

CIFF Trustees:



Business Intelligence & Data Mining

Profesor:
Pedro Pasquau

Octubre 2015

MASTER EN BUSINESS ANALYTICS & BIG DATA
2015– 2016

A.- Data Mining

- Qué es Data Mining.
- Procesos de Data Mining
- Algoritmos de Data Mining
- Herramienta de Data Mining: Introducción a WEKA
- Ejemplos sencillos sobre distintos algoritmos y análisis de los resultados obtenidos



A.- Data Mining

- Qué es Data Mining.
- Procesos de Data Mining
- Algoritmos de Data Mining
- **Herramienta de Data Mining: Introducción a WEKA**
- Ejemplos sencillos sobre distintos algoritmos y análisis de los resultados obtenidos



WEKA:

Weka es un conjunto de librerías JAVA para la extracción de conocimientos desde bases de datos.

Es un software ha sido desarrollado bajo licencia [GNU General Public License](#) lo cual ha impulsado que sea una de las suites más utilizadas en el área en los últimos años.



WEKA Instalación:

<http://www.cs.waikato.ac.nz/~ml/weka/>

<http://www.cs.waikato.ac.nz/~ml/weka/downloading.html>

Memoria en WEKA :

Aunque no va a ser necesario para este caso, en ocasiones es importante incrementar la cantidad de memoria que Weka puede utilizar.

Esto se puede hacer creando un fichero .bat (en Windows) como este:

@echo off

javaw -Xmx1024m -classpath "C:\Program Files (x86)\Weka-3-6\weka.jar"
weka.gui.Main

para ello habrá que localizar previamente el directorio donde está instalado el .jar de Weka.

Acceso a WEKA :

Ejecutar Weka 3.6 (with console)

WEKA .ARFF:

Nativamente Weka trabaja con un formato denominado arff ,
acrónimo de *Attribute-Relation File Format*

Estructura:

1. Cabecera. Se define el nombre de la relación.
2. Declaraciones de atributos. En esta sección se declaran los atributos que compondrán nuestro archivo junto a su tipo.

Tipos de Datos:

- 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.
- d) **STRING** Expresa cadenas de texto,
- 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.

Por ejemplo, si tenemos un atributo que indica el tiempo podría definirse:

@attribute tiempo {soleado,lluvioso,nublado}

Tipos de Datos:

Un ejemplo de un archivo de prueba.

prueba.arff

1 % Archivo de prueba para Weka.

2 @relation prueba

3

4 @attribute nombre STRING

5 @attribute ojo_izquierdo {Bien,Mal}

6 @attribute dimension NUMERIC

7 @attribute fecha_analisis DATE "dd-MM-yyyy HH:mm"

8

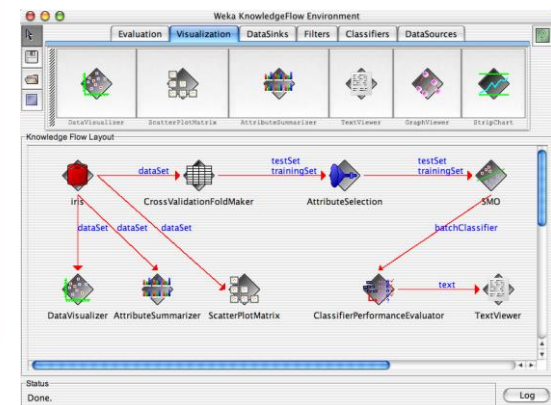
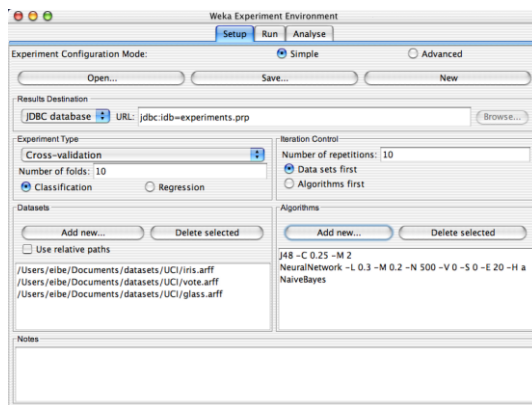
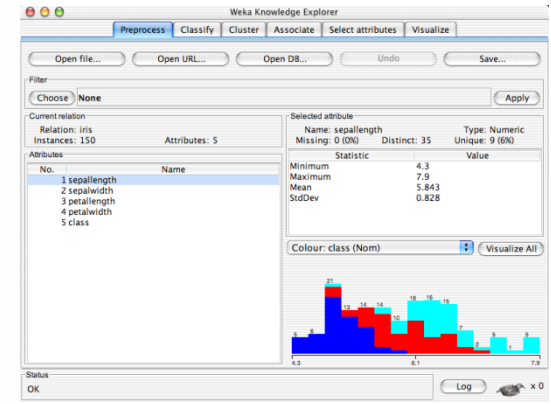
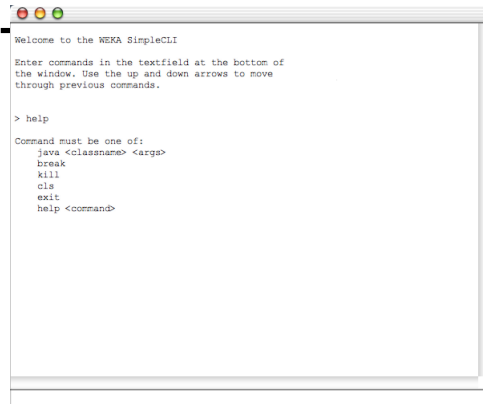
9 @data

10 Antonio,Bien,38.43,"12-04-2003 12:23"

11 'Maria Jose',?,34.53,"14-05-2003 13:45"

12 Juan,Bien,43,"01-01-2004 08:04"

13 Maria,?,?, "03-04-2003 11:03"

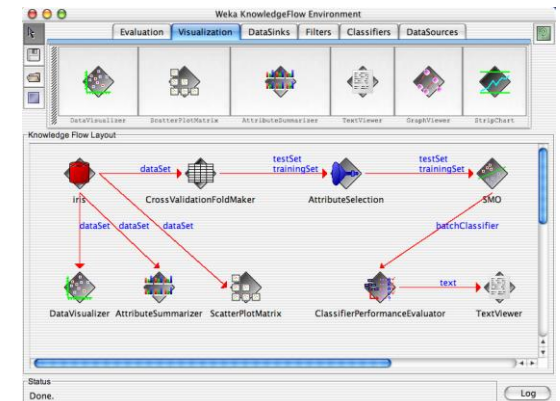
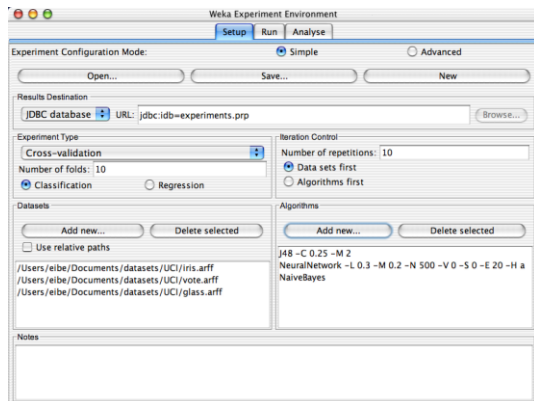
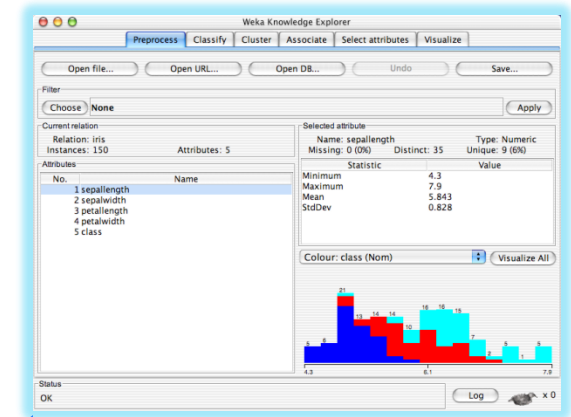


Simple CLI

Simple CLI es una abreviación de **Simple Client**.

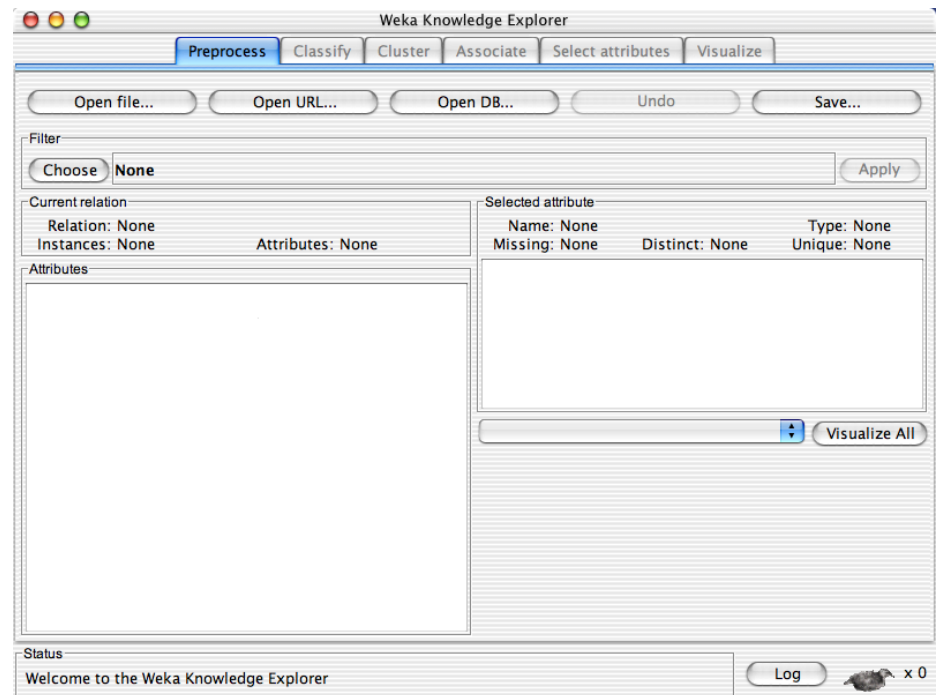
- consola para poder introducir instrucciones
- Apariencia muy simple
- Extremadamente potente
- Complicada de manejar ya que es necesario un conocimiento completo de la aplicación.
- Actualmente ya prácticamente sólo es útil como una herramienta de ayuda a la fase de pruebas gracias al resto de elementos.

Las instrucciones soportadas son → `Img`



Explorer

- Modo más Sencillo y Descriptivo
- Permite realizar operaciones sobre un mismo archivo
- permite tareas de:
 1. Preprocesado de los datos y aplicación de filtros.
 2. Clasificación.
 3. Clustering.
 4. Búsqueda de Asociaciones.
 5. Selección de atributos
 6. Visualización de datos.



Explorer: sub-entornos de ejecución:

- **Preprocess:** Incluye las herramientas y filtros para cargar y manipular los datos
- **Classification:** Acceso a las técnicas de clasificación y regresión
- **Cluster:** Integra varios métodos de agrupamiento
- **Associate:** Incluye una pocas técnicas de reglas de asociación
- **Select Attributes:** Permite aplicar diversas técnicas para la reducción del número de atributos
- **Visualize:** En este apartado podemos estudiar el comportamiento de los datos mediante técnicas de visualización.

Preprocesado & Filtrado:

Definir origen de Datos. Por defecto Arff.

Tipos:

- Arff → Estandar
- CSV → Separado por comas o Tab.
- C4.5 → Codificación C4.5. Requiere de un .names y .data
- Instancias Serializadas
- Open Url → Dirección http de nuestro fichero
- Open DB → Requiere configuración

Abrir iris.arff:

Balanceados

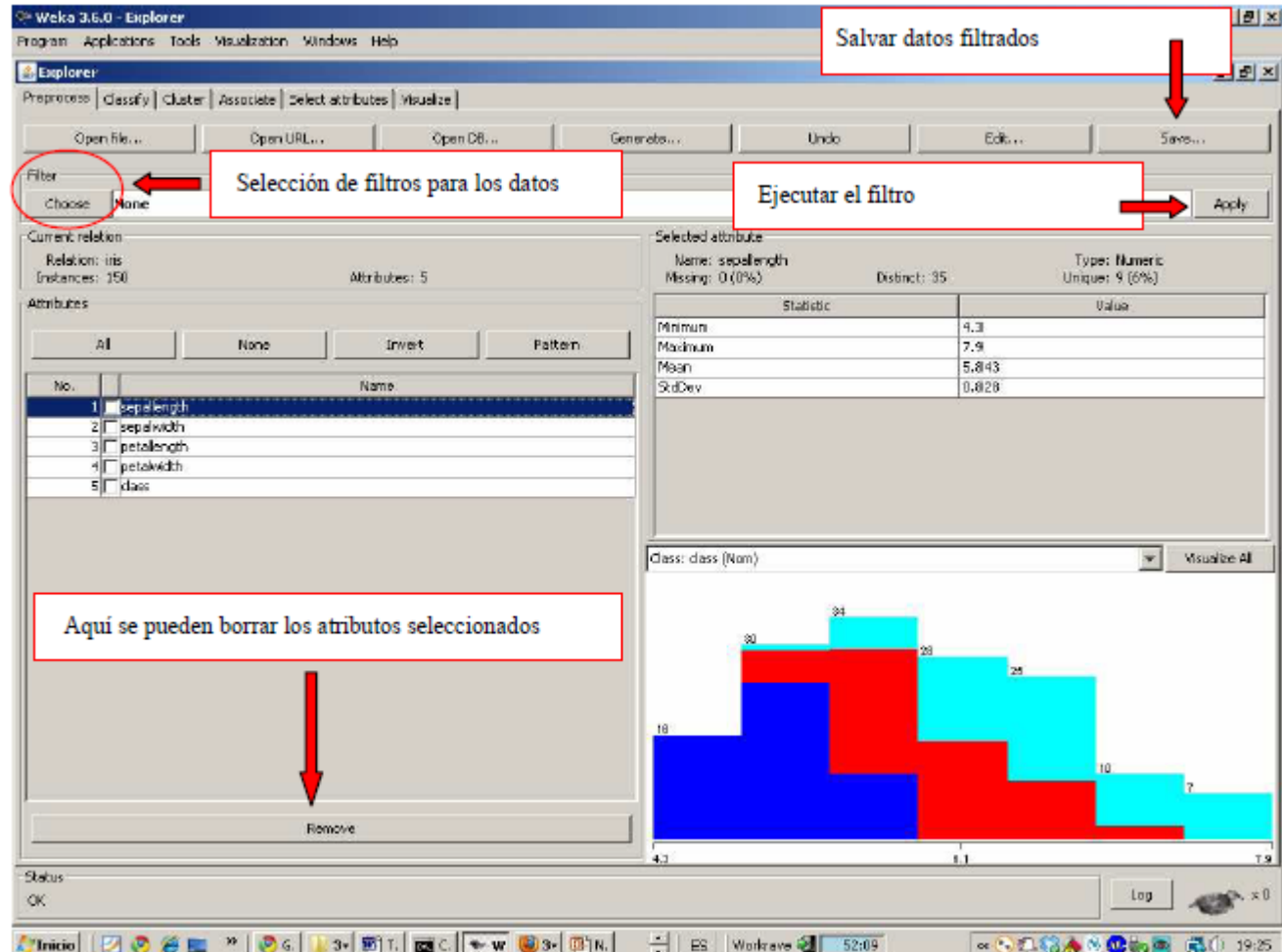
Selección de filtros para los datos

Nombres de atributos

Estadísticas de los datos

Desglose del atributo seleccionado en la izquierda

Selección de Filtros:



The screenshot shows the WEKA 3.6.0 Explorer window. The 'Filter' button is circled in red, with an arrow pointing to it from the text 'Selección de filtros para los datos'. The 'Apply' button is also circled in red, with an arrow pointing to it from the text 'Ejecutar el filtro'. A red arrow points from the 'Save...' button to the text 'Salvar datos filtrados'. A red arrow points from the 'Remove' button to the text 'Aquí se pueden borrar los atributos seleccionados'. The 'Current relation' section shows 'Relation: iris' and 'Instances: 150'. The 'Attributes' section shows a list of attributes: 'sepal.length', 'sepal.width', 'petal.length', 'petal.width', and 'class'. The 'Selected attribute' section shows 'sepal.length' with statistics: Minimum 4.3, Maximum 7.9, Mean 5.843, StdDev 0.828. The 'Class' section shows 'class (Nom)' and a 'Visualize All' button. A bar chart is displayed at the bottom right, showing the distribution of the 'class' attribute across the 'sepal.length' attribute.

Salvar datos filtrados

Selección de filtros para los datos

Ejecutar el filtro

Aquí se pueden borrar los atributos seleccionados

Remove

Visualize All

Selección de Filtros:

Add Añade un atributo más. Como parámetros debemos proporcionarle la posición que va a ocupar este nuevo atributo (esta vez comenzando desde el 1), el nombre del atributo y los posibles valores de ese atributo separados entre comas. Si no se especifican, se sobreentiende que el atributo es numérico.

AddExpression Este filtro es muy útil puesto que permite agregar al final un atributo que sea el valor de una función.

Por ejemplo:

$(a_3^{3.4}) * a_1 + \sqrt{\text{floor}(\tan(a_4))}$

AddNoise Añade ruido a un determinado atributo que debe ser nominal.

ClusterMembership Filtro que dado un conjunto de atributos y el atributo que define la clase de los mismos, devuelve la probabilidad de cada uno de los atributos de estar clasificados en una clase u otra.

Selección de Filtros:

Ver → Descripción Basica de Filtros.pdf



Selección de Filtros:

Ejemplo: Vamos a aplicar un filtro a un atributo concreto.

Discretizar el atributo



Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

Open file...

Open URL...

Open DB...

Undo

Save...

Filter

Choose

None

Apply

Current relation

Relation: iris

Instances: 150

Attributes: 5

Attributes

No.	Name
1	sepal.length
2	sepal.width
3	petal.length
4	petal.width
5	class

Selected attribute

Name: petal.length

Type: Numeric

Missing: 0 (0%)

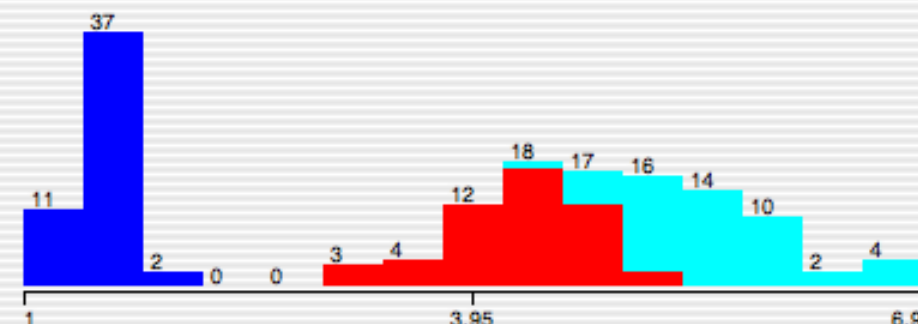
Distinct: 43

Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom)

Visualize All



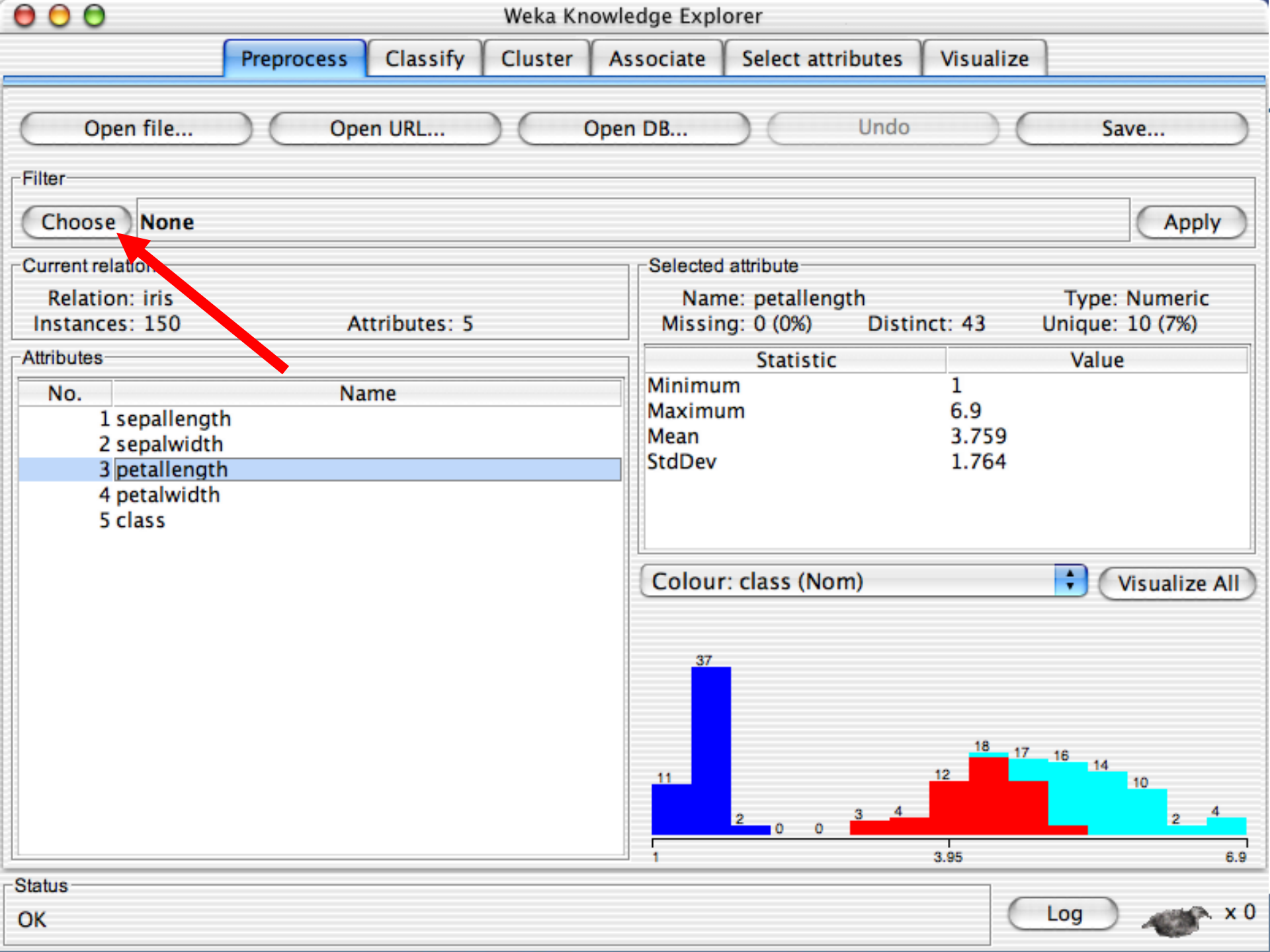
Status

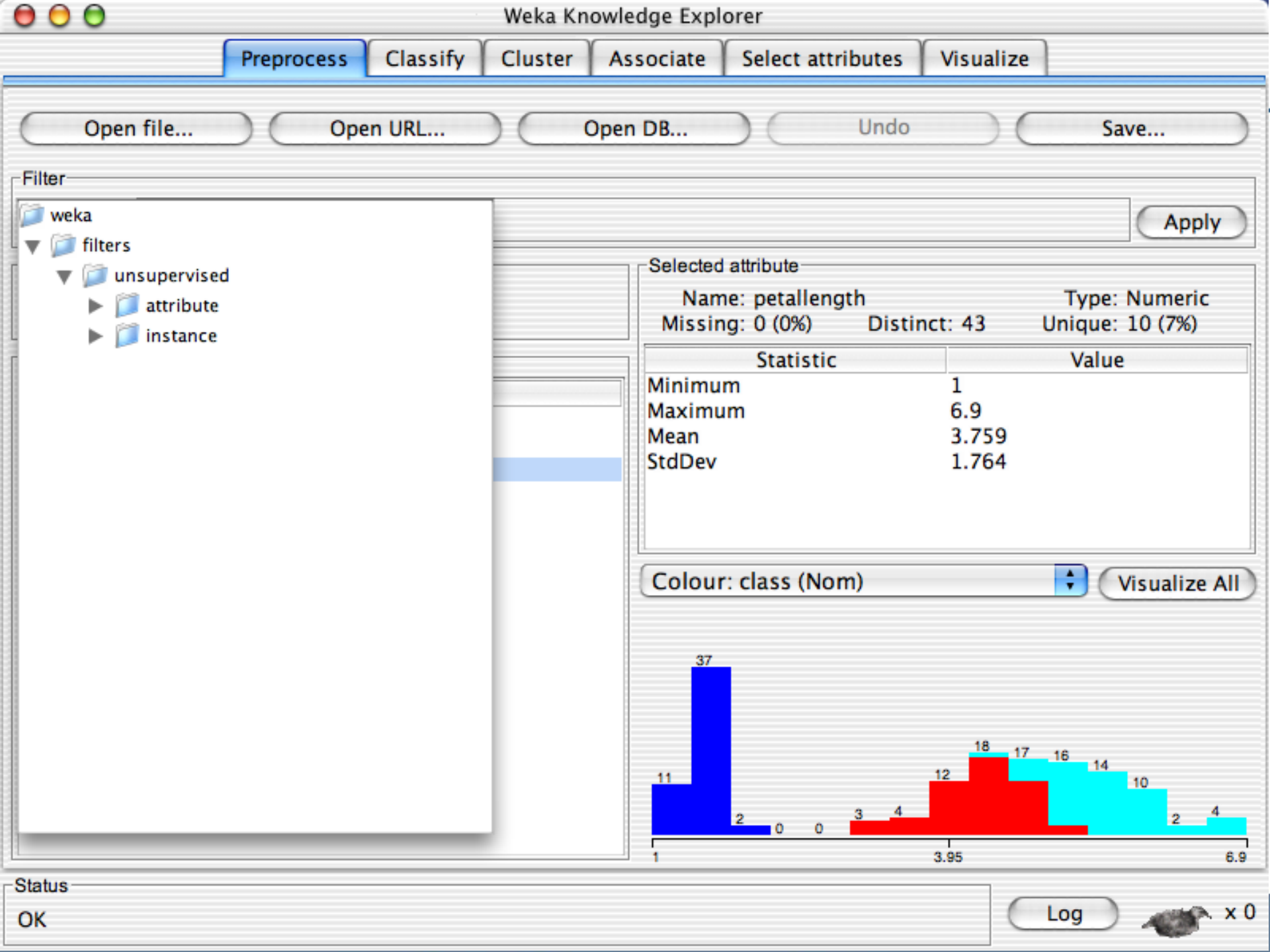
OK

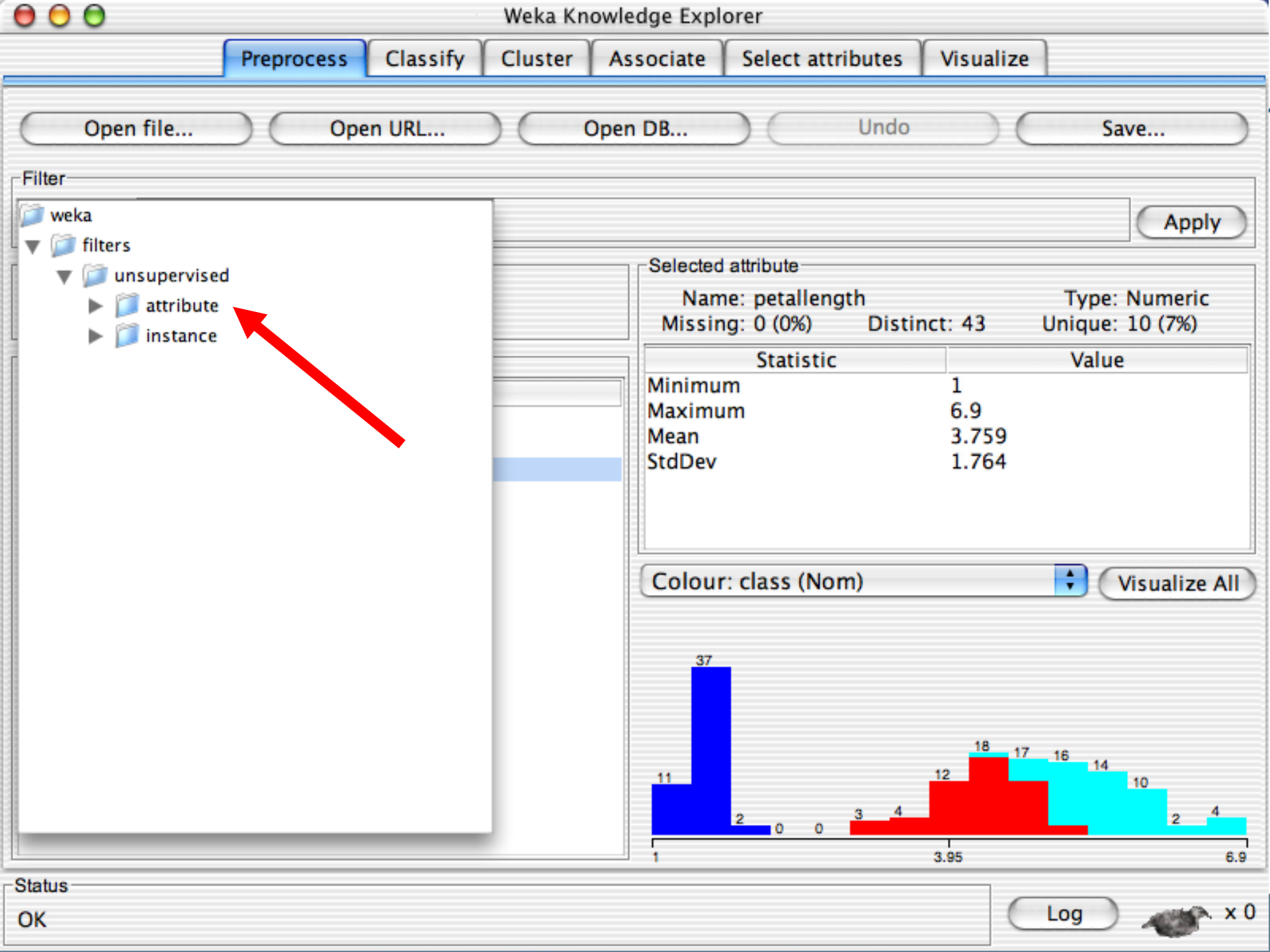
Log

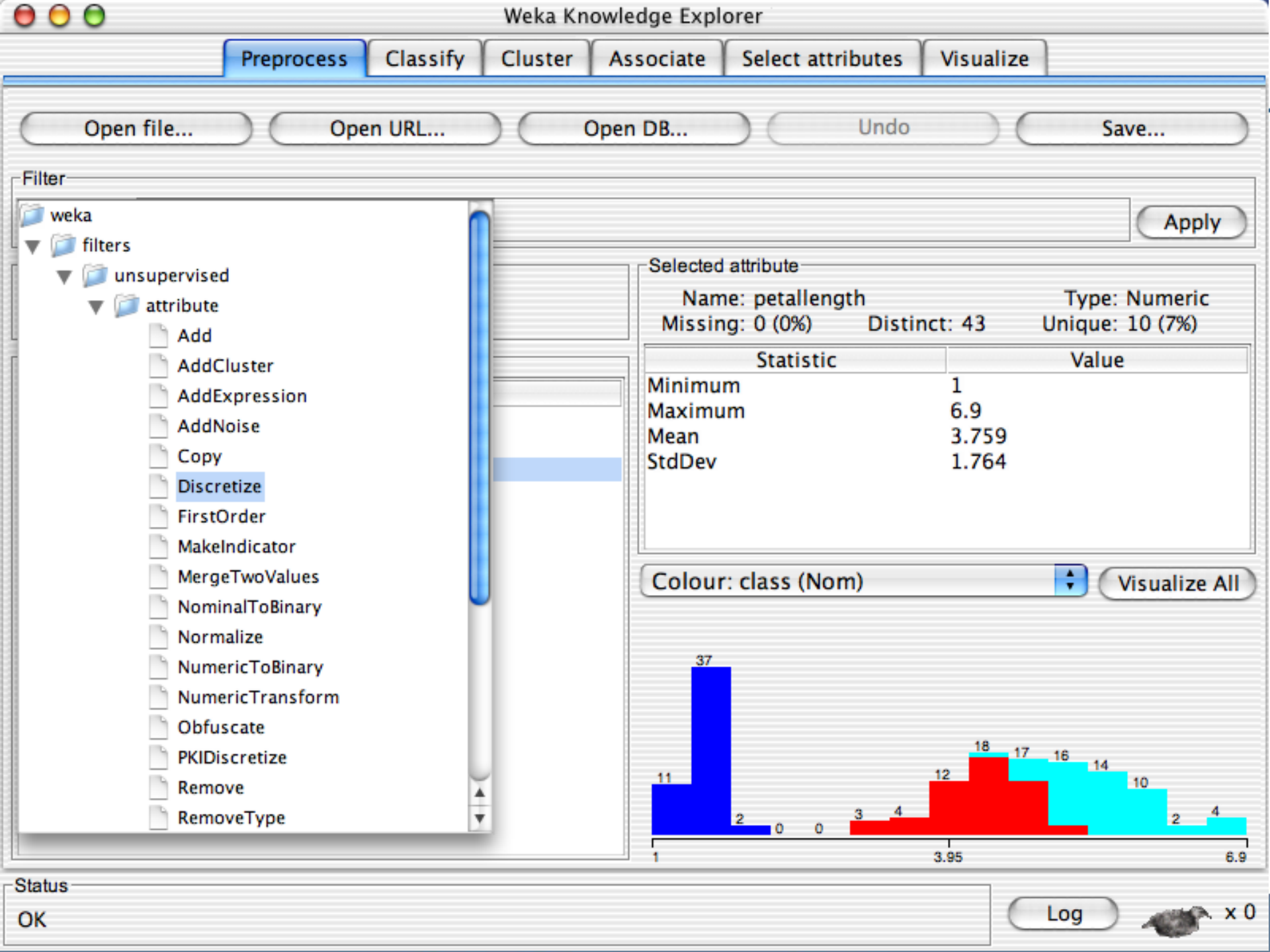


x 0











Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

Open file...

Open URL...

Open DB...

Undo

Save...

Filter

Choose

Discretize -B 10 -R first-last

Apply

Current relation

Relation: iris

Instances: 150

Attributes: 5

Attributes

No.	Name
1	sepal.length
2	sepal.width
3	petal.length
4	petal.width
5	class

Selected attribute

Name: petal.length

Type: Numeric

Missing: 0 (0%)

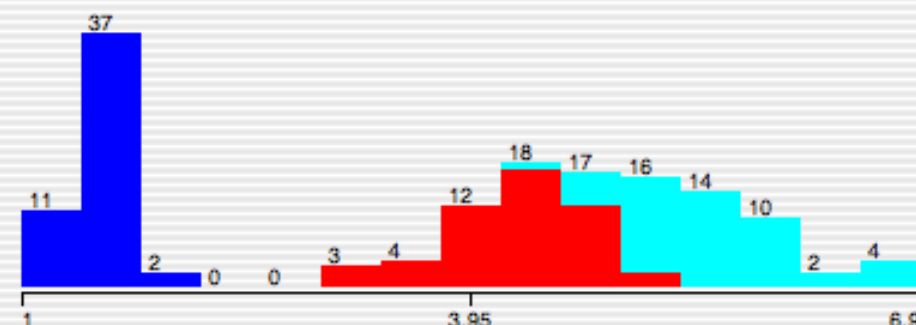
Distinct: 43

Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom)

Visualize All



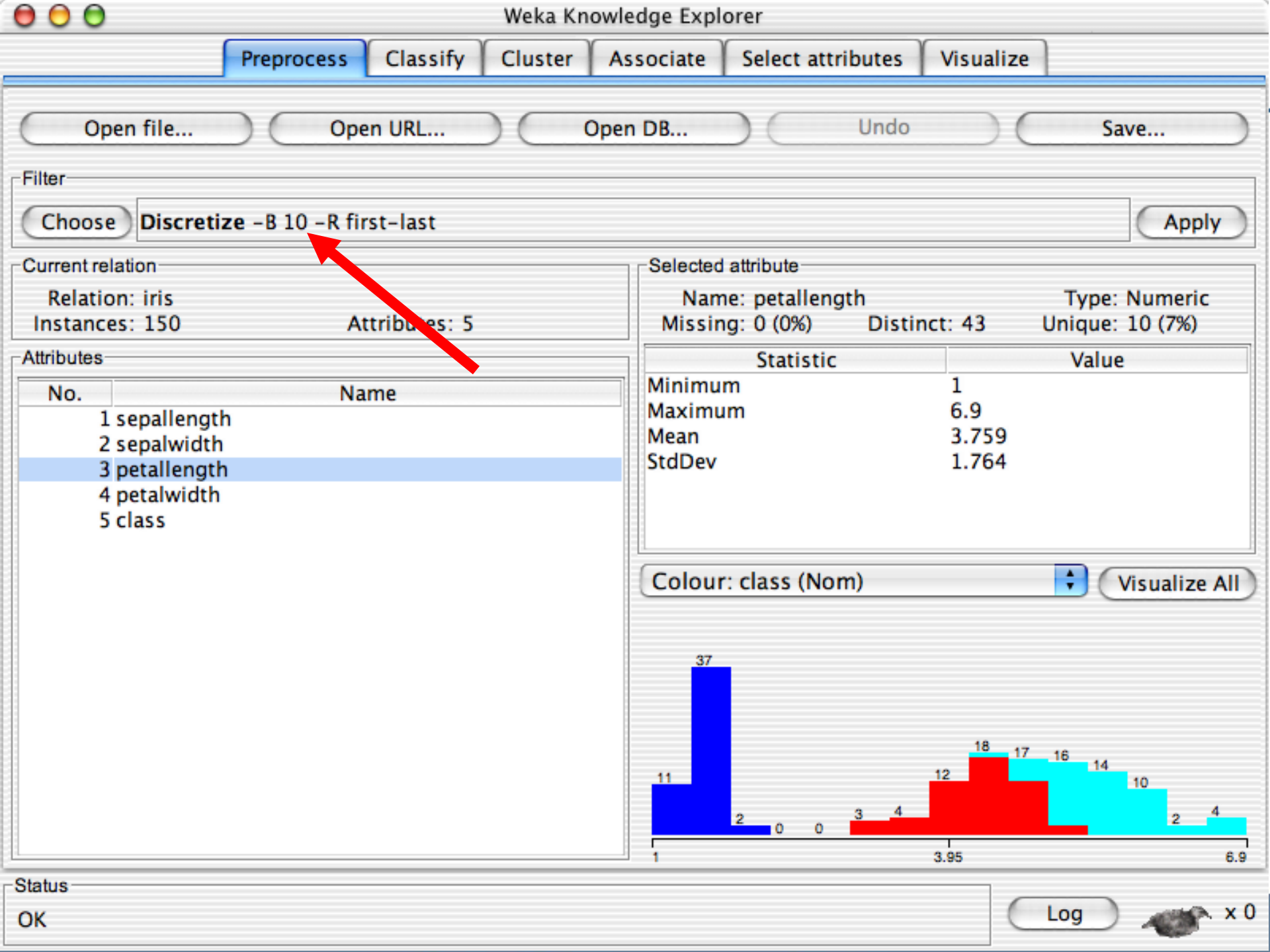
Status

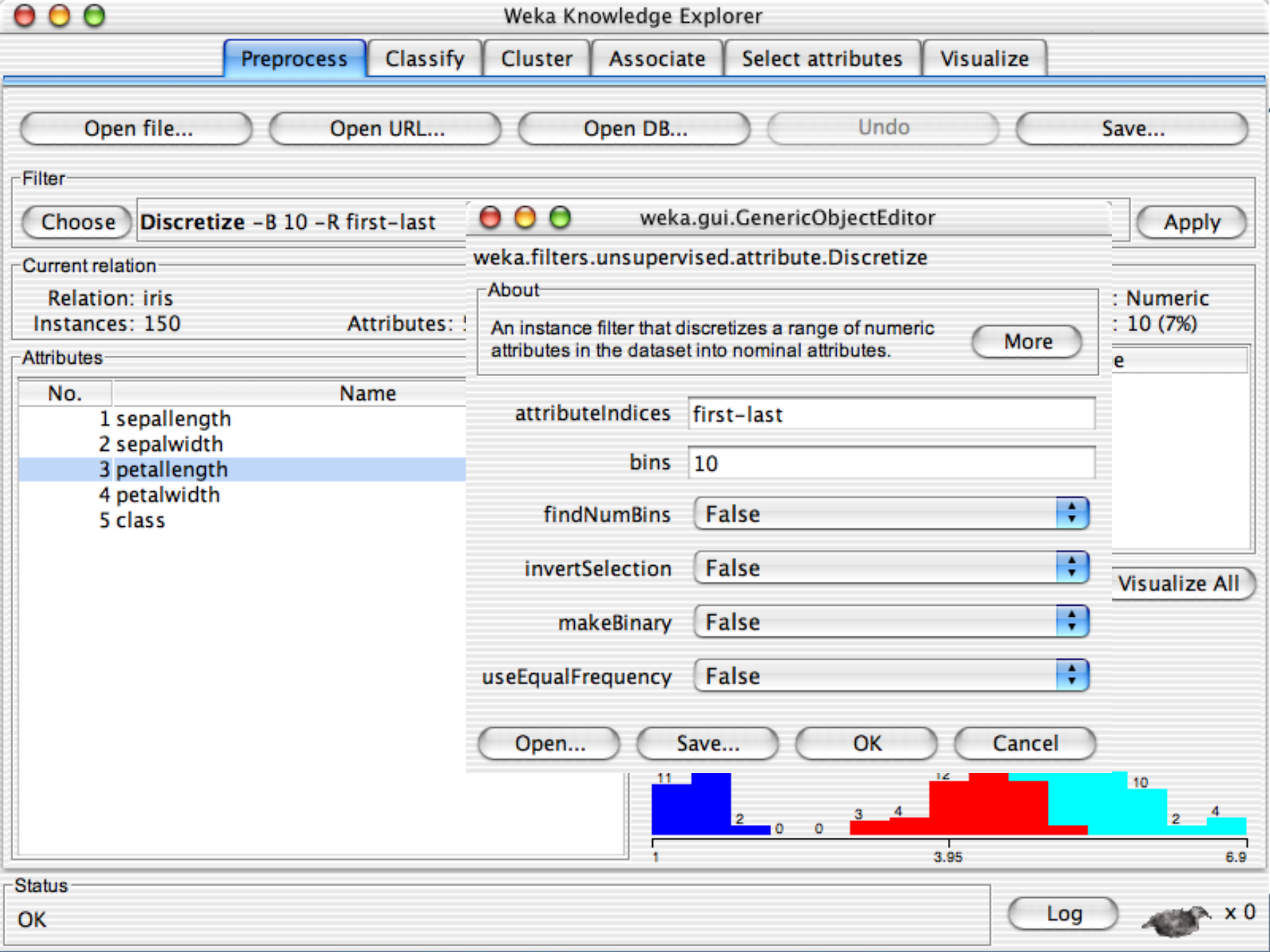
OK

Log



x 0

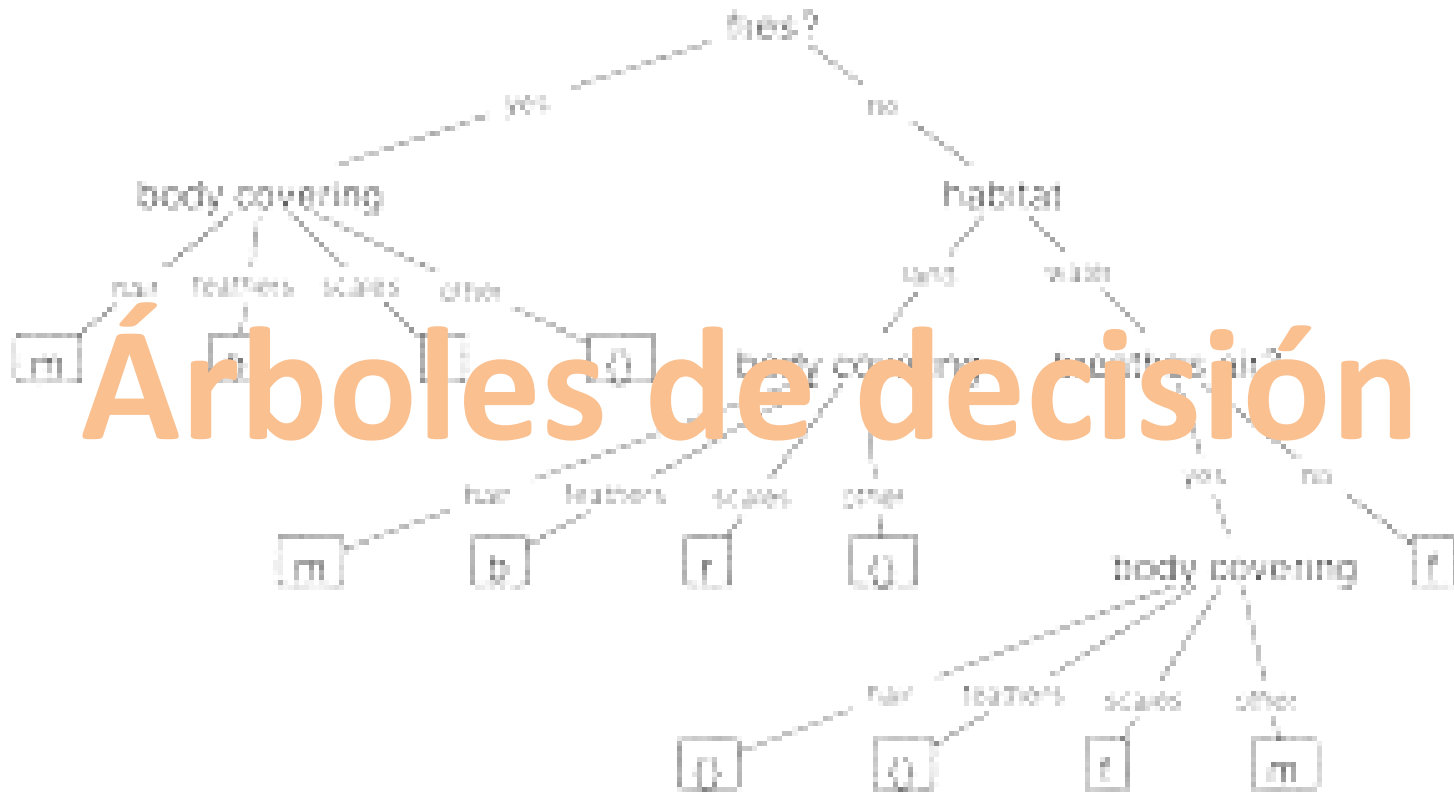




Selección de Filtros:

Ejemplo: Vamos a aplicar un filtro a un atributo concreto.

Discretizar el atributo



Arboles de decisión

- Los árboles de decisión son uno de los modelos de clasificación más utilizados.
- Su interpretación es bastante natural, su relación con técnicas de clasificación establecidas y probadas, en particular, y también los distintos métodos existentes para traducir los modelos resultados en colecciones de reglas de clasificación los hacen versátiles y aplicables

Arboles de decisión

- Un árbol de decisión tiene unas entradas las cuales pueden ser un objeto o una situación descrita por medio de un conjunto de atributos y a partir de esto devuelve una respuesta la cual es una decisión que es tomada a partir de las entradas. Se utilizan más los valores discretos por simplicidad, cuando se utilizan valores discretos en las funciones de una aplicación se denomina clasificación y cuando se utilizan los continuos se denomina regresión.

Arboles de decisión

- Un árbol de decisión lleva a cabo un test a medida que este se recorre hacia las hojas para alcanzar así una decisión. El árbol de decisión suele contener nodos internos, nodos hojas y arcos.
 - Un nodo interno contiene un test sobre algún valor de una de las propiedades.
 - Un nodo hoja representa el valor que devolverá el árbol de decisión
 - Las ramas brindan los posibles caminos que se tienen de acuerdo a la decisión tomada.

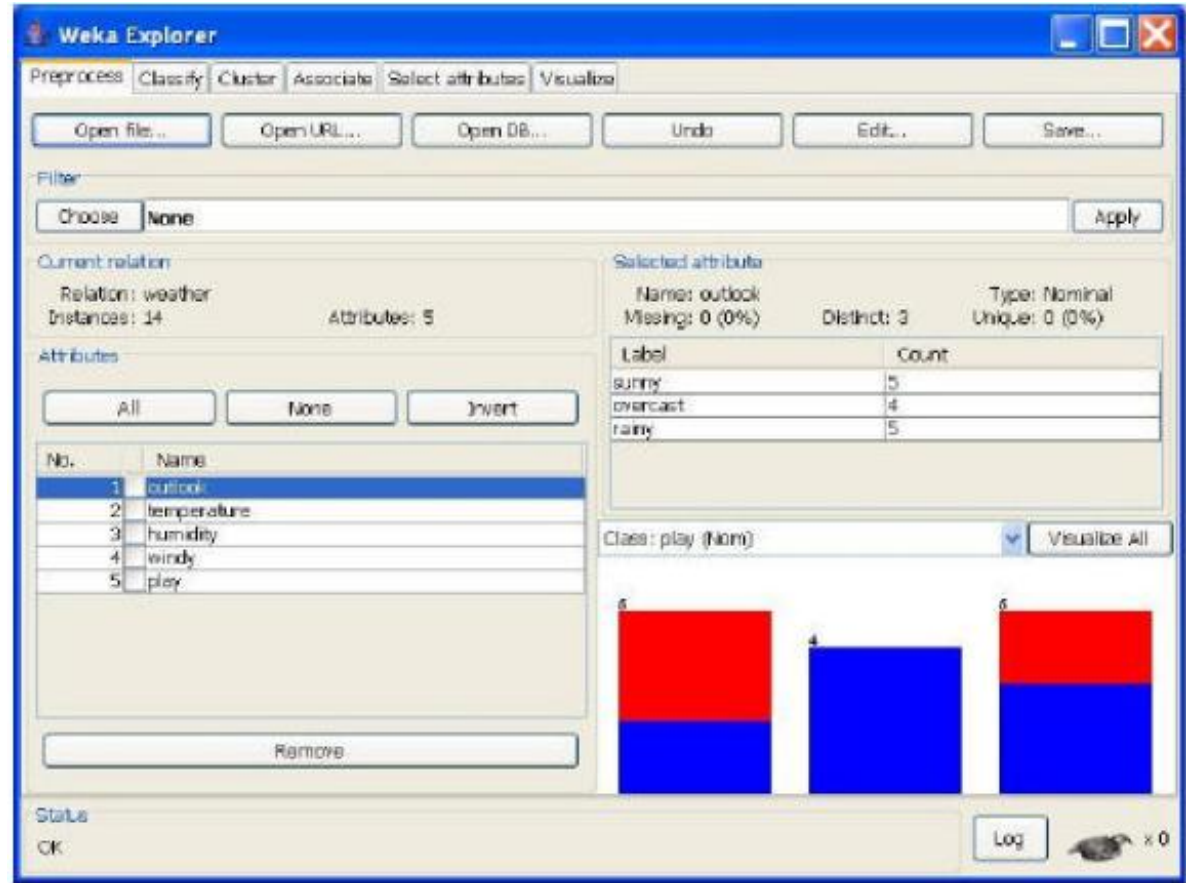
Nuestro Primer Clasificador/Predictor

- **objetivo:** poder determinar (predecir) si hoy podremos jugar al tenis.
- **Datos:**

Sky	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Nuestro Primer Clasificador/Predictor

- weather.nominal.arff



Nuestro Primer Clasificador/Predictor

- **Preproceso:**

Pulsando en el botón Choose en Filter, tendremos acceso a multitud de herramientas para el preprocesamiento de datos.

- · Realizar un filtrado de atributos
- · Cambiar el tipo de los atributos (discretizar o numerizar)
- · Realizar muestreos sobre los datos
- · Normalizar atributos numéricos
- · Unificar valores de un mismo atributo

Nuestro Primer Clasificador/Predictor

- Selección de Clasificador o Regresión:
 - **Bayes**. Métodos basados en el paradigma del aprendizaje de Bayes
 - **Funciones. Métodos “matemáticos”**: Redes neuronales, regresiones, SVM...
 - **Lazy**. Métodos que utilizan el paradigma de aprendizaje perezoso, es decir no construyen un modelo
 - **Meta**. Métodos que permiten combinar diferentes métodos de aprendizaje
 - **Trees**. Métodos que aprenden mediante la generación de árboles de decisión
 - **Rules**. Métodos que aprenden modelos que se pueden expresar como reglas.

Nuestro Primer Clasificador/Predictor

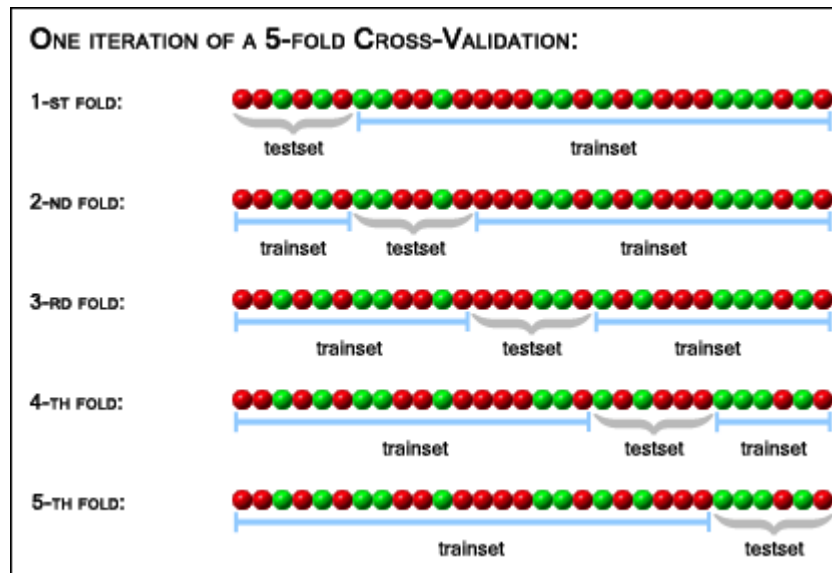
Selección del proceso de VALIDACIÓN del modelo resultante o Aprendido:

- **Use training set.**: Con esta opción Weka entrenará el método con todos los datos disponibles y a posteriori realiza la evaluación sobre los mismos datos.
- **Supplied test set**: Con esta opción podemos cargar un conjunto de datos (normalmente diferentes a los de aprendizaje) con los cuales se realizará la evaluación
- **Cross-validation**: Definido un número de pliegues a utilizar, 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.
- **Percentage split**. Se define un porcentaje con el que se aprende el modelo. La evaluación se realiza con los datos restantes. **IMP: PRESERVE ORDER FOR PERCENTAGE SPLIT**

Nuestro Primer Clasificador/Predictor

Selección del proceso de VALIDACIÓN del modelo resultante o Aprendido:

- Cross-validation: Definido un número de pliegues a utilizar $n=5$



Una validación-cruzada es estratificada cuando cada una de las partes conserva las propiedades de la muestra original (porcentaje de elementos de cada clase).

Nuestro Primer Clasificador/Predictor

Más Opciones del proceso de VALIDACIÓN del modelo resultante o Aprendido:

- **Output Model** Si la activamos una vez construido y probado el clasificador, nos mostrará en la salida del clasificador el modelo que ha construido.
- **Output per-class stats** Activada muestra estadísticas referentes a cada clase.
- **Output entropy evaluation measures** Muestra información de mediciones de la entropía en la clasificación.
- **Output confusion matrix** Muestra la matriz de confusión del clasificador. Esta tabla cuyo número de columnas es el número de atributos muestra la clasificación de las instancias.

Da una información muy útil porque no sólo refleja los errores producidos sino también informa del tipo de éstos.

Nuestro Primer Clasificador/Predictor

Matriz de Confusión de VACAS:

Clasificar un conjunto de vacas en dos clases gordas y flacas.

Matriz de Confusión Ejemplo resultado de Clasificador:

Gordas	Flacas	
32	4	Gordas
4	43	Flacas

M diagonal principal → Aciertos

Da una información muy útil porque no sólo refleja los errores producidos sino también informa del tipo de éstos.

Nuestro Primer Clasificador/Predictor

Clasificador:

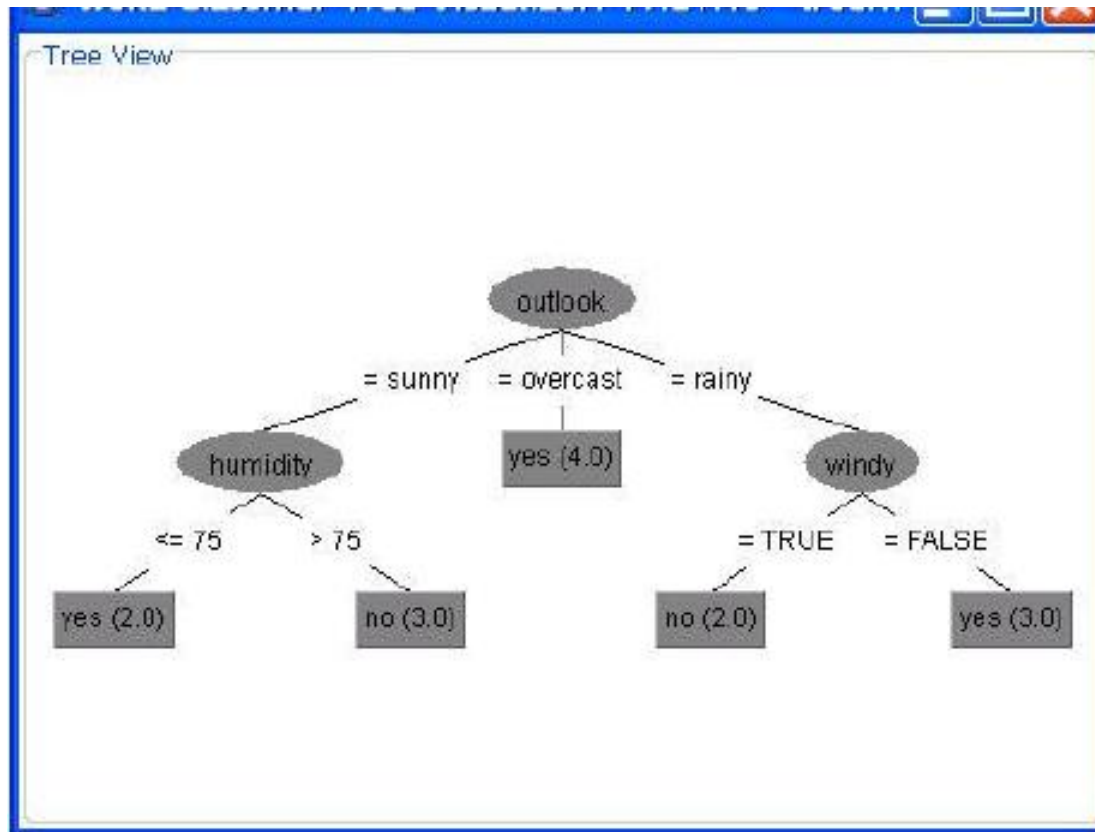
- Algoritmo clásico de aprendizaje de árboles de decisión - Predictor
- C4.5 = J48
- Use Training Set
- Resultados??
 - precisión máxima (100%) por lo que todas las medidas de error son 0

Arboles de decisión

- Algoritmo C4.5 (J48 en Java)
 - Es capaz de utilizar atributos tanto continuos como discretos: para gestionar los atributos continuos, crea umbrales y divide los nodos en función de dichos umbrales.
 - Gestiona datos con valores perdidos: A la hora de calcular la ganancia de información y la entropía, estos atributos simplemente no se utilizan.
 - Gestión de atributos con diferente coste
 - Poda de los árboles después de la creación

Nuestro Primer Clasificador/Predictor

Visualización:



Nuestro Primer Clasificador/Predictor

Mejora → AID:

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Datos en: iris.arff

- Atributos numéricos
- Clase Nominal
- Balanceado
- Punto de partida?? → Clasificador ZeroR → Por qué??

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

ZeroR → Por qué??

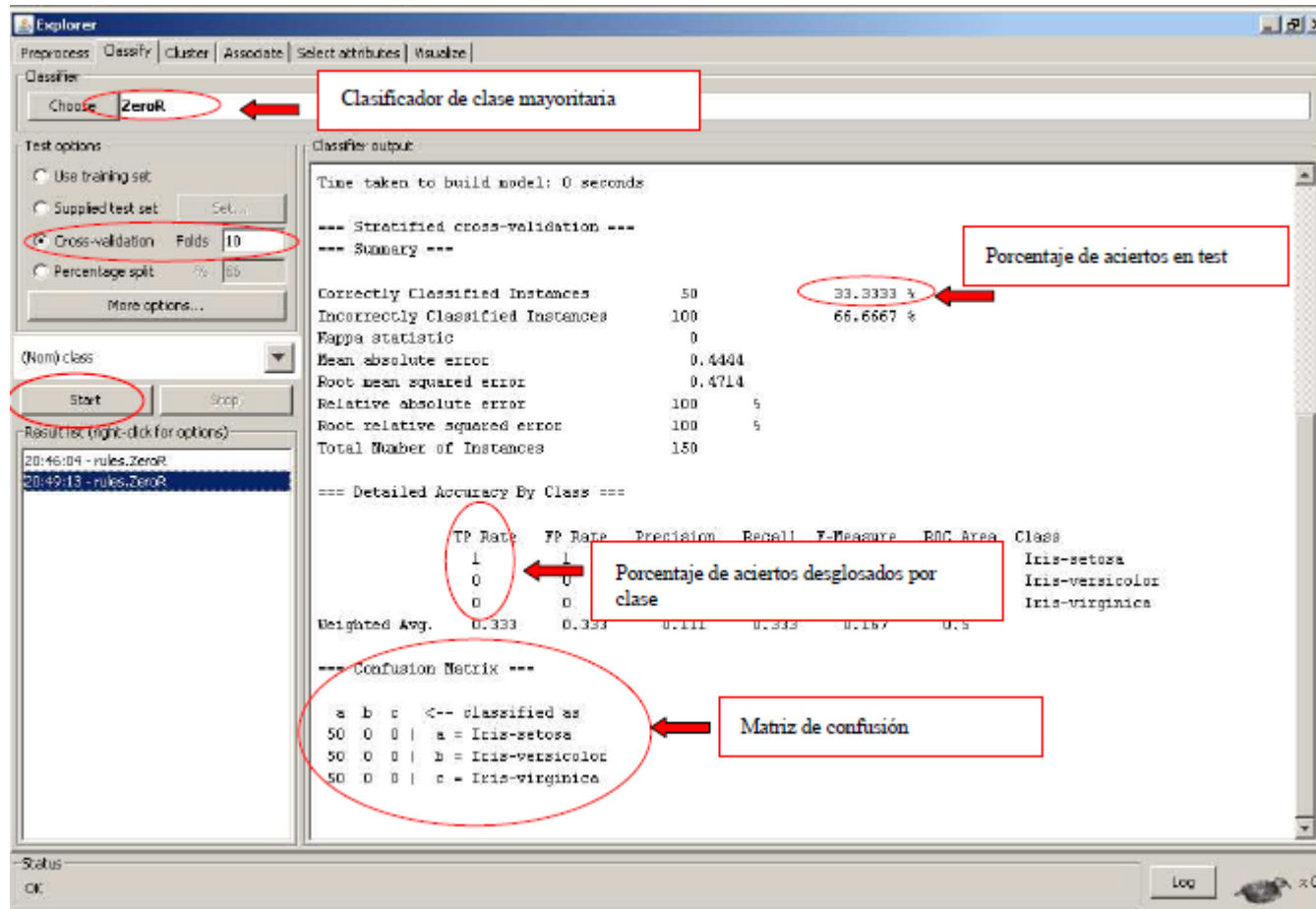
- Clasifica a todos los datos con la clase de la clase mayoritaria. Es decir, si el 90% de los datos son positivos y el 10% son negativos, clasificará a todos los datos como positivos.
- El porcentaje de aciertos que obtengamos con el, es el que habrá que superar con el resto de clasificadores.

Nuestro caso:

- Crossvalidation (10) → 33% de aciertos??? → lógico, puesto que las tres clases tienen 50 datos y ninguna es la mayoritaria. Sólo acierta la clase mayoritaria, o la primera, en caso de que ninguna sea mayoritaria.
- En el porcentaje de aciertos desglosado por clase → TP rate, o True Positive Rate → 100%

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.



The screenshot shows the WEKA Explorer interface with the ZeroR classifier selected. The 'Test options' section shows 'Cross-validation' with 'Folds' set to 10. The 'Start' button is highlighted. The 'Classifier output' section displays the following results:

Time taken to build model: 0 seconds

--- Stratified cross-validation ---

--- Summary ---

Metric	Value	Percentage
Correctly Classified Instances	50	33.3333 %
Incorrectly Classified Instances	100	66.6667 %
Kappa statistic	0	
Mean absolute error	0.4444	
Root mean squared error	0.4714	
Relative absolute error	100 %	
Root relative squared error	100 %	
Total Number of Instances	150	

--- Detailed Accuracy By Class ---

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	1	1	1	1	Iris-setosa
	0	0	0	0	0	0	Iris-versicolour
	0	0	0	0	0	0	Iris-virginica
Weighted Avg.	0.333	0.333	0.111	0.333	0.167	0.5	

--- Confusion Matrix ---

a	b	c	<-- classified as
50	0	0	a = Iris-setosa
50	0	0	b = Iris-versicolour
50	0	0	c = Iris-virginica

Annotations in the image highlight the following elements:

- Clasificador de clase mayoritaria**: Points to the ZeroR classifier selection.
- Porcentaje de aciertos en test**: Points to the 33.3333 % value in the summary table.
- Porcentaje de aciertos desglosados por clase**: Points to the TP Rate column in the Detailed Accuracy By Class table.
- Matriz de confusión**: Points to the Confusion Matrix table.

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

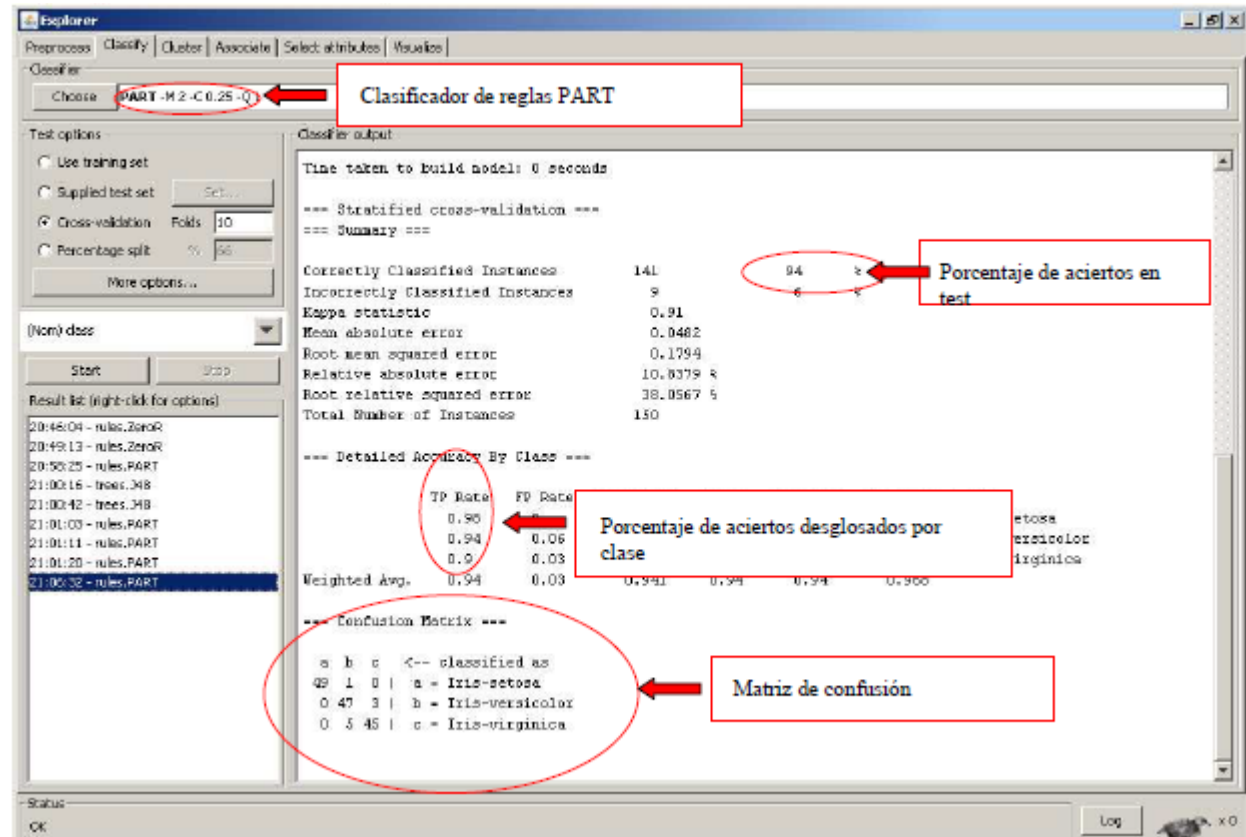
Mejora → PART, que construye reglas.

- Evol en C4.5
- Basado en divide y vencerás
- No realiza proceso de optimización Global
- Mayor coste Computacional

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Mejora → PART



Clasificador de reglas PART

Test options:

- ☐ Use training set
- ☐ Supplied test set
- ☒ Cross-validation Folds: 10
- ☐ Percentage split %: 50

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 20:46:04 - rules.ZeroR
- 20:49:13 - rules.ZeroR
- 20:50:25 - rules.PART
- 21:00:16 - trees.J48
- 21:00:42 - trees.J48
- 21:01:00 - rules.PART
- 21:01:11 - rules.PART
- 21:01:20 - rules.PART
- 21:06:32 - rules.PART

Classifier output

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	141
Incorrectly Classified Instances	9
Kappa statistic	0.91
Mean absolute error	0.0482
Root mean squared error	0.1794
Relative absolute error	10.8379 %
Root relative squared error	38.0567 %
Total Number of Instances	150

94 % **Porcentaje de aciertos en test**

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate
setosa	0.90	0.06
versicolor	0.94	0.03
virginica	0.9	0.03
Weighted Avg.	0.94	0.03

Porcentaje de aciertos desglosados por clase

=== Confusion Matrix ===

a	b	c	<-- classified as
49	1	0	a = Iris-setosa
0	47	3	b = Iris-versicolor
0	5	45	c = Iris-virginica

Matriz de confusión

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Mejora → PART

- Porcentaje de aciertos es 94% → bastante bueno vs 33% del caso base (ZeroR).
- Desglose de los aciertos por clase bastante → partíamos de un conjunto de datos balanceado
(si no hubiera estado balanceado, es muy posible que la clase mayoritaria se hubiera acertado mejor que las minoritarias).
- En la matriz de confusión, podemos ver como los datos acertados están en la diagonal, y los fallados fuera de ella.

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

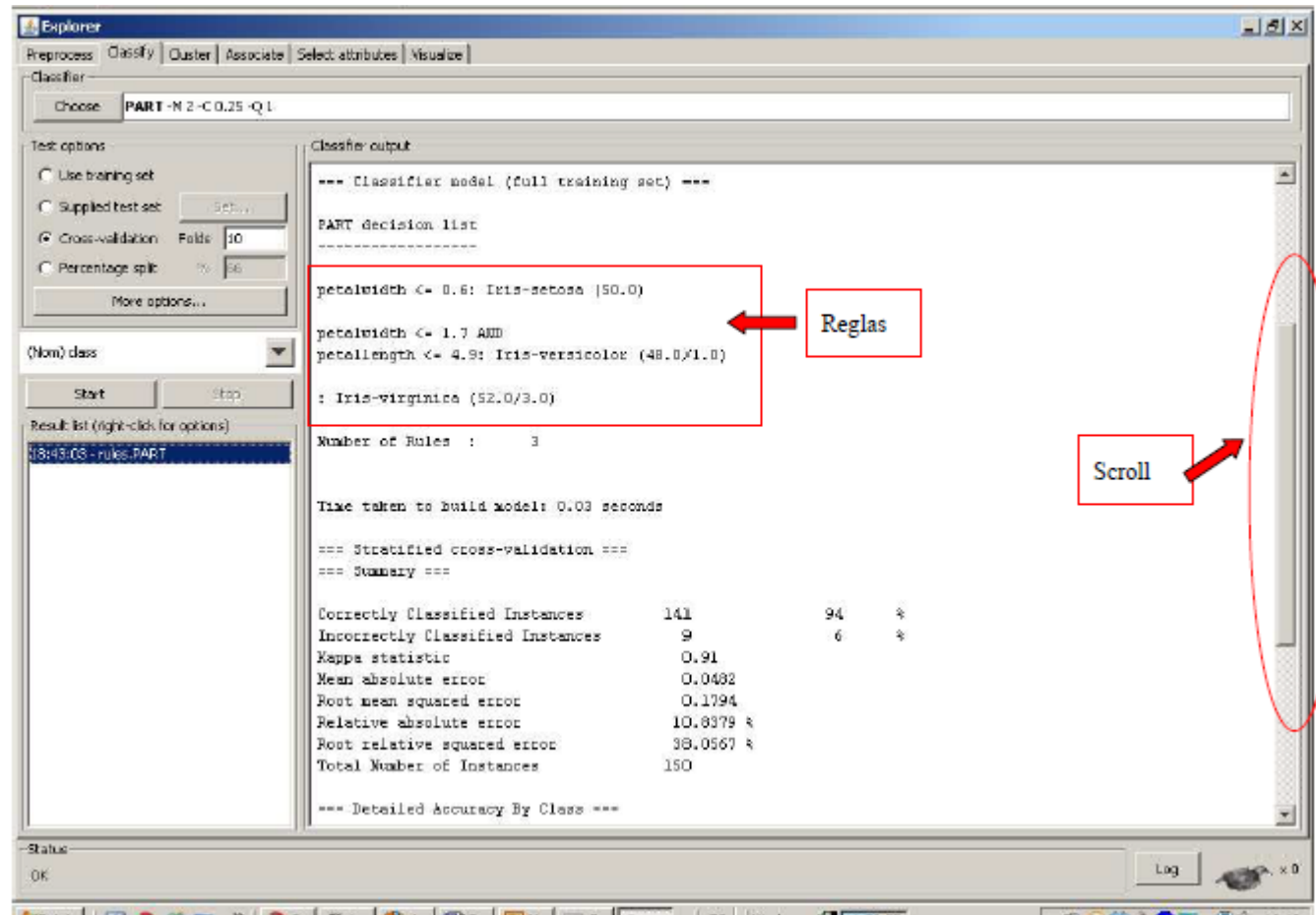
Mejora → PART

- Porcentaje de aciertos es 94% → bastante bueno vs 33% del caso base (ZeroR).
- Desglose de los aciertos por clase bastante → partíamos de un conjunto de datos balanceado
(si no hubiera estado balanceado, es muy posible que la clase mayoritaria se hubiera acertado mejor que las minoritarias).
- En la matriz de confusión, podemos ver como los datos acertados están en la diagonal, y los fallados fuera de ella.
- Visualización?? → No generan Arbol → Reglas !!

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Mejora → PART



Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Mejora → PART

- Hay tres reglas
- Los números (a/b) entre paréntesis indican cuantos datos clasifica la regla (a) y cuantos de ellos lo hace incorrectamente (b)
- IF (petalwidth \leq 0.6) THEN clase = Iris-setosa (50.0)
- IF (petalwidth \leq 1.7) AND (petallength \leq 4.9) THEN clase = Iris-versicolor (48.0/1.0)

Y si no se cumplen ninguna de las condiciones, entonces clasificar por omisión

- Clase = Iris-virginica (52.0/3.0)
- Obsérvese que las reglas sólo utilizan los atributos **petalength** y **petalwidth**, que son los que habíamos comprobado antes que eran los mejores (al menos, considerados de manera individual cada uno de ellos).

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Mejora → J48 y SVM

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Mejora → Parámetros básicos de configuración de J48

- **Confidence factor**, está relacionado con la complejidad, el cual controla el tamaño del árbol de decisión construido
- La complejidad del clasificador tiene que ver con el sobreaprendizaje (overfitting)
- Cuanto más pequeño es este parámetro, mas simple tiende a ser el clasificador (menos nodos), y viceversa.
- Varía entre 0 y 1

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Mejora → Parámetros básicos de configuración de J48

- **Confidence factor**, está relacionado con la complejidad, el cual controla el tamaño del árbol de decisión construido
- La complejidad del clasificador tiene que ver con el sobreaprendizaje (overfitting)
- Cuanto más pequeño es este parámetro, mas simple tiende a ser el clasificador (menos nodos), y viceversa.
- Varía entre 0 y 1
- Test: Con 0.001 y con 1

Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

Conclusion:

Con $CF = 0.001$ el árbol se simplifica (4 hojas y 7 nodos) → % de aciertos disminuye (94%).

Con $CF = 1.0$, se mantiene (5 hojas y 9 nodos) → mantiene % de aciertos (96%).

Este caso no **sobreaprendizaje** y que la complejidad del modelo era la adecuada.

Caso de que hubiese habido

Si **sobreaprendizaje** → modelo más simple con mejor % de aciertos mejor

Si el modelo muy simple → **underfitting** (es decir, que el modelo no es lo suficientemente complejo para llevar a cabo el aprendizaje de manera correcta).

En este caso, la única ventaja del árbol más pequeño es que es (ligeramente) más sencillo de entender, pero tiene la contraprestación de ser menos preciso.

Mejorar Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

El objetivo es mejorar una de las clases concretas de un clasificador.

Partimos del Clasificador PART (reglas) sobre Iris.arff

Recordamos nuestra matriz de confusión:

=== Confusion Matrix ===

TP Rate Class

0.98	Iris-setosa
0.94	Iris-versicolor
0.9	Iris-virginica

a	b	c	<-- classified as
49	1	0	a = Iris-setosa
0	47	3	b = Iris-versicolor
0	5	45	c = Iris-virginica

El objetivo es mejorar la clase menor (0.9) Iris-Virginica

Mejorar Segundo Ejemplo Clasificador/Predictor

Clasificación de Flores en función de alto y ancho de los pétalos y Sépalos.

El objetivo es mejorar una de las clases concretas de un clasificador.

- PART asigna erróneamente 5 datos de la clase c a la clase b, 3 datos de la b a la clase c y 1 dato de a, a b
- Ya sabemos que la clase que peor clasifica es la iris-virginica (la c) y en la matriz de confusión vemos que tiende a confundirla con la b (iris-versicolor)
- Mejorar el % de aciertos de la clase c, utilizando un meta-clasificador, el **costsensitive-Classifer**. → **Modificación de matriz de Costes**

Explorer, selección y filtrado de atributos

Tres posibilidades:

- Evaluación de atributos. Por ejemplo:
 - Método de búsqueda = Ranker
 - Método de evaluación = InfoGainAttributeEval
- Evaluación de conjuntos de atributos
 - Filter. Por ejemplo:
 - Método de búsqueda = Greedy Stepwise
 - Método de evaluación = CfsSubsetEval
 - Método Wrapper. Por ejemplo:
 - Método de búsqueda = Greedy Stepwise
 - Método de evaluación = ClassifierSubsetEval

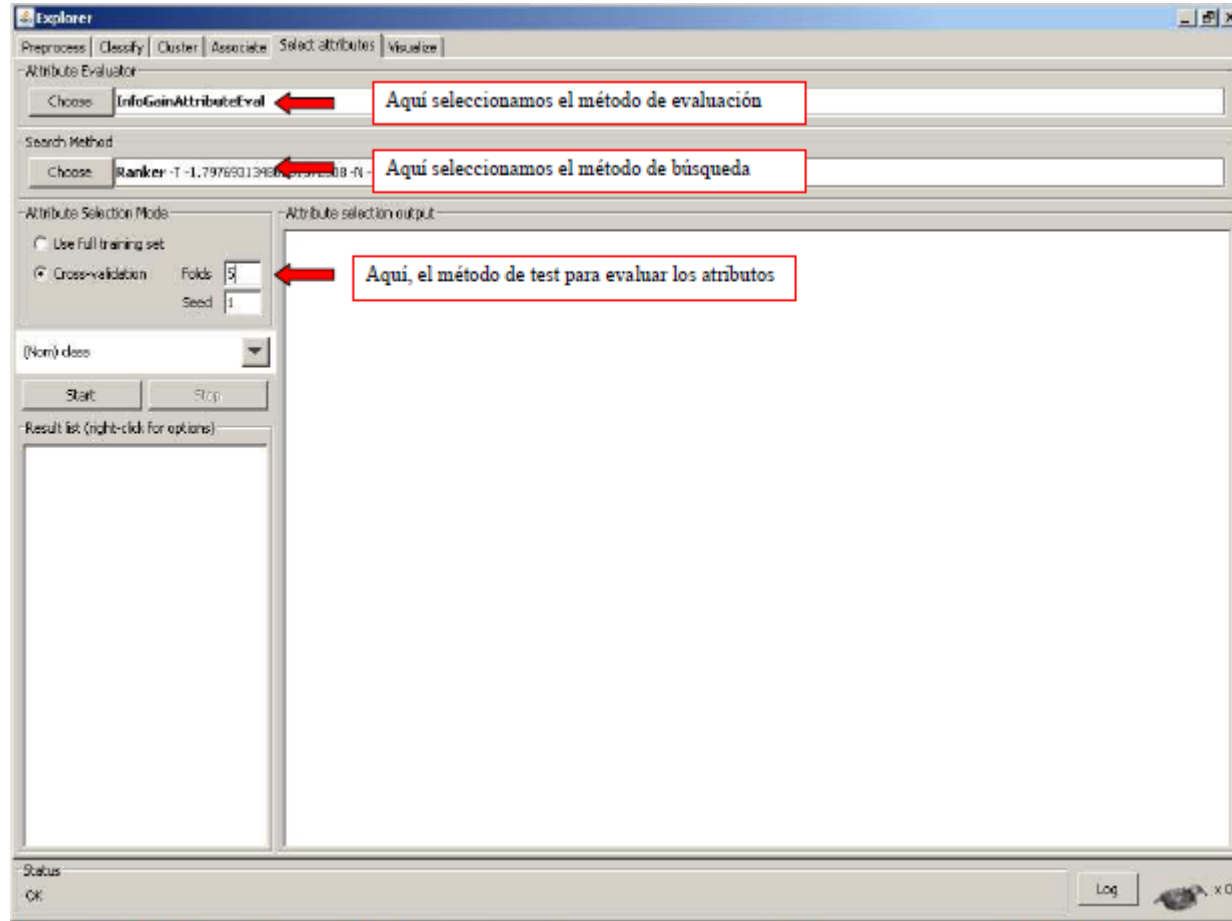
Explorer, selección y filtrado de atributos

Ejemplo: Seleccionar-Filtrar atributos sobre clasificación de Flores

- Ranker, con InfoGain.
- El método de test para evaluar los atributos será de **crossvalidation** de 5 hojas.
- Evaluación 5 veces más lenta, pero más precisa
- Compromiso: En entornos con muchos datos o atributos, habrá que reducir este número, o usar Use full training set, para que Weka tarde un tiempo razonable → reducirá la confianza que tengamos en el subconjunto de atributos seleccionado

Explorer, selección y filtrado de atributos

Ejemplo: Seleccionar-Filtrar atributos sobre clasificación de Flores



Explorer, selección y filtrado de atributos

Ejemplo: Seleccionar-Filtrar atributos sobre clasificación de Flores

=== Attribute selection 5 fold cross-validation (stratified), seed: 1 ===

average merit	average rank	attribute
1.386 +- 0.022	1.4 +- 0.49	4 petalwidth
1.368 +- 0.017	1.6 +- 0.49	3 petallength
0.711 +- 0.053	3 +- 0	1 sepallength
0.323 +- 0.059	4 +- 0	2 sepalwidth

- Ranker → ordenaba los atributos
- **average merit** (y su desviación típica): se refiere a la media de la correlaciones (medidas con InfoGain) en los cinco ciclos de validación cruzada.
- **average rank** (y su desviación típica): se refiere al orden medio en el que quedó cada atributo en cada uno de los cinco ciclos.

Explorer, selección y filtrado de atributos

Ejemplo: Seleccionar-Filtrar atributos sobre clasificación de Flores

=== Attribute selection 5 fold cross-validation (stratified), seed: 1 ===

average merit	average rank	attribute
1.386 +- 0.022	1.4 +- 0.49	4 petalwidth
1.368 +- 0.017	1.6 +- 0.49	3 petallength
0.711 +- 0.053	3 +- 0	1 sepallength
0.323 +- 0.059	4 +- 0	2 sepalwidth

- Por ejemplo, como para sepallength y sepalwidth, la desviación típica es de cero, eso quiere decir que sepallength quedó como tercer atributo en los cinco ciclos de validación cruzada, y que sepalwidth quedó siempre el cuarto.
- Petalwidth y petallength debieron de quedar ambos a veces primero y a veces segundo, por eso el orden medio es de 1.4.
- En resumen, petalwidth y petallength son los mejores atributos, a bastante diferencia de los 2 siguientes. Sepalwidth se ve que es particularmente malo.

Explorer, selección y filtrado de atributos

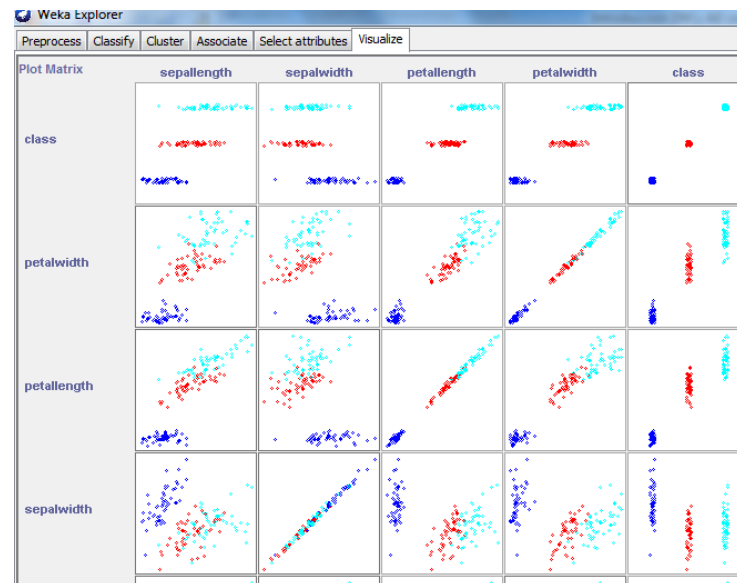
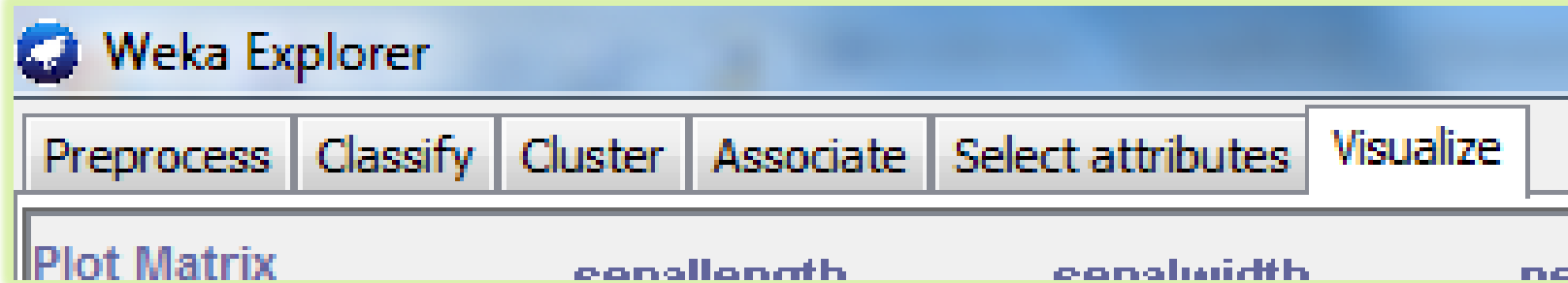
Ejemplo: Seleccionar-Filtrar atributos sobre clasificación de Flores

=== Attribute selection 5 fold cross-validation (stratified), seed: 1 ===

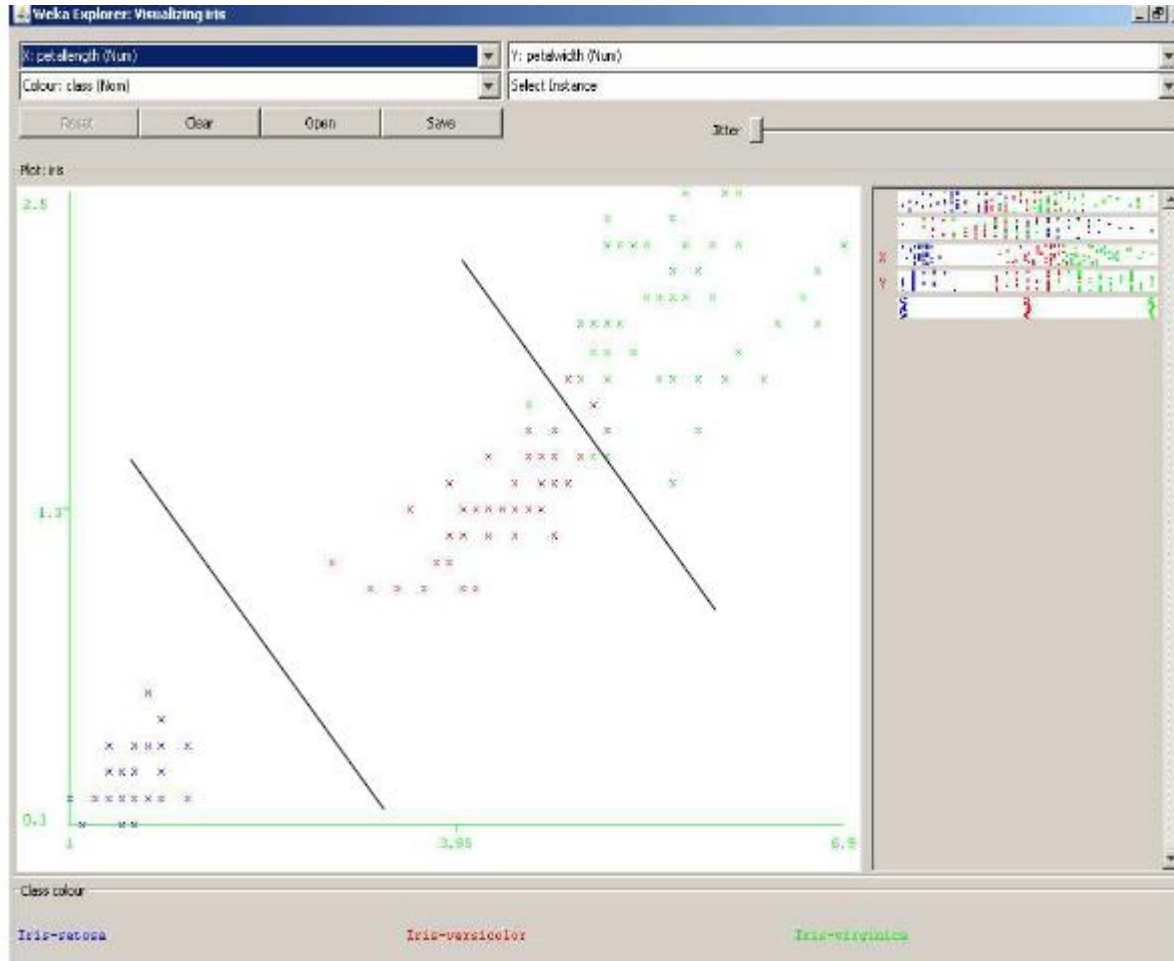
number of folds (%)	attribute
0(0 %)	1 sepallength
0(0 %)	2 sepalwidth
5(100 %)	3 petallength
5(100 %)	4 petalwidth

- Método de búsqueda = Greedy Stepwise
- Método de evaluación = CfsSubsetEval
- Interpretación → petallength y petalwidth fueron seleccionados en cada uno de los 5 folds de validación cruzada (o sea, siempre), mientras que sepallength y sepalwidth no fueron seleccionados en ningún fold (o sea, nunca).
- Nuevamente, petallength y petalwidth vuelven a ser seleccionados..

Visualizado de atributos



Visualizado de atributos



Visualizado de atributos

En la gráfica:

- Atributos de Petalos → Mas importantes. juntos, separamos casi perfectamente las tres clases.
- Se puede ver porque la clase iris-setosa (azul oscuro) era la que se clasificaba mejor y porqué las otras dos (roja y verde) presentaban ciertos errores (es casi imposible trazar una frontera sencilla entre ambas clases que las separe de manera perfecta).
- Si se prueba con los dos atributos de los sépalos, se verá que hay mucho más solape entre las clases. A la hora de visualizar, es importante que los dos atributos no sean redundantes. Si lo son, ambos aportarán la misma información, con lo que la separación será menos clara.

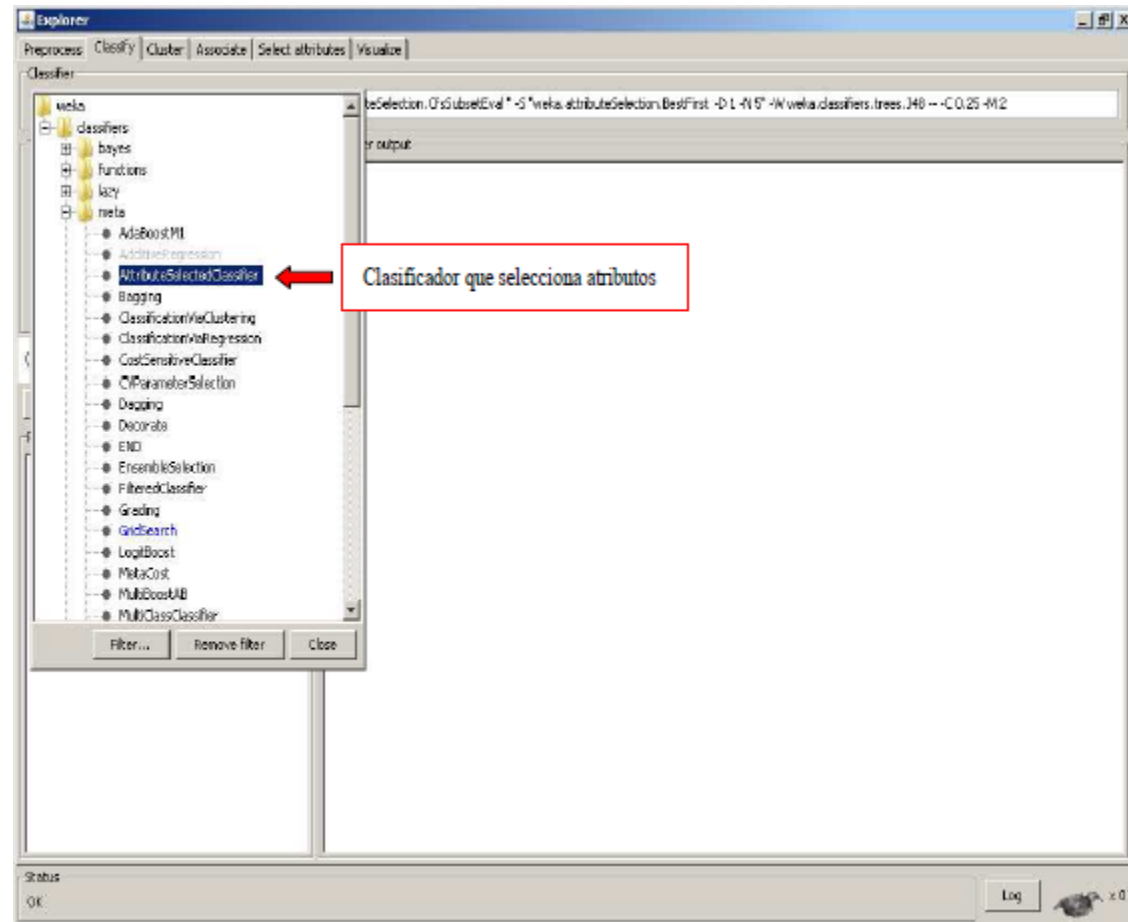
Clasificadores con Filtrado de Atributos

- La selección de atributos mejora (o empeora) el porcentaje de aciertos esperado?
- La selección de atributos afecta a si los modelos que se construyen son más complejos o más simples?
- Lo que necesitamos es un **meta-clasificador** que primero pase un filtro de selección de atributos, y después realice aprendizaje (y test) utilizando exclusivamente los atributos seleccionados.

selected classifier

Clasificadores con Filtrado de Atributos

selected classifier



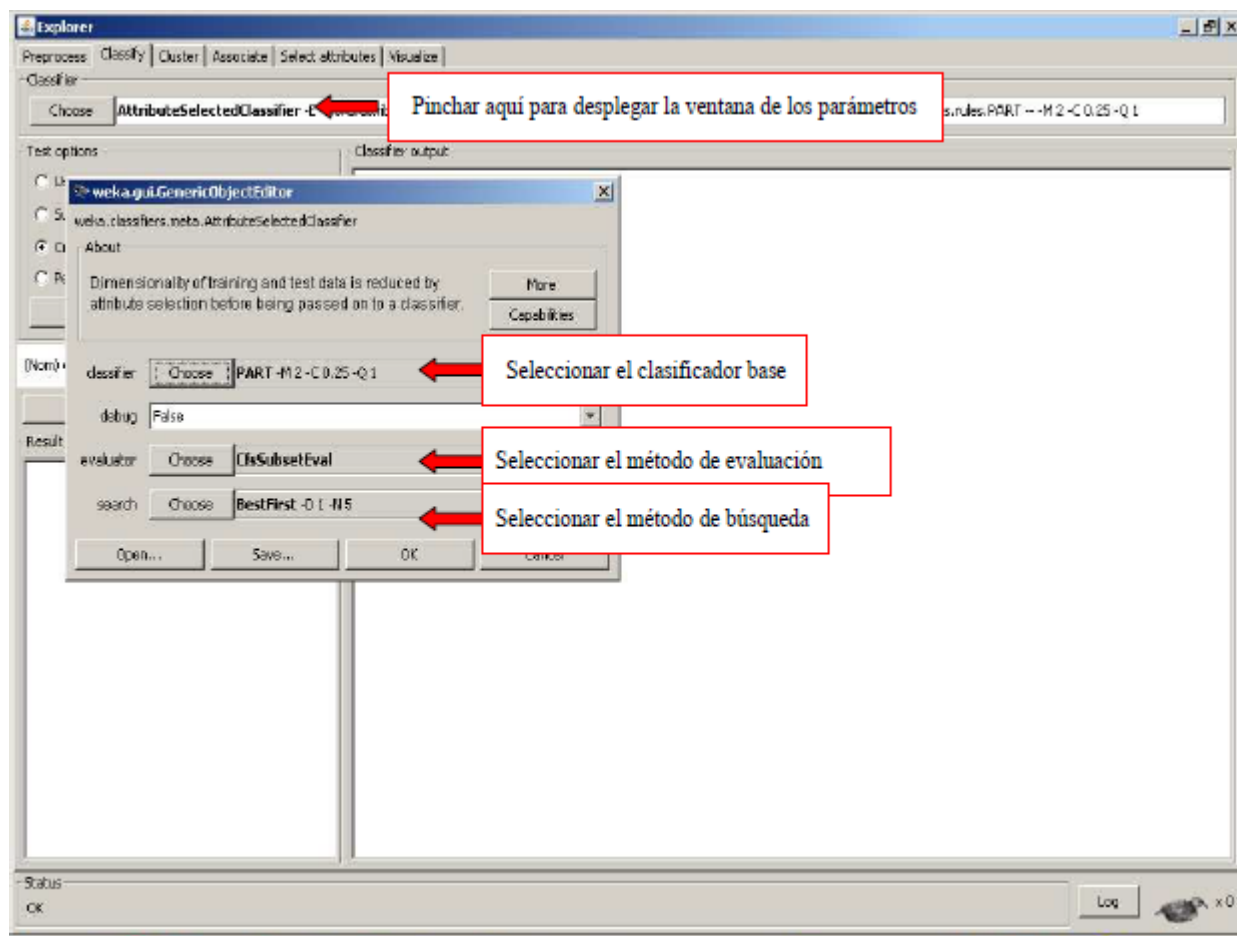
Clasificadores con Filtrado de Atributos

selected classifier:

búsqueda=CfsSubsetEval

evaluación = BestFirst

Clasificado = PART



Clasificadores con Filtrado de Atributos

selected classifier:

búsqueda=CfsSubsetEval

evaluación = BestFirst

Clasificado = PART

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	143	95.3333 %
--------------------------------	-----	-----------

Selected attributes: 3,4 : 2

petallength
petalwidth|

Clasificadores con Filtrado de Atributos

selected classifier:

búsqueda=CfsSubsetEval

evaluación = BestFirst

Clasificado = PART

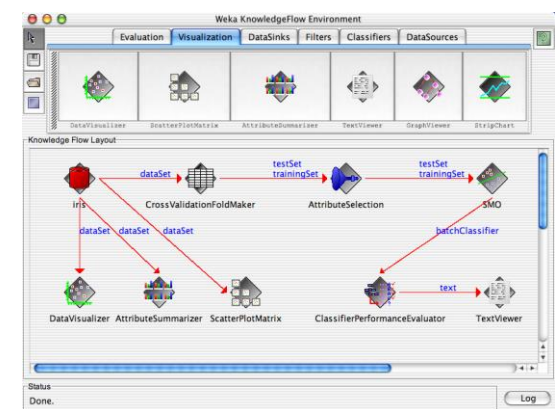
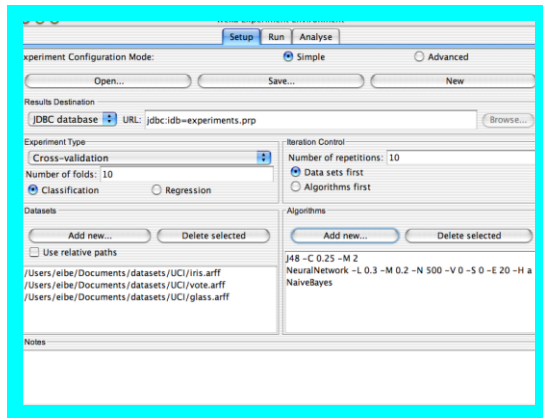
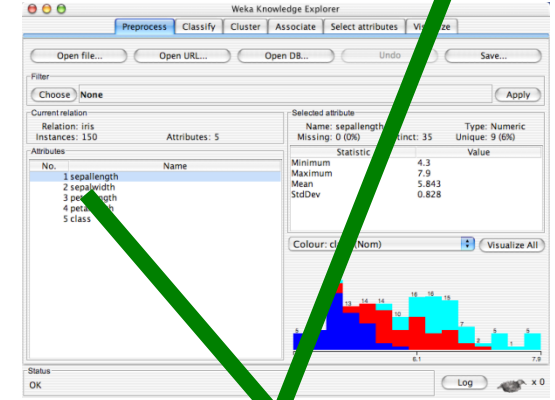
=== Stratified cross-validation ===

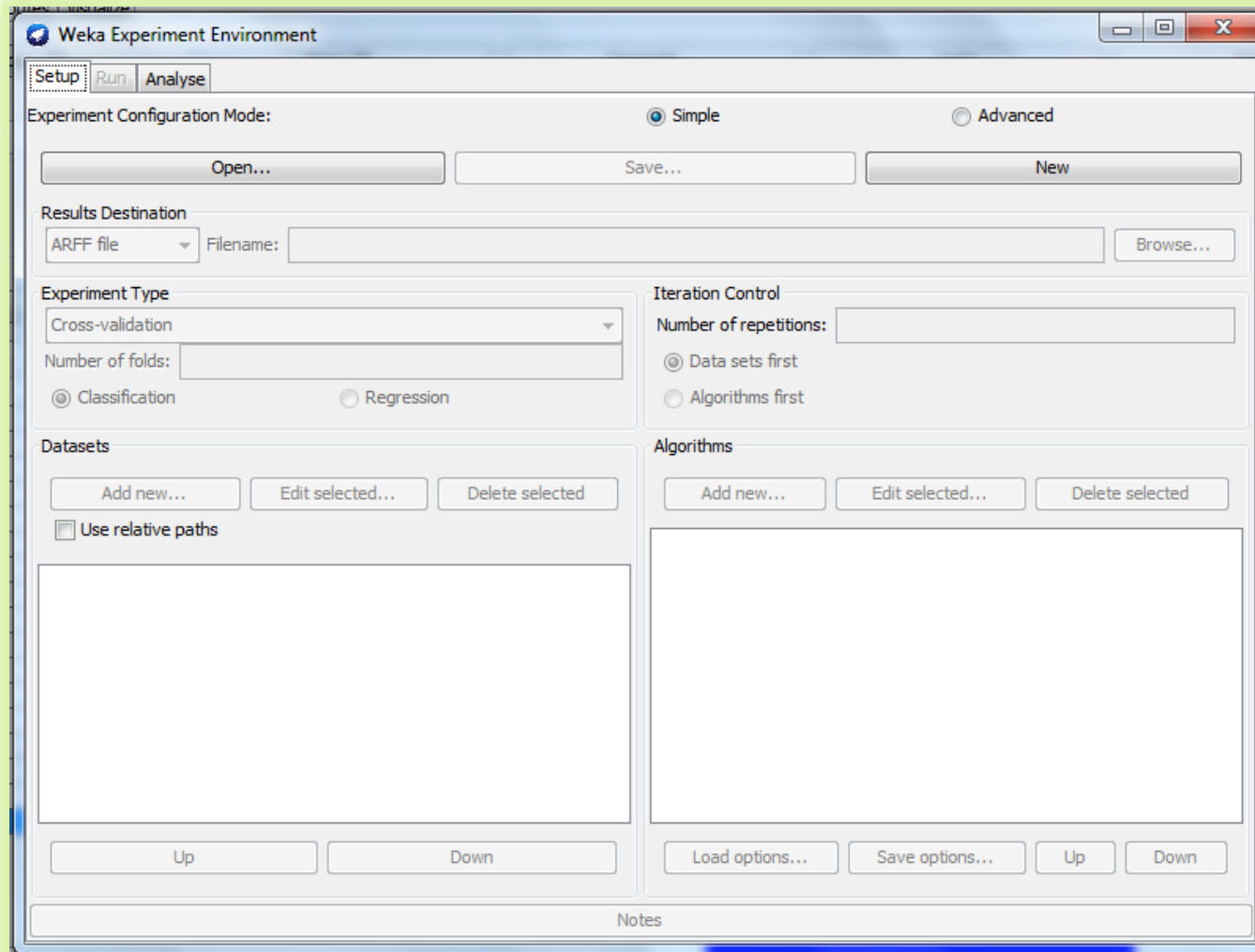
=== Summary ===

Correctly Classified Instances	143	95.3333 %
--------------------------------	-----	-----------

Selected attributes: 3,4 : 2

petallength
petalwidth|





Experimenter

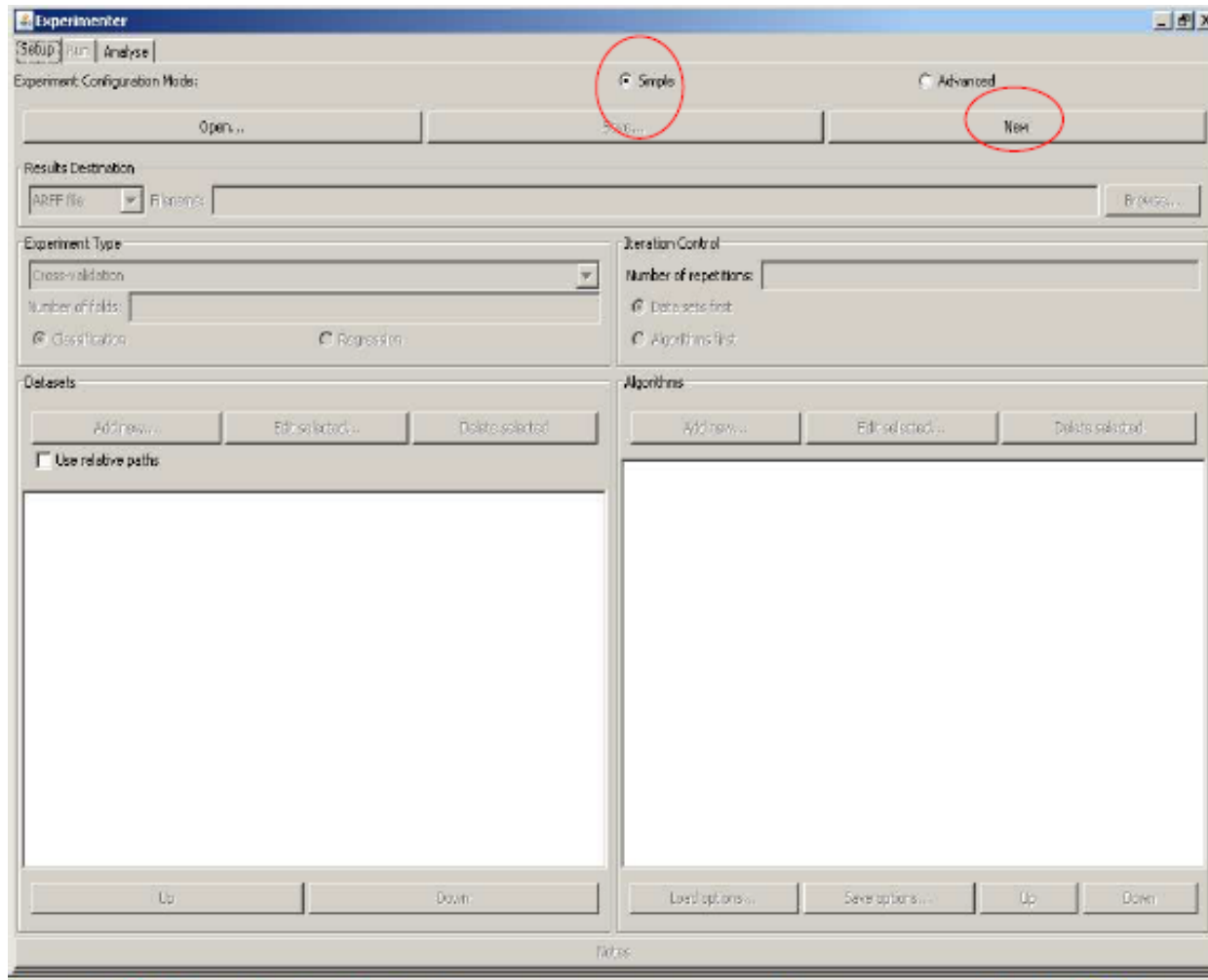
- El modo experimentador (**Experimenter**) es un modo muy útil para aplicar método de clasificación/regresión sobre un gran conjunto de datos
- También se usa para comparar uno o varios métodos y luego poder realizar contrastes estadísticos entre ellos y obtener otros índices estadísticos.
- El Experimenter nos dirá si las diferencias aparentes en porcentajes de aciertos de distintos algoritmos son estadísticamente significativas, o son debidas al azar.

Experimenter

- Los Modulos de uso de Experimenter son:
 - Setup (configura)
 - Run (ejecuta)
 - Analyse (análisis estadístico)
- Comenzaremos configurando nuestro experimento. Tenemos dos opciones: **Simple** y Advanced.

Experimenter

- Los Modulos de uso de Experimenter son:
 - Setup (configura)
 - Run (ejecuta)
 - Analyse (análisis estadístico)
- Comenzaremos configurando nuestro experimento. Tenemos dos opciones: **Simple** y Advanced.
- Definir fichero de Configuración: ajustes, ficheros involucrados, notas, etc, pertenecientes a un experimento
- Save → Para generar el .exp (No Autoguardado)



Experimenter

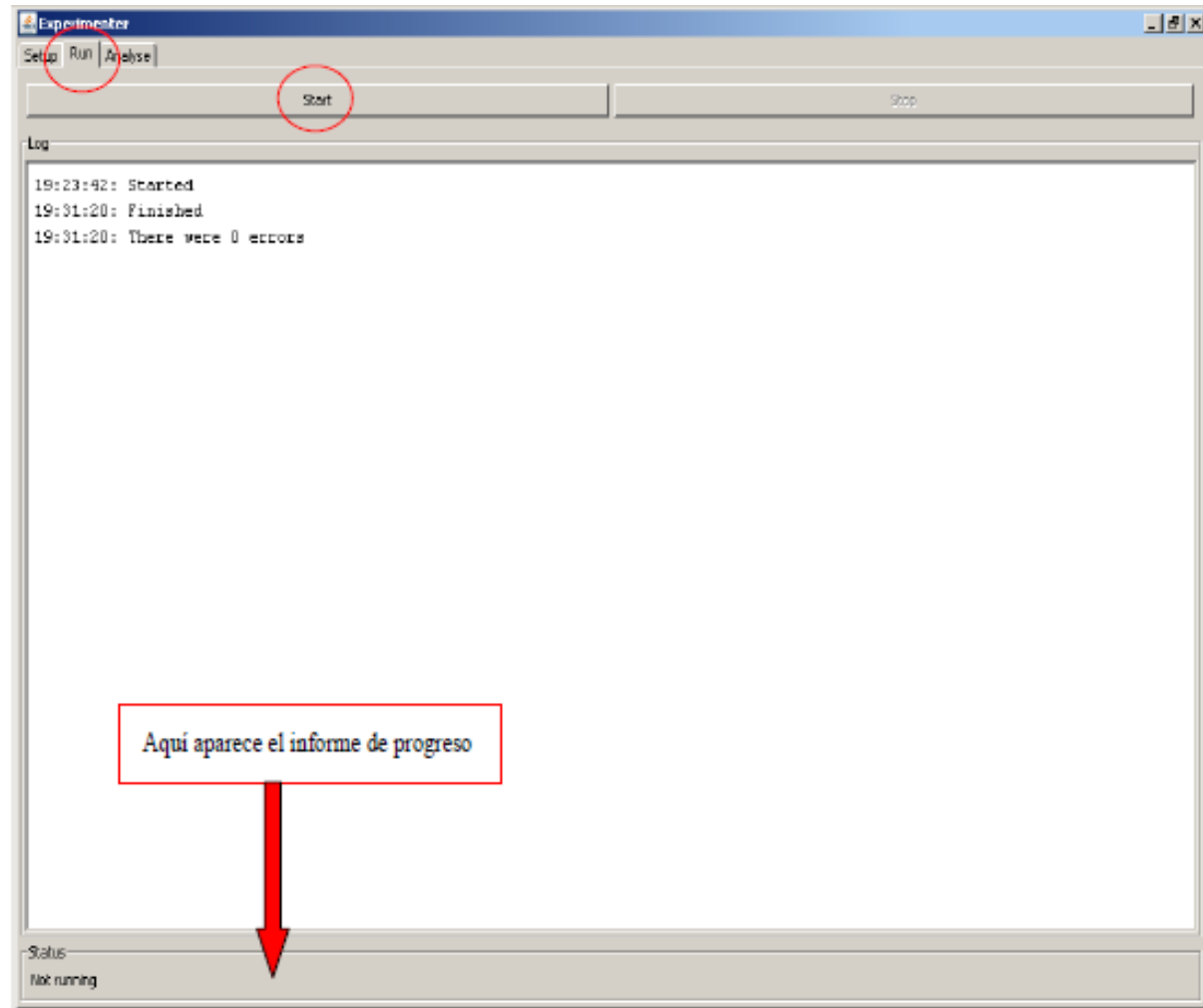
- Configurar ruta y formato de guardado
- Experiment type: Modo de Training y validación
 - · Crossvalidation
 - · Train-test percentage split (data randomized): primero desordena aleatoriamente los datos y después coge el primer 66% para construir el clasificador y el 34% restante para hacer el test (calcular el porcentaje de aciertos esperado)
 - · Train-test percentage split (order preserved): hace lo mismo que el anterior pero no desordena el conjunto de datos antes de dividirlos en entrenamiento y test

Experimenter

- Iteration control: definiremos el número de repeticiones de nuestro experimento, especificando si queremos que se realicen primero los archivos de datos o los algoritmos
- Datasets: indicar a Weka qué archivos de datos queremos que formen parte de nuestro experimento
- Algoritmos: Los algoritmos serán:
 - ZeroR (rules)
 - J48 (trees) (0,25 – 0,8)
 - PART (rules) (0,25 – 0,8)
 - IB1 (lazy).

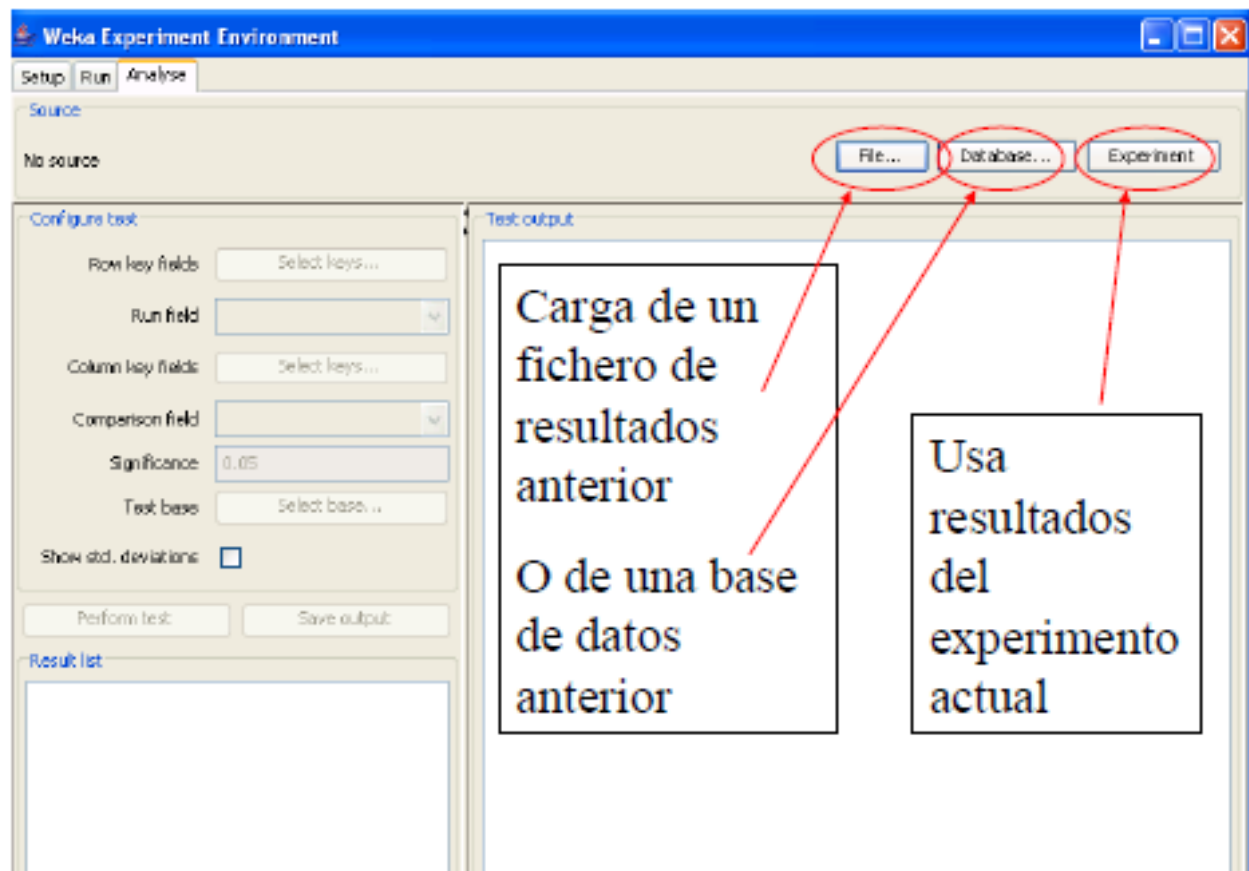
Experimenter : Run

- Start



Experimenter : Run

- Seleccionaremos Experiment.

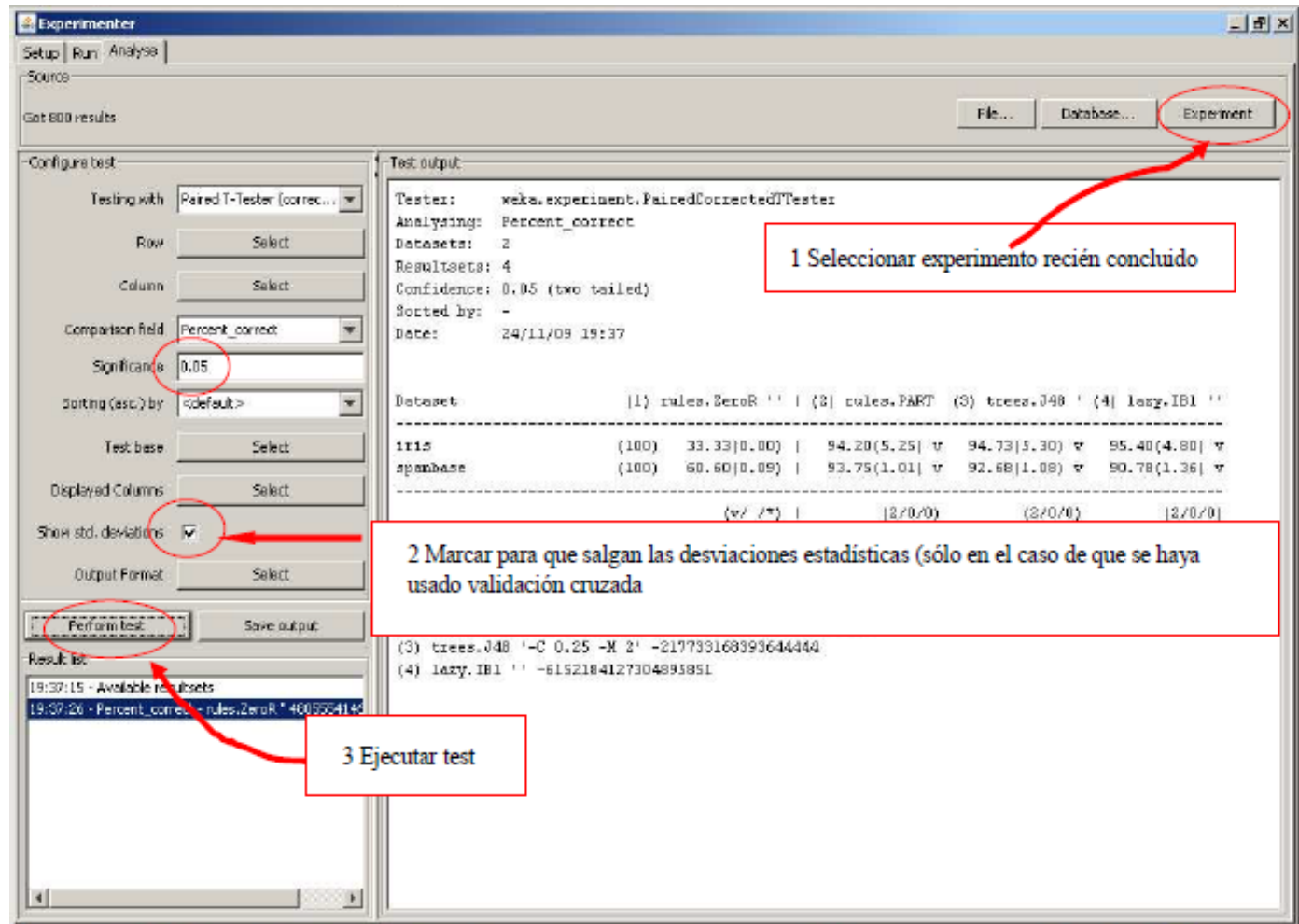


Experimenter : Run

- Seleccionaremos Experiment.
- Segundo, marcamos la opción de mostrar las desviaciones típicas show std. deviations (caso de que se haya usado crossvalidation y queramos ver las desviaciones con respecto a la media).
 - El resto de parámetros se pueden dejar como están.
 - El Experimenter encuentra que las diferencias son estadísticamente significativas, con un nivel alfa (significance) del 0.05. → Esto mas o menos quiere decir que la probabilidad de que el Experimenter os diga que una diferencia es significativa cuando realmente es debida al azar, es del 5%.
- Tercero, ejecutamos el test propiamente dicho (Perform Test).
- Resultados.

Experimenter : Run

- Experiment



The screenshot shows the WEKA Experimenter window with the following components and annotations:

- Buttons:** File..., Database..., Experiment (circled in red with annotation 1).
- Configure test section:**
 - Testing with: Paired T-Tester (corrected)
 - Row: Select
 - Column: Select
 - Comparison field: Percent correct
 - Significance: 0.05 (circled in red with annotation 2)
 - Sorting (asc.) by: <default>
 - Test base: Select
 - Displayed Columns: Select
 - Show std. deviations: ☒ (circled in red with annotation 2)
 - Output Format: Select
 - Perform test: (circled in red with annotation 3)
 - Save output
- Test output section:**
 - Tester: weka.experiment.PairedConnectedTTester
 - Analysing: Percent correct
 - Datasets: 2
 - Resultsets: 4
 - Confidence: 0.05 (two tailed)
 - Sorted by: -
 - Date: 24/11/09 19:37
- Table:**

Dataset	(1) rules.ZeroR	(2) rules.PART	(3) trees.J48	(4) lazy.IB1
iris	(100) 33.33(0.00)	94.20(5.25) v	94.73(5.30) v	95.40(4.80) v
svmbase	(100) 60.60(0.09)	93.75(1.01) v	92.68(1.08) v	90.78(1.36) v
	(v/ /*)	(2/0/0)	(2/0/0)	(2/0/0)
- Result list:**
 - 19:37:15 - Available resultsets
 - 19:37:26 - Percent correct - rules.ZeroR * 4605554144

Experimenter

- Resultados

Dataset	(1) rules.ZeroR ''	(2) rules.PART	(3) rules.PART	(4) trees.J48 '	(5) trees.J48 '	(6) lazy.IB1 ''
iris	(100) 33.33(0.00)	94.20(5.25) v	94.20(5.25) v	94.73(5.30) v	94.80(5.24) v	95.40(4.80) v
	(v/ /*)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)

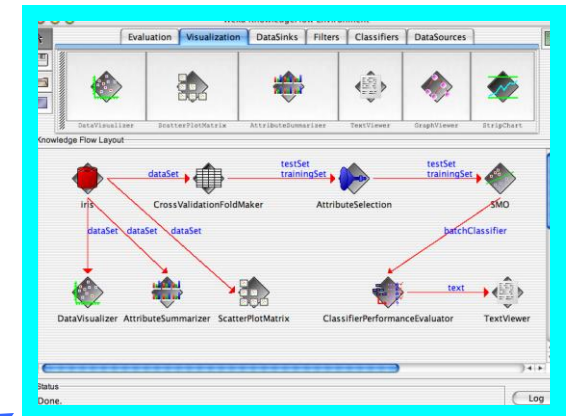
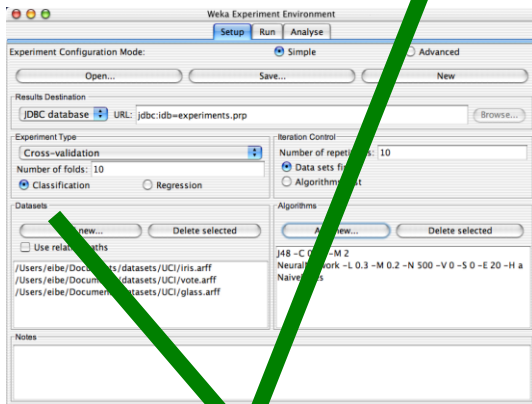
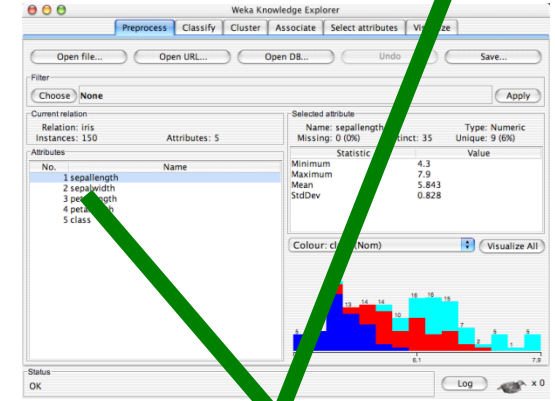
- El algoritmo con respecto al cual se hace la comparación, es el de más a la izquierda
- La v significa que la mejora de PART sobre ZeroR es estadísticamente significativa
- (2/0/0) Experimentos en los que mejora

Experimenter

- Resultados

Dataset	(1) rules.ZeroR ''	(2) rules.PART	(3) rules.PART	(4) trees.J48 '	(5) trees.J48 '	(6) lazy.IB1 ''
iris	(100) 33.33(0.00)	94.20(5.25) v	94.20(5.25) v	94.73(5.30) v	94.80(5.24) v	95.40(4.80) v
	(v/ /*)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)

- El algoritmo con respecto al cual se hace la comparación, es el de más a la izquierda
- La v significa que la mejora de PART sobre ZeroR es estadísticamente significativa
- (2/0/0) Experimentos en los que mejora

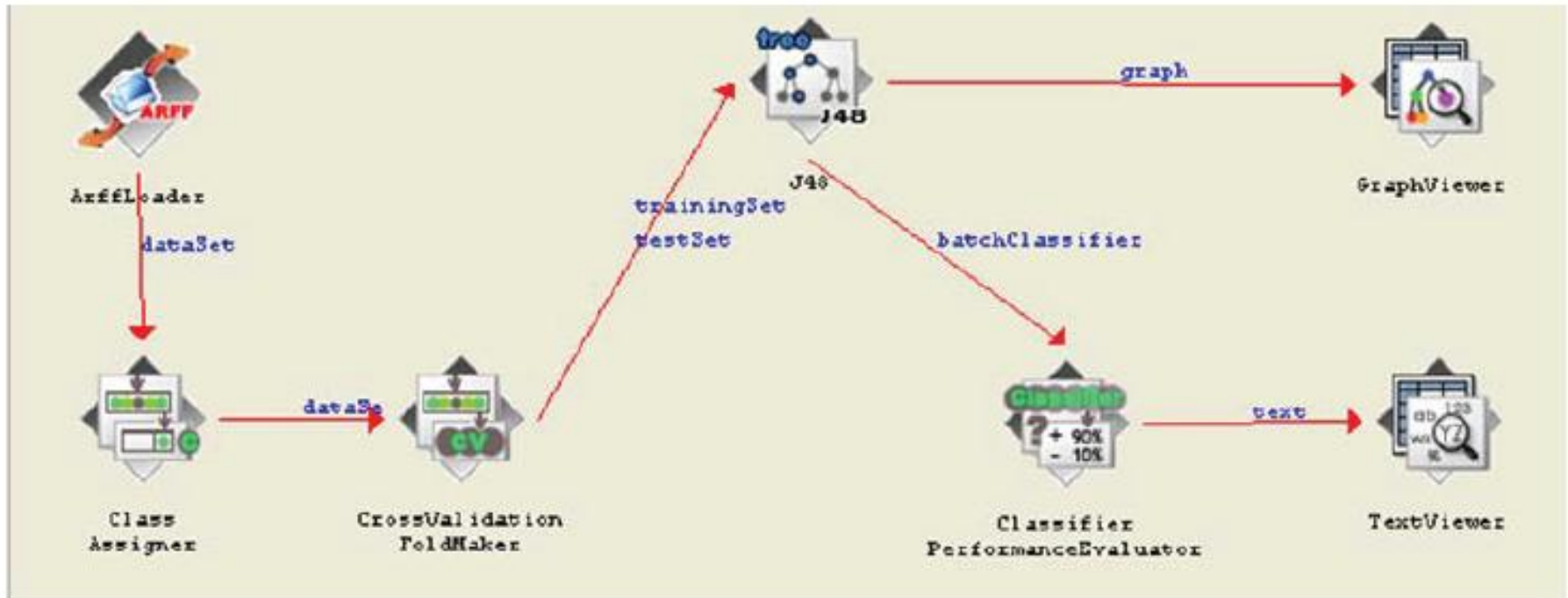


Knowledge flow

- Muestra de una forma más explícita el funcionamiento interno del programa
- Gráfico y se basa en situar en el panel de trabajo (zona gris), elementos base (situados en la parte superior de la ventana) de manera que creemos un “circuito” que defina nuestro experimento
- Permite crear configuraciones imposibles por Explorer
- Ejemplo Objetivo: Cargar un fichero ARFF y llevar a cabo una validación cruzada con C4.5

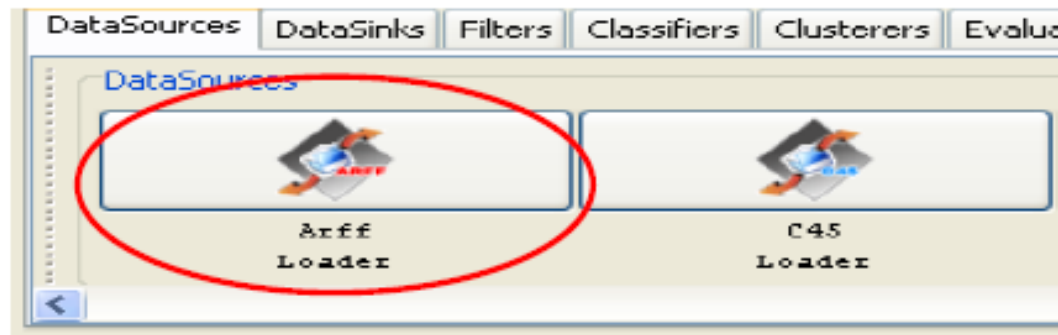
Knowledge flow

- Ejemplo



Knowledge flow

- Ejemplo
 - Objetivo: Cargar un fichero ARFF y llevar a cabo una validación cruzada con C4.5
 - Pulsar *Arff Loader* dentro de la pestaña *Data Sources*. Pinchar en el panel de Dibujo.

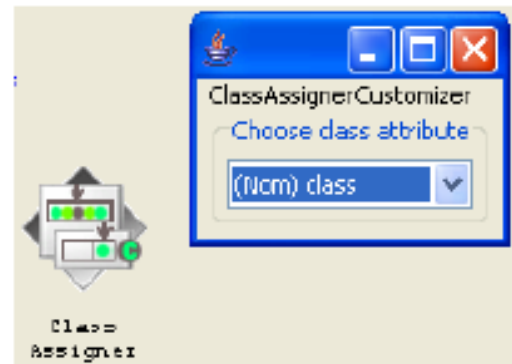


Knowledge flow

- Ejemplo
 - Con el botón derecho en el nodo *Arff Loader*, ir a *Configure*. Por ejemplo, trabajaremos con *iris.arff*.
 - Para indicar el atributo que será la clase, introducir un nodo *Class Asigner* (Etiqueta *Evaluation*).

Knowledge flow

- Ejemplo
 - Para conectar ambos nodos, pulsar con el botón derecho sobre *Arff Loader* y seleccionar la opción *Connections->dataset*.
 - Configurar el *Class Asigner* para tomar el atributo *Nom* como clase.

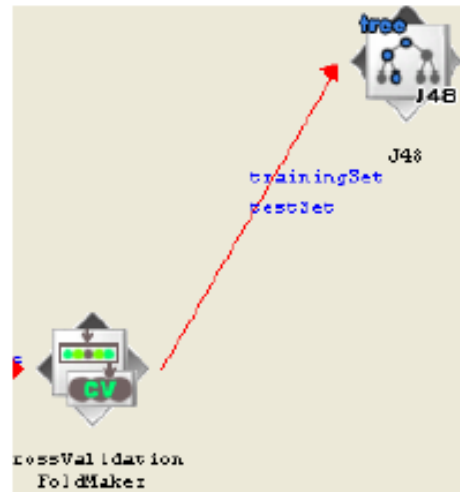


Knowledge flow

- Ejemplo
 - Hacer la misma operación para un *CrossValidation Foldmaker*.
 - Configurarlos con 10 folds y poner una semilla cualquiera.

Knowledge flow

- Ejemplo
 - Introducir un nodo *J48* (en pestaña *classifiers*).
 - Conectarlo con *CrossValidation Foldmaker* mediante la conexión *trainingSet* y *testSet*.



Knowledge flow

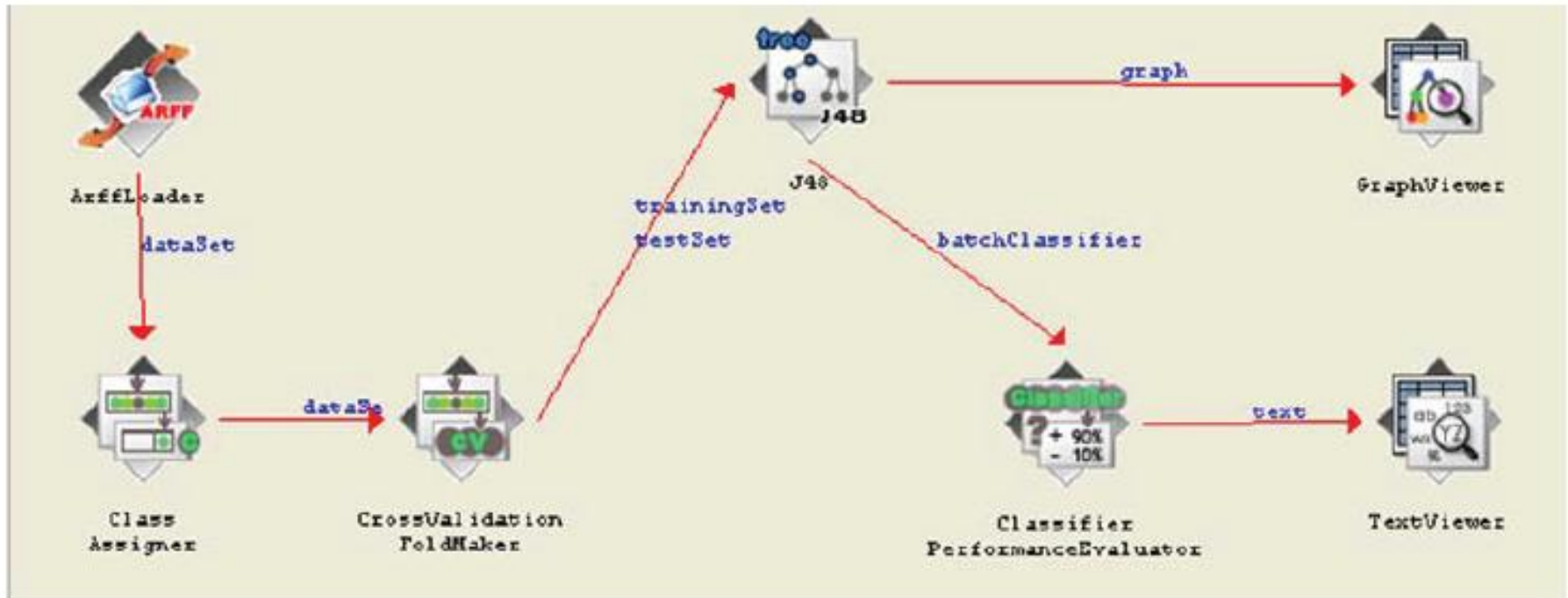
- Ejemplo
 - Introducir un nodo *Classifier* *PerformanceEvaluator* (en pestaña *evaluation*).
 - Conectarlo con *J48* mediante la conexión *batchClassifier*.

Knowledge flow

- Ejemplo
 - Para finalizar, incluimos un *GraphViewer* y un *TextViewer* conectados a *J48* y *Classifier PerformanceEvaluator* mediante conexiones *graph* y *text*, respectivamente.
 - Para ejecutar todo el experimento, seleccionar *Start Loading* en *Arff Loader*.
 - Los resultados pueden verse en cualquiera de los dos visores finales

Knowledge flow

- Ejemplo



Ejemplos Clasificadores

- **Ejemplo: Selección de Farmaco**
- En este caso se trata de predecir el tipo de fármaco (drug) que se debe administrar a un paciente afectado de **rinitis alérgica** según distintos parámetros/variables.
- Las variables que se recogen en los historiales clínicos de cada paciente son:
 - · Age: Edad
 - · Sex: Sexo
 - · BP (Blood Pressure): Tensión sanguínea.
 - · Cholesterol: nivel de colesterol.
 - · Na: Nivel de sodio en la sangre.
 - · K: Nivel de potasio en la sangre.

Ejemplos Clasificadores

- Ejemplo: Selección de Farmaco
- Hay cinco fármacos posibles: **DrugA**, **DrugB**, **DrugC**, **DrugX**, **DrugY**. Se han recogido los datos del medicamento idóneo para muchos pacientes en cuatro hospitales.
- **Objetivo:** Se pretende, para nuevos pacientes, determinar el mejor medicamento a probar.
- “drug1n.arff”

Ejemplos Clasificadores

- Ejemplo: Selección de Fármaco
- La primera pregunta que nos podemos hacer es ver qué fármacos son más comunes en general, para ver si todos suelen ser igualmente efectivos en términos generales.
- Para ello seleccionamos el atributo drug, y de esta manera vemos al distribución por clases. Podemos concluir que el fármaco más efectivo es el Y, que se administra con éxito en casi la mitad de los pacientes.
- Una regla vulgar (**REF**) sería aplicar el fármaco Y, en el caso que falle, el fármaco X, y así sucesivamente siguiendo las frecuencias de uso con éxito. →ZeroR

Ejemplos Clasificadores

- Ejemplo: Selección de Fármaco
- ZeroR → 45.5%
- Mejoramos la clase mayoritaria con J48 por ejemplo → 97%
- **Análisis J48:**
 - Mejora significativa
 - Buena aproximación
 - Bastantes reglas (12) → algo complejo
- **¿Se puede Mejorar?** → Siempre se pueden buscar otros algoritmos pero, intentaremos analizar los datos

Ejemplos Clasificadores

- **Ejemplo: Selección de Fármaco**
- Vamos a analizar, con más detenimiento, los atributos de entrada del problema. Es posible que se puedan establecer mejores modelos si combinamos algunos atributos → Podemos analizar pares de atributos utilizando diferentes gráficos.
- Para comparar la relación entre atributos en Weka debemos acudir al entorno **Visualize**, donde podemos realizar gráficas entre pares de atributos y ver si tienen alguna relación con las clases.
- De entre todas las combinaciones posibles, destaca la que utiliza los parámetros de los niveles de sodio y potasio (**K y Na**) ver

Ejemplos Clasificadores

- Ejemplo: Selección de Fármaco
- ¿Qué vemos?
- En este gráfico sí que se ven algunas características muy significativas. Parece haber una clara separación lineal entre una relación **K/Na** alta y una relación **K/Na** baja.
- Para las concentraciones **K/Na bajas**, el fármaco **Y** es el más efectivo de una manera clara y parece mostrarse que por encima de un cierto cociente K/Na ese medicamento deja de ser efectivo y se debe recurrir a los otros cuatro
- ¿Qué puedo hacer?

Ejemplos Clasificadores

- Ejemplo: Selección de Fármaco
- Podemos utilizar este conocimiento que acabamos de extraer para mejorar nuestros modelos.
- Hemos establecido que el medicamento a administrar depende en gran medida del cociente entre **K/Na**. Por tanto, vamos a realizar un nuevo modelo que utilice este cociente
- Crear un nuevo atributo derivado (también llamados atributos **pick & mix**) mediante el uso de los filtros del entorno “preprocess”.

Ejemplos Clasificadores

- Ejemplo: Selección de Fármaco
- Preprocesos → Añadimos
Filtro → `Unsupervised.Attribute.Addexpression`.
- Este atributo nos permite la creación de nuevos atributos mediante la combinación de atributos ya existentes (Pick&Mix).
- Expression=“a5/a6” (a5 corresponde a K y a6 corresponde a NA)
- Name= nuevo atributo “Na_to_Ka”.

Ejemplos Clasificadores

- Ejemplo: Selección de Fármaco
- Aplicar J48 → Resultados??
- Modelo mucho más simple y corto que el anterior, en el que se ve la importancia del atributo que hemos creado Na_to_K.
- Usar CrossValidation
- Se mejora la precisión

Ejemplos: Clasificadores Sensibles al Coste

- **Ejemplo: Préstamo Bancario**
- En un entorno bancario tenemos un modelo que nos recomienda si conceder o no un crédito a un determinado cliente a partir de las características propias de ese cliente.
- Obviamente, y desde el punto de vista del banco, es mucho más costoso que el sistema se equivoque dando un crédito a una persona que no lo devuelve, que la situación contraria, denegar un crédito a un cliente que sí lo devolvería
- **Como ajustar Clasificador??**
- credit-g.arff

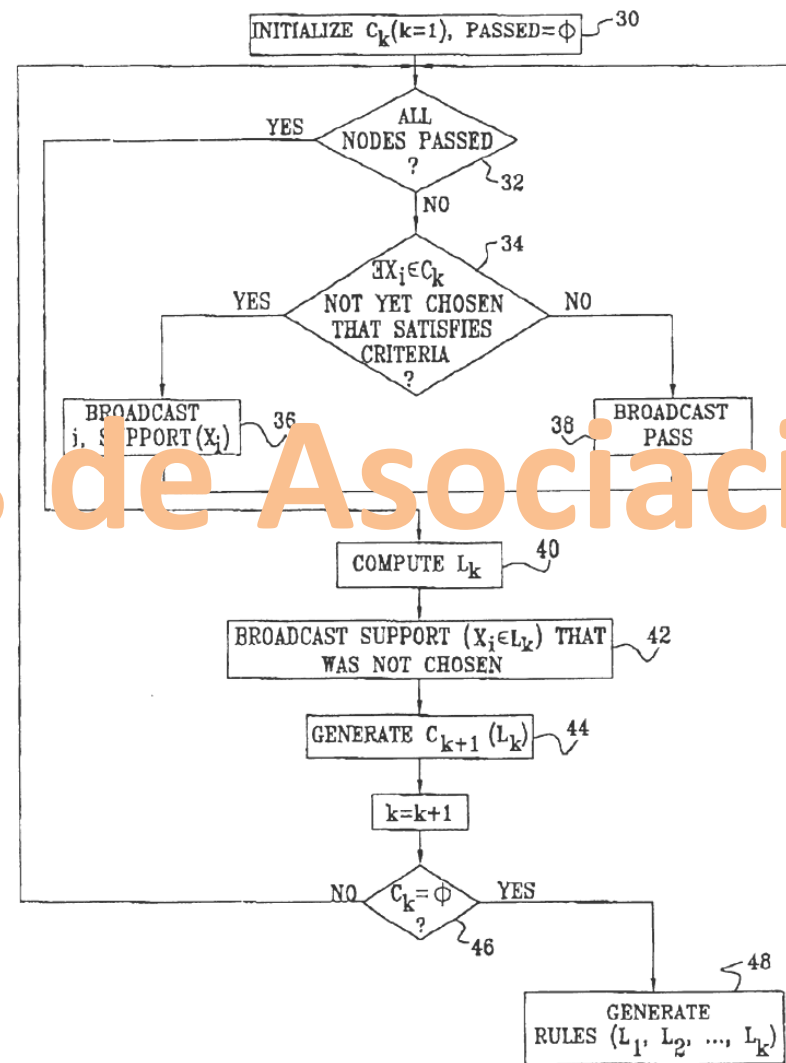
Ejemplos: Clasificadores Sensibles al Coste

- **Ejemplo: Préstamo Bancario**
- conjunto de datos contiene 1.000 ejemplos que representan clientes de una entidad bancaria que demandaron un crédito.
- Existen siete atributos numéricos y 13 nominales. Los atributos de cada objeto indican información sobre el cliente en cuestión: estado civil, edad, parado, etc., así como información del crédito: propósito del crédito, cantidad solicitada, etc.
- La clase es binaria e indica si el cliente puede ser considerado como fiable para concederle el crédito o no.
- **Pasos a seguir?**

Ejemplos: Clasificadores Sensibles al Coste

- Ejemplo: Préstamo Bancario
- Directamente ZeroR y Bayes →? Pesos?
- Meta
- Posible Mejora?

Reglas de Asociación



Reglas de Asociación

- Descubrir hechos que ocurren en común dentro de un determinado conjunto de datos.
- Descubrir combinaciones de pares atributo-valor que ocurren con frecuencia en un grupo de datos.
- De aplicación áreas como web mining, detección de intrusos o bioinformática.
- Ejemplo típico cesta de la compra.



Reglas de Asociación

Dificultad

- Gran número de posibles reglas a considerar, incluso para conjuntos de datos pequeños. Consecuentemente puede contener cualquier combinación de pares atributo-valor.
 - **Soporte:** ejemplos que satisfacen antecedente y consecuente o número de instancias que cubre correctamente la regla
 - **Confianza:** % de ejemplos que satisfacen el consecuente de los que hacen cierto el antecedente o cociente entre el soporte y el número de instancias que satisfacen el antecedente

Reglas de Asociación - Ejemplo

Cielo	Temp	Humedad	Viento	Jugar
Soleado	Alta	Alta	Falso	No
Soleado	Alta	Alta	Cierto	No
Cubierto	Alta	Alta	Falso	Si
Lluvioso	Suave	Alta	Falso	Si
Lluvioso	Fría	Normal	Falso	Si
Lluvioso	Fría	Normal	Cierto	No
Cubierto	Fría	Normal	Cierto	Si
Soleado	Suave	Alta	Falso	No
Soleado	Fría	Normal	Falso	Si
Lluvioso	Suave	Normal	Falso	Si
Soleado	Suave	Normal	Cierto	Si
Cubierto	Suave	Alta	Cierto	Si
Cubierto	Alta	Normal	Falso	Si
Lluvioso	Suave	Alta	Cierto	No

- Si temperatura=alta Entonces humedad= alta
- **Soporte:** n° de instancias con temperatura=alta & humedad= alta, 3
- **Confianza:** soporte dividido por n° de instancias con temperatura=alta, 75% (3/4)

Reglas de Asociación - Generación

1. Buscar combinaciones de pares atributo-valor con suficiente soporte
2. Generar, a partir de ellas, reglas con suficiente confianza
3. Se recurre a una estructura intermedia, denominada item-set donde
 - Item: par atributo-valor
 - Item set: conjunto de pares atributo-valor
 - K-item set: conjunto con k items

Reglas de Asociación - Algoritmos

- Algoritmos que realizan búsquedas de reglas de asociación en BBDD.
- **A priori:** Se basa en el conocimiento previo de los conjuntos frecuentes. Reduce el espacio de búsqueda aumentando la eficiencia.
- **Partition:** Basado en A priori pero más eficiente al hacer menos pasadas sobre la BBDD

Es importante reseñar que estos métodos sólo funcionan con datos nominales.

Reglas de Asociación – Ejemplo datos del hundimiento del Titanic:

“titanic.arff” y corresponden a las características de los 2.201 pasajeros del Titanic. Estos datos son reales.

Para este ejemplo sólo se van a considerar cuatro variables:

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

Reglas de Asociación – Ejemplo datos del hundimiento del Titanic:

“**titanic.arff**” y corresponden a las características de los 2.201 pasajeros del Titanic. Estos datos son reales.

“**UpperBoundMinSupport**” límite superior de cobertura requerido para aceptar un conjunto de ítems. Si no se encuentran conjuntos de ítems suficientes para generar las reglas requeridas se va disminuyendo el límite hasta llegar al límite inferior (opción “**LowerBoundMinSupport**”)

“**minMetric**” indicamos la confianza mínima (u otras métricas dependiendo del criterio de ordenación) para mostrar una regla de asociación

“**numRules**” indicamos el número de reglas que deseamos que aparezcan en pantalla.

La ordenación de estas reglas en pantalla puede configurarse mediante la opción “**MetricType**”, algunas opciones que se pueden utilizar son: confianza de la regla, lift (confianza dividido por el número de ejemplos cubiertos por la parte derecha de la regla),

Reglas de Asociación – Ejemplo datos del hundimiento del Titanic:

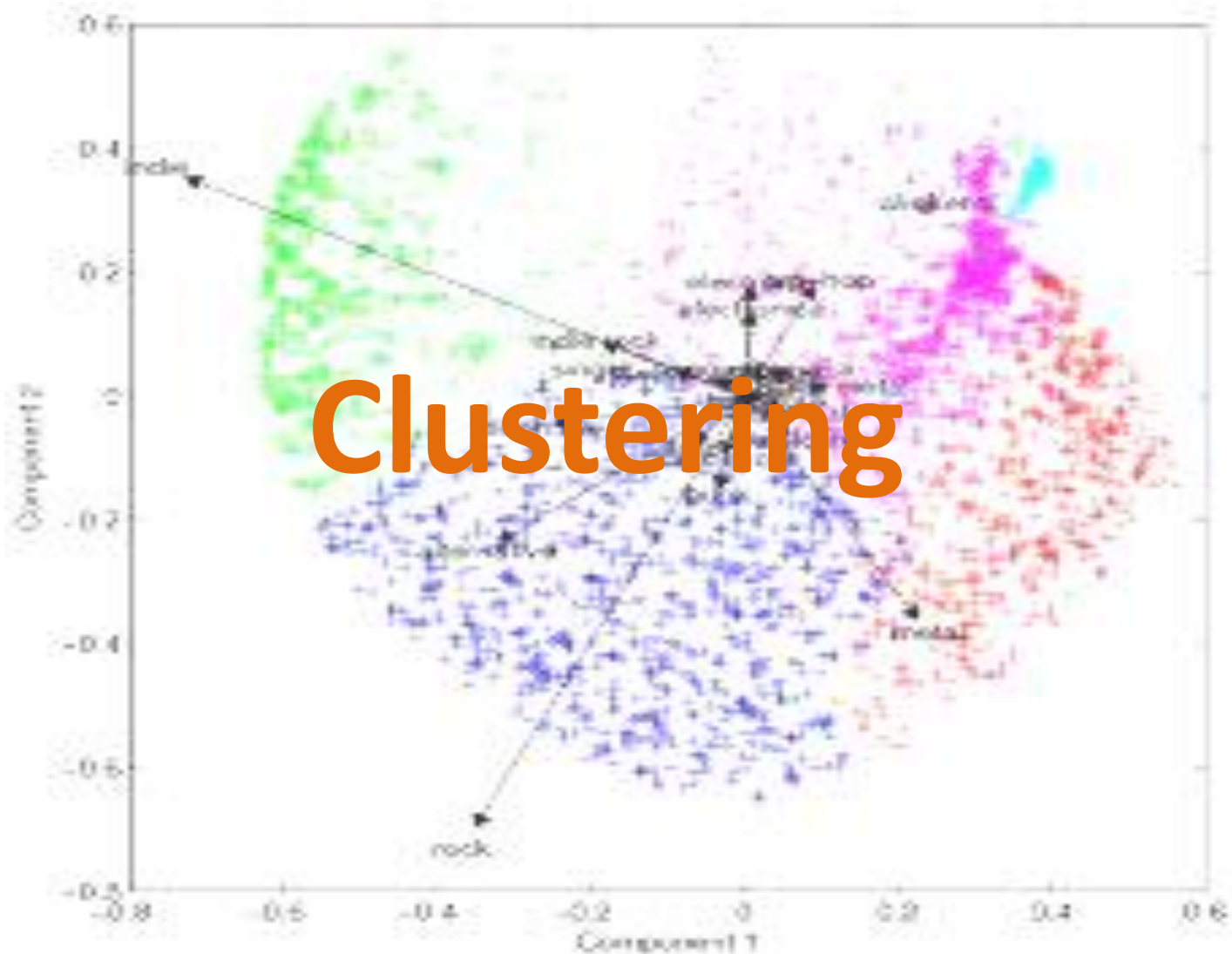
1. Clase=0 885 ==> Edad=1 885 conf:(1)
2. Clase=0 Sexo=1 862 ==> Edad=1 862 conf:(1)
3. Sexo=1 Sobrevivió?=0 1364 ==> Edad=1 1329 conf:(0.97)
4. Clase=0 885 ==> Edad=1 Sexo=1 862 conf:(0.97)
5. Clase=0 Edad=1 885 ==> Sexo=1 862 conf:(0.97)
6. Clase=0 885 ==> Sexo=1 862 conf:(0.97)
7. Sobrevivió?=0 1490 ==> Edad=1 1438 conf:(0.97)
8. Sexo=1 1731 ==> Edad=1 1667 conf:(0.96)
9. Edad=1 Sobrevivió?=0 1438 ==> Sexo=1 1329 conf:(0.92)
10. Sobrevivió?=0 1490 ==> Sexo=1 1364 conf:(0.92)

Reglas de Asociación – Ejemplo datos del hundimiento del Titanic:

- 1 indica que, como era de esperar toda la tripulación es adulta.
- La regla 2 nos indica lo mismo, pero teniendo en cuenta a los varones.
- Parecidas conclusiones podemos sacar de las reglas 4, 5 y 6.
- La regla 3 nos indica que los varones que murieron fueron en su mayoría adultos (97%).
- La regla 7 destaca que la mayoría que murieron fueron adultos (97%).
- Y finalmente la 10 informa que la mayoría de los muertos fueron hombres (92%).

Reglas de Asociación – Ejemplo datos Votos EEUU y Supermercado

- Usar votos.arff
- Usar supermarket.arff



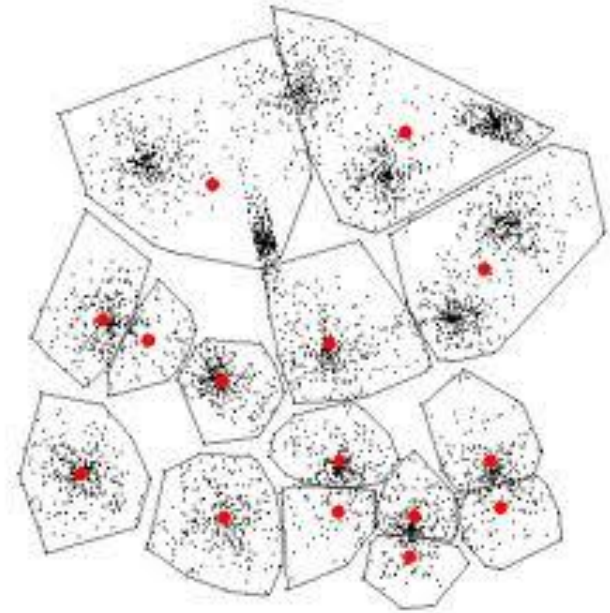
Clustering

Un algoritmo de agrupamiento es un procedimiento de agrupación de una serie de **vectores** de acuerdo con un **criterio de cercanía**.

Esta cercanía se define en términos de una determinada **función de distancia**, como la euclídea, aunque existen otras más robustas o que permiten extenderla a variables discretas.

Generalmente, los vectores de un mismo grupo (o clústers) comparten propiedades comunes.

El conocimiento de los grupos puede permitir una descripción sintética de un conjunto de datos multidimensional complejo.



Clustering

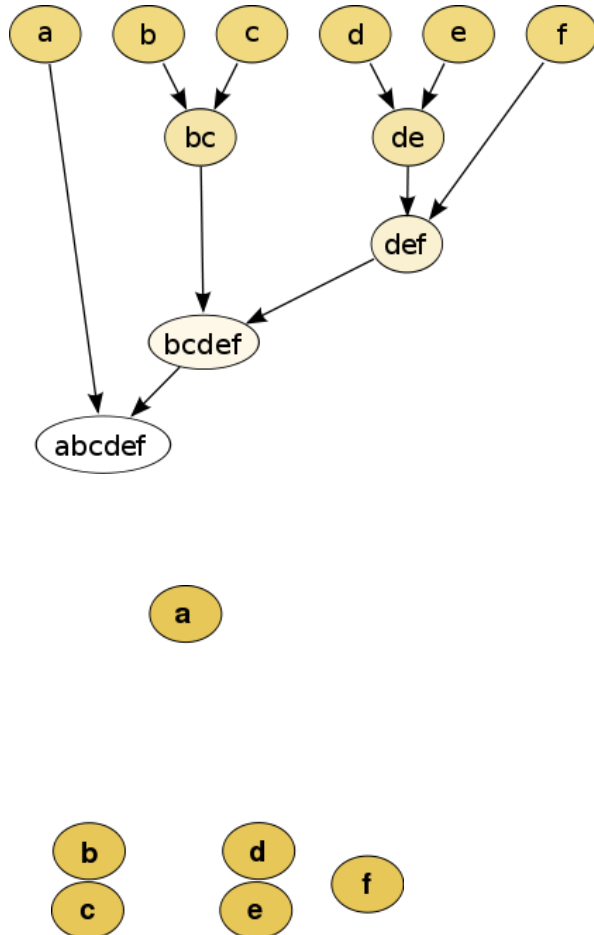
Esta descripción sintética se consigue sustituyendo la descripción de todos los elementos de un grupo por la de un representante característico del mismo → De ahí su uso en minería de datos.

Se considera una técnica de aprendizaje **no supervisada** puesto que busca encontrar relaciones entre variables descriptivas pero no la que guardan con respecto a una variable objetivo.

Las técnicas de agrupamiento se dividen en **dos grandes categorías**:

- **Jerárquicas**, que construyen una jerarquía de grupos escindiéndolos iterativamente.
- De **particionamiento**, en los que el número de grupos se determina de antemano y las observaciones se van asignando a los grupos en función de su cercanía.

Clustering



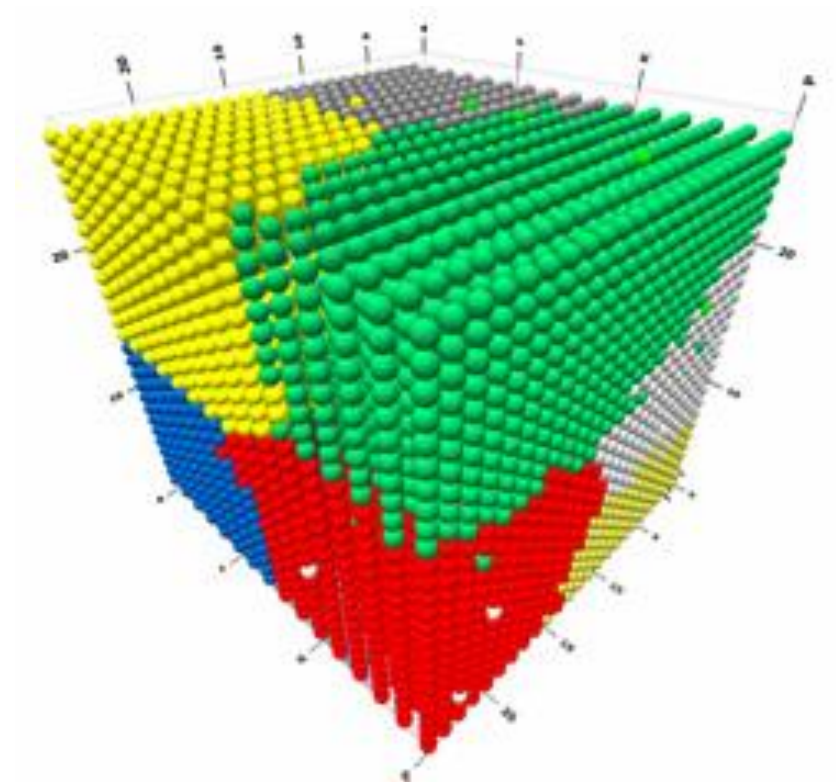
Clustering Jerárquico:

- **Aglomerativo:** Se trata de una aproximación “bottom up” o de abajo arriba : cada elemento pertenece a su propio cluster y según avanza el algoritmo, va juntando elementos en nuevos clusters
- **Divisivo:** Es una aproximación “top down”, es decir, de arriba abajo en la que todas las observaciones comienzan perteneciendo a un único cluster y se van separando.

Clustering

Clustering por particionamiento:

- K-medias
- C-medias difuso
- QT (Quality Threshold)



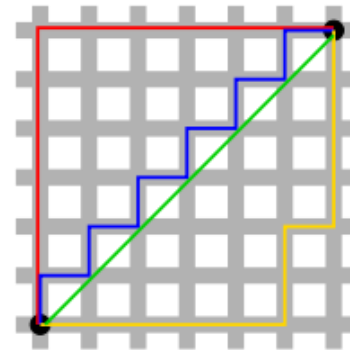
Funciones de distancia habituales:

- Distancia Euclídea:

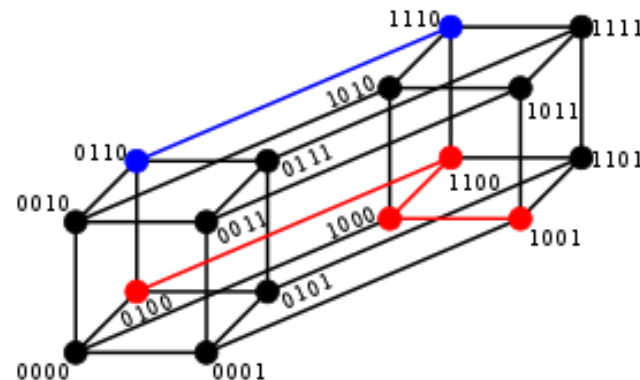
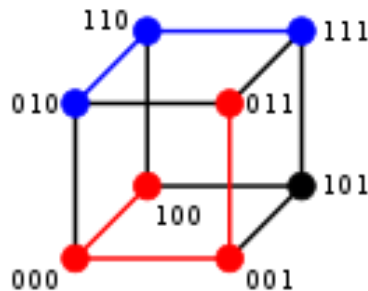
$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.$$

- Distancia Manhattan:

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^n |p_i - q_i|,$$



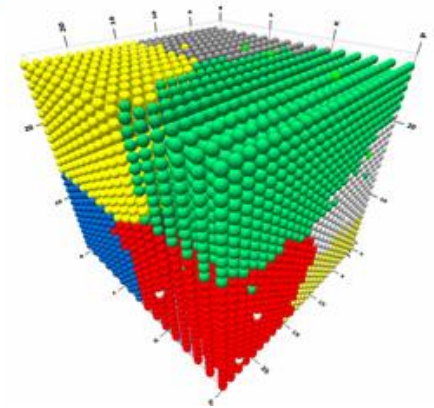
- Distancia Hamming: Mide el mínimo número de sustituciones necesarias para pasar de un elemento a otro



Ejemplo de Clustering: Provincias

Datos en `Caixa_Provincias.arff`

- 52 provincias españolas
- Características sociodemográficas.
- Datos Reales
- Fuente: Anuario económico de La Caixa del año 2006
- Fuente: Página web del Ministerio de Vivienda



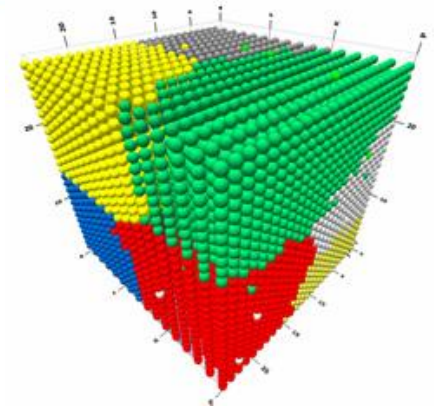
Ejemplo de Clustering: Provincias

Preprocesado:

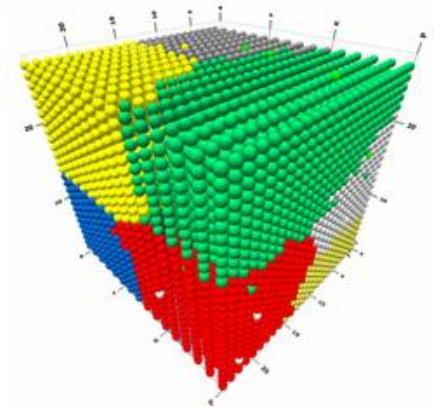
- Nombre de la provincia
- Población
- Ratio varones/mujeres
- Ratio extranjeros/españoles
- Extensión de la provincia (en Km2)
- Paro
- Número de teléfonos fijos registrados
- Número de vehículos de motor matriculados
- Número de oficinas bancarias
- Precio medio del m2 de vivienda

Selección de Algoritmo de Clustering → SimpleKMeans

SimpleKMeans:



Ejemplo de Clustering: Provincias



Kmedias → SimpleKMeans = Aglomeramiento →

- Es necesario definir número de cluster objetivo
- Pero... podemos elegir el algoritmo?????
- Que debo hacer????

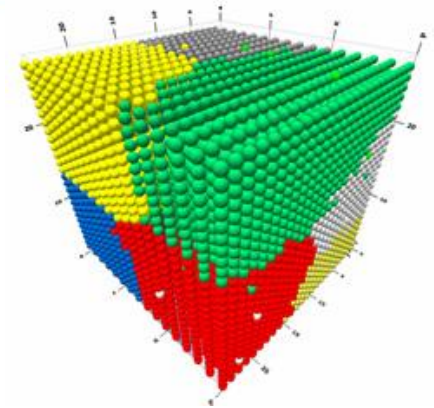
Ejemplo de Clustering: Provincias

Atributo String → Atributo Nominal

Ahora sí podemos aplicar Kmeans

Distancia Euclidea

3 Clusters



Antes de ejecutar nuestro primer clustering tenemos que seleccionar los **atributos** que **NO** queremos usar en el proceso de entre los que contiene el fichero ARFF inicial de datos. Por ejemplo, no tiene sentido utilizar el nombre de la provincia ya que no aporta ninguna información útil para la separación en clusters de las provincias.

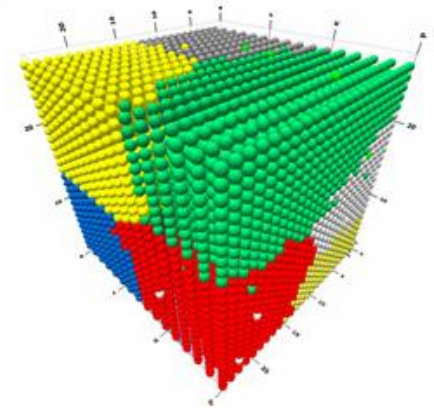
En el ejemplo vamos a quedarnos con las siguientes columnas de datos: **población, ratio de extranjeros/españoles, paro y precio medio del m2 de vivienda**. Para ello pinchamos en Ignore attributes y seleccionamos los demás atributos.

Ejemplo de Clustering: Provincias

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

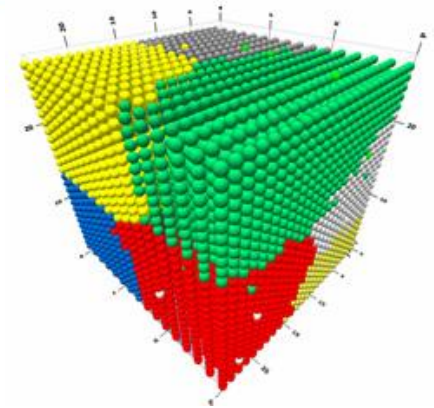
```
Clustered Instances
```

```
0      25 ( 48%)
1      21 ( 40%)
2       6 ( 12%)
```



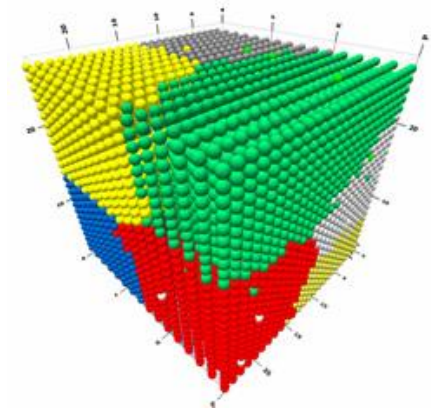
- 3 clusters con 25, 21 y 6 provincias respectivamente.
- Podemos ver la distribución de una manera más gráfica pinchando con el botón derecho sobre la entrada correspondiente del listado de la derecha y después en *Visualize cluster assignments*.
- Interpretar los resultados!!!

Ejemplo de Clustering: Provincias



- Guardar Modelo desde Visualización → .arff
- Editar y ver resultado

Ejemplo de Clustering: Empleados



La empresa de software para Internet “**Memolum Web**” quiere extraer **tipologías** de empleados, con el objetivo de hacer una política de personal más fundamentada y seleccionar a qué grupos **incentivar**.

Ejemplo de Clustering: Empleados

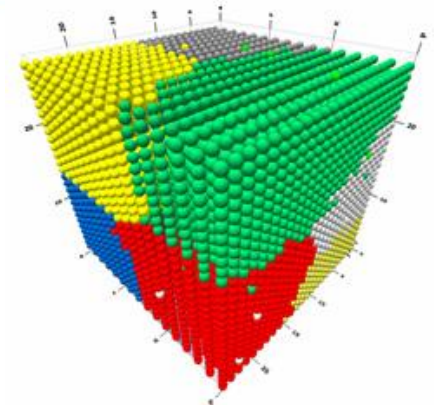
15 empleados

Las variables son:

- **Sueldo:** sueldo anual en euros.
- **Casado:** si está casado o no.
- **Coche:** si viene en coche a trabajar (o al menos si lo aparca en el parking de la empresa).
- **Hijos:** si tiene hijos.
- **Alq/Prop:** si vive en una casa alquilada o propia.
- **Sindic.:** si pertenece al sindicato revolucionario de Internet
- **Bajas/Año:** media del nº de bajas por año
- **Antigüedad:** antigüedad en la empresa
- **Sexo:** H: hombre, M: mujer.

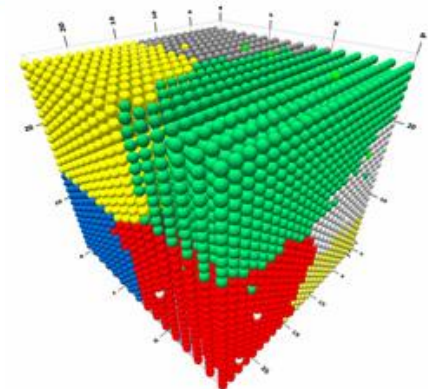
fichero “empleados.arff”.

Objetivo: Se intenta extraer grupos de entre estos quince empleados.



Ejemplo de Clustering: Empleados

- Probar con tres grupos
- Kmedias
- Simple? Resultado?



cluster 1	cluster 2	cluster 3
6 Ejemplos	5 examples	4 examples
Sueldo : 29166.6667 Casado : No Coche : No Hijos : 0.1667 Alq/Prop : Alquiler Sindic. : Sí Bajas/Año : 6.16 Antigüedad : 8.333 Sexo : M	Sueldo : 16600 Casado : Sí Coche : Sí Hijos : 1.8 Alq/Prop : Prop Sindic. : No Bajas/Año : 3.4 Antigüedad : 8.4 Sexo : H	Sueldo : 14500 Casado : Sí Coche : Sí Hijos : 0.25 Alq/Prop : Alquiler Sindic. : No Bajas/Año : 6.25 Antigüedad : 7.75 Sexo : H