Tema3_Ejercicio

Fran Camacho

2024-12-28

Tema3 - Ejercicio

La base de datos Diabetes en "Pima Indian Women", incluida en la librería MASS, contiene una muestra de 532 pacientes en edad adulta, y es utilizada para realizar un estudio sobre la diabetes.

[Observación: para conseguir los 532 registros de pacientes debe agregar los datasets Pima.tr y Pima.te usando la función rbind]

Cargar datos del paquete MASS

(Y también otros paquetes para funciones diversas).

```
if (!require(MASS)) install.packages('MASS', dependencies = T)
```

```
## Cargando paquete requerido: MASS
```

```
library(MASS)

if (!require(gmodels)) install.packages('gmodels', dependencies = T)  # cross tables
```

```
## Cargando paquete requerido: gmodels
```

```
library(gmodels)

if (!require(caret)) install.packages('caret', dependencies = T)     # data partitioning, confusion mat
```

```
## Cargando paquete requerido: caret
```

```
## Cargando paquete requerido: ggplot2
```

```
## Cargando paquete requerido: lattice
```

```
library(caret)

if (!require(class)) install.packages('class', dependencies = T)   # knn algorithm
```

```
## Cargando paquete requerido: class
```

```
library(class)
```

Paso 2: Explorar y preparar los datos

Examinar contenido (estructura y algunos registros) de los datasets Pima.tr y Pima.te:

```
#Structure
str(Pima.tr)
```

```
## 'data.frame':    200 obs. of  8 variables:
##  $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
##  $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
##  $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
##  $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
##  $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
##  $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
##  $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
##  $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

```
str(Pima.te)
```

```
## 'data.frame':    332 obs. of  8 variables:
##  $ npreg: int  6 1 1 3 2 5 0 1 3 9 ...
##  $ glu  : int  148 85 89 78 197 166 118 103 126 119 ...
##  $ bp   : int  72 66 66 50 70 72 84 30 88 80 ...
##  $ skin : int  35 29 23 32 45 19 47 38 41 35 ...
##  $ bmi  : num  33.6 26.6 28.1 31 30.5 25.8 45.8 43.3 39.3 29 ...
##  $ ped  : num  0.627 0.351 0.167 0.248 0.158 0.587 0.551 0.183 0.704 0.263 ...
##  $ age  : int  50 31 21 26 53 51 31 33 27 29 ...
##  $ type : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 2 1 1 2 ...
```

```
#Have a look at some data
head(Pima.tr,5)
```

```
##   npreg glu bp skin  bmi   ped age type
## 1     5  86 68   28 30.2 0.364  24   No
## 2     7 195 70   33 25.1 0.163  55  Yes
## 3     5  77 82   41 35.8 0.156  35   No
## 4     0 165 76   43 47.9 0.259  26   No
## 5     0 107 60   25 26.4 0.133  23   No
```

```
head(Pima.te,5)
```

```
##   npreg glu bp skin  bmi   ped age type
## 1     6 148 72   35 33.6 0.627  50  Yes
## 2     1  85 66   29 26.6 0.351  31   No
## 3     1  89 66   23 28.1 0.167  21   No
## 4     3  78 50   32 31.0 0.248  26  Yes
## 5     2 197 70   45 30.5 0.158  53  Yes
```

Se puede ver que los datasets tienen exactamente la misma estructura, así que podemos unirlos en un único dataset como se pide:

```
#Join datasets
pima <-  rbind(Pima.tr,Pima.te)
#write.csv(pima,file='Chapter03/pima.csv',na='') <- export data to be used lately with Python & scikit-
str(pima)
```

```
## 'data.frame':    532 obs. of  8 variables:
##  $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
##  $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
##  $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
##  $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
##  $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
##  $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
##  $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
##  $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

Variable dependiente

El atributo "type" es de especial interés, ya que es la variable que se quiere predecir.

Como dicho nombre "type" es demasiado genérico, le damos otro nombre más semántico ("diagnóstico"). También aprovechamos para cambiar el nombre de los niveles (a "positivo" y "negativo")

```
#Rename target variable and its levels
colnames(pima)[8] = "diagnosis"
levels(pima$diagnosis) <- c("Negative", "Positive")
```

Resumen estadístico de la variable dependiente

```
#Statistics of type/diagnosis
table(pima$diagnosis)
```

```
##
## Negative Positive
##      355      177
```

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

```
##
## Negative Positive
##    66.73    33.27
```

Normalización de los datos

El resultado del algoritmo KNNs depende en gran medida del modo en que se calculan las distancias entre las observaciones. Una variable con un rango de valores más grande, tendrá mucha mayor influencia que una con un rango más acotado. Es por tanto conveniente proceder a normalizar las variables numéricas.

Resumen estadístico antes de normalizar:

```r
#Summary before normalization
summary(pima)
```

```
##      npreg             glu               bp              skin
##  Min.   : 0.000   Min.   : 56.00   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 98.75   1st Qu.: 64.00   1st Qu.:22.00
##  Median : 2.000   Median :115.00   Median : 72.00   Median :29.00
##  Mean   : 3.517   Mean   :121.03   Mean   : 71.51   Mean   :29.18
##  3rd Qu.: 5.000   3rd Qu.:141.25   3rd Qu.: 80.00   3rd Qu.:36.00
##  Max.   :17.000   Max.   :199.00   Max.   :110.00   Max.   :99.00
##       bmi              ped              age           diagnosis
##  Min.   :18.20   Min.   :0.0850   Min.   :21.00   Negative:355
##  1st Qu.:27.88   1st Qu.:0.2587   1st Qu.:23.00   Positive:177
##  Median :32.80   Median :0.4160   Median :28.00
##  Mean   :32.89   Mean   :0.5030   Mean   :31.61
##  3rd Qu.:36.90   3rd Qu.:0.6585   3rd Qu.:38.00
##  Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

Normalización ("min-max scaling"):

```r
# Normalization (with the help of package caret):
# The preProcess() function scales the values to a range of 0 to 1 using method = c('range') as an argu
# The predict() method applies the actions of the preProcess() function on the entire data
process <- preProcess(pima, method=c("range"))
pima_norm <- predict(process, pima)
```

Resumen estadístico después de normalizar:

```r
#Summary after normalization
summary(pima_norm)
```

```
##      npreg              glu               bp              skin
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.05882   1st Qu.:0.2990   1st Qu.:0.4651   1st Qu.:0.1630
##  Median :0.11765   Median :0.4126   Median :0.5581   Median :0.2391
##  Mean   :0.20688   Mean   :0.4548   Mean   :0.5524   Mean   :0.2411
##  3rd Qu.:0.29412   3rd Qu.:0.5962   3rd Qu.:0.6512   3rd Qu.:0.3152
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       bmi               ped              age            diagnosis
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Negative:355
##  1st Qu.:0.1979   1st Qu.:0.07441   1st Qu.:0.03333   Positive:177
##  Median :0.2986   Median :0.14176   Median :0.11667
##  Mean   :0.3004   Mean   :0.17900   Mean   :0.17691
##  3rd Qu.:0.3824   3rd Qu.:0.24561   3rd Qu.:0.28333
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
```

Se comprueba que todos los valores están ahora entre 0 y 1.

```r
#Summary after normalization with aux. function
#
# normalize <- function(x) {
```

```
#   return ((x - min(x)) / (max(x) - min(x)))
# }
# pima_norm_2 <- as.data.frame(lapply(pima[1:7], normalize))
#
# summary(pima_norm_2)  <- se obtienen los mismos resultados, claro
```

Crear conjuntos de datos para entrenamiento y para test

Creamos los dos conjuntos con el siguiente código:

```
#Set seed to make the process reproducible
set.seed(9)

#partitioning data frame into training and testing sets
train_indices <- createDataPartition(pima_norm$diagnosis, times=1, p=.7, list=FALSE)

#create training set
pima_norm_train <- pima_norm[train_indices, 1:7]

#create testing set
pima_norm_test  <- pima_norm[-train_indices, 1:7]

#view number of rows in each set
#nrow(pima_norm_train)  # 373
#nrow(pima_norm_test)   # 159
```

Crear el etiquetado. (Lo necesitaremos para evaluar posteriormente el algoritmo).

```
#Labels for training and tests
pima_norm_train_labels <- pima_norm[train_indices , 8]
pima_norm_test_labels <- pima_norm[-train_indices, 8]
#length(pima_norm_train_labels)  # 373
#length(pima_norm_test_labels)   # 159
```

Paso 3: Entrenamiento

El algoritmo KNN, no construye un modelo, el proceso de entrenamiento consiste simplemente en almacenar los datos en un un formato estructurado. Para clasificar las instancias del conjunto test, utilizamos la implementación del algoritmo knn del paquete "class" (importado en el primer paso). Elegimos para k el valor 19, por ser 19 un valor aproximado a la raíz cuadrada del total de observaciones (373).

```
#> ?knn
pima_pred <- knn(train = pima_norm_train, test = pima_norm_test, cl = pima_norm_train_labels, k = 19)
```

Paso 4: Evaluación

Para evaluar el algoritmo, comparamos el resultado obtenido, con el vector donde guardamos anteriormente los datos etiquetados (pima_norm_test_labels). Para realizar dicha comparación, utilizaremos primero la función **CrossTable** del paquete "gmodels":

```r
# CrossTable/ConfusionMatrix
CrossTable(x = pima_norm_test_labels, y = pima_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  159
##
##
##                      | pima_pred
## pima_norm_test_labels |  Negative |  Positive | Row Total |
## ----------------------|-----------|-----------|-----------|
##              Negative |        94 |        12 |       106 |
##                       |     0.887 |     0.113 |     0.667 |
##                       |     0.810 |     0.279 |           |
##                       |     0.591 |     0.075 |           |
## ----------------------|-----------|-----------|-----------|
##              Positive |        22 |        31 |        53 |
##                       |     0.415 |     0.585 |     0.333 |
##                       |     0.190 |     0.721 |           |
##                       |     0.138 |     0.195 |           |
## ----------------------|-----------|-----------|-----------|
##          Column Total |       116 |        43 |       159 |
##                       |     0.730 |     0.270 |           |
## ----------------------|-----------|-----------|-----------|
##
##
```

Análisis de la tabla:

- El algoritmo ha predicho 94 resultados negativos de manera correcta.
- Pero en cambio ha devuelto 22 resultados como negativos, cuando en realidad son positivos (falsos negativos). Esto puede tener consecuencias bastante importantes, porque los pacientes pueden pensar que no tienen un problema (diabetes en este caso), y retrasar el tratamiento.
- El algoritmo también ha devuelto bastantes falsos positivos: ha devuelto 12 como casos de diabetes que en realidad no lo son.
- Sí ha devuelto 31 positivos de manera correcta.

Evaluación utilizando la función confusionMatrix del paquete "caret":

```r
# confusionMatrix from package caret
confusionMatrix(reference = pima_norm_test_labels, data = pima_pred, mode = "everything")
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction Negative Positive
##   Negative       94       22
##   Positive       12       31
##
##                Accuracy : 0.7862
##                  95% CI : (0.7142, 0.8471)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.0006448
##
##                   Kappa : 0.495
##
##  Mcnemar's Test P-Value : 0.1227126
##
##             Sensitivity : 0.8868
##             Specificity : 0.5849
##          Pos Pred Value : 0.8103
##          Neg Pred Value : 0.7209
##               Precision : 0.8103
##                  Recall : 0.8868
##                      F1 : 0.8468
##              Prevalence : 0.6667
##          Detection Rate : 0.5912
##    Detection Prevalence : 0.7296
##       Balanced Accuracy : 0.7358
##
##        'Positive' Class : Negative
##
```

Esta función nos devuelve directamente el valor de la **exactitud** (**"accuracy"**):

```
TP + TN / TP + FP + TN + FN (total) = 31 + 94 / 31 + 12 + 94 + 22 = 31 + 94 / 159 = 0.7862
```

Es decir, los clasificados correctamente sobre el total.

Con el parámetro mode = "everything", devuelve tambiém la Precision, el Recall y el F1:

Precisión

Esta función está considerando que lo "positivo" es no tener diabetes (" 'Positive' Class: Negative"), así que el valor se obtiene de esta manera

Precision = TN / TN + FN = 94 / 94 + 22 = 0.8103

Recall (Recuperación, o Tasa de verdaderos positivos (TPR))

Proporción de todos los positivos reales que se clasificaron correctamente como positivos, también se conoce como recuperación.

Recall = TN / TN + FP = 94 / 94 + 12 = 0.8868

F1 score (Puntuación F1)

La puntuación F1 es la media armónica (un tipo de promedio) de la precisión y la recuperación. Se obtiene de la siguiente manera:

F1 = 2 x (precision x recall) / (precision + recall) = 2TP / 2TP + FP + FN = 0.8468

(Al igual que en los dos casos anteriores, se considera como positivo a la clase negativa

```
 2x94 / 2x94 + 12 + 22 = 0.8468
```

)

"Esta métrica equilibra la importancia de la precisión y la recuperación, y es preferible a la precisión para los conjuntos de datos con desequilibrio de clases. Cuando la precisión y la recuperación tienen puntuaciones perfectas de 1.0, F1 también tendrá una puntuación perfecta de 1.0. En términos más generales, cuando la precisión y la recuperación sean similares en valor, F1 estará cerca de su valor. Cuando la precisión y la recuperación estén muy separadas, F1 será similar a la métrica que sea peor."

Tomado de:

https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

]

Paso 5: intentar mejorar el modelo

i) Usar otra normalización

Ahora probamos a normalizar utilizando la normalización "z score". Para ello usamos la función "scale":

```
# use the scale() function to z-score standardize a data frame (all columns but the target variable)
pima_z <- as.data.frame(scale(pima[-8]))
pima_z <- cbind(pima_z,pima[8])
```

Resumen estadístico después de normalizar con z:

```
#Summary after normalization z-score
summary(pima_z)
```

```
##      npreg              glu                bp                skin        
##  Min.   :-1.0619   Min.   :-2.0978   Min.   :-3.85903   Min.   :-2.10781  
##  1st Qu.:-0.7599   1st Qu.:-0.7187   1st Qu.:-0.60971   1st Qu.:-0.68248  
##  Median :-0.4580   Median :-0.1945   Median : 0.04016   Median :-0.01733  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000  
##  3rd Qu.: 0.4478   3rd Qu.: 0.6523   3rd Qu.: 0.69002   3rd Qu.: 0.64783  
##  Max.   : 4.0709   Max.   : 2.5152   Max.   : 3.12702   Max.   : 6.63422  
##       bmi                ped               age            diagnosis  
##  Min.   :-2.13486   Min.   :-1.2131   Min.   :-0.9863   Negative:355  
##  1st Qu.:-0.72884   1st Qu.:-0.7088   1st Qu.:-0.8005   Positive:177  
##  Median :-0.01311   Median :-0.2524   Median :-0.3359                 
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000                 
##  3rd Qu.: 0.58272   3rd Qu.: 0.4514   3rd Qu.: 0.5933                 
##  Max.   : 4.97155   Max.   : 5.5639   Max.   : 4.5890
```

Los valores no están entre 0 y 1, como ocurre con la normalizacón min-max.

Repetimos ahora los mismos pasos:

Dividir en 2 conjuntos (train y test)

```r
#Set seed to make the process reproducible
set.seed(9)

train_indices_z <- createDataPartition(pima_z$diagnosis, times=1, p=.7, list=FALSE)

pima_z_train <- pima_z[train_indices, 1:7]

pima_z_test  <- pima_z[-train_indices, 1:7]
```

Etiquetado

```r
pima_z_train_labels <- pima_z[train_indices, 8]
pima_z_test_labels <- pima_z[-train_indices, 8]
```

Entrenamiento

```r
pima_z_pred <- knn(train = pima_z_train, test = pima_z_test, cl = pima_z_train_labels, k = 19)
```

Evaluación

```r
confusionMatrix(reference = pima_z_test_labels, data = pima_z_pred, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Negative Positive
##   Negative       96       22
##   Positive       10       31
##
##                Accuracy : 0.7987
##                  95% CI : (0.7279, 0.8581)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.000168
##
##                   Kappa : 0.52
##
##  Mcnemar's Test P-Value : 0.051830
##
##             Sensitivity : 0.9057
##             Specificity : 0.5849
##          Pos Pred Value : 0.8136
##          Neg Pred Value : 0.7561
##               Precision : 0.8136
##                  Recall : 0.9057
##                      F1 : 0.8571
##              Prevalence : 0.6667
##          Detection Rate : 0.6038
##    Detection Prevalence : 0.7421
##       Balanced Accuracy : 0.7453
##
##        'Positive' Class : Negative
##
```

z-score: min-max norm.:

Accuracy : 0.7987 Accuracy : 0.7862
Precision : 0.8136 Precision : 0.8103
Recall : 0.9057 Recall : 0.8868 F1 : 0.8571 F1 : 0.8468

Se obtiene una pequeña mejora con la normalización z.

ii) Usando otros valores de k

k = 15 - min-max

```
pima_pred_k15 <- knn(train = pima_norm_train, test = pima_norm_test, cl = pima_norm_train_labels, k = 15
confusionMatrix(reference = pima_norm_test_labels, data = pima_pred_k15, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Negative Positive
##   Negative       93       19
##   Positive       13       34
##
##                Accuracy : 0.7987
##                  95% CI : (0.7279, 0.8581)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.000168
##
##                   Kappa : 0.534
##
##  Mcnemar's Test P-Value : 0.376759
##
##             Sensitivity : 0.8774
##             Specificity : 0.6415
##          Pos Pred Value : 0.8304
##          Neg Pred Value : 0.7234
##               Precision : 0.8304
##                  Recall : 0.8774
##                      F1 : 0.8532
##              Prevalence : 0.6667
##          Detection Rate : 0.5849
##    Detection Prevalence : 0.7044
##       Balanced Accuracy : 0.7594
##
##        'Positive' Class : Negative
##
```

k = 25 min-max

```
pima_pred_k25 <- knn(train = pima_norm_train, test = pima_norm_test, cl = pima_norm_train_labels, k = 25
confusionMatrix(reference = pima_norm_test_labels, data = pima_pred_k25, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction Negative Positive
##   Negative       96       21
##   Positive       10       32
##
##                Accuracy : 0.805
##                  95% CI : (0.7348, 0.8635)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 8.118e-05
##
##                   Kappa : 0.5373
##
##  Mcnemar's Test P-Value : 0.07249
##
##             Sensitivity : 0.9057
##             Specificity : 0.6038
##          Pos Pred Value : 0.8205
##          Neg Pred Value : 0.7619
##               Precision : 0.8205
##                  Recall : 0.9057
##                      F1 : 0.8610
##              Prevalence : 0.6667
##          Detection Rate : 0.6038
##    Detection Prevalence : 0.7358
##       Balanced Accuracy : 0.7547
##
##        'Positive' Class : Negative
##
```

k = 15 - z-score

```
pima_pred_z_k15 <- knn(train = pima_z_train, test = pima_z_test, cl = pima_z_train_labels, k = 15)
confusionMatrix(reference = pima_z_test_labels, data = pima_pred_z_k15, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Negative Positive
##   Negative       94       21
##   Positive       12       32
##
##                Accuracy : 0.7925
##                  95% CI : (0.7211, 0.8526)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.0003351
##
##                   Kappa : 0.5123
##
##  Mcnemar's Test P-Value : 0.1637344
##
##             Sensitivity : 0.8868
##             Specificity : 0.6038
##          Pos Pred Value : 0.8174
```

```
##          Neg Pred Value : 0.7273
##               Precision : 0.8174
##                  Recall : 0.8868
##                      F1 : 0.8507
##              Prevalence : 0.6667
##          Detection Rate : 0.5912
##    Detection Prevalence : 0.7233
##       Balanced Accuracy : 0.7453
##
##        'Positive' Class : Negative
##
```

k = 25 - z-score

```
pima_pred_z_k25 <- knn(train = pima_z_train, test = pima_z_test, cl = pima_z_train_labels, k = 25)
confusionMatrix(reference = pima_z_test_labels, data = pima_pred_z_k25, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Negative Positive
##   Negative       98       22
##   Positive        8       31
##
##                Accuracy : 0.8113
##                  95% CI : (0.7417, 0.8689)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 3.776e-05
##
##                   Kappa : 0.5455
##
##  Mcnemar's Test P-Value : 0.01762
##
##             Sensitivity : 0.9245
##             Specificity : 0.5849
##          Pos Pred Value : 0.8167
##          Neg Pred Value : 0.7949
##               Precision : 0.8167
##                  Recall : 0.9245
##                      F1 : 0.8673
##              Prevalence : 0.6667
##          Detection Rate : 0.6164
##    Detection Prevalence : 0.7547
##       Balanced Accuracy : 0.7547
##
##        'Positive' Class : Negative
##
```

R Python & scikit-learn (*)

k=15 , z-score k=15, min-max norm. k=15 , z-score k=15, min-max norm.

Accuracy : 0.7925 Accuracy : 0.7987 Accuracy : 0.7938 Accuracy : 0.7687

Precision : 0.8174 Precision : 0.8304 Precision : 0.6304 Precision : 0.7021

Recall : 0.8868 Recall : 0.8774 Recall : 0.6444 Recall : 0.5893

F1 : 0.8507 F1 : 0.8532 F1 : 0.6374 F1 : 0.6408

k=19 , z-score: k=19, min-max norm.: k=19 , z-score: k=19, min-max norm.:

Accuracy : 0.7987 Accuracy : 0.7862 Accuracy : 0.8438 Accuracy : 0.775000

Precision : 0.8136 Precision : 0.8103 Precision : 0.7500 Precision : 0.764706

Recall : 0.9057 Recall : 0.8868 Recall : 0.6667 Recall : 0.464286

F1 : 0.8571 F1 : 0.8468 F1 : 0.7059 F1 : 0.577778

k=25 , z-score: k=25, min-max norm.:	k=25 , z-score: k=25, min-max norm.:		
Accuracy : **0.8113**	Accuracy : 0.805	Accuracy : 0.8250	Accuracy : 0.7625

Precision : 0.8167 Precision : 0.8205 Precision : 0.7073 Precision : 0.6875

Recall : 0.9245 Recall : 0.9057 Recall : 0.6444 Recall : 0.5893

F1 : 0.8673 F1 : 0.8610 F1 : 0.6744 F1 : 0.6346

Diría que la opción que he probado primero (k=19, min-max), es la que obtiene peor resultado. Con Ks algo menor y algo mayor, se obtiene mejor exactitud. Y con la normalización z-score, se mejoran los valores de exactitud (y precision,recall y F1 también) para los 3 Ks elegidos.

```
# Others Ks (not better results obtained)
#
#pima_pred_z_k10 <- knn(train = pima_z_train, test = pima_z_test, cl = pima_z_train_labels, k = 10)
#confusionMatrix(reference = pima_z_test_labels, data = pima_pred_z_k10, mode = "everything")       # -
#pima_pred_z_k30 <- knn(train = pima_z_train, test = pima_z_test, cl = pima_z_train_labels, k = 30)
#confusionMatrix(reference = pima_z_test_labels, data = pima_pred_z_k30, mode = "everything")        #
#pima_pred_z_k35 <- knn(train = pima_z_train, test = pima_z_test, cl = pima_z_train_labels, k = 35)
#confusionMatrix(reference = pima_z_test_labels, data = pima_pred_z_k35, mode = "everything")        #
```