MAS: Trabalho de Grupo (14)

Hélio Henrique Ferreira Samuel

25 de março, 2022

```
rm(list=ls(all=TRUE)) #Remover tudo

#Libraries necessárias:
library(psych)

## Warning: package 'psych' was built under R version 4.0.5

library(lsr)

## Warning: package 'lsr' was built under R version 4.0.5

library(Metrics)

## Warning: package 'Metrics' was built under R version 4.0.5

library(tree)

## Warning: package 'tree' was built under R version 4.0.5

## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli

library(nnet)

## Warning: package 'nnet' was built under R version 4.0.5

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5

## Error : 'format_warning' is not an exported object from
'namespace:cli'

## Warning: replacing previous import 'ellipsis::check_dots_unnamed' by
## 'rlang::check_dots_unnamed' when loading 'pillar'

## Warning: replacing previous import 'ellipsis::check_dots_used' by
## 'rlang::check_dots_used' when loading 'pillar'

## Warning: replacing previous import 'ellipsis::check_dots_empty' by
## 'rlang::check_dots_empty' when loading 'pillar'

## Warning: replacing previous import 'ellipsis::check_dots_unnamed' by
## 'rlang::check_dots_unnamed' when loading 'tibble'
```

```
## Warning: replacing previous import 'ellipsis::check_dots_used' by
## 'rlang::check_dots_used' when loading 'tibble'

## Warning: replacing previous import 'ellipsis::check_dots_empty' by
## 'rlang::check_dots_empty' when loading 'tibble'

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ISLR)

## Warning: package 'ISLR' was built under R version 4.0.5

library(MASS)

## Warning: package 'MASS' was built under R version 4.0.5

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

library(FNN)

## Warning: package 'FNN' was built under R version 4.0.5
```

1. Leitura dos dados Cellular e análise preliminar dos mesmos

1.1) Leitura os dados usando cellular<-read.csv("Cellular.csv", header=TRUE, dec=".", sep=";").

```
cellular = read.csv("cellular.csv", sep = ";", header=TRUE) #É necessário
selecionar a diretoria usada para a sessão
names(cellular)

## [1] "minutes"  "bill"     "business" "los"      "income"   "score"
"score_r"

dim(cellular)

## [1] 250   7

#View(cellular) - código de visualização do DataFrame
```

1.2) [1 valor] Apresentação de head(cellular), definição do fator score_r e apresentação da tabela de frequências absolutas correspondente

```
#Head do cellular
head(cellular)
```

```
##   minutes  bill business  los income score score_r
## 1  276.46 48.43    28.11 3.50  68.86 64.98       1
## 2  189.01 61.93    22.57 2.42  77.31 52.65       1
## 3  197.49 47.90    27.48 2.42  56.89 63.72       1
## 4  256.77 66.92    44.84 2.34  75.23 72.11       1
## 5  274.82 72.78    37.56 3.38  87.60 83.45       1
## 6  207.29 55.83    36.89 3.18  72.72 70.41       1
```

```
#Fatorização do score_r
cellular$score_r = as.factor(cellular$score_r)

#Tabela de frequências absolutas
table(cellular$score_r)
```

```
##
##   0   1
## 200  50
```

1.3)[0.5 valores] Realização de uma análise descritiva dos dados apresentando o número de observações, mínimo, máximo, média, desvio padrão, medida de assimetria e de achatamento

```
summary(cellular)
```

```
##     minutes            bill           business          los
##  Min.   : 53.64   Min.   :  8.01   Min.   : 5.65   Min.   :1.020
##  1st Qu.:131.80   1st Qu.: 49.68   1st Qu.:27.19   1st Qu.:2.290
##  Median :158.47   Median : 63.70   Median :32.43   Median :2.680
##  Mean   :162.19   Mean   : 63.40   Mean   :32.68   Mean   :2.680
##  3rd Qu.:188.84   3rd Qu.: 77.19   3rd Qu.:38.24   3rd Qu.:3.087
##  Max.   :326.25   Max.   :121.24   Max.   :59.23   Max.   :4.370
##      income          score       score_r
##  Min.   :30.15   Min.   :16.71   0:200
##  1st Qu.:55.28   1st Qu.:33.03   1: 50
##  Median :60.93   Median :37.91
##  Mean   :61.59   Mean   :41.54
##  3rd Qu.:68.81   3rd Qu.:44.86
##  Max.   :95.44   Max.   :83.45
```

```
describe = describe(cellular)
describe[1:5] # primeiras descritivas
```

```
##          vars   n   mean    sd median
## minutes     1 250 162.19 46.57 158.47
## bill        2 250  63.40 19.80  63.70
## business    3 250  32.68  9.07  32.43
```

```
## los          4 250   2.68  0.60   2.68
## income       5 250  61.59 11.12  60.93
## score        6 250  41.54 13.32  37.92
## score_r*     7 250   1.20  0.40   1.00

describe[6:11] # restantes descritivas

##           trimmed   mad   min    max  range  skew
## minutes    159.72 41.91 53.64 326.25 272.61  0.54
## bill        63.49 20.67  8.01 121.24 113.23  0.02
## business    32.53  8.46  5.65  59.23  53.58  0.09
## los          2.69  0.59  1.02   4.37   3.35 -0.10
## income      61.71 10.70 30.15  95.44  65.29 -0.06
## score       40.09  8.19 16.71  83.45  66.74  1.02
## score_r*     1.12  0.00  1.00   2.00   1.00  1.49
```

1.4) [0.5 valores] Divisão dos dados em amostra de treino (65%) e de teste (35%) usando set.seed(888) e apresentação de tabela de frequências absolutas de score_r em cada amostra

```
set.seed(888)
ind_treino = sample(nrow(cellular), nrow(cellular)*.65)

# Treino
cellular_treino = cellular[ind_treino,]

# Teste
cellular_teste = cellular[-ind_treino,]

# Tabela de frequências absolutas de score_r para amostra de treino
table(cellular_treino$score_r)

##
##   0   1
## 125  37

# Tabela de frequências absolutas de score_r para amostra de teste
table(cellular_teste$score_r)

##
##  0  1
## 75 13
```

1.5) [0.5 valores] Obtenção dos dados dos preditores normalizados (normalização 0-1), nas amostras de treino e teste, e apresentação das primeiras 6 linhas destas amostras após normalização

```
# Definir a função de normalização
normalize_min_max <- function(x){
  return ((x - min(x)) / (max(x)-min(x)))}
```

```r
# Aplicação da função de normalização nas amostras de treino e de teste
cellular_treino_n = cellular_treino
cellular_treino_n[, 1:5] = sapply(cellular_treino_n[,1:5],
normalize_min_max)

cellular_teste_n = cellular_teste
cellular_teste_n[, 1:5] = sapply(cellular_teste_n[, 1:5],
normalize_min_max)
```

1.6) [1 valor] Completação das frases seguintes em comentário do script (com eventual obtenção de resultados adicionais):

```r
# A dimensão de "Cellular.csv" é de 250 linhas e 7 colunas; na amostra
original encontram-se 200 casos com score_r = "No churn" e no conjunto de
teste desta categoria corresponde a 85.23 % das observações.
```

2. Aprendizagem, sobre a amostra de treino, do 3-Nearest Neighbour (baseado em dois preditores) para prever score_r e avaliação do seu desempenho

2.1) [1.5 valores] Escolha dos preditores, justificando

```r
# Vamos analisar os preditores mais promissores

correlacoes = round(cor(cellular_treino_n[, 1:6]), 2)
correlacoes[1:5, 6]
```

```
##  minutes     bill business      los   income
##     0.64     0.31     0.19     0.18     0.22
```

```r
# Vamos escolher minutes e bill como os dois melhores preditores pois são
os que apresentam correlações mais altas com a variável target score
```

2.2) [2 valores] Obtenção do modelo e das correspondentes estimativas de score_r sobre amostra de teste

```r
knn_cellular = knn.reg(cellular_treino_n[, c(1,2)],
                       test = cellular_teste_n[, c(1,2)],
                       y = cellular_treino_n$score, k=3,
                       algorithm="brute")

str(knn_cellular)
```

```
## List of 7
##  $ call     : language knn.reg(train = cellular_treino_n[, c(1, 2)],
test = cellular_teste_n[,      c(1, 2)], y = cellular_treino_n$scor|
__truncated__
##  $ k        : num 3
##  $ n        : int 88
```

```
##  $ pred     : num [1:88] 65.6 69 58.3 67.7 68.2 ...
##  $ residuals: NULL
##  $ PRESS    : NULL
##  $ R2Pred   : NULL
##  - attr(*, "class")= chr "knnReg"
```

```
pred_knn_cellular = knn_cellular[4]
pred_knn_cellular # Nova lista com todos os valores previstos sobre a
amostra de teste na variável score
```

```
## $pred
##  [1] 65.63667 69.02667 58.31333 67.66333 68.18667 61.50667 64.00000
55.55667
##  [9] 65.63667 69.02667 63.22000 72.45333 56.56333 51.70333 28.69333
67.34667
## [17] 46.17000 30.46000 33.68333 62.21333 30.77333 41.27333 34.24667
30.77333
## [25] 35.34667 67.66333 43.37333 46.56333 31.28000 47.40333 50.12000
35.47333
## [33] 30.20000 50.12000 29.57333 29.80333 34.58000 38.54000 37.44333
34.10667
## [41] 33.90667 36.11667 35.42000 34.77000 38.56000 55.55667 37.78333
72.98000
## [49] 40.04333 41.79667 36.71333 47.97000 39.42000 37.84333 28.75333
37.89667
## [57] 28.43000 35.95667 36.74000 44.72667 38.61000 51.70333 46.56333
36.86667
## [65] 42.43333 28.43000 40.78333 36.70000 55.55667 31.97667 62.36000
43.83667
## [73] 28.75333 39.61000 30.07000 36.21000 35.93667 35.87333 51.70333
33.90667
## [81] 35.81667 38.36000 35.47333 42.58000 37.27000 60.28333 56.93667
35.26000
```

2.3) [2 valores] Apresentação da Confusion matrix sobre amostra de teste e do índice de Huberty correspondente

```
pred_knn_cellular_r = knn_cellular$pred
pred_knn_cellular_r[pred_knn_cellular_r < 50] = 0
pred_knn_cellular_r[pred_knn_cellular_r >= 50] = 1
pred_knn_cellular_r
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0
1 0 0 0 0
## [39] 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1
0 0 0 0 0
## [77] 0 0 1 0 0 0 0 0 0 1 1 0
```

```
confusion_mat_ = table(cellular_teste_n$score_r, pred_knn_cellular_r)
confusion_mat_
```

```
##    pred_knn_cellular_r
##      0  1
##   0 61 14
##   1  0 13
```

```
# Accuracy do modelo
accuracy_cellular = accuracy(cellular_teste_n$score_r,
pred_knn_cellular_r)
accuracy_cellular
```

```
## [1] 0.8409091
```

```
# Default-p
default_p = max(table(cellular$score_r) / nrow(cellular))
default_p
```

```
## [1] 0.8
```

```
# Huberty
cellular_huberty = (accuracy_cellular - default_p) / (1-default_p)
cellular_huberty
```

```
## [1] 0.2045455
```

```
# Valor mais próximo da primeira observação no conjunto de teste

knn_cellular_cv = knn.cv(cellular[,1:5], cellular$score_r, k=3,
prob=TRUE, algorithm="brute")

attr(knn_cellular_cv,"nn.index")[4,]
```

```
## [1] 41 46 17
```

2.4) [2 valores] Completação das frases seguintes em comentário do script (com eventual obtenção de resultados adicionais):

```
# Na aprendizagem foram usados dados normalizados;
# as observações mais próximas da primeira observação do conjunto de
teste são 41, 46, 17;
# a probabilidade da última observação do conjunto de teste pertencer à
classe alvo "No churn", estimada pelo modelo, é ______ ;
# segundo os resultados estimados, o churn dos clientes na amostra de
teste será 22.73 %.
```

3. Aprendizagem, sobre a amostra de treino, de uma Árvore de Regressão para prever score e avaliação do seu desempenho

3.1) [1.5 valores] Obtenção do modelo, com cerca de 10 nós folha, e apresentação da árvore correspondente

```
cellular_tree = tree(cellular_treino$score~. ,data = cellular_treino,
control=tree.control(nrow(cellular_treino), mincut = 1,  minsize = 2,
mindev = 0.001), split = "deviance")

# cellular_tree

cellular_tree_10 = prune.tree(cellular_tree, best=10)
cellular_tree_10
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 162 29300.000 42.41
##    2) score_r: 0 125  5053.000 35.96
##      4) bill < 29.465 5   228.300 28.15 *
##      5) bill > 29.465 120  4507.000 36.29
##       10) business < 25.265 27   763.300 38.40
##         20) minutes < 91.09 3    86.120 29.79 *
##         21) minutes > 91.09 24   427.100 39.48 *
##       11) business > 25.265 93  3589.000 35.68
##         22) business < 27.665 7   230.600 29.91
##           44) business < 27.32 5    14.740 33.38 *
##           45) business > 27.32 2     4.836 21.22 *
##         23) business > 27.665 86  3106.000 36.15
##           46) los < 2.03 8   273.300 40.81 *
##           47) los > 2.03 78  2641.000 35.67 *
##    3) score_r: 1 37  1490.000 64.20
##      6) income < 56.135 6    20.890 57.10 *
##      7) income > 56.135 31  1109.000 65.57
##       14) income < 66.68 12   444.400 68.68 *
##       15) income > 66.68 19   475.100 63.61 *
```

3.2) [1.5 valores] Estimação de score sobre amostra de teste, a partir da árvore obtida, e apresentação das estimativas correspondentes às 6 primeiras observações desta amostra

```
pred_cellular_tree = predict(cellular_tree_10, cellular_teste)
#pred_cellular_tree

# 6 primeiras observações

pred_cellular_tree[1:6]
```

```
##        4        5       12       13       21       23
## 63.60579 63.60579 68.67917 68.67917 63.60579 57.10000
```

3.3) [1.5 valores] Apresentação de 3 métricas de regressão associadas ao modelo aplicado sobre a amostra de teste

```
# Deviance
deviance(cellular_tree_10)
```

```
## [1] 4615.478
```

```
# MSE
mse = mse(cellular_teste$score, cellular_tree_10$y)
```

```
## Warning in actual - predicted: longer object length is not a multiple
of shorter
## object length
```

```
mse
```

```
## [1] 403.4733
```

```
# MAE
mae = mae(cellular_teste$score, cellular_tree_10$y)
```

```
## Warning in actual - predicted: longer object length is not a multiple
of shorter
## object length
```

```
mae
```

```
## [1] 14.77556
```

```
# R-square
r_square_tree = 1-sum((cellular_teste$score -
pred_cellular_tree)^2)/sum((cellular_teste$score -
mean(cellular_teste$score))^2)
r_square_tree
```

```
## [1] 0.6277124
```

```
# Erro de máquina, o erro dado foi argument "data0" is missing, with no
default.
```

3.4) [1 valor] Apresentação, com base nas estimativas obtidas em 3.2), de uma tabela de frequências para as categorias churn e No churn

```
pred_tree_cellular_r = pred_cellular_tree
pred_tree_cellular_r[pred_tree_cellular_r < 50] = 0
pred_tree_cellular_r[pred_tree_cellular_r >= 50] = 1
pred_tree_cellular_r
```

```
##   4   5  12  13  21  23  26  30  37  38  39  41  48  53  56  59  65
66  67  69
```

```
##   1   1   1   1   1   1   1   1   1   1   1   1   1   0   0   0   0
0   0   0
##  75  84  87  88  89  91  92  96 105 106 110 113 114 115 117 119 125
127 128 131
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0
## 132 134 139 142 143 144 150 151 155 157 158 161 163 170 172 174 176
178 184 186
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0
## 188 189 190 195 197 198 199 200 201 211 213 214 220 222 223 227 229
232 233 235
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0
## 237 240 241 244 246 247 248 249
##   0   0   0   0   0   0   0   0

tabela_cellular_tree = table(pred_tree_cellular_r)
tabela_cellular_tree

## pred_tree_cellular_r
##  0  1
## 75 13

names(tabela_cellular_tree) = c("No churn", "Churn")
tabela_cellular_tree

## No churn    Churn
##       75       13
```

3.5) [2 valores] Completação das frases seguintes em comentário do script (com eventual obtenção de resultados adicionais):

```
# Na aprendizagem foram usados dados não normalizados;
# o R-Square associado ao modelo sobre o teste é 0.6277124;
# o nó folha com menor frequência inclui 2 observações do teste;
# segundo os resultados estimados, a % de observações da amostra de teste
suscetíveis de fazer churn será 14.77 %.
```