K-Vecinos más cercanos (KNN) de iris

Omar Sanchez Hernandez

5/6/2022

Introducción

K-vecinos más cercanos es un método para clasificar casos basándose en sus similitudes a otros casos.

Librerias necesarias

```
library(MASS)
```

Cargar los datos iris

```
Z<-as.data.frame(iris)
colnames(Z)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

Definir la matriz de datos y la variable respuesta con las clasificaciones

```
x<-Z[,1:4]
y<-Z[,5]
```

Se definen las variables y observaciones

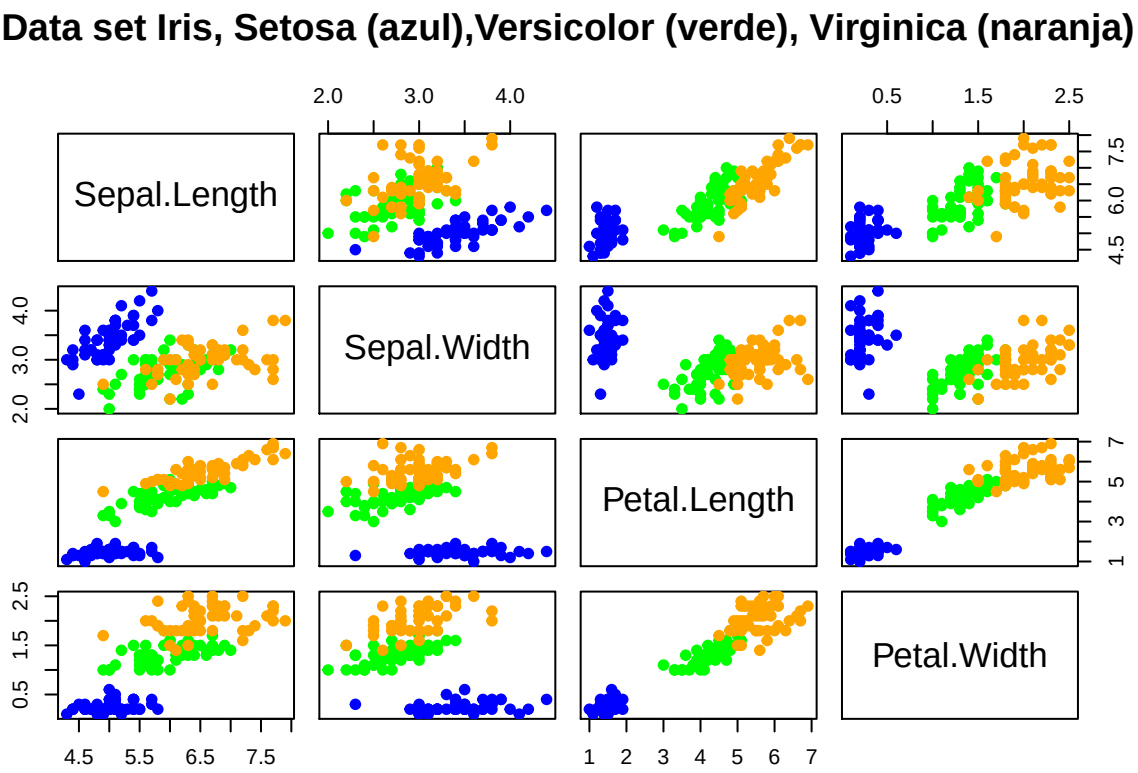
```
n<-nrow(x)
p<-ncol(x)
dim(Z)
```

```
## [1] 150   5
```

La matriz de datos contiene 150 variables y 5 variables.

Scatter plot de las variables

```
col.iris<-c("blue","green","orange")[y]
pairs(x, main="Data set Iris, Setosa (azul),Versicolor (verde), Virginica (naranja)",
      pch=19,col=col.iris)
```



Se observa que una especie de las flores se alcanza a diferenciar correctamente de las demás mientras que hay 2 que tienen características similares que las hace confundirse o traslaparse entre ellas.

Se fija una "semilla" para tener valores iguales

```
library(class)
library(knitr)
set.seed(1000)
```

Creacion de los ciclos para k=1 hasta k=20

Selecciona el valor de k que tenga el error mas bajo.

```
# Inicialización de una lista vacia de tamaño 20
knn.class<-vector(mode="list",length=20)
knn.tables<-vector(mode="list", length=20)
```

Clasificaciones erroneas

```
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
```

Aplicacion del ciclo

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

Se presenta el vector del ciclo en el que se obtienen cual podria ser el numero optimo de vecinos mas cercanos.

Numero optimo de k-vecinos

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

```
## [1] 14 18 19
```

```
knn.tables[[14]]
```

```
##
## y            setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

```
knn.tables[[18]]
```

```
##
## y            setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

```
knn.tables[[19]]
```

```
##
## y            setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

El numero mas eficiente es k=14.

Se presentan los resultados del k = 14

```
k.opt<-14
knn.cv.opt<-knn.class[[k.opt]]
knn.cv.opt
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor virginica  versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor virginica
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  versicolor virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

Tabla de contingencia con las clasificaciones buenas y malas

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

```
##
## y            setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

Cantidad de observaciones mal clasificadas

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

```
## [1] 3
```

Error de clasificacion (MR)

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

```
## [1] 0.02
```

Grafico de clasificaciones correctas y erroneas

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