# TEMA 3: Técnicas de minería de datos en WEKA

## 3.1. Introducción

En este capítulo se introduce la herramienta de minería de datos WEKA. WEKA, acrónimo de *Waikato Environment for Knowledge Analysis*, es un entorno para experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático, sobre cualquier conjunto de datos del usuario. Para ello únicamente se requiere que los datos a analizar se almacenen con un cierto formato, conocido como *ARFF* (*Attribute-Relation File Format*).

WEKA se distribuye como software de libre distribución desarrollado en Java. Está constituido por una serie de paquetes de código abierto con diferentes técnicas de preprocesado, clasificación, agrupamiento, asociación, y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados. Estos paquetes pueden ser integrados en cualquier proyecto de análisis de datos, e incluso pueden extenderse con contribuciones de los usuarios que desarrollen nuevos algoritmos. Con objeto de facilitar su uso por un mayor número de usuarios, WEKA incluye además una interfaz gráfica de usuario para acceder y configurar las diferentes herramientas integradas.

A lo largo de este capítulo se desarrollan varios ejemplos. En ellos se utilizan datos correspondientes a alumnos que realizaron las pruebas de acceso a la universidad en los años 1993 a 2003, procedentes de diferentes centros de enseñanza secundaria de la comunidad de Madrid.

## 3.2. Preparación de los datos

Los datos de entrada a la herramienta, sobre los que operarán las técnicas implementadas, deben estar codificados en un formato específico, denominado *Attribute-Relation File Format* (extensión "arff"). La herramienta permite cargar los datos en tres soportes: desde un fichero de texto, desde una base de datos y mediante acceso a través de internet a la dirección URL de un servidor web. En nuestros casos trabajaremos con ficheros de texto. Los datos deben estar dispuestos en el fichero de la forma siguiente: cada instancia en una fila, y con los atributos separados por comas.

El formato de un fichero arff sigue la siguiente estructura:

```
% comentarios
@relation NOMBRE_RELACION
@attribute r1 real
@attribute r2 real ...
...
@attribute i1 integer
@attribute i2 integer
…
@attribute s1 {v1_s1, v2_s1,…vn_s1}
@attribute s2 {v1_s1, v2_s1,…vn_s1}
…
@data
DATOS
```

Por tanto, los atributos pueden ser principalmente de dos tipos: numéricos, tanto de tipo real como entero (indicado con las palabra *real* o *integer* tras el nombre del atributo), y simbólicos, en cuyo caso se indican entre llaves los valores posibles que puede tomar.

### 3.2.1. Muestra de datos

El fichero que vamos a utilizar contiene datos correspondientes a 18802 alumnos presentados a las pruebas de acceso a la universidad y los resultados obtenidos en las pruebas. Los datos para cada alumno contienen la siguiente información: año, convocatoria, localidad del centro, opción cursada (de entre 5 posibles), calificaciones parciales obtenidas en lengua, historia, idioma y las tres asignaturas opcionales, así como los nombres de las asignaturas de idioma y las tres opcionales cursadas, calificación en el bachillerato, calificación final, y si el alumno se presentó o no a la prueba. Puede comprobarse que la cabecera del fichero de datos, "selectividad.arff", sigue el formato mencionado anteriormente:

```
@relation selectividad
@attribute Año_académico real
@attribute convocatoria {J, S}
@attribute localidad {ALPEDRETE, ARANJUEZ, ... }
@attribute opcion1ª {1,2,3,4,5}
@attribute nota_Lengua real
@attribute nota_Historia real
@attribute nota_Idioma real
@attribute des_Idioma {INGLES, FRANCES, ALEMAN}
@attribute des_asig1 {BIOLOGIA, DIB.ARTISTICO_II,... }
@attribute calif_asig1 real
@attribute des_asig2 {BIOLOGIA, C.TIERRA, ...}
@attribute calif_asig2 real
@attribute des_asig3 {BIOLOGIA, C.TIERRA, ...}
@attribute calif_asig3 real
@attribute cal_prueba real
@attribute nota_bachi real
@attribute cal_final real
@attribute Presentado {SI, NO}
@data
...
```

### 3.2.1. Objetivos del análisis

Antes de comenzar con la aplicación de las técnicas de WEKA a los datos, es muy conveniente revisar los objetivos perseguidos en el análisis. Un paso previo a la búsqueda de relaciones y modelos subyacentes en los datos ha de ser la comprensión del dominio de aplicación y establecer una idea clara acerca de los objetivos del usuario final. De esta manera, el proceso de análisis de datos (proceso *KDD*), permitirá dirigir la búsqueda y hacer refinamientos, con una interpretación adecuada de los resultados generados. Los objetivos, utilidad, aplicaciones, etc., del análisis efectuado no "emergen" de los datos, sino que deben ser considerados con detenimiento como primer paso del estudio.

En nuestro caso, uno de los objetivos perseguidos podría ser el intentar relacionar los resultados obtenidos en las pruebas con características o perfiles de los alumnos, si bien la descripción disponible no es muy rica y habrá que atenerse a lo que está disponible. Algunas de las preguntas que podemos plantearnos como objetivos del análisis podrían ser las siguientes:

- ¿Qué características comunes tienen los alumnos que superan la prueba?
- ¿existen grupos de alumnos, no conocidos de antemano, con características similares?
- ¿hay diferencias significativas en los resultados obtenidos según las opciones, localidades, años, etc.?
- ¿la opción seleccionada y el resultado están influidos por el entorno?

Como veremos, muchas veces el resultado alcanzado puede ser encontrar relaciones triviales o conocidas previamente, o puede ocurrir que el hecho de no encontrar relaciones significativas sea muy relevante. Por ejemplo, saber después de un análisis exhaustivo que la opción o la localidad no condicionan significativamente la calificación, o que la prueba es homogénea a lo largo de los años, puede ser una conclusión valiosa, y en este caso "tranquilizadora".

Se recomienda instalar WEKA antes de continuar. La comprensión del tema será mucho mayor si se van realizando todos los pasos que se detallan a continuación.

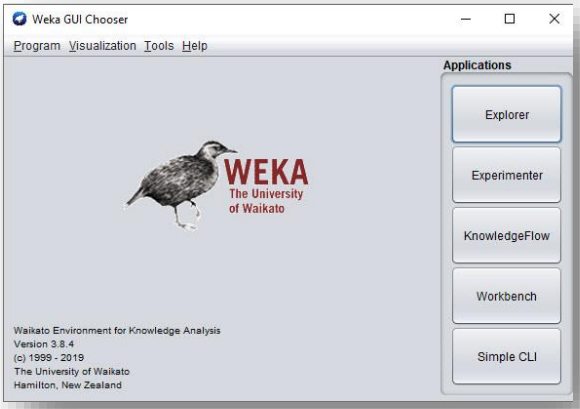
Puedes descargar WEKA gratuitamente en la dirección

http://www.cs.waikato.ac.nz/ml/weka/
https://www.cs.waikato.ac.nz/ml/weka/downloading.html

## 3.3. Ejecución de WEKA

WEKA se distribuye como un fichero ejecutable comprimido de java (fichero "jar"), que se invoca directamente sobre la máquina virtual JVM.

Una vez invocada, aparece la ventana de entrada a la interfaz gráfica (*GUIChooser*),




Weka dispone de cuatro opciones posibles de trabajo:

- **Simple CLI**: la interfaz "Command-Line Interfaz" es simplemente una ventana de comandos java para ejecutar las clases de WEKA. La primera distribución de WEKA no disponía de interfaz gráfica y las clases de sus paquetes se podían ejecutar desde la línea de comandos pasando los argumentos adecuados.
- **Explorer**: es la opción que permite llevar a cabo la ejecución de los algoritmos de análisis implementados sobre los ficheros de entrada, una ejecución independiente por cada prueba.
- **Experimenter**: esta opción permite definir experimentos más complejos, con objeto de ejecutar uno o varios algoritmos sobre uno o varios conjuntos de datos de entrada, y comparar estadísticamente los resultados.
- **KnowledgeFlow**: esta opción permite llevar a cabo las mismas acciones del "Explorer", con una configuración totalmente gráfica, inspirada en herramientas de tipo "data-flow" para seleccionar componentes y conectarlos en un proyecto de minería de datos, desde que se cargan los datos, y se aplican algoritmos de tratamiento y análisis, hasta el tipo de evaluación deseada.

Nos centraremos únicamente en la *Explorer*. Una vez seleccionada, aparece una ventana con 6 pestañas en la parte superior que se corresponden con diferentes tipos de operaciones, en etapas independientes, que se pueden realizar sobre los datos:

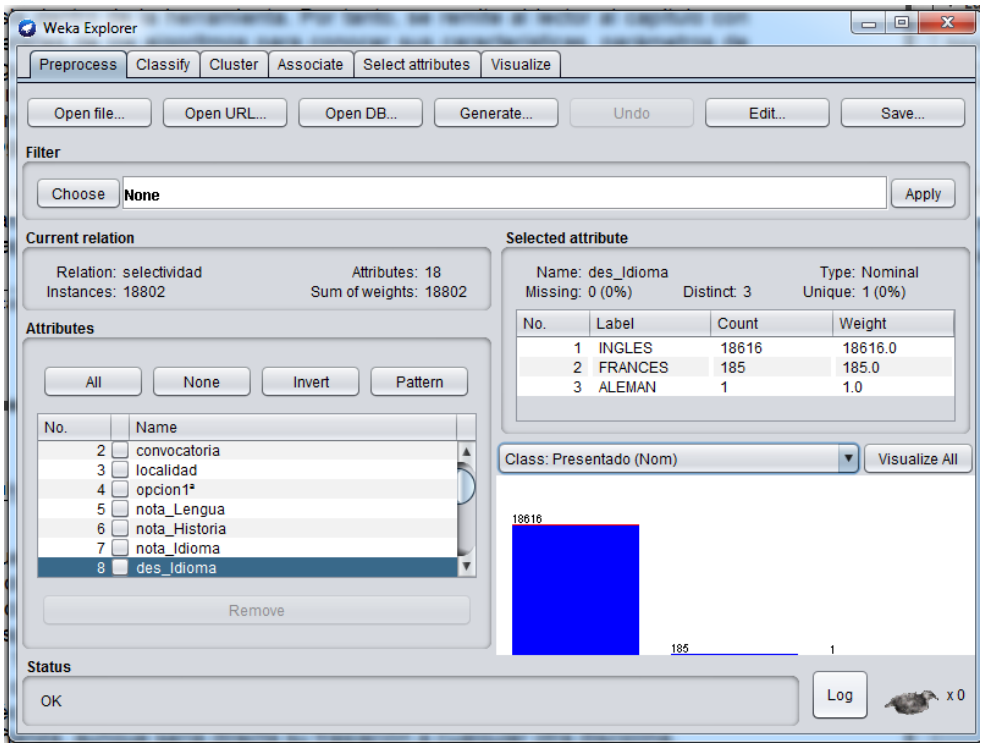
- **Preprocess**: selección de la fuente de datos y su preparación (filtrado).
- **Classify**: Herramientas para aplicar esquemas de clasificación, entrenar modelos y evaluar su precisión.
- **Cluster**: Algoritmos de agrupamiento.
- **Associate**: Algoritmos de búsqueda de reglas de asociación.
- **Select Attributes**: Búsqueda supervisada de subconjuntos de atributos representativos.
- **Visualize**: Herramienta interactiva de presentación gráfica en 2D.

Además de estas pestañas de selección, en la parte inferior de la ventana aparecen dos elementos comunes. Uno es el botón "**Log**", que al activarlo presenta una ventana textual donde se indica la secuencia de todas las operaciones que se han llevado a cabo

dentro del Explorer, sus tiempos de inicio y fin, así como los mensajes de error más frecuentes. Junto al botón de log aparece un icono de actividad (el pájaro WEKA, que se mueve cuando se está realizando alguna tarea) y un indicador de status, que indica qué tarea se está realizando en este momento dentro del Explorer.

## 3.4. Preprocesado de los datos

Antes de seleccionar ninguna otra opción, debe pasarse por esta operación, ya que se precisan datos para poder llevar a cabo cualquier análisis. La disposición de la parte de preprocesado del *Explorer*, **Preprocess**, es la que se indica a continuación.




Como se indicó anteriormente, hay tres posibilidades para obtener los datos: un fichero de texto, una dirección URL, o una base de datos, dadas respectivamente por las opciones: **Open file**, **Open URL** y **Open DB**. En nuestro caso utilizaremos siempre los datos almacenados en un fichero, que en pequeños tamaños es rápido y cómodo de utilizar.

Para este ejemplo abrimos el fichero "selectividad.arff" con la opción **Open File**.

### 3.4.1. Características de los atributos

Una vez cargados los datos, aparece un cuadro resumen, *Current relation*, con el nombre de la relación que se indica en el fichero (en la línea @relation del fichero arff), el número de instancias y el número de atributos. Más abajo, aparece una lista con todos

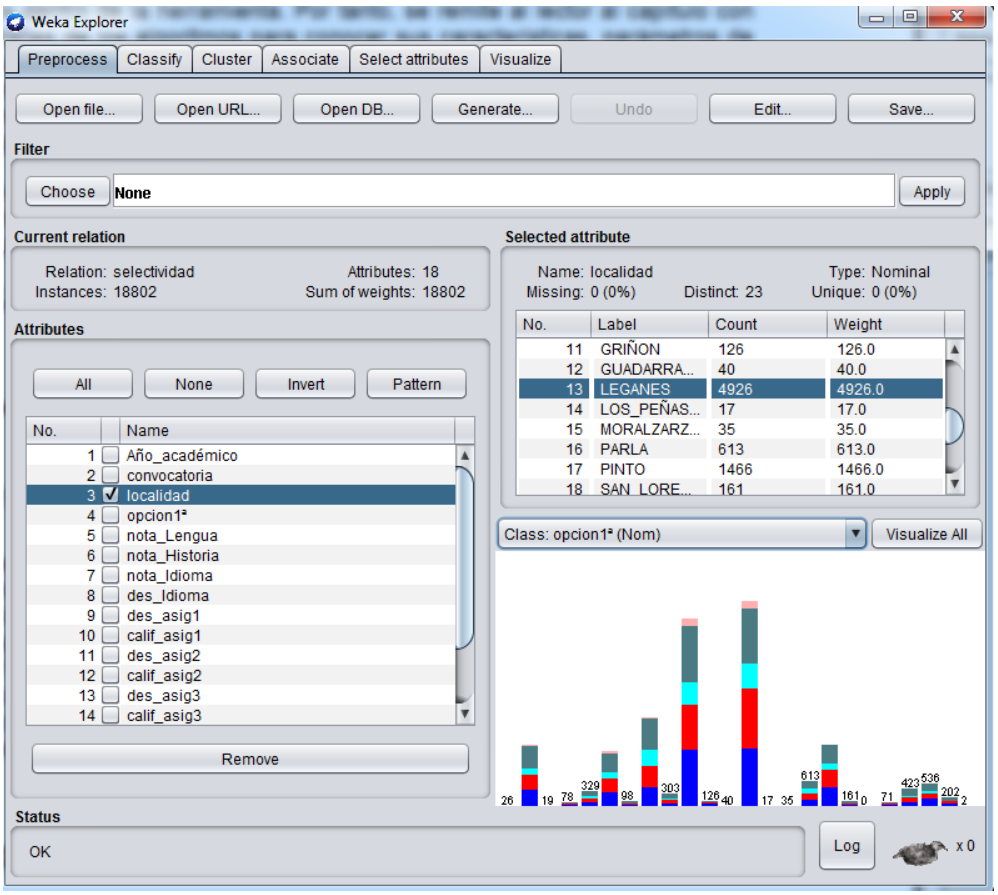
los atributos disponibles, con los nombres especificados en el fichero, de modo que se pueden seleccionar para ver sus detalles y propiedades.

En la parte derecha aparecen las propiedades del atributo seleccionado. Si es un atributo simbólico, se presenta la distribución de valores de ese atributo (número de instancias que tienen cada uno de los valores). Si es numérico aparece los valores máximo, mínimo, valor medio y desviación estándar. Otras características que se destacan del atributo seleccionado son el tipo (*Type*), el número de valores distintos (*Distinct*), el número y porcentaje de instancias con valor desconocido para el atributo (*Missing*, codificado en el fichero arff con "?"), y los valores de atributo que solamente se dan en una instancia (*Unique*).

Además, en la parte inferior se representan gráficamente los valores que toma el atributo. Si es simbólico, la distribución de frecuencias de los valores, si es numérico, un histograma con intervalos uniformes. En el histograma se puede presentar además con colores distintos la distribución de un segundo atributo para cada valor del atributo visualizado.

Por último, hay un botón "*Visualize All*" que permite representar los histogramas de todos los atributos simultáneamente.

A modo de ejemplo, a continuación se muestra el gráfico por localidades, indicando con colores las distribuciones por opciones elegidas.



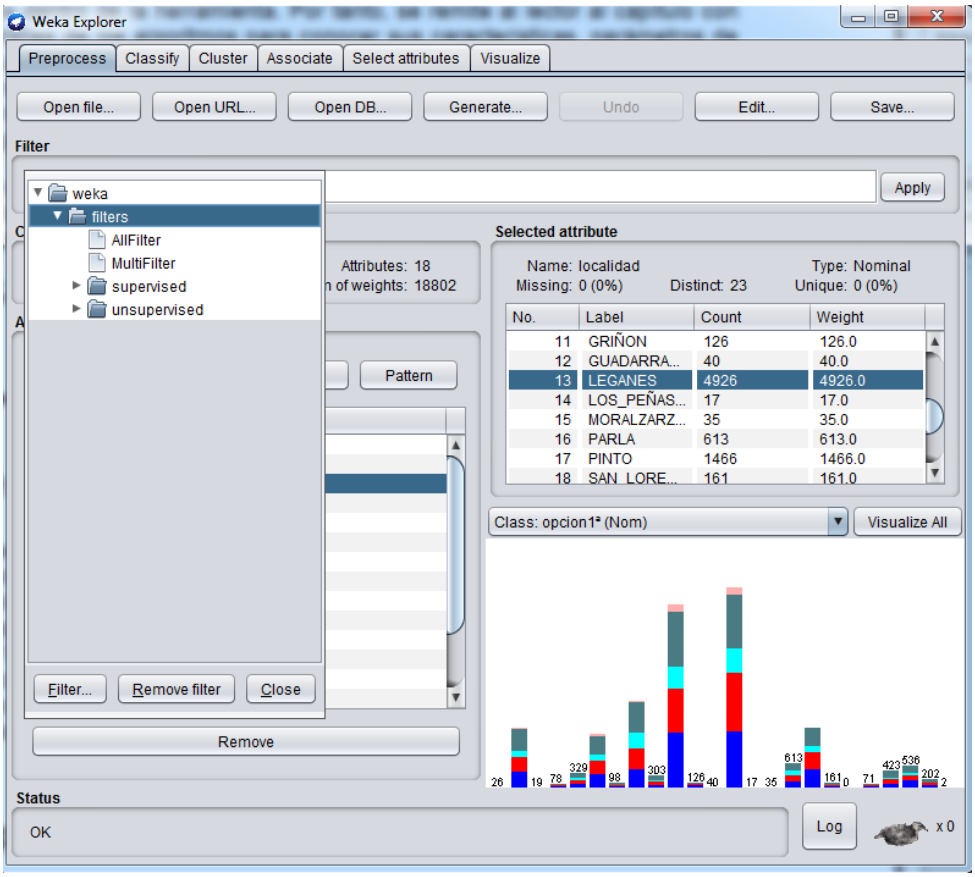
La columna de la localidad de Leganés es la que tiene más instancias. Puede observarse que la proporción de las opciones científicas (1 y 2) es superior a otras localidades, como Getafe, la segunda localidad en número de alumnos presentados.

**Actividad 3.1.** Realiza los histogramas de las calificaciones de bachillerato y calificación final de la prueba, indicando como segundo atributo la convocatoria en la que se presentan los alumnos.

### 3.4.2. Trabajo con Filtros. Preparación de ficheros de muestra

WEKA tiene integrados filtros que permiten realizar manipulaciones sobre los datos en dos niveles: atributos e instancias. Las operaciones de filtrado pueden aplicarse "en cascada", de manera que cada filtro toma como entrada el conjunto de datos resultante de haber aplicado un filtro anterior. Una vez que se ha aplicado un filtro, la relación cambia para el resto de operaciones llevadas a cabo en el *Experimenter*, existiendo siempre la opción de deshacer la última operación de filtrado aplicada utilizando el botón **Undo**. Además, pueden guardarse los resultados de aplicar filtros en nuevos ficheros, que también serán de tipo ARFF, para manipulaciones posteriores.

Para aplicar un filtro a los datos, se selecciona con el botón **Choose** de **Filter**, desplegándose el árbol con todos los que están integrados.



Puede verse que aparecen filtros de tipo supervisado y no supervisado. Hay disponibles una gran cantidad de filtros, de los cuales destacaremos únicamente algunos de las más frecuentes y que pertenecen a los filtros no supervisados.

### 3.4.2.1. Filtros de atributos

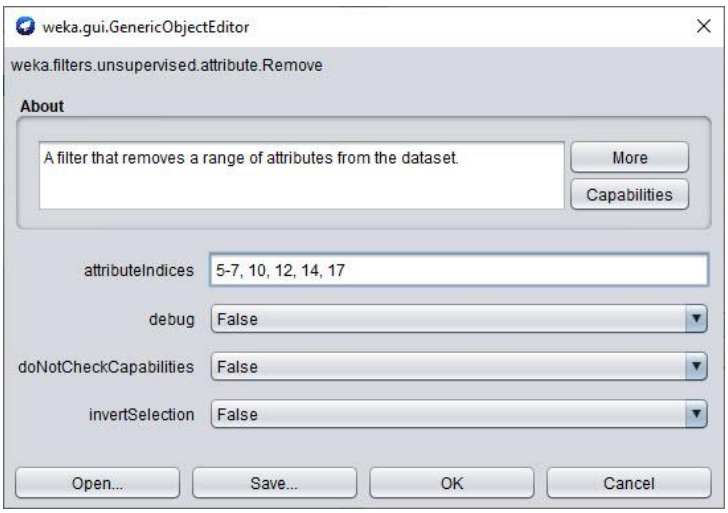
De entre todas las posibilidades implementadas tan solo se describe la utilización de filtros para eliminar atributos, para discretizar atributos numéricos, y para añadir nuevos atributos con expresiones, por la frecuencia con la que se realizan estas operaciones.

**Filtros de selección**

Vamos a utilizar el filtro de atributos "*Remove*", que permite eliminar una serie de atributos del conjunto de entrada. En primer lugar procedemos a seleccionarlo desde el árbol desplegado con el botón **Choose** de los filtros. A continuación lo configuraremos para determinar qué atributos queremos filtrar.

La configuración de un filtro sigue el esquema general de configuración de cualquier algoritmo integrado en WEKA. Una vez seleccionado el filtro específico con el botón **Choose**, aparece su nombre dentro del área de filtro (el lugar donde antes aparecía la palabra **None**). Se pueden configurar sus parámetros haciendo clic sobre esta área, momento en el que aparece la ventana de configuración correspondiente a ese filtro particular. Si no se realiza esta operación se utilizarían los valores predeterminados del filtro seleccionado.

Como primer filtro de selección, vamos a eliminar de los atributos de entrada todas las calificaciones parciales de la prueba y la calificación final, quedando como únicas calificaciones la nota de bachillerato y la calificación de la prueba. Por tanto tenemos que seleccionar los índices 5,6,7,10,12,14 y 17, indicándolo en el cuadro de configuración del filtro *Remove*:



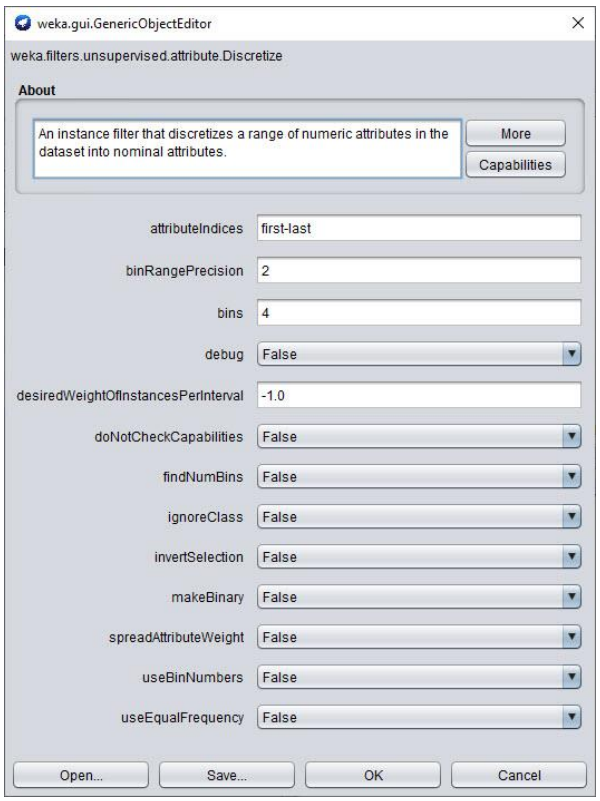

Como puede verse, en el conjunto de atributos a eliminar se pueden poner series de valores contiguos delimitados por guión (5-7) o valores sueltos, separados por comas (10, 12, 14, 17). Además, puede usarse "first" y "last" para indicar el primer y último atributo, respectivamente. La opción **invertSelection** es útil cuando realmente queremos

seleccionar un pequeño subconjunto de todos los atributos y eliminar el resto. **Open** y **Save** nos permiten guardar configuraciones de interés en archivos. El botón **More**, que aparece opcionalmente en algunos elementos de WEKA, muestra información de utilidad acerca de la configuración de los mismos. Estas opciones para designar y seleccionar atributos, para obtener ayuda, y para guardar y cargar configuraciones específicas, son comunes a otros elementos de WEKA.

Una vez configurado, al accionar el botón **Apply** del área de filtros se modifica el conjunto de datos (se filtra) y se genera una relación transformada. Esto se refleja en "Current Relation", que pasa a ser la resultante de aplicar la operación correspondiente (esta información se puede ver con más nitidez en la ventana de log, que además indicará la cascada de filtros aplicados a la relación operativa). La relación transformada tras aplicar el filtro podría almacenarse en un nuevo fichero ARFF con el botón **Save**, dentro de la ventana **Preprocess.**

### Filtros de discretización

Estos filtros son muy útiles cuando se trabaja con atributos numéricos, puesto que muchas herramientas de análisis requieren datos simbólicos, y por tanto se necesita aplicar esta transformación antes. También son necesarios cuando queremos hacer una clasificación sobre un atributo numérico, por ejemplo clasificar los alumnos aprobados y suspensos. Este filtrado transforma los atributos numéricos seleccionados en atributos simbólicos, con una serie de etiquetas resultantes de dividir la amplitud total del atributo en intervalos, con diferentes opciones para seleccionar los límites. Por defecto, se divide la amplitud del intervalo en tantas "cajas" como se indique en **bins** (por defecto 10), todas ellas de la misma amplitud.



Por ejemplo, para discretizar las calificaciones numéricas en 4 categorías, todas de la misma amplitud, se configuraría tal y como se muestra en la imagen anterior.

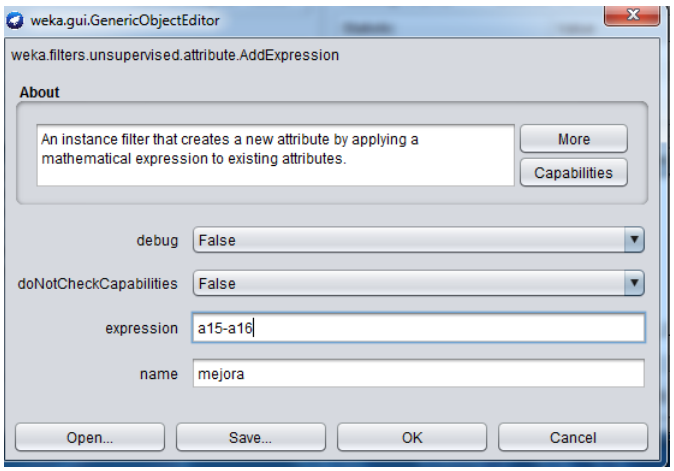
En este caso se ha aplicado a todos los atributos numéricos con la misma configuración (los atributos seleccionados son first-last, no considerando los atributos que antes del filtrado no eran numéricos). Observa que la relación de trabajo ahora ("current relation") es el resultado de aplicar en secuencia el filtro anterior y el actual.

A veces es más útil no fijar todas las cajas de la misma anchura sino forzar a una distribución uniforme de instancias por categoría, con la opción **useEqualFrequency**. La opción **findNumBins** permite optimizar el número de cajas (de la misma amplitud), con un criterio de clasificación de mínimo error en función de las etiquetas.

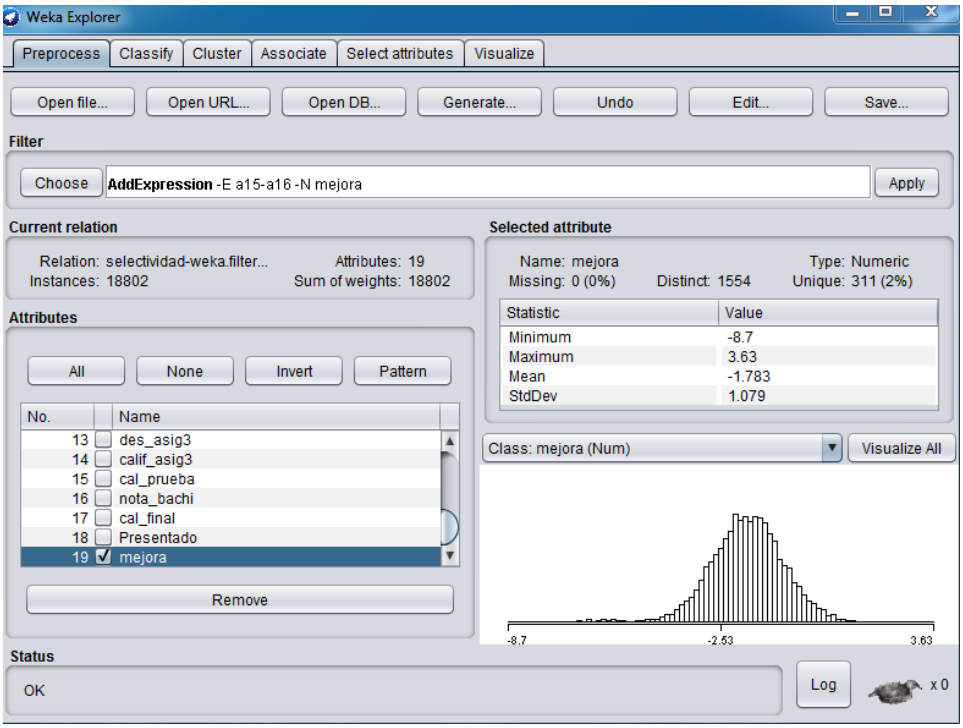
**Actividad 3.2.** Realiza una nueva discretización de la relación (eliminando el efecto del filtro anterior y dejando la relación original con el botón **Undo**) que divida las calificaciones en 4 intervalos de la misma frecuencia, lo que permite determinar los cuatro cuartiles (intervalos al 25%) de la calificación en la prueba: los intervalos delimitados por los valores {4, 4.8, 5.76}.

### Filtros de añadir expresiones

Muchas veces es interesante incluir nuevos atributos resultantes de aplicar expresiones a los existentes, lo que puede traer información de interés o formular cuestiones interesantes sobre los datos. Por ejemplo, vamos a añadir como atributo de interés la "mejora" sobre la nota de bachillerato, lo que puede servir para calificar el "éxito" en la prueba. Seleccionamos el filtro de atributos **AddExpression**, configurado para obtener la diferencia entre los atributos calificación en la prueba y nota de bachillerato, en las posiciones 15 y 16:




Después de aplicarlo aparece este atributo en la relación, sería el número 19, con el histograma indicado en la figura:




### 3.4.2.2. Filtros de instancias

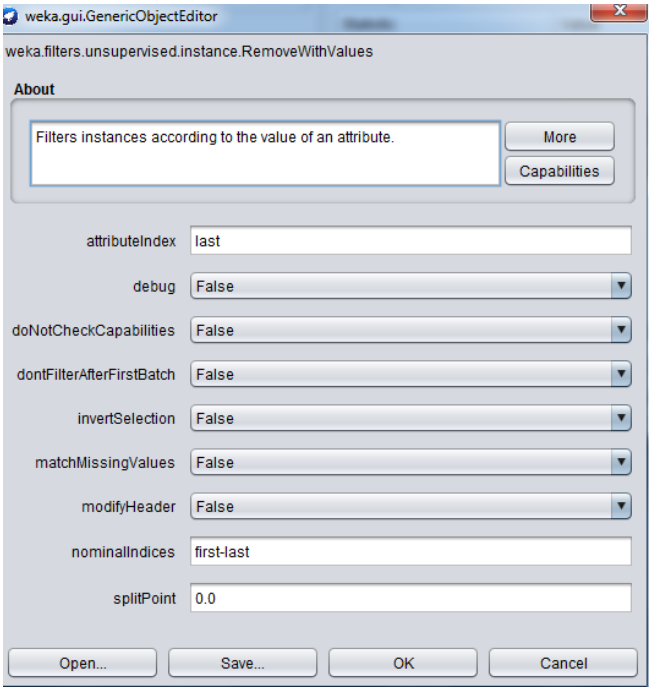
De entre todas las posibilidades implementadas para filtros de selección de instancias (selección de rangos, muestreos, etc.), nos centraremos en la utilización de filtros para seleccionar instancias cuyos atributos cumplen determinadas condiciones.

**Selección de instancias con condiciones sobre atributos**

Vamos a utilizar el filtro **RemoveWithValues**, que elimina las instancias de acuerdo con condiciones definidas sobre uno de los atributos.

Este filtro permite verificar una condición simple sobre un atributo. Sin embargo, es posible hacer un filtro más complejo con varias condiciones aplicadas a uno o varios atributos sin más que aplicar en cascada varios filtros.

Las opciones que aparecen en la ventana de configuración son las siguientes:




El atributo utilizado para filtrar se indica en "attributeIndex". Si es un atributo nominal, se indican los valores a filtrar en parámetro, "nominalIndices". Si es numérico, se filtran las instancias con un valor inferior al punto de corte, "splitPoint". Se puede invertir el criterio de filtrado mediante el campo "invertSelection".

**Actividad 3.3.** Utiliza tres filtros de este tipo para seleccionar los alumnos de Getafe y Leganés con una calificación de la prueba entre 6.0 y 8.0. Comprueba el efecto de filtrado visualizando los histogramas de los atributos correspondientes (localidad y calificación en la prueba).

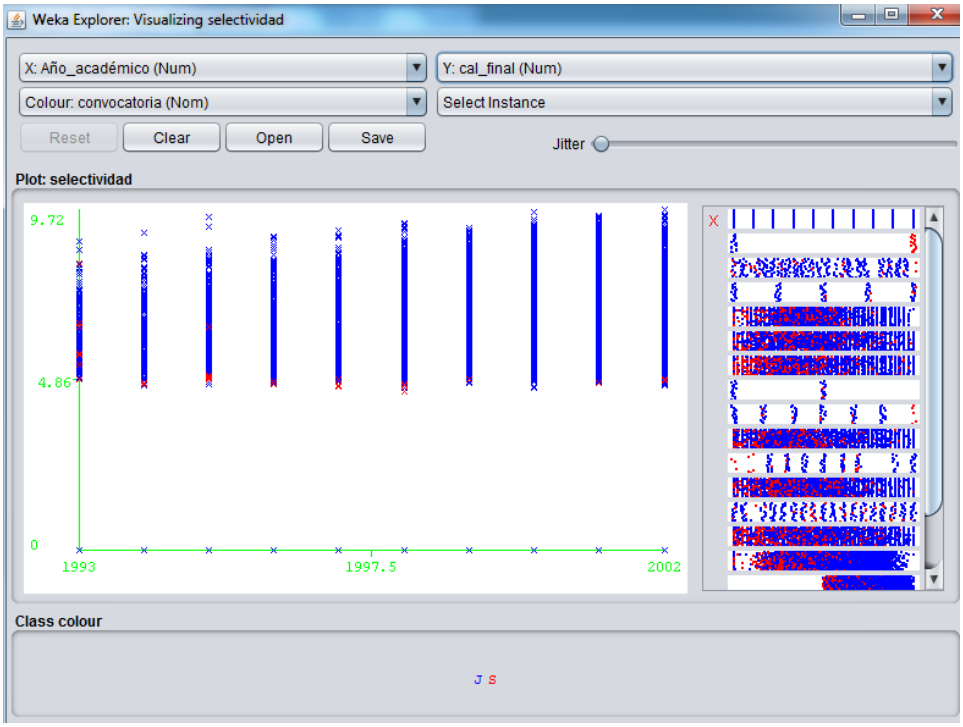
## 3.5. Visualización

Una de las primeras etapas del análisis de datos puede ser el mero análisis visual de estos, lo que en ocasiones es de gran utilidad para desvelar relaciones de interés utilizando nuestra capacidad para comprender imágenes. La herramienta de visualización de WEKA permite presentar gráficas 2D que relacionen pares de atributos, con la opción de utilizar además los colores para añadir información de un tercer atributo. Además, tiene incorporada una facilidad interactiva para seleccionar instancias con el ratón.

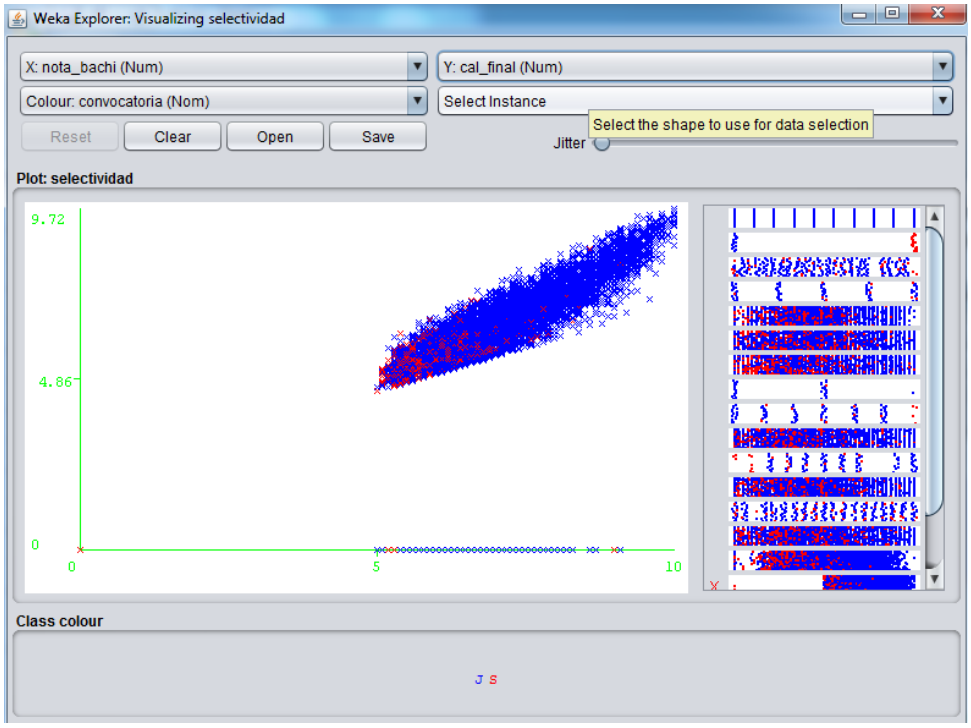
### 3.5.1. Representación 2D de los datos

Las instancias se pueden visualizar en gráficas 2D que relacionan pares de atributos. Al seleccionar la opción **Visualize** del *Explorer* aparecen todas los pares posibles de atributos en las coordenadas horizontal y vertical. La idea es que se selecciona la gráfica deseada para verla en detalle en una ventana nueva. En nuestro caso, aparecerán

todas las combinaciones posibles de atributos. Como primer ejemplo vamos a visualizar el rango de calificaciones finales de los alumnos a lo largo de los años, poniendo la convocatoria (junio o septiembre) como color de la gráfica.




Vamos a visualizar ahora dos variables cuya relación es de gran interés, la calificación de la prueba en función de la nota de bachillerato, usando como color la convocatoria (junio o septiembre).



En esta gráfica podemos apreciar la relación entre ambas magnitudes, que si bien no es directa al menos define una cierta tendencia creciente, y cómo la convocatoria está bastante relacionada con ambas calificaciones.

Cuando lo que se relacionan son variables simbólicas, se presentan sus posibles valores a lo largo del eje. Sin embargo, en estos casos todas las instancias que comparten cada valor de un atributo simbólico pueden ocultarse (serían un único punto en el plano), razón por la que se utiliza la opción de **Jitter**. Esta opción permite introducir un desplazamiento aleatorio (ruido) en las instancias, con objeto de poder visualizar todas aquellas que comparten un par de valores de atributos simbólicos, de manera que puede visualizarse la proporción de instancias que aparece en cada región.

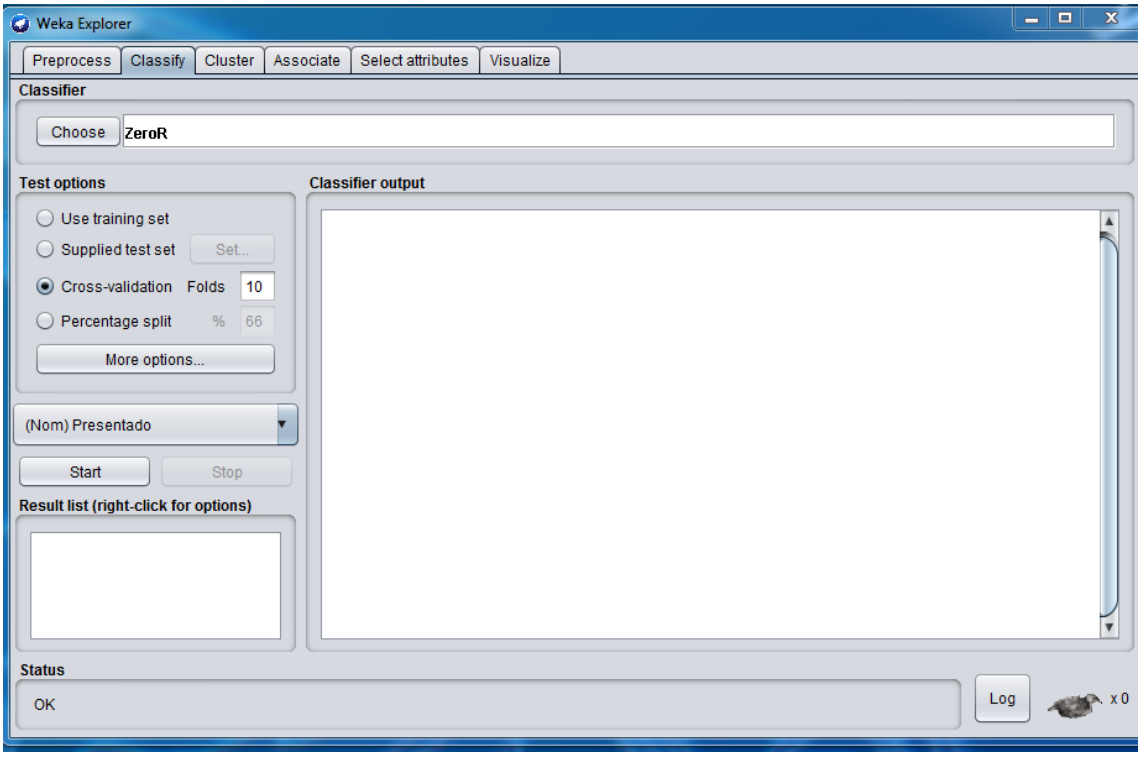
**Actividad 3.4.** Visualiza la relación entre las tres asignaturas optativas, y con la opción cursada como color

## 3.6. Clasificación

En esta sección abordamos el problema de la clasificación, que es el más frecuente en la práctica. En ocasiones, el problema de clasificación se formula como un refinamiento en el análisis, una vez que se han aplicado algoritmos no supervisados de agrupamiento y asociación para describir relaciones de interés en los datos.

WEKA incorpora una lista amplia para resolver tareas de clasificación. En este apartado se presentan varios ejemplos de aplicación de algoritmos de clasificación. El epígrafe 3.7 está dedicado a la exposición de los principales algoritmos de clasificación implementados en WEKA, además, se describirán los principales parámetros de configuración de los mismos.

Se pretende construir un modelo que permita predecir la categoría de las instancias en función de una serie de atributos de entrada. En el caso de WEKA, la clase es simplemente uno de los atributos simbólicos disponibles, que se convierte en la variable objetivo a predecir. Por defecto, es el último atributo (última columna) a no ser que se indique otro explícitamente. La configuración de la clasificación se efectúa en la ventana siguiente:




En la parte superior se puede seleccionar el algoritmo de clasificación y configurarlo. El resto de elementos a definir en esta ventana se describen a continuación.

### 3.6.1. Modos de evaluación del clasificador

El resultado de aplicar el algoritmo de clasificación se efectúa comparando la clase predicha con la clase real de las instancias. Esta evaluación puede realizarse de diferentes modos, según la selección en el cuadro **Test options**:

- **Use training set:** esta opción evalúa el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo para determinar el error, que en este caso se denomina "error de resustitución". Por tanto, esta opción puede proporcionar una estimación demasiado optimista del comportamiento del clasificador, al evaluarlo sobre el mismo conjunto sobre el que se hizo el modelo.
- **Supplied test set**: evaluación sobre conjunto independiente. Esta opción permite cargar un conjunto nuevo de datos. Sobre cada dato se realizará una predicción de clase para contar los errores.
- **Cross-validation**: evaluación con validación cruzada. Esta opción es la más elaborada y costosa. Se realizan tantas evaluaciones como se indica en el parámetro **Folds**. Se dividen las instancias en tantas carpetas como indica este parámetro y en cada evaluación se toman las instancias de cada carpeta como datos de test, y el resto como datos de entrenamiento para construir el modelo. Los errores calculados son el promedio de todas las ejecuciones.
- **Percentage split :** esta opción divide los datos en dos grupos, de acuerdo con el porcentaje indicado (**%**). El valor indicado es el porcentaje de instancias para construir el modelo, que a continuación es evaluado sobre

las que se han dejado aparte. Cuando el número de instancias es suficientemente elevado, esta opción es suficiente para estimar con precisión las prestaciones del clasificador en el dominio.

Además de estas opciones para seleccionar el modo de evaluación, el botón **More Options** abre un cuadro con otras opciones adicionales:

- **Output model**: permite visualizar (en modo texto y, con algunos algoritmos, en modo gráfico) el modelo construido por el clasificador (árbol, reglas, etc.)
- **Output per-class stats**: obtiene estadísticas de los errores de clasificación por cada uno de los valores que toma el atributo de clase
- **Output entropy evaluation measures**: genera medidas de evaluación de entropía
- **Store predictions for visualization:** permite analizar los errores de clasificación en una ventana de visualización
- **Cost-sensitive evaluation**: con esta opción se puede especificar una función con costes relativos de los diferentes errores, que se rellena con el botón **Set**.

En nuestro ejemplo utilizaremos los valores predeterminados de estas últimas opciones.

**Evaluación del clasificador en ventana de texto**

Una vez se ejecuta el clasificador seleccionado sobre los datos de la relación, en la ventana de texto de la derecha aparece información de ejecución, el modelo generado con todos los datos de entrenamiento y los resultados de la evaluación. Por ejemplo, al predecir el atributo "presentado", con un árbol de decisión de tipo J48, aparece el modelo textual siguiente:

```
J48 pruned tree
------------------
cal_prueba <= 0: NO (153.0)
cal_prueba > 0: SI (18649.0/2.0)
Number of Leaves : 2
Size of the tree : 3
```

A partir de los datos se obtiene esta relación trivial, salvo dos únicos casos de error: los presentados son los que tienen una calificación superior a 0.

Con referencia al informe de evaluación del clasificador, podemos destacar tres elementos:

- **Resumen** (*Summary*): es el porcentaje global de errores cometidos en la evaluación
- **Precisión detallada por clase**: para cada uno de los valores que puede tomar el atributo de clase: el porcentaje de instancias con ese valor que son correctamente predichas (TP: true positives), y el porcentaje de instancias con otros valores que son incorrectamente predichas a ese valor aunque tenían otro (FP: false positives).

Las otras columnas, *precision*, *recall*, *Fmeasure*, … se relacionan con estas dos anteriores.

- **Matriz de confusión**: aquí aparece la información detallada de cuantas instancias de cada clase son predichas a cada uno de los valores posibles. Por tanto, es una matriz con $N^2$ posiciones, siendo N el número de valores que puede tomar la clase. En cada fila, i, i=1...N, aparecen las instancias que realmente son de la clase i, mientras que las columnas, j, j=1...N, son las que se han predicho al valor j de la clase. En el ejemplo anterior, la matriz de confusión que aparece es la siguiente:

```
=== Confusion Matrix ===
    a   b <-- classified as
18647   0 | a = SI
    2 153 | b = NO
```

por tanto, los valores en la diagonal son los aciertos, y el resto de valores son los errores. De los 18647 alumnos presentados, todos son correctamente clasificados, mientras que de los 155 no presentados, hay 153 correctamente clasificados y 2 con error.

**Lista de resultados**

Al igual que con otras opciones de análisis, la ventana izquierda de la lista de resultados contiene el resumen de todas las aplicaciones de clasificadores sobre conjuntos de datos en la sesión del *Explorer*. Se puede acceder a esta lista para presentar los resultados, y al pulsar el botón derecho aparecen diferentes opciones de visualización, entre las que podemos destacar las siguientes:

- Salvar y cargar modelos: **Load model, Save model**. Estos modelos pueden salvarse a un fichero para posteriormente recuperarlos y aplicarlos a nuevos conjuntos de datos
- Visualizar árbol y errores de predicción: **Visualize tree, Visualize classifier errors**,...

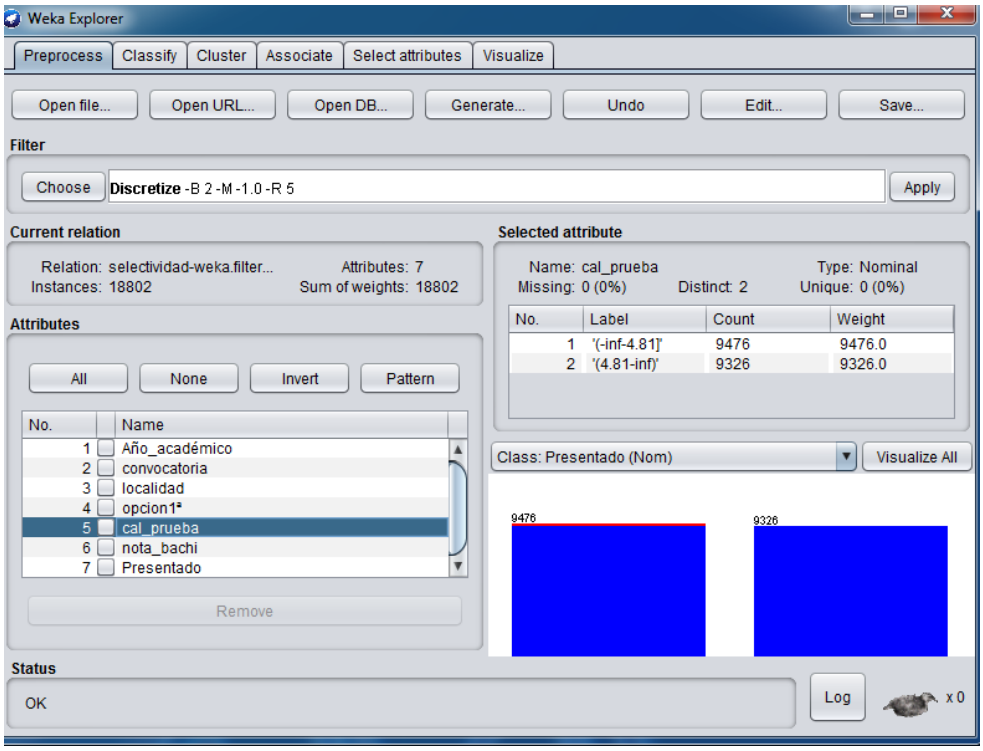
**Actividad 3.5.** Obtén el árbol de decisión (gráfico) del ejemplo de clasificación anterior.

## 3.6.2. Selección y configuración de clasificadores

Vamos a ilustrar la aplicación de algoritmos de clasificación a diferentes problemas de predicción de atributos definidos sobre los datos de entrada en este ejemplo. Solo aplicaremos algunos algoritmos de clasificación. Como ya se ha comentado, más adelante se presenta una lista de los principales algoritmos de clasificación implementados en WEKA, así como las características principales de cada uno de ellos.

El problema de clasificación siempre se realiza sobre un atributo simbólico, en el caso de utilizar un atributo numérico se precisa por tanto discretizarlo antes en intervalos

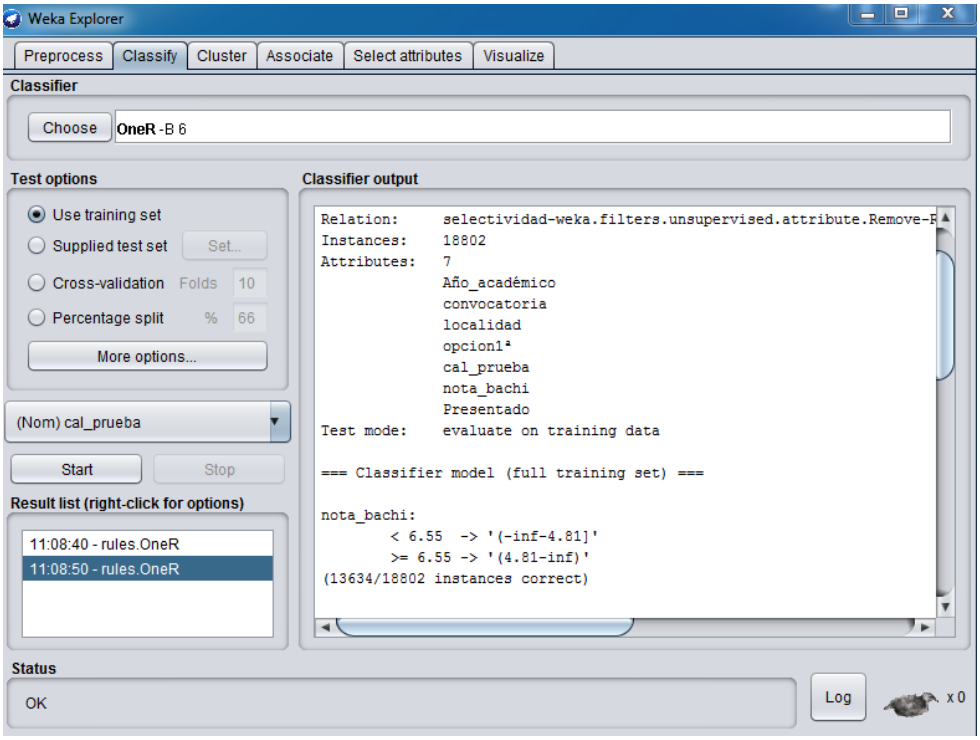
que representarán los valores de clase. En primer lugar efectuaremos análisis de predicción de la calificación en la prueba de selectividad a partir de los siguientes atributos: año, convocatoria, localidad, opción, presentado y nota de bachillerato. Se van a realizar dos tipos de predicciones: aprobados, e intervalos de clasificación. Por tanto, tenemos que aplicar en primer lugar una combinación de filtros que elimine los atributos no deseados relativos a calificaciones parciales y asignaturas opcionales (Remove (5-14, 17), y un filtro que discretice la calificación en la prueba en dos partes (discretize (Bin =2). Se debe marcar que solo aplique el filtro a la calificación en la prueba, en caso de marcar first-last, aplicará el filtro a todos los atributos):




Se ha optado por realizar las predicciones sobre la calificación en la prueba, puesto que la calificación final depende explícitamente de la nota del bachillerato.

**Clasificador "OneR"**

Este es uno de los clasificadores más sencillos y rápidos, aunque en ocasiones sus resultados son sorprendentemente buenos en comparación con algoritmos mucho más complejos. Simplemente selecciona el atributo que mejor "explica" la clase de salida. Si hay atributos numéricos, busca los umbrales para hacer reglas con mejor tasa de aciertos. Lo aplicaremos al problema de predicción de aprobados en la prueba a partir de los atributos de entrada, para llegar al resultado siguiente:




Por tanto, el algoritmo llega a la conclusión que la mejor predicción posible con un solo atributo es la nota del bachillerato, fijando el umbral que determina el éxito en la prueba en 6.55. La tasa de aciertos sobre el propio conjunto de entrenamiento es del 72.5%.

**Actividad 3.6.** Compara este resultado con el obtenido al utilizar los otros modos de evaluación del clasificador posibles.

### Clasificador como árbol de decisión: J48

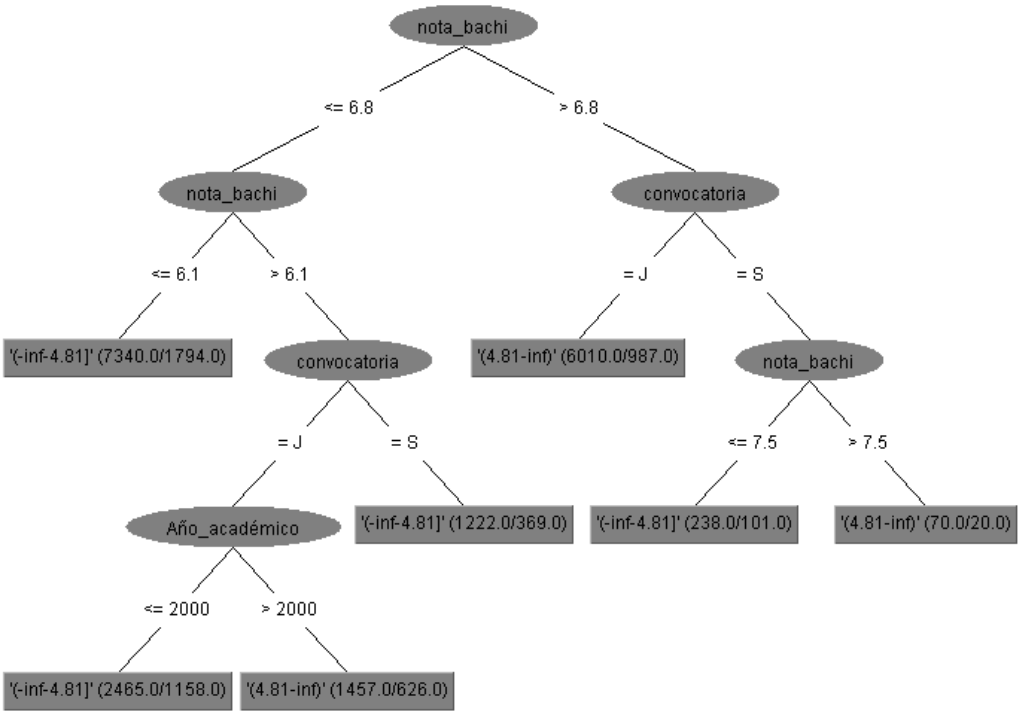
El algoritmo J48 de WEKA es una implementación del algoritmo C4.5, uno de los algoritmos de minería de datos que se ha utilizado en multitud de aplicaciones. No vamos a entrar en los detalles de todos los parámetros de configuración y tan solo resaltaremos uno de los más importantes, el factor de confianza para la poda, **confidence level**, puesto que influye notoriamente en el tamaño y capacidad de predicción del árbol construido.

Una explicación simplificada de este parámetro de construcción del árbol es la siguiente: para cada operación de poda, define la probabilidad de error que se permite a la hipótesis de que el empeoramiento debido a esta operación es significativo. Cuanto más baja se haga esa probabilidad, se exigirá que la diferencia en los errores de predicción antes y después de podar sea más significativa para no podar. El valor por defecto de este factor es del 25%, y conforme va bajando se permiten más operaciones de poda y por tanto llegar a árboles cada vez más pequeños. Otra forma de variar el tamaño del árbol es a través de un parámetro que especifica el mínimo número de instancias por nodo, si bien

es menos elegante puesto que depende del número absoluto de instancias en el conjunto de partida.

Construiremos el árbol de decisión con los parámetros por defecto del algoritmo J48: se llega a un clasificador con más de 250 nodos, con una probabilidad de acierto ligeramente superior al del clasificador *OneR*.

Vamos a modificar la configuración del algoritmo para llegar a un árbol más manejable, como el que se presenta a continuación.




En este caso, se ha seleccionado un valor del factor de confianza de 0.01 para la poda y como mínimo número de instancias por nodo 50.

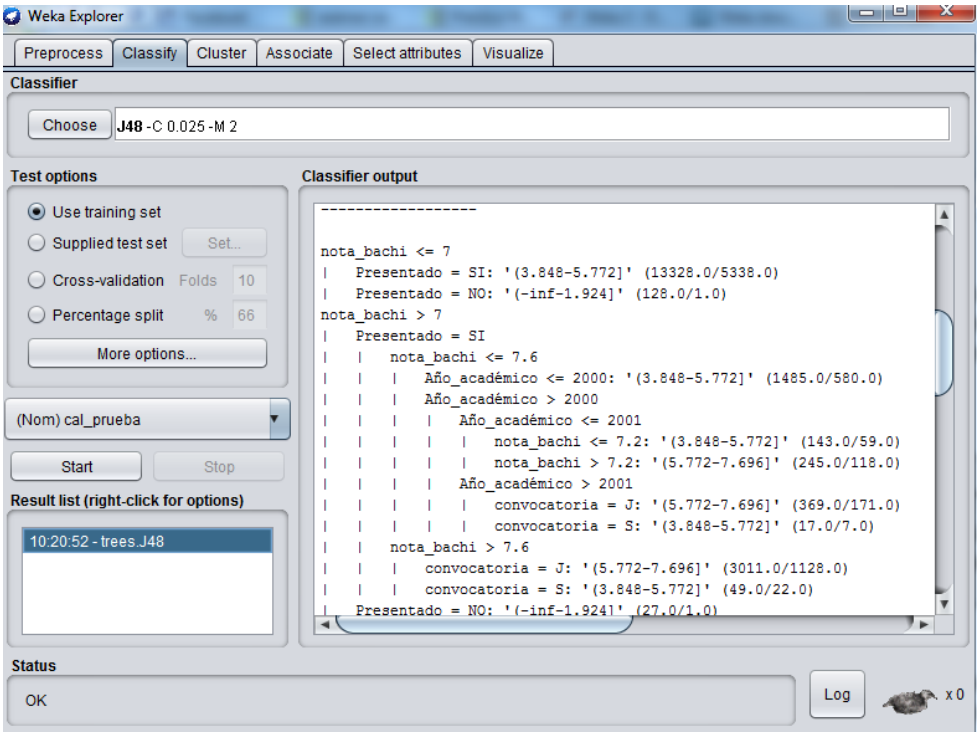
Este modelo es un refinamiento del generado con OneR, que supone una mejora moderada en las prestaciones. De nuevo los atributos más importantes son la calificación de bachillerato, la convocatoria, y después el año, antes que la localidad o las opciones.

**Actividad 3.7.** Realiza de nuevo el árbol utilizando en este caso un valor del factor de confianza de 0.05 para la poda y como mínimo número de instancias por nodo 50. Compara los resultados obtenidos.

Otros problemas de clasificación pueden formularse sobre cualquier atributo de interés. A continuación mostramos algunos ejemplos a título ilustrativo.

### Clasificación multinivel de las calificaciones

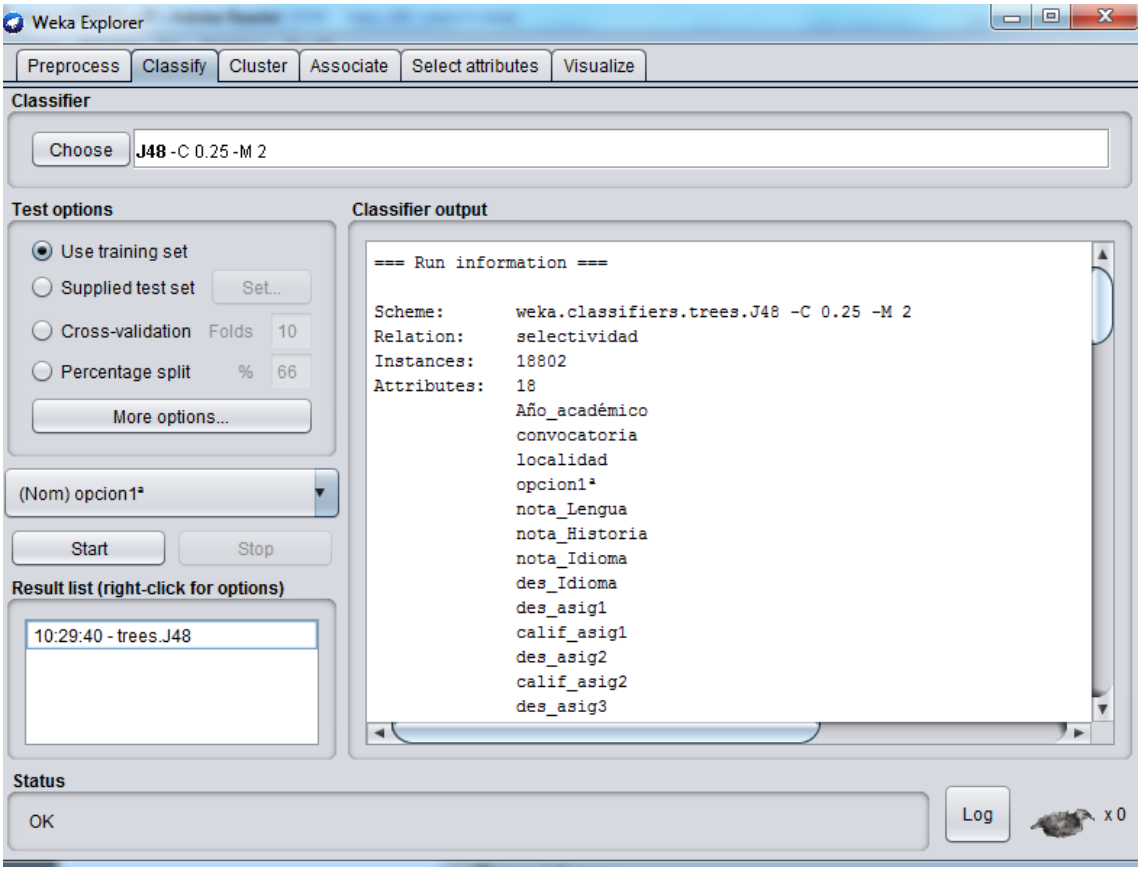
El problema anterior puede intentar refinarse y dividir el atributo de interés, la calificación final, en más niveles, en este caso 5. Los resultados se muestran a continuación.

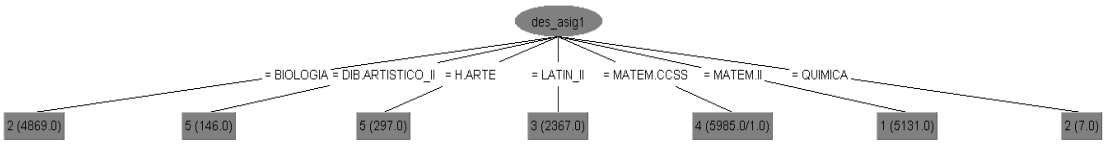



**Actividad 3.8.** Comenta los resultados sobre la precisión y tamaño del ejemplo anterior.

### Predicción de la opción

Si dejamos todos los atributos en la muestra y aplicamos el clasificador a la opción *cursado*, se desvela una relación trivial entre opción y asignaturas en las opciones que predice bien el 100% de los casos.




## Mejora en la prueba

Un problema de clasificación interesante puede ser determinar qué alumnos tienen más "éxito" en la prueba, en el sentido de mejorar su calificación de bachillerato con la calificación en la prueba. Para ello utilizaremos el atributo "mejora", introducido anteriormente, y lo discretizamos en dos valores con la misma frecuencia. Obtenemos una mediana de -1.75, de manera que dividimos los alumnos en dos grupos: los que obtienen una diferencia inferior a este valor y los que la obtienen superior, para diferenciar los alumnos según que el resultado se atenga más o menos a sus expectativas. Evidentemente, para evitar construir modelos triviales, tenemos que eliminar los atributos relacionados con las calificaciones en la prueba. Por tanto, se utilizan los siguientes atributos:

```
Attributes: 7
Año_académico
convocatoria
localidad
opcion1ª
nota_bachi
Presentado
mejora
```

**Actividad 3.9.** Realiza el problema de clasificación anterior y comenta los resultados obtenidos.

## 3.7. Implementación de las técnicas de clasificación en Weka

### 3.7.1. Tabla de Decisión en WEKA

El algoritmo de tabla de decisión implementado en la herramienta WEKA se encuentra en la clase *weka.classifiers.DecisionTable.java*. Las opciones de configuración de que dispone son las indicadas en la siguiente tabla:

| Opción | Descripción |
|---|---|
| displayRules (False) | Predeterminadamente no se muestran las reglas del clasificador, concretamente la tabla de decisión construida. |
| maxStale (5) | Indica el número máximo de conjuntos que intenta mejorar el algoritmo para encontrar una tabla mejor sin haberla encontrado en los últimos *n-1* subconjuntos. |
| crossVal (1) | Predeterminadamentese evalúa el sistema mediante el proceso *leave-one-out*. Si se aumenta el valor de *1* se realiza validación cruzada con *n* carpetas |

En primer lugar, en cuanto a los atributos que permite el sistema, estos pueden ser tanto numéricos (que se discretizarán) como simbólicos. La clase también puede ser numérica o simbólica.

El algoritmo consiste en ir seleccionando uno a uno los subconjuntos, añadiendo a cada uno de los ya probados cada uno de los atributos que aún no pertenecen a él. Se prueba la precisión del subconjunto, bien mediante validación cruzada o *leave-one-out* y,

si es mejor, se continúa con él. Se continúa así hasta que se alcanza *maxStale*. Para ello, una variable comienza siendo *0* y aumenta su valor en una unidad cuando a un subconjunto no se le puede añadir ningún atributo para mejorarlo, volviendo a *0* si se añade un nuevo atributo a un subconjunto.

En cuanto al proceso *leave-one-out*, es un método de estimación del error. Es una validación cruzada en la que el número de conjuntos es igual al número de ejemplos de entrenamiento. Cada vez se elimina un ejemplo del conjunto de entrenamiento y se entrena con el resto. Se juzgará el acierto del sistema con el resto de instancias según se acierte o se falle en la predicción del ejemplo que se eliminó. El resultado de las *n* pruebas (siendo *n* el número inicial de ejemplos de entrenamiento) se promedia y dicha media será el error estimado. Por último, para clasificar un ejemplo pueden ocurrir dos cosas. En primer lugar, que el ejemplo corresponda exactamente con una de las reglas de la tabla de decisión, en cuyo caso se devolverá la clase de dicha regla. Si no se corresponde con ninguna regla, se puede utilizar *Ibk* (si se seleccionó dicha opción) para predecir la clase, o la media o moda de la clase según el tipo de clase del que se trate (numérica o simbólica).

### 3.7.2 ID3 en WEKA

La clase en la que está codificado el algoritmo ID3 es *weka.classifiers.ID3.java*. En primer lugar, en cuanto a la implementación, no permite ningún tipo de configuración. Esta implementación se ajusta exactamente a lo descrito anteriormente. Lo único reseñable es que para determinar si un nodo es hoja o no, se calcula la ganancia de información y, si la máxima ganancia es *0* se considera nodo hoja, independientemente de que haya ejemplos de distintas clases en dicho nodo. Los atributos introducidos al sistema deben ser simbólicos, al igual que la clase.

### 3.7.3. C4.5 en WEKA (J48)

La clase en la que se implementa el algoritmo C4.5 en WEKA es *weka.classifers.j48.J48.java*. Las opciones que permite este algoritmo son las que se muestran en la siguiente tabla.

| **Opción** | **Descripción** |
|---|---|
| minNumObj (2) | Número mínimo de instancias por hoja. |
| saveInstanceData(False) | Una vez finalizada la creación del árbol de decisión se eliminan todas las instancias que se clasifican en cada nodo, que hasta el momento se mantenían almacenadas. |
| binarySplits (False) | Con los atributos nominales tampoco se divide (por defecto) cada nodo en dos ramas. |
| unpruned (False) | En caso de no activar la opción, se realiza la poda del árbol. |
| subtreeRaising (True) | Se permite realizar el podado con el proceso *subtree raising*. |

| Opción | Descripción |
|---|---|
| confidenceFactor (0.25) | Factor de confianza para el podado del árbol. |
| reducedErrorPruning (False) | Si se activa esta opción, el proceso de podado no es el propio de C4.5, sino que el conjunto de ejemplos se divide en un subconjunto de entrenamiento y otro de test, de los cuales el último servirá para estimar el error para la poda. |
| numFolds (3) | Define el número de subconjuntos en que hay que dividir el conjunto de ejemplos para emplear el último de ellos como conjunto de test si se activa la opción *reducedErrorPruning*. |
| useLaplace (False) | Si se activa esta opción, cuando se intenta predecir la probabilidad de que una instancia pertenezca a una clase, se emplea el *suavizado de Laplace*. |

El algoritmo J48 se ajusta al algoritmo C4.5, ampliando funcionalidades tales como permitir la realización del proceso de podado mediante *reducedErrorPruning* o que las divisiones sean siempre binarias *binarySplits*.

Pueden admitirse atributos simbólicos y numéricos y se permiten ejemplos con faltas en dichos atributos, tanto en el momento de entrenamiento como en la predicción de dicho ejemplo. En cuanto a la clase, esta debe ser simbólica.

### 3.7.4. Árbol de Decisión de un solo nivel en WEKA

La clase en la que se implementa este algoritmo en WEKA es *weka.classifers.DecisionStump.java*. No tiene opciones de configuración, pero la implementación es muy completa, dado que admite tanto atributos numéricos como simbólicos y clases de ambos tipos también. El árbol de decisión tendrá tres ramas: una de ellas será para el caso de que el atributo sea desconocido, y las otras dos serán para el caso de que el valor del atributo del ejemplo de test sea igual a un valor concreto del atributo o distinto a dicho valor, en caso de los atributos simbólicos, o que el valor del ejemplo de test sea mayor o menor que un determinado valor en el caso de atributos numéricos.

En el caso de los atributos simbólicos se considera cada valor posible del mismo y se calcula la ganancia de información con el atributo igual al valor, distinto al valor y valores perdidos del atributo. En el caso de atributos simbólicos se busca el mejor punto de ruptura.

Lo que se busca es el valor mínimo de la ecuación calculada, ya sea la entropía o la varianza. De esta forma se obtiene el atributo que será raíz del árbol de decisión y sus tres ramas. Lo que se hará por último es construir dicho árbol: cada rama finaliza en un nodo *hoja* con el valor de la clase, que será la media o la moda de los ejemplos que se clasifican por ese camino, según se trate de una clase numérica o simbólica.

### 3.7.5. 1R en WEKA

La clase *weka.classifers.OneR.java* implementa el algoritmo 1R. La única opción configurable es *minBucketSize* que indica el número mínimo de ejemplos que deben pertenecer a un conjunto en caso de atributo numérico.

La implementación que se lleva a cabo en WEKA de este algoritmo cumple exactamente con lo descrito anteriormente.

1R es un clasificador muy sencillo, que únicamente utiliza un atributo para la clasificación. Sin embargo, aún hay otro clasificador más sencillo, el 0R, implementado en *weka.classifers.ZeroR.java*, que simplemente calcula la media en el caso de tener una clase numérica o la moda, en caso de una clase simbólica. No tiene ningún tipo de opción de configuración.

### 3.7.6. PRISM en WEKA

La clase *weka.classifers.Prism.java* implementa el algoritmo PRISM. No tiene ningún tipo de configuración posible. Únicamente permite atributos nominales, la clase debe ser también nominal y no puede haber atributos con valores desconocidos. La implementación de esta clase sigue completamente el algoritmo expuesto en el tema anterior.

### 3.7.7. PART en WEKA

La clase *weka.classifers.j48.PART.java* implementa el algoritmo PART. A continuación, se muestran las opciones de configuración de dicho algoritmo.

| Opción | Descripción |
|---|---|
| minNumObj (2) | Número mínimo de instancias por hoja. |
| binarySplits (False) | Con los atributos nominales tampoco se divide (por defecto) cada nodo en dos ramas. |
| confidenceFactor(0.25) | Factor de confianza para el podado del árbol. |
| reducedErrorPruning (False) | Si se activa esta opción, el proceso de podado no es el propio de C4.5, sino que el conjunto de ejemplos se divide en un subconjunto de entrenamiento y otro de test, de los cuales el último servirá para estimar el error para la poda. |
| numFolds (3) | Define el número de subconjuntos en que hay que dividir el conjunto de ejemplos para emplear el último de ellos como conjunto de test si se activa la opción *reducedErrorPruning*. |

Como se ve en la tabla anterior, las opciones del algoritmo PART son un subconjunto de las ofrecidas por J48, que implementa el sistema C4.5. Esto es debido a que PART emplea muchas de las clases que implementan C4.5, por lo que los cálculos de la entropía, del error esperado,... son los mismos.

La implementación que se realiza en WEKA del sistema PART se corresponde exactamente con lo comentado anteriormente, y más teniendo en cuenta que los implementadores de la versión son los propios creadores del algoritmo.

Por último, en cuanto a los tipos de datos admitidos por el algoritmo, estos son numéricos y simbólicos para los atributos, y simbólicos para la clase.

### 3.7.8. *Naive* Bayesiano en WEKA

El algoritmo *naive* Bayesiano se encuentra implementado en la clase *weka.classifiers.NaiveBayesSimple.java*. No dispone de ninguna opción de configuración. El algoritmo que implementa esta clase se corresponde completamente con el expuesto anteriormente. En este caso no se usa el *estimador de Laplace*, sino que la aplicación muestra un error si hay menos de dos ejemplos de entrenamiento para una terna *atributo-valor-clase* o si la desviación típica de un atributo numérico es igual a *0*.

Una alternativa a esta clase que también implementa un clasificador *naive* Bayesiano es la clase *weka.classifiers.NaiveBayes.java*. Tan solo tiene una opción configurable, useKernelEstimator(False), que permite emplear un estimador de densidad de núcleo para modelar los atributos numéricos en lugar de una distribución normal.

### 3.7.9. KNN en WEKA (IBk)

En WEKA se implementa el clasificador KNN con el nombre IBk, concretamente en la clase *weka.classifiers.IBk.java*. Además, en la clase *weka.classifiers.IB1.java* hay una versión simplificada del mismo, concretamente un clasificador NN (Nearest Neighbor), sin ningún tipo de opción, en el que, como su propio nombre indica, tiene en cuenta únicamente el voto del vecino más cercano. El clasificador IBk permite las siguientes opciones:

| Opción | Descripción |
|---|---|
| KNN (1) | Número de vecinos más cercanos. |
| distanceWeighting (No distance weighting) | Los valores posibles son: *No distance weighting*, *Weight by 1-distance* y *Weight by 1/distance*. Permite definir si se deben "pesar" o no a los vecinos a la hora de votar, y, en su caso, hacerlo bien según su semejanza, bien con la inversa de su distancia con respecto al ejemplo a clasificar. |
| crossValidate (False) | Si se activa esta opción, cuando se vaya a clasificar una instancia se selecciona el número de vecinos (hasta el número especificado en la opción KNN) mediante el proceso *hold-one-out*. |
| meanSquared (False) | Minimiza el error cuadrático en lugar del error absoluto para el caso de clases numéricas cuando se activa la opción *crossValidate*. |

| Opción | Descripción |
|---|---|
| windowSize (0) | Si es *0* el número de ejemplos de entrenamiento es ilimitado. Si es mayor que *0*, únicamente se almacenan los *n* últimos ejemplos de entrenamiento, siendo *n* el número que se ha especificado. |
| debug (False) | Muestra el proceso de construcción del clasificador. |
| noNormalization (False) | No normaliza los atributos. |

El algoritmo implementado en WEKA consiste en crear el clasificador a partir de los ejemplos de entrenamiento, simplemente almacenando todas las instancias disponibles (a menos que se restrinja con la opción *windowSize*). Posteriormente, se clasificarán los ejemplos de test a partir del clasificador generado, bien con el número de vecinos especificados o comprobando el mejor *k* si se activa la opción *crossValidate*.

En cuanto a los tipos de datos permitidos y las propiedades de la implementación, estos son:

- Admite atributos numéricos y simbólicos.
- Admite clase numérica y simbólica. Si la clase es numérica se calculará la media de los valores de la clase para los *k* vecinos más cercanos.
- Permite dar peso a cada ejemplo.
- El proceso de *hold-one-out* consiste en, para cada *k* entre *1* y el valor configurado en KNN, calcular el error en la clasificación de los ejemplos de entrenamiento. Se escoge el *k* con un menor error obtenido. El error cometido para cada *k* se calcula como el error medio absoluto o el cuadrático si se trata de una clase numérica.

### 3.7.10. K* en WEKA

La clase en la que se implementa el algoritmo K* en WEKA es *weka.classifers.kstar.KStar.java*. Las opciones que admite este algoritmo son:

| Opción | Descripción |
|---|---|
| entropicAutoBlend (False) | Si se activa esta opción se calcula el valor de los parámetros $x_0$ (o *s*) basándose en la entropía en lugar del parámetro de mezclado. |
| globalBlend (20) | Parámetro de mezclado, expresado en tanto por ciento. |
| missingMode (Average column entropy curves) | Define cómo se tratan los valores desconocidos en los ejemplos de entrenamiento; las opciones posibles son *Ignore the Instance with missing value* (no se tienen en cuenta los ejemplos con atributos desconocidos), *Treat missing value as maximally different* (diferencia igual al del vecino más lejano considerado), *Normalize over the attributes* (se ignora el atributo desconocido) y *Average column entropy curves*. |

Dado que los autores de la implementación de este algoritmo en WEKA son los autores del propio algoritmo, dicha implementación se corresponde perfectamente con lo visto anteriormente. Simplemente son destacables los siguientes puntos:

- Admite atributos numéricos y simbólicos, así como pesos por cada instancia.
- Permite que la clase sea simbólica o numérica.
- Proporciona cuatro modos de actuación frente a pérdidas en los atributos en ejemplos de entrenamiento.
- Para el cálculo de los parámetros $x_0$ y $s$ permite basarse en el parámetro $b$ o en el cálculo de la entropía.
- Como ya se comentó en su momento, las ecuaciones para el cálculos de $P^*$ y de la *esfera de influencia* varían ligeramente.

### 3.7.11. Redes de Neuronas en WEKA

La clase en la que se implementan las redes de neuronas en WEKA es *weka.classifiers.neural.NeuralNetwork.java*. Las opciones que permite configurar son:

| Opción | Descripción |
|---|---|
| momentum (0.2) | Factor que se utiliza en el proceso de actualización de los pesos. Se multiplica este parámetro por el peso en el momento actual (el que se va a actualizar) y se suma al peso actualizado. |
| validationSetSize (0) | Determina el porcentaje de patrones que se emplearán como test del sistema. De esta forma, tras cada entrenamiento se validará el sistema, y terminará el proceso de entrenamiento si la validación da un valor menor o igual a *0*, o si se superó el número de entrenamientos configurado. |
| nominalToBinaryFilter (False) | Transforma los atributos nominales en binarios. |
| learningRate (0.3) | Razón de aprendizaje. Tiene valores entre *0* y *1*. |
| hiddenLayers (a) | Determina el número de neuronas ocultas. Sus posibles valores son: *'a'=(atribs+clases)/2*, *'i'=atribs*, *'o'=clases*, *'t'=atribs+clases*. |
| validationThreshold (20) | Si el proceso de validación arroja unos resultados en cuanto al error que empeoran durante *n* veces consecutivas (siendo *n* el valor de esta variable), se detiene el aprendizaje. |
| reset (True) | Permite al sistema modificar la razón de aprendizaje automáticamente (la divide entre *2*) y comenzar de nuevo el proceso de aprendizaje si el proceso de entrenamiento no converge. |
| GUI (False) | Visualización de la red de neuronas. Si se activa esta opción se puede modificar la red de neuronas, parar el proceso de entrenamiento en cualquier momento, modificar parámetros como el de la razón de aprendizaje, etc. |

| Opción | Descripción |
|---|---|
| autoBuild (True) | El sistema construye automáticamente la red basándose en las entradas, salidas y el parámetro *hiddenLayers*. |
| normalizeNumericClass (True) | Normaliza los posibles valores de la clase si esta es numérica, de forma que estén entre *–1* y *1*. |
| decay (False) | La razón de ganancia se modifica con el ciclo de aprendizaje: $\alpha = \alpha/n$, donde $n$ es el número de ciclo de aprendizaje actual. |
| trainingTime (500) | Número total de ciclos de aprendizaje. |
| normalizeAttributes (True) | Normaliza los atributos numéricos para que estén entre *–1* y *1*. |
| randomSeed (0) | Semilla para generar los números aleatorios que inicializarán los parámetros de la red. |

La implementación de redes de neuronas que se realiza en la herramienta se ciñe al algoritmo de retropropagación.

Algunas características que se pueden destacar de esta implementación son:

- Se admiten atributos numéricos y simbólicos.
- Se admiten clases numéricas (predicción) y simbólicas (clasificación).
- Permite la generación manual de redes que no se ciñan a la arquitectura mostrada anteriormente, por ejemplo, eliminando conexiones de neuronas de una capa con la siguiente.
- Como función sigmoidal se utiliza la restringida entre *0* y 1.
- Los ejemplos admiten pesos: Cuando se aprende con dicho ejemplo se multiplica la razón de aprendizaje por el peso del ejemplo. Todo esto antes de dividir la razón de aprendizaje por el número de ciclo de aprendizaje si se activa *decay*.

**Actividad 3.10.** Utilizando el fichero weather.nominal.arff , ejecuta el algoritmo de clasificación Id3 en los 3 casos siguientes:

- Use training set
- Cross validation
- Percentage split;

Describe el árbol obtenido. ¿Con que método de validación se han obtenido mejores porcentajes de bien clasificados?

**Actividad 3.11.** Aplica los siguientes clasificadores sobre el fichero de datos Drug1n.arff:

- ZeroR
- OneR
- Ibk

- NaiveBayes
- Id3
- j48

La validación se realizará sobre el mismo conjunto de aprendizaje. ¿Cuáles son los modelos que proporcionan los mejores resultados? ¿Has conseguido ejecutar todos los algoritmos? ¿Qué problemas has encontrado? ¿Cómo se pueden resolver?

**Actividad 3.12.**

Debes contestar de la forma más formal posible, además recuerda incluir capturas de pantalla de los pasos intermedios.

**1. Obtención de los datos**

Descarga el conjunto de datos iris.arff. Abre el fichero de datos con un editor, y estudia su contenido:

- ¿Cuántos atributos caracterizan los datos de esta tabla de datos?
- Si suponemos que queremos predecir el último atributo a partir de los anteriores, ¿estaríamos ante un problema de clasificación o de regresión?

**2. Estudio estadístico de los datos**

- Abre el fichero iris.arff en el Explorer de WEKA. Recuerda que en la sección attributes se puede pinchar sobre cada atributo para obtener información estadística del mismo.
- ¿Cuál es el rango de valores del atributo *petalwitdth*?
- ¿Con la información que puedes obtener visualmente, ¿qué atributos crees que son los que mejor permitirían predecir el atributo *class*?

**3. Aplicación de filtros**

- Aplica el filtro filters/unsupervised/attribute/normalize sobre el conjunto de datos. ¿Qué efecto tiene este filtro?
- Aplica el filtro filters/unsupervised/instance/RemovePercentage sobre el conjunto de datos. ¿Qué efecto tiene este filtro?
- Graba el conjunto de datos como iris2.arff.
- Aplica el filtro filters/unsupervised/attribute/Discretize sobre el conjunto de datos. ¿Qué efecto tiene este filtro?

**4. Visualización**

- Carga el conjunto de datos iris2.arff. Pulsa la pestaña Visualize. Aumenta Point Size a 5 para visualizar los datos mejor. Aumenta el valor de Jitter, ¿qué efecto tiene?

## 5. Clasificación

### 5.1. Clasificador ZeroR

Carga el conjunto de datos iris.arff. Selecciona el clasificador ZeroR y Use trainning set.

- ¿Qué modelo genera el clasificador ZeroR?
- ¿Cuántas instancias del conjunto de entrenamiento clasifica bien?
- ¿Qué porcentaje de instancias clasifica bien?
- ¿Qué crees que indica la matriz de confusión?

### 5.2. Clasificador J48

Carga el conjunto de datos iris.arff. Selecciona el clasificador J48 y Use trainning set.

- ¿Cuántas hojas tiene el árbol generado con J48?
- ¿Cuántas instancias del conjunto de entrenamiento clasifica bien?
- ¿Qué porcentaje de instancias clasifica bien?
- Pulsar el botón de *More Options* y selecciona la opción *Output predictions*. ¿En qué instancias se ha equivocado?
- Obtén el gráfico correspondiente al árbol generado.
- ¿Cómo podrías reducir el tamaño de este árbol en caso de que fuese necesario?

### 5.3. Clasificador ID3

Carga el conjunto de datos iris.arff. Selecciona el clasificador ID3 y utilízalo para generar un árbol de decisión.

- ¿Has podido ejecutar el algoritmo ID3 sobre el conjunto de datos directamente? ¿Por qué?
- ¿Qué acciones has llevado a cabo para poder ejecutarlo?
- ¿Qué porcentaje de éxito sobre el conjunto de entrenamiento has obtenido?
- ¿Qué porcentaje de éxito obtienes si utilizas como mecanismo de evaluación la validación cruzada?