# KNN para el análisis de variables cualitativas con R

## Contents



## 1 Introducción

Se quiere predecir la especie de dos pingüinos con las siguientes características:

```
##   long_pico prof_pico long_aleta peso   isla genero
## 1      39.8      18.4        192 3250  Dream hembra
## 2      46.1      15.5        202 4000 Biscoe  macho
```

utilizando los datos del archivo pinguinos.csv y el algoritmo KNN.

## 2 Lectura y preparación de los datos

En primer lugar se leen los datos y se crean los factores correspondientes

```
## se leen los datos
d = read.csv("datos/pinguinos.csv", sep = ";")
```

```
# es conveniente convertir a factores las variables cualitativas
d$especie = factor(d$especie)
d$isla = factor(d$isla)
d$genero = factor(d$genero)
```

- Se normalizan los regresores numéricos:

```
# se lee la funcion que normaliza
source("funciones/knn_funciones.R")
```

```
# se normalizan y se guardan en un data.frame
d1 = data.frame(
  long_pico = knn_normaliza(d$long_pico),
  prof_pico = knn_normaliza(d$prof_pico),
  long_aleta = knn_normaliza(d$long_aleta),
  peso = knn_normaliza(d$peso)
)
```

- Se definen las variables auxiliares. En este caso se deciden utilizar una variable menos que el número de niveles:

```
#
d1$isla_Bis = ifelse(d$isla == "Biscoe", 1, 0)
d1$isla_Dre = ifelse(d$isla == "Dream", 1, 0)
#
d1$genero_H = ifelse(d$genero == "hembra", 1, 0)
```

- Se preparan los valores que se quieren predecir:

```
# variables cuantitativas normalizadas
xp1 = data.frame(
  long_pico = knn_normaliza(c(39.8,46.1), min(d$long_pico), max(d$long_pico)),
  prof_pico = knn_normaliza(c(18.4,15.5), min(d$prof_pico), max(d$prof_pico)),
  long_aleta = knn_normaliza(c(192,202), min(d$long_aleta), max(d$long_aleta)),
  peso = knn_normaliza(c(3250,4000), min(d$peso), max(d$peso))
)
# variables auxiliares
xp1$isla_Bis = c(0,1)
xp1$isla_Dre = c(1,0)
#
xp1$genero_H = c(1,0)
```

# 3 Algoritmo KNN con R

Para aplicar el algoritmo se va a utilizar el paquete FNN de R:

```
# se utiliza la funcion knn.reg del paquete FNN
yp = FNN::knn(d1, test = xp1, cl = d$especie, k = 3)
yp
```

```
## [1] Adelie Gentoo
## attr(,"nn.index")
##      [,1] [,2] [,3]
## [1,]   28   84   79
## [2,]  189  170  110
## attr(,"nn.dist")
##           [,1]      [,2]      [,3]
## [1,] 0.1000516 0.1180674 0.1201057
## [2,] 0.3456152 0.3673515 0.3703687
## Levels: Adelie Gentoo
```

- Se observa que se devuelve la predicción: Adelie, Gentoo
- attr(,“nn.index”) devuelve los k puntos más cercanos.
- attr(,“nn.dist”) devuelve la distancia de los k puntos más cercanos.

```
# la especie de los k puntos más cercanos es
(p1 = attr(yp,"nn.index")[1,])
```

```
## [1] 28 84 79
```

```
d$especie[p1]
```

```
## [1] Adelie Adelie Adelie
## Levels: Adelie Chinstrap Gentoo
```

Luego la predicción es Adelie.

## 3.1 Cálculo del error de predicción: matriz de confusión

```
# se crean los datos de entrenamiento y los datos test (80%-20%)
set.seed(678)
n = nrow(d)
pos_train = sample(1:n,round(0.8*n), replace = F)
train_x = d1[pos_train,]
test_x = d1[-pos_train,]
train_y = d$especie[pos_train]
test_y = d$especie[-pos_train]
```

```
# se utiliza la funcion knn.reg del paquete FNN
yp = FNN::knn(train_x, test = test_x, cl = train_y, k = 1)
```

El método más sencillo para calcular el error de predicción es la **matriz de confusión**:

```
# matriz de confusion
(t = table(test_y, yp))
```

```
##            yp
## test_y      Adelie Chinstrap Gentoo
##   Adelie        26         2      0
##   Chinstrap      0        13      0
##   Gentoo         0         0     26
```

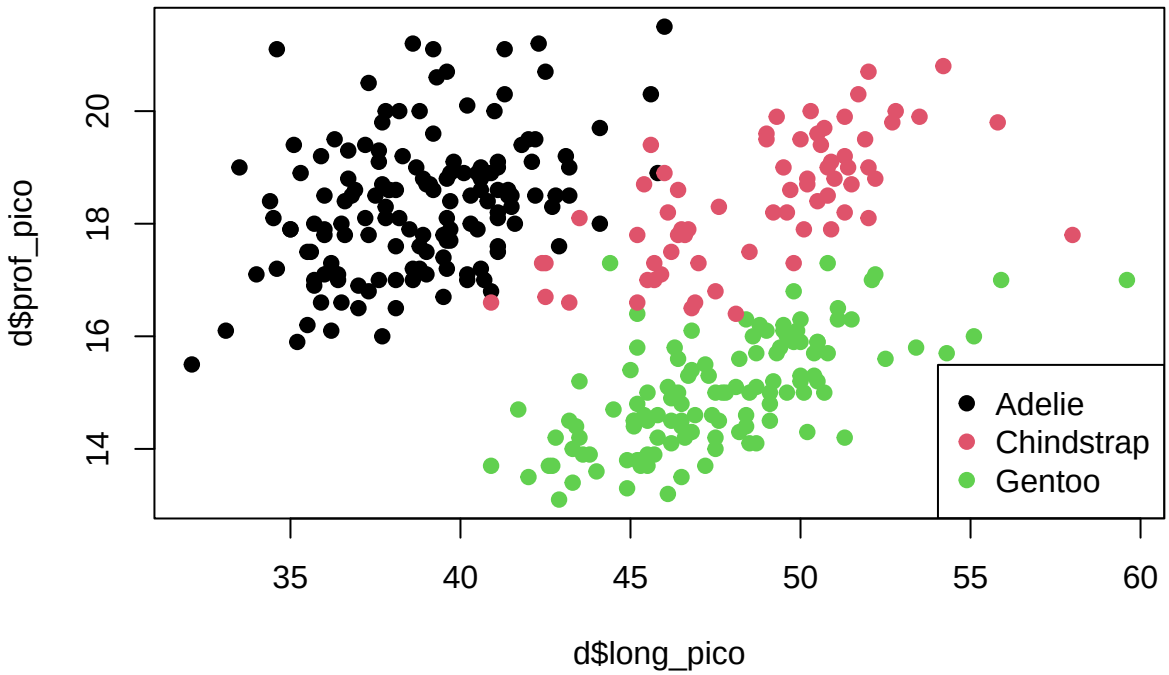
Los valores de la diagonal de la matriz están bien predichos, los de fuera de la diagonal son errores de predicción:

```
# error
1 - sum(diag(t))/sum(t)
```

```
## [1] 0.02985075
```

Sólo hay dos pingüinos mal predichos, la predicción es prácticamente perfecta. Esto ocurre porque la longitud del pico y la profundidad del pico caracterizan muy bien la especie de los pingüinos:

```
plot(d$long_pico, d$prof_pico, col = d$especie, pch = 19)
legend("bottomright", legend = c("Adelie","Chindstrap","Gentoo"), pch = 19, col = 1:3)
```




Si se observa el gráfico, todas las especies están agrupadas en regiones muy bien delimitadas. Luego k = 1 es suficiente para realizar la predicción final:

```
# se utilizan todos los datos
(yp = FNN::knn(d1, test = xp1, cl = d$especie, k = 1))
```

```
## [1] Adelie Gentoo
## attr(,"nn.index")
##      [,1]
## [1,]   28
## [2,]  189
## attr(,"nn.dist")
##           [,1]
## [1,] 0.1000516
## [2,] 0.3456152
## Levels: Adelie Gentoo
```

En caso contrario se podría calcular el k óptimo utilizando los subconjuntos train-test o validación cruzada.

## 3.2 Cálculo de k con validación cruzada

Cuando se trata de un problema de clasicación se tiene que utilizar la función knn.cv:

```
yp=FNN::knn.cv(train_x,train_y, k = 3)
str(yp)
```

```
##  Factor w/ 3 levels "Adelie","Chinstrap",..: 1 3 2 1 3 1 1 1 2 3 ...
##  - attr(*, "nn.index")= int [1:266, 1:3] 261 10 40 255 162 160 110 129 178 2 ...
##  - attr(*, "nn.dist")= num [1:266, 1:3] 0.0813 0.04 0.0552 0.1912 0.0729 ...
```

El significado es el mismo que lo obtenido con *knn.reg*. La matriz de confusion nos da una medida del error de prediccion:

```
(t = table(train_y,yp))
```

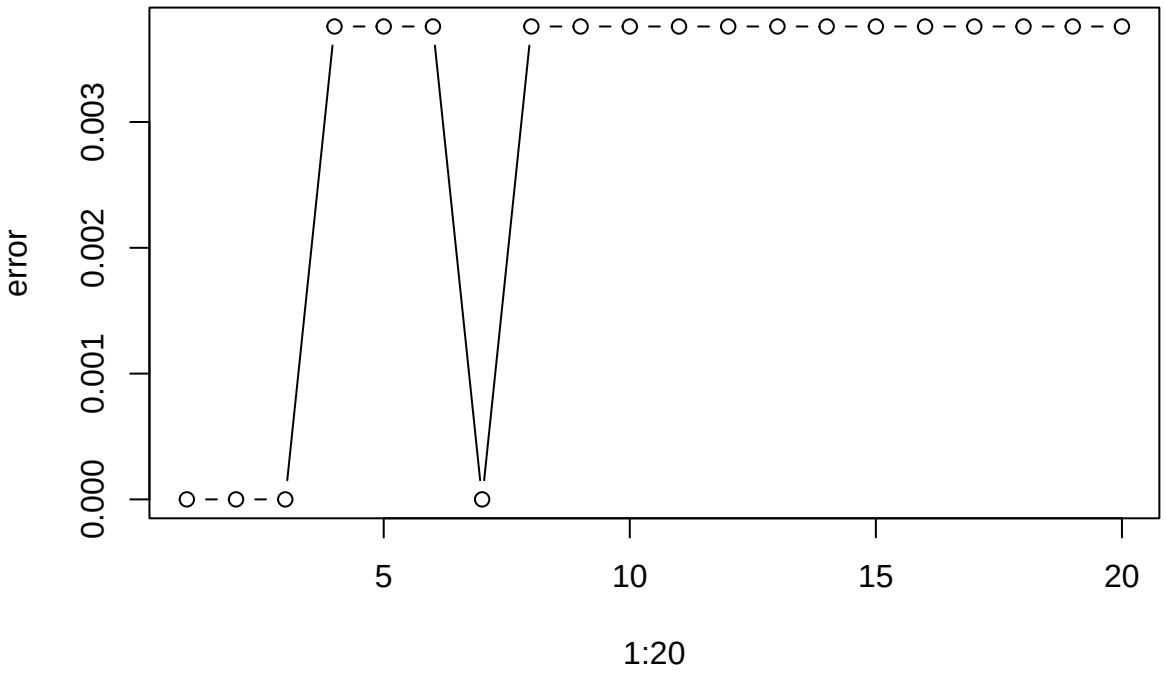
```
##           yp
```

```
## train_y     Adelie Chinstrap Gentoo
##   Adelie       118         0      0
##   Chinstrap      0        55      0
##   Gentoo         0         0     93
```

Por tanto, utilizando diferentes valores de k se obtiene:

```
error = rep(0,20)
for (ii in 1:20){
  yp = FNN::knn.cv(train_x,train_y, k = ii)
  t = table(train_y,yp)
  error[ii] = 1 - sum(diag(t))/sum(t)
}
```

```
plot(1:20,error, type = "b")
```




Se observa que el error siempre es pequeño, como máximo se clasifica mal una observación: error = $1/333 = 0.003$