Classificação de tumores usando kNN

2. Bibliotecas utilizadas

More on `tidyverse` here.

More on `caret` here.

More on `class` here.

```
library(tidyverse)
library(caret)
library(class)
```

3. Importando os dados

You can also embed plots, for example:

```
data <- read.csv("wisc_bc_data.csv")[,-1]
data$diagnosis <- factor(data$diagnosis, levels = c("B","M"), ordered = T)
data[,2:ncol(data)] <- sapply(data[,2:ncol(data)], as.numeric)
row.names(data) <- as.numeric(row.names(data))
```

4. Análise exploratória dos dados

4.1 Verificando os tipos dos dados e ausência de valores

Verificando as primeiras linhas da tabela:

```
head(data)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1         M       17.99        10.38         122.80    1001.0
## 2         M       20.57        17.77         132.90    1326.0
## 3         M       19.69        21.25         130.00    1203.0
## 4         M       11.42        20.38          77.58     386.1
## 5         M       20.29        14.34         135.10    1297.0
## 6         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
```

```
##    area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst
## 1     0.03003             0.006193        25.38         17.33
## 2     0.01389             0.003532        24.99         23.41
## 3     0.02250             0.004571        23.57         25.53
## 4     0.05963             0.009208        14.91         26.50
## 5     0.01756             0.005115        22.54         16.67
## 6     0.02165             0.005082        15.47         23.75
##   perimeter_worst area_worst smoothness_worst compactness_worst
## 1          184.60     2019.0           0.1622            0.6656
## 2          158.80     1956.0           0.1238            0.1866
## 3          152.50     1709.0           0.1444            0.4245
## 4           98.87      567.7           0.2098            0.8663
## 5          152.20     1575.0           0.1374            0.2050
## 6          103.40      741.6           0.1791            0.5249
##   concavity_worst concave.points_worst symmetry_worst
## 1          0.7119               0.2654         0.4601
## 2          0.2416               0.1860         0.2750
## 3          0.4504               0.2430         0.3613
## 4          0.6869               0.2575         0.6638
## 5          0.4000               0.1625         0.2364
## 6          0.5355               0.1741         0.3985
##   fractal_dimension_worst
## 1                 0.11890
## 2                 0.08902
## 3                 0.08758
## 4                 0.17300
## 5                 0.07678
## 6                 0.12440
```

Verificando a existência de *missing values*:

```
colSums(is.na(data))
```

```
##               diagnosis             radius_mean            texture_mean
##                       0                       0                       0
##          perimeter_mean               area_mean         smoothness_mean
##                       0                       0                       0
##        compactness_mean          concavity_mean     concave.points_mean
##                       0                       0                       0
##           symmetry_mean  fractal_dimension_mean               radius_se
##                       0                       0                       0
##              texture_se            perimeter_se                 area_se
##                       0                       0                       0
##           smoothness_se          compactness_se            concavity_se
##                       0                       0                       0
##       concave.points_se             symmetry_se    fractal_dimension_se
##                       0                       0                       0
##            radius_worst           texture_worst         perimeter_worst
```

```
##                       0                        0                        0
##              area_worst         smoothness_worst        compactness_worst
##                       0                        0                        0
##         concavity_worst    concave.points_worst           symmetry_worst
##                       0                        0                        0
## fractal_dimension_worst
##                       0
```

4.2 Verificando quantidade da variável dependente no dataset

Total de Benignos e Malignos:

```r
table(data$diagnosis)
```

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

Benignos e Malignos em %:

```r
round(prop.table(table(data$diagnosis))*100, digits = 2)
```

```
##
##     B     M
## 62.74 37.26
```

4.3 Normalizando as variáveis quantitativas

4.3.1 Normalização Min-Max

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

```r
norm.minmax <- function(x) {
  return((x - min(x))/(max(x) - min(x)))
}
```

4.3.2 Normalização Z-score

$$x_{\text{norm}} = \frac{x - \text{mean}(x)}{\text{sd}(x)}$$

```r
norm.zscore <- function(x) {
  return((x - mean(x))/sd(x))
}
```

4.3.3 Normalizando os dados

```r
data_norm1 <- as.data.frame(lapply(data[,2:ncol(data)], norm.minmax))
data_norm2 <- as.data.frame(lapply(data[,2:ncol(data)], norm.zscore))
```

5. Construindo o modelo de Classificação k-NN

5.1 Criando dataset de treino e teste

```
test.size <- 0.20
split_row <- as.integer((1 - test.size) * nrow(data))
```

Dataset de treino e teste utilizando a normalização min-max:

```
train1 <- data_norm1[1:split_row,]
test1 <- data_norm1[(split_row+1):nrow(data_norm1),]
```

Dataset de treino e teste utilizando o Z-score para normalização dos dados:

```
train2 <- data_norm2[1:split_row,]
test2 <- data_norm2[(split_row+1):nrow(data_norm2),]
```

Criando as labels de saída:

```
label.train <- data[1:split_row, 1]
label.test <- data[(split_row+1):nrow(data), 1]
```

5.2 Criando o modelo k-NN

Uma sugestão acadêmica para a escolha do k é calcular a raiz quadrada do tamanho da amostra e usar o valor obtido:

```
k <- ceiling(sqrt(nrow(data)))
```

Modelo com normalização min-max:

```
#label.train
data_pred1 <- knn(train = train1, test = test1, cl = label.train, k = k)
```

Modelo normalizado com Z-score:

```
data_pred2 <- knn(train = train2, test = test2, cl = label.train, k = k)
```

5.2.1 Matriz de Confusão usando o modelo normalizado com min-max

```
confusionMatrix(data_pred1, label.test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 88  2
##          M  0 24
##
##                Accuracy : 0.9825
##                  95% CI : (0.9381, 0.9979)
##     No Information Rate : 0.7719
##     P-Value [Acc > NIR] : 9.116e-11
##
##                   Kappa : 0.9488
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
```

```
##             Specificity : 0.9231
##          Pos Pred Value : 0.9778
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7719
##          Detection Rate : 0.7719
##    Detection Prevalence : 0.7895
##       Balanced Accuracy : 0.9615
##
##        'Positive' Class : B
##
```

5.2.2 Matriz de Confusão usando o modelo por Z-score

```
confusionMatrix(data_pred2, label.test)
```

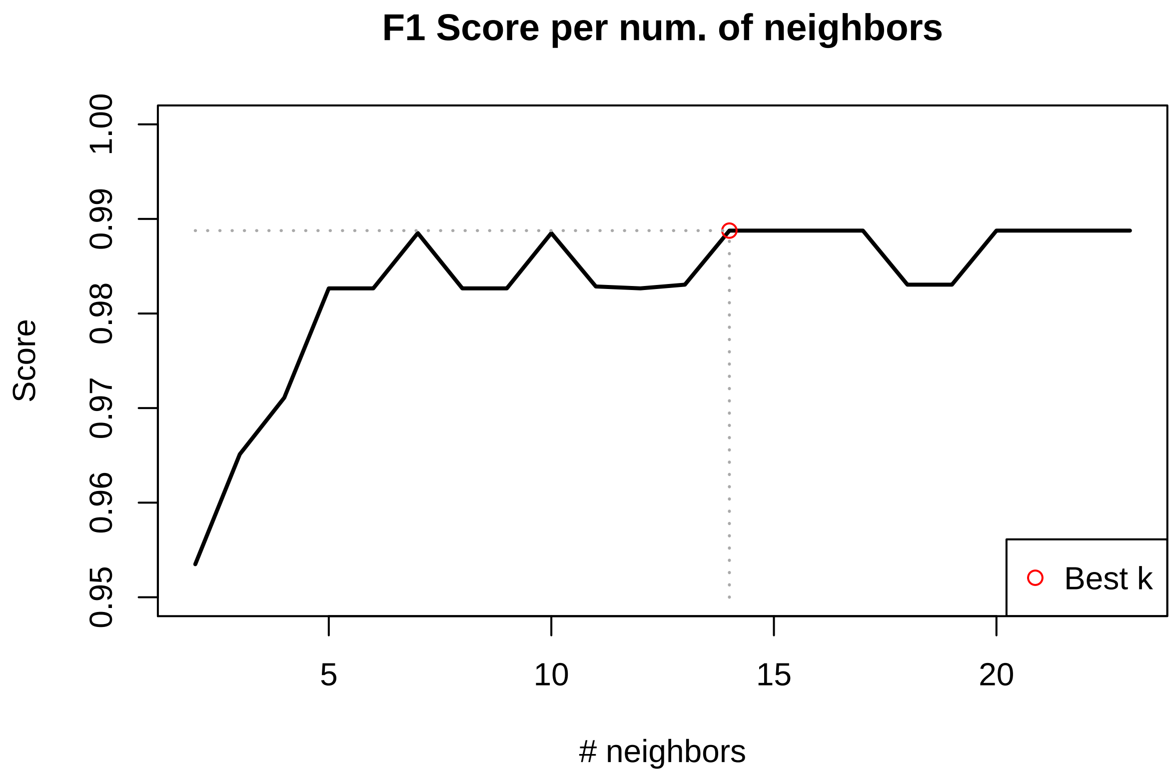
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 88  2
##          M  0 24
##
##                Accuracy : 0.9825
##                  95% CI : (0.9381, 0.9979)
##     No Information Rate : 0.7719
##     P-Value [Acc > NIR] : 9.116e-11
##
##                   Kappa : 0.9488
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9231
##          Pos Pred Value : 0.9778
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7719
##          Detection Rate : 0.7719
##    Detection Prevalence : 0.7895
##       Balanced Accuracy : 0.9615
##
##        'Positive' Class : B
##
```

Extra: Tuning k

```
##
## Attaching package: 'mltools'

## The following object is masked from 'package:tidyr':
##
##     replace_na
```

```r
scores <- c()
ks <- 2:as.integer(sqrt(nrow(data)))

for (k in ks) {
  preds <- knn(train = train1, test = test1, cl = label.train, k = k)
  f1 <- as.numeric(confusionMatrix(preds, label.test)$byClass["F1"])
  scores <- append(scores, f1)
}
```

```r
ymin <- 0.95
ymax <- 1.00

plot(ks, scores, type = "l", lwd=2,
     main="F1 Score per num. of neighbors",
     xlab = "# neighbors", ylab = "Score", ylim = c(ymin, ymax))

# Best k
points(ks[which.max(scores)], max(scores), pch=21, col="red")

# Vertical line
lines(x = c(ks[which.max(scores)], ks[which.max(scores)]),
      y = c(ymin, max(scores)),
      lty=3, col="darkgray", lwd=1.5)

# Horizontal line
lines(x = c(min(ks), ks[which.max(scores)]),
      y = c(max(scores), max(scores)),
      lty=3, col="darkgray", lwd=1.5)

legend("bottomright", pch=21, col="red", legend = "Best k")
```


F1 Score per num. of neighbors
Score
1.00
0.99
0.98
0.97
0.96
0.95
5
10
15
20
neighbors
Best k