PRÁCTICA MINERÍA DE DATOS: WEKA

Hernández Sánchez, Víctor

Escuela Politécnica Superior de Elche, Universidad Miguel Hernández

Grado en Ingeniería Informática en Tecnologías de la Información

Curso 2022-2023, Minería de Datos

19 de octubre de 2022

1. Descriptiva del Dataset utilizado

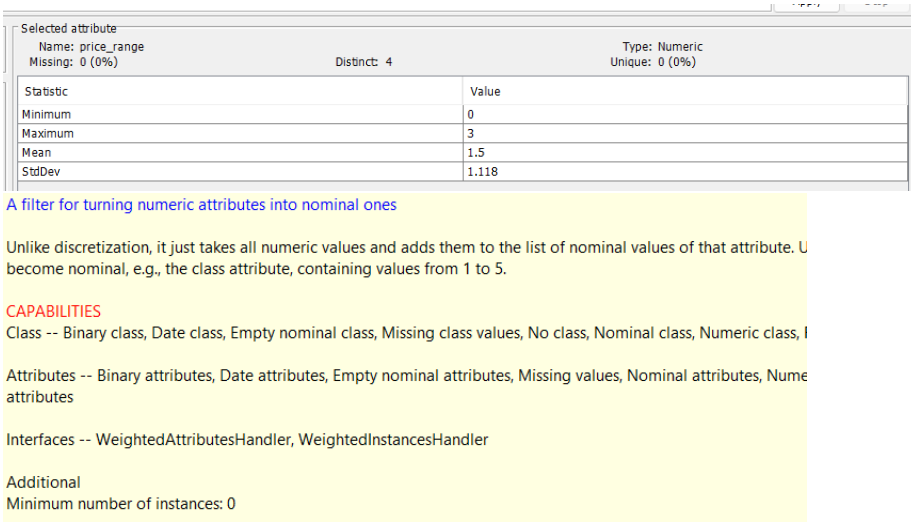
Para la realización de esta práctica, usaremos el Dataset encontrado en la página web:
https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?select=train.csv

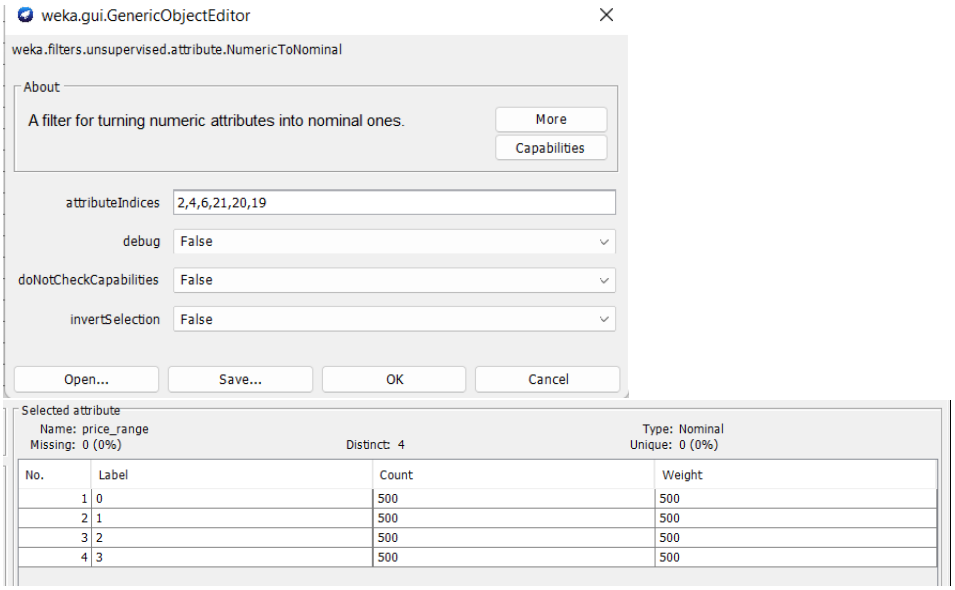
Dentro de esta web, vamos a usar los datos del archivo “moviles.arff”.

Este archivo nos proporciona datos detallados de una comparativa de características de móviles con variable objetivo el precio del teléfono.
Gracias a esto, podremos encontrar relaciones interesantes entre las distintas características y el precio de los móviles que las tienen.

Inicialmente, el Dataset entero venía con un formato de atributos numéricos, pero al analizarlo, muchos de ellos eran booleanas (0,1) o se dividían en pocos grupos, como por ejemplo la variable objetivo, dividida en 4 rangos de precios.

A estos atributos, se les aplicó un filtro de conversión de numérico a nominal para poder estudiarlas de forma más cómoda con Weka.





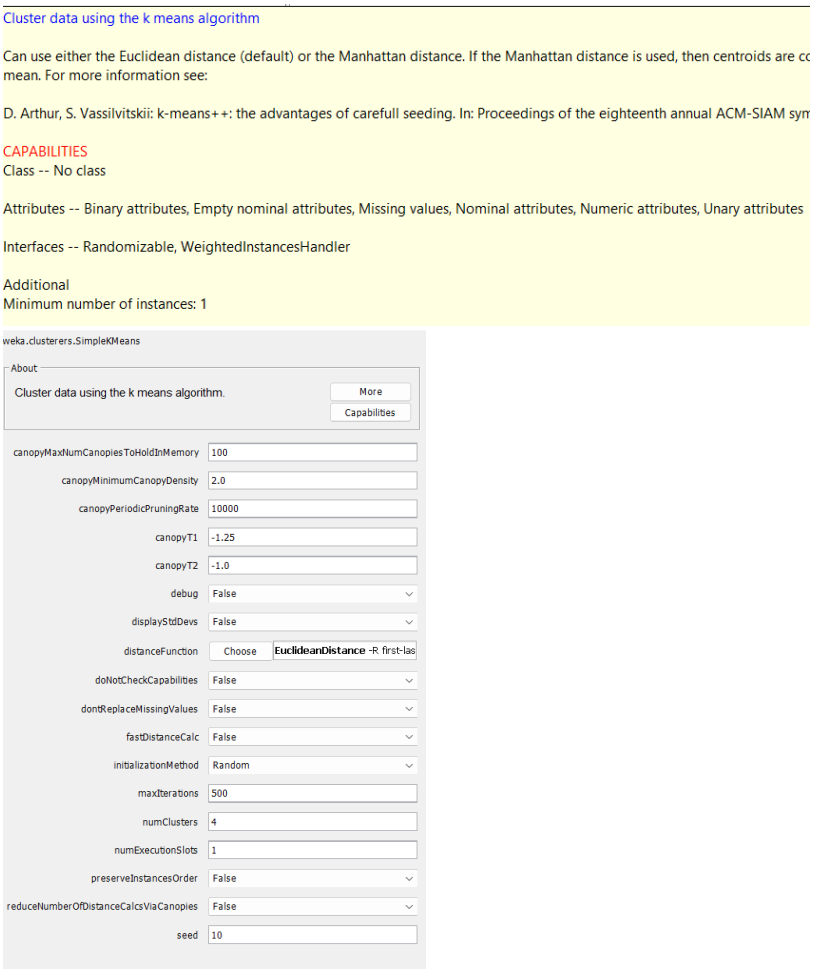
Una vez este filtro está aplicado, podemos comenzar a usar y probar los algoritmos.

2. Algoritmos

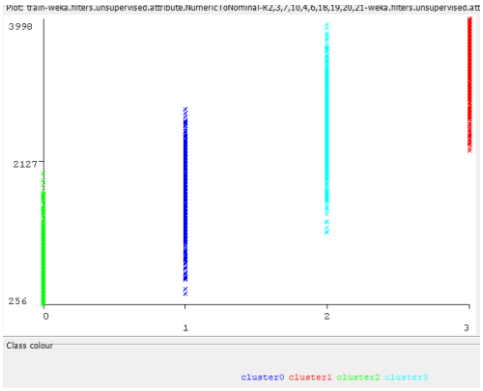
a. Clustering o segmentación

El Clustering es un proceso de minería de datos en el que se agrupan datos de un set de atributos concreto basándose en sus características y agregándolos por sus similitudes.

En este caso, usaremos el algoritmo llamado “SimpleKMeans”, que sirve para agregar datos de varios tipos, entre ellos numéricos y nominales.



Para la primera ejecución, usaremos los valores por defecto excepto el valor numClusters, ya que como nuestra variable objetivo tiene 4 rangos de precios, deberemos indicar un 4 en lugar del 2 por defecto. Seleccionaremos para el estudio los atributos de Precio (variable objetivo) y RAM.

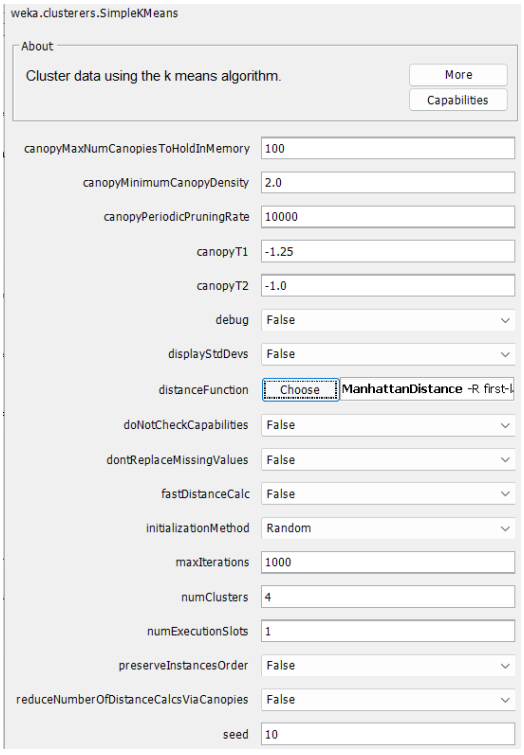


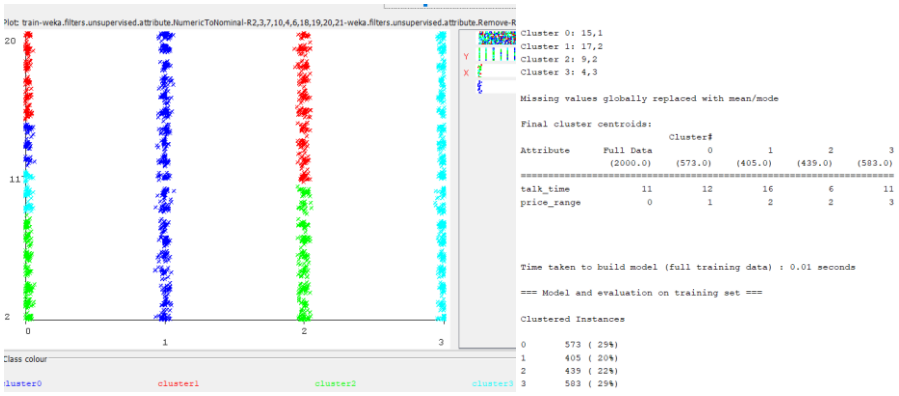

```
Final cluster centroids:
                           Cluster#
Attribute     Full Data          0          1          2          3
                (2000.0)   (500.0)    (500.0)    (500.0)    (500.0)
============================================================================
ram             2124.213   1679.49  3449.232    785.314   2582.816
price_range            0         1         3          0          2



Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       500 ( 25%)
1       500 ( 25%)
2       500 ( 25%)
3       500 ( 25%)
```

Podemos apreciar que los valores aparecen distribuidos en los 4 rangos de precios y que, a mayor RAM, mayor precio en general tendrá el teléfono.

También podemos observar que, en tanto en el primer nivel como en el último, se encuentran mucho más definidos los grupos de los clústers, incluso en el primer grupo llegando a no superar en ningún caso la mitad del total de RAM.

Si ahora cambiamos algunos valores de la ejecución, veremos que resultados obtendremos. En este caso vamos a cambiar el método de la función de distancia (de Euclídea a Manhattan) y a duplicar el número de iteraciones. Además, usaremos el atributo duración de la llamada en lugar de la RAM.

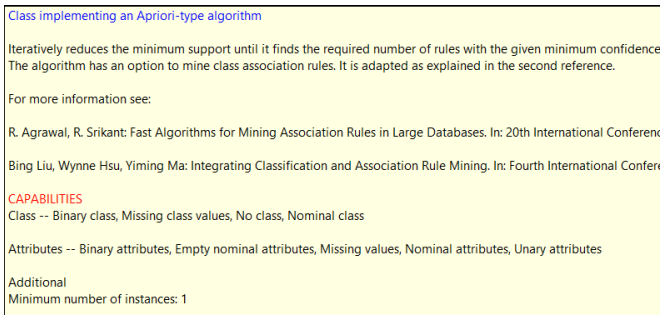




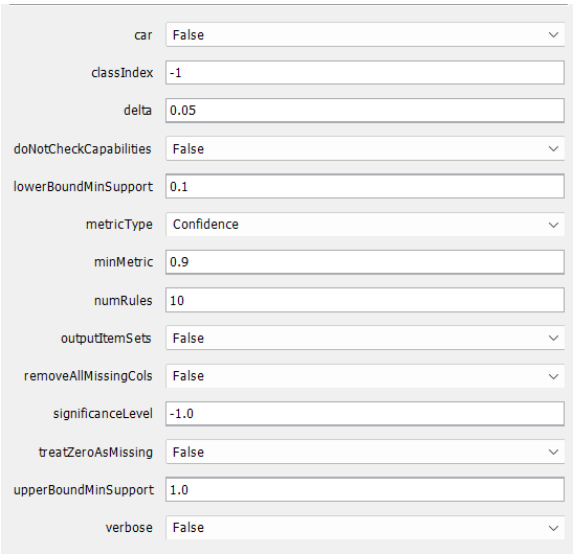
En este caso, observamos que no hay una diferencia real entre los 4 grupos, ya que todos, en algún caso concreto, hacen el mínimo y el máximo tiempo de llamadas hasta que la batería se agota.

b. Asociación

La asociación en minería de datos y análisis estadístico sirve para generar una serie de reglas o funciones con un índice de probabilidad de que se cumplan. Se usa estudiando un conjunto de datos y cruzando todas las instancias de cada uno con todas las del resto.



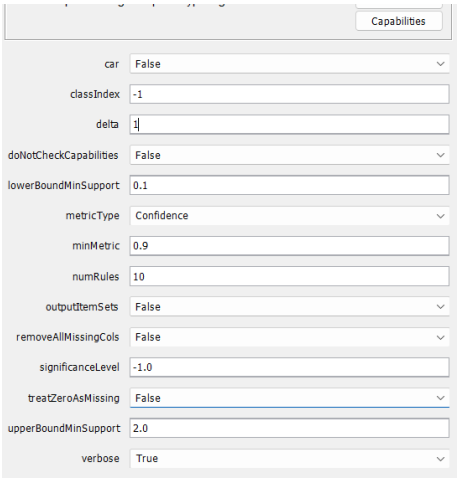
En este caso usaremos el algoritmo Apriori, con el que usaremos únicamente variables nominales y estudiaremos las reglas que nos indica. Lo ejecutaremos con las variables por defecto y con todos nuestros atributos nominales.




```
Apriori
=======

Minimum support: 0.2 (400 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 17

Size of set of large itemsets L(2): 51

Size of set of large itemsets L(3): 10

Best rules found:

 1. four_g=1 1043 ==> three_g=1 1043    <conf:(1)> lift:(1.31) lev:(0.12) [248] conv:(248.76)
 2. dual_sim=1 four_g=1 533 ==> three_g=1 533    <conf:(1)> lift:(1.31) lev:(0.06) [127] conv:(127.12)
 3. four_g=1 touch_screen=1 533 ==> three_g=1 533    <conf:(1)> lift:(1.31) lev:(0.06) [127] conv:(127.12)
 4. blue=1 four_g=1 523 ==> three_g=1 523    <conf:(1)> lift:(1.31) lev:(0.06) [124] conv:(124.74)
 5. four_g=1 wifi=0 523 ==> three_g=1 523    <conf:(1)> lift:(1.31) lev:(0.06) [124] conv:(124.74)
 6. blue=0 four_g=1 520 ==> three_g=1 520    <conf:(1)> lift:(1.31) lev:(0.06) [124] conv:(124.02)
 7. four_g=1 wifi=1 520 ==> three_g=1 520    <conf:(1)> lift:(1.31) lev:(0.06) [124] conv:(124.02)
 8. dual_sim=0 four_g=1 510 ==> three_g=1 510    <conf:(1)> lift:(1.31) lev:(0.06) [121] conv:(121.64)
 9. four_g=1 touch_screen=0 510 ==> three_g=1 510    <conf:(1)> lift:(1.31) lev:(0.06) [121] conv:(121.64)
10. three_g=0 477 ==> four_g=0 477    <conf:(1)> lift:(2.09) lev:(0.12) [248] conv:(248.76)
```

Obtenemos de resultado 10 reglas de a priori, todas con índice de confianza del 1. Esto ha ocurrido porque se cumple en todos los casos la regla de “si hay 3G, hay 4G y si hay 4G, hay 3G”, por lo tanto, en todas las combinaciones de atributos que se encuentre el 4G, sabremos con seguridad 1 que habrá 3G también.

Ahora, ejecutaremos eliminando 3G y 4G y modificando las variables del algoritmo, duplicaremos delta y el nivel superior máximo, añadiremos verbose.



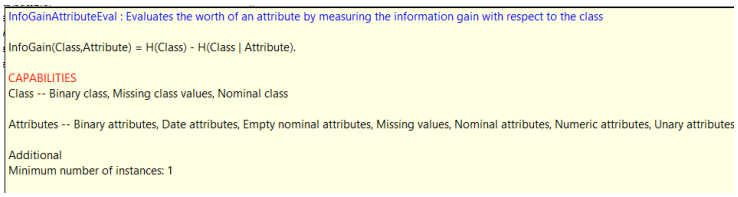

```
Apriori
=======

Minimum support: 0.1 (200 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 1

Generated sets of large itemsets:

Size of set of large itemsets L(1): 23

Size of set of large itemsets L(2): 89

Size of set of large itemsets L(3): 80

Best rules found:
```

No se ha encontrado ninguna regla. Si modificamos valores o añadimos y quitamos atributos, observamos que ocurre lo mismo.
Esto quiere decir que la única relación que ha podido encontrar ha sido entre 3G y 4G y eliminando una de ellas o ambas nos deja sin reglas.

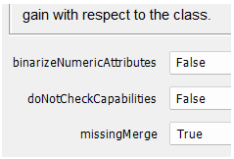
c. Selección de Características

La selección de características es un método que usaremos para observar cuales son los atributos que más nos conviene usar o cuales de ellos nos darán mejores resultados por ser más significativos.

En este caso, el algoritmo “InfoGainAttributeVal” es el que emplearemos, ya que nos permite incluir todo tipo de atributos.



Lo ejecutaremos con valores por defecto y con todos nuestros atributos.



```
Attribute Evaluator (supervised, Class (nominal): 21 price_range):
        Information Gain Ranking Filter

Ranked attributes:
 1.200329   14 ram
 0.079119    7 int_memory
 0.034715    1 battery_power
 0.023732    3 clock_speed
 0.023251   13 px_width
 0.020199   12 px_height
 0.007417   10 n_cores
 0.0014     19 touch_screen
 0.001148    6 four_g
 0.000517    2 blue
 0.000494   18 three_g
 0.000464    4 dual_sim
 0.000309   20 wifi
 0           5 fc
 0          15 sc_h
 0          16 sc_w
 0          11 pc
 0           9 mobile_wt
 0           8 m_dep
 0          17 talk_time

Selected attributes: 14,7,1,3,13,12,10,19,6,2,18,4,20,5,15,16,11,9,8,17 : 20
```

Obtenemos el siguiente resultado, en el que apreciamos que para la variable objetivo, el valor más significativo es la RAM y el resto tienen mucha menos importancia real.

d. Clasificación

El método de clasificación lo usaremos para crear árboles muy visuales que nos indican con porcentajes de certeza acciones que suceden o no dependiendo de los atributos que sean elegidos.

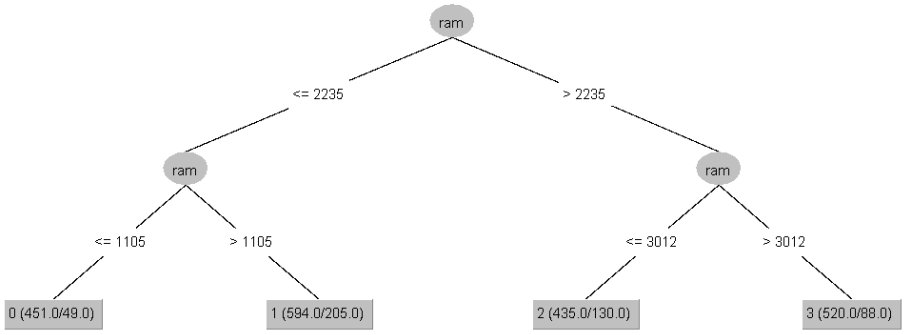
Para este ejemplo usaremos el algoritmo J48 con los valores por defecto y con atributos RAM y precio.

```
Class for generating a pruned or unpruned C4

5 decision tree. For more information, see

Ross Quinlan (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publ

CAPABILITIES
Class -- Binary class, Missing class values, Nominal class

Attributes -- Binary attributes, Date attributes, Empty nominal attributes, Missing v

Interfaces -- Drawable, PartitionGenerator, Sourcable, WeightedInstancesHandler

Additional
Minimum number of instances: 0
```

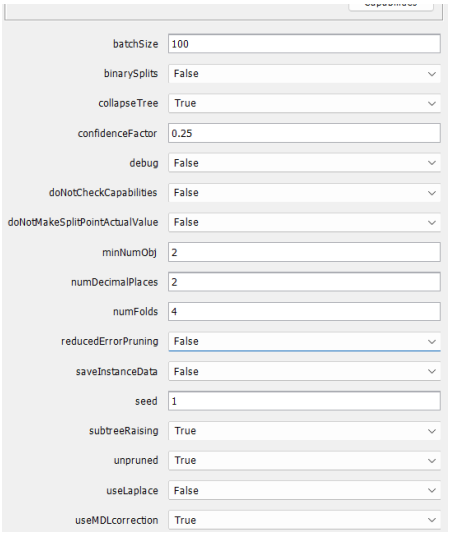
batchSize	100
binarySplits	False
collapseTree	True
confidenceFactor	0.25
debug	False
doNotCheckCapabilities	False
doNotMakeSplitPointActualValue	False
minNumObj	2
numDecimalPlaces	2
numFolds	3
reducedErrorPruning	False
saveInstanceData	False
seed	1
subtreeRaising	True
unpruned	False
useLaplace	False
useMDLcorrection	True

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1506               75.3    %
Incorrectly Classified Instances       494               24.7    %
Kappa statistic                          0.6707
Mean absolute error                      0.1883
Root mean squared error                  0.3099
Relative absolute error                 50.2079 %
Root relative squared error             71.5719 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,808    0,042    0,865      0,808    0,836      0,784    0,944     0,810     0
                 0,744    0,138    0,642      0,744    0,690      0,579    0,839     0,561     1
                 0,600    0,090    0,690      0,600    0,642      0,535    0,817     0,573     2
                 0,860    0,059    0,829      0,860    0,844      0,791    0,944     0,782     3
Weighted Avg.    0,753    0,082    0,756      0,753    0,753      0,672    0,886     0,681
```

Obtenemos un porcentaje bueno, del 75.3% y si analizamos el árbol, obtenemos lo siguiente.



Si ahora modificamos algunos valores, bajando el confidencefactor, marcando como true unpruned y numfolds=4, con los mismos atributos y añadiendo los pixeles de la cámara principal, obtenemos este resultado.




```
=== Summary ===

Correctly Classified Instances        1495               74.75   %
Incorrectly Classified Instances       505               25.25   %
Kappa statistic                          0.6633
Mean absolute error                      0.1846
Root mean squared error                  0.3096
Relative absolute error                 49.2167 %
Root relative squared error             71.4902 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,832    0,058    0,827      0,832    0,830      0,772    0,956     0,828     0
                 0,698    0,129    0,644      0,698    0,670      0,555    0,834     0,573     1
                 0,600    0,091    0,688      0,600    0,641      0,534    0,838     0,585     2
                 0,860    0,059    0,829      0,860    0,844      0,791    0,944     0,782     3
Weighted Avg.    0,748    0,084    0,747      0,748    0,746      0,663    0,893     0,692
```

En general, prácticamente el mismo resultado en porcentaje de error, salvo que en este caso tenemos un atributo más. Si generamos el árbol, podemos apreciar que aparecerá este atributo añadiendo nuevas ramas a las decisiones.