KNN_Cancer_de_Mama

Alyster Fernandes

7/2/2021

Introdução

Sobre a base de dados

A base de dados utilizar nesse projeto foi disponibilizada no repositório de machine learning da Universidade da Califórnia, Irvine (UCI). O dataset apresentado traz dados reais obtidos em testes em células provenientes de biópsias de nódulos mamários com anomalias. A utilização dessa base é interessante, pois representa a significância que o aprendizado de máquina e a estatística podem ter em aplicações na área médica.

Mais detalhes sobre a base de dados e também sobre as variáveis, podem ser encontrados nesse LINK

Pacotes e arquivos

Ativação dos pacotes

```
#pacote com funções para classificação
library(class)
library(gmodels)

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

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.0.5
```

Leitura do arquivo

```
dados <-
read.csv('https://raw.githubusercontent.com/AlysterF/KNN_cancer_de_mama/main/database/bc_data.csv')
```

Análise Exploratória

Quais dados existem nessa base?

É importante conhecer os dados com os quais estamos trabalhando. Então essa é a hora de perguntar: que informações existem aqui? Como esses dados estão distribuídos? Há um equilíbrio nos dados em relação a variável target? A exploração

de dados é o passo inicial para que possamos deixar tudo pronto para que o modelo de machine learning possa nos ajudar a prever respostas. Se os dados estão limpos e tratados, teremos um modelo mais assertivo.

Com o comando abaixo, é possível visualizar as primeiras seis linhas do dataframe, o que auxilia a compreender como os dados estão apresentados.

```
#Visualização das primeiras 6 linhas do dataframe
head(dados)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1 87139402         B       12.32        12.39          78.85     464.1
## 2  8910251         B       10.60        18.95          69.28     346.4
## 3   905520         B       11.04        16.83          70.92     373.2
## 4   868871         B       11.28        13.39          73.00     384.8
## 5  9012568         B       15.19        13.21          97.65     711.8
## 6   906539         B       11.57        19.04          74.20     409.7
##   smoothness_mean compactness_mean concavity_mean points_mean
symmetry_mean
## 1         0.10280          0.06981        0.03987     0.03700
0.1959
## 2         0.09688          0.11470        0.06387     0.02642
0.1922
## 3         0.10770          0.07804        0.03046     0.02480
0.1714
## 4         0.11640          0.11360        0.04635     0.04796
0.1771
## 5         0.07963          0.06934        0.03393     0.02657
0.1721
## 6         0.08546          0.07722        0.05485     0.01428
0.2031
##   dimension_mean radius_se texture_se perimeter_se area_se
smoothness_se
## 1        0.05955    0.2360     0.6656        1.670   17.43
0.008045
## 2        0.06491    0.4505     1.1970        3.430   27.10
0.007470
## 3        0.06340    0.1967     1.3870        1.342   13.54
0.005158
## 4        0.06072    0.3384     1.3430        1.851   26.33
0.011270
## 5        0.05544    0.1783     0.4125        1.338   17.72
0.005012
## 6        0.06267    0.2864     1.4400        2.206   20.30
0.007278
##   compactness_se concavity_se points_se symmetry_se dimension_se
radius_worst
## 1       0.011800      0.01683  0.012410     0.01924     0.002248
13.50
## 2       0.035810      0.03354  0.013650     0.03504     0.003318
```

```
11.88
## 3       0.009355      0.01056  0.007483     0.01718     0.002198
12.41
## 4       0.034980      0.02187  0.019650     0.01580     0.003442
11.92
## 5       0.014850      0.01551  0.009155     0.01647     0.001767
16.20
## 6       0.020470      0.04447  0.008799     0.01868     0.003339
13.07
##   texture_worst perimeter_worst area_worst smoothness_worst
compactness_worst
## 1         15.64           86.97      549.1           0.1385
0.1266
## 2         22.94           78.28      424.8           0.1213
0.2515
## 3         26.44           79.93      471.4           0.1369
0.1482
## 4         15.77           76.53      434.0           0.1367
0.1822
## 5         15.73          104.50      819.1           0.1126
0.1737
## 6         26.98           86.43      520.5           0.1249
0.1937
##   concavity_worst points_worst symmetry_worst dimension_worst
## 1         0.12420      0.09391         0.2827         0.06771
## 2         0.19160      0.07926         0.2940         0.07587
## 3         0.10670      0.07431         0.2998         0.07881
## 4         0.08669      0.08611         0.2102         0.06784
## 5         0.13620      0.08178         0.2487         0.06766
## 6         0.25600      0.06664         0.3035         0.08284
```

Já o comando str nos apresenta os formatos em que os dados estão no dataframe, assim, é possível identificar quais variáveis são texto, números e etc.

```
#Descrição do formato das variáveis
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 ...
```

A grande maioria dos dados está em valor numérico, com exeção da variável diagnosis que é char/caractere. Os dados nesse formato estão conforme o esperado.

O comando summary exibe um resumo estatístico de todas as variáveis do dataframe.

```
#Resumo estatístico das variáveis
summary(dados)
```

```
##        id             diagnosis          radius_mean      texture_mean
##  Min.   :     8670   Length:569         Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   Class :character   1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024   Mode  :character   Median :13.370   Median :18.84
##  Mean   : 30371831                      Mean   :14.127   Mean   :19.29
##  3rd Qu.:  8813129                      3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502                      Max.   :28.110   Max.   :39.28
##  perimeter_mean     area_mean      smoothness_mean   compactness_mean
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492
##  Median : 86.24   Median : 551.1   Median :0.09587   Median :0.09263
##  Mean   : 91.97   Mean   : 654.9   Mean   :0.09636   Mean   :0.10434
##  3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity_mean     points_mean      symmetry_mean    dimension_mean
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.1060   Min.   :0.04996
##  1st Qu.:0.02956   1st Qu.:0.02031   1st Qu.:0.1619   1st Qu.:0.05770
##  Median :0.06154   Median :0.03350   Median :0.1792   Median :0.06154
##  Mean   :0.08880   Mean   :0.04892   Mean   :0.1812   Mean   :0.06280
##  3rd Qu.:0.13070   3rd Qu.:0.07400   3rd Qu.:0.1957   3rd Qu.:0.06612
##  Max.   :0.42680   Max.   :0.20120   Max.   :0.3040   Max.   :0.09744
##    radius_se        texture_se      perimeter_se       area_se
```

```
##  Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
##  1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
##  Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
##  Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
##  3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
##  Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
##  smoothness_se      compactness_se      concavity_se       points_se
##  Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.
:0.000000
##  1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st
Qu.:0.007638
##  Median :0.006380   Median :0.020450   Median :0.02589   Median
:0.010930
##  Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean
:0.011796
##  3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd
Qu.:0.014710
##  Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.
:0.052790
##   symmetry_se        dimension_se        radius_worst   texture_worst
##  Min.   :0.007882   Min.   :0.0008948   Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015160   1st Qu.:0.0022480   1st Qu.:13.01   1st Qu.:21.08
##  Median :0.018730   Median :0.0031870   Median :14.97   Median :25.41
##  Mean   :0.020542   Mean   :0.0037949   Mean   :16.27   Mean   :25.68
##  3rd Qu.:0.023480   3rd Qu.:0.0045580   3rd Qu.:18.79   3rd Qu.:29.72
##  Max.   :0.078950   Max.   :0.0298400   Max.   :36.04   Max.   :49.54
##  perimeter_worst    area_worst     smoothness_worst  compactness_worst
##  Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729
##  1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720
##  Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190
##  Mean   :107.26   Mean   : 880.6   Mean   :0.13237   Mean   :0.25427
##  3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910
##  Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800
##  concavity_worst   points_worst     symmetry_worst   dimension_worst
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.1145   1st Qu.:0.06493   1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2267   Median :0.09993   Median :0.2822   Median :0.08004
##  Mean   :0.2722   Mean   :0.11461   Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3829   3rd Qu.:0.16140   3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :1.2520   Max.   :0.29100   Max.   :0.6638   Max.   :0.20750
```

Correções do dataframe

Perceba que nos resultados apresentados anteriormente, a variável id não parece fazer muito sentido para nossas análises estatísticas, então o ideal é que a variável id seja removida.

```
#Remoção da primeira coluna do dataframe.
dados <- dados[-1]
head(dados)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean
smoothness_mean
## 1         B       12.32        12.39          78.85     464.1
0.10280
## 2         B       10.60        18.95          69.28     346.4
0.09688
## 3         B       11.04        16.83          70.92     373.2
0.10770
## 4         B       11.28        13.39          73.00     384.8
0.11640
## 5         B       15.19        13.21          97.65     711.8
0.07963
## 6         B       11.57        19.04          74.20     409.7
0.08546
##   compactness_mean concavity_mean points_mean symmetry_mean
dimension_mean
## 1          0.06981        0.03987     0.03700        0.1959
0.05955
## 2          0.11470        0.06387     0.02642        0.1922
0.06491
## 3          0.07804        0.03046     0.02480        0.1714
0.06340
## 4          0.11360        0.04635     0.04796        0.1771
0.06072
## 5          0.06934        0.03393     0.02657        0.1721
0.05544
## 6          0.07722        0.05485     0.01428        0.2031
0.06267
##   radius_se texture_se perimeter_se area_se smoothness_se
compactness_se
## 1    0.2360     0.6656        1.670   17.43      0.008045
0.011800
## 2    0.4505     1.1970        3.430   27.10      0.007470
0.035810
## 3    0.1967     1.3870        1.342   13.54      0.005158
0.009355
## 4    0.3384     1.3430        1.851   26.33      0.011270
0.034980
## 5    0.1783     0.4125        1.338   17.72      0.005012
0.014850
## 6    0.2864     1.4400        2.206   20.30      0.007278
0.020470
##   concavity_se points_se symmetry_se dimension_se radius_worst
texture_worst
## 1      0.01683  0.012410     0.01924     0.002248        13.50
15.64
## 2      0.03354  0.013650     0.03504     0.003318        11.88
22.94
## 3      0.01056  0.007483     0.01718     0.002198        12.41
26.44
```

```
## 4      0.02187  0.019650     0.01580     0.003442        11.92
15.77
## 5      0.01551  0.009155     0.01647     0.001767        16.20
15.73
## 6      0.04447  0.008799     0.01868     0.003339        13.07
26.98
##   perimeter_worst area_worst smoothness_worst compactness_worst
concavity_worst
## 1           86.97      549.1           0.1385            0.1266
0.12420
## 2           78.28      424.8           0.1213            0.2515
0.19160
## 3           79.93      471.4           0.1369            0.1482
0.10670
## 4           76.53      434.0           0.1367            0.1822
0.08669
## 5          104.50      819.1           0.1126            0.1737
0.13620
## 6           86.43      520.5           0.1249            0.1937
0.25600
##   points_worst symmetry_worst dimension_worst
## 1      0.09391         0.2827         0.06771
## 2      0.07926         0.2940         0.07587
## 3      0.07431         0.2998         0.07881
## 4      0.08611         0.2102         0.06784
## 5      0.08178         0.2487         0.06766
## 6      0.06664         0.3035         0.08284
```

A variável diagnosis, apesar de ter sido apresentada corretamente, seria mais interessante se fossem apresentadas as quantidades de cada valor possível dentro dessa variável.

```
#proporção de valores da variável diagnosis
table(dados$diagnosis)

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

Existem 2 valores diferentes na coluna diagnosis. É possível categorizar esses dados, para que quando as análises estatísticas sejam feitas, os dados sejam considerados como tendo 2 níveis (B e M). Isso também auxiliará no funcionamento dos algoritmos de classificação.

```
#convertendo de char para factor

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 ...
```

Se fizermos novamente a descrição estatística dos dados, será possível ver que a variável diagnosis terá resultados mais interessantes.

```
summary(dados$diagnosis)
```

```
## Benigno Maligno
##     357     212
```

Nas descrições estatísticas, é possível ver que os valores estão em escalas completamentes diferentes, alguns iniciam-se em 0, outros em 185. Essa falta de normalização pode afetar o desempenho do algoritmo de classificação, fazendo com que algumas variáveis acabem recebendo mais importância do que outras nos cálculos. Pode-se corrigir esse problema através da normalização, que consiste em aplicar o seguinte cálculo ao valor x: (x - min(x)) / (max(x) - min(x)), esse cálculo retornará todos os valores em uma escala de 0 a 1.

```
#Criação da função de normalização
f_normalizacao <- function(x){
  return((x - min(x))/(max(x)-min(x)))
}
```

Após a criação da função, é necessário aplicar a função aos dados do dataframe.

```
#Aplicação da normalização em todos os dados numéricos do dataframe.
dados_norm <- as.data.frame(lapply(dados[2:31], f_normalizacao))
summary(dados_norm)
```

```
##   radius_mean      texture_mean    perimeter_mean     area_mean
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2233   1st Qu.:0.2185   1st Qu.:0.2168   1st Qu.:0.1174
##  Median :0.3024   Median :0.3088   Median :0.2933   Median :0.1729
##  Mean   :0.3382   Mean   :0.3240   Mean   :0.3329   Mean   :0.2169
##  3rd Qu.:0.4164   3rd Qu.:0.4089   3rd Qu.:0.4168   3rd Qu.:0.2711
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  smoothness_mean  compactness_mean concavity_mean     points_mean
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.3046   1st Qu.:0.1397   1st Qu.:0.06926   1st Qu.:0.1009
##  Median :0.3904   Median :0.2247   Median :0.14419   Median :0.1665
##  Mean   :0.3948   Mean   :0.2606   Mean   :0.20806   Mean   :0.2431
##  3rd Qu.:0.4755   3rd Qu.:0.3405   3rd Qu.:0.30623   3rd Qu.:0.3678
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##  symmetry_mean    dimension_mean     radius_se         texture_se
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.2823   1st Qu.:0.1630   1st Qu.:0.04378   1st Qu.:0.1047
##  Median :0.3697   Median :0.2439   Median :0.07702   Median :0.1653
##  Mean   :0.3796   Mean   :0.2704   Mean   :0.10635   Mean   :0.1893
##  3rd Qu.:0.4530   3rd Qu.:0.3404   3rd Qu.:0.13304   3rd Qu.:0.2462
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##   perimeter_se        area_se         smoothness_se    compactness_se
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.04000   1st Qu.:0.02064   1st Qu.:0.1175   1st Qu.:0.08132
```

```
##  Median :0.07209   Median :0.03311   Median :0.1586   Median :0.13667
##  Mean   :0.09938   Mean   :0.06264   Mean   :0.1811   Mean   :0.17444
##  3rd Qu.:0.12251   3rd Qu.:0.07170   3rd Qu.:0.2187   3rd Qu.:0.22680
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
##   concavity_se       points_se       symmetry_se      dimension_se
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.03811   1st Qu.:0.1447   1st Qu.:0.1024   1st Qu.:0.04675
##  Median :0.06538   Median :0.2070   Median :0.1526   Median :0.07919
##  Mean   :0.08054   Mean   :0.2235   Mean   :0.1781   Mean   :0.10019
##  3rd Qu.:0.10619   3rd Qu.:0.2787   3rd Qu.:0.2195   3rd Qu.:0.12656
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##   radius_worst    texture_worst    perimeter_worst    area_worst
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.1807   1st Qu.:0.2415   1st Qu.:0.1678   1st Qu.:0.08113
##  Median :0.2504   Median :0.3569   Median :0.2353   Median :0.12321
##  Mean   :0.2967   Mean   :0.3640   Mean   :0.2831   Mean   :0.17091
##  3rd Qu.:0.3863   3rd Qu.:0.4717   3rd Qu.:0.3735   3rd Qu.:0.22090
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##  smoothness_worst compactness_worst concavity_worst    points_worst
##  Min.   :0.0000   Min.   :0.0000    Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.3000   1st Qu.:0.1163    1st Qu.:0.09145   1st Qu.:0.2231
##  Median :0.3971   Median :0.1791    Median :0.18107   Median :0.3434
##  Mean   :0.4041   Mean   :0.2202    Mean   :0.21740   Mean   :0.3938
##  3rd Qu.:0.4942   3rd Qu.:0.3025    3rd Qu.:0.30583   3rd Qu.:0.5546
##  Max.   :1.0000   Max.   :1.0000    Max.   :1.00000   Max.   :1.0000
##  symmetry_worst   dimension_worst
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1851   1st Qu.:0.1077
##  Median :0.2478   Median :0.1640
##  Mean   :0.2633   Mean   :0.1896
##  3rd Qu.:0.3182   3rd Qu.:0.2429
##  Max.   :1.0000   Max.   :1.0000
```

Também é importante verificar se existem dados nulos, para que os mesmos sejam tratados.

```
anyNA(dados_norm)
```

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

Machine Learning

Após os tratamentos necessários, já é possível aplicar os dados ao algoritmo de machine learning.

Treino do modelo

Para que se treine o modelo, é necessária a separação dos dados em amostras de treino e teste, sendo a de treino para efetivamente treinar o modelo e a de teste para

validar se o modelo responde bem para dados que nunca tenham sido apresentados antes. Essa divisão de dados deve ser feita de forma aleatória, para que não haja viés no treino. Utilizarei uma proporção de 70% para treino e 30% para teste.

```
#Split dos dados de treino e teste
set.seed(2021)
dt = sort(sample(nrow(dados_norm), nrow(dados_norm) * 0.7))
treino <- dados_norm[dt,]
teste <- dados_norm[-dt,]
```

Também é necessário, no algoritmo K-Nearest Neighbors, apresentar variáveis de labels, para que os resultados previstos sejam comparados aos resultados reais.

```
#Labels para comparação
treino_labels <- dados[dt, 1]
teste_labels <- dados[-dt, 1]
```

Tendo todas essas variáveis preparadas, já é possível treinar o modelo KNN. No modelo KNN, é necessário apresentar um número de parâmetro K. Esse parâmetro é a quantidade de menores distâncias que o algoritmo irá utilizar para comparar e decidir em qual classe os dados se encontram. Utilizarei 20, mas explorarei outras opções em sequência.

```
#Treino do modelo knn
modelo <- knn(train = treino, test = teste, cl = treino_labels, k=20)
```

Avaliação de performance do modelo

Após o treino do modelo, é importante avaliar qual foi a performance do modelo e para isso utilizarei uma matriz de confusão para verificar se os valores previstos estão de acordo com os valores reais.

```
#matriz de confusão
tab <- table(modelo, teste_labels)
tab
```

```
##          teste_labels
## modelo    Benigno Maligno
##   Benigno     104      10
##   Maligno       0      57
```

No nosso caso, o modelo, quando a classificação era benigna, acertou 100%. Já na previsão dos casos em que o câncer era maligno, o modelo acabou prevendo 10 erroneamente, pois foram classificados como benignos, quando deveriam ser malignos.

Para testar a acurácia do modelo, o cálculo é bem simples, pois trata-se da soma dos acertos dividida pelo total de dados.

```
#Acurácia do modelo
acuracia <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
acuracia(tab)

## [1] 94.15205
```

O modelo obteve uma acurácia de 94%. É uma excelente acurácia, e esse valor sempre depende do seu objetivo. A acurácia de um modelo deve estar dentro do range pré-definido pelo cientista de dados.

Otimização do modelo

É possível que o modelo seja otimizado substituindo a normalização dos dados por uma padronização, ou até mesmo testando outros valores para o parâmetro K. Testarei o desempenho do algoritmo caso os dados fossem padronizados. Os dados normalizados, como vimos anteriormente, criam uma escala de 0 a 1. Já a padronização altera a média dos dados para 0 com desvio padrão 1.

```
#padronização do z score
dados_padr <- as.data.frame(scale(dados[-1]))
summary(dados_padr)

##   radius_mean       texture_mean     perimeter_mean      area_mean
##  Min.   :-2.0279   Min.   :-2.2273   Min.   :-1.9828   Min.   :-1.4532
##  1st Qu.:-0.6888   1st Qu.:-0.7253   1st Qu.:-0.6913   1st Qu.:-0.6666
##  Median :-0.2149   Median :-0.1045   Median :-0.2358   Median :-0.2949
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4690   3rd Qu.: 0.5837   3rd Qu.: 0.4992   3rd Qu.: 0.3632
##  Max.   : 3.9678   Max.   : 4.6478   Max.   : 3.9726   Max.   : 5.2459
##  smoothness_mean    compactness_mean  concavity_mean     points_mean
##  Min.   :-3.10935   Min.   :-1.6087   Min.   :-1.1139   Min.   :-
1.2607
##  1st Qu.:-0.71034   1st Qu.:-0.7464   1st Qu.:-0.7431   1st Qu.:-
0.7373
##  Median :-0.03486   Median :-0.2217   Median :-0.3419   Median :-
0.3974
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   :
0.0000
##  3rd Qu.: 0.63564   3rd Qu.: 0.4934   3rd Qu.: 0.5256   3rd Qu.:
0.6464
##  Max.   : 4.76672   Max.   : 4.5644   Max.   : 4.2399   Max.   :
3.9245
##  symmetry_mean      dimension_mean      radius_se         texture_se
##  Min.   :-2.74171   Min.   :-1.8183   Min.   :-1.0590   Min.   :-
1.5529
##  1st Qu.:-0.70262   1st Qu.:-0.7220   1st Qu.:-0.6230   1st Qu.:-
0.6942
##  Median :-0.07156   Median :-0.1781   Median :-0.2920   Median :-
0.1973
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   :
0.0000
```

```
##  3rd Qu.: 0.53031   3rd Qu.: 0.4706   3rd Qu.: 0.2659   3rd Qu.:
0.4661
##  Max.   : 4.48081   Max.   : 4.9066   Max.   : 8.8991   Max.   :
6.6494
##   perimeter_se        area_se         smoothness_se     compactness_se
##  Min.   :-1.0431   Min.   :-0.7372   Min.   :-1.7745   Min.   :-1.2970
##  1st Qu.:-0.6232   1st Qu.:-0.4943   1st Qu.:-0.6235   1st Qu.:-0.6923
##  Median :-0.2864   Median :-0.3475   Median :-0.2201   Median :-0.2808
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.2428   3rd Qu.: 0.1067   3rd Qu.: 0.3680   3rd Qu.: 0.3893
##  Max.   : 9.4537   Max.   :11.0321   Max.   : 8.0229   Max.   : 6.1381
##   concavity_se       points_se         symmetry_se       dimension_se
##  Min.   :-1.0566   Min.   :-1.9118   Min.   :-1.5315   Min.   :-1.0960
##  1st Qu.:-0.5567   1st Qu.:-0.6739   1st Qu.:-0.6511   1st Qu.:-0.5846
##  Median :-0.1989   Median :-0.1404   Median :-0.2192   Median :-0.2297
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.3365   3rd Qu.: 0.4722   3rd Qu.: 0.3554   3rd Qu.: 0.2884
##  Max.   :12.0621   Max.   : 6.6438   Max.   : 7.0657   Max.   : 9.8429
##   radius_worst     texture_worst      perimeter_worst     area_worst
##  Min.   :-1.7254   Min.   :-2.22204   Min.   :-1.6919   Min.   :-
1.2213
##  1st Qu.:-0.6743   1st Qu.:-0.74797   1st Qu.:-0.6890   1st Qu.:-
0.6416
##  Median :-0.2688   Median :-0.04348   Median :-0.2857   Median :-
0.3409
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   :
0.0000
##  3rd Qu.: 0.5216   3rd Qu.: 0.65776   3rd Qu.: 0.5398   3rd Qu.:
0.3573
##  Max.   : 4.0906   Max.   : 3.88249   Max.   : 4.2836   Max.   :
5.9250
##  smoothness_worst  compactness_worst concavity_worst    points_worst
##  Min.   :-2.6803   Min.   :-1.4426   Min.   :-1.3047   Min.   :-1.7435
##  1st Qu.:-0.6906   1st Qu.:-0.6805   1st Qu.:-0.7558   1st Qu.:-0.7557
##  Median :-0.0468   Median :-0.2693   Median :-0.2180   Median :-0.2233
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5970   3rd Qu.: 0.5392   3rd Qu.: 0.5307   3rd Qu.: 0.7119
##  Max.   : 3.9519   Max.   : 5.1084   Max.   : 4.6965   Max.   : 2.6835
##  symmetry_worst    dimension_worst
##  Min.   :-2.1591   Min.   :-1.6004
##  1st Qu.:-0.6413   1st Qu.:-0.6913
##  Median :-0.1273   Median :-0.2163
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4497   3rd Qu.: 0.4504
##  Max.   : 6.0407   Max.   : 6.8408

str(dados_padr)

## 'data.frame':    569 obs. of  30 variables:
##  $ radius_mean      : num  -0.513 -1.001 -0.876 -0.808 0.302 ...
```

```
##  $ texture_mean     : num  -1.604 -0.079 -0.572 -1.372 -1.414 ...
##  $ perimeter_mean   : num  -0.54 -0.934 -0.866 -0.781 0.234 ...
##  $ area_mean        : num  -0.542 -0.877 -0.8 -0.767 0.162 ...
##  $ smoothness_mean  : num  0.458 0.037 0.806 1.425 -1.19 ...
##  $ compactness_mean : num  -0.654 0.196 -0.498 0.175 -0.663 ...
##  $ concavity_mean   : num  -0.614 -0.313 -0.732 -0.532 -0.688 ...
##  $ points_mean      : num  -0.3072 -0.5798 -0.6216 -0.0247 -0.576 ...
##  $ symmetry_mean    : num  0.538 0.403 -0.356 -0.148 -0.331 ...
##  $ dimension_mean   : num  -0.46 0.2992 0.0853 -0.2943 -1.0421 ...
##  $ radius_se        : num  -0.61 0.163 -0.752 -0.241 -0.818 ...
##  $ texture_se       : num  -0.999 -0.036 0.308 0.229 -1.458 ...
##  $ perimeter_se     : num  -0.592 0.279 -0.754 -0.502 -0.756 ...
##  $ area_se          : num  -0.504 -0.291 -0.589 -0.308 -0.497 ...
##  $ smoothness_se    : num  0.334 0.143 -0.627 1.408 -0.676 ...
##  $ compactness_se   : num  -0.764 0.577 -0.9 0.531 -0.593 ...
##  $ concavity_se     : num  -0.499 0.0545 -0.7067 -0.3321 -0.5428 ...
##  $ points_se        : num  0.0995 0.3005 -0.699 1.2729 -0.428 ...
##  $ symmetry_se      : num  -0.158 1.754 -0.407 -0.574 -0.493 ...
##  $ dimension_se     : num  -0.585 -0.18 -0.604 -0.133 -0.766 ...
##  $ radius_worst     : num  -0.5729 -0.9081 -0.7985 -0.8998 -0.0143 ...
##  $ texture_worst    : num  -1.633 -0.445 0.124 -1.612 -1.618 ...
##  $ perimeter_worst  : num  -0.6039 -0.8625 -0.8134 -0.9146 -0.0822 ...
##  $ area_worst       : num  -0.582 -0.801 -0.719 -0.784 -0.108 ...
##  $ smoothness_worst : num  0.269 -0.485 0.198 0.19 -0.866 ...
##  $ compactness_worst: num  -0.8114 -0.0176 -0.6741 -0.458 -0.5121 ...
##  $ concavity_worst  : num  -0.709 -0.386 -0.793 -0.889 -0.652 ...
##  $ points_worst     : num  -0.315 -0.538 -0.613 -0.434 -0.499 ...
##  $ symmetry_worst   : num  -0.1192 0.0634 0.1572 -1.2911 -0.6688 ...
##  $ dimension_worst  : num  -0.899 -0.447 -0.284 -0.892 -0.902 ...
```

O processo de separação dos dados de treino e teste devem ser repetidos para que o novo modelo seja criado e avaliado.

```
#separação dos dados de treino e teste
dt_padr <- sort(sample(nrow(dados_padr), nrow(dados_padr)*0.7))
treino_padr <- dados_padr[dt_padr,]
teste_padr <- dados_padr[-dt_padr,]

#Labels para comparação
treino_labels2 <- dados[dt_padr, 1]
teste_labels2 <- dados[-dt_padr, 1]

#treino do modelo com os novos dados
modelo2 <- knn(treino_padr, teste_padr, treino_labels2, k = 20)

tab2 <- table(modelo, teste_labels2)
tab2

##          teste_labels2
## modelo    Benigno Maligno
```

```
##   Benigno      71      43
##   Maligno      33      24
```

```
#Acurácia do modelo
acuracia(tab2)
```

```
## [1] 55.55556
```

Perceba que a acurácia do novo modelo está muito inferior ao modelo que utiliza os dados padronizados, portanto, utilizarei o modelo inicial para variar o parâmetro K e observar os resultados das taxas de erro.

Taxas de erro em função da variação do parâmetro K

Utilizarei a taxa de erro médio para apresentar na avaliação abaixo. Irei considerar também a variação de K de 1 a 25.

```
### Calculo da taxa de erro para cada K
prev = NULL
taxa_erro = NULL
suppressWarnings(
  for(i in 1:25){
    set.seed(2021)
    prev = knn(train = treino, test = teste, cl = treino_labels, k = i)
    taxa_erro[i] = mean(dados$diagnosis != prev)
  }
)
```

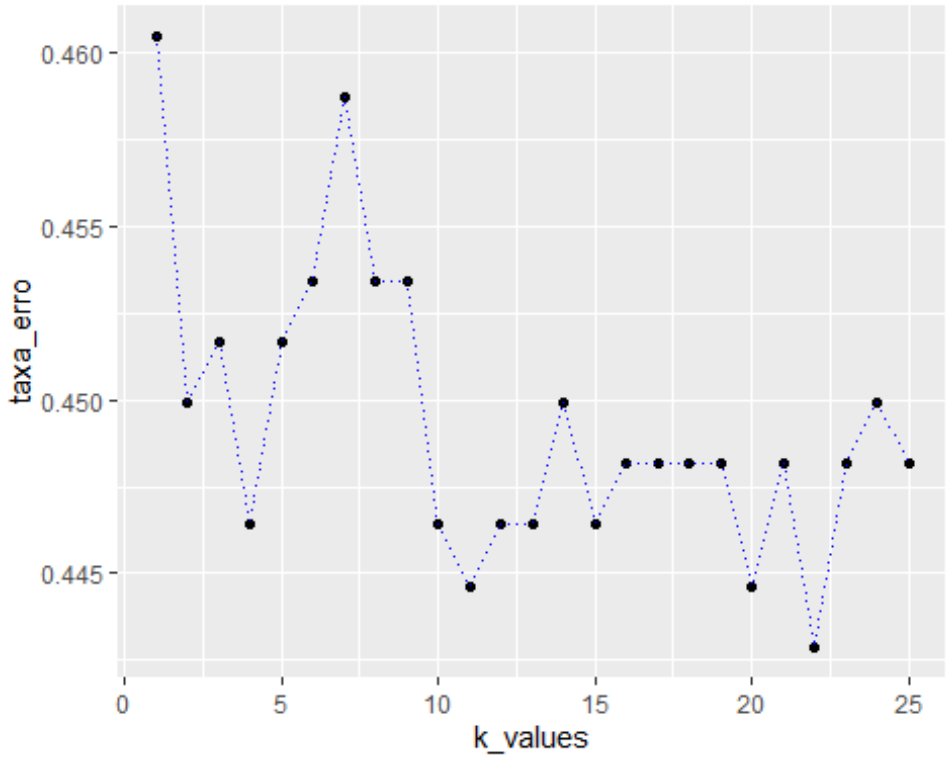
```
#criacao do dataframe com taxas de erro em funcao dos valores de k
k_values <- 1:25
df_erro <- data.frame(taxa_erro, k_values)
df_erro
```

```
##    taxa_erro k_values
## 1  0.4604569        1
## 2  0.4499121        2
## 3  0.4516696        3
## 4  0.4463972        4
## 5  0.4516696        5
## 6  0.4534271        6
## 7  0.4586995        7
## 8  0.4534271        8
## 9  0.4534271        9
## 10 0.4463972       10
## 11 0.4446397       11
## 12 0.4463972       12
## 13 0.4463972       13
## 14 0.4499121       14
## 15 0.4463972       15
## 16 0.4481547       16
## 17 0.4481547       17
```

```
## 18 0.4481547       18
## 19 0.4481547       19
## 20 0.4446397       20
## 21 0.4481547       21
## 22 0.4428822       22
## 23 0.4481547       23
## 24 0.4499121       24
## 25 0.4481547       25
```

Com todos os dados armazenados, o ideal é apresentá-los em um gráfico de linhas para melhor entendimento.

```
ggplot(df_erro, aes(x = k_values, y = taxa_erro)) +
  geom_point() +
  geom_line(lty = "dotted", color = 'blue')
```



Através do gráfico acima, é possível ver que conforme o número de K aumenta, menor a taxa de erro. E claramente é possível identificar que a menor taxa de erro do modelo ocorre quando o valor de K é 22.

Conclusão

Comentários

Decidi realizar esse projeto de análise para praticar a utilização de dados no algoritmo KNN. Apesar dos dados não precisarem de muito tratamento, esse foi o meu primeiro

modelo de machine learning de classificação em que eu procurei realizar o processo sozinho do início ao fim após alguns estudos. Estou bem orgulhoso do resultado obtido e dos conhecimentos adquiridos, é claro.

Contatos

Linkedin: Alyster Fernandes **GitHub:** AlysterF **Tableau:** alyster.fernandes