Introdução à Aprendizagem Automática

1.1 Tipos de Aprendizagem

Aprendizagem Supervisionada (LR, ANN, DT)

Os dados estão rotulados. O modelo aprende a mapear inputs para outputs conhecidos.

Aprendizagem Não Supervisionada (Clustering, PCA)

Os dados não têm rótulos. O objetivo é descobrir padrões ou estruturas nos dados.

Aprendizagem Profunda (Deep Learning) (CNN, RNN)

Usa redes neuronais com múltiplas camadas. Ideal para dados complexos como imagens, texto ou áudio.

Q2: Explique a diferença entre aprendizagem supervisionada e não supervisionada.

Na **aprendizagem supervisionada** os dados têm labels e o modelo aprende a mapear inputs para outputs conhecidos (ex: classificar emails). Na não supervisionada os dados não têm labels, e o objetivo é descobrir padrões ou estruturas nos dados (ex: agrupar clientes por comportamento de compra).

1.2 Regressão vs Classificação

Q1: Explique a diferença entre regressão e classificação. Dê um exemplo de cada.

Regressão prevê valores contínuos com um output de número real(ex: prever o preço de uma casa com base na área). **Classificação** prevê categorias discretas sendo o output uma classe (ex: detetar se um email é spam ou não).

1.3 Pipeline de ML

Todo o projeto de ML segue um pipeline:

1. Seleção e recolha de dados
2. Exploração e visualização dos dados
3. Pré-processamento
    - Feature extraction e Feature Selection
    - Data cleaning (missing values / entradas incorretas)
    - Scaling / Normalização
4. Escolha e treino do modelo
5. Avaliação do modelo
6. Ajuste/otimização (hyperparameter tuning)
7. Deployment e monitorização

Q3: Porque é que o scaling/normalização das features é importante?

Porque métodos baseados em distâncias (como K-NN e SVM) e em gradiente (como redes neuronais) são sensíveis à escala. Se uma feature tem valores de 0 a 1000 e outra de 0 a 1, a primeira domina o cálculo de distâncias. A normalização garante que todas as features contribuem de forma equilibrada. A desvantagem é que o significado físico original das variáveis se perde.

Q4: Tem um conjunto de dados com 100.000 registos de clientes, cada um com mais de 50 variáveis. O objetivo é prever cancelamentos de contrato. Descreva os passos principais que seguiria.

1) Exploração de dados: analisar distribuições, visualizar correlações, identificar missing values e outliers.

2) Tratamento de dados: lidar com valores em falta, corrigir inconsistências, normalizar variáveis numéricas, codificar variáveis categóricas.

3) Seleção de variáveis: remover features não informativas/redundantes usando correlação, importância de features ou métodos wrapper/filter.

4) Tratar desequilíbrio de classes: se os cancelamentos são raros, usar SMOTE, undersampling ou penalização de algoritmos.

5) Escolha do modelo: testar vários (Logistic Regression como baseline, Random Forest, XGBoost). Usar cross-validation para avaliação robusta.

6) Validação: usar k-fold, métricas adequadas (F1-score, AUC-ROC) em vez de accuracy simples.

7) Interpretação: analisar importância das features e erros sistemáticos.

2. Aprendizagem Não Supervisionada - Clustering e PCA

2.1 Clustering

Agrupar exemplos similares em clusters, sem rótulos.

2.2 K-Means

Q5: Explique o algoritmo K-Means, incluindo como funciona e como se escolhe K.

K-Means é um algoritmo de clustering iterativo.

Passos: (1) inicializar K centróides aleatoriamente;

(2) atribuir cada ponto ao centróide mais próximo;

(3) recalcular centróides como média dos pontos de cada cluster;

(4) repetir 2-3 até convergência.

Para escolher K: usar visualização dos dados, conhecimento do domínio, ou o Elbow Method.

2.3 Redução de Dimensionalidade - PCA

Q6: O que é PCA e qual a sua utilidade em ML?

PCA é uma técnica que reduz o número de features, mantendo o máximo de informação possível dos dados originais.

É útil para: reduzir o número de features antes de treinar um modelo (menos parâmetros = treino mais rápido), visualizar dados de alta dimensão em 2D/3D, e remover correlações entre features.

Limitações: funciona só com relações lineares, perde informação, e os componentes principais são difíceis de interpretar.

3. Regressão Linear - **prever valores contínuos**

A regressão é um algoritmo de **aprendizagem supervisionada** e um método estatístico utilizado para modelar a relação entre uma variável dependente (target - y) e uma ou mais variáveis independentes (features - x), ajustando uma equação linear aos dados observados

3.1 Overfitting e Regularização

Situação	Bias	Variância	Sintoma
Underfit	Alto	Baixo	Erro alto no treino E no teste
Good Fit	Equilibrado	Equilibrado	Erro aceitável em ambos
Overfit	Baixo	Alto	Erro baixo no treino, alto no teste

Como combater o overfitting:

1. Reduzir o numero de features
2. Regularização (requerem normalização)

Q8: Explique a diferença entre Ridge e Lasso. Quando usaria cada um?

Ridge (L2) adiciona uma penalidade proporcional ao quadrado dos coeficientes. Reduz a magnitude mas nunca os leva a zero, mantendo todas as features. Ideal quando todas as features contribuem para a previsão.

Lasso (L1) adiciona uma penalidade proporcional ao valor absoluto dos coeficientes. Pode forçar alguns coeficientes exatamente a zero, fazendo seleção automática de features. Ideal quando se suspeita que muitas features são irrelevantes e se quer um modelo mais interpretável.

Elastic Net combina ambos: útil quando há muitas features correlacionadas.

Q7: Num projeto de previsão de procura, o modelo apresenta erros sistemáticos em determinados períodos (fim de semana ou férias). O que pode estar a causar este comportamento?

O modelo provavelmente não captura padrões temporais específicos desses períodos.

Causas: (1) **Features insuficientes** - o modelo pode não ter variáveis que indiquem dia da semana, feriados ou eventos especiais;

(2) **Dados de treino desequilibrados** - se fins de semana têm poucos exemplos nos dados de treino;

(3) **O modelo assume linearidade** quando a relação real é não-linear nesses períodos.

Ações: adicionar features como 'dia da semana', 'é feriado', 'mês do ano'; usar modelos de séries temporais (ARIMA, SARIMA) que modelam sazonalidade;

4. Regressão Logística e Classificação – **prever classificação binaria**

Vantagens:

- Simples e interpretável (pesos mostram importância das features)
- Eficiente computacionalmente
- Output probabilístico

Desvantagem:

- Assume relação linear entre features e log-odds
- Pressupostos de independência das features
- Pode underfit quando a fronteira é muito não-linear

Q9: Qual a diferença entre regressão linear e regressão logística?

Regressão linear prevê valores contínuos (output é um número real), usa MSE como função de custo; usada para regressão

Regressão logística prevê probabilidades de pertencer a uma classe (output entre 0 e 1), usa Log Loss como função de custo;

4.2 Fronteira de Decisão

A fronteira de decisão é **onde o modelo muda de opinião**, de um lado prevê classe 0, do outro prevê classe 1.

Linear vs Não-linear:

- **Linear**: a fronteira é uma linha reta (ou plano em 3D). Funciona quando os dados são separáveis por uma linha.
- **Não-linear**: quando os dados não são separáveis por uma linha reta, adicionas features como x^2 para criar fronteiras curvas (círculos, elipses, formas complexas).

4.3 Função de Custo - Log Loss

Situação	Penalização
Previu 90% de probabilidade → era mesmo classe 1 ✓	Penalização pequena- acertou com confiança
Previu 50% de probabilidade → era classe 1	Penalização média- acertou mas sem certeza
Previu 10% de probabilidade → era classe 1 ✗	Penalização ENORME- estava muito confiante e errou

Q10: Como funciona a estratégia One vs All para classificação multiclasse?

Para K classes, treina-se K classificadores binários - cada um responde à pergunta "é esta classe ou não?". Para classificar um novo exemplo, corre-se todos os classificadores e prevê-se a classe que retornou a maior probabilidade (winner-takes-all).

Exemplo: 3 classes (gato/cão/pássaro) → treinar P(gato|x), P(cão|x), P(pássaro|x);

prever a classe com maior probabilidade.

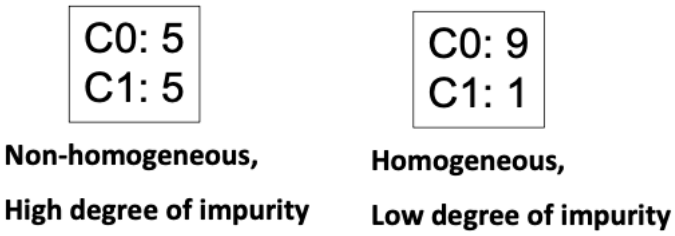
5. Naive Bayes, Árvores de Decisão e SVM

Modelo baseado em estrutura de árvore onde cada nó interior é uma pergunta sobre uma feature e cada folha é uma classe.

Prós	Contras
Interpretável e visualizável	Muito propenso a overfitting
Não precisa de scaling	Instável (pequenas mudanças → árvore diferente)
Lida com features numéricas e categóricas	Biasado com classes desequilibradas

Como escolher o melhor split?

- obter a **árvore mais pequena possível** com **folhas puras** (quase todos os exemplos da mesma classe)
- Em cada nó, escolhe-se a feature que **mais reduz a impureza** após a divisão
- **Impureza = mistura de classes** num nó - quanto mais misturado, pior



5.3 SVM Linear - Support Vector Machine

O SVM encontra a fronteira de decisão (hiperplano) que maximiza a margem (distancia entre hiperplano e ponto mais próximo) entre as duas classes.

Objetivo: maximizar a margem → melhor generalização

• **Support Vectors**: os pontos mais próximos do hiperplano - definem a margem

Parâmetro C: controla o trade-off entre margem e erros de classificação

- C grande → mais importância ao ajuste dos dados de treino (menor margem, mais overfitting)
- C pequeno → maior margem, mais tolerante a erros de treino (mais generalização)

Q11: O que são Support Vectors no SVM e qual a sua importância?

Support Vectors são os pontos de treino mais próximos do hiperplano de separação. São os únicos pontos que determinam a posição e orientação do hiperplano.

São importantes porque: ao maximizar a distância (margem) em relação a eles, o SVM encontra a fronteira mais robusta, que generaliza melhor para dados novos. Se removermos um support vector, o hiperplano muda;

5.4 SVM Não-Linear - Kernel SVM

Para dados não linearmente separáveis, o kernel SVM transforma os dados para um espaço de maior dimensão onde se tornam separáveis, sem calcular explicitamente a transformação.

Parâmetros do SVM-RBF

C grande → baixo bias, alta variância (overfitting)

C pequeno → alto bias, baixa variância (underfitting)

σ^2 grande (γ pequeno) → fronteira mais suave (bias alto)

σ^2 pequeno (γ grande) → fronteira mais complexa (variância alta)

Q12: Compare Árvores de Decisão com Naive Bayes para um problema de classificação de texto.

Naive Bayes é geralmente preferível para classificação de texto porque é eficiente com muitas features,funciona bem com dados esparsos e requer menos dados de treino.

Árvores de Decisão têm vantagem na interpretabilidade mas tendem a fazer overfitting com muitas features e são instáveis.

Para texto, Naive Bayes é o baseline clássico (spam detection, análise de sentimento). Se a interpretabilidade for prioritária, a árvore pode ser melhor, mas será necessário poda e controlo da profundidade.

6. Avaliação de Classificadores e Desequilíbrio de Classes

6.1 Matriz de Confusão

	Previsto: Positivo	**Previsto: Negativo**
Real: Positivo	TP (True Positive)	FN (False Negative)
Real: Negativo	FP (False Positive)	TN (True Negative)

Q13: Considere a seguinte matriz de confusão (Positivo=20, FN=80, FP=15, TN=85). Interprete e comente o desempenho.

Total = 200 exemplos.

Accuracy = (20+85)/200 = 52.5% - performance medíocre.

Recall (TPR) = 20/(20+80) = 20% - o classificador deteta apenas 20% dos casos positivos reais, péssimo se os positivos são importantes (ex: diagnóstico de doença).

Precision = 20/(20+15) = 57% - dos que classifica como positivos, 57% são corretos.

O principal problema é o Recall muito baixo - há 80 Falsos Negativos. O modelo falha em detetar a maioria dos casos positivos. Se esta for uma aplicação crítica (medicina, fraude), o modelo é claramente insatisfatório.

Deve-se reduzir o threshold de decisão para aumentar o Recall, mesmo que a Precision diminua.

6.2 Métricas de Avaliação

Métrica	Fórmula	O que mede	Quando usar
Accuracy	(TP+TN)/(TP+TN+FP+FN)	% total de acertos	Apenas com classes balanceadas
Precisão (Precision)	TP/(TP+FP)	Dos que previu positivo, % corretos	Quando FP é custoso (ex: spam)
Recall (Sensibilidade)	TP/(TP+FN)	Dos reais positivos, % detetados	Quando FN é custoso (ex: diagnóstico)
F1-Score	2×(P×R)/(P+R)	Média harmónica de Precision e Recall	Classes desequilibradas
Balanced Accuracy	(TPR+TNR)/2	Média do recall de cada classe	Datasets desequilibrados
AUC-ROC	Área sob curva ROC	Capacidade geral de distinguir classes	Avaliação geral do classificador

Limitações da Accuracy

- **Desequilíbrio de classes**
  - A accuracy pode ser enganosa quando uma classe é muito mais frequente que as outras
  - Exemplo: 95% de accuracy pode significar que o modelo está sempre a prever a classe maioritária

- **Não distingue tipos de erro**
  - A accuracy não diferencia entre falsos positivos e falsos negativos
  - Crítico em domínios como saúde ou deteção de fraude

Q15: Um modelo de classificação tem alta precisão mas baixa sensibilidade. A equipa considera o desempenho satisfatório. Concorda?

Depende do contexbto. **Alta precisão** significa que quando o modelo prevê positivo, raramente erra. Mas **baixa sensibilidade (recall)** significa que muitos casos positivos reais não são detetados (muitos FN). Se a aplicação é diagnóstico médico ou deteção de fraude, os FN têm alto custo - o modelo pode falhar em detetar doenças/fraudes reais, o que é perigoso. Neste caso, o desempenho NÃO é satisfatório. Se o objetivo é enviar emails de marketing só quando há alta certeza (FP custosos), então alta precisão com recall moderado pode ser aceitável. A avaliação correta depende sempre do problema de negócio e do custo relativo dos FP e FN.

Métrica baixa	Significa	Na matriz de confusão
Accuracy baixa	Muitos erros no geral	Muitos FP + FN
Recall baixo	Falha em detetar os positivos reais	Muitos **FN**
Precision baixa	Muitos falsos alarmes	Muitos **FP**
F1 baixo	Mau balanço entre Precision e Recall	Muitos FP e/ou FN

Métrica alta	Significa	Na matriz de confusão
Accuracy alta	Muitos acertos no geral	Muitos TP + TN
Recall alto	Deteta quase todos os positivos reais	Poucos **FN**
Precision alta	Quando prevê positivo, raramente erra	Poucos **FP**
F1 alto	Bom balanço geral	Poucos FP e poucos FN

6.3 O Problema do Desequilíbrio de Classes

Técnicas para lidar com desequilíbrio:

Nível	Técnica	Descrição
Data Level	Undersampling	Remover amostras da classe maioritária
Data Level	Oversampling	Duplicar/criar amostras da classe minoritária
Data Level	SMOTE	Criar amostras sintéticas da minoria (mais diversas que duplicação simples)
Metric Level	Usar F1, AUC, Balanced Accuracy	Em vez de accuracy simples
Classifier Level	Penalização (class_weight)	Aumentar o custo de erros na classe minoritária
Classifier Level	One-Class Classification	Modelar só a classe minoritária; o resto é anomalia

Q14: Num problema de classificação binária com dados desequilibrados (90% classe A, 10% classe B): a) a accuracy é enganadora? b) que métricas usar?

a) Sim, a accuracy é enganadora. Um classificador que sempre prevê classe A tem accuracy=90% mas nunca deteta a classe B (Recall=0%). Parece bom mas é inútil para o objetivo real.

b) Métricas mais adequadas: Recall (fundamental se não detetar a classe B for perigoso), Precision (se os FP são custosos), F1-Score (balanço entre Precision e Recall), Balanced Accuracy (média do recall de cada classe), AUC-ROC (avaliação geral independente do threshold).

Para tratar o desequilíbrio: usar SMOTE/undersampling nos dados, ou penalizar o algoritmo com class_weight.

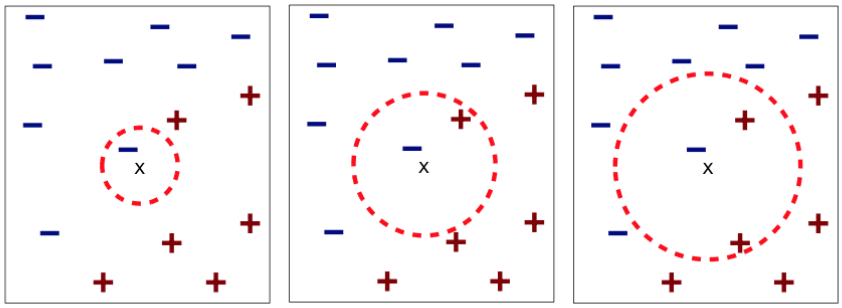
6.5 K-NN — K-Nearest Neighbors

Classificador não paramétrico e sem fase de treino. Para classificar um novo ponto, encontra os K vizinhos mais próximos no conjunto de treino e prevê a classe maioritária.

Prós	**Contras**
Simples e intuitivo	Lento na previsão (calcula todas as distâncias)
Não assume distribuição dos dados	Sensível à escala → **normalizar obrigatório**
Funciona com fronteiras não-lineares	Sensível a features irrelevantes/ruidosas
Versátil (classificação e regressão)	Difícil de usar com alta dimensionalidade

Quando usar:

- Tarefas simples de classificação ou regressão onde a **interpretabilidade** é importante e o dataset é pequeno
- Quando não tens **nenhuma suposição prévia** sobre a distribuição dos dados
- Dados **não-lineares** onde modelos lineares simples (regressão logística, SVM) falham
- Datasets **pequenos** onde o tempo de treino não é preocupação e a velocidade de previsão não é crítica



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

7. Cross-Validation, Bias vs Variância e Learning Curves

7.1 Divisão dos Dados

Método	Como funciona	Quando usar
Holdout	x% treino, (100-x)% teste (ex: 70/30)	Datasets grandes
Stratified Holdout	Mesma proporção de cada classe em treino e teste	Quando há desequilíbrio de classes
Leave-One-Out	Treinar com N-1 exemplos, testar com 1; repetir N vezes	Datasets muito pequenos
K-Fold CV	Dividir em K partes; K-1 para treino, 1 para teste; repetir K vezes	Datasets médios - mais robusto
Bootstrap	Amostragem com reposição para criar conjunto de treino	Estimativa de incerteza

Q17: Porque se usa K-Fold Cross-Validation em vez de uma simples divisão treino/teste?

A **divisão treino/teste** depende de como os dados foram divididos - uma divisão 'sortuda' pode dar resultados demasiado otimistas ou pessimistas. **K-Fold CV** usa o dataset K vezes: cada 'fold' serve como conjunto de teste uma vez, e os restantes K-1 como treino.

Vantagens: avaliação mais robusta e menos dependente da divisão; usa todos os dados para treino e teste; estima melhor a variância do desempenho.

Especialmente útil para datasets pequenos onde não se pode 'desperdiçar' dados num conjunto de teste fixo.

7.2 Processo de Modelação

Nunca **avalias o modelo com os mesmos dados com que o treinaste.**

Passo 1 - Treino Treinas vários modelos (ex: diferentes algoritmos ou diferentes configurações) nos dados de treino.

Passo 2 - Validação Testas esses modelos nos dados de validação. Escolhes o modelo com menor erro de validação. Este passo serve para **comparar e selecionar**.

Passo 3 - Teste Pegas no melhor modelo, re-treinas com treino+validação e avalias **uma única vez** no conjunto de teste. Este é o resultado final honesto.

7.3 Bias vs Variância

Q16: Explique o trade-off Bias-Variância e como identificar cada problema.

Bias é o erro sistemático do modelo - ocorre quando o modelo é demasiado simples e não captura os padrões reais (underfitting).

Variância é a sensibilidade do modelo às flutuações do conjunto de treino - ocorre quando o modelo é demasiado complexo e memoriza o ruído (overfitting).

Para identificar: **(1) High Bias**: erro alto TANTO no treino como na validação - as curvas de aprendizagem convergem mas para um platô alto;

(2) High Variance: erro baixo no treino mas muito mais alto na validação - grande gap entre as curvas.

Q: O teu modelo apresenta erros grandes em dados novos. O que fazes para melhorar?

Resposta: O primeiro passo é sempre **diagnosticar** o problema através das learning curves, comparando o erro de treino com o erro de validação, antes de tomar qualquer ação.

Se o erro for alto tanto no treino como na validação, o problema é **High Bias (underfitting)**, ou seja, o modelo é demasiado simples. Neste caso as ações a tomar são adicionar mais features relevantes, adicionar features polinomiais, usar um modelo mais complexo ou reduzir a regularização.

Se o erro for baixo no treino mas alto na validação, o problema é **High Variance (overfitting)**, ou seja, o modelo memorizou os dados de treino e não generaliza. Neste caso deve-se obter mais dados de treino, reduzir o número de features ou aumentar a regularização.

Em qualquer dos casos, pode-se ainda otimizar os hiperparâmetros com Grid Search ou experimentar algoritmos diferentes.

8. Ensemble Classifiers e Gestão de Modelos

8.1 Conceito de Ensemble

Em vez de usar um único classificador, combinar múltiplos modelos. A decisão final é obtida por agregação.

Porquê funciona: modelos individuais cometem erros diferentes; a combinação pode compensar as fraquezas individuais.

8.2 Bagging vs Boosting vs Stacking

Q18: Explique a diferença entre Bagging e Boosting, com exemplos.

Resposta: Bagging (ex: Random Forest): treina múltiplos modelos em paralelo, cada um em um subconjunto aleatório dos dados (bootstrap). A decisão final é por votação maioritária. Reduz principalmente a variância (overfitting). Robusto a ruído. -> **usa-se quando há high variance/overfitting**

Boosting (ex: XGBoost, AdaBoost): treina modelos sequencialmente. Cada modelo tenta corrigir os erros do anterior, aumentando o peso das amostras mal classificadas. Reduz bias e variância. Muito preciso mas sensível a ruído (pode overfitting com dados ruidosos) -> **usa-se quando há high bias/underfitting**.

8.4 Data Drift e Concept Drift

Nenhum modelo dura para sempre. O desempenho pode degradar-se com o tempo.

- **Data Drift:** a distribuição dos dados de input mudou (ex: comportamento de clientes pós-COVID)
- **Concept Drift:** a relação entre inputs e outputs mudou (ex: o que define 'fraude' evoluiu)
- Monitorização contínua do modelo é essencial em produção

9. Redes Neuronais Artificiais (ANN)

9.1 Porquê Redes Neuronais?

Para problemas com muitas features e relações não-lineares complexas (ex: reconhecimento de imagens).

As redes neuronais foram desenhadas para aprender representações não-lineares complexas de grandes volumes de dados.

9.2 Neurônio Artificial - Funcionamento

Estrutura base: Uma rede neuronal é composta por camadas de nós ligados entre si:

Input Layer → Hidden Layer(s) → Output Layer
(features) (aprende padrões) (previsão)

- **Input Layer:** uma camada com tantos nós quantas as features
- **Hidden Layers:** uma ou mais camadas intermediárias que aprendem representações
- **Output Layer:** uma camada com output adequado ao problema (1 neurônio para binário, K para multiclasse)

Deep Learning = redes com múltiplas hidden layers.

Q19: Qual a operação de um nó/'node' numa rede neuronal?

Cada nó realiza três operações:
(1) Ponderação - multiplica cada input pelo respetivo peso
(2) Soma - soma todos os inputs ponderados + bias;
(3) Ativação - aplica uma função de ativação ao resultado
Se a saída a exceder um certo threshold, o **nó 'dispara'** e passa o valor para os nós da camada seguinte.

9.3 Funções de Ativação

Q20: O que é uma função de ativação e porque é essencial?

Uma **função de ativação** é uma função matemática aplicada à saída de cada neurônio. É essencial porque **introduz não-linearidade na rede.** Sem estas, uma rede com múltiplas camadas seria equivalente a um único modelo linear, incapaz de aprender relações complexas). A escolha da função de ativação afeta: velocidade de treino, risco de vanishing gradient, e tipo de output.

10. Deep Learning e Redes Convolucionais (CNN)

Q:Descreve as diferencas entre deep learning and conventional ml

O Deep Learning é um subconjunto do ML que usa redes neuronais com múltiplas camadas.

A principal diferença está no tratamento das features: o **ML convencional** requer feature engineering manual**,** enquanto o **Deep Learning** aprende automaticamente as features diretamente dos dados brutos, sem intervenção humana.

O **ML convencional** funciona bem com datasets pequenos e é menos exigente computacionalmente, sendo também mais interpretável.

O **Deep Learning** requer grandes quantidades de dados e hardware poderoso, mas oferece performance superior em problemas complexos como imagens, texto ou séries temporais, onde os modelos convencionais falham. No entanto, é difícil interpretar como chegou a uma decisão, o que pode ser problemático em aplicações críticas.

10.2 CNN - Convolutional Neural Networks

CNNs são especialmente desenhadas para processar dados com estrutura em grelha (imagens). Exploram localidade espacial e invariância à translação.

Q21: Explique os componentes base de uma CNN e a sua função.

Uma CNN é composta por:

(1) **Camadas convolucionais**: aplicam filtros que deslizam pela imagem e produzem feature maps. Cada filtro deteta uma feature específica (é treinável e aprende automaticamente o que detetar)

(2) **Camadas de pooling**: reduzem o tamanho dos feature maps (downsampling), diminuindo parâmetros e controla overfitting.

(3) **Camadas fully connected (FC):** no final da rede, conectam todos os neurônios para fazer a classificação final com base nas features extraídas.

(4) **Softmax:** camada de output que converte os scores em probabilidades por classe.

11. Séries Temporais, RNN, LSTM e GRU

11.1 Séries Temporais - Porquê uma Metodologia Especial?

Dados de séries temporais são caracterizados pela sua natureza sequencial. Cada observação é influenciada pelas anteriores.

Estratégias para Séries Temporais

1. Divisão temporal correta: NUNCA dividir aleatoriamente. Treino = passado, Teste = futuro.

2. Feature engineering temporal: lag features, rolling statistics, indicadores sazonais.

3. Modelos específicos: ARIMA, SARIMA, LSTM, GRU.

4. Deteção de stationaridade: transformar séries não-estacionárias (ex: diferenciação).

Q22: O tratamento de séries temporais carece de uma metodologia especial? Justifique e apresente estratégias.

Sim, definitivamente. Dados de séries temporais violam o pressuposto de independência dos algoritmos tradicionais de ML. Cada observação é influenciada pelas anteriores, existindo autocorrelação.

Principais desafios: (1) **Divisão temporal:** não se pode dividir aleatoriamente (data leakage) — o treino deve ser sempre com dados anteriores ao teste.

(2) **Sazonalidade:** padrões que se repetem em períodos fixos.

(3) **Tendência:** crescimento/declínio de longo prazo.

(4) **Não-estacionaridade**: média e variância mudam ao longo do tempo

11.4 RNN - Recurrent Neural Networks

RNNs são desenhadas para dados sequenciais.

A característica principal é a memória: o hidden state de cada passo temporal é passado para o passo seguinte, permitindo capturar dependências temporais

Q23: Explique o problema do Vanishing Gradient em RNNs e como LSTM o resolve.

Em RNNs standard, durante o Backpropagation Through Time (BPTT), os gradientes são multiplicados por matrizes de pesos a cada passo temporal. Se esses pesos são < 1, os gradientes encolhem exponencialmente ao propagar para trás — as camadas iniciais recebem gradientes ~0 e deixam de aprender. Resultado: a rede esquece dependências de longo prazo. O LSTM resolve com o Cell State — uma 'memória de longo prazo' que percorre toda a sequência com apenas adições/multiplicações elementares (sem multiplicações de matriz), mantendo gradientes estáveis. Os gates (forget, update, output) com funções sigmoid controlam o fluxo de informação: o forget gate decide o que esquecer do passado; o update gate decide o que adicionar da informação atual. Isto permite à rede memorizar informação relevante por longos períodos.

Q24: Compare RNN, LSTM e GRU. Quando usar cada um?

Resposta: RNN standard: simples, mas sofre de vanishing gradient — adequado apenas para sequências curtas. LSTM: adiciona cell state e 3 gates para controlo fino da memória. Resolve vanishing gradient. Ideal para sequências muito longas onde dependências de longo prazo são importantes (ex: tradução de texto, previsão de séries longas). Mais parâmetros → mais lento. GRU: versão simplificada da LSTM com 2 gates. Performance comparável à LSTM na maioria dos casos, com menos parâmetros e treino mais rápido. Preferido para datasets menores ou quando velocidade é importante. Regra prática: começar com GRU; se o problema envolver dependências muito longas ou se a performance não for suficiente, experimentar LSTM.

12. Anomaly Detection & Ethics in AI

12.1 Anomaly Detection

Q: O que é Anomaly Detection e quais os seus tipos?

Anomaly Detection é o processo de identificar padrões ou pontos nos dados que desviam significativamente do comportamento normal, podendo indicar erros, fraude ou falhas. Existem três tipos:
Point Anomaly: um único ponto muito diferente dos restantes (ex: transação bancária anormal);
Contextual Anomaly: um ponto que é anómalo apenas num determinado contexto (ex: temperatura alta em inverno);
Collective Anomaly: um grupo de pontos que individualmente parecem normais mas que coletivamente revelam um padrão anómalo (ex: várias pequenas transações suspeitas seguidas).

12.4 Ética e Responsible AI

Onde entra o bias:

- **Dados** - dados históricos refletem desigualdades sociais; grupos marginalizados estão sub-representados
- **Labeling** - quem rotula os dados impõe a sua interpretação
- **Problem framing** - quem decide o que é um problema relevante?
- **Objetivos do modelo** - a métrica escolhida reflete valores (ex: accuracy esconde erros desiguais)
- **Deployment** - um sistema comporta-se diferente em laboratório vs. no mundo real

Q: O que é Responsible AI e quais os seus princípios fundamentais?

Responsible AI refere-se ao conjunto de práticas que tentam operacionalizar princípios éticos no desenvolvimento de sistemas de inteligência artificial. Os seus princípios fundamentais são: Fairness: garantir que o modelo não discrimina grupos; Reliability: o sistema funciona de forma consistente; Privacy: proteção dos dados pessoais; Transparency: as decisões devem ser explicáveis e auditáveis; Accountability: alguém deve ser responsável pelas decisões do sistema; e Sustainability: desenvolvimento sustentável a longo prazo. A ideia central é que os riscos éticos da AI não são primariamente falhas técnicas, mas falhas de governança e responsabilidade.