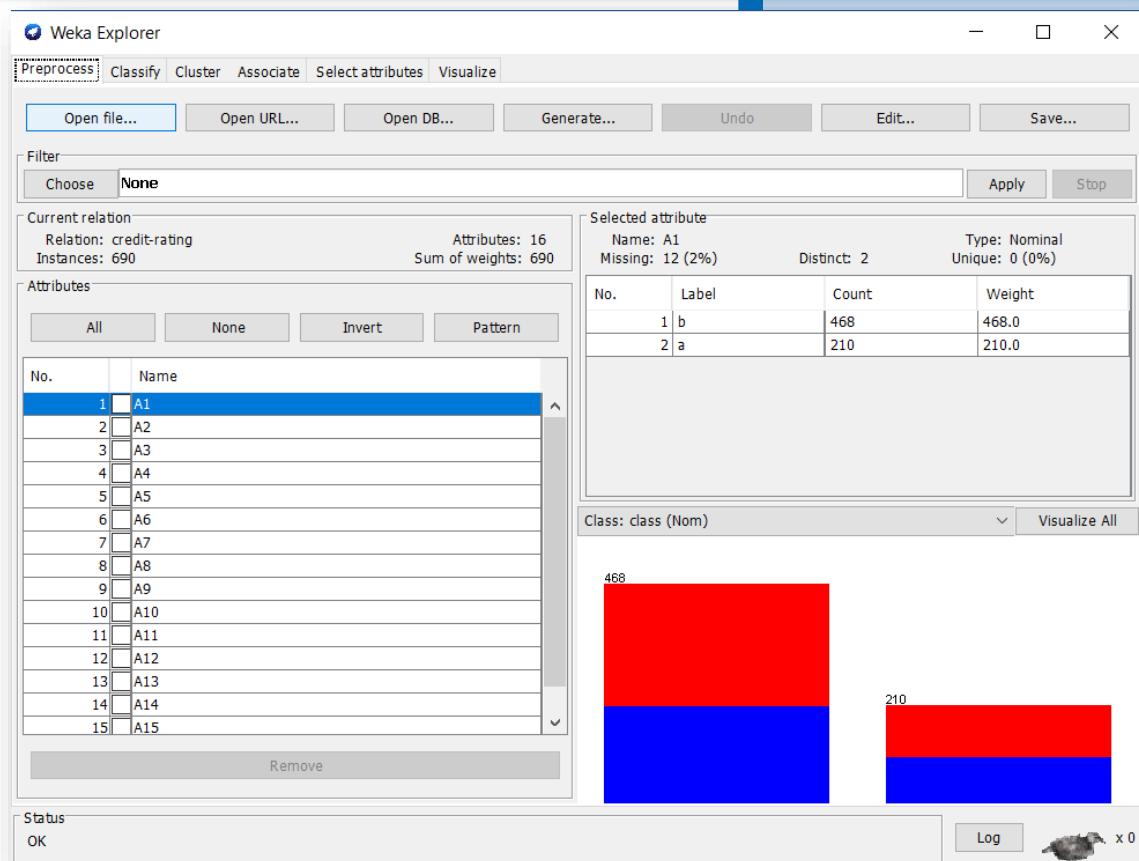




Análisis de datos con Weka



PAULA POLEY CEBALLOS
SISTEMAS INTELIGENTES
2021-2022

ÍNDICE

EXPLORER	3
PREPROCESAMIENTO DE DATOS (PREPROCESS)	3
REMOVE.....	12
DISCRETIZE.....	13
NOMINALTOBINARY	15
CLASSCONDITIONALPROBABILITIES	16
ATTRIBUTE SELECTION	17
PARTITIONMEMBERSHIP	18
ADDEXPRESSION.....	19
CLASIFICADORES (CLASSIFY)	21
J48.....	22
TABLA RESUMEN MODIFICACIONES (J48)	33
NAIVE BAYES.....	33
TABLA RESUMEN MODIFICACIONES (NAIVE BAYES)	36
NAIVE BAYES MULTINOMINAL TEXT	37
TABLA RESUMEN MODIFICACIONES (NAIVE BAYES MULTINOMINAL TEXT)	39
REPTree (DECISION TREE).....	39
TABLA RESUMEN MODIFICACIONES (REPTree)	45
LMT (LOGISTIC MODEL TREES)	46
IBk (K-VECINOS MÁS CERCANOS)	49
TABLA RESUMEN MODIFICACIONES (IBk)	55
SMO (MÁQUINAS DE VECTORES DE SOPORTE).....	56
TABLA RESUMEN MODIFICACIONES (SMO)	60
Logistic (REGRESIÓN LOGÍSTICA)	60
TABLA RESUMEN MODIFICACIONES (Logistic)	64
MLP (MULTILAYER PERCEPTRON)	65
TABLA RESUMEN MODIFICACIONES (MultilayerPerceptron)	68
BAGGING	68
TABLA RESUMEN MODIFICACIONES (Bagging)	72
CREAR UN NUEVO CLASIFICADOR	73
AGRUPACIÓN (CLUSTER)	74
SimpleKMeans	74
EM	79
HIERARCHICALCLUSTERER.....	82

ASOCIACIÓN (ASSOCIATE)	84
APRIORI.....	84
SELECCIÓN DE FUNCIONES (Select attributes)	86
KNOWLEDGEFLOW	90
SIMPLE CLI	97
TÉRMINOS EXPLICADOS	98

WEKA



El conjunto de datos utilizado en este trabajo tiene el nombre: grupo_22.arff proporcionado por el profesor de la asignatura.

Al final del documento se explicarán algunos términos que se ha mencionado a lo largo del trabajo.

Para empezar abrimos weka y lo primero que nos sale es la siguiente imagen.

EXPLORER

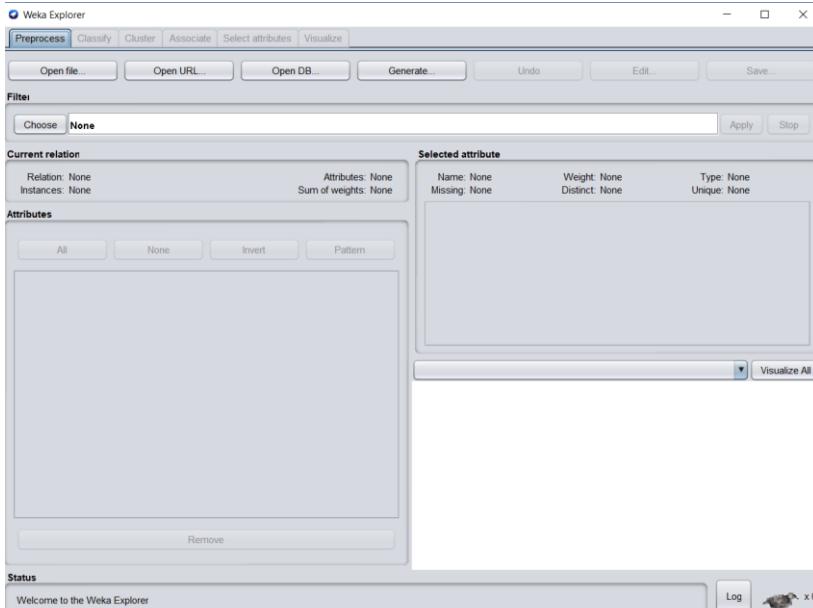
Le damos en Applications a Explorer es el modo más usado y descriptivo. Éste permite realizar operaciones sobre un sólo archivo de datos. Directamente nos sale la ventana de preprocesamiento. Este es el primer paso en el aprendizaje automático. Se seleccionará el archivo de datos, lo procesará y lo ajustará para aplicar los distintos algoritmos de aprendizaje automático. En Visualize permite visualizar sus datos procesados para su análisis.

El explorador permite tareas de:

- a) Preprocesado de los datos y aplicación de filtros.
- b) Clasificación.
- c) Clustering.
- d) Búsqueda de Asociaciones.
- e) Selección de atributos.
- f) Visualización de datos

Como veremos y comentaremos a continuación.

PREPROCESAMIENTO DE DATOS (PREPROCESS)



Preprocess

Para cargar los datos y definir el origen de estos en la ventana **Preprocess** le damos a Open file y aparece una ventana de selección de fichero. Seleccionamos el conjunto de datos que vamos a (queremos) estudiar. Este paso se realiza si el archivo lo tenemos en nuestro ordenador. Se hace el procesamiento porque los datos pueden tener muchas cosas no deseadas que dan lugar a un incorrecto análisis.

Una vez seleccionado el origen de los datos podremos aplicar algún filtro sobre él o bien pasar a las siguientes secciones y realizar otras tareas. Los botones que acompañan a abrir el fichero:

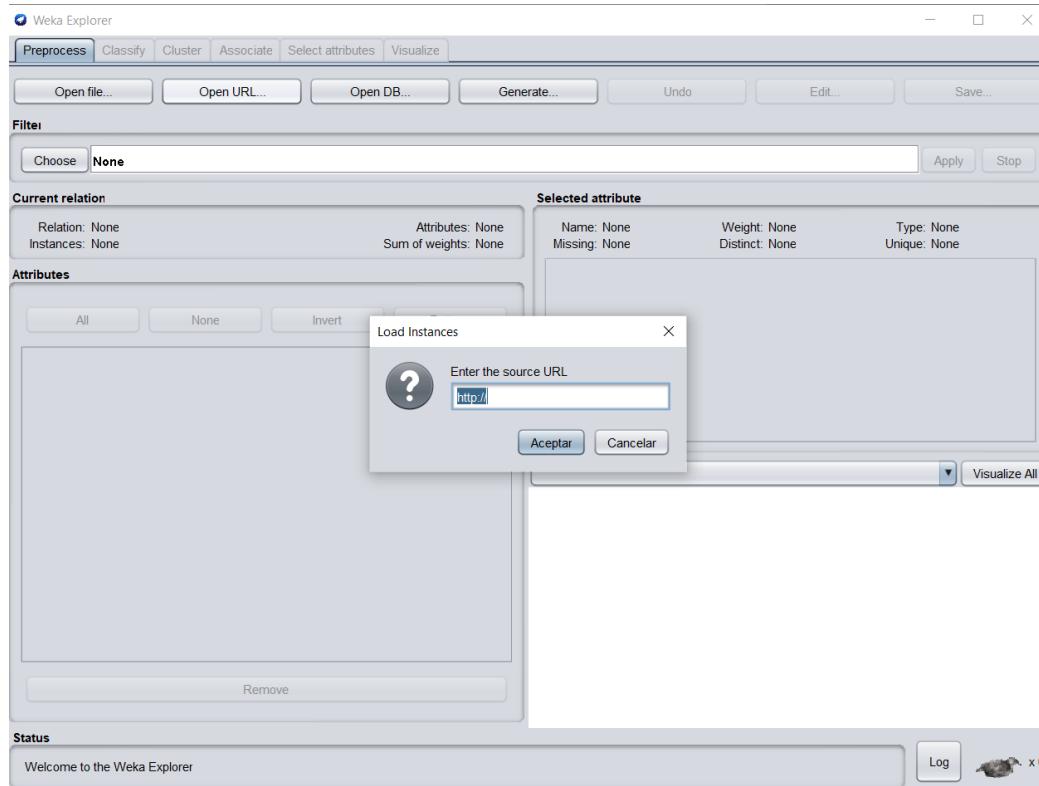
Undo

Save...

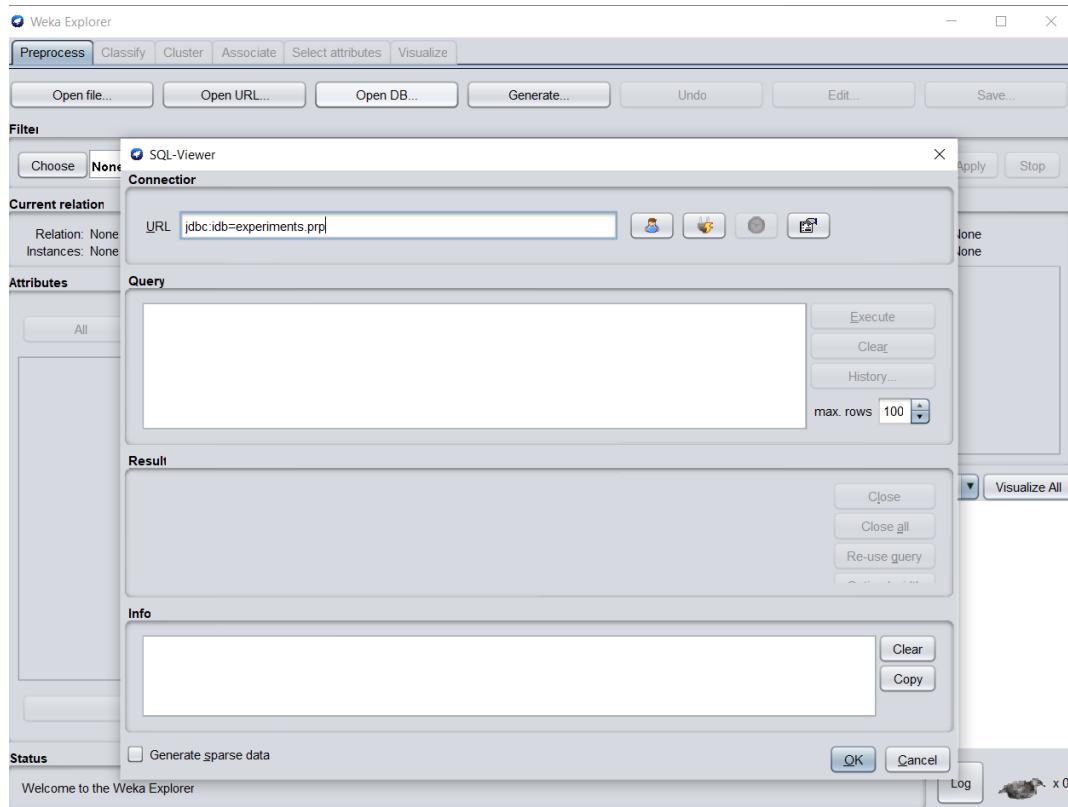
y , nos permiten deshacer los cambios y guardar los nuevos datos ya transformados (en formato arff). Además, se muestra en la ventana (captura de abajo) cada uno de los atributos que componen los datos, junto con un resumen con estadísticas de estos (media aritmética, rango de los datos, desviación estándar, número de instancias distintas, de qué tipo son, etc.)

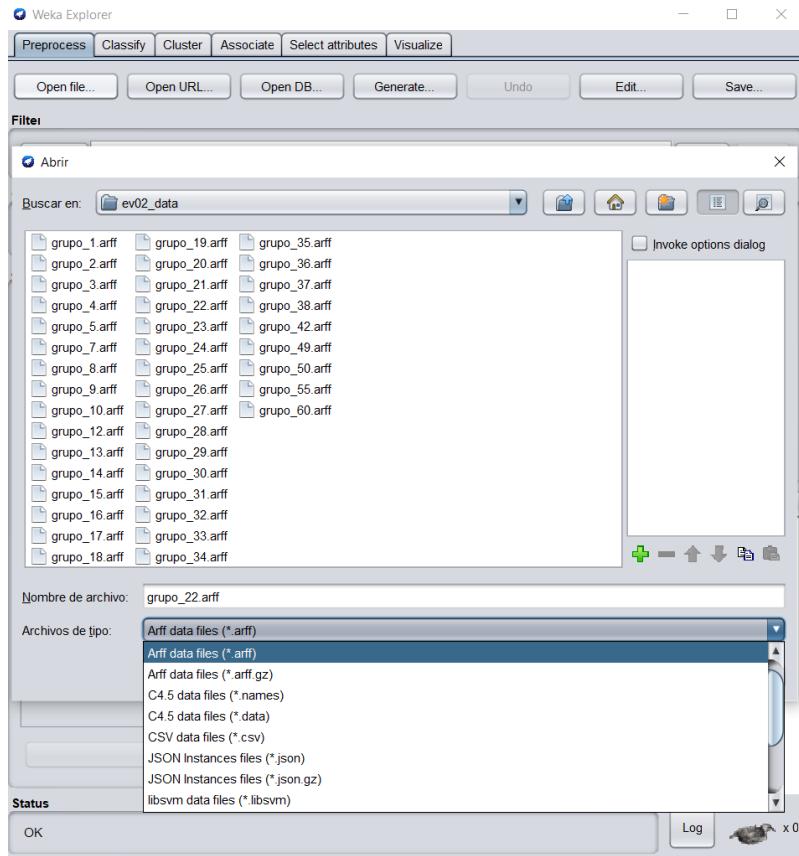
The screenshot shows the Weka Explorer interface with the 'Preprocess' tab active. The main workspace displays the 'credit-rating' dataset. The 'Current relation' section shows 690 instances and 16 attributes. The 'Selected attribute' section shows the 'A1' attribute, which is nominal with 2 distinct values ('b' and 'a') and 0 unique values. A histogram below shows the count of each class: 468 for 'b' and 210 for 'a'. The bottom part of the interface shows the 'Open file...' dialog box, which is a standard file selection dialog with a list of files in the 'Escritorio' folder.

En cambio si lo que queremos cargar datos desde la web en vez de Open file le damos a Open URL e introducimos la url con los datos a estudiar.



Y por último si lo que queremos es cargar los datos desde la base de datos le damos a Open BD.





Weka admite muchos formatos de archivo para los datos. Estos son arff, arff.gz, bsi, csv, dat, data, json, json.gz, libsvm, m, names, xrff, xrff.gz. Se puede ver en la captura anterior.

El tipo de archivo que se va a utilizar es el formato arff acrónimo de Attribute-Relation File Format. Este formato está compuesto por una estructura diferenciada en tres partes:

1. **Cabecera.** Se define el nombre de la relación. Su formato es el siguiente:

```
@relation <nombre-de-la-relacion>
```

Donde <nombre-de-la-relacion> es de tipo String (Entendiendo como tipo String el ofrecido por Java.) . Si dicho nombre contiene algún espacio será necesario expresarlo entrecomillado.

2. **Declaraciones de atributos.** En esta sección se declaran los atributos que compondrán nuestro archivo junto a su tipo. La sintaxis es la siguiente:

```
@attribute <nombre-del-atributo> <tipo>
```

Donde <nombre-del-atributo> es de tipo String teniendo las mismas restricciones que el caso anterior. Weka acepta diversos tipos, estos son:

- a) **NUMERIC:** Expresa números reales
- b) **INTEGER:** Expresa números enteros.
- c) **DATE:** Expresa fechas, para ello este tipo debe ir precedido de una etiqueta de formato entrecomillada. La etiqueta de formato está compuesta por caracteres separadores (guiones y/o espacios) y unidades de tiempo:

- dd Día.
- MM Mes.
- yyyy Año.
- HH Horas.
- mm Minutos.
- ss Segundos.

- d) **STRING:** Expresa cadenas de texto, con las restricciones del tipo String comentadas anteriormente.

- e) ENUMERADO El identificador de este tipo consiste en expresar entre llaves y separados por comas los posibles valores (caracteres o cadenas de caracteres) que puede tomar el atributo.
3. Sección de datos: Declaramos los datos que componen la relación separando entre comas los atributos y con saltos de línea las relaciones.

@data

4.3.2

Aunque éste es el modo "completo" es posible definir los datos de una forma abreviada (sparse data). Si tenemos una muestra en la que hay muchos datos que sean 0 podemos expresar los datos prescindiendo de los elementos que son nulos, rodeando cada una de las filas entre llaves y situando delante de cada uno de los datos el número de atributo .

Un ejemplo de esto es el siguiente

@data

{1 4, 3 3}

En este caso hemos prescindido de los atributos 0 y 2 (como mínimo) y asignamos al atributo 1 el valor 4 y al atributo 3 el valor 3.

En el caso de que algún dato sea desconocido se expresará con un símbolo de cerrar interrogación ("?"). Es posible añadir comentarios con el símbolo "%", que indicará que desde ese símbolo hasta el final de la línea es todo un comentario. Los comentarios pueden situarse en cualquier lugar del fichero

Cuando ya hemos cargado los datos, en la pantalla principal en unos de los recuadros esta Current relation

Current relation	
Relation: credit-rating	Attributes: 16
Instances: 690	Sum of weights: 690

donde se ve el nombre del archivo de datos que se ha cargado. Hay 690 instancias es decir esto es el número de filas de la tabla (690 ejemplos). También esta contiene 16 atributos que van a ser los campos a tratar.

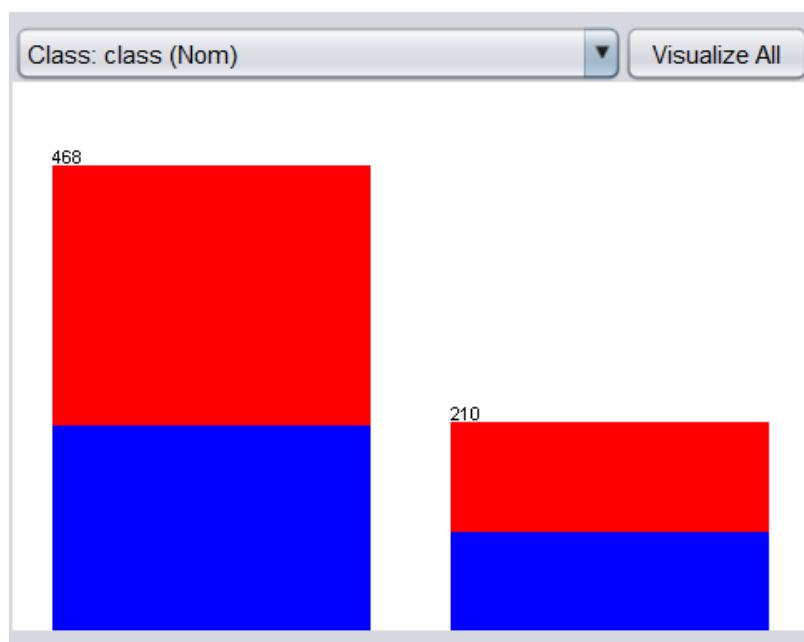
Justo debajo de este cuadro tenemos Attributes donde podemos ver los distintos campos del conjunto de datos. Esta contiene 16 campos: A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, class. Es decir, tenemos 16 atributos.

Attributes	
All	None
1	A1
2	A2
3	A3
4	A4
5	A5
6	A6
7	A7
8	A8
9	A9
10	A10
11	A11
12	A12
13	A13
14	A14
15	A15
16	class

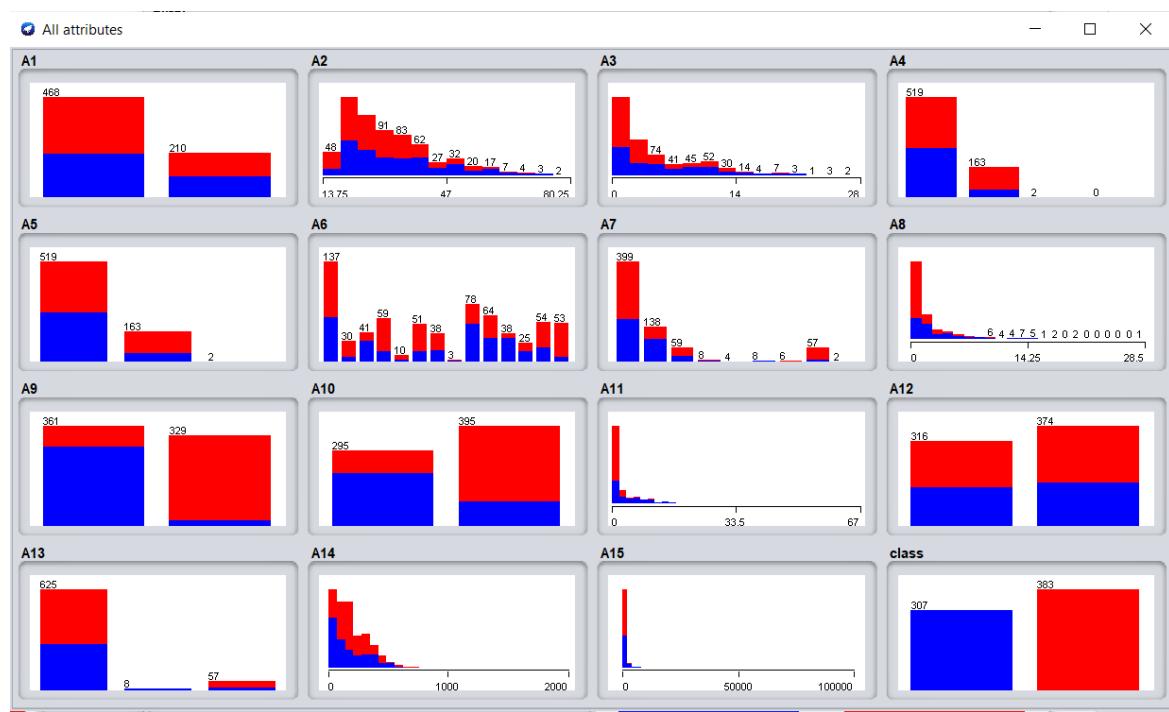
Como tengo seleccionado el atributo A1 en el lado derecho se muestra más detalles de este atributo. (Captura abajo). Pero puedo ir pinchando sobre el nombre de los otros atributos y puedo ver su distribución al igual que con este. En la sección Selected attribute podemos ver que este atributo tiene dos valores asociados a y b. Está el nombre y el tipo de atributo en este caso es Nominal . Hay 2 valores distintos sin un valor único (Distinct: 2) . Se muestra también los valores nominales para A1 como b,a con su recuento y peso.

Selected attribute			
Name: A1		Type: Nominal	
Missing: 12 (2%)		Distinct: 2	Unique: 0 (0%)
No.	Label	Count	Weight
1	b	468	468.0
2	a	210	210.0

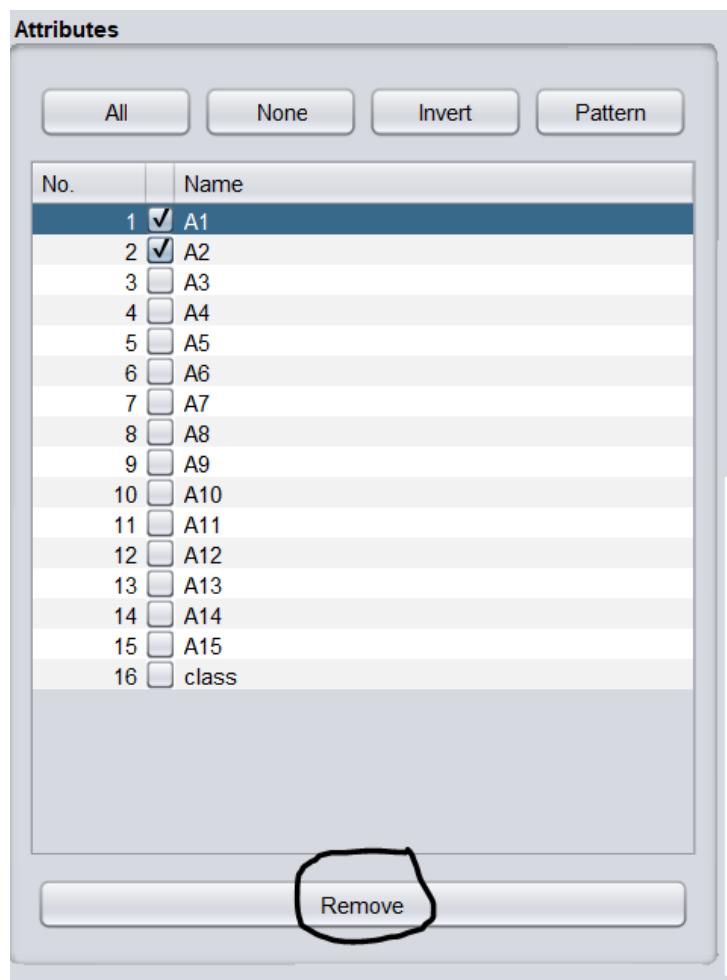
Por último en la parte inferior de la ventana vemos la representación visual de ese atributo. Los valores asociados b y a del atributo A1 que tienen 468 y 210 ejemplos respectivamente. Que estén pintados de colores hace que nos permita ver que proporción de estos ejemplos son positivos (de color azul) y cuales negativos (de color rojo).



Si le damos a Visualize All (en la captura anterior vemos dicho botón) se ven todas las funciones en una ventana.



Para eliminar atributos simplemente se selecciona el atributo que se desea eliminar de la lista de Attributes y se le da a remove. En este caso se eliminaría el atributo A1 y A2.



Como experimento en la ventana arriba a la derecha al pulsar Edit vamos a modificar el valor del atributo A1 en el primer ejemplo de "b" a "a".

1º Paso

The Weka Explorer interface shows the 'Edit...' button highlighted in yellow, indicating it is selected. The 'Selected attribute' panel displays 'A1' with values 'b' (468) and 'a' (210). The 'Attributes' panel lists attributes A1 through A15 and class.

2º Paso

The Weka Viewer interface displays the 'credit-rating' dataset. The first row, which previously had 'b' in column A1, now has 'a'. The 'Add instance' button is visible at the bottom.

No.	1: A1	2: A2	3: A3	4: A4	5: A5	6: A6	7: A7	8: A8	9: A9	10: A10	11: A11	12: A12	13: A13	14: A14	15: A15	16: class
1	a	30.83	0.0	u	g	w	v	1.25	t	1.0	f	g	202.0	0.0	+	
2	a	58.67	4.46	u	g	q	h	3.04	t	6.0	f	g	43.0	560.0	+	
3	a	24.5	0.5	u	g	q	h	1.5	t	0.0	f	g	280.0	824.0	+	
4	b	27.83	1.54	u	g	w	v	3.75	t	5.0	t	g	100.0	3.0	+	
5	b	20.17	5.625	u	g	w	v	1.71	t	0.0	f	s	120.0	0.0	+	
6	b	32.08	4.0	u	g	m	v	2.5	t	0.0	t	g	360.0	0.0	+	
7	b	33.17	1.04	u	g	r	h	6.5	t	0.0	t	g	164.0	3128...	+	
8	a	22.92	11.585	u	g	cc	v	0.04	t	0.0	f	g	80.0	1349.0	+	
9	b	54.42	0.5	y	p	k	h	3.96	t	0.0	f	g	180.0	314.0	+	
10	b	42.5	4.915	y	p	w	v	3.165	t	0.0	t	g	52.0	1442.0	+	
11	b	22.08	0.83	u	g	c	h	2.165	f	0.0	t	g	128.0	0.0	+	
12	b	29.92	1.835	u	g	c	h	4.335	t	0.0	f	g	260.0	200.0	+	
13	a	38.25	6.0	u	g	k	v	1.0	t	0.0	t	g	0.0	0.0	+	
14	b	48.08	6.04	u	g	k	v	0.04	f	0.0	f	g	0.0	2690.0	+	
15	a	45.83	10.5	u	g	q	v	5.0	t	7.0	t	g	0.0	0.0	+	
16	b	36.67	4.415	y	p	k	v	0.25	t	10.0	t	g	320.0	0.0	+	
17	b	28.25	0.875	u	g	m	v	0.96	t	3.0	t	g	396.0	0.0	+	
18	a	23.25	5.875	u	g	q	v	3.17	t	10.0	f	g	120.0	245.0	+	
19	b	21.83	0.25	u	g	d	h	0.665	t	0.0	t	g	0.0	0.0	+	
20	a	19.17	8.585	u	g	cc	h	0.75	t	7.0	f	g	96.0	0.0	+	
21	b	25.0	11.25	u	g	c	v	2.5	t	17.0	f	g	200.0	1208.0	+	
22	b	23.25	1.0	u	g	c	v	0.835	t	0.0	f	s	300.0	0.0	+	
23	a	47.75	8.0	u	g	c	v	7.875	t	6.0	t	g	0.0	1260.0	+	
24	a	27.42	14.5	u	g	x	h	3.085	t	1.0	f	g	120.0	11.0	+	
25	a	41.17	6.5	u	g	q	v	0.5	t	3.0	t	g	145.0	0.0	+	
26	a	15.83	0.585	u	g	c	h	1.5	t	2.0	f	g	100.0	0.0	+	
27	a	47.0	13.0	u	g	i	bb	5.165	t	9.0	t	g	0.0	0.0	+	
28	b	56.58	18.5	u	g	d	bb	15.0	t	17.0	t	g	0.0	0.0	+	
29	b	57.42	8.5	u	g	e	h	7.0	t	3.0	f	g	0.0	0.0	+	
30	b	42.09	1.04	u	g	w	v	5.0	t	6.0	t	g	500.0	1000.0	+	

3º Paso

No.	1: A1	2: A2	3: A3	4: A4	5: A5	6: A6	7: A7	8: A8	9: A9	10: A10	11: A11	12: A12	13: A13	14: A14	15: A15	16: class
	Nominal	Numeric	Numeric	Nominal												
1	b	30.83	0.0	u	g	w	v	1.25	t	t	1.0	f	g	202.0	0.0	+
2		58.67	4.46	u	g	q	h	3.04	t	t	6.0	f	g	43.0	560.0	+
3	a	24.5	0.5	u	g	q	h	1.5	t	f	0.0	f	g	280.0	824.0	+
4	b	27.83	1.54	u	g	w	v	3.75	t	t	5.0	t	g	100.0	3.0	+
5	a	20.17	5.625	u	g	w	v	1.71	t	f	0.0	f	s	120.0	0.0	+
6	b	32.08	4.0	u	g	m	v	2.5	t	f	0.0	t	g	360.0	0.0	+
7	b	33.17	1.04	u	g	r	h	6.5	t	f	0.0	t	g	164.0	3128...	+
8	a	22.92	11.585	u	g	cc	v	0.04	t	f	0.0	f	g	80.0	1349.0	+
9	b	54.42	0.5	y	p	k	h	3.96	t	f	0.0	f	g	180.0	314.0	+
10	b	42.5	4.915	y	p	w	v	3.165	t	f	0.0	t	g	52.0	1442.0	+
11	b	22.08	0.83	u	g	c	h	2.165	f	f	0.0	t	g	128.0	0.0	+
12	b	29.92	1.835	u	g	c	h	4.335	t	f	0.0	f	g	260.0	200.0	+
13	a	38.25	6.0	u	g	k	v	1.0	t	f	0.0	t	g	0.0	0.0	+
14	b	48.08	6.04	u	g	k	v	0.04	f	f	0.0	f	g	0.0	2690.0	+
15	a	45.83	10.5	u	g	q	v	5.0	t	t	7.0	t	g	0.0	0.0	+
16	b	36.67	4.415	y	p	k	v	0.25	t	t	10.0	t	g	320.0	0.0	+
17	b	28.25	0.875	u	g	m	v	0.96	t	t	3.0	t	g	396.0	0.0	+
18	a	23.25	5.875	u	g	q	v	3.17	t	t	10.0	f	g	120.0	245.0	+
19	b	21.83	0.25	u	g	d	h	0.665	t	f	0.0	t	g	0.0	0.0	+
20	a	19.17	8.585	u	g	cc	h	0.75	t	t	7.0	f	g	96.0	0.0	+
21	b	25.0	11.25	u	g	c	v	2.5	t	t	17.0	f	g	200.0	1208.0	+
22	b	23.25	1.0	u	g	c	v	0.835	t	f	0.0	f	s	300.0	0.0	+
23	a	47.75	8.0	u	g	c	v	7.875	t	t	6.0	t	g	0.0	1280.0	+
24	a	27.42	14.5	u	g	x	h	3.085	t	t	1.0	f	g	120.0	11.0	+
25	a	41.17	6.5	u	g	q	v	0.5	t	t	3.0	t	g	145.0	0.0	+
26	a	15.83	0.585	u	g	c	h	1.5	t	t	2.0	f	g	100.0	0.0	+
27	a	47.0	13.0	u	g	i	bb	5.165	t	t	9.0	t	g	0.0	0.0	+
28	b	56.58	18.5	u	g	d	bb	15.0	t	t	17.0	t	g	0.0	0.0	+
29	b	57.42	8.5	u	g	e	h	7.0	t	t	3.0	f	g	0.0	0.0	+
30	b	42.08	1.04	u	g	w	v	5.0	t	t	6.0	t	g	500.0	1000	+

Add instance Undo OK Cancel

4º Paso

No.	1: A1	2: A2	3: A3	4: A4	5: A5	6: A6	7: A7	8: A8	9: A9	10: A10	11: A11	12: A12	13: A13	14: A14	15: A15	16: class
	Nominal	Numeric	Numeric	Nominal												
1	a	30.83	0.0	u	g	w	v	1.25	t	t	1.0	f	g	202.0	0.0	+
2	a	58.67	4.46	u	g	q	h	3.04	t	t	6.0	f	g	43.0	560.0	+
3	a	24.5	0.5	u	g	q	h	1.5	t	f	0.0	f	g	280.0	824.0	+
4	b	27.83	1.54	u	g	w	v	3.75	t	t	5.0	t	g	100.0	3.0	+
5	b	20.17	5.625	u	g	w	v	1.71	t	f	0.0	f	s	120.0	0.0	+
6	b	32.08	4.0	u	g	m	v	2.5	t	f	0.0	t	g	360.0	0.0	+
7	b	33.17	1.04	u	g	r	h	6.5	t	f	0.0	t	g	164.0	3128...	+
8	a	22.92	11.585	u	g	cc	v	0.04	t	f	0.0	f	g	80.0	1349.0	+
9	b	54.42	0.5	y	p	k	h	3.96	t	f	0.0	f	g	180.0	314.0	+
10	b	42.5	4.915	y	p	w	v	3.165	t	f	0.0	t	g	52.0	1442.0	+
11	b	22.08	0.83	u	g	c	h	2.165	f	f	0.0	t	g	128.0	0.0	+
12	b	29.92	1.835	u	g	c	h	4.335	t	f	0.0	f	g	260.0	200.0	+
13	a	38.25	6.0	u	g	k	v	1.0	t	f	0.0	t	g	0.0	0.0	+
14	b	48.08	6.04	u	g	k	v	0.04	f	f	0.0	f	g	0.0	2690.0	+
15	a	45.83	10.5	u	g	q	v	5.0	t	t	7.0	t	g	0.0	0.0	+
16	b	36.67	4.415	y	p	k	v	0.25	t	t	10.0	t	g	320.0	0.0	+
17	b	28.25	0.875	u	g	m	v	0.96	t	t	3.0	t	g	396.0	0.0	+
18	a	23.25	5.875	u	g	q	v	3.17	t	t	10.0	f	g	120.0	245.0	+
19	b	21.83	0.25	u	g	d	h	0.665	t	f	0.0	t	g	0.0	0.0	+
20	a	19.17	8.585	u	g	cc	h	0.75	t	t	7.0	f	g	96.0	0.0	+
21	b	25.0	11.25	u	g	c	v	2.5	t	t	17.0	f	g	200.0	1208.0	+
22	b	23.25	1.0	u	g	c	v	0.835	t	f	0.0	f	s	300.0	0.0	+
23	a	47.75	8.0	u	g	c	v	7.875	t	t	6.0	t	g	0.0	1280.0	+
24	a	27.42	14.5	u	g	x	h	3.085	t	t	1.0	f	g	120.0	11.0	+
25	a	41.17	6.5	u	g	q	v	0.5	t	t	3.0	t	g	145.0	0.0	+
26	a	15.83	0.585	u	g	c	h	1.5	t	t	2.0	f	g	100.0	0.0	+
27	a	47.0	13.0	u	g	i	bb	5.165	t	t	9.0	t	g	0.0	0.0	+
28	b	56.58	18.5	u	g	d	bb	15.0	t	t	17.0	t	g	0.0	0.0	+
29	b	57.42	8.5	u	g	e	h	7.0	t	t	3.0	f	g	0.0	0.0	+
30	b	42.08	1.04	u	g	w	v	5.0	t	t	6.0	t	g	500.0	1000	+

Add instance Undo OK Cancel

Ahora le damos a ok y vemos como cambia el histograma. Para que todo siga igual volvemos a poner el valor b de nuevo antes de seguir, de manera más rápida le damos al botón Undo (y deshace los cambios). Se aprecia muy bien como el número cambia y ahora b y a tienen respectivamente 467 y 211 ejemplos.

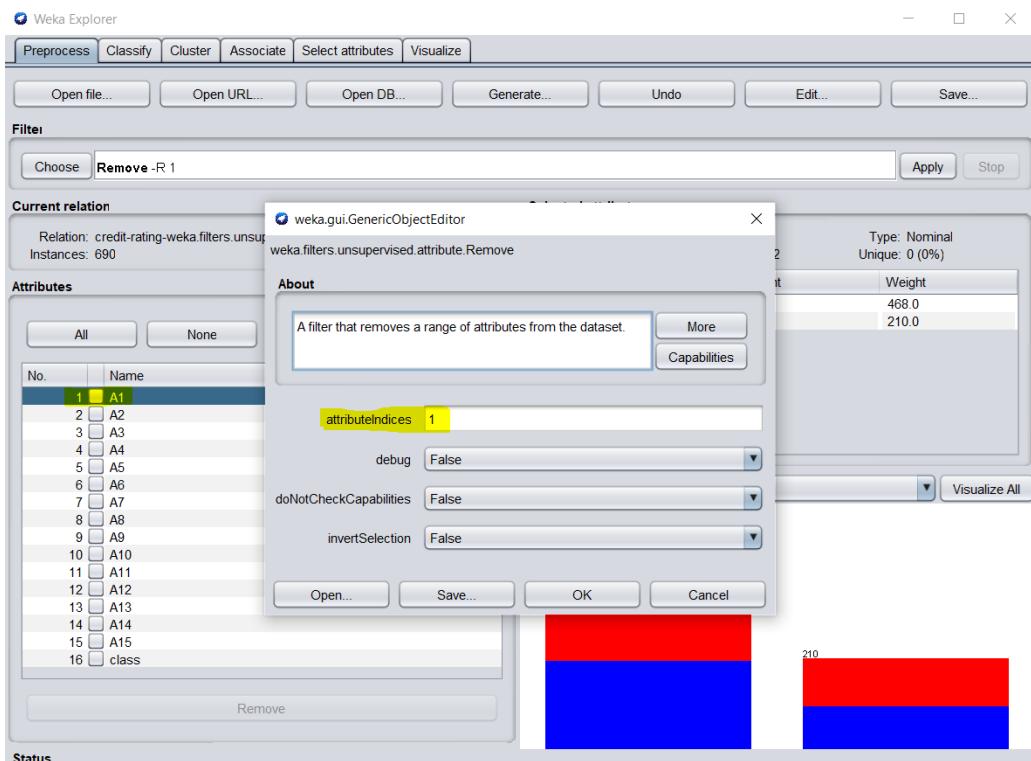


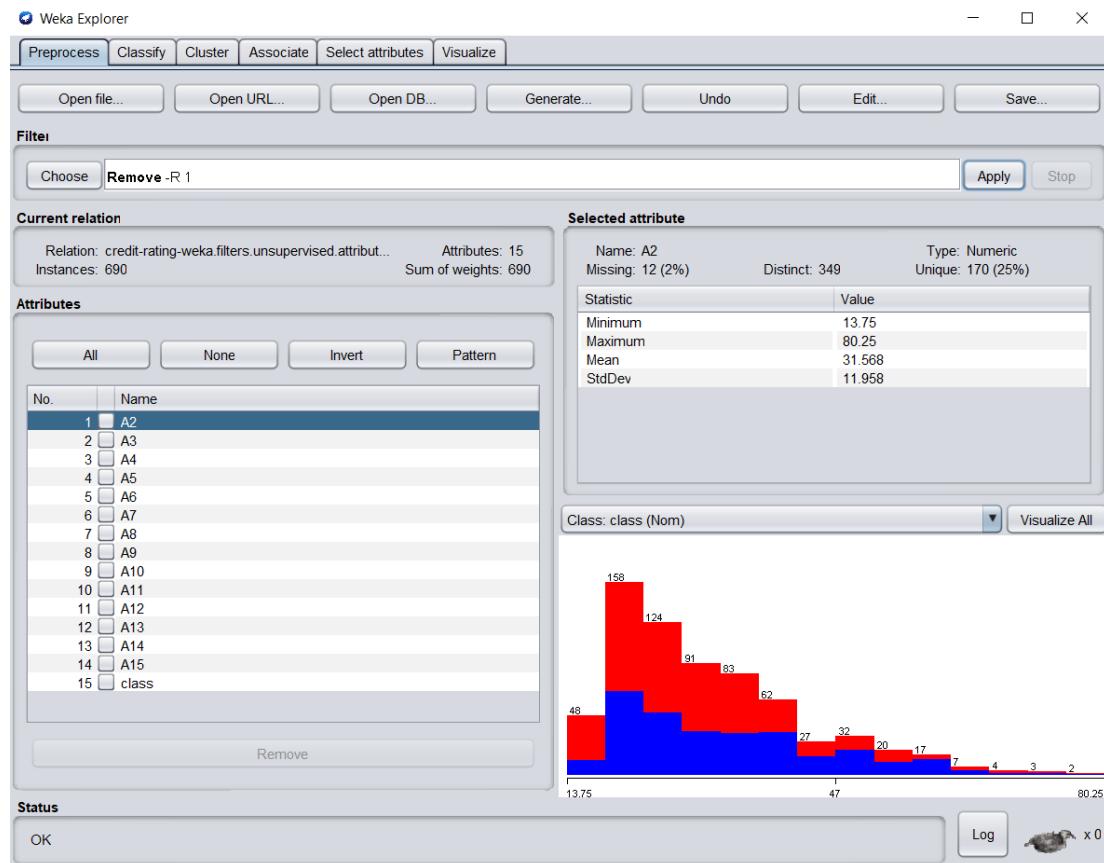
Una vez visto todo esto vamos a preprocesar los datos aplicando filtros a estos datos. Ya que Weka permite aplicar una gran diversidad de filtros sobre los datos, permitiendo realizar transformaciones sobre ellos de todo tipo. Al pulsar el botón Choose dentro del recuadro Filter se despliega un árbol en el que seleccionar los filtr a escoger.

REMOVE

El filtro más simple es Remove, su función es eliminar algunos atributos elegidos. Para ello hacemos clic en Choose > weka > filters > unsupervised > attribute > Remove.

Hacemos clic en el filtro para editarlo. En attributeIndices ponemos el índice del atributo que queremos eliminar. En este caso si quiero eliminar el atributo A1 pongo 1 ya que está en esa posición y le doy a ok seguido de Apply.



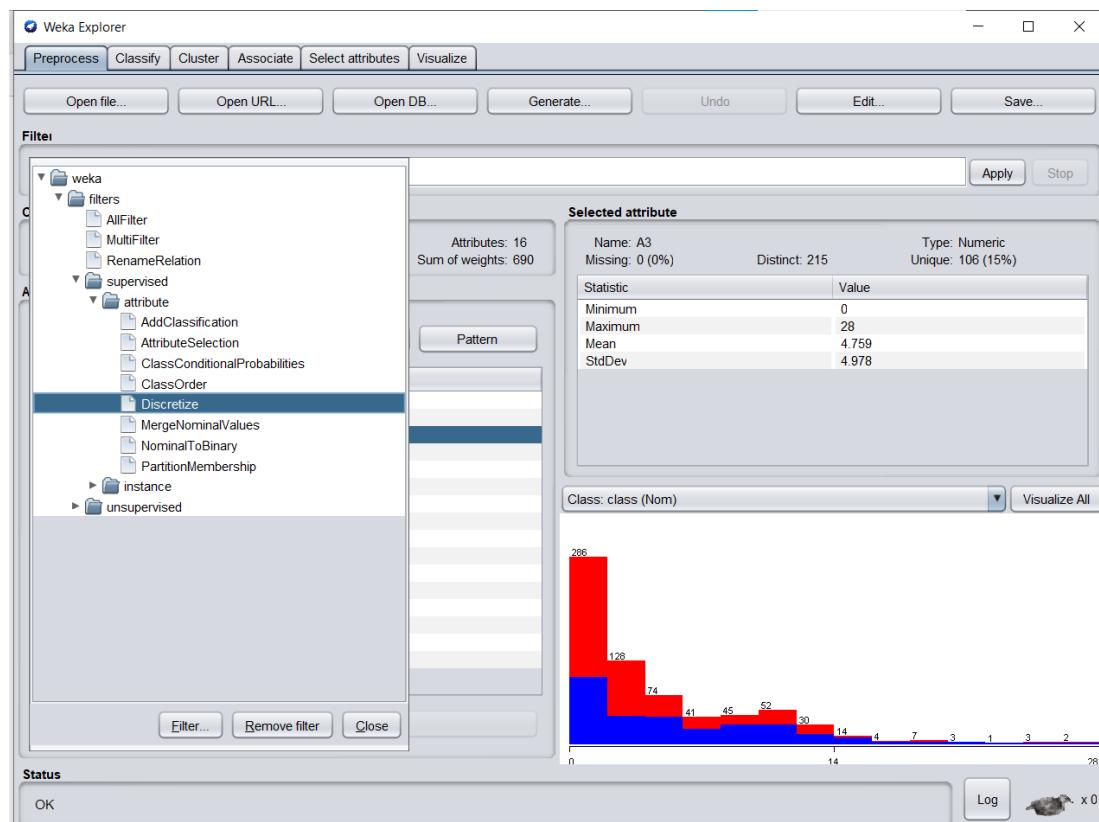


Como hemos visto anteriormente y veremos otra manera de eliminar atributos sin utilizar filtros es seleccionando el atributo o los atributos que quieras eliminar y dándole a Remove.

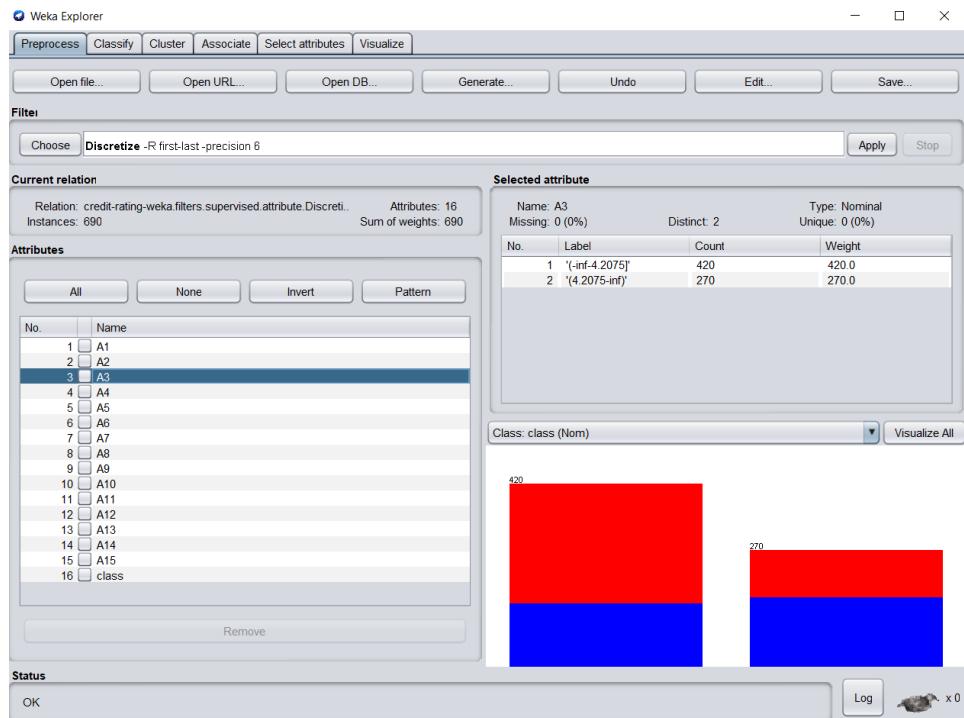
DISCRETIZE

Discretiza atributos numéricos y los convierte en nominales según ciertos parámetros.

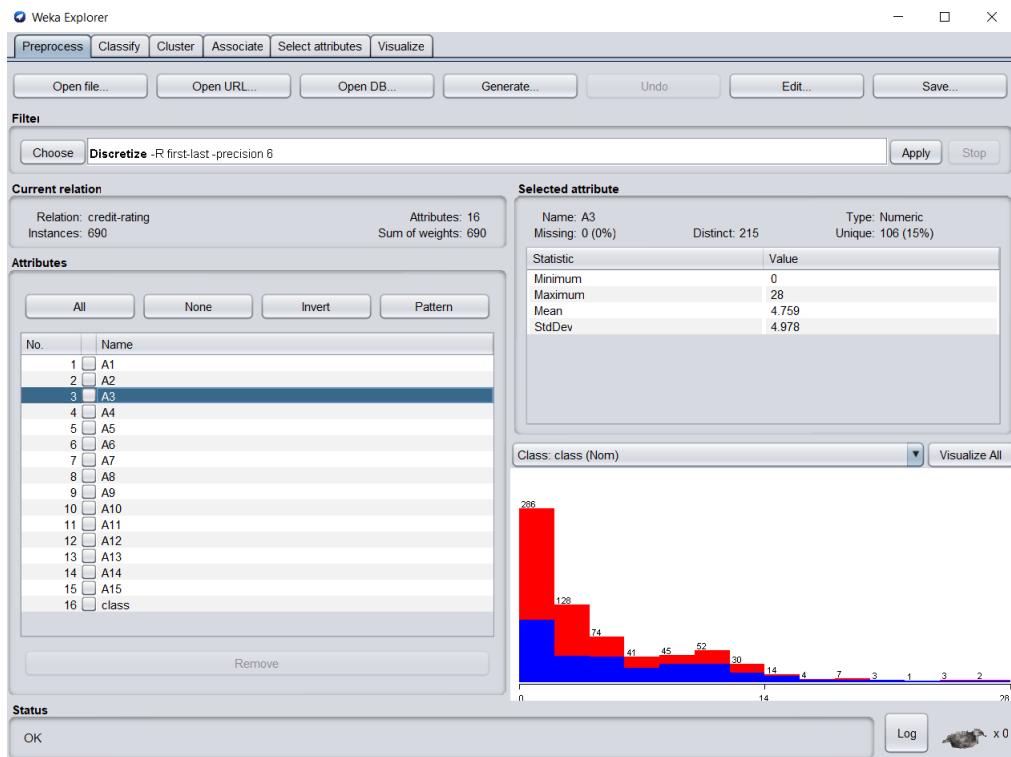
Para ello le damos a Choose > Weka > filters > supervised > attribute > Discretize



Al darle a apply nos queda:



Antes teníamos:



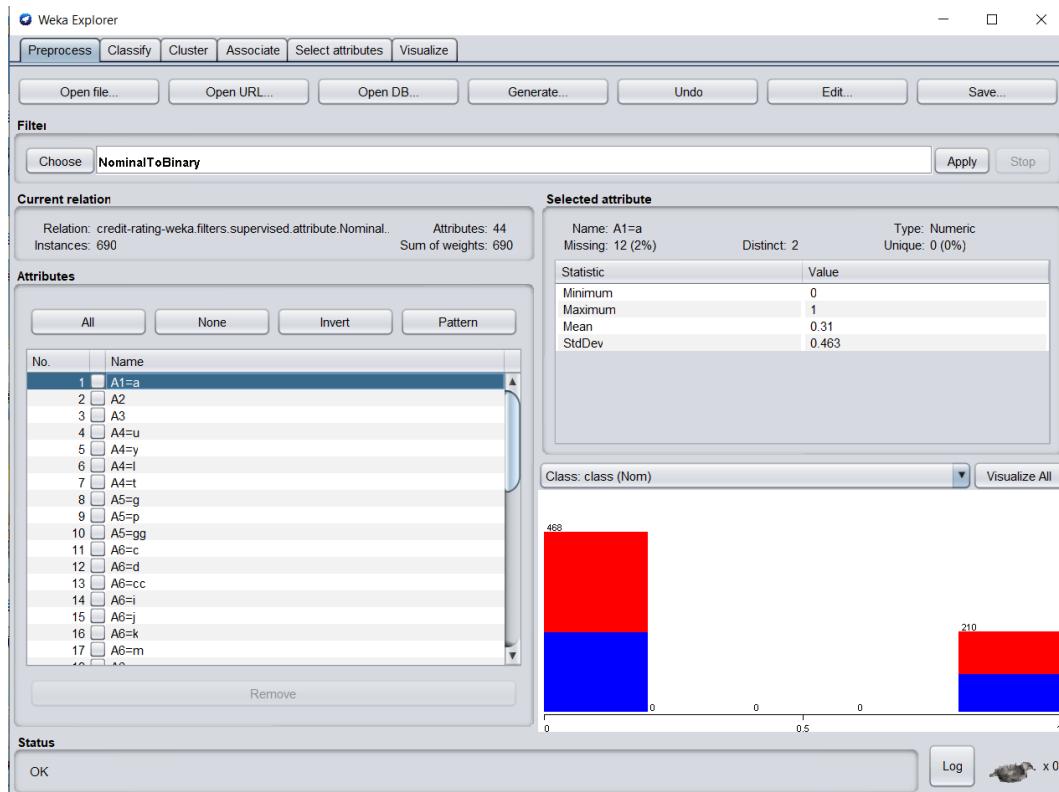
Lo que ha pasado es que ha cambiado de tipo numérico a tipo nominal como podemos apreciar en las capturas anteriores. Claro ejemplo en el atributo A3. Así tenemos todos los atributos de tipo nominal. Para retroceder, es decir, quedarnos como al principio se le da siempre a Undo.

NOMINALTOBINARY

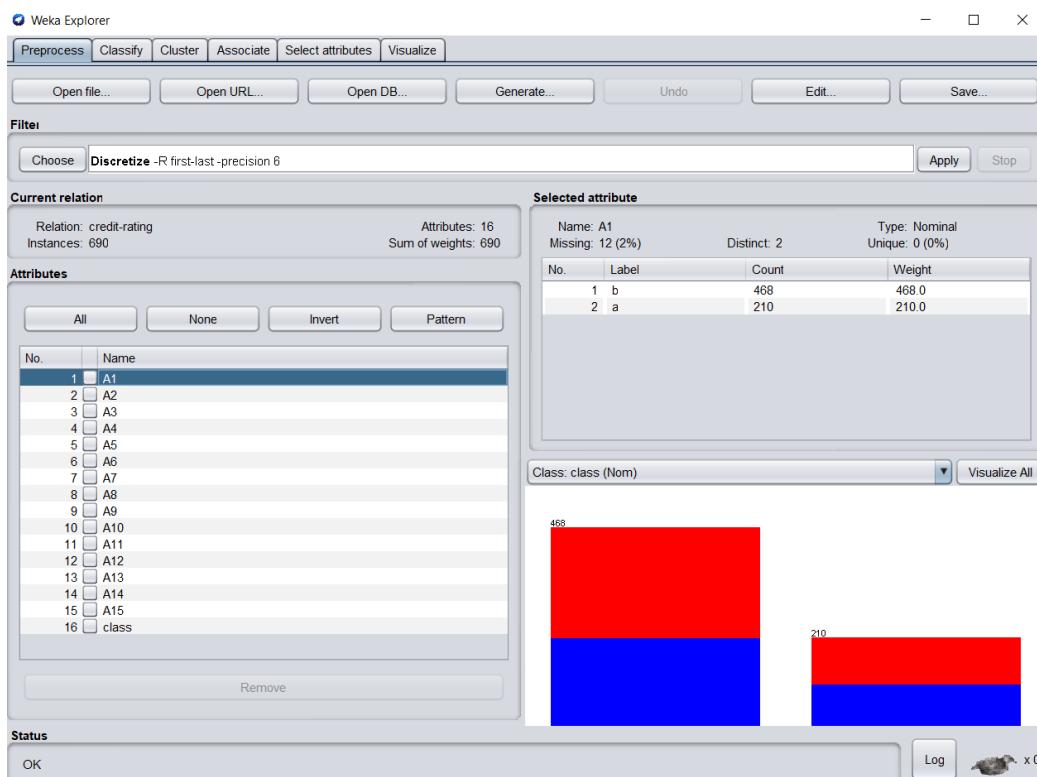
Convierte los atributos nominales en numéricos binarios, de manera que un atributo nominal con k valores se convierte en k atributos binarios.

Choose> Weka > filters > supervised > attribute > NominalToBinary

Se nos queda todos los atributos de tipo numérico.



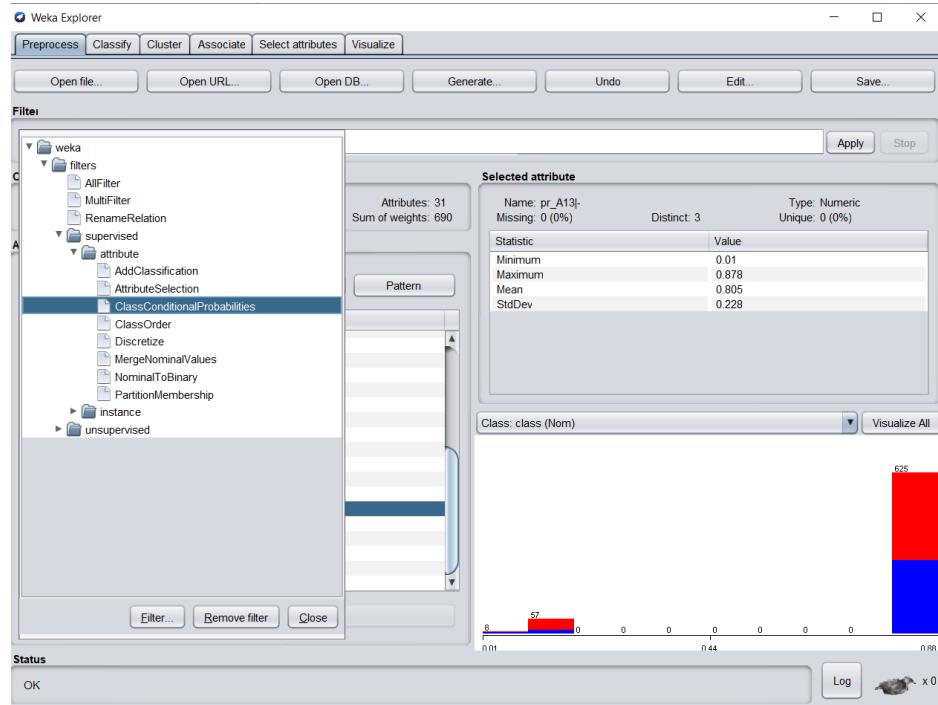
Y antes:



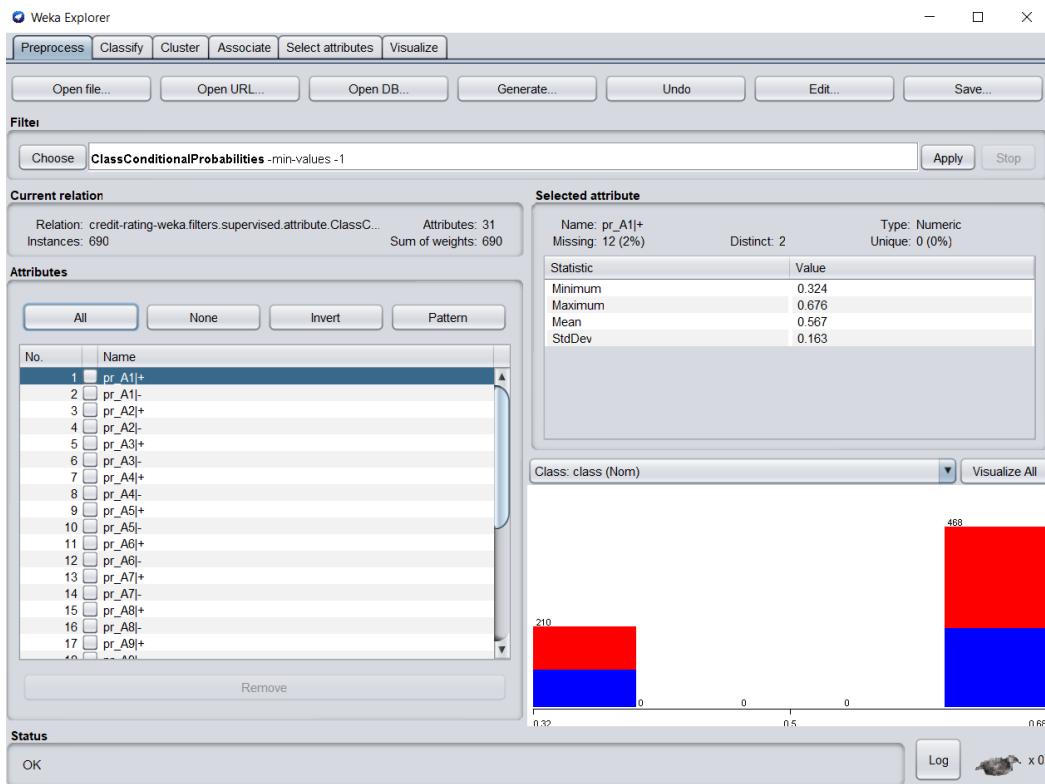
CLASS CONDITIONAL PROBABILITIES

Convierte los valores de atributos nominales y / o numéricos en probabilidades condicionales de clase.

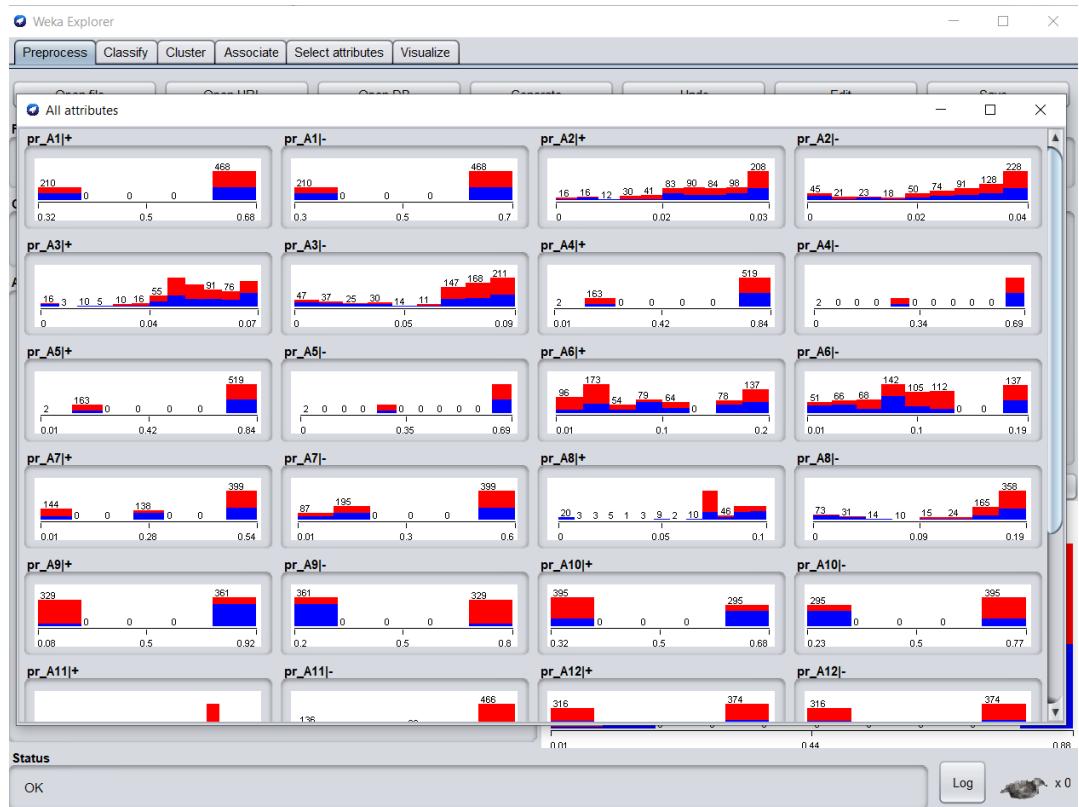
En choose seleccionamos filters > supervised > attribute > ClassConditionalProbabilities



Y obtenemos:



Le damos a `visualize all` para ver todos los atributos.

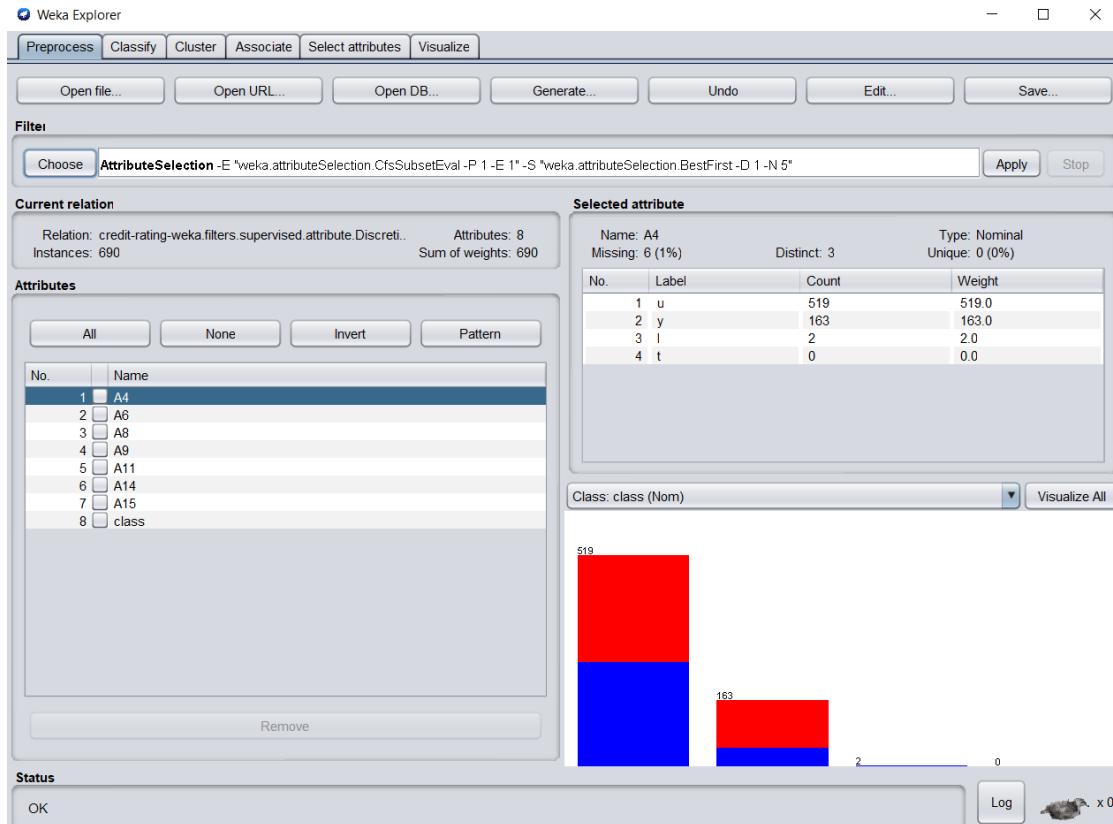


ATTRIBUTE SELECTION

Choose > Weka > filters > supervised > attribute > AttributeSelection

Permite seleccionar atributos mediante métodos de búsqueda y evaluación. Es un filtro muy flexible.

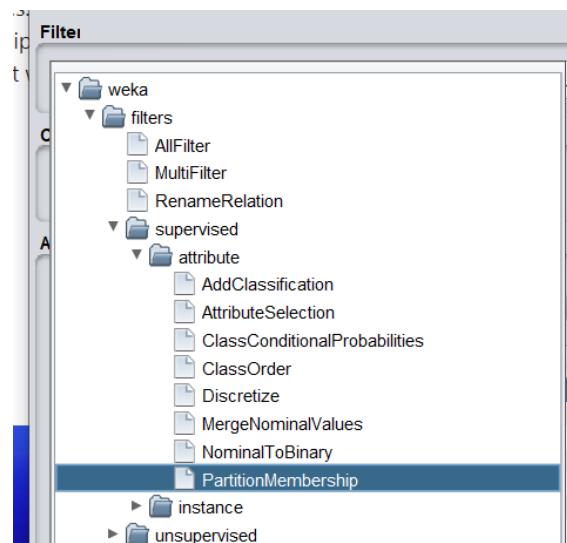
Como se puede ver en la captura adjuntada se han eliminado los atributos de A1, A2, A3, A5, A7, A10, A12, A13, A14



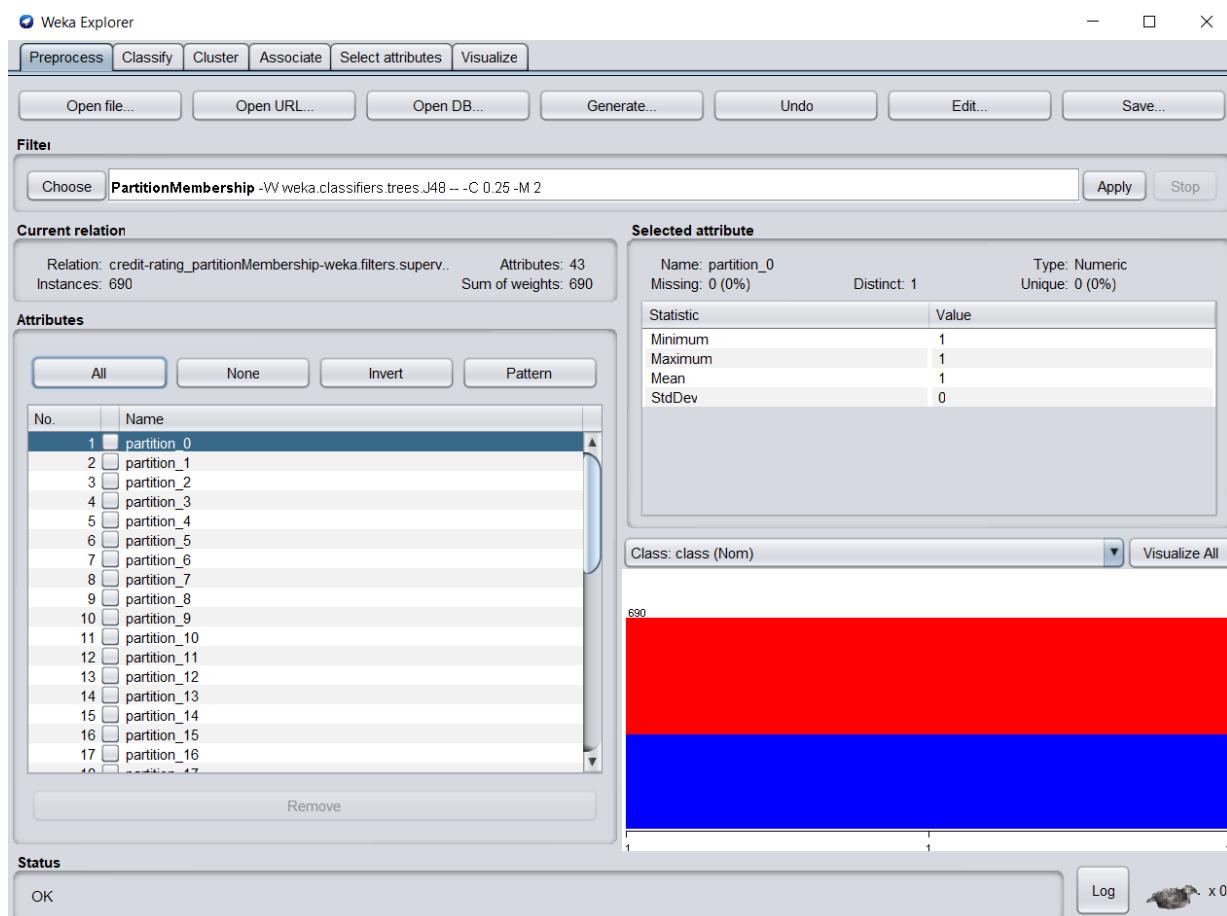
PARTITIONMEMBERSHIP

Un filtro que usa un generador de particiones para generar valores de pertenencia a particiones. Las instancias filtradas se componen de estos valores más el atributo de clase (si se establece en los datos de entrada) y se representan como instancias dispersas.

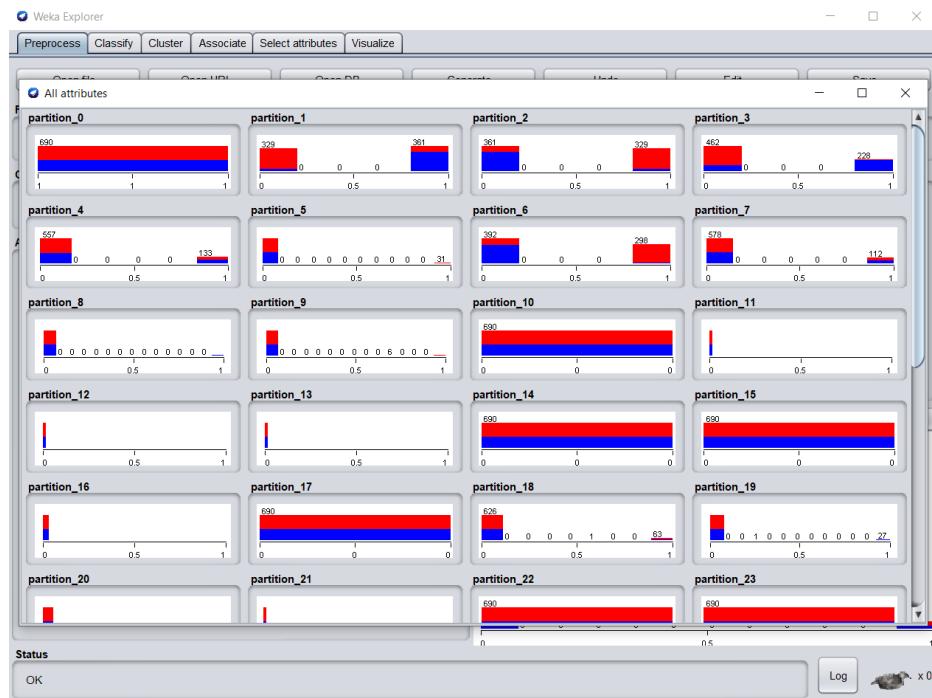
En choose elegir filters > supervised > attribute > PartitionMemberShip



Obtenemos:



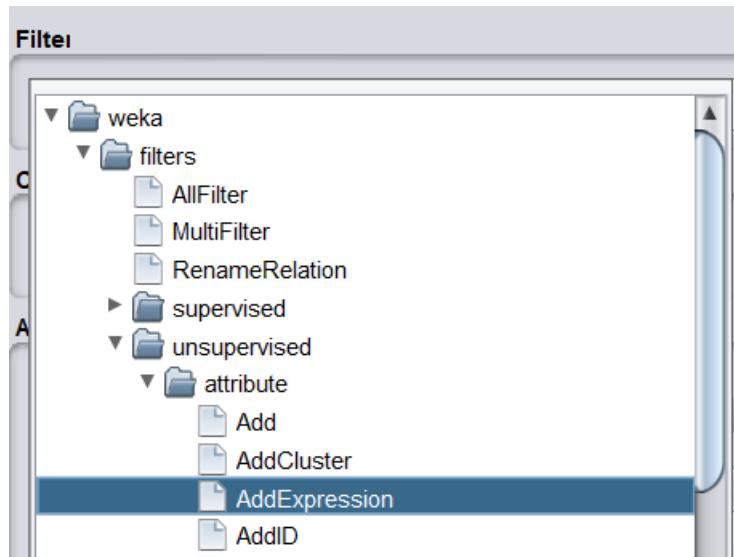
Al darle a Visualize all vemos todas las particiones que hay.



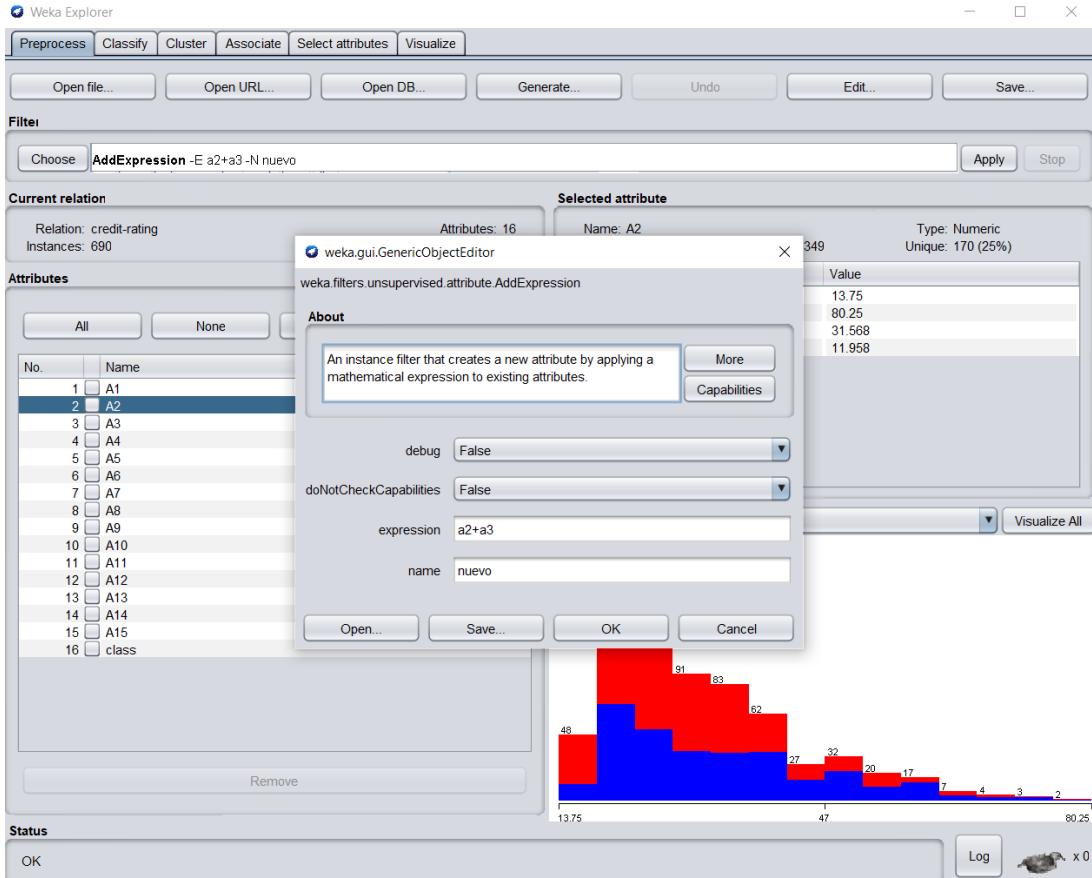
ADDEXPRESSION

Es un filtro de instancia que crea un nuevo atributo aplicando una expresión matemática a los atributos existentes.

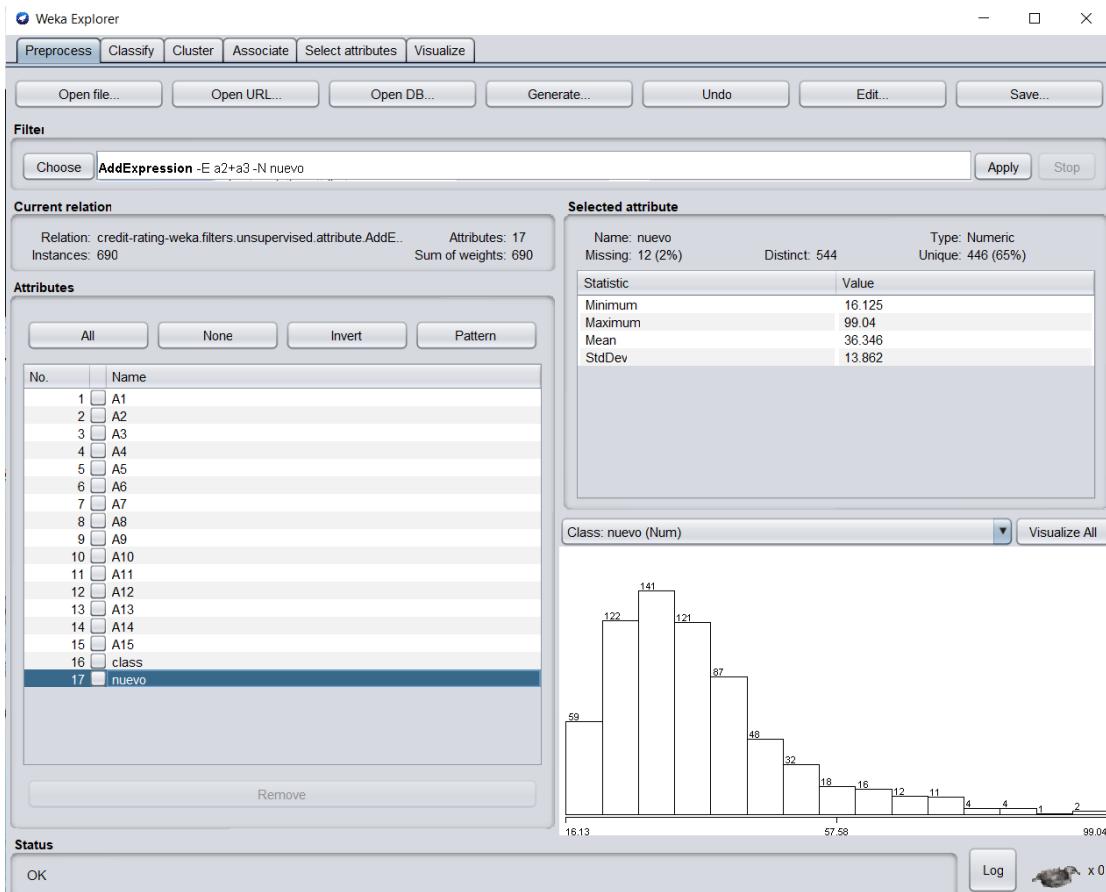
En choose seleccionamos filters> unsupervised >attribute > AddExpression



Antes de darle a apply hacemos clic en el filtro para modificarlo. En name se pone el nombre del nuevo atributo y en expresión la expresión matemática que queremos aplicar (suma, resta, multiplicación, división..)



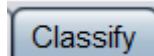
Al aplicar el filtro vemos en la siguiente captura como en la lista de Attributes (esquina izquierda abajo) aparece el nuevo atributo que hemos creado sumando $a2+a3$. Como ya sabemos en Selected attribute sale la información del atributo. El nombre, de que tipo es, los valores... Y en la esquina derecha inferior lo vemos de manera gráfica.



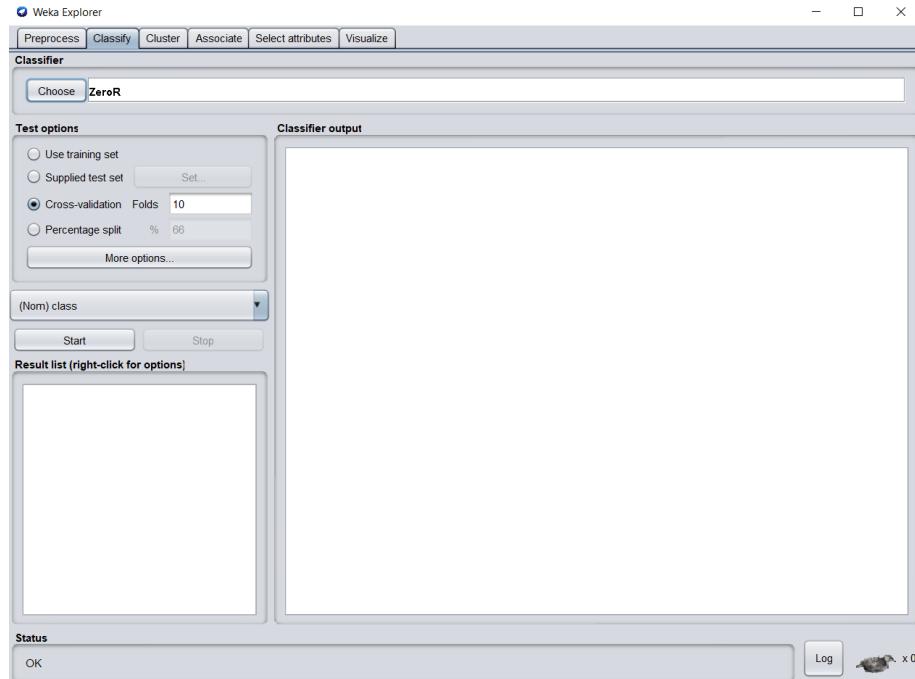
CLASIFICADORES (CLASSIFY)

Proporciona varios algoritmos de aprendizaje automático para la clasificación de sus datos. Para enumerar algunos, se puede aplicar algoritmos como Regresión lineal, Regresión logística, Máquinas de vectores de soporte, Árboles de decisión, RandomTree, RandomForest, NaiveBayes, etc. La lista proporciona algoritmos de aprendizaje automático supervisados y no supervisados.

Por ejemplo si se juega hoy al tenis o no dependiendo de cómo esté el tiempo, condiciones climáticas. (Ejercicio visto en clase). Esto depende de varias características que gracias a la construcción de árboles se puede tomar la decisión si ir a jugar al tenis hoy o no.



Le damos a la segunda pestaña del explorador y nos sale esta pantalla.

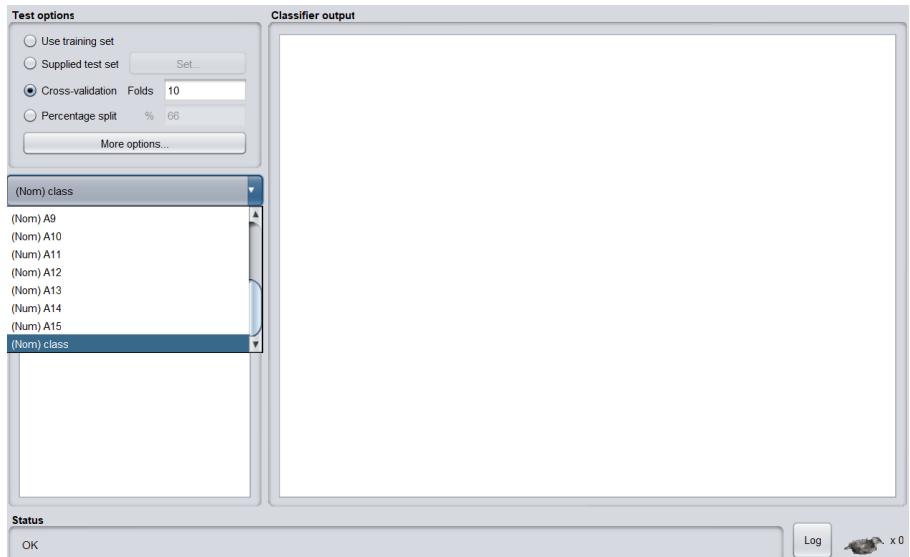


Como vemos arriba en Test options tenemos 4 opciones de prueba:

- Equipo de entrenamiento
- Equipo de prueba suministrado
- Validación cruzada (Folds es el número de pliegues en los que se dividirán y se puede modificar)
- División porcentual (Con el porcentaje de división que pongamos se dividirá los datos entre entrenamiento y prueba)

Se utilizará validación cruzada o división porcentual a no ser que tenga el conjunto de entrenamiento o un conjunto de prueba ajustado por el cliente.

Seleccionamos uno de los atributos que tenemos.

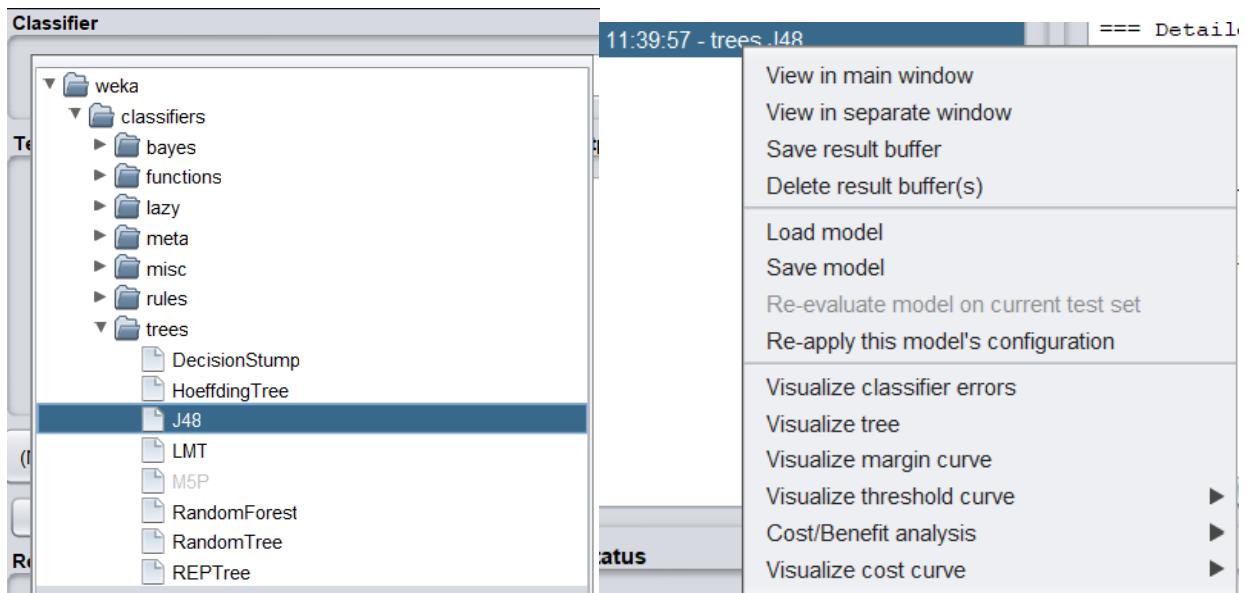


Como queremos realizar una clasificación vamos a elegir un clasificador pulsando en Classifier → Choose (que nos deja elegir entre los clasificadores disponibles con los cuales nos haremos una idea de la bondad del modelo elegido)

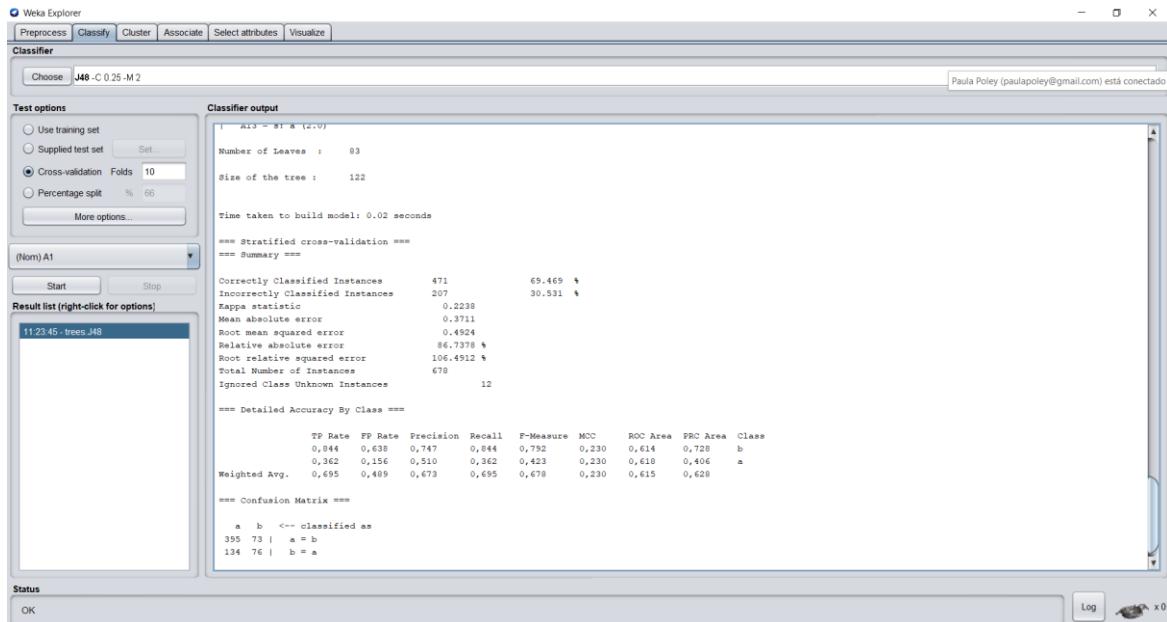
J48

J48 es una implementación del algoritmo ID3 con pequeñas modificaciones.

Choose → weka → Classifier → tres → J48

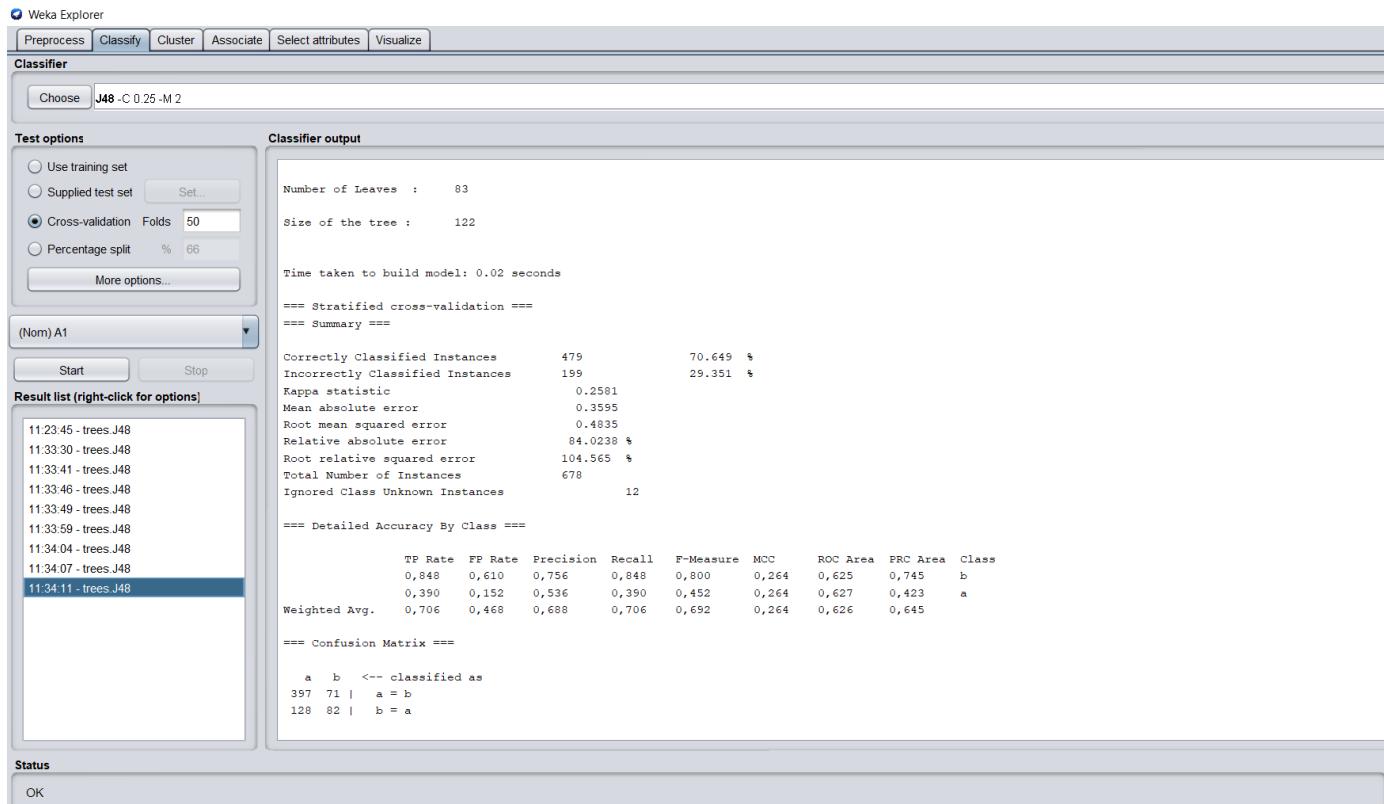


Y para iniciar el proceso de clasificación le damos a star. En el lado derecho de la captura de pantalla se ven los resultados de la clasificación.



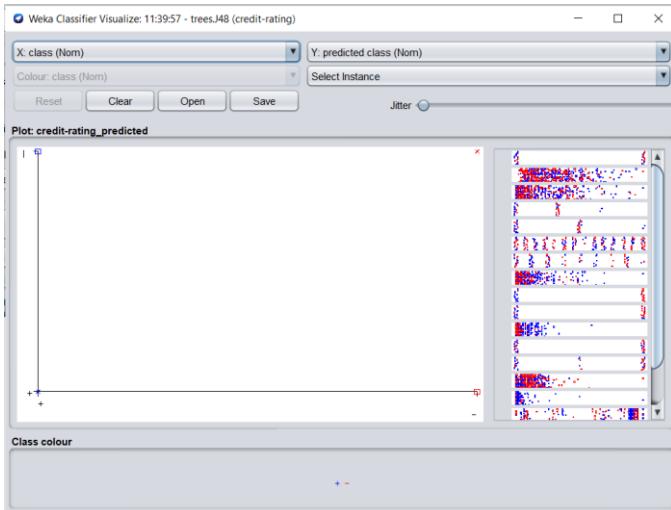
Vemos que el tamaño del árbol (size of the tree) es 122. Las instancias correctamente clasificadas 471 (69.469 %) y las instancias clasificadas incorrectamente 207 (30.531 %) . El número total de instancias 678. El error absoluto relativo es de 86,7378%. Podemos deducir gracias a estos resultados mostrados que la clasificación no es muy aceptable.

En cambio si el número de pliegues (Folds) lo cambiamos de 10 a 50 es decir, dividimos en 50 partes el conjunto de datos. Vemos como el número de estancias correctamente clasificadas aumenta (479 → 70.649 % pero de todas formas la clasificación no es muy aceptable.



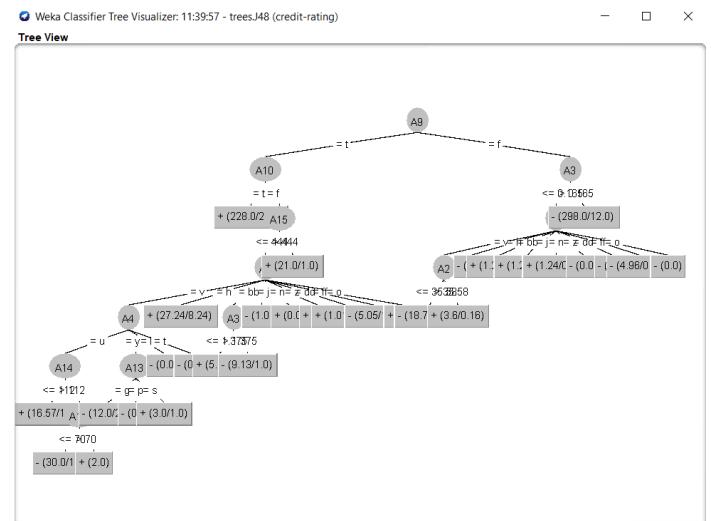
Nos vamos a la caja Result list nos colocamos encima del resultado que queremos y le damos click derecho. Seleccionamos Visualize tree y así vemos la representación visual del árbol transversal. A parte de esta opción hay muchas otras más. Veremos sus capturas abajo.

VISUALIZAR ERRORES DE CLASIFICADOR

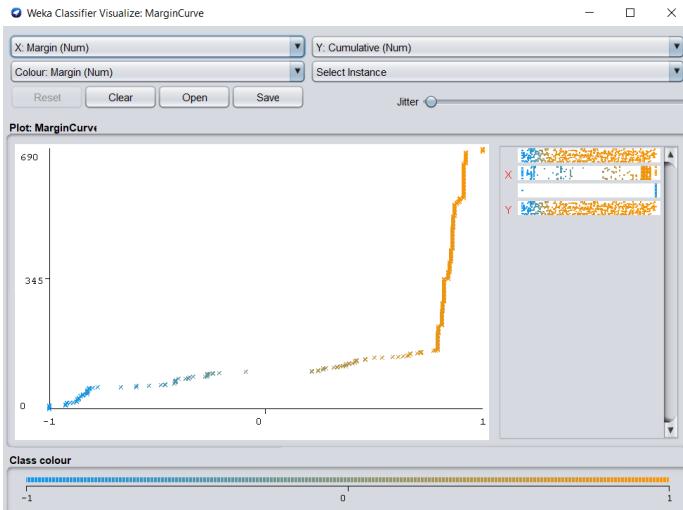


Si le cambiamos, seleccionamos distintos atributos en X e Y. Estos ejes cambian al igual que la gráfica mostrada. Los mismo sucede si seleccionamos las franjas que estan en el lado derecho. Si le damos con el click izquierdo seleccionamos la X y si le damos con el click derecho seleccionamos la Y.

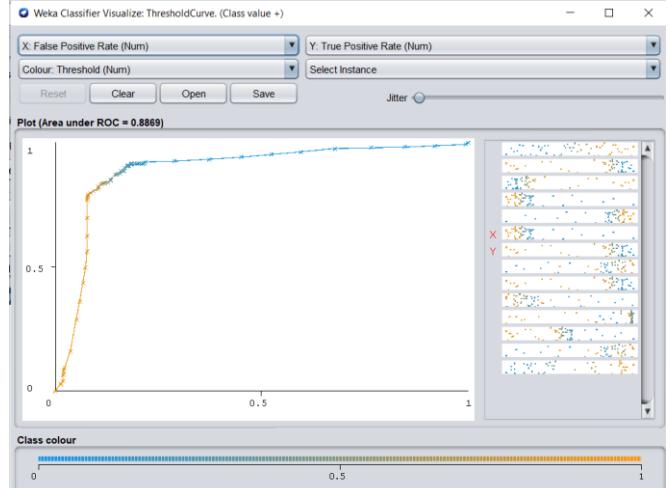
VISUALIZAR ÁRBOL



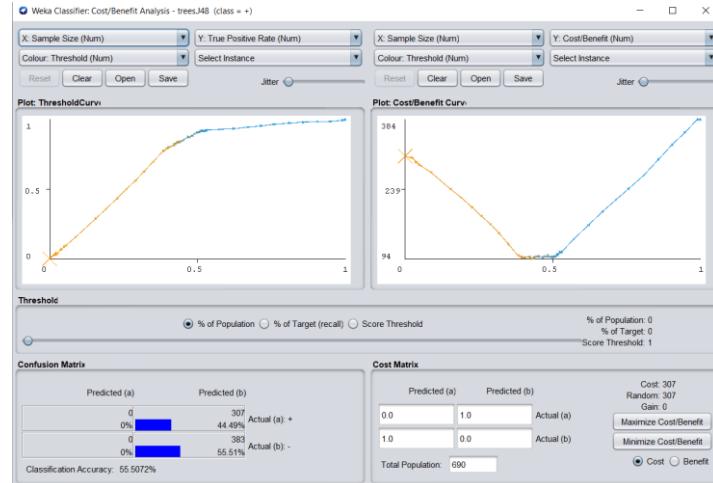
VISUALIZAR CURVA DE MARGEN



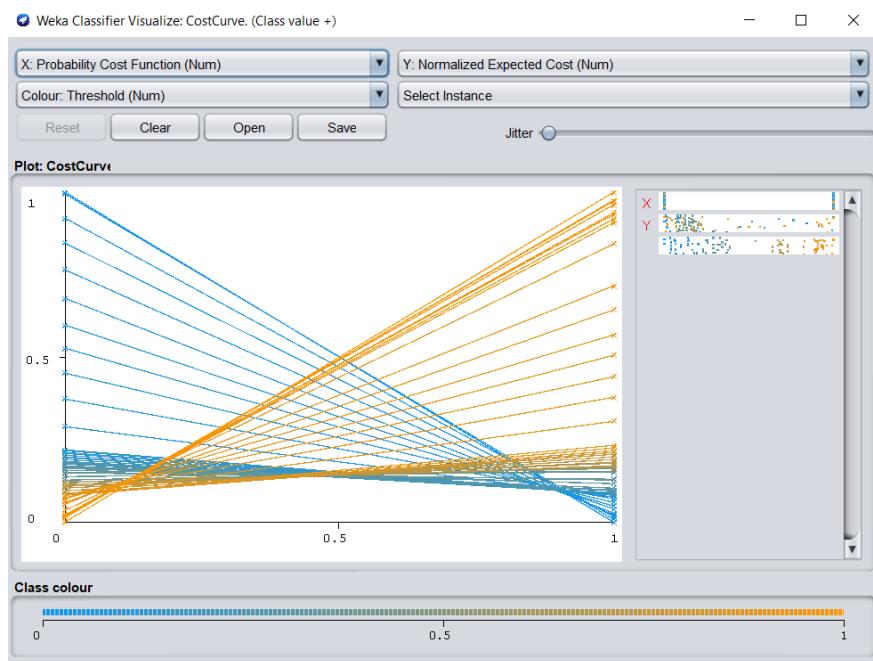
VISUALIZAR LA CURVA DE UMBRAL +



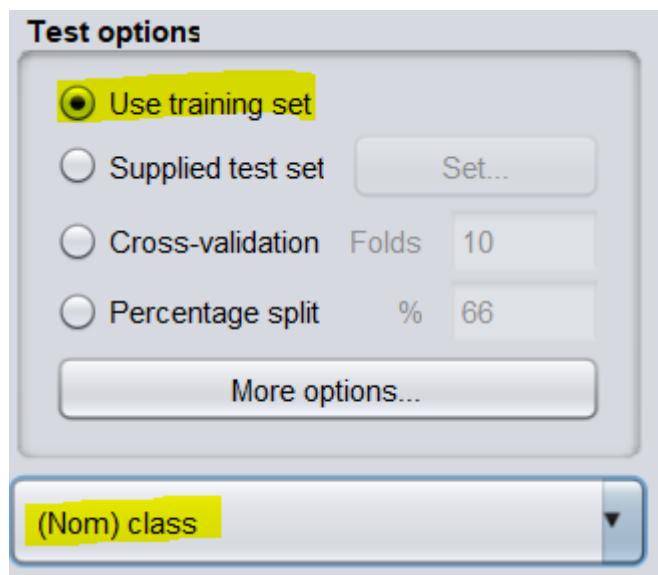
ANÁLISIS COSTE-BENEFICIO +



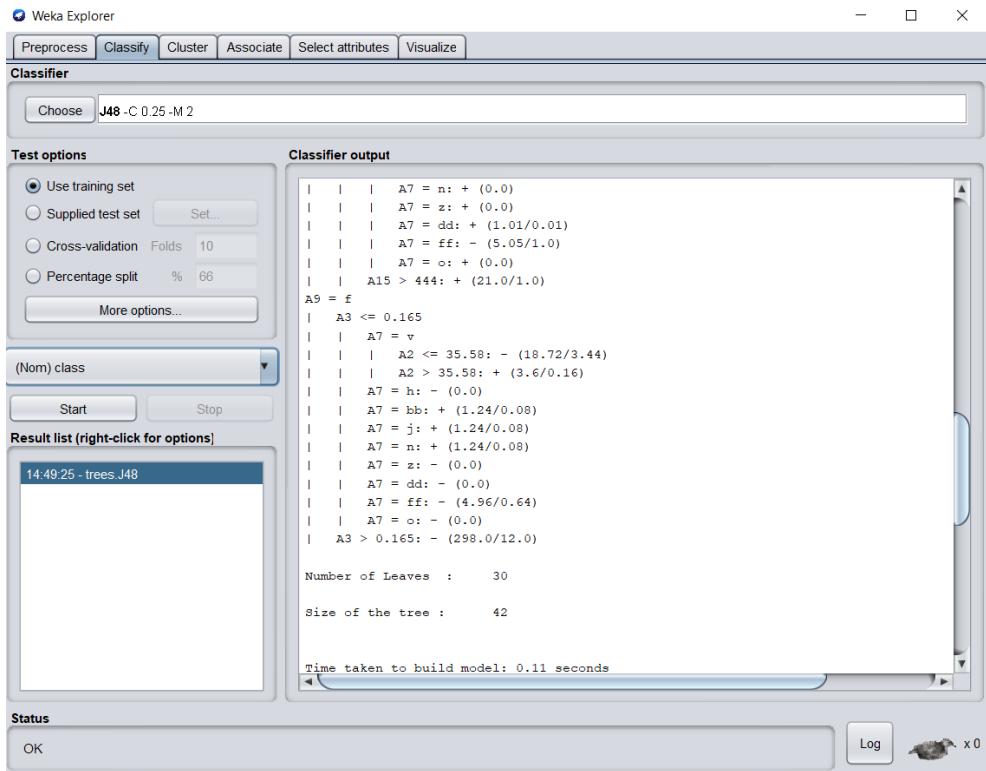
VISUALIZAR LA CURVA DE COSTOS +



Cambiemos, ahora vamos a irnos a Test options y vamos a seleccionar use training set y observamos que aparece como atributo de clasificación (Nom) class que es el último que aparece en la lista.



Le damos a star para generar el árbol donde aparecerá en modo texto con la información.

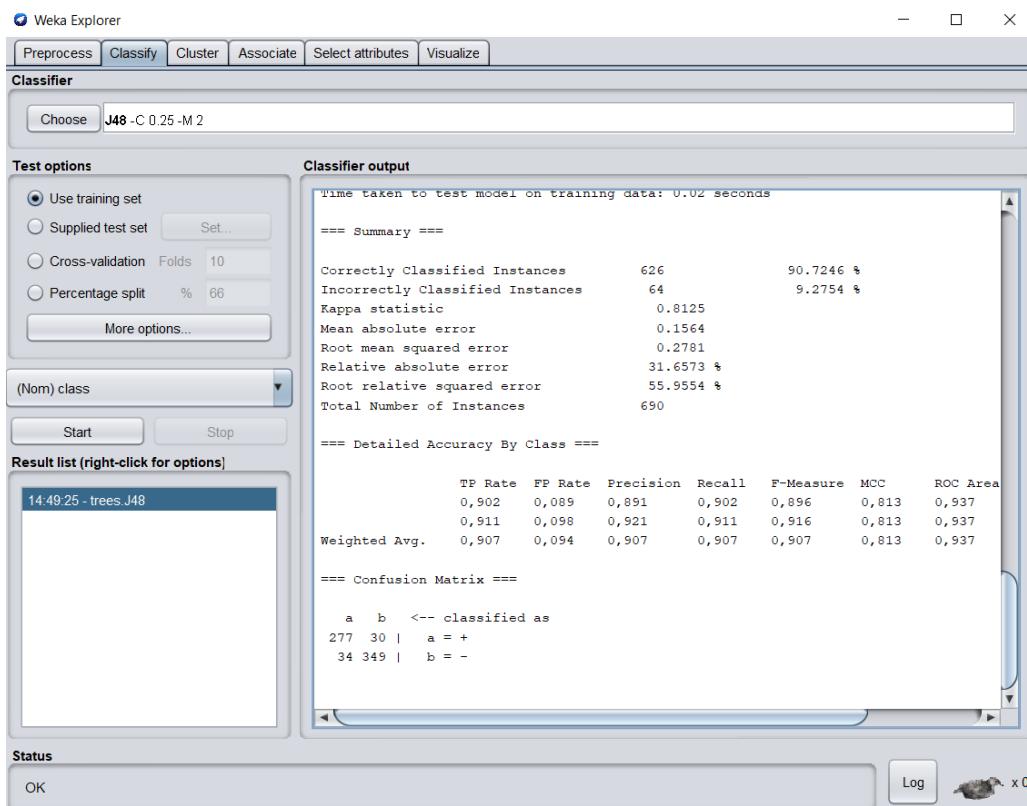


Ha encontrado un árbol con 42 nodos (Size of the tree) de los cuales 30 son hojas (number of leaves). Ha clasificado correctamente los 626 ejemplos de la base de datos (el 90.7246 %).

==== Confusion Matrix ===

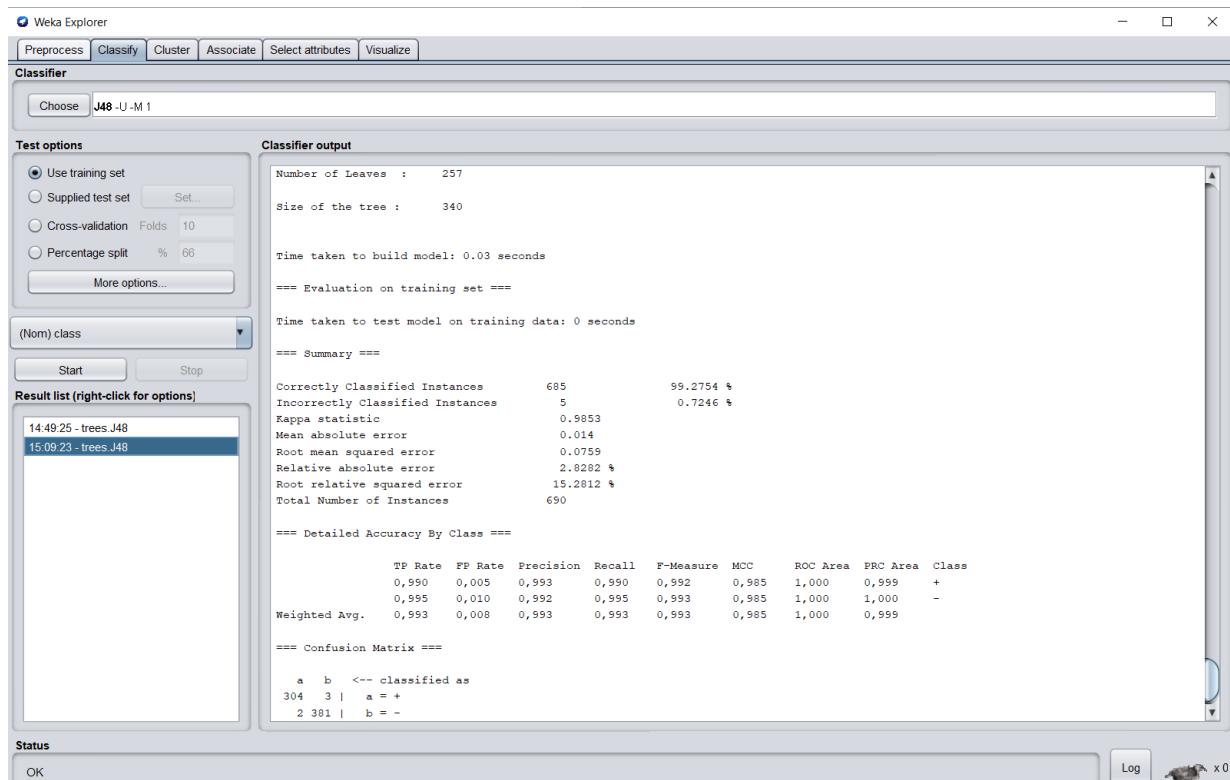
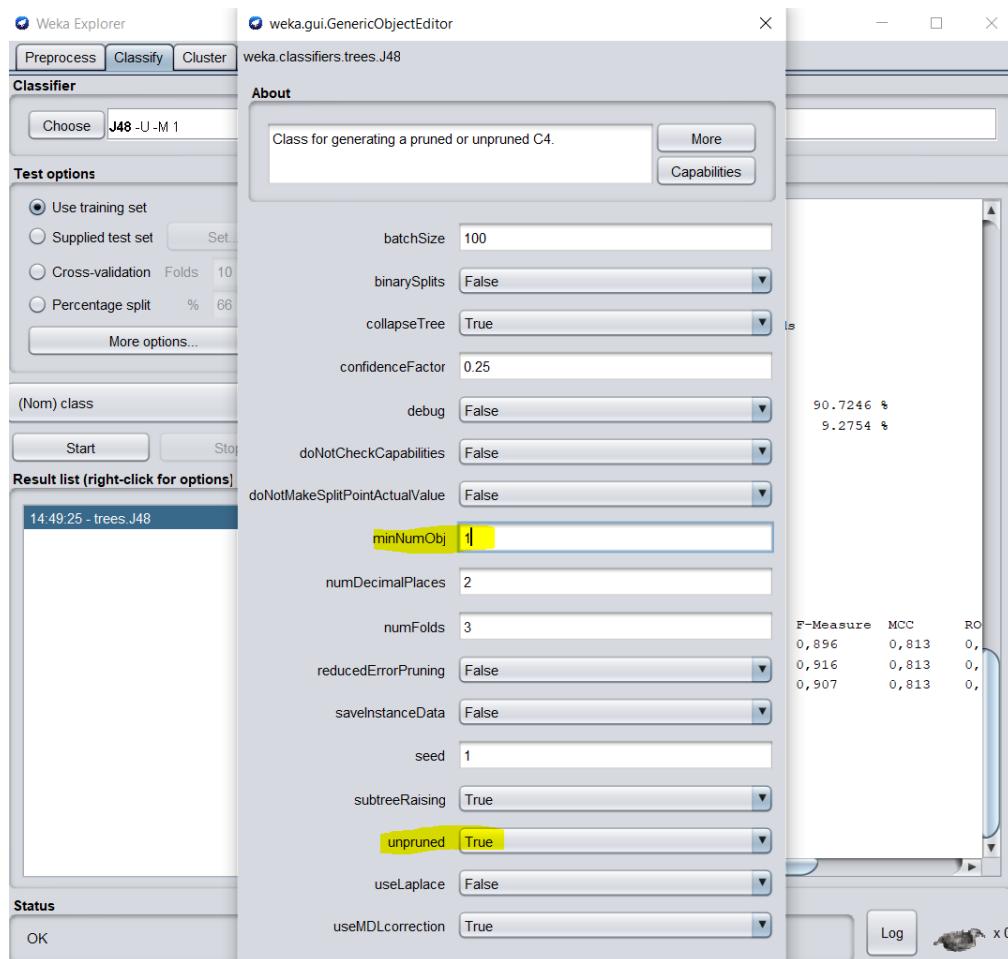
		a b <-- classified as
a	277	30 a = +
b	34	349 b = -

La Matriz de confusión que aparece al final nos dice que de los 307 ejemplos con clasificación+, el árbol clasifica 277 como + y 30 como -. Análogamente, de los 383 ejemplos con clasificación -, el árbol clasifica 349 como - y 34 como +.



Ahora veamos lo que ocurre cuando no podamos el árbol.

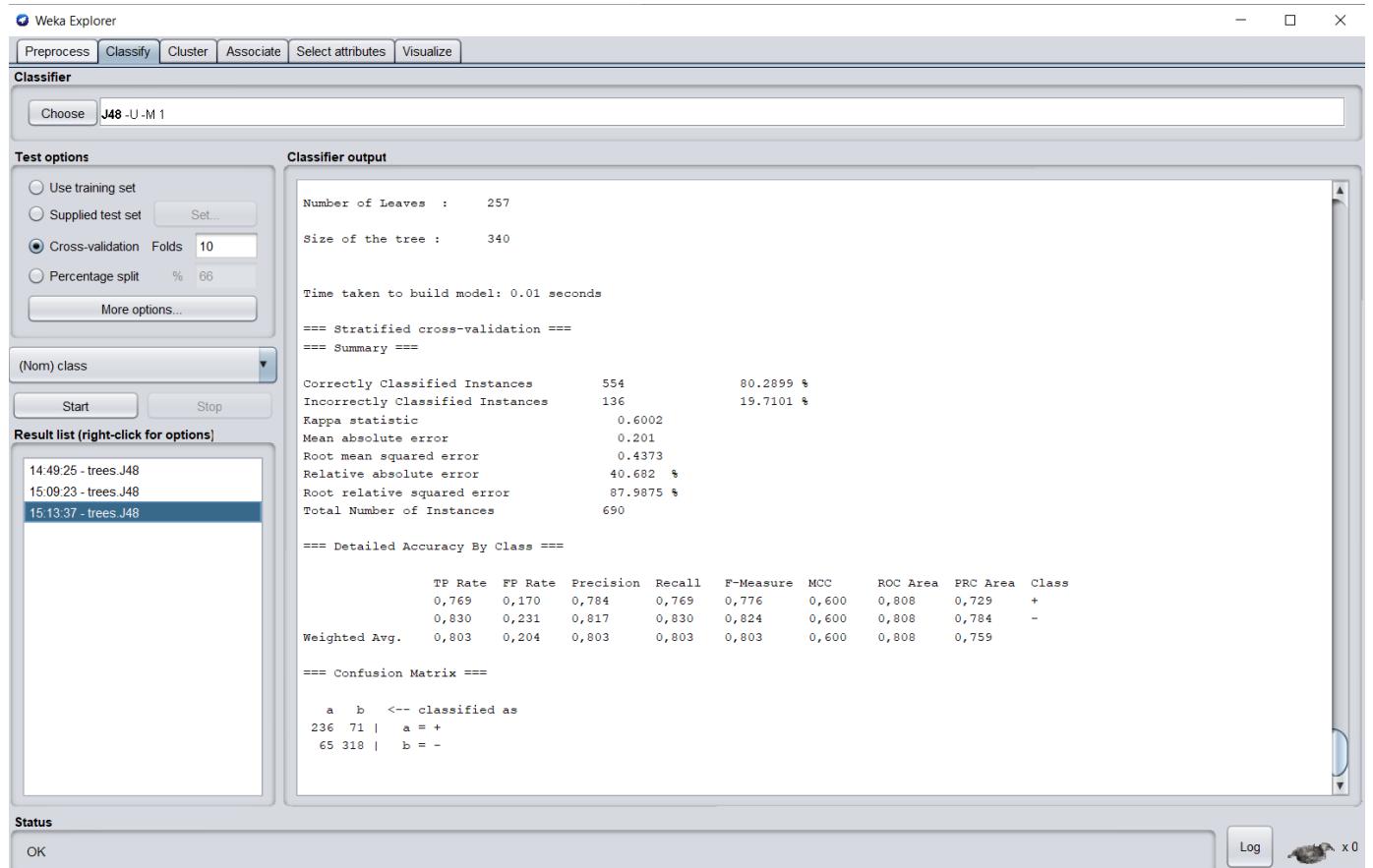
Hacemos click en el filtro para modificarlo. En Unpruned lo podemos a True para que no poda el árbol obtenido y minNumObj a 1 para que admita hojas con un único ejemplo.



En la descripción ha cambiado a J48 -U -M 1. Ha encontrado un árbol con 340 nodos (size of the tree), de los cuales 257 son hojas (number of leaves) . Ha clasificado correctamente los 685 ejemplos de la base de datos (el 99,2754 %). La Matriz de confusión que aparece al final nos dice que de los 307 ejemplos con clasificación +, el árbol clasifica 304 como + y 3 como -. Análogamente, de los 383 ejemplos con clasificación -, el árbol clasifica 381 como - y 2 como +.

Esto ha sido una mejora en cuanto a los anteriores.

En cambio si no podamos el árbol, admitimos hojas con un único ejemplo y realizamos para cada uno de ellos validación cruzada seleccionando la opción de Cross-validation con el Folds predeterminado el rendimiento empeora, este tiene un 80.2899 %, es decir 554 instancias clasificadas correctamente.



Ahora vamos a ver como es el árbol con validación cruzada con Folds 50 , sin podar el árbol y admite hojas con un único ejemplo.

Seleccionamos en Test Options Cross-validation con un Folds de 50 y hacemos click en el filtro J48 para editarlo. Ponemos minNumObj a 1 y unpruned a True.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 -U -M 1

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 50
- Percentage split % 66

More options...

(Nom) A13

Start Stop

Result list (right-click for options)

- 14:49:25 - trees.J48
- 15:09:23 - trees.J48
- 15:13:37 - trees.J48
- 16:09:14 - trees.J48
- 16:09:20 - trees.J48
- 16:10:13 - rules.OneR
- 17:04:28 - trees.J48

Classifier output

```
==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances          608           88.1159 %
Incorrectly Classified Instances        82            11.8841 %
Kappa statistic                         0.3173
Mean absolute error                     0.0801
Root mean squared error                 0.2779
Relative absolute error                 68.7465 %
Root relative squared error            115.8164 %
Total Number of Instances               690

==== Detailed Accuracy By Class ====

```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC
0,934	0,615	0,936	0,934	0,935	0,317	
0,625	0,009	0,455	0,625	0,526	0,527	
0,333	0,057	0,345	0,333	0,339	0,281	
Weighted Avg.	0,881	0,562	0,882	0,881	0,881	

```
==== Confusion Matrix ====

```

	a	b	c	<-- classified as
584	5	36	1	a = g
3	5	0	1	b = p
37	1	19	1	c = s

Status OK Log x 0

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 -U -M 1

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 50
- Percentage split % 66

More options...

(Nom) A13

Start Stop

Classifier output

```

|   |   A7 = n: g (1.01)
|   |   A7 = z: g (1.01)
|   |   A7 = dd: g (0.0)
|   |   A7 = ff: g (11.12)
|   |   A7 = o: g (0.0)
|   A4 = l: p (1.0)
|   A4 = t: g (0.0)

```

```
Number of Leaves : 153
```

```
Size of the tree : 221
```

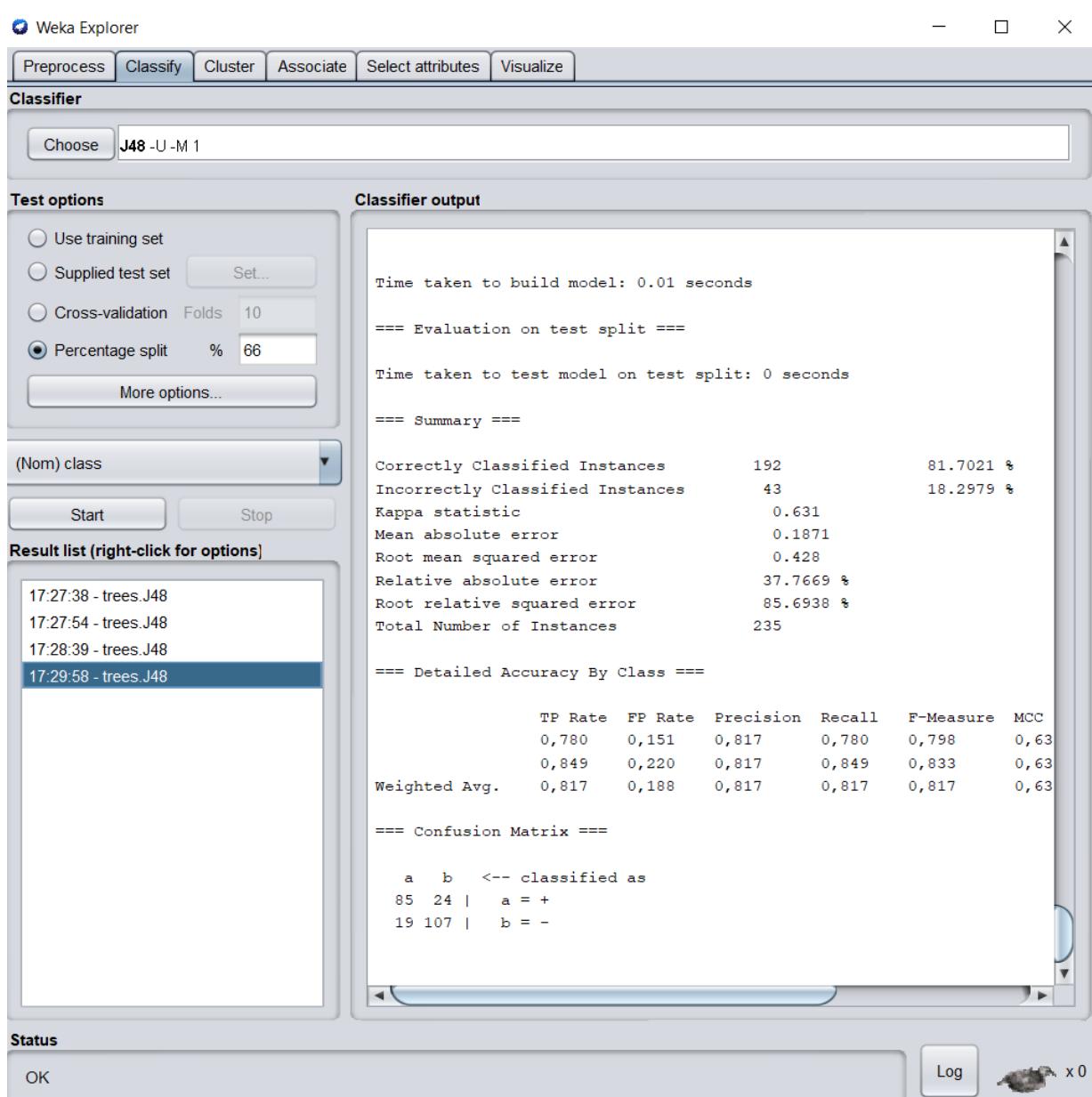
Con estas modificaciones hemos obtenido un rendimiento del 88,1159 % es decir 608 instancias correctamente clasificadas. Frente a un 11,8841% de instancias mal clasificadas. También podemos ver que el árbol tiene 221 nodos de los cuales 153 son hojas.

Haremos otro experimento, seleccionaremos percentage split con % predeterminado que es 66%, haciendo click en el filtro cambiamos minNumObjt = 1 y unpruned = True para que no se poda el árbol y que admita hojas con un único ejemplo. Obtenemos un 81,7021 % de rendimiento ya que tiene 192 instancias correctamente clasificadas. El número de nodos del árbol es de 340 de los cuales 257 son hojas.

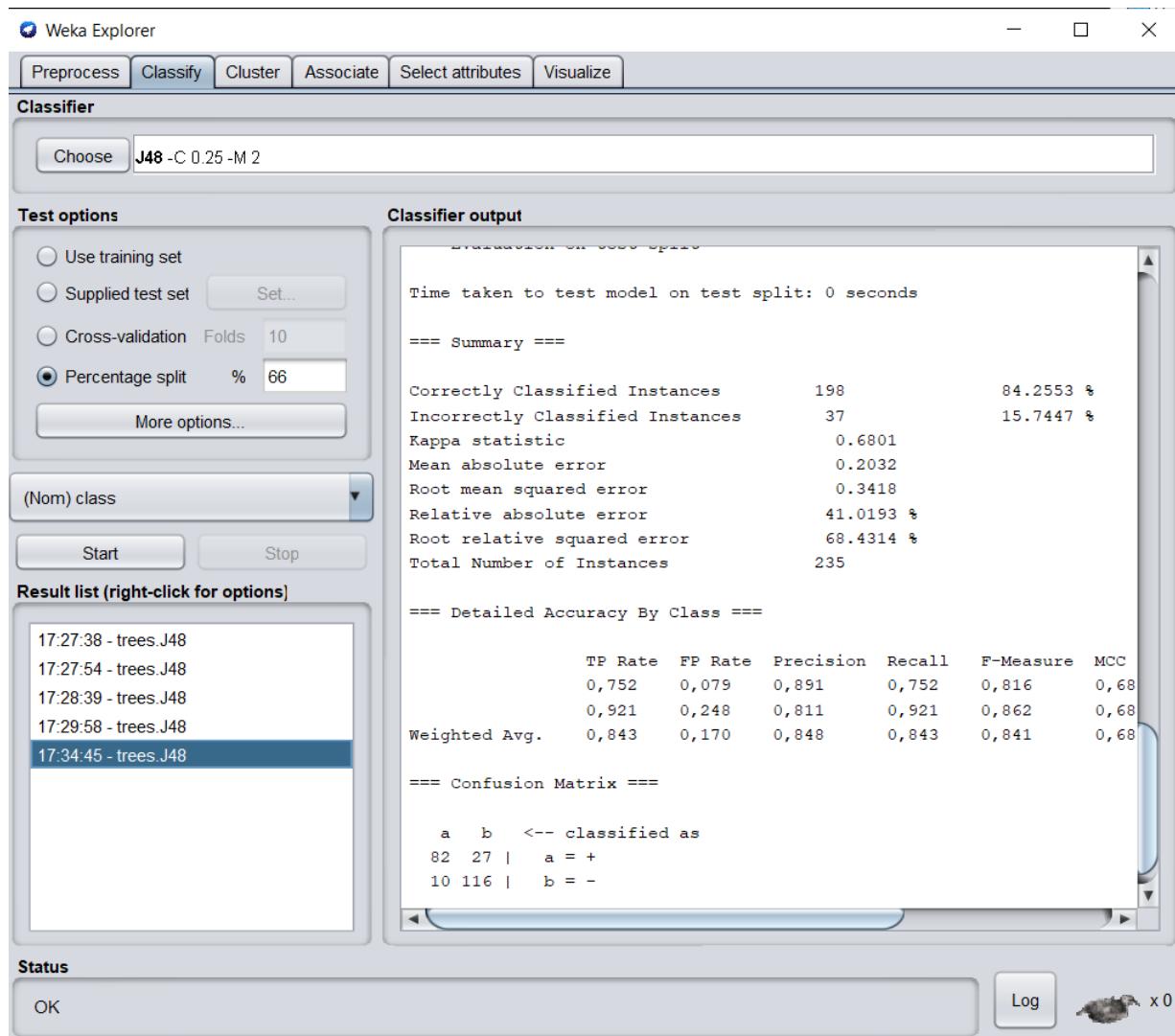
```
|   |   A4 = l: + (2.0)
|   |   A4 = t: - (0.0)

Number of Leaves :      257

Size of the tree :      340
```



Si en cambio dejamos los valores predeterminados (es decir minNumObj = 2 y unpruned = False) aumenta aunque poco el rendimiento a un 84,2553%.



Si dejamos los valores antes mencionados (predeterminados) y cambiamos el % a 79% el rendimiento sigue aumentando a un 90,3448% . El árbol se forma por 42 nodos de los cuales 30 son hojas.

```

|   |   A7 = o: - (0.0)
|   A3 > 0.165: - (298.0/12.0)

Number of Leaves :      30

Size of the tree :      42

Time taken to build model: 0.01 seconds

```

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 -C 0.25 -M 2

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 10
- Percentage split % 79
- More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 17:27:38 - trees.J48
- 17:27:54 - trees.J48
- 17:28:39 - trees.J48
- 17:29:58 - trees.J48
- 17:34:45 - trees.J48
- 17:37:23 - trees.J48
- 17:37:28 - trees.J48
- 17:37:32 - trees.J48
- 17:37:39 - trees.J48
- 17:37:45 - trees.J48
- 17:37:48 - trees.J48

Classifier output

```
Time taken to test model on test split: 0 seconds

==== Summary ===

Correctly Classified Instances           131          90.3448 %
Incorrectly Classified Instances        14          9.6552 %
Kappa statistic                         0.7995
Mean absolute error                     0.1775
Root mean squared error                 0.2975
Relative absolute error                 36.0245 %
Root relative squared error            60.0794 %
Total Number of Instances               145

==== Detailed Accuracy By Class ===

      TP Rate   FP Rate   Precision   Recall   F-Measure   MCC
      0,823     0,036     0,944      0,823    0,879      0,80
      0,964     0,177     0,879      0,964    0,920      0,80
Weighted Avg.   0,903     0,117     0,907      0,903    0,902      0,80

==== Confusion Matrix ===

  a   b   <-- classified as
51 11 |  a = +
 3 80 |  b = -
```

Status OK Log x 0

Ahora vamos a ver como si con un 79% no hacemos poda y admitimos hojas con un único ejemplo el rendimiento baja a un 82.069 %. El árbol tiene 340 nodos de los cuales 257 son hojas.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 -U -M 1

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 10
- Percentage split % 79
- More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 17:27:38 - trees.J48
- 17:27:54 - trees.J48
- 17:28:39 - trees.J48
- 17:29:58 - trees.J48
- 17:34:45 - trees.J48
- 17:37:23 - trees.J48
- 17:37:28 - trees.J48
- 17:37:32 - trees.J48
- 17:37:39 - trees.J48
- 17:37:45 - trees.J48
- 17:37:48 - trees.J48
- 17:40:34 - trees.J48

Classifier output

```
Number of Leaves : 257
Size of the tree : 340

Time taken to build model: 0.01 seconds

==== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

==== Summary ===

Correctly Classified Instances           119          82.069 %
Incorrectly Classified Instances        26          17.931 %
Kappa statistic                         0.6276
Mean absolute error                     0.1901
Root mean squared error                 0.4274
Relative absolute error                 38.5877 %
Root relative squared error            86.3144 %
Total Number of Instances               145

==== Detailed Accuracy By Class ===

      TP Rate   FP Rate   Precision   Recall   F-Measure   MCC
      0,726     0,108     0,833      0,726    0,776      0,63
      0,892     0,274     0,813      0,892    0,851      0,63
Weighted Avg.   0,821     0,203     0,822      0,821    0,819      0,63

==== Confusion Matrix ===

  a   b   <-- classified as
```

Status OK Log x 0

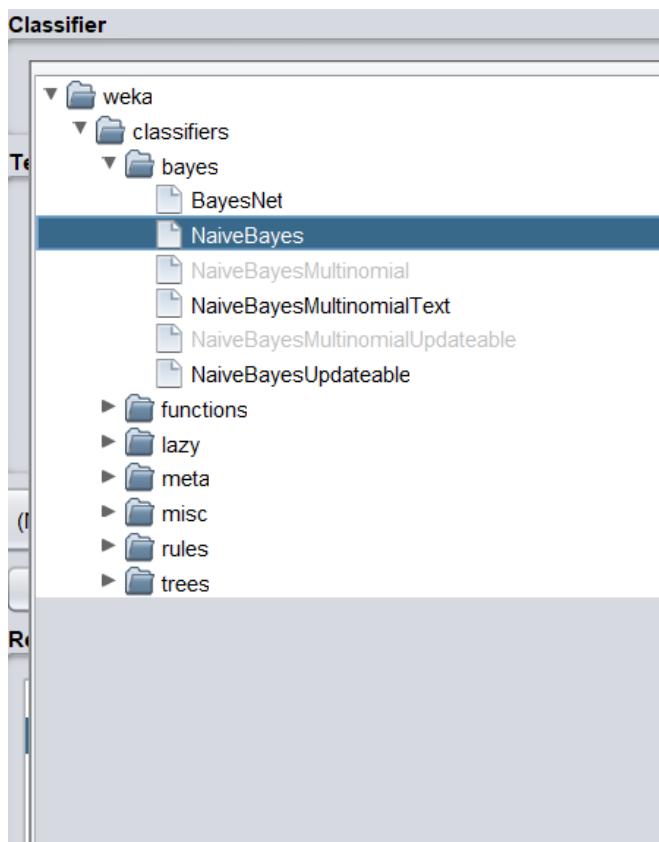
TABLA RESUMEN J48 ordenada de peor a mejor rendimiento. Siendo el mejor rendimiento el último (color verde)

TABLA RESUMEN MODIFICACIONES (J48)		% DE RENDIMIENTO DEL ÁRBOL
cross validation con valores predeterminados		69,469%
cross validation con Folds = 50		70,649%
cross validation sin podar el árbol, hojas con un único ejemplo y Folds = 10		80,2899%
Percentage split 66% sin podar el árbol y que admite hoja con un único ejemplo		81,7021 %
Percentage split 79% sin podar el árbol y que admite hoja con un único ejemplo		82.069 %
Percentage split 66% con valores predeterminados		84,2553%
cross validation sin podar el árbol, hojas con un único ejemplo y Folds = 50		88,1159 %
use training set con valores predeterminados		90,2754%
Percentage split 79% con valores predeterminados		90.3448 %
Use training set sin podar el árbol y que admite hoja con un único ejemplo		99,2754%

NAIVE BAYES

Este es un algoritmo de clasificación. Utiliza una implementación simple del Teorema de Bayes donde la probabilidad previa para cada clase se calcula a partir de los datos de entrenamiento y se asume que son independientes entre sí (técnica llamado condicionalmente independiente). Calcula la probabilidad posterior para cada clase y hace una predicción para la clase con mayor probabilidad. Como tal, admite problemas de clasificación binaria y de clases múltiples.

En choose seleccionamos Naive Bayes , dejamos los valores predeterminados y le damos a star.



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayes

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 10
- Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 20.30.27 - functions.MultilayerPerceptron
- 20.41.54 - functions.Logistic
- 20.58.28 - bayes.NaiveBayes

Classifier output

```
Time taken to build model: 0.02 seconds
==== Stratified cross-validation ====
==== Summary ====

Correctly Classified Instances      536          77.6812 %
Incorrectly Classified Instances   154          22.3188 %
Kappa statistic                   0.534
Mean absolute error               0.2228
Root mean squared error           0.4356
Relative absolute error            45.0979 %
Root relative squared error       87.6429 %
Total Number of Instances         690

==== Detailed Accuracy By Class ====

      TP Rate   FP Rate   Precision   Recall   F-Measure   MCC   ROC Area   PRC Area   Cl
          0,596    0,078    0,859     0,596    0,704     0,557   0,896    0,861    +
          0,922    0,404    0,740     0,922    0,821     0,557   0,896    0,906    -
Weighted Avg.      0,777    0,259    0,793     0,777    0,769     0,557   0,896    0,886

==== Confusion Matrix ====

      a     b   <-- classified as
183 124 |  a = +
30 353 |  b = -
```

Status

OK

Log x 0

En este algoritmo conseguimos una precisión del 77.6812 %. Si cambiamos el número de pliegues a 11 nos sale una precisión del 78.1159 %. Hemos obtenido un 0,4347% más de precisión que en el caso de 10 pliegues.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayes

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 11
- Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 20.30.27 - functions.MultilayerPerceptron
- 20.41.54 - functions.Logistic
- 20.58.28 - bayes.NaiveBayes
- 20.59.40 - bayes.NaiveBayes
- 20.59.45 - bayes.NaiveBayes
- 20.59.52 - bayes.NaiveBayes
- 20.59.54 - bayes.NaiveBayes
- 20.59.57 - bayes.NaiveBayes
- 21.00.05 - bayes.NaiveBayes
- 21.00.08 - bayes.NaiveBayes

Classifier output

```
Time taken to build model: 0 seconds
==== Stratified cross-validation ====
==== Summary ====

Correctly Classified Instances      539          78.1159 %
Incorrectly Classified Instances   151          21.8841 %
Kappa statistic                   0.5429
Mean absolute error               0.2216
Root mean squared error           0.4338
Relative absolute error            44.8576 %
Root relative squared error       87.2885 %
Total Number of Instances         690

==== Detailed Accuracy By Class ====

      TP Rate   FP Rate   Precision   Recall   F-Measure   MCC   ROC Area   PRC Area   Cl
          0,599    0,073    0,868     0,599    0,709     0,567   0,895    0,862    +
          0,927    0,401    0,743     0,927    0,825     0,567   0,895    0,906    -
Weighted Avg.      0,781    0,255    0,798     0,781    0,773     0,567   0,895    0,887

==== Confusion Matrix ====

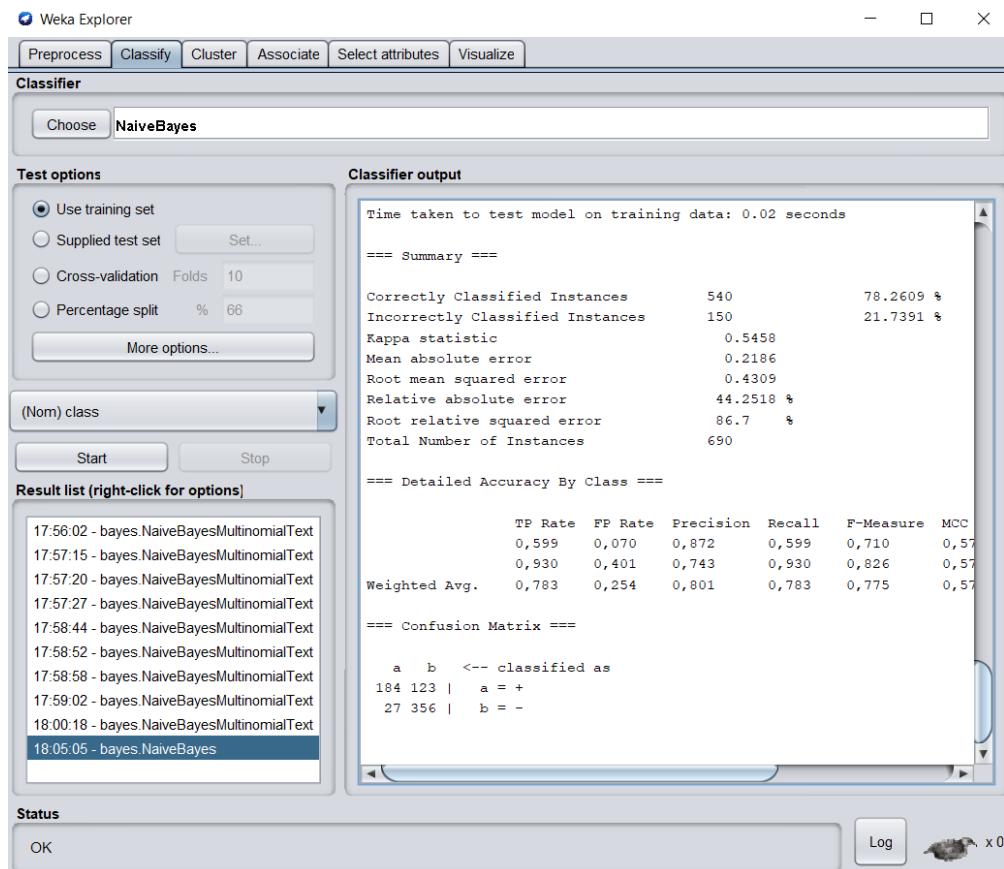
      a     b   <-- classified as
184 123 |  a = +
28 355 |  b = -
```

Status

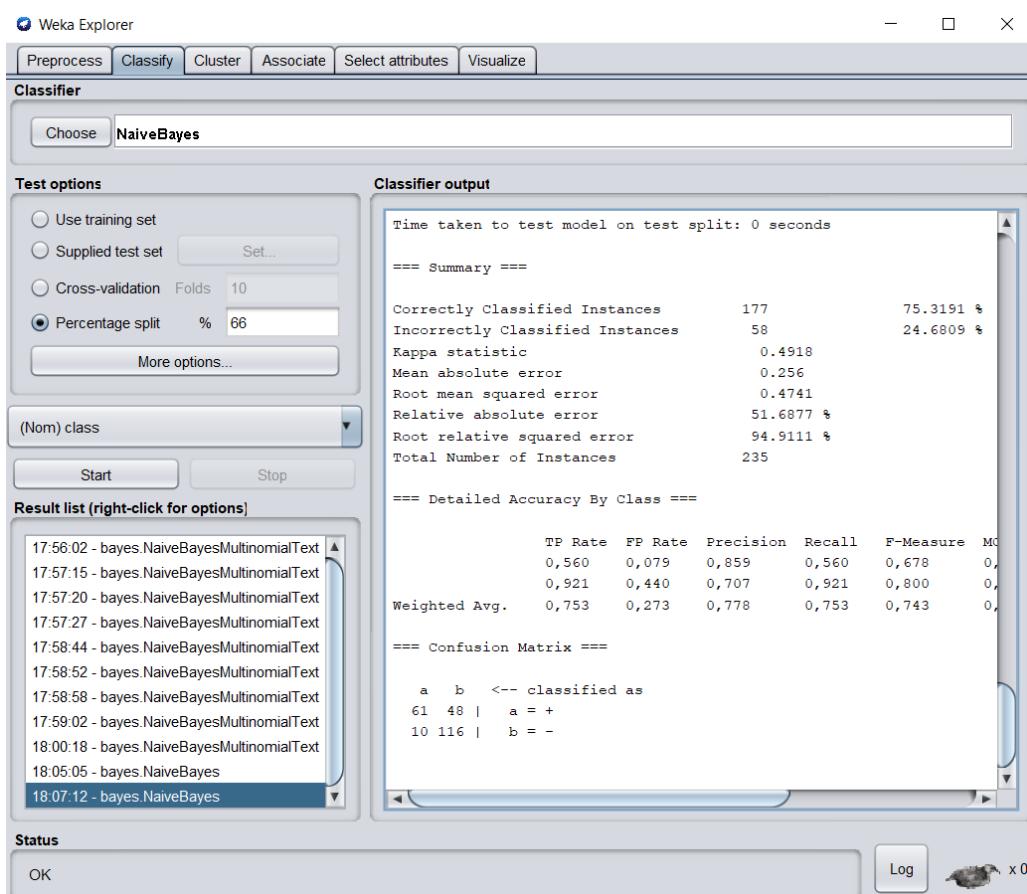
OK

Log x 0

Para mejorar este rendimiento en Test options seleccionamos Use training set y vemos como mejora con 540 instancias correctamente clasificadas con un 78.2609%



Miremos ahora lo que sucede si en Test options seleccionamos Percentage Split y dejamos los valores predeterminados. El rendimiento baja a un 75.3191 %



¿Qué sucede si cambiamos el % del Percentage Split?. Si cambiamos de un 66% a un 85% el rendimiento mejora bastante teniendo ahora un 80.5825 %

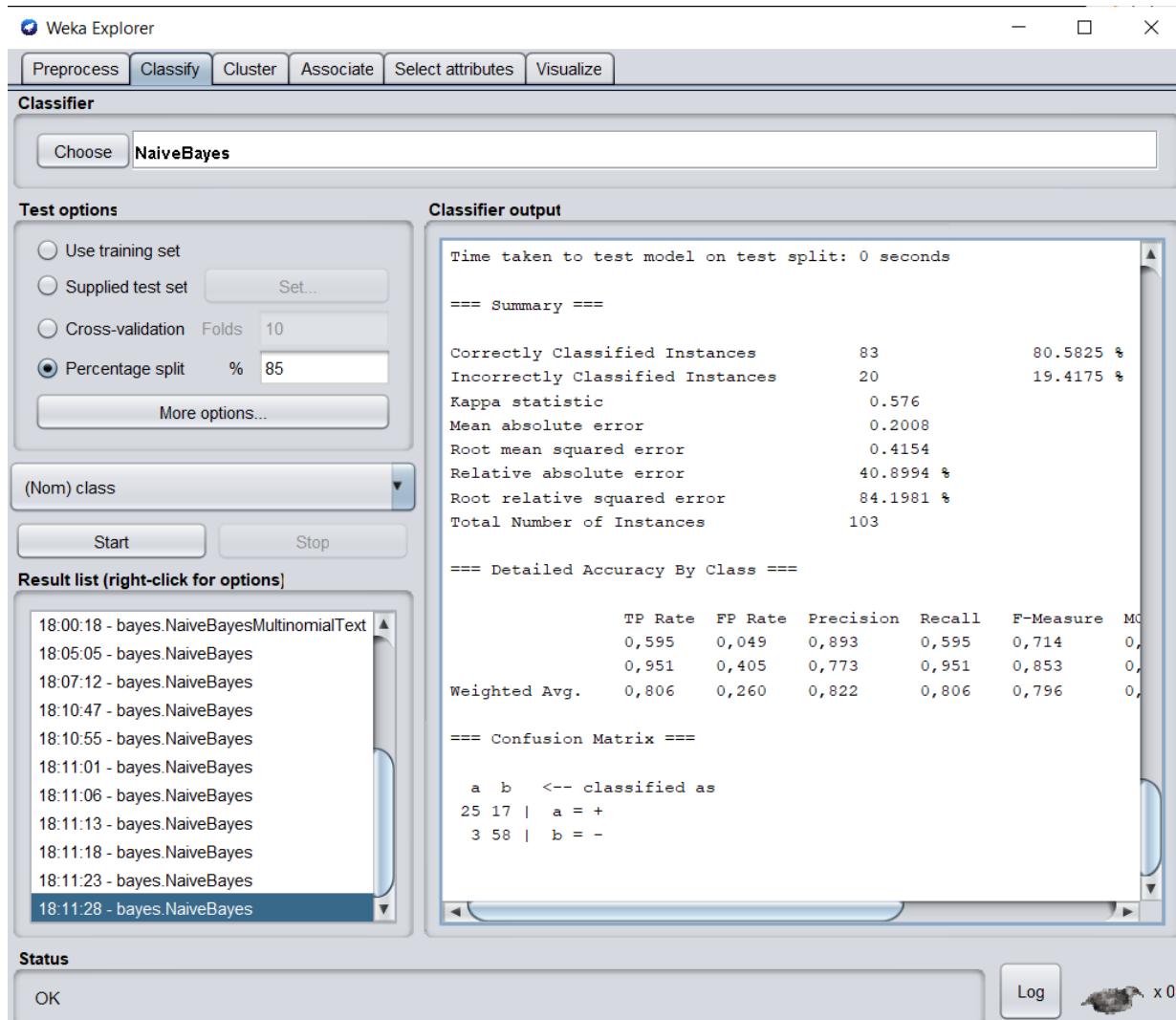


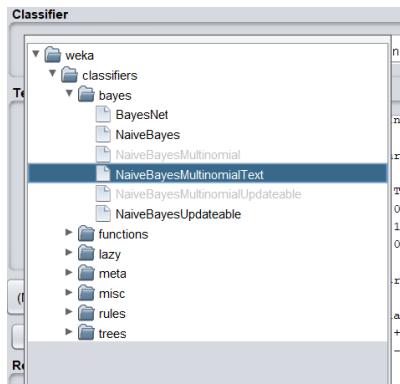
TABLA RESUMEN NAIVE BAYES ordenada de peor a mejor rendimiento. Siendo el mejor rendimiento el último (color verde)

TABLA RESUMEN MODIFICACIONES (NAIVE BAYES)		% DE RENDIMIENTO DEL ÁRBOL
Percentage Split con valor predeterminado		75.3191 %
Cross-validation con valor predeterminado		77.6812 %.
Cross-validation Folds= 11		78.1159 %.
Use training set		78.2609 %
Percentage Split con 85%		80.5825 %

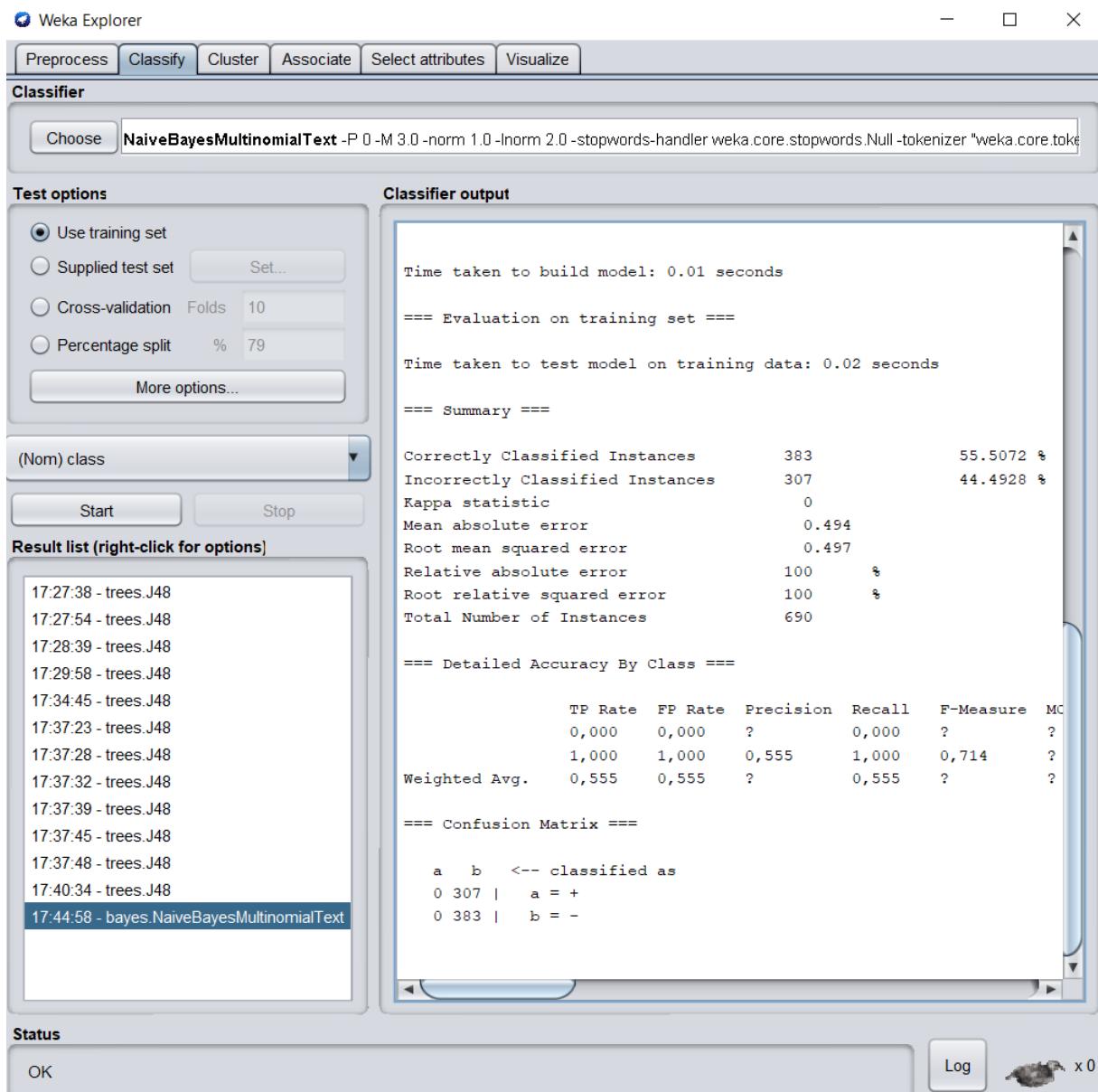
NAIVE BAYES MULTINOMINAL TEXT

Naive Bayes Multinomial Text opera directamente (y solo) en atributos de cadena (String). Se aceptan otros tipos de atributos de entrada, pero se ignoran durante el entrenamiento y la clasificación.

Para seleccionarlo le damos a Choose en Classifier > Bayes > NaiveBayesMultinomialText

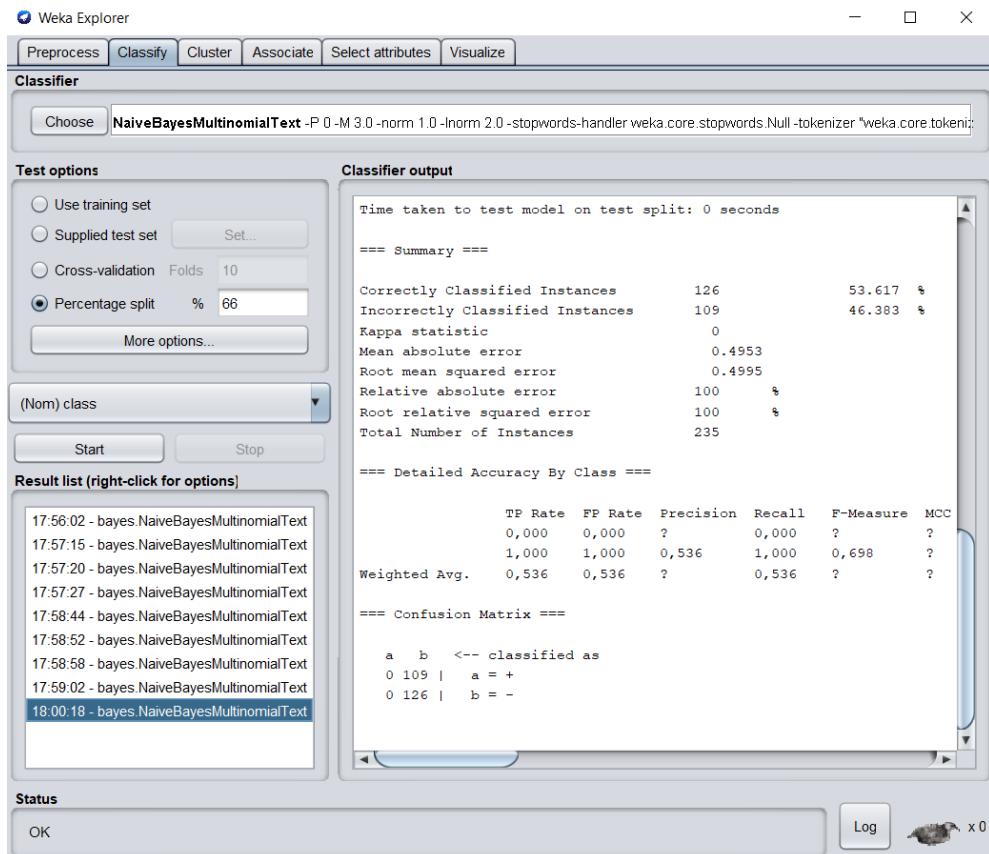


Seleccionamos Use training set y le damos a star.



Obtenemos un rendimiento 55.5072% ya que 383 instancias han sido clasificadas correctamente frente a 307 clasificadas incorrectamente. El peor rendimiento de todos hechos hasta ahora.

Ya que con cross-validation el rendimiento sale igual que el anterior si seleccionamos percentage Split y dejamos el valor predeterminado el rendimiento sube un poco a 53.617 %



Si ponemos percentage Split al 77% es aún mejor el rendimiento ya que aumenta al 57,2412%

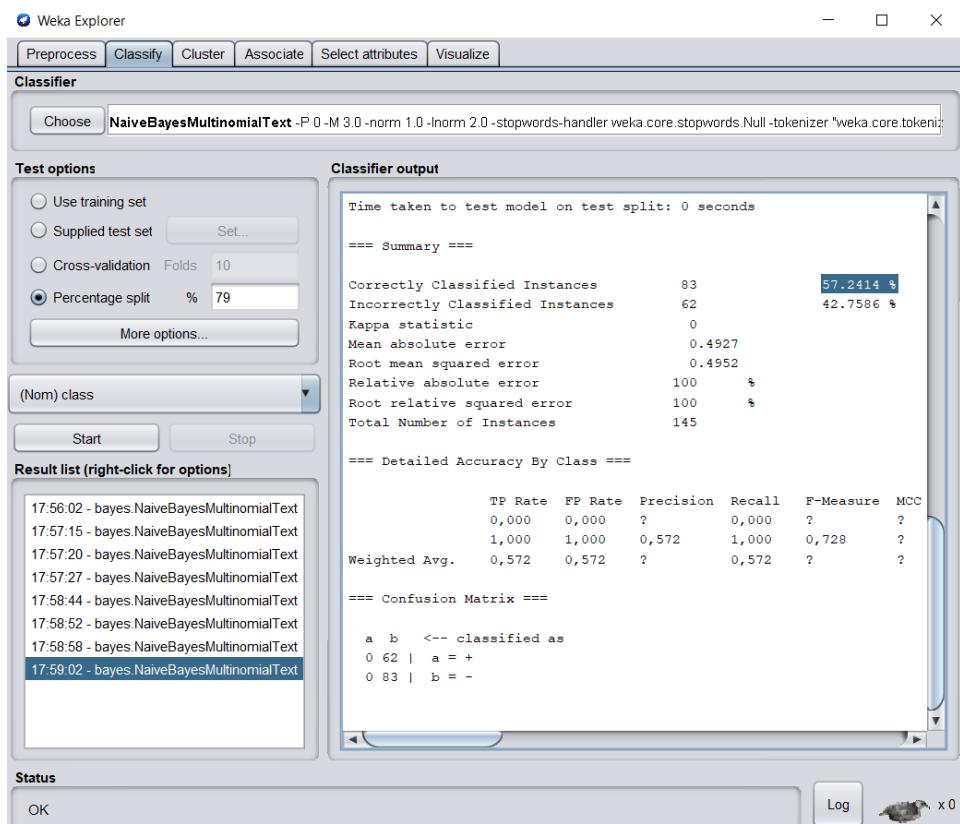


TABLA RESUMEN NAIVE BAYES MULTINOMINAL TEXT ordenada de peor a mejor rendimiento. Siendo el mejor rendimiento el último (color verde)

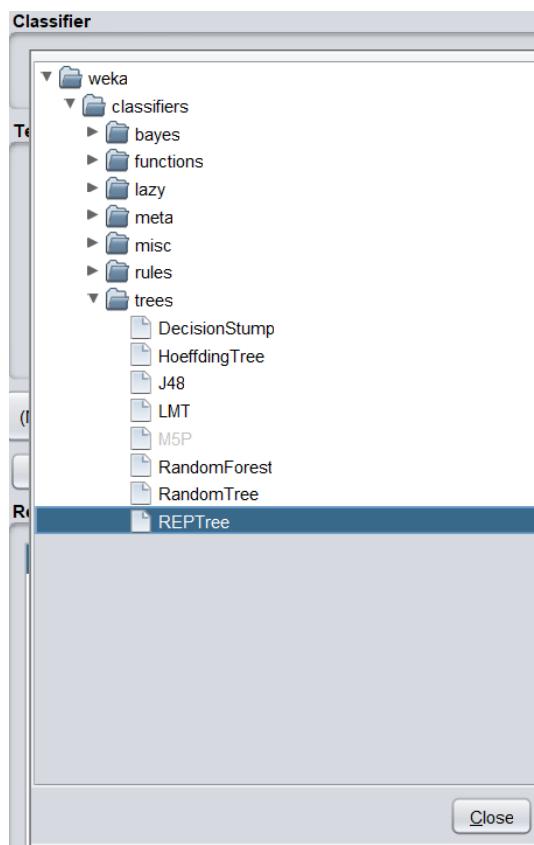
TABLA RESUMEN MODIFICACIONES (NAIVE BAYES MULTINOMINAL TEXT)		% DE RENDIMIENTO DEL ÁRBOL
Use training set		55.5072%
Percentage Split con valores predeterminados		53.617%
Percentage Split 79%		57.2414 %

REPTree (DECISION TREE)

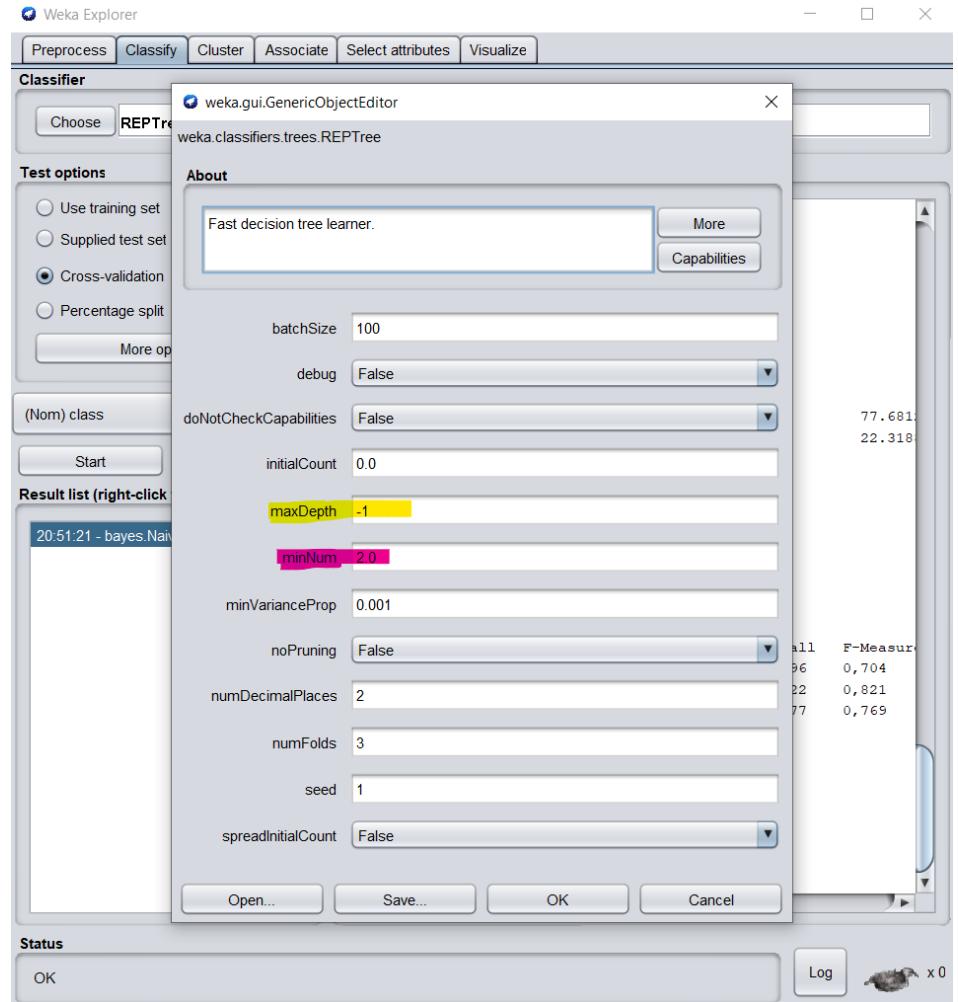
Los árboles de decisión se conocen como árboles de clasificación y regresión (CART). Trabajan creando un árbol para evaluar una instancia de datos, comienzan en la raíz del árbol y mueven hacia las hojas (raíces) hasta que se pueda hacer una predicción. El proceso de creación de un árbol de decisiones funciona seleccionando con avidez el mejor punto de división para hacer predicciones y repitiendo el proceso hasta que el árbol tenga una profundidad fija.

Una vez construido el árbol, se poda para mejorar la capacidad del modelo de generalizar a nuevos datos.

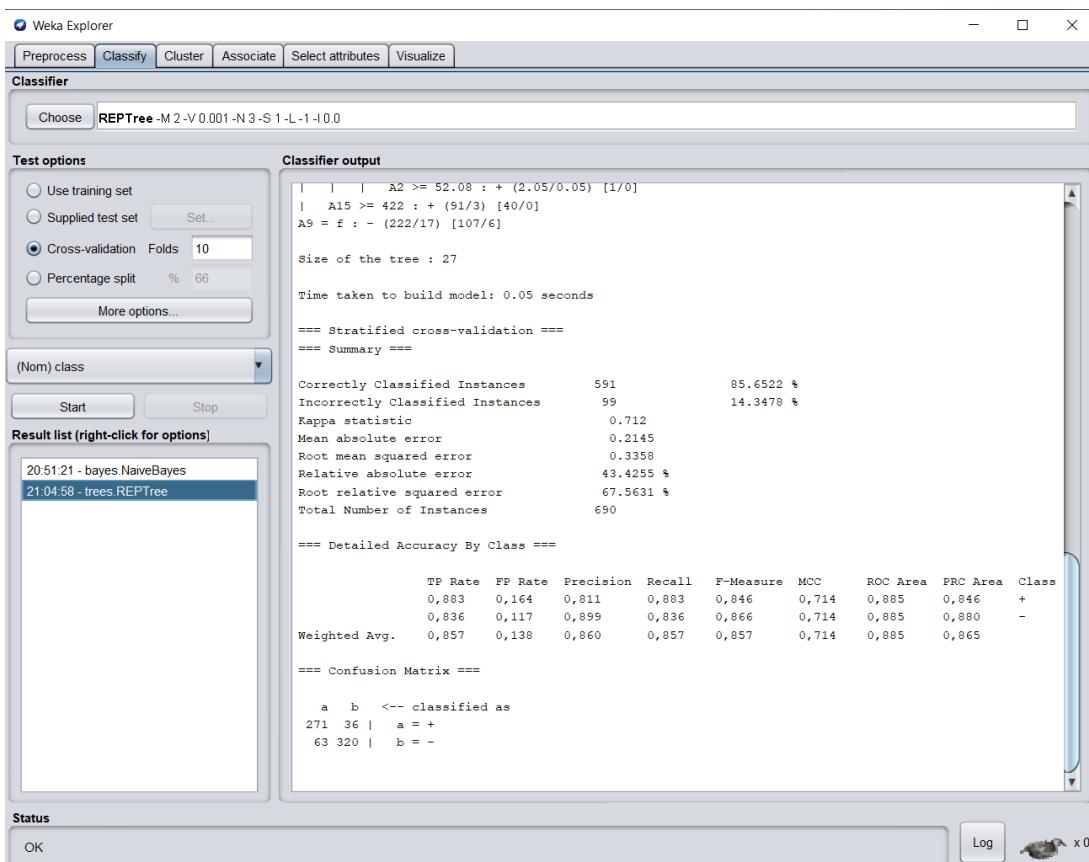
Para hacerlo le damos a **Choose > weka > classifiers > trees > REPTree**



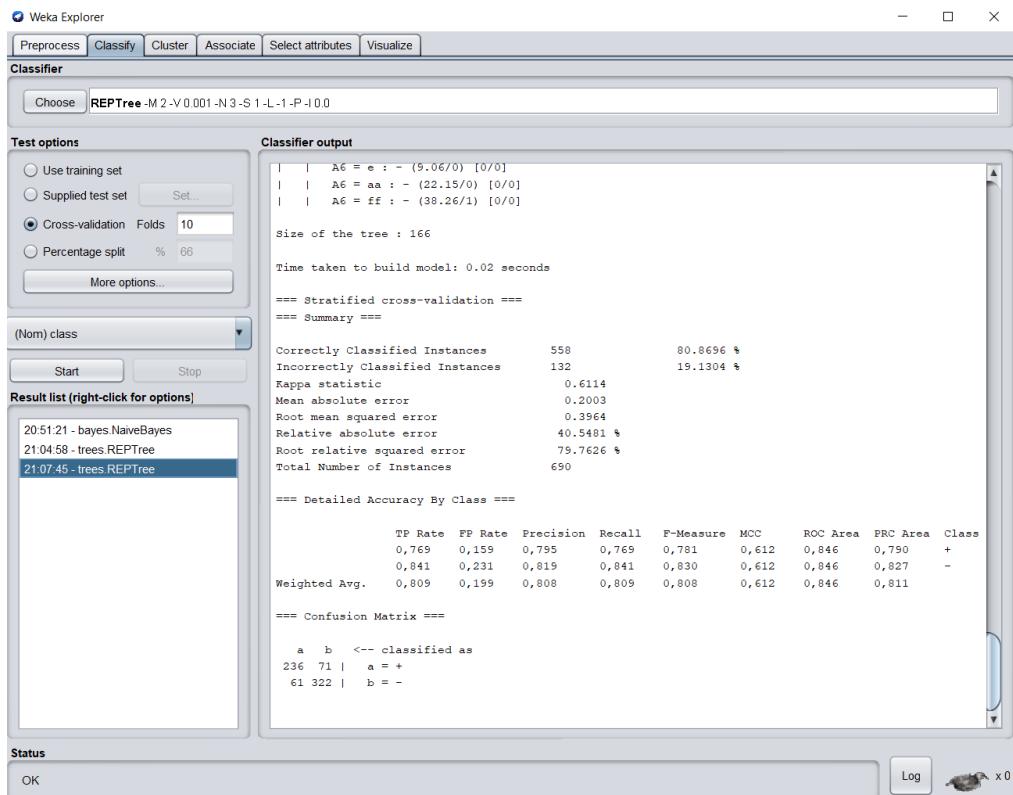
Si le damos al algoritmo nos sale la configuración de este. La profundidad del árbol podemos especificarla en **maxDepth** (subrayado en amarillo) aunque se define automáticamente. Para definir el número mínimo de instancias admitidas por el árbol en un nodo hoja al construir el árbol a partir de los datos de entrenamiento con el parámetro **minNum** (subrayado en rosa)



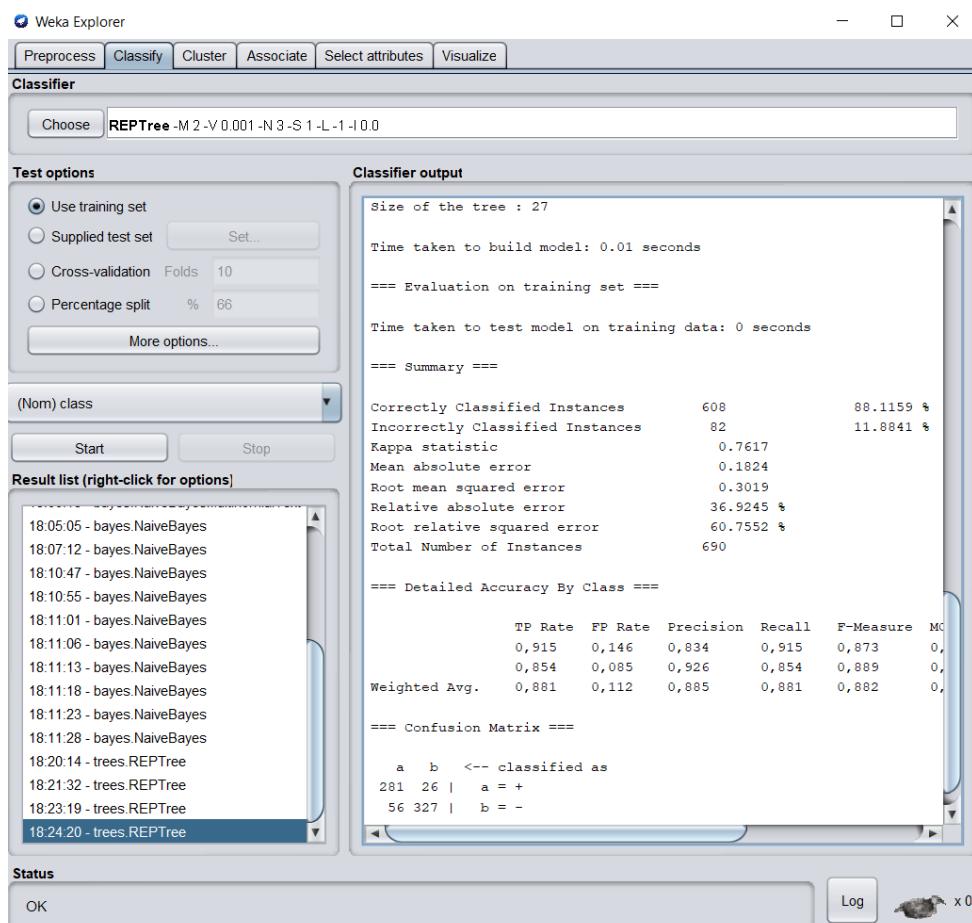
Al darle a star vemos que nos sale una precisión de 85.6522%.



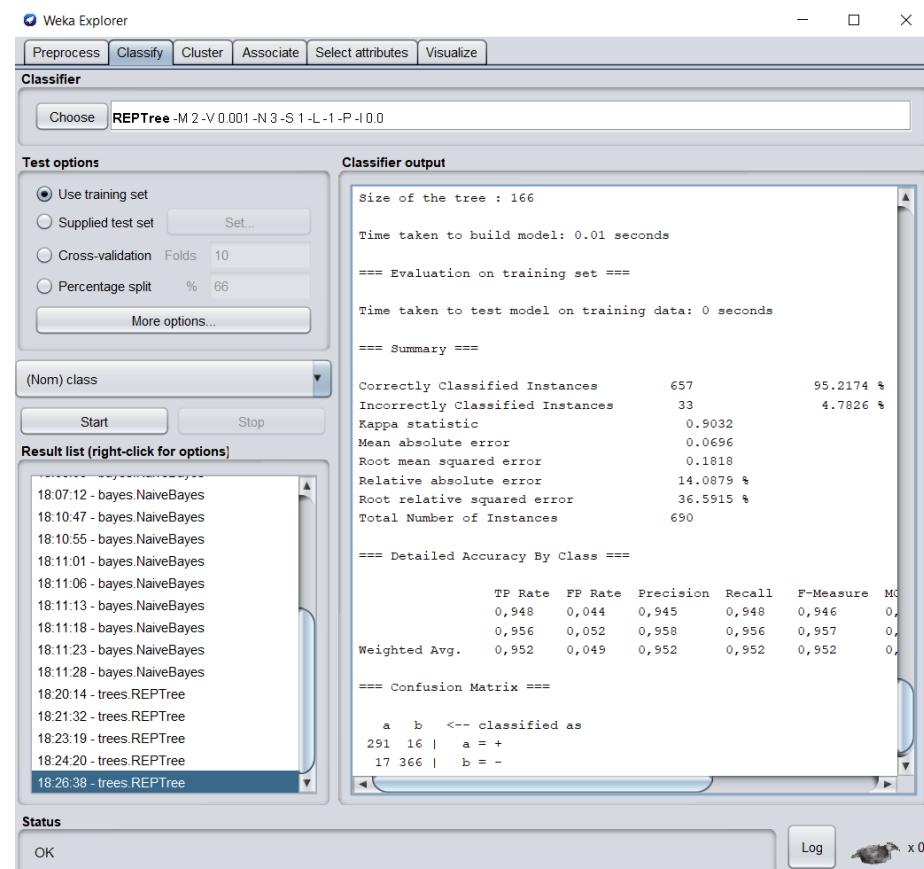
Ahora vamos a hacer otra prueba, en noPruning se puede desactivar la poda al ponerlo en True. Y vemos como al no hacer la poda nos sale un peor rendimiento 80.8696 % frente a un 85.6522%. con poda.



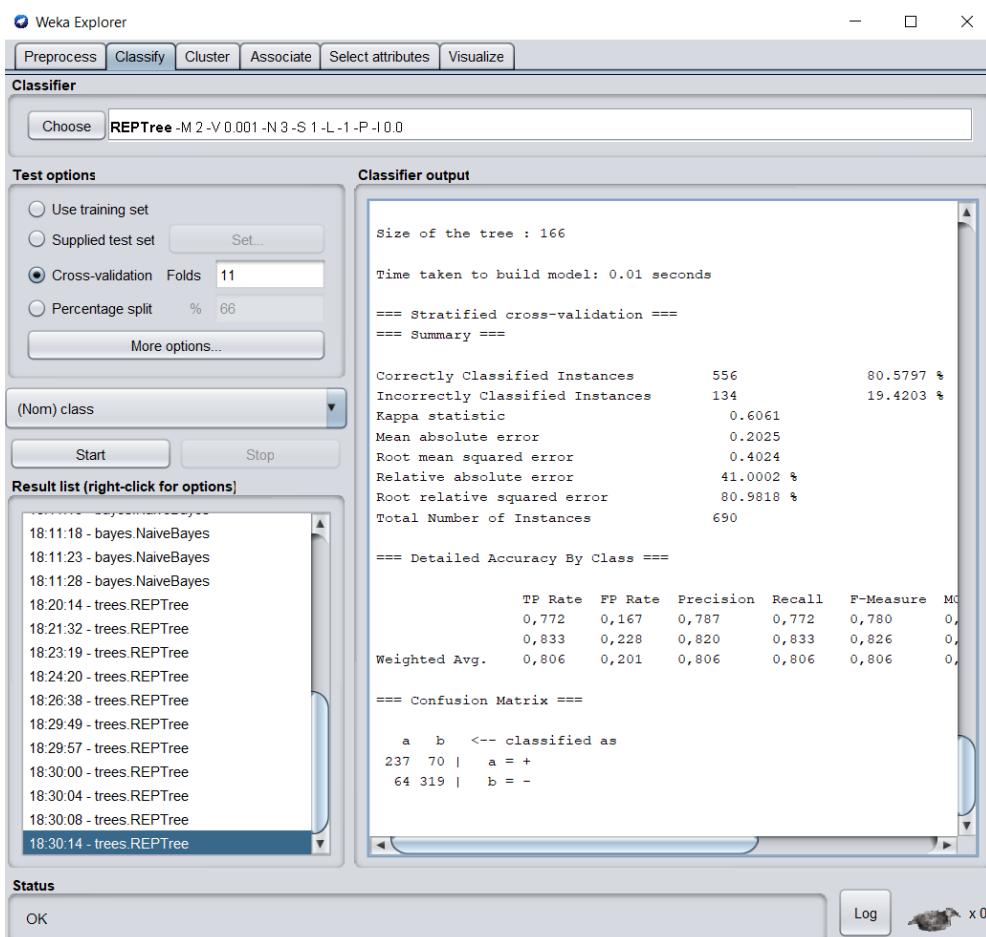
Para mejorar el rendimiento en test options seleccionamos use training set y dejamos los valores predeterminados. Así obtenemos un 88,1159% de rendimiento . El árbol está formado por 27 nodos.



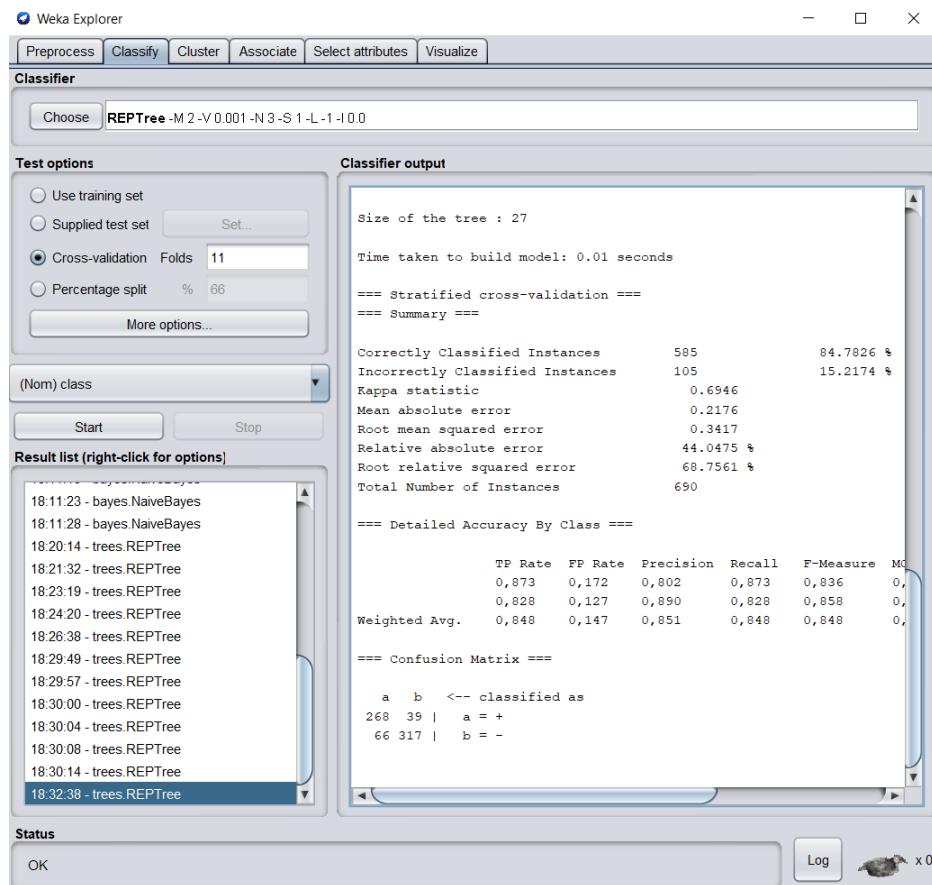
Ahora vamos a dejarlo como el anterior pero hacemos clic en el filtro y ponemos a True noPruning para no hacer la poda del árbol y el rendimiento aumenta bastante llegando a un 95.2174 %. Y con 166 nodos.



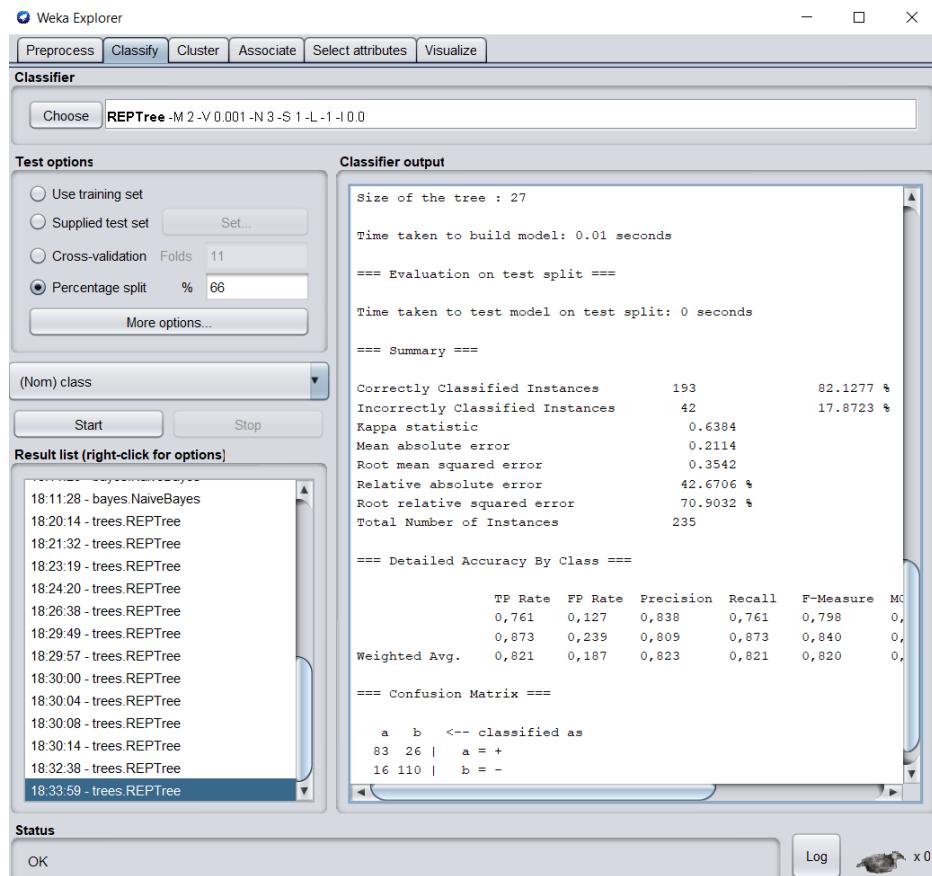
Ahora vamos a cambiar el Folds a 11 de Cross-validation para ver si mejora o no el rendimiento.



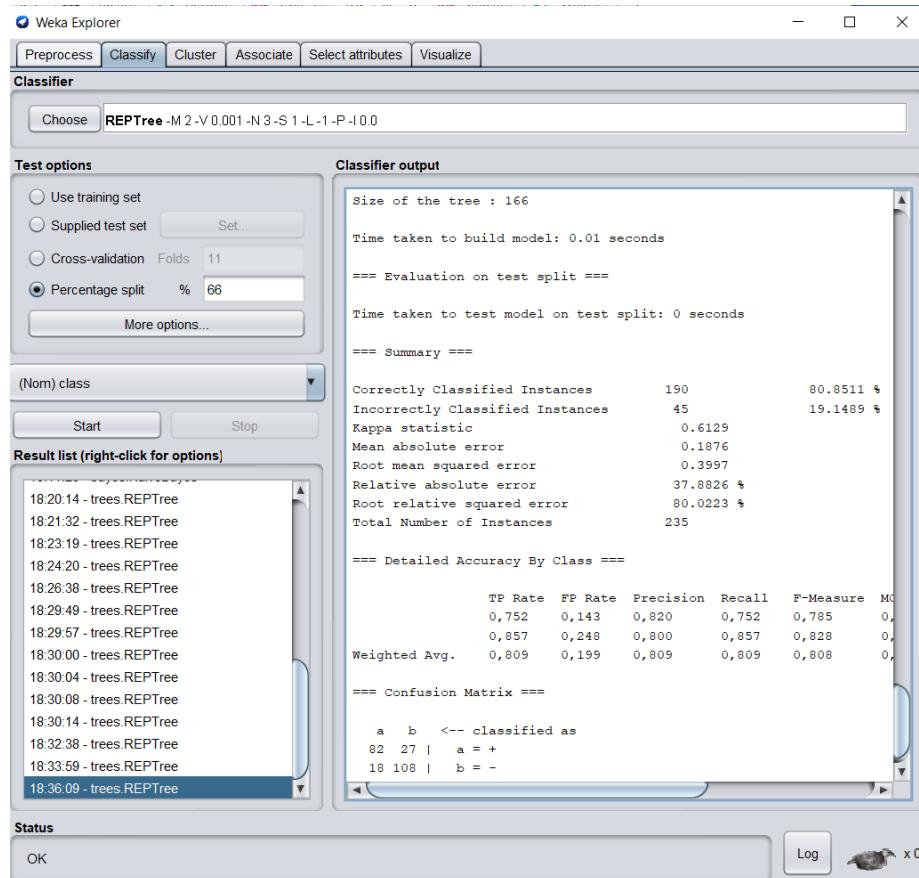
Como podemos ver no mejora, es más es el peor rendimiento hasta ahora. (80.5797 %) frente a un 84.7826 % si con este ejemplo mismo hacemos poda. Formado por 27 nodos el árbol.



En cuanto a la sección test options en Percentage Split si dejamos los valores predeterminados obtenemos un 82,1277% de rendimiento con 27 nodos formando el árbol.

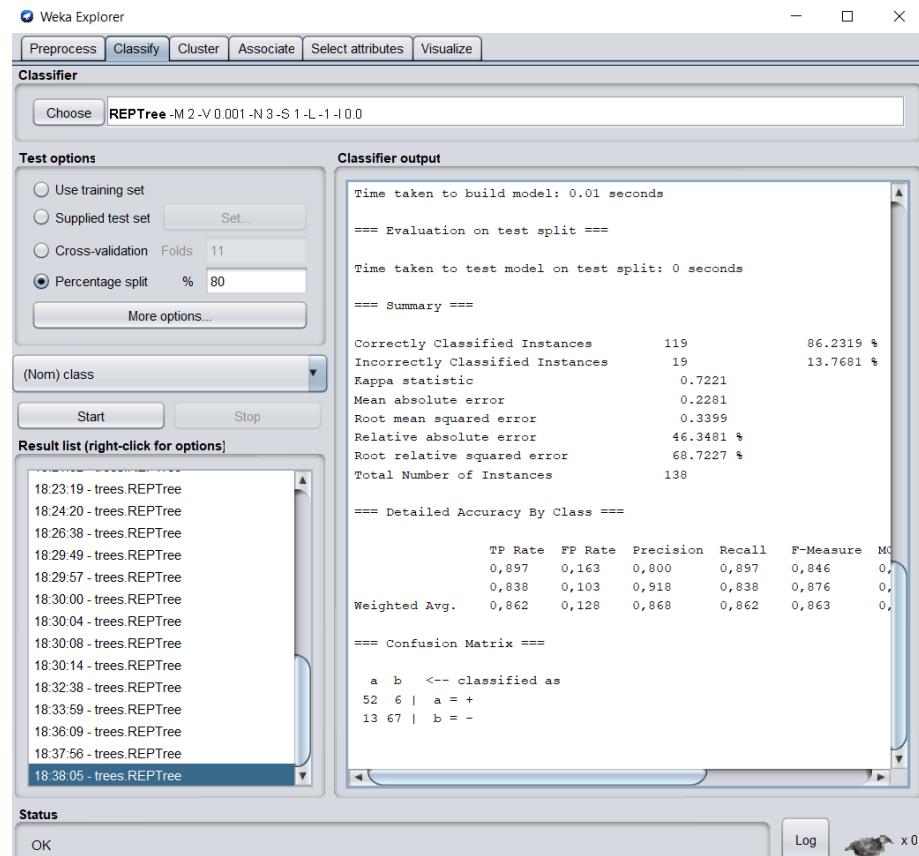


Y si hacemos la poda a este empeora obteniendo un rendimiento del 80,8511%



¿Y si cambiamos el % del Percentage Split que sucede?

Si lo cambiamos de un 66% a un 80% el rendimiento mejora con creces a un 86.2319 % frente a un 84.058 % si no podamos el árbol (noPruning = True) . Lo que obtenemos peor rendimiento.



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -P -I 0.0

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 11
- Percentage split % 80

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 18:24:20 - trees.REPTree
- 18:26:38 - trees.REPTree
- 18:29:49 - trees.REPTree
- 18:29:57 - trees.REPTree
- 18:30:00 - trees.REPTree
- 18:30:04 - trees.REPTree
- 18:30:08 - trees.REPTree
- 18:30:14 - trees.REPTree
- 18:32:38 - trees.REPTree
- 18:33:59 - trees.REPTree
- 18:36:09 - trees.REPTree
- 18:37:56 - trees.REPTree
- 18:38:05 - trees.REPTree
- 18:39:44 - trees.REPTree

Classifier output

```
Time taken to build model: 0.01 seconds
==== Evaluation on test split ====
Time taken to test model on test split: 0 seconds
==== Summary ====
Correctly Classified Instances          116           84.058 %
Incorrectly Classified Instances        22            15.942 %
Kappa statistic                         0.6697
Mean absolute error                     0.1768
Root mean squared error                 0.3568
Relative absolute error                  35.9181 %
Root relative squared error             72.152 %
Total Number of Instances                138
==== Detailed Accuracy By Class ====
      TP Rate   FP Rate   Precision   Recall   F-Measure   MCC
      0,776     0,113     0,833      0,776    0,804      0,
      0,888     0,224     0,845      0,888    0,866      0,
Weighted Avg.      0,841     0,177     0,840      0,841    0,840      0,
==== Confusion Matrix ====
      a   b   <-- classified as
45  13 |  a = +
  9  71 |  b = -
```

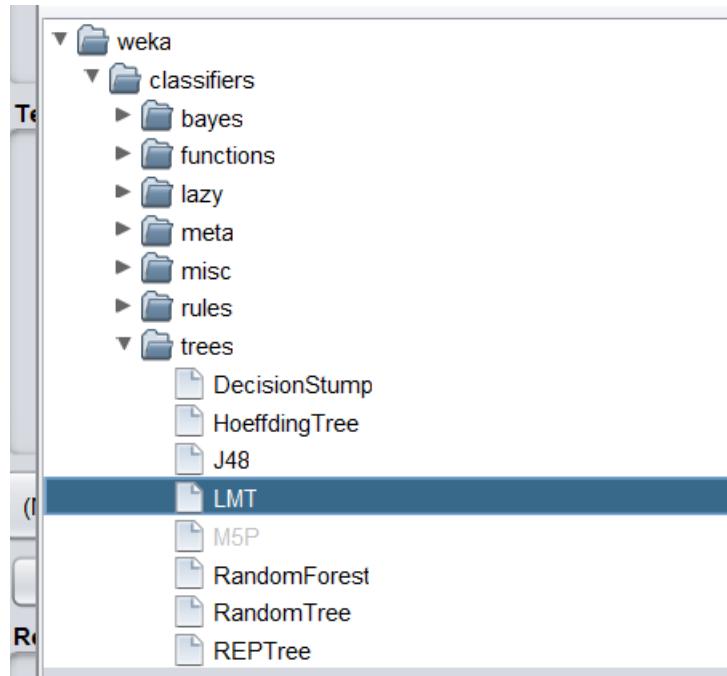
Status OK Log x 0

TABLA RESUMEN MODIFICACIONES (REPTree)		% RENDIMIENTO DEL ÁRBOL
Cross-validation, Folds = 11		80,5797%
Percentage Split , 66%		80.8511 %
Cross-validation , Folds = 10 y no podar el árbol		80.8696 %
Percentage Split con valores predeterminados		82.1277 %
Percentage Split , 66% y sin podar el árbol		84.058 %
Cross-validation, Folds = 11 y no podar el árbol		84.7826 %
Cross-validation con valores predeterminados		85.6522%.
Percentage Split 80%		86.2319 %
Use training set con valores predeterminados		88,1159%
Use training set sin podar el árbol		95.2174 %

LMT (LOGISTIC MODEL TREES)

Este filtro lo que hace es combinar los modelos de regresión logística con árboles de inducción. Es decir, es un modelo analógico de árboles para problemas de clasificación.

Los pasos son los siguientes: Choose > classifiers > trees > LMT



Al utilizar use training set obtenemos un rendimiento el 86.2319% es decir 595 instancias correctamente clasificadas frente a 95 mal clasificadas

A screenshot of the Weka Classifier output window. The 'Choose' button is set to 'LMT -I-1-M 15 -W 0.0'. The 'Test options' panel shows 'Use training set' selected. The 'Classifier output' panel displays the following evaluation results:

```
==== Evaluation on training set ====
Time taken to test model on training data: 0 seconds
==== Summary ====
Correctly Classified Instances      595          86.2319 %
Incorrectly Classified Instances   95           13.7681 %
Kappa statistic                   0.7247
Mean absolute error               0.2029
Root mean squared error          0.3097
Relative absolute error          41.0821 %
Root relative squared error     62.3127 %
Total Number of Instances        690

==== Detailed Accuracy By Class ====

          TP Rate  FP Rate  Precision  Recall   F-Measure  MCC    ROC Area  PRC Area  Cl
          0,909   0,175    0,806    0,909    0,855    0,729   0,936   0,929   +
          0,825   0,091    0,919    0,825    0,869    0,729   0,936   0,936   -
Weighted Avg.   0,862   0,128    0,869    0,862    0,863    0,729   0,936   0,933   -
```

The 'Result list' panel shows the history of runs: '13:59:02 - trees.LMT', '14:02:22 - trees.LMT', and '14:04:48 - trees.LMT' (which is selected).

Si utilizamos validación cruzada (cross-validation) y dejamos el Folds a 10, es decir dividimos en 10 partes el conjunto de datos. El rendimiento es menor en un 84.7826 %, 585 instancias bien clasificadas.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose LMT -I-1-M 15 -W 0.0

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 10
- Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 13:59:02 - trees.LMT
- 14:02:22 - trees.LMT
- 14:04:48 - trees.LMT
- 14:14:54 - trees.LMT
- 14:17:00 - trees.LMT

Classifier output

```
Time taken to build model: 0.42 seconds

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

Correctly Classified Instances      585          84.7826 %
Incorrectly Classified Instances   105          15.2174 %
Kappa statistic                      0.6948
Mean absolute error                  0.2073
Root mean squared error              0.3241
Relative absolute error              41.959 %
Root relative squared error         65.2199 %
Total Number of Instances           690

== Detailed Accuracy By Class ==

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area Cl
0,876   0,175   0,801    0,876   0,837    0,697   0,920   0,903   +
0,825   0,124   0,893    0,825   0,858    0,697   0,920   0,917   -
Weighted Avg.  0,848   0,147   0,852    0,848   0,848    0,697   0,920   0,911

== Confusion Matrix ==

a   b   <- classified as
269 38 |  a = +
67 316 |  b = -
```

Status

OK Log x 0

Aunque cambie el Folds no varía mucho la cantidad de instancias correctamente clasificadas. Por ejemplo si el Folds es 19 el rendimiento es del 84.9275 % de la misma forma si lo cambio a 69 el rendimiento es de 85.0725 %.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose LMT -I-1-M 15 -W 0.0

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 19
- Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 14:14:54 - trees.LMT
- 14:17:00 - trees.LMT
- 14:19:58 - trees.LMT
- 14:20:13 - trees.LMT
- 14:20:24 - trees.LMT
- 14:20:27 - trees.LMT
- 14:20:30 - trees.LMT
- 14:20:33 - trees.LMT
- 14:20:40 - trees.LMT
- 14:20:51 - trees.LMT
- 14:21:45 - trees.LMT
- 14:22:31 - trees.LMT
- 14:23:57 - trees.LMT

Classifier output

```
Time taken to build model: 0.31 seconds

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

Correctly Classified Instances      586          84.9275 %
Incorrectly Classified Instances   104          15.0725 %
Kappa statistic                      0.6976
Mean absolute error                  0.2043
Root mean squared error              0.3253
Relative absolute error              41.3613 %
Root relative squared error         65.4609 %
Total Number of Instances           690

== Detailed Accuracy By Class ==

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area Cl
0,876   0,172   0,803    0,876   0,838    0,700   0,919   0,901   +
0,828   0,124   0,893    0,828   0,859    0,700   0,919   0,919   -
Weighted Avg.  0,849   0,145   0,853    0,849   0,850    0,700   0,919   0,911

== Confusion Matrix ==

a   b   <- classified as
269 38 |  a = +
66 317 |  b = -
```

Status

OK Log x 0

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose LMT -I-1 -M 15 -VV 0.0

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 69
- Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 14:04:48 - trees.LMT
- 14:14:54 - trees.LMT
- 14:17:00 - trees.LMT
- 14:19:58 - trees.LMT
- 14:20:13 - trees.LMT
- 14:20:24 - trees.LMT
- 14:20:27 - trees.LMT
- 14:20:30 - trees.LMT
- 14:20:33 - trees.LMT
- 14:20:40 - trees.LMT
- 14:20:51 - trees.LMT
- 14:21:45 - trees.LMT
- 14:22:31 - trees.LMT

Classifier output

```

Time taken to build model: 0.35 seconds

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

Correctly Classified Instances      587      85.0725 %
Incorrectly Classified Instances   103      14.9275 %
Kappa statistic                   0.7008
Mean absolute error               0.2035
Root mean squared error           0.3228
Relative absolute error            41.1947 %
Root relative squared error       64.941 %
Total Number of Instances         690

==== Detailed Accuracy By Class ====


|               | TP Rate | FP Rate | Precision | Recall | F-Measure | MCC   | ROC Area | PRC Area | Class |
|---------------|---------|---------|-----------|--------|-----------|-------|----------|----------|-------|
| 0,883         | 0,175   | 0,802   | 0,883     | 0,840  | 0,704     | 0,921 | 0,903    | +        |       |
| 0,825         | 0,117   | 0,898   | 0,825     | 0,860  | 0,704     | 0,921 | 0,923    | -        |       |
| Weighted Avg. | 0,851   | 0,143   | 0,855     | 0,851  | 0,851     | 0,704 | 0,921    | 0,914    |       |


==== Confusion Matrix ====


|   |     | <-- classified as |       |
|---|-----|-------------------|-------|
|   |     | a                 | b     |
| a | 271 | 36                | a = + |
|   | 67  | 316               | b = - |


```

Status

OK Log x 0

El % de rendimiento sigue sin variar si seleccionamos Percentage split con el % predeterminado dando un 83.4043 % de instancias correctamente clasificadas.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose LMT -I-1 -M 15 -VV 0.0

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 19
- Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 14:17:00 - trees.LMT
- 14:19:58 - trees.LMT
- 14:20:13 - trees.LMT
- 14:20:24 - trees.LMT
- 14:20:27 - trees.LMT
- 14:20:30 - trees.LMT
- 14:20:33 - trees.LMT
- 14:20:40 - trees.LMT
- 14:20:51 - trees.LMT
- 14:21:45 - trees.LMT
- 14:22:31 - trees.LMT
- 14:23:57 - trees.LMT
- 14:24:23 - trees.LMT

Classifier output

```

==== Evaluation on test split ====
Time taken to test model on test split: 0 seconds

==== Summary ====

Correctly Classified Instances      196      83.4043 %
Incorrectly Classified Instances   39      16.5957 %
Kappa statistic                   0.6653
Mean absolute error               0.198
Root mean squared error           0.3243
Relative absolute error            39.9774 %
Root relative squared error       64.932 %
Total Number of Instances         235

==== Detailed Accuracy By Class ====


|               | TP Rate | FP Rate | Precision | Recall | F-Measure | MCC   | ROC Area | PRC Area | Class |
|---------------|---------|---------|-----------|--------|-----------|-------|----------|----------|-------|
| 0,798         | 0,135   | 0,837   | 0,798     | 0,817  | 0,666     | 0,920 | 0,916    | +        |       |
| 0,865         | 0,202   | 0,832   | 0,865     | 0,848  | 0,666     | 0,920 | 0,920    | -        |       |
| Weighted Avg. | 0,834   | 0,171   | 0,834     | 0,834  | 0,834     | 0,666 | 0,920    | 0,918    |       |


==== Confusion Matrix ====


|   |    | <-- classified as |       |
|---|----|-------------------|-------|
|   |    | a                 | b     |
| a | 87 | 22                | a = + |
|   | 17 | 109               | b = - |


```

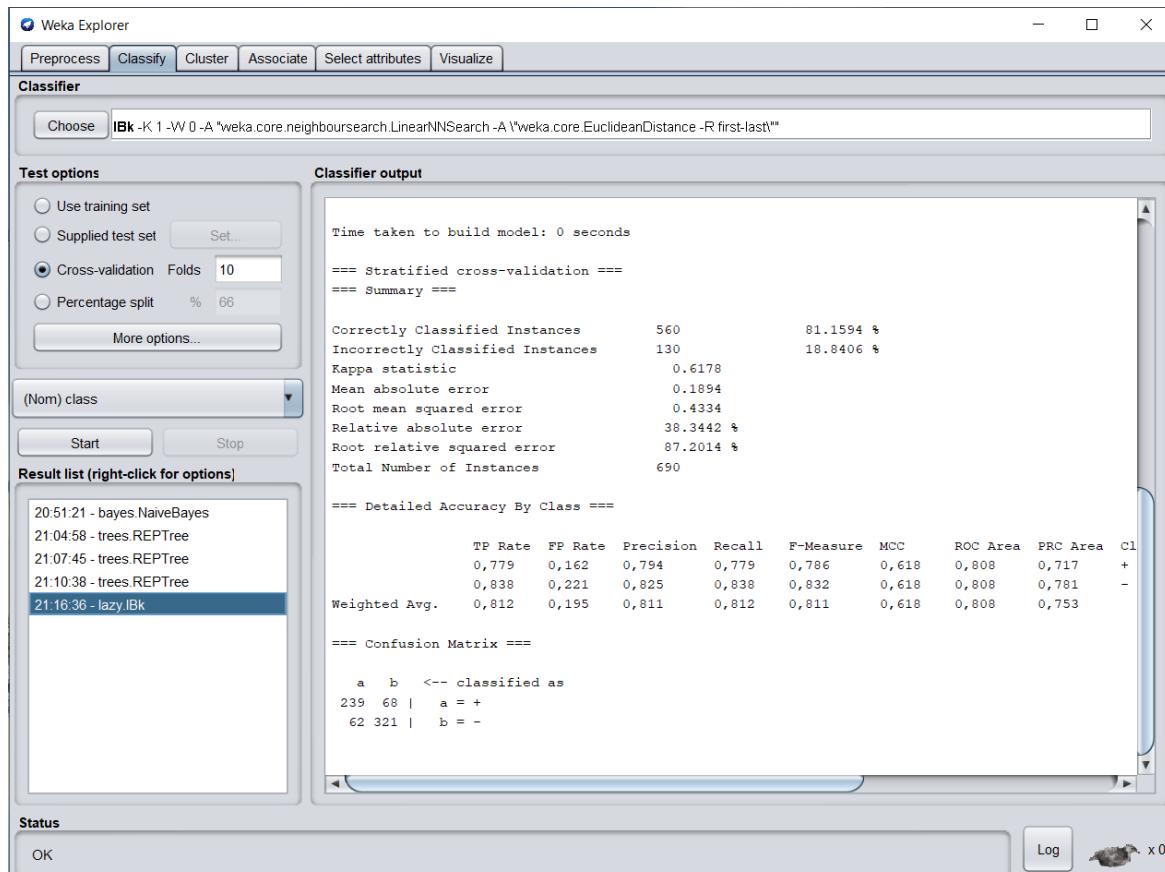
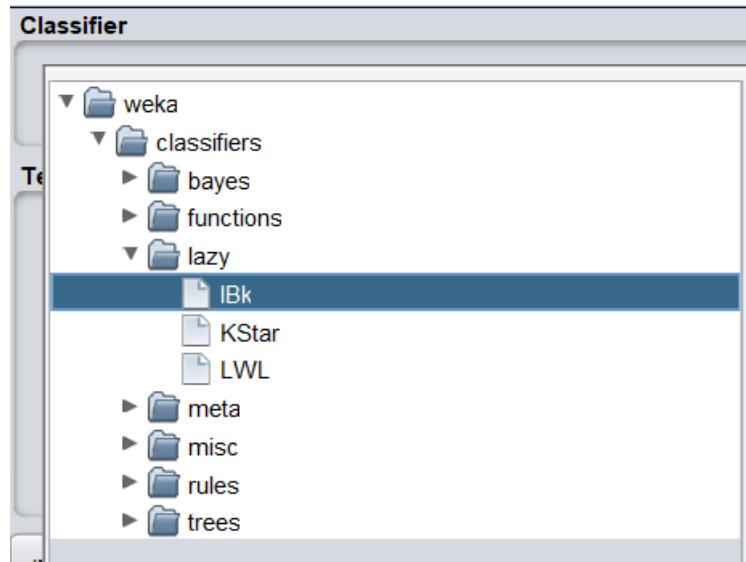
Status

OK Log x 0

IBk (K-VECINOS MÁS CERCANOS)

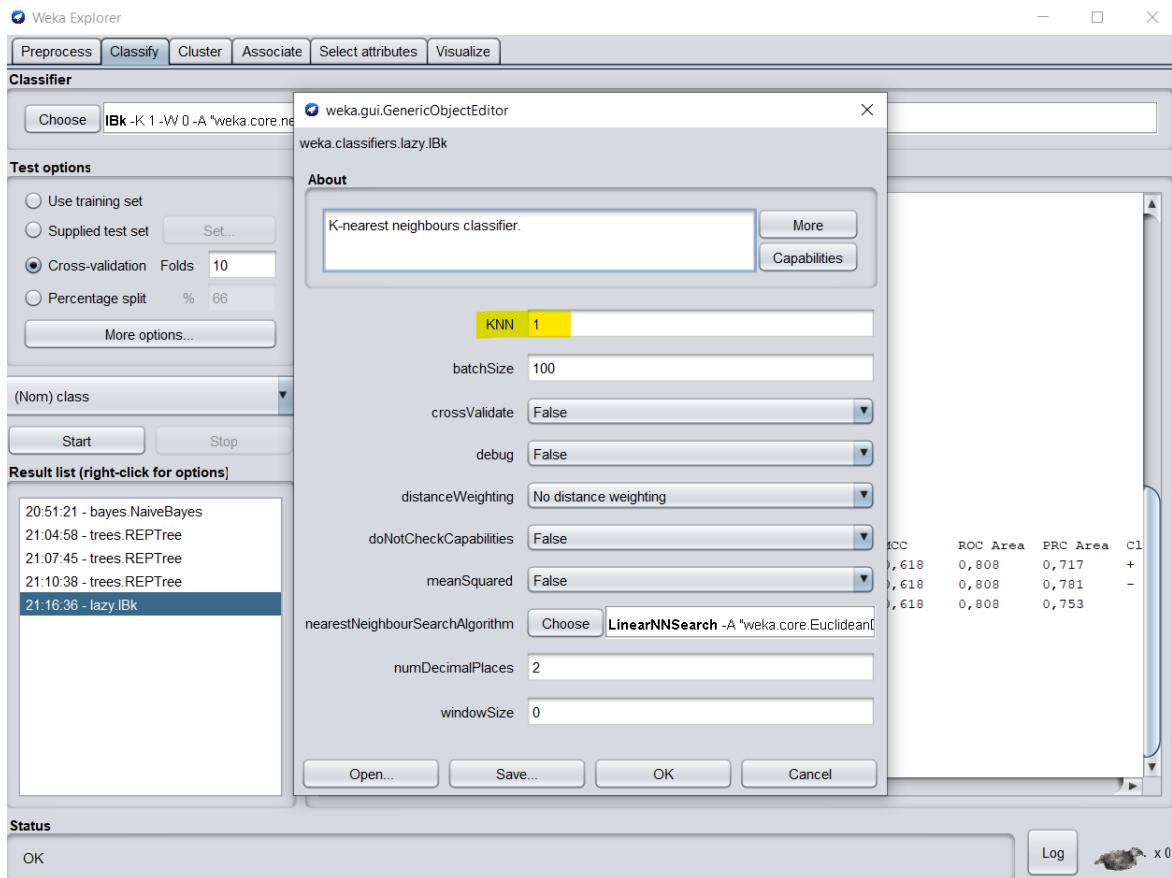
También se le llama KNN para abreviar. Admite clasificación y regresión. Es un algoritmo simple, pero uno que no asume mucho sobre el problema aparte de que la distancia entre instancias de datos es significativa para hacer predicciones. Como tal, a menudo logra un muy buen rendimiento.

En choose seleccionamos weka > Classifier > lazy > IBk y star

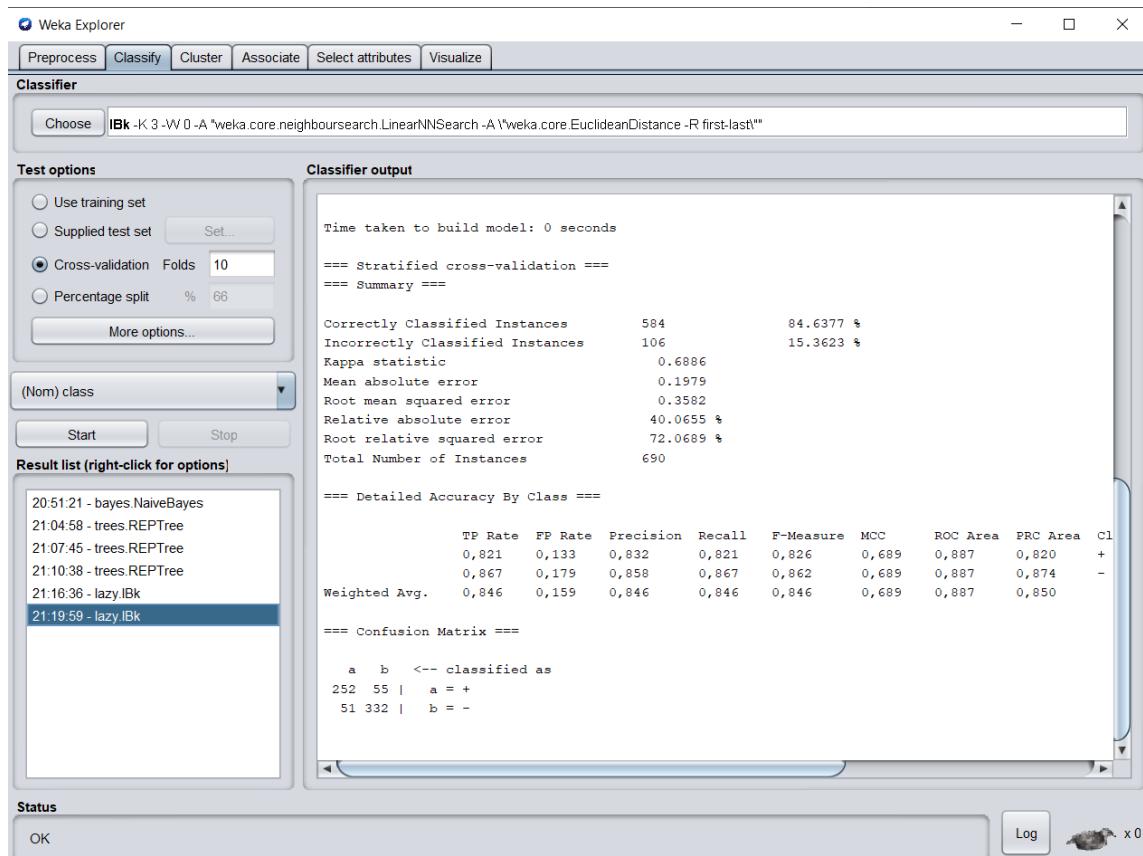


Con este algoritmo obtenemos un 81.1594 % de precisión.

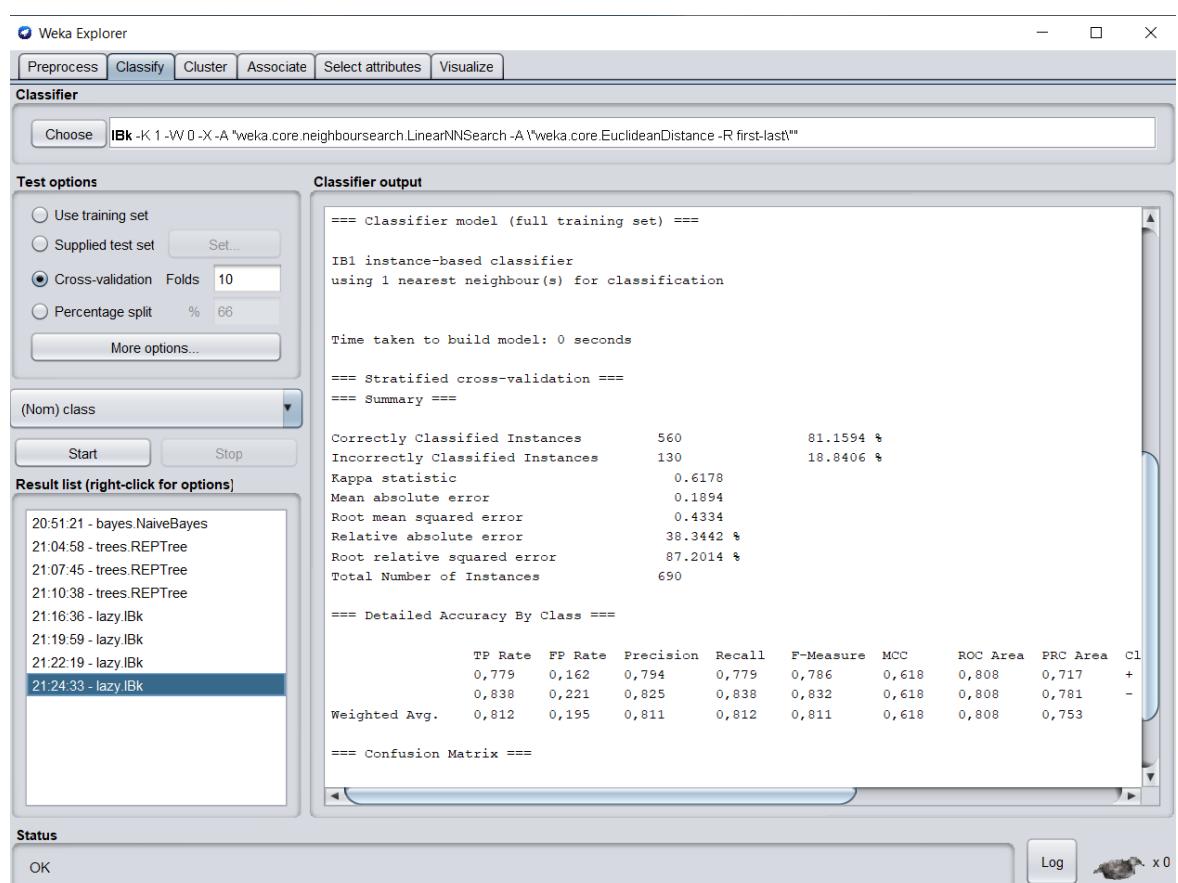
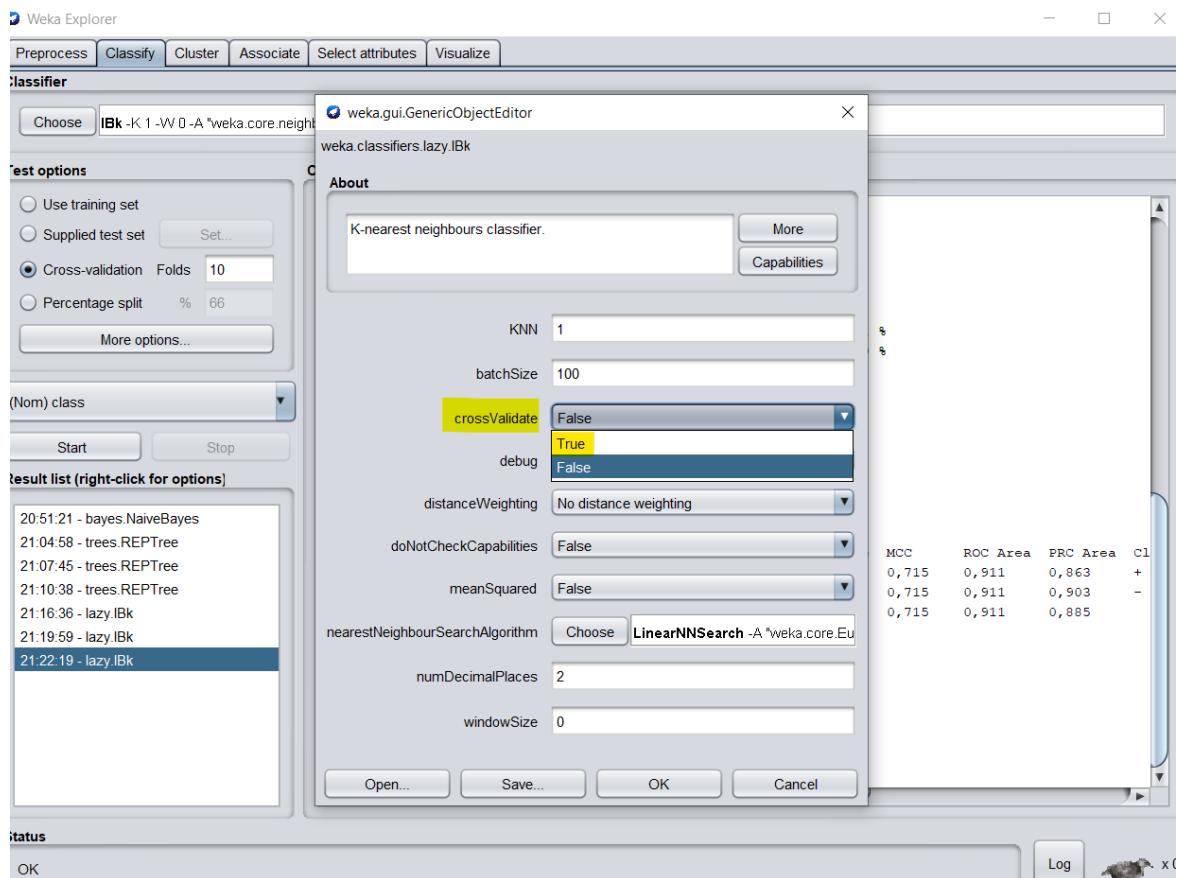
El tamaño de la vecindad está controlado por el parámetro KNN (esto lo hacemos dando clic al algoritmo). Estaba establecido KNN = 1 pero vamos a cambiarlo a 3 a ver qué precisión nos sale, si esta mejora o no.



Vemos como al cambiarlo la precisión aumenta en a un 84.6377 %.



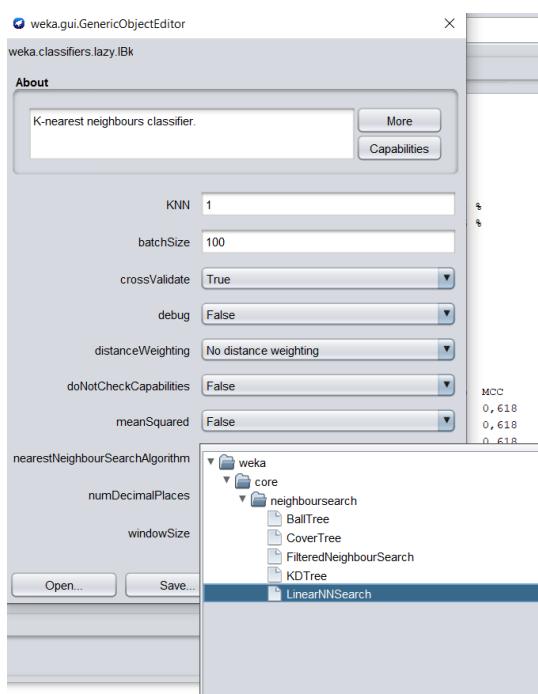
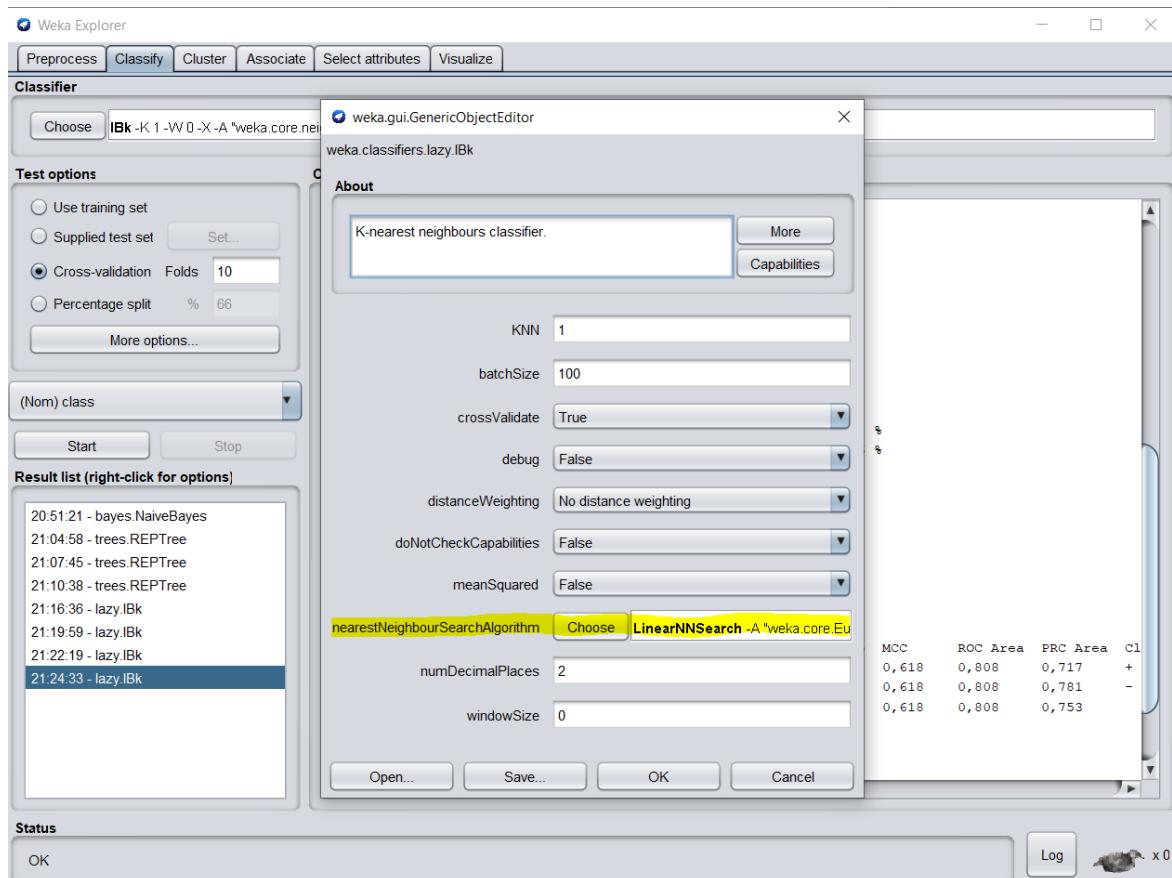
Un dato curioso es que Weka puede descubrir automáticamente un buen valor para KNN usando la validación cruzada dentro del algoritmo estableciendo el parámetro crossValidate en True. La validación cruzada consiste en: dado un número n se divide los datos en n partes y, por cada parte, se construye el clasificador con las n-1 partes restantes y se prueba con esa. Así por cada una de las n particiones.



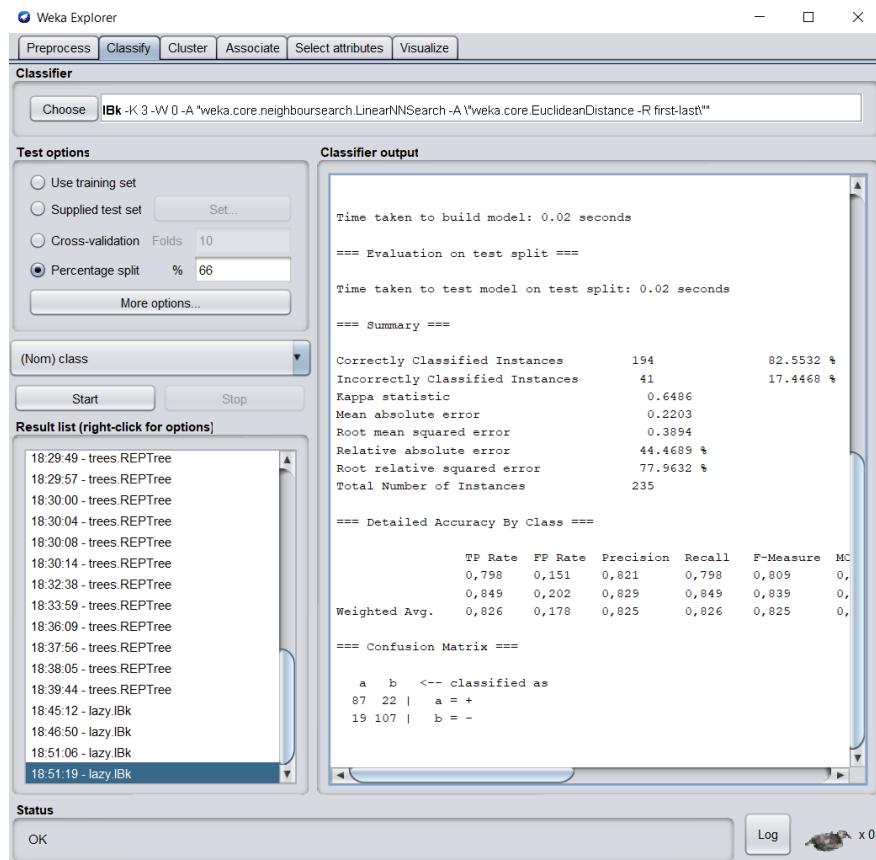
Obtenemos así un 81,1594% de rendimiento.

El parámetro medida de distancia utilizada se configura en `nearestNeighbourSearchAlgorithm` que controla la forma en que se almacenan y buscan los datos de entrenamiento. Este el valor predeterminado es `LinearNNSearch`. Si le damos a `choose` se abre una ventana de configuración donde se puede elegir un

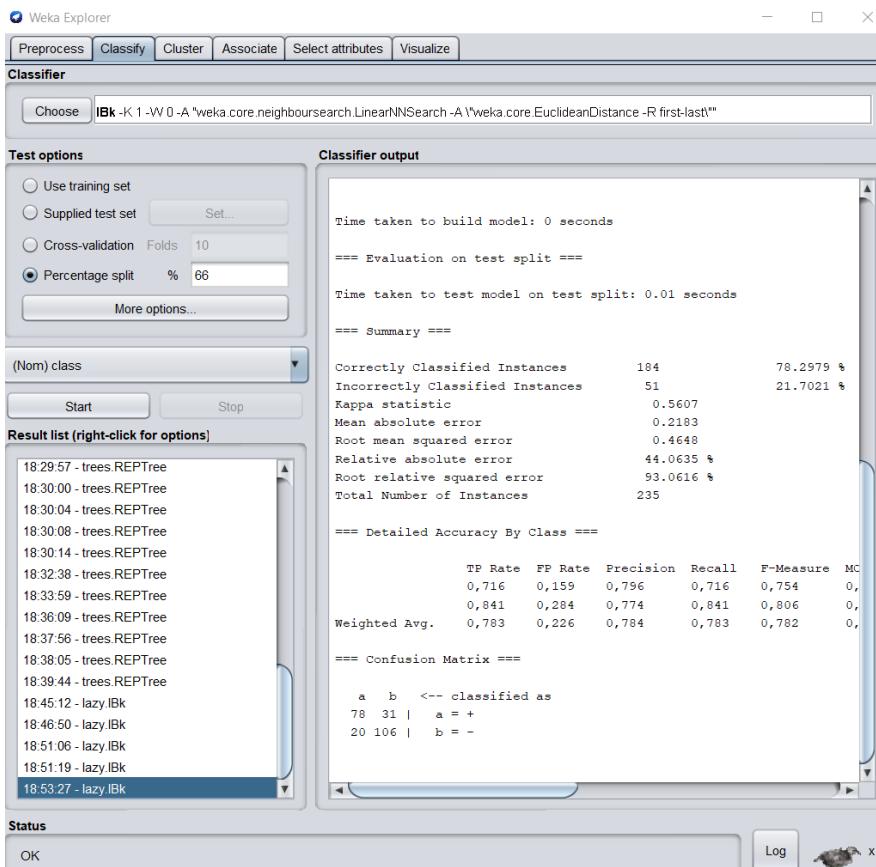
parámetro de función de distancia. De forma predeterminada, la distancia euclídea se usa para calcular la distancia entre instancias, lo cual es bueno para datos numéricos con la misma escala. La distancia de Manhattan es buena si sus atributos difieren en medidas o tipo.



Veremos qué pasa si seleccionamos Percentage Split y ponemos KNN =3. El rendimiento es de un 82.5532 % con 194 instancias clasificadas correctamente.

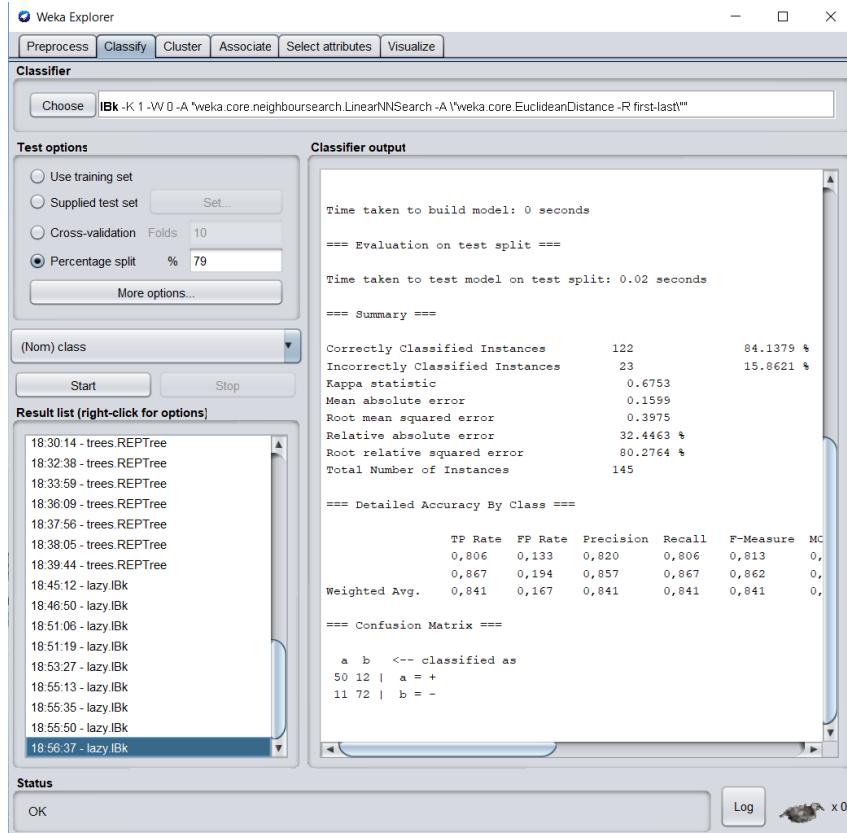


¿Y si dejamos los valores predeterminados que sucede?

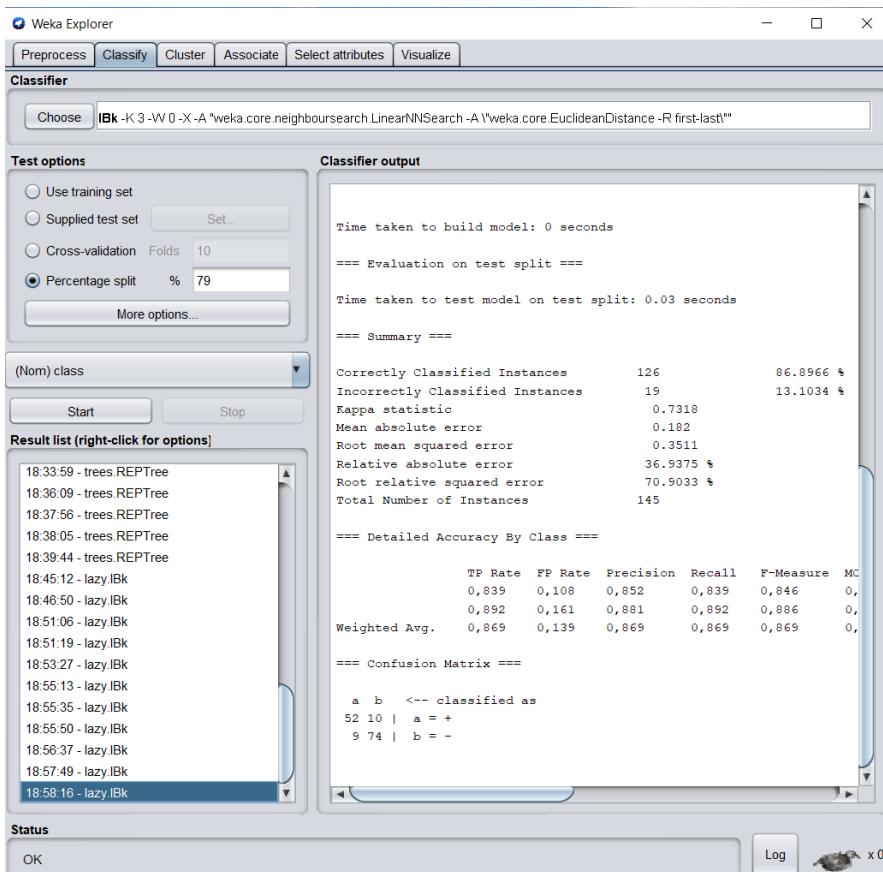


Que el rendimiento baja bastante a un 78.2979 %. Lo que no salgan en la tabla es porque al realizar el crossValidate = True me salían el mismo valor de rendimiento.

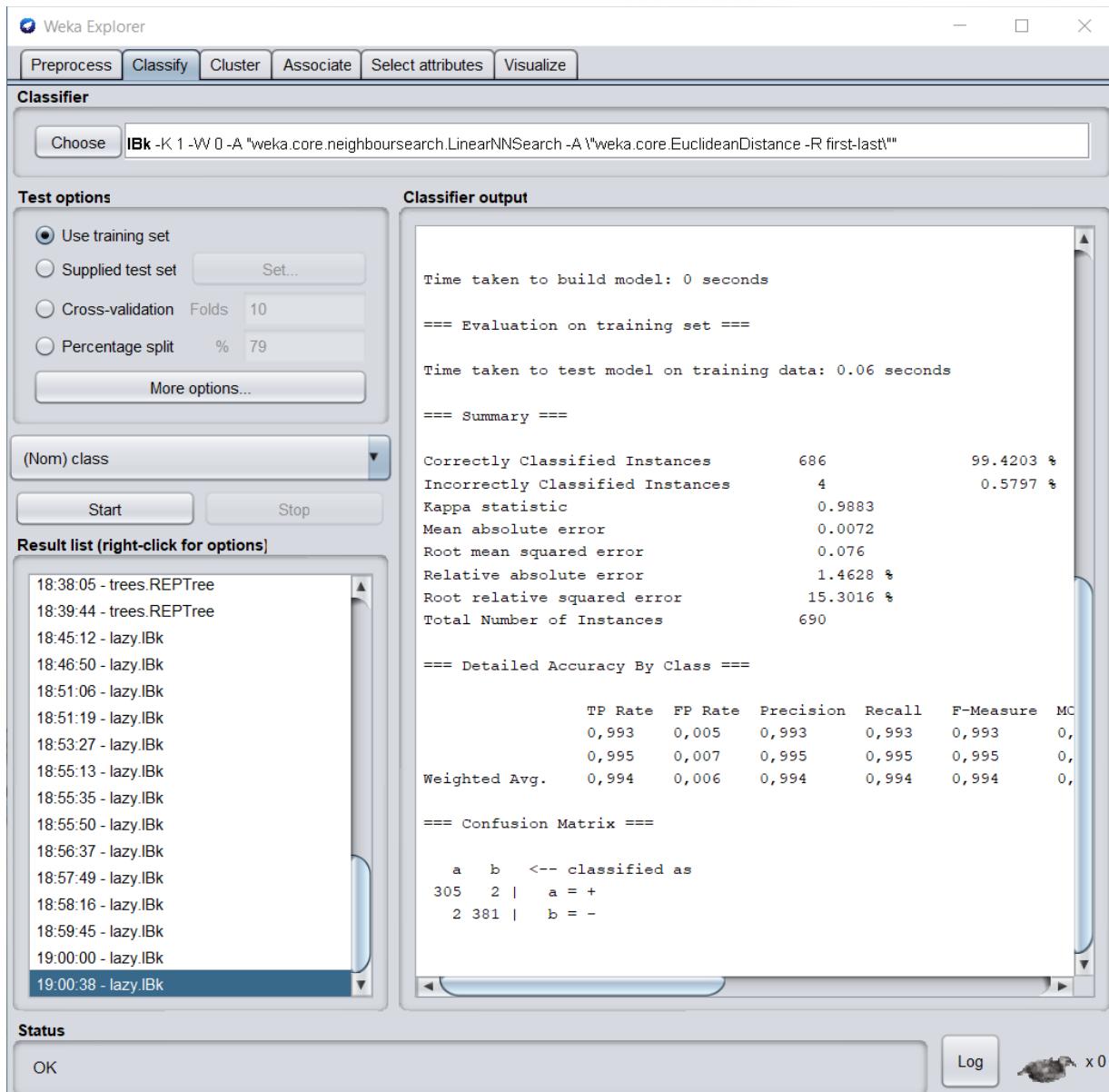
Si editamos manualmente el % y lo cambiamos a 79% el rendimiento mejora a un 84,1379%



Si le realizo el crossValidate = True me sale el mismo %. Entonces dejemos el crossValidate = True y cambiemos el KNN a 3



Mejora el rendimiento, siendo este el más alto hasta ahora con un 86.8966 %. Pero esto dura poco porque si en test options seleccionamos Use training set el rendimiento sigue aumentando hasta un 99.4203 %



Si ponemos KNN= 3 nos sale el mismo rendimiento. En cambio si dejamos KNN=3 pero el crossValidate lo ponemos a True el rendimiento baja aunque es el segundo más alto con un 91.1594 %.

TABLA RESUMEN MODIFICACIONES (IBk)		% RENDIMIENTO DEL ÁRBOL
Percentage Split		78.2979 %
Cross-validation con valores predeterminados		81.1594 %
Cross-validation con KNN = 1 y crossValidate = True		81,1594%
Percentage Split con KNN =3		82.5532 %
Percentage Split 79%		84.1379 %
Cross-validation con KNN = 3		84.6377 %.
Percentage Split 79% , KNN = 3		86.8966 %
Use training, KNN=3, crossValidate = True		91.1594 %
Use training set con valores predeterminados		99.4203 %

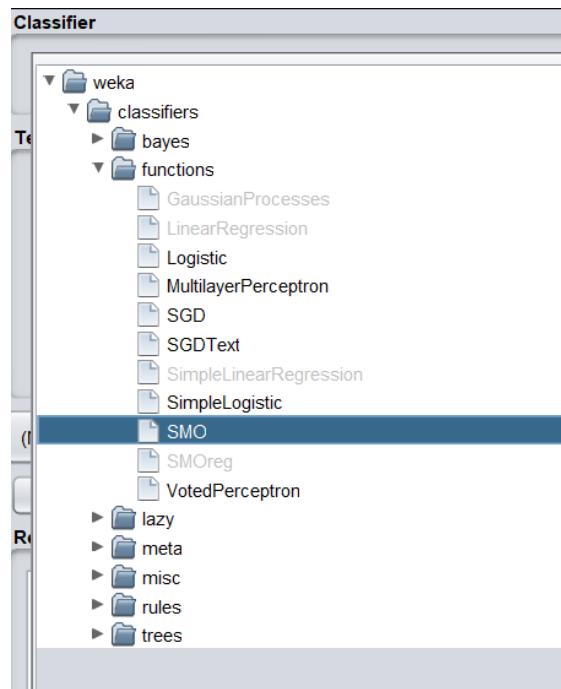
SMO (MÁQUINAS DE VECTORES DE SOPORTE)

Se desarrollaron para problemas de clasificación binaria, aunque se han realizado extensiones a la técnica para admitir problemas de regresión y clasificación de clases múltiples. El algoritmo a menudo se conoce como SVM para abreviar.

SVM trabaja buscando la línea que mejor separa los datos en los dos grupos. Esto se hace mediante un proceso de optimización que solo considera aquellas instancias de datos en el conjunto de datos de entrenamiento que están más cerca de la línea que mejor separa las clases. Las instancias se denominan vectores de soporte, de ahí el nombre de la técnica.

SMO se refiere al algoritmo de optimización eficiente específico utilizado dentro de la implementación de SVM, que significa Optimización mínima secuencial.

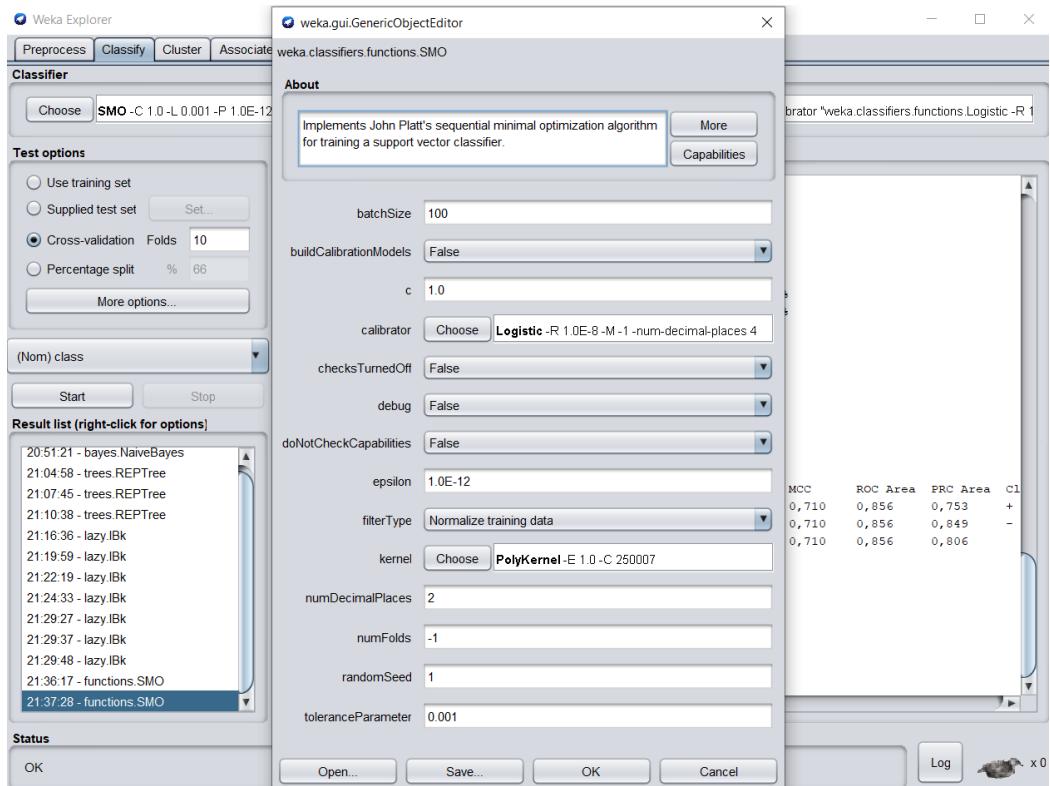
En choose seleccionamos weka> classifiers > functions > SMO y star



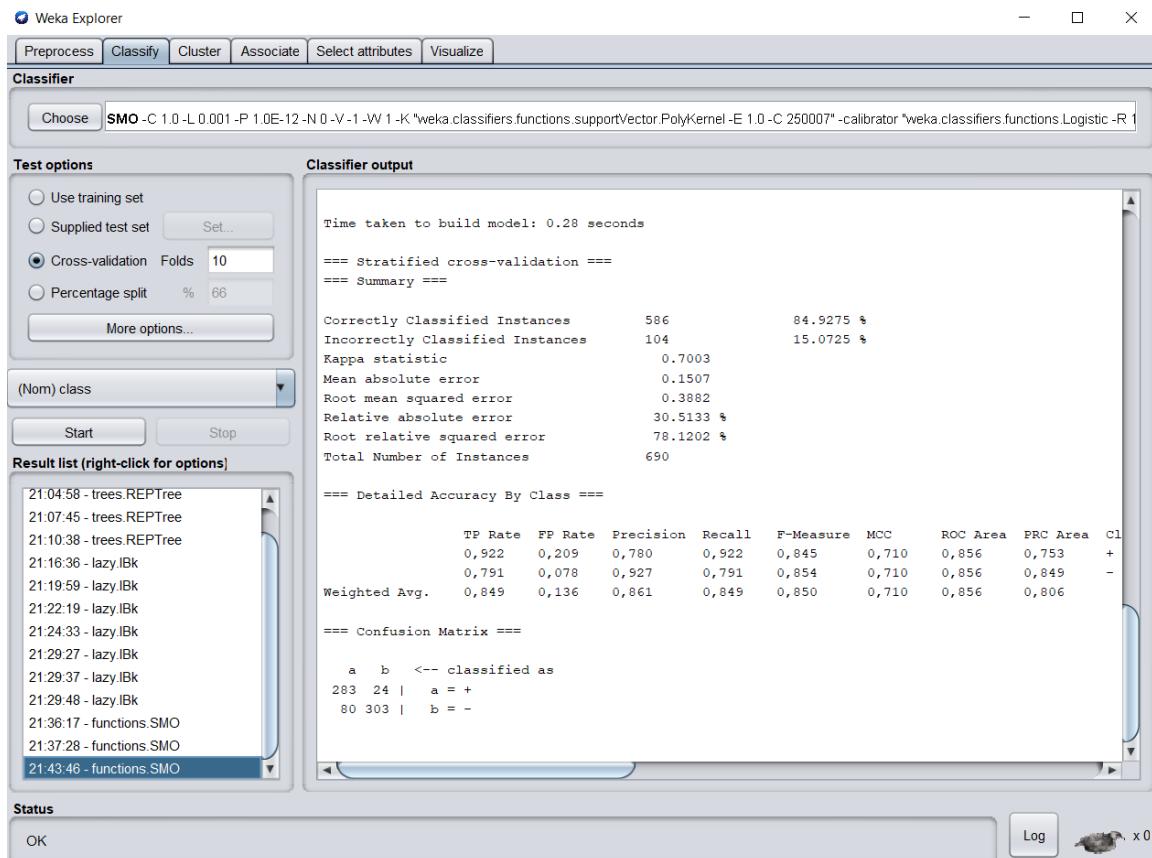
El parámetro C que vemos en la captura de abajo (parámetro de complejidad) controla qué tan flexible puede ser el proceso para trazar la línea para separar las clases. Un valor de 0 no permite violaciones del margen, mientras que el valor predeterminado es 1. Haremos una prueba a continuación.

Un parámetro clave en SVM es el tipo de kernel a utilizar. El kernel más simple es un kernel lineal que separa los datos con una línea recta o hiperplano. El valor predeterminado en Weka es un núcleo polinómico que separará las clases usando una línea curva o ondulada, cuanto más alto sea el polinomio, más ondulado (el valor del exponente).

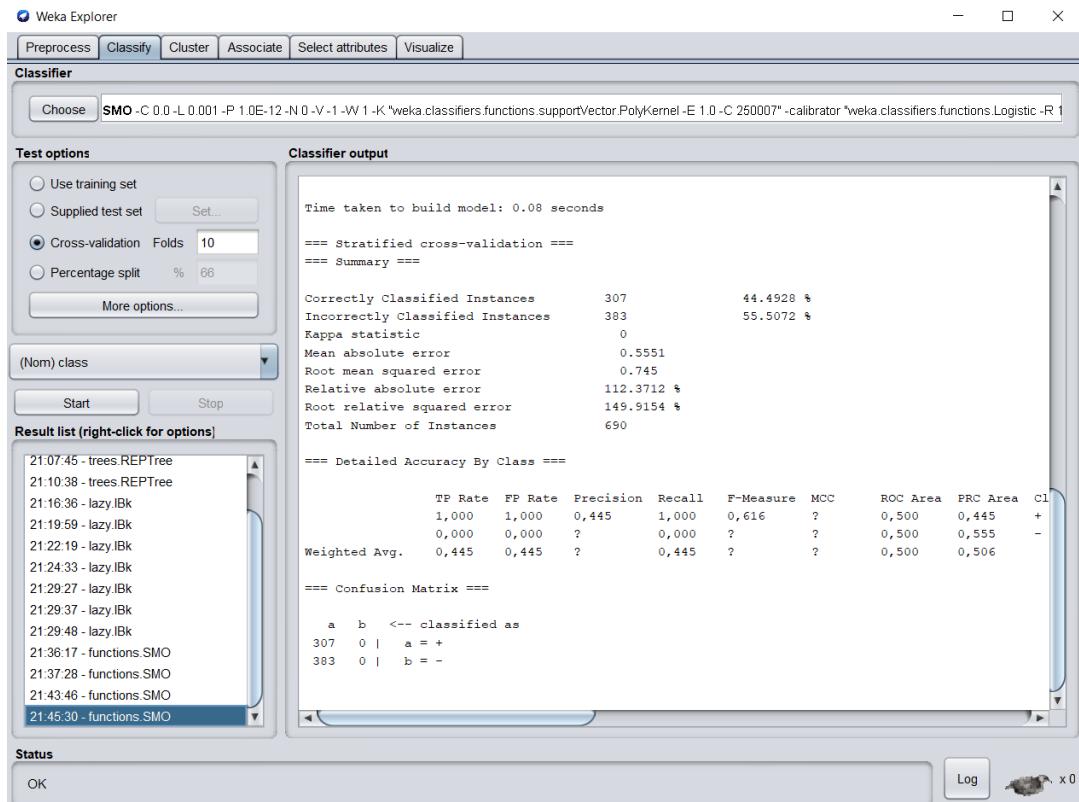
Un núcleo popular y poderoso es el núcleo RBF o núcleo de función de base radial que es capaz de aprender polígonos cerrados y formas complejas para separar las clases.



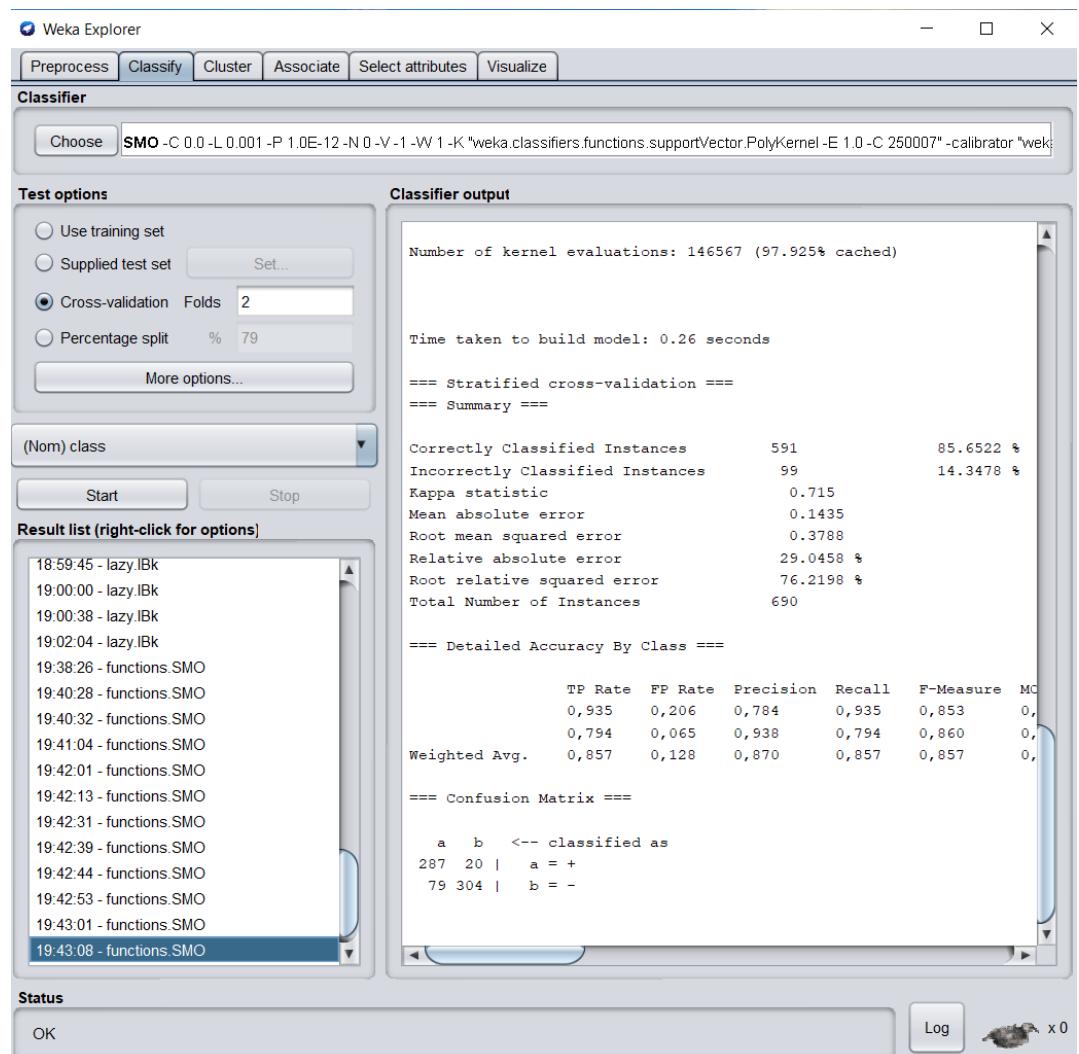
Este resultado nos da con el valor predeterminado de C (c=1) y la precisión nos sale del 84.9275 %.



En cambio vamos a poner que el valor de C sea 0 (c=0). Y vemos como la precisión ha bajada y es nefasta 44.4928 % ya que no permite violaciones del margen.



Si ahora utilizamos un Folds = 2 el rendimiento sube un poquito (85.6522 %) comparado con los valores predeterminados.



Si en test options utilizamos Use training set y nos mejora el rendimiento 85.942 %.

The screenshot shows the Weka Explorer interface with the following details:

- Top Bar:** Weka Explorer, Preprocess, Classify (selected), Cluster, Associate, Select attributes, Visualize.
- Classifier Panel:** Choose SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.calculators.regression.Ridge".
- Test options Panel:** Use training set (radio button selected). Other options: Supplied test set (Set...), Cross-validation (Folds 2), Percentage split (% 66), More options... (button).
- Result list (right-click for options):** A scrollable list of log entries from 19:02:04 to 19:46:35, mostly related to SMO classifier runs.
- Classifier output Panel:**
 - Time taken to build model: 0.25 seconds
 - ==== Evaluation on training set ====
 - Time taken to test model on training data: 0 seconds
 - ==== Summary ====

	Correctly Classified Instances	593	85.942 %
Incorrectly Classified Instances	97	14.058 %	
Kappa statistic	0.7211		
Mean absolute error	0.1406		
Root mean squared error	0.3749		
Relative absolute error	28.4602 %		
Root relative squared error	75.447 %		
Total Number of Instances	690		
 - ==== Detailed Accuracy By Class ====

	TP Rate	FP Rate	Precision	Recall	F-Measure	...
0,945	0,209	0,784	0,945	0,857	0,	
0,791	0,055	0,947	0,791	0,862	0,	
Weighted Avg.	0,859	0,124	0,874	0,859	0,860	0,
 - ==== Confusion Matrix ====

	a	b	<- classified as
290	17	17	a = +
80	303	303	b = -
- Status Panel:** OK.
- Log Panel:** Shows a log entry with a file icon and x 0.

Cuando usamos Percentage split con valores predeterminados el rendimiento empeora 83.8298 % pero si lo modificamos y lo ponemos a un 79% el rendimiento mejora bastante a un 86.2069 % siendo este el mejor de todos los probados. Con número de evaluaciones del kernel: 146567 (97,925% en caché)

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.calibration.BayesianCalibration -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007"

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 2
- Percentage split % 79

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 19:40:28 - functions.SMO
- 19:40:32 - functions.SMO
- 19:41:04 - functions.SMO
- 19:41:04 - functions.SMO

Classifier output

```

+ 0.4997 * (normalized) A13=s
+ 0.0195 * (normalized) A14
+ -0.0919 * (normalized) A15
- 2.162

Number of kernel evaluations: 146567 (97.925% cached)

Time taken to build model: 0.28 seconds

==== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

==== Summary ===

Correctly Classified Instances      125          86.2069 %
Incorrectly Classified Instances   20           13.7931 %

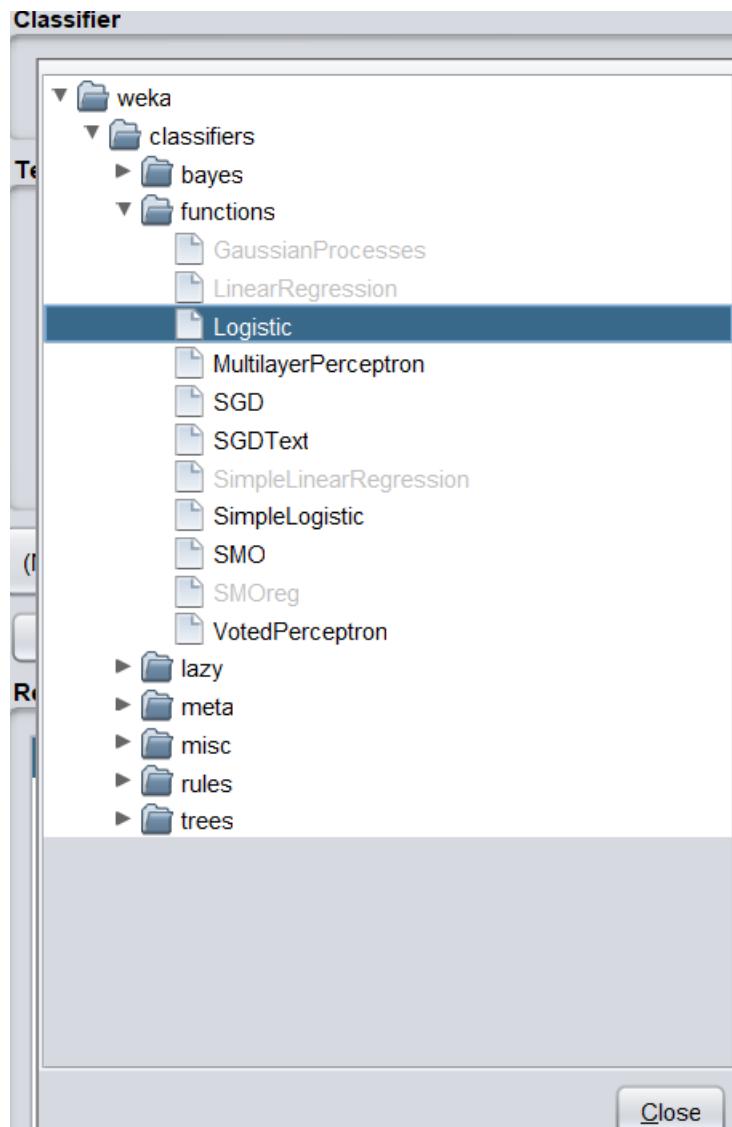
```

TABLA RESUMEN MODIFICACIONES (SMO)		% DE RENDIMIENTO DEL ÁRBOL
Cross-validation , C = 0.0		44.4928 %
Percentage split con valores predeterminados		83.8298 %
Cross-validation con valores predeterminados		84.9275 %
Cross.validation , Folds = 2		85.6522 %
Use training set		85.942 %
Percentage Split 79%		86.2069 %

Logistic (REGRESIÓN LOGÍSTICA)

Es un algoritmo de clasificación binaria. En choose seleccionamos Logistic. Asume que las variables de entrada son numéricas y tienen una distribución gaussiana (curva de campana). La regresión logística aún puede lograr buenos resultados si sus datos no son gaussianos. En el caso del conjunto de datos de la ionosfera, algunos atributos de entrada tienen una distribución similar a la de Gauss, pero muchos no la tienen.

El algoritmo aprende un coeficiente para cada valor de entrada, que se combinan linealmente en una función de regresión y se transforman mediante una función logística (en forma de s). La regresión logística es una técnica rápida y sencilla, pero puede resultar muy eficaz en algunos problemas.



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **Logistic -R 1.0E-8 -M 1 -num-decimal-places 4**

Test options

Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66
More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 20:30:27 - functions MultilayerPerceptron
- 20:41:54 - functions Logistic**

Classifier output

```
==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances      588          85.2174 %
Incorrectly Classified Instances    102          14.7826 %
Kappa statistic                      0.7024
Mean absolute error                  0.1954
Root mean squared error              0.3329
Relative absolute error              39.5619 %
Root relative squared error         66.9767 %
Total Number of Instances            690

==== Detailed Accuracy By Class ====

           TP Rate   FP Rate   Precision   Recall   F-Measure   MCC   ROC Area   PRC Area   Cl
          0.863     0.157     0.815     0.863     0.839     0.703   0.904     0.858     +
          0.643     0.137     0.685     0.843     0.864     0.703   0.904     0.874     -
Weighted Avg.   0.852     0.146     0.854     0.852     0.853     0.703   0.904     0.867

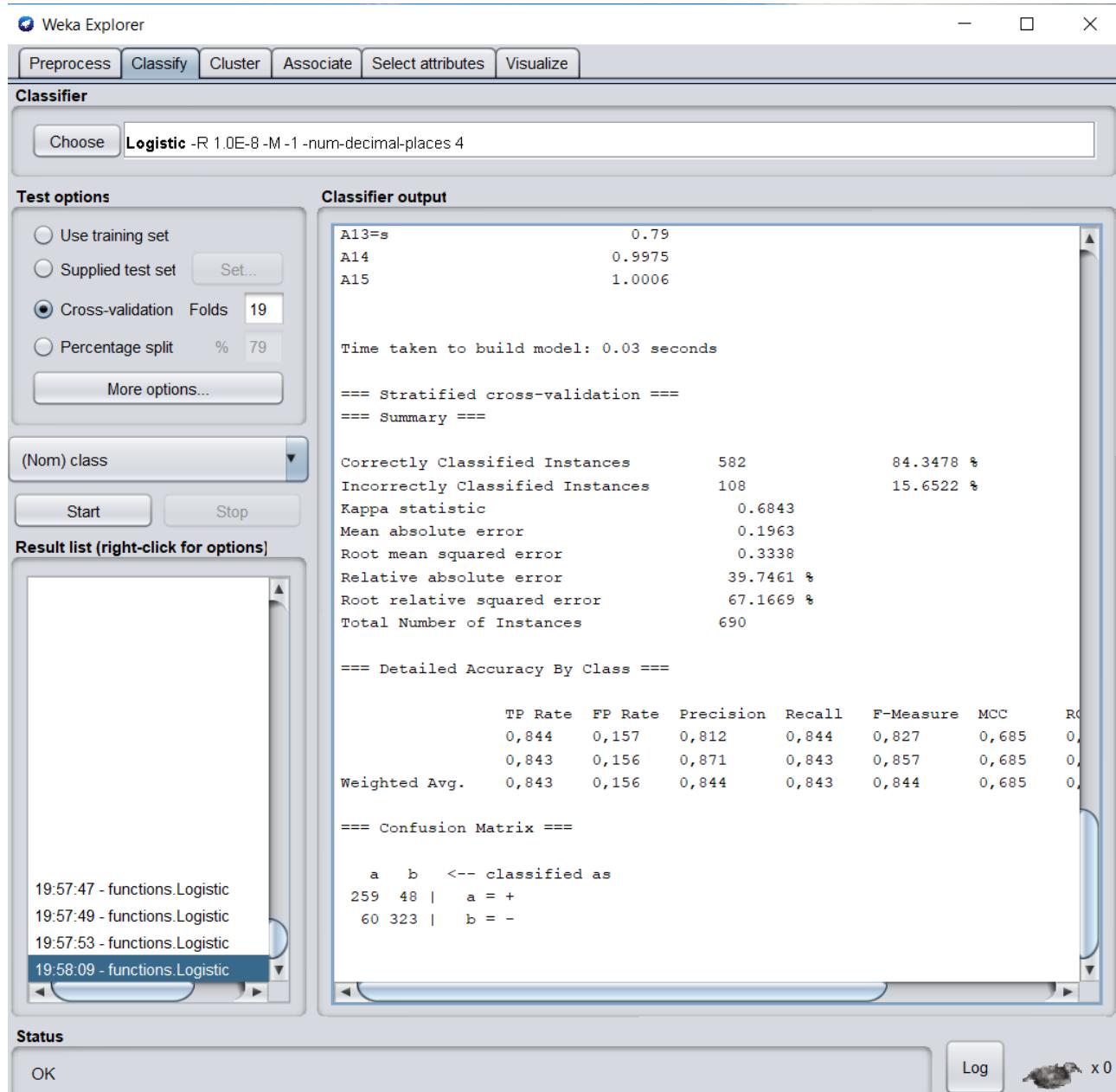
==== Confusion Matrix ====

      a   b   <- classified as
265  42 |   a = +
 60 323 |   b = -
```

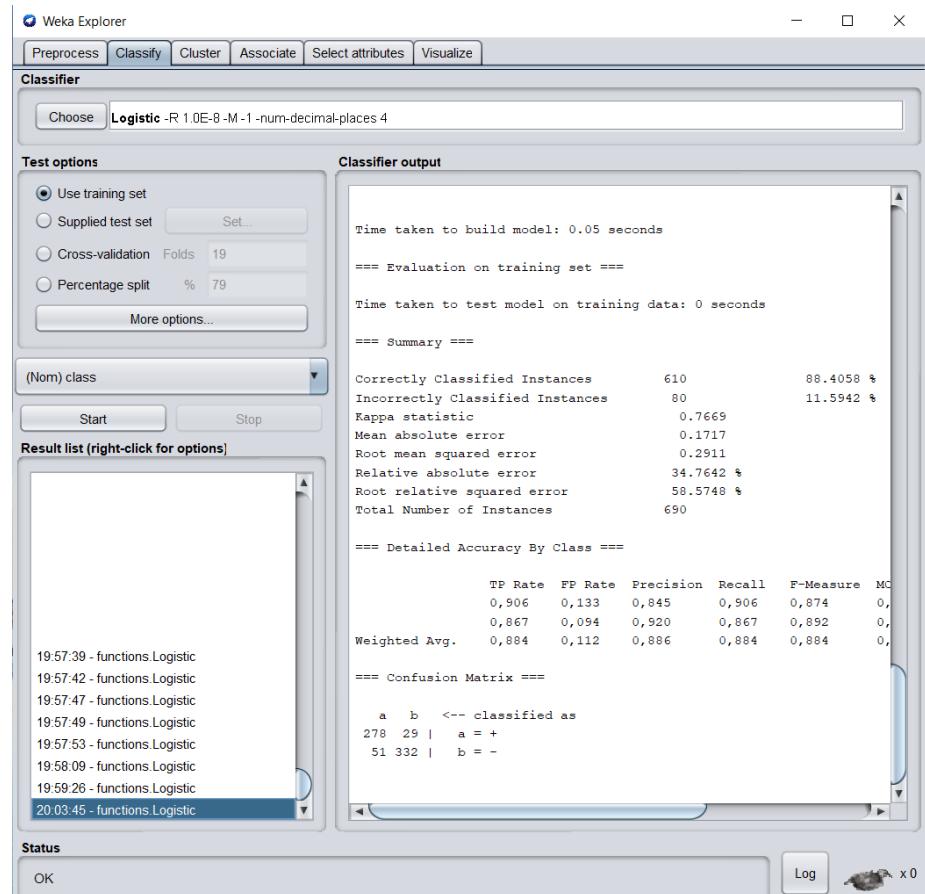
Status

OK Log x 0

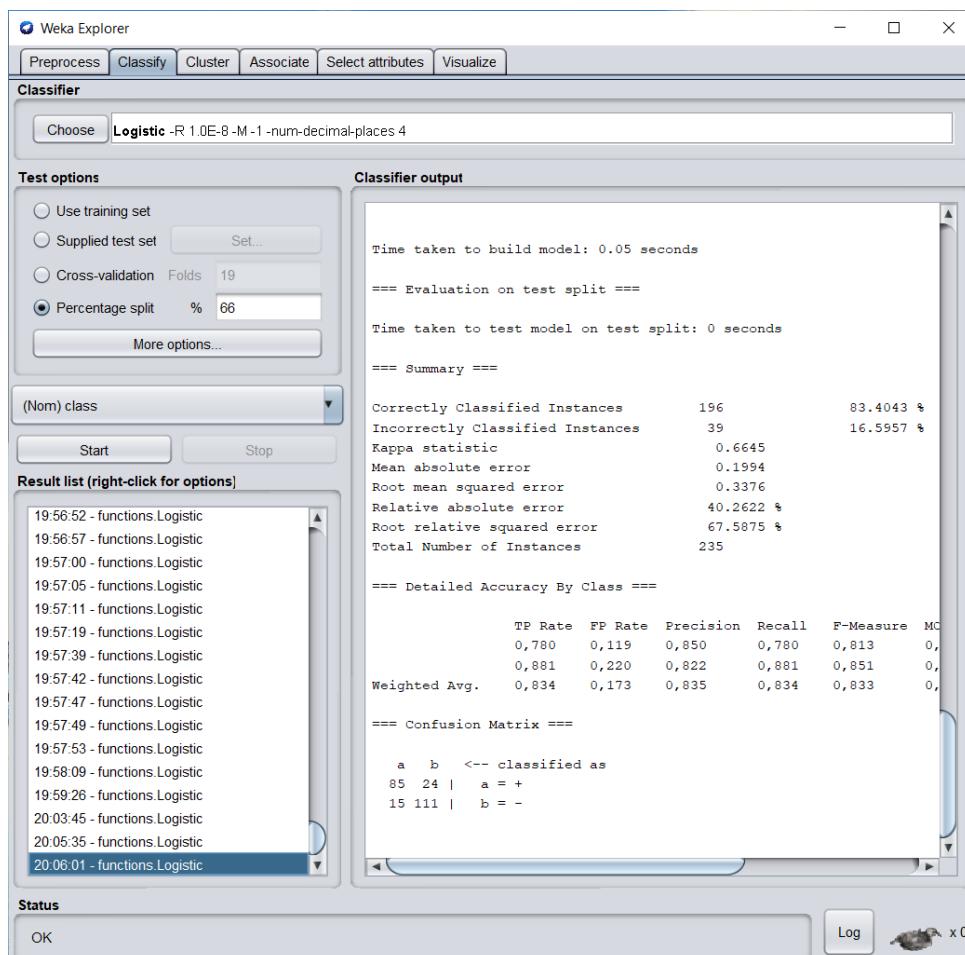
Como podemos ver la regresión logística de nuestro conjunto de datos logra una precisión del 85.2174 % y si cambiamos el Folds a 19 el rendimiento no mejora lo que hace es empeorar un poco 84.3478 %.



Ahora vamos a seleccionar la sección Use training set y dejamos los valores predeterminados. Así obtenemos un rendimiento del 88.4058%



Queriendo mejorar el rendimiento seleccionamos Percentage split con los valores predeterminados pero obtenemos el peor rendimiento que hemos obtenido hasta ahora 83.4043 %



Para mejorar este rendimiento último vamos a modificar el % y lo pondremos a 79% viendo así como se obtiene un 86,8966% frente a un 13.1034% de instancias incorrectamente clasificados.

The screenshot shows the Weka Explorer interface with the 'Classifier' tab selected. In the 'Test options' section, 'Percentage split' is chosen with 79% selected. The 'Classifier output' pane displays the evaluation results:

```

Time taken to build model: 0.05 seconds

==== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

==== Summary ===

Correctly Classified Instances           126          86.8966 %
Incorrectly Classified Instances        19          13.1034 %
Kappa statistic                         0.7307
Mean absolute error                     0.1697
Root mean squared error                 0.3015
Relative absolute error                 34.4449 %
Root relative squared error            60.8735 %
Total Number of Instances               145

==== Detailed Accuracy By Class ===

      TP Rate   FP Rate   Precision   Recall   F-Measure   M
      0,823     0,096    0,864       0,823    0,843      0,
      0,904     0,177    0,872       0,904    0,888      0,
Weighted Avg.   0,869     0,143    0,869       0,869    0,869      0

==== Confusion Matrix ===

  a   b   <-- classified as
51  11 |  a = +
  8  75 |  b = -

```

The 'Result list' pane shows a log of function calls, and the 'Status' pane shows 'OK'.

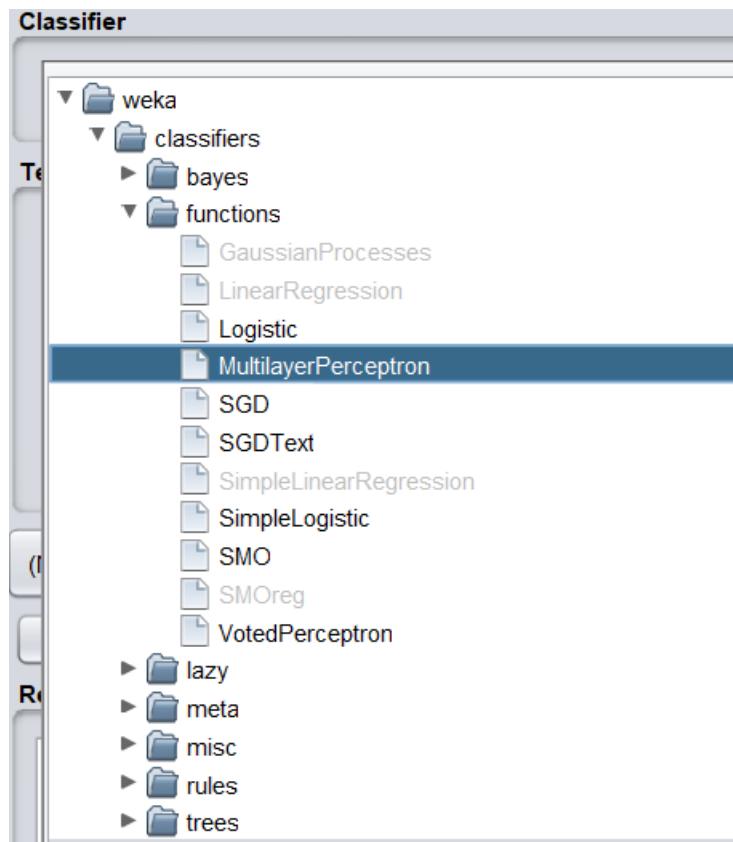
Al hacer click en el filtro para modificar valores estaba debug = False (que es la depuración) al ponerlo a True en las modificaciones de la tabla de abajo no cambia el % rendimiento de ellos.

TABLA RESUMEN MODIFICACIONES (Logistic)		% RENDIMIENTO DEL ÁRBOL
Percentage split con valores predeterminados		83.4043 %
Cross-validation, Folds = 19		84.3478 %
Cross-validation con valores predeterminados		85.2174 %
Percentage split 79%		86.8966 %
Use training set con valores predeterminados		88.4058%

MLP (MULTILAYER PERCEPTRON)

MLP es un algoritmo diseñado para la clasificación supervisada.

Dentro de Classifier le damos a choose y Multilayer perception y a star.

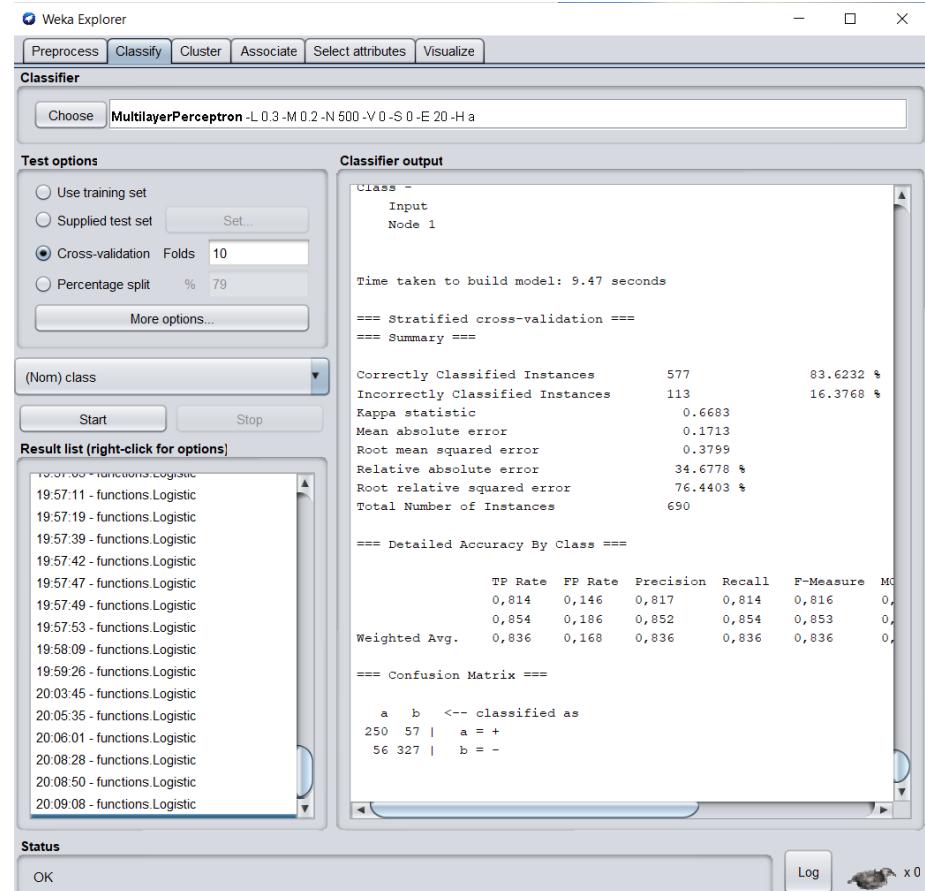


Nos sale una ventana (classifier output) donde podemos ver los atributos del conjunto y mucha más información respectiva al mismo.

A screenshot of the Weka Classifier output window. The top bar shows tabs: Preprocess, Classify, Cluster, Associate, Select attributes, and Visualize. The 'Classifier' tab is selected. In the center, there's a 'Choose' button followed by the text 'MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a'. Below this are sections for 'Test options' and 'Classifier output'. The 'Test options' section includes radio buttons for 'Use training set', 'Supplied test set', 'Cross-validation' (set to 10), and 'Percentage split' (set to 66). The 'Classifier output' section displays the following text:

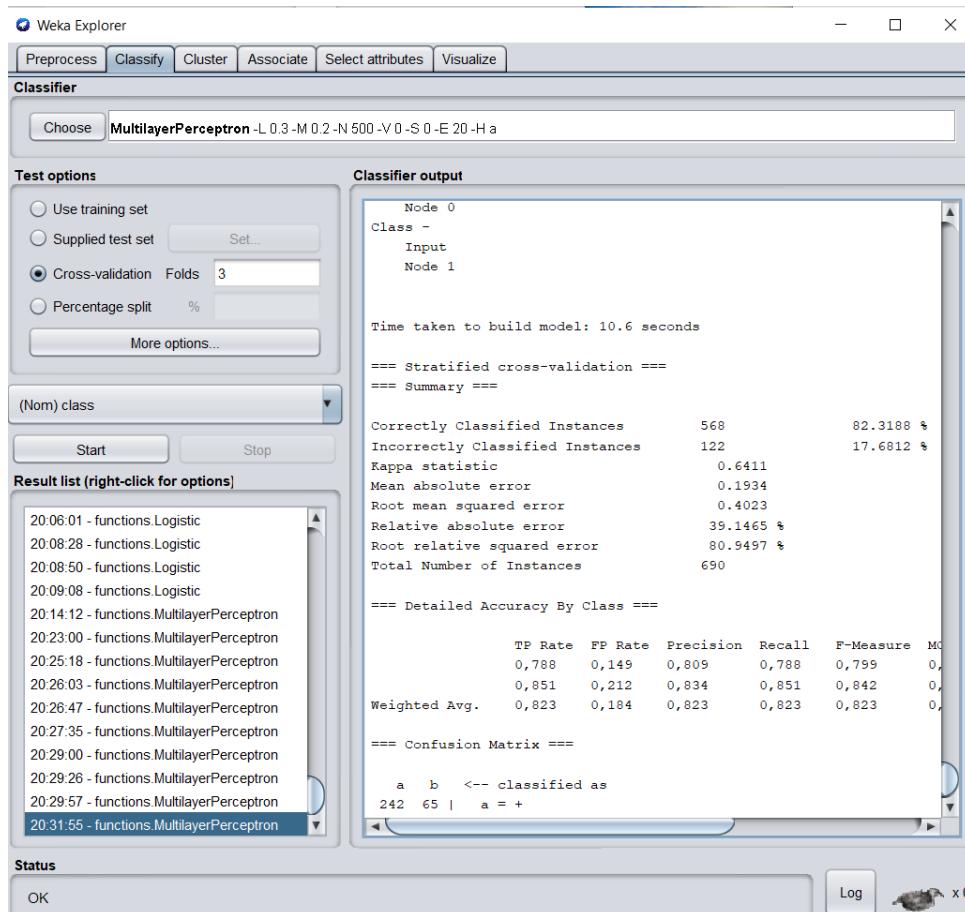
```
==== Run information ====
Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation:    credit-rating
Instances:   690
Attributes:  16
             A1
             A2
             A3
             A4
             A5
             A6
             A7
             A8
             A9
             A10
             A11
             A12
             A13
             A14
             A15
             class
Test mode:   10-fold cross-validation
==== Classifier model (full training set) ====
Sigmoid Node 0
  Inputs      Weights
  Threshold   -5.0591489824362
  Node 2     -1.88788575131983
  Node 3     -7.487329084290409
  Node 4     4.870250407019657
  Node 5     0.268209013928206
  Node 6     4.785423044577189
```

The bottom status bar says 'Building model for fold 7...'. There are also 'Start' and 'Stop' buttons for the process.

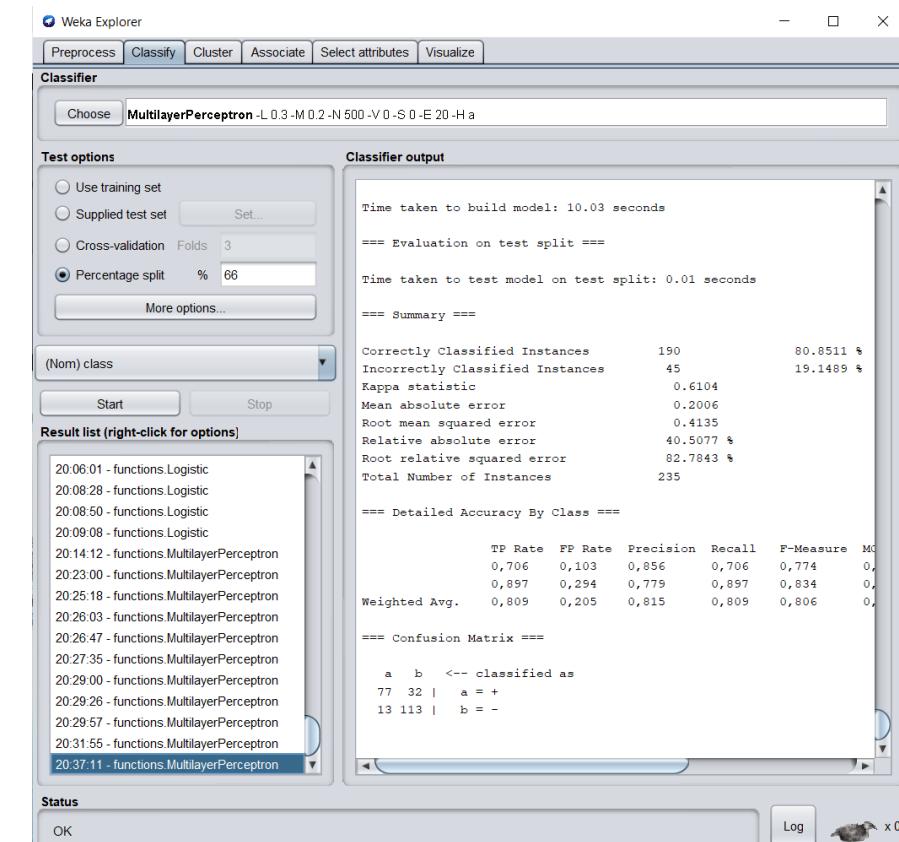


Vemos que este ha salido un rendimiento del 83.6232%

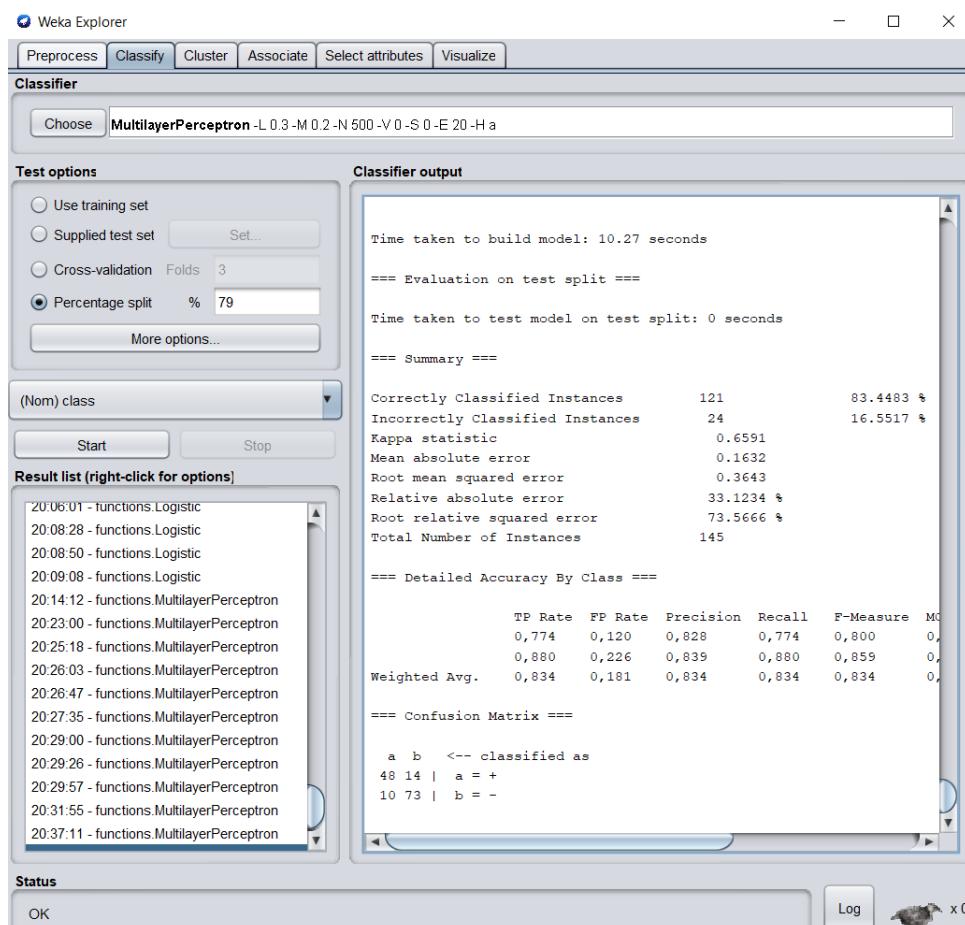
Si cambiamos el número de Folds y lo ponemos a 3 y no en 10 que era el valor predeterminado el rendimeinto baja 82.3188 %



Llega a bajar más el rendimiento si seleccionamos Percentage split con los valores predeterminados que nos da 80,8511%



Ahora vamos a cambiar el % manualmente y lo pondremos a 79% para ver si el rendimiento aumenta. Como podemos ver efectivamente aumenta el rendimiento a un 83.4483 %.



Por último en test options seleccionamos Use training set con los valores predeterminados. Y obtenemos un rendimiento del 96,2319% es decir 664 instancias clasificadas correctamente. Frente a 26 instancias que estan clasificadas incorrectamente.

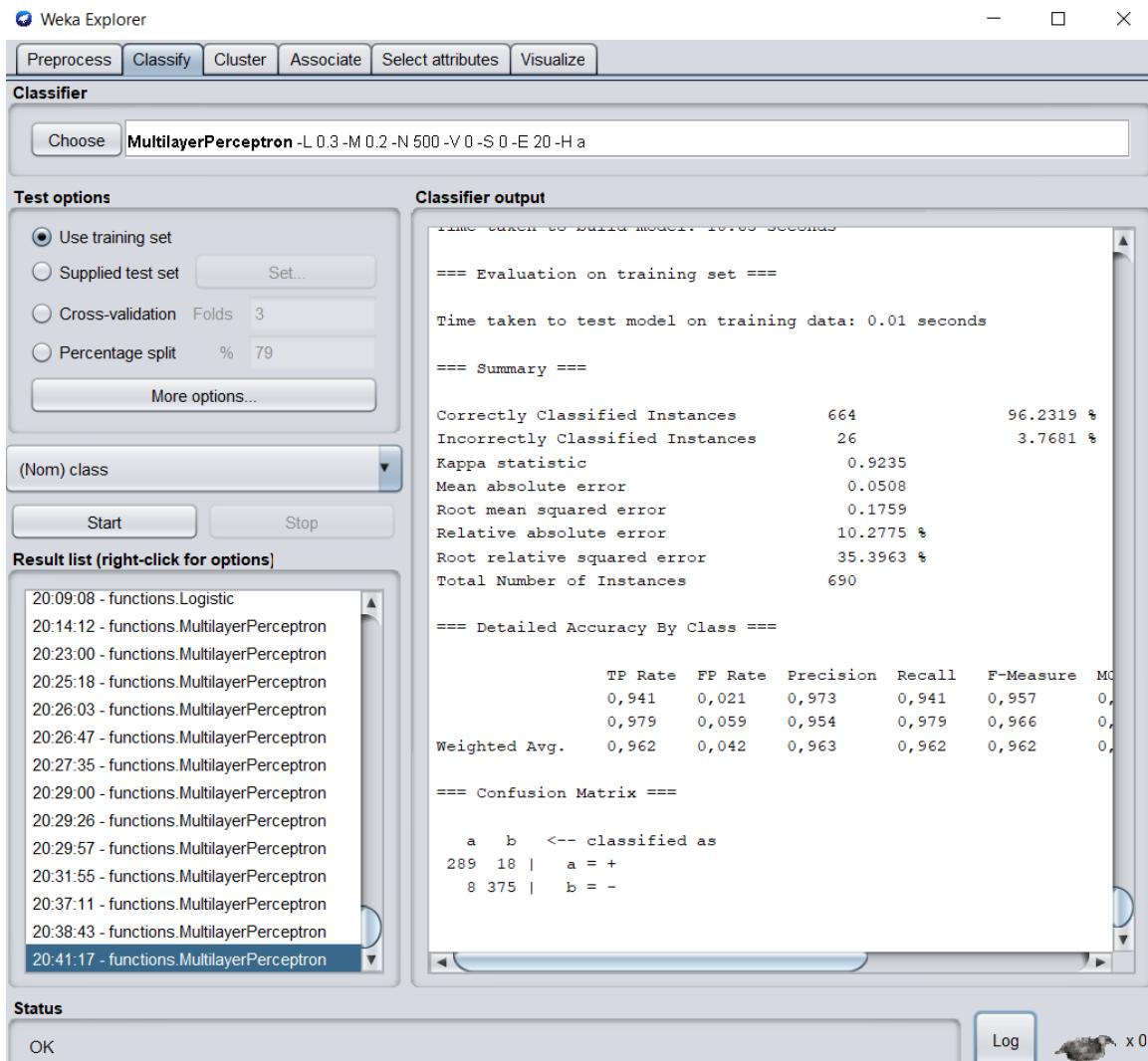
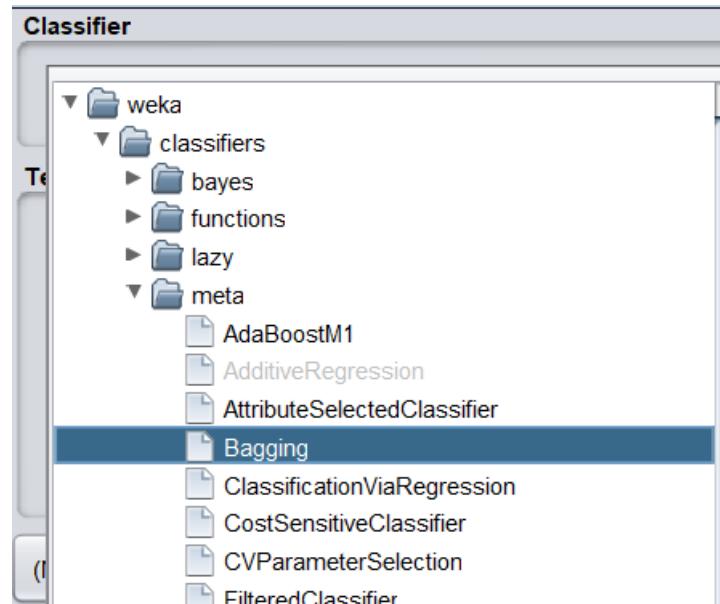


TABLA RESUMEN MODIFICACIONES (MultilayerPerceptron)		% RENDIMIENTO DEL ÁRBOL
Percentage split con valores predeterminados		80,8511%
Cross-validation , Folds = 3		82.3188 %
Percentage con un 79%		83.4483 %
Cross-validation con los valores predeterminados		83.6232 %
Use training set		96.2319 %

BAGGING

El Bagging es una de las técnicas de construcción de conjuntos que también se conoce como Agregación Bootstrap. Dada una muestra de datos, se extraen varias muestras, bootstrapped. Esta selección se realiza de manera aleatoria, es decir, cada variable se puede elegir de la población original, de modo que cada variable es igualmente probable que se seleccione en cada iteración del proceso de arranque. Puede hacer clasificación y regresión dependiendo del algoritmo base.

En choose se selecciona meta y Bagging



Si en Test opstions elegimos Use training set obtenemos un rendimiento del 90.5797 % es decir, 625 instancias clasificadas correctamente. En cambio en cross-validation con los valores predeterminados el rendimiento es menor 85.6522 % , 591 instancias clasificadas correctamente. Si cambiamos el número de Folds de esta última y lo ponemos a 19 hay una pequeña mejoría en el rendimiento aunque es muy pequeña 86.5217 %. En la sección Percentage split si dejamos los valores predeterminados obtenemos un rendimiento del 85,5319% en cambio si el % lo cambio manualmente y elijo 79% sube un poco el rendimiento 86.2069% y de otro modo si pongo un 50% el rendimiento baja un poco a un 84,6377%.

Las capturas para verificar estos valores se adjuntan a continuación de manera ordenada.

This screenshot shows the Weka Explorer interface with the 'Classifier' tab selected. The 'Choose' dropdown is set to 'Bagging -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0'. The 'Test options' panel shows 'Use training set' selected. The 'Classifier output' panel displays the evaluation results on the training set:

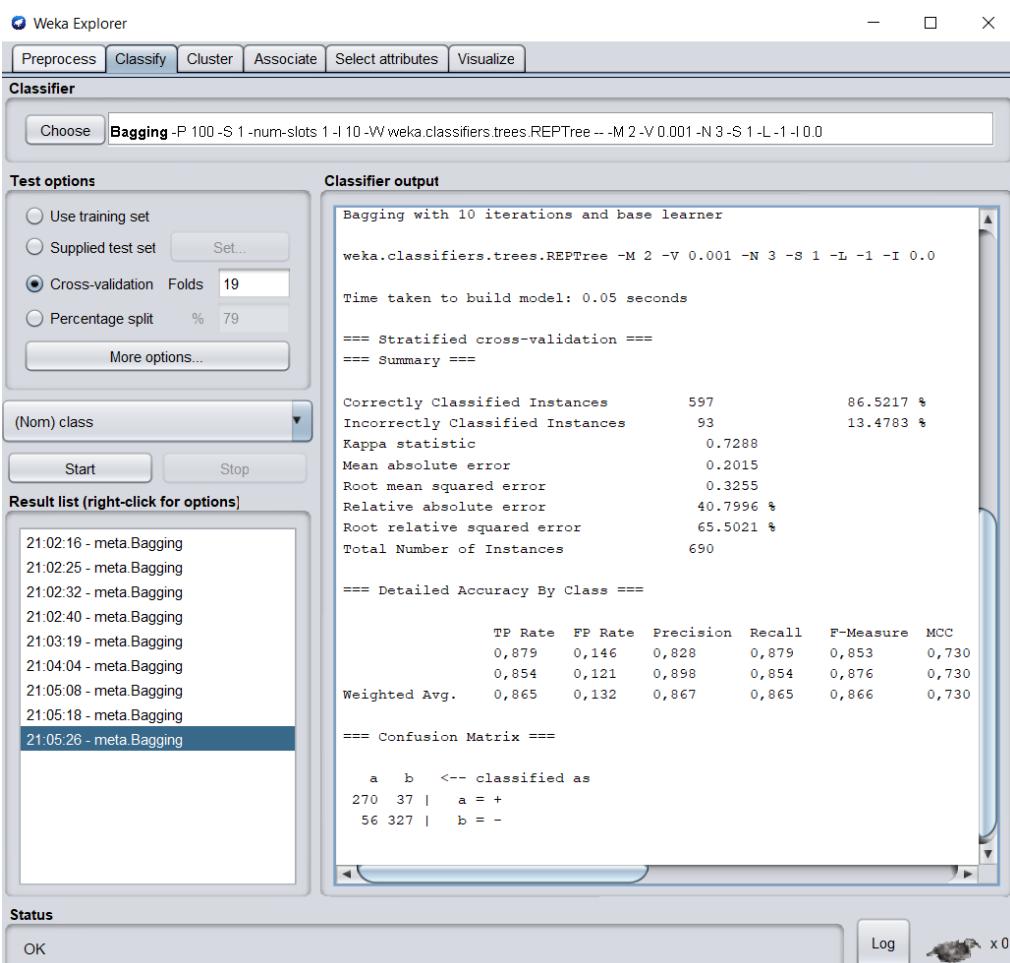
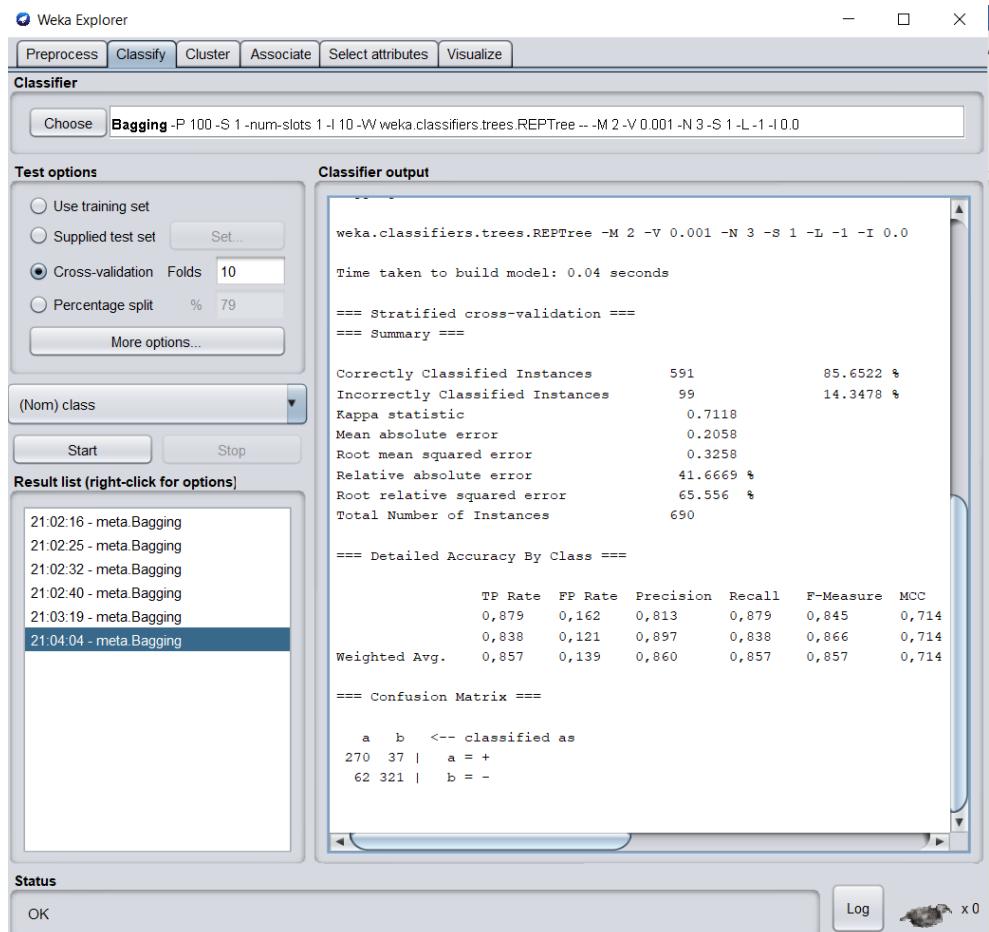
```

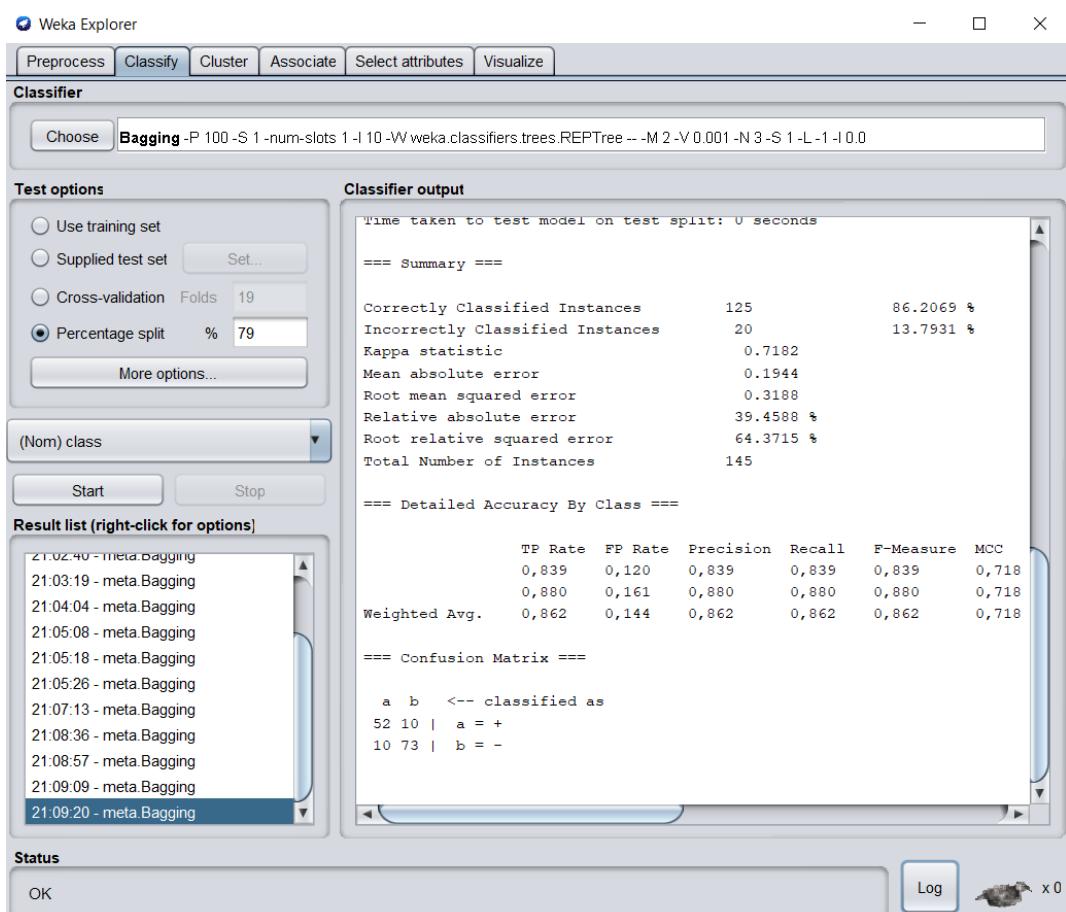
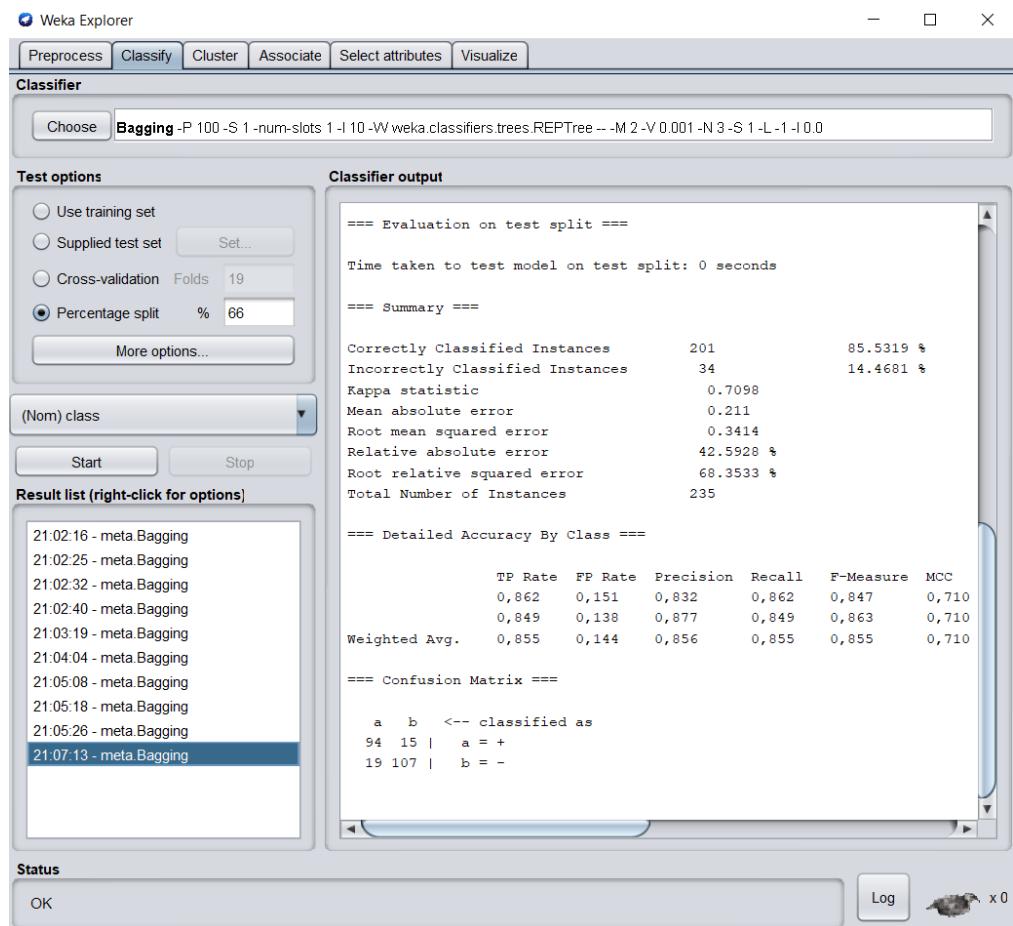
==== Evaluation on training set ====
Time taken to test model on training data: 0.01 seconds

==== Summary ====
Correctly Classified Instances      625          90.5797 %
Incorrectly Classified Instances   65           9.4203 %
Kappa statistic                   0.8097
Mean absolute error               0.1579
Root mean squared error          0.2626
Relative absolute error          31.9586 %
Root relative squared error     52.8418 %
Total Number of Instances        690

```

The 'Result list' panel shows a list of recent runs, with '21:03:19 - meta.Bagging' selected. The 'Status' panel at the bottom shows 'OK'.





Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **Bagging -P 100 -S 1 -num-slots 1 -I 10 -V weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0**

Test options

- Use training set
- Supplied test set [Set...](#)
- Cross-validation Folds 19
- Percentage split % 50

[More options...](#)

(Nom) class [▼](#)

Start Stop

Result list (right-click for options)

- 21:02:16 - meta.Bagging
- 21:02:25 - meta.Bagging
- 21:02:32 - meta.Bagging
- 21:02:40 - meta.Bagging
- 21:03:19 - meta.Bagging
- 21:04:04 - meta.Bagging
- 21:05:08 - meta.Bagging
- 21:05:18 - meta.Bagging
- 21:05:26 - meta.Bagging
- 21:07:13 - meta.Bagging
- 21:08:36 - meta.Bagging
- 21:08:57 - meta.Bagging
- 21:09:09 - meta.Bagging
- 21:09:20 - meta.Bagging
- 21:10:33 - meta.Bagging
- 21:10:40 - meta.Bagging**

Classifier output

```

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Time taken to build model: 0.05 seconds

==== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

==== Summary ===

Correctly Classified Instances           292          84.6377 %
Incorrectly Classified Instances        53           15.3623 %
Kappa statistic                         0.6927
Mean absolute error                     0.2189
Root mean squared error                 0.3508
Relative absolute error                  43.9611 %
Root relative squared error            69.0809 %
Total Number of Instances               345

==== Detailed Accuracy By Class ===

      TP Rate   FP Rate   Precision   Recall   F-Measure   MCC
      0,846     0,153     0,841      0,846    0,844     0,693
      0,847     0,154     0,851      0,847    0,849     0,693
Weighted Avg.   0,846     0,154     0,846      0,846    0,846     0,693

==== Confusion Matrix ===

      a     b  <- classified as
143   26 |  a = +
  27  149 |  b = -

```

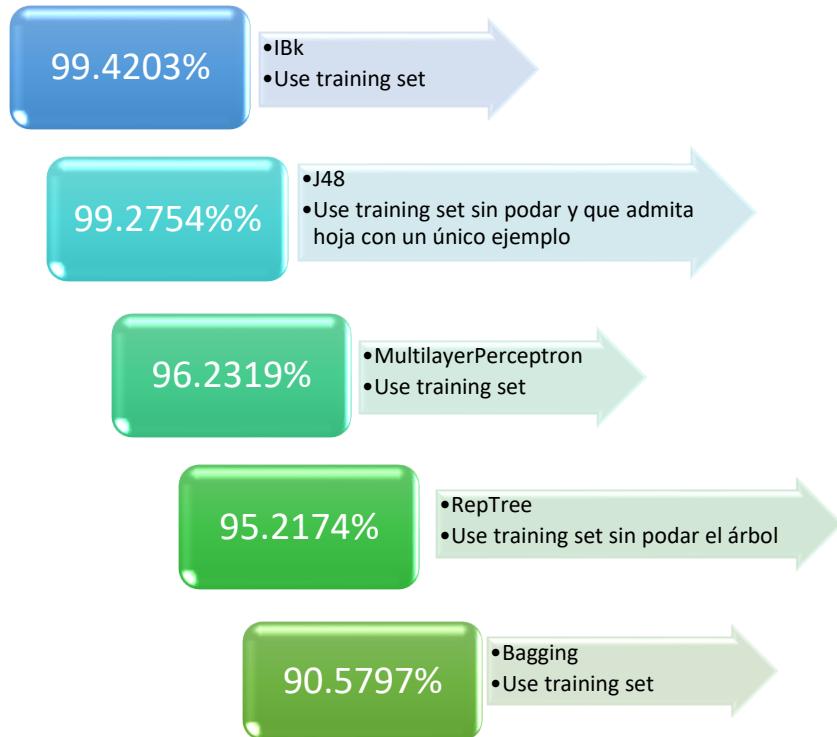
Status

OK [Log](#)  x 0

Gracias a la tabla vemos el orden a elegir para tener un rendimiento u otro.

TABLA RESUMEN MODIFICACIONES (Bagging)		% RENDIMIENTO DEL ÁRBOL
Percentage split 50%		84.6377%
Percentage split con valores predeterminados		85.5319%
Cross-validation con valores predeterminados		85.6522%
Percentage split 79%		86.2069%
Cross-validation, Folds = 19		86.5217%
Use training set		90.5797%

En resumen, estos top 5 de clasificadores con mayor rendimiento.



CREAR UN NUEVO CLASIFICADOR

Como Weka está orientado al desarrollo abierto y comunitario es sencillo programar nuevos clasificadores. Los pasos por seguir son:

1. En primer lugar es crear una nueva clase y situarla dentro de weka/classifiers/.
2. Una vez creada la clase, cualquier clasificador debe heredar de las clases:
 - import weka.classifiers.*
 - import weka.core.*
 - import weka.util.*
3. Debe extender la clase Classifier. Si el clasificador calcula la distribución de clases debe extender también de DistributionClassifier.
4. Todo clasificador debe tener los siguientes métodos:
 - void buildClassifier(Instances instancias) Este método debe implementar la construcción del clasificador obteniendo como parámetro las instancias. Este método debe inicializar todas las variables y nunca debe modificar ningún valor de las instancias.
 - float classifyInstance(Instance instancia) Clasifica una instancia concreta una vez que el clasificador ya está construido. Devuelve la clase en la que se ha clasificado o Instance.missingValue() si no se consigue clasificar.
 - double[] distributionForInstance(Instance instancia) Este método devuelve la distribución de una instancia en forma de vector. Si el clasificador no ha logrado clasificar la instancia dada devolverá un vector lleno de ceros. Si lo consigue y las clases son numéricas devolverá el valor obtenido y en los demás casos devolverá un vector con las probabilidades de cada elemento en cada clase
5. Y, en según qué circunstancias, deberá implementar las siguientes interfaces:
 - UpdateableClassifier Si el clasificador tiene un carácter incremental.
 - WeightedInstanceHandler Si el clasificador hace uso de los pesos de cada instancia

6. Es muy útil conocer las clases Instance e Instances que son aquellas que almacenan (siempre con double) todas las instancias, incluso las nominales que utilizan como índice la declaración del atributo y las indexa según este orden. Los métodos más útiles de la clase Instance (se encuentra en el paquete weka.core) son:

- Instances classAttribute() Devuelve la clase de la que procede el atributo.
- double classValue() Devuelve el valor de la clase del atributo.
- double value(int i) Devuelve el valor del atributo que ocupa la posición i.
- Enumeration enumerateAttributes() Devuelve una enumeración de los atributos que contiene esa instancia.
- double weight() Devuelve el peso de una instancia en concreto.

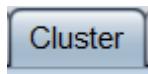
Y de la clase Instances:

- int numInstances() Devuelve el número de instancias que contiene el conjunto de datos.
- Instance instance(int) Devuelve la instancia que ocupa la posición i.
- Enumeration enumerateInstances() Devuelve una enumeración de las instancias que posee el conjunto de datos.

AGRUPACIÓN (CLUSTER)

En Cluster se proporcionan varios algoritmos de agrupación en clústeres, como SimpleKMeans, FilteredClusterer, HierarchicalClusterer, etc. Es una tarea descriptiva que pretende como su nombre indica agrupar los individuos disponibles en grupos o clusters de comportamiento similar. A diferencia de la tarea de clasificación y las distintas técnicas utilizadas en ella , en el análisis cluster (en sentido estricto) y en otras técnicas de agrupación, los grupos son desconocidos a priori y son precisamente lo que queremos

determinar. Seleccionamos la casilla



SimpleKMeans

Puede usar la distancia euclídea (predeterminada) o la distancia de Manhattan. Si se utiliza la distancia de Manhattan, los centroides se calculan como la mediana por componentes en lugar de como la media. Permite ejecutar este análisis. Entre sus principales opciones tenemos la determinación del número de cluster a construir (por defecto 2) y la elección de la función de distancia a considerar.

Las capturas que vamos a ver a continuación por defecto agrupa los datos en dos conjuntos.

Si seleccionamos Use training set en Cluster mode con los valores predeterminados vemos que hay 525 (76%) instancias para el grupo 0 y 165 (24%) instancias para el otro grupo. El tiempo necesario para crear el modelo (datos de entrenamiento completos) es de 0,12 segundos. El número de interacciones es 5.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance"

Cluster mode

- Use training set
- Supplied test set Set...
- Percentage split % 66
- Classes to clusters evaluation (Nom) class
- Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

22:38:22 - SimpleKMeans

Clusterer output

Attribute	Full Data	0	1
	(690.0)	(525.0)	(165.0)
A1	b	b	b
A2	31.5682	32.2719	29.3289
A3	4.7587	4.9669	4.0963
A4	u	u	y
A5	g	g	p
A6	c	c	c
A7	v	v	v
A8	2.2234	2.3545	1.8062
A9	t	t	f
A10	f	f	f
A11	2.4	2.7105	1.4121
A12	f	f	f
A13	g	g	g
A14	184.0148	182.3889	189.1881
A15	1017.3855	997.2495	1081.4545
class	-	-	-

Time taken to build model (full training data) : 0.12 seconds

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

Clustered Instances

0	525 (76%)
1	165 (24%)

weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

More Capabilities

canopyMaxNumCanopiesToHoldInMemory 100

canopyMinimumCanopyDensity 2.0

canopyPeriodicPruningRate 10000

canopyT1 -1.25

canopyT2 -1.0

debug False

displayStdDevs False

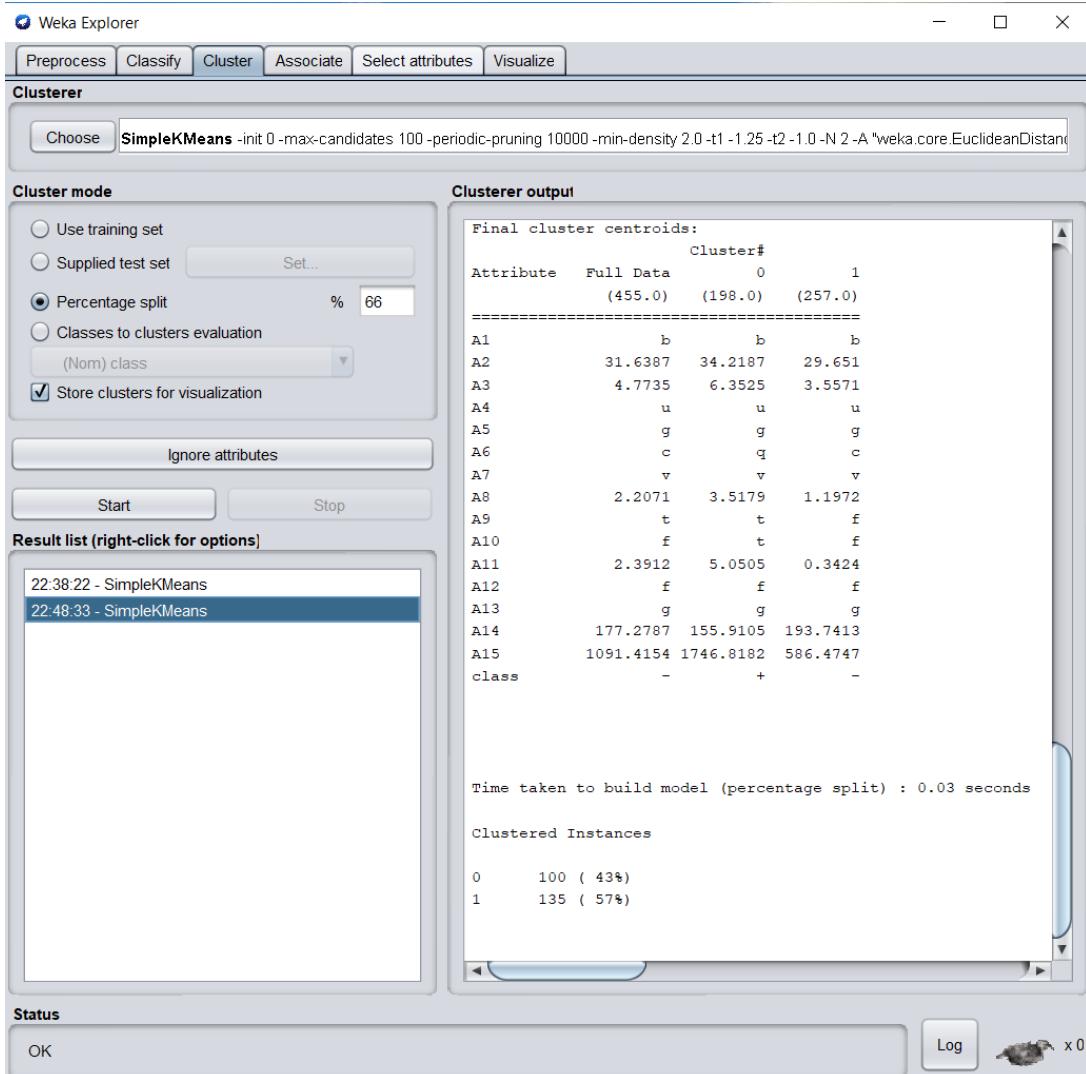
distanceFunction Choose EuclideanDistance -R first-last

doNotCheckCapabilities False

dontReplaceMissingValues False

Open... Save... OK Cancel

Si por el contrario seleccionamos Percentage split en Cluster mode y dejamos los valores predeterminados. Se parecen los números más. Es decir en el grupo 0 hay 100 instancias (43%) y en el grupo 1 hay 135 instancias (57%). Este modo tarda menos tiempo, en concreto 0.03 segundos. El número de interacciones es 4.



Si modificamos manualmente el % de Percentage split y lo ponemos a 79% los números se desestabilizan un poco y queda un 39% (56 instancias) para el grupo 0 y un 61% (89 instancias) en el grupo 1. Aquí el número de interacciones es 5.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance"

Cluster mode

- Use training set
- Supplied test set Set...
- Percentage split % 79
- Classes to clusters evaluation (Nom) class
- Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

- 22:38:22 - SimpleKMeans
- 22:48:33 - SimpleKMeans
- 22:49:07 - SimpleKMeans

Clusterer output

```

Final cluster centroids:
                         Cluster#
Attribute   Full Data      0       1
                  (545.0) (232.0) (313.0)
=====
A1           b       b       b
A2           31.9612  35.5018  29.3368
A3           4.8549   6.6725   3.5076
A4           u       u       u
A5           g       g       g
A6           c       c       c
A7           v       v       v
A8           2.2974   4.0372   1.0078
A9           t       t       f
A10          f       t       f
A11          2.3541   5.0043   0.3898
A12          f       t       f
A13          g       g       g
A14          183.1327 164.6649 196.8213
A15          988.3908 1976.9569 255.6518
class        -       +       -

```

Time taken to build model (percentage split) : 0.02 seconds

Clustered Instances

0	56 (39%)	
1	89 (61%)	

Status OK Log x 0

Elegimos Classes to clusters Evaluation y por defecto elegimos (Nom) class que es el último atributo de los datos. Vemos como las instancias agrupadas incorrectamente son 297.0 es decir, 43.0435 %. Para cluster 0 se asigna + y para cluster 1 se le asigna -. Nos dice que de los 307 ejemplos con clasificación +, el árbol clasifica 264 como + y 43 como -. Análogamente, de los 383 ejemplos con clasificación -, el árbol clasifica 129 como - y 254 como +. También podemos ver como 518 instancias pertenecen al grupo 0 y 172 instancias al grupo 1.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance"

Cluster mode

- Use training set
- Supplied test set Set...
- Percentage split % 79
- Classes to clusters evaluation
 - (Nom) class
- Store clusters for visualization

Clusterer output

```

A10          f          f          f
A11          2.4        2.8108    1.1628
A12          f          f          f
A13          g          g          g
A14          184.0148   182.6258   188.1979
A15          1017.3855  998.2104   1075.1337

```

Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

	0	1	(%)
0	518	172	75%
1	172	518	25%

Class attribute: class
 Classes to Clusters:

```

0   1  <-- assigned to cluster
264  43 | +
254  129 | -

```

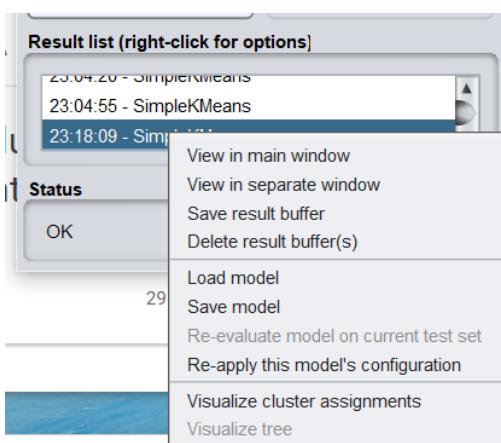
Cluster 0 <-- +
 Cluster 1 <-- -

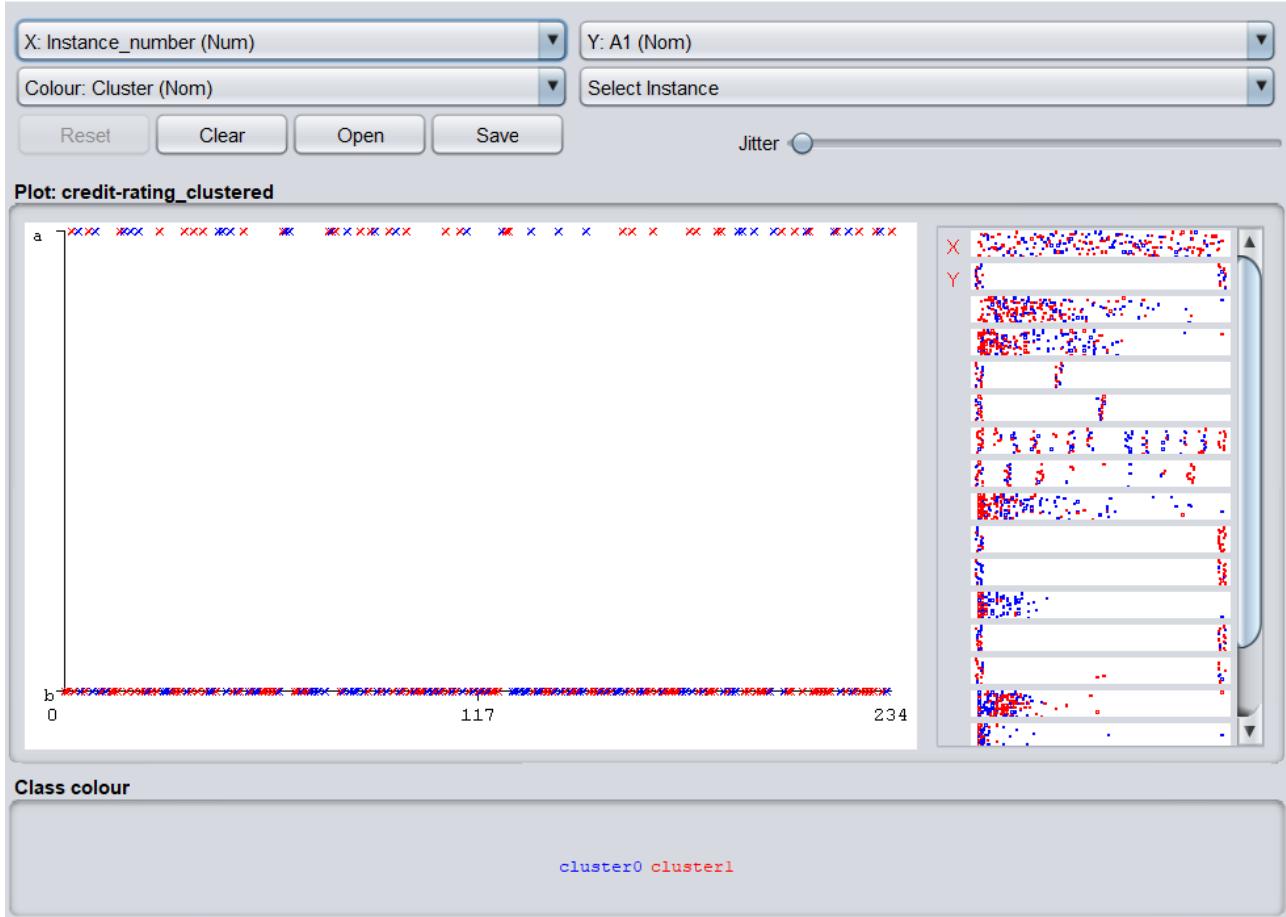
Incorrectly clustered instances : 297.0 43.0435 %

Status

OK Log x 0

De manera que la mejor agrupación sería Percentage split con los valores predeterminados ya que es lo más cercano a tener 50% instancias para cada grupo. Vamos a visualizar las asignaciones de clústeres dando clic derecho en el SimpleKMeans en la sección de Result list.

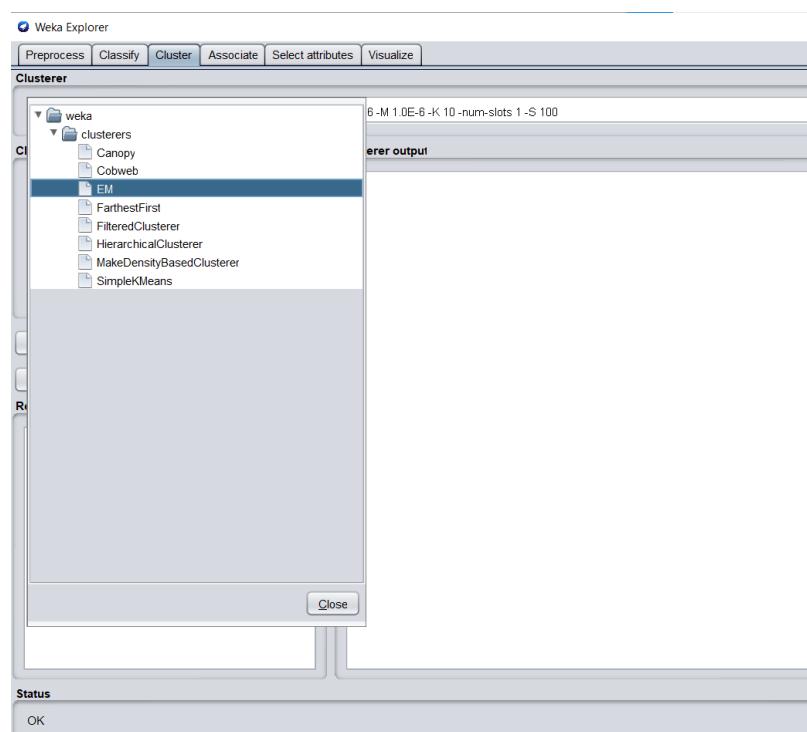




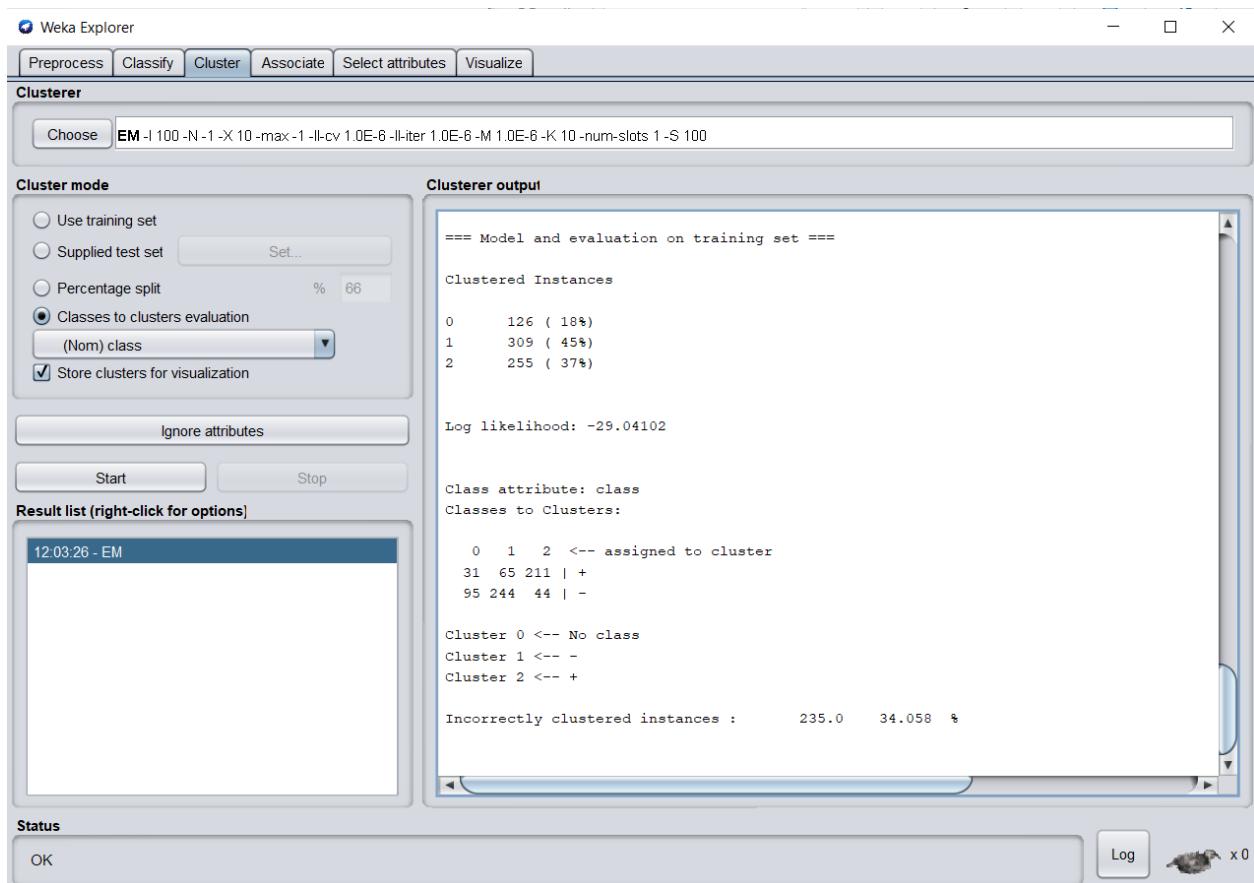
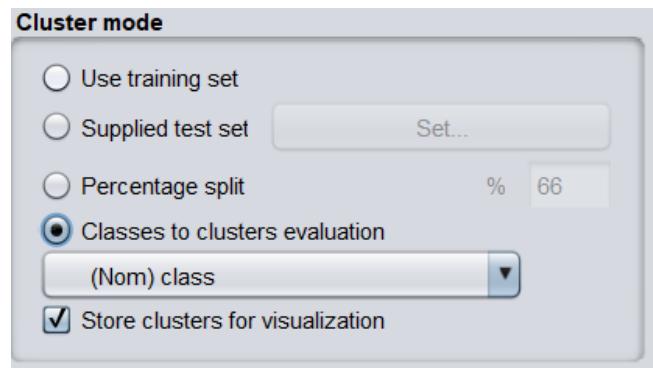
EM

Esto lo que hace es buscar grupos de instancias similares en todo el conjunto de datos. Tenemos distintos algoritmos de agrupación en clústeres como EM, FilteredClusterer, HierarchicalClusterer, SimpleKMeans.....

Choose → weka → clusterers → EM



En la ventana Cluster mode seleccionamos Classes to clusters Evaluation y le damos a star.



En la salida del procesamiento vemos como hay 3 instancias agrupadas detectadas en la base de datos. El cluster 0 no tiene ninguna clase asociada. También se puede ver la media y la desviación estándar de cada uno de los atributos en los distintos grupos asociados.

weka explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose EM -l 100 -N 1 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Cluster mode

- Use training set
- Supplied test set Set...
- Percentage split % 66
- Classes to clusters evaluation (Nom) class
- Store clusters for visualization

Clusterer output

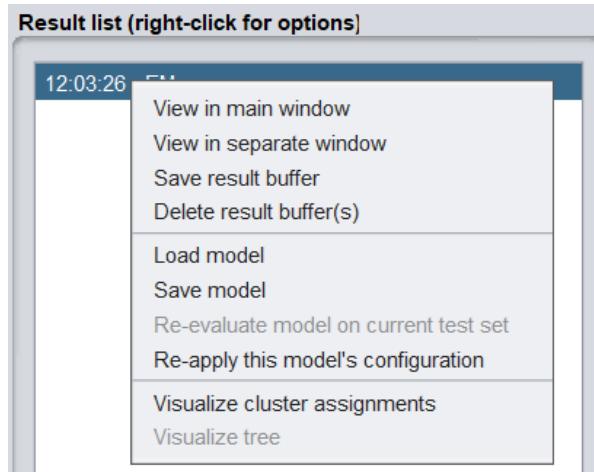
Number of clusters selected by cross validation: 3
Number of iterations performed: 17

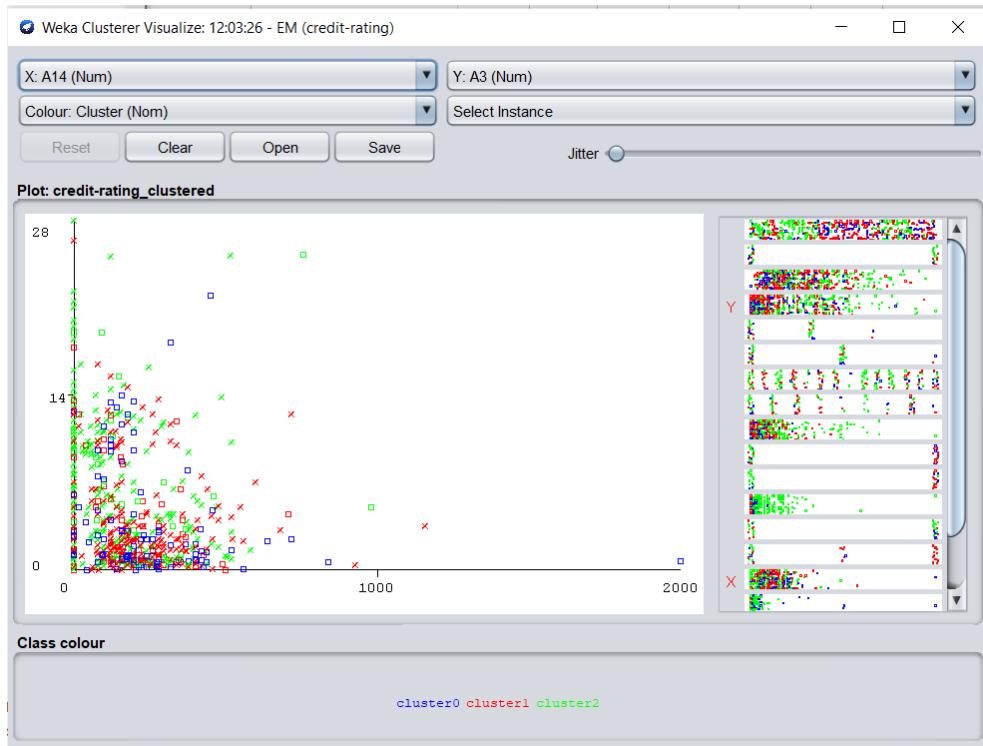
Attribute	Cluster		
	0 (0.17)	1 (0.44)	2 (0.39)
A1			
b	85.0768	220.1794	177.7438
a	32.8488	84.1394	96.0118
[total]	117.9256	304.3189	273.7556
A2			
mean	27.0788	29.875	35.3668
std. dev.	7.4568	10.3973	13.639
A3			
mean	4.0399	3.6176	6.3348
std. dev.	4.5819	3.9009	5.7267
A4			
u	79.8193	208.7045	239.4762
y	37.1063	94.6251	34.2686
l	2	1.9892	1.0108
t	1	1	1
[total]	119.9256	306.3189	275.7556
A5			
g	79.8193	208.7045	239.4762
p	37.1063	94.6251	34.2686
gg	2	1.9892	1.0108
[total]	118.9256	305.3189	274.7556
A6			
c	25.7493	68.5125	54.7382
d	3.907	21.9652	7.1277

Status

OK

Para ver la representación visual de los clusters el ratón se coloca encima de EM de result list y clic derecho. Y selecciona visualize clusters assignments.

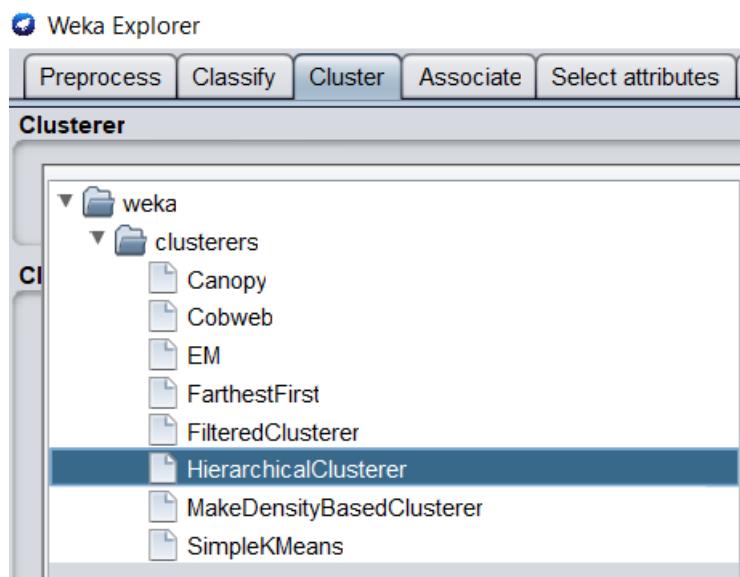


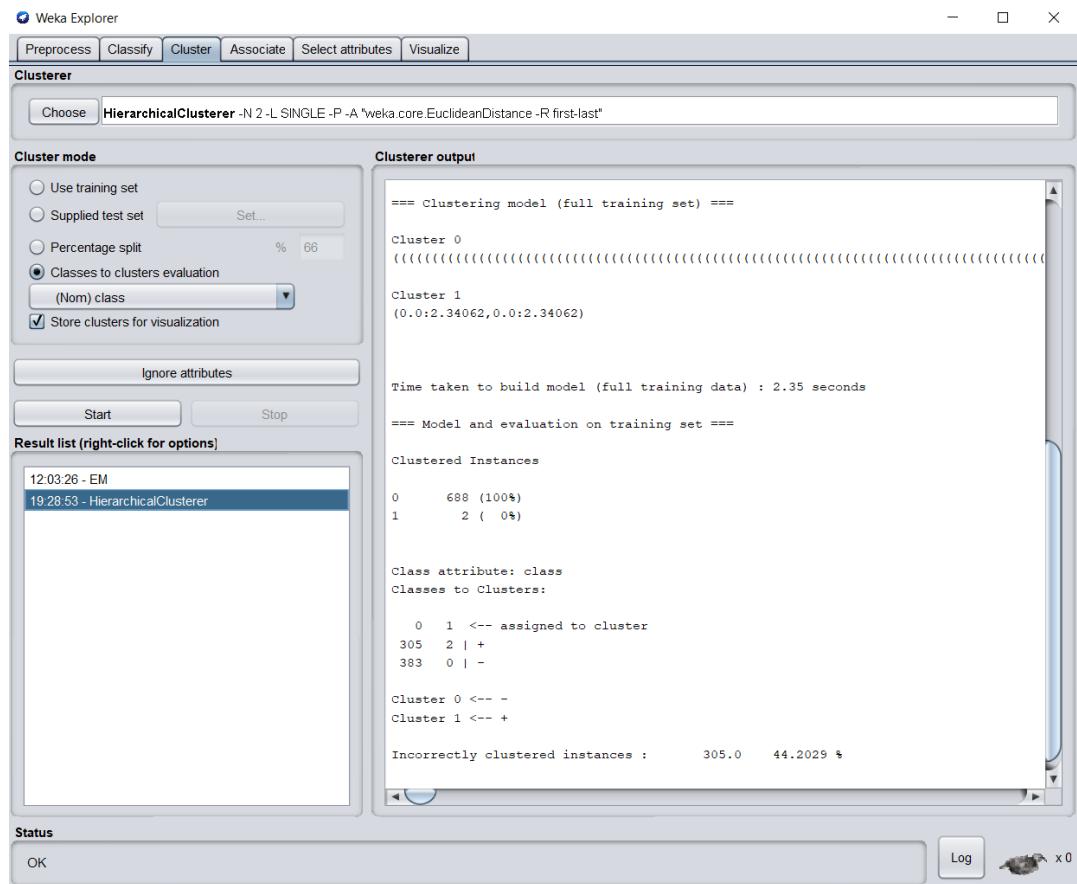


Sucede lo mismo de antes para analizar los resultados podemos cambiar los ejes X e Y de dos formas. O en la parte derecha de la captura o con las barras del lateral (clic derecho para Y , clic izquierdo para X)

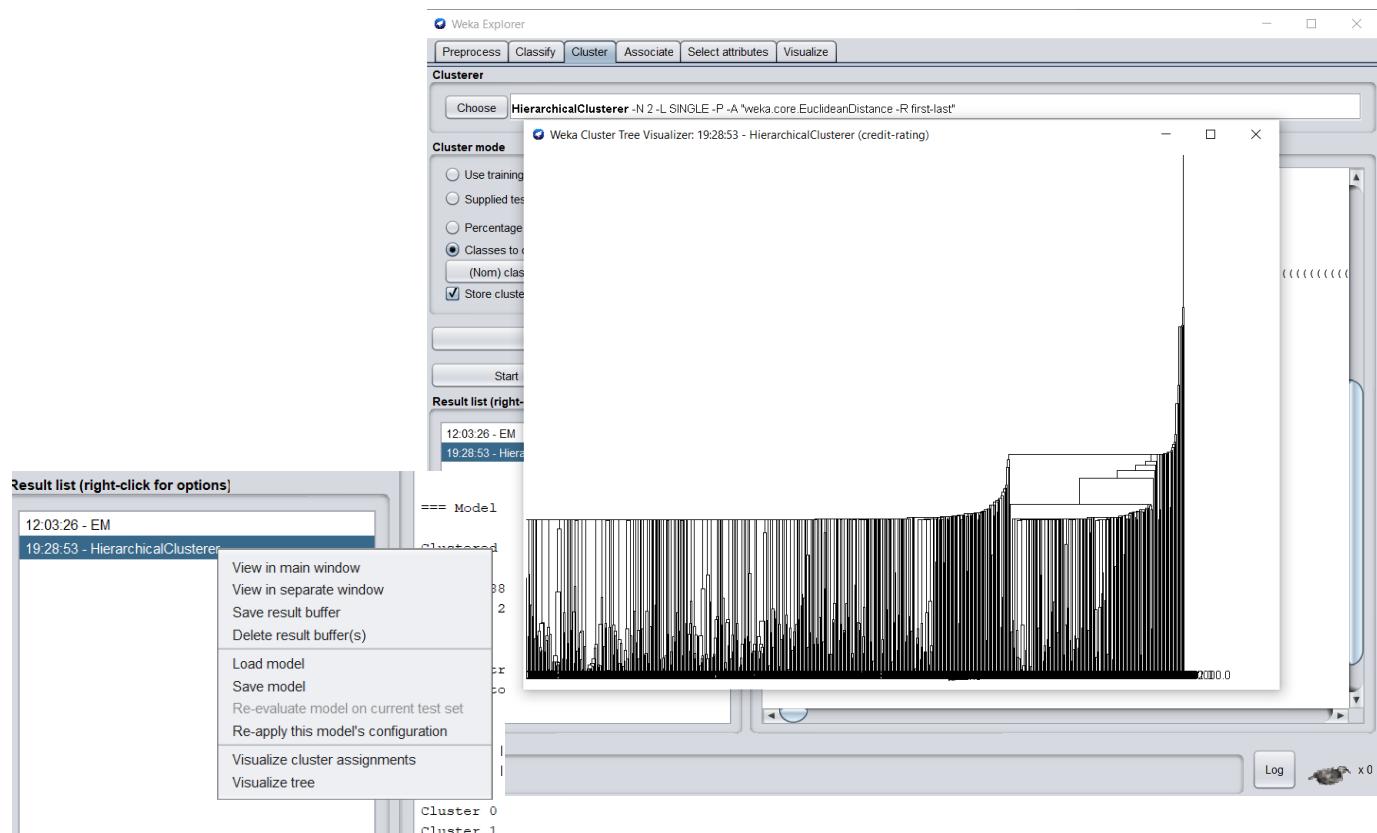
HIERARCHICALCLUSTERER

Este es otro algoritmo de agrupamiento. Le damos al botón Choose > HierarchicalClusterer y en la sección de Cluster mode seleccionamos Classes to cluster Evaluation.





Ahora vamos a ver el árbol producido por este algoritmo. Clic derecho en HierarchicalClusterer y visualize tree.



ASOCIACIÓN (ASSOCIATE)

Asociaciones entre x e y , siendo x e y cualquier cosa. Permiten la búsqueda automática de reglas que relacionan conjuntos de atributos entre sí. Son algoritmos no supervisados, en el sentido de que no existen relaciones conocidas a priori con las que contrastar la validez de los resultados, sino que se evalúa si esas

Associate

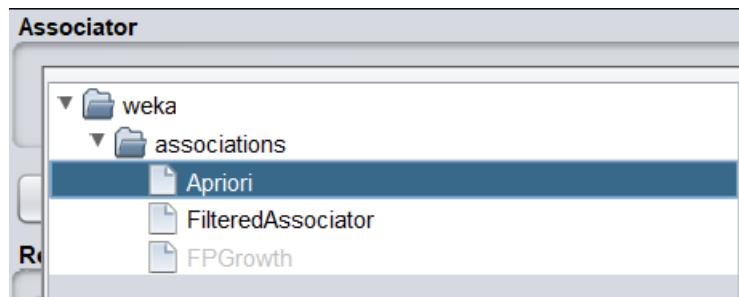
reglas son estadísticamente significativas . Seleccionamos la ventana

Se encontrará Apriori, FilteredAssociator y FPGrowth.

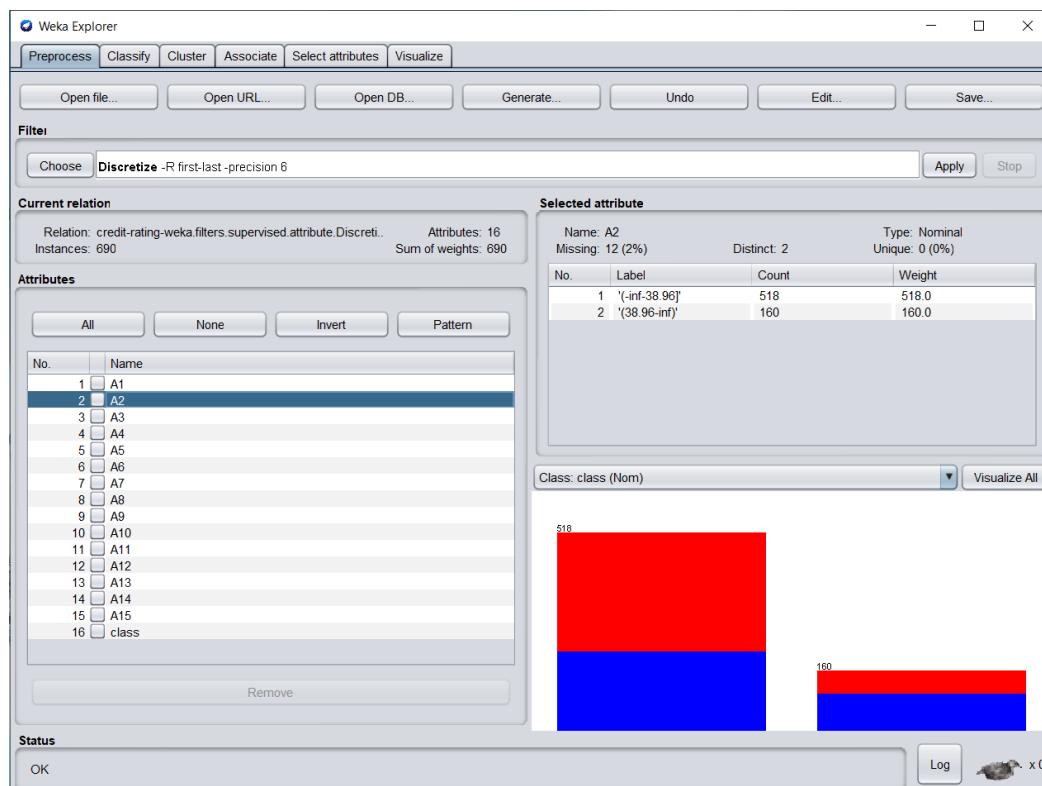
APRIORI

Es el principal algoritmo de asociación implementado en WEKA. Algoritmo en ML que muestra las asociaciones probables y crea reglas de asociación. Es decir, busca reglas entre atributos simbólicos, razón por la que se requiere haber discretizado todos los atributos numéricos.

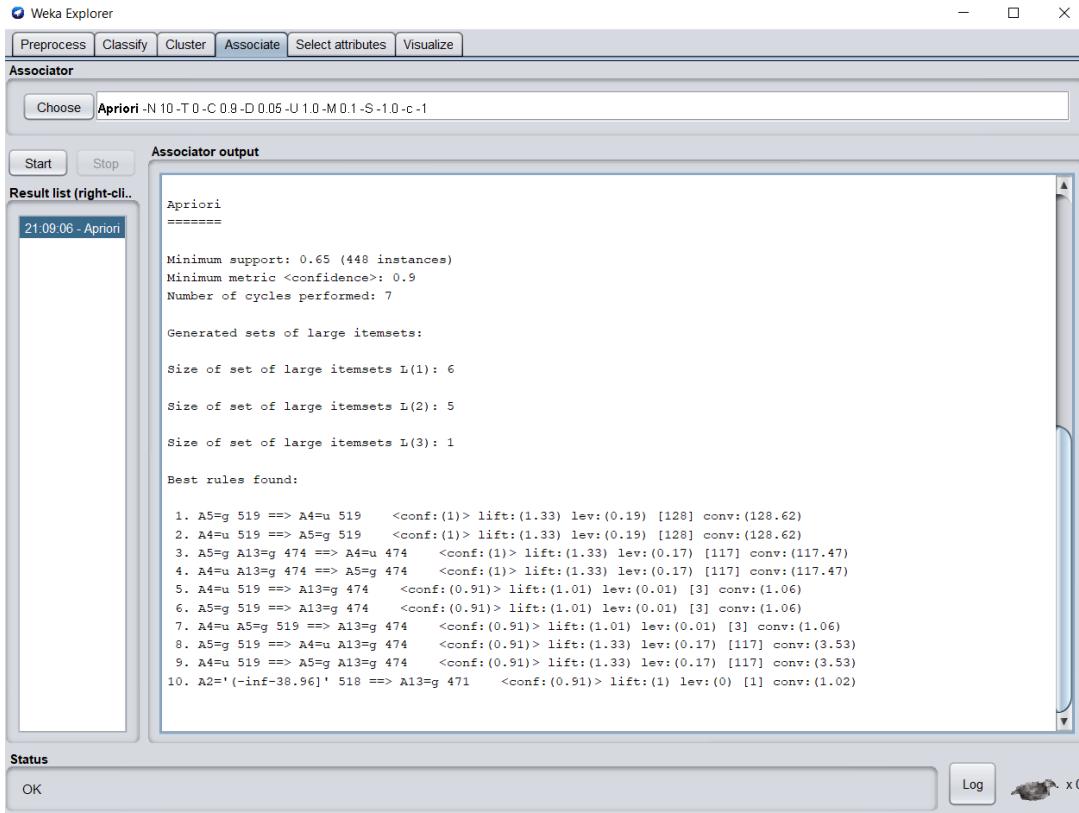
Le damos a choose para seleccionar el algoritmo de asociación > associations > Apriori



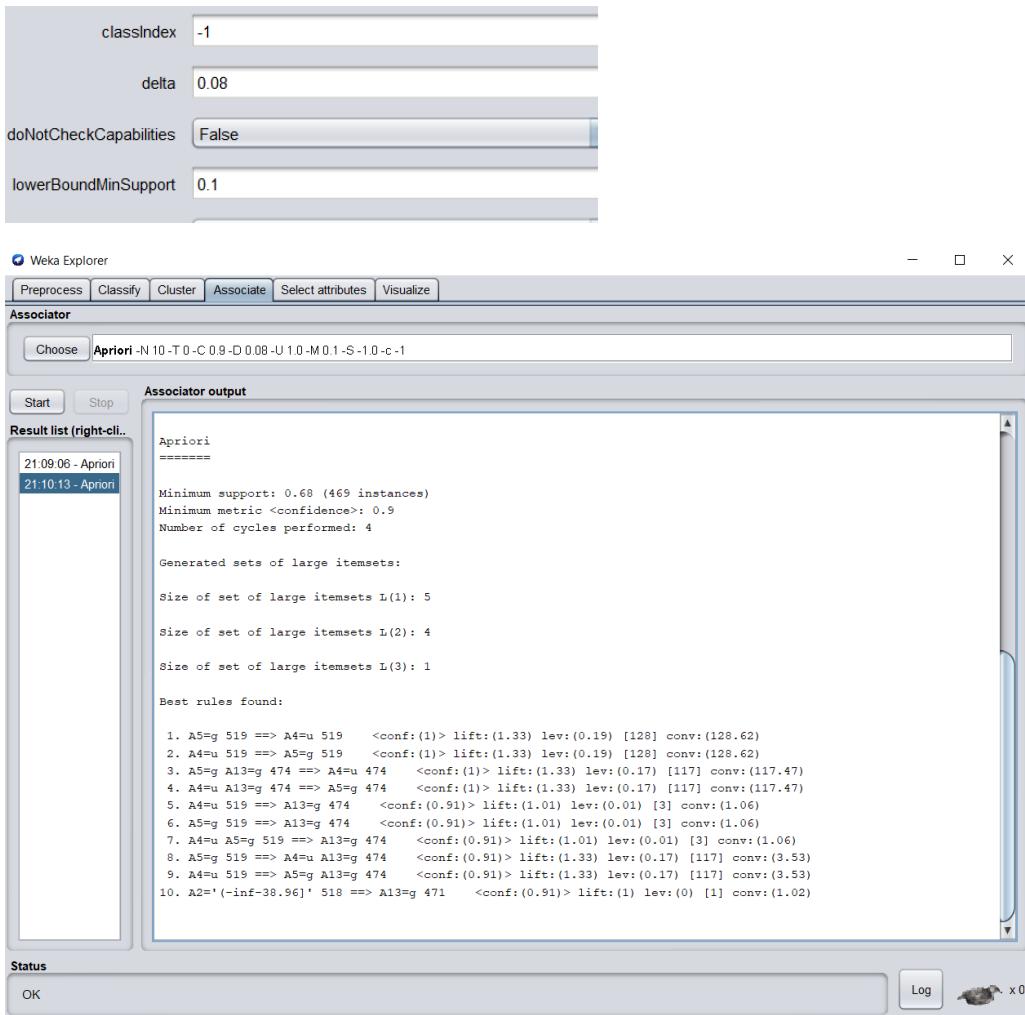
Antes de esto en Preprocess en Choose hemos elegido usar el filtro NumericToNominal pero vamos a elegir discretize porque es más adecuado. . Vemos como el atributo A2 que era de tipo Numeric pasa a ser de tipo Nominal.



Ahora le damos a star y obtenemos la siguiente captura. En la cual vemos que el número de ciclos realizados es 7 y la métrica mínima de confianza es 0.9. Y el soporte mínimo son 448 instancias.

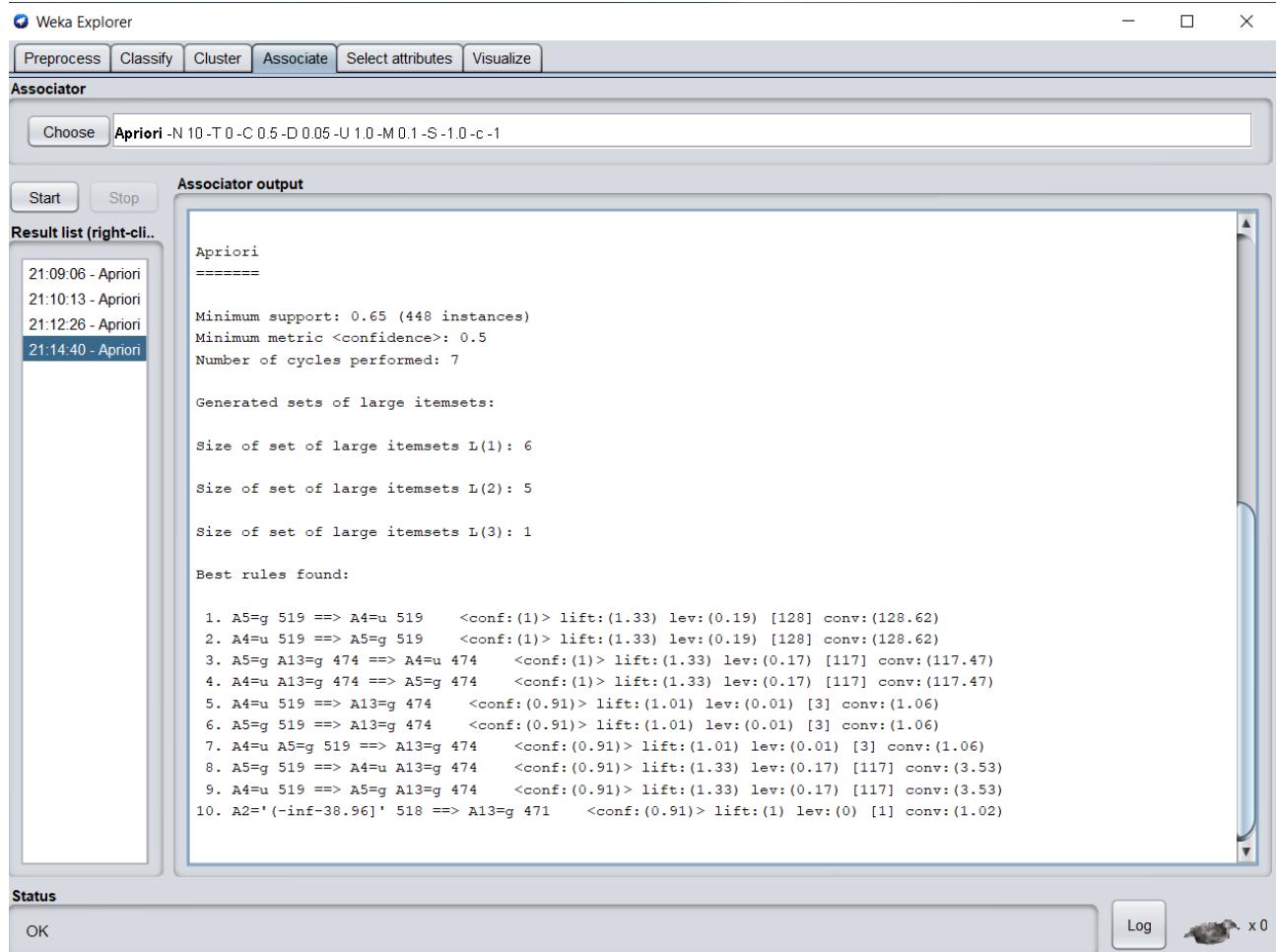


Si por otra parte hacemos clic en el filtro y cambiamos el parámetro delta de 0.05 a 0.08. Su función es disminuir iterativamente el apoyo por este factor. El soporte mínimo son 469 instancias. El número de ciclos realizados es 4. En lo último de la captura vemos las mejores reglas que ha encontrado.



Si por el contrario disminuyo delta a 0.01 el soporte mínimo se queda igual que el anterior es de 469 instancias y 32 ciclos realizados .

Si ahora modificamos y hacemos que el umbral mínimo aceptable para una regla sea el 50% en vez del 90% que sale como predeterminado. (minMetric = 0.5) Vemos lo que ocurre. El número de ciclos realizados es 7 y el soporte mínimo son 448 instancias.



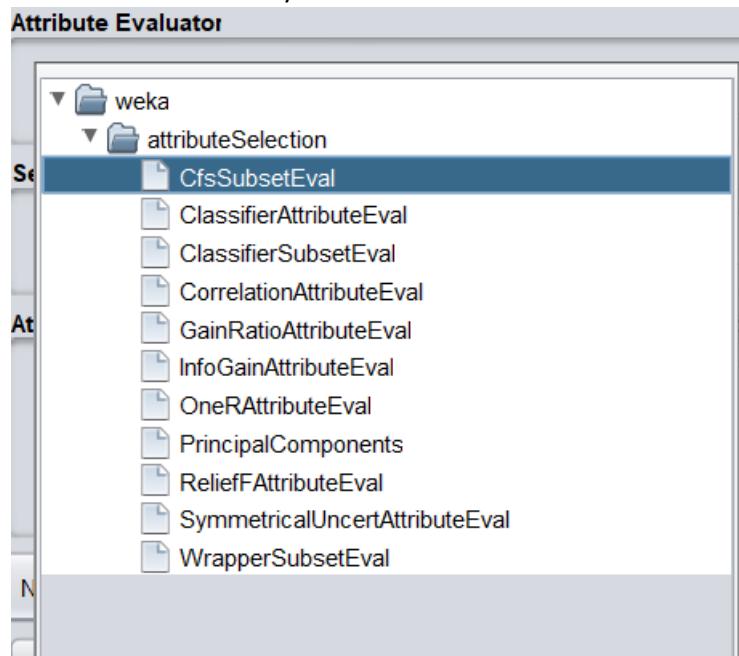
SELECCIÓN DE FUNCIONES (Select attributes)

En Select attributes Permite seleccionar funciones basadas en varios algoritmos como ClassifierSubsetEval, PrincipalComponents, etc.

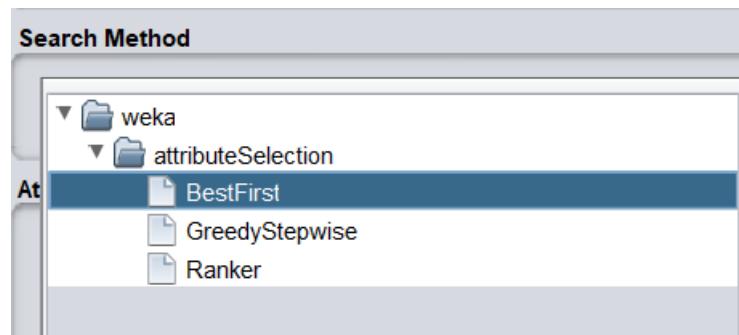
Nos ayuda a eliminar varios atributos no significativos (irrelevantes) en el análisis para el buen desarrollo de

un buen modelo de aprendizaje automático. Nos vamos a la ventana a la parte

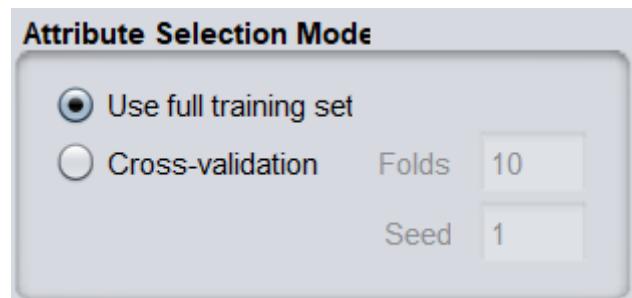
Attribute Evaluator y le damos a Choose > CfsSubsetEval



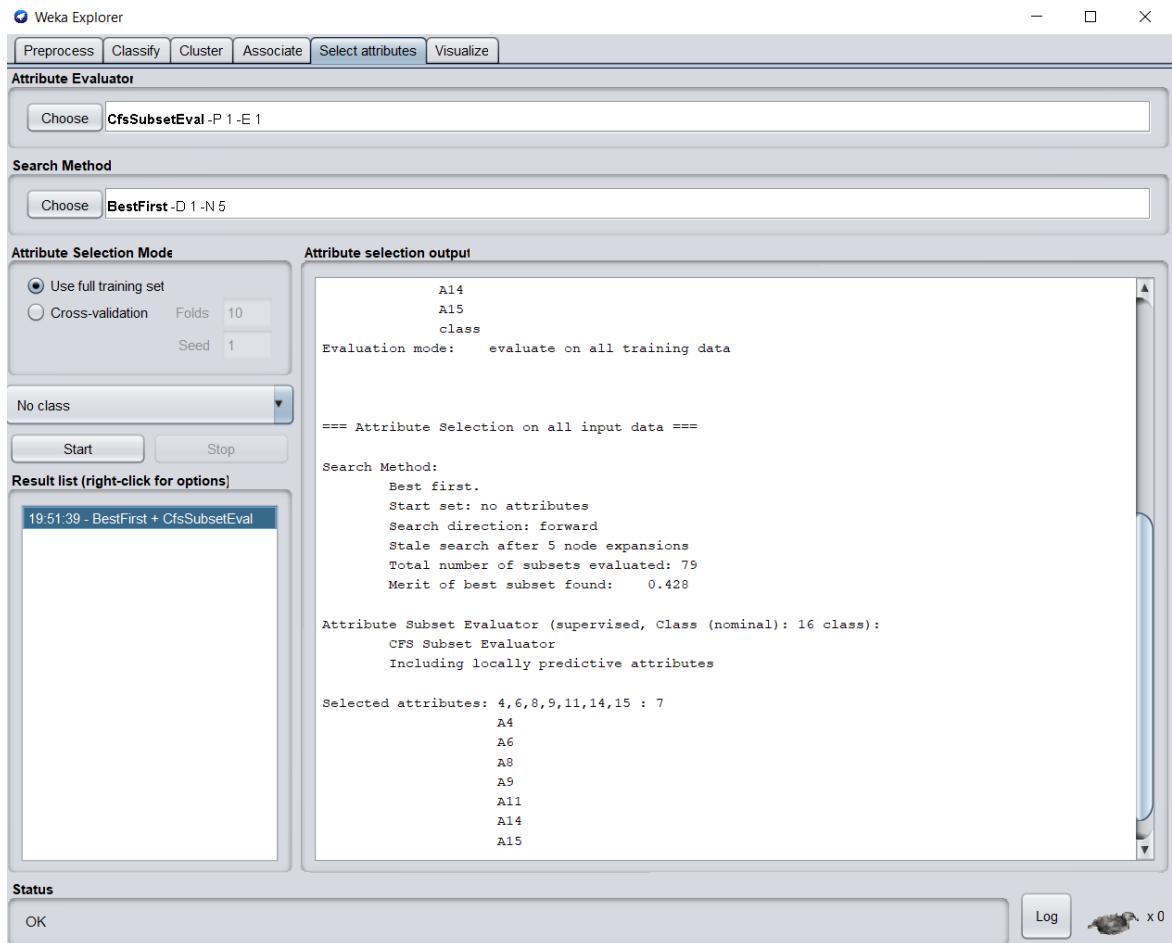
En la parte Search Method > BestFirst



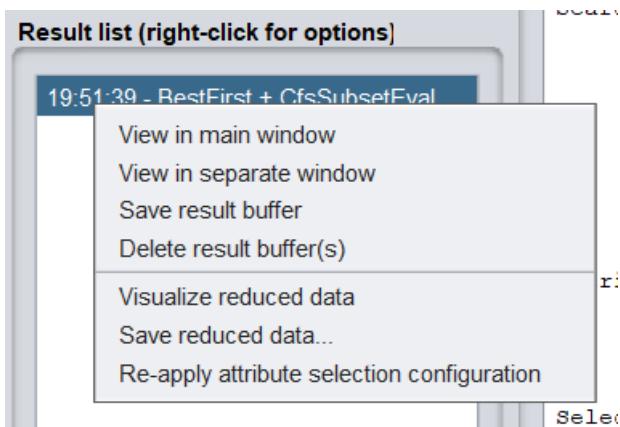
Y por último en attribute Selection Mode a use full training set

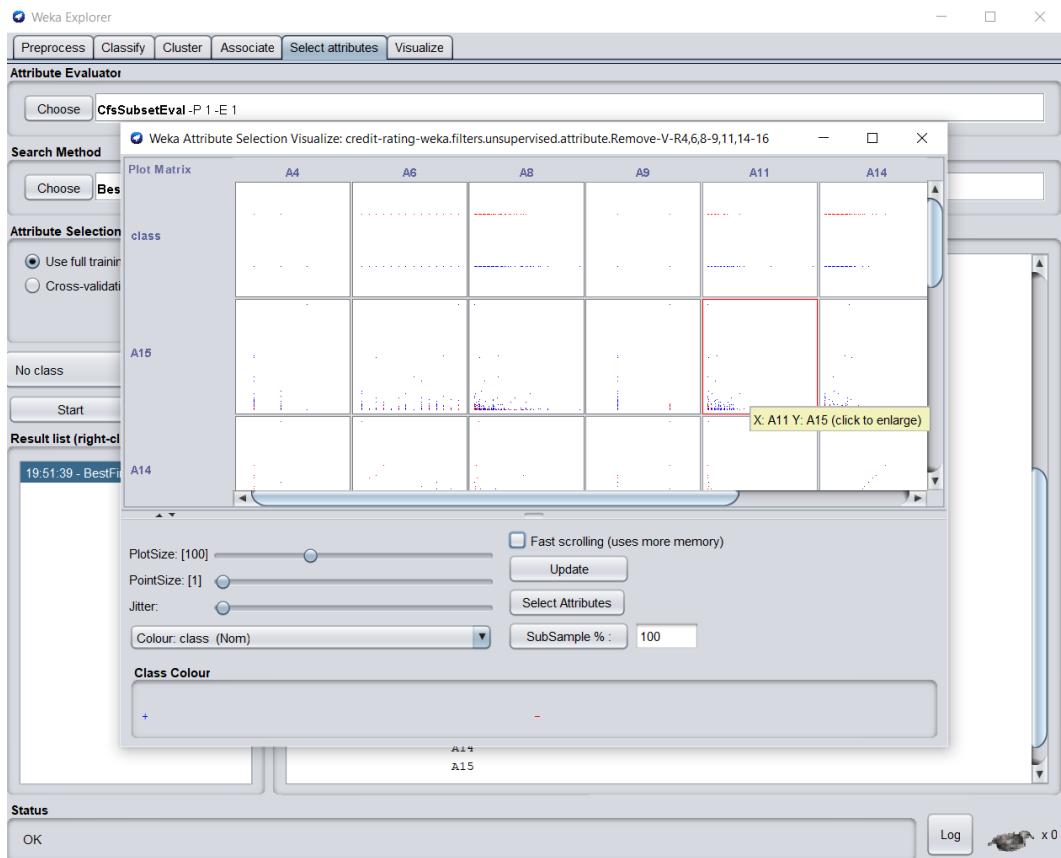


Podemos como en todos los casos anteriores cambiar los valores dando clic o dejar los valores predeterminados. Le damos a star y nos sale la ventana de resultados. Donde vemos que nos da la lista de atributos (Selected attributes)



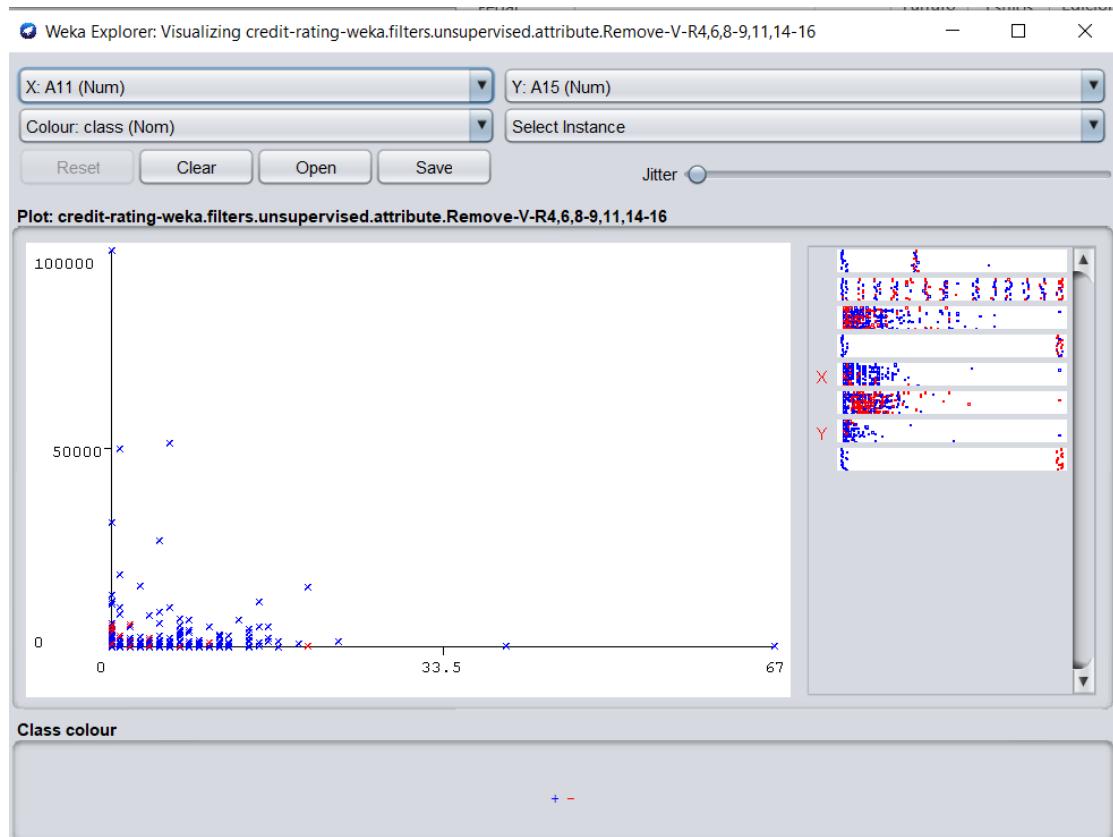
Hacemos clic derecho a lo seleccionado en la captura y seleccionamos visualize reduced data.





Las capturas anteriores nos muestran la representación visual de lo obtenido anteriormente. Al darle clic a alguno de los cuadrados nos sale otra ventana donde se ve la gráfica de los datos para después analizarlos. Yo he seleccionado un cuadrado (borde rojo) y esto es lo que nos sale.

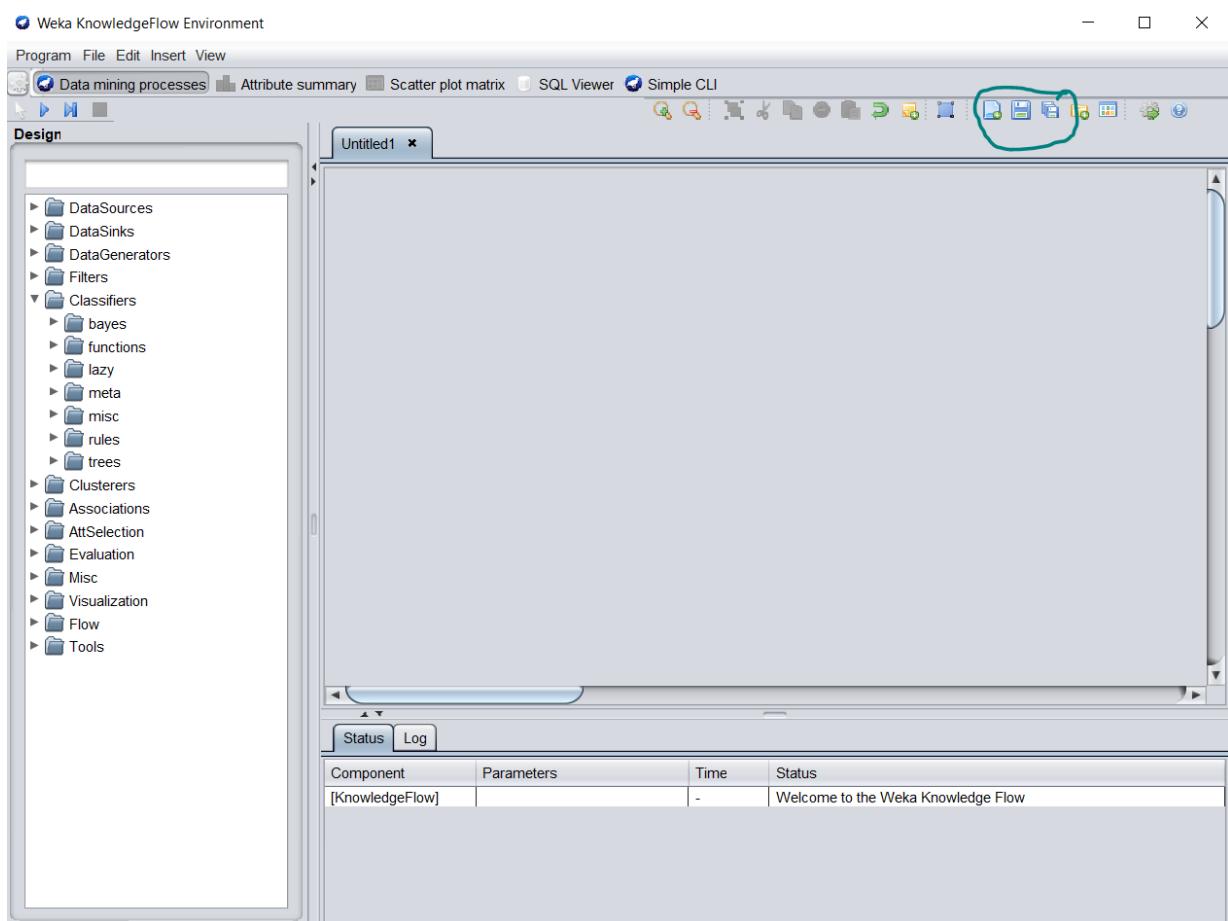
Como anteriormente hemos dicho cambiando las diferentes opciones de X e Y se analizaran los resultados.



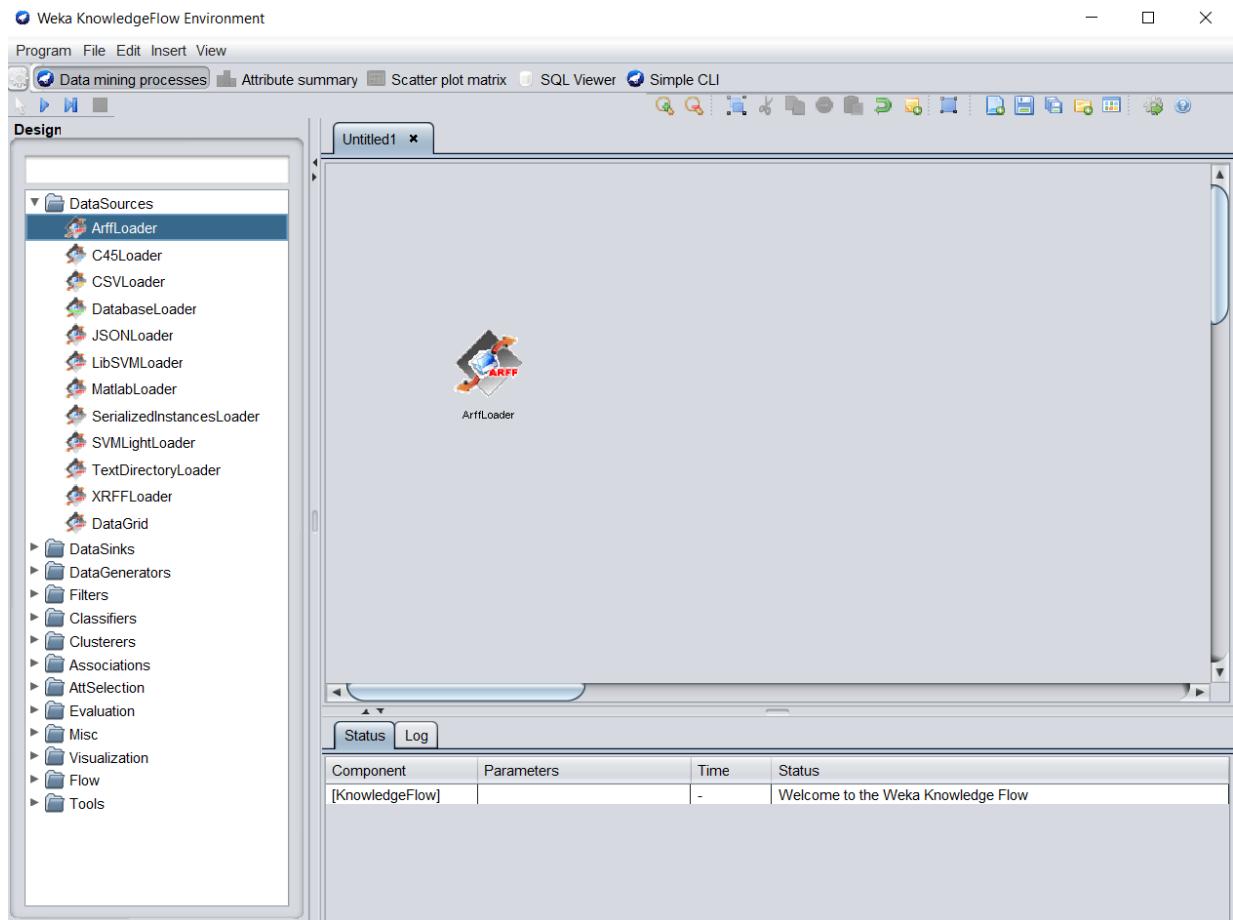
KNOWLEDGEFLOW

Knowledge Flow es una interfaz de Weka que muestra de una forma más explícita el funcionamiento interno del programa. Este entorno admite esencialmente las mismas funciones que el Explorador, pero con una interfaz de arrastrar y soltar (manera gráfica). Una ventaja es que apoya el aprendizaje incremental.

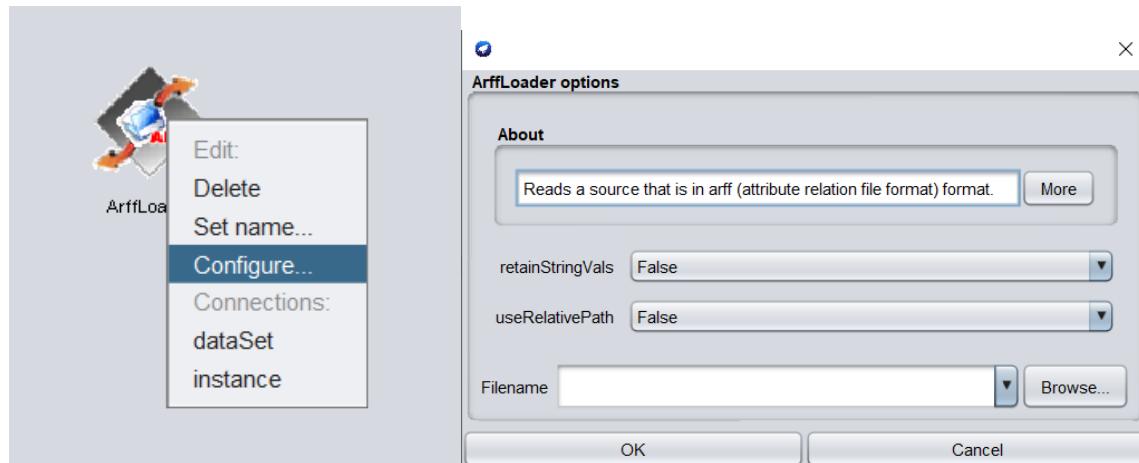
Se puede seleccionar componentes WEKA de una paleta, colocarlos en un lienzo de diseño y conectarlos para formar un flujo de conocimiento para procesar y analizar datos. En la actualidad, todos los clasificadores, filtros, agrupadores, asociadores, cargadores y ahorradores de WEKA están disponibles en KnowledgeFlow junto con algunas herramientas adicionales.



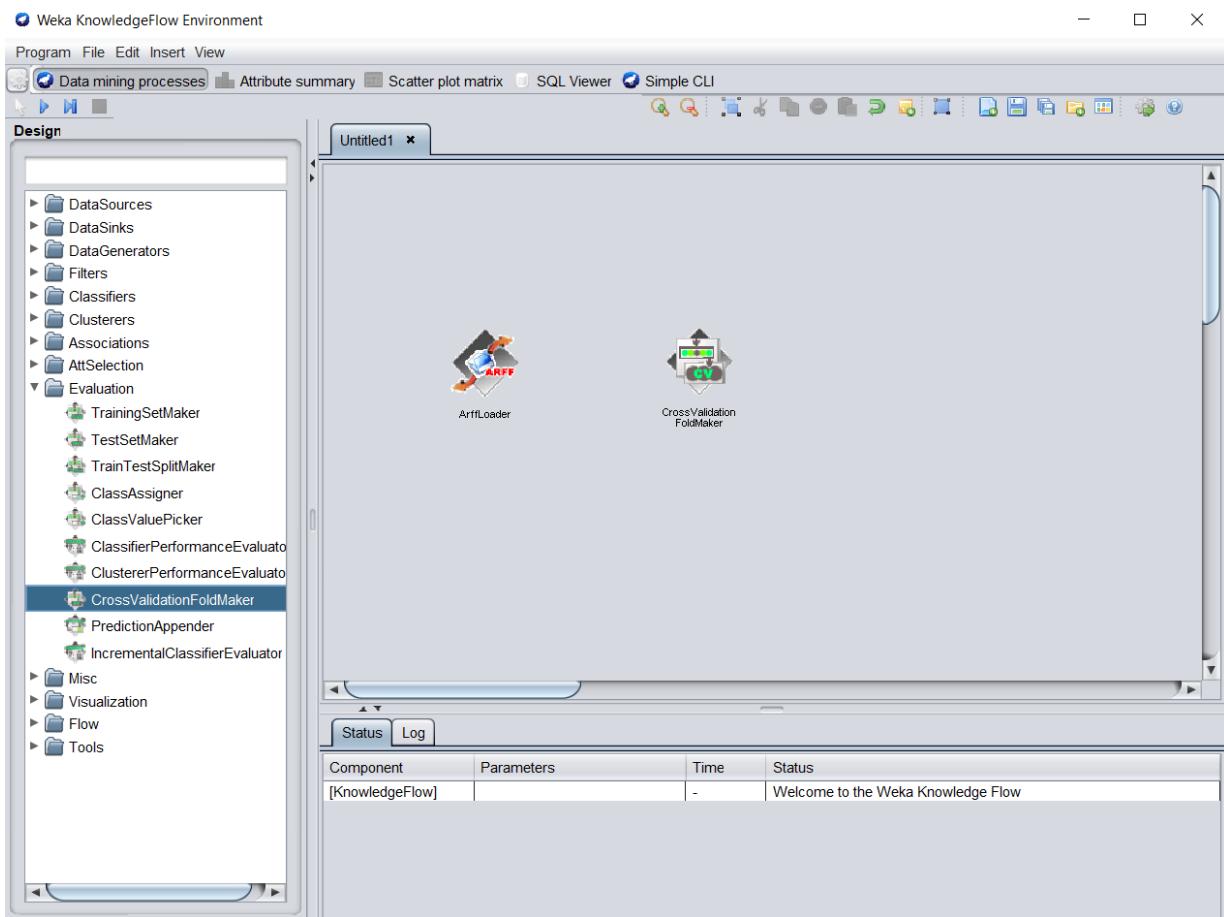
En la captura anterior en la parte superior derecha en un círculo vemos las funciones de borrar, salvar y cargar layout.



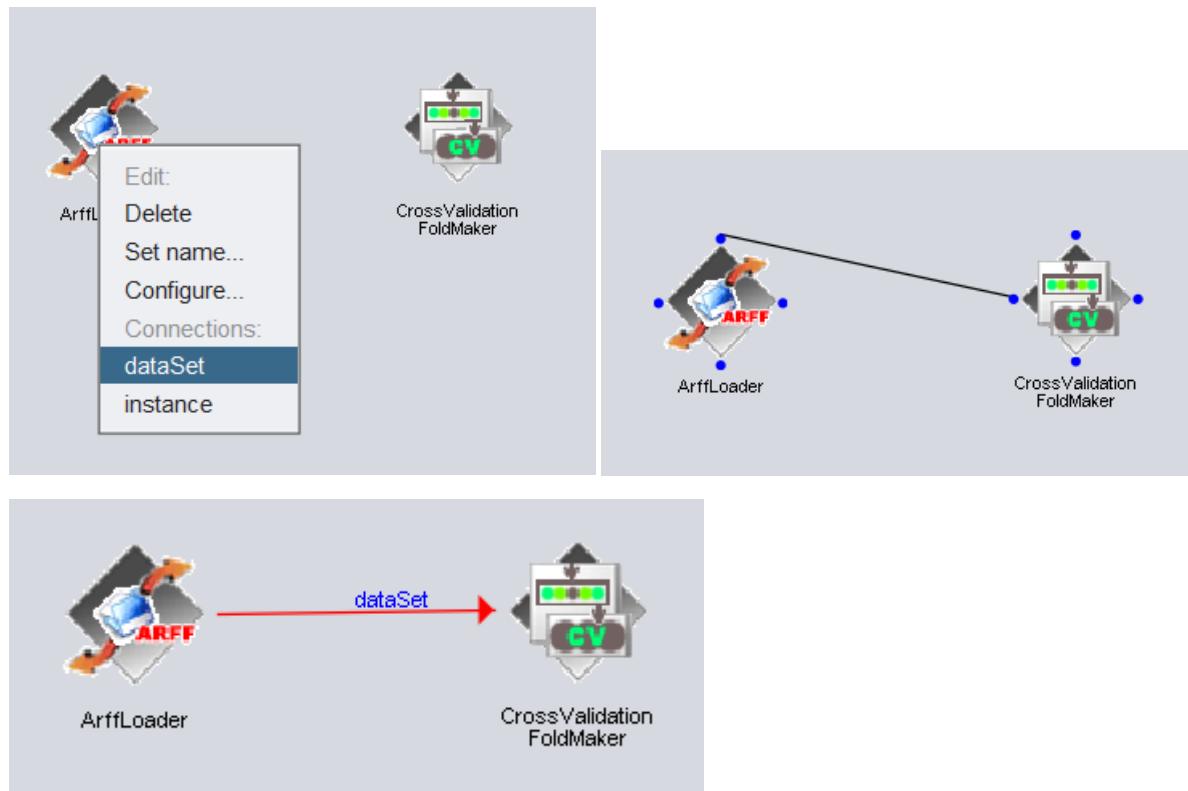
En la columna Design seleccionamos DataSources y hacemos clic con el botón izquierdo a ArffLoader soltamos y hacemos clic otra vez con el botón izquierdo en el lugar donde queremos colocarlo. Para configurarlo hacemos clic derecho encima y le damos a Configure.



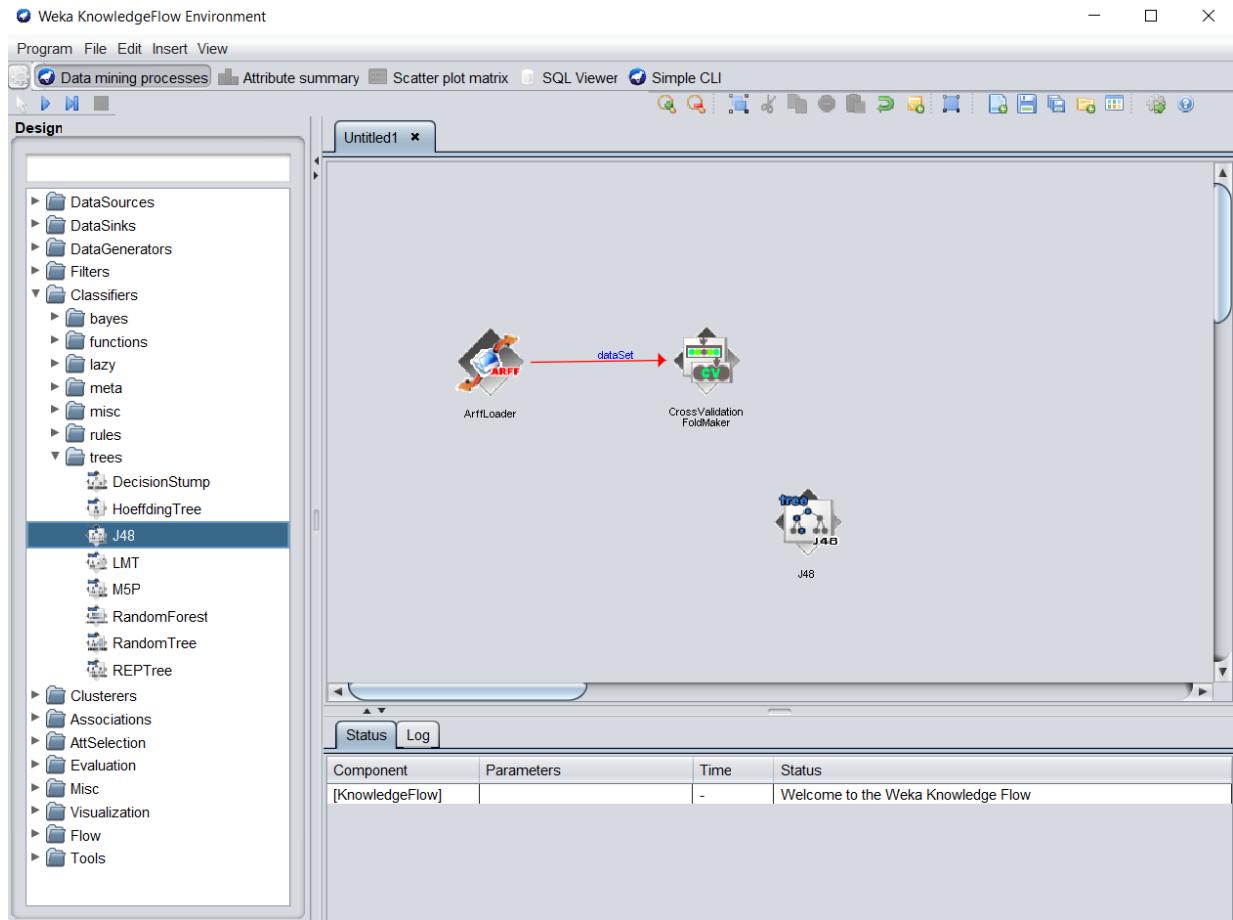
A continuación seleccionamos en la columna de Design Evaluation y CrossValidationFoldMaker. Hacemos clic con el botón izquierdo como antes y lo colocamos donde queramos.



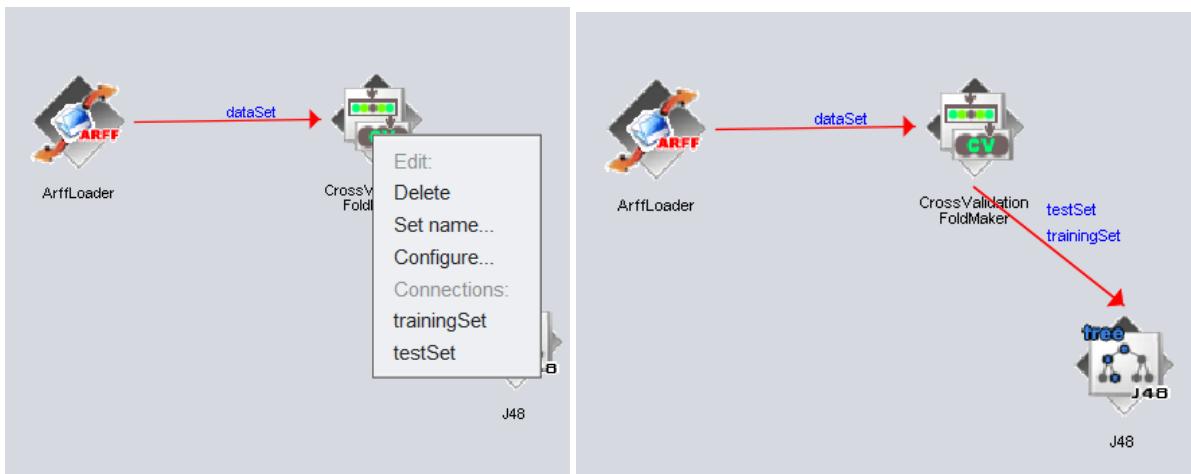
Para crear conexión dataset del primer elemento (ArffLoader) al segundo elemento (CrossValidationFoldMaker) . Le damos clic derecho al primer elemento y seleccionamos dataSet. Nos sale directamente una flecha para que la conectemos al elemento que queramos como se puede ver en las capturas siguientes.



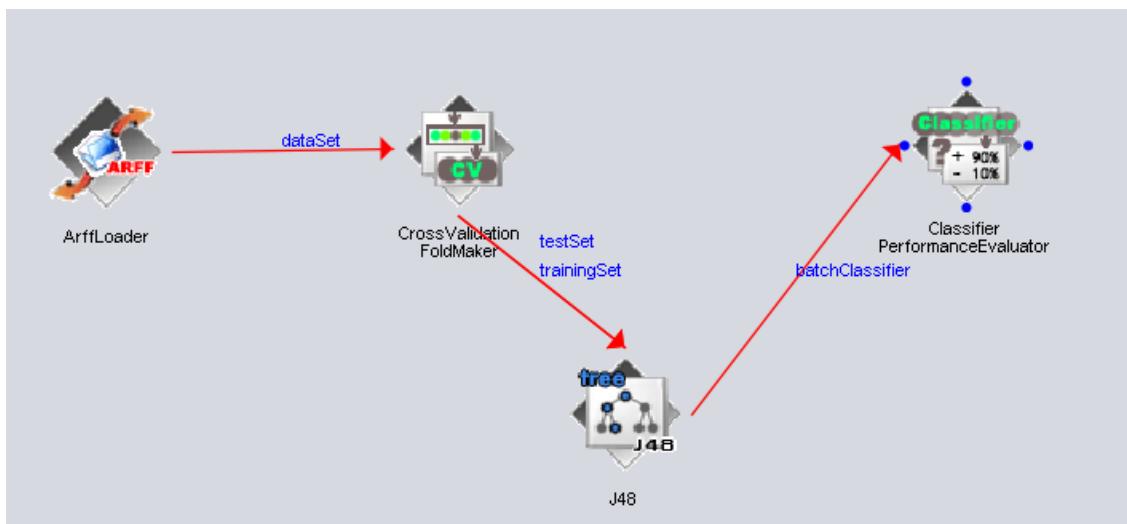
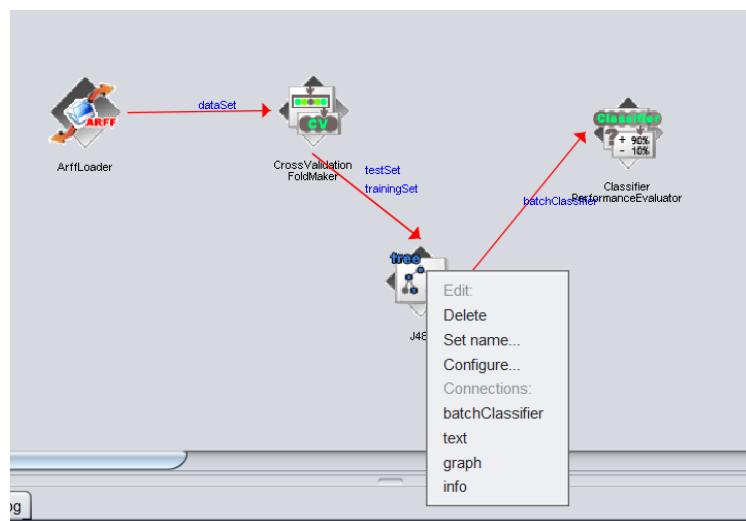
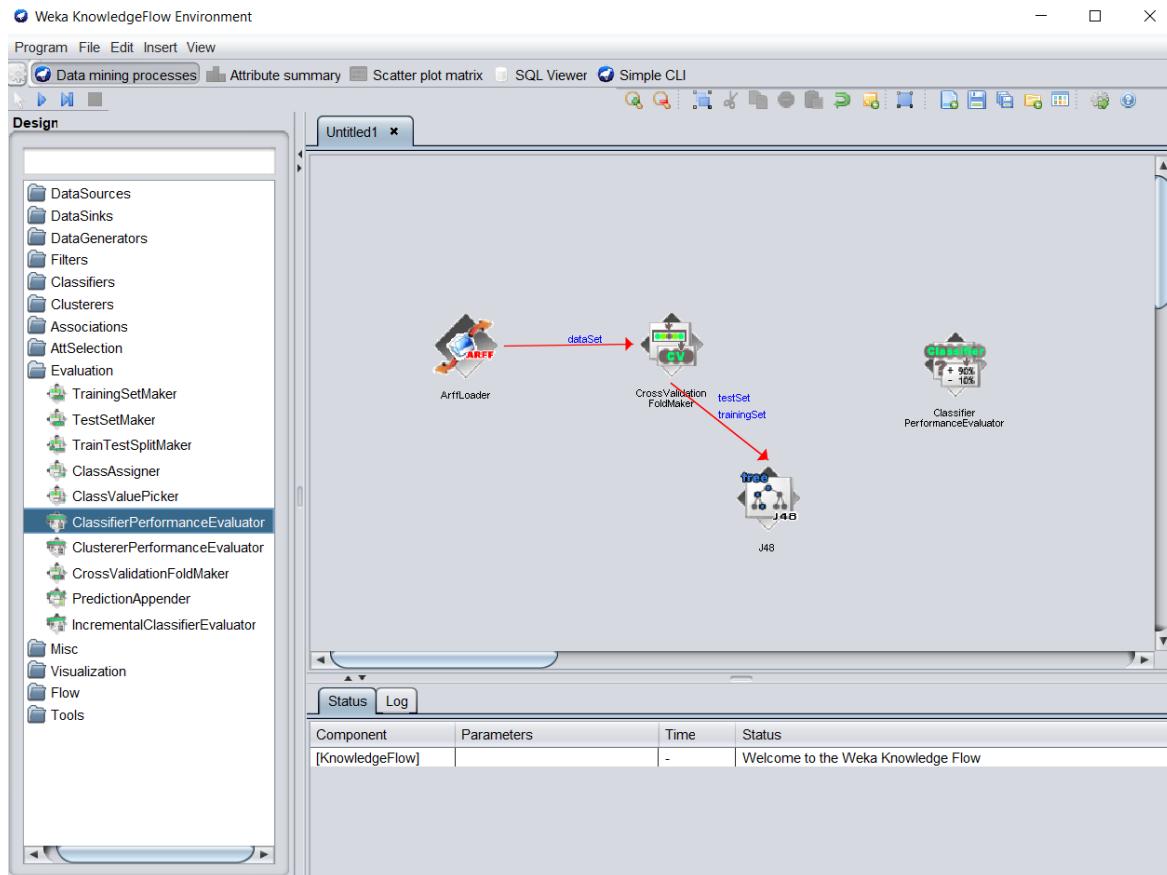
Ahora vamos a añadir el clasificador J48 para ello en la columna de Design seleccionamos Classifiers , trees y J48.



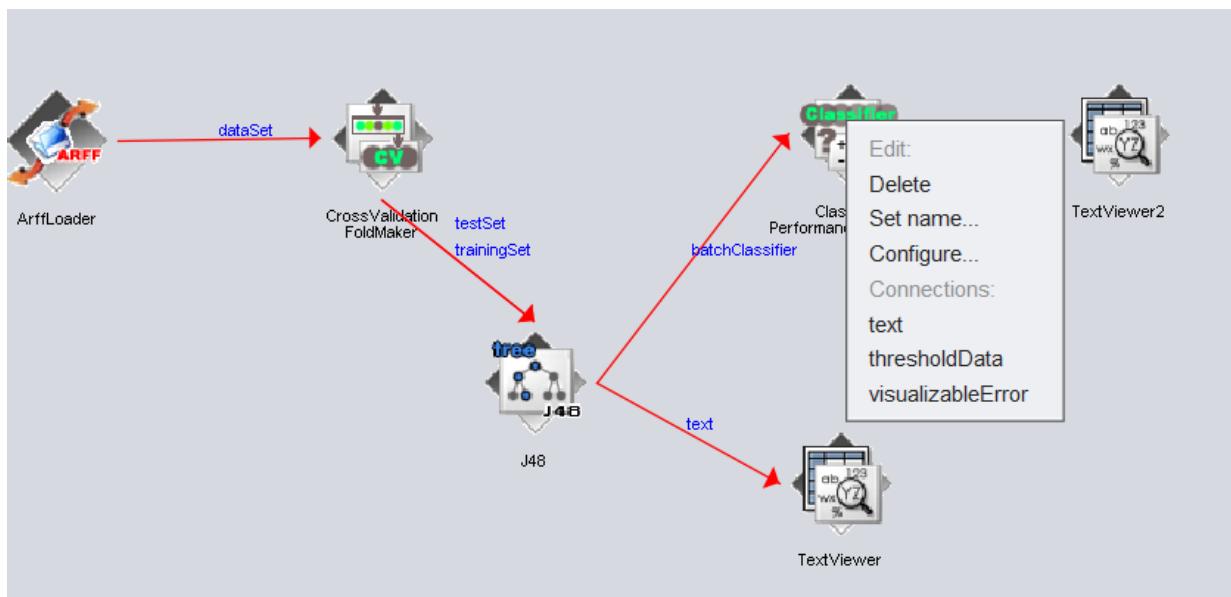
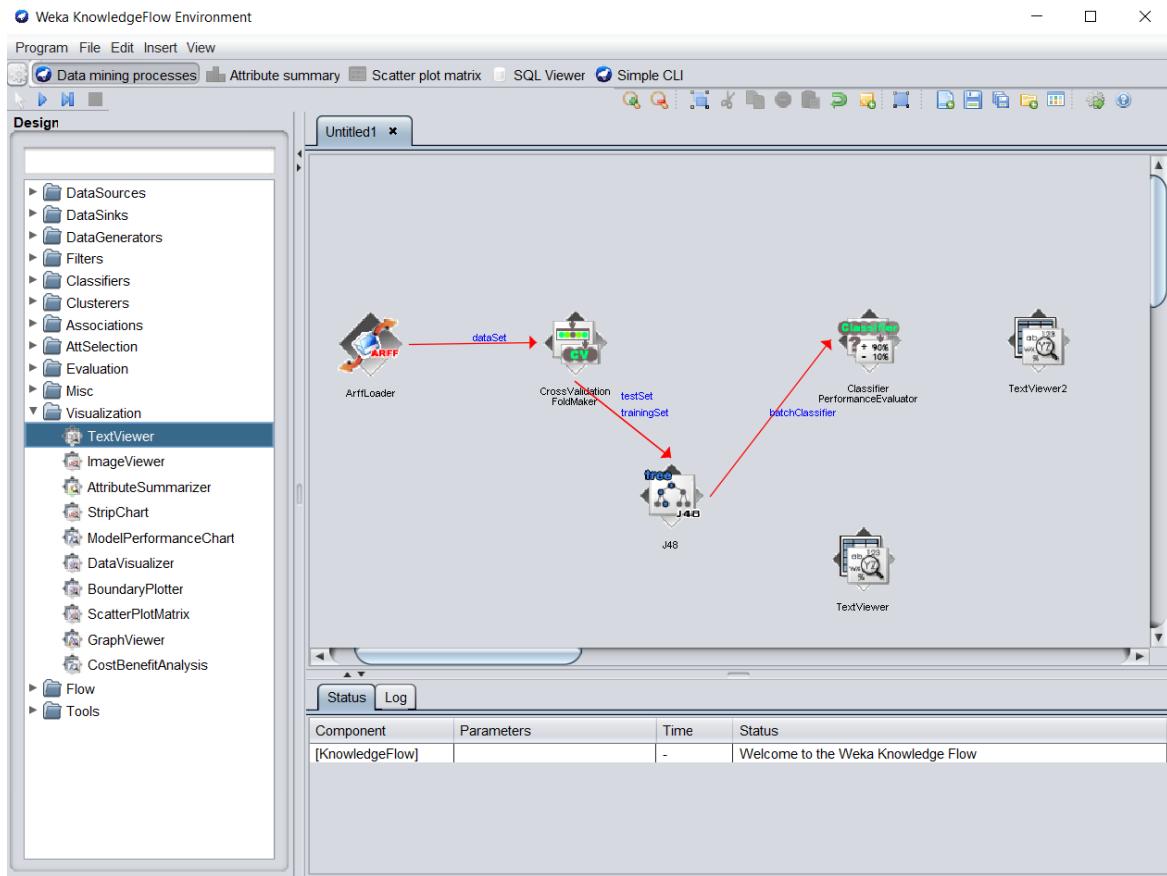
Hacemos igual para crear conexiones training y test del segundo al tercer elemento. Clic derecho en el segundo elemento, seleccionar trainingSet y testSet y con la línea conectarlo al tercer elemento.

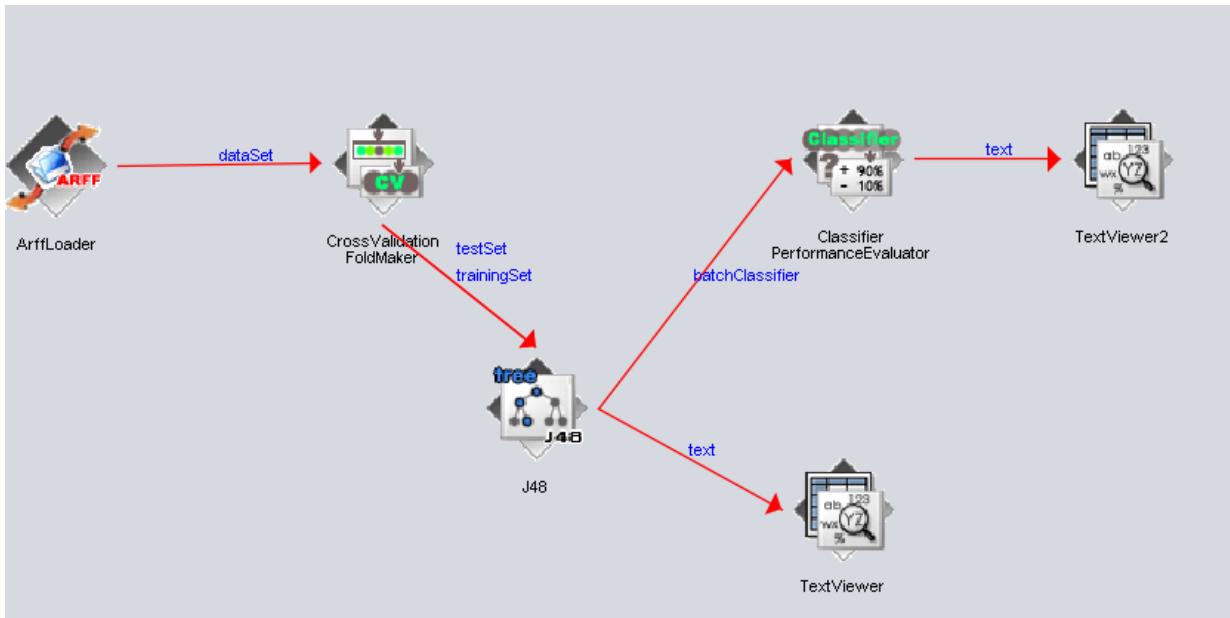


Añadimos Classifier performance evaluator (Evaluation) y lo conectamos con enlace batchclassifier. En la columna Design seleccionamos Evaluation y ClassifierPerformanceEvaluator. Hacemos clic izquierdo y lo colocamos donde queramos.

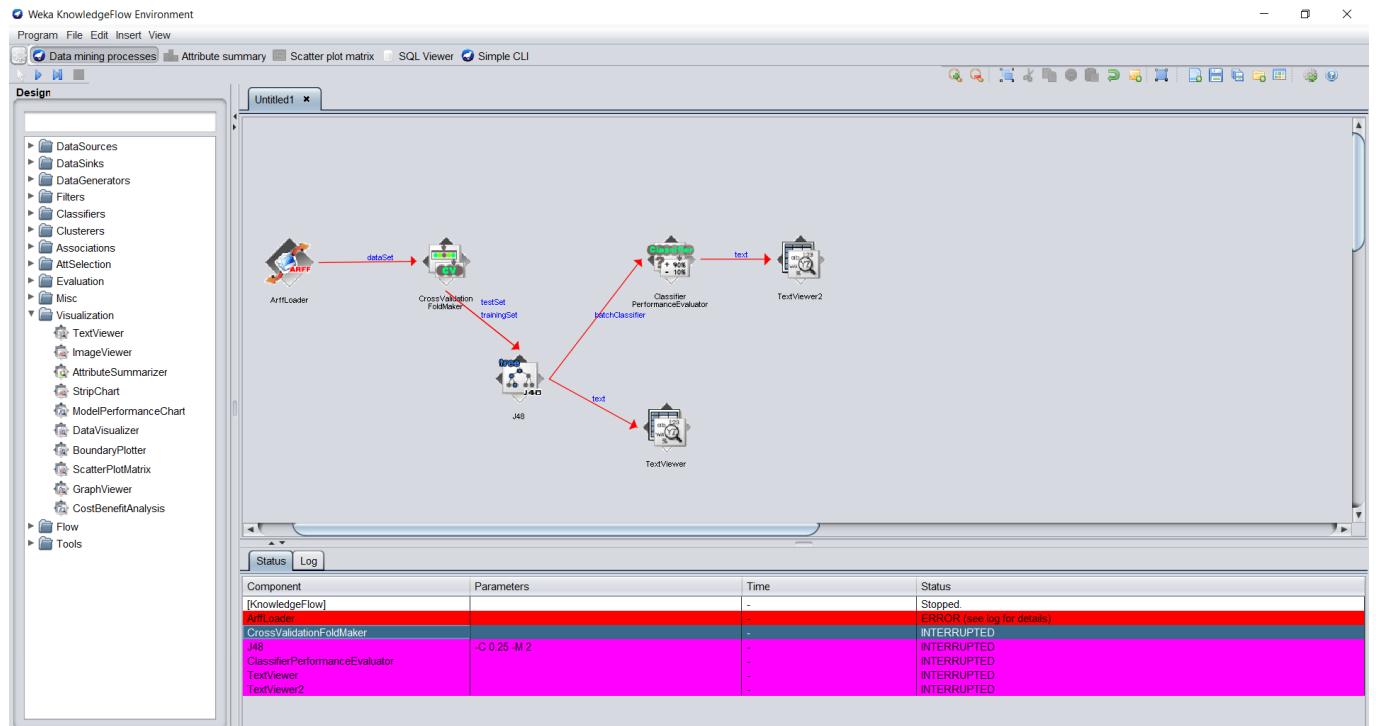


A continuación añadimos dos textviewer y lo conectamos con enlace text.

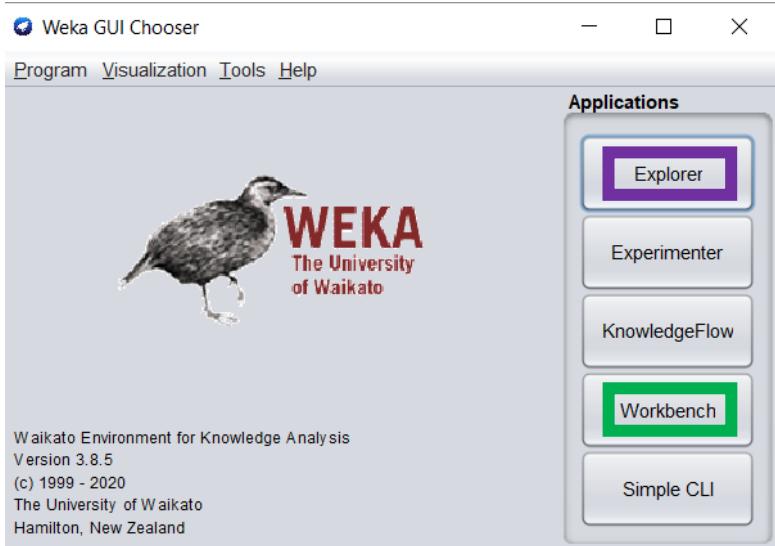




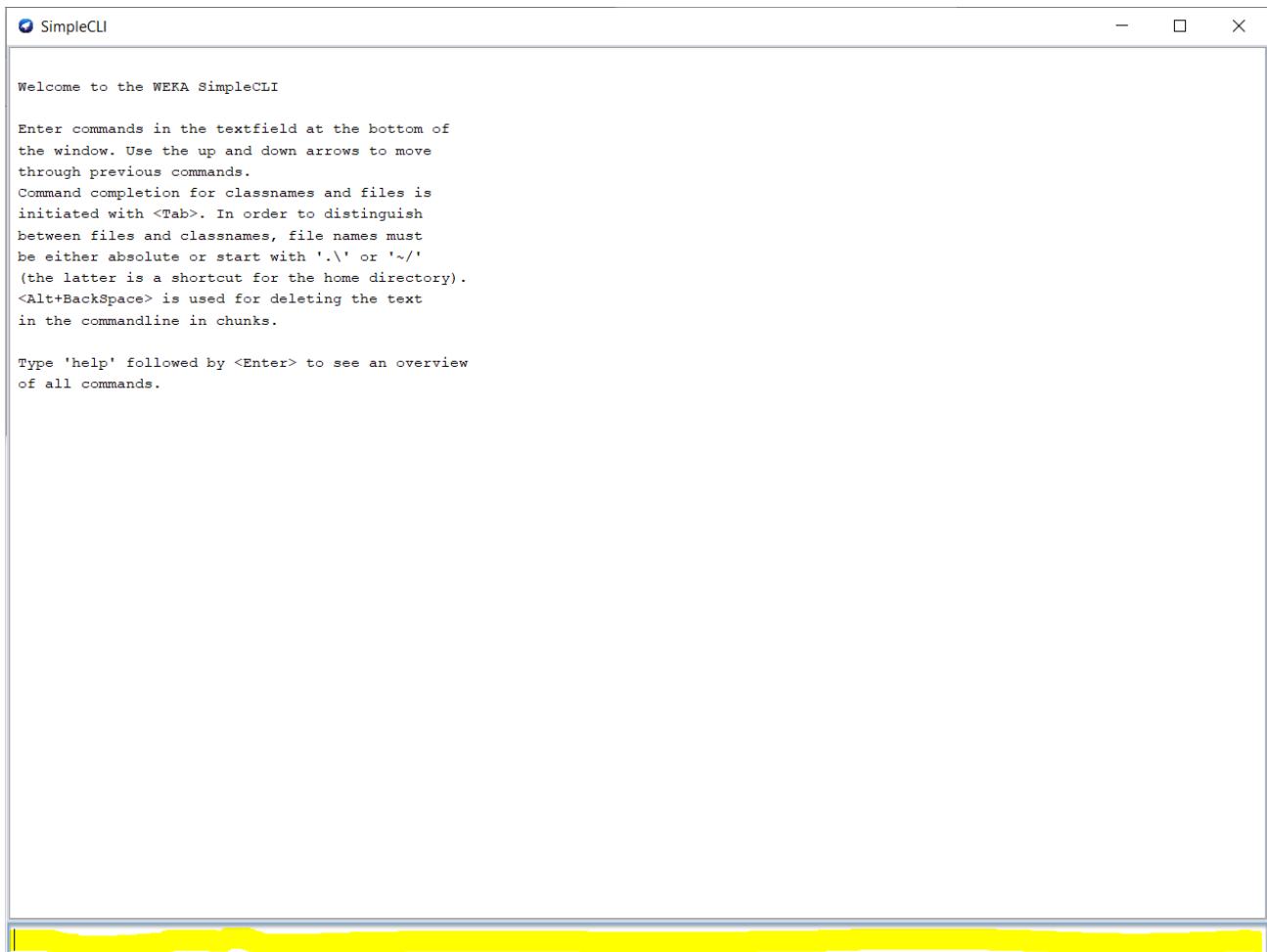
Arrancamos dándole a star loading en la esquina superior izquierda.



SIMPLE CLI



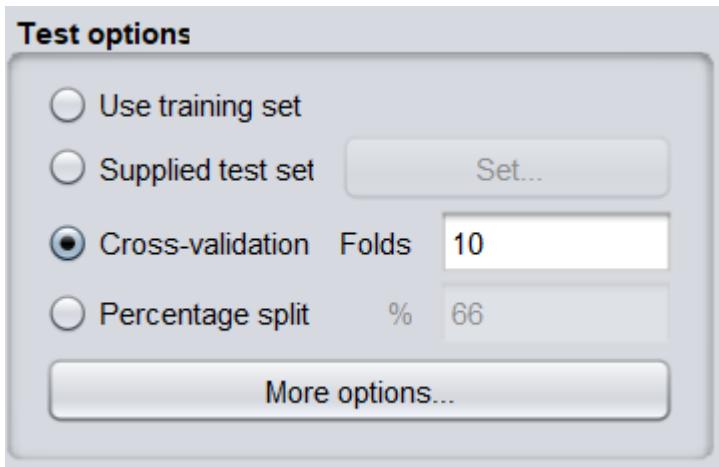
En clase hemos usado la herramienta gráfica EXPLORER para desarrollar rápidamente modelos de aprendizaje automático. Pero si le damos a Simple CLI esto es una interfaz de línea de comandos que tiene más potencia que la que da el explorador



Esta captura es la que nos sale y los comandos se escriben en la parte inferior (pintado de amarillo). Se puede hacer todo lo anterior hecho.

TÉRMINOS EXPLICADOS

En la ventana Classify y en la sección Test options vemos varios términos.



Use training set = Genera un árbol con apenas errores ya que el algoritmo conoce de antemano las soluciones.

Cross-validation = Dado un número n Folds, se divide los datos en n Folds partes y por cada parte se construye el clasificador con las n Folds -1 partes restantes y se prueba con esa. Así por cada una de las n Folds particiones.

Percentage split = Se puede dividir un conjunto de datos en dos partes expresadas en porcentajes. El primer porcentaje que suele ser el mayor de los dos, se destinará a los datos de entrenamiento, mientras que el segundo porcentaje se destinará a los datos de prueba. Para que la separación en porcentajes sea realmente efectiva es necesario que se repitan lo menos posible los datos de prueba. EL % que le pongamos será el % de separación para datos de entrenamiento y el restante a datos de prueba.