Prácticas de Minería de Datos

Grado en Ingeniería Informática

Curso 2023/24

Manuel Ramírez Ballesteros

ÍNDICE

1. Introducción: Entorno WEKA

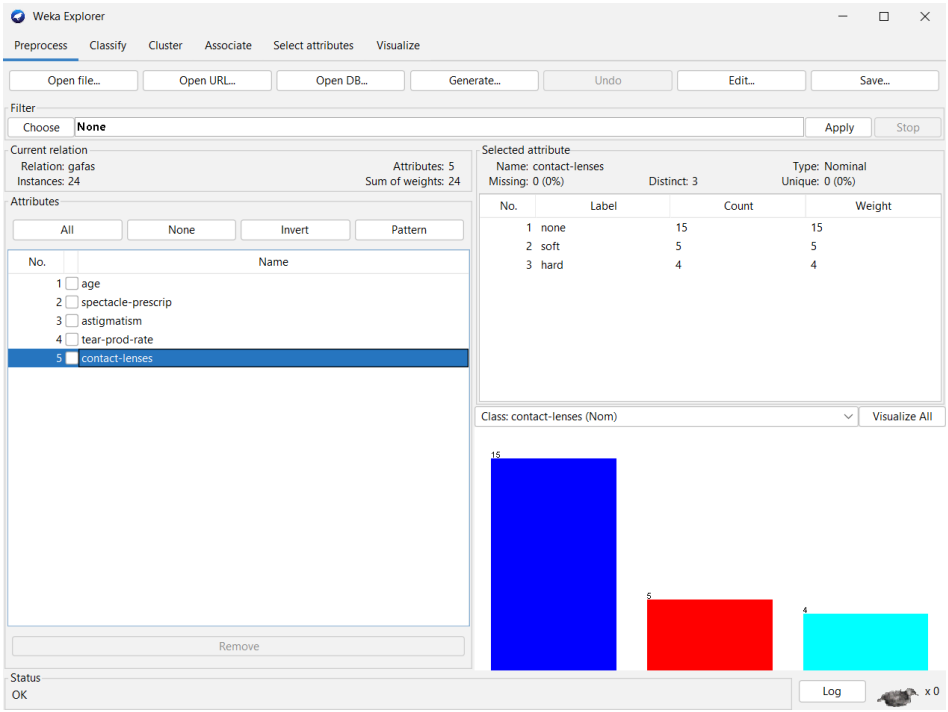
WEKA es una herramienta gratuita para experimentación de minería de datos, escrita en lenguaje Java, que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático, sobre cualquier conjunto de datos del usuario.
Para ello únicamente se requiere que los datos a analizar se almacenen con un cierto formato, conocido como *arff*. En estas prácticas abordaremos las diferentes tareas que nos encontramos en la Minería de datos y a diferentes *datasets* que nos permitan profundizar en este ámbito.

2. Práctica 1: Conocimiento del entorno WEKA.

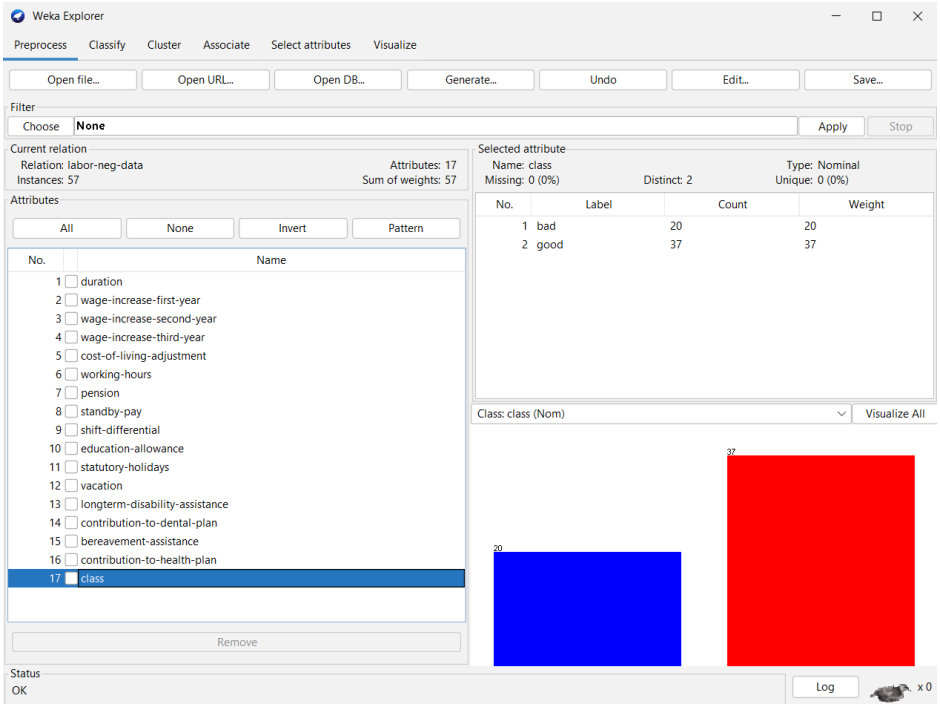
Para este ejercicio se ha generado el fichero *gafas arff* a partir de Excel adjunto:

	A	B	C	D	E
1	age	spectacle-prescrip	astigmatism	tear-prod-rate	contact-lenses
2	young	myope	no	reduced	none
3	young	myope	no	normal	soft
4	young	myope	yes	reduced	none
5	young	myope	yes	normal	hard
6	young	hypermetrope	no	reduced	none
7	young	hypermetrope	no	normal	soft
8	young	hypermetrope	yes	reduced	none
9	young	hypermetrope	yes	normal	hard
10	pre-presbyopic	myope	no	reduced	none
11	pre-presbyopic	myope	no	normal	soft
12	pre-presbyopic	myope	yes	reduced	none
13	pre-presbyopic	myope	yes	normal	hard
14	pre-presbyopic	hypermetrope	no	reduced	none
15	pre-presbyopic	hypermetrope	no	normal	soft
16	pre-presbyopic	hypermetrope	yes	reduced	none
17	pre-presbyopic	hypermetrope	yes	normal	none
18	presbyopic	myope	no	reduced	none
19	presbyopic	myope	no	normal	none
20	presbyopic	myope	yes	reduced	none
21	presbyopic	myope	yes	normal	hard
22	presbyopic	hypermetrope	no	reduced	none
23	presbyopic	hypermetrope	no	normal	soft
24	presbyopic	hypermetrope	yes	reduced	none
25	presbyopic	hypermetrope	yes	normal	none

Al abrirlo en nuestro entorno WEKA:



Para el segundo ejercicio, trabajaremos con el fichero labor arff:



Donde podemos observar que se trata de una base de datos con acuerdos laborales que consta de 57 instancias o contratos laborales, cada uno de ellos definido por 17 atributos. De estos atributos, el último es el atributo de clase, que distingue la calidad del contrato mediante dos valores: bueno o malo.
El atributo que contiene más valores ausentes es *standby-pay*, del cual se desconocen 48 valores, el equivalente al 84%:

Selected attribute

Name: standby-pay | Type: Numeric
Missing: 48 (84%) | Distinct: 7 | Unique: 6 (11%)

Statistic	Value
Minimum	2
Maximum	14
Mean	7.444
StdDev	5.028

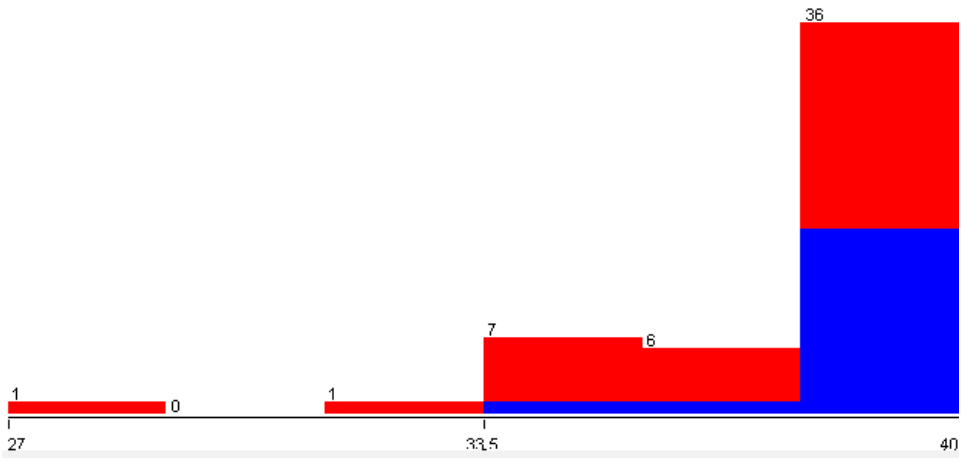
Estudiando el histograma del atributo *working-hours*:

Selected attribute

Name: working-hours | Type: Numeric
Missing: 6 (11%) | Distinct: 8 | Unique: 3 (5%)

Statistic	Value
Minimum	27
Maximum	40
Mean	38.039
StdDev	2.506

Class: class (Nom) | Visualize All



Podemos observar que la mayoría de los contratos que se realizaron eran de 40 horas, tanto buenos como malos, y que la media de los contratos para esta muestra está en 38 horas, siendo poco frecuentes los contratos de menos de 33,5 horas.

Para ver el atributo que mejor y peor dividen a la clase, podemos hacer algunas reglas del tipo:

- Si *wage-increase-first-year* es alto o muy alto, la clase es Good -> (4.5, 7]
- Si *wage-increase-second-year* es alto o muy alto, la clase es Good -> (5, 7]
- Si *wage-increase-third-year* es de medio a muy alto, la clase es Good -> (3.03, 5.1]
- Si *working-hours* es bajo o muy bajo, la clase es Good -> [27, 33.5]
- Si *pension* es muy bajo (sin pensión), la clase es Bad.
- Si *standby-pay* es bajo o muy bajo, la clase es Bad -> [2, 6]
- Si *standby-pay* es medio, alto o muy alto, la clase es Good -> (6, 14]
- Si *shift-differential* es medio o muy alto, la clase es Good -> (8.33, 16.67] y (20.83, 25]
- Si *statutory-holidays* es alto o muy alto, la clase es Good-> (12.6, 15]
- Si *longterm-disability-assitance* es bajo (no, al ser nominal), la clase es Bad.
- Si *contribution-to-dental-plan* es muy alto (completo), la clase es Good.
- Si *bereavement-assistance* es muy bajo (no), la clase es Bad.
- Si *contribution-to-health-plan* es muy bajo (no), la clase es Bad.
- No se han generado reglas para los atributos *duration*, *cost.of.living-adjustment, education-allowance* y *vacation*, ya que todos los valores del atributo contienen ambas clases, por lo que entre ellos decidiremos cual es el que peor divide a la clase.

Dadas estás reglas, parece que el atributo que mejor divide a la clase es *standby-pay*, ya que, dado un valor del atributo, no hay solapamiento de clases, aunque cabe destacar que es el atributo con más valores ausentes.

3. Práctica 2: Conocimiento del entorno WEKA.

Para el primer ejercicio, hemos convertido el fichero de texto a formato *arff*. Para ello, se ha etiquetado y reescrito corrigiendo algunos valores que podrían generar problemas, como los sí, Si, o yes que encontramos en algunos atributos. También se han omitido los valores que se encontraban vacíos y se ha utilizado la interrogación para los datos faltantes:

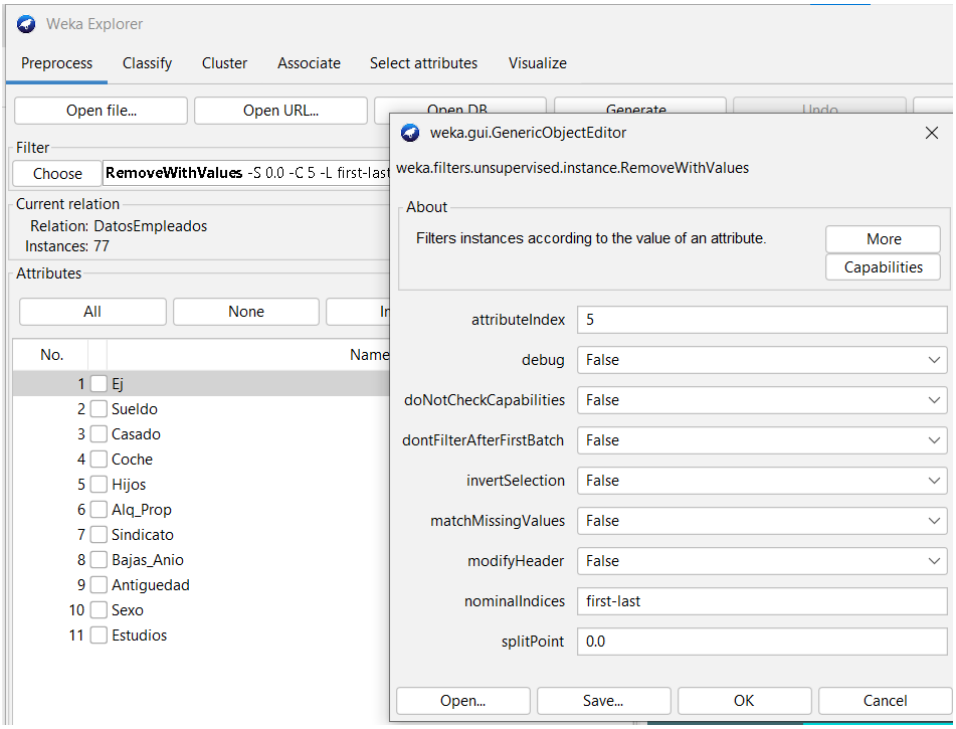
DatosEmpleados: Bloc de notas

Archivo Edición Formato Ver Ayuda

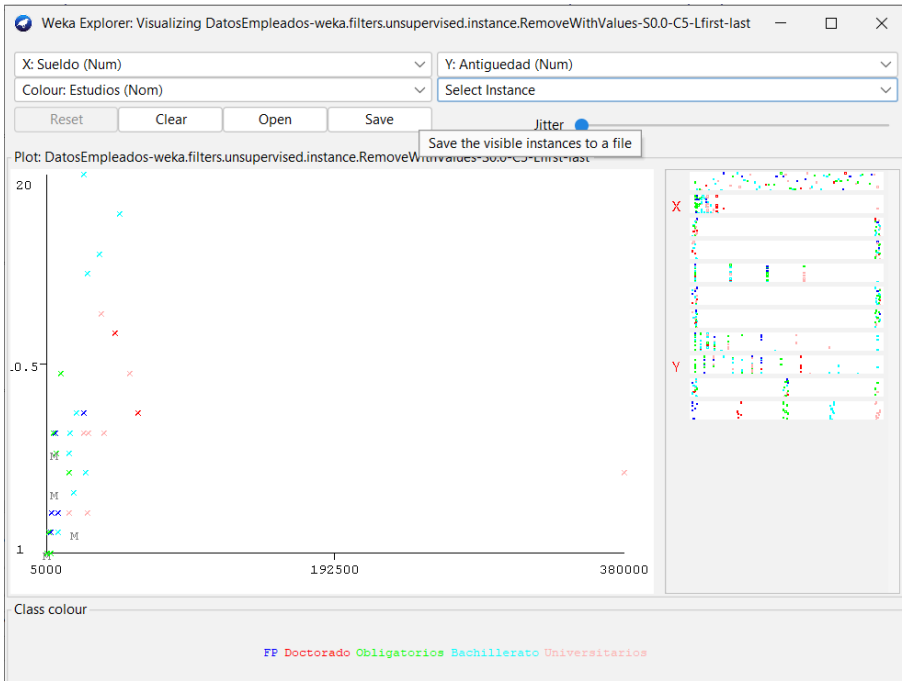
```
@RELATION DatosEmpleados

@ATTRIBUTE 'Ej' numeric
@ATTRIBUTE 'Sueldo' numeric
@ATTRIBUTE 'Casado' {'No','Si'}
@ATTRIBUTE 'Coche' {'No','Si'}
@ATTRIBUTE 'Hijos' numeric
@ATTRIBUTE 'Alq_Prop' {'Alquiler', 'Propio'}
@ATTRIBUTE 'Sindicato' {'No','Si'}
@ATTRIBUTE 'Bajas_Anio' numeric
@ATTRIBUTE 'Antiguedad' numeric
@ATTRIBUTE 'Sexo' {'M','H','V'}
@ATTRIBUTE 'Estudios' {'FP','Doctorado','Obligatorios','Bachillerato','Universitarios'}

@DATA
1,13000,'No','Si',0,'Propio','No',2,3,'M','Obligatorios'
2,32000,'Si','No',2,'Propio','Si',1,15,'M','Bachillerato'
3,12000,'No','No',0,'Alquiler','No',0,6,'H','Obligatorios'
4,41000,'Si','Si',3,'Propio','No',3,13,'H','Universitarios'
5,5000,'No','No',0,?,'Si',0,1,'H',?
6,65000,'No','Si',0,'Propio','No',3,8,'M','Doctorado'
7,53000,'Si','Si',5,'Propio','No',4,18,'M','Bachillerato'
8,23000,'No','Si',0,'Alquiler','Si',7,2,'H',?
9,31000,'Si','No',?,'Propio','Si',0,5,'H','Bachillerato'
10,30000,'Si','Si',2,'Propio','No',1,20,'H','Bachillerato'
11,20000,'No','Si',1,'Alquiler','Si',3,3,'M','Universitarios'
12,13000,'No','No',0,'Propio','No',12,2,'H','Bachillerato'
13,11000,'No','Si',0,'Alquiler','No',0,7,'H','FP'
14,9000,'No','Si',1,'Propio','Si',2,3,'H','FP'
15,60000,'Si','Si',4,'Propio','No',0,10,'M','Universitarios'
16,380000,'No','Si',?,'Propio','No',2,5,'V','Universitarios'
17,6000,'No','Si',0,?,'No',0,1,'V','Obligatorios'
18,30000,'Si','Si',-7,'Propio','Si',10,10,'V','Universitarios'
19,23000,'No','Si',0,'Propio','No',2,4,'M','Bachillerato'
20,43000,'No','Si',3,'Alquiler','Si',20,7,'H','Universitarios'
21,13000,'No','Si',0,'Alquiler','Si',3,3,'M','FP'
22,21000,'Si','Si',1,'Propio','No',1,7,'M','Bachillerato'
23,15000,'Si','Si',2,'Propio','Si',5,10,'H','Obligatorios'
24,30000,'Si','Si',1,'Alquiler','No',15,7,'M','Universitarios'
```

Es una base de datos con los datos personales de ciertos empleados donde tenemos como atributos el número que identifica al empleado, el sueldo de tipo numérico, su estado civil que puede ser casado o no, si tiene coche, la cantidad de hijos, si la vivienda es propia o en alquiler, si pertenece o no a un sindicato, el número de bajas al año, el tiempo de antigüedad, el sexo y el nivel de estudios.
En total, tenemos 77 instancias con 11 atributos:

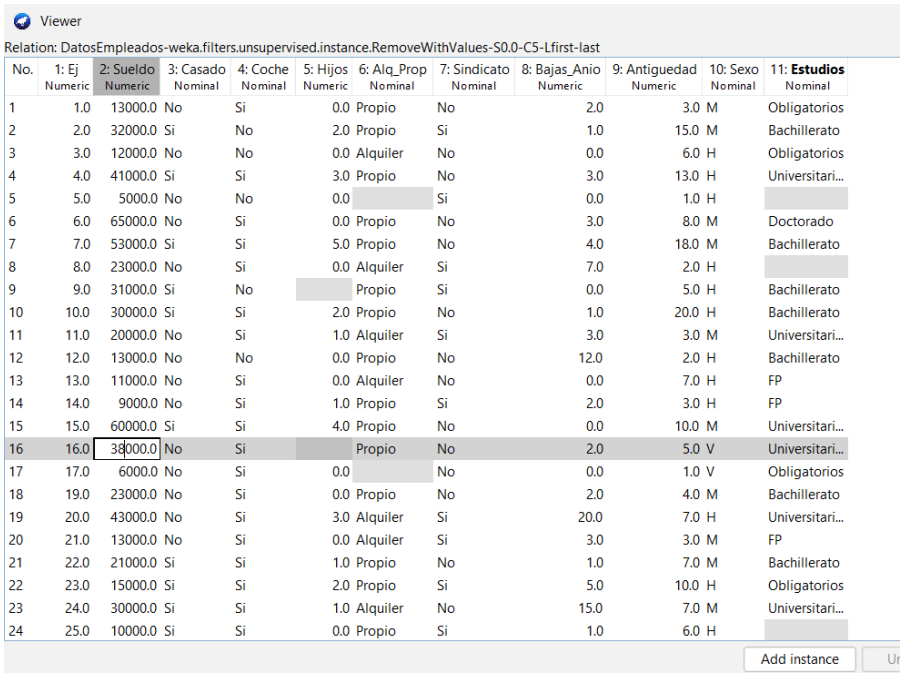
El atributo con más valores ausentes es el nivel de estudios, con 6 valores ausentes (que corresponden al 8%). También tenemos el número de hijos y si la vivienda es alquilada o propia que tienen 4 valores ausentes cada uno (5%).
Ya se han eliminado algunas de las instancias que estaban vacías a la hora de crear el fichero en formato arff. También podríamos eliminar algunas instancias que contienen datos anómalos como la 18, en la que el número de hijos es –7. Para ello podemos hacer lo siguiente:



Tras aplicar este filtro, nos quedaremos con 76 instancias.
Otra instancia que parece tener datos erróneos al estudiar los histogramas es la 16, ya que el sueldo de 380000 es muy distante al resto.
En cuanto a atributos con valores únicos, obviando el primero que es un identificador y cada empleado tendrá el suyo propio, el que más valores únicos tiene es sueldo, con 12 valores únicos, que corresponde al 12%.
El atributo que hace de clase es el nivel de estudios, representado por 5 valores: FP, Doctorado, Bachillerato, Universitarios o Doctorado. A simple vista, podemos obtener resultados como que los Doctorados, aun siendo de los que menos instancias tenemos, son los que tienen mayor sueldo y menos bajas tienen e hijos tienen. Tampoco pertenecen a sindicatos.
Por último, podemos emplear la herramienta de visualización 2D para mostrar el atributo antigüedad respecto al sueldo para ver los datos erróneos antes mencionados:



En mi caso, al generar el archivo arff ya se corrigieron los errores en los atributos nominales y, por lo tanto, sólo nos quedan fallos en los atributos numéricos. Para corregir, por ejemplo, el sueldo anómalo de la instancia 16, a cuya cifra le quitaremos un 0 mediante el siguiente procedimiento:



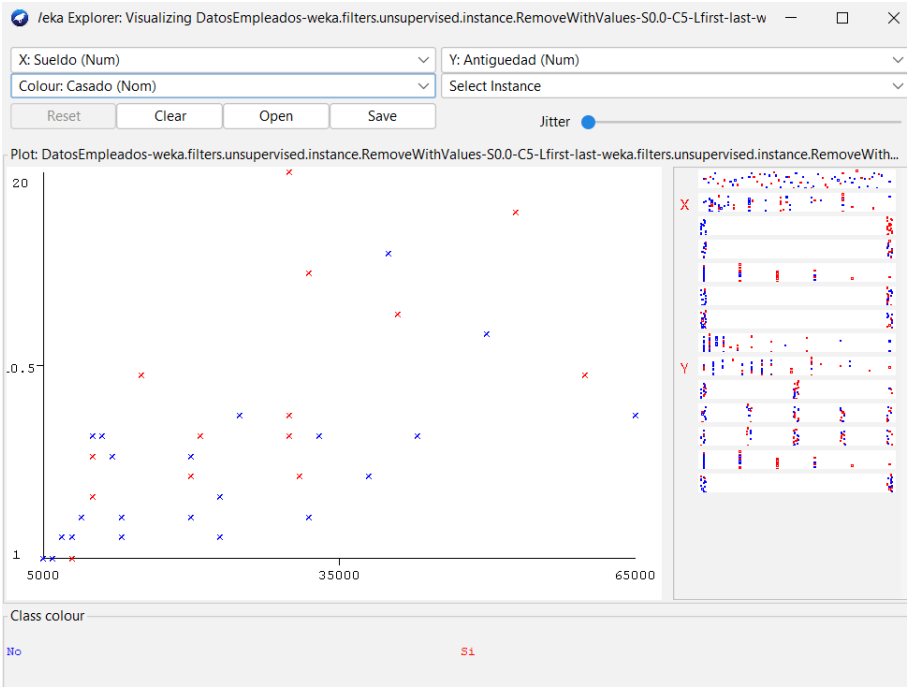
No.	1: Ej Numeric	2: Sueldo Numeric	3: Casado Nominal	4: Coche Nominal	5: Hijos Numeric	6: Alq_Prop Nominal	7: Sindicato Nominal	8: Bajas_Anio Numeric	9: Antiguedad Numeric	10: Sexo Nominal	11: **Estudios** Nominal
1	1.0	13000.0	No	Si	0.0	Propio	No	2.0	3.0	M	Obligatorios
2	2.0	32000.0	Si	No	2.0	Propio	Si	1.0	15.0	M	Bachillerato
3	3.0	12000.0	No	No	0.0	Alquiler	No	0.0	6.0	H	Obligatorios
4	4.0	41000.0	Si	Si	3.0	Propio	No	3.0	13.0	H	Universitari...
5	5.0	5000.0	No	No	0.0		Si	0.0	1.0	H	
6	6.0	65000.0	No	Si	0.0	Propio	No	3.0	8.0	M	Doctorado
7	7.0	53000.0	Si	Si	5.0	Propio	No	4.0	18.0	M	Bachillerato
8	8.0	23000.0	No	Si	0.0	Alquiler	Si	7.0	2.0	H	
9	9.0	31000.0	Si	No		Propio	Si	0.0	5.0	H	Bachillerato
10	10.0	30000.0	Si	Si	2.0	Propio	No	1.0	20.0	H	Bachillerato
11	11.0	20000.0	No	Si	1.0	Alquiler	Si	3.0	3.0	M	Universitari...
12	12.0	13000.0	No	No	0.0	Propio	No	12.0	2.0	H	Bachillerato
13	13.0	11000.0	No	Si	0.0	Alquiler	No	0.0	7.0	H	FP
14	14.0	9000.0	No	Si	1.0	Propio	Si	2.0	3.0	H	FP
15	15.0	60000.0	Si	Si	4.0	Propio	No	0.0	10.0	M	Universitari...
16	16.0	38000.0	No	Si		Propio	No	2.0	5.0	V	Universitari...
17	17.0	6000.0	No	Si	0.0		No	0.0	1.0	V	Obligatorios
18	19.0	23000.0	No	Si	0.0	Propio	No	2.0	4.0	M	Bachillerato
19	20.0	43000.0	No	Si	3.0	Alquiler	Si	20.0	7.0	H	Universitari...
20	21.0	13000.0	No	Si	0.0	Alquiler	Si	3.0	3.0	M	FP
21	22.0	21000.0	Si	Si	1.0	Propio	No	1.0	7.0	M	Bachillerato
22	23.0	15000.0	Si	Si	2.0	Propio	Si	5.0	10.0	H	Obligatorios
23	24.0	30000.0	Si	Si	1.0	Alquiler	No	15.0	7.0	M	Universitari...
24	25.0	10000.0	Si	Si	0.0	Propio	Si	1.0	6.0	H	

También podríamos considerar erróneos los valores ‘V’ para el atributo sexo, pero al estar repetido hasta en 4 instancias, se ha decidido tomarlo como un dato válido.
Continuando con los datos ausentes, vamos a considerar que, si el campo Estudios está ausente, el nivel de estudios será obligatorio. En cuanto al

número de hijos, se ha considerado que, para estudios obligatorios y bachillerato, la cantidad de hijos sea 1, y 2 para universitarios, a modo de promedio de los datos que tenemos. Y para la vivienda, supondremos que siempre será propia, al ser la más frecuente. Para ello crearemos copias de dichos atributos, con el fin de conservar los originales.
Por último, tras guardar el nuevo archivo con datos correctos, vamos a visualizar nuevamente los datos:

- Si vemos la antigüedad frente al sueldo en función de los estudios:

- Si ahora lo hacemos en función del estado civil:



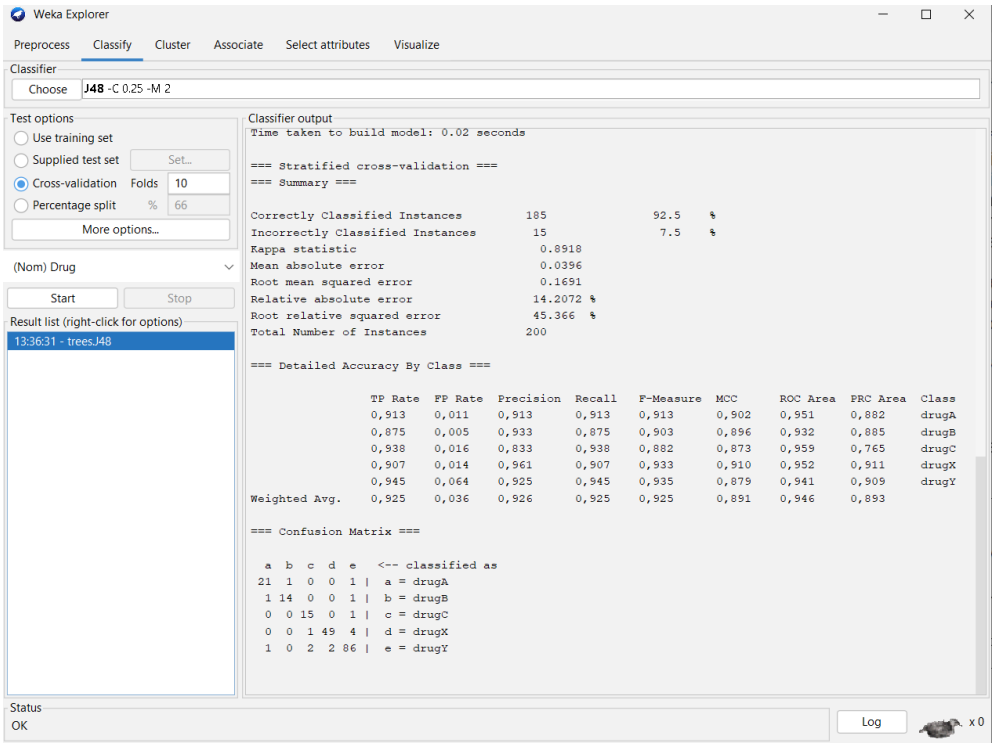
- Por último, comentaremos el sueldo frente a los estudios:

4. Práctica 3: Técnicas básicas de preprocesamiento con WEKA.

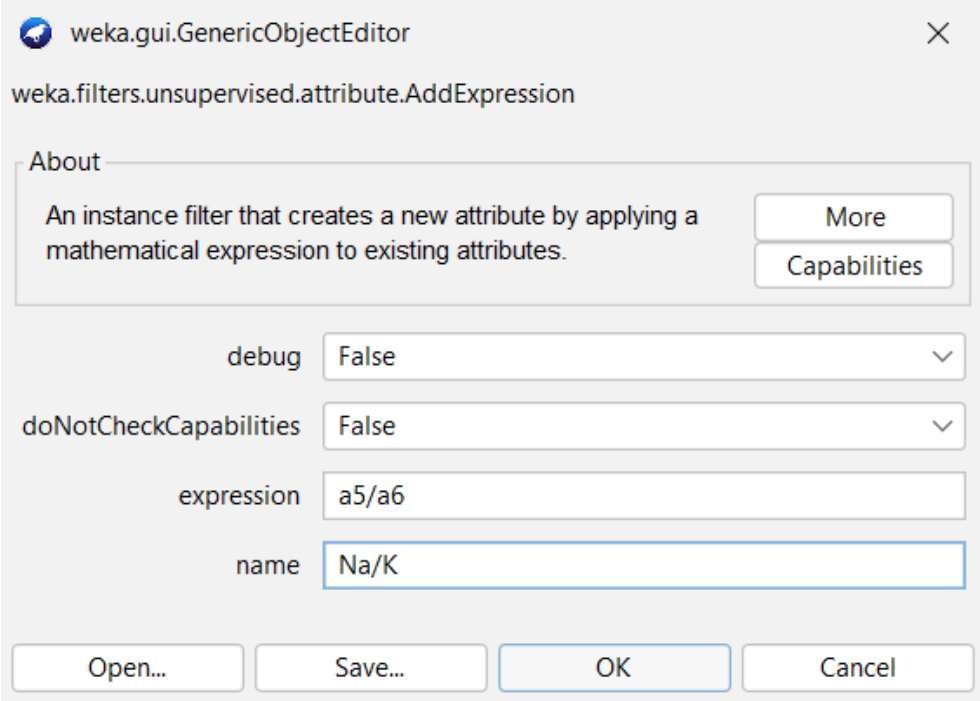
En esta parte trabajaremos con el set de datos drug1n.arff. En el vemos el registro de 200 personas en base a siete atributos, entre los cuales se distingue el de clase: Drug. En este distinguen los 5 fármacos posibles: DrugA, DrugB, DrugC, DrugX, DrugY.

El resto de los atributos son:
- Age: Edad.
- Sex: Sexo.
- BP (Blood Pressure): Tensión sanguínea.
- Cholesterol: nivel de colesterol.
- Na: Nivel de sodio en la sangre.
- K: Nivel de potasio en la sangre.

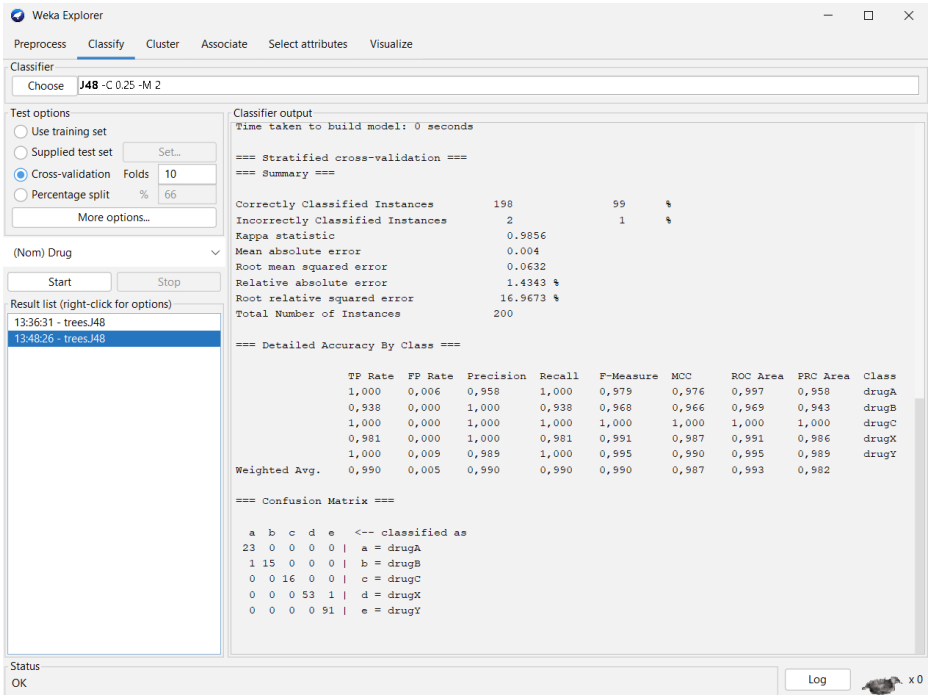
Al examinar los datos, no se observan datos faltantes ni anómalos en del set. Si aplicamos el algoritmo J48, podemos obtener los siguientes resultados:



Al observar los datos, podemos ver que entre los campos Na y K, existe algún tipo de relación donde la drugY puede separarse del resto en función del valor de K. Para ello vamos a combinar ambos campos atributos de la siguiente manera:



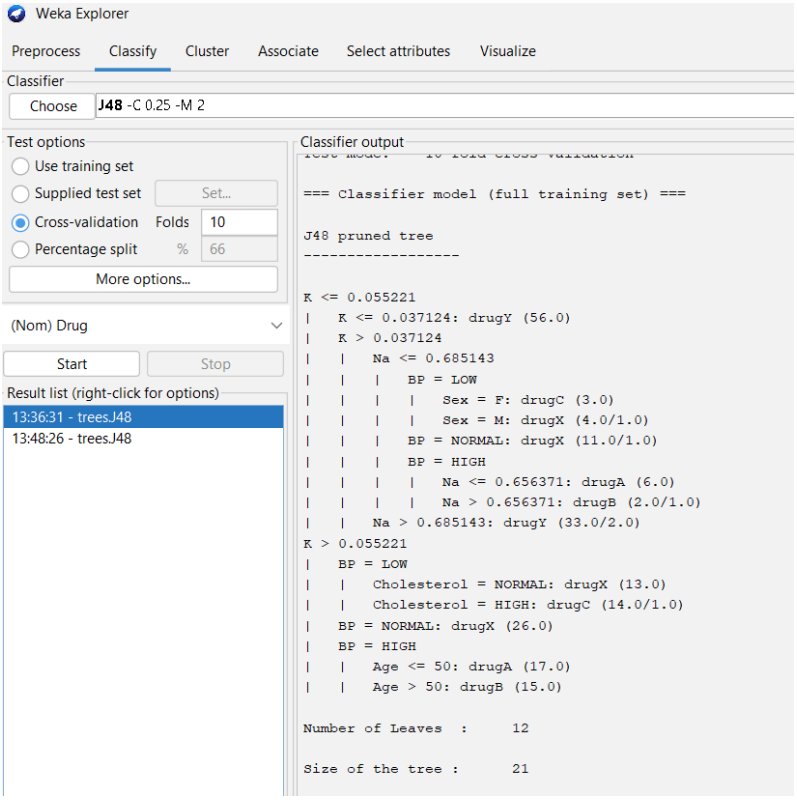
Si volvemos a probar a clasificarlos mediante el método J48, obtenemos:



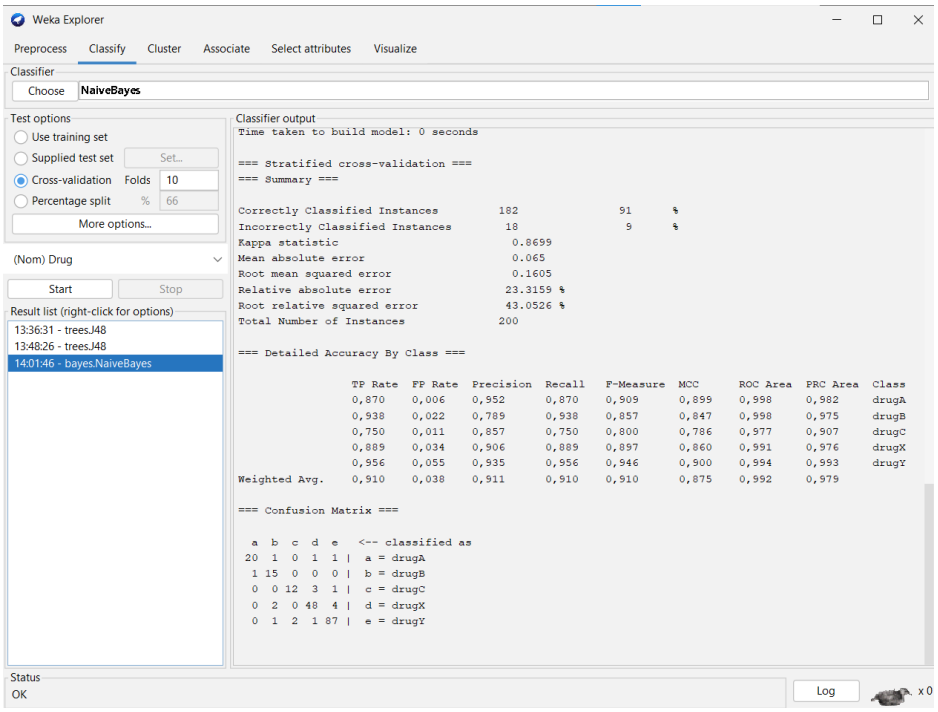
Como vemos, antes de aplicar la modificación, el modelo erraba al clasificar 15 instancias, lo correspondiente a predecir el 92.5% de los casos. Con el nuevo atributo combinado, se consigue una precisión del 99%, fallando sólo en la clasificación de dos ejemplares.
El tamaño del árbol generado es de 10, mientras que antes se generaba un árbol de tamaño 21.

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     DrugData-weka.filters.unsupervised.attribute.AddExpression-Ea5/a6-NNa/K
Instances:    200
Attributes:   8
              Age
              Sex
              BP
              Cholesterol
              Na
              K
              Drug
              Na/K
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

Na/K <= 14.641961
|   BP = LOW
|   |   Cholesterol = NORMAL: drugX (18.0)
|   |   Cholesterol = HIGH: drugC (16.0)
|   BP = NORMAL: drugX (36.0)
|   BP = HIGH
|   |   Age <= 50: drugA (23.0)
|   |   Age > 50: drugB (16.0)
Na/K > 14.641961: drugY (91.0)

Number of Leaves  :     6

Size of the tree :      10
```



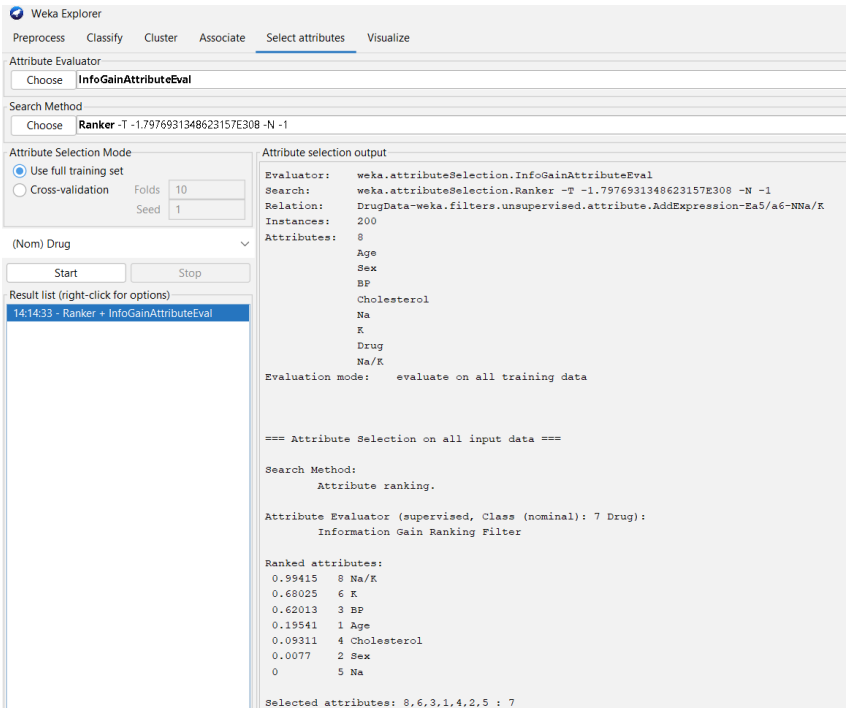
Si a continuación aplicamos el método Naive Bayes, que es un método que reduce su calidad en función de la cantidad de atributos que no son relevantes, se tiene:



Vemos que con este modelo se alcanza una precisión del 91%, con 182 instancias correctamente clasificadas.

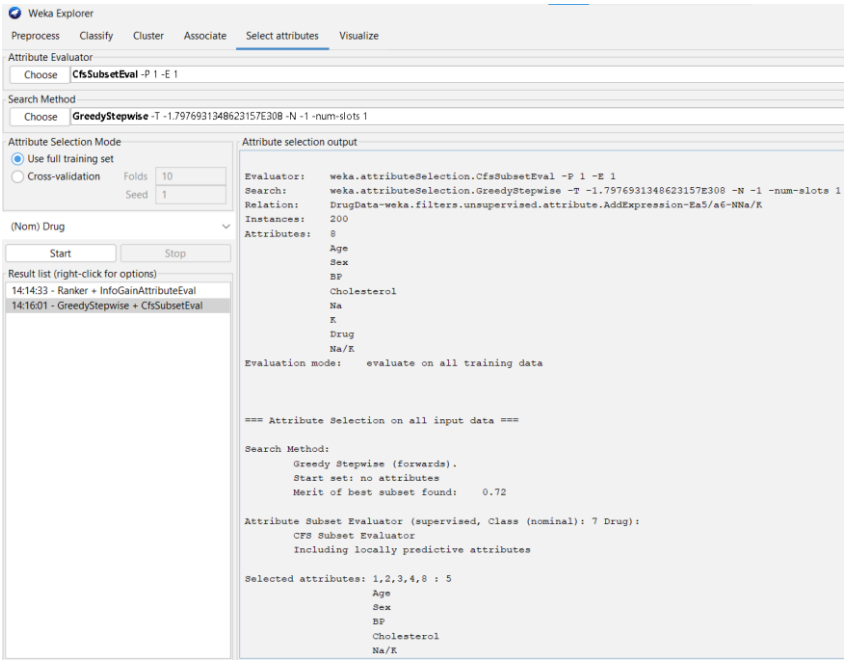
Continuaremos realizando una selección de características para eliminar los atributos que no son relevantes para el problema mediante diferentes técnicas:

- Evaluación individual de atributos (método de búsqueda Ranker):



Según el evaluador de ganancia de la información como métrica, los atributos que son más importantes para predecir la variable objetivo Drug son principalmente Na/K, K y BP, siendo Na, Sex y Cholesterol los menos relevantes.

- Evaluación de conjuntos de atributos *filter* (método de búsqueda Greedy):

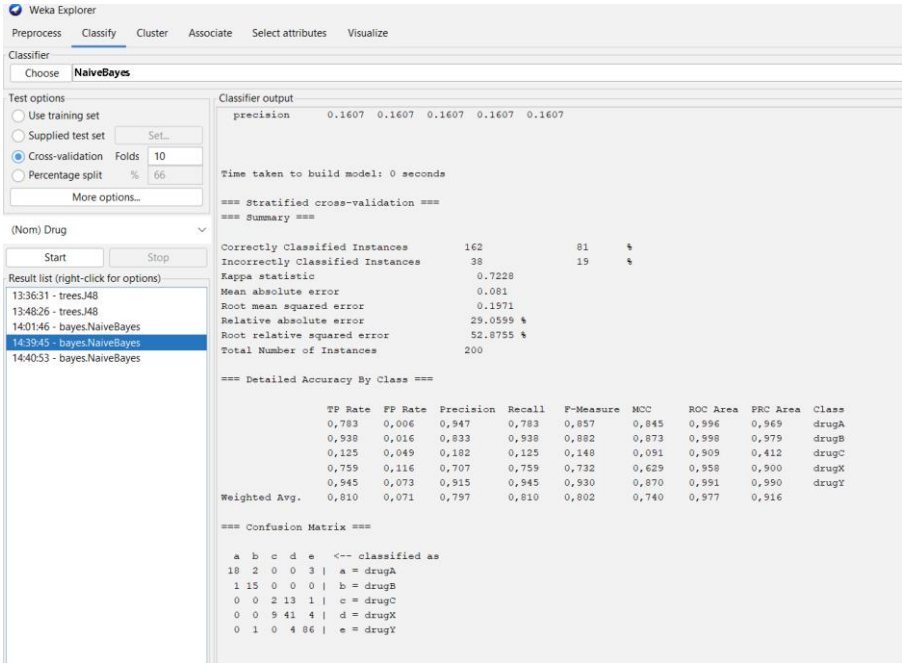


Según el método greedy y el evaluador CFS, los atributos más relevantes son: Age, Sex, BP, Cholesterol y NA/K, siendo Na y K los menos relevantes.

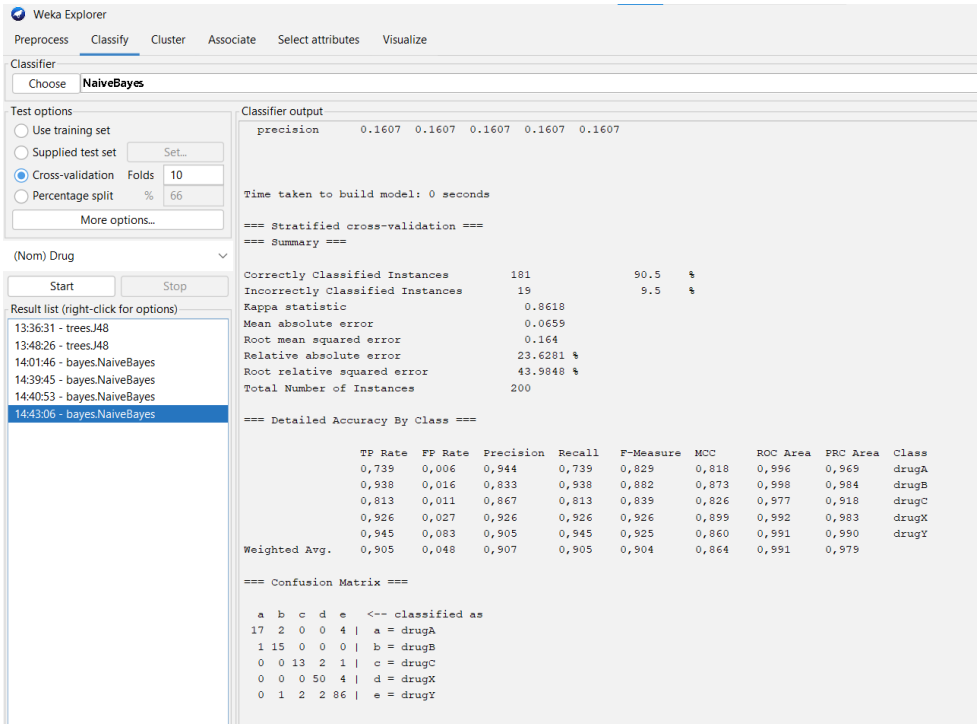
- Evaluación de conjuntos de atributos *wrapper* (método de búsqueda exhaustiva):

```
Relation:     DrugData-weka.filters.unsupervised.attribute.AddExpression-Ea5/a6-NNa/K
Instances:    200
Attributes:   8
              Age
              Sex
              BP
              Cholesterol
              Na
              K
              Drug
              Na/K
Evaluation mode:    evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 30
        Merit of best subset found:    0.455

Attribute Subset Evaluator (supervised, Class (nominal): 7 Drug):
        Wrapper Subset Evaluator
        Learning scheme: weka.classifiers.rules.ZeroR
        Scheme options:
        Subset evaluation: classification accuracy
        Number of folds for accuracy estimation: 5

Selected attributes:
```

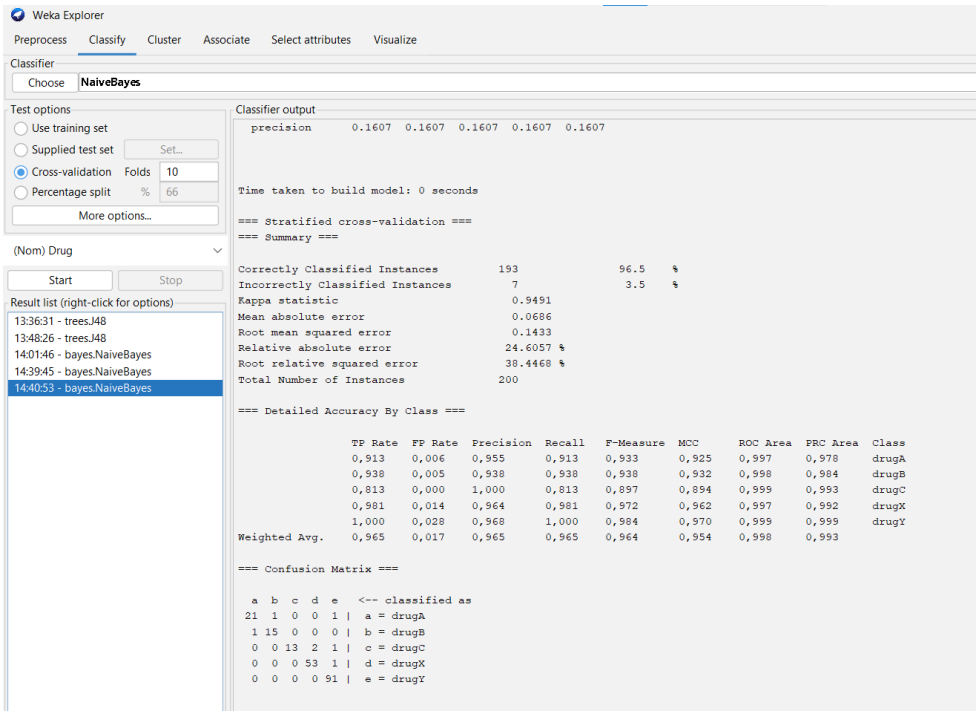
Si ahora volvemos a aplicar el método *Naive Bayes* a cada una de las muestras omitiendo los atributos menos relevantes, obtenemos los siguientes resultados:
Para el método de búsqueda *Ranker*, si omitimos los 3 atributos menos relevantes, no se tiene una precisión demasiado buena:



Sin embargo, al omitir solo los atributos Sex y Na, se obtiene una mayor precisión:



Aunque, quien mejor resultado consigue, es la del método Greedy al omitir los atributos Na y K, con un 96.5% de acierto:



Si ahora aplicamos el algoritmo J48 que hemos entrenado con los datos del primer hospital para cada uno de los hospitales restantes en la fase de test, obtenemos:

```
Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         362               90.5    %
Incorrectly Classified Instances        38                9.5    %
Kappa statistic                          0.8653
Mean absolute error                      0.0444
Root mean squared error                  0.1871
Relative absolute error                 15.756  %
Root relative squared error             49.6531 %
Total Number of Instances              400

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,022    0,814      1,000    0,897      0,892    0,989     0,814     drugA
                 0,742    0,005    0,920      0,742    0,821      0,814    0,870     0,762     drugB
                 0,894    0,020    0,857      0,894    0,875      0,858    0,929     0,732     drugC
                 0,855    0,014    0,962      0,855    0,905      0,872    0,952     0,916     drugX
                 0,953    0,074    0,905      0,953    0,928      0,874    0,964     0,941     drugY
Weighted Avg.    0,905    0,040    0,909      0,905    0,904      0,868    0,951     0,884

=== Confusion Matrix ===

   a   b   c   d   e   <-- classified as
  35   0   0   0   0 |   a = drugA
   5  23   0   0   3 |   b = drugB
   0   0  42   2   3 |   c = drugC
   0   0   6 100  11 |   d = drugX
   3   2   1   2 162 |   e = drugY
```

Classifier output

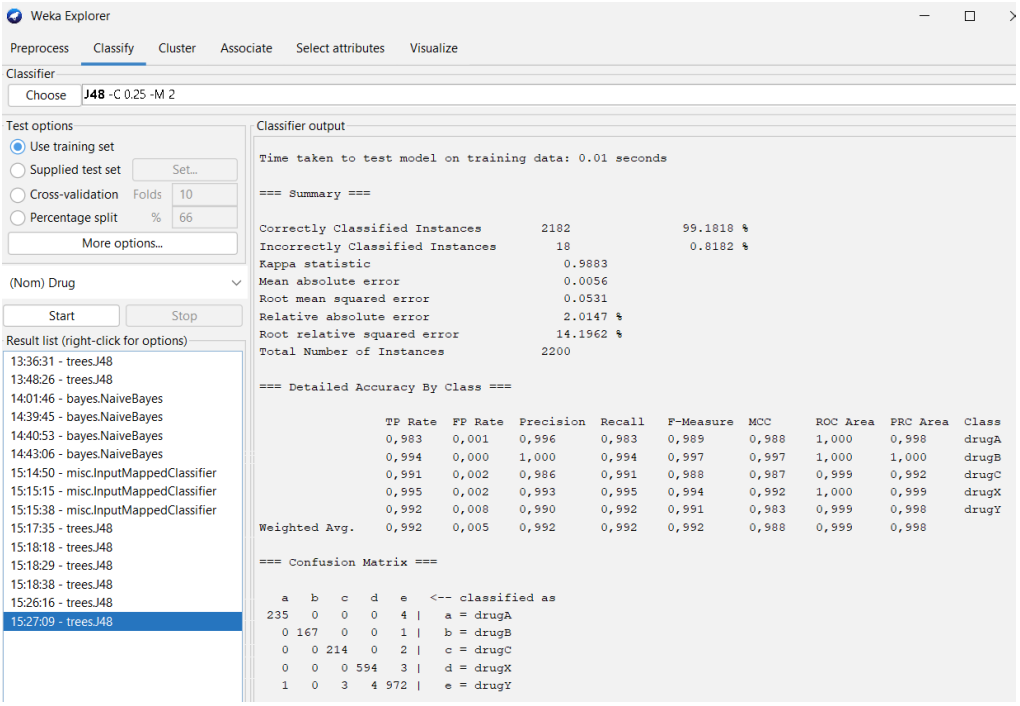
```
Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         538               89.6667 %
Incorrectly Classified Instances        62               10.3333 %
Kappa statistic                          0.8512
Mean absolute error                      0.0481
Root mean squared error                  0.1958
Relative absolute error                 17.2061 %
Root relative squared error             52.3653 %
Total Number of Instances              600

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,886    0,034    0,775      0,886    0,827      0,804    0,926     0,700     drugA
                 0,692    0,011    0,857      0,692    0,766      0,751    0,844     0,719     drugB
                 0,760    0,009    0,884      0,760    0,817      0,805    0,872     0,661     drugC
                 0,923    0,032    0,911      0,923    0,917      0,888    0,975     0,933     drugX
                 0,949    0,058    0,931      0,949    0,940      0,889    0,966     0,955     drugY
Weighted Avg.    0,897    0,040    0,898      0,897    0,895      0,860    0,945     0,874

=== Confusion Matrix ===

   a   b   c   d   e   <-- classified as
  62   6   0   0   2 |   a = drugA
  14  36   0   0   2 |   b = drugB
   0   0  38   6   6 |   c = drugC
   0   0   3 144   9 |   d = drugX
   4   0   2   8 258 |   e = drugY
```

Classifier output

```
Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         894               89.4    %
Incorrectly Classified Instances       106               10.6    %
Kappa statistic                          0.8482
Mean absolute error                      0.0501
Root mean squared error                  0.1985
Relative absolute error                 17.9057 %
Root relative squared error             53.0307 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,883    0,029    0,790      0,883    0,834      0,814    0,927     0,711     drugA
                 0,681    0,009    0,855      0,681    0,758      0,748    0,839     0,699     drugB
                 0,816    0,021    0,816      0,816    0,816      0,794    0,890     0,638     drugC
                 0,911    0,032    0,914      0,911    0,913      0,881    0,959     0,933     drugX
                 0,937    0,054    0,933      0,937    0,935      0,883    0,962     0,945     drugY
Weighted Avg.    0,894    0,039    0,895      0,894    0,893      0,856    0,941     0,867

=== Confusion Matrix ===

   a   b   c   d   e   <-- classified as
  98   5   0   0   8 |   a = drugA
  18  47   0   0   4 |   b = drugB
   0   0  84  12   7 |   c = drugC
   0   0  13 246  11 |   d = drugX
   8   3   6  11 419 |   e = drugY
```

Para el fichero drug2n como test -> 90.5%
Para el fichero drug3n como test -> 89.667%
Para el fichero drug4n como test -> 89.4%

Por último, se han fusionado todos los datos en el fichero drugn.arff y mediante cross-validation obtenemos una precisión del 99.18%:



A continuación, podemos generar otros conjuntos de datos de test mediante los filtros indicados en la guía, cuyos resultados son 99.09% para datos balanceados y 98.18% para los datos seleccionados aleatoriamente:

Classifier output

```
Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         216               98.1818 %
Incorrectly Classified Instances         4                1.8182 %
Kappa statistic                          0.9738
Mean absolute error                      0.0088
Root mean squared error                  0.0752
Relative absolute error                  3.1678 %
Root relative squared error             20.2219 %
Total Number of Instances              220

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,005    0,960      1,000    0,980      0,977    1,000     0,995     drugA
                 0,941    0,000    1,000      0,941    0,970      0,968    1,000     1,000     drugB
                 1,000    0,010    0,895      1,000    0,944      0,941    0,998     0,945     drugC
                 1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     drugX
                 0,970    0,008    0,990      0,970    0,980      0,963    0,998     0,997     drugY
Weighted Avg.    0,982    0,005    0,983      0,982    0,982      0,974    0,999     0,994

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 24  0  0  0  0 |  a = drugA
  0 16  0  0  1 |  b = drugB
  0  0 17  0  0 |  c = drugC
  0  0  0 63  0 |  d = drugX
  1  0  2  0 96 |  e = drugY
```

Classifier output

```
Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         218               99.0909 %
Incorrectly Classified Instances         2                0.9091 %
Kappa statistic                          0.987
Mean absolute error                      0.0072
Root mean squared error                  0.058
Relative absolute error                  2.5758 %
Root relative squared error             15.4997 %
Total Number of Instances              220

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     drugA
                 1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     drugB
                 1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     drugC
                 0,967    0,000    1,000      0,967    0,983      0,977    0,997     0,992     drugX
                 1,000    0,016    0,980      1,000    0,990      0,982    0,997     0,993     drugY
Weighted Avg.    0,991    0,007    0,991      0,991    0,991      0,986    0,998     0,995

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 23  0  0  0  0 |  a = drugA
  0 17  0  0  0 |  b = drugB
  0  0 22  0  0 |  c = drugC
  0  0  0 58  2 |  d = drugX
  0  0  0  0 98 |  e = drugY
```

Por lo que no puede apreciarse mucha diferencia en este caso cuando los datos están o no balanceados.

5. Práctica 4: Clasificación: Almacenamiento de modelos y predicción de nuevos casos.

En esta práctica se utilizará el fichero vehicle arff con el que construiremos 3 modelos de clasificación utilizando 10-validación cruzada:

- Para el algoritmo J48:

Classifier output

```
Number of Leaves  :     98

Size of the tree :      195


Time taken to build model: 0.02 seconds

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

Correctly Classified Instances         613               72.4586 %
Incorrectly Classified Instances       233               27.5414 %
Kappa statistic                          0.6328
Mean absolute error                      0.1415
Root mean squared error                  0.3355
Relative absolute error                 37.7493 %
Root relative squared error             77.4887 %
Total Number of Instances              846

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,613    0,166    0,553      0,613    0,582      0,433    0,784     0,544     opel
                 0,456    0,143    0,524      0,456    0,488      0,328    0,758     0,473     saab
                 0,950    0,019    0,945      0,950    0,947      0,929    0,977     0,945     bus
                 0,889    0,040    0,872      0,889    0,881      0,843    0,932     0,793     van
Weighted Avg.    0,725    0,093    0,722      0,725    0,722      0,631    0,862     0,687

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 130  74   2   6 |   a = opel
  96  99   6  16 |   b = saab
   3   4 207   4 |   c = bus
   6  12   4 177 |   d = van
```

- Para el algoritmo Random Forest:

Classifier output

```
RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.25 seconds

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

Correctly Classified Instances         643               76.0047 %
Incorrectly Classified Instances       203               23.9953 %
Kappa statistic                          0.6801
Mean absolute error                      0.1569
Root mean squared error                  0.2669
Relative absolute error                 41.8471 %
Root relative squared error             61.6464 %
Total Number of Instances              846

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,547    0,123    0,598      0,547    0,571      0,437    0,882     0,693     opel
                 0,562    0,145    0,573      0,562    0,567      0,420    0,871     0,665     saab
                 0,982    0,014    0,960      0,982    0,971      0,960    0,999     0,996     bus
                 0,960    0,039    0,884      0,960    0,920      0,896    0,995     0,985     van
Weighted Avg.    0,760    0,081    0,752      0,760    0,755      0,676    0,936     0,833

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 116  84   2  10 |   a = opel
  76 122   6  13 |   b = saab
   0   2 214   2 |   c = bus
   2   5   1 191 |   d = van
```

- Para el algoritmo 1R:

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

Correctly Classified Instances         439               51.8913 %
Incorrectly Classified Instances       407               48.1087 %
Kappa statistic                          0.3594
Mean absolute error                      0.2405
Root mean squared error                  0.4905
Relative absolute error                 64.1715 %
Root relative squared error            113.288  %
Total Number of Instances              846

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,368    0,159    0,436      0,368    0,399      0,221    0,604     0,319     opel
                 0,410    0,172    0,452      0,410    0,430      0,246    0,619     0,337     saab
                 0,592    0,131    0,611      0,592    0,601      0,466    0,731     0,467     bus
                 0,719    0,179    0,552      0,719    0,624      0,496    0,770     0,463     van
Weighted Avg.    0,519    0,160    0,512      0,519    0,512      0,356    0,680     0,395

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  78  69  25  40 |   a = opel
  62  89  22  44 |   b = saab
  29  28 129  32 |   c = bus
  10  11  35 143 |   d = van
```

Como vemos, quien consigue mejores resultados es el algoritmo Random Forest con un 76.0047% de instancias correctamente clasificadas (643 instancias, frente a 203 mal clasificadas, que se corresponden al 23.9953% de los datos), seguido por el J48 con un 72.4586% (613 instancias, frente a 233 mal clasificadas, que se corresponden al 27.5414% de los datos) y, por último, el algoritmo 1R con 439 instancias correctamente clasificadas (51.8913%, frente a 407 mal clasificados, que se corresponden con el 48.1087%).
Para la clase saab, tenemos que el primer modelo consigue una precisión del 52.4%, el segundo una precisión del 57.3% y el último una precisión del 45.2%.
Las medidas que suelen emplearse para evaluar los modelos son:

- **TP Rate**: Proporción de casos positivos que fueron correctamente identificados. Se define como: TP Rate = TP/(TP+FN)
- **FP Rate**: Proporción de casos negativos que fueron incorrectamente identificados como positivos. Se define como: FP Rate = FP/(FP+TN)
- **Precision**: Proporción de identificaciones positivas que fueron correctas. Se define como: Precision = TP/(TP+FP)
- **Recall:** Proporción de casos positivos que fueron correctamente identificados respecto al total de casos positivos reales. Se define como: Recall = TP/(TP+FN)
- **F-Measure**: Medida que combina la precisión y el recall en un solo valor. Es útil para clases desbalanceadas y se mide:
  F-Measure = 2*(Precision*Recall) / (Precision+Recall)

Para la marca Opel, empleando el modelo generado mediante el algoritmo J48, tenemos que:

- Los verdaderos positivos (TP) son 130.
- Los FP clasificados como opel que no lo son: 96 + 3 + 6 = 105.

- Los TN clasificados correctamente como no opel son: 121 + 215 + 193 = 529.
- Los FN clasificados como no opel que son opel son: 74 + 2 + 6 = 82.
- Considerando los datos anteriores obtenidos de la matriz de confusión, podemos realizar los cálculos solicitados:

  TP Rate = 130 / (130 + 82) = 0.6132
  FP Rate = 105 / (105 + 529) = 0.1656
  Precision = 130 / (130 + 105) = 0.5532
  Recall = 130 / (130 + 82) = 0.6132
  F-Measure = 2 * (0.5532 * 0.6132) / (0.5532 + 0.6132) = 0.5816

Los coches de tipo van clasificados como opel son 6 mientras que los clasificados como saab son 12. Basándonos en la métrica de precisión para cada marca, tenemos que los correctamente clasificados son:

- Opel: 55.3%
- Saab: 52.4%
- Bus: 94.5%
- Van: 87.2%

Los errores cometidos para la clase opel en los distintos modelos podemos obtenerlos de cualquiera de las métricas anteriores:

- **J48**:

- TR Rate: 0,613
- FP Rate: 0,166
- Precision: 0,553
- Recall: 0,613
- F-Measure: 0,582

- **Random Forest**:

- TR Rate: 0,547
- FP Rate: 0,123
- Precision: 0,598
- Recall: 0,547
- F-Measure: 0,571

- **R1**:

- TR Rate: 0,368
- FP Rate: 0,123
- Precision: 0,436
- Recall: 0,368
- F-Measure: 0,399

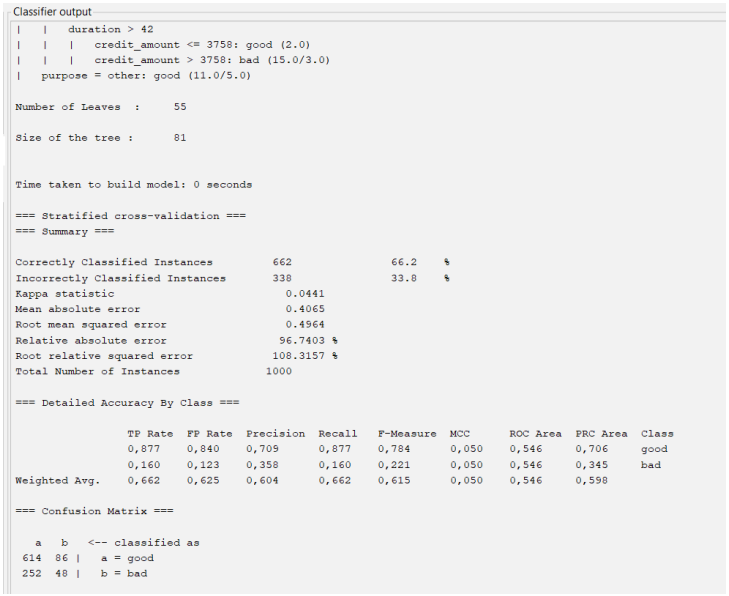
Como vemos, el **J48** consigue menores tasas de error si consideramos *TR Rate, Recall y F-Measure*, mientras que ***Random Forest*** tiene menor *FP Rate* y mejor *Precision*. Además, vemos que los errores al clasificar correctamente esta clase crecen en comparación con los del resto de clases.
A continuación, estudiaremos la base de dato contenida en el fichero credit-g_simple.arff, que contiene 1000 ejemplos con clientes de una entidad bancaria que demandaron un crédito. Consta de 7 atributos numéricos y 13 nominales.

La clase es binaria e indica si el cliente puede ser considerado como fiable para concederle el crédito o no.
El modelo que prediga si un cliente es fiable se construirá en base a la siguiente información:

- duración del crédito (en meses)
- propósito del crédito
- cantidad que solicita
- estado civil
- edad
- situación laboral

Se han construido tres modelos de clasificación usando empleando los siguientes métodos: J48, Naive Bayes y R1, utilizando 10-validación cruzada para cada uno de los modelos y se han almacenado los modelos generados:

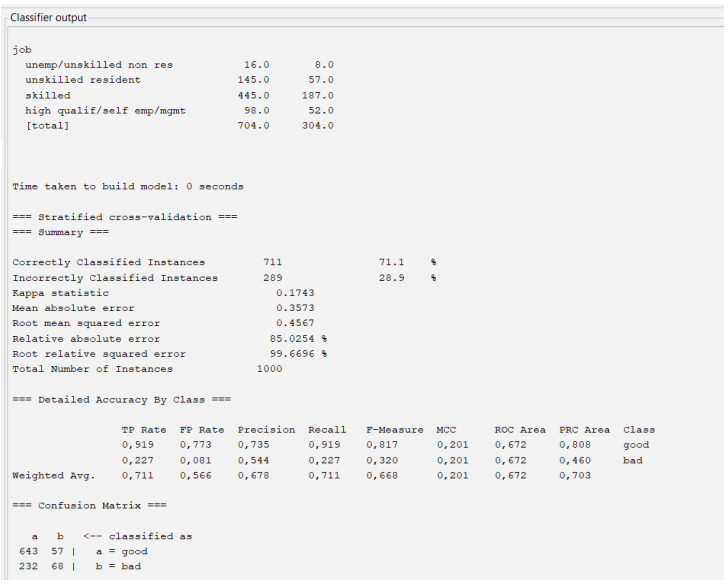
- Con el J48 tenemos:



Classifier output

```
|   |   duration > 42
|   |   |   credit_amount <= 3758: good (2.0)
|   |   |   credit_amount > 3758: bad (15.0/3.0)
|   purpose = other: good (11.0/5.0)

Number of Leaves  :     55

Size of the tree :      81


Time taken to build model: 0 seconds

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

Correctly Classified Instances         662               66.2    %
Incorrectly Classified Instances       338               33.8    %
Kappa statistic                          0.0441
Mean absolute error                      0.4065
Root mean squared error                  0.4964
Relative absolute error                 96.7403 %
Root relative squared error            108.3157 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,877    0,840    0,709      0,877    0,784      0,050    0,546     0,706     good
                 0,160    0,123    0,358      0,160    0,221      0,050    0,546     0,345     bad
Weighted Avg.    0,662    0,625    0,604      0,662    0,615      0,050    0,546     0,598

=== Confusion Matrix ===

   a   b   <-- classified as
 614  86 |   a = good
 252  48 |   b = bad
```

- Con Naive Bayes:



Classifier output

```
job
  unemp/unskilled non res          16.0       8.0
  unskilled resident              145.0      57.0
  skilled                         445.0     187.0
  high qualif/self emp/mgmt        98.0      52.0
  [total]                         704.0     304.0



Time taken to build model: 0 seconds

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

Correctly Classified Instances         711               71.1    %
Incorrectly Classified Instances       289               28.9    %
Kappa statistic                          0.1743
Mean absolute error                      0.3573
Root mean squared error                  0.4567
Relative absolute error                 85.0254 %
Root relative squared error             99.6696 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,919    0,773    0,735      0,919    0,817      0,201    0,672     0,808     good
                 0,227    0,081    0,544      0,227    0,320      0,201    0,672     0,460     bad
Weighted Avg.    0,711    0,566    0,678      0,711    0,668      0,201    0,672     0,703

=== Confusion Matrix ===

   a   b   <-- classified as
 643  57 |   a = good
 232  68 |   b = bad
```

- Con Random Forest:

```
Correctly Classified Instances         684               68.4    %
Incorrectly Classified Instances       316               31.6    %
Kappa statistic                          0.1702
Mean absolute error                      0.37
Root mean squared error                  0.4537
Relative absolute error                 88.0629 %
Root relative squared error             98.9998 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,844    0,690    0,741      0,844    0,789      0,176    0,666     0,823     good
                 0,310    0,156    0,460      0,310    0,371      0,176    0,666     0,445     bad
Weighted Avg.    0,684    0,530    0,657      0,684    0,663      0,176    0,666     0,709

=== Confusion Matrix ===

   a   b   <-- classified as
 591 109 |   a = good
 207  93 |   b = bad
```

- Con R1:

```
Time taken to build model: 0.02 seconds

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

Correctly Classified Instances         661               66.1    %
Incorrectly Classified Instances       339               33.9    %
Kappa statistic                          0.0552
Mean absolute error                      0.339
Root mean squared error                  0.5822
Relative absolute error                 80.6802 %
Root relative squared error            127.0545 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,867    0,820    0,712      0,867    0,782      0,061    0,524     0,710     good
                 0,180    0,133    0,367      0,180    0,242      0,061    0,524     0,312     bad
Weighted Avg.    0,661    0,614    0,608      0,661    0,620      0,061    0,524     0,591

=== Confusion Matrix ===

   a   b   <-- classified as
 607  93 |   a = good
 246  54 |   b = bad
```

Utilizando los tres modelos, vamos a predecir si deberíamos darles el crédito a los siguientes clientes:

12	Vacaciones	1500	Hombre soltero	Trabajador no cualificado	37	¿?
48	Electrónica de consumo	3000	Hombre casado	Trabajador no cualificado	25	¿?
6	Negocios	6500	Mujer soltera	Trabajador cualificado	45	¿?

Las predicciones realizadas por el modelo del J48 son:

```
=== Predictions on test set ===

    inst#     actual  predicted error prediction
        1        1:?     1:good       0.794
        2        1:?      2:bad       0.667
        3        1:?     1:good       0.794
```

Las predicciones realizadas por el modelo de Navie Bayes son:

```
=== Predictions on test set ===

    inst#     actual  predicted error prediction
        1        1:?     1:good       0.762
        2        1:?      2:bad       0.683
        3        1:?     1:good       0.508
```

Las predicciones realizadas por el modelo de Random Forest son:

```
=== Predictions on test set ===

    inst#     actual  predicted error prediction
        1        1:?     1:good       0.737
        2        1:?      2:bad       0.737
        3        1:?     1:good       0.724
```

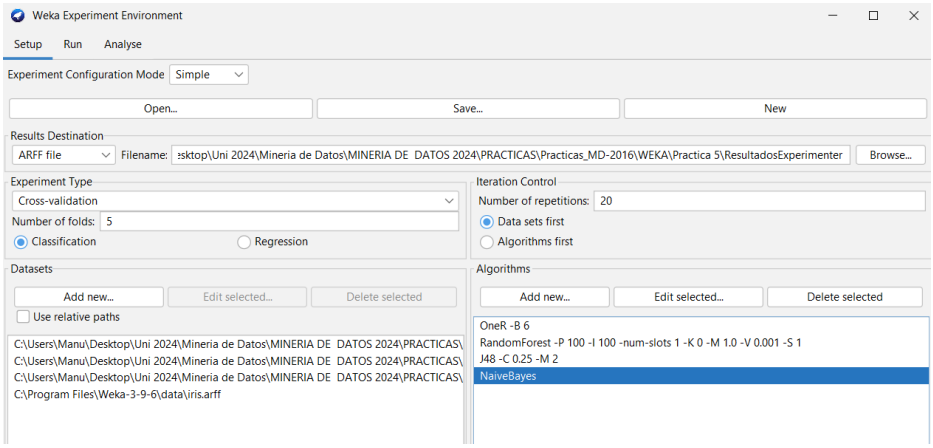
Las predicciones realizadas por el modelo del R1 son:

```
=== Predictions on test set ===

    inst#     actual  predicted error prediction
        1        1:?     1:good       1
        2        1:?     1:good       1
        3        1:?      2:bad       1
```

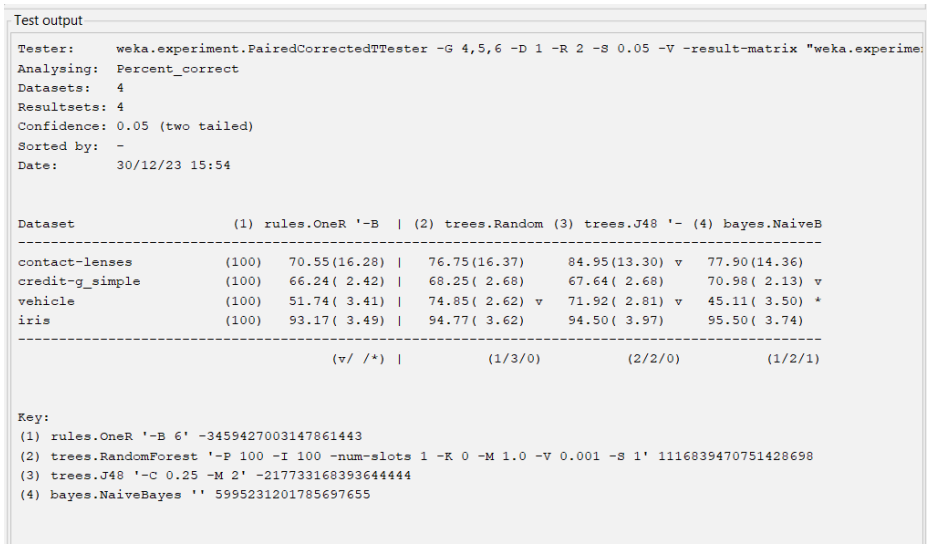
Según los modelos vistos, obviando el R1 donde el error es 1 y no proporcionaría unos resultados confiables, los modelos generados mediante Naive Bayes, Random Forest y J48 nos indican que tanto el primero como el tercer cliente podría recibir el crédito, mientras que el segundo no. Aun así, el error en ambos modelos para el primer cliente es grande también y quizás no proporcione un resultado fiable. Las razones por las que pueden considerar los modelos aptos para el crédito a dichos clientes pueden ser sus edades superiores a 35 años, las duraciones del crédito bajas y sus profesiones.

6. Práctica 5: Clasificación: Uso del *Experimenter* de WEKA y clasificación sensible al coste.

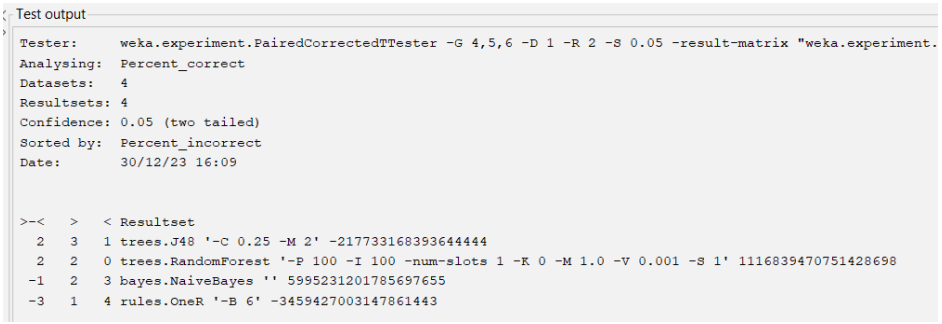
En esta práctica, trabajaremos con el modo Experimenter que nos permite aplicar varios métodos de clasificación para distintos datasets. Para ello, se emplearán los modelos generados por los algoritmos 1R, J48 y Random Forest de la práctica anterior a las bases de datos credit-g.arff, contact-lenses.arff, iris.arff y vehicle.arff, ejecutando 20 veces cada algoritmo usando validación cruzada de 5 hojas:



Al ejecutarlo no aparecen errores y podemos pasar a nalizarlo usando como base el algoritmo 1R, mostrando las desviaciones típicas respecto a la medida:

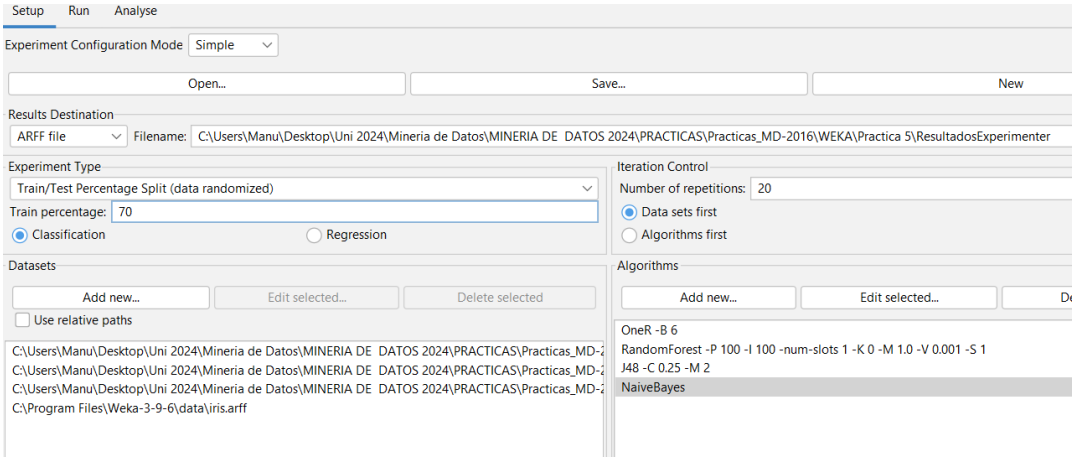


Para la base de datos credit-g.arff, observamos que los resultados son muy similares, al igual que las desviaciones típicas. Quien mejores resultados obtiene es el algoritmo Naive Bayes, sin embargo, para contact-lenses.arff quien consigue mejores resultados, y con diferencia, es el J48, mientras que para vehicle.arff es Random Forest.
Si ahora usamos como base Ranking, los resultados son los siguientes:



Donde se puede ver que los que mejor resultados consiguen son J48 y Random Forest.

A continuación, haremos de nuevo el experimento repitiendo 20 veces cada algoritmo, pero con un 70% de los datos para el entrenamiento, escogidos de forma aleatoria.



Al ejecutarlo, vemos que quien mejores resultados en Ranking consigue es Random Forest, el cual utilizaremos como base:

```
>-<   >   < Resultset
  3   3   0 trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
  1   2   1 trees.J48 '-C 0.25 -M 2' -217733168393644444
 -2   1   3 bayes.NaiveBayes '' 5995231201785697655
 -2   1   3 rules.OneR '-B 6' -3459427003147861443
```

Test output

```
Tester:     weka.experiment.PairedCorrectedTTester -G 3,4,5 -D 1 -R 2 -S 0.05 -V -result-matrix "weka.
Analysing:  Percent_correct
Datasets:   4
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       30/12/23 16:25


Dataset                   (2) trees.RandomFo | (1) rules.OneR ' (3) trees.J48 ' (4) bayes.Naive
-------------------------------------------------------------------------------------------------
contact-lenses            (20)   75.42(6.88) |   63.07(13.25)     85.68(8.83)     75.62(9.04)
credit-g_simple           (20)   67.95(2.56) |   66.57( 1.98)     67.57(1.78)     71.07(1.76)
vehicle                   (20)   75.33(1.85) |   51.46( 2.89) *   70.40(2.26) *   44.07(3.45) *
iris                      (20)   95.11(2.46) |   93.67( 3.08)     94.22(3.77)     96.22(2.98)
-------------------------------------------------------------------------------------------------
                                     (v/ /*) |          (0/3/1)          (0/3/1)          (0/3/1)


Key:
(1) rules.OneR '-B 6' -3459427003147861443
(2) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(3) trees.J48 '-C 0.25 -M 2' -217733168393644444
(4) bayes.NaiveBayes '' 5995231201785697655
```

Podemos observar que 1R consigue peores resultados para todos los datasets, mientras que J48 consigue sólo mejores resultados en la base de datos *contact-lenses* y Naive Bayes es mejor en el set de datos *credit-g*. Aun así, el modelo obtenido para Random Forest es estadísticamente superior al resto de modelos.

Para la segunda parte de la práctica, trabajaremos con la base de datos *credit-g.arff* que tiene asociada una matriz de coste de manera que indique es que 5 veces más costoso otorgar un crédito a una persona que no lo devuelve. En primero lugar, aplicaremos el algoritmo 0R:

```
Test mode:    10-fold cross-validation
Evaluation cost matrix:
 0 1
 5 0


=== Classifier model (full training set) ===

ZeroR predicts class value: good

Time taken to build model: 0 seconds

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

Correctly Classified Instances         700               70      %
Incorrectly Classified Instances       300               30      %
Kappa statistic                          0
Total Cost                            1500
Average Cost                             1.5
Mean absolute error                      0.4202
Root mean squared error                  0.4583
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              1000

=== Confusion Matrix ===

   a   b   <-- classified as
 700   0 |   a = good
 300   0 |   b = bad
```

Obtenemos como resultado un 70% de precisión al clasificar y un coste total de 1500.
Si ahora utilizamos el algoritmo IBK, obtenemos:

```
Test mode:    10-fold cross-validation
Evaluation cost matrix:
 0 1
 5 0


=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

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

Correctly Classified Instances         671               67.1    %
Incorrectly Classified Instances       329               32.9    %
Kappa statistic                          0.2114
Total Cost                            1001
Average Cost                             1.001
Mean absolute error                      0.3294
Root mean squared error                  0.573
Relative absolute error                 78.3905 %
Root relative squared error            125.0279 %
Total Number of Instances              1000
```

Para k = 1, la tasa de aciertos no supera los resultados obtenidos mediante 0R pero, para k = 12, ya empezamos a obtener mejores resultados con un coste total de 1345:

```
Time taken to build model: 0 seconds

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

Correctly Classified Instances         703               70.3    %
Incorrectly Classified Instances       297               29.7    %
Kappa statistic                          0.0978
Total Cost                            1345
Average Cost                             1.345
Mean absolute error                      0.3776
Root mean squared error                  0.4507
Relative absolute error                 89.8682 %
Root relative squared error             98.3465 %
Total Number of Instances             1000

=== Confusion Matrix ===

   a   b   <-- classified as
 665  35 |   a = good
 262  38 |   b = bad
```

Para k = 25, se consigue una precisión del 71.5% y un coste total de 1349:

```
Test mode:    10-fold cross-validation
Evaluation cost matrix:
 0 1
 5 0


=== Classifier model (full training set) ===

IB1 instance-based classifier
using 25 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

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

Correctly Classified Instances         715               71.5    %
Incorrectly Classified Instances       285               28.5    %
Kappa statistic                          0.1127
Total Cost                            1349
Average Cost                             1.349
Mean absolute error                      0.3876
Root mean squared error                  0.4501
Relative absolute error                 92.2396 %
Root relative squared error             98.218  %
Total Number of Instances             1000

=== Confusion Matrix ===

   a   b   <-- classified as
 681  19 |   a = good
 266  34 |   b = bad
```

Si ahora empleamos como algoritmo Naive Bayes:

```
Time taken to build model: 0 seconds

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

Correctly Classified Instances         711               71.1    %
Incorrectly Classified Instances       289               28.9    %
Kappa statistic                          0.1743
Total Cost                            1217
Average Cost                             1.217
Mean absolute error                      0.3573
Root mean squared error                  0.4567
Relative absolute error                 85.0254 %
Root relative squared error             99.6696 %
Total Number of Instances             1000

=== Confusion Matrix ===

   a   b   <-- classified as
 643  57 |   a = good
 232  68 |   b = bad
```

Vemos que la precisión es similar a las obtenidas por IBK, pero el coste total es menor.
Podemos calcular los costes de la siguiente manera:

- **0R**: 300*5 = 1500
- **IBK (k = 12)**: 262*5 + 35 = 1345
- **IBK (k = 25)**: 266*5 + 19 = 1349
- **Naive Bayes**: 232*5 + 57 = 1217

Ahora vamos a emplear como clasificador el paquete costSensitiveClassifier para realizar un aprendizaje sensible al coste, dejando ZeroR como clasificador base y empleando la misma matriz de coste:

```
=== Classifier model (full training set) ===

CostSensitiveClassifier using reweighted training instances

weka.classifiers.rules.ZeroR

Classifier Model
ZeroR predicts class value: bad

Cost Matrix
 0 1
 5 0


Time taken to build model: 0 seconds

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

Correctly Classified Instances         300               30      %
Incorrectly Classified Instances       700               70      %
Kappa statistic                          0
Total Cost                             700
Average Cost                             0.7
Mean absolute error                      0.5726
Root mean squared error                  0.5962
Relative absolute error                136.2677 %
Root relative squared error            130.1057 %
Total Number of Instances             1000

=== Confusion Matrix ===

   a   b   <-- classified as
   0 700 |   a = good
   0 300 |   b = bad
```

Ahora vemos que la precisión sigue siendo la misma, pero el coste total se ha reducido a 700. Esto se debe a que antes el modelo basado en ZeroR asignaba todas las instancias a la clase *good*, que era donde teníamos mayor coste al fallar. Sin embargo, ahora clasifica todas las instancias como *bad*, permitiendo que, aunque crezca el número de errores a más del doble, el coste se reduzca a menos de la mitad.
Si usamos como base el algoritmo IBK con k = 12:

```
Cost Matrix
 0 1
 5 0


Time taken to build model: 0 seconds

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

Correctly Classified Instances         439               43.9    %
Incorrectly Classified Instances       561               56.1    %
Kappa statistic                          0.1038
Total Cost                             653
Average Cost                             0.653
Mean absolute error                      0.489
Root mean squared error                  0.5494
Relative absolute error                116.3763 %
Root relative squared error            119.8952 %
Total Number of Instances             1000

=== Confusion Matrix ===

   a   b   <-- classified as
 162 538 |   a = good
  23 277 |   b = bad
```

Para k = 25:

```
Cost Matrix
 0 1
 5 0


Time taken to build model: 0 seconds

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

Correctly Classified Instances         440               44      %
Incorrectly Classified Instances       560               56      %
Kappa statistic                          0.0909
Total Cost                             700
Average Cost                             0.7
Mean absolute error                      0.5197
Root mean squared error                  0.5589
Relative absolute error                123.6911 %
Root relative squared error            121.9609 %
Total Number of Instances             1000

=== Confusion Matrix ===

   a   b   <-- classified as
 175 525 |   a = good
  35 265 |   b = bad
```

Para Naive Bayes:

```
Cost Matrix
 0 1
 5 0


Time taken to build model: 0.01 seconds

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

Correctly Classified Instances         528               52.8    %
Incorrectly Classified Instances       472               47.2    %
Kappa statistic                          0.1631
Total Cost                             696
Average Cost                             0.696
Mean absolute error                      0.4847
Root mean squared error                  0.5289
Relative absolute error                115.3458 %
Root relative squared error            115.4105 %
Total Number of Instances             1000

=== Confusion Matrix ===

   a   b   <-- classified as
 284 416 |   a = good
  56 244 |   b = bad
```

Podemos observar que la precisión en todos estos modelos se ha reducido, pero también el coste total, ya que los clasificados incorrectamente como *good* ahora son mucho menores, pero también los clasificados correctamente como *good*, que se han reducido casi a la mitad.

Podemos resumir todo esto en la siguiente tabla:

Algoritmos	Precisión (sin basarse en coste)	Coste (sin basarse en coste)	Precisión (basándose en coste)	Coste (basándose en coste)
ZeroR				
IBK (k=12)				
IBK (k=25)				
Naive Bayes				

7. Práctica 6: Clasificación: Combinación de modelos y métodos.

En esta práctica, vamos a trabajar con los métodos de *bagging y boosting* sobre el set de datos *credit-g.arff.* Para ello, primero veremos los resultados que se obtienen con un modelo basado en el J48:

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

Correctly Classified Instances         705               70.5    %
Incorrectly Classified Instances       295               29.5    %
Kappa statistic                          0.2467
Mean absolute error                      0.3467
Root mean squared error                  0.4796
Relative absolute error                 82.5233 %
Root relative squared error            104.6565 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,840    0,610    0,763      0,840    0,799      0,251    0,639     0,746     good
                 0,390    0,160    0,511      0,390    0,442      0,251    0,639     0,449     bad
Weighted Avg.    0,705    0,475    0,687      0,705    0,692      0,251    0,639     0,657

=== Confusion Matrix ===

   a   b   <-- classified as
 588 112 |   a = good
 183 117 |   b = bad
```

Como podemos observar, se obtiene una precisión del 70.5%. Si ahora observamos los modelos de *bagging* sobre J48, tenemos:

- Para 10 repeticiones:

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

Correctly Classified Instances         733               73.3    %
Incorrectly Classified Instances       267               26.7    %
Kappa statistic                          0.3154
Mean absolute error                      0.3268
Root mean squared error                  0.4193
Relative absolute error                 77.7745 %
Root relative squared error             91.5015 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,863    0,570    0,779      0,863    0,819      0,321    0,753     0,858     good
                 0,430    0,137    0,573      0,430    0,491      0,321    0,753     0,586     bad
Weighted Avg.    0,733    0,440    0,718      0,733    0,721      0,321    0,753     0,777

=== Confusion Matrix ===

   a   b   <-- classified as
 604  96 |   a = good
 171 129 |   b = bad
```

- Para 20 repeticiones:

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

Correctly Classified Instances         744               74.4    %
Incorrectly Classified Instances       256               25.6    %
Kappa statistic                          0.3469
Mean absolute error                      0.3259
Root mean squared error                  0.4181
Relative absolute error                 77.5746 %
Root relative squared error             91.2324 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,867    0,543    0,788      0,867    0,826      0,353    0,755     0,859     good
                 0,457    0,133    0,596      0,457    0,517      0,353    0,755     0,586     bad
Weighted Avg.    0,744    0,420    0,731      0,744    0,733      0,353    0,755     0,777

=== Confusion Matrix ===

   a   b   <-- classified as
 607  93 |   a = good
 163 137 |   b = bad
```

- Para 40 repeticiones:

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

Correctly Classified Instances         746               74.6    %
Incorrectly Classified Instances       254               25.4    %
Kappa statistic                          0.344
Mean absolute error                      0.3238
Root mean squared error                  0.4197
Relative absolute error                 77.0721 %
Root relative squared error             91.5822 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,877    0,560    0,785      0,877    0,829      0,352    0,750     0,856     good
                 0,440    0,123    0,606      0,440    0,510      0,352    0,750     0,569     bad
Weighted Avg.    0,746    0,429    0,731      0,746    0,733      0,352    0,750     0,770

=== Confusion Matrix ===

   a   b   <-- classified as
 614  86 |   a = good
 168 132 |   b = bad
```

Si aplicamos boosting sobre J48, obtenemos:

- Para 10 repeticiones:

```
Time taken to build model: 0.05 seconds

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

Correctly Classified Instances         696               69.6    %
Incorrectly Classified Instances       304               30.4    %
Kappa statistic                          0.2678
Mean absolute error                      0.2925
Root mean squared error                  0.5066
Relative absolute error                 69.6069 %
Root relative squared error            110.5483 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,791    0,527    0,778      0,791    0,785      0,268    0,724     0,843     good
                 0,473    0,209    0,493      0,473    0,483      0,268    0,724     0,558     bad
Weighted Avg.    0,696    0,431    0,693      0,696    0,694      0,268    0,724     0,758

=== Confusion Matrix ===

   a   b   <-- classified as
 554 146 |   a = good
 158 142 |   b = bad
```

- Para 20 repeticiones:

```
Time taken to build model: 0.06 seconds

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

Correctly Classified Instances         728               72.8    %
Incorrectly Classified Instances       272               27.2    %
Kappa statistic                          0.3267
Mean absolute error                      0.2766
Root mean squared error                  0.4902
Relative absolute error                 65.8263 %
Root relative squared error            106.962  %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,834    0,520    0,789      0,834    0,811      0,328    0,728     0,844     good
                 0,480    0,166    0,554      0,480    0,514      0,328    0,728     0,522     bad
Weighted Avg.    0,728    0,414    0,719      0,728    0,722      0,328    0,728     0,747

=== Confusion Matrix ===

   a   b   <-- classified as
 584 116 |   a = good
 156 144 |   b = bad
```

- Para 40 repeticiones:

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

Correctly Classified Instances         722               72.2    %
Incorrectly Classified Instances       278               27.8    %
Kappa statistic                          0.3173
Mean absolute error                      0.2785
Root mean squared error                  0.4965
Relative absolute error                 66.2867 %
Root relative squared error            108.3297 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,824    0,517    0,788      0,824    0,806      0,318    0,727     0,850     good
                 0,483    0,176    0,541      0,483    0,511      0,318    0,727     0,509     bad
Weighted Avg.    0,722    0,414    0,714      0,722    0,717      0,318    0,727     0,748

=== Confusion Matrix ===

   a   b   <-- classified as
 577 123 |   a = good
 155 145 |   b = bad
```

Modelo J48 con 10 repeticiones:

- **Porcentaje de acierto:** 70.5%
- **Media F (F-Measure):** 69.2%
- **AUC:** 0.639

Modelo bagging sobre J48 con 10 repeticiones:

- **Porcentaje de acierto**: 73.3%
- **Media F (F-Measure):** 72.1%
- **AUC:** 0.753

Modelo bagging sobre J48 con 20 repeticiones:

- **Porcentaje de acierto**: 74.4%
- **Media F (F-Measure):** 73.3%
- **AUC:** 0.755

Modelo bagging sobre J48 con 40 repeticiones:

- **Porcentaje de acierto**: 74.6%
- **Media F (F-Measure):** 73.3%
- **AUC:** 0.750

Modelo boosting sobre J48 con 10 repeticiones:

- **Porcentaje de acierto**: 69.6%
- **Media F (F-Measure):** 69.4%
- **AUC:** 0.724

Modelo boosting sobre J48 con 20 repeticiones:

- **Porcentaje de acierto**: 72.8%
- **Media F (F-Measure):** 72.2%
- **AUC:** 0.728

Modelo boosting sobre J48 con 40 repeticiones:

- **Porcentaje de acierto**: 72.2%
- **Media F (F-Measure):** 71.7%
- **AUC:** 0.727

Con esto, podemos decir que quien mejores resultados obtiene para las distintas métricas es el modelo generado mediante bagging sobre J48 para 40 repeticiones, aunque el que se obtiene con 20 es muy similar, tanto por el equilibrio entre precisión y sensibilidad dado por el *F-Measure*, como por la tasa de aciertos como por el rendimiento del modelo que observamos la

medida de su AUC. Quienes peores resultados obtienen en general son los modelos del J48 y de *boosting* sobre J48 con 10 repeticiones.
A continuación, se realizará el mismo procedimiento para la base de datos *contact-lenses.* Con J48 tenemos:

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

Correctly Classified Instances          20               83.3333 %
Incorrectly Classified Instances         4               16.6667 %
Kappa statistic                          0.71
Mean absolute error                      0.15
Root mean squared error                  0.3249
Relative absolute error                 39.7059 %
Root relative squared error             74.3898 %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,053    0,833      1,000    0,909      0,889    0,947     0,833     soft
                 0,750    0,100    0,600      0,750    0,667      0,596    0,813     0,592     hard
                 0,800    0,111    0,923      0,800    0,857      0,669    0,811     0,865     none
Weighted Avg.    0,833    0,097    0,851      0,833    0,836      0,703    0,840     0,813

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  0  3  1 |  b = hard
  1  2 12 |  c = none
```

Con *bagging* sobre J48:

- Con 10 repeticiones:

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

Correctly Classified Instances          18               75      %
Incorrectly Classified Instances         6               25      %
Kappa statistic                          0.5355
Mean absolute error                      0.2017
Root mean squared error                  0.3362
Relative absolute error                 53.3889 %
Root relative squared error             76.9675 %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,800    0,053    0,800      0,800    0,800      0,747    0,947     0,710     soft
                 0,500    0,100    0,500      0,500    0,500      0,400    0,869     0,533     hard
                 0,800    0,333    0,800      0,800    0,800      0,467    0,852     0,936     none
Weighted Avg.    0,750    0,236    0,750      0,750    0,750      0,514    0,875     0,822

=== Confusion Matrix ===

  a  b  c   <-- classified as
  4  0  1 |  a = soft
  0  2  2 |  b = hard
  1  2 12 |  c = none
```

- Con 20 repeticiones:

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

Correctly Classified Instances          19               79.1667 %
Incorrectly Classified Instances         5               20.8333 %
Kappa statistic                          0.625
Mean absolute error                      0.2025
Root mean squared error                  0.3266
Relative absolute error                 53.155  %
Root relative squared error             74.0016 %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,053    0,833      1,000    0,909      0,889    0,947     0,710     soft
                 0,500    0,100    0,500      0,500    0,500      0,400    0,913     0,692     hard
                 0,800    0,222    0,857      0,800    0,828      0,567    0,852     0,938     none
Weighted Avg.    0,792    0,167    0,793      0,792    0,790      0,606    0,882     0,850

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  0  2  2 |  b = hard
  1  2 12 |  c = none
```

Con *boosting* sobre J48:

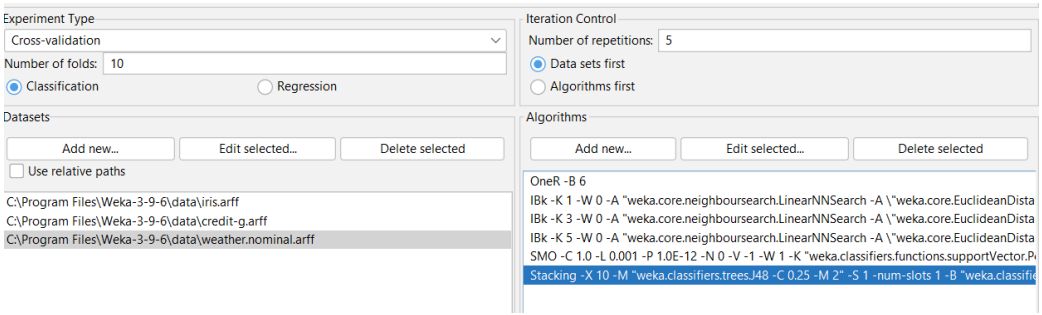
- Con 10 repeticiones:

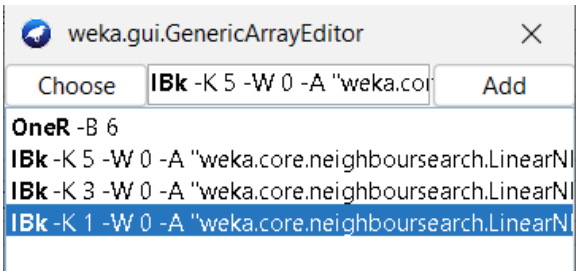
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          17               70.8333 %
Incorrectly Classified Instances         7               29.1667 %
Kappa statistic                          0.4766
Mean absolute error                      0.1803
Root mean squared error                  0.3831
Relative absolute error                 47.7325 %
Root relative squared error             87.7266 %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,800    0,053    0,800      0,800    0,800      0,747    0,916     0,654     soft
                 0,500    0,150    0,400      0,500    0,444      0,321    0,850     0,702     hard
                 0,733    0,333    0,786      0,733    0,759      0,393    0,770     0,846     none
Weighted Avg.    0,708    0,244    0,724      0,708    0,715      0,455    0,814     0,782

=== Confusion Matrix ===

  a  b  c   <-- classified as
  4  0  1 |  a = soft
  0  2  2 |  b = hard
  1  3 11 |  c = none
```

- Con 20 repeticiones:

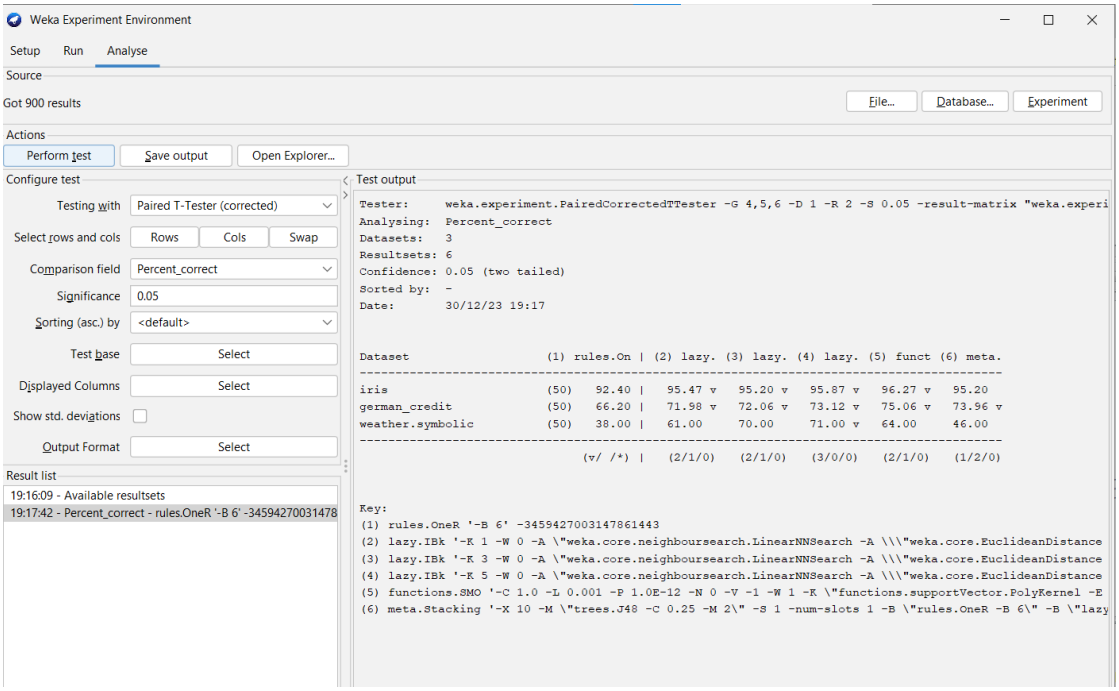
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          18               75      %
Incorrectly Classified Instances         6               25      %
Kappa statistic                          0.5017
Mean absolute error                      0.167
Root mean squared error                  0.3643
Relative absolute error                 43.8477 %
Root relative squared error             82.5251 %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,600    0,053    0,750      0,600    0,667      0,596    0,926     0,668     soft
                 0,500    0,050    0,667      0,500    0,571      0,507    0,938     0,813     hard
                 0,867    0,444    0,765      0,867    0,813      0,450    0,815     0,903     none
Weighted Avg.    0,750    0,297    0,745      0,750    0,742      0,490    0,858     0,839

=== Confusion Matrix ===

  a  b  c   <-- classified as
  3  0  2 |  a = soft
  0  2  2 |  b = hard
  1  1 13 |  c = none
```

En este caso, quien mejor tasa de aciertos y *F-Measure* tiene es el modelo generado por el algoritmo J48, con un 83.33% y un 83.66% respectivamente, seguido del modelo de *bagging* sobre J48 con 20 repeticiones, el cual obtiene un 79.17% y un 79.00% respectivamente, que es a su vez quien obtiene una mayor área sobre la curva ROC.
Por último, se han analizado en el modo *Experimenter* las bases de datos i*ris, weather y credit-g*, con los algoritmos *OneR, KNN* con *k=1, KNN* con *k=3, KNN* con *k=5, SMO* y *staking*. Para este último, se ha tomado como algoritmo de meta-aprendizaje el *J48* y se han tomado como clasificadores base los tres *KNN* anteriores y el *OneR* de la siguiente manera:





Con ello, se han obtenido los siguientes resultados:



- **iris**: La mayoría de los algoritmos tienen una alta precisión, pero con SMO se obtienen las mejores puntuaciones.
- **credit-g**: Nuevamente, *SMO* consigue los mejores resultados, seguido de *Stacking*, aunque en general no se obtienen mejores resultados que con el *dataset* anterior.
- **weather.symbolic**: Este *dataset* parece ser más desafiante. Los algoritmos *k-NN* parecen funcionar mejor con K=3 y K=5.
- **lazy.IBk** muestra un patrón donde el rendimiento generalmente mejora a medida que aumenta k (es decir, al considerar más vecinos cercanos).
- **Stacking** ha demostrado ser efectivo en la mayoría de los *datasets*, lo que sugiere que combinar diferentes clasificadores puede ser beneficioso, especialmente cuando se trata de *datasets* variados.

8. Práctica 7: Clustering (Agrupamiento).

En esta sección estudiaremos otra de las grandes tareas dentro de la minería de datos. Utilizaremos la base de datos de *provicias.arff* con el fin de obtener tres *clusters* que agrupen dichas provincias. Para ello se utilizado el método *SimpleKMeans* y se han seleccionado los siguientes atributos:

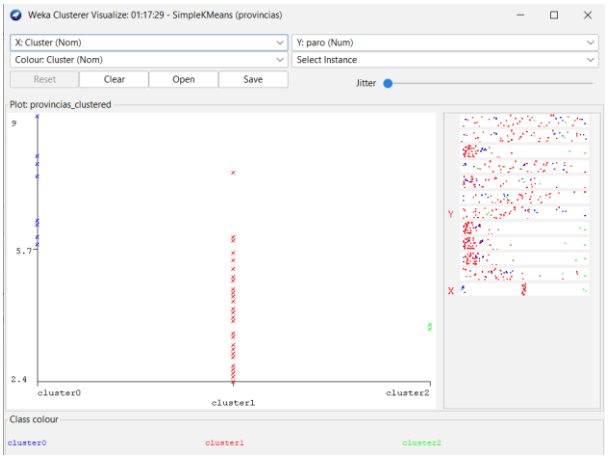
- *Población, extensión* y *paro*: Se han seleccionado estos atributos para estudiar las características socioeconómicas y demográficas de las distintas provincias.

```
kMeans
======

Number of iterations: 4
Within cluster sum of squared errors: 4.074124511502811

Initial starting points (random):

Cluster 0: 339555,7273,6
Cluster 1: 198045,10561,4.9
Cluster 2: 5964143,8022,3.7

Missing values globally replaced with mean/mode

Final cluster centroids:
                            Cluster#
Attribute      Full Data            0            1            2
                  (52.0)        (8.0)       (42.0)        (2.0)
============================================================================
población     848240.9615    711804.25   648180.9286    5595248.5
extension       9727.1923      4329.75   10843.4762         7875
paro               4.5346          7.1       4.0833         3.75
```

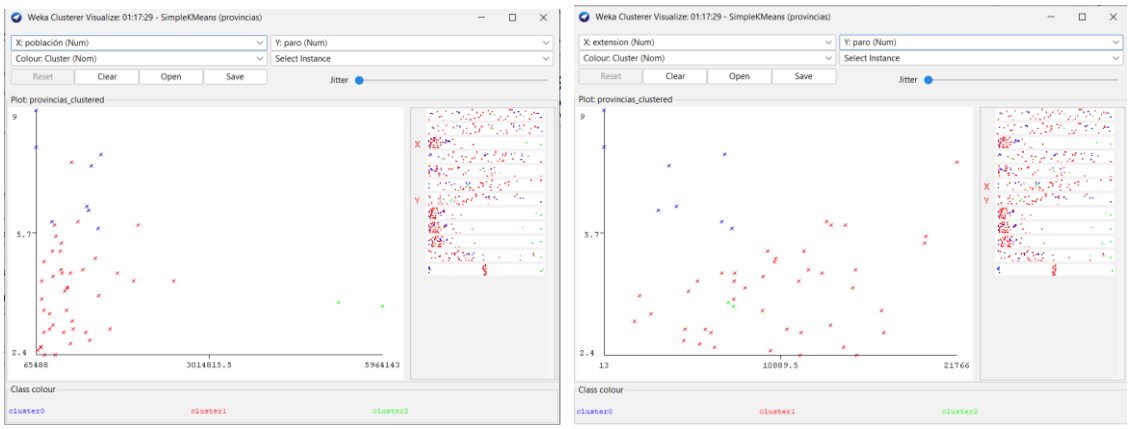
- El cluster 0 tiene una población media de aproximadamente 711,804 habitantes, una extensión de 4,329.75 unidades y una tasa de desempleo de 7.1. Se puede interpretar como áreas con menor población y mayor tasa de desempleo en comparación con los otros clusters. Está compuesto por provincias como Melilla, Coruña, Tenerife, La Palma o Ourense.

- El cluster 1 es el más grande con 42 instancias, lo que indica que abarca la mayoría de las provincias. Tiene una población promedio de 648,180 habitantes, una extensión de 10,843.48 unidades y una tasa de desempleo de 4.08. Esto sugiere áreas más pobladas con una extensión promedio y una tasa de desempleo moderada. Está compuesto por provincias como Badajoz, Álava, León, Huelva o Castellón.
- El cluster 2 es el cluster más pequeño con solo 2 instancias: Madrid y Barcelona. Sin embargo, tiene la mayor población promedio y una tasa de desempleo más baja, lo que sabemos al ser las grandes ciudades de España.

Si representamos los atributos seleccionados en función del cluster asignado:



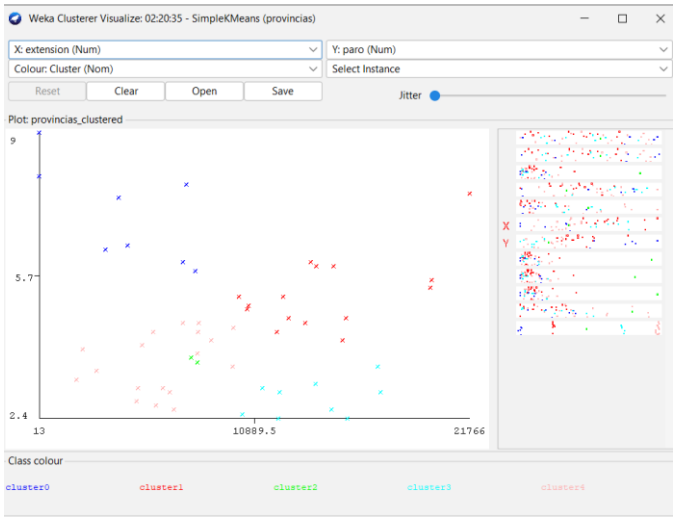
Donde vemos que el atributo que mejor se adapta a los clusters es el paro, aunque Badajoz es la ciudad que parece que se distingue más dentro del cluster1. Si vemos las relaciones entre los atributos, se distinguen mejor los diferentes clusters generados por el modelo:



Si probamos a modificar el número de *clusters* a 5:

```
Initial starting points (random):

Cluster 0: 339555,7273,6
Cluster 1: 198045,10561,4.9
Cluster 2: 5964143,8022,3.7
Cluster 3: 704907,6303,3.1
Cluster 4: 299957,2932,3.5

Missing values globally replaced with mean/mode

Final cluster centroids:
                        Cluster#
Attribute      Full Data          0          1          2          3          4
                  (52.0)      (8.0)     (15.0)      (2.0)      (9.0)     (18.0)
=============================================================================================
población     848240.9615   711804.25   780647.8  5595248.5  429973.6667   646895.5
extension      9727.1923     4329.75  14459.6667       7875  13863.5556  6319.9444
paro              4.5346         7.1      5.2867       3.75      2.8667     3.6889



Time taken to build model (full training data) : 0 seconds

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

Clustered Instances

0       8 ( 15%)
1      15 ( 29%)
2       2 (  4%)
3       9 ( 17%)
4      18 ( 35%)
```

Vemos que se generan 2 *clusters* más grandes, con 18 y 15 instancias, mantenemos el *cluster* formado por Madrid y Barcelona y se generan otros dos más pequeños con 8 y 9 instancias.



A continuación, trabajaremos con la base de datos *vote.arff,* la cual contiene preferencias sobre los votantes estadounidenses, con el fin de obtener dos *clusters* cuyo porcentaje de error sea inferior al 14%. Para ello se han probado distintas semillas, ya que para el valor por defecto 10 se superaba el porcentaje de error:

- Si la semilla vale 3:

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

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

Clustered Instances

0      232 ( 53%)
1      203 ( 47%)


Class attribute: Class
Classes to Clusters:

   0   1  <-- assigned to cluster
 220  47 | democrat
  12 156 | republican

Cluster 0 <-- democrat
Cluster 1 <-- republican

Incorrectly clustered instances :       59.0     13.5632 %
```

- Si la semilla vale 20:

```
Time taken to build model (full training data) : 0 seconds

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

Clustered Instances

0      218 ( 50%)
1      217 ( 50%)


Class attribute: Class
Classes to Clusters:

   0   1  <-- assigned to cluster
  54 213 | democrat
 164   4 | republican

Cluster 0 <-- republican
Cluster 1 <-- democrat

Incorrectly clustered instances :       58.0     13.3333 %
```

- Si la semilla vale 26:

```
Time taken to build model (full training data) : 0 seconds

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

Clustered Instances

0      209 ( 48%)
1      226 ( 52%)


Class attribute: Class
Classes to Clusters:

   0   1  <-- assigned to cluster
  47 220 | democrat
 162   6 | republican

Cluster 0 <-- republican
Cluster 1 <-- democrat

Incorrectly clustered instances :       53.0     12.1839 %
```

Este valor es el que parece conseguir mejores resultados. Podemos caracterizar a los Republicanos y a los Demócratas en función de las características más distintivas de estos *clusters*:

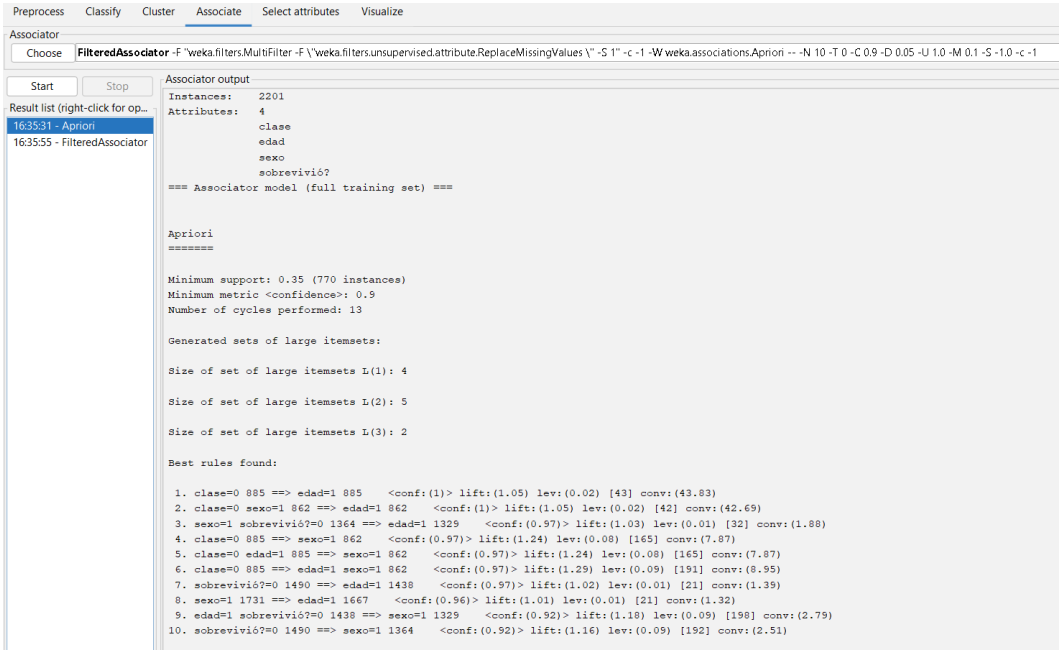
```
Final cluster centroids:
                                                Cluster#
Attribute                            Full Data          0          1
                                       (435.0)    (209.0)    (226.0)
=============================================================================
handicapped-infants                          n          n          y
water-project-cost-sharing                   y          y          y
adoption-of-the-budget-resolution            y          n          y
physician-fee-freeze                         n          y          n
el-salvador-aid                              y          y          n
religious-groups-in-schools                  y          y          n
anti-satellite-test-ban                      y          n          y
aid-to-nicaraguan-contras                    y          n          y
mx-missile                                   y          n          y
immigration                                  y          y          n
synfuels-corporation-cutback                 n          n          n
education-spending                           n          y          n
superfund-right-to-sue                       y          y          n
crime                                        y          y          n
duty-free-exports                            n          n          y
export-administration-act-south-africa       y          y          y
```

Donde vemos que los republicanos, representados por cluster0, están a favor de la congelación de las tarifas médicas, de ayudar al salvador, de grupos religiosos en las escuelas, de la inmigración, del gasto en educación... Mientras que el *cluster* de los demócratas se caracteriza por el apoyo a bebés discapacitados, la adopción de la resolución presupuestaria, la prohibición de pruebas anti-satélites, la ayuda a los contras nicaragüenses y las exportaciones libres de impuestos.

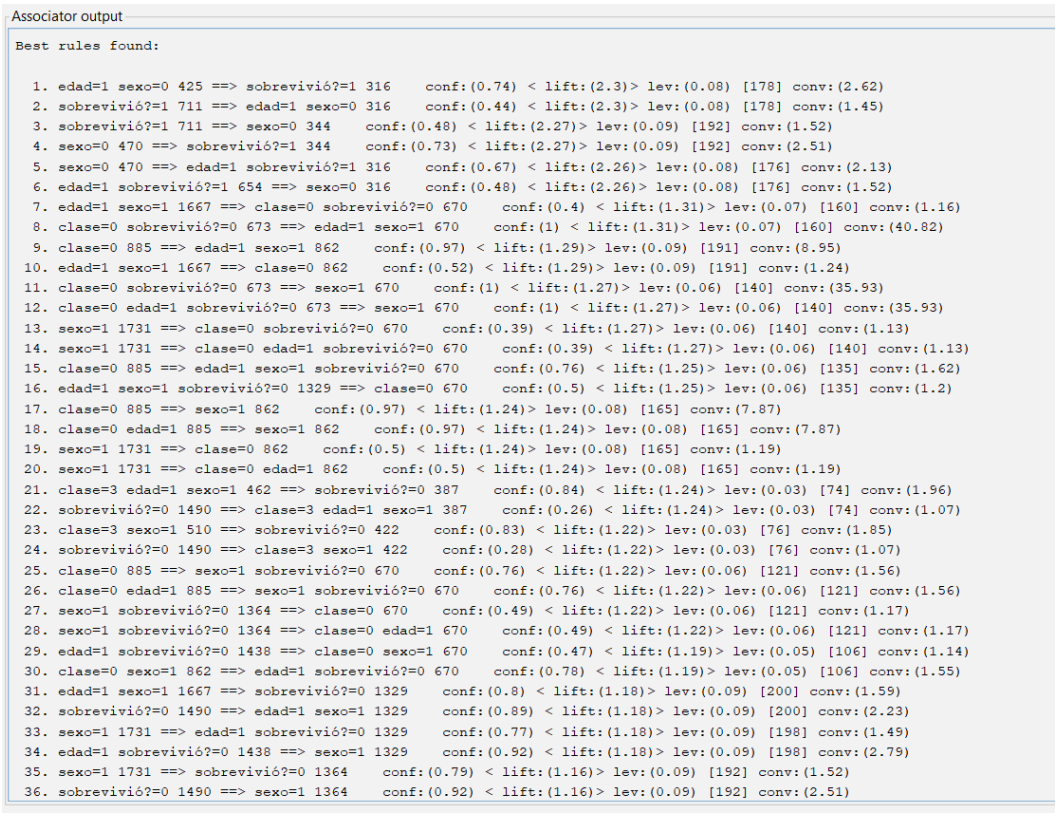
9. Práctica 8: Reglas de asociación.

En esta última práctica, trabajaremos la última de las grandes tareas de la minería de datos: Las reglas de asociación. Para ello, utilizaremos la base de datos *titanic.arff* para responder a las preguntas planteadas y como algoritmo el *a priori.* Este fichero consta de los siguientes atributos:

- Clase (0 = tripulación, 1 = primera, 2 = segunda, 3 = tercera).
- Edad (1 = adulto, 0 = niño).
- Sexo (1 = hombre, 0 = mujer).
- Sobrevivió (1 = sí, 0 = no).



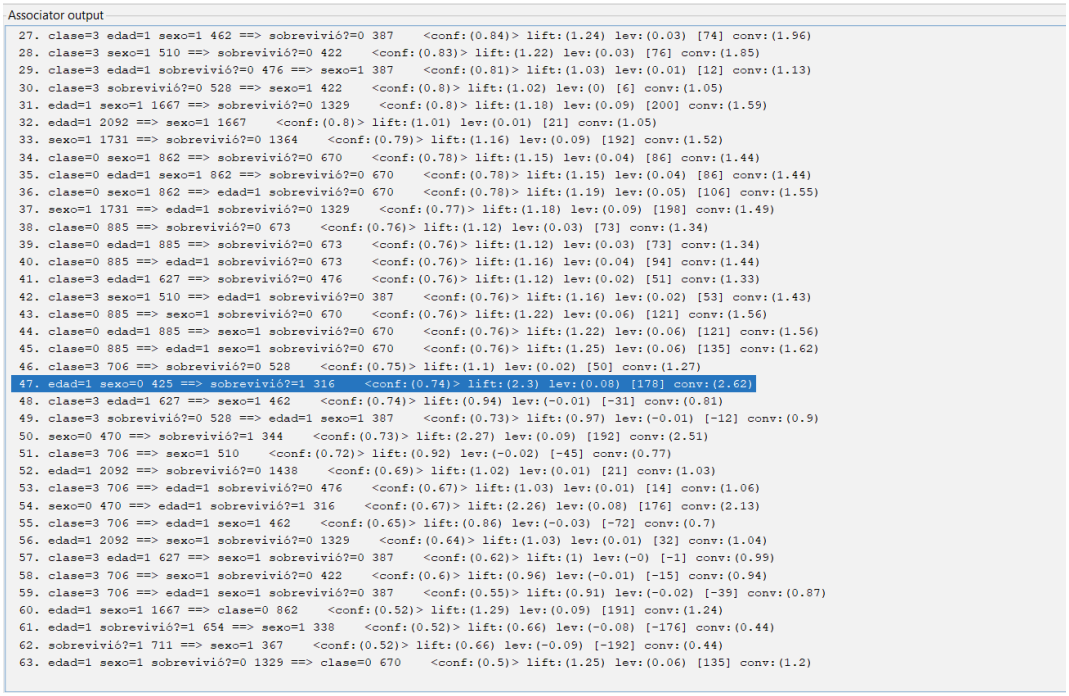
Para la primera regla, vemos que el factor de interés es 1.05, que al ser mayor que 1 indica que hay una asociación positiva entre clase=0 y edad=1, más fuerte de lo que se esperaría por casualidad. Esta regla nos indica que toda la tripulación es adulta.
Si aplicamos ahora el algoritmo para un máximo de 100 reglas y utilizando el factor de interés como métrica:



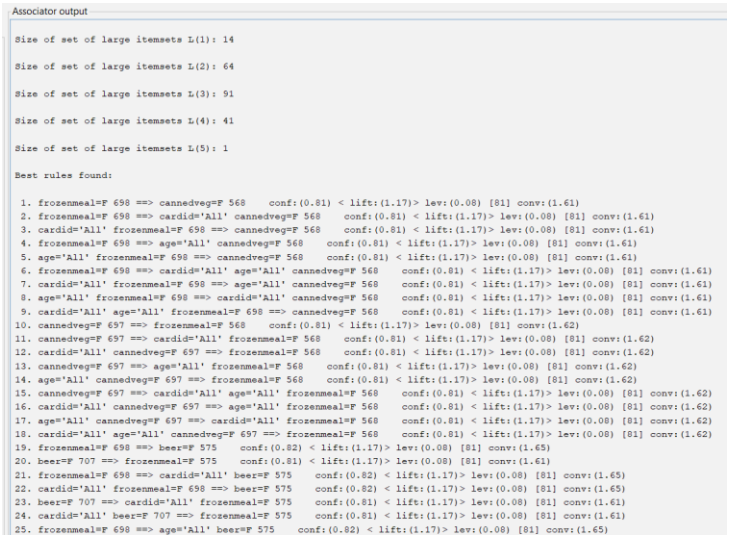
Se generan un total de 52 reglas, donde las más interesantes son las primeras. La primera y la segunda regla encontrada tienen ambas el mismo valor de interés 2.3:

- 1. edad=1 sexo=0 425 ==> sobrevivió?=1 316 conf:(0.74): Indica que, de las 425 mujeres adultas, sobrevivieron 316, es decir, el 74% que nos indica el factor de confianza. El *lift* nos dice que las mujeres adultas tenían 2.3 veces más probabilidades de sobrevivir comparado con lo que se esperaría al azar.
- 2. sobrevivió?=1 711 ==> edad=1 sexo=0 316 conf:(0.44): Indica que, de los 711 supervivientes, 316 eran mujeres adultas (44%). Esta regla es similar a la anterior y nos indica que de los supervivientes tenían 2.3 veces más probabilidades de ser mujeres adultas que al azar.

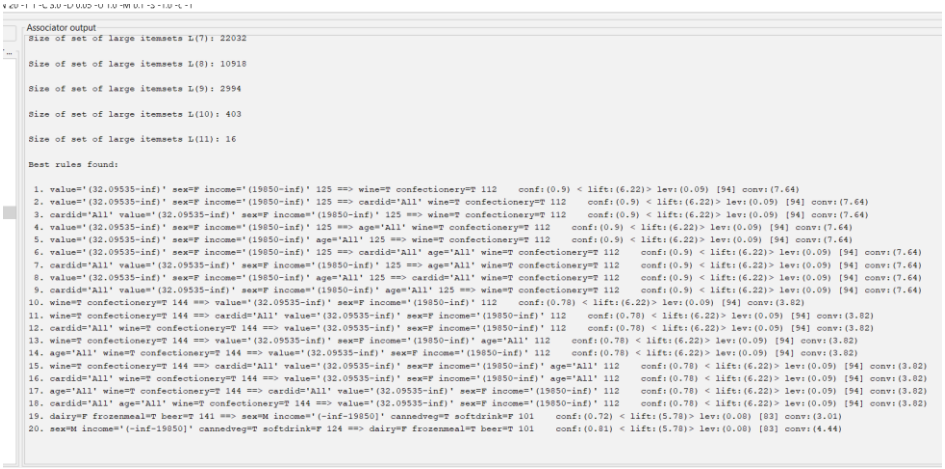
Si ahora volvemos a probar el algoritmo con la métrica inicial de *Confidence*, con un máximo de 100 reglas y una métrica mínima de 0.5, vemos que la primera regla la encontramos en la posición 47:



En la segunda parte, trabajaremos con la base de datos *Cesta_compra.arff*. Para poder aplicar el algoritmo *A priori,* es necesario discretizar el atributo *cardid*. Tras esto, se ha aplicado el algoritmo con la métrica Lift y un máximo de 50 reglas:



Se ha probado también a generar un máximo de 20 reglas con Lift mayor:



Entre las que podemos destacar que las mujeres mayores de 32 con una renta superior a 19850 suelen comprar vino y confitería, o que los hombres con rentas inferiores a 19850 que compran verduras enlatadas, pero no compran refrescos, tampoco compran diarios, pero sí compran comida congelada y cerveza.

10. Conclusiones.

En estas prácticas, nos hemos familiarizado con el entorno de Weka en una primera aproximación a la minería de datos. Hemos trabajado las distintas tareas más genéricas que nos encontramos en esta rama del conocimiento y hemos abordado diversos problemas mediante diferentes enfoques y técnicas. También hemos mejorado nuestra capacidad de análisis para escoger los algoritmos que mejor resolverían cierto problema en función del *dataset* y nos hemos familiarizado con distintos conceptos que son esenciales en este ámbito, como trabajar con clases desbalanceadas, el meta-aprendizaje o las diferentes métricas con las que evaluar los modelos.