Práctica 4: Weka – Clasificación con redes bayesianas

1. Introducción

En este guion vamos a aprender distintos clasificadores para el conjunto de datos "ledLXMn30.arff" con Weka. Como bien dice el nombre del archivo, tenemos un conjunto de datos con 10000 muestras y un 30% de ruido, por lo que vamos a tener que aplicar preprocesamiento a los datos.

2. Inspeccionando los datos

Lo primero, será visualizar el conjunto de datos que tenemos:

Figura 1: Distribución de frecuencias absolutas para cada atributo

Viendo todas las variables y el conjunto de datos que tiene cada uno. Podemos observar como los atributos att1 al att24 son atributos binarios en los que su valor será 0 o 1. La variable clase por el contrario tiene 10 clases, siendo del 0 al 9 sus etiquetas.

3. Sin selección de características

Vamos a probar que pasaría en la clasificación sin selección de características ni eliminación de ruido para tener una base.

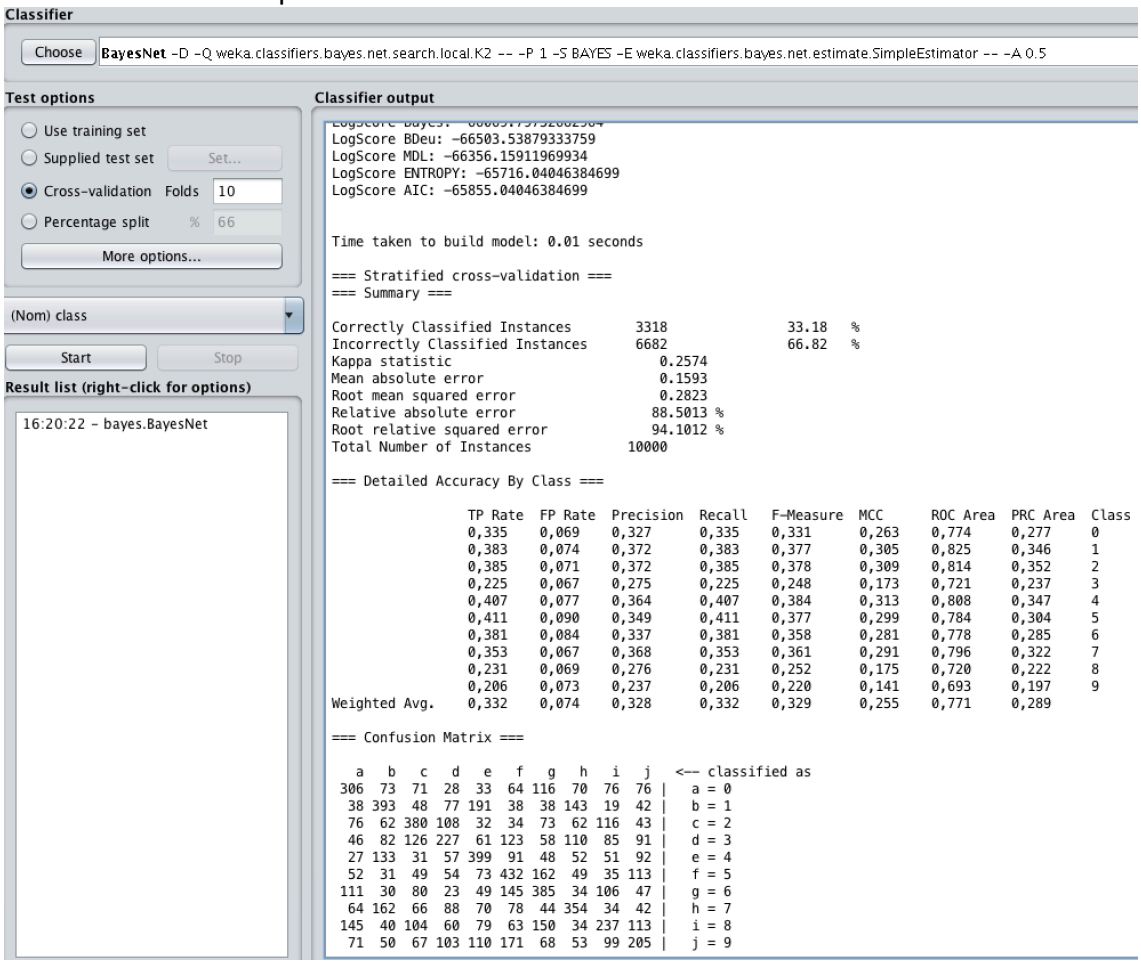

Figura 2: Clasificación con BayesNet

Donde podemos ver que el accuracy es de 33.18% con un clasificador BayesNet, por lo que ya tenemos una idea de que tenemos que mejorar.
Veamos como es el grafo de clasificación.

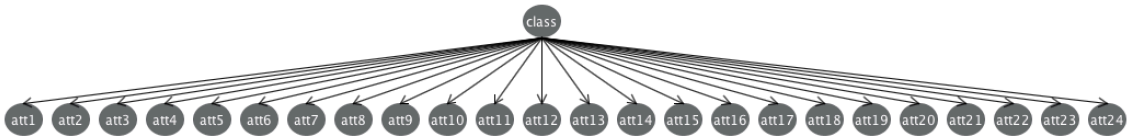

Figura 3: Grafo de clasificación con BayesNet

4. Selección de características

4.1 Selección de características

Lo primero será seleccionar características ya que tenemos 24 atributos de las cuales seguramente podamos desechar algunas. Por lo tanto, vamos a la pestaña “Select attributes” y aplicamos un evaluador “CfsSubsetEval” y como método de búsqueda el “GreedyStepwise”, con una cross-validation de 10 Folds, quedando como resultado:

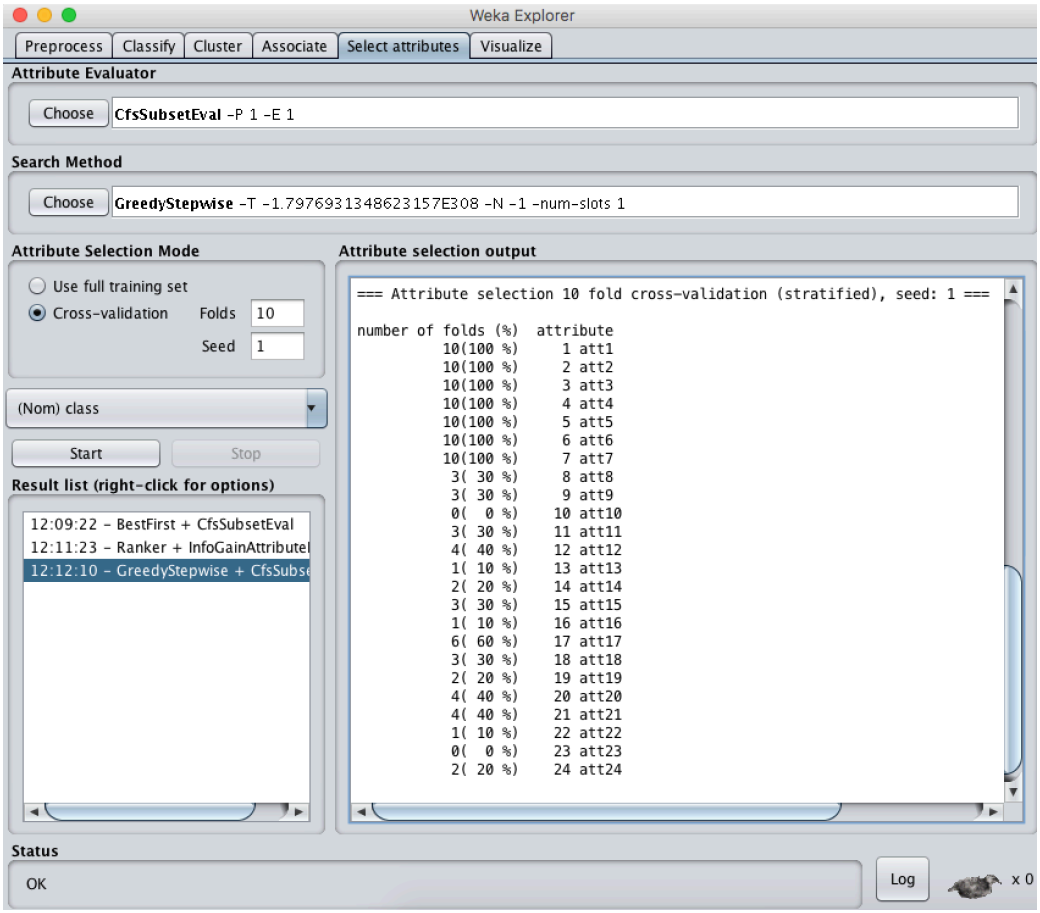

Figura 4: Distribución de frecuencias absolutas para cada atributo

Por lo que nos quedaremos con los atributos del 1 al 7. Quedando los datos:

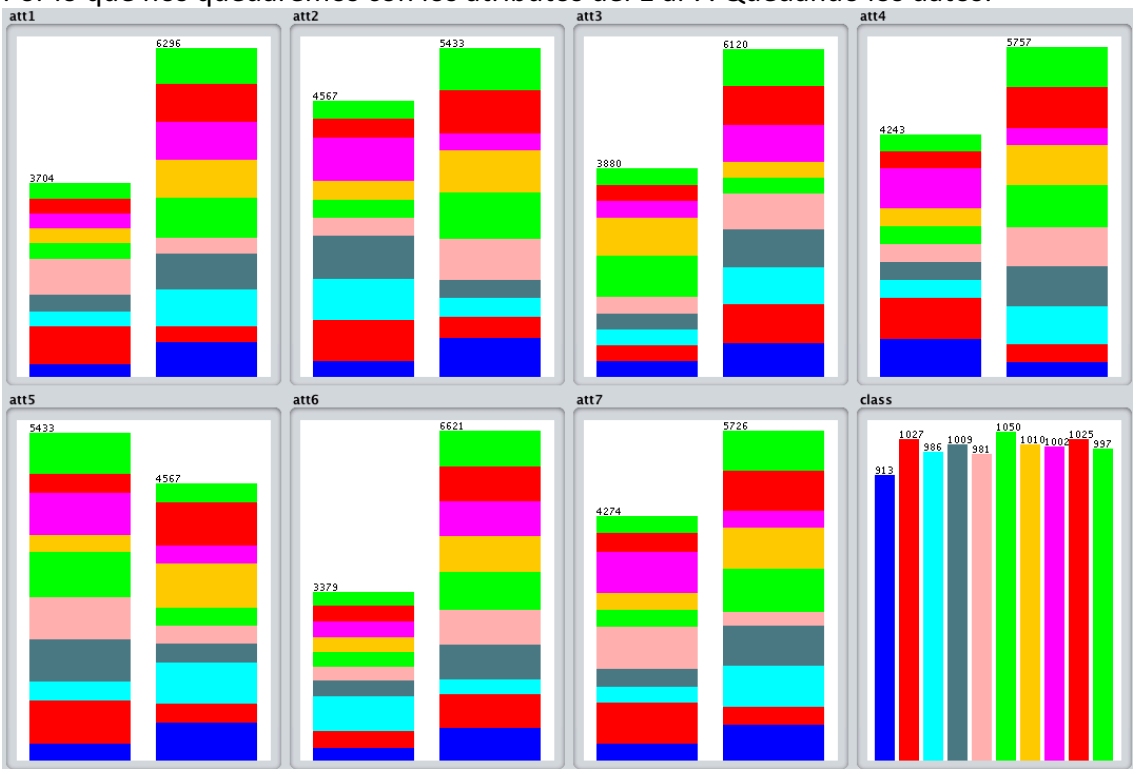

Figura 5: Distribución de frecuencias absolutas para cada atributo

4.2 Clasificación con selección de características

Si primeramente aplicamos un clasificador ZeroR para obtener una visión preliminar de cómo funciona el clasificador para este conjunto de datos obtenemos:

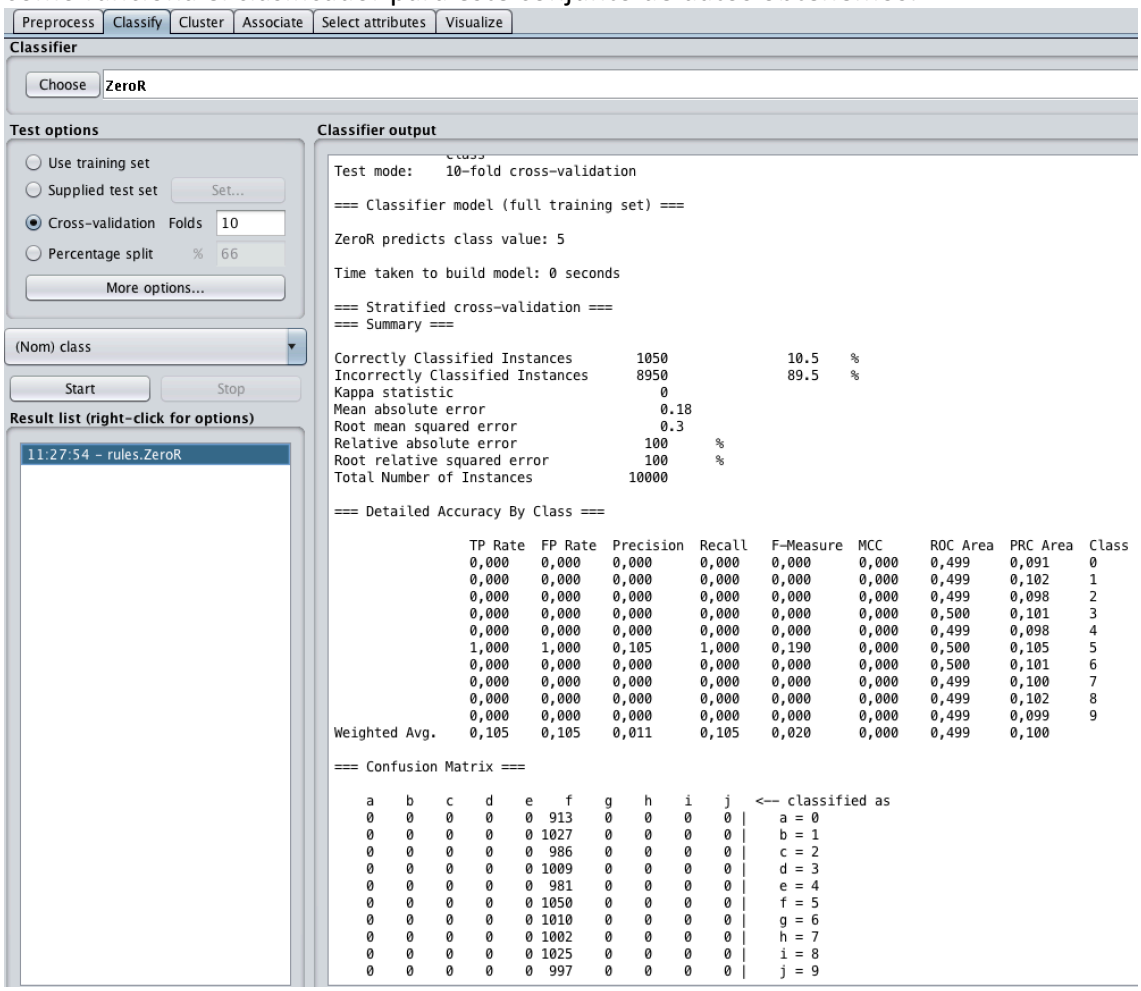

Figura 6: Clasificación con ZeroR

Donde tenemos un accuracy del 10,5%.

Si por ejemplo aplicamos un árbol C4.5 obtenemos un árbol con un número de hojas de 50 y un tamaño del árbol de 99.

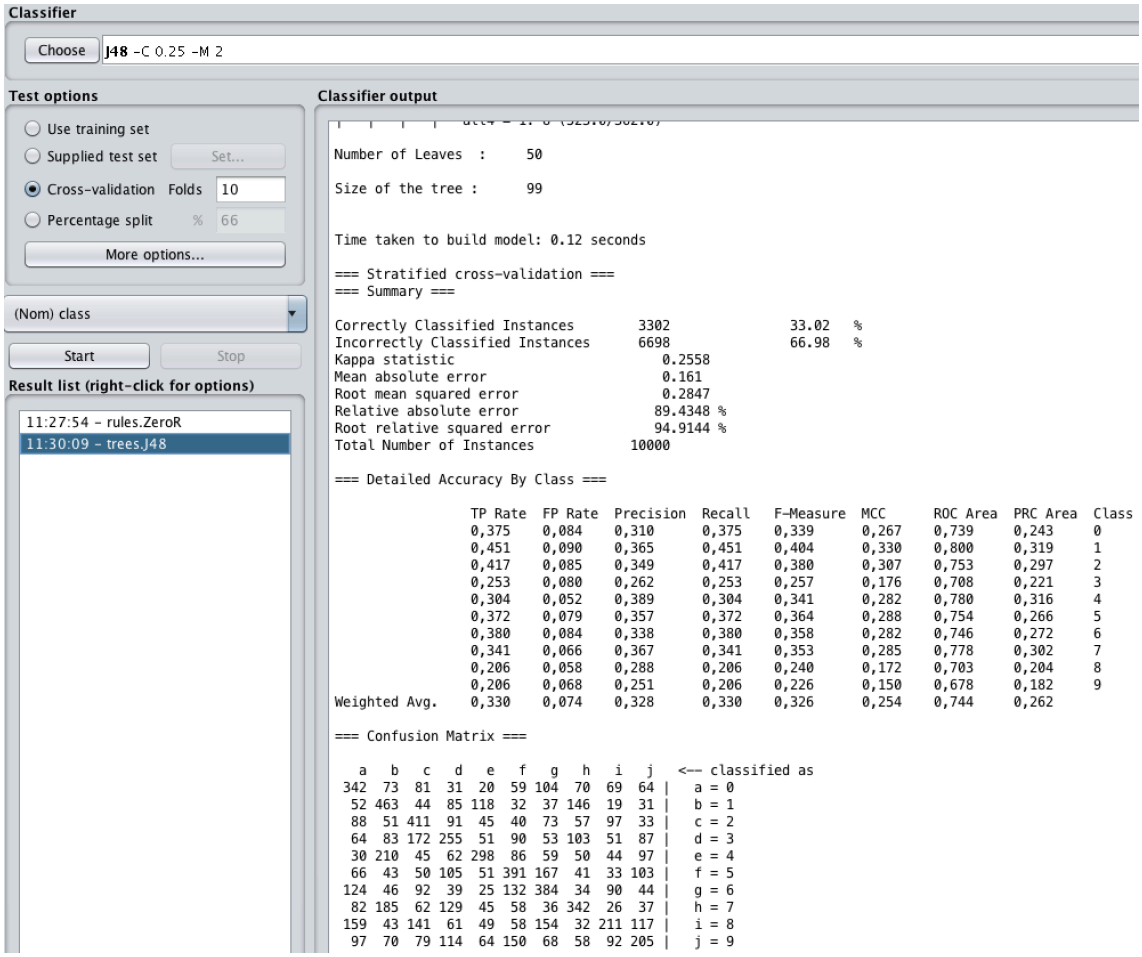

Figura 7: Clasificación con C4.5

Donde el accuracy es del 33,02%, mucho mejor que el ZeroR, pero menos que elegir aleatoriamente al 50%.

Si pasamos a usar técnicas basadas en bayes, podemos tener los siguientes resultados:

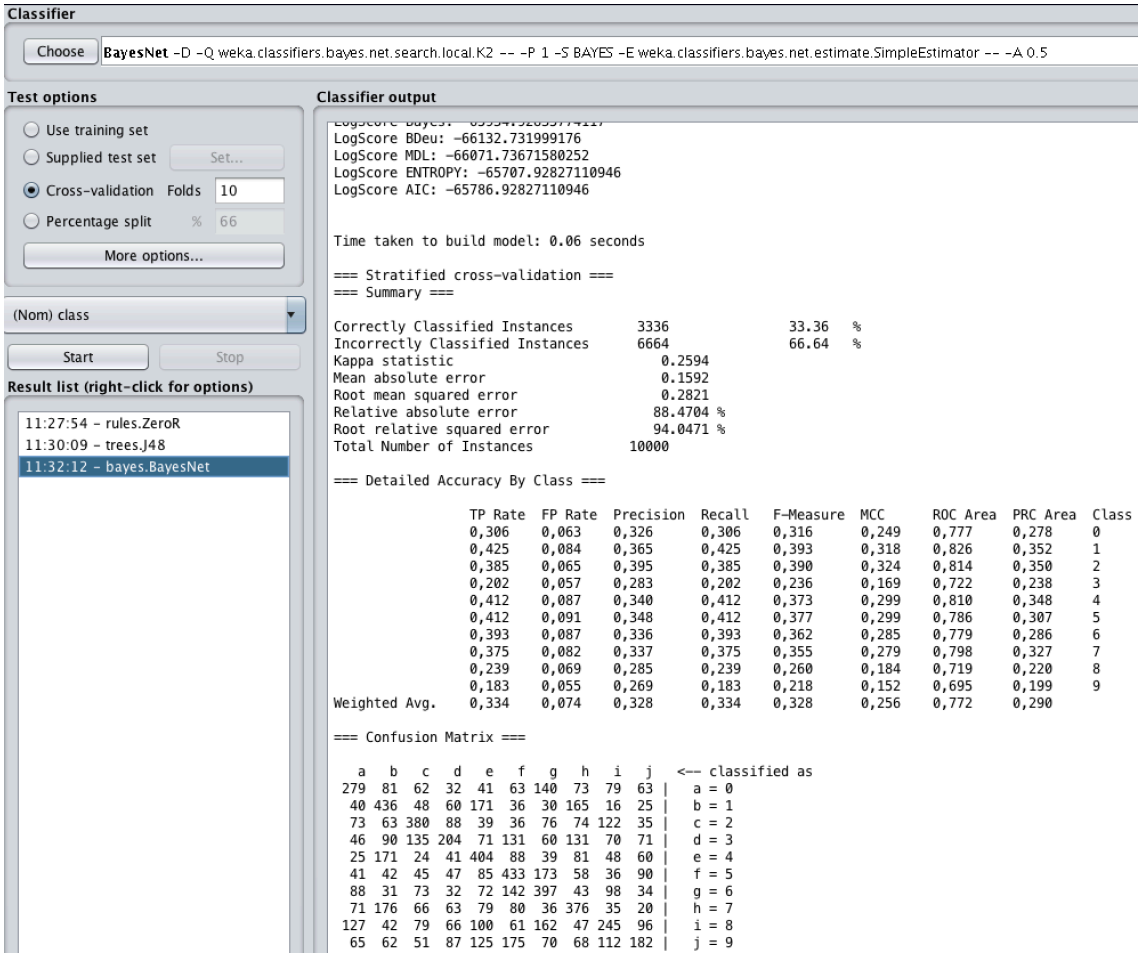

Figura 8: Clasificación con BayesNet

Donde el accuracy ha sido del 33.36%, mejor que los anteriores. Donde podemos ver su grafo:

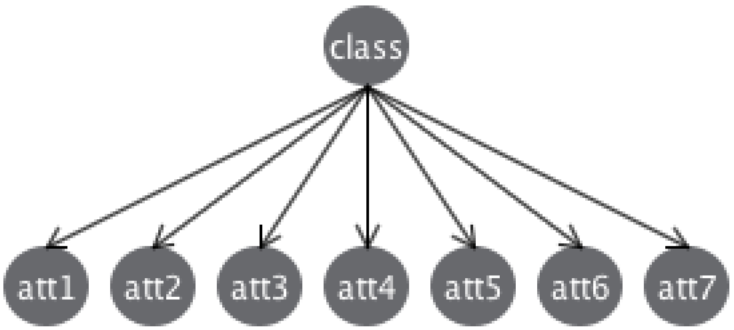

Figura 9: Grafo de clasificación con BayesNet

Si probamos distintas configuraciones del algoritmo BayesNet, como es cambiar el algoritmo de búsqueda podemos llegar a un accuracy del 33,37% con el algoritmo ICSSearchAlgorithm.

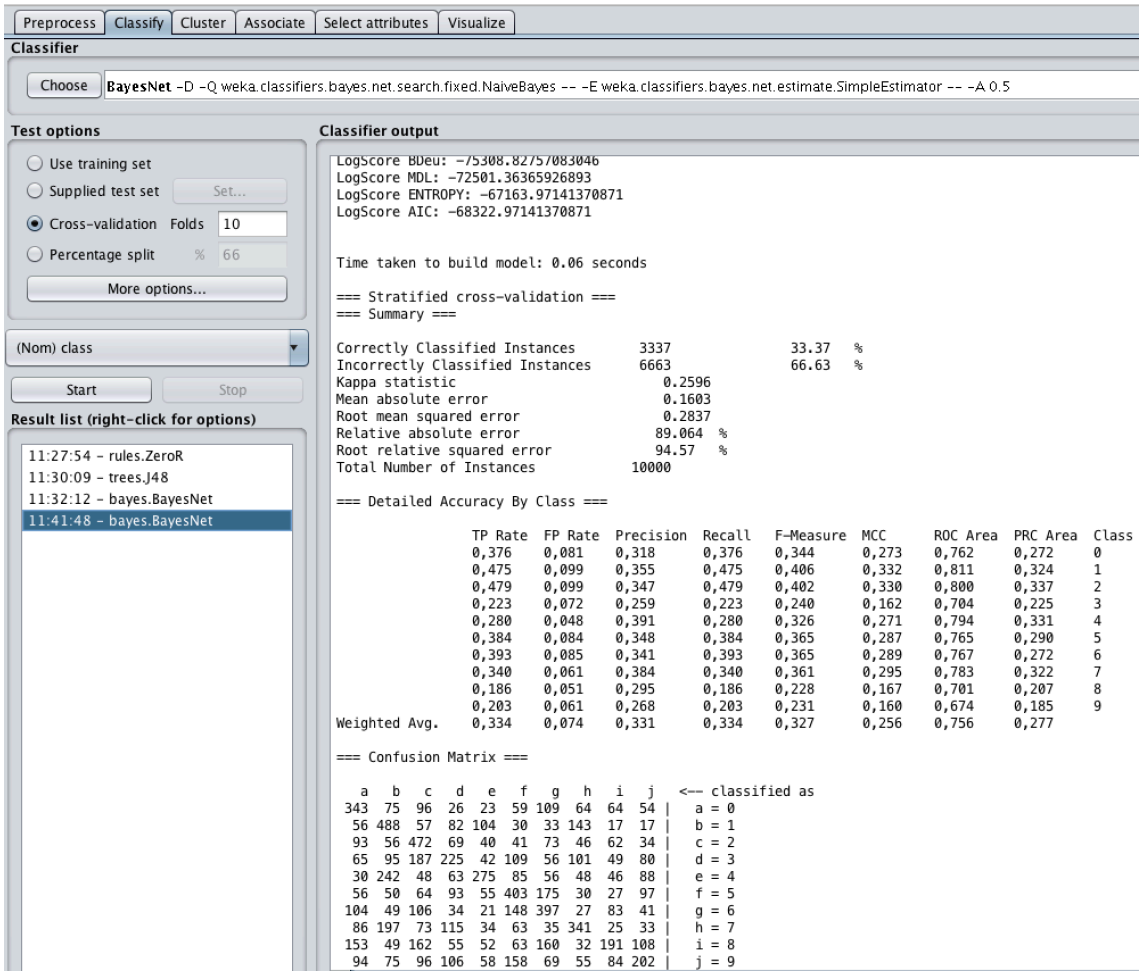

Figura 10: Clasificación con BayesNet y algoritmo ICCSearchAlgorithm

Probando distintos algoritmos de búsqueda podemos ver que no se mejora la clasificación por lo que vamos a probar otro método. Además, hemos mejorado a la clasificación base, sin selección de características.

5 Eliminación del ruido

Para eliminar el ruido en Weka tenemos distintos filtros, vamos a probar algunos de ellos.

5.1 Filtro Resample

Vamos a realizar este filtro para ver si así conseguimos eliminar un poco de ruido sobre los atributos seleccionados. Al aplicar el filtro nos queda una distribución:

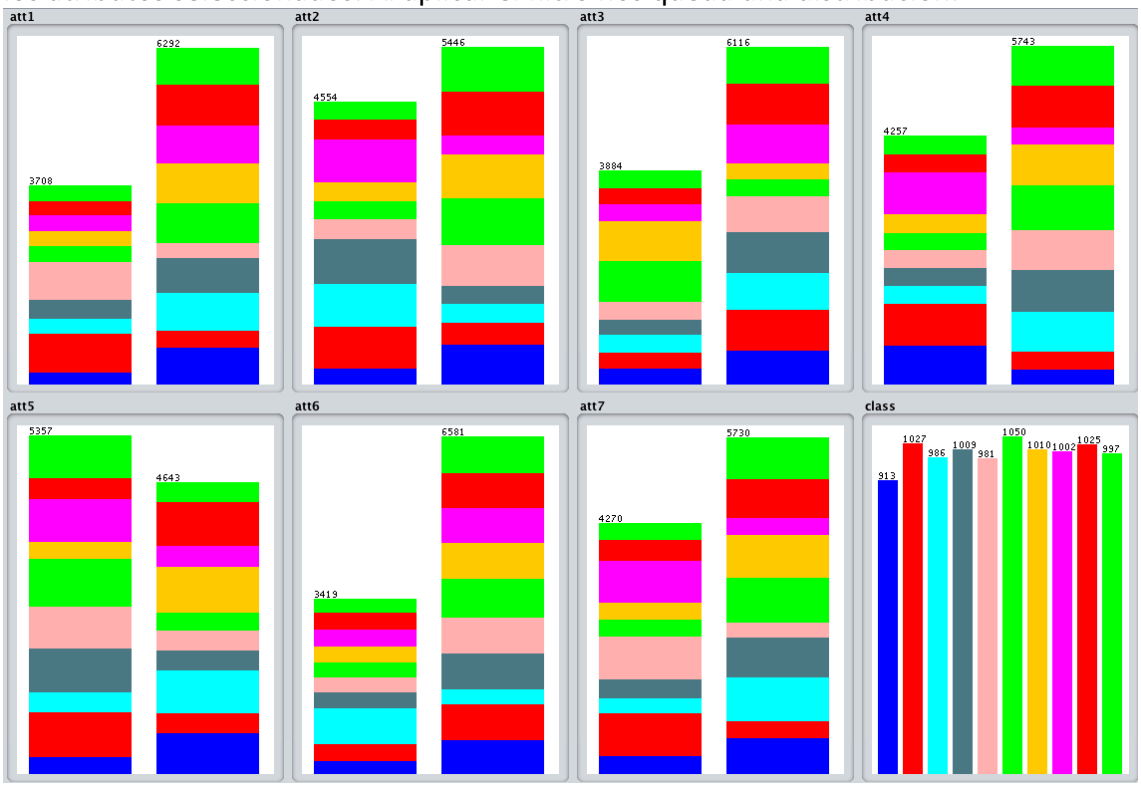

Figura 11: Distribución de frecuencias absolutas para cada atributo

Si aplicamos un BayesNet con el algoritmo de búsqueda ICSSearchAlgorithm, que ha sido el que mejor nos ha funcionado, obtenemos la siguiente clasificación.

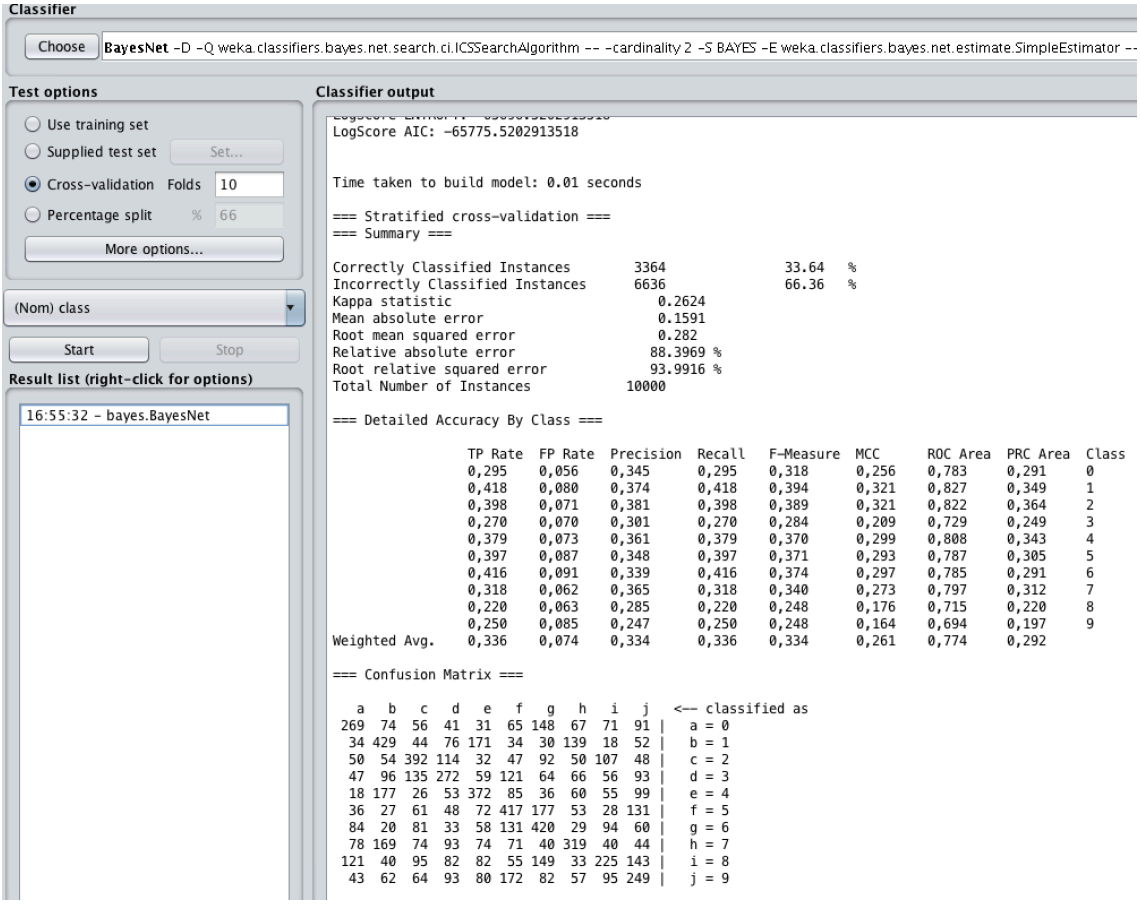

Figura 12: Clasificación con BayesNet y algoritmo ICCSearchAlgorithm

5.2 Filtro StratifiedRemoveFolds

Como tenemos un conjunto de datos bastante grande y solo 7 atributos, vamos a probar a eliminar instancias para ver que tal funciona esta técnica. La distribución resultante es:

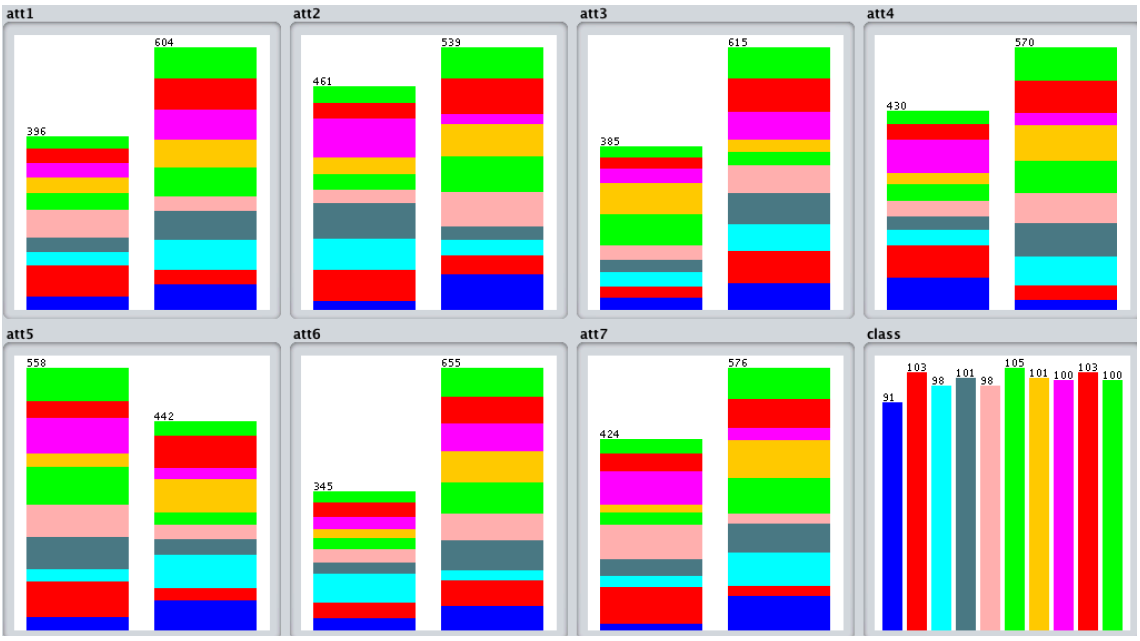

Figura 13: Distribución de frecuencias absolutas para cada atributo

Si ahora aplicamos una clasificación podemos obtener:

```
LogScore MDL: -6852.08444065325
LogScore ENTROPY: -6579.228107133455
LogScore AIC: -6658.228107133455


Time taken to build model: 0 seconds

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

Correctly Classified Instances         343               34.3    %
Incorrectly Classified Instances       657               65.7    %
Kappa statistic                          0.2701
Mean absolute error                      0.1576
Root mean squared error                  0.2815
Relative absolute error                 87.5401 %
Root relative squared error             93.8238 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,495    0,094    0,346      0,495    0,407      0,343    0,825     0,386     0
                 0,466    0,100    0,348      0,466    0,398      0,322    0,807     0,313     1
                 0,367    0,052    0,434      0,367    0,398      0,340    0,822     0,416     2
                 0,307    0,087    0,284      0,307    0,295      0,213    0,711     0,217     3
                 0,296    0,074    0,302      0,296    0,299      0,224    0,784     0,336     4
                 0,410    0,087    0,355      0,410    0,381      0,303    0,767     0,280     5
                 0,406    0,067    0,406      0,406    0,406      0,339    0,790     0,361     6
                 0,320    0,076    0,320      0,320    0,320      0,244    0,822     0,344     7
                 0,204    0,042    0,356      0,204    0,259      0,208    0,703     0,225     8
                 0,170    0,051    0,270      0,170    0,209      0,147    0,698     0,190     9
Weighted Avg.    0,343    0,073    0,342      0,343    0,337      0,268    0,772     0,305

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j   <-- classified as
 45  7  8  2  3 11  2  3  5  5 |  a = 0
  7 48  3  7 12  4  2 16  2  2 |  b = 1
 10  5 36 11  5  5  6 10  6  4 |  c = 2
  8  9  6 31  6 11  9 13  4  4 |  d = 3
  5 25  1 10 29  7  5  6  5  5 |  e = 4
 11  4  2  3  7 43 13  9  3 10 |  f = 5
 14  3  4  9  5 17 41  0  5  3 |  g = 6
  4 27  9  9  7  4  5 32  1  2 |  h = 7
 14  4  9  9 11  5 11  8 21 11 |  i = 8
 12  6  5 18 11 14  7  3  7 17 |  j = 9
```

Figura 14: Clasificación con BayesNet y algoritmo HillClimber

Obteniendo un accuracy del 34.3%, el mejor obtenido hasta el momento. Por lo tanto, este ha sido el mejor resultado obtenido en todas las pruebas realizadas, quedando como resumen la selección de características, el filtro de ruido y aplicación del clasificador BayesNet.

5.3 Filtro AddClassification

Una vez probadas muchas técnicas (en este documento no las he puesto todas, ya que eran muchas), el mejor resultado se ha obtenido con este filtro. Para ser más específicos la configuración usada ha sido la de obtener una primera clasificación con un clasificador BayesNet y sustituir nuestro atributo class por esta nueva clasificación:

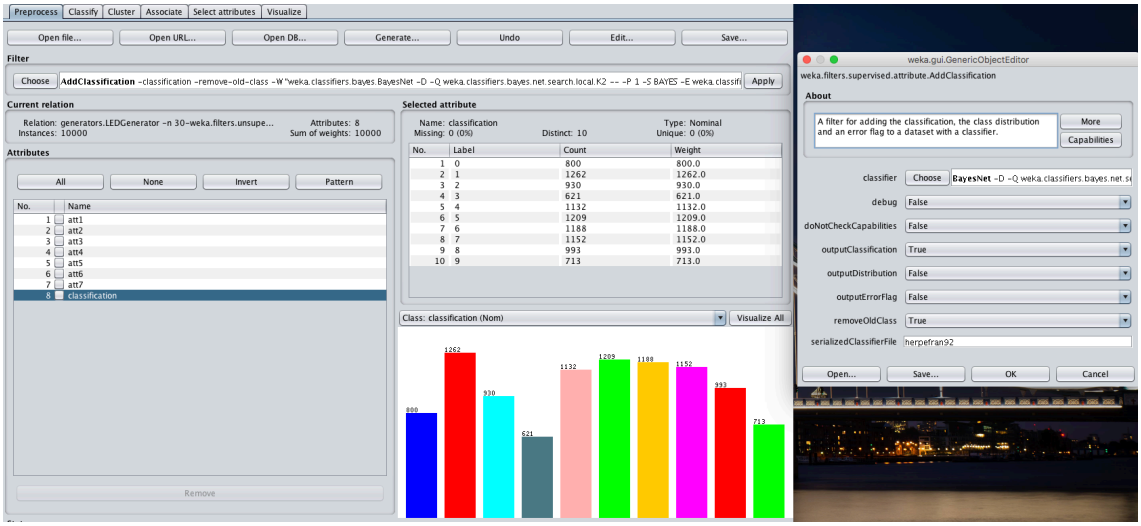

Figura 15: Filtro AddClassification con parámetros

Una vez pasado este filtro, realizamos un clasificador como el usado para el filtro como es BayesNet obteniendo el siguiente resultado:

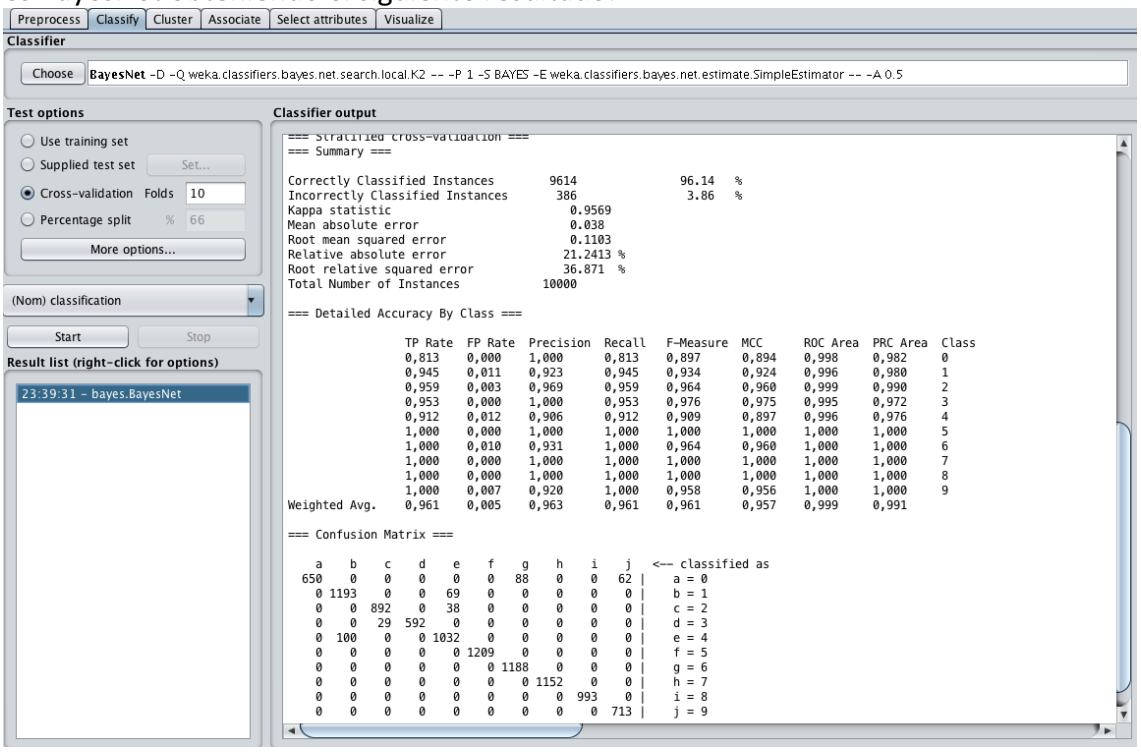

Figura 16: Clasificación con BayesNet

De esta forma, hemos obtenido un accuracy del 96,14%, el valor más alto conseguido hasta el momento y con un filtro diferente. Tenemos el peligro de tener un modelo sobre ajustado al ver este accuracy pero como hemos aplicado CrossValidation, podemos estar algo más tranquilos al ver nuestros resultados.