Otro ejemplo de clasificación con k-NN

k-Nearest Neighbors (k-NN) es un algoritmo que es útil para hacer clasificaciones/predicciones cuando existen posibles límites no lineales que separan clases o valores de interés.

Conceptualmente, k-NN examina las clases/valores de los puntos a su alrededor (es decir, sus vecinos) para determinar el valor del punto de interés. El valor mayoritario o medio se asignará al punto de interés.

Nota: Usamos la clasificación k-NN al predecir un resultado categórico y la regresión k-NN al predecir un resultado continuo.

Paso 1: Leer los datos y los paquetes necesarios

```
> data <- read.table("student-mat.csv",sep=";",header=TRUE)
> var.names.data <-tolower(colnames(data))
> colnames(data) <- var.names.data
> head(data)
```

	school	sex	age	address	famsize	pstatus	medu	fedu	mjob	fjob	reason	guardian	traveltime	studytime
1	GP	F	18	U	GT3	A	4	4	at_home	teacher	course	mother	2	2
2	GP	F	17	U	GT3	T	1	1	at_home	other	course	father	1	2
3	GP	F	15	U	LE3	T	1	1	at_home	other	other	mother	1	2
4	GP	F	15	U	GT3	T	4	2	health	services	home	mother	1	3
5	GP	F	16	U	GT3	T	3	3	other	other	home	father	1	2
6	GP	M	16	U	LE3	T	4	3	services	other	reputation	mother	1	2

	failures	schoolsup	famsup	paid	activities	nursery	higher	internet	romantic	famrel	freetime	goout	dalc	walc	health
1	0	yes	no	no	no	yes	yes	no	no	4	3	4	1	1	3
2	0	no	yes	no	no	no	yes	yes	no	5	3	3	1	1	3
3	3	yes	no	yes	no	yes	yes	yes	no	4	3	2	2	3	3
4	0	no	yes	yes	yes	yes	yes	yes	yes	3	2	2	1	1	5
5	0	no	yes	yes	no	yes	yes	no	no	4	3	2	1	2	5
6	0	no	yes	yes	yes	yes	yes	yes	no	5	4	2	1	2	5

	absences	g1	g2	g3
1	6	5	6	6
2	4	5	5	6
3	10	7	8	10
4	2	15	14	15
5	4	6	10	10
6	10	15	15	15

Se necesitan las siguientes librerías, sino están instaladas, instalas de la forma acostumbrada.

```
> library(caret)
> library(class)
> library(dplyr)
> library(e1071)
> library(FNN)
```

```
> library(gmodels)
> library(psych)
```

Paso 2. Clasificación k-NN

Para la clasificación k-NN, vamos a predecir el trabajo de la variable categórica mother's job ("mjob") utilizando todas las demás variables dentro del conjunto de datos.

Preparación de datos

Haremos una copia de nuestro conjunto de datos para que podamos prepararlo para nuestra clasificación k-NN.

```
> data_class <- data
```

A continuación, colocaremos nuestra variable de resultado, mother's job ("mjob"), en su propio objeto y lo eliminaremos del conjunto de datos.

```
> # put outcome in its own object
> mjob_outcome <- data_class %>% select(mjob)
> # remove original variable from the data set
> data_class <- data_class %>% select(-mjob)
```

Ten en cuenta que debido a que k-NN implica calcular distancias entre puntos de datos, debemos usar solo variables numéricas. Esto solo se aplica a las variables predictoras. La variable de resultado para la clasificación k-NN debe seguir siendo una variable de factor.

Primero, escalamos los datos en caso de que nuestras características estén en diferentes métricas. Por ejemplo, si tuviéramos "ingresos (income)" como variable, estaría en una escala mucho mayor que "edad (age)", lo que podría ser problemático dado que k-NN depende de las distancias. Ten en cuenta que estamos usando la función de "escala (scale)" aquí, lo que significa que estamos escalando a una métrica de puntuación z (z-score).

Determina qué variables son números enteros

```
> str(data_class)
```

Vemos que las variables "age," "medu," "fedu," "traveltime," "studytime," "failures," "famrel," "freetime," "goout," "dalc," "walc," "health," "absences," "g1," "g2," y "g3" son variables enteras, lo que significa que se pueden escalar.

```
'data.frame':   395 obs. of  32 variables:
 $ school    : chr  "GP" "GP" "GP" "GP" ...
 $ sex       : chr  "F" "F" "F" "F" ...
 $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
 $ address   : chr  "U" "U" "U" "U" ...
 $ famsize   : chr  "GT3" "GT3" "LE3" "GT3" ...
 $ pstatus   : chr  "A" "T" "T" "T" ...
 $ medu      : int  4 1 1 4 3 4 2 4 3 3 ...
 $ fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
 $ fjob      : chr  "teacher" "other" "other" "services" ...
 $ reason    : chr  "course" "course" "other" "home" ...
 $ guardian  : chr  "mother" "father" "mother" "mother" ...
 $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
 $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
 $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
 $ schoolsup : chr  "yes" "no" "yes" "no" ...
 $ famsup    : chr  "no" "yes" "no" "yes" ...
 $ paid      : chr  "no" "no" "yes" "yes" ...
 $ activities: chr  "no" "no" "no" "yes" ...
 $ nursery   : chr  "yes" "no" "yes" "yes" ...
 $ higher    : chr  "yes" "yes" "yes" "yes" ...
 $ internet  : chr  "no" "yes" "yes" "yes" ...
 $ romantic  : chr  "no" "no" "no" "yes" ...
 $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
 $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
 $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
 $ dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
 $ walc      : int  1 1 3 1 2 2 1 1 1 1 ...
 $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
 $ absences  : int  6 4 10 2 4 10 0 6 0 0 ...
 $ g1        : int  5 5 7 15 6 15 12 6 16 14 ...
 $ g2        : int  6 5 8 14 10 15 12 5 18 15 ...
 $ g3        : int  6 6 10 15 10 15 11 6 19 15 ...
```

```
> data_class[, c("age", "medu", "fedu", "traveltime", "studytime", "failures", "famrel", "freetime", "goout", "dalc", "walc",
+ "health", "absences", "g1", "g2", "g3")] <- scale(data_class[, c("age", "medu", "fedu", "traveltime", "studytime", "failures",
+ "famrel", "freetime", "goout", "dalc", "walc", "health", "absences", "g1", "g2", "g3")])
> head(data_class)
```

```
  school sex        age address famsize pstatus       medu       fedu     fjob     reason guardian traveltime
1     GP   F  1.0217506       U     GT3       A  1.1424068  1.3586476  teacher     course   mother  0.7912473
2     GP   F  0.2380778       U     GT3       T -1.5979820 -1.3981972    other     course   father -0.6424347
3     GP   F -1.3292678       U     LE3       T -1.5979820 -1.3981972    other      other   mother -0.6424347
4     GP   F -1.3292678       U     GT3       T  1.1424068 -0.4792490 services       home   mother -0.6424347
5     GP   F -0.5455950       U     GT3       T  0.2289439  0.4396993    other       home   father -0.6424347
6     GP   M -0.5455950       U     LE3       T  1.1424068  0.4396993    other reputation   mother -0.6424347
    studytime   failures schoolsup famsup paid activities nursery higher internet romantic      famrel   freetime
1 -0.04223229 -0.4493737       yes     no   no         no     yes    yes       no       no  0.06211528 -0.2357113
2 -0.04223229 -0.4493737        no    yes   no         no      no    yes      yes       no  1.17736694 -0.2357113
3 -0.04223229  3.5847768       yes     no  yes         no     yes    yes      yes       no  0.06211528 -0.2357113
4  1.14932149 -0.4493737        no    yes  yes        yes     yes    yes      yes      yes -1.05313638 -1.2368505
5 -0.04223229 -0.4493737        no    yes  yes         no     yes    yes       no       no  0.06211528 -0.2357113
6 -0.04223229 -0.4493737        no    yes  yes        yes     yes    yes      yes       no  1.17736694  0.7654280
        goout       dalc       walc     health    absences        g1         g2         g3
1  0.80046413 -0.5400138 -1.0025178 -0.3987837  0.03637833 -1.780209 -1.2532017 -0.96371171
2 -0.09778397 -0.5400138 -1.0025178 -0.3987837 -0.21352497 -1.780209 -1.5190528 -0.96371171
3 -0.99603207  0.5826465  0.5504019 -0.3987837  0.53618492 -1.177653 -0.7214996 -0.09062427
4 -0.99603207 -0.5400138 -1.0025178  1.0397512 -0.46342827  1.232570  0.8736068  1.00073503
5 -0.99603207 -0.5400138 -0.2260579  1.0397512 -0.21352497 -1.478931 -0.1897975 -0.09062427
6 -0.99603207 -0.5400138 -0.2260579  1.0397512  0.53618492  1.232570  1.1394578  1.00073503
```

En segundo lugar, necesitamos codificar de forma ficticia cualquier factor o variable categórica.

Examina la estructura de los datos para determinar qué variables deben codificarse de forma ficticia.

> str(data_class)

```
'data.frame':   395 obs. of  32 variables:
 $ school    : chr  "GP" "GP" "GP" "GP" ...
 $ sex       : chr  "F" "F" "F" "F" ...
 $ age       : num  1.022 0.238 -1.329 -1.329 -0.546 ...
 $ address   : chr  "U" "U" "U" "U" ...
 $ famsize   : chr  "GT3" "GT3" "LE3" "GT3" ...
 $ pstatus   : chr  "A" "T" "T" "T" ...
 $ medu      : num  1.142 -1.598 -1.598 1.142 0.229 ...
 $ fedu      : num  1.359 -1.398 -1.398 -0.479 0.44 ...
 $ fjob      : chr  "teacher" "other" "other" "services" ...
 $ reason    : chr  "course" "course" "other" "home" ...
 $ guardian  : chr  "mother" "father" "mother" "mother" ...
 $ traveltime: num  0.791 -0.642 -0.642 -0.642 -0.642 ...
 $ studytime : num  -0.0422 -0.0422 -0.0422 1.1493 -0.0422 ...
 $ failures  : num  -0.449 -0.449 3.585 -0.449 -0.449 ...
 $ schoolsup : chr  "yes" "no" "yes" "no" ...
 $ famsup    : chr  "no" "yes" "no" "yes" ...
 $ paid      : chr  "no" "no" "yes" "yes" ...
 $ activities: chr  "no" "no" "no" "yes" ...
 $ nursery   : chr  "yes" "no" "yes" "yes" ...
 $ higher    : chr  "yes" "yes" "yes" "yes" ...
 $ internet  : chr  "no" "yes" "yes" "yes" ...
 $ romantic  : chr  "no" "no" "no" "yes" ...
 $ famrel    : num  0.0621 1.1774 0.0621 -1.0531 0.0621 ...
 $ freetime  : num  -0.236 -0.236 -0.236 -1.237 -0.236 ...
 $ goout     : num  0.8005 -0.0978 -0.996 -0.996 -0.996 ...
 $ dalc      : num  -0.54 -0.54 0.583 -0.54 -0.54 ...
 $ walc      : num  -1.003 -1.003 0.55 -1.003 -0.226 ...
 $ health    : num  -0.399 -0.399 -0.399 1.04 1.04 ...
 $ absences  : num  0.0364 -0.2135 0.5362 -0.4634 -0.2135 ...
 $ g1        : num  -1.78 -1.78 -1.18 1.23 -1.48 ...
 $ g2        : num  -1.253 -1.519 -0.721 0.874 -0.19 ...
 $ g3        : num  -0.9637 -0.9637 -0.0906 1.0007 -0.0906 ...
```

Podemos ver que las variables “fjob”, “reason” y “guardian” son variables factoriales que tienen tres o más niveles, y que las variables “school”, “sex”, “address”, “famsize”, “pstatus”, “schoolsup”, “famsup”, “paid”, “activities”, “nursery”, “higher”, “internet” y “romantic” son variables factoriales que tienen solo dos niveles.

Ahora codificamos variables ficticias que tienen solo dos niveles y están codificadas 1/0.

```
> data_class$schoolsup <- ifelse(data_class$schoolsup == "yes", 1, 0)
> data_class$famsup <- ifelse(data_class$famsup == "yes", 1, 0)
> data_class$paid <- ifelse(data_class$paid == "yes", 1, 0)
> data_class$activities <- ifelse(data_class$activities == "yes", 1, 0)
```

```
> data_class$nursery <- ifelse(data_class$nursery == "yes", 1, 0)
> data_class$higher <- ifelse(data_class$higher == "yes", 1, 0)
> data_class$internet <- ifelse(data_class$internet == "yes", 1, 0)
> data_class$romantic <- ifelse(data_class$romantic == "yes", 1, 0)
```

Luego, variables de código ficticio que tienen dos niveles, pero no son numéricas.

```
> data_class$school <- dummy.code(data_class$school)
> data_class$sex <- dummy.code(data_class$sex)
> data_class$address <- dummy.code(data_class$address)
> data_class$famsize <- dummy.code(data_class$famsize)
> data_class$pstatus <- dummy.code(data_class$pstatus)
```

A continuación, simulamos variables de código que tienen tres o más niveles.

```
> fjob <- as.data.frame(dummy.code(data_class$fjob))
> reason <- as.data.frame(dummy.code(data_class$reason))
> guardian <- as.data.frame(dummy.code(data_class$guardian))
```

Cambiamos el nombre de las columnas "other" en "fjob", "reason" y "guardian", y cambiamos el nombre de "health" en "fjob" (para que no tengamos columnas duplicadas más adelante).

```
> fjob <- rename(fjob, other_fjob = other)
> fjob <- rename(fjob, health_fjob = health)
> reason <- rename(reason, other_reason = other)
> guardian <- rename(guardian, other_guardian = other)
```

Combinamos nuevas variables ficticias con el conjunto de datos original.

```
> data_class <- cbind(data_class, fjob, guardian, reason)
```

Eliminamos las variables originales que tenían que codificarse de forma ficticia.

```
> data_class <- data_class %>% select(-one_of(c("fjob", "guardian", "reason")))
> head(data_class)
```

```
  school.GP school.MS sex.F sex.M        age address.U address.R famsize.GT3 famsize.LE3 pstatus.T pstatus.A
1         1         0     1     0  1.0217506         1         0           1           0         0         1
2         1         0     1     0  0.2380778         1         0           1           0         1         0
3         1         0     1     0 -1.3292678         1         0           0           1         1         0
4         1         0     1     0 -1.3292678         1         0           1           0         1         0
5         1         0     1     0 -0.5455950         1         0           1           0         1         0
6         1         0     0     1 -0.5455950         1         0           0           1         1         0
       medu       fedu traveltime   studytime   failures schoolsup famsup paid activities nursery higher internet
1 1.1424068  1.3586476  0.7912473 -0.04223229 -0.4493737         1      0    0          0       1      1        0
2 -1.5979820 -1.3981972 -0.6424347 -0.04223229 -0.4493737         0      1    0          0       0      1        1
3 -1.5979820 -1.3981972 -0.6424347 -0.04223229  3.5847768         1      0    1          0       1      1        1
4 1.1424068 -0.4792490 -0.6424347  1.14932149 -0.4493737         0      1    1          1       1      1        1
5 0.2289439  0.4396993 -0.6424347 -0.04223229 -0.4493737         0      1    1          0       1      1        0
6 1.1424068  0.4396993 -0.6424347 -0.04223229 -0.4493737         0      1    1          1       1      1        1
  romantic      famrel   freetime      goout       dalc       walc     health    absences        g1         g2
1        0  0.06211528 -0.2357113  0.80046413 -0.5400138 -1.0025178 -0.3987837  0.03637833 -1.780209 -1.2532017
2        0  1.17736694 -0.2357113 -0.09778397 -0.5400138 -1.0025178 -0.3987837 -0.21352497 -1.780209 -1.5190528
3        0  0.06211528 -0.2357113 -0.99603207  0.5826465  0.5504019 -0.3987837  0.53618492 -1.177653 -0.7214996
4        1 -1.05313638 -1.2368505 -0.99603207 -0.5400138 -1.0025178  1.0397512 -0.46342827  1.232570  0.8736068
5        0  0.06211528 -0.2357113 -0.99603207 -0.5400138 -0.2260579  1.0397512 -0.21352497 -1.478931 -0.1897975
6        0  1.17736694  0.7654280 -0.99603207 -0.5400138 -0.2260579  1.0397512  0.53618492  1.232570  1.1394578
          g3 other_fjob services teacher at_home health_fjob mother father other_guardian course home reputation
1 -0.96371171         0        0       1       0          0      1      0              0      1    0          0
2 -0.96371171         1        0       0       0          0      0      1              0      1    0          0
3 -0.09062427         1        0       0       0          0      1      0              0      0    0          0
4  1.00073503         0        1       0       0          0      1      0              0      0    1          0
5 -0.09062427         1        0       0       0          0      0      1              0      0    1          0
6  1.00073503         1        0       0       0          0      1      0              0      0    0          1
  other_reason
1            0
2            0
3            1
4            0
5            0
6            0
```

¡Ahora estamos listos para la clasificación k-NN!

Dividimos los datos en conjuntos de entrenamiento y prueba. Dividimos el 75% de los datos en el conjunto de entrenamiento y el 25% restante en el conjunto de prueba.

```
> set.seed(1234)
> smp_size <- floor(0.75 * nrow(data_class))
> train_ind <- sample(seq_len(nrow(data_class)), size = smp_size)
> class_pred_train <- data_class[train_ind, ]
> class_pred_test <- data_class[-train_ind, ]
```

Dividimos la variable de resultado en conjuntos de entrenamiento y prueba utilizando la misma partición que la anterior.

```
> mjob_outcome_train <- mjob_outcome[train_ind, ]
> mjob_outcome_test <- mjob_outcome[-train_ind, ]
```

Usar el paquete class. Ejecutamos la clasificación k-NN

Tenemos que decidir el número de vecinos (k). Hay varias reglas generales, una de las cuales es la raíz cuadrada del número de observaciones en el conjunto de entrenamiento. En este caso, seleccionamos 17 como número de vecinos, que es aproximadamente la raíz cuadrada de nuestro tamaño de muestra N = 296.

```
> mjob_pred_knn <- knn(train = class_pred_train, test = class_pred_test, cl = mjob_outcome_train, k=17)
```

Evaluación del modelo

```
> mjob_outcome_test <- data.frame(mjob_outcome_test)
> class_comparison <- data.frame(mjob_pred_knn, mjob_outcome_test)
> names(class_comparison) <- c("PredictedMjob", "ObservedMjob")
> head(class_comparison)
  PredictedMjob ObservedMjob
1         other      at_home
2      services      at_home
3         other        other
4         other        other
5         other     services
6       teacher      teacher
```

```
> CrossTable(x = class_comparison$ObservedMjob, y = class_comparison$PredictedMjob, prop.chisq=FALSE, prop.c = FALSE, prop.r = FALSE, prop.t = FALSE)
```

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

Total Observations in Table:  99

                            | class_comparison$PredictedMjob
class_comparison$ObservedMjob |   at_home |    health |     other |  services |   teacher | Row Total |
------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                      at_home |         1 |         0 |        14 |         3 |         0 |        18 |
------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                       health |         0 |         0 |         5 |         4 |         0 |         9 |
------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                        other |         0 |         1 |        25 |         1 |         0 |        27 |
------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                     services |         1 |         0 |        18 |         5 |         0 |        24 |
------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                      teacher |         0 |         1 |         9 |         7 |         4 |        21 |
------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                 Column Total |         2 |         2 |        71 |        20 |         4 |        99 |
------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
```

Los resultados de Cross Table indican que nuestro modelo no predijo muy bien el mother's job. Para leer la tabla cruzada, comenzamos examinando la diagonal superior izquierda a la inferior derecha de la matriz. La diagonal de la matriz representa el número de casos que se clasificaron correctamente para cada categoría.

Si el modelo clasificara correctamente todos los casos, la matriz tendría ceros en todas partes menos en la diagonal. En este caso, vemos que los números son bastante altos fuera de las diagonales, lo que indica que nuestro modelo no clasificó correctamente nuestro resultado en función de nuestros predictores.

Para examinar el éxito de la clasificación dada una determinada categoría, uno lee las filas de la matriz. Por ejemplo, al leer la primera fila, vemos que el modelo clasificó correctamente 1 de 18 casos "at home", 14 de 18 casos "at home" como "other" y 3 de 18 casos "at home" como "services".

Utilizamos ahora el paquete caret. Ejecutamos la clasificación k-NN.

En este paquete, la función elige el número óptimo de vecinos (k) para ti.

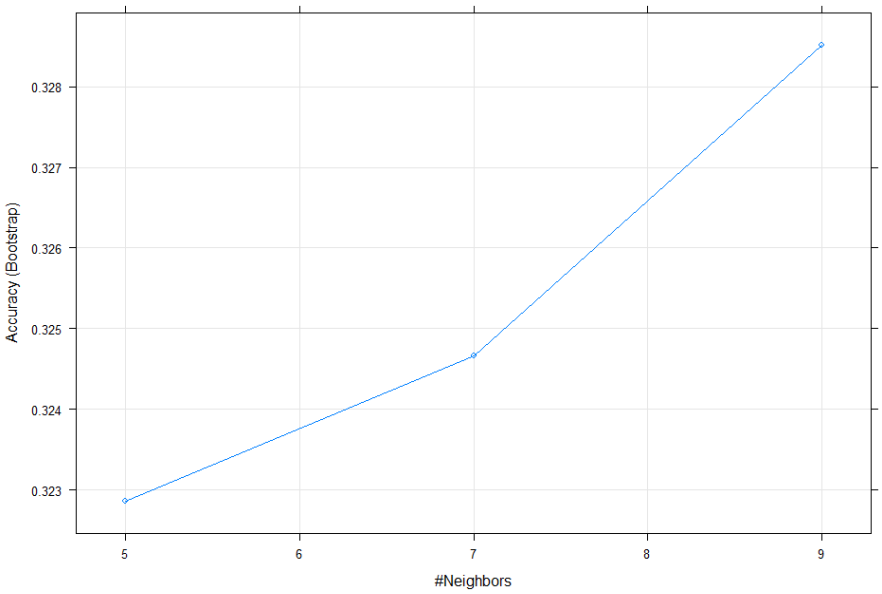
```
> mjob_pred_caret <- train(class_pred_train, mjob_outcome_train, method = "knn", preProcess = c("center","scale"))
```

Al observar la salida del modelo k-NN del paquete caret, podemos ver que eligió k = 9, dado que este fue el número en el que la precisión y kappa alcanzaron su punto máximo.

```
> mjob_pred_caret
k-Nearest Neighbors

296 samples
 41 predictor
  5 classes: 'at_home', 'health', 'other', 'services', 'teacher'

Pre-processing: centered (36), scaled (36), ignore (5)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 296, 296, 296, 296, 296, 296, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.3228596  0.08042517
  7  0.3246592  0.07108124
  9  0.3285163  0.07099429

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

Esta grafica también muestra un pico de precisión de 9.

```
> plot(mjob_pred_caret)
```


Accuracy (Bootstrap)
0.328
0.327
0.326
0.325
0.324
0.323
5
6
7
8
9
#Neighbors