Data Science Academy - Mini-Projeto 2

Equipe DSA

11 Julho, 2018

Mini-Projeto 2 - Prevendo a Ocorrência de Câncer

Este projeto é parte integrante do curso Big Data Analytics com R e Microsoft Azure da Formação Cientista de Dados. O objetivo é analisar dados reais sobre exames de câncer de mama realizado com mulheres nos EUA e então prever a ocorrência de novos casos.

Os dados do câncer da mama incluem 569 observações de biópsias de câncer, cada um com 32 características (variáveis). Uma característica é um número de identificação (ID), outro é o diagnóstico de câncer, e 30 são medidas laboratoriais numéricas. O diagnóstico é codificado como "M" para indicar maligno ou "B" para indicar benigno.

Todo o projeto será descrito de acordo com suas etapas.

Etapa 1 - Coletando os Dados

Aqui está a coleta de dados, neste caso um arquivo csv.

```
# Coletando dados
dados <- read.csv("http://datascienceacademy.com.br/blog/aluno/RFundamentos/Datasets/ML/bc_data.csv",
                  stringsAsFactors = FALSE)
str(dados)
```

```
## 'data.frame':    569 obs. of  32 variables:
##  $ id                : int  87139402 8910251 905520 868871 9012568 906539 925291 87880 862989 89827 .
##  $ diagnosis         : chr  "B" "B" "B" "B" ...
##  $ radius_mean       : num  12.3 10.6 11 11.3 15.2 ...
##  $ texture_mean      : num  12.4 18.9 16.8 13.4 13.2 ...
##  $ perimeter_mean    : num  78.8 69.3 70.9 73 97.7 ...
##  $ area_mean         : num  464 346 373 385 712 ...
##  $ smoothness_mean   : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
##  $ compactness_mean  : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
##  $ concavity_mean    : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
##  $ points_mean       : num  0.037 0.0264 0.0248 0.048 0.0266 ...
##  $ symmetry_mean     : num  0.196 0.192 0.171 0.177 0.172 ...
##  $ dimension_mean    : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
##  $ radius_se         : num  0.236 0.451 0.197 0.338 0.178 ...
##  $ texture_se        : num  0.666 1.197 1.387 1.343 0.412 ...
##  $ perimeter_se      : num  1.67 3.43 1.34 1.85 1.34 ...
##  $ area_se           : num  17.4 27.1 13.5 26.3 17.7 ...
##  $ smoothness_se     : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
##  $ compactness_se    : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
##  $ concavity_se      : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
##  $ points_se         : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
##  $ symmetry_se       : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
##  $ dimension_se      : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
##  $ radius_worst      : num  13.5 11.9 12.4 11.9 16.2 ...
##  $ texture_worst     : num  15.6 22.9 26.4 15.8 15.7 ...
##  $ perimeter_worst   : num  87 78.3 79.9 76.5 104.5 ...
```

```
##  $ area_worst       : num  549 425 471 434 819 ...
##  $ smoothness_worst : num  0.139 0.121 0.137 0.137 0.113 ...
##  $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
##  $ concavity_worst  : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
##  $ points_worst     : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
##  $ symmetry_worst   : num  0.283 0.294 0.3 0.21 0.249 ...
##  $ dimension_worst  : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
```

```
#head(dados)
```

Etapa 2 - Explorando os Dados

Independentemente do método de aprendizagem de máquina, deve sempre ser excluídas variáveis de ID. Caso contrário, isso pode levar a resultados errados porque o ID pode ser usado para unicamente "prever" cada exemplo. Por conseguinte, um modelo que inclui um identificador que sofrem de superajuste, e é improvável que generalizar bem a outros dados.

```
# Excluindo a coluna ID
dados <- dados[-1]
str(dados)
```

```
## 'data.frame':    569 obs. of  31 variables:
##  $ diagnosis        : chr  "B" "B" "B" "B" ...
##  $ radius_mean      : num  12.3 10.6 11 11.3 15.2 ...
##  $ texture_mean     : num  12.4 18.9 16.8 13.4 13.2 ...
##  $ perimeter_mean   : num  78.8 69.3 70.9 73 97.7 ...
##  $ area_mean        : num  464 346 373 385 712 ...
##  $ smoothness_mean  : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
##  $ compactness_mean : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
##  $ concavity_mean   : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
##  $ points_mean      : num  0.037 0.0264 0.0248 0.048 0.0266 ...
##  $ symmetry_mean    : num  0.196 0.192 0.171 0.177 0.172 ...
##  $ dimension_mean   : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
##  $ radius_se        : num  0.236 0.451 0.197 0.338 0.178 ...
##  $ texture_se       : num  0.666 1.197 1.387 1.343 0.412 ...
##  $ perimeter_se     : num  1.67 3.43 1.34 1.85 1.34 ...
##  $ area_se          : num  17.4 27.1 13.5 26.3 17.7 ...
##  $ smoothness_se    : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
##  $ compactness_se   : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
##  $ concavity_se     : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
##  $ points_se        : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
##  $ symmetry_se      : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
##  $ dimension_se     : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
##  $ radius_worst     : num  13.5 11.9 12.4 11.9 16.2 ...
##  $ texture_worst    : num  15.6 22.9 26.4 15.8 15.7 ...
##  $ perimeter_worst  : num  87 78.3 79.9 76.5 104.5 ...
##  $ area_worst       : num  549 425 471 434 819 ...
##  $ smoothness_worst : num  0.139 0.121 0.137 0.137 0.113 ...
##  $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
##  $ concavity_worst  : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
##  $ points_worst     : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
##  $ symmetry_worst   : num  0.283 0.294 0.3 0.21 0.249 ...
##  $ dimension_worst  : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
```

```r
# Verificando se existem dados NA
any(is.na(dados))
```

```
## [1] FALSE
```

```r
# Muitos classificadores requerem que as variáveis sejam do tipo Fator
table(dados$diagnosis)
```

```
##
##   B   M
## 357 212
```

```r
dados$diagnosis <- factor(dados$diagnosis, levels = c("B", "M"), labels = c("Benigno", "Maligno"))
str(dados$diagnosis)
```

```
##  Factor w/ 2 levels "Benigno","Maligno": 1 1 1 1 1 1 1 2 1 1 ...
```

```r
# Verificando a proporção
round(prop.table(table(dados$diagnosis)) * 100, digits = 1)
```

```
##
## Benigno Maligno
##    62.7    37.3
```

```r
# Medidas de Tendência Cetral
# Detectamos aqui um problema de escala entre os dados, que então precisam ser normalizados
summary(dados[c("radius_mean", "area_mean", "smoothness_mean")])
```

```
##   radius_mean       area_mean      smoothness_mean
##  Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
##  1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
##  Median :13.370   Median : 551.1   Median :0.09587
##  Mean   :14.127   Mean   : 654.9   Mean   :0.09636
##  3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
##  Max.   :28.110   Max.   :2501.0   Max.   :0.16340
```

```r
# Criando um função de normalização
normalizar <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

# Testando a função de normalização - os resultados devem ser idênticos
normalizar(c(1, 2, 3, 4, 5))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```r
normalizar(c(10, 20, 30, 40, 50))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```r
# Normalizando os dados
dados_norm <- as.data.frame(lapply(dados[2:31], normalizar))

# Confirmando que a normalização funcionou
summary(dados[c("radius_mean", "area_mean", "smoothness_mean")])
```

```
##   radius_mean       area_mean      smoothness_mean
##  Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
##  1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
```

```
##  Median :13.370   Median : 551.1   Median :0.09587
##  Mean   :14.127   Mean   : 654.9   Mean   :0.09636
##  3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
##  Max.   :28.110   Max.   :2501.0   Max.   :0.16340
```

```
summary(dados_norm[c("radius_mean", "area_mean", "smoothness_mean")])
```

```
##   radius_mean       area_mean      smoothness_mean
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2233   1st Qu.:0.1174   1st Qu.:0.3046
##  Median :0.3024   Median :0.1729   Median :0.3904
##  Mean   :0.3382   Mean   :0.2169   Mean   :0.3948
##  3rd Qu.:0.4164   3rd Qu.:0.2711   3rd Qu.:0.4755
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Etapa 3 - Treinando o modelo

Com os dados devidamente normalizados, podemos agora começar o processo de treinamento do modelo. Para isso, vamos dividir nosso conjunto de dados em dados de treino e dados de teste.

```
# Carregando o pacote library
# install.packages("class")
library(class)

# Criando dados de treino e dados de teste
dados_treino <- dados_norm[1:469, ]
dados_teste <- dados_norm[470:569, ]

# Criando os labels para os dados de treino e de teste
dados_treino_labels <- dados[1:469, 1]
dados_teste_labels <- dados[470:569, 1]

#?knn
# Criando o modelo
modelo <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k = 21)

# A função knn() retorna um objeto do tipo fator com as previsões para cada exemplo no dataset de teste
class(modelo)
```

```
## [1] "factor"
```

Etapa 4 - Avaliando a Performance do Modelo

Vamos agora avaliar a performance do modelo.

```
# Carregando o gmodels
# install.packages("gmodels")
library(gmodels)

# Criando uma tabel acruzada dos dados previstos x dados atuais
CrossTable(x = dados_teste_labels, y = modelo, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
```

```
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                     | modelo
## dados_teste_labels |   Benigno |   Maligno | Row Total |
## -------------------|-----------|-----------|-----------|
##            Benigno |        61 |         0 |        61 |
##                    |     1.000 |     0.000 |     0.610 |
##                    |     0.968 |     0.000 |           |
##                    |     0.610 |     0.000 |           |
## -------------------|-----------|-----------|-----------|
##            Maligno |         2 |        37 |        39 |
##                    |     0.051 |     0.949 |     0.390 |
##                    |     0.032 |     1.000 |           |
##                    |     0.020 |     0.370 |           |
## -------------------|-----------|-----------|-----------|
##       Column Total |        63 |        37 |       100 |
##                    |     0.630 |     0.370 |           |
## -------------------|-----------|-----------|-----------|
##
##
```

```
# Interpretando os Resultados
# A tabela cruzada mostra 4 possíveis valores, que representam os falso/verdadeiro positivo e negativo
# A primeira coluna lista os labels originais nos dados observados
# As duas colunas do modelo (Benigno e Maligno) do modelo, mostram os resultados da previsão
# Temos:
# Cenário 1: Célula Benigno (label) x Benigno (Modelo) - 61 casos - true negative
# Cenário 2: Célula Benigno (label) x Maligno (Modelo) - 00 casos - false positive
# Cenário 3: Célula Maligno (label) x Benigno (Modelo) - 02 casos - false negative (o modelo errou)
# Cenário 4: Célula Maligno (label) x Maligno (Modelo) - 37 casos - true positive

# Lendo a Confusion Matrix (Perspectva de ter ou não a doença):

# True Negative  = nosso modelo previu que a pessoa NÃO tinha a doença e os dados mostraram que realmen
# False Positive = nosso modelo previu que a pessoa tinha a doença e os dados mostraram que NÃO, a pess
# False Negative = nosso modelo previu que a pessoa NÃO tinha a doença e os dados mostraram que SIM, a
# True Positive = nosso modelo previu que a pessoa tinha a doença e os dados mostraram que SIM, a pesso

# Falso Positivo - Erro Tipo I
# Falso Negativo - Erro Tipo II

# Taxa de acerto do Modelo: 98% (acertou 98 em 100)
```

Etapa 5 - Otimização do Modelo

```
## Etapa 5: Otimizando a perfomance do modelo

# Usando a função scale() para padronizar o z-score
dados_z <- as.data.frame(scale(dados[-1]))

# Confirmando transformação realizada com sucesso
summary(dados_z$area_mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.4532 -0.6666 -0.2949  0.0000  0.3632  5.2459
```

```
# Criando novos datasets de treino e de teste
dados_treino <- dados_z[1:469, ]
dados_teste <- dados_z[470:569, ]

dados_treino_labels <- dados[ 1: 469, 1]
dados_teste_labels <- dados[ 470: 569, 1]

# Reclassificando
modelo_v2 <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k = 21)

# Criando a cross table para comparar dados previstos com os dados reais
CrossTable(x = dados_teste_labels, y = modelo_v2, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                     | modelo_v2
## dados_teste_labels |   Benigno |   Maligno | Row Total |
## -------------------|-----------|-----------|-----------|
##            Benigno |        61 |         0 |        61 |
##                    |     1.000 |     0.000 |     0.610 |
##                    |     0.924 |     0.000 |           |
##                    |     0.610 |     0.000 |           |
## -------------------|-----------|-----------|-----------|
##            Maligno |         5 |        34 |        39 |
##                    |     0.128 |     0.872 |     0.390 |
##                    |     0.076 |     1.000 |           |
##                    |     0.050 |     0.340 |           |
## -------------------|-----------|-----------|-----------|
##       Column Total |        66 |        34 |       100 |
```

```
##                    |     0.660 |     0.340 |           |
## -------------------|-----------|-----------|-----------|
##
##
```

```
# Testando diferentes valores para k
# Criando dados de treino e dados de teste
# dados_treino <- dados_norm[1:469, ]
# dados_teste <- dados_norm[470:569, ]

# Criando os labels para os dados de treino e de teste
# dados_treino_labels <- dados[1:469, 1]
# dados_teste_labels <- dados[470:569, 1]

# dados_test_pred <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k=1)
# CrossTable(x = dados_teste_labels, y = dados_test_pred, prop.chisq=FALSE)

# dados_test_pred <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k=5)
# CrossTable(x = dados_teste_labels, y = dados_test_pred, prop.chisq=FALSE)

# dados_test_pred <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k=11)
# CrossTable(x = dados_teste_labels, y = dados_test_pred, prop.chisq=FALSE)

# dados_test_pred <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k=15)
# CrossTable(x = dados_teste_labels, y = dados_test_pred, prop.chisq=FALSE)

# dados_test_pred <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k=21)
# CrossTable(x = dados_teste_labels, y = dados_test_pred, prop.chisq=FALSE)

# dados_test_pred <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k=27)
# CrossTable(x = dados_teste_labels, y = dados_test_pred, prop.chisq=FALSE)
```

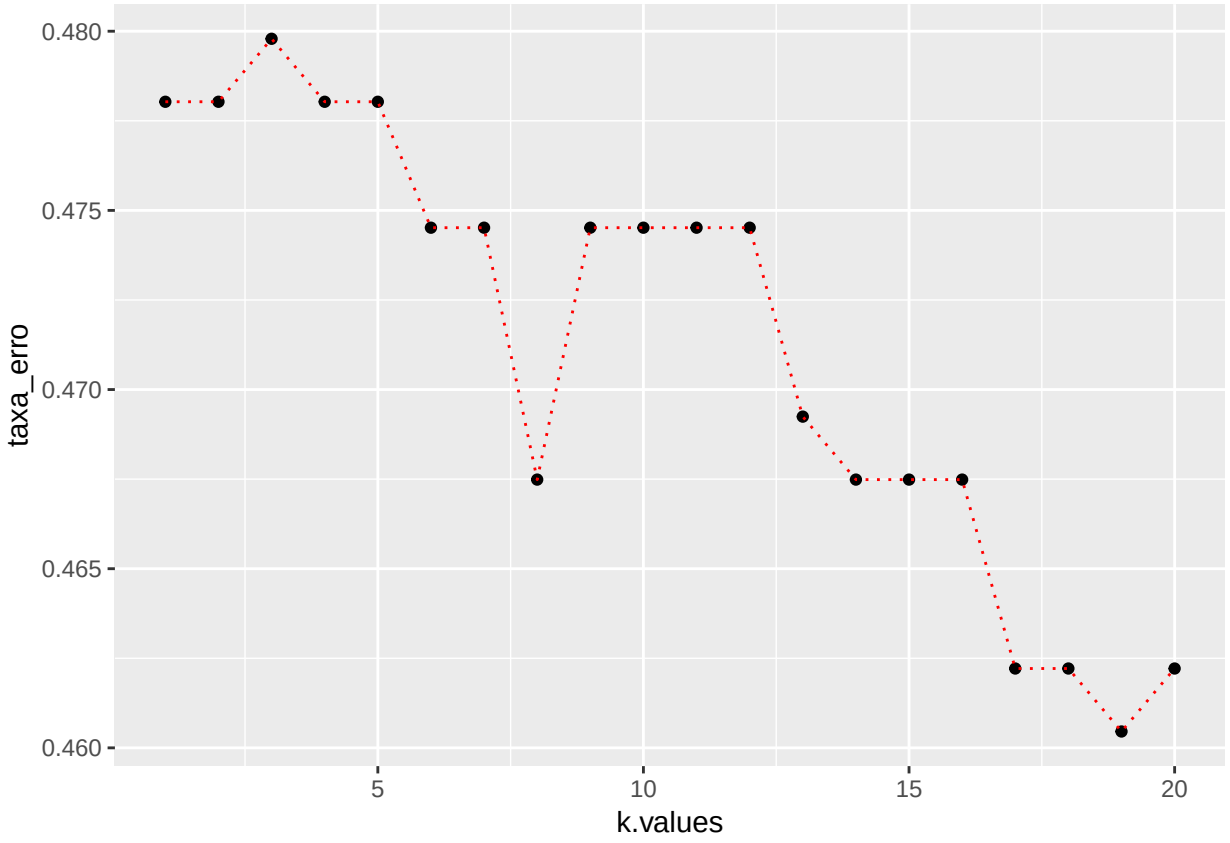
Etapa 6 - Calculando a Taxa de Erro

```
## Calculando a taxa de erro
prev = NULL
taxa_erro = NULL

suppressWarnings(
for(i in 1:20){
  set.seed(101)
  prev = knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k = i)
  taxa_erro[i] = mean(dados$diagnosis != prev)
})

# Obtendo os valores de k e das taxas de erro
library(ggplot2)
k.values <- 1:20
df_erro <- data.frame(taxa_erro, k.values)
df_erro
```

```
##    taxa_erro k.values
## 1  0.4780316        1
```

```
## 2  0.4780316        2
## 3  0.4797891        3
## 4  0.4780316        4
## 5  0.4780316        5
## 6  0.4745167        6
## 7  0.4745167        7
## 8  0.4674868        8
## 9  0.4745167        9
## 10 0.4745167       10
## 11 0.4745167       11
## 12 0.4745167       12
## 13 0.4692443       13
## 14 0.4674868       14
## 15 0.4674868       15
## 16 0.4674868       16
## 17 0.4622144       17
## 18 0.4622144       18
## 19 0.4604569       19
## 20 0.4622144       20
```

```
# À medida que aumentamos k, diminuímos a taxa de erro do modelo
ggplot(df_erro, aes(x = k.values, y = taxa_erro)) + geom_point()+ geom_line(lty = "dotted", color = 'red
```



Fim

www.datascienceacademy.com.br