Comandos R - Estimativa de volumes de árvores

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

```
# --- 00 Instalação e carregamento de pacotes necessários e funções básicas ---

# Função para log

log <- function(msg) {
  cat("\n", format(Sys.time(), "%d-%m-%Y %H:%M:%S"), "-", msg, "\n")
}
```

← saída

```
N/A
```

→ comando

```
# Instalação dos pacotes necessários

mirror <- "cran-r.c3sl.ufpr.br"

log(paste("Instalando e carregando pacotes necessários. Mirror: ", mirror))

options(repos = mirror)
install.packages("e1071")
install.packages("randomForest")
install.packages("kernlab")
install.packages("neuralnet")
install.packages("caret")
```

← saída

```
  18-04-2024 08:19:50 - Instalando e carregando pacotes necessários. Mirror:  cran-r.c3sl.u

tentando a URL 'cran-r.c3sl.ufpr.br/bin/macosx/big-sur-arm64/contrib/4.3/e1071_1.7-14.tgz'
Content type 'application/x-gzip' length 662464 bytes (646 KB)
==================================================
downloaded 646 KB


The downloaded binary packages are in
        /var/folders/40/p6_z3q455bz4wy3rkbmlnmp40000gn/T//RtmpQotcSh/downloaded_packages
> install.packages("randomForest")
tentando a URL 'cran-r.c3sl.ufpr.br/bin/macosx/big-sur-arm64/contrib/4.3/randomForest_4.7-1
Content type 'application/x-gzip' length 256002 bytes (250 KB)
==================================================
downloaded 250 KB

...
Mais informações sobre os pacotes baixados, mas que foram
ocultadas aqui para não tomar tanto espaço
```

→ comando

```r
# Carregamento dos pacotes

library("neuralnet")
library("caret")
```

← saída

```
  Carregando pacotes exigidos: ggplot2
  Carregando pacotes exigidos: lattice
```

→ comando

```r
# --- 01 Carregar o arquivo Volumes.csv (http://www.razer.net.br/datasets/Volumes.csv) ---

url_dataset <- "http://www.razer.net.br/datasets/Volumes.csv"

# Carregando a base de dados (ex 1)

log(paste("Carregando base de dados de volumes de árvores. URL:", url_dataset))

dataset <- read.csv2(url_dataset, header = TRUE, sep = ";")
```

← saída

```
  18-04-2024 08:19:54 - Carregando base de dados de volumes de árvores. URL: http://www.raz
```

→ comando

```r
# --- 02 Eliminar a coluna NR, que só apresenta um número sequencial ---

dataset <- dataset[, !names(dataset) %in% "NR"]

# Visualizando a base de dados

log("Estrutura da base do dataset")
str(dataset)

log("Primeiras linhas da base do dataset")
head(dataset)

log("Sumário da base do dataset")
summary(dataset)
```

← saída

```
  18-04-2024 08:19:55 - Estrutura da base do dataset
  'data.frame': 100 obs. of  4 variables:
  $ DAP: num  34 41.5 29.6 34.3 34.5 29.9 28.4 29.5 36.3 36.3 ...
  $ HT : num  27 27.9 26.4 27.1 26.2 ...
  $ HP : num  1.8 2.75 1.15 1.95 1 1.9 2.3 2.4 1.8 1.5 ...
  $ VOL: num  0.897 1.62 0.801 1.079 0.98 ...

  18-04-2024 08:19:55 - Primeiras linhas da base do dataset
    DAP    HT   HP       VOL
1 34.0 27.00 1.80 0.8971441
2 41.5 27.95 2.75 1.6204441
3 29.6 26.35 1.15 0.8008181
4 34.3 27.15 1.95 1.0791682
5 34.5 26.20 1.00 0.9801112
6 29.9 27.10 1.90 0.9067022

  18-04-2024 08:19:55 - Sumário da base do dataset
  Min.   :27.50   Min.   :22.80   Min.   :1.000   Min.   :0.6982
  1st Qu.:32.50   1st Qu.:25.50   1st Qu.:1.700   1st Qu.:1.0553
  Median :35.25   Median :26.40   Median :2.225   Median :1.3006
  Mean   :35.86   Mean   :26.24   Mean   :2.135   Mean   :1.3583
  3rd Qu.:39.05   3rd Qu.:27.15   3rd Qu.:2.562   3rd Qu.:1.6191
  Max.   :49.00   Max.   :28.40   Max.   :3.400   Max.   :2.5245
```

→ comando

```r
# --- 03 Criar partição de dados: treinamento 80%, teste 20% ---

# Setando uma semente de aleatoriedade

set.seed(123)

# Criando índices para o treino

log("Particionando dados em treino e teste")

indices <- createDataPartition(dataset$VOL, p = 0.8, list = FALSE)

# Separando dados em treino e teste

dados_treino <- dataset[indices, ]
dados_teste <- dataset[-indices, ]

# Verificando quantidade de observações há em cada partição

paste("Observações nos dados de treinamento:", nrow(dados_treino))
paste("Observações nos dados de teste:", nrow(dados_teste))
```

← saída

```
  18-04-2024 08:19:55 - Particionando dados em treino e teste
[1] "Observações nos dados de treinamento: 80"
[1] "Observações nos dados de teste: 20"
```

→ comando

```
# --- 04 Usando o pacote "caret", treinar os modelos: Random Forest (rf), SVM (svmRadial),
# O modelo alométrico é dado por: Volume = b0 + b1 * dap ^ 2 * Ht

# Treinando os modelos

log("Treinando modelo Random Forest")
rf <- train(VOL ~ ., data = dados_treino, method = "rf")

log("Treinando modelo SVM")
svm <- train(VOL ~ ., data = dados_treino, method = "svmRadial")

log("Treinando modelo Neural Network")
rna <- train(VOL ~ ., data = dados_treino, method = "neuralnet")

log("Treinando modelo Alométrico de SPURR")
alom <- nls(VOL ~ b0 + b1 * (DAP ^ 2) * HT, data = dados_treino, start = list(b0 = 0.5, b1
```

← saída

```
  18-04-2024 08:19:55 - Treinando modelo Random Forest
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .


  18-04-2024 08:19:57 - Treinando modelo SVM

  18-04-2024 08:19:57 - Treinando modelo Neural Network
There were 37 warnings (use warnings() to see them)

  18-04-2024 08:21:41 - Treinando modelo Alométrico de SPURR
```

→ comando

```
# --- 05 Efetue as predições nos dados de teste

log("Realizando predições")
predicoes_rf <- predict(rf, dados_teste)
predicoes_svm <- predict(svm, dados_teste)
predicoes_rna <- predict(rna, dados_teste)
predicoes_alom <- predict(alom, dados_teste)
```

← saída

```
  18-04-2024 08:21:41 - Realizando predições
```

→ comando

```
# --- 06 Crie suas próprias funções (UDF) e calcule as seguintes métricas entre a predição

# Função para cálculo do coeficiente de determinação R2
calcular_coef_r2 <- function(observacoes, predicoes) {
  return(1 - sum((observacoes - predicoes) ^ 2) / sum((observacoes - mean(observacoes)) ^ 2
}

# Função para erro padrão de estimativ: Syx
calcular_erro_syx <- function(observacoes, predicoes) {
  return(sqrt(sum((observacoes - predicoes) ^ 2) / (length(observacoes) - 2)))
}

# Função para o calculo da porcentagem de erro Syx
calcular_erro_syx_percent <- function(observacoes, predicoes) {
  return((calcular_erro_syx(observacoes, predicoes) / mean(observacoes)) * 100)
}

# Função para calcular um score com base no valor de R2 e Syx
calcular_score <- function(r2, syx) {
  return((r2 + (1 - syx)) / 2)
}

# Função para retornar as metricas de avaliação
calcular_metricas <- function(observacoes, predicoes, nome_modelo) {
  r2 <- calcular_coef_r2(observacoes, predicoes)
  syx <- calcular_erro_syx(observacoes, predicoes)
  syx_percent <- calcular_erro_syx_percent(observacoes, predicoes)
  score <- calcular_score(r2, syx)

  return(data.frame(model = nome_modelo, r2 = r2, syx = syx, syxPercentage = syx_percent, s
}
```

← saída

```
N/A
```

→ comando

```r
# --- 07 Escolha o melhor modelo ---

log("Calculando métricas para os modelos")

metricas_df <- calcular_metricas(dados_teste$VOL, predicoes_rf, "rf")
metricas_df <- rbind(metricas_df, calcular_metricas(dados_teste$VOL, predicoes_svm, "svm"))
metricas_df <- rbind(metricas_df, calcular_metricas(dados_teste$VOL, predicoes_rna, "rna"))
metricas_df <- rbind(metricas_df, calcular_metricas(dados_teste$VOL, predicoes_alom, "alom"

metricas_df <- metricas_df[order(metricas_df$score, decreasing=TRUE), ]

log("Métricas com base no score (melhor para o pior):")
print(metricas_df)
```

← saída

```
  18-04-2024 08:21:41 - Calculando métricas para os modelos
  18-04-2024 08:21:41 - Métricas com base no score (melhor para o pior):
  model        r2       syx syxPercentage     score
3   rna 0.8867930 0.1354473      10.06305 0.8756729
4  alom 0.8694429 0.1454567      10.80670 0.8619931
1    rf 0.8486654 0.1566040      11.63489 0.8460307
2   svm 0.7900779 0.1844433      13.70321 0.8028173
```