K-vecinos más cercanos (kNN) - Penguins

Karina Itzel Rodríguez Conde

26/5/2022

Introducción

El método de K vecinos más cercanos es también conocido como **kNN** y consiste en un método de clasificación no paramétrico que se basa en buscar para una observación, sus k vecinos más cercanos, es decir, aquellas observaciones que están más cercanas a una determinada distancia de dicha observación.

Matriz de datos

Para esta práctica, se trabajó con la matriz **penguins**, que se encuentra precargada en R y muestra las medidas morfológicas de tres diferentes especies de pingüinos así como otras características.

Cargar los datos penguins

```
library(readxl)
penguins <- read_excel("C:/Users/TOSHIBA/Downloads/penguins.xlsx")
head(penguins)
```

```
## # A tibble: 6 x 9
##   ID    especie isla      largo_pico_mm grosor_pico_mm largo_aleta_mm
##   <chr> <chr>   <chr>             <dbl>          <dbl>          <dbl>
## 1 i1    Adelie  Torgersen          39.1           18.7            181
## 2 i2    Adelie  Torgersen          39.5           17.4            186
## 3 i3    Adelie  Torgersen          40.3           18              195
## 4 i4    Adelie  Torgersen          37.8           18.1            190
## 5 i5    Adelie  Torgersen          36.7           19.3            193
## 6 i6    Adelie  Torgersen          39.3           20.6            190
## # ... with 3 more variables: masa_corporal_g <dbl>, genero <chr>, año <dbl>
```

```
Z<-as.data.frame(penguins)
```

Exploración de la matriz

1.- Dimensión

```r
dim(Z)
```

```
## [1] 344   9
```

La base de datos cuenta con 344 observaciones y 9 variables.

2.- Nombre de las variables

```r
colnames(Z)
```

```
## [1] "ID"              "especie"         "isla"            "largo_pico_mm"
## [5] "grosor_pico_mm"  "largo_aleta_mm"  "masa_corporal_g" "genero"
## [9] "año"
```

3.- Tipo de variables

```r
str(Z)
```

```
## 'data.frame':    344 obs. of  9 variables:
##  $ ID             : chr  "i1" "i2" "i3" "i4" ...
##  $ especie        : chr  "Adelie" "Adelie" "Adelie" "Adelie" ...
##  $ isla           : chr  "Torgersen" "Torgersen" "Torgersen" "Torgersen" ...
##  $ largo_pico_mm  : num  39.1 39.5 40.3 37.8 36.7 39.3 38.9 39.2 34.1 42 ...
##  $ grosor_pico_mm : num  18.7 17.4 18 18.1 19.3 20.6 17.8 19.6 18.1 20.2 ...
##  $ largo_aleta_mm : num  181 186 195 190 193 190 181 195 193 190 ...
##  $ masa_corporal_g: num  3750 3800 3250 3700 3450 ...
##  $ genero         : chr  "male" "female" "female" "female" ...
##  $ año            : num  2007 2007 2007 2007 2007 ...
```

4.- Saber si existen datos nulos

```r
anyNA(Z)
```

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

Esta base de datos no contiene datos nulos.

Tratamiento de la matriz

1.- Definir la matriz de datos y la variable respuesta con las clasificaciones

```r
x<-Z[,4:7]
y<-Z[,2]
y <- as.factor(y)
```

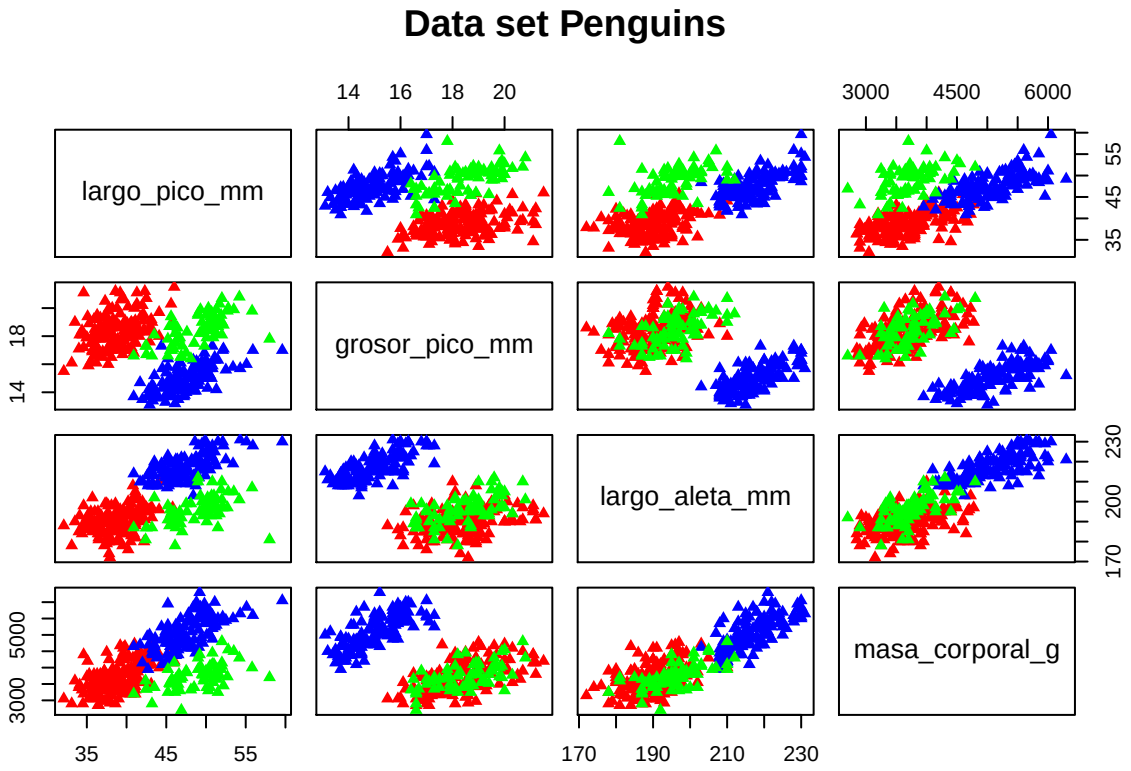
2.- Se definen las variables y observaciones

```
n<-nrow(x)
p<-ncol(x)
```

3.- Gráfico scatter plot

Creación de un vector de colores

```
col.penguins <- c("red","green", "blue")[y]
pairs(x, main="Data set Penguins", pch=17,col=col.penguins)
```



Aplicación del kNN

1.- Paquetería y librería a utilizar

```
library(class)
```

2.- Se fija una “semilla” para tener valores iguales

```
set.seed(12345)
```

3.- Creación de los ciclos para k=1 hasta k=20.

Se selecciona el valor de k que tenga el error más bajo.

3.1.- Inicialización de una lista vacía de tamaño 20

```
knn.class<-vector(mode="list",length=20)
knn.tables<-vector(mode="list", length=20)
```

3.2.- Clasificaciones erróneas

```
knn.mis<-matrix(NA, nrow=20, ncol=1)
knn.mis
```

```
##       [,1]
##  [1,]   NA
##  [2,]   NA
##  [3,]   NA
##  [4,]   NA
##  [5,]   NA
##  [6,]   NA
##  [7,]   NA
##  [8,]   NA
##  [9,]   NA
## [10,]   NA
## [11,]   NA
## [12,]   NA
## [13,]   NA
## [14,]   NA
## [15,]   NA
## [16,]   NA
## [17,]   NA
## [18,]   NA
## [19,]   NA
## [20,]   NA
```

```
for(k in 1:20){
  knn.class[[k]]<-knn.cv(x,y,k=k)
  knn.tables[[k]]<-table(y,knn.class[[k]])
  #La suma de las clasificaciones menos las correctas
  knn.mis[k]<- n-sum(y==knn.class[[k]])
}

knn.mis
```

```
##       [,1]
##  [1,]   44
##  [2,]   58
##  [3,]   72
##  [4,]   75
##  [5,]   71
##  [6,]   79
##  [7,]   79
##  [8,]   74
##  [9,]   74
## [10,]   73
## [11,]   73
## [12,]   72
```

```
## [13,]   73
## [14,]   75
## [15,]   80
## [16,]   88
## [17,]   89
## [18,]   85
## [19,]   84
## [20,]   82
```

4.- Número óptimo de k-vecinos

```
which(knn.mis==min(knn.mis))
```

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

5.- Visualización de los números óptimos de k-vecinos

```
knn.tables[[1]]
```

```
##
## y           Adelie Chinstrap Gentoo
##   Adelie       136        12      4
##   Chinstrap     18        46      4
##   Gentoo         2         4    118
```

¿Cuál es el número óptimo de K-vecinos cercanos? Se muestra que el más eficiente es k=1

6.- Se señala el k más eficiente

```
k.opt<-1
knn.cv.opt<-knn.class[[k.opt]]
knn.cv.opt
```

```
##   [1] Adelie    Adelie    Chinstrap Adelie    Adelie    Adelie    Adelie
##   [8] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
##  [15] Adelie    Adelie    Adelie    Chinstrap Adelie    Adelie    Adelie
##  [22] Chinstrap Adelie    Adelie    Adelie    Adelie    Adelie    Chinstrap
##  [29] Adelie    Adelie    Chinstrap Adelie    Adelie    Adelie    Adelie
##  [36] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
##  [43] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
##  [50] Adelie    Adelie    Adelie    Adelie    Chinstrap Adelie    Adelie
##  [57] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
##  [64] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie    Chinstrap
##  [71] Adelie    Adelie    Adelie    Chinstrap Adelie    Adelie    Adelie
##  [78] Adelie    Adelie    Adelie    Adelie    Gentoo    Adelie    Adelie
##  [85] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
##  [92] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie    Gentoo
##  [99] Adelie    Adelie    Adelie    Gentoo    Adelie    Adelie    Adelie
## [106] Adelie    Adelie    Adelie    Adelie    Gentoo    Adelie    Adelie
## [113] Adelie    Adelie    Adelie    Adelie    Adelie    Chinstrap Adelie
```

```
## [120] Adelie    Adelie    Adelie    Adelie    Chinstrap Adelie    Adelie
## [127] Adelie    Adelie    Adelie    Adelie    Adelie    Chinstrap Adelie
## [134] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
## [141] Chinstrap Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
## [148] Adelie    Adelie    Adelie    Adelie    Adelie    Chinstrap Gentoo
## [155] Gentoo    Gentoo    Gentoo    Chinstrap Gentoo    Gentoo    Gentoo
## [162] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [169] Adelie    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [176] Gentoo    Gentoo    Gentoo    Chinstrap Gentoo    Gentoo    Gentoo
## [183] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [190] Gentoo    Adelie    Gentoo    Chinstrap Gentoo    Gentoo    Gentoo
## [197] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [204] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [211] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [218] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [225] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [232] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [239] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [246] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [253] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [260] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [267] Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo    Gentoo
## [274] Gentoo    Gentoo    Gentoo    Chinstrap Chinstrap Chinstrap Chinstrap
## [281] Adelie    Chinstrap Adelie    Adelie    Adelie    Chinstrap Chinstrap
## [288] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [295] Chinstrap Adelie    Adelie    Chinstrap Adelie    Chinstrap Chinstrap
## [302] Adelie    Chinstrap Chinstrap Adelie    Gentoo    Adelie    Chinstrap
## [309] Chinstrap Chinstrap Chinstrap Chinstrap Adelie    Gentoo    Adelie
## [316] Gentoo    Chinstrap Chinstrap Adelie    Adelie    Chinstrap Adelie
## [323] Chinstrap Gentoo    Chinstrap Chinstrap Adelie    Chinstrap Chinstrap
## [330] Chinstrap Chinstrap Chinstrap Adelie    Chinstrap Chinstrap Chinstrap
## [337] Chinstrap Chinstrap Chinstrap Adelie    Chinstrap Chinstrap Chinstrap
## [344] Chinstrap
## Levels: Adelie Chinstrap Gentoo
```

7.- Tabla de contingencia con las clasificaciones buenas y malas

```
knn.tables[[k.opt]]
```

```
##
## y           Adelie Chinstrap Gentoo
##   Adelie       136        12      4
##   Chinstrap     18        46      4
##   Gentoo         2         4    118
```

8.- Cantidad de observaciones mal clasificadas

```
knn.mis[k.opt]
```

```
## [1] 44
```

¿Cuál es la cantidad de observaciones mal clasificadas? Se puede observar que hay 44 malas clasificaciones, puesto que están fuera de la diagonal.

9.- Error de clasificación (MR)

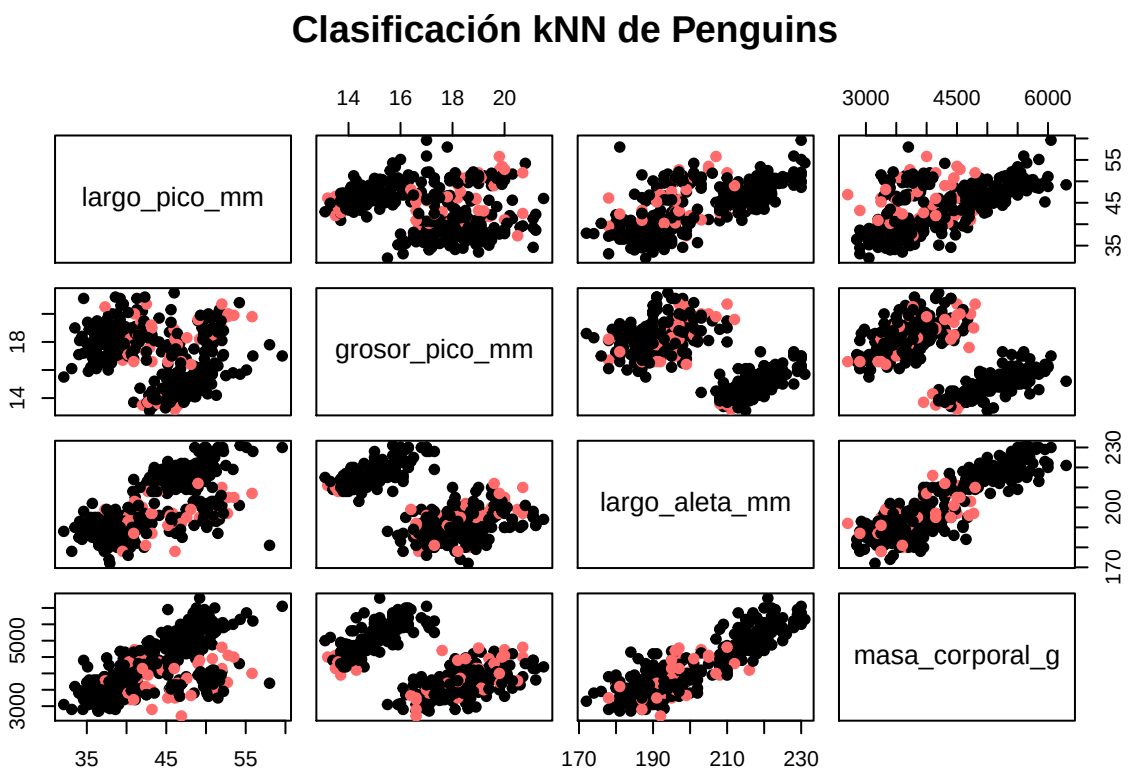
```
knn.mis[k.opt]/n
```

```
## [1] 0.127907
```

¿Cuál es el ratio de mala clasificación (MR)? El ratio de clasificación es de un 0.127907%

10.- Gráfico de clasificaciones correctas y erróneas

```
col.knn.penguins<-c("indianred1","black")[1*(y==knn.cv.opt)+1]
pairs(x, main="Clasificación kNN de Penguins",
      pch=19, col=col.knn.penguins)
```



Aquellas observaciones marcadas en rojo son las mal clasificadas y las de negro son las correctas.