclat, que entrega conjuntos frequentes em vez de regras. Em agrupamento, do K-means, que exige K como parâmetro, ao hierárquico, que permite escolher o corte no dendrograma, ao DBSCAN, que descobre clusters por densidade e ainda identifica ruído. Em regressão e séries temporais, do linear simples ao múltiplo, das árvores de regressão à Random Forest, e do Naive às médias móveis, ao amortecimento exponencial, ao ARIMA e aos modelos auto-regressivos não lineares. Em todos os casos, o modelo mais simples é **baseline obrigatório**, não uma etapa a ser pulada.

O quarto fio condutor é o **rigor na avaliação**. Cada bloco traz suas métricas e suas armadilhas: acurácia enganosa em bases desbalanceadas e a necessidade de matriz de confusão, precisão, recall, Kappa e F1; suporte, confiança e lift em associação, com a advertência de que ordenar por confiança devolve o óbvio e ordenar por lift revela o interessante; WCSS e o Elbow Method em agrupamento, com o cuidado de que WCSS zero significa um cluster por ponto; MAPE, RMSE, R quadrado e R quadrado ajustado em regressão, este último existindo justamente para punir a inclusão de regressores irrelevantes; e a distinção entre one step ahead e multi step em séries temporais, na qual o erro se propaga.

Por fim, o curso encerra onde muitos cursos param: no **deploy**. A separação entre notebook de exploração, notebook de treinamento e script de inferência, junto com `requirements.txt` e `README.md`, é o que transforma um experimento em um sistema. O trabalho final consolida todo esse percurso, exigindo do aluno as mesmas decisões que um projeto real exige — qual tarefa, quais dados, qual pré-processamento, quais algoritmos, quais métricas e qual forma de entrega.