Algoritmo k-Nearest Neighbors (kNN)

Alejandro Navas González

Fecha: 11 de septiembre, 2020

Contents

Introducción: Nearest Neighbors

Los clasificadores nearest neighbor se definen por su carcaterística de clasificar ejemplos no etiquetados asignándoles la clase de aquellos ejemplos etiquetados que son más similares. A pesar de lo simple de esta propuesta, los métodos nearest neighbor son muy poderosos. De hecho, se ha empleado este método para:

- Aplicaciones de visión por ordenador, incluyendo reconocimiento óptico de caracteres y reconocimiento facial.
- Predecir si una persona disfruta de una película que le han recomendado (Netflix).
- Identificar patrones en los datos genéticos, para su uso en la detección de proteínas o enfermedades específicas.

Este método es útil cuando se trabaja con clases numerosas, complejas y que de otra forma serían muy difíciles de manejar, pero con el requisito que los elementos dentro de una clase son similares, de que existe homogeneidad y claridad de distinción entre los grupos. Por contra, si no se ofrece una clara diferencia entre grupos, este algoritmo no se recomienda.

Algoritmo k-NN: Fortalezas y Debilidades

Las fortalezas y debilidades de este algoritmo son:

Fortalezas	Debilidades
Simple y efectivo	No produce un modelo, lo que limita la habilidad de encontrar nuevas relaciones entre las variables objeto de estudio.
No realiza supuestos sobre una distribución subyacente de los datos	Su fase de clasificación es lenta.
Tiene una fase de entrenamiento rápida	Requiere de una gran cantidad de memoria. Las variables nominales y los valores perdidos (NA) requieren de un procesamiento adicional.

Por último, como detalle a destacar, este algoritmo se dice que es vago o *lazy* porque, técnicamente hablando, no hay proceso de abstracción. Los procesos de abstracción y generalización ocurren simultáneamente. Esto permite que la fase de entrenamiento sea en extremo rápido, y conlleva la posible desventaja de que el proceso de predicción tienda a ser relativamente lento. Además, al basarse en casos concretos, no construye un modelo. Por ello se dice que el método se encuentra en una clase de **métodos de aprendizaje no paramétricos**, puesto que no se aprenden parámetros sobre los datos.

Cálculo de la Distancia

Para establecer el algoritmo k-NN se precisa de una función de distancias o una fórmula que permita medir la similaridad entre dos sujetos. Existen muchas formas de calcular las distancias. De forma tradicional, este algoritmo utiliza las **distancia euclídea**, que es una distancia que se daría uniendo mediante una regla dos puntos. Otra distancia habitual es la de **Manhattan**, que es aquella basada en los caminos que seguiría un peatón al caminar por las manzanas de la ciudad.

La **distancia euclídea** se especifica en la siguiente fórmula

$$dist(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

Elección de la K

Elegir cuántos vecinos seleccionar en el k-NN determinará lo bien que el modelo se pueda utilizar para generalizar datos en un futuro. El balance entre el ***overfitting*** o **sobreajuste** y el ***underfitting*** o **infraajuste** del conjunto de datos de entrenamiento es un problema conocido como **bias-variance tradeoff**. Elegir un valor de k alto reduce el impacto o variación causada por el ruido del conjunto de datos, pero el sesgo o *bias* que se puede cometer entonces es ignorar pequeños, pero relevantes, patrones (*underfitting*), y viceversa, esto es, tomar un valor de k pequeño permitirá explorar detalles dentro del conjunto de datos pero no permitirá una generalización (*overfitting*). Esto se explica en la medida en que al fijar una k muy grande, póngase que igual al total número de observaciones en los datos de entrenamiento, como cada instancia de entrenamiento es representada en la votación final, la clase de entrenamiento más común siempre tiene la mayoría de los votantes. El modelo, por lo tanto, siempre predeciría la clase mayoritaria, independientemente de qué vecinos están más cerca. En el extremo opuesto, el uso de un solo vecino más cercano permite que datos ruidosos o valores atípicos puedan influir indebidamente en la clasificación de los sujetos. Por ejemplo, supóngase que algunos de los sujetos del marco de entrenamiento fueron accidentalmente mal etiquetados. Cualquier ejemplo no etiquetado que esté más cerca del vecino incorrectamente etiquetado se predecirá que pertenece a la clase incorrecta, incluso si los otros nueve vecinos más cercanos presentaran una clase diferente. Así pues, siempre convendrá encontrar el mejor k en algún punto entre ambos extremos.

Diagnóstico mediante kNN

En este ejemplo se va a investigar la utilidad del Machine Learning para la detección del cáncer de mama aplicando un algoritmo k-NN a medidas de biopsisas de mujeres que presentaron masas anormales en los pechos.

Recolección de los Datos

Se utilizará el marco de datos de **Breast Cancer Wisconsin Diagnostic** del repositorio de la UCI para Machine Learning, el cual está disponible en el siguiente **respositorio**. Este conjunto de datos fue donado por investigadores de la **Universidad de Wisconsin** y presenta medidas de imágenes digitales de masas extraídas de pecho. Este marco incluye un total de 569 biopsisas y 32 variables. Una de estas variables es el ID, otra el diagnóstico y las treinta restantes son medidas cuantitativas llevadas a cabo en laboratorio. El diagnóstico es tal que se utiliza **M** para indicar maligno y **B** para indicar beningno.

Exploración & Preparación

En primer lugar, se importará el archivo csv con los datos de Wisconsin.

```
Wisc<-read.csv(file.path(params$folder.data, params$file1), stringsAsFactors = TRUE)
```

A continuación, se estudiará la estructura del marco de datos, que habrá de contar con 569 observaciones y 32 variables.

```
head(Wisc)
```

```
str(Wisc)
```

```
'data.frame':   569 obs. of  32 variables:
 $ id                : int  87139402 8910251 905520 868871 9012568 906539 925291 87880 862989 89827 ...
 $ diagnosis         : Factor w/ 2 levels "B","M": 1 1 1 1 1 1 1 2 1 1 ...
 $ radius_mean       : num  12.3 10.6 11 11.3 15.2 ...
 $ texture_mean      : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean    : num  78.8 69.3 70.9 73 97.7 ...
 $ area_mean         : num  464 346 373 385 712 ...
 $ smoothness_mean   : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
 $ compactness_mean  : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
 $ concavity_mean    : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
 $ points_mean       : num  0.037 0.0264 0.0248 0.048 0.0266 ...
 $ symmetry_mean     : num  0.196 0.192 0.171 0.177 0.172 ...
 $ dimension_mean    : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
 $ radius_se         : num  0.236 0.451 0.197 0.338 0.178 ...
 $ texture_se        : num  0.666 1.197 1.387 1.343 0.412 ...
 $ perimeter_se      : num  1.67 3.43 1.34 1.85 1.34 ...
 $ area_se           : num  17.4 27.1 13.5 26.3 17.7 ...
 $ smoothness_se     : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
 $ compactness_se    : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
 $ concavity_se      : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
 $ points_se         : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
 $ symmetry_se       : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
 $ dimension_se      : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
 $ radius_worst      : num  13.5 11.9 12.4 11.9 16.2 ...
 $ texture_worst     : num  15.6 22.9 26.4 15.8 15.7 ...
 $ perimeter_worst   : num  87 78.3 79.9 76.5 104.5 ...
 $ area_worst        : num  549 425 471 434 819 ...
 $ smoothness_worst  : num  0.139 0.121 0.137 0.137 0.113 ...
 $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
 $ concavity_worst   : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
 $ points_worst      : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
 $ symmetry_worst    : num  0.283 0.294 0.3 0.21 0.249 ...
 $ dimension_worst   : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
```

La primera de las variabes es el ID o identificador único del paciente. Esta variable no aporta información

útil, por lo que es preciso excluirla del modelo.

```
# ID está en la primera columna. Se procede a su eliminación.
rownames(Wisc)<-Wisc[,1]
Wisc<-Wisc[, -1]
# Se comprueba que se ha eliminado la variable ID.
head(Wisc)
```

```
str(Wisc)
```

```
'data.frame':   569 obs. of  31 variables:
 $ diagnosis        : Factor w/ 2 levels "B","M": 1 1 1 1 1 1 1 2 1 1 ...
 $ radius_mean      : num  12.3 10.6 11 11.3 15.2 ...
 $ texture_mean     : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean   : num  78.8 69.3 70.9 73 97.7 ...
 $ area_mean        : num  464 346 373 385 712 ...
 $ smoothness_mean  : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
 $ compactness_mean : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
 $ concavity_mean   : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
 $ points_mean      : num  0.037 0.0264 0.0248 0.048 0.0266 ...
 $ symmetry_mean    : num  0.196 0.192 0.171 0.177 0.172 ...
 $ dimension_mean   : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
 $ radius_se        : num  0.236 0.451 0.197 0.338 0.178 ...
 $ texture_se       : num  0.666 1.197 1.387 1.343 0.412 ...
 $ perimeter_se     : num  1.67 3.43 1.34 1.85 1.34 ...
 $ area_se          : num  17.4 27.1 13.5 26.3 17.7 ...
 $ smoothness_se    : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
 $ compactness_se   : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
 $ concavity_se     : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
 $ points_se        : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
 $ symmetry_se      : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
 $ dimension_se     : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
 $ radius_worst     : num  13.5 11.9 12.4 11.9 16.2 ...
 $ texture_worst    : num  15.6 22.9 26.4 15.8 15.7 ...
 $ perimeter_worst  : num  87 78.3 79.9 76.5 104.5 ...
 $ area_worst       : num  549 425 471 434 819 ...
 $ smoothness_worst : num  0.139 0.121 0.137 0.137 0.113 ...
 $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
 $ concavity_worst  : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
 $ points_worst     : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
 $ symmetry_worst   : num  0.283 0.294 0.3 0.21 0.249 ...
 $ dimension_worst  : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
```

La variable *diagnosis* es de peculiar interés, ya que es aquello que se pretende predecir. Se puede ver cuántos sujetos tienen tumores beningnos frente a aquellos que tienen masas malignas. Por otro lado, muchos clasificadores de Machine Learning precisan de estar codificados como *factor*, por lo que es necesario también recodificar la variable *diagnosis* en caso de que no se presentara como *factor*. En esta situación dicha tarea se ha realizado al importar el marco de datos, si bien se añadirá este paso para generar etiquetas más informativas sobre los valores que puede tomar esta variable.

```
# Frecuencias absolutas.
table(Wisc$diagnosis)
```

```
  B   M
357 212
```

```r
# Frecuencias relativas.
prop.table(table(Wisc$diagnosis))
```

```
        B         M
0.6274165 0.3725835
```

```r
# Recodificación.
Wisc$diagnosis<-factor(Wisc$diagnosis, levels = c("B", "M"),
                       labels = c("Benigno", "Maligno"))
# Comprobación de la recodificación.
round(prop.table(table(Wisc$diagnosis)) * 100, digits = 1)
```

```
Benigno Maligno
   62.7    37.3
```

El resto de variables son todas numéricas y consisten en tres medidas diferentes de diez características. Se van a estudiar detalladamente los estadísticos básicos de tres de ellas.

```r
summary(Wisc[c("radius_mean", "area_mean", "smoothness_mean")])
```

```
  radius_mean       area_mean      smoothness_mean
 Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
 1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
 Median :13.370   Median : 551.1   Median :0.09587
 Mean   :14.127   Mean   : 654.9   Mean   :0.09636
 3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
 Max.   :28.110   Max.   :2501.0   Max.   :0.16340
```

Observando los resultados de estos estadísticos se tiene un problema, pues mientras que la media de *smoothness* va de 0.05 a 0.16, la media del área va de 143,5 a 2501.0, de modo que el impacto del área será muy superior al de *smoothness* al calcular las distancias. Por este motivo es preciso aplicar una normalización, para reescalar de las características a un rango estándar de valores.

Transformación. La Normalización de los Datos Numéricos

Para llevar a cabo la normalización es preciso crear una función en R que lo haga. Esta función tomará el conjunto de valores de un vector x y para cada valor en x, le restará el valor mínimo de x y lo dividirá por el rango de x. Finalmente, devolverá el vector resultante de la transformaión.

```r
# Se crea la función para normalizar.
Normalizar<-function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}
# Se aplica a las 30 variables cuantitativas.
Wisc.Normalizado<-as.data.frame(lapply(Wisc[2:31], Normalizar))

# Se confirma que la normalización se ha efectuado correctamente.
summary(Wisc.Normalizado$area_mean)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.1174  0.1729  0.2169  0.2711  1.0000
```

Preparación de los Datos. Los Marcos de Entrenamiento & Prueba

En primer lugar, ante la ausencia de muestras cuyo status nos sea desconocido para testear el modelo, se va a simular dicho escenario mediante la división de nuestro marco en dos partes. Una de ellas, el conjunto de

entrenamiento, será utilizada para crear el modelo kNN, y la otra, el conjunto de prueba o testeo, que será utilizado para ver la precisión predictiva del modelo creado. De los 569 sujetos, cien serán utilizados para el conjunto de testeo y 469 para el de entrenamiento.

```
Wisc_train <- Wisc.Normalizado[1:469, ]
Wisc_test <- Wisc.Normalizado[470:569, ]
```

Al construir estos marcos de datos se ha de excluir además la variable diana, la que se quiere explicar, séase, *diagnosis*. Para el entrenamiento del modelo kNN se almacenará esta variable como un vector de tipo factor, dividido según los marcos.

```
Wisc_train_labels<-Wisc[1:469, 1]
Wisc_test_labels<-Wisc[470:569, 1]
```

Entrenamiento del Modelo

Para los algoritmos kNN, la fase de entrenamiento no involucra la creación del modelo, sino el almacenamiento de los datos de entrada en un formato estructurado. Para realizar la clasificación de nuestro marco de prueba, se utilizará una implementacioón kNN del paquete **class**. Concretamente la función *knn()* de este paquete ofrece una implementación estándar o clásica de un algoritmo kNN, utilizando las distancias euclídeas.

Con todo dispuesto, lo único que falta es especificar qué k es la apropiada. Como nuestro marco de entrenamiento ofrece 469 sujetos, se probará en primera instancia con un k=21, siendo 21 la raíz cuadrada de 469 si se truncan los decimales (realmente la raíz cuadrada de 469 es 21.656, pero dicho valor no se puede emplear como k). Por otro lado, usar un número impar reducirá la posibilidad de terminar con un voto de empate. Ahora, sí se está en disposición de aplicar el algoritmos kNN.

```
library(class)
Wisc_test_pred<-knn(train = Wisc_train, test = Wisc_test, cl = Wisc_train_labels, k=21)
```

Evaluación del Modelo

El próximo paso es la evaluación del modelo para saber cómo de bien las clases predichas en Wisc_test_pred se ajustan con los valores conocidos en Wisc_test_labels. Para realizar dicha evaluación se puede usar *CrossTable()*, una función del paquete **gmodels**.

```
library(gmodels)
CrossTable(x=Wisc_test_labels, y=Wisc_test_pred, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                 | Wisc_test_pred
Wisc_test_labels |   Benigno |   Maligno | Row Total |
-----------------|-----------|-----------|-----------|
         Benigno |        61 |         0 |        61 |
```

```
                 |     1.000 |     0.000 |     0.610 |
                 |     0.968 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
         Maligno |         2 |        37 |        39 |
                 |     0.051 |     0.949 |     0.390 |
                 |     0.032 |     1.000 |           |
                 |     0.020 |     0.370 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        63 |        37 |       100 |
                 |     0.630 |     0.370 |           |
-----------------|-----------|-----------|-----------|
```

Los resultados obtenidos indican lo siguiente. En la celda superior izquierda se tienen los verdaderos negativos. Se han obtenido 61 de 100 sujetos con masas benignas, y los kNN los han identificado como tal. En la celda inferior derecha se indican los verdaderos positivos, aquellos sujetos cuyos tumores se han identificados como malignos. Se han obtenido 37 verdaderos positivos. Los dos sujetos obtenidos en la parte inferior izquierda son falsos negativos, sujetos que se predice que tienen tumores benignos, pero realmente presentan tumores malignos. Los errores de esta clase son extremadamente peligrosos, ya que llevan al paciente a creer que no tiene cáncer cuando realmente éste avanza. Por otro lado, los sujetos de la celda superior derecha son los falsos positivos, que son aquellos sujetos clasificados como de tumores malignos cuando realmente su diagnóstico es benigno. En este caso no se han obtenido falsos positivos. A pesar de que este segundo tipo de error es menos peligroso que el del falso negativo, se ha evitar ya que supone un coste adicional, una carga para el sistema sanitario y/o un estrés adicional para el paciente.

Mejora del Modelo

El resultado obtenido se puede mejorar y reducir la tasa de falsos negativos. Para ello se van a aplicar dos variaciones respecto del clasificador previo. Primero, se va a utilizar un método alternativo para reescalar las variables numéricas. Segundo, se utilizarán otros valores de k.

Transformación Alternativa: Estandarización por el Z-Score

Aunque la normalización es el método tradicional utilizado para la clasificación vía kNN, no siempre es la opción más apropiada. Al no predefinir un mínimo y un máximo, los valores estandarizados por el z-score no presentan valores extremos comprimidos hacia los valores centrales. Se puede pensar que para un tumor maligno es probable que se prersenten valores atípicos, pues el tumor crece descontroladamente, de tal modo que es razonable permitir que los valores atípicos pesen más en el cálculo de las distancias. Con este sistema de estandarización se puede comprobar si mejora la precisión del modelo. Para llevarla a cabo se recurre a la función scale(), que permite, por defecto, una estandarización vía z-score. Esta función permite trabajar sobre un data frame, de modo que no es necesaria la función lapply.

```
# Se estandafrizan todas las columnas menos la de diagnóstico.
Wisc.Z<-as.data.frame(scale(Wisc[, -c(1)]))
# Se confirma la transformación.
summary(Wisc.Z$area_mean)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.4532 -0.6666 -0.2949  0.0000  0.3632  5.2459
```

La media de una transfromación Z-score siempre ha de ser nula y el rango compacto. Un valor Z-score mayor que tres o menor que menos tres indica valores en extremo extraños. Se procede tras esta verificación a realizar dividir de nuevo el marco de datos en un set de prueba y otro de entrenamiento y en aplicar el algoritmo k-NN.

```r
Wisc_train.Z<-Wisc.Z[1:469, ]
Wisc_test.Z<-Wisc.Z[470:569, ]
# El paso de crear los vectores de etiquetas no es necesario, pues son los mismos.
Wisc_test_pred.Z <- knn(train = Wisc_train.Z, test = Wisc_test.Z,
cl = Wisc_train_labels, k=21)
CrossTable(x = Wisc_test_labels, y = Wisc_test_pred.Z, prop.chisq=FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                 | Wisc_test_pred.Z
Wisc_test_labels |   Benigno |   Maligno | Row Total |
-----------------|-----------|-----------|-----------|
         Benigno |        61 |         0 |        61 |
                 |     1.000 |     0.000 |     0.610 |
                 |     0.924 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
         Maligno |         5 |        34 |        39 |
                 |     0.128 |     0.872 |     0.390 |
                 |     0.076 |     1.000 |           |
                 |     0.050 |     0.340 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        66 |        34 |       100 |
                 |     0.660 |     0.340 |           |
-----------------|-----------|-----------|-----------|
```

Desafortunadamente, el resultado obtenido es una pérdida de precisión del test, han aumentado los falsos negativos, no se ha conseguido su mejora.

Valores Alternativos de K

Quizá se obtiene una mejora probando valores alternativos de k. Sobre esta base, recurriendo a los marcos de datos de prueba y entrenamiento normalizados, se aplican a continuación diferentes valores de k.

```r
Wisc_test_pred.1<-knn(train = Wisc_train, test = Wisc_test, cl = Wisc_train_labels, k=1)
CrossTable(x=Wisc_test_labels, y=Wisc_test_pred.1, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
```

```
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                 | Wisc_test_pred.1
Wisc_test_labels |   Benigno |   Maligno | Row Total |
-----------------|-----------|-----------|-----------|
         Benigno |        58 |         3 |        61 |
                 |     0.951 |     0.049 |     0.610 |
                 |     0.983 |     0.073 |           |
                 |     0.580 |     0.030 |           |
-----------------|-----------|-----------|-----------|
         Maligno |         1 |        38 |        39 |
                 |     0.026 |     0.974 |     0.390 |
                 |     0.017 |     0.927 |           |
                 |     0.010 |     0.380 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        59 |        41 |       100 |
                 |     0.590 |     0.410 |           |
-----------------|-----------|-----------|-----------|
```

```
Wisc_test_pred.5<-knn(train = Wisc_train, test = Wisc_test, cl = Wisc_train_labels, k=5)
CrossTable(x=Wisc_test_labels, y=Wisc_test_pred.5, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                 | Wisc_test_pred.5
Wisc_test_labels |   Benigno |   Maligno | Row Total |
-----------------|-----------|-----------|-----------|
         Benigno |        61 |         0 |        61 |
                 |     1.000 |     0.000 |     0.610 |
                 |     0.968 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
         Maligno |         2 |        37 |        39 |
```

```
                |     0.051 |     0.949 |     0.390 |
                |     0.032 |     1.000 |           |
                |     0.020 |     0.370 |           |
----------------|-----------|-----------|-----------|
    Column Total |        63 |        37 |       100 |
                |     0.630 |     0.370 |           |
----------------|-----------|-----------|-----------|
```

```
Wisc_test_pred.11<-knn(train = Wisc_train, test = Wisc_test, cl = Wisc_train_labels, k=11)
CrossTable(x=Wisc_test_labels, y=Wisc_test_pred.11, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                | Wisc_test_pred.11
Wisc_test_labels |   Benigno |   Maligno | Row Total |
-----------------|-----------|-----------|-----------|
         Benigno |        61 |         0 |        61 |
                 |     1.000 |     0.000 |     0.610 |
                 |     0.953 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
         Maligno |         3 |        36 |        39 |
                 |     0.077 |     0.923 |     0.390 |
                 |     0.047 |     1.000 |           |
                 |     0.030 |     0.360 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        64 |        36 |       100 |
                 |     0.640 |     0.360 |           |
-----------------|-----------|-----------|-----------|
```

```
Wisc_test_pred.15<-knn(train = Wisc_train, test = Wisc_test, cl = Wisc_train_labels, k=15)
CrossTable(x=Wisc_test_labels, y=Wisc_test_pred.15, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
```

```
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                 | Wisc_test_pred.15
Wisc_test_labels |   Benigno |   Maligno | Row Total |
-----------------|-----------|-----------|-----------|
         Benigno |        61 |         0 |        61 |
                 |     1.000 |     0.000 |     0.610 |
                 |     0.953 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
         Maligno |         3 |        36 |        39 |
                 |     0.077 |     0.923 |     0.390 |
                 |     0.047 |     1.000 |           |
                 |     0.030 |     0.360 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        64 |        36 |       100 |
                 |     0.640 |     0.360 |           |
-----------------|-----------|-----------|-----------|
```

```
Wisc_test_pred.27<-knn(train = Wisc_train, test = Wisc_test, cl = Wisc_train_labels, k=27)
CrossTable(x=Wisc_test_labels, y=Wisc_test_pred.27, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                 | Wisc_test_pred.27
Wisc_test_labels |   Benigno |   Maligno | Row Total |
-----------------|-----------|-----------|-----------|
         Benigno |        61 |         0 |        61 |
                 |     1.000 |     0.000 |     0.610 |
                 |     0.938 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
         Maligno |         4 |        35 |        39 |
                 |     0.103 |     0.897 |     0.390 |
                 |     0.062 |     1.000 |           |
```

```
                 |     0.040 |     0.350 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        65 |        35 |       100 |
                 |     0.650 |     0.350 |           |
-----------------|-----------|-----------|-----------|
```

A pesar de que el clasificador en ningún caso es perfecto, el enfoque 1NN es capaz de evitar uno de los dos falsos negativos a expensas de introducir tres falsos positivos. Sin embargo, es importante tener en cuenta que no sería prudente adaptar el enfoque demasiado a la evaluación del rendimiento sobre el marco de prueba y tomar el 1NN a la ligera, pues es probable que un conjunto diferente de 100 registros de pacientes sea algo distinto de los que se utilizan para medir el rendimiento de nuestro modelo, es decir, es más que probable que al optar por un valor de k de uno se esté pecando de sobreajuste a los datos del set empleado para la evaluación.

Bibliografía

Lantz, Brett. 2015. *Machine Learning with R*. Packt Publishing Ltd. http://www.packtpub.com/books/content/machine-learning-r.