Comandos R - Pesquisa com Dados de Satélite (Satellite)

Relação dos comandos executados no script e suas respectivas saídas

OBS: Quando a saída for `N/A` significa que o comando não gerou um output

---

→ comando

```
mirror <- "cran-r.c3sl.ufpr.br"
options(repos = mirror)

# Instale o pacote mlbench se ainda não o tiver instalado
install.packages("mlbench")

# Carregue o pacote
library(mlbench)
```

← saída

```
tentando a URL 'cran-r.c3sl.ufpr.br/bin/macosx/big-sur-arm64/contrib/4.3/mlbench_2.1-3.1.tg
Content type 'application/x-gzip' length 1051850 bytes (1.0 MB)
==================================================
downloaded 1.0 MB


The downloaded binary packages are in
        /var/folders/40/p6_z3q455bz4wy3rkbmlnmp40000gn/T//RtmpEm2yFz/downloaded_packages
```

→ comando

```
# 1- Carregue a base de dados Satellite
data(Satellite)

# Visualize a estrutura da base de dados
str(Satellite)
```

← saída

```
'data.frame':   6435 obs. of  37 variables:
 $ x.1    : num  92 84 84 80 84 80 76 76 76 76 ...
 $ x.2    : num  115 102 102 102 94 94 102 102 89 94 ...
 $ x.3    : num  120 106 102 102 102 98 106 106 98 98 ...
 $ x.4    : num  94 79 83 79 79 76 83 87 76 76 ...
 $ x.5    : num  84 84 80 84 80 80 76 80 76 76 ...
 $ x.6    : num  102 102 102 94 94 102 102 98 94 98 ...
 $ x.7    : num  106 102 102 102 98 102 106 106 98 102 ...
 $ x.8    : num  79 83 79 79 76 79 87 79 76 72 ...
 $ x.9    : num  84 80 84 80 80 76 80 76 76 76 ...
 $ x.10   : num  102 102 94 94 102 102 98 94 98 94 ...
 $ x.11   : num  102 102 102 98 102 102 106 102 102 90 ...
 $ x.12   : num  83 79 79 76 79 79 79 76 72 76 ...
 $ x.13   : num  101 92 84 84 84 76 80 80 80 76 ...
 $ x.14   : num  126 112 103 99 99 99 107 112 95 91 ...
 $ x.15   : num  133 118 104 104 104 104 118 118 104 104 ...
 $ x.16   : num  103 85 81 78 81 81 88 88 74 74 ...
 $ x.17   : num  92 84 84 84 76 76 80 80 76 76 ...
 $ x.18   : num  112 103 99 99 99 99 112 107 91 95 ...
 $ x.19   : num  118 104 104 104 104 108 118 113 104 100 ...
 $ x.20   : num  85 81 78 81 81 85 88 85 74 78 ...
 $ x.21   : num  84 84 84 76 76 76 80 80 76 76 ...
 $ x.22   : num  103 99 99 99 99 103 107 95 95 91 ...
 $ x.23   : num  104 104 104 104 108 118 113 100 100 100 ...
 $ x.24   : num  81 78 81 81 85 88 85 78 78 74 ...
 $ x.25   : num  102 88 84 84 84 84 79 79 75 75 ...
 $ x.26   : num  126 121 107 99 99 103 107 103 91 91 ...
 $ x.27   : num  134 128 113 104 104 104 113 104 96 96 ...
 $ x.28   : num  104 100 87 79 79 79 87 83 75 71 ...
 $ x.29   : num  88 84 84 84 84 79 79 79 75 79 ...
 $ x.30   : num  121 107 99 99 103 107 103 103 91 87 ...
 $ x.31   : num  128 113 104 104 104 109 104 104 96 93 ...
 $ x.32   : num  100 87 79 79 79 87 83 79 71 71 ...
 $ x.33   : num  84 84 84 84 79 79 79 79 79 79 ...
 $ x.34   : num  107 99 99 103 107 107 103 95 87 87 ...
 $ x.35   : num  113 104 104 104 109 109 104 100 93 93 ...
 $ x.36   : num  87 79 79 79 87 87 79 79 71 67 ...
 $ classes: Factor w/ 6 levels "red soil","cotton crop",..: 3 3 3 3 3 3 3 3 4 4 ...
```

→ comando

```r
# 2. Crie 2 partições contendo 80% para treino e 20% para teste
# Instale os pacotes necessários se ainda não os tiver instalado
install.packages("caret")

# Carregue o pacote
library(caret)

# Defina a semente para reprodução dos resultados
set.seed(123)

# Crie a partição
particao <- createDataPartition(Satellite$classes, p = 0.8, list = FALSE)

# Separe os dados de treinamento e teste
dados_treino <- Satellite[particao, ]
dados_teste <- Satellite[-particao, ]

# Verifique o tamanho dos dados de treinamento e teste
print(paste("Tamanho dos dados de treinamento:", nrow(dados_treino)))
print(paste("Tamanho dos dados de teste:", nrow(dados_teste)))
```

← saída

```
tentando a URL 'cran-r.c3sl.ufpr.br/bin/macosx/big-sur-arm64/contrib/4.3/caret_6.0-94.tgz'
Content type 'application/x-gzip' length 3586369 bytes (3.4 MB)
==================================================
downloaded 3.4 MB


The downloaded binary packages are in
        /var/folders/40/p6_z3q455bz4wy3rkbmlnmp40000gn/T//RtmpEm2yFz/downloaded_packages

Carregando pacotes exigidos: ggplot2
Keep up to date with changes at https://tidyverse.org/blog/
Carregando pacotes exigidos: lattice

[1] "Tamanho dos dados de treinamento: 5151"
[1] "Tamanho dos dados de teste: 1284"
```

→ comando

```r
# 3- Treine modelos RandomForest, SVM e RNA para predição destes dados.
install.packages("neuralnet")
# Carregue os pacotes necessários
library(randomForest)
library(e1071)
library(neuralnet)

# 3.1 Treinamento do modelo Random Forest
modelo_rf <- randomForest(classes ~ ., data = dados_treino)

# 3.2 Treinamento do modelo SVM
modelo_svm <- svm(classes ~ ., data = dados_treino)

# 3.3 Treinamento do modelo RNA
modelo_rna <- neuralnet(classes ~ ., data = dados_treino, hidden = c(5, 2), linear.output =

# Exiba os modelos treinados
print(modelo_rf)
print(modelo_svm)
print(modelo_rna)
```

← saída

```
tentando a URL 'cran-r.c3sl.ufpr.br/bin/macosx/big-sur-arm64/contrib/4.3/neuralnet_1.44.2.t
Content type 'application/x-gzip' length 122191 bytes (119 KB)
==================================================
downloaded 119 KB


The downloaded binary packages are in
        /var/folders/40/p6_z3q455bz4wy3rkbmlnmp40000gn/T//RtmpEm2yFz/downloaded_packages

randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.

Attaching package: ‘randomForest’

The following object is masked from ‘package:ggplot2’:

    margin

Call:
 randomForest(formula = classes ~ ., data = dados_treino)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6

        OOB estimate of  error rate: 8.66%
Confusion matrix:
                    red soil cotton crop grey soil damp grey soil
red soil                1206           4        13              0
cotton crop                0         545         1              5
grey soil                  8           1      1044             23
damp grey soil             5           4        94            293
vegetation stubble        22           5         2              3
very damp grey soil        0           0        20             55
                    vegetation stubble very damp grey soil class.error
red soil                             4                   0  0.01711491
cotton crop                          7                   5  0.03197158
grey soil                            2                   9  0.03955842
damp grey soil                       4                 101  0.41516966
vegetation stubble                 509                  25  0.10070671
very damp grey soil                 24                1108  0.08202154


Call:
svm(formula = classes ~ ., data = dados_treino)


Parameters:
```

```
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  1670

$call
neuralnet(formula = classes ~ ., data = dados_treino, hidden = c(5,
    2), linear.output = FALSE)

$response
     grey soil damp grey soil vegetation stubble very damp grey soil
1        FALSE          FALSE               TRUE               FALSE
2        FALSE          FALSE               TRUE               FALSE
3        FALSE          FALSE               TRUE               FALSE
4        FALSE          FALSE               TRUE               FALSE
5        FALSE          FALSE               TRUE               FALSE
6        FALSE          FALSE               TRUE               FALSE
7        FALSE          FALSE               TRUE               FALSE
8        FALSE           TRUE              FALSE               FALSE
9        FALSE           TRUE              FALSE               FALSE
10       FALSE           TRUE              FALSE               FALSE
11       FALSE           TRUE              FALSE               FALSE
12       FALSE           TRUE              FALSE               FALSE
13       FALSE           TRUE              FALSE               FALSE
14       FALSE           TRUE              FALSE               FALSE
15       FALSE          FALSE               TRUE               FALSE
16       FALSE          FALSE               TRUE               FALSE
...

A saída ao executar o print do modelo_rna é extremamente longa.
Portanto, somente o começo da saída foi adicionada a este documento
```

→ comando

```r
# 4. Escolha o melhor modelo com base em suas matrizes de confusão.
# Carregue o pacote 'caret' para calcular a matriz de confusão
library(caret)

# Função para calcular métricas de desempenho
calcular_metricas <- function(matriz_confusao) {
  # Precisão (precision)
  precisao <- diag(matriz_confusao) / colSums(matriz_confusao)

  # Recall
  recall <- diag(matriz_confusao) / rowSums(matriz_confusao)

  # F1-score
  f1_score <- 2 * (precisao * recall) / (precisao + recall)

  # Retornar as métricas
  return(data.frame(precisao = precisao, recall = recall, f1_score = f1_score))
}

# Função para imprimir as métricas
imprimir_metricas <- function(nome_modelo, matriz_confusao) {
  cat("\nModelo:", nome_modelo, "\n")
  print(calcular_metricas(matriz_confusao))
}

# Função para plotar a matriz de confusão
plotar_matriz_confusao <- function(nome_modelo, matriz_confusao) {
  confusionMatrix(matriz_confusao, main = nome_modelo)
}
```

← saída

```
N/A
```

→ comando

```r
# Prever os rótulos usando cada modelo
predicoes_rf <- predict(modelo_rf, newdata = dados_teste)
predicoes_svm <- predict(modelo_svm, newdata = dados_teste)
predicoes_rna <- predict(modelo_rna, newdata = dados_teste)

# Prever as probabilidades usando a RNA
probabilidades_rna <- predict(modelo_rna, newdata = dados_teste)

# Obter os nomes das classes
nomes_classes <- levels(dados_teste$classes)

# Transformar probabilidades em rótulos de classe
predicoes_rna <- apply(probabilidades_rna, 1, function(x) nomes_classes[which.max(x)])

# Converter as predições em fatores
predicoes_rna <- factor(predicoes_rna, levels = nomes_classes)
```

← saída

```
N/A
```

→ comando

```r
# Calcular a matriz de confusão para cada modelo
matriz_confusao_rf <- confusionMatrix(predicoes_rf, dados_teste$classes)
matriz_confusao_svm <- confusionMatrix(predicoes_svm, dados_teste$classes)
matriz_confusao_rna <- confusionMatrix(predicoes_rna, dados_teste$classes)

# Imprimir as métricas de desempenho para cada modelo
imprimir_metricas("Random Forest", matriz_confusao_rf$table)
imprimir_metricas("SVM", matriz_confusao_svm$table)
imprimir_metricas("RNA", matriz_confusao_rna$table)

# Plotar as matrizes de confusão
plotar_matriz_confusao("Random Forest", matriz_confusao_rf$table)
plotar_matriz_confusao("SVM", matriz_confusao_svm$table)
plotar_matriz_confusao("RNA", matriz_confusao_rna$table)
```

← saída

```
Modelo: Random Forest
                     precisao    recall  f1_score
red soil            0.9934641 0.9712460 0.9822294
cotton crop         0.9857143 0.9857143 0.9857143
grey soil           0.9667897 0.9065744 0.9357143
damp grey soil      0.7200000 0.8181818 0.7659574
vegetation stubble  0.8510638 0.9600000 0.9022556
very damp grey soil 0.9169435 0.8990228 0.9078947

Modelo: SVM
                     precisao    recall  f1_score
red soil            0.9934641 0.9589905 0.9759230
cotton crop         0.9785714 0.9785714 0.9785714
grey soil           0.9667897 0.8704319 0.9160839
damp grey soil      0.6320000 0.7117117 0.6694915
vegetation stubble  0.8085106 0.9500000 0.8735632
very damp grey soil 0.8704319 0.8881356 0.8791946

Modelo: RNA
                    precisao     recall  f1_score
red soil                   0        NaN       NaN
cotton crop                0        NaN       NaN
grey soil                  0        NaN       NaN
damp grey soil             1 0.09735202 0.1774308
vegetation stubble         0        NaN       NaN
very damp grey soil        0        NaN       NaN
```

→ comando

```r
# 5- Indique qual modelo dá o melhor o resultado e a métrica u2lizada
# Extrair os valores de F1-score para cada modelo
f1_rf <- calcular_metricas(matriz_confusao_rf$table)$f1_score
f1_svm <- calcular_metricas(matriz_confusao_svm$table)$f1_score
f1_rna <- calcular_metricas(matriz_confusao_rna$table)$f1_score

# Criar um data frame com os valores de F1-score
df_f1 <- data.frame(Modelo = c("Random Forest", "SVM", "RNA"),
                    F1_Score = c(f1_rf, f1_svm, f1_rna))

# Ordenar o data frame pelo F1-score
df_f1 <- df_f1[order(df_f1$F1_Score, decreasing = TRUE), ]

# Imprimir o data frame
print(df_f1)

# Identificar o melhor modelo
melhor_modelo <- df_f1[1, "Modelo"]
cat("\nO melhor modelo é:", melhor_modelo, "\n")
```

← saída

```
          Modelo  F1_Score
2            SVM 0.9857143
1  Random Forest 0.9822294
8            SVM 0.9785714
7  Random Forest 0.9759230
3            RNA 0.9357143
9            RNA 0.9160839
6            RNA 0.9078947
5            SVM 0.9022556
12           RNA 0.8791946
11           SVM 0.8735632
4  Random Forest 0.7659574
10 Random Forest 0.6694915
16 Random Forest 0.1774308
13 Random Forest       NaN
14           SVM       NaN
15           RNA       NaN
17           SVM       NaN
18           RNA       NaN

O melhor modelo é: SVM
```