



Teoria do Aprendizado Estatístico

Avaliação 1

Luiz Henrique Barretta Francisco - 202100155302

maio/2025

Exercício 1 (ML): Diferencie os seguintes termos e utilize um exemplo:

- a. Função de perda e erro;
- b. Validação e data splitting;
- c. Overfitting e Complexidade;
- d. Risco e Risco empírico;
- e. Limite de generalização e coef. de shattering.

a) Função de perda e erro

Função de perda quantifica o quanto errado é uma única predição do modelo durante o treinamento. Erro é a função usada pelo algoritmo de otimização para ajustar parâmetros do modelo.

Erro é frequentemente expresso como uma curva ou taxa de erro e é uma medida de performance mais geral. O erro é calculado sobre um conjunto de dados (treino ou teste) para avaliar o desempenho de uma forma mais intuitiva, como a proporção de classificações incorretas.

Exemplo: Conjunto de dados: $(x_1, y_1) = (2, 3)$
 $(x_2, y_2) = (4, 5)$

Predições $\hat{y}_1 = 2.5$, $\hat{y}_2 = 5.5$.

Função de perda: Erro quadrático médio

$$L_1(y_1, \hat{y}_1) = (y_1 - \hat{y}_1)^2 = (3 - 2.5)^2 = 0.5^2 = 0.25$$

$$L_2(y_2, \hat{y}_2) = (y_2 - \hat{y}_2)^2 = (5 - 5.5)^2 = 0.5^2 = 0.25$$

Erro: Erro absoluto médio

$$E = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| = \frac{1}{2} (|3 - 2.5| + |5 - 5.5|) = \frac{1}{2} (0.5 + 0.5) = \frac{1}{2} (1) = 0.5$$

b) Validação e data splitting

Data splitting é o método de particionar um conjunto de dados em subconjuntos menores, geralmente de treino, validação e teste.

Validação é o processo de usar essas partições dos dados para calcular os parâmetros e estruturas do modelo. Isso é feito através do modelo no conjunto de validação depois de ter treinado no conjunto de treino.

Exemplo: Considere um conjunto de dados Z com 1000 observações, dividido em Z_{treino} com 900 observações e Z_{teste} com 100 observações de tal forma que $Z_{\text{treino}} \cap Z_{\text{teste}} = \emptyset$ e $Z_{\text{treino}} \cup Z_{\text{teste}} = Z$.

A validação consiste em dizer que o modelo está sendo ajustado re a função de perda (L) em círculos (Z_{treino}) e $L(Z_{\text{teste}})$.

c) Overfitting e complexidade

Overfitting (sobreajuste) ocorre quando um modelo é muito complexo e não aprende realmente os padrões mas também se ajusta expecificamente ao conjunto de teste, resultando num desempenho ruim no conjunto de teste.

Complexidade refere-se à capacidade do modelo de reconhecer padrões nos dados. Modelos mais complexos (como polinomiais de grau alto ou árvores de decisão profundas) conseguem captar relações mais intrincadas.

Exemplo: Considere 5 pontos de dados (x_i, y_i)

Modelo de baixa complexidade: $f_1(x) = \beta_0 + \beta_1 x$.

Modelo de alta complexidade $f_2(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$

Overfitting: Se houver novos pontos de teste, o modelo de baixa complexidade, a regressão linear (baixa complexidade) pode se ajustar melhor a esses novos dados do que o modelo de alta complexidade que define uma curva que passa por todos os 5 dados e nulos de três novos

d) Risco e risco empírico

Risco (ou risco esperado) de um modelo é o valor esperado da função de perda sobre a distribuição, ou verdadeira e desconhecida, é o erro que seria cometido em média em todo a população.

Risco empírico é a média da função de perda calculada sobre o conjunto de dados de treinamento disponível. É uma aproximação do risco verdadeiro.

Exemplo: Considere uma função de perda de classificação binária involucrando 3. Sendo $(x_i, y_i) = \{(A, 0), (B, 1), (C, 0)\}$ e $y = (0, 0, 1)$.

$$\text{Risco empírico}(f) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i)) = \frac{1}{3} [1(0 \neq 0) + 1(1 \neq 1) + 1(0 \neq 1)] = \frac{1}{3}[0 + 1 + 1] = 2/3.$$

$$\text{Risco: } R(f) = E(x, y) \sim P[L(y, f(x))] = \int \sum L(y, f(x)) dP(x, y)$$

Esse valor não pode ser calculado para $P(x, y)$ é desconhecida.

e) Limite de generalização e coeficiente de shattering

Limite de generalização é uma das qualidades matemática que relaciona o risco verdadeiro $R(f)$ com o risco empírico $R_{emp}(f)$ mais um termo que depende da complexidade da classe de modelos G (do qual $f \in G$ é uma) e da amostra N do conjunto de treino.

O coeficiente de shattering $M(G, n)$ também chamado de função de crescimento é o número máximo de classificações (maneiras de rotular os n pontos em duas classes) que podem ser geradas pela classe G .

limite de generalização: $R(g_{ue}) \leq R_{emp}(g_{ue}) + \sqrt{\frac{4}{n} (\log 2 M(6, 2n) - \log S)}$

Seja $n = 1000$; $R_{emp}(g_{ue}) = \# \text{erros} = \frac{5}{1000} = 0.05$; $S = 0.05$.

$M(6, N) \approx N^{dvc}$; $M(6, 2000) \approx 2000^3$. Então:

$$R(g_{ue}) \leq 0.05 + \sqrt{\frac{4}{1000} (\log 2 (2000^3) - \log 0.05)}$$

$$R(g_{ue}) \leq 0.05 + \sqrt{\frac{4}{1000} (23.496 - (-2.9957))}$$

$$R(g_{ue}) \leq 0.05 + \sqrt{\frac{4}{1000} (26.4977)}$$

$$R(g_{ue}) \leq 0.05 + \sqrt{0.1059668} \Rightarrow R(g_{ue}) \leq 0.05 + 0.3255$$

$$R(g_{ue}) \leq 0.3755.$$

Coeficiente de shattering $M(6, n)$

Considere $f(x)$ uma classe linear unidimensional à seguir. $f(x) : \mathbb{R}^1 \rightarrow \mathbb{R} \leq a$ e $x_1 < x_2$.

Existem $2^n = 2^2 = 4$ maneiras de rotular os dados: $(x_1, x_2) \Rightarrow (-1, -1) (-1, 1) (1, -1) \cancel{(1, 1)} (1, 1)$.

Portanto $M(f(x), 2) = 3$.

Exercício 2: Escolha dois métodos de aprendizado de máquina diferentes de Reg. Logística e KNN. Pesquise sobre o coef. de shattering de cada um. Crie um texto explicativo sobre sua pesquisa e vincule com o que foi visto na disciplina.

Redes Neurais: A capacidade de aprendizado de uma rede neural, e também seu coeficiente de shattering, dependem diretamente da sua arquitetura: o número de camadas, a quantidade de nós por camada e as funções de ativação utilizadas. Para uma rede neural mais simples, definida com W pesos (parâmetros), U unidade de computação (também chamadas de neurônios ou nós), L camadas e com uma função de ativação não polinomial, como a *ReLU* (Rectified Linear Unit), definida por $\text{ReLU}(x) = \max(0, x)$, foi provado por Bartlett *et al.* (2019) a existência dos seguintes limites para a dimensão *Vapnik-Chervonenkis* (VC), denotada como $VCdim(F)$:

- **Limite Superior:**

$$VCdim = O(WL \log W)$$

- **Limite Inferior:**

$$VCdim = \Omega(WL \log \frac{W}{L})$$

Utilizando o limite superior acima, podemos inferir que o coeficiente de *shattering* para a rede neural *ReLU* satisfaz:

$$\mathcal{M}(F, n) \leq (n + 1)^{cWL \log W}$$

onde c é uma constante implícita na notação $O(\cdot)$.

Isso demonstra que a riqueza da classe de funções da rede, e portanto sua capacidade de classificar conjuntos de dados de diversas maneiras, cresce de acordo com n , e sendo a sua taxa de crescimento governada exponencialmente pela dimensão VC, a qual, por sua vez, está ligada às características de W pesos e L camadas da rede. É importante notar que no artigo referenciado, limites mais estreitos ou largos para a dimensão VC também foram definidos e podem ser obtidos ao se especificar com mais detalhes as características da rede neural. No caso apresentado, utilizou-se um modelo de rede neural mais simplificado.

Bartlett, P. L., Harvey, N., Liaw, C., & Mehrabian, A. (2019). *Nearly-tight VC-dimension bounds for piecewise linear neural networks*. *Journal of Machine Learning Research*, 20(63), 1–17. Disponível em: <https://jmlr.org/papers/volume20/17-612/17-612.pdf>

Árvores de Decisão: A capacidade de generalização de uma árvore de decisão, e também sua complexidade combinatória, podem ser caracterizadas pela dimensão VC. Essa dimensão depende da estrutura da árvore: o número de folhas L_T , o número de atributos reais l utilizados e o tipo das regras de decisão.

Recentemente, Leboeuf *et al.* (2022) apresentaram resultados para a dimensão VC em árvores de decisão binárias que realizam divisões nos eixos em atributos reais. Os autores mostraram que, para uma árvore de L_T folhas e l atributos reais, a dimensão VC da classe de funções correspondente é assintoticamente limitada por:

- **Limite Superior:**

$$VCdim = O(L_T \log(L_T \cdot l))$$

Esse resultado foi obtido por meio de um modelo recursivo baseado em funções de particionamento, assim generalizando o comportamento para árvores arbitrárias. Utilizando um resultado clássico de Mansour (1997), sabe-se que existe um limite inferior para a dimensão VC em função do número de nós internos N , onde $L_T = N + 1$:

- **Limite Inferior:**

$$VCdim = \Omega(L_T)$$

Portanto, a dimensão VC cresce ao menos linearmente com o número de folhas e, no pior caso, até proporcionalmente a $L_T \log(L_T l)$.

A partir do Lema de Sauer-Shelah, podemos inferir um limite superior para o coeficiente de *shattering*, que representa o número máximo de rotulações distintas que a classe F pode implementar sobre n exemplos:

$$\mathcal{M}(F, n) \leq (n + 1)^{c \cdot L_T \log(L_T l)}$$

onde c é uma constante implícita de $O(\cdot)$.

Esse crescimento exponencial do coeficiente de *shattering* em relação ao número de n evidencia a complexidade da classe de funções representada pela árvore de decisão. Tal crescimento é controlado pela dimensão VC, que por sua vez é governada pela profundidade estrutural da árvore e pela complexidade das regras de divisão utilizadas.

Leboeuf, J.-S., LeBlanc, F., & Marchand, M. (2022). *Generalization Properties of Decision Trees on Real-valued and Categorical Features*. Journal of Machine Learning Research, 1–81. Disponível em: [https://arxiv.org/pdf/2210.10781](https://arxiv.org/pdf/2210.10781.pdf)

Mansour, Y. (1997). *Pessimistic Decision Tree Pruning Based on Tree Size*. Fourteenth International Conference on Machine Learning (pp. 195–201). Morgan Kaufmann. Disponível em: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b6fce37612db10a9756b904b5e79e1144ca12574>

Exercício 3 (ML): Leia artigo RAPER, Simon. Leo Breiman's "two cultures". Significance, v. 17, n. 1, p. 34-37, 2020. Acesso em: (<https://rss.onlinelibrary.wiley.com/doi/epdf/10.1111/j.1740-9713.2020.01357.x>). Argumente relacionando o artigo com os conteúdos de aprendizado de máquina vistos na disciplina.

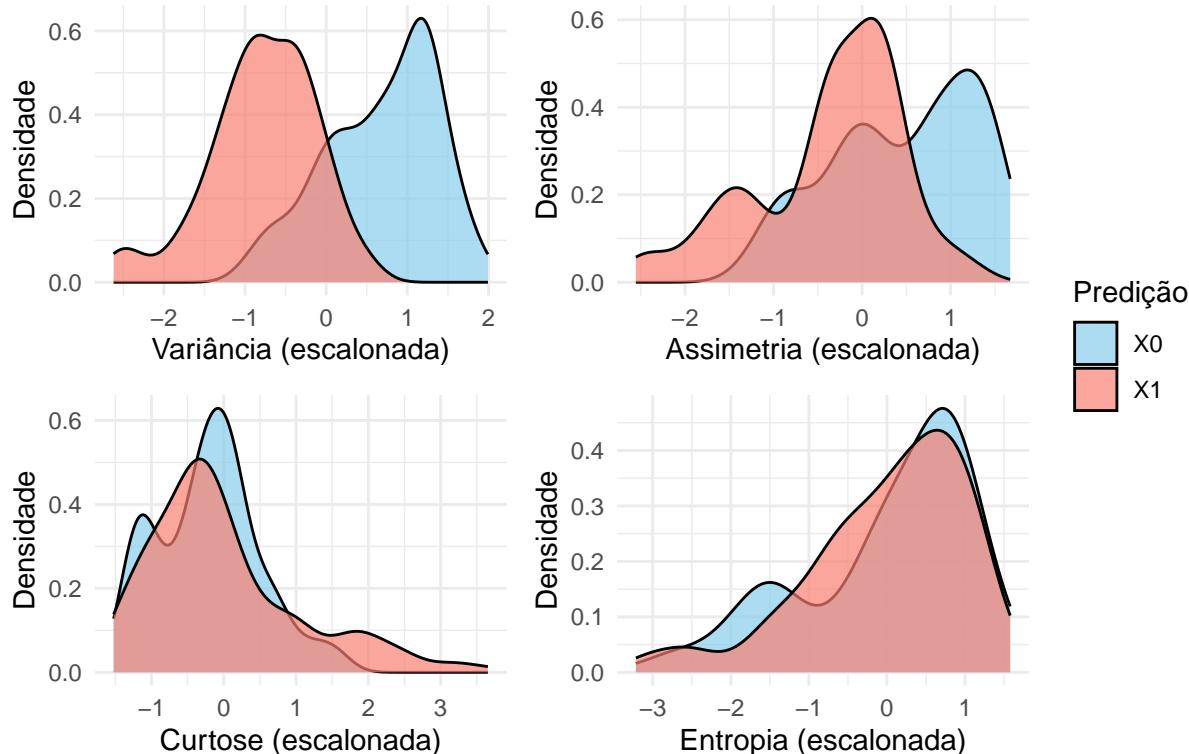
3)

O artigo "Leo Breiman's Two Cultures", publicado na revista *Significance* de 2020 e escrito por Simon Raper, revisita as ideias do clássico artigo de 2001 de Breiman sobre as duas culturas na análise de dados. Nele, Breiman distingue a cultura de modelagem de dados, focada em modelos explicativos, da cultura de modelagem algorítmica, que prioriza a capacidade preditiva. A ênfase de Breiman na previsão como critério de medição e sucesso alinha-se com a fase preditiva do aprendizado de máquina deslocada para as aulas. Sua crítica ~~à~~ à excessiva dependência de modelos específicos tem como real propósito a alerta relevante, e o "efeito Borromean" ilustra a importância de calibrar modelos, como visto nas técnicas de "data splitting" e validações cruzadas vistos na disciplina.

A cultura algorítmica, que Breiman defende, abrange a complexidade e a ideia de "tecidos-pjetos", se elas fornecerem melhores previsões, e que é visível na variação de algoritmos como KNN e SVMs, onde a interpretabilidade pode ser sacrificada pela performance. Breiman incentiva o uso de um círculo vicioso de ferramentas, escolhendo a mais adequada ao problema, refletindo na aplicação das diversas métodos em aula. Ele também adverte contra a campanha e a aplicação ~~à~~ acrítica de qualquer método, um ponto crucial mesmo com a ascensão da cultura algorítmica. A discussão sobre a trade-off entre ciência e computação e a minimização de riscos empíricos são fundamentais para construir modelos que generalizem bem, um objetivo central para ambas as culturas, mas desafiador com diferentes prioridades. A busca por funções que minimizem os riscos é o cerne do aprendizado supervisionado, seja para classificação ou regressão.

Exercício 4: Pesquise duas métricas de capacidade preditiva não vistas em sala de aula para o contexto de classificação. Utilize os dados banknote (<https://archive.ics.uci.edu/dataset/267/banknote+authentication>) e o classificador KNN (<https://archive.ics.uci.edu/dataset/267/banknote+authentication>)) e o classificador KNN.

Distribuições Marginais das Features – KNN (k = 33)



O gráfico acima apresenta as distribuições de densidade marginal para cada uma das quatro variáveis preditoras do conjunto de dados (variância, assimetria, curtose e entropia), condicionadas às classes (“X0” e “X1”) preditas pelo modelo KNN no conjunto de teste. Ajustou-se um KNN com $k=33$, um conjunto de treinamento correspondente a 75% das observações e com normalização das variáveis.

Matriz de Confusão – KNN (k = 33)



Este gráfico exibe a matriz de confusão aplicada no modelo KNN ajustado. Os valores obtidos de Verdadeiros Negativos (TN), Verdadeiros Positivos (TP), Falsos Negativos (FN) e Falsos Positivos (FP), visualizados nesta matriz, servirão de base para o cálculo das duas novas métricas de capacidade preditiva (Coeficiente Kappa de Cohen e Razão de Chances Diagnóstica - DOR) que serão exploradas em seguida:

O Coeficiente Kappa de Cohen é calculado pela fórmula:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

onde: P_o é a concordância observada, proporção de acertos ou acurácia:

$$P_o = \frac{TP + TN}{N}$$

sendo $N = TP + TN + FP + FN$, o número total de observações.

P_e é a concordância esperada por acaso. Para um problema de classificação binária como é o caso, é calculada como a soma da probabilidade de ambos (modelo e real) classificarem como positivo por acaso e a probabilidade de ambos classificarem como negativo por acaso:

$$P_e = P_{\text{positivos por acaso}} + P_{\text{negativos por acaso}}$$

$$P_e = \left(\frac{TP + FP}{N} \times \frac{TP + FN}{N} \right) + \left(\frac{FN + TN}{N} \times \frac{FP + TN}{N} \right)$$

TP: 152 TN: 188 FP: 2 FN: 0

Coeficiente Kappa de Cohen: 0.9882

O Coeficiente Kappa de Cohen obtido para o modelo KNN foi de 0.9882. Esta métrica avalia a concordância entre as classes preditas pelo modelo e as classes reais, levando em consideração a concordância que poderia

ocorrer puramente ao acaso. Um valor de Kappa como o observado, bem superior a 0.8, indica um acordo quase perfeito entre as previsões do modelo e os valores verdadeiros, sugerindo que o classificador possui um desempenho excelente e consistente além da simples sorte.

A Razão de Chances de Diagnóstico é calculada como a razão entre as chances de um teste (no nosso caso, a previsão do modelo) ser positivo em um indivíduo que realmente tem a condição (classe positiva) e as chances de um teste ser positivo em um indivíduo que não tem a condição (classe negativa). A fórmula é:

$$DOR = \frac{\frac{TP}{FN}}{\frac{FP}{TN}}$$

Simplificando, obtemos:

$$DOR = \frac{TP \times TN}{FP \times FN}$$

É importante notar que, se FP ou FN (ou ambos) forem zero, a DOR pode se tornar indefinida ou infinita. Nesses casos, uma prática comum é adicionar uma pequena constante (como 0.5) a todas as células da matriz de confusão antes de se calcular para obter um valor finito e estável.

Razão de Chances de Diagnóstico (DOR): 22997

A Razão de Chances de Diagnóstico calculada para o modelo, resultou em um valor extraordinariamente alto de 22997. A DOR quantifica o poder discriminatório do modelo, indicando o quanto mais provável é uma previsão correta de uma classe em relação a uma incorreta, comparando as chances de acerto para as duas classes. Um valor tão elevado (muito acima de 100) sugere uma capacidade discriminatória excepcional do modelo KNN em distinguir entre as notas verdadeiras e falsificadas, especialmente porque o número de Falsos Negativos (FN) foi zero, o que impulsionou significativamente esta métrica.

Acurácia: 0.9942

Sensibilidade (Recall para classe 'X1'): 1

Especificidade: 0.9895

Precisão (VPP para classe 'X1'): 0.987

F1-Score para classe 'X1'): 0.9935

Acima temos outras métricas de desempenho preditivas que corroboram com o bom ajuste oferecido por esse modelo.

Exercício 5: Escolha um conjunto de dados para classificação multiclasse (diferente do wallrobot-navigation) e qualquer classificador de seu interesse. Aplique os métodos de validação holdout, holdout repetido, k-fold, k-fold repetido e leave-one-out. Compare os resultados.

Tabela 1: Comparação das Métricas de Desempenho do KNN por Método de Validação

Método	Accuracy	Kappa	Sens.M	Espec.M	PPV.M	NPV.M	AcurBal.M	F1.M
Holdout	0.6857	0.5809	0.6885	0.8950	0.6721	0.8999	0.7917	0.6723
RepeatedHoldout	0.7100	0.6131	0.7126	0.9030	0.7015	0.9054	0.8078	0.7037
KFold	0.7269	0.6357	0.7291	0.9088	0.7254	0.9108	0.8189	0.7218
RepeatedKFold	0.7219	0.6291	0.7240	0.9071	0.7161	0.9093	0.8156	0.7155
LOOCV	0.7305	0.6407	0.7328	0.9100	0.7179	0.9126	0.8214	0.7219

A tabela acima mostra os resultados do classificador KNN no dataset ‘Vehicle’, o qual é utilizado para

classificar veículos em quatro categorias distintas com base em 18 características numéricas dos carros. Os métodos de reamostragem como tendem a fornecer estimativas das métricas de desempenho mais estáveis e confiáveis que o Holdout simples, cuja estimativa é mais sensível à divisão particular daqueles dados. A consistência nos resultados entre os métodos de reamostragem mais robustos, incluindo o LOOCV, sugere uma avaliação de desempenho mais fidedigna. Para datasets multiclasse como o ‘Vehicle’, métricas como Acurácia Balanceada Média e F1 Médio são particularmente importantes para entender a generalização do modelo entre as diferentes classes, que podem ser desbalanceadas.

Considerando um equilíbrio entre a confiabilidade da estimativa de desempenho, a variância dessa estimativa e o custo computacional, o método de k-Fold Repetido (no exemplo, um *10-Fold* repetido 3 vezes) se destaca como a melhor escolha para uma avaliação robusta. Ele melhora a estabilidade de um único k-Fold ao mitigar os efeitos de uma partição específica e é, na maioria das vezes, mais prático computacionalmente que o LOOCV em datasets maiores, embora o LOOCV ofereça uma estimativa com baixo viés.

Exercício 6 (ML): Considere $n = 3$ observações $Y = 0, 1$ e $\mathcal{X} = \mathbb{R}$, seja $G = \{I_{[a,b)}(x) \mid a < b \in \mathbb{R}\}$. $I_A(x) = 1$ se $x \in A$ e 0 c.c.. Calcule o coeficiente de shattering $\mathcal{M}(G, n)$.

6) Calcular $M(G, 3)$ com $G = \{I(a)\mid a \in [a, b]\}$

Sejam os 3 pontos $x_1 < x_2 < x_3$.

$g \in G$ classifica x_i como 1 se $x_i \in [a, b]$ e 0 caso contrário.

Pois $n=3$ existem $2^3 = 8$ classificações possíveis

1. $(0, 0, 0)$: $a = x_3 + 2$ e $b = x_3 + 3$. $\forall x_i \notin [a, b]$

2. $(0, 0, 1)$: $a = x_3$ e $b = x_3 + 1$. Apenas $x_3 \in [a, b]$

3. $(0, 1, 0)$: $a = x_2$ e $b = (x_2 + x_3)/2$. Apenas $x_2 \in [a, b]$

4. $(0, 1, 1)$: $a = x_2$ e $b = x_3 + 1$. $x_2, x_3 \in [a, b]$

5. $(1, 0, 0)$: $a = x_1$ e $b = (x_1 + x_2)/2$. Apenas $x_1 \in [a, b]$

6. $(1, 0, 1)$: Classificação impossível

Se um intervalo $[a, b] \in (x_1, x_3)$ com $x_1 < x_3$ entâo $a \leq x_1$ e $x_3 \leq b$.

Dado que $x_1 < x_2 < x_3$ e $a \leq x_1 \leq x_3 \leq b$ implica

que $a \leq x_1 < x_2 < x_3 \leq b$. Portanto, x_2 deve

estar no intervalo $[a, b]$. Não é possível construir a classificação $(1, 0, 1)$ com a 6 dada e o intervalo $[a, b]$.

7. $(1, 1, 0)$: $a = x_1$ e $b = (x_2 + x_3)/2$. $x_1, x_2 \in [a, b]$. $x_3 \notin [a, b]$

8. $(1, 1, 1)$: $a = x_1$ e $b = x_3 + 1$. $x_1, x_2, x_3 \in [a, b]$.

Das 8 dicatões, 7 são possíveis.

$M(G, 3) = 7$. Pelo lema de Sauer-Shelah

$$M(G, n) \leq \sum_{i=0}^d \binom{n}{i}$$

$$M(G, 3) \leq \sum_{i=0}^2 \binom{3}{i}$$

$$M(G, 3) \leq \binom{3}{0} + \binom{3}{1} + \binom{3}{2} = 7.$$

Exercício 7 (ML): Mostre matematicamente que o princípio de minimização do risco empírico não é consistente para o coef. de shattering com crescimento exponencial. Interprete o resultado.

7) Mostrar que $M(G, m)$ exponencial e-moço consistente.

(aplicando o lema de shattering exponencial)
 $M(G, m) = 2^m \sqrt{m}$.

Pelo lema da simetrização, temos a desigualdade:

$$\begin{aligned} P(\sup_{g \in G} |R_{\text{emp}}(g) - R(g)| > \epsilon) &\leq 2M(G, 2m) \exp(-m \frac{\epsilon^2}{4}) \\ &\leq 2 \cdot 2^{2m} \exp(-m \frac{\epsilon^2}{4}) \\ &\leq 2 \cdot (e^{\ln 2})^{2m} \exp(-m \frac{\epsilon^2}{4}) \\ &\leq 2 \exp(2m \ln 2) \exp(-m \frac{\epsilon^2}{4}) \end{aligned}$$

$$P(\sup_{g \in G} |R_{\text{emp}}(g) - R(g)| > \epsilon) \leq 2 \exp(m(2 \ln 2 - \frac{\epsilon^2}{4}))$$

Para a probabilidade de tender a 0 então o expoente $m(2 \ln 2 - \frac{\epsilon^2}{4})$ deve tender a $-\infty$.

Como m é positiva, então a parte $2 \ln 2 - \frac{\epsilon^2}{4}$ deve ser negativa.

$$2 \ln 2 - \frac{\epsilon^2}{4} < 0 \Rightarrow \frac{\epsilon^2}{4} > 2 \ln 2 \Rightarrow \epsilon > \sqrt{8 \ln 2}$$

$$\text{Se } 0 < \epsilon < \sqrt{8 \ln 2} \text{ então } 2 \ln 2 - \frac{\epsilon^2}{4} \geq 0.$$

Portanto no $\lim_{n \rightarrow \infty} 2 \exp(m(2 \ln 2 - \frac{\epsilon^2}{4})) \neq 0$. para um ϵ escolhido entre $0 < \epsilon < \sqrt{8 \ln 2}$.

Exercício 8 (ML): Justifique a utilização de métodos de classificação binária com base na Teoria do Aprendizado Estatístico.

8)

A utilização de métodos de classificação binária é justificada pelo teorema de Aprendizagem Estatística para ela se basear numa função que minimiza o erro da classificação.

Essa função de perda L é usada para encontrar um classificador $g(x)$ que, para observações x , estime corretamente a qual das duas classes de y (0 e 1) essa classe pertence.

A teoria também se preocupa em garantir que o risco empírico $R_{emp}(g) = \frac{1}{n} \sum L(y_i, g(x_i))$ rejeite um bom critério de risco verdadeiro $R(g) = E_{x,y}[L(y, g(x))]$, permitindo assim sua generalização, ou seja, que $|R_{emp}(g) - R(g)|$ rejeite um valor pequeno.

Isto é garantido ao manter a complexidade da classe de funções G adequada e não muito grande, através de uma dimensão Vapnik-Chervonenkis finita (~~dim(G)~~) ($VC(G) < \infty$). Assim, assegura-se que o modelo aprendido se aproxima realista das regras particulares do amostra, rende robustez entre a generalização.

O classificador de Bayes, $g(x) = \arg \max P(Y=c|x)$ é o ótimo teoricamente que se busca aproximando-o iterativamente, calculado por $P(R(g) - R(g^*) > \epsilon) \rightarrow 0$ quando $n \rightarrow \infty$) garante que, com n grande, o classificador g obtido terá convergência para a melhor solução g^* dentro da classe de funções G .

Códigos

```
#Exercício 4
library(readr)
library(caret)
library(class)
library(dplyr)
library(ggplot2)
library(patchwork)

col_names <- c("variance", "skewness", "curtosis", "entropy", "class")
banknote_data <- read_csv("data_banknote_authentication.txt",
                           col_names = col_names, col_types = cols())

banknote_data$class <- as.factor(banknote_data$class)
levels(banknote_data$class) <- make.names(levels(banknote_data$class))

set.seed(123)
# Data Splitting (75%) e teste (25%)
trainIndex <- createDataPartition(banknote_data$class, p = 0.75, list = FALSE, times = 1)
train_data <- banknote_data[trainIndex, ]
test_data <- banknote_data[-trainIndex, ]

train_features <- train_data[, -5]
train_labels <- train_data$class
test_features <- test_data[, -5]
test_labels <- test_data$class

# Normalização
preProcValues <- preprocess(train_features, method = c("center", "scale"))
train_features_scaled <- predict(preProcValues, train_features)
test_features_scaled <- predict(preProcValues, test_features)

# KNN
k_val <- floor(sqrt(nrow(train_data)))
if (k_val %% 2 == 0) k_val <- k_val + 1
knn_pred <- knn(train = train_features_scaled, test = test_features_scaled,
                  cl = train_labels, k = k_val)
pos_class <- make.names("1")

plot_data_dist <- as.data.frame(test_features_scaled)
plot_data_dist$predicted_class <- knn_pred
pred_colors <- c("X0" = "skyblue", "X1" = "salmon")

p_variance <- ggplot(plot_data_dist, aes(x = variance, fill = predicted_class)) +
  geom_density(alpha = 0.7) + scale_fill_manual(values = pred_colors, name = "Predição") +
  labs(x = "Variância (escalonada)", y = "Densidade") +
  theme_minimal() + theme(plot.title = element_text(hjust = 0.5, size = 10),
                          legend.position = "none")

p_skewness <- ggplot(plot_data_dist, aes(x = skewness, fill = predicted_class)) +
  geom_density(alpha = 0.7) + scale_fill_manual(values = pred_colors, name = "Predição") +
  labs(x = "Assimetria (escalonada)", y = "Densidade") +
  theme_minimal() + theme(plot.title = element_text(hjust = 0.5, size = 10),
```

```

            legend.position = "none")

p_curtosis <- ggplot(plot_data_dist, aes(x = curtosis, fill = predicted_class)) +
  geom_density(alpha = 0.7) + scale_fill_manual(values = pred_colors, name = "Predição") +
  labs(x = "Curtose (escalonada)", y = "Densidade") +
  theme_minimal() + theme(plot.title = element_text(hjust = 0.5, size = 10),
                           legend.position = "none")

p_entropy <- ggplot(plot_data_dist, aes(x = entropy, fill = predicted_class)) +
  geom_density(alpha = 0.7) + scale_fill_manual(values = pred_colors, name = "Predição") +
  labs(x = "Entropia (escalonada)", y = "Densidade") +
  theme_minimal() + theme(plot.title = element_text(hjust = 0.5, size = 10))

combined_dist_plot <- (p_variance | p_skewness) / (p_curtosis | p_entropy) +
  plot_layout(guides = 'collect') +
  plot_annotation(
    title = paste("Distribuições Marginais das Features - KNN (k =", k_val, ")"),
    theme = theme(plot.title = element_text(hjust = 0.5, size = 16, face="bold")))
print(combined_dist_plot)

cm <- confusionMatrix(data = knn_pred, reference = test_labels, positive = pos_class)
cm_table <- as.data.frame(cm$table)
names(cm_table) <- c("Prediction", "Reference", "Frequency")
plot_title <- paste("Matriz de Confusão - KNN (k =", k_val, ")")

#Exercício 5

library(mlbench)
data(Vehicle, package = "mlbench")
vehicle_data <- Vehicle
levels(vehicle_data$Class) <- make.names(levels(vehicle_data$Class))

df_metrics <- list()
metrics <- c("Accuracy", "Kappa", "Mean_Sensitivity", "Mean_Specificity", "Mean_Pos_Pred_Value",
            "Mean_Neg_Pred_Value", "Mean_Balanced_Accuracy", "Mean_F1")

train_index_holdout <- createDataPartition(vehicle_data$Class, p = 0.75, list = FALSE, times = 1)
train_data_holdout <- vehicle_data[train_index_holdout, ]
test_data_holdout <- vehicle_data[-train_index_holdout, ]
target_col_name <- "Class"
predictor_names <- setdiff(names(vehicle_data), target_col_name)
train_features_holdout <- train_data_holdout[, predictor_names]
train_labels_holdout <- train_data_holdout[[target_col_name]]
test_features_holdout <- test_data_holdout[, predictor_names]
test_labels_holdout <- test_data_holdout[[target_col_name]]

# Normalização
preproc_holdout <- preProcess(train_features_holdout, method = c("center", "scale"))
train_features_holdout_scaled <- predict(preproc_holdout, train_features_holdout)
test_features_holdout_scaled <- predict(preproc_holdout, test_features_holdout)

# Holdout
k_holdout <- floor(sqrt(nrow(train_features_holdout_scaled)))

```

```

if (k_holdout %% 2 == 0) k_holdout <- k_holdout + 1

predictions_holdout <- knn(train = train_features_holdout_scaled, test = test_features_holdout_scaled,
                           cl = train_labels_holdout, k = k_holdout)

# Calcular Métricas
cm_holdout <- confusionMatrix(data = predictions_holdout, reference = test_labels_holdout)
holdout_metrics_vector <- c(
  cm_holdout$overall[c("Accuracy", "Kappa")],
  Mean_Sensitivity = mean(cm_holdout$byClass[, "Sensitivity"], na.rm = TRUE),
  Mean_Specificity = mean(cm_holdout$byClass[, "Specificity"], na.rm = TRUE),
  Mean_Pos_Pred_Value = mean(cm_holdout$byClass[, "Pos Pred Value"], na.rm = TRUE),
  Mean_Neg_Pred_Value = mean(cm_holdout$byClass[, "Neg Pred Value"], na.rm = TRUE),
  Mean_Balanced_Accuracy = mean(cm_holdout$byClass[, "Balanced Accuracy"], na.rm = TRUE),
  Mean_F1 = mean(cm_holdout$byClass[, "F1"], na.rm = TRUE)
)
names(holdout_metrics_vector) <- metrics
df_metrics[["Holdout"]] <- as.data.frame(t(holdout_metrics_vector))

common_train_control_args <- list(summaryFunction = multiClassSummary, classProbs = FALSE,
                                    savePredictions = "none", allowParallel = FALSE)

# Holdout Repetido (10x 75/25)
tc_repeated_holdout <- do.call(trainControl, c(list(method = "LGOCV", p = 0.75,
                                                       number = 10), common_train_control_args))
train_knn_lgocv <- train(Class ~ ., data = vehicle_data, method = "knn",
                           trControl = tc_repeated_holdout, preProcess = c("center", "scale"),
                           tuneLength = 3)
best_k_lgocv <- train_knn_lgocv$bestTune$k
lgocv_results <- train_knn_lgocv$results[train_knn_lgocv$results$k == best_k_lgocv,
                                           metrics, drop = FALSE]
df_metrics[["RepeatedHoldout"]] <- lgocv_results

# k-Fold (k=10)
tc_kfold <- do.call(trainControl, c(list(method = "cv", number = 10), common_train_control_args))
train_knn_kfold <- train(Class ~ ., data = vehicle_data, method = "knn",
                           trControl = tc_kfold, preProcess = c("center", "scale"),
                           tuneLength = 3)
best_k_kfold <- train_knn_kfold$bestTune$k
kfold_results <- train_knn_kfold$results[train_knn_kfold$results$k == best_k_kfold,
                                            metrics, drop = FALSE]
df_metrics[["KFold"]] <- kfold_results

# k-Fold Repetido (10-fold, 3 repetições)
tc_rkf <- do.call(trainControl, c(list(method = "repeatedcv", number = 10, repeats = 3),
                                    common_train_control_args))
train_knn_rkf <- train(Class ~ ., data = vehicle_data, method = "knn", trControl = tc_rkf,
                        preProcess = c("center", "scale"), tuneLength = 3)
best_k_rkf <- train_knn_rkf$bestTune$k
rep_kfold_res <- train_knn_rkf$results[train_knn_rkf$results$k == best_k_rkf, metrics, drop = FALSE]
df_metrics[["RepeatedKFold"]] <- rep_kfold_res

```

```

# LOOCV
tc_loocv <- do.call(trainControl, c(list(method = "LOOCV"), common_train_control_args))
train_knn_loocv <- train(Class ~ ., data = vehicle_data, method = "knn", trControl = tc_loocv,
                           preProcess = c("center", "scale"), tuneLength = 3)
best_k_loocv <- train_knn_loocv$bestTune$k
loocv_results <- train_knn_loocv$results[train_knn_loocv$results$k == best_k_loocv,
                                           metrics, drop = FALSE]
df_metrics[["LOOCV"]] <- loocv_results

final_results_list_dfs <- lapply(names(df_metrics), function(method_name) {
  df_row <- df_metrics[[method_name]]
  df_row_with_method <- data.frame(Method = method_name)
  for(col_name in names(df_row)) {
    df_row_with_method[[col_name]] <- df_row[[col_name]]}
  df_row_with_method <- df_row_with_method[, c("Method", metrics)]
  return(df_row_with_method)})}

results_df <- do.call(rbind, final_results_list_dfs)
rownames(results_df) <- NULL
numeric_cols_final <- setdiff(names(results_df), "Method")
results_df[numeric_cols_final] <- lapply(results_df[numeric_cols_final],
                                         function(x) if(is.numeric(x)) round(x, 4) else x)
novos_nomes_colunas <- c("Método", "Accuracy", "Kappa", "Sens.M", "Espec.M",
                          "PPV.M", "NPV.M", "AcurBal.M", "F1.M")
colnames(results_df) <- novos_nomes_colunas
library(knitr)
kable(results_df, caption = "Comparação das Métricas de Desempenho do KNN por Método de Validação",
      booktabs = TRUE, digits = 4, align = 'c')

confusion_plot <- ggplot(data = cm_table, aes(x = Reference, y = Prediction, fill = Frequency)) +
  geom_tile(colour = "white") + geom_text(aes(label = sprintf("%d", Frequency)), vjust = 1, size = 5) +
  scale_fill_gradient(low = "lightblue", high = "steelblue") +
  labs(x = "Classe Real (Reference)", y = "Classe Predita (Prediction)",
       title = plot_title, fill = "Frequência") +
  coord_equal() + theme_minimal(base_size = 12) +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        axis.title = element_text(size = 14), axis.text = element_text(size = 12),
        legend.title = element_text(size = 12), legend.text = element_text(size = 10),
        axis.text.y = element_text(angle = 0, hjust = 1),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank()) +
  scale_x_discrete(limits = sort(levels(cm_table$Reference))) +
  scale_y_discrete(limits = rev(sort(levels(cm_table$Prediction))))+
print(confusion_plot)

TP <- cm$table[pos_class, pos_class]
TN <- cm$table[levels(test_labels)[levels(test_labels) != pos_class],
               levels(test_labels)[levels(test_labels) != pos_class]]
FP <- cm$table[pos_class, levels(test_labels)[levels(test_labels) != pos_class]]
FN <- cm$table[levels(test_labels)[levels(test_labels) != pos_class], pos_class]
cat("TP:", TP, " TN:", TN, " FP:", FP, " FN:", FN, "\n")
cohen_kappa <- cm$overall["Kappa"]
cat("Coeficiente Kappa de Cohen:", round(cohen_kappa, 4), "\n")

TP_adj <- TP + 0.5

```



```

TN_adj <- TN + 0.5
FP_adj <- FP + 0.5
FN_adj <- FN + 0.5
dor <- (TP_adj * TN_adj) / (FP_adj * FN_adj)

cat("Razão de Chances de Diagnóstico (DOR):", round(dor, 4), "\n")

cat("Acurácia:", round(cm$overall["Accuracy"], 4))
cat("Sensibilidade (Recall para classe 'X1'):", round(cm$byClass["Sensitivity"], 4))
cat("Especificidade:", round(cm$byClass["Specificity"], 4))
cat("Precisão (VPP para classe 'X1'):", round(cm$byClass["Pos Pred Value"], 4))
cat("F1-Score para classe 'X1':", round(cm$byClass["F1"], 4))

```