

UNIVERSIDADE DO MINHO

Mestrado em Engenharia Biomédica

Aprendizagem e Extração de Conhecimento

Fundamentos, metodologia e prática laboratorial

Sebenta pedagógica · CRISP-DM · AI Studio · Python

Hugo Peixoto

Versão de trabalho revista · junho de 2026

Apresentação e orientação de estudo

Esta sebenta reúne os fundamentos e as atividades de Aprendizagem e Extração de Conhecimento, com aplicação ao contexto da Engenharia Biomédica. O CRISP-DM constitui o referencial de organização do trabalho, desde a formulação do problema à avaliação e utilização dos resultados.

Pré-requisitos. Noções de estatística descritiva, bases de dados relacionais e programação elementar em Python. Os exemplos visuais permitem introduzir os conceitos antes da sua implementação em código.

Percurso de estudo. A Parte I contém nove capítulos de fundamentos. A Parte II apresenta nove fichas de exercícios correspondentes. Recomenda-se estudar o capítulo teórico, executar o caso guiado e resolver a ficha, documentando escolhas, resultados e limitações.

Convenções. Os nomes dos atributos e dos operadores são conservados quando necessários à execução. Os valores numéricos dos casos guiados, exceto os cálculos explicitamente demonstrados, são os indicados nos materiais originais; não constituem resultados reproduzidos nesta revisão. Cada execução deve identificar ficheiro, versão, parâmetros e semente.

Dados e utilização responsável. Os ficheiros das fichas devem ser acompanhados de origem, licença, dicionário de variáveis e instruções de acesso. Exemplos com dados de saúde destinam-se à aprendizagem metodológica e não sustentam decisões clínicas. O estatuto sintético, anonimizado ou autorizado dos dados deve ser confirmado antes de qualquer utilização ou partilha.

Ambiente. Registrar as versões de Python e das bibliotecas, bem como a versão e extensões do AI Studio. A disponibilidade de operadores, portas, exemplos e licenças pode variar entre versões. Os exemplos de código completos desta edição encontram-se também na pasta `exemplos`.

Índice

1	Data Mining e a Metodologia CRISP-DM	6
1.1	O que é Data Mining?	6
1.2	Data Mining vs. KDD	6
1.3	Técnicas de Data Mining	6
1.4	A metodologia CRISP-DM	7
2	Correlação e Análise Exploratória	10
2.1	Correlação: definição formal	10
2.2	Pressupostos e limitações	10
2.3	Correlação não implica causalidade	11
2.4	Análise exploratória de dados (EDA)	11
3	Recolha de Dados: APIs e Integração	12
3.1	APIs REST e o formato JSON	12
3.2	ETL vs. ELT	12
3.3	Fluxo de trabalho: da API à base de dados	13
3.4	Idempotência e integridade referencial	13
4	Fundamentos da Preparação de Dados	14
4.1	Combinação de dados: tipos de junção	14
4.2	Mecanismos de dados em falta	14
4.3	Estratégias de imputação	15
4.4	Normalização e escalonamento	15
4.5	Deteção de outliers	15
5	Regras de Associação: Fundamentos	17
5.1	O que são regras de associação?	17
5.2	Métricas fundamentais	17
5.3	O princípio Apriori (anti-monotonicidade)	18
5.4	Comparação de algoritmos	18
6	Regressão Linear: Fundamentos	19
6.1	Regressão linear: conceito	19
6.2	Mínimos quadrados ordinários (OLS)	19
6.3	Pressupostos da regressão linear	19
6.4	Avaliação do modelo: R^2 e RMSE	20
6.5	Multicolinearidade	20
6.6	Regularização: Ridge, Lasso e Elastic Net	20
6.7	Interpretação de coeficientes e p-values	21
7	Classificação: Fundamentos	22
7.1	Classificação: conceito	22
7.2	Árvores de decisão	22
7.3	Comparação de algoritmos de classificação	23
7.4	O compromisso Bias-Variância	23
7.5	Validação cruzada	23
7.6	Métricas de avaliação	24
7.7	Curva ROC e AUC	24
8	Extração de Conhecimento em Código: Fundamentos	25
8.1	Ferramentas visuais vs. código	25
8.2	Reprodutibilidade	25
8.3	Pipelines e separação de responsabilidades	26
8.4	Notebooks vs. scripts	26
9	Clustering: Fundamentos	27
9.1	O algoritmo k-Means	27

9.2	Inicialização: k-means++	27
9.3	Escolha do número de clusters k	27
9.4	Comparação com outros métodos de clustering	28
9.5	Distâncias	28
10	Exercícios — Introdução à Metodologia CRISP-DM	31
11	Exercícios — AI Studio: Correlações	32
11.1	O AI Studio (RapidMiner)	32
11.2	Caso prático guiado	32
12	Exercícios — Recolha de Dados via API	34
12.1	Exemplo de referência: dados de doentes e registos clínicos	34
13	Exercícios — Preparação de Dados no AI Studio	36
13.1	Operadores de referência	36
14	Exercícios — AI Studio: Regras de Associação	38
14.1	Caso prático guiado	38
15	Exercícios — AI Studio: Regressão Linear	40
15.1	Caso prático guiado	40
16	Exercícios — AI Studio: Classificação	42
16.1	Caso prático guiado	42
17	Exercícios — Extração de Conhecimento com Python	44
17.1	Fluxo de trabalho em código	44
18	Exercícios — k-Means Clustering	47
18.1	Caso prático guiado	47
18.2	k-Means em Python	47
	Orientações para a autoavaliação	49
	Glossário	50
	Referências e documentação de apoio	52

PARTE I

Fundamentos

Conceitos, preparação de dados, modelação e avaliação

CAPÍTULO 1

Data Mining e a Metodologia CRISP-DM

Objetivos de aprendizagem

- Definir Data Mining e situá-lo no processo mais amplo de Descoberta de Conhecimento em Bases de Dados (KDD).
- Identificar e comparar as principais técnicas de extração de conhecimento.
- Explicar a origem, características e objetivos da metodologia CRISP-DM, e compará-la com metodologias alternativas.
- Descrever, em detalhe, as seis fases do processo CRISP-DM e as suas tarefas típicas.

1.1 O que é Data Mining?

Data Mining. Processo de extração de padrões ou conhecimento de interesse — não triviais, implícitos, anteriormente desconhecidos e potencialmente úteis — a partir de grandes quantidades de dados.

A crescente disponibilidade de dados não se traduz automaticamente em conhecimento útil. Esta diferença resulta, em parte, da conjugação de dois fatores:

1. **Aplicações que geram enormes quantidades de dados:** a internet, sistemas de informação, experiências laboratoriais, transações comerciais, simulações, dados clínicos, dados bancários, dados pessoais, sistemas de CCTV, dispositivos de IoT, entre outros.
2. **Tecnologias capazes de recolher e armazenar esses dados:** satélites, câmaras, scanners, wearables, dispositivos móveis, cloud computing, bases de dados e data warehouses.

A recolha, a qualidade e a representatividade dos dados continuam a ser desafios; a extração de conhecimento exige ainda métodos adequados e avaliação crítica.

1.2 Data Mining vs. KDD

É comum usar-se “Data Mining” como sinónimo do processo completo de descoberta de conhecimento, mas, formalmente, o Data Mining é apenas **uma etapa** dentro de um processo mais amplo: o **KDD** (*Knowledge Discovery in Databases*).

KDD — Knowledge Discovery in Databases. Processo global, não trivial, de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis, a partir de dados. O KDD engloba a **seleção, pré-processamento, transformação, Data Mining** (a etapa de aplicação de algoritmos) e **interpretação/avaliação** dos padrões descobertos.

Ou seja: o Data Mining é o “motor” algorítmico (deteção de padrões), enquanto o KDD é a jornada completa — desde a pergunta de negócio até ao conhecimento acionável. Ao longo desta sebenta, e seguindo a prática comum, o termo “Data Mining” será por vezes usado de forma mais lata para designar todo o processo, mas é importante reconhecer esta distinção formal.

1.3 Técnicas de Data Mining

O Data Mining não é uma técnica única, mas sim uma família de abordagens, cada uma adequada a um tipo diferente de pergunta. Uma primeira grande divisão é entre **aprendizagem supervisionada** (existe um atributo-alvo conhecido, que se pretende prever) e **aprendizagem não supervisionada** (não existe atributo-alvo — o objetivo é descobrir estrutura latente nos dados).

Técnica	Tipo	Objetivo
Regras de Associação	Não supervisionada	Encontrar padrões de coocorrência entre itens (ex.: cabazes de compra).
Classificação	Supervisionada	Construir modelos que distingam classes discretas, para prever a que classe pertencem novas observações.
Regressão	Supervisionada	Prever um valor numérico contínuo a partir de variáveis explicativas.
Agrupamento (Clustering)	Não supervisionada	Agrupar observações semelhantes entre si, sem classes pré-definidas.
Deteção de outliers	Não supervisionada	Identificar observações que não seguem o comportamento geral dos dados.
Análise de séries temporais	Supervisionada / não sup.	Analisar tendência, sazonalidade e periodicidade ao longo do tempo.

Tabela 1: Principais técnicas de Data Mining, por tipo de aprendizagem

Esta sebenta desenvolve correlação, preparação de dados, regras de associação, regressão, classificação e agrupamento. A análise de séries temporais é apenas contextualizada; não constitui um capítulo autónomo.

1.4 A metodologia CRISP-DM

1.4.1 Origem e objetivos

CRISP-DM — *CRoss Industry Standard Process for Data Mining* — é uma metodologia resultante de um esforço financiado pela Comunidade Europeia para desenvolver um processo estruturado para projetos de Data Mining. Os seus principais objetivos são encorajar a utilização de ferramentas interoperáveis ao longo de todo o processo, e permitir retirar conhecimento valioso mesmo de tarefas simples.

1.4.2 Características fundamentais

Confiável e repetível. O processo deve poder ser seguido de forma consistente, mesmo por pessoas com pouco conhecimento em Data Mining.

Guiado por diretrizes. A metodologia reúne diretrizes e documentação da experiência associada ao processo.

Flexível. O processo deve adaptar-se a diferentes problemas de negócio e conjuntos de dados.

1.4.3 Comparação com outras metodologias

O CRISP-DM não é a única metodologia estruturada para projetos de Data Mining. Duas alternativas frequentemente referidas na literatura são o processo **KDD** (na sua formulação clássica de 5 a 9 passos, conforme o autor) e o **SEMMA** (*Sample, Explore, Modify, Model, Assess*), proposto pela SAS Institute.

Aspeto	CRISP-DM	SEMMA
Origem	Consórcio industrial (Europa, anos 1990)	SAS Institute
Foco	Todo o ciclo de vida do projeto, incluindo negócio e deployment	Sobretudo as etapas técnicas de modelação
Contexto de negócio	Explícito (Business Understanding)	Não contemplado diretamente
Deployment	Fase explícita, com monitorização	Não contemplado diretamente
Neutralidade de ferramenta	Independente de fornecedor	Historicamente associado ao SAS Enterprise Miner

Tabela 2: CRISP-DM vs. SEMMA

Nota metodológica

O CRISP-DM oferece um referencial transversal ao ciclo de vida do projeto e independente de uma ferramenta específica. É adotado nesta sebenta para articular objetivos, preparação dos dados, modelação e avaliação. Ver a documentação de referência no final do documento.

1.4.4 As seis fases do processo

Fase	Foco principal
1. Business Understanding	Compreender os objetivos e requisitos do projeto, do ponto de vista do negócio.
1. Compreensão do problema e dos dados	Recolher, explorar e familiarizar-se com os dados disponíveis.
3. Data Preparation	Selecionar, limpar, construir, integrar e formatar os dados para modelação.
3. Modelação	Selecionar e aplicar técnicas de modelação, calibrando parâmetros.
4. Avaliação	Avaliar se os modelos obtidos cumprem os objetivos de negócio definidos.
6. Deployment	Colocar o conhecimento obtido em prática, monitorizando e mantendo a solução.

Tabela 3: As seis fases da metodologia CRISP-DM

Business Understanding (Compreender o Negócio)

- **Determinar os objetivos do negócio:** documentar o contexto (*background*), definir os objetivos e o critério de sucesso do negócio.
- **Avaliar a situação:** avaliar recursos disponíveis, identificar constrangimentos, documentar custos e benefícios.
- **Determinar os objetivos de Data Mining:** traduzir o objetivo de negócio num objetivo técnico e definir o critério de sucesso da aplicação de DM.
- **Produzir um plano de projeto:** definir o plano de implementação, as técnicas e as ferramentas a utilizar.

Data Understanding (Compreender os Dados)

- **Recolher os dados iniciais:** identificar as fontes de dados, definir métodos de recolha e extrair os dados.
- **Descrever os dados:** avaliar a quantidade de dados, o tipo de atributos, o seu significado para o negócio e estatísticas básicas.
- **Explorar os dados:** analisar em detalhe os atributos de interesse, definir relações e agrupamentos, e produzir um relatório de exploração.
- **Verificar a qualidade dos dados:** avaliar a existência de campos nulos, atributos em falta e a coerência entre valor e significado.

Data Preparation (Preparar os Dados)

- **Selecionar os dados:** aplicar critérios de inclusão/exclusão, selecionar e criar atributos.
- **Limpar os dados:** remover dados desnecessários e tratar valores em falta.
- **Construir dados:** derivar novos atributos e validar a reconstrução de campos nulos.
- **Integrar e formatar dados:** combinar atributos de outras fontes e organizar o dataset final.

Modeling (Criar os Modelos)

- **Selecionar a técnica,** desenhar o modelo de teste (divisão treino/teste), construir o modelo e fazer uma primeira avaliação interna dos resultados.

Evaluation (Avaliar os Resultados)

- **Avaliar os resultados** face aos objetivos de negócio, **rever o processo** criticamente e **determinar os próximos passos**.

Deployment (Implementação)

- **Planear o deployment e a monitorização, produzir o relatório final e rever o projeto** para retirar lições futuras.

Nota metodológica

O CRISP-DM é, na prática, um processo **iterativo e cíclico**: é comum, por exemplo, que a fase de Modeling revele problemas de qualidade nos dados que obriguem a voltar à Data Preparation, ou mesmo à Data Understanding.

CAPÍTULO 2

Correlação e Análise Exploratória

Objetivos de aprendizagem

- Definir formalmente o coeficiente de correlação de Pearson e conhecer alternativas para outros tipos de relação.
- Distinguir correlação de causalidade.
- Comparar diferentes medidas de associação entre atributos.
- Reconhecer as condições em que a correlação de Pearson é adequada (e quando não é).

2.1 Correlação: definição formal

Correlação de Pearson. Medida estatística da força e direção da associação linear entre duas variáveis numéricas com variância não nula. Outras medidas de correlação caracterizam outros tipos de associação.

O **coeficiente de correlação de Pearson**, r , entre dois atributos X e Y , com n observações, é definido como:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

onde $\text{cov}(X, Y)$ é a **covariância** entre X e Y , e σ_X , σ_Y são os respetivos desvios-padrão. O resultado é sempre um valor em $[-1, 1]$:

- Um coeficiente em $]0, 1]$ indica uma **correlação positiva**.
- Um coeficiente em $[-1, 0[$ indica uma **correlação negativa**.
- Quanto mais próximo de 1 ou -1 , mais forte é a correlação; um valor próximo de 0 indica ausência de relação **linear** (mas não necessariamente ausência de qualquer relação — ver adiante).

Numa regressão linear simples ajustada por mínimos quadrados, com termo independente e avaliada na mesma amostra, $R^2 = r^2$. É nesse contexto que r^2 representa a fração da variabilidade amostral de Y captada pelo ajuste linear; esta interpretação não estabelece causalidade.

2.2 Pressupostos e limitações

Referência de apoio: documentação SciPy, *pearsonr* (ver referências).

O coeficiente de Pearson pode ser calculado para pares numéricos com variância não nula, sem exigir normalidade marginal. A sua interpretação requer atenção a três aspetos:

1. **Forma da relação:** mede associação linear; uma relação quadrática forte pode produzir um valor próximo de zero.
2. **Observações influentes:** valores extremos podem alterar substancialmente o coeficiente.
3. **Escala e desenho amostral:** códigos de categorias não têm, em geral, distâncias quantitativas interpretáveis. Medidas ordinais e observações repetidas exigem cuidados específicos.

Os testes e intervalos de confiança associados a r têm pressupostos próprios, distintos do simples cálculo do coeficiente. A escolha entre inferência paramétrica, permutação ou reamostragem deve respeitar o desenho do estudo.

Quando estes pressupostos não se verificam, existem alternativas:

Medida	Deteta	Quando usar
Pearson (r)	Relações lineares	Variáveis numéricas; interesse na associação linear; inspeção de pontos influentes
Spearman (ρ)	Relações monótonas (não necessariamente lineares)	Atributos ordinais, ou numéricos com outliers/relações não-lineares monótonas
Kendall (τ)	Concordância de ordenação entre pares de observações	Amostras pequenas, muitos empates, dados ordinais

Tabela 4: Medidas de correlação/associação e o seu domínio de aplicação

O coeficiente de **Spearman** é, na prática, o coeficiente de Pearson calculado sobre as **ordens** (*ranks*) dos valores, em vez dos valores originais — o que reduz a influência das magnitudes extremas, sem o tornar imune a observações influentes, e permite detetar relações monótonas não-lineares (por exemplo, exponenciais).

Nota metodológica

Um coeficiente de correlação **não** é uma percentagem. Um valor de 0.776 não significa “77.6% de variabilidade explicada” — numa regressão linear simples com termo independente, ajustada e avaliada na mesma amostra, $R^2 = 0.776^2 \approx 0.602$ (60.2%) tem essa interpretação descritiva. O coeficiente de correlação, por si só, mede apenas a força e direção da associação linear.

2.3 Correlação não implica causalidade

Este é, provavelmente, o erro de interpretação mais comum em análise de dados: duas variáveis podem estar fortemente correlacionadas sem que exista qualquer relação causal direta entre elas. As explicações alternativas mais comuns para uma correlação observada incluem:

- **Relação causal:** pode contribuir para uma associação observada, mas não pode ser estabelecida apenas pelo coeficiente. Requer um desenho e pressupostos de identificação adequados.
- **Causa comum (variável confundidora)** — uma terceira variável Z , não observada, influencia tanto X como Y , criando uma correlação espúria entre elas.
- **Coincidência estatística** — em datasets com muitos atributos, é estatisticamente esperado encontrar algumas correlações fortes por puro acaso (o problema das **comparações múltiplas**).

Caso de estudo • Vendas de gelados e afogamentos

Num dataset de dados mensais, o número de vendas de gelados está fortemente correlacionado com o número de afogamentos em praias. Comer gelado não causa afogamentos — ambas as variáveis são influenciadas por uma terceira: a temperatura (e a época do ano). Este é o exemplo clássico de uma variável confundidora.

2.4 Análise exploratória de dados (EDA)

A correlação é uma das várias ferramentas de **Análise Exploratória de Dados** (*Exploratory Data Analysis*, EDA) — o conjunto de técnicas usadas na fase de Data Understanding do CRISP-DM para sumarizar as principais características de um dataset, frequentemente com recurso a métodos visuais. Além da matriz de correlação, as ferramentas de EDA mais comuns incluem:

- **Estatísticas descritivas:** média, mediana, moda, desvio-padrão, quartis, mínimo e máximo, por atributo.
- **Histogramas:** para observar a forma da distribuição de um atributo numérico (simétrica, enviesada, bimodal, etc.).
- **Boxplots:** para comparar distribuições entre grupos e identificar outliers visualmente.
- **Gráficos de dispersão** (*scatter plots*): para observar visualmente a relação entre dois atributos numéricos, complementando o valor numérico da correlação.

CAPÍTULO 3

Recolha de Dados: APIs e Integração

Objetivos de aprendizagem

- Definir formalmente API, REST e JSON, e os principais métodos HTTP.
- Comparar as abordagens ETL e ELT para integração de dados.
- Compreender os requisitos de um modelo relacional para dados recolhidos externamente.
- Discutir boas práticas de segurança e de qualidade de dados na recolha via API.

3.1 APIs REST e o formato JSON

API — Application Programming Interface. Conjunto de regras e protocolos que permite que duas aplicações comuniquem entre si, expondo funcionalidades ou dados sem expor a implementação interna.

REST — Representational State Transfer. Estilo arquitetural para sistemas distribuídos em que os recursos (dados) são identificados por URLs e manipulados através de um conjunto uniforme de operações, tipicamente correspondentes aos métodos HTTP.

Os métodos HTTP mais usados numa API REST correspondem, na prática, às quatro operações fundamentais sobre dados (**CRUD**: *Create, Read, Update, Delete*):

Método HTTP	Operação CRUD	Uso típico
GET	Read	Obter um recurso (ex.: o JSON com a lista de doentes).
POST	Create	Criar um novo recurso.
PUT / PATCH	Update	Atualizar um recurso existente (total ou parcialmente).
DELETE	Delete	Remover um recurso.

Tabela 5: Métodos HTTP e a sua correspondência com operações CRUD

Os dados são tipicamente devolvidos em formato **JSON** (*JavaScript Object Notation*) — um formato de texto legível por humanos, baseado em pares chave-valor, listas e objetos aninhados, amplamente usado na troca de dados na web; outros formatos, incluindo XML, continuam a ser utilizados.

3.2 ETL vs. ELT

A integração de dados provenientes de fontes externas (como uma API) numa base de dados centralizada segue tipicamente um de dois padrões:

Padrão	Ordem das etapas	Quando usar
ETL (<i>Extract, Transform, Load</i>)	Os dados são extraídos, transformados/limpos fora da base de dados destino, e só depois carregados.	Volumes moderados, transformações complexas, destino com esquema relacional rígido (o caso do script apresentado neste capítulo).
ELT (<i>Extract, Load, Transform</i>)	Os dados são extraídos e carregados “em bruto”, e transformados dentro do sistema destino (tipicamente um data warehouse).	Grandes volumes, quando o destino tem grande capacidade de processamento (ex.: data warehouses na cloud).

Tabela 6: ETL vs. ELT

A ficha prática ilustra a extração e transformação inicial em Python, seguindo-se uma proposta de esquema MySQL. O carregamento transacional e idempotente constitui uma tarefa a implementar e validar pelo estudante.

3.3 Fluxo de trabalho: da API à base de dados

O processo típico de recolha de dados a partir de uma API, até à sua disponibilização para análise, segue os seguintes passos:

1. **Verificar a disponibilidade dos dados:** aceder diretamente aos endpoints da API e confirmar que a resposta é válida.
2. **Configurar o ambiente Python:** garantir que estão instaladas as bibliotecas necessárias para pedidos HTTP (`requests`) e para acesso à base de dados.
3. **Obter os dados:** escrever um script que faça o pedido HTTP a cada endpoint, interprete a resposta JSON e a guarde.
4. **Desenhar e criar o modelo relacional:** definir tabelas, chaves primárias e chaves estrangeiras, respeitando as relações entre entidades.
5. **Carregar os dados:** inserir os dados na base de dados, com validação e normalização dos valores.

3.4 Idempotência e integridade referencial

Dois conceitos são particularmente importantes num script de carregamento de dados como o apresentado na Parte II deste capítulo:

Idempotência. Uma operação é idempotente quando pode ser executada múltiplas vezes produzindo sempre o mesmo resultado final, sem efeitos colaterais adicionais em execuções repetidas. Um script de carregamento idempotente pode ser reexecutado em segurança (por exemplo, após uma falha a meio) sem duplicar dados.

Na prática, a idempotência é conseguida com padrões como `INSERT ... ON DUPLICATE KEY UPDATE` (MySQL) ou `INSERT ... ON CONFLICT` (PostgreSQL), que atualizam um registo existente em vez de falharem por violação de chave primária.

Integridade referencial. Propriedade de uma base de dados relacional em que cada chave estrangeira não nula referencia uma chave válida na tabela correspondente. A admissibilidade de valores nulos depende do esquema. Garante que não existem “órfãos” — por exemplo, um registo clínico associado a um doente inexistente.

CAPÍTULO 4

Fundamentos da Preparação de Dados

Objetivos de aprendizagem

- Classificar formalmente os mecanismos de dados em falta (MCAR, MAR, MNAR).
- Comparar estratégias de imputação de valores em falta.
- Comparar métodos de normalização/escalonamento de atributos.
- Comparar métodos de deteção de outliers, com as respetivas fórmulas.

A fase de **Data Preparation** do CRISP-DM é, frequentemente, a que consome mais tempo num projeto real de Data Mining — e é também a que mais condiciona a qualidade dos resultados obtidos nas fases seguintes.

4.1 Combinação de dados: tipos de junção

Combinar dados de múltiplas fontes ou tabelas — a operação de **Join** — segue a mesma teoria de álgebra relacional usada em bases de dados:

Tipo de Join	Resultado
Inner Join	Apenas os registos com correspondência em ambos os conjuntos de dados.
Left (Outer) Join	Todos os registos do conjunto à esquerda, com valores nulos onde não há correspondência à direita.
Right (Outer) Join	Todos os registos do conjunto à direita, com valores nulos onde não há correspondência à esquerda.
Full (Outer) Join	Todos os registos de ambos os conjuntos, com nulos onde não há correspondência.

Tabela 7: Tipos de junção entre conjuntos de dados

A operação de **Aggregate** (agrupar + agregar) corresponde, em termos conceptuais, a uma cláusula `GROUP BY` em SQL: agrupa registos por um ou mais atributos, e calcula, para cada grupo, uma função de agregação (contagem, soma, média, moda, etc.) sobre outros atributos.

4.2 Mecanismos de dados em falta

Antes de decidir **como** tratar valores em falta, é importante compreender **porque** faltam — a literatura estatística distingue três mecanismos:

Mecanismo	Significado
MCAR (<i>Missing Completely At Random</i>)	A probabilidade de faltar um valor é independente de qualquer outro atributo, observado ou não. Caso mais favorável — a remoção de casos não introduz enviesamento.
MAR (<i>Missing At Random</i>)	A probabilidade de faltar depende de outros atributos observados no dataset (ex.: doentes mais velhos têm mais valores em falta num exame). Pode permitir métodos de imputação condicionados nesses atributos, desde que o modelo de imputação seja adequado; não garante ausência de enviesamento.
MNAR (<i>Missing Not At Random</i>)	A probabilidade de faltar depende do próprio valor em falta (ex.: doentes com rendimentos muito altos recusam-se a reportá-los). O caso mais difícil de tratar corretamente.

Tabela 8: Mecanismos de dados em falta

4.3 Estratégias de imputação

Estratégia	Descrição	Risco principal
Remoção de casos (<i>listwise deletion</i>)	Eliminar linhas com valores em falta.	Perda de dados; enviesamento se não for MCAR.
Remoção de atributos	Eliminar colunas com proporção elevada de valores em falta.	Perda de informação potencialmente relevante.
Imputação pela média/mediana	Substituir pela média (atributos numéricos simétricos) ou mediana (assimétricos).	Reduz artificialmente a variância do atributo.
Imputação pela moda	Substituir pelo valor mais frequente (atributos categóricos).	Pode reforçar desequilíbrios de classe existentes.
Imputação por regressão / kNN	Prever o valor em falta a partir de outros atributos.	Mais complexo; pode introduzir correlações artificiais.

Tabela 9: Comparação de estratégias de imputação de valores em falta

Nota metodológica

A ordem das operações importa: filtrar exemplos com valores em falta **antes** de remover atributos problemáticos ou substituir valores pode eliminar observações desnecessariamente — um atributo com 90% de valores em falta exige analisar a sua utilidade, o mecanismo de ausência e as alternativas; nenhuma percentagem determina, isoladamente, a sua exclusão.

4.4 Normalização e escalonamento

Muitos algoritmos (k-Means, k-NN, regressão com regularização, redes neuronais) são sensíveis à **escala** dos atributos — um atributo medido em milhares (ex.: rendimento anual) domina artificialmente a distância ou o gradiente, comparado com um atributo medido em unidades (ex.: número de filhos).

Método	Fórmula	Quando usar
Min-Max	$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$	Mapeia o intervalo de treino para $[0, 1]$; novos valores podem sair desse intervalo. Sensível a outliers.
Z-score (Padronização)	$x' = \frac{x - \bar{x}}{\sigma}$	Centra e reescala; não exige nem produz uma distribuição normal.
Robust Scaling	$x' = \frac{x - \text{mediana}(x)}{\text{IQR}(x)}$	Quando existem outliers significativos — usa mediana e amplitude interquartil em vez de média e desvio-padrão.

Tabela 10: Métodos de normalização/escalonamento de atributos numéricos

Nota metodológica

Em tarefas preditivas, separar treino e teste antes de aprender imputações, escalas, seleção de atributos ou limiares. Na validação cruzada, cada transformação é ajustada apenas no treino de cada partição. Uma transformação sem acesso ao teste não impede outras formas de fuga de informação, como incluir variáveis medidas depois do desfecho.

4.5 Detecção de outliers

Outlier. Observação que se desvia significativamente do comportamento geral dos restantes dados, ao ponto de levantar suspeita de ter sido gerada por um mecanismo diferente. Um outlier não é necessariamente ruído a descartar — pode ser precisamente o que se procura (ex.: deteção de fraude).

Método	Princípio
Z-score	Marca como outlier qualquer valor a mais de k desvios-padrão da média (tipicamente $k = 2$ ou $k = 3$): $ z = \left \frac{x-\bar{x}}{\sigma}\right > k$.
IQR (amplitude interquartil)	Marca como outlier valores fora de $[Q_1 - 1.5 \times \text{IQR}, Q_3 + 1.5 \times \text{IQR}]$, onde $\text{IQR} = Q_3 - Q_1$. Mais robusto que o Z-score, por não depender da média/desvio-padrão.
Distância / k-vizinhos mais próximos	Marca como outlier observações cuja distância média aos k vizinhos mais próximos é anormalmente alta.
LOF (<i>Local Outlier Factor</i>)	Compara a densidade local de uma observação com a densidade dos seus vizinhos — deteta outliers “locais” mesmo em datasets com clusters de densidades diferentes, onde o Z-score ou a distância simples falhariam.

Tabela 11: Métodos de deteção de outliers

Nota metodológica

As médias e desvios-padrão usados em muitos destes métodos são, eles próprios, sensíveis a outliers — um único valor extremo pode “esconder-se” ao inflacionar o desvio-padrão. Por isso, em datasets com outliers muito extremos, os métodos baseados em mediana/IQR ou em densidade (LOF) são preferíveis aos baseados em média/desvio-padrão (Z-score).

CAPÍTULO 5

Regras de Associação: Fundamentos

Objetivos de aprendizagem

- Definir formalmente suporte, confiança, lift e conviction.
- Explicar o princípio de anti-monotonicidade (princípio Apriori).
- Comparar os algoritmos Apriori, FP-Growth e Eclat.
- Reconhecer os limites de interpretação das regras de associação.

5.1 O que são regras de associação?

Regra de associação. Implicação da forma $A \rightarrow C$ (“se A , então C ”), onde A e C são conjuntos de itens (*itemsets*) disjuntos, que expressa um padrão de coocorrência frequente entre atributos de um dataset.

São particularmente comuns na **análise de cabazes de compra** (*market basket analysis*), mas aplicam-se a qualquer domínio com atributos binários de presença/ausência — incluindo, por exemplo, a coocorrência de sintomas ou comorbilidades em registos clínicos.

5.2 Métricas fundamentais

Seja N o número total de observações (transações), e $\text{suporte}(X)$ a proporção de observações que contêm o itemset X :

Suporte

$$\text{suporte}(A \rightarrow C) = \text{suporte}(A \cup C) = \frac{n(A \cup C)}{N}$$

onde $n(X)$ conta as transações que contêm todos os itens de X . O suporte é uma frequência relativa; $|A \cup C|$ contaria itens, não transações.

Confiança

$$\text{confiança}(A \rightarrow C) = \frac{\text{suporte}(A \cup C)}{\text{suporte}(A)}$$

Mede a fiabilidade condicional da regra: entre as observações com A , que proporção também tem C ?

Lift

$$\text{lift}(A \rightarrow C) = \frac{\text{confiança}(A \rightarrow C)}{\text{suporte}(C)} = \frac{\text{suporte}(A \cup C)}{\text{suporte}(A) \times \text{suporte}(C)}$$

Mede o quanto a presença de A aumenta (ou diminui) a probabilidade de C , relativamente à probabilidade **base** de C (ou seja, corrige a confiança pela popularidade geral de C).

Conviction

$$\text{conviction}(A \rightarrow C) = \frac{1 - \text{suporte}(C)}{1 - \text{confiança}(A \rightarrow C)}$$

Mede o quanto a regra falharia mais frequentemente se A e C fossem independentes — valores mais altos indicam regras mais fortes.

Interpretação do Lift.

- lift = 1: na distribuição empírica, os eventos de presença de A e C satisfazem a igualdade de independência; isto não prova independência na população.
- lift > 1: a presença de A **aumenta** a probabilidade de C — associação positiva.
- lift < 1: a presença de A **diminui** a probabilidade de C — associação negativa (substituição).

Nota metodológica

A confiança, isoladamente, pode induzir em erro: uma regra pode ter confiança elevada apenas porque C é, por si só, muito frequente no dataset (independentemente de A). O Lift corrige este efeito, e é por isso um complemento importante à leitura de suporte e confiança.

5.3 O princípio Apriori (anti-monotonicidade)

O desafio computacional central na descoberta de regras de associação é que o número de itemsets possíveis cresce exponencialmente com o número de atributos (2^n subconjuntos possíveis, para n atributos). O **princípio Apriori** torna o problema tratável:

Princípio Apriori (anti-monotonicidade). Se um itemset é frequente (satisfaz o suporte mínimo), então todos os seus subconjuntos são também frequentes. Equivalentemente: se um itemset **não** é frequente, nenhum dos seus super-conjuntos pode ser frequente.

Este princípio permite podar drasticamente o espaço de procura: basta descartar um itemset infrequente para descartar, automaticamente, todos os itemsets que o contêm — sem necessidade de os contar explicitamente.

5.4 Comparação de algoritmos

Algoritmo	Estratégia	Vantagens / Desvantagens
Apriori	Gera candidatos iterativamente (itemsets de tamanho k , depois $k + 1$, ...), podando com o princípio da anti-monotonicidade. Requer múltiplas passagens pelo dataset.	Simples e intuitivo; mas pode ser lento em datasets grandes ou com muitos itens frequentes, devido à geração explícita de candidatos.
FP-Growth	Constrói uma estrutura compacta em árvore (<i>FP-tree</i> , <i>Frequent Pattern tree</i>) que resume o dataset, e extrai os itemsets frequentes diretamente da árvore, sem gerar candidatos explicitamente.	Tipicamente muito mais rápido que o Apriori (apenas duas passagens pelo dataset); estrutura de dados mais complexa de implementar.
Eclat	Representa cada item pela lista de transações (IDs) em que ocorre (formato vertical), e calcula suportes por interseção dessas listas.	O desempenho e o consumo de memória dependem da representação dos conjuntos de transações e da densidade dos dados; deve ser comparado empiricamente.

Tabela 12: Comparação de algoritmos de descoberta de itemsets frequentes

No RapidMiner/AI Studio, os atributos usados nestes operadores devem ser do tipo **binominal** (não confundir com *binomial*: binominal significa “um de dois valores”, que tanto podem ser numéricos como categóricos).

Caso de estudo · Cálculo de suporte, confiança e lift

Considere o seguinte conjunto de transações:

id	Items comprados
1	A, B, C
2	A, C
3	A, D
4	B, E, F

Para a regra $A \rightarrow C$:

- $\text{suporte}(\{A\}) = 75\%$, $\text{suporte}(\{C\}) = 50\%$, $\text{suporte}(\{A, C\}) = 50\%$
- $\text{confiança}(A \rightarrow C) = \frac{\text{suporte}(\{A, C\})}{\text{suporte}(\{A\})} = \frac{50}{75} \approx 66.7\%$
- $\text{lift}(A \rightarrow C) = \frac{\frac{2}{4}}{0.5} = 1.33$ — a presença de A aumenta em 33% a probabilidade de C , relativamente à sua probabilidade base.

Nota metodológica

O processo CRISP-DM é de natureza cíclica: é normal que uma primeira tentativa de modelação não produza qualquer regra (por exemplo, com um suporte ou confiança mínimos demasiado exigentes) — nesse caso, o passo seguinte não é desistir, mas voltar atrás e ajustar os parâmetros ou reconsiderar a preparação dos dados.

CAPÍTULO 6

Regressão Linear: Fundamentos

Objetivos de aprendizagem

- Formular o critério dos mínimos quadrados ordinários (OLS) e interpretar a solução matricial.
- Enumerar e explicar os pressupostos da regressão linear.
- Interpretar R^2 , R^2 ajustado e multicolinearidade.
- Comparar a regressão linear simples com métodos de regularização (Ridge, Lasso, Elastic Net).

6.1 Regressão linear: conceito

Regressão Linear. Modelo estatístico que assume uma relação linear entre uma variável dependente (resposta) y e uma ou mais variáveis independentes x , usando essa relação para prever valores numéricos contínuos.

Regressão linear simples

$$y = mx + b$$

(uma única variável independente)

Regressão linear múltipla

$$y = m_1x_1 + m_2x_2 + \dots + m_nx_n + b$$

(várias variáveis independentes, cada uma com o seu coeficiente)

Nestas fórmulas, y é o que se pretende prever (o *label*), cada m_i é o **coeficiente** do atributo x_i (o seu peso no modelo) e b é a **constante** (*intercept*).

6.2 Mínimos quadrados ordinários (OLS)

Como são os coeficientes m_i e b determinados? O método mais comum é o dos **Mínimos Quadrados Ordinários** (*Ordinary Least Squares*, OLS): escolher os coeficientes que minimizam a soma dos quadrados dos **resíduos** (a diferença entre o valor observado e o valor previsto):

$$\min_{m,b} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \min_{m,b} \sum_{i=1}^n (y_i - (mx_i + b))^2$$

Em notação matricial, com X a matriz de atributos (incluindo uma coluna de 1's para o intercept) e β o vetor de coeficientes, a solução fechada (*equações normais*) é:

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

Esta expressão exige que X tenha posto completo por colunas. Na implementação, preferem-se métodos numericamente estáveis, como QR ou SVD; se a matriz for singular, pode usar-se a pseudoinversa. O modelo estatístico inclui ainda um erro: $y = X\beta + \varepsilon$, com média condicional $E(\varepsilon | X) = 0$.

Minimizar o quadrado dos resíduos (em vez do seu valor absoluto, por exemplo) penaliza mais fortemente erros grandes, e tem a vantagem de possuir uma solução analítica fechada.

6.3 Pressupostos da regressão linear

A validade estatística de um modelo de regressão linear (nomeadamente dos seus p-values e intervalos de confiança) depende de um conjunto de pressupostos, frequentemente resumidos pelo mnemónico **LINE**:

Pressuposto	Significado
L — Linearidade	A relação entre as variáveis independentes e a resposta é, de facto, linear.
I — Independência	A dependência entre observações deve ser modelada; os erros não devem apresentar a dependência ignorada pelo modelo inferencial adotado.
N — Normalidade	Na inferência clássica exata em pequenas amostras, assume-se normalidade dos erros condicionais; a normalidade não é necessária para calcular OLS.
E — Homoscedasticidade (<i>Equal variance</i>)	A variância dos resíduos é constante ao longo de todos os valores previstos (não deve “abrir em leque”).

Tabela 13: Pressupostos LINE da regressão linear

A estes junta-se, tipicamente, a **ausência de multicolinearidade severa** entre variáveis independentes (ver adiante).

6.4 Avaliação do modelo: R^2 e RMSE

Referência de apoio: documentação scikit-learn, `r2_score` (ver referências).

O **coeficiente de determinação**, R^2 , mede a proporção da variância da variável dependente que é explicada pelo modelo:

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

No ajuste OLS com termo independente, avaliado nos próprios dados de treino e com resposta não constante, R^2 situa-se entre 0 e 1 e não diminui ao adicionar preditores. Em teste, ou noutras condições, pode ser negativo: o erro é então superior ao da previsão constante pela média do conjunto avaliado. O **R^2 ajustado** penaliza a inclusão de variáveis no ajuste:

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

onde n é o número de observações e p o número de variáveis independentes. O erro de previsão é ainda frequentemente reportado através do **Root Mean Squared Error** (RMSE):

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}}$$

que tem a vantagem, sobre o R^2 , de estar na mesma unidade da variável prevista — facilitando a interpretação prática do erro.

6.5 Multicolinearidade

Multicolinearidade. Situação em que duas ou mais variáveis independentes estão fortemente correlacionadas entre si, tornando instável a estimação individual dos seus coeficientes (pequenas mudanças nos dados podem alterar drasticamente os coeficientes, ou até inverter o seu sinal).

O diagnóstico mais comum é o **Fator de Inflação da Variância** (*Variance Inflation Factor*, VIF) de cada atributo — valores de VIF acima de 5 ou 10 são tipicamente considerados problemáticos, sugerindo a remoção ou combinação de atributos redundantes.

6.6 Regularização: Ridge, Lasso e Elastic Net

Quando existem muitas variáveis independentes (possivelmente correlacionadas entre si), a regressão linear “pura” (OLS) pode sofrer de *overfitting*. As técnicas de **regularização** acrescentam uma penalização à magnitude dos coeficientes, ao critério de otimização:

Método	Penalização adicionada	Efeito
Ridge (L2)	$\lambda \sum_j m_j^2$	Encolhe todos os coeficientes em direção a zero, sem os eliminar — útil quando todos os atributos são potencialmente relevantes, mas correlacionados.
Lasso (L1)	$\lambda \sum_j m_j $	Pode reduzir coeficientes exatamente a zero — funciona como seleção automática de atributos.
Elastic Net	Combinação ponderada de L1 e L2.	Compromisso entre os dois: seleção de atributos (como o Lasso) mantendo alguma estabilidade (como o Ridge) quando há grupos de atributos correlacionados.

Tabela 14: Comparação de técnicas de regularização em regressão

Nestas fórmulas, λ (lambda) é um hiperparâmetro que controla a força da penalização — quanto maior, mais os coeficientes são “encolhidos” em direção a zero, reduzindo o risco de overfitting à custa de maior enviesamento (*bias*).

6.7 Interpretação de coeficientes e p-values

- O **coeficiente** descreve a variação da média prevista por unidade do atributo, mantendo os restantes preditores constantes. Esta associação condicional não constitui, por si só, um efeito causal.
- O **valor-p** quantifica, sob a hipótese nula de coeficiente igual a zero e os pressupostos do teste, a probabilidade de obter uma estatística tão ou mais extrema do que a observada. Não é a probabilidade de a hipótese ser verdadeira. Um limiar como 0,05 deve ser previamente justificado; intervalos de confiança, dimensão da associação e comparações múltiplas também importam.

Nota metodológica

Um atributo estatisticamente significativo não é necessariamente relevante na prática: com amostras muito grandes, até efeitos muito pequenos (sem relevância prática) se tornam estatisticamente significativos. É sempre importante olhar também para a magnitude do coeficiente, não apenas para o seu p-value.

CAPÍTULO 7

Classificação: Fundamentos

Objetivos de aprendizagem

- Explicar os critérios de divisão em árvores de decisão (Gini e entropia).
- Comparar algoritmos de classificação: árvores, Naive Bayes, ensembles (Random Forest, XGBoost) e SVM.
- Definir formalmente as métricas de avaliação de classificadores e a validação cruzada.
- Explicar o compromisso bias-variância e a curva ROC.

7.1 Classificação: conceito

Classificação. Problema de aprendizagem supervisionada em que o objetivo é prever a que classe (categoria discreta) pertence uma nova observação, com base num conjunto de atributos.

7.2 Árvores de decisão

As **árvores de decisão** são constituídas por **nós** (que testam um atributo) e **folhas** (que atribuem uma classe), ligados por ramos. Em cada nó, o algoritmo escolhe o atributo (e o ponto de corte, se numérico) que melhor separa as classes — segundo um critério de **impureza**.

Impureza de Gini. Para um conjunto de dados com k classes, e p_i a proporção de observações da classe i :

$$\text{Gini} = 1 - \sum_{i=1}^k p_i^2$$

Varia entre 0 (conjunto puro) e $1 - \frac{1}{k}$ (classes equilibradas); no caso binário, o máximo é 0,5. O algoritmo escolhe a divisão que minimiza a impureza de Gini ponderada dos subconjuntos resultantes.

Entropia e Ganho de Informação. A entropia mede o grau de “desordem” de um conjunto:

$$\text{Entropia} = - \sum_{i=1}^k p_i \log_2(p_i)$$

O **ganho de informação** de uma divisão é a redução de entropia entre o conjunto original e os subconjuntos resultantes — o algoritmo escolhe a divisão com maior ganho de informação.

Gini e entropia produzem, na prática, árvores muito semelhantes — a diferença está sobretudo no custo computacional (Gini evita o cálculo de logaritmos, sendo ligeiramente mais rápida) e em detalhes de sensibilidade a classes muito desequilibradas.

7.2.1 Overfitting e pruning

Uma árvore sem restrições pode crescer até produzir folhas puras, quando os dados e os critérios de divisão o permitem — o que tipicamente resulta em **overfitting** (memorização do ruído dos dados de treino, com fraca generalização a dados novos). O **pruning** (poda) combate este efeito:

- **Pre-pruning:** impor limites durante a construção da árvore (profundidade máxima, número mínimo de observações por folha, ganho mínimo de informação para dividir).
- **Post-pruning:** construir a árvore completa e, depois, remover ramos que não melhorem significativamente o desempenho num conjunto de validação.

7.3 Comparação de algoritmos de classificação

Algoritmo	Princípio	Quando usar
Árvore de decisão	Divisões sucessivas do espaço de atributos, minimizando impureza.	Interpretabilidade elevada; dados de tipos mistos; risco de overfitting sem poda.
Naive Bayes	Aplica o Teorema de Bayes assumindo independência condicional entre atributos, dada a classe.	Rápido, funciona bem com muitos atributos (ex.: classificação de texto); a suposição de independência raramente é realista, mas o modelo é surpreendentemente robusto na prática.
Random Forest	<i>Ensemble</i> de várias árvores de decisão, treinadas em amostras aleatórias dos dados e dos atributos (<i>bagging</i>), com voto/média final.	Reduz a variância (overfitting) de uma árvore única; permite analisar importâncias de atributos, que não equivalem a explicações causais; mais lento e menos interpretável do que uma única árvore.
XGBoost / Gradient Boosting	<i>Ensemble</i> sequencial: cada nova árvore corrige os erros das anteriores (<i>boosting</i>).	Frequentemente competitivo em dados tabulares; desempenho e custo dependem dos dados, da implementação e da afinação dos hiperparâmetros.
SVM (<i>Support Vector Machine</i>)	Procura o hiperplano que maximiza a margem entre classes; com o “kernel trick”, pode capturar fronteiras não-lineares.	Eficaz em espaços de alta dimensão; menos intuitivo de interpretar; sensível à escolha de kernel e parâmetros.

Tabela 15: Comparação de algoritmos de classificação

Bagging vs. Boosting.

- **Bagging** (*Bootstrap Aggregating*, usado no Random Forest): treina modelos em paralelo, sobre amostras aleatórias dos dados (com reposição), e combina-os por votação/média. Reduz sobretudo a **variância**.
- **Boosting**: combina modelos sequenciais. O AdaBoost repondera observações; o gradient boosting ajusta novos modelos a uma direção de melhoria da função de perda. O comportamento depende da perda, da complexidade e da regularização.

7.4 O compromisso Bias-Variância

Compromisso Bias-Variância. Na regressão com perda quadrática, sob o modelo habitual de erro aditivo com média zero, o erro esperado de previsão num ponto decompõe-se em:

$$\text{Erro} = \text{Bias}^2 + \text{Variância} + \text{Erro irreduzível}$$

- **Bias** (enviesamento): erro sistemático por o modelo ser demasiado simples para capturar a verdadeira relação (*underfitting*).
- **Variância**: sensibilidade excessiva do modelo às particularidades dos dados de treino (*overfitting*).

Modelos mais complexos (árvores profundas, muitos parâmetros) tendem a ter baixo bias mas alta variância; modelos mais simples (regressão linear, árvores rasas), o inverso. Técnicas como o pruning, a regularização e os ensembles procuram, cada uma à sua maneira, encontrar um bom equilíbrio.

7.5 Validação cruzada

Validação cruzada (k-fold). Técnica de avaliação que divide o dataset em k partições (*folds*) de tamanho semelhante. O modelo é treinado k vezes, usando sempre $k - 1$ partições para treino e a partição restante para teste (rodando qual delas é usada para teste em cada iteração). O desempenho final é a média (e desvio-padrão) das k avaliações.

Face a uma única divisão treino/teste (*Split Data/holdout*), a validação cruzada dá uma estimativa mais robusta e menos dependente de uma divisão aleatória “sortuda” ou “azarada” dos dados — ao custo de k vezes mais tempo de treino.

Nota metodológica

A escolha de modelos e hiperparâmetros deve usar apenas o conjunto de desenvolvimento, por exemplo através de validação cruzada. O conjunto de teste fica reservado para a avaliação final. Se existirem várias observações do mesmo doente, separar por doente; em previsão temporal, respeitar a ordem temporal. Divisões aleatórias por linha podem produzir estimativas excessivamente otimistas.

7.6 Métricas de avaliação

A partir da **matriz de confusão** — que cruza as classes previstas com as classes reais — calculam-se as métricas de desempenho mais comuns, usando as contagens de verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP) e falsos negativos (FN):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

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

$$\text{Recall (V Sensitivity)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

O **F1-score** é a média harmónica entre precisão e recall, sendo especialmente útil quando se quer um único número que equilibre ambas as métricas (em vez de otimizar apenas uma, à custa da outra).

Nota metodológica

Nenhuma destas métricas deve ser lida isoladamente. Num dataset com classes desequilibradas (por exemplo, 95% de casos negativos), um modelo que preveja sempre “negativo” tem 95% de *accuracy* — mas **recall** nulo para a classe positiva, o que o torna inútil na prática se o objetivo for detetar os casos positivos (por exemplo, deteção de doença).

7.7 Curva ROC e AUC

A curva ROC representa a sensibilidade em função da taxa de falsos positivos (1 – especificidade), variando o limiar de um score. A área sob a curva (ROC-AUC) resume a ordenação: 1 corresponde a separação perfeita; 0,5, a ausência de capacidade discriminativa global. Um valor inferior a 0,5 exige verificar a orientação do score, as classes e a variabilidade amostral.

Mantidas as distribuições condicionais dos scores, as taxas ROC não dependem diretamente da prevalência. Contudo, uma ROC-AUC elevada pode coexistir com muitos falsos positivos numa classe rara. Complementar com precisão-recall, prevalência e desempenho no limiar escolhido. A ROC-AUC não avalia a calibração das probabilidades.

CAPÍTULO 8

Extração de Conhecimento em Código: Fundamentos

Objetivos de aprendizagem

- Comparar ferramentas visuais (no-code/low-code) com fluxos de trabalho baseados em código.
- Discutir reprodutibilidade, controlo de versão e automação em projetos de Data Mining.
- Compreender o papel de um *pipeline* e a diferença entre notebooks exploratórios e scripts de produção.

8.1 Ferramentas visuais vs. código

As sessões anteriores desta unidade curricular usaram o AI Studio (RapidMiner) como ferramenta visual — cada etapa da metodologia CRISP-DM é construída ligando operadores num fluxo, o que torna o processo explícito e fácil de inspecionar visualmente. A generalidade dos projetos reais de Data Mining e Machine Learning, no entanto, acaba por ser implementada em **código**, sobretudo em Python.

Critério	Ferramentas visuais (AI Studio)	Código (Python)
Curva de aprendizagem	Baixa — ideal para exploração inicial e não-programadores.	Mais elevada — requer conhecimento de programação.
Reprodutibilidade	Processos exportáveis, mas menos fáceis de versionar linha-a-linha.	Scripts/notebooks controlados por Git, com histórico completo de alterações.
Automação e integração	Limitada fora do próprio ambiente.	Fácil de integrar em pipelines de produção, APIs, tarefas agendadas (<i>cron</i>), CI/CD.
Flexibilidade algorítmica	Restrita aos operadores disponíveis (ou extensões).	Acesso direto a qualquer biblioteca (<i>scikit-learn</i> , <i>PyTorch</i> , etc.) e a algoritmos personalizados.
Escalabilidade	Pode ser limitada para datasets muito grandes.	Integração natural com ferramentas de big data (Spark, Dask).

Tabela 16: Ferramentas visuais vs. fluxos de trabalho em código

Na prática, muitas equipas usam ambas as abordagens de forma complementar: ferramentas visuais para exploração rápida e comunicação com stakeholders não-técnicos, e código para os modelos que efetivamente chegam a produção.

8.2 Reprodutibilidade

Reprodutibilidade. Possibilidade de repetir um procedimento com os mesmos dados, código e condições documentadas, obtendo resultados compatíveis dentro da tolerância definida.

Em Machine Learning, a reprodutibilidade exige atenção especial a fontes de aleatoriedade: divisão treino/teste, inicialização de modelos, ordem de processamento em paralelo, etc. A prática recomendada é fixar explicitamente uma **semente aleatória** (*random seed*, por exemplo `random_state=42` em *scikit-learn*), reduzindo a variabilidade entre execuções. A semente, por si só, não garante identidade entre versões, plataformas ou operações paralelas — essencial tanto para depuração como para comparação justa entre modelos.

8.3 Pipelines e separação de responsabilidades

Pipeline (de Machine Learning). Sequência encadeada e reutilizável de transformações de dados e etapas de modelação, tratada como uma única unidade — desde os dados brutos até à previsão final.

Um pipeline bem construído (por exemplo, com `sklearn.pipeline.Pipeline`) encapsula os passos de pré-processamento (normalização, imputação, codificação) juntamente com o modelo, garantindo que exatamente as mesmas transformações aprendidas nos dados de treino são aplicadas aos dados de teste ou de produção — evitando um erro metodológico comum: o **vazamento de dados** (*data leakage*), em que informação dos dados de teste “contamina” indevidamente o treino (por exemplo, normalizar todo o dataset antes de dividir em treino/teste).

8.4 Notebooks vs. scripts

Formato	Uso típico
Notebooks (Jupyter, Google Colab)	Exploração interativa, visualização imediata de resultados, comunicação e ensino — a execução célula-a-célula facilita a experimentação, mas dificulta a reprodutibilidade se as células forem executadas fora de ordem.
Scripts Python (.py)	Código de produção: mais fácil de testar automaticamente, versionar, agendar e integrar noutros sistemas.

Tabela 17: Notebooks vs. scripts como formato de trabalho

CAPÍTULO 9

Clustering: Fundamentos

Objetivos de aprendizagem

- Definir formalmente a função-objetivo do k-Means (inércia / WCSS).
- Explicar o algoritmo k-Means passo a passo, e a inicialização k-means++.
- Comparar métodos para escolher o número de clusters k .
- Comparar k-Means com métodos alternativos de clustering (hierárquico, DBSCAN, GMM).

9.1 O algoritmo k-Means

k-Means. Algoritmo de agrupamento (*clustering*) não supervisionado que particiona um dataset em k grupos (clusters), minimizando a soma das distâncias quadráticas entre cada ponto e o centróide do seu cluster.

Formalmente, o k-Means procura minimizar a **inércia** (também chamada *Within-Cluster Sum of Squares*, WCSS):

$$\text{WCSS} = \sum_{j=1}^k \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

onde C_j é o conjunto de pontos atribuídos ao cluster j , e μ_j é o centróide (a média) desse cluster. O algoritmo funciona de forma iterativa (algoritmo de Lloyd):

1. **Inicialização:** escolhem-se k pontos como centróides iniciais.
2. **Atribuição:** cada ponto é atribuído ao cluster cujo centróide está mais próximo (tipicamente por distância euclidiana).
3. **Atualização:** recalcula-se cada centróide como a média dos pontos atribuídos a ele.
4. **Repetição:** os passos 2 e 3 repetem-se até os centróides estabilizarem (convergência) ou até um número máximo de iterações.

O algoritmo garante convergência (a inércia nunca aumenta entre iterações), mas apenas para um **mínimo local** — não garante encontrar a solução ótima global.

9.2 Inicialização: k-means++

A escolha dos centróides iniciais afeta fortemente o resultado final. A inicialização aleatória simples pode gerar clusters de má qualidade, ou exigir muitas repetições para encontrar uma boa solução. **k-means++.** Estratégia de inicialização em que o primeiro centróide é escolhido aleatoriamente, e cada centróide seguinte é escolhido com probabilidade proporcional ao quadrado da distância ao centróide mais próximo já escolhido — favorecendo centróides iniciais bem espalhados pelo espaço de dados.

Esta estratégia (por omissão em `scikit-learn`) reduz significativamente a probabilidade de convergência para um mau mínimo local, face à inicialização puramente aleatória. Na prática, executa-se ainda o algoritmo várias vezes com inicializações diferentes (parâmetro `n_init`), escolhendo o resultado com menor inércia final.

9.3 Escolha do número de clusters k

Ao contrário da classificação, não existe uma “resposta certa” óbvia para o número de clusters — é necessário um critério objetivo:

Método	Princípio
Método do cotovelo (<i>Elbow Method</i>)	Representa a inércia (WCSS) para vários valores de k ; escolhe-se o k a partir do qual o ganho marginal de reduzir a inércia começa a diminuir claramente (o “cotovelo” da curva).
Coeficiente de silhueta (<i>Silhouette Score</i>)	Para cada ponto, compara a distância média aos pontos do seu próprio cluster com a distância média ao cluster vizinho mais próximo. Varia entre -1 e 1 ; valores mais altos indicam clusters mais bem separados e coesos.
Índice de Davies-Bouldin	Mede a razão entre a dispersão dentro de cada cluster e a separação entre clusters. Ao contrário da silhueta, quanto menor , melhor a qualidade do agrupamento.
Estatística do Gap (<i>Gap Statistic</i>)	Compara a inércia observada com a inércia esperada sob uma distribuição de referência aleatória (sem estrutura de clusters) — uma regra habitual escolhe o menor k cujo gap é pelo menos o gap de $k + 1$ menos a respetiva incerteza estimada; maximizar o gap não é a única regra.

Tabela 18: Métodos para escolher o número de clusters k

Coeficiente de silhueta (fórmula). Para um ponto i , sejam $a(i)$ a distância média aos restantes pontos do seu cluster, e $b(i)$ a menor distância média aos pontos de cada um dos outros clusters:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

9.4 Comparação com outros métodos de clustering

O k-Means assume implicitamente clusters de forma aproximadamente esférica e de tamanhos semelhantes, e é sensível a outliers (que distorcem a média usada como centróide). Diferentes problemas beneficiam de algoritmos alternativos:

Método	Princípio	Quando preferir a k-Means
Clustering Hierárquico	Constrói uma árvore de agrupamentos sucessivos (<i>dendrograma</i>), aglomerativa (de baixo para cima) ou divisiva (de cima para baixo).	Quando não se sabe k à partida, ou se quer explorar a estrutura a múltiplos níveis de granularidade; mais lento em datasets grandes.
DBSCAN	Agrupa por densidade: pontos em regiões densamente povoadas formam clusters; pontos isolados são marcados como ruído.	Quando os clusters têm formas irregulares (não convexas) ou tamanhos muito diferentes, ou quando existem outliers claros a identificar como ruído.
Gaussian Mixture Models (GMM)	Modela os dados como uma mistura de distribuições gaussianas, atribuindo a cada ponto uma probabilidade de pertença a cada cluster (<i>soft clustering</i>), em vez de uma atribuição rígida.	Quando se espera sobreposição entre clusters, ou se pretende uma medida de incerteza na atribuição.

Tabela 19: k-Means vs. métodos alternativos de clustering

9.5 Distâncias

O k-Means clássico minimiza distâncias euclidianas quadráticas e atualiza os centros por médias. As seguintes distâncias são relevantes em agrupamento, mas não podem ser trocadas livremente mantendo o mesmo algoritmo:

Distância	Fórmula / uso
Euclidiana	$\sqrt{\sum_i (x_i - y_i)^2}$ — distância “em linha reta”; a mais comum, mas sensível a diferenças de escala entre atributos (daí a importância de normalizar antes de aplicar k-Means).
Manhattan	$\sum_i x_i - y_i $ — soma das diferenças absolutas; objetivos baseados nesta distância requerem atualizações adequadas, como medianas, em vez das médias do k-Means clássico.
Cosseno	Mede o ângulo entre dois vetores, ignorando a sua magnitude — muito usada em dados de texto ou alta dimensionalidade.

Tabela 20: Métricas de distância usadas em clustering

Nota metodológica

As médias usadas para calcular os centróides são particularmente suscetíveis à influência de valores extremos — é importante tratar outliers e normalizar os dados antes de aplicar k-Means.

PARTE II

Exercícios

CAPÍTULO 10

Exercícios — Introdução à Metodologia CRISP-DM

Este capítulo aplica a teoria do primeiro capítulo (Data Mining e a metodologia CRISP-DM) a um exercício de análise de um problema à escolha, estruturado pelas seis fases do CRISP-DM.

Ficha 01 • Introdução à Metodologia CRISP-DM

Identifique um problema que possa ser enquadrado no processo de Data Mining. Para esse problema, descreva sucintamente as seguintes fases do processo CRISP-DM:

1. Business Understanding — qual o objetivo de negócio? Qual o critério de sucesso?
2. Data Understanding — que dados seriam necessários? Que fontes de dados existiriam?
3. Data Preparation — que passos de limpeza, seleção ou construção de atributos seriam previsíveis?
4. Modeling — que técnica(s) de Data Mining (das apresentadas no Capítulo 1) se aplicariam ao problema escolhido, e porquê?
5. Evaluation — como se avaliaria se o modelo cumpre o objetivo de negócio definido em 1?
6. Deployment — como se colocaria o conhecimento obtido em prática?

Nota metodológica

Sugestão de trabalho: escolha um problema realista do domínio da engenharia biomédica ou saúde (por exemplo, previsão de readmissões hospitalares, deteção de eventos adversos, ou triagem de doentes) para tornar o exercício mais próximo do contexto da unidade curricular.

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

CAPÍTULO 11

Exercícios — AI Studio: Correlações

11.1 O AI Studio (RapidMiner)

O **AI Studio** — anteriormente conhecido por **RapidMiner Studio** — é uma ferramenta comercial de análise de dados que utiliza Machine Learning, e pode ser vista como uma alternativa a ferramentas como o Weka. Ao longo desta unidade curricular, será a ferramenta principal de trabalho: os processos de Data Mining são construídos visualmente, ligando **operadores** (blocos funcionais) num fluxo de dados, o que torna explícita cada etapa da metodologia CRISP-DM.

Nota metodológica

O download do AI Studio requer registo/login (usar o login de aluno). Consultar: <https://docs.rapidminer.com/latest/studio/installation/index.html>.

11.2 Caso prático guiado

Caso de estudo · Consumo de gasóleo de aquecimento

Contexto: a Sara é gerente regional de vendas de um fornecedor nacional de combustíveis fósseis para aquecimento doméstico. A recente volatilidade dos preços do gasóleo de aquecimento, e a grande variabilidade no tamanho de cada pedido, preocupam-na. Que fatores influenciam a procura por gasóleo de aquecimento?

Business Understanding: o objetivo da Sara é compreender melhor os fatores que influenciam o consumo de gasóleo de aquecimento, de forma a monitorizar e responder melhor à procura. A correlação foi escolhida como forma de modelar a relação entre esses fatores.

Data Understanding: a partir da base de dados de cobrança da empresa, foi construído um dataset com os seguintes atributos:

- **Insulation** — classificação de densidade de isolamento da casa (1 a 10).
- **Temperature** — temperatura ambiente média externa (°F).
- **Heating_Oil** — número total de unidades de óleo de aquecimento adquiridas no último ano.
- **Num_Occupants** — número de ocupantes da casa.
- **Avg_Age** — idade média dos ocupantes.
- **Home_Size** — classificação do tamanho da casa (1 a 8).

Data Preparation: importar o ficheiro `pl03-dataset.csv` para o repositório do AI Studio (Import Data → My Computer) e inspecionar os dados nas vistas **Data** e **Statistics**.

Modeling: na perspetiva de *Design*, procurar o operador **Correlation Matrix** na caixa de pesquisa de operadores, arrastá-lo para a janela de processo, ligar o dataset à sua entrada e executar (*Run*).

Evaluation: os materiais originais indicam os seguintes valores, a confirmar na execução do ficheiro:

- **Heating_Oil × Insulation:** correlação positiva de 0.736.
- **Heating_Oil × Avg_Age:** a correlação mais forte do estudo, 0.848 (positiva).
- **Avg_Age × Temperature:** correlação negativa forte, -0.673.

Deployment: algumas leituras possíveis dos resultados:

- O número de ocupantes da casa, apesar de intuitivamente relevante, não apresentou, no exemplo original, associações lineares descritas como fortes; sem teste e dimensão amostral não se deve afirmar significância estatística.
- O isolamento correlacionou-se com vários outros atributos — o que justifica investigar a escala da variável, confundimento e hipóteses alternativas antes de propor intervenções.
- O dataset tem uma granularidade temporal baixa (temperatura média **anual**); dados mensais ou semanais dariam resultados mais próximos da realidade.

Questões de reflexão

A correlação entre `Avg_Age` e `Heating_Oil` é forte e positiva — mas será razoável concluir que a idade dos ocupantes de uma casa **causa** um maior consumo de gasóleo? E a correlação negativa entre `Avg_Age` e `Temperature` — fará sentido lógico que a idade dos moradores influencie a temperatura externa da casa? (Ver o Capítulo 2, secção “Correlação não implica causalidade”).

Ficha 02 · Introdução ao AI Studio — Correlações

Parte I

1. Quais as principais limitações de modelos de correlações?
2. O que é um coeficiente de correlação e como é interpretado?
3. Qual a diferença entre uma correlação negativa e uma correlação positiva?
 - Se dois atributos diminuem essencialmente à mesma taxa, é uma correlação positiva ou negativa? Explique.
4. Como é medida a força de uma correlação? Quais os limites para essa força?
5. Consegue pensar em atributos que poderiam ser interessantes incluir no dataset estudado no exemplo desta aula?

Parte II

Aceda ao ficheiro `mpg_dataset.csv`, com os seguintes atributos: `cylinders` (número de cilindros), `displacement` (cilindrada), `horsepower` (potência), `weight` (peso do veículo), `acceleration` (tempo em segundos para acelerar de 0 a 60), `model year` (ano do modelo, séc. XX), `origin` (1: Americana, 2: Europeia, 3: Japonesa) e `mpg` (consumo/eficiência de combustível).

1. Execute a etapa de Data Preparation no AI Studio: importe o csv e acrescente-o ao design.
2. Documente quais os atributos que podem influenciar ou explicar o consumo/eficiência de combustível (`mpg`) de um veículo, com base na matriz de correlação obtida.

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

CAPÍTULO 12

Exercícios — Recolha de Dados via API

12.1 Exemplo de referência: dados de doentes e registos clínicos

O código seguinte extrai três recursos JSON e conserva uma cópia local. Não cria tabelas nem efetua inserções. O esquema SQL apresentado a seguir é uma proposta para a segunda etapa do exercício, cujo carregamento deve ser implementado pelo estudante.

```
from pathlib import Path
import json
import requests

FONTES = {
    "cod_postal": "https://hpeixoto.me/cod_postal.json",
    "doentes": "https://hpeixoto.me/doentes.json",
    "registos": "https://hpeixoto.me/registos.json",
}

def obter_json(url):
    resposta = requests.get(url, timeout=30)
    resposta.raise_for_status()
    dados = resposta.json()
    if not isinstance(dados, (list, dict)):
        raise ValueError("Esperava-se um array ou objeto JSON.")
    return dados

def main():
    destino = Path("dados_extraidos")
    destino.mkdir(exist_ok=True)
    for nome, url in FONTES.items():
        dados = obter_json(url)
        # Conserva a estrutura recebida; validar o esquema antes do CSV.
        temporario = destino / f"{nome}.json.tmp"
        temporario.write_text(
            json.dumps(dados, ensure_ascii=False, indent=2),
            encoding="utf-8",
        )
        temporario.replace(destino / f"{nome}.json")
        print(f"Recurso guardado: {nome}")

if __name__ == "__main__":
    main()
```

O esquema de referência seguinte deve ser confrontado com a estrutura efetivamente recebida. Os tipos e nomes são uma proposta de trabalho, não um contrato validado dos recursos remotos:

```
CREATE TABLE cod_postal (
    cod_postal VARCHAR(10) PRIMARY KEY,
    localidade VARCHAR(100)
);

CREATE TABLE doentes (
    id_paciente INT PRIMARY KEY,
    nome VARCHAR(100),
    data_nascimento DATE,
    sexo VARCHAR(10),
    cod_postal VARCHAR(10),
    FOREIGN KEY (cod_postal) REFERENCES cod_postal(cod_postal)
);

CREATE TABLE registos (
    id_registo VARCHAR(100) PRIMARY KEY,
    id_paciente INT NOT NULL,
    data_registo DATE,
    temperatura VARCHAR(30),
    dor_cabeca VARCHAR(30),
    dor_muscular VARCHAR(30),
    tosse VARCHAR(30),
```

```

diarreia VARCHAR(30),
olfato_paladar VARCHAR(30),
avaliacao_global VARCHAR(30),
FOREIGN KEY (id_paciente) REFERENCES doentes(id_paciente)
);

```

Nota metodológica

Para o carregamento, usar instruções parametrizadas, transações e uma chave estável por registo da origem. Um identificador local auto-incrementado não evita duplicações entre importações. Se a origem não fornecer ID, definir e justificar uma chave de negócio; não presumir que doente e data identificam um único evento. Validar as chaves estrangeiras antes de confirmar a transação. Não completar códigos postais de quatro algarismos com “000”: isso inventaria informação. Guardar o valor original, validar os sete algarismos quando disponíveis e registar valores incompletos como tal.

Boas práticas de segurança: credenciais de acesso a bases de dados nunca devem ficar escritas diretamente no código-fonte. A abordagem recomendada é usar variáveis de ambiente (por exemplo, com a biblioteca `python-dotenv`) ou um gestor de segredos.

Ficha 03 • Data Collection from API

Contexto: considere que exerce funções de analista de dados num projeto para estudar o acompanhamento de pacientes com COVID-19, monitorizados em casa através de uma aplicação web. Esta aplicação disponibiliza uma API REST que permite aceder e recuperar dados relevantes. O objetivo é aceder a esta fonte, obter os dados e prepará-los para análise futura.

Verificar a fonte de dados

1. Verificar a disponibilidade dos ficheiros JSON fornecidos pela API REST:
 - Abrir o navegador e visitar https://hpeixoto.me/cod_postal.json, observando a resposta.
 - Repetir para <https://hpeixoto.me/doentes.json> e <https://hpeixoto.me/registos.json>.

Configurar o ambiente Python

1. Preparar o ambiente Python necessário (instalação das bibliotecas `requests` e de um conector para a base de dados escolhida).

Obter dados das APIs

1. Criar um script Python para obter os dados JSON das três APIs e guardá-los como ficheiros CSV.

Criação de base de dados centralizada

1. Criar o modelo relacional (seguindo o exemplo apresentado neste capítulo, ou um desenho próprio) e carregar os dados para a base de dados criada (exemplo: MySQL).

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

CAPÍTULO 13

Exercícios — Preparação de Dados no AI Studio

13.1 Operadores de referência

- **Join:** combina dois datasets a partir de um atributo-chave comum (ex.: `Products + Transactions`, via `Product ID`).
- **Aggregate:** agrupa registos (*group by*) e calcula uma função de agregação — `count`, `mode`, etc.
- **Generate Attributes:** cria novos atributos a partir de expressões (ex.: `Total = Amount * Price`).
- **Select Attributes:** mantém/remove um subconjunto de atributos (`attribute filter type = subset`).
- **Replace Missing Values / Filter Examples:** substituem ou removem valores/exemplos em falta.
- **Normalize:** reescala atributos (ver Capítulo 4 para as fórmulas).
- **Detect Outlier (Distances) / Detect Outlier (LOF):** identificam outliers (ver Capítulo 4).

Questões de reflexão

Depois de criar o atributo `Total` ($\text{Amount} \times \text{Price}$), como descreveria a forma da sua distribuição? Um histograma ou a aba de Estatísticas do AI Studio ajudam a responder — distribuições fortemente assimétricas (*skewed*) são um sinal comum quando se combinam quantidade e preço unitário.

Ficha 04 · Data Preparation

1. Juntar e agrupar dados

1. Colocar no workflow os datasets de exemplo `Products` e `Transactions` (repositório `Samples`).
2. Juntar os dois datasets com o operador **Join**, usando `Product ID` como atributo-chave.
3. Agrupar os dados com o operador **Aggregate**, ligado à saída do `Join`:
 - `Customer ID` com a função `count`.
 - `Product Name` com a função `mode`.
 - Agrupar por (*group by*) `Product ID`.

Questões de reflexão

Na vista de Resultados, é possível descobrir o ID do cliente que pagou mais por um único produto? Quanto pagou? É possível responder à mesma questão usando apenas operadores (sem inspeção manual)?

2. Criar e remover atributos

1. Remover o operador **Aggregate** do processo anterior.
2. Adicionar um operador **Generate Attributes**, ligado ao `Join`, criando o atributo `Total = Amount * Price`.
3. Adicionar um operador **Select Attributes** (tipo `subset`) para manter apenas `Customer ID`, `Product Name` e `Total`.
4. Executar o processo.

3. Lidar com valores nulos

1. Carregar o dataset `Titanic` e inspecionar as colunas com valores em falta.
2. Remover os atributos `Cabin` e `Life Boat` (elevada proporção de valores em falta) com **Select Attributes**.
3. Substituir os valores em falta no atributo `Age` com **Replace Missing Values** (filtro `single`).
4. Remover exemplos remanescentes com valores em falta, usando **Filter Examples** (`no_missing_attributes`).

Questões de reflexão

Depois destes três passos, a aba de Estatísticas ainda mostra valores em falta nalguma coluna? Porque não seria boa ideia filtrar logo os exemplos, em vez de primeiro remover atributos e substituir valores na coluna *Age*? Quantos exemplos restam no dataset?

4. Normalização e deteção de outliers

1. Carregar novamente o dataset *Titanic* e remover *Cabin*, *Life Boat*, *Name* e *Ticket Number*.
2. Selecionar os atributos quantitativos adequados à distância escolhida; excluir identificadores e o desfecho. Normalizar apenas os atributos quantitativos selecionados com **Normalize**.
3. Detetar outliers com **Detect Outlier (Distances)** (configurações padrão).
4. Criar uma variante que exclui os casos assinalados com **Filter Examples** (*Outlier equals false*) e comparar com os dados completos. Justificar cada exclusão; uma observação rara pode ser válida.

Questões de reflexão

Como alteraria o processo para encontrar 20 outliers em vez de 10? Como o alteraria para mostrar apenas os outliers, em vez de os remover? Que diferenças observa ao substituir o operador de deteção de outliers por *Detect Outlier (LOF)*?

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

CAPÍTULO 14

Exercícios — AI Studio: Regras de Associação

14.1 Caso prático guiado

Caso de estudo · Organizações comunitárias

Contexto: o Pedro é gestor municipal de uma cidade em crescimento, com recursos limitados. Sabe que os cidadãos são ativos em organizações comunitárias (igrejas, clubes sociais, grupos de interesse) e acredita que estes grupos podem colaborar para responder a necessidades da comunidade — mas precisa primeiro de descobrir se existem associações naturais entre os diferentes tipos de grupos.

Data Understanding: através de um questionário online, foi construído um dataset com 12 atributos, incluindo `Time_in_Community`, `Gender`, `Working`, `Age`, e um conjunto de atributos sim/não indicando pertença a organizações: `Family`, `Hobbies`, `Social_Club`, `Political`, `Professional`, `Religious`, `Support_Group`.

Data Preparation:

1. Importar o dataset (`pl06-dataset.csv`) e inspecionar estatísticas — sem valores em falta, 12 atributos.
2. Os atributos sim/não estão codificados como inteiros (0/1); os operadores de regras de associação exigem o tipo **binominal** — usar o operador **Numerical to Binomial** sobre os atributos relevantes.
3. Reduzir o dataset aos atributos de interesse com **Select Attributes**: `Family`, `Hobbies`, `Social_Club`, `Political`, `Professional`, `Religious`, `Support_Group`.

Modeling:

1. Aplicar o operador **FP-Growth** sobre o dataset preparado, com um min support inicial.
2. Ligar as portas `exa` e `fre` às portas de resultado (`exa`: exemplos e estatísticas; `fre`: matriz de padrões frequentes).
3. Aplicar o operador **Create Association Rules** sobre a matriz de padrões, com min confidence = 0.5.

Evaluation: nos resultados descritos no material original, não reproduzidos nesta revisão, as regras sugeriram associações compatíveis com a hipótese inicial — as organizações religiosas têm ligações naturais com organizações de família e de hobbies. Uma das regras apresentou confiança de aproximadamente 80%, e todas as regras eram suportadas por mais de 20% das observações.

Deployment: as associações podem orientar uma investigação posterior sobre colaboração entre organizações ligadas a igrejas, família e hobbies — os grupos onde a análise revelou sobreposição de membros.

Nota metodológica

O processo CRISP-DM é de natureza cíclica: é normal que uma primeira tentativa de modelação não produza qualquer regra (por exemplo, com um suporte ou confiança mínimos demasiado exigentes) — nesse caso, o passo seguinte não é desistir, mas voltar atrás e ajustar os parâmetros ou reconsiderar a preparação dos dados.

Ficha 05 · AI Studio — Regras de Associação

Parte I

1. O que são regras de associação? Para que servem?
2. Quais são as duas principais métricas calculadas em regras de associação, e como são calculadas?

3. De que tipo devem ser os atributos de um dataset para usar os operadores *Frequent Pattern* no AI Studio?
4. Como são interpretados os resultados das regras de associação? No exemplo desta aula, qual foi a regra mais forte, e como se sabe?

Parte II

1. Faça download do dataset `order.csv`, importe-o para o AI Studio e proceda à etapa de Data Understanding.
2. Execute os passos de Data Preparation necessários, garantindo tipos de dados apropriados para o operador FP-Growth.
3. Gere regras de associação com o método FP-Growth:
 - Modifique os valores de confiança e suporte mínimos para encontrar regras interessantes.
 - Analise outras medidas de força, como Laplace ou Lift.
 - Documente as suas descobertas: que regras encontrou? Que atributos estão mais fortemente associados? Alguma regra é suficientemente forte para basear decisões nela?
4. Repita o exercício com o operador **W-FPGrowth** (extensão Weka — instalar via `Extensions` → `Marketplace`). Note que este operador usa os parâmetros `u` (suporte) e `c` (confiança), e cria as suas próprias regras sem operadores adicionais.
5. Explore o algoritmo **Apriori** (W-Apriori) sobre o mesmo dataset, consultando a documentação de ajuda do AI Studio para compreender os seus parâmetros. Apresente e discuta os resultados obtidos.

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

CAPÍTULO 15

Exercícios — AI Studio: Regressão Linear

15.1 Caso prático guiado

Caso de estudo · Previsão do preço de uma casa

Contexto: o Pedro está a pensar mudar de casa e quer ter a certeza de que o preço que vai pagar é adequado. Reuniu um dataset com 120 observações de casas do seu bairro e de bairros vizinhos.

Data Understanding: os atributos disponíveis são `House_sqft` (área), `Num_of_bedrooms`, `Num_of_bathrooms`, `Year_built`, `Tax_assessed_value` e `Last_sold_price` (o atributo-alvo).

Data Preparation:

1. Importar o dataset para o repositório do AI Studio.
2. Definir `last_sold_price` como **label** (atributo-alvo) com o operador **Set Role**.
3. Dividir o dataset em treino (70%) e teste (30%) com o operador **Split Data**.

Modeling:

1. Aplicar o operador **Linear Regression** ao fluxo de treino.
2. Ligar o modelo treinado ao fluxo de teste através do operador **Apply Model**, conectando as portas `lab` e `mod` às portas de resultado (`res`).

Evaluation:

1. O separador de resultados `LinearRegression` mostra, para cada atributo, o seu **coeficiente** e o respetivo **p-value** (ver Capítulo 6).
2. Se um atributo não aparecer no modelo, consultar os parâmetros de seleção, a colinearidade e os registos de execução. A ausência não demonstra, por si só, irrelevância preditiva nem equivale a uma regra universal de remoção por valor-p.
3. Com os coeficientes obtidos (`house_sqft`: 1.737, `num_of_bathrooms`: -1192.239, `year_built`: 44.031, `tax_assessed_value`: 0.996, `intercept` -85701.545), a previsão para uma casa com `house_sqft` = 1770, `num_of_bathrooms` = 2, `year_built` = 1990, `tax_assessed_value` = 195000 é:

$$y = 1.737 \times 1770 - 1192.239 \times 2 + 44.031 \times 1990 + 0.996 \times 195000 - 85701.545 \approx 196830.16$$

Avaliação de desempenho: o operador **Performance** permite calcular o RMSE (ver Capítulo 6). Neste exemplo, obteve-se $RMSE = 1968.844$.

Nota metodológica

Evitar extrapolação não justificada e verificar compatibilidade com a população de desenvolvimento. Estar dentro dos intervalos marginais de treino não garante uma previsão fiável: combinações novas, mudanças de distribuição e incerteza também devem ser avaliadas.

Ficha 06 · AI Studio — Regressão Linear

1. Regressão Linear

Importe o dataset `insurance.csv`.

1. Execute o passo de Data Understanding da metodologia CRISP-DM.
2. Repita os passos descritos nos slides da aula e, após executar o modelo, examine os coeficientes dos atributos e as previsões de custo no cenário de teste.
3. Relate os resultados:
 - Que atributos têm maior peso?
 - Algum atributo foi removido por não ter boa capacidade de previsão? Se sim, quais, e porquê?
 - Que outros atributos poderiam ajudar o modelo a prever melhor o custo anual do seguro de saúde?
4. Utilizando a visualização do AI Studio, descreva o que observa nos gráficos:

- Relação entre sexo e prémio do seguro.
- Relação entre número de dependentes e prémio do seguro.
- Relação entre ser ou não fumador e prémio do seguro.
- Relação entre localização e prémio do seguro.

2. HappyScore

Importe o dataset `happyscore_income.csv` e prepare o processo de Data Mining da forma que considerar mais adequada:

1. Execute os processos de Business e Data Understanding para perceber que processo de DM deve ser aplicado. Inclua:
 - Uma frase que caracterize o objetivo de negócio.
 - Uma frase que caracterize o objetivo do processo de DM.
 - Uma tabela-resumo dos atributos encontrados (tipo, min, max, média, nulos, etc.).
2. Crie o modelo de DM no AI Studio, justificando todos os operadores de preparação de dados utilizados.
3. Descreva as principais conclusões obtidas.

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

CAPÍTULO 16

Exercícios — AI Studio: Classificação

16.1 Caso prático guiado

Caso de estudo · Adoção de um novo eReader

Contexto: o Ricardo trabalha para uma grande loja online que vai lançar um novo eReader. Quer maximizar a efetividade do marketing, distinguindo clientes ansiosos por comprar a nova versão de clientes dispostos a esperar.

Business Understanding: o Ricardo quer prever, para cada cliente, a que categoria de adoção pertencerá — Innovator, Early Adopter, Early Majority ou Late Majority — a partir do comportamento de consumo no site.

Data Understanding: os dados incluem User_ID, Gender, Age, Marital_Status, Website_Activity (Seldom/Regular/Frequent), Browsed_Electronics_12Mo, Bought_Electronics_12Mo, Bought_Digital_Media_18Mo, Bought_Digital_Books, Payment_Method e o atributo-alvo eReader_Adoption.

Data Preparation:

1. Importar o dataset (pl08-dataset.csv) e avaliar os dados importados.
2. Atribuir o papel de **label** (classe) ao atributo eReader_Adoption, e avaliar quais os atributos a retirar do modelo (por exemplo, identificadores como User_ID).
3. Dividir o dataset em treino (70%) e teste (30%) com **Split Data**.

Modeling:

1. Aplicar o operador **Decision Trees** ao fluxo de treino.
2. Ligar o modelo ao fluxo de teste com **Apply Model**, conectando as portas lab e mod a res.

Evaluation:

1. Examinar a árvore de decisão resultante (separador Tree).
2. Adicionar o operador **Performance** para avaliar o modelo através da matriz de confusão.
3. Comparar com outros algoritmos de classificação disponíveis no AI Studio: **Naive Bayes**, **Random Forest**, **W-J48** e **XGBoost** (ver Capítulo 7 para a comparação teórica entre estes algoritmos).

Nota metodológica

A **validação cruzada** (*cross-validation*) — em alternativa a uma única divisão treino/teste — permite obter uma estimativa mais robusta do desempenho de um modelo (ver Capítulo 7). Diferenças entre a exatidão global (accuracy) obtida por **Split Data** e por **Cross Validation** são normais, e refletem a variabilidade introduzida por uma única divisão aleatória dos dados.

Ficha 07 · AI Studio — Classificação

O dataset usado neste exercício descreve fatores de risco de doença cardíaca (heart-c.csv, ficheiro disponibilizado para a ficha; identificar a origem e a versão). O atributo num representa a classe: 0 — nenhuma doença; 1 a 4 — códigos não nulos cujo significado deve ser confirmado no dicionário de dados; não presumir uma escala de gravidade clínica. O objetivo é prever doença cardíaca a partir dos restantes atributos — um problema de classificação, a resolver com o RapidMiner.

1. Compreensão do problema e dos dados

1. Para cada atributo, obtenha (recorrendo às abas Estatísticas e Visualizações): tipo de atributo, percentagem de valores ausentes, máximo, mínimo, média, moda e desvio-padrão (quando aplicável). Existem registos únicos nalgum atributo?
2. Estude os histogramas dos atributos numéricos e descreva informalmente como parecem influenciar o risco de doença cardíaca.
3. Usando um gráfico de dispersão (*Scatter*): os atributos thalach e ca parecem mais ou menos associados a doença cardíaca? Investigue o gráfico thalach (X) / oldpeak (Y) — existem áreas “densas” de doença cardíaca?

2. Preparação e desenho de validação (documentar operadores e parâmetros)

Definir primeiro uma separação estratificada treino/teste. Os passos seguintes são aprendidos apenas no treino e repetidos dentro de cada partição da validação cruzada.

1. Tratar valores ausentes: não eliminar registos — substituir pela média (atributos numéricos) ou moda (atributos nominais). Guardar como `fe06-p1`.
2. Avaliar o peso dos atributos com **Weight by Information Gain**, seleccionando um subconjunto com peso > 0.4 . Guardar como `fe06-p2`. Comparar com as conclusões da exploração, sem usar o teste para seleccionar atributos.
3. Investigar observações assinaladas por **Detect Outlier**. Comparar uma análise sem exclusão com uma exclusão justificada apenas no treino. Não remover automaticamente casos difíceis do teste. Guardar como `fe06-p3`.
4. Normalizar os dados com **Normalize**. Guardar como `fe06-p4`.
5. Combinar os passos 2, 3 e 4 num único processo. Guardar como `fe06-p5`.

3. Modelação

Repetir para cada um dos processos da Parte 3, e também para o dataset original:

1. Classificador **Decision Trees**: que conclusões pode tirar (atributos importantes)? Compare a precisão obtida com **Split Data** e com **Cross Validation**. Crie um classificador com e sem *pruning* — qual se comporta melhor?
2. Explore mais 2 algoritmos de classificação, usando **Split Data** e **Cross Validation** para todos.

4. Avaliação

1. Escolha medidas de desempenho e justifique a escolha.
2. Resuma numa tabela as medidas de desempenho para cada classificador e cada dataset.
3. O que pode concluir?

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

CAPÍTULO 17

Exercícios — Extração de Conhecimento com Python

17.1 Fluxo de trabalho em código

Referência de apoio: scikit-learn, *Common pitfalls and recommended practices* (ver referências). Este capítulo replica, em Python (usando o Google Colab como ambiente de referência), o mesmo problema de classificação de doença cardíaca já trabalhado no capítulo anterior — mas agora através de código, o que exige tornar explícitas decisões que, no AI Studio, eram tomadas através de operadores visuais.

O código seguinte ilustra a estrutura típica de um *pipeline* de classificação em Python, organizado segundo as mesmas fases do CRISP-DM já estudadas:

```
import numpy as np
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import (
    train_test_split, StratifiedKFold, cross_validate,
)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.dummy import DummyClassifier
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_auc_score,
    average_precision_score,
)

def main():
    df = pd.read_csv("heart-c.csv", na_values=["?", "NA", ""])
    if "num" not in df:
        raise ValueError("Falta o alvo 'num'; confirmar o dicionário.")
    alvo = pd.to_numeric(df["num"], errors="coerce")
    # Não imputar o desfecho: excluir e contabilizar alvos ausentes.
    print("Alvos excluídos:", int(alvo.isna().sum()))
    validos = alvo.notna()
    df, alvo = df.loc[validos].copy(), alvo.loc[validos]
    if not alvo.isin([0, 1, 2, 3, 4]).all():
        raise ValueError("Códigos do alvo inesperados.")
    y = (alvo > 0).astype(int)

    # Lista explícita: exclui num, class e identificadores.
    numericas = ["age", "trestbps", "chol", "thalach", "oldpeak"]
    categoricas = ["sex", "cp", "fbs", "restecg", "exang",
                  "slope", "ca", "thal"]
    colunas = numericas + categoricas
    ausentes = set(colunas) - set(df.columns)
    if ausentes:
        raise ValueError(f"Confirmar nomes/esquema: {sorted(ausentes)}")
    X = df[colunas].apply(pd.to_numeric, errors="coerce")

    # Pressuposto do exemplo: uma linha por indivíduo independente.
    X_tr, X_te, y_tr, y_te = train_test_split(
        X, y, test_size=0.20, stratify=y, random_state=42,
    )
    n_folds = min(5, int(y_tr.value_counts().min()))
    if n_folds < 2:
        raise ValueError("Casos insuficientes para validação estratificada.")
    cv = StratifiedKFold(n_folds, shuffle=True, random_state=42)

    numerico = Pipeline([
```

```

        ("imputar", SimpleImputer(strategy="median",
                                   keep_empty_features=True)),
        ("escalar", StandardScaler()),
    ])
categorico = Pipeline([
    ("imputar", SimpleImputer(strategy="most_frequent",
                              keep_empty_features=True)),
    ("codificar", OneHotEncoder(handle_unknown="ignore",
                                sparse_output=False)),
])
preparar = ColumnTransformer([
    ("numericas", numerico, numericas),
    ("categoricas", categorico, categoricas),
])
modelos = {
    "Referencia": DummyClassifier(strategy="prior"),
    "Logistica": LogisticRegression(max_iter=2000),
    "Random Forest": RandomForestClassifier(
        n_estimators=200, random_state=42),
    "SVM": SVC(),
}
pipelines, comparacao = {}, {}
for nome, modelo in modelos.items():
    pipe = Pipeline([("preparar", preparar), ("modelo", modelo)])
    pipelines[nome] = pipe
    resultados = cross_validate(
        pipe, X_tr, y_tr, cv=cv,
        scoring={"ap": "average_precision", "f1": "f1",
                 "balanced_accuracy": "balanced_accuracy"},
        error_score="raise",
    )
    comparacao[nome] = resultados["test_ap"].mean()
    print(nome, "AP media:", comparacao[nome],
          "DP:", resultados["test_ap"].std())

# AP é a métrica predefinida para esta demonstração binária.
escolhido = max(comparacao, key=comparacao.get)
final = pipelines[escolhido].fit(X_tr, y_tr)
previsto = final.predict(X_te)
if hasattr(final, "predict_proba"):
    score = final.predict_proba(X_te)[:, 1]
else:
    score = final.decision_function(X_te)
print("Modelo selecionado:", escolhido)
print(classification_report(y_te, previsto, zero_division=0))
print("Matriz de confusão:", confusion_matrix(y_te, previsto))
print("ROC-AUC:", roc_auc_score(y_te, score))
print("AP:", average_precision_score(y_te, score))
print("Prevalência no teste:", float(y_te.mean()))

if __name__ == "__main__":
    main()

```

Nota metodológica

Este código é um exemplo ilustrativo, adaptado à estrutura do exercício deste capítulo — não é uma transcrição de um notebook específico. Serve como esqueleto de referência: cada secção pode e deve ser adaptada ao dataset e às perguntas concretas de cada fase.

Questões de reflexão

Compare este *pipeline* em código com o processo equivalente construído visualmente no capítulo anterior. Que vantagens e desvantagens identifica em cada abordagem (ver também Capítulo 8)?

Ficha 08 · Extração de Conhecimento com Python

O dataset é novamente sobre doenças cardíacas (`heart-c.csv`, ficheiro disponibilizado para a ficha). O atributo `num` representa a classe: 0 — sem doença; 1 a 4 — códigos não nulos a interpretar segundo o dicionário de dados, sem presumir gravidade crescente. Objetivo: prever doença cardíaca a partir dos restantes atributos. Ferramenta: Python, via Google Colab.

1. Compreensão dos Dados

1. Carregar os dados e imprimir as primeiras linhas. Examinar a estrutura básica e o tipo de dados de cada atributo.
2. Verificar a existência de valores em falta e identificar colunas que necessitem de limpeza ou transformação adicional.
3. Calcular estatísticas básicas (média, mediana, etc.) e analisar a distribuição de cada atributo.

2. Exploração dos Dados

1. Explorar a distribuição da classe (casos positivos vs. negativos). Converter a classe em 0 (sem doença) / 1 (com doença).
2. Comparar diferentes atributos (`age`, `thalach`, `oldpeak`) com a classe, recorrendo a *boxplots*. Usar histogramas (`histplot`) para comparar `thalach` e `ca` com a possibilidade de doença.
3. Identificar padrões ou correlações entre atributos, utilizando matrizes de correlação (excluindo `id` e `num`).

3. Processamento de Dados

1. Analisar os valores em falta no treino e justificar imputação ou exclusão. Um limiar de 30% pode ser explorado como hipótese, mas não é uma regra universal; a decisão faz parte do procedimento de validação.
2. Escalar/normalizar as características numéricas.
3. Codificar variáveis categóricas em valores numéricos.
4. Remover atributos desnecessários.

4. Modelação

1. Separar treino e teste (80/20) antes de ajustar o pré-processamento. Se existirem observações repetidas por doente, a separação deve ser por doente.
2. Construir e treinar um modelo de classificação simples (por exemplo, `LogisticRegression`).
3. Construir e treinar 2 modelos mais complexos (por exemplo, `Random Forest`, `SVM`).

5. Avaliação

1. Avaliar o desempenho de cada modelo com Precisão, Recall e F1-score.
2. Avaliar a exatidão global (`accuracy`) global de cada modelo, com um gráfico de barras comparativo.
3. Gerar e interpretar a matriz de confusão de cada modelo.
4. Comparar os modelos por validação cruzada apenas no treino. Selecionar o procedimento pela métrica definida e avaliar o modelo escolhido uma única vez no teste reservado.

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

CAPÍTULO 18

Exercícios — k-Means Clustering

18.1 Caso prático guiado

Caso de estudo · Exploração de perfis quantitativos

Contexto: pretende-se explorar a heterogeneidade de um conjunto didático de indivíduos, sem classes conhecidas nem um modelo de risco validado.

Compreensão dos dados: o material original descreve 547 observações com `weight`, `cholesterol` e `gender`. Confirmar tamanho, unidades, origem e significado das variáveis no ficheiro `pl12-dataset.csv` antes da análise.

Preparação: usar peso e colesterol como atributos quantitativos, documentando valores em falta e escalonamento. A variável categórica `gender` não entra na distância euclidiana deste exemplo; pode ser usada para descrever a composição dos grupos, após discussão da sua definição no dicionário de dados.

Modelação: comparar vários valores de k usando inércia, silhueta e estabilidade. Uma expectativa inicial de quatro grupos é uma hipótese, não uma conclusão.

Avaliação: interpretar os centróides na escala original. Os números atribuídos aos clusters são arbitrários e podem mudar entre execuções. Não equivalem a categorias de risco clínico nem demonstram a utilidade de uma intervenção.

Utilização dos resultados: formular hipóteses para análise posterior; qualquer aplicação em saúde requer validação independente e apreciação do contexto de utilização.

18.2 k-Means em Python

O mesmo algoritmo pode ser aplicado diretamente em Python com `scikit-learn`, com a vantagem de facilitar a escolha objetiva do número de clusters através do **método do cotovelo** (ver Capítulo 9):

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score

def main():
    df = pd.read_csv("pl12-dataset.csv", na_values=["?", "NA", ""])
    atributos = ["weight", "cholesterol"]
    if not set(atributos).issubset(df.columns):
        raise ValueError("Confirmar nomes dos atributos no ficheiro.")
    X = df[atributos].apply(pd.to_numeric, errors="coerce")
    # Exclusão explícita so para este exemplo exploratorio.
    completos = X.dropna()
    print("Observacoes excluidas:", len(X) - len(completos))
    scaler = StandardScaler()
    Z = scaler.fit_transform(completos)
    limite = min(10, len(Z) - 1, len(np.unique(Z, axis=0)))
    if limite < 2:
        raise ValueError("Observacoes distintas insuficientes.")
    linhas, candidatos = [], {}
    for k in range(2, limite + 1):
        modelo = KMeans(n_clusters=k, n_init=20, random_state=42)
        grupos = modelo.fit_predict(Z)
        if len(np.unique(grupos)) != k:
            continue
        candidatos[k] = modelo
    linhas.append({
        "k": k, "inercia": modelo.inertia_,
        "silhueta": silhouette_score(Z, grupos),
        "davies_bouldin": davies_bouldin_score(Z, grupos),
```

```

    })
    if not linhas:
        raise ValueError("Nao foi obtida uma particao valida.")
    resumo = pd.DataFrame(linhas).set_index("k")
    print(resumo)
    # Heuristica explicita; discutir estabilidade e plausibilidade.
    k_escolhido = int(resumo["silhueta"].idxmax())
    modelo = candidatos[k_escolhido]
    centros = scaler.inverse_transform(modelo.cluster_centers_)
    print("Centroides na escala original:")
    print(pd.DataFrame(centros, columns=atributos))
    resumo["inerzia"].plot(marker="o")
    plt.ylabel("Inercia")
    plt.show()
    plt.scatter(completos["weight"], completos["cholesterol"],
                c=modelo.labels_, cmap="viridis", s=18)
    plt.xlabel("Peso (unidade a confirmar)")
    plt.ylabel("Colesterol (unidade a confirmar)")
    plt.show()

if __name__ == "__main__":
    main()

```

Nota metodológica

O **índice de Davies-Bouldin** é uma métrica alternativa e complementar ao método do cotovelo (ver Capítulo 9): quanto **menor** o valor, melhor a separação obtida.

Ficha 09 · AI Studio — k-Means Clustering

1. Introdução

1. O que significa k-Means clustering?
2. Como se identificam os clusters, e que processo usa o AI Studio para atribuir observações a um cluster?
3. O que revela a *Centroid Table* ao utilizador? Como se interpretam os seus valores?
4. Depois do exercício introdutório, pense num problema que possa ser resolvido agrupando observações em clusters. Encontre um dataset online e execute os seguintes passos:
 - **Preparação dos dados:** selecionar apenas os atributos numéricos e lidar com valores nulos.
 - **Normalização:** avaliar e justificar o escalonamento; a padronização iguala escalas, mas não garante igual relevância ou influência dos atributos no agrupamento.
 - **Aplicar K-Means:** usar o K-Means da biblioteca *sklearn* para executar o clustering.
 - **Visualização:** fazer o *plot* dos resultados e interpretá-los.

Elementos a entregar e critérios de apreciação

Entregar o processo ou código, a identificação dos dados e das versões das ferramentas, os resultados essenciais e uma discussão das limitações. A apreciação incide na correção metodológica, na reprodutibilidade, na interpretação dos resultados e na clareza da comunicação. Resultados numéricos sem descrição do procedimento são insuficientes.

Orientações para a autoavaliação

Estas orientações não substituem a discussão em aula nem constituem soluções únicas para as fichas. Explicitam critérios mínimos para verificar a coerência do trabalho.

Correlação. Um coeficiente positivo significa que valores elevados de uma variável tendem a acompanhar valores elevados da outra; duas variáveis que diminuem em conjunto também podem estar positivamente correlacionadas. Antes da interpretação, inspecionar o gráfico de dispersão, a escala das variáveis e observações influentes. Associação não identifica, por si só, um mecanismo causal.

Regras de associação. Nas quatro transações do exemplo teórico, A surge em três e C em duas, ambas contendo A . Assim, o suporte da regra $A \rightarrow C$ é $\frac{2}{4} = 0.5$, a confiança é $\frac{2}{3} \approx 0.667$ e o lift é $\frac{\frac{2}{3}}{0.5} \approx 1.333$. Suporte e confiança elevados não demonstram utilidade de uma decisão; verificar também redundância, estabilidade e contexto de aplicação.

Regressão. A substituição nos coeficientes apresentados no caso da habitação produz aproximadamente 196 830,16 unidades monetárias. Este é um cálculo sobre coeficientes fornecidos, não a reprodução do ajuste. A apreciação de um modelo deve incluir erro em dados reservados, comparação com uma referência e análise das limitações do desenho.

Classificação. Ao construir `class = (num > 0)`, remover `num` dos preditores é obrigatório: mantê-lo revelaria a resposta ao algoritmo. Imputação, escalonamento e seleção de atributos aprendem-se no treino de cada partição. Escolher o modelo por validação cruzada no desenvolvimento; reservar o teste para a avaliação final. Uma melhoria anormalmente grande de desempenho deve motivar uma auditoria de fuga de informação.

Agrupamento. Os rótulos dos clusters não têm ordem nem significado clínico intrínseco. Comparar métricas, tamanhos, centróides na escala original e estabilidade entre sementes. Uma silhueta maior é um critério interno de comparação, não uma prova de que existem grupos naturais ou de que estes permitem decidir uma intervenção.

Reprodutibilidade. Outra pessoa deve conseguir identificar os dados, preparar o ambiente e repetir os passos sem decisões implícitas. Registar versões, parâmetros, exclusões e semente. Nas tarefas de integração, testar ainda uma segunda execução e demonstrar que esta não duplica eventos nem perde informação.

Glossário

- AI Studio / RapidMiner** Ferramenta comercial visual de Data Mining e Machine Learning usada como referência prática nesta unidade curricular.
- Apriori** Algoritmo de descoberta de itemsets frequentes, baseado na geração iterativa de candidatos e no princípio da anti-monotonicidade.
- AUC** Área sob a curva ROC; resume num único número o desempenho de um classificador em todos os limiares de decisão.
- Bagging** Técnica de ensemble que treina modelos em paralelo sobre amostras aleatórias dos dados, combinando-os por votação/média (ex.: Random Forest).
- Bias-Variância (compromisso)** Decomposição do erro de generalização em erro sistemático (bias) e sensibilidade aos dados de treino (variância).
- Boosting** Técnica de ensemble que treina modelos sequencialmente, corrigindo os erros dos anteriores (ex.: XGBoost).
- Business Understanding** Primeira fase do CRISP-DM — compreensão dos objetivos e requisitos de negócio.
- Classificação** Técnica de Data Mining supervisionada que prevê uma classe (categoria discreta).
- Clustering** Técnica de Data Mining não supervisionada que agrupa observações semelhantes entre si.
- Confiança (regras de associação)** Proporção de observações com o antecedente que também têm o consequente de uma regra.
- Conviction** Métrica de força de uma regra de associação, baseada na frequência esperada de falha da regra sob independência.
- Correlação** Família de medidas de associação; Pearson caracteriza a relação linear e Spearman a relação monótona entre ordens.
- CRISP-DM** *CRoss Industry Standard Process for Data Mining* — metodologia de seis fases para projetos de Data Mining.
- Cross-validation (validação cruzada)** Técnica de avaliação que particiona o dataset em k folds, treinando e testando o modelo k vezes.
- Data Mining** Processo de extração de padrões ou conhecimento de interesse a partir de grandes quantidades de dados.
- Data Understanding** Fase do CRISP-DM dedicada à recolha, exploração e familiarização com os dados.
- Data Preparation** Fase do CRISP-DM dedicada à seleção, limpeza, construção e formatação dos dados.
- Deployment** Fase do CRISP-DM dedicada à implementação e monitorização da solução em produção.
- DBSCAN** Algoritmo de clustering baseado em densidade, capaz de identificar clusters de forma irregular e pontos de ruído.
- Entropia** Medida de desordem/impureza de um conjunto de dados, usada em árvores de decisão.
- Evaluation** Fase do CRISP-DM dedicada à avaliação dos resultados face aos objetivos de negócio.
- FP-Growth** Algoritmo de descoberta de itemsets frequentes baseado numa estrutura em árvore compacta (FP-tree).
- Gini (impureza de)** Medida de impureza usada em árvores de decisão para escolher a melhor divisão.
- Idempotência** Propriedade de uma operação que, executada múltiplas vezes, produz sempre o mesmo resultado final.
- Imputação** Substituição de valores em falta por valores estimados (média, mediana, moda, regressão, etc.).
- Inércia (WCSS)** Soma das distâncias quadráticas entre cada ponto e o centróide do seu cluster; função-objetivo do k-Means.
- Integridade referencial** Cada chave estrangeira não nula referencia uma chave válida na tabela correspondente.
- JSON** *JavaScript Object Notation* — formato de texto para troca de dados estruturados.
- k-Means** Algoritmo de clustering que particiona os dados em k grupos, minimizando a inércia.
- k-means++** Estratégia de inicialização do k-Means que espalha os centróides iniciais de forma mais eficaz do que a inicialização aleatória.
- KDD** *Knowledge Discovery in Databases* — processo global de descoberta de conhecimento, do qual o Data Mining é uma etapa.
- Lift** Métrica de força de uma regra de associação que corrige a confiança pela popularidade base do consequente.

Matriz de confusão Tabela que cruza as classes previstas por um classificador com as classes reais.

MCAR / MAR / MNAR Classificação dos mecanismos de dados em falta, consoante a probabilidade de faltar depender (ou não) de outros atributos ou do próprio valor.

Modeling Fase do CRISP-DM dedicada à seleção e aplicação de técnicas de modelação.

Multicolinearidade Correlação forte entre variáveis independentes de um modelo de regressão, que instabiliza a estimação dos coeficientes.

Naive Bayes Classificador probabilístico baseado no Teorema de Bayes, assumindo independência condicional entre atributos.

Normalização Reescalonamento de atributos numéricos para um intervalo ou distribuição comum.

OLS *Ordinary Least Squares* — método de estimação dos coeficientes de regressão linear, minimizando a soma dos quadrados dos resíduos.

Outlier Observação que se desvia significativamente do comportamento geral dos dados.

Overfitting Sobreajuste de um modelo aos dados de treino, com fraca capacidade de generalização.

Pipeline Sequência encadeada e reutilizável de transformações de dados e modelação.

Pruning Poda de uma árvore de decisão, para reduzir overfitting.

Valor-p Probabilidade, sob a hipótese nula e o modelo do teste, de obter uma estatística tão ou mais extrema do que a observada; não é a probabilidade da hipótese nula.

R² Comparação do erro quadrático com o de uma previsão pela média; pode ser negativo fora das condições do ajuste OLS com termo independente na amostra de treino.

Random Forest Ensemble de árvores de decisão treinadas por bagging.

Regularização (Ridge/Lasso) Técnicas que penalizam a magnitude dos coeficientes de regressão, reduzindo overfitting.

REST *Representational State Transfer* — estilo arquitetural para APIs baseadas em recursos identificados por URLs.

RMSE *Root Mean Squared Error* — raiz quadrada do erro quadrático médio de previsão.

ROC (curva) Representação gráfica do desempenho de um classificador em todos os limiares de decisão possíveis.

Silhueta (coeficiente de) Medida de qualidade de um agrupamento, baseada na coesão interna e separação entre clusters.

Spearman (correlação de) Medida de associação monótona, calculada sobre as ordens (*ranks*) dos valores.

Suporte Proporção de observações que contêm um dado itemset.

SVM *Support Vector Machine* — classificador que procura o hiperplano de margem máxima entre classes.

Validação cruzada Ver Cross-validation.

VIF *Variance Inflation Factor* — diagnóstico de multicolinearidade num modelo de regressão.

XGBoost Implementação popular e eficiente de gradient boosting para árvores de decisão.

Referências e documentação de apoio

As referências seguintes apoiam os conceitos e as correções metodológicas desta edição. A documentação de ferramentas deve ser consultada para a versão utilizada. Consulta das fontes em linha: 16 de outubro de 2026.

- IBM. *CRISP-DM Help Overview*. Documentação CRISP-DM.
- scikit-learn. *Common pitfalls and recommended practices*. Separação de dados e prevenção de fuga de informação.
- scikit-learn. *StandardScaler*. Padronização de atributos.
- scikit-learn. *r2_score*. Definição e domínio de R^2 .
- scikit-learn. *Metrics and scoring: quantifying the quality of predictions*. Avaliação de classificadores e regressores.
- scikit-learn. *Clustering*. Agrupamento, distâncias e métricas.
- SciPy. *pearsonr*. Coeficiente de Pearson e inferência.
- RapidMiner. *Installing RapidMiner AI Studio*. Instalação e documentação do ambiente.

Rastreabilidade dos casos de estudo. Os casos do aquecimento doméstico, organizações comunitárias, preço de habitação e adoção de eReader foram conservados dos materiais fornecidos. A sua fonte bibliográfica, autoria original, licença e correspondência com os ficheiros de dados devem ser confirmadas pelos autores antes de publicação.