Gabarito Lista 03

Ciência de Dados - 2025

Alex Monito Nhancololo

Critérios usados na correção

Independentemente do software empregado, aplicam-se os seguintes critérios de avaliação:

- Justificar sempre as escolhas metodológicas, não bastando apresentar apenas o código e/ou sua saída.
- Manter o documento limpo, sem `#` desnecessários, warnings, erros ou descrições triviais.
- Incluir tabelas e gráficos bem formatados, com títulos, legendas e rótulos claros para interpretação autônoma.
- Organizar todo o código em apêndice/anexo, mantendo coerência, consistência e redação técnica objetiva.

Nota: Comentários sobre funções básicas (como set.seed() ou outras funções triviais) não devem ser incluídos.

Sugestões

- Use `pacman` para instalar e importar pacotes numa só vez, evitando `install.packages()`, `library()` ou `require` para cada pacote.
- Use `knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE, comment = "")`, para mostrar o código e a saída, ocultar mensagens automáticas, ocultar mensagens de aviso (warnings) na saída e remover o símbolo padrão (como ##) antes da saída do código, respectivamente.
- Use pacotes como `textreg` ou `stargazer`, `finalfit`, `flextable`, `equatiomatic`, entre outros, para extração das estimativas do modelo e suas métricas em diferentes formatos (LaTeX, HTML, Word, colocar tabela única para vários modelos etc.), geração de tabelas formatadas e extração automática das equações/expressões matemáticas do modelo ajustado com ou sem as estimativas, respectivamente.

Pacotes usados:

```
if(!require(pacman)) install.packages(pacman)
p_load(tree, rpart,rpart.plot,ggpubr,tidyverse,knitr,kableExtra,readxl,
       readr,ipred,randomForest,e1071,caret,dplyr,class,broom)
```

EXERCÍCIO 01

Utilizou-se a função `rpart` do pacote `rpart` para ajustar uma árvore de classificação da espécie de flor, definindo *minsplit* (número mínimo de observações em um nó para que ele seja dividido) como 5. Todo o conjunto de dados foi empregado no ajuste, e a estrutura resultante, desenhada com a função `prp` do pacote `rpart.plot` está exibida na Figura 1 (A).

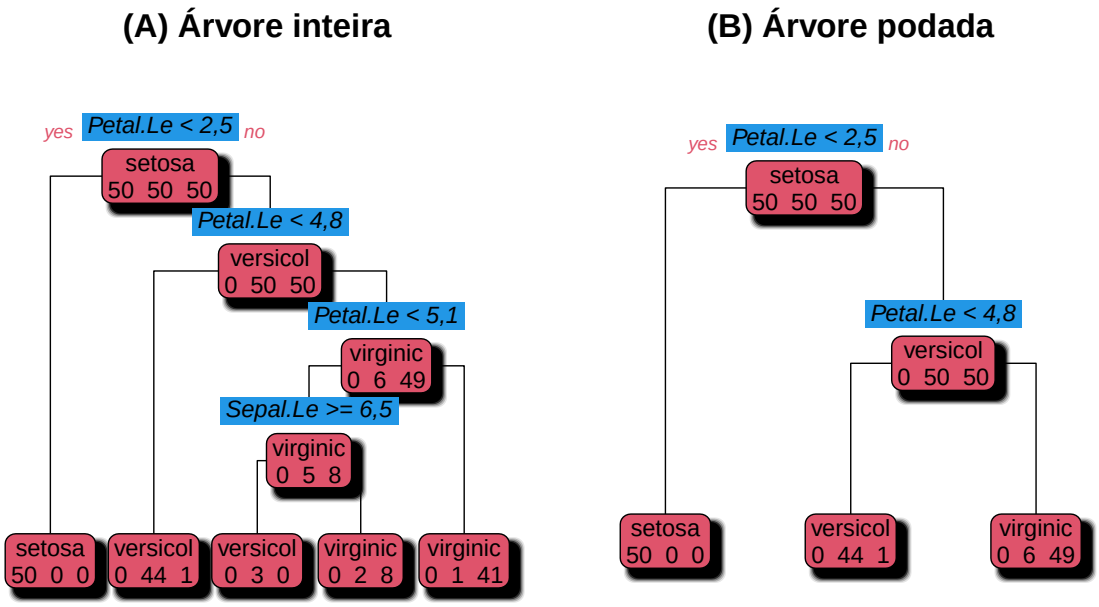

Figura 1: Árvores ajustadas.

Definiu-se $X_1 =$ Petal.Length e $X_2 =$ Sepal.Length. A partir da árvore ajustada, distinguiu-se cinco regiões no plano (X_1, X_2):

$$
\begin{aligned}
R_1 &= \{X_1 < 2{,}5\}, \\
R_2 &= \{X_1 \geq 2{,}5\} \cap \{X_1 < 4{,}8\}, \\
R_3 &= \{X_1 \geq 4{,}8\} \cap \{X_1 < 5{,}1\} \cap \{X_2 \geq 6{,}5\}, \\
R_4 &= \{X_1 \geq 4{,}8\} \cap \{X_1 < 5{,}1\} \cap \{X_2 < 6{,}5\}, \\
R_5 &= \{X_1 \geq 5{,}1\}.
\end{aligned}
$$

Com base nessas regiões, atribuiu-se as classes às espécies de flor da seguinte forma:

- Setosa: R_1;
- Versicolor: $R_2 \cup R_3$;
- Virginica: $R_4 \cup R_5$.

A representação gráfica dessas zonas de classificação é apresentada na Figura 2 (A).

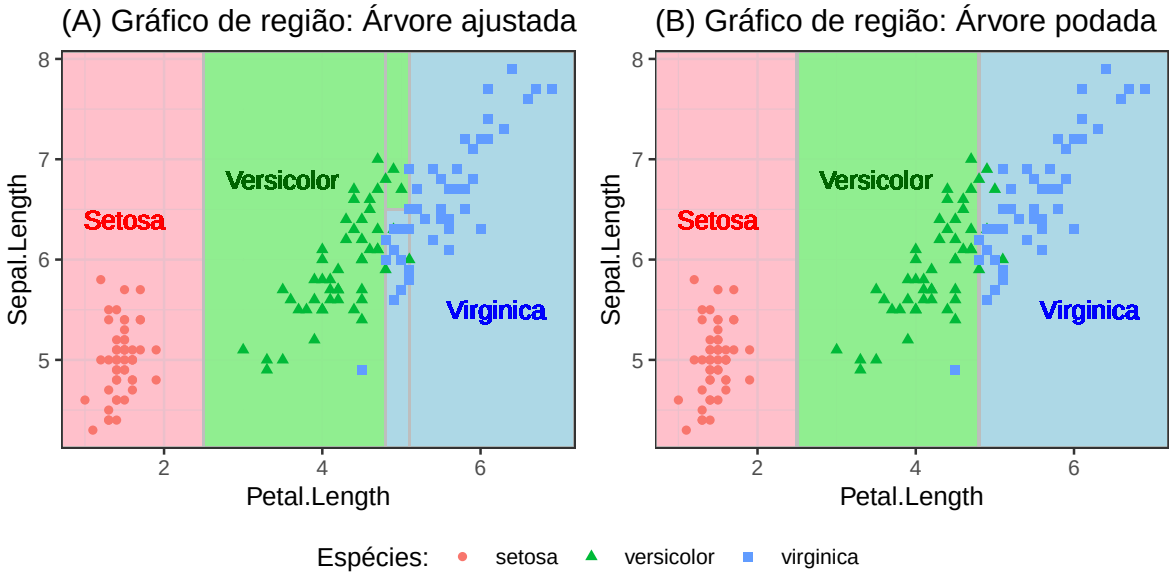

Figura 2: Plano separador.

A Tabela 1 apresenta a matriz de confusão resultante da aplicação da árvore ajustada sobre o mesmo conjunto de dados. Observa-se uma taxa de erro de 2,667 %, o que evidencia a eficácia do modelo. Note-se ainda que todas as observações de Setosa foram classificadas corretamente, havendo apenas uma pequena sobreposição entre Versicolor e Virginica.

Tabela 1: Matriz confusão do método de classificação usando a árvore ajustada.

	Referência		
	setosa	**versicolor**	**virginica**
Predição			
setosa	50	0	0
versicolor	0	47	1
virginica	0	3	49

Note:
Acurácia: 0,973; Taxa de erro: 0,027

Após observar-se que a árvore inicial apresentava ramos em excesso possivelmente reflexo do pequeno número de variáveis e de observações nas folhas mais profundas optou-se por realizar uma poda visando reduzir sua complexidade e mitigar riscos de sobreajuste. Para tanto, aplicou-se a função `prune` do pacote `rpart` com $cp = 0.1$, obtendo a árvore ilustrada na

Figura 1 (B). Esta versão é notavelmente mais enxuta, gerando apenas três regiões no plano (X_1, X_2), onde $X_1 = \text{Petal.Length}$ e $X_2 = \text{Sepal.Length}$:

$$\begin{aligned} R_1 &: \{\, X_1 < 2{,}5\}, \\ R_2 &: \{\, 2{,}5 \leq X_1 < 4{,}8\}, \\ R_3 &: \{\, X_1 \geq 4{,}8\}. \end{aligned}$$

Diferentemente da árvore original, cada região está agora associada exclusivamente a uma única classe de flor:

$$\begin{aligned} \text{Setosa} &: \; R_1, \\ \text{Versicolor} &: \; R_2, \\ \text{Virginica} &: \; R_3. \end{aligned}$$

A representação gráfica dessas zonas de decisão encontra-se na Figura 2 (B).

A Tabela 2 apresenta a matriz de confusão resultante da aplicação desta árvore podada ao conjunto de dados original. Observa-se uma taxa de erro de

$$(1 - \text{Ex01}_{\text{C}}\text{onfMat02\$overall[1]}) \times 100\% \;=\; (1 - \text{acurácia}) \times 100\%.$$

confirmando um bom desempenho do modelo. Assim como na versão sem poda, todas as observações de Setosa foram corretamente classificadas, restando apenas um pequeno grau de confusão entre Versicolor e Virginica.

Tabela 2: Matriz confusão do método de classificação usando a árvore podada.

	Referência		
	setosa	**versicolor**	**virginica**
Predição			
setosa	50	0	0
versicolor	0	44	1
virginica	0	6	49

Note:

Acurácia: 0,953; Taxa de erro: 0,047

Por fim, vale ressaltar que, embora a árvore podada apresente uma taxa de erro ligeiramente superior à da árvore original, ambas mantêm níveis de erro muito baixos, o que atesta o bom desempenho de ambos os modelos. A pequena redução na acurácia da versão podada é amplamente compensada por sua estrutura mais simples e pela interpretabilidade aprimorada, características que podem favorecer sua capacidade de generalização.

EXERCÍCIO 02

Dividiu-se o conjunto de dados em Treino (75%) e Teste (25%) para comparar o desempenho de dois métodos de classificação *bagging* e Floresta Aleatória na predição da variável *Diabetes* a partir de *HDL*, *LDL*, *Triglic*, *Glicose* e *Peso*.

Para o *bagging*, utilizou-se a função `bagging` do pacote `ipred` com `nbagg = 25` (número de réplicas bootstrap) e árvores construídas com `minsplit = 5` e `cp = 0`. A matriz de confusão resultante, mostrada na Tabela 3, indica uma taxa de erro de 17,2%.

Devido ao desbalanceamento entre as classes (14 casos "Presente" vs. 50 "Ausente"), observa-se que, embora a acurácia global seja razoável, a sensibilidade para identificar corretamente a presença de diabetes é apenas 50%. Isso evidencia a necessidade de avaliar métricas específicas por classe, além da acurácia agregada, para compreender melhor o desempenho do modelo em situações de desbalanceamento.

Tabela 3: Matriz confusão do Bagging.

	Referência	
	Ausente	**Presente**
Predição		
Ausente	46	7
Presente	4	7

Note:
Acurácia: 0,828; Taxa de erro: 0,172

Para a Floresta Aleatória, utilizamos a função `randomForest` do pacote `randomForest`, definindo `mtry = 2` (número de variáveis sorteadas em cada nó), `ntree = 500` (quantidade de árvores) e `nodesize = 5` (mínimo de observações em cada folha). A Tabela 4 apresenta a matriz de confusão obtida sobre o conjunto de teste, com uma taxa de erro de 14,1%.

Em razão do desbalanceamento entre os grupos (14 casos "Presente" vs. 50 "Ausente"), novamente verifica-se que, embora a acurácia global seja satisfatória, a sensibilidade para detectar corretamente a presença de diabetes permanece em apenas 50%, evidenciando a necessidade de métricas específicas para avaliar o desempenho em classes minoritárias.

Tabela 4: Matriz confusão do método de classificação com Floresta Aleatória.

	Referência	
	Ausente	**Presente**
Predição		
Ausente	48	7
Presente	2	7

Note:
Acurácia: 0,859; Taxa de erro: 0,141

Tabela 5: Principais resultados do ajuste do Bagging e Floresta Aleatória

Classificadores	**Bagging**	**Floresta Aleatória**
Taxa de Erro	0,172	0,141
Acurácia	0,828	0,859
Kappa	0,455	0,528
Lim. Inf. Acurácia	0,713	0,750
Lim. Sup. Acurácia	0,911	0,934
Valor p da acurácia	0,229	0,082
Valor p McNemar	0,546	0,182
Sensibilidade	0,500	0,500
Especificidade	0,920	0,960
Precisão	0,636	0,778
Prevalência	0,219	0,219

A Tabela 5 sintetiza o desempenho dos dois algoritmos. Observa-se que *bagging* e Floresta Aleatória apresentam resultados muito próximos, com leve vantagem para esta última. Assim, sob o critério de acurácia, a Floresta Aleatória se mostra o método preferível.

Observação: No *bagging* também seria possível empregar a função `randomForest`; entretanto, seria necessário ajustar `mtry = 5` para que todas as variáveis participem de cada divisão, em vez de selecionar apenas um subconjunto aleatório.

EXERCÍCIO 03

Dados os pares de treinamento $\{(x_i, y_i)\}_{i=1}^n$, onde $x_i \in \mathbb{R}^p$ são vetores de características e $y_i \in \{-1, +1\}$ são seus rótulos, o classificador de margem máxima busca um hiperplano $\mathbf{w} \cdot x + b = 0$ que separe perfeitamente as duas classes: pontos com $\mathbf{w} \cdot x_i + b > 0$ ficam na classe $+1$ e com $\mathbf{w} \cdot x_i + b < 0$ na classe -1. A margem é a distância mínima entre o hiperplano e qualquer ponto de treinamento, dada por $\min_i \big|\mathbf{w} \cdot x_i + b\big| / \|\mathbf{w}\|$; maximizar essa margem torna o classificador mais robusto. Os vetores de suporte são exatamente os pontos que atingem essa distância mínima, ou seja, satisfazem $y_i(\mathbf{w} \cdot x_i + b) = 1$ quando normalizamos $\|\mathbf{w}\|$. Uma formulação equivalente apresenta-se como

$$\max_{\beta_0, \beta_1, \ldots, \beta_p, M} M, \quad \text{sujeito a,} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \quad y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M, \;\; i = 1, \ldots, n,$$

onde a normalização $\sum_{j=1}^p \beta_j^2 = 1$ faz com que M seja diretamente metade da distância ao hiperplano. Em vez de fixar $\|\beta\| = 1$ e otimizar M, adota-se também a formulação "primal":

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{sujeito a} \quad y_i(\mathbf{w} \cdot x_i + b) \geq 1, \;\; i = 1, \ldots, n,$$

cuja solução $(\mathbf{w}^\star, b^\star)$ determina um hiperplano que separa as classes e cujo valor ótimo satisfaz $M = 1/\|\mathbf{w}^\star\|$. A regra de classificação para uma nova observação $\mathbf{x}^*$ é: $\hat{y}(\mathbf{x}^*) = \text{sgn}(\mathbf{w} \cdot \mathbf{x}^* + b)$. A margem (distância mínima ao hiperplano) é dada por: $M = \frac{1}{\|\mathbf{w}\|}$.

Tabela 6: Dados fornecidos

Obs	X1	X2	Y
1	3	4	Azul
2	2	2	Azul
3	4	4	Azul
4	1	4	Azul
5	2	1	Vermelho
6	4	3	Vermelho
7	4	1	Vermelho

Item A)

Para ajustar o SVM linear (CVM) com custo muito alto (aproximando-se do hard-margin), utilizou-se o pacote `e1071`:

Função objetivo e restrições:

$$\min_{w_0, w_1, w_2} \quad \frac{1}{2}\left(w_1^2 + w_2^2\right)$$

$$\text{sujeito a} \quad \begin{cases} w_0 + w_1 x_{i1} + w_2 x_{i2} \geq 1 & \text{para } y_i = \text{Azul}, \\ w_0 + w_1 x_{i1} + w_2 x_{i2} \leq -1 & \text{para } y_i = \text{Vermelho}. \end{cases}$$

A equação do hiperplano é $-2\,X_1 + 2\,X_2 + 1 = 0$. Assim, para uma nova observação $(X_1^*,\ X_2^*)$:

$$\text{Classificação} \ = \ \begin{cases} \text{Azul:} & \text{se, } -2\,X_1^* + 2\,X_2^* + 1 > 0, \\ \text{Vermelho:} & \text{se, } -2\,X_1^* + 2\,X_2^* + 1 < 0. \end{cases}$$

Item B)

Na Figura 3, apresenta-se o hiperplano separador (em preto) e as fronteiras da margem superior e inferior (linhas tracejadas), e destacam-se os vetores de suporte.

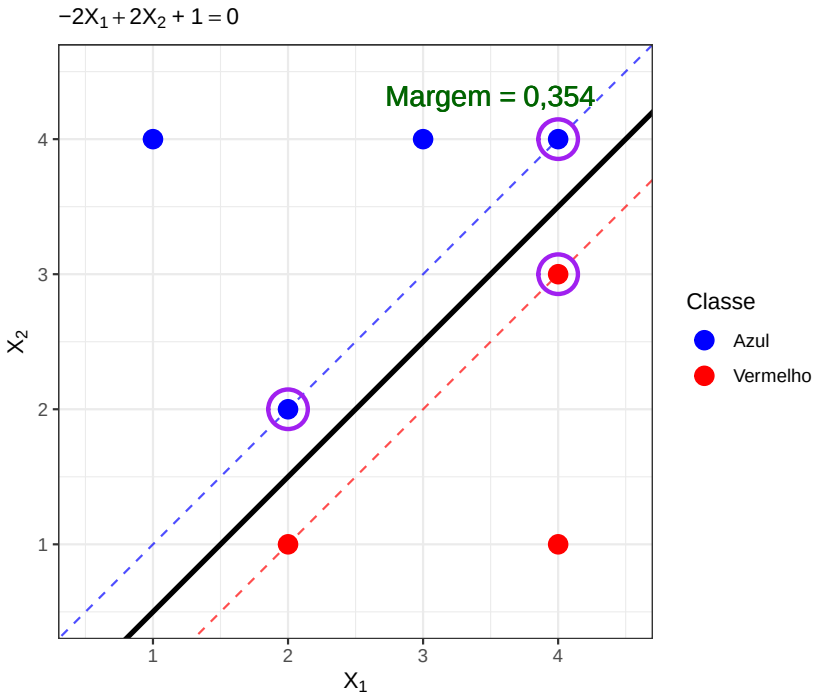

Figura 3: Classificador de Margem Máxima

O hiperplano separador $-2X_1 + 2X_2 + 1 = 0$ está em preto; as linhas tracejadas em azul e vermelho representam as fronteiras da margem superior e inferior, respectivamente; e os círculos roxos destacam os vetores de suporte.

Item C)

A margem máxima M é calculada por:

$$M = \frac{1}{\|\mathbf{w}\|} = \frac{1}{\sqrt{w_1^2 + w_2^2}} = \frac{1}{\sqrt{(-2)^2 + 2^2}} = \frac{1}{\sqrt{8}} \approx 0{,}354,$$

onde $\mathbf{w} = (-2, 2)$.

> **Regra prática em SVM**: quaisquer alterações feitas em *vetores de suporte* **afetam diretamente o hiperplano** (e, portanto, o classificador). Por outro lado, modificações em observações que **não** são vetores de suporte **não alteram w** nem b, de modo que o hiperplano permanece inalterado.

A Tabela 7 a seguir mostra quais pontos são vetores de suporte (destacados no ajuste original):

Tabela 7: Vetores de Suporte do Modelo Original.

	Obs	X1	X2	Y
2	2	2	2	Azul
3	3	4	4	Azul
6	6	4	3	Vermelho

Para ilustrar o efeito de alterar um vetor de suporte, modificamos a sétima observação de (4, 1) para (4.5, 1.5) e ajustamos novamente o SVM:

Tabela 8: Comparação entre o modelo original e o modelo com 7^a observação modificada.

Modelo	w1	w2	b	Margem
Original	-2	2	1	0,354
Modificado	-2	2	1	0,354

Observa-se que na Tabela 8, que ao alterar um vetor de suporte, os coeficientes (w_1, w_2), o termo b e, consequentemente, a margem M não mudam, pois a última observação não faz parte do vetor suporte.

Referências por ordem de abrangência

1. Morettin, Pedro Alberto, and Júlio da Motta Singer. *Estatística e ciência de dados.* (2025). (**Recomendo**)
2. Boehmke, Brad, and Brandon M. Greenwell. *Hands-on machine learning with R.* Chapman and Hall/CRC, 2019. Disponível online **GRATUITAMENTE**
3. Manual, A. Beginner'S. *An introduction to statistical learning with applications in R.* (2013). (**Existe versão Python, ops. é pago**)

Apêndice

Exercício 1

Ajuste:

```
ajusteArvore = rpart::rpart(Species ~ Sepal.Length + Petal.Length,
                            data = iris, minsplit = 5)
poda = rpart::prune(ajusteArvore, cp = 0.1)

Ex01_pred01 = predict(ajusteArvore, type = "class")
Ex01_pred02 = predict(poda, type = "class")

Ex01_ConfMat01 = confusionMatrix(Ex01_pred01, iris$Species)
Ex01_ConfMat02 = confusionMatrix(Ex01_pred02, iris$Species)

# arv = tree(Species ~ Sepal.Length + Petal.Length, data = iris)
# arv2 = cv.tree(arv)
# plot(arv)
# text(arv)
```

Gráfico das árvores:

```
par(mfrow = c(1,2))
rpart.plot::prp(ajusteArvore, type = 1, extra = 1, fallen.leaves = TRUE, Margin=0,
                box.col = 2, shadow.col = 1, under.percent = 3,
                nn.col = 2, split.box.col = 4, split.font = 3, split.col = 1,
                space = 0.6, main = "(A) Árvore inteira", cex = 0.8)
rpart.plot::prp(poda, type = 1, extra = 1, fallen.leaves = TRUE, Margin=0,
                box.col = 2, shadow.col = 1, under.percent = 3,
                nn.col = 2, split.box.col = 4, split.font = 3, split.col = 1,
                space = 0.6, main= "(B) Árvore podada", cex = 0.8)
```

Gráfico dos planos:

```
plano01 = ggplot(iris, aes(y = Sepal.Length, x= Petal.Length,
                           color = Species, shape = Species, fill = Species)) +
    theme_bw() +
    geom_rect(aes(ymin = -Inf, ymax = Inf, xmin = -Inf, xmax = 2.5),
              fill = "pink", alpha = 0.02, color = "gray") +
    geom_rect(aes(ymin = -Inf, ymax = Inf, xmin = 2.5, xmax = 4.8),
              fill = "lightgreen", alpha = 0.02, color = "gray") +
    geom_rect(aes(ymin =  6.5, ymax = Inf, xmin = 4.8, xmax = 5.1),
              fill = "lightgreen", alpha = 0.02, color = "gray") +
    geom_rect(aes(ymin = -Inf, ymax = 6.5, xmin = 4.8, xmax = 5.1),
              fill = "lightblue", alpha = 0.02, color = "gray") +
    geom_rect(aes(ymin = -Inf, ymax = Inf, xmin = 5.1, xmax = Inf),
              fill = "lightblue", alpha = 0.02, color = "gray") +
        # geom_text(x=c(4,4,4), y=c(6.2,4,3), label=c(levels(iris$Species))) +
    geom_text(y = 5.5, x = 6.2, label = "Virginica", color = "blue") +
    geom_text(y = 6.8, x = 3.5, label = "Versicolor", color = "darkgreen") +
    geom_text(y = 6.4, x = 1.5, label = "Setosa", color = "red") +
    labs(color = "Espécies:", shape = "Espécies:", fill = "Espécies:",
         title = "(A) Gráfico de região: Árvore ajustada") +
    geom_point()

plano02 = ggplot(iris, aes(y = Sepal.Length, x = Petal.Length,
                           color = Species, shape = Species, fill = Species)) +
    theme_bw() +
    geom_rect(aes(ymin = -Inf, ymax = Inf, xmin = -Inf, xmax = 2.5),
              fill = "pink", alpha = 0.02, color = "gray") +
    geom_rect(aes(ymin = -Inf, ymax = Inf, xmin = 2.5, xmax = 4.8),
              fill = "lightgreen", alpha = 0.02, color = "gray") +
    geom_rect(aes(ymin = -Inf, ymax = Inf, xmin = 4.8, xmax = Inf),
```

```
              fill = "lightblue", alpha = 0.02, color = "gray") +
    # geom_text(x=c(4,4,4), y=c(6.2,4,3), label=c(levels(iris$Species))) +
    geom_text(y = 5.5, x = 6.2, label = "Virginica", color = "blue") +
    geom_text(y = 6.8, x = 3.5, label = "Versicolor", color = "darkgreen") +
    geom_text(y = 6.4, x = 1.5, label = "Setosa", color = "red") +
    labs(color = "Espécies:", shape = "Espécies:", fill = "Espécies:",
         title = "(B) Gráfico de região: Árvore podada") +
    geom_point()

ggarrange(plano01, plano02, ncol = 2, nrow = 1, common.legend = TRUE, legend ="bottom")
```

Exercício 2

Código:

```
## Fazendo download e carregando o conjunto de dados
file = tempfile(fileext = ".xls")
download.file("http://www.ime.usp.br/~jmsinger/MorettinSinger/rehabcardio.xls",
              file, mode = "wb")
data02 = read_xls(file, 1)
rm(file)

## Selecionando apenas as colunas necessárias e as linhas completas
dat02 = data02 %>%
    dplyr::select(Diabete, HDL, LDL, Triglic, Glicose, Peso) %>%
    na.omit()
dat02$Diabete = as.factor(ifelse(dat02$Diabete == 0, "Ausente", "Presente"))
dat02$Triglic = as.numeric(dat02$Triglic)

## Separando em conjuntos de teste e treino
set.seed(2023)
idxTrain = createDataPartition(y = dat02$Diabete, p = .75, list = FALSE)
trainDat02 = dat02[idxTrain,]
testDat02 = dat02[-idxTrain,]

## Ajustando bagging
bag = bagging(Diabete ~ ., data = trainDat02, nbagg = 25, coob = TRUE,
              control = rpart.control(minsplit = 5, cp = 0))

## Ajustando random florest
forest = randomForest(Diabete ~ ., data = trainDat02, ntree=500,
```

```
                      mtry=2, nodesize = 5)

## Predições
Ex02_predBag = predict(bag, testDat02)
Ex02_predforest = predict(forest, testDat02)

Ex02_ConfMat01 = confusionMatrix(Ex02_predBag, testDat02$Diabete,
                                 positive = "Presente")
Ex02_ConfMat02 = confusionMatrix(Ex02_predforest, testDat02$Diabete,
                                 positive = "Presente")
```

Exercício 3

Item A

```
modelo <- e1071::svm(Y ~ X1 + X2,
                     data   = dados,
                     kernel = "linear",
                     cost   = 1e10,
                     scale  = FALSE)

# coeficientes w e b
w <- t(modelo$coefs) %*% modelo$SV
b <- -modelo$rho
```

Item B

```
# Normas e margem
norma_w   <- sqrt(sum(w^2))
margem    <- 1 / norma_w
w1_fmt    <- round(w[1], 2)
w2_fmt    <- round(w[2], 2)
b_abs     <- round(abs(b), 2)
b_sign    <- if (b >= 0) "+" else "-"
margem_fmt <- round(margem, 3)

expr_hiperplano <- bquote(
  .(w1_fmt)*X[1] + .(w2_fmt)*X[2] ~ .(b_sign) ~ .(b_abs) == 0
```

```
)
expr_margem <- bquote(
  "Margem =" ~ .(margem_fmt)
)


vetores_suporte <- dados[modelo$index, ]

# Gráfico
ggplot(dados, aes(x = X1, y = X2, color = Y)) +
  geom_point(size = 4) +
  geom_abline(
    slope     = - w[1] / w[2],
    intercept = - b / w[2],
    size      = 1.2,
    color     = "black"
  ) +
  # Margem superior (y = (1 - b)/w[2] - (w[1]/w[2]) x)
  geom_abline(
    slope     = - w[1] / w[2],
    intercept = (1 - b) / w[2],
    linetype  = "dashed",
    color     = "blue",
    alpha     = 0.7
  ) +
  # Margem inferior (y = (-1 - b)/w[2] - (w[1]/w[2]) x)
  geom_abline(
    slope     = - w[1] / w[2],
    intercept = (-1 - b) / w[2],
    linetype  = "dashed",
    color     = "red",
    alpha     = 0.7
  ) +
  # Vetores de suporte
  geom_point(
    data  = vetores_suporte,
    aes(x = X1, y = X2),
    shape  = 1,
    size   = 8,
    stroke = 1.5,
    color  = "purple"
  ) +
```

```
  scale_color_manual(values = c("Azul" = "blue", "Vermelho" = "red")) +
  labs(
    subtitle = expr_hiperplano,
    x        = expression(X[1]),
    y        = expression(X[2]),
    color    = "Classe"
  ) +
  theme_bw() +
  coord_fixed(ratio = 1, xlim = c(0.5, 4.5), ylim = c(0.5, 4.5)) +
  annotate(
    "text",
    x     = 3.5,
    y     = 4.3,
    label = expr_margem,
    color = "darkgreen",
    size  = 5
  )
```

Item C

```
vetores_suporte <- dados[modelo$index, ]
knitr::kable(vetores_suporte)
```

```
dados_mod <- dados
dados_mod[7, c("X1", "X2")] <- c(4.5, 1.5)

modelo_mod <- e1071::svm(
  Y ~ X1 + X2,
  data   = dados_mod,
  kernel = "linear",
  cost   = 1e10,
  scale  = FALSE
)

w_mod    <- t(modelo_mod$coefs) %*% modelo_mod$SV
b_mod    <- -modelo_mod$rho
norma_w_mod <- sqrt(sum(w_mod^2))
margem_mod  <- 1 / norma_w_mod

resultado <- data.frame(
```

```r
  Modelo = c("Original", "Modificado"),
  w1     = c(round(w[1], 4),    round(w_mod[1], 4)),
  w2     = c(round(w[2], 4),    round(w_mod[2], 4)),
  b      = c(round(b,    4),    round(b_mod,    4)),
  Margem = c(round(margem, 4),  round(margem_mod, 4))
)

knitr::kable(resultado)
```