MACHINE LEARNING EN RSTUDIO

METODO K VECINOS PROXIMOS

EDUARD LARA

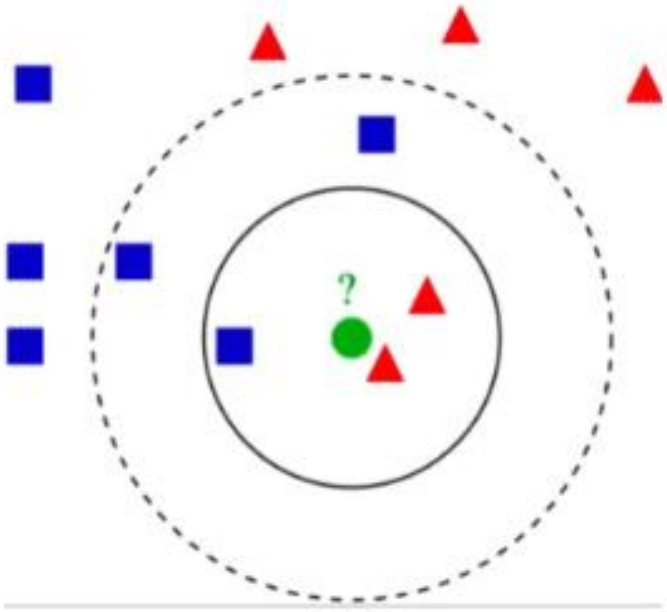
1. INDICE

1. INTRODUCCION METODO K VECINOS MAS CERCANOS

- El método de los vecinos más cercanos es un método de clasificación supervisada, que estima la probabilidad de que un elemento 'X' pertenezca a una clase 'C' a partir de la información proporcionada
- En el reconocimiento de patrones, este algoritmo es utilizado como método de clasificación de objetos, basado en un entrenamiento mediante ejemplos cercanos en el espacio de los elementos
- Es sencillo, fácil de entrenar y trabaja con cualquier número de clases.

1. INTRODUCCION METODO K VECINOS MAS CERCANOS

- Por ejemplo, tenemos un círculo verde con un interrogante verde como el que aparece en el diagrama, y queremos saber a que clase pertenece, si pertenece a los triángulos rojos o a los rectángulos azules.



1. INTRODUCCION METODO K VECINOS MAS CERCANOS

1) Primero calculamos la distancia del círculo verde a todos los elementos.
2) Segundo ordenamos los puntos de menor a mayor distancia que los separa del circulo verde.
3) Por último, predecimos el grupo al que pertenece el círculo verde en función de la clase a la que pertenecen los "K" elementos más cercanos,

1. INTRODUCCION METODO K VECINOS MAS CERCANOS

- En la figura del diagrama aparecen dos círculos negros cuyo centro es el círculo verde, uno solido y otro discontinuo.
- Si consideramos el circulo negro solido, donde hay dos triángulos rojos y uno azul, o lo que es lo mismo k=3 vecinos más cercanos, como ganan los rojos, el circulo verde será un triángulo rojo.
- En cambio, si consideramos k=5 vecinos mas cercanos, que se corresponde con el círculo punteado exterior, como hay 3 elementos azules y 2 rojos, el circulo verde pertenecerá a los rectángulos azules.
- Así que en función de K podremos asignar el círculo verde a una clase o a la otra.

2. EJEMPLO K VECINOS MAS CERCANOS I

Paso 1. Vamos a ver el algoritmo de los vecinos más cercanos. Instalaremos el paquete ISLR, y lo cargamos en memoria

```
> install.packages('ISLR')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/ISLR_1.4.zip'
Content type 'application/zip' length 2924157 bytes (2.8 MB)
downloaded 2.8 MB

package 'ISLR' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\RtmpqUg9Oh\downloaded_packages
> library(ISLR)
>
```

2. EJEMPLO K VECINOS MAS CERCANOS I

Paso 2. Dentro de este paquete hay un dataset que se llama Caravan. Lo asignaremos a una variable datos.

Es una base de datos de seguros que tiene 5822 observaciones o filas y 86 variables o columnas.

```
> datos = Caravan
```

Data	
datos	5822 obs. of 86 variables

2. EJEMPLO K VECINOS MAS CERCANOS I

Paso 3. Si visualizamos las 6 primeras filas con los datos de las 86 columnas. Estimaremos la última columna llamada Purchase mediante el algoritmo de k vecinos más cercanos.

El cliente ha comprado o no ha comprado según las columnas anteriores. Estimaremos si el cliente compra o no compra el seguro, en función de esas características.

```
> datos = Caravan
> head(datos)
  MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE
1      33        1        3         2        8      0      5      1      3
2      37        1        2         2        8      1      4      1      4
3      37        1        2         2        8      0      4      2      4
4       9        1        3         3        3      2      3      2      4
5      40        1        4         2       10      1      4      1      4
6      23        1        2         1        5      0      5      0      5
  MRELGE MRELSA MRELOV MFALLEEN MFGEKIND MFWEKIND MOPLHOOG MOPLMIDD
1      7      0      2        1        2        6        1        2
2      6      2      2        0        4        5        0        5
3      3      2      4        4        4        2        0        5
4      5      2      2        2        3        4        3        4
5      7      1      2        2        4        4        5        4
6      0      6      3        3        5        2        0        5
  Purchase
1       No
2       No
3       No
4       No
5       No
6       No
```

2. EJEMPLO K VECINOS MAS CERCANOS I

Paso 4. Usamos la función str() para obtener más información sobre este dataset. Vemos que aparecen 5822 observaciones con 86 variables. La ultima la 86 es la de la compra o purchase, la que indica si ha comprado o no el seguro.

```
> str(datos)
'data.frame':   5822 obs. of  86 variables:
 $ MOSTYPE : num  33 37 37 9 40 23 39 33 33 11 ...
 $ MAANTHUI: num  1 1 1 1 1 1 2 1 1 2 ...
 $ MGEMOMV : num  3 2 2 3 4 2 3 2 2 3 ...
 $ MGEMLEEF: num  2 2 2 3 2 1 2 3 4 3 ...
 $ MOSHOOFD: num  8 8 8 3 10 5 9 8 8 3 ...
 $ MGODRK  : num  0 1 0 2 1 0 2 0 0 3 ...
 $ MGODPR  : num  5 4 4 3 4 5 2 7 1 5 ...
 $ MGODOV  : num  1 1 2 2 1 0 0 0 3 0 ...
 $ MGODGE  : num  3 4 4 4 4 5 5 2 6 2 ...
 $ MRELGE  : num  7 6 3 5 7 0 7 7 6 7 ...
 $ MRELSA  : num  0 2 2 2 1 6 2 2 0 0 ...
```

```
 $ ABRAND  : num  1 1 1 1 1 0 0 0 0 1 ...
 $ AZEILPL : num  0 0 0 0 0 0 0 0 0 0 ...
 $ APLEZIER: num  0 0 0 0 0 0 0 0 0 0 ...
 $ AFIETS  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ AINBOED : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ABYSTAND: num  0 0 0 0 0 0 0 0 0 0 ...
 $ Purchase: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
>
```

2. EJEMPLO K VECINOS MAS CERCANOS I

Paso 5. Mediante Summary podemos ver información de una columna concreta, la de compra. Vemos que hay 5474 que no han comprado y 348 que si han comprado

```
> summary(datos$Purchase)
  No  Yes
5474  348
>
```

Paso 6. Si queremos visualizar si el dataset tiene algún valor nulo, la hacemos con la función any

```
> any(is.na(datos))
[1] FALSE
>
```

Si es false es que no hay ningún valor nulo, con lo cual no tenemos que hacer ningún tipo de arreglo a este dataset antes de trabajar con él.

2. EJEMPLO K VECINOS MAS CERCANOS I

Paso 7. Vamos a estandarizar los datos. Primero vamos a asignar a la variable datos.compra, la columna 86, que es la columna de compra.

```
> datos.compra = datos[,86]
>
```

Paso 8. Por otro lado, vamos a estandarizar los datos mediante la función scale. Le pasamos todos los datos, menos la columna 86.

```
> datos.estandarizados = scale(datos[,-86])
>
```

Vamos a tener por un lado los datos de Purchase en datos.compra y el resto de las columnas en datos.estandarizados.

2. EJEMPLO K VECINOS MAS CERCANOS I

Paso 9. Ahora vamos a dividir los datos de nuestro dataset en datos de entrenamiento para entrenar nuestro modelo, y en datos de pruebas, para realizar las estimaciones.

Primero creamos la variable filas con los valores del 1 al 1000, por ejemplo. Si hacemos un head de filas, vemos que tiene los valores 1 2 0 4, 5, 6 y así hasta el 1000

```
> filas = 1:1000
> head(filas)
[1] 1 2 3 4 5 6
>
```

2. EJEMPLO K VECINOS MAS CERCANOS I

Paso 10. Utilizaremos la variable filas para poner las 1000 primeras filas para los datos de pruebas, y los demás para los datos de entrenamiento, tanto para los datos estandarizados como para los datos de la columna Purchase

```
> pruebas.datos = datos.estandarizados[filas,]
> pruebas.compra = datos.compra[filas]
>
```

```
> entrenamiento.datos = datos.estandarizados[-filas,]
> entrenamiento.compra = datos.compra[-filas]
>
```

De los datos estandarizados (los valores de todas las columnas menos la de compra), hemos seleccionado las primeras 1000 filas, y las hemos puesto en pruebas.datos

Y el resto, la 1001, 1002 hasta 5874 pasarían a ser datos de entrenamiento. Y para la columna Purchase, lo mismo.

3. EJEMPLO K VECINOS MAS CERCANOS II

Paso 1. Una vez hemos separado los datos de pruebas y el entrenamiento, vamos a ejecutar el modelo y obtener las predicciones. Para ello necesitamos cargar la librería class

```
> library(class)
>
```

Paso 2. Establecemos la semilla con valor 90. De esta manera se consigue que las pruebas que hagamos, den siempre el mismo resultado. Se puede poner siempre cualquier valor u omitir esta línea.

```
> set.seed(90)
>
```

3. EJEMPLO K VECINOS MAS CERCANOS II

Paso 3. Calculamos la variable prediccion.compra mediante la ejecución del modelo Knn, los k vecinos más cercanos, pasándole los datos de entrenamiento, las columnas características de los datos de prueba, el entrenamiento de compra y y k=1.

```
> prediccion.compra = knn(entrenamiento.datos,pruebas.datos, entrenamiento.compra, k=1)
>
```

Hemos obtenido las predicciones de la columna compra o purchase para estos datos de pruebas

3. EJEMPLO K VECINOS MAS CERCANOS II

Paso 4. Entonces, si queremos ver algunos valores, hacemos el head de predicion.compra

```
> head(prediccion.compra)
[1] No No No No No No
Levels: No Yes
>
```

Vemos los primeros valores y que hay solo dos valores posibles: No y Yes.

3. EJEMPLO K VECINOS MAS CERCANOS II

Paso 5. Ahora evaluaremos que el modelo funciona correctamente para este dataset, calculando el error. Si es bajo, el modelo es correcto/bueno para estos datos.

```
> error = mean(pruebas.compra!=prediccion.compra)
> error
[1] 0.116
>
```

El error sería la media de los valores de prueba de la columna objetivo compra distintos de la predicción compra.

En este caso, el modelo se equivoca el 11% de las veces .

Es un error que puede ser correcto para este modelo, pero ha sido con k=1, es decir, que se ha hecho la estimación teniendo en cuenta un único vecino, el vecino más cercano

3. EJEMPLO K VECINOS MAS CERCANOS II

Paso 6. En este caso si aumentamos el número de vecinos, posiblemente este error bajaría y nuestro modelo sería más óptimo.

Por ejemplo, si pusiéramos un valor de k=5, y calculamos nuevamente el error:

```
> prediccion.compra = knn(entrenamiento.datos,pruebas.datos, entrenamiento.compra, k=5)
> error = mean(pruebas.compra!=prediccion.compra)
> error
[1] 0.066
>
```

Veremos que ahora el error ha bajado a 0.066.

El error ha bajado de un 11% a un 6%, con lo cual nuestro modelo funciona mejor para k=5

3. EJEMPLO K VECINOS MAS CERCANOS II

Paso 7. ¿ Cuál sería el valor óptimo que tendríamos que poner en k para que nuestro modelo sea el mejor a la hora de obtener resultados?

Para ello vamos a hacer un bucle corriendo k de 1 hasta 20 y viendo cuál es el que tiene un error menor.

```
> prediccion.compra = NULL
> errores = NULL
> for (i in 1:20) {
+     set.seed(90)
+     prediccion.compra = knn(entrenamiento.datos, pruebas.datos, entrenamiento.compra, k=i)
+     errores[i] = mean(pruebas.compra != prediccion.compra)
+ }
>
```

Tarda un poco porque va a ejecutar 20 veces el modelo knn para k=1, 2, 3 asi hasta llegar a 20

Vemos que ha finalizado cuando sale el símbolo ">"

3. EJEMPLO K VECINOS MAS CERCANOS II

Paso 8. Una vez ha realizado las 20 primeras predicciones, en el vector errores tenemos todos los errores que hay. Hacemos un print de estos errores

```
> print(errores)
 [1] 0.116 0.117 0.074 0.073 0.066 0.065 0.062 0.061 0.058 0.059 0.059 0.059 0.059 0.059 0.059 0.059
[17] 0.059 0.059 0.059 0.059
>
```

3. EJEMPLO K VECINOS MAS CERCANOS II

Paso 9. Vamos a construir una tabla de errores para visualizar más fácilmente el error con el valor de k, para poder elegir cuál es el valor óptimo.

```
> valores.k = 1:20
> tabla.errores = data.frame(errores, valores.k)
> tabla.errores
   errores valores.k
1    0.116         1
2    0.117         2
3    0.074         3
4    0.073         4
5    0.066         5
6    0.065         6
7    0.062         7
8    0.061         8
9    0.058         9
10   0.059        10
11   0.059        11
12   0.059        12
13   0.059        13
14   0.059        14
15   0.059        15
16   0.059        16
17   0.059        17
18   0.059        18
19   0.059        19
20   0.059        20
> |
```

Según aumenta el valor de k, el valor del error va bajando: 0.61, 0.58, 0.59 y aquí se mantiene fijo.

El valor más bajo está en k=9.

Para este modelo de k vecinos más cercanos, y para este dataset, el valor K=9 es el mas optimo porque da el menor error posible.