K Nearest Neighbours (KNN) para clasificación

Contents

1 Introduccion a la clasificacion

1.1 Lectura de datos

```
d = read.csv("datos/MichelinNY.csv")
str(d)
```

```
## 'data.frame':    164 obs. of  6 variables:
##  $ InMichelin     : int  0 0 0 1 0 0 1 1 1 0 ...
##  $ Restaurant.Name: chr  "14 Wall Street" "212" "26 Seats" "44" ...
##  $ Food           : int  19 17 23 19 23 18 24 23 27 20 ...
##  $ Decor          : int  20 17 17 23 12 17 21 22 27 17 ...
##  $ Service        : int  19 16 21 16 19 17 22 21 27 19 ...
##  $ Price          : int  50 43 35 52 24 36 51 61 179 42 ...
```

2 Algoritmo K-vecinos más próximos

- Datos: (yi, x1i, x2i,…,xpi), i = 1,…,n

y	x1	x2		xp
y1	x11	x21	…	xp1
y1	x12	x22	…	xp2
…	…	…	…	…
y1	x1n	x2n	…	xpn

- Se calcula la distancia euclídea del dato que se quiere clasificar (x1a, x2a,…,xpa) con cada uno de los puntos de la base de datos

$$d_i = \sqrt{(x_{1i} - x_{1a})^2 + (x_{2i} - x_{2a})^2 + \cdots + (x_{pi} - x_{pa})^2}$$

- se ordenan las distancias de menor a mayor y se le asigna al nuevo dato la categoría mayoritaria dentro de los k-datos con menor distancia.
- es decir, si K = 1, se le asigna la categoría del punto más cercano.
- se suelen utilizar K impares para evitar empates.
- a menudo se utilizan regresores estandarizados para que todos los regresores tengan la misma contribución a la distancia.

3 Análisis de los datos InMichlelin

Los regresores cualitativos son dificiles de modelar (cual es la distancia entre colores, por ejemplo?). Por eso no los vamos a incluir en el análisis (punto débil del algoritmo).

3.1 Training, validation and test sets

- Seleccionamos 80% de los datos para el training set y 20% de los datos para el test set:

```
set.seed(123)
n = nrow(d)
pos_train = sample(1:n,round(0.8*n), replace = F)
train_x = d[pos_train,3:6]
test_x = d[-pos_train,3:6]
train_y = d$InMichelin[pos_train]
test_y = d$InMichelin[-pos_train]
```

3.2 Resultado

- k = 1

```
library(class)
test_pred = knn(train_x, test_x, train_y, k = 1)
table(test_y,test_pred)
```

```
##       test_pred
## test_y  0  1
##      0 16  3
##      1  5  9
```

- k = 3

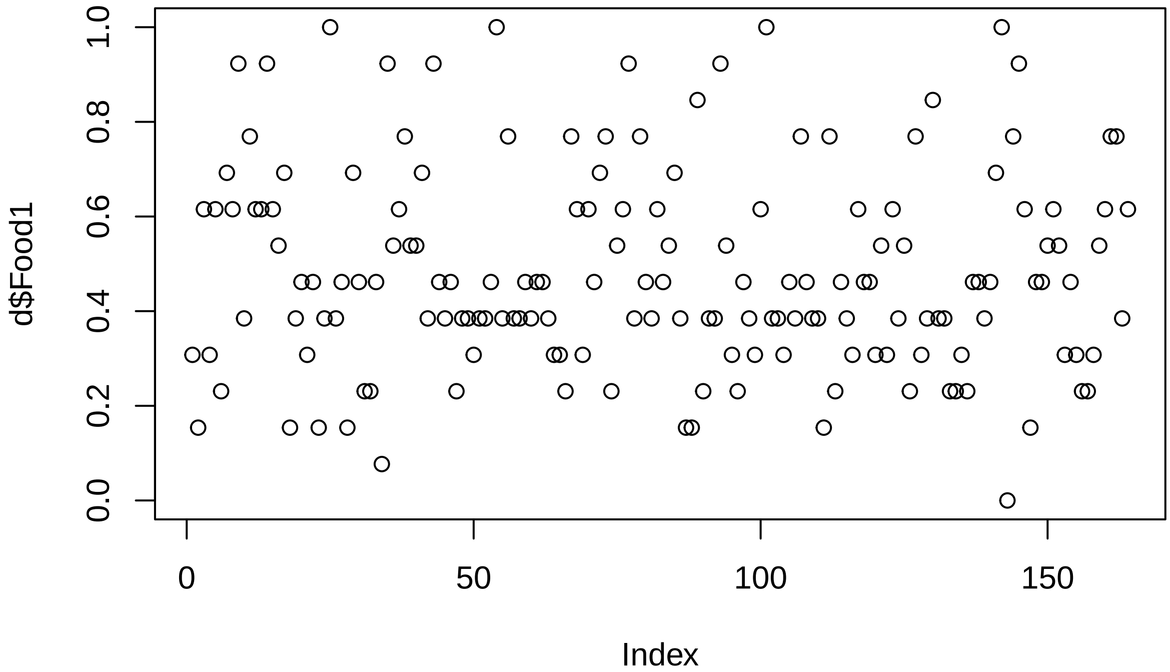
```
test_pred = knn(train_x, test_x, train_y, k = 3)
table(test_y,test_pred)
```

```
##       test_pred
## test_y  0  1
##      0 17  2
##      1  5  9
```

3.3 Escalado de variables

Para que todas las variables tengan la misma importancia, escalamos las variables numericas:

```
d$Food1 = (d$Food - min(d$Food))/(max(d$Food) - min(d$Food))
plot(d$Food1)
```




```
normalization = function(x){
  x1 = (x - min(x))/(max(x) - min(x))
  return(x1)
}
```

```
d$Decor1 = normalization(d$Decor)
d$Service1 = normalization(d$Service)
d$Price1 = normalization(d$Price)
```

```
train_x = d[pos_train,7:10]
test_x = d[-pos_train,7:10]
test_pred = knn(train_x, test_x, train_y, k = 1)
table(test_y,test_pred)
```

```
##       test_pred
## test_y  0  1
##      0 15  4
##      1  3 11
```

Como vemos, la clasificación mejora.

3.4 Comparación con regresión logística

```
train = d[pos_train,]
test = d[-pos_train,]
m = glm(InMichelin ~ Food + Decor + Service + Price, data = train, family = binomial)
prob = predict(m, newdata = test, type = "response")
```

```
pred = rep(0,length(prob))
pred[prob > 0.5] = 1
# Matriz de confusion
table(test$InMichelin, pred)
```

```
##    pred
```

```
##      0  1
##   0 19  0
##   1  5  9
```

Como se observa, el modelo logit predice mejor.

4 Analisis de los datos IRIS

```
d = iris
str(d)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
d$Sepal.Length1 = normalization(d$Sepal.Length)
d$Sepal.Width1 = normalization(d$Sepal.Width)
d$Petal.Length1 = normalization(d$Sepal.Length)
d$Petal.Width1 = normalization(d$Petal.Width)
```

```
set.seed(123)
n = nrow(d)
pos_train = sample(1:n,round(0.8*n), replace = F)
train_x = d[pos_train,6:9]
test_x = d[-pos_train,6:9]
train_y = d[pos_train,5]
test_y = d[-pos_train,5]
```

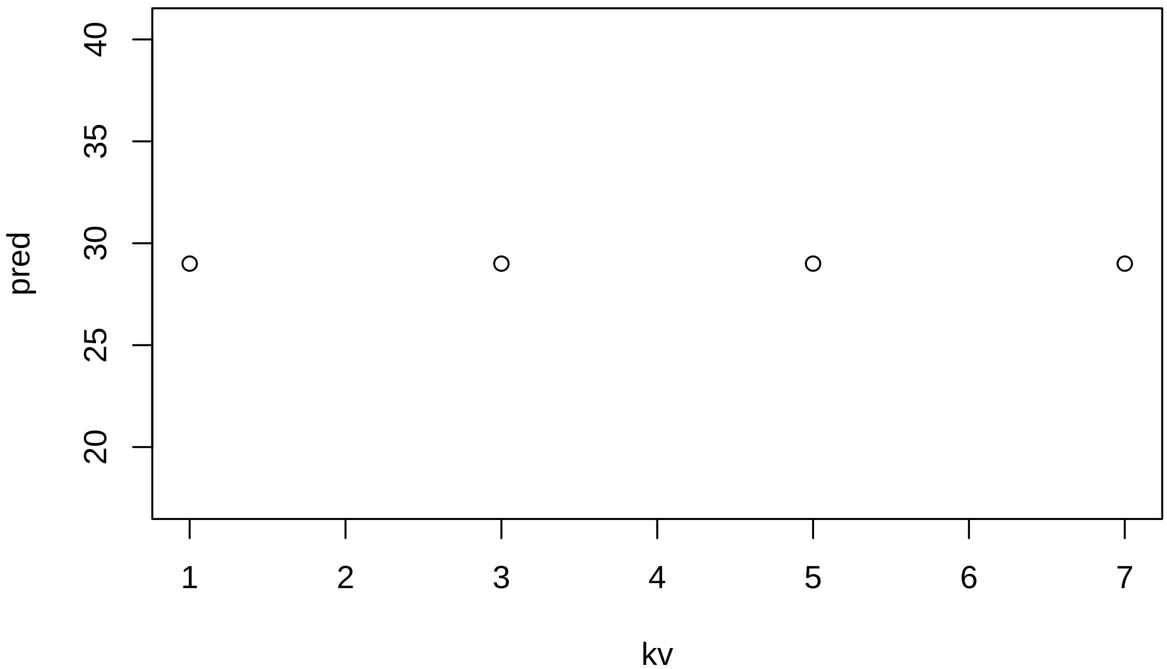
```
pred = knn(train_x,test_x,train_y, k=3)
```

```
table(test_y, pred)
```

```
##             pred
## test_y       setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         14         1
##   virginica       0          0         5
```

Calculamos el valor optimo de k:

```
kv = c(1,3,5,7)
pred = rep(0,4)
ii = 1
for (i in kv){
  pred_i = knn(train_x,test_x,train_y, k=i)
  pred[ii] = sum(diag(table(test_y,pred_i)))
  ii = ii + 1
}
plot(kv,pred)
```


pred
40
35
30
25
20
1
2
3
4
5
6
7
kv