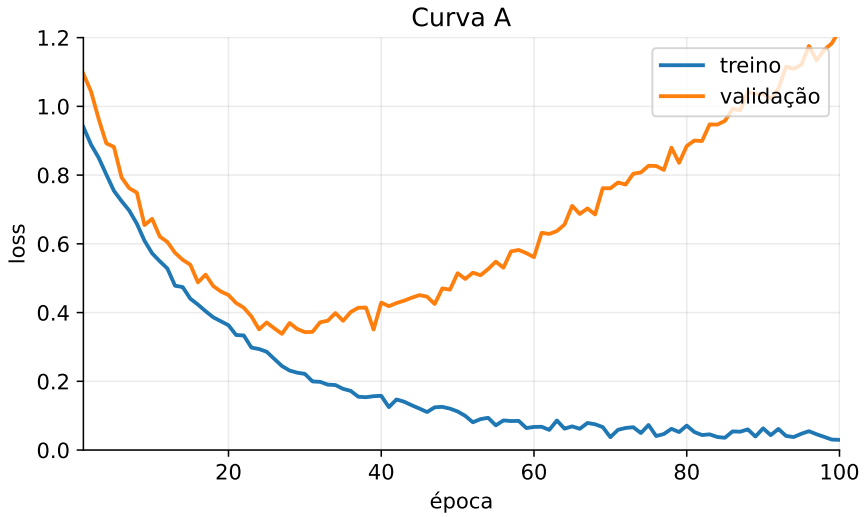
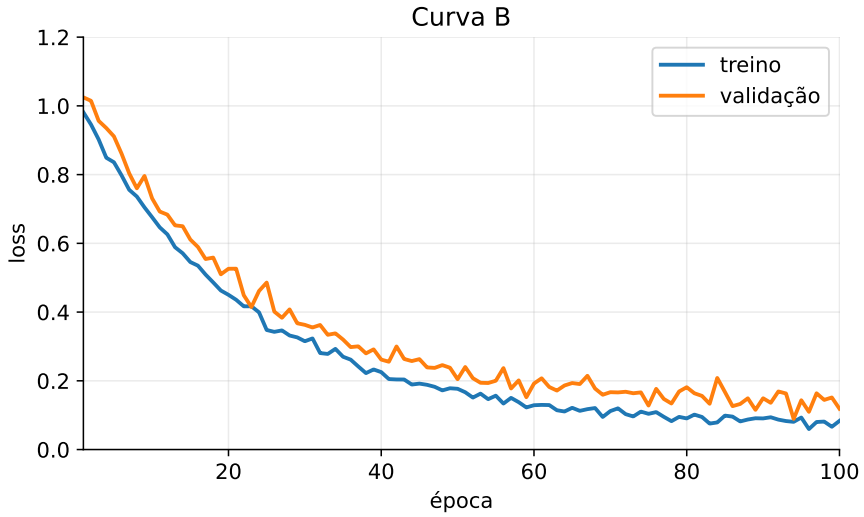
Exercício 1 — Overfitting e o trade-off bias-variance

Objetivos de aprendizagem

- Identificar visualmente a presença (ou ausência) de overfitting em curvas de loss de treino e validação.
- Discutir o trade-off bias-variance qualitativamente, sem recorrer à fórmula da decomposição.
- Relacionar flexibilidade do modelo, número de épocas e o ponto ótimo de *early stopping* / seleção de hiperparâmetros.

As figuras abaixo mostram duas curvas de treinamento de modelos de machine learning. Em cada figura, o eixo x representa o número de épocas (iterações de treino) e o eixo y representa a *loss* avaliada em dois conjuntos: o conjunto de treinamento (azul) e o conjunto de validação (laranja).





(a) Explique, em até dois parágrafos e com suas próprias palavras, o que é **overfitting**. Em particular, como ele se manifesta numa curva de treino vs. validação? Qual a relação intuitiva entre overfitting e a *flexibilidade* do modelo (por exemplo, o grau do polinômio em uma regressão polinomial)?

(b) Para cada uma das duas curvas (A e B), responda:

- Há indícios de overfitting? Justifique o que está vendo no gráfico (ou a ausência do que esperaria ver).
- Se houver, em torno de qual época o overfitting começa? O que seria a estratégia razoável de *early stopping* ali? *(Early stopping é a prática de interromper o treinamento antes do número máximo de épocas, tipicamente quando a loss de validação para de melhorar ou começa a piorar — evitando que o modelo continue se ajustando ao ruído do conjunto de treino.)*
- Se não houver, isso significa que o modelo está bem ajustado, ou poderíamos suspeitar de outro problema (e.g., *underfitting*)?

(c) Discuta as duas curvas em termos do **trade-off bias-variance**:

- Em qual fase de cada treinamento o **viés** é o termo dominante do erro de generalização? E em qual fase é a **variância**?
- Em que ponto da curva A o equilíbrio entre os dois é melhor, e o que isso significa para a escolha dos hiperparâmetros do modelo?
- A curva B parece estar mais para o lado de alto viés / baixa variância, ou alto viés baixo e variância também baixa? Justifique.

(d) Suponha agora que, em vez de "número de épocas" no eixo x, tivéssemos **flexibilidade do modelo** (por exemplo, o grau do polinômio em uma regressão polinomial). Em até um parágrafo, descreva qualitativamente como você esperaria que as curvas de loss em treino e em validação se comportassem ao longo desse eixo, e por quê.

Exercício 2 — Métricas a partir da matriz de confusão

Objetivos de aprendizagem

- Calcular accuracy, precision, recall e F1 a partir de uma matriz de confusão binária.
- Escolher a métrica apropriada para diferentes cenários (spam vs. screening médico).
- Distinguir entre precision *macro* e *micro* em problemas multi-classe.

Um classificador binário foi avaliado em um conjunto de teste de 200 observações. A matriz de confusão resultante é (positivo = classe rara):

	$\hat{y} = 1$ (predito positivo)	$\hat{y} = 0$ (predito negativo)
$y = 1$ (positivo)	$TP = 45$	$FN = 15$
$y = 0$ (negativo)	$FP = 20$	$TN = 120$

(a) Calcule **accuracy**, **precision**, **recall** e **F1-score** a partir dos quatro contadores. Apresente cada conta passo a passo.

(b) O mesmo classificador será aplicado em dois cenários distintos:

- **Filtro de *spam***: marcar erroneamente um e-mail legítimo como spam (FP) é uma falha grave para o usuário; deixar um spam passar (FN) é apenas um incômodo.
- ***Screening* de uma doença grave**: deixar de detectar um caso positivo (FN) é uma falha grave; um falso alarme (FP) gera um exame adicional, mas pouco mais.

Para cada cenário, indique qual das métricas (precision, recall, F1) você usaria como critério principal de seleção do classificador, e justifique em uma ou duas frases.

(c) Considere agora a versão multi-classe do mesmo problema, com 3 classes (A, B, C). A matriz de confusão observada é (linhas = verdade, colunas = predição), em um conjunto de teste de 100 exemplos:

	$\hat{y} = A$	$\hat{y} = B$	$\hat{y} = C$
$y = A$	40	5	5
$y = B$	4	25	1
$y = C$	5	5	10

Calcule **precision macro** e **precision micro**. Mostre as duas contas separadamente. Em qual situação prática você usaria cada uma? Por que os dois números são diferentes neste caso?

Exercício 3 — K-Fold à mão e em Python

Objetivos de aprendizagem

- Aplicar K-Fold cross-validation passo a passo em um dataset pequeno.
- Comparar K-Fold sem embaralhamento com Stratified K-Fold.
- Reproduzir o cálculo manual com a API do `sklearn`.

Considere um dataset com $n = 15$ amostras, divididas em 3 classes (A, B, C) com 5 amostras cada, indexadas pelos IDs 1 a 15. As amostras estão **ordenadas por classe**:

A: IDs 1–5 B: IDs 6–10 C: IDs 11–15

Aplicamos K-Fold cross-validation com $K = 5$ e `shuffle=False`. Como o dataset não é embaralhado, os folds de validação são contíguos:

Fold	IDs de validação
1	1, 2, 3
2	4, 5, 6
3	7, 8, 9
4	10, 11, 12
5	13, 14, 15

Em cada fold, o classificador usado é o `DummyClassifier(strategy='most_frequent')` do `sklearn`, que prevê sempre a classe **mais comum** no conjunto de treinamento. Em caso de empate, vence a classe com o menor rótulo (A < B < C).

(a) Para cada um dos 5 folds, preencha uma tabela com:
- As contagens de A, B, C no conjunto de **treinamento** (os 12 IDs fora do fold);
- A classe prevista pelo `DummyClassifier`;
- Os rótulos verdadeiros no conjunto de **validação**;
- O número de erros (0, 1, 2 ou 3) e a taxa de erro do fold (Erro_k).

(b) Calcule a estimativa final $\text{CV}_{(5)} = \frac{1}{5}\sum_{k=1}^{5} \text{Erro}_k$ para este K-Fold sem embaralhamento.

(c) Repita o cálculo usando **Stratified K-Fold** (com $K = 5$), que produz folds com exatamente **1 A, 1 B e 1 C** cada. Mostre o Erro_k por fold e a estimativa final $\text{CV}_{(5)}^{\text{strat}}$.

(d) Compare os dois números obtidos em (b) e (c). Em até dois parágrafos, explique a discrepância: por que o K-Fold *shuffle=False* produz uma estimativa tão pessimista neste dataset? E o que muda na estratificação?

(e) Em Python, reproduza os dois resultados usando:

```
import numpy as np
from sklearn.dummy import DummyClassifier
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score

X = np.arange(15).reshape(-1, 1)               # feature dummy
y = np.array(['A'] * 5 + ['B'] * 5 + ['C'] * 5)
clf = DummyClassifier(strategy='most_frequent')

kf  = KFold(n_splits=5, shuffle=False)
skf = StratifiedKFold(n_splits=5, shuffle=False)

# accuracy por fold; o "erro" do exercício é (1 - accuracy)
acc_kf  = cross_val_score(clf, X, y, cv=kf,  scoring='accuracy')
acc_skf = cross_val_score(clf, X, y, cv=skf, scoring='accuracy')
```

Verifique que os valores empíricos batem com os calculados nas partes (b) e (c).

Exercício 4 — Por que estratificar?

Objetivos de aprendizagem

- Quantificar via combinatória a probabilidade de folds patológicos no K-Fold não estratificado.
- Confirmar o cálculo teórico por simulação em Python.
- Entender a garantia algorítmica que o Stratified K-Fold oferece.

Considere um dataset com $n = 100$ amostras, das quais apenas 5 são positivas (classe minoritária) e 95 são negativas. Aplicamos 5-Fold cross-validation **sem estratificação**, dividindo o dataset aleatoriamente em 5 folds de 20 amostras cada (cada partição aleatória é equiprovável).

(a) Calcule a probabilidade de que **um fold específico** (digamos, o fold 1) não contenha nenhuma das 5 amostras positivas. Apresente o cálculo via

$$P(\text{fold 1 sem positivos}) \;=\; \frac{\binom{95}{20}}{\binom{100}{20}}.$$

Simplifique para uma forma fechada $\prod_{i=0}^{4} \frac{80-i}{100-i}$ e dê o valor numérico aproximado.

(b) Calcule a probabilidade de que **cada um dos 5 folds tenha exatamente 1 amostra positiva** — a distribuição que o Stratified K-Fold garante por construção. Use o argumento combinatório:

- O número total de maneiras de escolher 5 slots entre 100 para serem os positivos é $\binom{100}{5}$.
- O número de maneiras *favoráveis* (1 positivo em cada um dos 5 folds de 20 slots) é 20^5.

Apresente o cálculo e o valor numérico.

(c) A partir de (b), conclua a probabilidade de que **pelo menos um fold tenha zero amostras positivas** (em outras palavras, a probabilidade de o split aleatório falhar em representar a classe minoritária em algum fold).

(d) Verifique empiricamente com Python. Use o seguinte esqueleto, executando 10 000 repetições:

```
import numpy as np
from sklearn.model_selection import KFold, StratifiedKFold

rng = np.random.default_rng(0)
y = np.array([1] * 5 + [0] * 95)        # 5 positivos, 95 negativos

def at_least_one_fold_empty(splitter, X, y):
    for _, val_idx in splitter.split(X, y):
        if (y[val_idx] == 1).sum() == 0:
            return True
    return False

X = np.zeros((100, 1))
hits_kf, hits_skf = 0, 0
for i in range(10_000):
    if at_least_one_fold_empty(KFold(5, shuffle=True, random_state=i), X, y):
        hits_kf += 1
    if at_least_one_fold_empty(StratifiedKFold(5, shuffle=True, random_state=i), X, y):
        hits_skf += 1

print(f'KFold (não estratificado): {hits_kf / 10_000:.4f}')
print(f'StratifiedKFold:           {hits_skf / 10_000:.4f}')
```

Compare o número da primeira linha com o valor teórico de (c). Comente o resultado da segunda linha.