Instituto Federal do Espírito Santo – Ifes

campus Cariacica

Bacharelado em Engenharia de Produção – 4º período

Modelos Econômicos e Quantitativos

Prof. Guilherme Guilhermino Neto

Alunos: Rayane Cardoso Silva e André Teles Cunha

Prática de laboratório 3 – Aprendizado de Máquina

Sua dupla ou trio deve trabalhar com a base de dados que foi designada. Esta base de dados faz parte do pacote fpp2 do R. Guarde os dados em um objeto do RStudio e siga o roteiro abaixo para as análises. Vocês deverão responder ao que se pede **neste arquivo**. Ao final, exporte em formato *.pdf*, com o nome no seguinte padrão: *Prática de laboratório 3 - Nome dos alunos.pdf* e envie pelo AVA.

Instruções para Análise

Dataset: land_mine

Detecção de minas enterradas no chão é um procedimento muito importante em termos de segurança da vida e da propriedade. Muitos métodos diferentes foram usados para isso; entretanto, ainda não foi possível atingir 100% de sucesso. O processo de detecção de minas consiste no projeto de um sensor, na análise de dados e na elaboração de algoritmos de decisão.

O método da anomalia magnética funciona de acordo com o princípio da medição de anomalias resultantes do objeto no campo magnético que perturba sua estrutura. O campo magnético e os dados obtidos são usados para determinar condições como movimento e posição. A determinação de parâmetros como posição, profundidade ou direção usando anomalia magnética tem sido feita desde 1970.

O dataset disponibilizado contém resultados de análises e minas já classificadas. Suas colunas são:

V: voltagem - Voltagem detectada pelo sensor FLC devido a distorção magnética (0V até 10,6V)

H: altura do sensor em relação ao chão (0cm até 20cm)

S: tipo de solo. Pode ser:

1 Seco e arenoso

2 Seco e húmus

3 Seco e calcário

4 Úmido e arenoso

5 Úmido e húmus

6 Úmido e calcário

M: detecção de mina

0 Não há mina

1 Há mina

Os dados das colunas V, H e S já foram padronizados.

Data de entrega 01/12 - Entrega parcial 2 - Prova 3 (1 ao 5)

Aula 03/12 - Exemplo de Random Forest / Prova 3 / Orientação ao trabalho

Data de entrega 08/12 - Entrega final da Prova 3

Aula 10/12 - Apresentação dos trabalhos

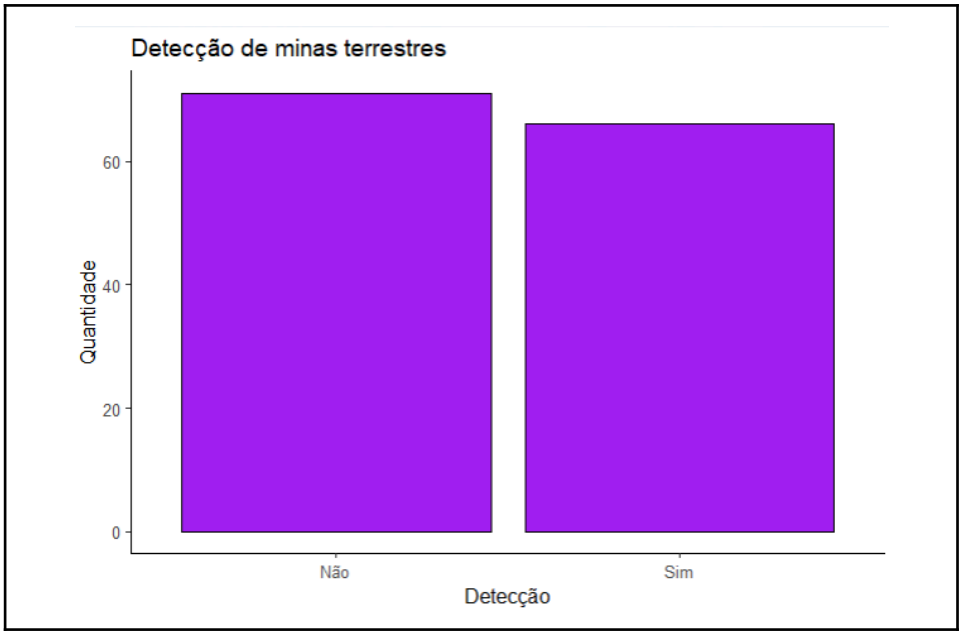
1 – Comece analisando os tipos de minas nesse conjunto de dados.

a) Faça uma tabela de frequências e um gráfico de barras para os dados da coluna **M**.

Tabela de frequência:

Detecção de Minas Terrestres:	
0 (não detectada)	**1 (detectada)**
71	*66*

Gráfico de barras:



b) Com no item **a)**, você diria que o conjunto de dados é **desbalanceado** (ou seja, alguma classe está sub-representada)? Se sim, qual classe está sub-representada? Como isso pode afetar o desenvolvimento de um modelo de classificação? Quais cuidados você deveria tomar na hora de separar as amostras de treino e teste?

Bom, o conjunto de dados não aparenta ser desbalanceado, já que a quantidade de posições em que há minas terrestres não é discrepante em relação à quantidade das mesmas em que não há minas terrestres.

Apesar de o conjunto de dados ser balanceado, é necessário se atentar se o mesmo não está classificado, pois se tiver, todas as posições em que há minas estarão juntas, assim como o contrário, o que na hora de separar as amostras de treino e teste, pode influenciar negativamente o teste do modelo. Portanto, a dupla enxerga como solução, separar as amostras de "treino" e "teste" por meio de uma seleção aleatória dos valores.

2 – Fixe uma semente aleatória. Depois:

a) Separe os dados em uma amostra de treino e outra de teste, aleatórias, seguindo a proporção 80-20. Abaixo, diga quantas observações ficaram na amostra de treino e quantas ficaram na amostra de teste.

*Na amostra treino foram selecionadas 109 observações (137*0.8=109.6) e na amostra teste restaram 28 observações (109-137=28).*

b) Faça uma tabela de frequências para a coluna **M** para a amostra de treino e outra para a amostra de teste. Você diria que a representação das classes está proporcional (ou próxima disso) ao que ocorria nos dados originais?

Tabela de frequência das variáveis de cada amostra:

Amostra	Categoria 0	Categoria 1	Total
Dados Treino	54	55	109
Dados Teste	17	11	28
Total	71	66	137

Resposta:

Portanto, a partir dos dados expostos na tabela, é possível concluir que tanto as amostras quanto o conjunto de dados estão balanceados

c) Se a distribuição ficou desbalanceada, proponha e implemente uma solução para que isso não ocorra.

*Apesar de as amostras de treino e teste estarem balanceadas, a possível solução seria separar a coluna "M" em dois conjuntos de dados, classificados por 'Há mina' e 'Não há mina', ou seja, conjuntos nomeados - M_0 e M_1, após isso, iria atribuir, de acordo com a proporção 80-20, para o conjunto de dados "treino" e "teste", exemplificando novamente - "dados_treino <- M_0*0.8 + M_1*0.8", e o mesmo para a amostra teste.*

Neste caso, não seria necessário aplicar a função 'sample()' pois os valores já foram selecionados aleatoriamente pela semente implementada.

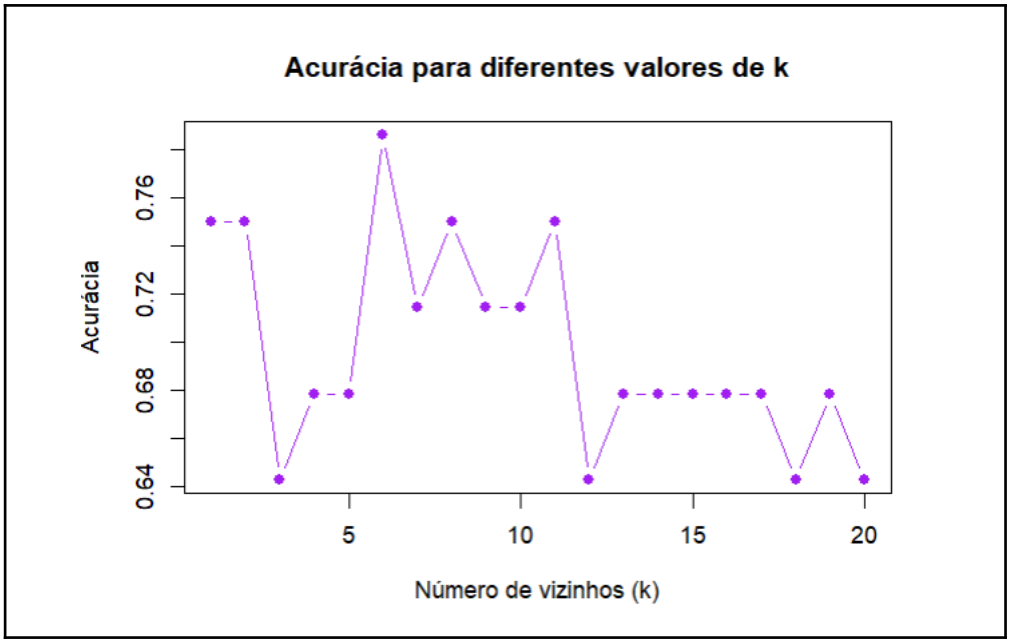
d) Quando fazemos e por que é uma boa prática fixar uma semente aleatória antes de executar um código?

Ao implementar uma semente aleatória, ocorre a padronização na aleatoriedade dos valores obtidos. Com isso, no contexto da nossa disciplina, é possível que façamos programas iguais e possamos corrigir nossos resultados, tanto com o professor quanto com os colegas da turma, de maneira mais eficiente, uma vez que, os resultados devem ser os mesmos ao implementar a mesma semente aleatória.

3 – Para o **kNN:**

a) Utilize a amostra de treino para escolher o número de vizinhos para o kNN. Abaixo, mostre o gráfico com o número de vizinhos *versus* a acurácia.

Gráfico de linha - número de vizinhos versus acurácia:



b) Com base em **a)**, qual número de vizinhos mais próximos você escolheria? Justifique.

Melhor valor de k: 6
Pois este foi o número de vizinhos em que o modelo melhor conseguiu classificar a maior parte dos dados, como observamos pelo valor da acurácia (sendo a maior que foi encontrada).

c) Utilizando o número de vizinhos escolhido em **b)**, utilize o kNN para fazer predições para a amostra de teste. Gere uma matriz de confusão e mostre os resultados abaixo.

Confusion Matrix and Statistics

Reference
Prediction 0 1
0 14 3
1 3 8

Accuracy : 0.7857
95% CI : (0.5905, 0.917)
No Information Rate : 0.6071
P-Value [Acc > NIR] : 0.03717

Kappa : 0.5508

Mcnemar's Test P-Value : 1.00000

Sensitivity : 0.7273
Specificity : 0.8235
Pos Pred Value : 0.7273
Neg Pred Value : 0.8235
Prevalence : 0.3929
Detection Rate : 0.2857
Detection Prevalence : 0.3929
Balanced Accuracy : 0.7754

'Positive' Class : 1

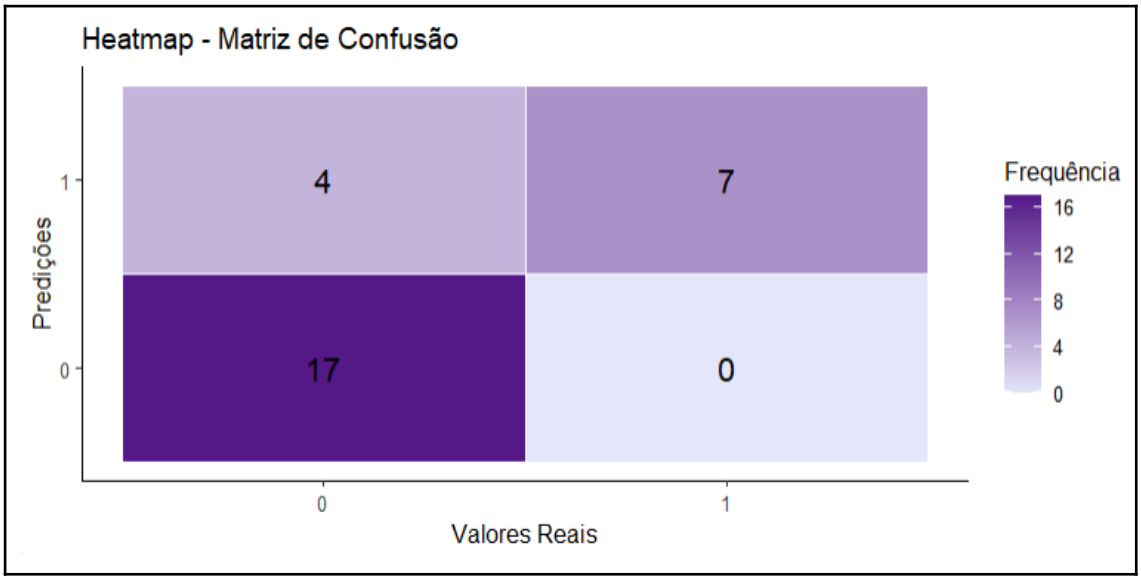
d) Qual foi a acurácia utilizando o kNN?

Acurácia correspondente: 0.7857143 ou 78.57%

4 – Para o **Naive Bayes:**

a) Utilize a amostra de treino para ajustar o Naive Bayes e a amostra de teste para fazer predições.

Gráfico calor - Valores Reais versus Predições:



b) Utilize o Naive Bayes para fazer predições para a amostra de teste. Gere uma matriz de confusão e mostre os resultados abaixo.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 17  4
         1  0  7

               Accuracy : 0.8571
                 95% CI : (0.6733, 0.9597)
    No Information Rate : 0.6071
    P-Value [Acc > NIR] : 0.003981

                  Kappa : 0.68

 Mcnemar's Test P-Value : 0.133614

            Sensitivity : 0.6364
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.8095
```

Prevalence : 0.3929

Detection Rate : 0.2500

Detection Prevalence : 0.2500

Balanced Accuracy : 0.8182

'Positive' Class : 1

c) Qual foi a acurácia utilizando o Naive Bayes?

Acurácia correspondente: 0.8571429 ou 85.71%

5 – Para a **árvore de decisão:**

a) Utilize a amostra de treino para construir uma árvore de decisão.

As amostras de treino e teste utilizadas pela dupla foram as mesmas das que utilizamos na questão 2, letra b.

b) Qual a variável mais importante para a classificação de acordo com seu modelo?

Gráfico de barras - Variáveis versus Importância:

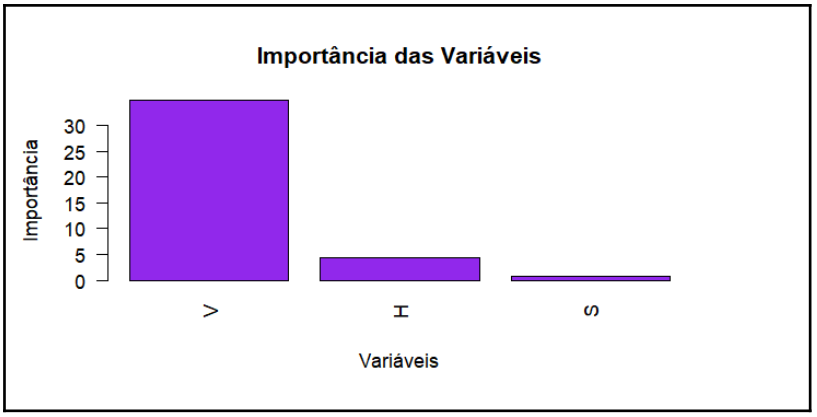


Tabela de Importância versus variáveis:

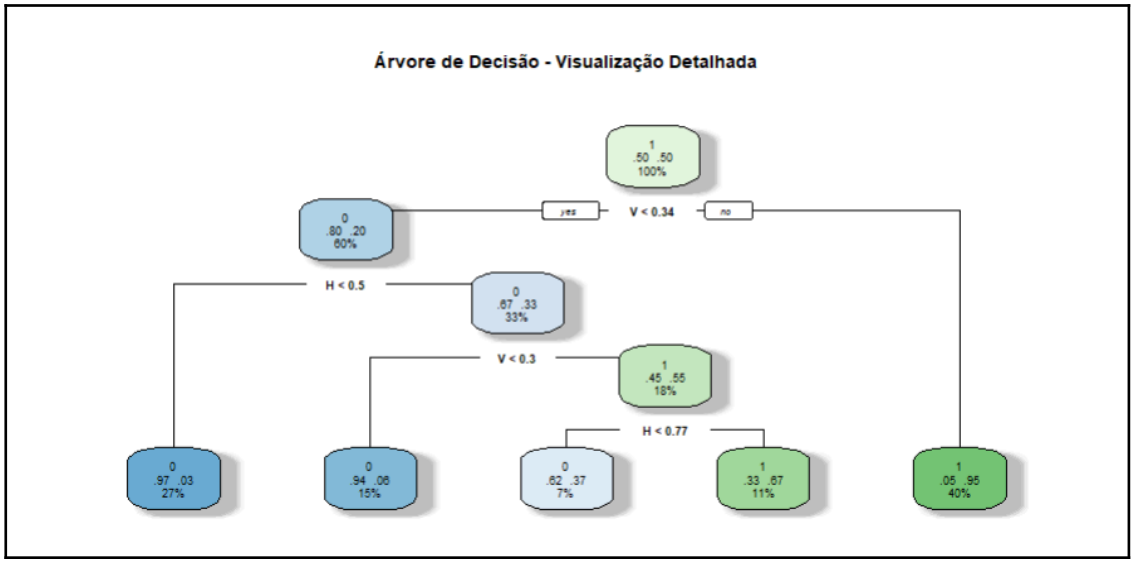
Variável	V	H	S
Valor	34.8010469	4.4778197	0.7921875

Resposta:

A partir dos resultados que obtivemos, nós podemos concluir que a variável "V" é 8 vezes mais importante que "H", fazendo o seguinte cálculo: 34.80 ÷ 4.48 ≈ 7.8. Assim como a variável "H" é 5.7 vezes mais influente que "S", dado pela mesma lógica: 4.48 ÷ 0.79 ≈ 5.7.

c) Faça uma figura para a árvore encontrada e mostre abaixo:

Gráfico - Árvore de decisão de classificação:



d) Utilize a árvore para fazer predições na amostra de teste e gere uma matriz de confusão.

Confusion Matrix and Statistics

Reference
Prediction 0 1
0 14 3
1 3 8

Accuracy : 0.7857
95% CI : (0.5905, 0.917)
No Information Rate : 0.6071
P-Value [Acc > NIR] : 0.03717

Kappa : 0.5508

Mcnemar's Test P-Value : 1.00000

Sensitivity : 0.7273

Specificity : 0.8235
Pos Pred Value : 0.7273
Neg Pred Value : 0.8235
Prevalence : 0.3929
Detection Rate : 0.2857
Detection Prevalence : 0.3929
Balanced Accuracy : 0.7754

'Positive' Class : 1

e) Qual foi a acurácia utilizando a árvore de decisão?

Acurácia da Árvore de Decisão: 0.7857143 ou 78.57 %.

Cole os códigos do R aqui:

```
############################################################
# "Prova" 3 - Modelos Econômicos Quantitativos
# Alunos: André Cunha e Rayane Cardoso - (19/11 até 10/12)
############################################################

# instalação de Pacotes
install.packages("MASS")
install.packages("rattle")
install.packages("rpart")
install.packages("rpart.plot")

# Carregamento de pacotes
library(readr)         # Leitura de arquivos
library(ggplot2)       # Gráficos
library(dplyr)         # Manipulação de dados
library(class)         # kNN
library(caret)         # Métricas e validação cruzada
library(MASS)          # Naive Bayes
library(rpart)         # Árvores de decisão
library(rattle)        # Visualização de árvores
library(rpart.plot)    # Plotagem de árvores

# Carregando Data source
dados <- read_csv("C:/Users/andre/OneDrive/GRADUAÇÃO_ENPRO/ENPRO
```

```
4/Modelos Econômicos Quantitativos/Provas/Prova 3/Dataset-minas
terrestres.csv")
View(dados)

#####################################################################
# 1 - Comece analisando os tipos de minas nesse conjunto de dados.
#####################################################################

# a) Faça uma tabela de frequências e um gráfico de barras para os
dados da coluna M.

table(dados$M) # Tabela

# gráfico de barras
ggplot(dados, aes(x = factor(M, labels = c("Não", "Sim")))) +
  geom_bar(fill = "purple", color = "black") +
  labs(
    title = "Detecção de minas terrestres",
    x = "Detecção",
    y = "Quantidade"
  ) +
  theme_classic()

########################################
#2 - Fixe uma semente aleatória. Depois:
########################################

set.seed(1234)

# a) Separe os dados em uma amostra de treino e outra de teste,
aleatórias (80-20).

sorteio <- sample(1:137,109) # Escolha das observações aleatóriamente

dados_treino <- dados[sorteio,]

dados_teste <- dados[-sorteio,]

# b) Faça uma tabela de frequências para a variável M para a
  # amostra de treino e outra para a amostra de teste.
```

```r
table(dados_treino$M)

table(dados_teste$M)

########################
# 3 - Para o kNN:
########################

# Normalizando os dados -
  # É importante pois o algoritmo calcula a distância entre os
  # pontos para determinar quais são os vizinhos mais próximos.

normaliza <- function(x) {
  return((x - min(x))/(max(x) - min(x)))
}

# Excluindo a variável M
dados_normalizados <- as.data.frame(lapply(dados[, -ncol(dados)],
normaliza))

# Adicionando a variável alvo de volta
dados_normalizados$M <- dados$M

# Separando as amostras conforme a seleção aleatória feita na questão
anterior
dados_treino_norm <- dados_normalizados[sorteio, ]
dados_teste_norm <- dados_normalizados[-sorteio, ]

# a) Testar diferentes valores de k e calcular acurácia
acuracias <- c()
ks <- 1:20 # Vetor para testar os valores de k de 1 a 20

for (k in ks) {
  pred <- knn(
    train = dados_treino_norm[, -ncol(dados_treino_norm)],  # Dados de
treino (sem a variável M)
    test = dados_teste_norm[, -ncol(dados_teste_norm)],    # Dados de
teste (sem a variável M)
    cl = dados_treino_norm$M,                              # Classes
reais do treino
    k = k                                                  # Número de
```

```
vizinhos
  )

  # Calculando a acurácia para o valor de k
  acuracia <- mean(pred == dados_teste_norm$M)
  acuracias <- c(acuracias, acuracia)
}

# Gráfico ocm número de vizinhos versus a acurácia
plot(ks, acuracias, type = "b", # conecta os pontos com linhas e
também exibe os marcadores de cada ponto
     col = "purple", pch = 19, # Define o formato e tamanho dos
marcadores nos pontos do gráfico.
     xlab = "Número de vizinhos (k)",
     ylab = "Acurácia",
     main = "Acurácia para diferentes valores de k")

# b) Escolha do melhor k (concatenando e exibindo texto e valores no
console)
  # Essas modificações foram feitas para facilitar o meu trabalho na
leitura dos dados
melhor_k <- ks[which.max(acuracias)]
cat("Melhor valor de k:", melhor_k, "\n")
cat("Acurácia correspondente:", max(acuracias), "\n")

# c) Aplicar o modelo com o melhor k e gerar a matriz de confusão

#Fazendo a predição com o melhor número de vizinhos
pred_final <- knn(
  train = dados_treino_norm[, -ncol(dados_treino_norm)],
  test = dados_teste_norm[, -ncol(dados_teste_norm)],
  cl = dados_treino_norm$M,
  k = melhor_k
)

# Aplicando a matriz de confusão para o melhor número de vizinhos
matriz_confusao_knn <- confusionMatrix(
  data = factor(pred_final),          # Predições do modelo
  reference = factor(dados_teste_norm$M), # Valores reais (rótulos
verdadeiros)
  positive = "1"                      # Especifica a classe positiva
```

```
)

print(matriz_confusao_knn)

##########################
# 4 - Para o Naive Bayes:
##########################
library(e1071) # Para o Naive Bayes

# a) Ajustando o modelo com a amostra treino
modelo_nb <- naiveBayes(M ~ ., data = dados_treino)

# b.1) Fazendo predições com a amostra teste
pred_nb <- predict(modelo_nb, newdata = dados_teste)

# b.2) Exibindo a matriz de confusão
matriz_confusao_nb <- confusionMatrix(
  data = factor(pred_nb),    # Predições do NB
  reference = factor(dados_teste$M),  # Rótulos reais
  positive = "1"    # Especifica a classe positiva
  )

print(matriz_confusao_nb)

############################
# Representando gráficamente (letra A)
############################

# Transformando a matriz de confusão em um data frame a partir da
conversão de matriz -> tabela
df_heatmap <- as.data.frame(as.table(matriz_confusao_nb))

# Corrigindo o nome das colunas
colnames(df_heatmap) <- c("Real", "Predito", "Freq")

# Revisando se os níveis de "Real" e "Predito" são os mesmos (0 e 1)
df_heatmap$Real <- factor(df_heatmap$Real, levels =
unique(c(df_heatmap$Real, df_heatmap$Predito)))
df_heatmap$Predito <- factor(df_heatmap$Predito, levels =
unique(c(df_heatmap$Real, df_heatmap$Predito)))
```

```
# Gerando o gráfico "heatmap"
ggplot(df_heatmap, aes(x = Real, y = Predito, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), color = "black", size = 5) +
  scale_fill_gradient(low = "lavender", high = "purple4") +
  labs(
    title = "Heatmap - Matriz de Confusão",
    x = "Valores Reais",
    y = "Predições",
    fill = "Frequência"
  ) +
  theme_classic()

# c) Determinando a acurácia do método Naive Bayes

# Acessando apenas a matriz de confusão (tabela) dentro do objeto
matriz_confusao_nb <- confusionMatrix(
  data = factor(pred_nb),
  reference = factor(dados_teste$M),
  positive = "1"
)$table

# Calculando a acurácia
acuracia_nb <- sum(diag(matriz_confusao_nb)) / sum(matriz_confusao_nb)
cat("Acurácia do Naive Bayes:",acuracia_nb, "ou", round(acuracia_nb *
100, 2), "%\n")

##############################
# Fim da entrega PARCIAL - 1
##############################

################################
# 5 - Para a Árvore de Decisão:
################################

# a) Ajustando o modelo com a amostra de treino
modelo_arvore <- rpart(M ~ ., data = dados_treino, method = "class")

# b.1) Fazendo predições com a amostra de teste
pred_arvore <- predict(modelo_arvore, newdata = dados_teste, type =
"class")
```

```
# b.2) Exibindo a matriz de confusão
matriz_confusao_arvore <- confusionMatrix(
  data = factor(pred_arvore),          # Predições da árvore
  reference = factor(dados_teste$M),   # Rótulos reais
  positive = "1"                       # Especifica a classe positiva
)

print(matriz_confusao_arvore)

# c) Determinando a acurácia do método Árvore de Decisão

# c.1) Acessando apenas a matriz de confusão (tabela) dentro do objeto
matriz_confusao_arvore <- confusionMatrix(
  data = factor(pred_arvore),
  reference = factor(dados_teste$M),
  positive = "1"
)$table

# c.2) Calculando a acurácia - Automatizando a leitura da matriz de
confusão
acuracia_arvore <- sum(diag(matriz_confusao_arvore)) /
sum(matriz_confusao_arvore)
cat("Acurácia da Árvore de Decisão:", acuracia_arvore, "ou",
round(acuracia_arvore * 100, 2), "%\n")

# d.1) Determinando a importância das variáveis no modelo ajustado
importancia_variaveis <- modelo_arvore$variable.importance
cat("\nImportância das variáveis:\n")
print(importancia_variaveis)

# d.2) Gráfico da importância das variáveis
barplot(importancia_variaveis,
        main = "Importância das Variáveis",
        col = "purple2",
        las = 2,
        horiz = FALSE,
        xlab = "Variáveis", ylab = "Importância")

# e) Visualização da árvore de decisão
# e.1) 1° forma de plotar a árvore de decisão (pacote rattle)
```

```
fancyRpartPlot(modelo_arvore)

# e.2) 2ª forma de plotar a árvore de forma mais detalhada (pacote
rpart.plot)
rpart.plot(
  modelo_arvore,
  type = 2,                # Tipo de plot (2 = texto nos nós)
  extra = 104,             # Mostra porcentagens e contagens
  fallen.leaves = TRUE,    # Coloca os nós terminais no mesmo nível
  shadow.col = "gray",     # Adiciona sombra para profundidade
  box.palette = "BuGn",    # Paleta de cores para os nós
  main = "Árvore de Decisão - Visualização Detalhada"
)

##############################
# Fim da entrega PARCIAL - 2
##############################
```