PROGRAMA DE DOCTORADO INTERUNIVERSITARIO

# APRENDIZAJE AUTOMÁTICO Y DATA MINING

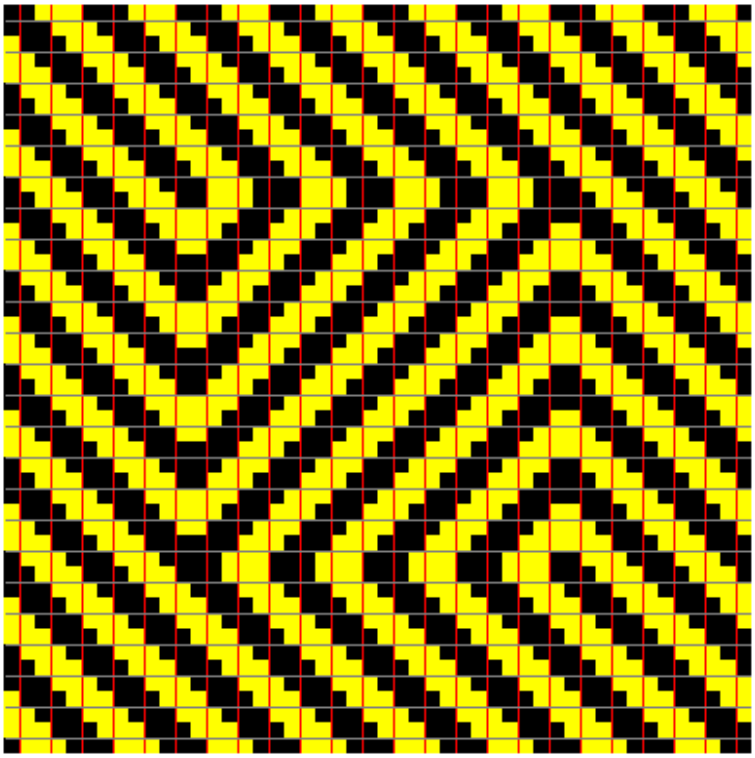
## Práctica 2:
## Utilización de WEKA desde la línea de comandos.

---

***Objetivos:***

- Utilización de WEKA desde la línea de comandos.
- Modificación de parámetros de los clasificadores
- Lectura automática de resultados.
- Lanzamiento de WEKA dentro de un bucle.

---



## 1. INTRODUCCIÓN

En la práctica anterior se utilizó WEKA desde uno de sus entornos gráficos.

En ocasiones es más práctico utilizar los algoritmos de WEKA desde la línea de comandos. Por ejemplo, si se quiere lanzar muchas veces un clasificador para obtener estadísticas de resultados, se puede automatizar el proceso mediante un bucle.

## 2. LANZAMIENTO DE UN COMANDO WEKA DESDE MS-DOS

Como ejemplo, se generará desde MS-DOS un árbol de decisión a partir de un fichero de datos de entrenamiento propio de WEKA.

Para ello, en primer lugar abriremos una ventana de comandos de MS-DOS.

Dentro de esa ventana, nos situaremos en el directorio donde se encuentra instalado WEKA. En la mayor parte de los ordenadores, este directorio será:

```
c:\iarp\weka_java
```

Una vez en el directorio correcto, se tecleará:

```
java weka.classifiers.trees.J48 –t data\weather.arff –C 0.30
```

El resultado que aparecerá en pantalla será el siguiente:

```
Options: -C 0.30

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

outlook = sunny
|   humidity <= 75: yes (2.0)
|   humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  :     5

Size of the tree :      8

Time taken to build model: 0.02 seconds
Time taken to test model on training data: 0 seconds
```

```
=== Error on training data ===

Correctly Classified Instances          14               100 %
Incorrectly Classified Instances         0                 0 %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               14

=== Confusion Matrix ===

 a b   <-- classified as
 9 0 | a = yes
 0 5 | b = no

=== Stratified cross-validation ===

Correctly Classified Instances           9               64.2857 %
Incorrectly Classified Instances         5               35.7143 %
Kappa statistic                          0.186
Mean absolute error                      0.2857
Root mean squared error                  0.4818
Relative absolute error                 60      %
Root relative squared error             97.6586 %
Total Number of Instances               14

=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 3 2 | b = no
```

Se trata el mismo resultado que se obtuvo en la práctica anterior desde el entorno gráfico.

Veamos en detalle la instrucción tecleada:

**`java`**
(lanza el intérprete java)

**`weka.classifiers.trees.J48`**
(elige un clasificador de entre todos los disponibles, en este caso el generador de árboles de decisión J48)

**`–t data\weather.arff`**
(indica cuál es el fichero de datos de entrenamiento a utilizar).

**`–C 0.30`**
(fija un parámetro del clasificador. En este caso es el umbral de confianza para la poda de los árboles).

La forma de lanzar cualquier otro clasificador o cualquier otra función de WEKA es similar. Para determinar el tipo de funciones disponibles, basta con comprobar los distintos directorios que existen a partir de **`weka_java\weka\:`**

- **Associations:** algoritmos para generar reglas a partir de datos.
- **AttributeSelection:** selección de características.
- **Classifiers:** clasificadores.
- **Clusterers:** agrupación de datos.
- **Core:** funciones núcleo de WEKA (no se usan directamente).
- **Datagenerators:** generación automática de datos (aleatorios).
- **Estimators:** estimadores estadísticos.
- **Experiment:** interfaz de usuario.
- **Filters:** filtros de datos y de atributos.
- **Gui:** interfaz de usuario.

Dentro de cada directorio existen múltiples subdirectorios, por lo tanto la cantidad de algoritmos disponible en WEKA es muy elevada.

## 3. LANZAMIENTO DE UN COMANDO WEKA DESDE MATLAB

Es práctico lanzar los comandos de WEKA desde Matlab para poder incluir bucles y funciones de lectura automática de resultados fácilmente.

Para lanzar WEKA desde Matlab se realiza una llamada al sistema (a MS-DOS) mediante el comando **!**:

Desde la ventana de comandos de Matlab, nos situaremos en el directorio correcto mediante el comando cd:

```
>> cd 'c:\iarp\weka_java'
```

Una vez en el directorio correcto, teclearemos:

```
>> !java weka.classifiers.trees.J48 –t data\weather.arff –C 0.30
```

Y obtendremos el mismo resultado que el obtenido sobre la ventana de MS-DOS. Una ventaja adicional de utilizar Matlab es que el comando se puede modificar a voluntad desde dentro del programa, mediante la instrucción **`eval`**.

Por ejemplo, para generar un árbol de decisión con un umbral de confianza ajustable para la poda podemos teclear las siguientes instrucciones de Matlab:

```
>> conf = 0.30;
>> orden = sprintf
 ('!java weka.classifiers.trees.J48 –t data/weather.arff –C %f', conf);
>> eval(orden);
```

El resultado será similar al obtenido en el resto de pruebas

Por último, será posible realizar un bucle para distintos valores del parámetro del clasificador, de la forma siguiente:

```
>> cf = [0.05 0.10 0.15 0.20 0.25 0.30];
>> for i=1:6
>> orden = sprintf
('!java weka.classifiers.trees.J48 –t data/weather.arff –C %f', cf(i));
>> eval(orden);
>> end;
```

## 4. LECTURA AUTOMÁTICA DE RESULTADOS DE WEKA

Comprobar los resultados de WEKA sobre la pantalla no es práctico cuando se realizan múltiples experimentos dentro de un bucle. Desde Matlab es posible leer automáticamente los resultados, siempre que éstos se hayan guardado anteriormente en un fichero.

Para guardar los resultados en un fichero, basta con utilizar el operador de redirección de MS-DOS (símbolo >) que hace que los datos no se muestren en pantalla sino que se escriban en el fichero que se indique.

Desde Matlab, bastaría con teclear un comando como el siguiente:

```
>> !java weka.classifiers.trees.J48 –t data\weather.arff > out.txt
```

La redirección hace que los resultados se guarden en el fichero ‘out.txt’ y que no aparezcan en pantalla. Para comprobar que todo ha funcionado correctamente, se buscará el fichero anterior desde el explorador de Windows y se abrirá con el programa Wordpad o EditPlus (el programa Notepad puede mostrar incorrectamente los saltos de línea).

El último paso consiste en extraer automáticamente la información deseada del fichero de resultados. Supongamos que los datos que nos interesan son los porcentajes de clasificaciones correctas tanto sobre los datos de entrenamiento como en un experimento de validación cruzada. Tales datos aparecen en el siguiente punto de los resultados:

1. Porcentaje de clasificaciones correctas sobre los datos de entrenamiento:

```
=== Error on training data ===

Correctly Classified Instances          14               100 %
```

2. Porcentaje de clasificaciones correctas en un experimento de validación cruzada:

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

Correctly Classified Instances           9               64.2857 %
```

Podemos escribir un programa Matlab que lea el fichero de resultados y busque precisamente esa información. Este programa se incluye como dato de la práctica. El programa es el siguiente:

```
% lee resultados de WEKA
function [porcent1, porcent2] = lee_weka (fichero)

% abre fichero de resultados
file = fopen(fichero, 'r');

% busca primer dato
cadena = busca_comienzo('=== Error on training data ===', file);
cadena = busca_comienzo('Correctly Classified Instances', file);
datos = sscanf(cadena(31:length(cadena)), '%f');
porcent1 = datos(2);

% busca segundo dato
cadena = busca_comienzo('=== Stratified cross-validation ===', file);
cadena = busca_comienzo('Correctly Classified Instances', file);
datos = sscanf(cadena(31:length(cadena)), '%f');
porcent2 = datos(2);

fclose(file);

return


% busca una cadena que comienza por unos ciertos caracteres
function cadena = busca_comienzo(comienzo, file);

carac = length(comienzo);
seguir=1;
while seguir==1
   cadena = fgets(file);
   if (length(cadena)>=carac)
      if cadena(1:carac)==comienzo
         seguir=0;
      end
   end
end

return
```

Y está disponible en la página web de la asignatura para evitar teclearlo:

http://isa.umh.es/isa/es/asignaturas/aprendizaje/lee_weka.m

La forma de utilizar el programa será la siguiente:

- En primer lugar, el programa se debe copiar en el directorio de la asignatura: C:\iarp\weka_java

- En Segundo lugar, se lanzará el programa desde Matlab de la forma siguiente:

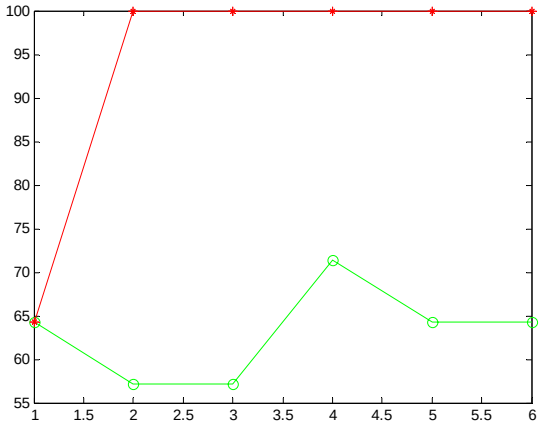
```
>> [porcent1, porcent2] = lee_weka('out.txt')

porcent1 = 100

porcent2 = 64.2857
```

Una vez que somos capaces de leer los resultados, es posible lanzar un clasificador con distintos valores para sus parámetros en un bucle y almacenar todos los resultados. Se probará el siguiente programa Matlab:

```
>> cf = [0.05 0.10 0.15 0.20 0.25 0.30];
>> for i=1:6
>> orden = sprintf
 ('!java  weka.classifiers.trees.J48  -t  data/weather.arff  -C  %f  >
out.txt', cf(i));
>> eval(orden);
>> [p1(i), p2(i)] = lee_weka('out.txt');
>> end;
```

Una vez ejecutado el programa anterior, los resultados estarán disponibles en los vectores p1 y p2, y podrán ser mostrados, por ejemplo, mediante un comando plot:

```
>> plot(p1, 'r-*');
>> plot(p2, 'g-o');
```



## A ENTREGAR: EJERCICIO NÚMERO 1

**A) Recuperar los ficheros datos_1.arff, datos_2.arff y datos_3.arff de la página web de la asignatura:**

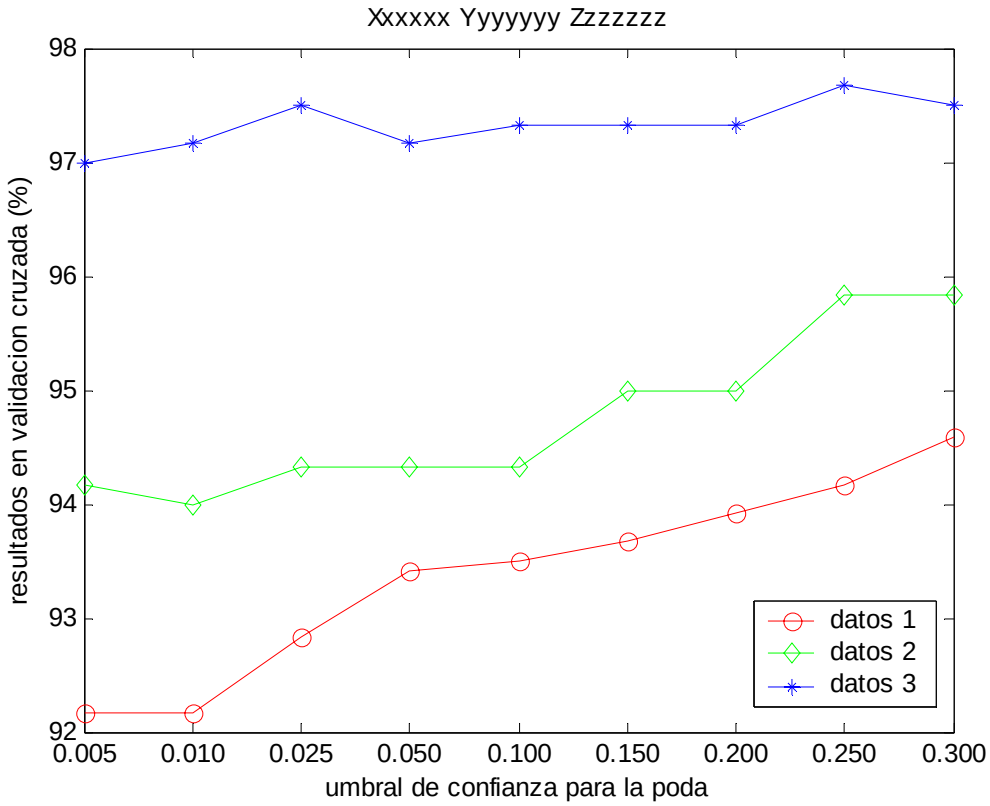
http://isa.umh.es/isa/es/ asignaturas/aprendizaje/datos_1.arff
http://isa.umh.es/isa/es/ asignaturas/aprendizaje/datos_2.arff
http://isa.umh.es/isa/es/ asignaturas/aprendizaje/datos_3.arff

**B) Lanzar el programa WEKA dentro de un bucle, generando árboles de decisión para cada uno de los ficheros de datos y para cada uno de los siguientes valores del parámetro de confianza para la poda;**

```
[0.005 0.010 0.025 0.050 0.10 0.15 0.20 0.25 0.30];
```

C) **Almacenar los resultados de clasificaciones correctas en un experimento de validación cruzada, mostrando posteriormente los resultados en un gráfico como el que se indica a continuación, donde Xxxxx Yyyyyy Zzzzzz se corresponden con el nombre y apellidos del alumno.**

**El resultado final debe tener un aspecto como el siguiente:**



**A ENTREGAR: EJERCICIO NÚMERO 2**

**A) Crear un programa lee_weka_plus.m tomando como base el programa lee_weka.m que, además de leer los valores de porcentajes de aciertos correctos, sea capaz de leer también el tamaño del árbol de decisión (el número de nodos).Tal información se encuentra en el siguiente punto de los resultados:**

```
Size of the tree :      8
```

B) **Lanzar de nuevo el programa WEKA dentro de un bucle, con los mismos ficheros de datos del ejercicio anterior, pero recopilando en este caso los resultados de número de nodos del árbol. El resultado se debe nombrar en un gráfico como el que se indica a continuación, donde Xxxxx Yyyyyy Zzzzzz se corresponden con el nombre y apellidos del alumno.**

**El resultado final debe tener un aspecto como el siguiente:**