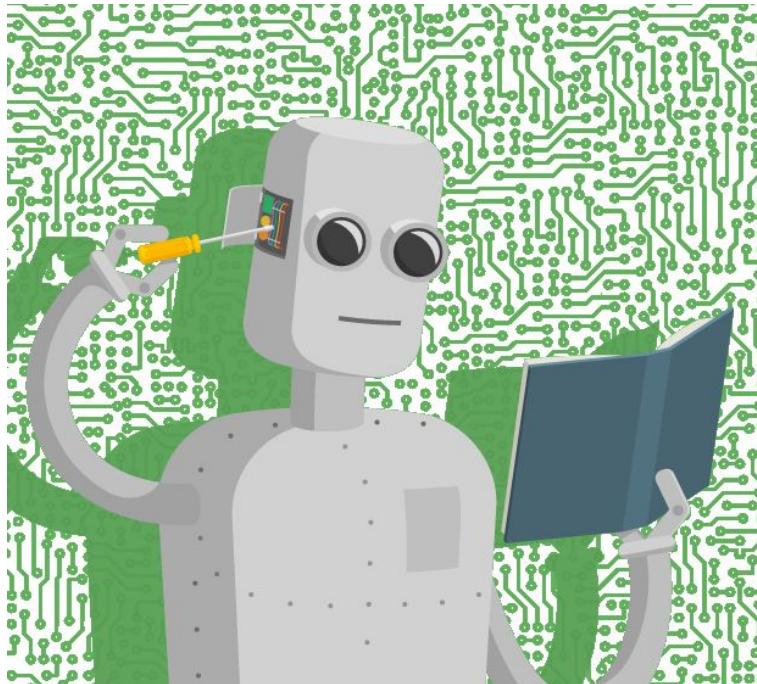




# Introducción al Aprendizaje Automático: prácticas



Base de datos usada:  
**‘vehicle’**

Víctor Monserrat Villatoro  
Curso académico 2015-2016



Práctica 1: Introducción a Weka.....	Página 4.
Ejercicio 1.....	Página 4.
Ejercicio 2.....	Página 7.
Ejercicio 3.....	Página 11.
Ejercicio 4.....	Página 14.
Ejercicio 5.....	Página 16.
Ejercicio 6.....	Página 19.
Ejercicio 7.....	Página 20.
Ejercicio 8.....	Página 20.
Ejercicio 9.....	Página 20.
Ejercicio 10.....	Página 21.
Práctica 2: Regresión con Weka.....	Página 22.
Algoritmo IB1.....	Página 22.
Ejercicio 1.....	Página 22.
Algoritmo IBK.....	Página 24.
Ejercicio 1.....	Página 24.
Ejercicio 2.....	Página 27.
Ejercicio 3.....	Página 31.
Ejercicio 4.....	Página 31.
Regresión.....	Página 32.
Ejercicio 1.....	Página 32.
Ejercicio 2.....	Página 33.
Ejercicio 3.....	Página 34.
Ejercicio 4.....	Página 34.
Ejercicio 5.....	Página 34.
Ejercicio 6.....	Página 35.
Ejercicio 7.....	Página 37.
Ejercicio 8.....	Página 39.
Regresión logística.....	Página 40.
Ejercicio 1.....	Página 40.
Ejercicio 2.....	Página 43.
Ejercicio 3.....	Página 45.
Ejercicio 4.....	Página 46.
Ejercicio 5.....	Página 47.



---

Práctica 3: Clasificación con Weka.....	Página 50.
Árboles de decisión.....	Página 50.
Ejercicio 1.....	Página 50.
Ejercicio 2.....	Página 52.
Ejercicio 3.....	Página 53.
Ejercicio 4.....	Página 54.
Ejercicio 5.....	Página 54.
Redes neuronales.....	Página 55.
Ejercicio 1.....	Página 55.
Ejercicio 2.....	Página 57.
Ejercicio 3.....	Página 58.
Ejercicio 4.....	Página 59.
Ejercicio 5.....	Página 60.
Ejercicio 6.....	Página 61.
Práctica 4: Clustering con Weka.....	Página 62.
Algoritmo K-means.....	Página 62.
Ejercicio 1.....	Página 62.
Ejercicio 2.....	Página 65.
Ejercicio 3.....	Página 68.
Ejercicio 4.....	Página 70.
Ejercicio 5.....	Página 70.
Ejercicio 6.....	Página 70.
Ejercicio 7.....	Página 71.
Clustering jerárquico.....	Página 74.
Ejercicio 1.....	Página 74.
Ejercicio 2.....	Página 75.
Algoritmo DBScan.....	Página 78.
Ejercicio 1.....	Página 78.
Ejercicio 2.....	Página 79.
Ejercicio 3.....	Página 79.
Práctica 5: Redes Neuronales Evolutivas (NNEP).....	Página 82.
Ejercicio 1.....	Página 84.
Ejercicio 2.....	Página 86.
Ejercicio 3.....	Página 87.
Ejercicio 4.....	Página 90.



## 1. Práctica 1: Introducción a Weka.

### 1.1. Ejercicio 1.

*Preparar el fichero .arff de la base de datos con la que vas a trabajar.  
Recuerda comprobar que está todo correcto cargándolo con Weka.*

La base de datos con la que se va a trabajar es ‘vehicle’. Para preparar el fichero ‘vehicle.arff’ de la base de datos debemos reescribir el nombre de los atributos, de la clase y de las subclases de esta.

Actualmente los nombres de los atributos son ‘x0’, ‘x1’, ‘x2’, ‘x3’, ‘x4’, ‘x5’, ‘x6’, ‘x7’, ‘x8’, ‘x9’, ‘x10’, ‘x11’, ‘x12’, ‘x13’, ‘x14’, ‘x15’, ‘x16’ y ‘x17’; el nombre de la clase es ‘Class’ y los nombres de las subclases son ‘y0’, ‘y1’, ‘y2’ e ‘y3’, como se puede observar en la siguiente imagen.

```
1 @relation vehicle
2 @attribute 'x0' numeric
3 @attribute 'x1' numeric
4 @attribute 'x2' numeric
5 @attribute 'x3' numeric
6 @attribute 'x4' numeric
7 @attribute 'x5' numeric
8 @attribute 'x6' numeric
9 @attribute 'x7' numeric
10 @attribute 'x8' numeric
11 @attribute 'x9' numeric
12 @attribute 'x10' numeric
13 @attribute 'x11' numeric
14 @attribute 'x12' numeric
15 @attribute 'x13' numeric
16 @attribute 'x14' numeric
17 @attribute 'x15' numeric
18 @attribute 'x16' numeric
19 @attribute 'x17' numeric
20 @attribute 'Class' {'y0','y1','y2','y3'}
21 @data
22 88.0,45.0,82.0,155.0,56.0,8.0,154.0,43.0,19.0,149.0,180.0,357.0,170.0,69.0,3.0,0.0,188.0,193.0,'y2'
23 94.0,46.0,77.0,169.0,60.0,8.0,158.0,42.0,20.0,148.0,181.0,373.0,181.0,67.0,12.0,2.0,193.0,199.0,'y2'
24 91.0,46.0,78.0,148.0,61.0,9.0,147.0,45.0,19.0,152.0,168.0,323.0,199.0,70.0,13.0,11.0,189.0,200.0,'y0'
25 101.0,52.0,105.0,162.0,53.0,10.0,212.0,31.0,24.0,163.0,226.0,669.0,204.0,74.0,12.0,11.0,186.0,194.0,'y3'
26 85.0,45.0,80.0,154.0,64.0,9.0,147.0,45.0,19.0,148.0,169.0,324.0,174.0,71.0,1.0,4.0,188.0,199.0,'y0'
27 85.0,45.0,70.0,130.0,58.0,8.0,151.0,45.0,19.0,146.0,171.0,334.0,187.0,79.0,2.0,5.0,181.0,186.0,'y1'
28 88.0,46.0,74.0,171.0,68.0,6.0,152.0,43.0,19.0,148.0,180.0,349.0,192.0,71.0,5.0,11.0,189.0,195.0,'y1'
29 90.0,37.0,80.0,171.0,58.0,9.0,157.0,42.0,20.0,132.0,172.0,373.0,115.0,60.0,3.0,18.0,201.0,209.0,'y2'
30 83.0,37.0,49.0,112.0,55.0,5.0,122.0,55.0,17.0,128.0,144.0,219.0,146.0,85.0,8.0,16.0,180.0,184.0,'y0'
31 95.0,45.0,105.0,208.0,64.0,10.0,187.0,36.0,22.0,150.0,202.0,520.0,158.0,64.0,7.0,32.0,198.0,211.0,'y2'
```

**Imagen 1.** Fichero ‘vehicle.arff’ original.



Para preparar el fichero debemos intercambiar estos nombres por identificadores que proporcionen información de estos atributos y clases.

Para ello leemos el fichero de información de la base de datos, ‘vehicle.txt’ y reemplazamos los valores de los atributos por ‘compactness’, ‘circularity’, ‘distance\_circularity’, ‘radius\_ratio’, ‘pr\_axis\_aspect\_ratio’, ‘max\_length\_aspect\_ratio’, ‘scatter\_ratio’, ‘elongatedness’, ‘pr\_axis\_rectangularity’, ‘max\_length\_rectangularity’, ‘scaled\_major\_variance’, ‘scaled\_minor\_variance’, ‘scaled\_radius\_of\_gyration’, ‘major\_skewness’, ‘minor\_skewness’, ‘minor\_kurtosis’, ‘major\_kurtosis’ y ‘hollows\_ratio’; el nombre de la clase por ‘Vehicle’ y el nombre de las subclases por ‘Van’, ‘Bus’, ‘Saab’ y ‘Opel’ respectivamente. Abajo, en la imagen, podemos ver el fichero ya preparado.

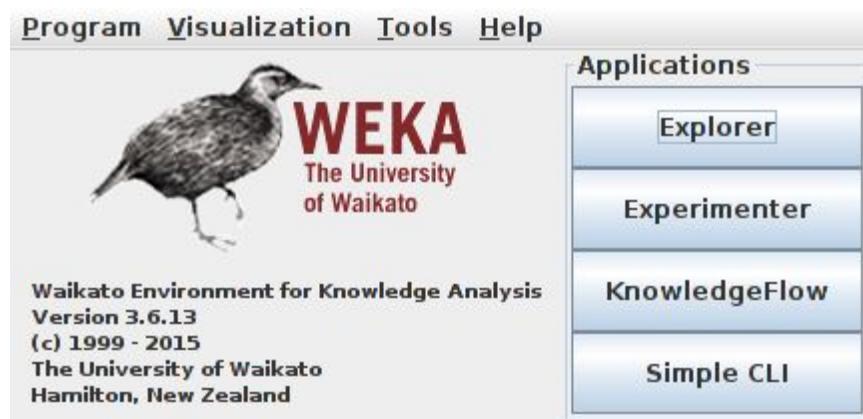
```
1 @relation vehicle
2 @attribute 'compactness' numeric
3 @attribute 'circularity' numeric
4 @attribute 'distance_circularity' numeric
5 @attribute 'radius_ratio' numeric
6 @attribute 'pr_axis_aspect_ratio' numeric
7 @attribute 'max_length_aspect_ratio' numeric
8 @attribute 'scatter_ratio' numeric
9 @attribute 'elongatedness' numeric
10 @attribute 'pr_axis_rectangularity' numeric
11 @attribute 'max_length_rectangularity' numeric
12 @attribute 'scaled_major_variance' numeric
13 @attribute 'scaled_minor_variance' numeric
14 @attribute 'scaled_radius_of_gyration' numeric
15 @attribute 'major_skewness' numeric
16 @attribute 'minor_skewness' numeric
17 @attribute 'minor_kurtosis' numeric
18 @attribute 'major_kurtosis' numeric
19 @attribute 'hollows_ratio' numeric
20 @attribute 'Vehicle' {'Van', 'Bus', 'Saab', 'Opel'}
21 @data
22 88.0,45.0,82.0,155.0,56.0,8.0,0,154.0,43.0,19.0,149.0,180.0,357.0,170.0,69.0,3.0,0,0,0,188.0,193.0, 'Saab'
23 94.0,46.0,77.0,169.0,60.0,8.0,0,158.0,42.0,20.0,148.0,181.0,373.0,181.0,67.0,12.0,2.0,193.0,199.0, 'Saab'
24 91.0,46.0,78.0,148.0,61.0,9.0,0,147.0,45.0,19.0,152.0,168.0,323.0,199.0,70.0,13.0,11.0,189.0,200.0, 'Van'
25 101.0,52.0,105.0,162.0,53.0,10.0,212.0,31.0,24.0,163.0,226.0,669.0,204.0,74.0,12.0,11.0,186.0,194.0, 'Opel'
26 85.0,45.0,80.0,154.0,64.0,9.0,0,147.0,45.0,19.0,148.0,169.0,324.0,174.0,71.0,1.0,4.0,188.0,199.0, 'Van'
27 85.0,45.0,70.0,130.0,58.0,8.0,0,151.0,45.0,19.0,146.0,171.0,334.0,187.0,79.0,2.0,5.0,181.0,186.0, 'Bus'
28 88.0,46.0,74.0,171.0,68.0,6.0,0,152.0,43.0,19.0,148.0,180.0,349.0,192.0,71.0,5.0,11.0,189.0,195.0, 'Bus'
29 90.0,37.0,80.0,0,171.0,58.0,9.0,0,157.0,42.0,20.0,132.0,172.0,373.0,115.0,60.0,3.0,18.0,201.0,209.0, 'Saab'
30 83.0,37.0,49.0,112.0,55.0,5.0,122.0,55.0,17.0,128.0,144.0,219.0,146.0,85.0,8.0,0,16.0,180.0,184.0, 'Van'
31 95.0,45.0,105.0,208.0,64.0,10.0,0,187.0,36.0,22.0,150.0,202.0,520.0,158.0,64.0,7.0,32.0,198.0,211.0, 'Saab'
```

**Imagen 2.** Fichero ‘vehicle.arff’ preparado.

Una vez preparado el fichero de la base de datos seleccionada, debemos cargarlo con Weka para comprobar que todo esté correcto.

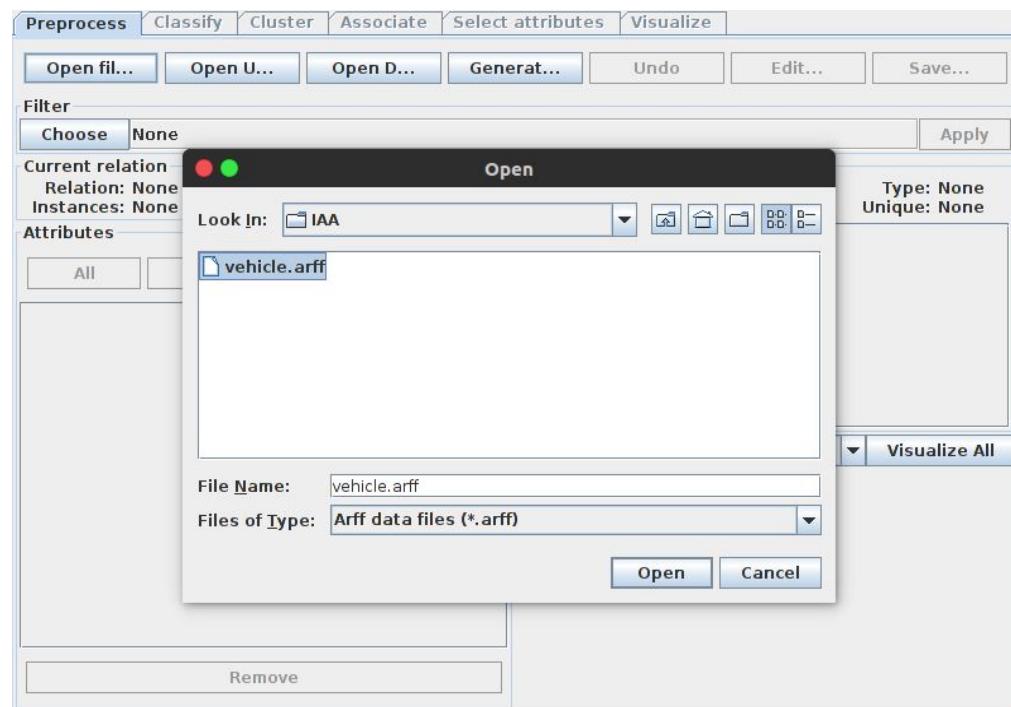


Para ello, abrimos Weka, seleccionamos la aplicación ‘Explorer’,



**Imagen 3.** Interfaz de Weka.

y abrimos el fichero preparado de la base de datos.



**Imagen 4.** Abriendo el fichero ‘vehicle.arff’ con Weka Explorer.



Si todo está correcto, se abrirá correctamente y el ‘Status’ será ‘OK’. Por el contrario, si existe algún error, Weka nos advertirá de este.

### 1.2. Ejercicio 2.

Desde el entorno Preprocess, abre la base de datos. Describe este fichero de datos, con la mayor precisión posible, a partir de la información que visualizas en este entorno.

Abrimos el entorno Preprocess y observamos la siguiente interfaz.

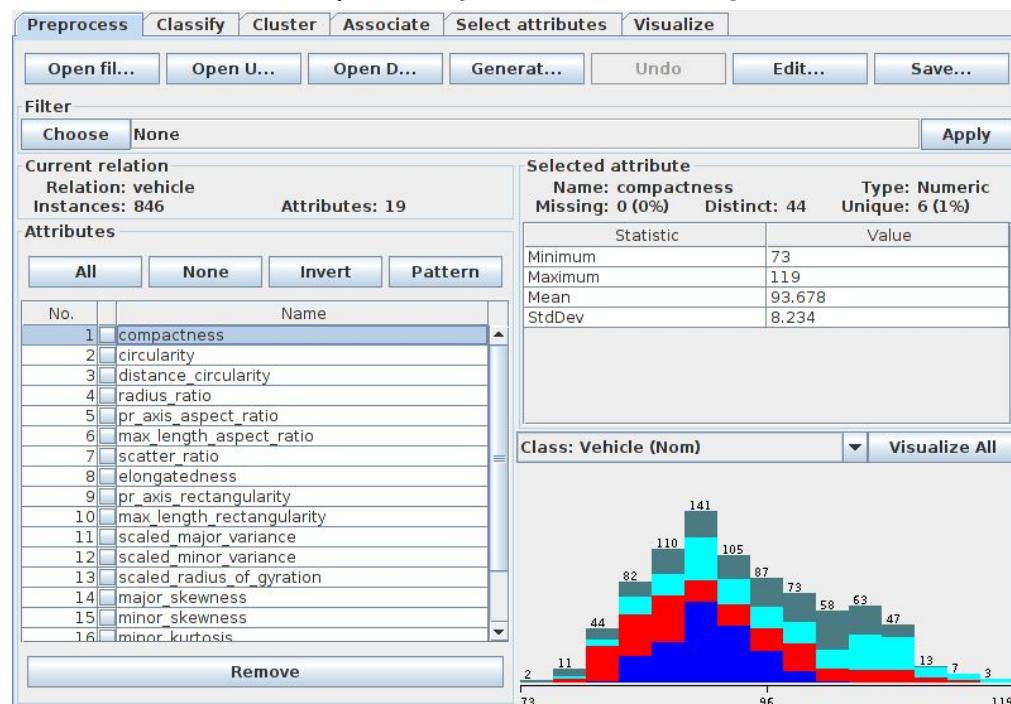


Imagen 5. Entorno Preprocess de Weka.

En el apartado ‘Current relation’ podemos ver que la actual es ‘vehicle’ de la cual tenemos 846 instancias y 19 atributos, que realmente son 18 más el atributo de subclase, que es un atributo que nos indica a qué subclase pertenece la instancia.

A continuación podemos observar la sección ‘Attributes’ donde se muestran cada uno de los atributos de la clase ‘Vehicle’. Podemos seleccionar cualquiera de estos atributos y ver más información de este en el siguiente apartado, ‘Selected attribute’. En esta sección



podremos ver el nombre del atributo seleccionado, su tipo (numérico o nominal), si hay valores perdidos, el número de valores del atributo diferentes que hay y cuántos de estos valores son únicos.

Además, podemos observar algunas estadísticas del atributo, como su valor máximo, su valor mínimo, su media o su desviación típica. A continuación se representan las estadísticas de cada atributo a través de la siguiente tabla.

Atributo	Máximo	Mínimo	Media	Desviación típica
compactness	119	73	93.678	8.234
circularity	59	33	44.862	6.17
distance_circularity	112	40	82.089	15.772
radius_ratio	333	104	168.941	33.472
pr_axis_aspect_ratio	138	47	61.694	7.888
max_length_aspect_ratio	55	2	8.567	4.601
scatter_ratio	265	112	168.839	33.245
elongatedness	61	26	40.934	7.812
pr_axis_rectangularity	29	17	20.583	2.592
max_length_rectangularity	188	118	147.999	14.516
scaled_major_variance	320	130	188.625	31.395
scaled_minor_variance	1018	184	439.911	176.693
scaled_radius_of_gyration	268	109	174.703	32.546
major_skewness	135	59	72.462	7.487
minor_skewness	22	0	6.377	4.918
minor_kurtosis	41	0	12.599	8.931
major_kurtosis	206	176	188.933	6.164
hollows_ratio	214	181	195.689	7.487



Atributo	Tipo	Distintos	Únicos	Faltan
compactness	Númerico	44	6	0
circularity	Númerico	27	1	0
distance_circularity	Númerico	63	7	0
radius_ratio	Númerico	134	14	0
pr_axis_aspect_ratio	Númerico	37	7	0
max_length_aspect_ratio	Númerico	21	7	0
scatter_ratio	Númerico	131	18	0
elongatedness	Númerico	35	1	0
pr_axis_rectangularity	Númerico	13	1	0
max_length_rectangularity	Númerico	66	4	0
scaled_major_variance	Númerico	128	22	0
scaled_minor_variance	Númerico	424	218	0
scaled_radius_of_gyration	Númerico	143	21	0
major_skewness	Númerico	39	9	0
minor_skewness	Númerico	23	0	0
minor_kurtosis	Númerico	41	4	0
major_kurtosis	Númerico	30	1	0
hollows_ratio	Númerico	32	0	0



El atributo nominal de subclase tiene cuatro valores distintos, no tiene valores únicos (0%) y no tiene ningún valor desaparecido y sus valores son:

No.	Label	Count
1	Van	204
2	Bus	218
3	Saab	217
4	Opel	207

Por último, podemos ver una gráfica por colores que nos muestra el número de instancias de cada subclase que existe por cada valor que toma el atributo seleccionado. El color cyan representa la subclase 'Van', el rojo la subclase 'Bus', el celeste la subclase 'Saab' y el azul la subclase 'Opel'.

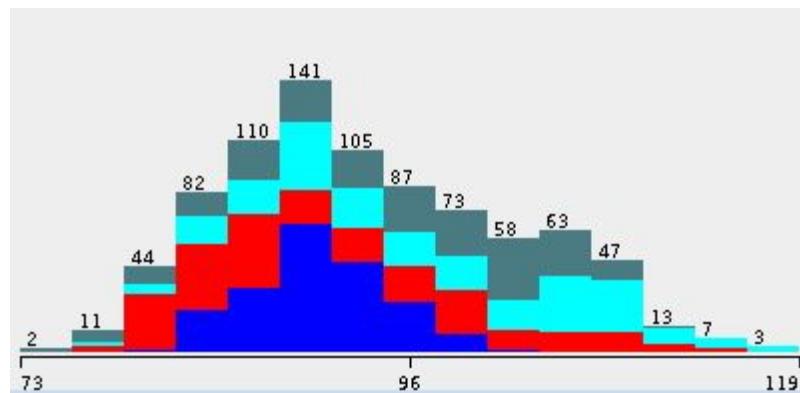


Imagen 6. Histograma de subclases por valores de 'compactness'.



### 1.3. Ejercicio 3.

*Incluye además, una descripción del problema tratado en la base de datos (puede hacer una búsqueda bibliográfica) y de los atributos de la misma.*

Las características fueron extraídas de las siluetas por el Sistema de Procesamiento Jerárquico de Imagen BINATTS, que extrae una combinación de características independientes de escala que utilizan medidas clásicas basadas en momentos tales como la varianza reducido, asimetría y curtosis de los ejes mayor/menor y medidas heurísticas tales como los huecos, la circularidad, la ortogonalidad y la compacidad.

Cuatro tipos de vehículos "Corgi" se utilizaron para el experimento. Estos vehículos fueron elegidos para que fuese fácil distinguir si se trataba de el autobús, la camioneta o uno de los coches pero más difícil distinguir entre los coches. Los cuatro vehículos fueron:

- Opel Manta 400.
- Saab 9000.
- Un autobús de dos pisos.
- Camioneta Chevrolet.

Las imágenes fueron adquiridas con una cámara encima del vehículo con un ángulo de elevación fijo (34.2 grados respecto a la horizontal). Los vehículos se colocaron en una superficie de iluminación de fondo difusa (caja de luz). Los vehículos fueron pintados en negro mate para minimizar los brillos. Las imágenes fueron capturadas utilizando una CRS4000 conectada a una VAX 750. Todas las imágenes fueron tomadas con una resolución de 128x128 píxeles con 64 niveles de gris. Estas imágenes fueron umbralizadas para obtener las siluetas binarias de los vehículos, negadas (para cumplir con los requisitos de procesamiento de BINATTS) y posteriormente sometidas a módulos de contracción y expansión del Sistema de Procesamiento Jerárquico de Imagen para eliminar el ruido de "sal y pimienta" de la imagen.



Los vehículos se rotaron y su ángulo de orientación se midió usando una retícula radial debajo del vehículo. Los 0 y 180 grados corresponden a la parte frontal y trasera, respectivamente, mientras que los 90 y 270 grados corresponden a los perfiles. Dos series de 60 imágenes, cubriendo los 360 grados, fueron capturados por cada vehículo. El vehículo fue girado un ángulo fijo entre imagen e imagen. Estos conjuntos de datos se conocen como e2 y e3 respectivamente.

Otros dos conjuntos de imágenes, e4 y e5, fueron capturados con la cámara a una altura de 37,5 y 30,8 grados respectivamente. Estos conjuntos también contienen 60 imágenes por vehículo exceptuando el conjunto e4 de la camioneta que contiene sólo 46 debido a la dificultad de enfocarla en la cámara en algunas posiciones. Los atributos elegidos han sido los siguientes:

- Compacidad. Un cuerpo compacto es aquel de textura apretada y poco porosa.

$$\text{Compacidad} = \frac{\overline{\text{perímetro}}^2}{\text{área}}$$

- Circularidad. Un cuerpo circular es aquel con forma de círculo.

$$\text{Circularidad} = \frac{\overline{\text{radio}}^2}{\text{área}}$$

- Distancia de circularidad.

$$\text{Distancia de circularidad} = \frac{\text{área}}{\overline{\text{distancia desde el borde}}^2}$$

- Relación del radio.

$$\text{Ratio del radio} = \frac{\text{máximo radio} - \text{mínimo radio}}{\text{radio}}$$

- Relación de aspecto.

$$\text{Relación de aspecto} = \frac{\text{eje menor}}{\text{eje mayor}}$$

- Relación de aspecto de máxima longitud.

$$\text{Relación de aspecto de máxima longitud} = \frac{\text{longitud} \perp \text{máxima longitud}}{\text{máxima longitud}}$$

- Relación de dispersión.

$$\text{Relación de dispersión} = \frac{\text{inerzia eje menor}}{\text{inerzia eje mayor}}$$



- Elongatedness. Relación de aspecto del rectángulo mínimo delimitador.

$$\text{Elongatedness} = \frac{\text{área}}{\text{anchura del mínimo rectángulo contenedor}^2}$$

- Rectangularidad de ejes.

$$\text{Rectangularidad de ejes} = \frac{\text{área}}{\text{eje mayor} \cdot \text{eje menor}}$$

- Rectangularidad de máxima longitud.

$$\text{Rectangularidad de máxima longitud} = \frac{\text{área}}{\text{máxima longitud} \cdot \text{longitud} \perp \text{máxima longitud}}$$

- Varianza mayor escalada.

$$\text{Varianza mayor escalada} = \frac{\text{segundo momento sobre el eje menor}}{\text{área sobre el eje mayor}}$$

- Varianza menor escalada.

$$\text{Varianza menor escalada} = \frac{\text{segundo momento sobre el eje mayor}}{\text{área sobre el eje menor}}$$

- Radio de giro escalado.

$$\text{Radio de giro escalado} = \frac{\text{mayor varianza} + \text{menor varianza}}{\text{área}}$$

- Mayor oblicuidad.

$$\text{Mayor oblicuidad} = \frac{\text{tercer momento sobre el eje mayor}}{\text{varianza sobre el eje menor}^{3/2}}$$

- Menor oblicuidad.

$$\text{Menor oblicuidad} = \frac{\text{tercer momento sobre el eje menor}}{\text{varianza sobre el eje mayor}^{3/2}}$$

- Menor curtosis.

$$\text{Menor curtosis} = \frac{\text{cuarto momento sobre el eje mayor}}{\text{varianza sobre el eje menor}^2}$$

- Mayor curtosis.

$$\text{Mayor curtosis} = \frac{\text{cuarto momento sobre el eje menor}}{\text{varianza sobre el eje mayor}^2}$$

- Relación de huecos.

$$\text{Relación de huecos} = \frac{\text{área del polígono que delimita el objeto} - \text{área del objeto}}{\text{área del polígono que delimita el objeto}}$$



#### 1.4. Ejercicio 4.

Visualiza la relación entre atributos. ¿Hay alguna relación que sea visualmente significativa?

Para visualizar la relación entre atributos debemos acceder al entorno ‘Visualize’.

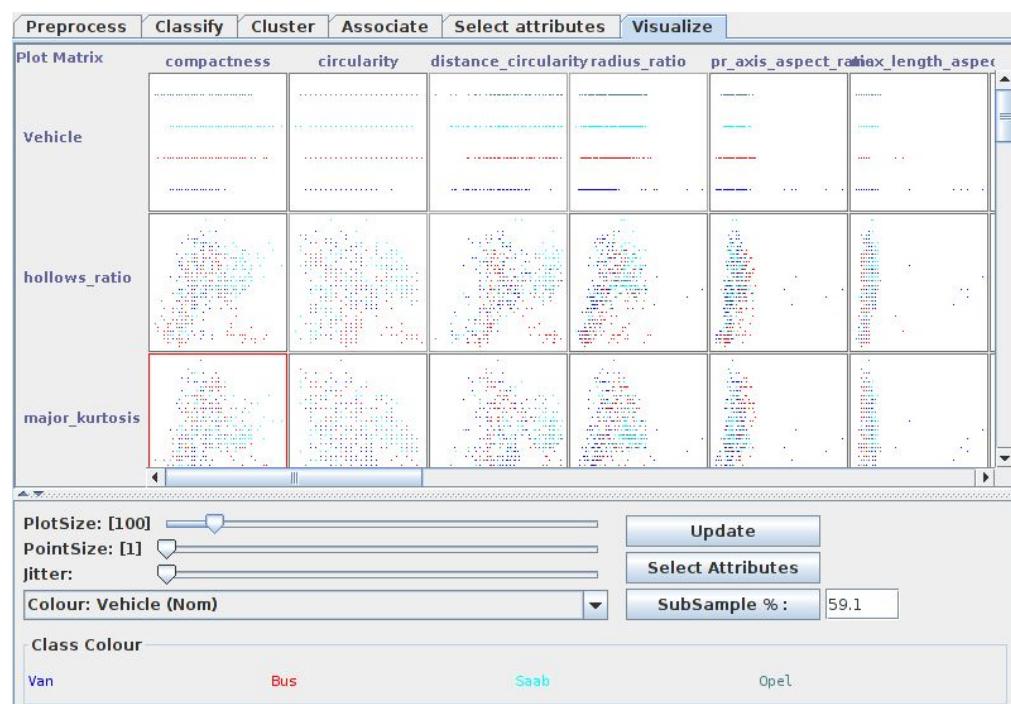


Imagen 7. Entorno ‘Visualize’ de Weka.

Podemos apreciar ahora que existen varios atributos muy relacionados entre sí. A continuación se muestran algunos ejemplos claros.

scaled\_minor\_variance con scatter\_ratio

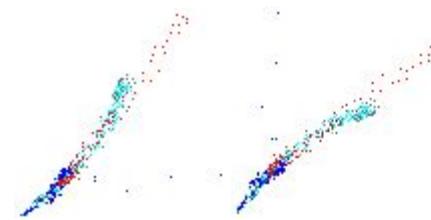




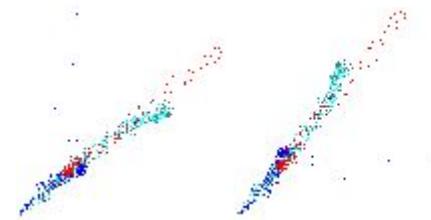
scaled\_minor\_variance con elongatedness



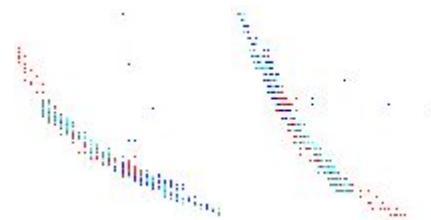
scaled\_minor\_variance con scaled\_mayor\_variance



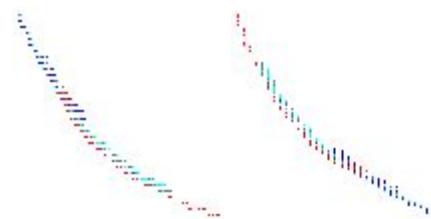
scaled\_mayor\_variance con scatter\_ratio



scaled\_mayor\_variance con elongatedness



elongatedness con scatter\_ratio



Por lo que se podría decir que estos atributos están muy relacionados entre sí, a veces de forma directa y otras de forma inversa.

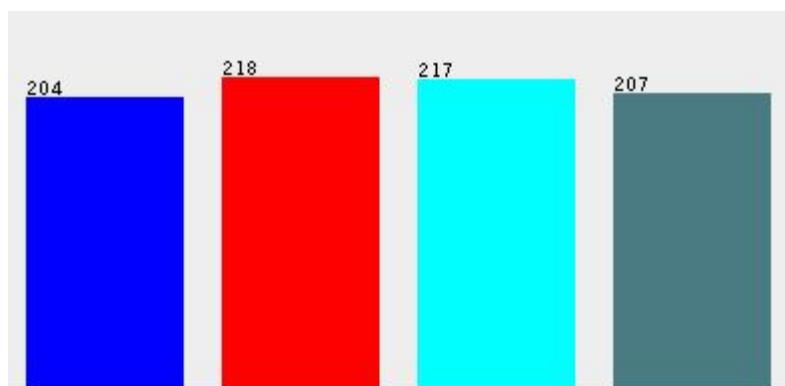


### 1.5. Ejercicio 5.

*Aplica cuatro filtros, uno de cada tipo (supervisado - no supervisado de atributos e instancias), comenta el resultado obtenido. Tras analizar los resultados, es necesario decidir si queremos conservar los cambios realizados por el filtro o no.*

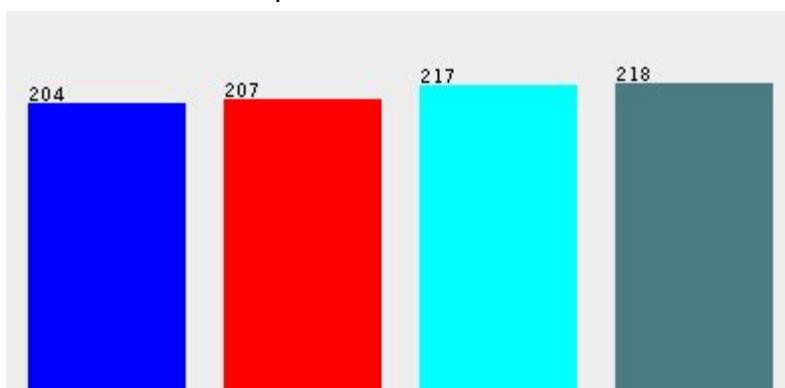
- Supervisado de atributos.

Si seleccionamos el atributo nominal de subclase ‘Vehicle’ podemos observar que el orden establecido se corresponde al de inserción.



**Imagen 8.** Histograma del atributo de subclase antes del filtro.

Aplicando el filtro ‘ClassOrder’ se puede observar que se ordenan las subclases por número de instancias de cada una.



**Imagen 9.** Histograma del atributo de subclase tras el filtro.



Podemos ver que se han ordenado de menor a mayor y no se han respetado los colores asignados a las subclases, sino que se han reasignado. Si queremos ordenarlas inversamente, de menor a mayor, cambiamos el parámetro -C 0 por -C 1.

No conservaremos los cambios porque de momento no necesitamos ordenar la base de datos, por lo que pulsamos ‘Undo’.

- Supervisado de instancias.

En este caso aplicaremos el filtro ‘Resample’ que produce una submuestra aleatoria de un conjunto de datos utilizando el muestreo con sustitución o sin sustitución.

Lo aplicaremos con un tamaño del 50% del conjunto de datos actual y observamos que el número de instancias se ha reducido efectivamente a la mitad.



**Imagen 10.** Filtro ‘Resample’.

Si observamos las medias y desviaciones no varían demasiado puesto que hemos reducido el número de instancias aleatoriamente y hemos seleccionado un subconjunto aleatorio, así que lo normal es que estas estadísticas se mantengan parecidas.

Este filtro podría ser útil para bases de datos demasiado extensas, sin embargo no es nuestro caso, por lo que no aplicaremos el filtro finalmente y volveremos a pulsar ‘Undo’.



● No supervisado de atributos.

El filtro ‘FirstOrder’ toma una serie de N atributos numéricos y los reemplaza con N-1, los valores de los cuales son la diferencia entre los valores de los atributos consecutivos a partir de la instancia original. Por ejemplo:

Valores originales: 0.1, 0.2, 0.3, 0.1, 0.3.

Nuevos valores: 0.1, 0.1, -0.2, 0.2.

Podría interesarnos de nuevo para reducir el tamaño de los valores de los atributos numéricos pero de nuevo no creo que sea suficientemente interesante, por lo que volveremos a deshacer el filtro.

● No supervisado de instancias.

Seleccionamos el filtro ‘Normalize’ que normaliza instancias teniendo en cuenta sólo los atributos numéricos y observamos como los valores de todos los atributos ahora se encuentran comprendidos entre 0 y 1 como se puede ver a continuación.

Statistic	Value
Minimum	0.09
Maximum	0.196
Mean	0.144
StdDev	0.022

**Imagen 10.** Algunos valores tras el filtro ‘Normalize’.

Este filtro podría sernos útil para trabajar con los valores de todos los atributos en relación a los otros y poder trabajar en la misma magnitud por lo que podríamos aplicar este filtro ya que puede sernos de utilidad en un futuro.



### 1.6. Ejercicio 6.

*Si lo consideras necesario, aplica un filtro de selección de características y/o selección de patrones sobre la base de datos original. Indica por qué has seleccionado ese filtro y cuáles son los resultados obtenidos.*

Para la selección de características, usaremos el filtro no supervisado de atributos ‘PrincipalComponents’ que realiza un análisis de los componentes principales y la transformación de los datos.

No.	Name
1	0.317scatter_ratio+0.314scaled_minor_varian...
2	0.541major_kurtosis+0.539hollows_ratio-0.49...
3	-0.643pr_axis_aspect_ratio-0.592max_length_a...
4	0.665minor_kurtosis-0.609minor_skewness-0.1...
5	0.725minor_skewness+0.604minor_kurtosis+0...
6	-0.465max_length_rectangularity-0.436max_len...
7	0.488max_length_aspect_ratio+0.464compact...
8	Vehicle

**Imagen 11.** Atributos tras el filtro ‘PrincipalComponents’.

Con este filtro reducimos los atributos de la base de datos a través de transformarlos en ecuaciones cuyas variables serán los atributos más importantes. Ahora comprobaremos si este filtro de selección de características mejoran la clasificación de nuestra base de datos.

La clasificación de la base de datos antes de aplicar el filtro a través del algoritmo IB1 con un 5-fold-cross-validation conseguía un porcentaje de acierto del 70%.

Correctly Classified Instances	594	70.2128 %
--------------------------------	-----	-----------

Mientras que, a través del mismo algoritmo e igual configuración, tras aplicar el filtro, el porcentaje de acierto baja a un 63% por lo que no usaremos este filtro.

Correctly Classified Instances	537	63.4752 %
--------------------------------	-----	-----------



Para la selección de patrones, usaremos el filtro supervisado de instancias 'Resample' que produce una submuestra aleatoria de un conjunto de datos utilizando el muestreo con sustitución o sin sustitución.

Lo aplicaremos con un tamaño del 50% del conjunto de datos actual y deshabilitaremos el reemplazamiento.

**Resample -B 0.0 -S 1 -Z 50.0 -no-replacement**

Observamos que el número de instancias se ha reducido efectivamente a la mitad y si observamos las medias y desviaciones no varían demasiado puesto que hemos seleccionado un subconjunto aleatorio, así que lo normal es que estas estadísticas se mantengan parecidas.

Current relation  
Relation: vehicle-weka.filters.supervised.instan...  
Instances: 423 Attributes: 19

La clasificación de la base de datos antes de aplicar el filtro a través del algoritmo IB1 con un 5-fold-cross-validation conseguía un porcentaje de acierto del 70%.

Correctly Classified Instances	594	70.2128 %
--------------------------------	-----	-----------

Mientras que, a través del mismo algoritmo e igual configuración, tras aplicar el filtro, el porcentaje de acierto baja a un 65% por lo que tampoco usaremos este filtro.

Correctly Classified Instances	276	65.2482 %
--------------------------------	-----	-----------

En casos en los que nuestra base de datos sea excesivamente grande podría interesarnos este filtro porque reduciremos bastante el tamaño de la base de datos sin tener una gran pérdida de información.

En nuestro caso lo desechamos puesto que el tamaño de nuestra base de datos no es excesivamente grande.



1.7. Ejercicio 7.

*¿Existen valores perdidos en la base de datos? Analiza en qué porcentaje y en qué variables y elimínalos.*

El número de valores perdidos en la base de datos es 0 (0%), ninguna variable posee valores perdidos y para eliminarlos habría que aplicar el filtro ‘ReplaceMissingValues’ que reemplazará los valores perdidos por la media y la moda del conjunto de entrenamiento.

1.8. Ejercicio 8.

*Convierte todos los atributos nominales a codificación binaria. Emplea la base de datos sin valores perdidos obtenida en el paso previo.*

No existen atributos nominales en esta base de datos, sólo numéricos. Para convertir todos los atributos nominales a codificación binaria es necesario aplicar el filtro ‘NominalToBinary’.

1.9. Ejercicio 9.

*Elimina los atributos identificadores (en caso de existir). Emplea la base de datos del paso previo.*

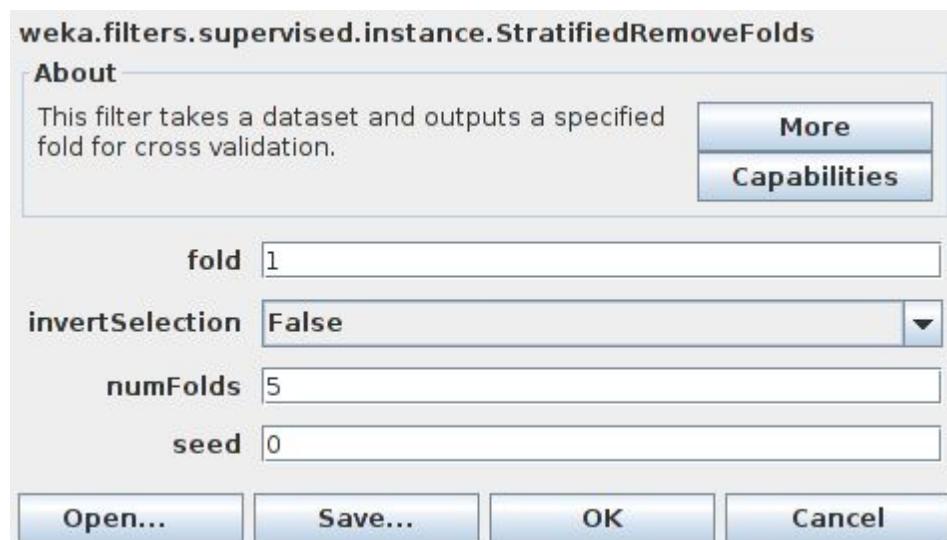
No existen atributos identificadores en esta base de datos. Para eliminar todos los atributos identificadores es necesario aplicar el filtro ‘RemoveUseless’.



### 1.10. Ejercicio 10.

Divide la base de datos obtenida en el paso anterior en 5 conjuntos train-test.

Para dividir la base de datos en 5 conjuntos train-test debemos emplear el filtro supervisado de instancias ‘StratifiedRemoveFolds’. Para ello seleccionamos el filtro y cambiamos el valor de ‘numFolds’ al número de conjuntos deseados, en este caso, 5.



**Imagen 12.** Parámetros del filtro ‘StratifiedRemoveFolds’.

Podemos observar ahora como el número de instancias se ha reducido a 170 (20% del total) porque hemos dividido la base de datos en 5 conjuntos y nos quedamos con sólo uno de ellos.



**Imagen 13.** Número de instancias tras ‘StratifiedRemoveFolds’.



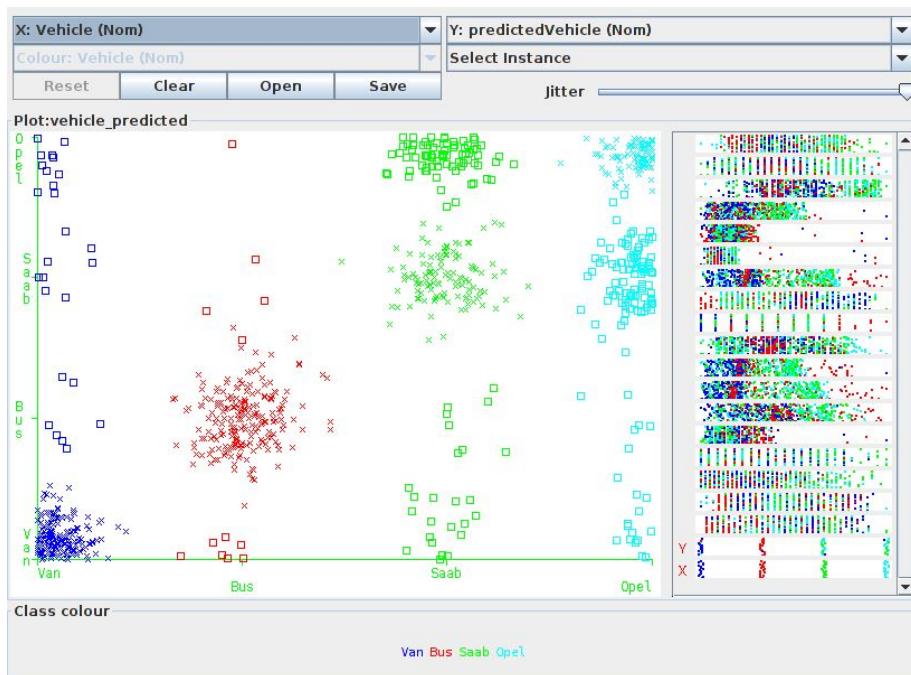
## 2. Práctica 2: Regresión con Weka.

### 2.1. Algoritmo IB1.

#### 2.1.1. Ejercicio 1.

*En el entorno Explorer utiliza el algoritmo IB1 con un 10-fold cross-validation. Visualiza la clasificación realizada (gráfica y matriz de confusión). Incluye ambas en la memoria, interpretando los resultados.*

Seleccionamos el algoritmo IB1 con un 10-fold cross-validation y visualizamos la gráfica de la clasificación realizada que se muestra a continuación.



**Imagen 14.** Gráfica de la clasificación tras ‘IB1’.



Ahora mostraremos la matriz de confusión.

==== Confusion Matrix ===

a	b	c	d	<- classified as
178	7	8	11	a = Van
7	206	4	1	b = Bus
15	10	117	75	c = Saab
14	6	95	92	d = Opel

**Imagen 15.** Matriz de confusión de la clasificación tras 'IB1'.

Podemos observar en la matriz de confusión como han sido clasificadas las instancias respecto a lo que son en realidad. En el eje x se muestra como se han clasificado y en el eje y lo que en realidad son. De esta manera, tenemos en la diagonal principal las clasificaciones realizadas correctamente y el resto de elementos de la matriz son clasificaciones incorrectas. Además podemos observar que en esta diagonal principal los valores son elevados, es decir, se consigue clasificar de manera correcta en muchas ocasiones.

Correctly Classified Instances	594	70.2128 %
Incorrectly Classified Instances	252	29.7872 %
Kappa statistic	0.6026	
Mean absolute error	0.1489	
Root mean squared error	0.3859	
Relative absolute error	39.7269 %	
Root relative squared error	89.1364 %	
Total Number of Instances	846	

En la salida del clasificador podemos ver que el 70.2128% de instancias han sido clasificadas correctamente y el valor del estadístico Kappa es de 0.6026, lo cual es bastante alto teniendo en cuenta que es un valor entre -1 y 1.

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.858	0.055	0.833	0.858	0.845	0.902	Van
	0.945	0.038	0.896	0.945	0.92	0.953	Bus
	0.548	0.173	0.522	0.548	0.535	0.688	Saab
	0.454	0.131	0.528	0.454	0.488	0.661	Opel
Weighted Avg.	0.702	0.1	0.695	0.702	0.697	0.801	

También podemos ver que la clase mejor clasificada es Bus.



## 2.2. Algoritmo IBK.

### 2.2.1. Ejercicio 1.

Ejecuta el algoritmo IBK-1 con un 5 fold (Experimenter o con los ficheros de la práctica anterior) para la base de datos elegida. Si se usa el Experimenter, fija el número de repeticiones a 1. Calcula la media y la desviación típica de las medidas: Accuracy, Kappa, RMSE y la media ponderada AUC.

Usaremos el entorno ‘Experimenter’ de Weka, creamos un nuevo experimento, exportamos el resultado a un archivo ‘experimenter.arff’, elegimos nuestra base de datos y el algoritmo que vamos a usar, cambiamos el número de ‘folds’ a 5 y el número de repeticiones a 1 y pulsamos ‘Run’ y ‘Start’ para ejecutarlo.

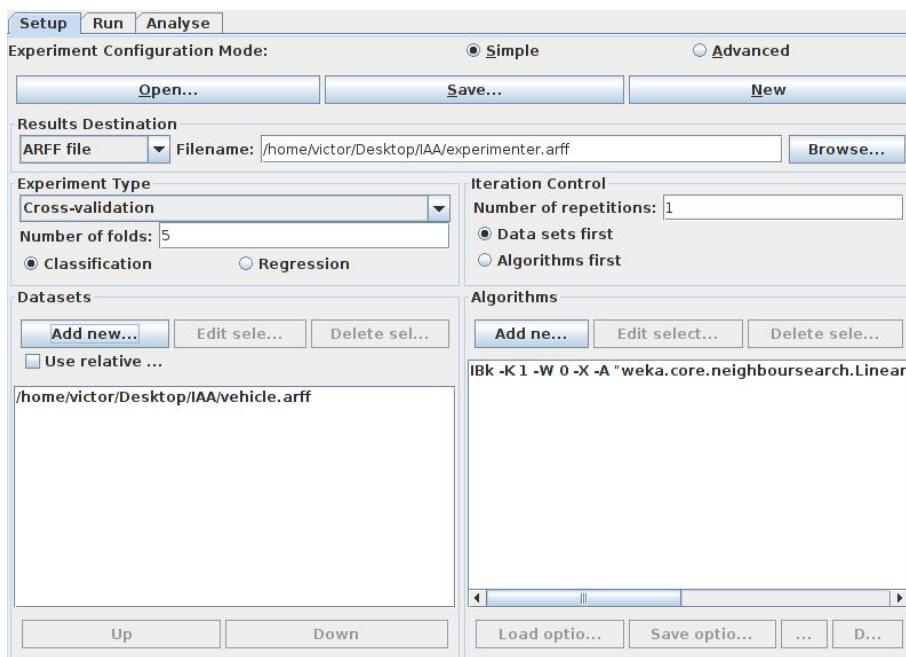


Imagen 15. Configuración para el experimento con ‘IB1’.



Una vez ejecutado, cargamos el fichero de salida ‘experimenter.arff’ con el entorno ‘Explorer’ y podremos ver los valores máximos, mínimos, la media y la desviación típica de los estadísticos del experimento con el algoritmo IBK-1.

De esta forma podremos ver fácilmente los estadísticos de los atributos deseados. A continuación se muestran los deseados.

- Accuracy.

Selected attribute	
Name: Percent_correct	Type: Numeric
Missing: 0 (0%)	Distinct: 4
Unique: 3 (60%)	
Statistic	Value
Minimum	67.456
Maximum	74.556
Mean	70.211
StdDev	3.026

- Kappa.

Selected attribute	
Name: Kappa_statistic	Type: Numeric
Missing: 0 (0%)	Distinct: 5
Unique: 5 (100%)	
Statistic	Value
Minimum	0.565
Maximum	0.66
Mean	0.603
StdDev	0.04

- RMSE.

Selected attribute	
Name: Root_mean_squared_e...	Type: Num...
Missing: 0 (0%)	Distinct: 4
Unique: 3 (60%)	
Statistic	Value
Minimum	0.356
Maximum	0.402
Mean	0.384
StdDev	0.02



- Media ponderada AUC.

Selected attribute		Type: Nu... Unique: 5 (...)
Name:	Weighted_avg_area_un...	
Missing:	0 (0%)	Distinct: 5
Statistic	Value	
Minimum	0.783	
Maximum	0.83	
Mean	0.801	
StdDev	0.02	



### 2.2.2. Ejercicio 2.

Ejecuta con la misma configuración IBK con  $k=3$ ,  $k=5$  y  $k=10$ . Para cada caso, incluye la media y la desviación típica de las medidas empleadas en el paso anterior.

Repetimos los pasos del ejercicio anterior pero esta vez exportamos un fichero para cada configuración.

- k=3.
  - Accuracy.

Selected attribute	
Name:	Percent_correct
Missing:	0 (0%)
Distinct:	5
Type:	Numeric
Unique:	5 (100%)
Statistic	Value
Minimum	64.497
Maximum	71.765
Mean	67.962
StdDev	3.455

- Kappa.

Selected attribute	
Name:	Kappa_statistic
Missing:	0 (0%)
Distinct:	5
Type:	Numeric
Unique:	5 (100%)
Statistic	Value
Minimum	0.526
Maximum	0.624
Mean	0.573
StdDev	0.046

- RMSE.

Selected attribute	
Name:	Root_mean_squared_e...
Missi...	0 (0%)
Distinct:	5
Type:	Num...
Unique:	5 (10...
Statistic	Value
Minimum	0.304
Maximum	0.375
Mean	0.342
StdDev	0.026



- Media ponderada AUC.

Selected attribute		
Name:	Weighted_avg_area_un...	Type: Nu...
Missing:	0 (0%)	Distinct: 5
		Unique: 5 (...)
Statistic	Value	
Minimum	0.812	
Maximum	0.89	
Mean	0.849	
StdDev	0.029	

- k=5.

- Accuracy.

Selected attribute		
Name:	Percent_correct	Type: Numeric
Missing:	0 (0%)	Distinct: 4
		Unique: 3 (60%)
Statistic	Value	
Minimum	65.68	
Maximum	75.148	
Mean	70.683	
StdDev	3.59	

- Kappa.

Selected attribute		
Name:	Kappa_statistic	Type: Numeric
Missing:	0 (0%)	Distinct: 5
		Unique: 5 (100%)
Statistic	Value	
Minimum	0.543	
Maximum	0.668	
Mean	0.609	
StdDev	0.048	

- RMSE.

Selected attribute		
Name:	Root_mean_squared_e...	Type: Num...
Missi...	0 (0%)	Distinct: 5
		Unique: 5 (10...)
Statistic	Value	
Minimum	0.289	
Maximum	0.338	
Mean	0.317	
StdDev	0.023	



- Media ponderada AUC.

Selected attribute		
Name:	Weighted_avg_area_un...	Type: Nu...
Missing:	0 (0%)	Distinct: 5
		Unique: 5 (...)
Statistic	Value	
Minimum	0.856	
Maximum	0.915	
Mean	0.88	
StdDev	0.029	

- k=10.
  - Accuracy.

Selected attribute		
Name:	Percent_correct	Type: Numeric
Missing:	0 (0%)	Distinct: 3
		Unique: 2 (40%)
Statistic	Value	
Minimum	69.822	
Maximum	75.148	
Mean	71.511	
StdDev	2.441	

- Kappa.

Selected attribute		
Name:	Kappa_statistic	Type: Numeric
Missing:	0 (0%)	Distinct: 5
		Unique: 5 (100%)
Statistic	Value	
Minimum	0.597	
Maximum	0.668	
Mean	0.62	
StdDev	0.033	

- RMSE.

Selected attribute		
Name:	Root_mean_squared_e...	Type: Num...
Missi...	0 (0%)	Distinct: 5
		Unique: 5 (10...)
Statistic	Value	
Minimum	0.289	
Maximum	0.332	
Mean	0.312	
StdDev	0.02	



**UNIVERSIDAD DE CÓRDOBA**  
**Escuela Politécnica Superior**

Ingeniería Informática  
Sistemas Interactivos  
Víctor Monserrat Villatoro  
i32moviv@uco.es



- 
- Media ponderada AUC.

Selected attribute		
Name:	Weighted_avg_area_un...	Type: Nu...
Missing:	0 (0%)	Distinct: 5
Statistic	Value	
Minimum	0.859	
Maximum	0.915	
Mean	0.886	
StdDev	0.026	



#### 2.2.3. Ejercicio 3.

Comenta los resultados obtenidos. Estos resultados serán usados en las siguientes prácticas para comparar con otros algoritmos.

Los resultados obtenidos son similares para todos los experimentos. Sin embargo, se puede apreciar una subida de la media de los estadísticos Accuracy, Kappa y de la media ponderada AUC, lo que indica una mejora del modelo a medida que se incrementa el número de vecindarios (K). Además podemos ver un decremento en la media del estadístico RMSE directamente proporcional al número de vecindarios, lo que también indica una mejora del modelo.

#### 2.2.4. Ejercicio 4.

Observa la media ponderada de la métrica Recall y el valor de Accuracy. ¿Qué ocurre? ¿Por qué? Justifica tu respuesta (preferiblemente de manera matemática).

	Media ponderada Recall	Accuracy
IBK-1	0.702	70.211
IBK-3	0.68	67.962
IBK-5	0.707	70.683
IBK-10	0.715	71.511

Como podemos ver son casi los mismos valores en diferente escala, la media ponderada Recall entre 0 y 1, Accuracy entre 0 y 100.



Accuracy es el porcentaje de todos los valores bien clasificados respecto a todos los valores, es decir, la suma de la diagonal principal de la matriz de confusión dividida por la suma de todos los valores de la matriz, o, expresado de forma matemática:

$$\text{Accuracy} = \frac{\sum_{i=1}^n a_{ii}}{\sum_{i=1}^n \sum_{j=1}^n a_{ij}}.$$

Siendo  $n$  la dimensión de la matriz.

Recall es el porcentaje de los positivos bien clasificados entre el total de patrones de la clase, es decir, el valor del elemento de misma fila y columna dividido por la suma de todos los valores de esa fila, o, expresado de forma matemática:

$$\text{Recall para la fila } i \text{ de la matriz} = \frac{a_{ii}}{\sum_{j=1}^n a_{ij}}.$$

Siendo  $n$  la dimensión de la matriz e  $i$  una fila de la matriz, que se corresponde con una clase.

La media ponderada Recall se calcula sumando el producto de cada métrica Recall por la suma de todos los valores de esa fila y dividiendo entre la suma de todos los valores de la matriz, o, expresado de forma matemática:

$$\text{Media ponderada Recall} = \frac{\sum_{i=1}^n \text{Recall}_i \cdot \sum_{j=1}^n a_{ij}}{\sum_{i=1}^n \sum_{j=1}^n a_{ij}}.$$

Siendo  $n$  la dimensión de la matriz.

Por lo que, se puede demostrar que:

$$\text{M.P. Recall} = \frac{\sum_{i=1}^n \text{Recall}_i \cdot \sum_{j=1}^n a_{ij}}{\sum_{i=1}^n \sum_{j=1}^n a_{ij}} = \frac{\sum_{i=1}^n \frac{a_{ii}}{\sum_{j=1}^n a_{ij}} \cdot \sum_{j=1}^n a_{ij}}{\sum_{i=1}^n \sum_{j=1}^n a_{ij}} = \frac{\sum_{i=1}^n a_{ii}}{\sum_{i=1}^n \sum_{j=1}^n a_{ij}} = \text{Accuracy}.$$



## 2.3. Regresión.

### 2.3.1. Ejercicio 1.

Observa y estudia los atributos de la base de datos *autoMpg.arff* (disponible en Moodle).

- Emplea diferentes métodos de regresión (*SimpleLinearRegression*, *LinearRegression* o cualquier otro).
- Utiliza un 80% de los patrones para entrenar y el 20% restante para generalizar.
- Comenta los resultados y los modelos obtenidos.

La base de datos ‘autoMpg.arff’ consta de 4 atributos numéricos, 3 nominales y 1 de clase.

‘Cylinders’ es de tipo nominal, no tiene valores perdidos ni valores únicos y existen 5 valores distintos para este parámetro.

‘Displacement’ es de tipo numérico, no tiene valores perdidos tiene 29 valores únicos (7%) y existen 82 valores distintos para este parámetro.

‘Horsepower’ es de tipo numérico, no tiene valores perdidos, tiene 35 valores únicos (9%) y existen 94 valores distintos para este parámetro.

‘Weight’ es de tipo numérico, no tiene valores perdidos, tiene 314 valores únicos (79%) y existen 351 valores distintos para este parámetro.

‘Acceleration’ es de tipo numérico, no tiene valores perdidos, tiene 28 valores únicos (7%) y existen 95 valores distintos para este parámetro.

‘Model’ es de tipo nominal, no tiene valores perdidos ni valores únicos y existen 13 valores distintos para este parámetro.



‘Origin’ es de tipo nominal, no tiene valores perdidos ni valores únicos y existen 3 valores distintos para este parámetro.

Configuramos en la pestaña ‘Classify’ con un 80% de los patrones para entrenar y con el método ‘LinearRegression’ ya que ‘SimpleLinearRegression’ no es posible ejecutarlo en bases de datos con atributos nominales, y pulsamos en ‘Start’, lo que genera una salida como la siguiente.

The screenshot shows the Weka interface with the 'Classify' tab selected. In the 'Test options' panel, 'Percentage split' is chosen at 80%. The 'Classifier output' panel displays the generated regression model:

```
Test mode:split 80.0% train, remainder test
--- Classifier model (full training set) ---
Linear Regression Model
class =
-2.2744 * cylinders=6,3,5,4 +
-4.4422 * cylinders=3,5,4 +
6.7401 * cylinders=5,4 +
0.012 * displacement +
-0.0359 * horsepower +
-0.0056 * weight +
1.6185 * model=75,71,76,74,77,78,79,81,82,80 +
1.8306 * model=77,78,79,81,82,80 +
1.8958 * model=79,81,82,80 +
1.7755 * model=81,82,80 +
1.167 * model=82,80 +
1.2523 * model=80 +
2.1362 * origin=2,3 +
37.9165
Time taken to build model: 0.05 seconds
```

**Imagen 16.** Configuración para ‘LinearRegression’.

Obtenemos así un modelo cuya variable dependiente es el atributo de clase ‘class’ que depende del resto de atributos. Podríamos estimar la clase de una instancia a través de este modelo y el valor de los atributos de esta instancia.



### 2.3.2. Ejercicio 2.

*¿Podrías ejecutar el método SimpleLinearRegression? Justifica tu respuesta. Soluciona el problema, sólo es necesario la aplicación de un filtro.*

No sería posible ejecutar el método 'SimpleLinearRegression' a esta base de datos porque posee atributos nominales y este método sólo admite atributos numéricos.

Para solucionar el problema se aplicará el filtro 'NominalToBinary', una vez aplicado, ejecutamos el método y se generará una salida como la siguiente.

The screenshot shows the Weka interface with the 'Classify' tab selected. Under 'Classifier', 'SimpleLinearRegression' is chosen. In the 'Test options' section, 'Percentage split' is selected with 80% for training. The 'Classifier output' pane displays the following results:

```
origin=3
class
Test mode:split 80.0% train, remainder test
==== Classifier model (full training set) ====
Linear regression on weight
-0.01 * weight + 46.32

Time taken to build model: 0.01 seconds
==== Evaluation on test split ====
==== Summary ===
Correlation coefficient          0.8164
Mean absolute error              3.4188
Root mean squared error          4.3557
Relative absolute error           54.4712 %
Root relative squared error      58.4038 %
Total Number of Instances        80
```

**Imagen 17.** Salida generada por 'SimpleLinearRegression'.



2.3.3. Ejercicio 3.

*Si tenemos el patrón: 6, 173, 115, 2595, 11.3, 79, 1 y sabemos que el valor esperado es 28.8. ¿Qué error obtienen los modelos calculados anteriormente?*

Con el modelo obtenido a través de ‘SimpleLinearRegression’,  $-0.01 \cdot \text{peso} + 46.32$ , siendo nuestro peso 2595 obtenemos un valor de 20.37, por lo que el error de este modelo es 8.43.

Con el modelo obtenido a través de ‘LinearRegression’, obtenemos un valor de 24.4025, por lo que el error de este modelo es 4.3975.

2.3.4. Ejercicio 4.

*¿Cuál es el atributo que nos aporta mayor información de la variable dependiente?*

Para ‘SimpleLinearRegression’ el único atributo que nos aporta información es el peso.

Para ‘LinearRegression’ el atributo que nos aporta más información es cilindros.

2.3.5. Ejercicio 5.

*¿Existe algún atributo que no aporte información al modelo?*

Para ‘SimpleLinearRegression’ el único atributo que nos aporta información es el peso.

Para ‘LinearRegression’ el atributo que no aporta información es la aceleración.

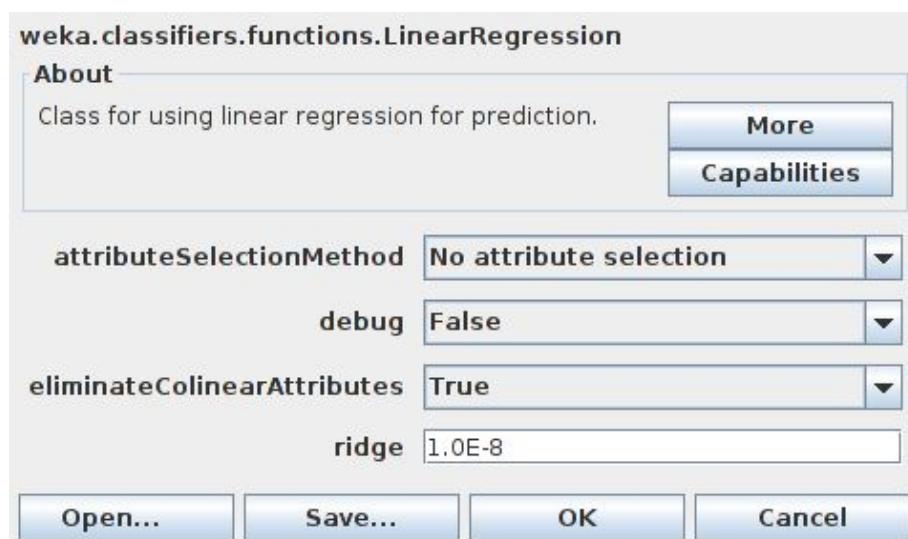


### 2.3.6. Ejercicio 6.

Modifica el parámetro `attributeSelectionMethod` del método `LinearRegression` a “No attribute Selection”.

- Comenta qué ocurre en el nuevo modelo obtenido.
- Analiza por qué “weight” presenta un peso menor que “acceleration”.

Cambiamos el parámetro ‘attributeSelectionMethod’ del método `LinearRegression` y ejecutamos pulsando ‘Start’.



**Imagen 18.** Cambio del parámetro ‘attributeSelectionMethod’.



Generando una salida como la que se muestra abajo.

```
Linear Regression Model

class =
-2.1609 * cylinders=6,3,5,4 +
-4.4201 * cylinders=3,5,4 +
6.5852 * cylinders=5,4 +
0.2542 * cylinders=4 +
0.0125 * displacement +
-0.0359 * horsepower +
-0.0055 * weight +
0.0199 * acceleration +
0.4689 * model=70,72,75,71,76,74,77,78,79,81,82,80 +
-0.3653 * model=72,75,71,76,74,77,78,79,81,82,80 +
1.3936 * model=75,71,76,74,77,78,79,81,82,80 +
-0.0097 * model=71,76,74,77,78,79,81,82,80 +
0.6091 * model=76,74,77,78,79,81,82,80 +
-0.2096 * model=74,77,78,79,81,82,80 +
1.7185 * model=77,78,79,81,82,80 +
-0.0459 * model=78,79,81,82,80 +
1.9437 * model=79,81,82,80 +
1.693 * model=81,82,80 +
1.1546 * model=82,80 +
1.2807 * model=80 +
1.9273 * origin=2,3 +
0.4219 * origin=3 +
36.9902

Time taken to build model: 0.04 seconds
```

**Imagen 18.** Salida tras cambiar de ‘attributeSelectionMethod’.

Podemos observar cómo, de esta manera, se genera un modelo que cuenta con todos los atributos como variables en el modelo. Esto se debe a que con la opción del parámetro ‘M5 method’, el atributo con menor influencia en el modelo se pierde mientras que con la opción ‘No attribute Selection’ se conservan todos los atributos.

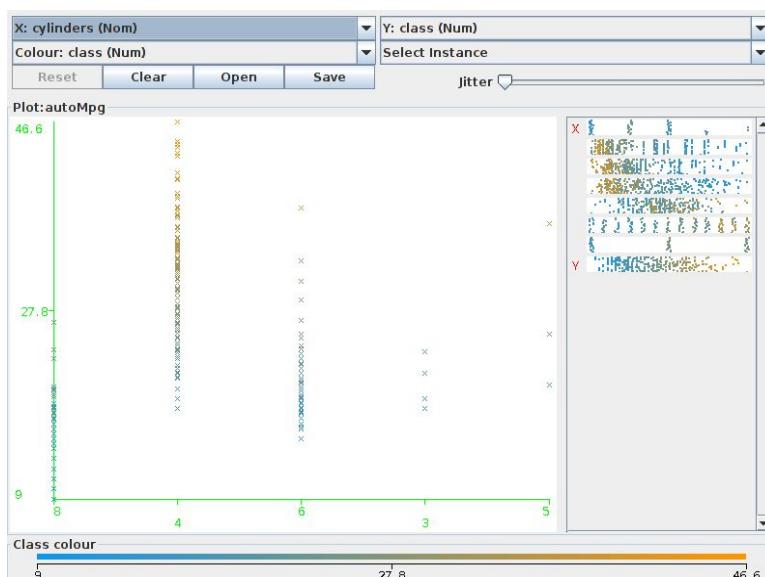
Al generar la nueva salida, vemos que la aceleración tiene mayor peso que el peso, por lo que este debería haber sido el atributo eliminado por ‘M5 method’. Como no ha sido así, intuimos que la razón es porque la eliminación del peso no mejoraba la estimación del error del modelo y por el contrario, la eliminación de la aceleración sí y ahí termina el algoritmo.



### 2.3.7. Ejercicio 7.

Analiza visualmente cada una de las variables continuas independientes frente a la dependiente. ¿Cuáles serían útiles para estimar un modelo simple con una única variable independiente?

Analizando visualmente las variables continuas independientes frente a la dependiente, observamos que las más útiles para estimar el modelo son ‘cylinders’, ‘model’ y ‘origin’ como observamos más abajo.



**Imagen 19.** ‘cylinders’ frente a ‘class’.



UNIVERSIDAD DE CÓRDOBA  
Escuela Politécnica Superior

Ingeniería Informática  
Sistemas Interactivos  
Víctor Monserrat Villatoro  
i32moviv@uco.es

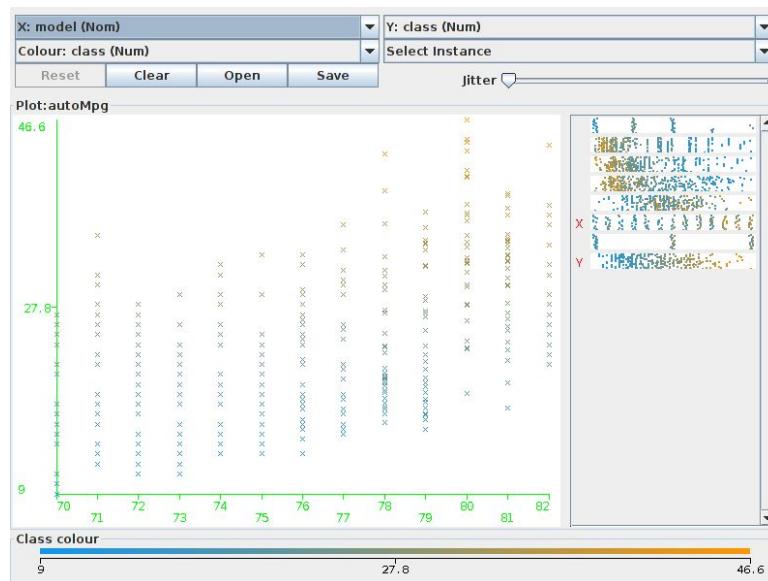


Imagen 19. ‘model’ frente a ‘class’.

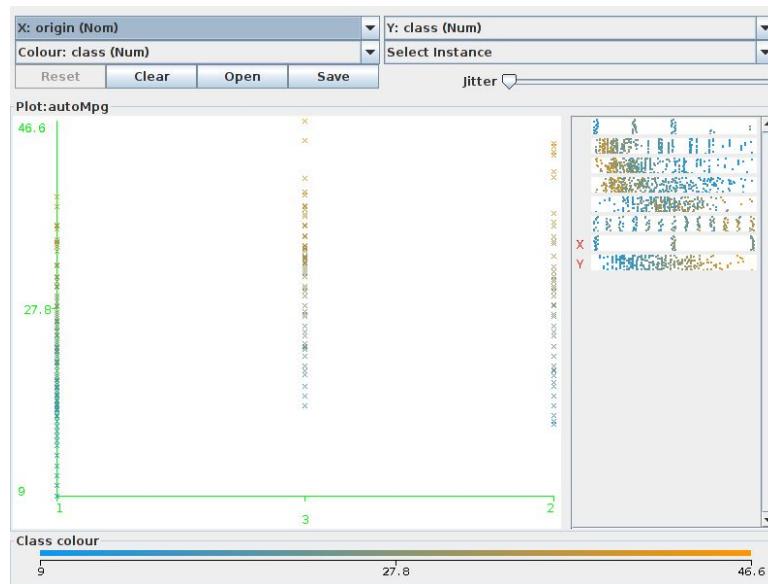


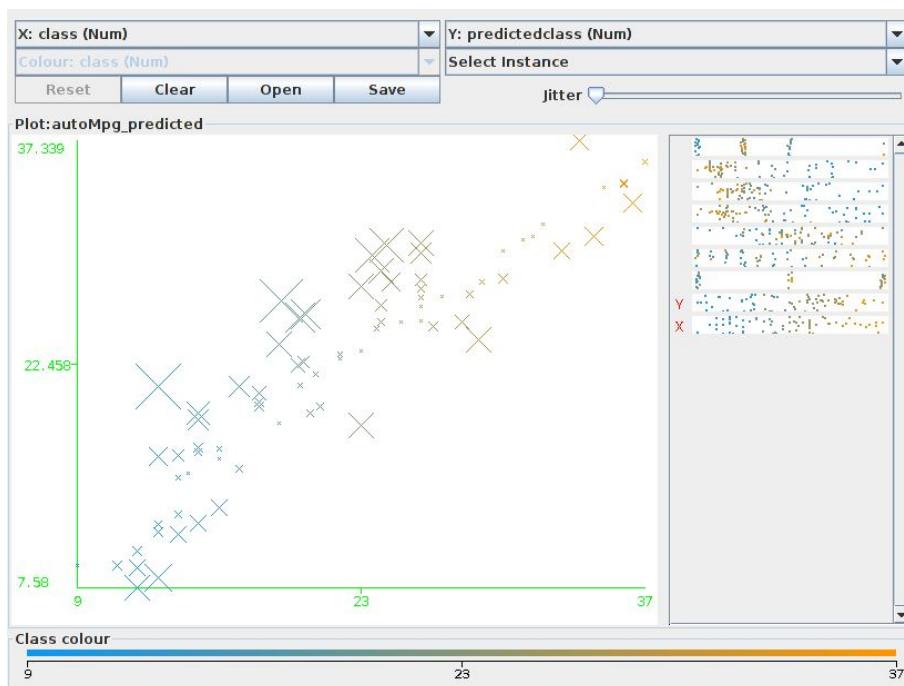
Imagen 20. ‘origin’ frente a ‘class’.



### 2.3.8. Ejercicio 8.

Visualiza gráficamente los errores cometidos. ¿Qué representan las diferentes cruces?

Para visualizar los errores cometidos debemos hacer click derecho y seleccionar ‘Visualize classifier errors’ y se generará una nueva ventana como la siguiente.



**Imagen 21.** Error cometido al estimar con el modelo.

Las cruces representan el error al estimar, cuanto mayor es la cruz mayor es el error cometido en la estimación.



## 2.4. Regresión logística.

### 2.4.1. Ejercicio 1.

Utiliza el entorno *Explorer* para ejecutar *Logistic* y *SimpleLogistic* con tu base de datos, usando un 5-fold como ya se hizo anteriormente con el algoritmo *IBK*. Analiza los modelos obtenidos, las variables que podrían ser más influyentes, variables que no se usan, etc.

Al ejecutar ‘*Logistic*’ se nos genera el siguiente modelo.

==== Classifier model (full training set) ===			
Variable	Class		
	Van	Bus	Saab
compactness	0.3119	-0.0352	0.2237
circularity	-0.5476	-0.5021	-0.6089
distance_circularity	0.2821	-0.1749	0.0092
radius_ratio	-0.0999	-0.7117	0.0172
pr_axis_aspect_ratio	0.4457	2.1148	-0.0015
max_length_aspect_ratio	0.7241	0.2571	-0.2016
scatter_ratio	-0.7699	-0.4518	0.1027
elongatedness	-0.0172	-1.459	-0.0829
pr_axis_rectangularity	-0.5036	-0.0287	0.578
max_length_rectangularity	0.657	0.2687	0.0801
scaled_major_variance	0.1059	0.3851	-0.022
scaled_minor_variance	0.0586	0.0436	-0.0394
scaled_radius_of_gyration	-0.0642	0.0426	0.0614
major_skewness	0.8211	0.2848	0.0636
minor_skewness	-0.0029	-0.2519	0.0326
minor_kurtosis	-0.0463	0.1289	-0.0336
major_kurtosis	-0.1765	1.9213	-0.5019
hollows_ratio	0.6596	-0.966	0.4437
Intercept	-191.9474	-171.7156	-19.0452

**Imagen 22.** Modelo con ‘*Logistic*’.

Podemos ver la influencia que tiene cada atributo sobre cada clase, a mayor coeficiente (en valor absoluto) más influencia.

La última clase no se muestra porque suponemos que toda instancia no clasificada en alguna de estas 3, se clasificará como Opel. La matriz es de ‘atributos x clases-1’.



Al ejecutar 'SimpleLogistic' se nos genera el siguiente modelo.

```
==== Classifier model (full training set) ====  
  
SimpleLogistic:  
  
Class 0 :  
-68.86 +  
[compactness] * 0.17 +  
[distance_circularity] * 0.1 +  
[radius_ratio] * -0.03 +  
[max_length_aspect_ratio] * 0.14 +  
[elongatedness] * 0.34 +  
[pr_axis_rectangularity] * -0.25 +  
[max_length_rectangularity] * 0.26 +  
[scaled_major_variance] * -0.02 +  
[scaled_minor_variance] * -0.01 +  
[scaled_radius_of_gyration] * -0.06 +  
[major_skewness] * 0.05 +  
[minor_skewness] * -0.02 +  
[minor_kurtosis] * -0.01 +  
[hollows_ratio] * 0.08  
  
Class 1 :  
-32.24 +  
[compactness] * -0.03 +  
[distance_circularity] * -0.04 +  
[radius_ratio] * -0.06 +  
[pr_axis_aspect_ratio] * 0.13 +  
[max_length_aspect_ratio] * -0.14 +  
[elongatedness] * -0.26 +  
[scaled_major_variance] * 0.01 +  
[scaled_radius_of_gyration] * 0.01 +  
[major_skewness] * 0.14 +  
[minor_skewness] * -0.11 +  
[minor_kurtosis] * 0.01 +  
[major_kurtosis] * 0.47 +  
[hollows_ratio] * -0.25  
  
Class 2 :  
48.68 +  
[compactness] * 0.09 +  
[circularity] * -0.04 +  
[distance_circularity] * 0.01 +  
[radius_ratio] * 0.03 +  
[pr_axis_aspect_ratio] * -0.19 +  
[max_length_aspect_ratio] * -0.03 +  
[max_length_rectangularity] * -0.08 +  
[scaled_major_variance] * 0 +  
[scaled_radius_of_gyration] * 0.01 +  
[major_skewness] * -0.18 +  
[minor_skewness] * 0.04 +  
[minor_kurtosis] * 0.02 +  
[major_kurtosis] * -0.21 +  
[hollows_ratio] * 0.07
```



```
Class 3 :  
59.87 +  
[compactness] * -0.06 +  
[circularity] * 0.02 +  
[distance_circularity] * 0.03 +  
[radius_ratio] * 0.04 +  
[pr_axis_aspect_ratio] * -0.25 +  
[scaled_major_variance] * 0 +  
[scaled_minor_variance] * 0 +  
[scaled_radius_of_gyration] * -0.02 +  
[major_skewness] * -0.22 +  
[minor_skewness] * 0.01 +  
[minor_kurtosis] * 0.02 +  
[major_kurtosis] * -0.03 +  
[hollows_ratio] * -0.13
```

Time taken to build model: 1.06 seconds

**Imagen 23.** Modelo con ‘SimpleLogistic’.

Podemos ver la influencia que tiene cada atributo sobre cada clase, a mayor coeficiente (en valor absoluto) más influencia. Las variables que no se usan no se muestran en el modelo.



#### 2.4.2. Ejercicio 2.

Visualiza gráficamente los errores cometidos por estos métodos sobre tu base de datos.

Para visualizar los errores cometidos por estos métodos hacemos click derecho y seleccionamos ‘Visualize classifier errors’.

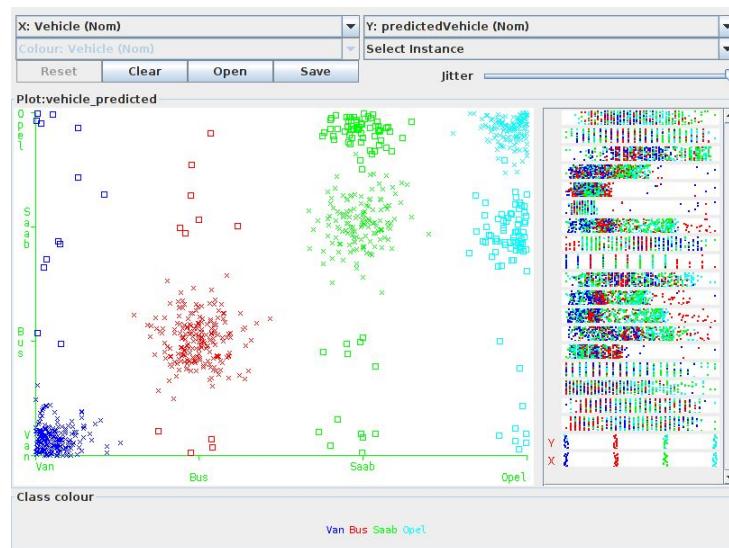


Imagen 24. Error cometido con ‘Logistic’.

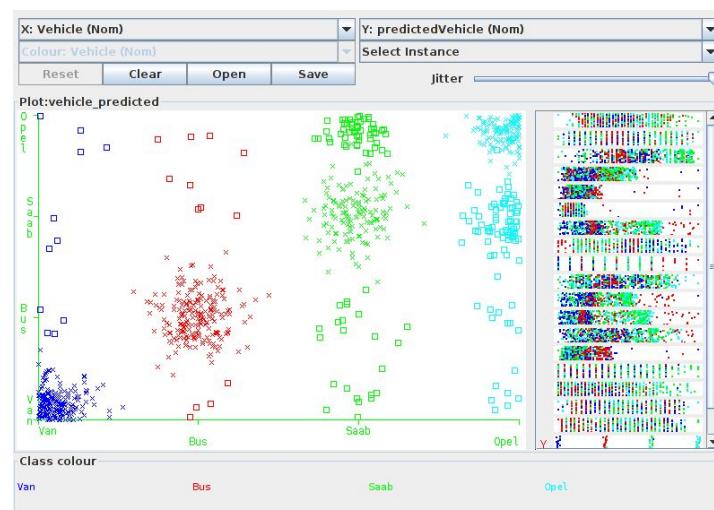


Imagen 24. Error cometido con ‘SimpleLogistic’.



#### 2.4.3. Ejercicio 3.

*Analiza los parámetros `maxBoostingIterations` y `heuristicStop` de `SimpleLogistic` y estudia qué ocurre al modificarlos.*

‘`maxBoostingIterations`’ es el parámetro que determina el número máximo de iteraciones para la ejecución del método.

Si incrementamos el valor de este encontraremos a priori mejores modelos y se aumentará el tiempo de ejecución si no se interrumpe a través de otra condición de parada y al contrario si disminuimos el valor.

‘`heuristicStop`’ es el parámetro que establece el número de iteraciones máximo sin mejora en el error mínimo seguidas.

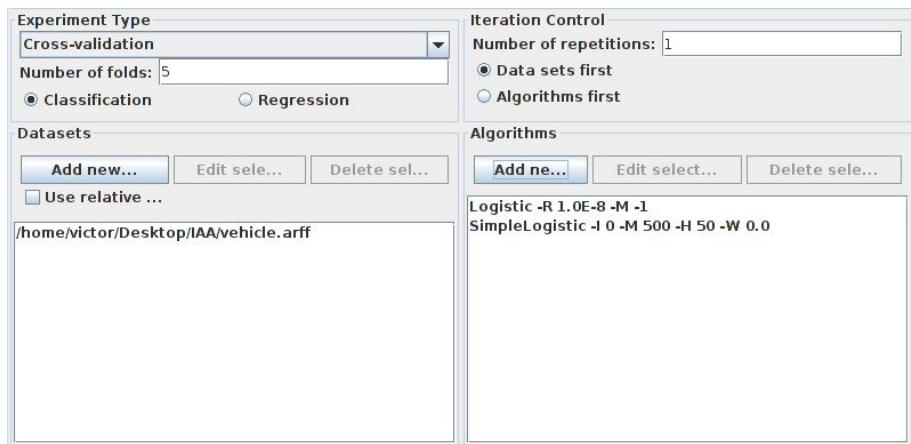
Si incrementamos el valor de este encontraremos a priori mejores modelos y se aumentará el tiempo de ejecución si no se interrumpe a través de otra condición de parada y al contrario si disminuimos el valor.

Ambos parámetros establecen condiciones de parada para el método.



#### 2.4.4. Ejercicio 4.

Utiliza el entorno *Experimenter* para ejecutar *Logistic* y *SimpleLogistic* con tu base de datos, también usando un 5-fold.



**Imagen 25.** Ejecución de ‘Logistic’ y ‘SimpleLogistic’.



2.4.5. Ejercicio 5.

Compara los resultados obtenidos con los que nos proporcionaron los métodos IBK (Accuracy, AUC, Kappa y RMSE).

- Logistic.
- Accuracy.

Selected attribute	
Name:	Percent_correct
Missing:	0 (0%)
Distinct:	5
Unique:	5 (100%)
Statistic	Value
Minimum	74.556
Maximum	84.615
Mean	78.724
StdDev	3.667

- Kappa.

Selected attribute	
Name:	Kappa_statistic
Missing:	0 (0%)
Distinct:	5
Unique:	5 (100%)
Statistic	Value
Minimum	0.661
Maximum	0.795
Mean	0.716
StdDev	0.049

- RMSE.

Selected attribute	
Name:	Root_mean_squared_e...
Missi...	0 (0%)
Distinct:	5
Unique:	5 (10...
Statistic	Value
Minimum	0.222
Maximum	0.288
Mean	0.261
StdDev	0.024



- Media ponderada AUC.

Selected attribute		
Name:	Weighted_avg_area_un...	Type: Nu...
Missing:	0 (0%)	Distinct: 5
		Unique: 5 (...)
Statistic	Value	
Minimum	0.929	
Maximum	0.969	
Mean	0.945	
StdDev	0.015	

- SimpleLogistic.
  - Accuracy.

Selected attribute		
Name:	Percent_correct	Type: Numeric
Missing:	0 (0%)	Distinct: 5
		Unique: 5 (100%)
Statistic	Value	
Minimum	74.556	
Maximum	83.432	
Mean	78.013	
StdDev	3.596	

- Kappa.

Selected attribute		
Name:	Kappa_statistic	Type: Numeric
Missing:	0 (0%)	Distinct: 5
		Unique: 5 (100%)
Statistic	Value	
Minimum	0.661	
Maximum	0.779	
Mean	0.707	
StdDev	0.048	

- RMSE.

Selected attribute		
Name:	Root_mean_squared_e...	Type: Num...
Missi...	0 (0%)	Distinct: 5
		Unique: 5 (100%)
Statistic	Value	
Minimum	0.258	
Maximum	0.297	
Mean	0.273	
StdDev	0.016	



- Media ponderada AUC.

Selected attribute		Type: Nu...	Unique: 5 (...)
Name:	Weighted_avg_area_un...	Missing: 0 (0%)	Distinct: 5
Statistic		Value	
Minimum	0.913		
Maximum	0.95		
Mean	0.935		
StdDev	0.014		

Los métodos ‘Logistic’ y ‘SimpleLogistic’ obtienen mejores valores para estos estadísticos que el algoritmo IBK.

El valor de Accuracy ha aumentado en torno a un 8%, el de Kappa alrededor de un 10%, el error ha disminuido como un 11% y por último la media ponderada ha aumentado AUC aproximadamente un 14%, llegando a rozar casi el 100%.

Por lo que podemos hablar de una mejora bastante clara del modelo con respecto al algoritmo anterior.



### 3. Práctica 3: Clasificación con Weka.

#### 3.1. Árboles de decisión.

##### 3.1.1. Ejercicio 1.

Descarga la Base de datos *zoo.arff* disponible en Weka y ejecuta el algoritmo C4.5 usando la base de datos completa para entrenar y generalizar. Analiza cómo cambia el árbol (nodo principal, tamaño y número de hojas) y los resultados al modificar los atributos *confidenceFactor*, *minNumObj* y *unpruned* por separado. Poner como mínimo 3 valores para *confidenceFactor* y *minNumObj*.

Configuramos el entorno ‘Explorer’, la pestaña ‘Classify’ con el algoritmo J48 y seleccionamos ‘Use training set’ para usar la base de datos completa para entrenar y generalizar. A la derecha se muestra la salida.

The screenshot shows the Weka 'Explorer' interface. The 'Classifier' tab is selected, showing 'J48 -C 0.25 -M 2'. Under 'Test options', 'Use training set' is selected. On the right, the 'Classifier output' pane displays the generated decision tree:

```
==== Classifier model (full training set) ====
J48 pruned tree
-----
feathers = false
|   milk = false
|   |   backbone = false
|   |   |   airborne = false
|   |   |   |   predator = false
|   |   |   |   |   legs <= 2: invertebrate (2.0)
|   |   |   |   |   |   legs > 2: insect (2.0)
|   |   |   |   |   |   |   predator = true: invertibrate (8.0)
|   |   |   |   |   |   |   |   airborne = true: insect (6.0)
|   |   |   |   |   |   |   |   |   backbone = true
|   |   |   |   |   |   |   |   |   |   fins = false
|   |   |   |   |   |   |   |   |   |   |   tail = false: amphibian (3.0)
|   |   |   |   |   |   |   |   |   |   |   tail = true: reptile (6.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   fins = true: fish (13.0)
|   |   |   |   |   |   |   |   |   |   |   milk = true: mammal (41.0)
|   |   |   |   |   |   |   |   |   |   |   feathers = true: bird (20.0)

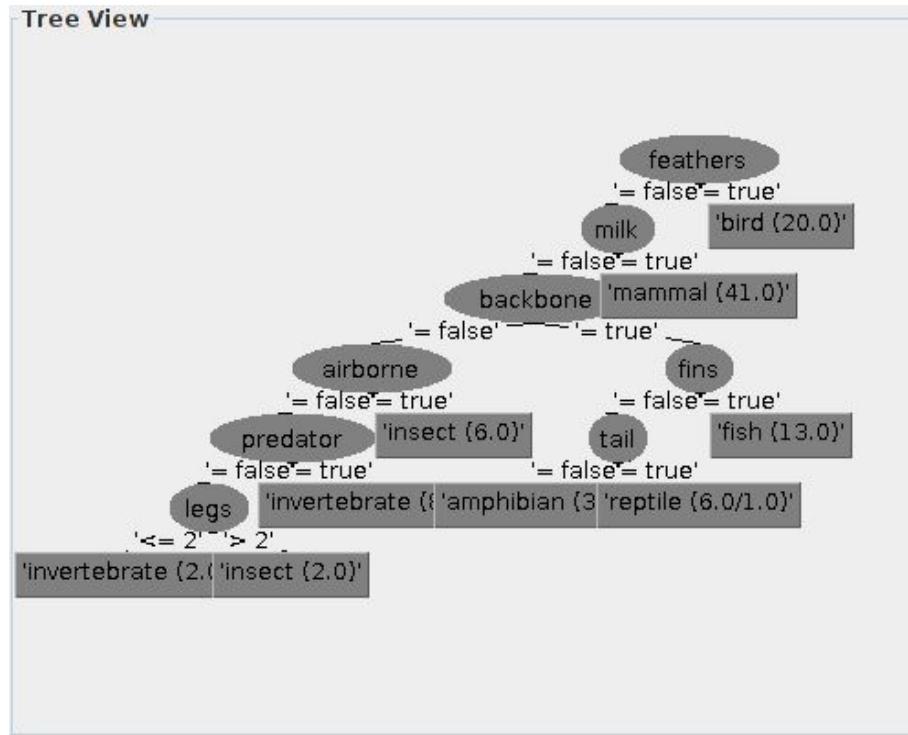
Number of Leaves : 9
Size of the tree : 17

Time taken to build model: 0.02 seconds
```

Imagen 26. Ejecución de ‘C4.5’ (‘J48’).



Si ahora hacemos click derecho y seleccionamos ‘Visualize tree’ se nos genera una nueva ventana con el árbol generado.



**Imagen 27.** Árbol generado por ‘C4.5’ (‘J48’).

El parámetro ‘confidenceFactor’ es el que controla la poda del árbol, valores menores del parámetro significan una poda mayor.

Probando con valores inferiores al por defecto, 0.25. Si disminuimos el valor el árbol se hace más simple pero disminuye la precisión del modelo y al contrario si aumentamos el valor, aunque llega un momento en el que deja de mejorar.

El parámetro ‘minNumObj’ establece el número mínimo de instancias por hoja. Por defecto, el valor es 2.

Si incrementamos el valor, se reduce el tamaño del árbol pero disminuye la precisión del modelo y al contrario si disminuimos el valor.



El parámetro ‘unpruned’ es binario, toma valor positivo o negativo e indica si se realiza la poda o no. Por defecto, toma valor negativo, lo que significa que se realiza la poda. Por el contrario, si toma valor positivo no se realiza la poda.

Si se realiza poda, se incrementa el tamaño del árbol y disminuye la precisión del modelo.

### 3.1.2. Ejercicio 2.

*Analiza lo mismo pero empleando un 80 % para entrenamiento y un 20% para generalización. ¿Qué diferencias observas?*

Para ello seleccionamos ahora ‘Percentage split’, ingresamos el valor 80 y volvemos a ejecutar.

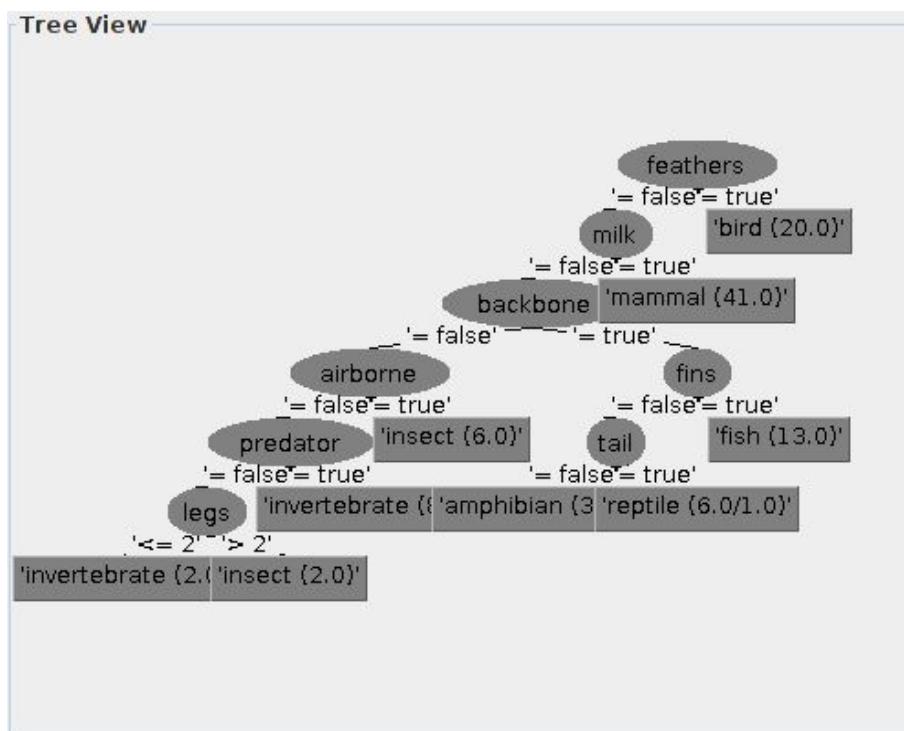
Observamos un incremento del error, un decremento del porcentaje de acierto y por lo tanto disminuye la precisión del modelo porque ahora se usa sólo el 80% para entrenamiento.



### 3.1.3. Ejercicio 3.

Muestra el árbol obtenido en el ejercicio anterior (con un 80/20%) con los parámetros por defecto del C4.5 y analízalo (nodo principal, número de nodos o hojas, variables presentes...).

El árbol obtenido es el que se muestra justo debajo.



**Imagen 28.** Árbol generado por ‘C4.5’ (‘J48’) con 80/20%.

El árbol obtenido tiene como nodo principal el atributo ‘feathers’ (plumas). El tamaño del árbol es 17 (8 nodos y 9 hojas) y las variables presentes son ‘feathers’, ‘milk’, ‘backbone’, ‘airborne’, ‘fins’, ‘predator’, ‘tail’ y ‘legs’.



### 3.1.4. Ejercicio 4.

*Analiza y justifica si podrías aplicar el algoritmo ID3 con tu base de datos. En caso afirmativo, ¿por qué no has tenido problemas al ejecutarlo? En caso negativo, ¿cómo podrías solucionarlo?*

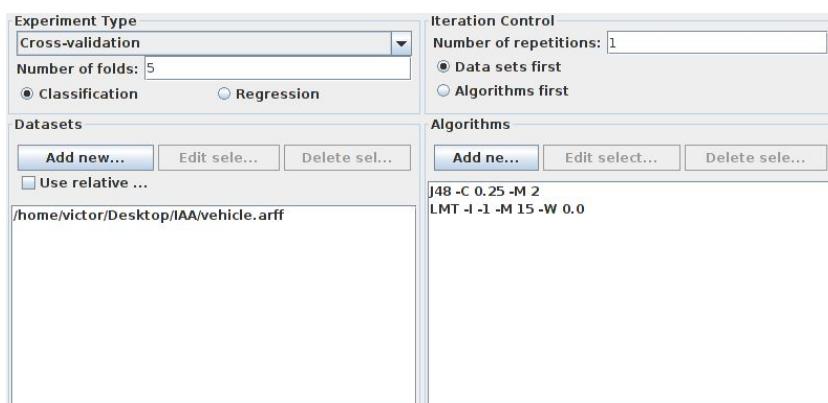
El algoritmo ID3 no se podría usar en nuestra base de datos puesto que posee atributos de tipo numérico y este algoritmo no es compatible con este tipo de atributos.

La solución para este problema es aplicar el filtro ‘NumericToBinary’ o ‘NumericToNominal’ para transformar los atributos numéricos en binarios pero no es interesante aplicarlos en nuestra base de datos.

### 3.1.5. Ejercicio 5.

*Usando el entorno Experimenter, ejecuta los algoritmos C4.5 y LMT usando un 5-fold (para el C4.5 puedes usar los parámetros con los que hayas obtenido el mejor resultado en los apartados anteriores).*

Configuramos el entorno para ejecutar los algoritmos y pulsamos ‘Run’ y ‘Start’ y podremos analizar los resultados.



**Imagen 29.** Configuración del entorno ‘Experimenter’.

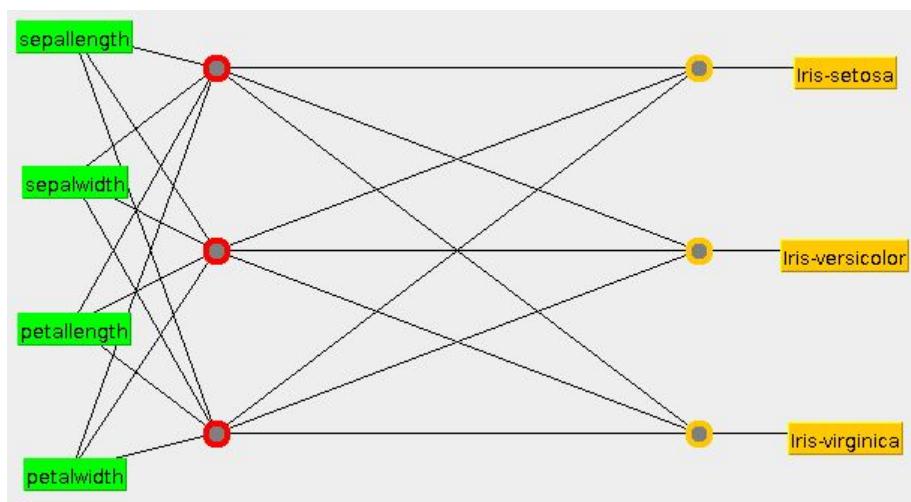
Comprobamos que LMT obtiene mejores resultados que C4.5 (llamado J48 en Weka).



### 3.2. Redes neuronales.

#### 3.2.1. Ejercicio 1.

Utilizando la base de datos “Iris” completa para entrenar y generalizar, representa gráficamente la red resultante de ejecutar el algoritmo MultilayerPerceptron con los valores por defecto. Añade en la gráfica el valor de todos los pesos de la red neuronal.



Esta es la red resultante y a continuación podemos ver los pesos de la red neuronal.



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

Sigmoid Node 0
    Inputs  Weights
    Threshold -0.0491382673800735
    Node 3  0.027023429587306844
    Node 4  0.047737802038675406
    Node 5  -8.617050637471438E-4
Sigmoid Node 1
    Inputs  Weights
    Threshold -0.001813155428890656
    Node 3  0.004644222421406531
    Node 4  -0.017963170264953733
    Node 5  -0.029948356201707205
Sigmoid Node 2
    Inputs  Weights
    Threshold -0.015015642691489917
    Node 3  -0.03513942801283209
    Node 4  -0.02923590167358281
    Node 5  -0.012147148731891946
Sigmoid Node 3
    Inputs  Weights
    Threshold 0.021914050730164863
    Attrib sepallength 0.0242525172953017
    Attrib sepalwidth -0.0305390009971664
    Attrib petallength 0.03355062157463565
    Attrib petalwidth -0.019746308149674242
Sigmoid Node 4
    Inputs  Weights
    Threshold 0.01390399970937567
    Attrib sepallength 0.022662858742601488
    Attrib sepalwidth 0.03201780630162036
    Attrib petallength 0.02401299820526416
    Attrib petalwidth -0.018754663650378568
Sigmoid Node 5
    Inputs  Weights
    Threshold -0.016249285177650778
    Attrib sepallength -0.031117935463524673
    Attrib sepalwidth 0.04259685106621908
    Attrib petallength 0.02133110393823469
    Attrib petalwidth 0.04388816188910931
Class Iris-setosa
    Input
    Node 0
Class Iris-versicolor
    Input
    Node 1
Class Iris-virginica
    Input
    Node 2
```

Time taken to build model: 84.32 seconds



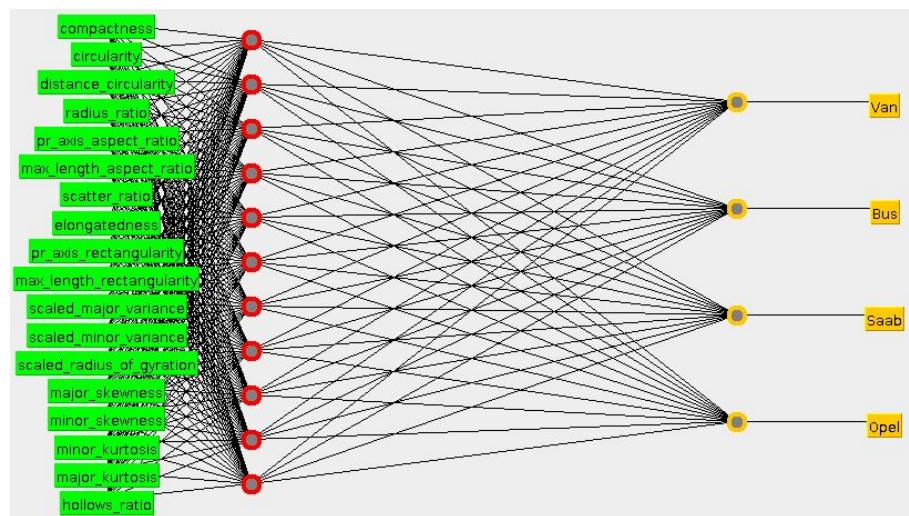
### 3.2.2. Ejercicio 2.

Considerando tu base de datos, ¿cuál sería el valor por defecto para el atributo `hiddenLayers`?

`hiddenLayers` -- This defines the hidden layers of the neural network. This is a list of positive whole numbers. 1 for each hidden layer. Comma separated. To have no hidden layers put a single 0 here. This will only be used if `autobuild` is set. There are also wildcard values '`a`' = (`attribs + classes`) / 2, '`i`' = `attribs`, '`o`' = `classes` , '`t`' = `attribs + classes`.

Como nuestra base de datos consta de 18 atributos y 4 clases, con un simple cálculo ( $\frac{18+4}{2}$ ) podemos conocer el valor por defecto de `hiddenLayers` para nuestra base de datos, 11.

Podemos comprobarlo ejecutando el algoritmo y observando la siguiente salida.



**Imagen 30.** Red resultante de ‘MultilayerPerceptron’.

Se puede apreciar que efectivamente el número de nodos en la capa oculta es 11.



### 3.2.3. Ejercicio 3.

Utilizando un diseño experimental de tipo 80/20% con el algoritmo MultilayerPerceptron, estima el valor que podría considerarse óptimo del parámetro hiddenLayers para tu base de datos (máximo 2 capas ocultas).

Tras varias ejecuciones de este algoritmo, estimamos que el valor óptimo del parámetro hiddenLayers para nuestra base de datos es 30 con sólo una capa, algunos de sus estadísticos podemos verlos abajo.

```
==== Evaluation on test split ===
==== Summary ====
Correctly Classified Instances      149          88.1657
Incorrectly Classified Instances    20           11.8343
Kappa statistic                      0.8422
Mean absolute error                  0.0701
Root mean squared error              0.2144
Relative absolute error              18.6758 %
Root relative squared error         49.4901 %
Total Number of Instances            169

==== Detailed Accuracy By Class ====
          TP Rate   FP Rate   Precision   Recall   F-Measure
          0.956     0.008     0.977     0.956     0.966
          0.976     0.024     0.932     0.976     0.953
          0.786     0.055     0.825     0.786     0.805
          0.8       0.07      0.78      0.8       0.79
Weighted Avg.      0.882     0.038     0.882     0.882     0.881

==== Confusion Matrix ====
  a   b   c   d   <-- classified as
43   0   0   2 |   a = Van
  1  41   0   0 |   b = Bus
  0   2  33   7 |   c = Saab
  0   1   7  32 |   d = Opel
```

Se puede observar como se ha logrado un 88% de acierto.



### 3.2.4. Ejercicio 4.

Con los parámetros por defecto del algoritmo *MultilayerPerceptron*, ¿qué observas al modificar el *learningRate*? ¿Y el *momentum*?

‘learningRate’ representa la cantidad de pesos que son actualizados. Un valor mayor indica que se actualizan menos.

‘momentum’ representa el momentum aplicado a los pesos durante su actualización. Influye en las modificaciones que se hacen en los pesos, un valor mayor indica que estas modificaciones son menos fuertes.

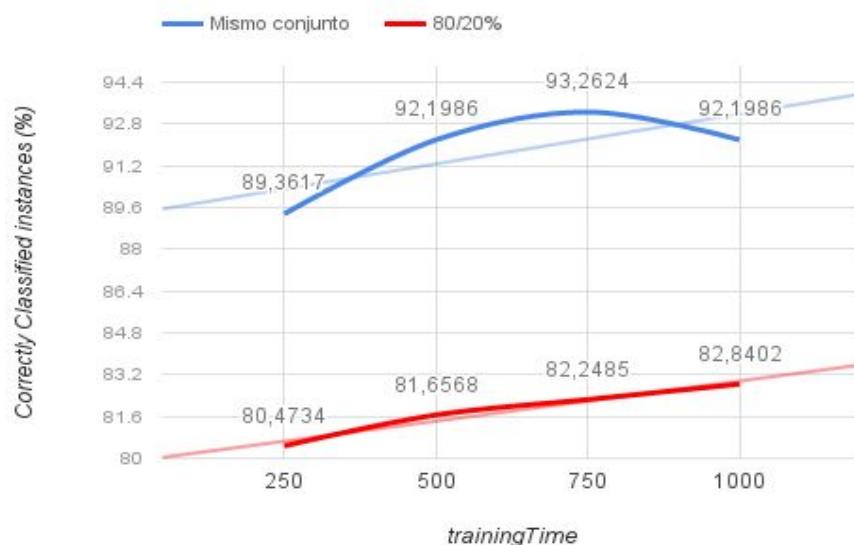
Conseguimos mejores resultados conforme ‘learningRate’ se aproxima a 0 porque el parámetro indica cuántos enlaces se modifican en cada iteración, cuantos más modifiquemos mayor será el error corregido.

Respecto al ‘momentum’ cuanto menor sea mejores resultados obtenemos debido a que estas modificaciones son más fuertes



### 3.2.5. Ejercicio 5.

Realiza una gráfica que muestre cómo cambia el valor de *Correctly Classified instances* al modificar el parámetro *trainingTime*, utilizando primero el mismo conjunto para entrenar y generalizar y después con un 80/20% de tu base de datos. Comentar las gráficas.



**Gráfico 1.** Correctly Classified Instances.

Podemos observar como la tendencia en los dos casos es de un incremento del porcentaje de instancias bien clasificadas a medida que se aumenta el tiempo de entrenamiento, al menos en el intervalo representado.



### 3.2.6. Ejercicio 6.

Usando el entorno *Experimenter*, ejecuta los algoritmos *MultilayerPerceptron* (con la mejor configuración que hayas obtenido) y *RBFNetwork* (por defecto) usando un 5-fold. Compara los resultados obtenidos con el resto de métodos vistos.

Usaremos los parámetros por defecto para todos los algoritmos.

Algoritmo	Accuracy	Kappa	RMSE	AUC
MultilayerPerceptron	81.443	0.753	0.272	0.943
RBFNetwork	64.54	0.527	0.341	0.865
C4.5	73.405	0.645	0.339	0.848
LMT	81.676	0.756	0.253	0.949
Logistic	78.724	0.716	0.261	0.945
SimpleLogistic	78.013	0.707	0.273	0.935
IBK-10	71.511	0.62	0.312	0.026

Los mejores resultados se obtienen en los algoritmos LMT y MultilayerPerceptron porque se obtienen menores valores de RMSE y mayores valores en Accuracy, Kappa y AUC.



#### 4. Práctica 4: Clustering con Weka.

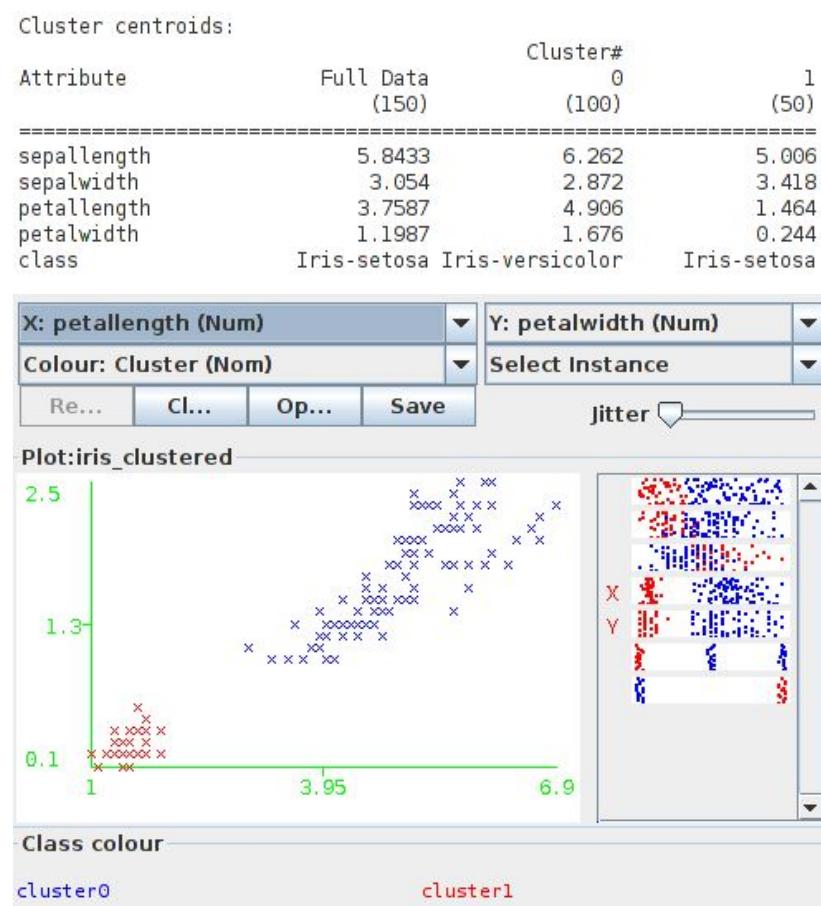
##### 4.1. Algoritmo K-means.

###### 4.1.1. Ejercicio 1.

Seleccionando la opción *Use training set*, analiza los clusters creados por el algoritmo K-means para la base de datos Iris (prueba 2, 3, 4 y 5 clusters). Visualiza los clusters resultantes al representar los atributos *petallength* y *petalwidth*.

En el entorno ‘Explorer’ accedemos a la pestaña ‘Cluster’ y seleccionamos ‘Use training set’ y elegimos ‘SimpleKMeans’.

- 2 clusters.



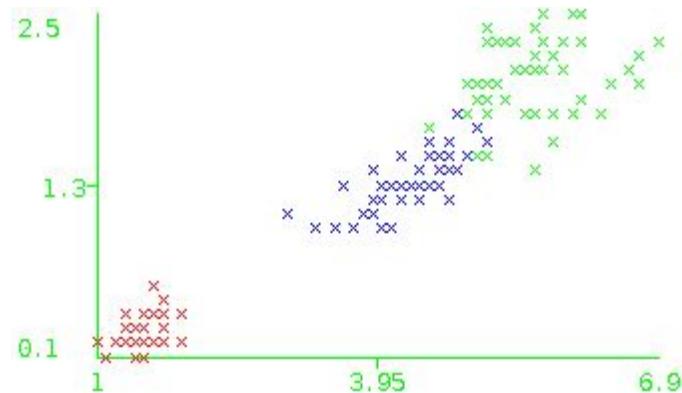
Usaremos la configuración que se muestra en la ventana anterior para las siguientes gráficas pero no se mostrará.



● 3 clusters.

Cluster centroids:

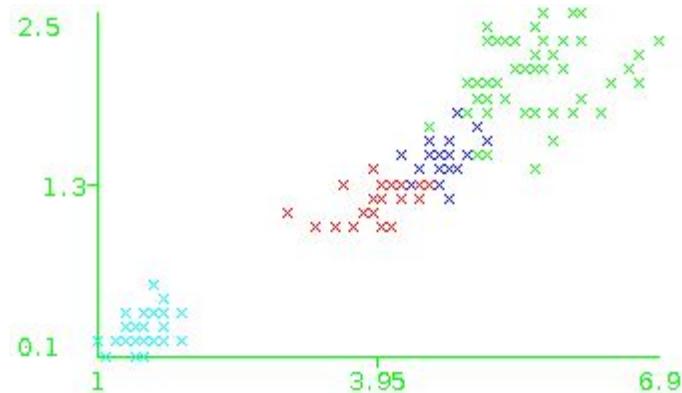
Attribute	Full Data (150)	Cluster# 0 (50)	1 (50)	2 (50)
sepallength	5.8433	5.936	5.006	6.588
sepalwidth	3.054	2.77	3.418	2.974
petallength	3.7587	4.26	1.464	5.552
petalwidth	1.1987	1.326	0.244	2.026
class	Iris-setosa Iris-versicolor		Iris-setosa	Iris-virginica



● 4 clusters.

Cluster centroids:

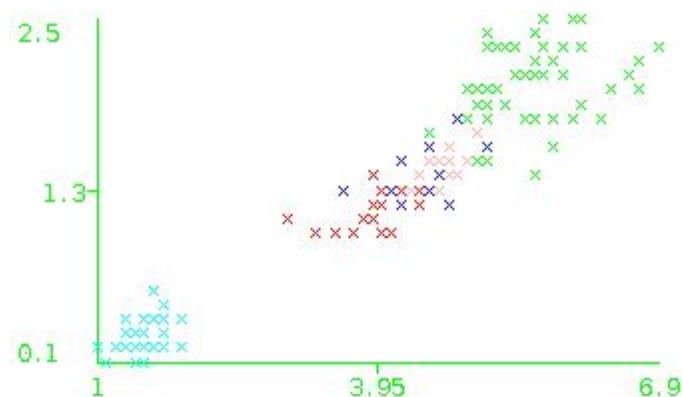
Attribute	Full Data (150)	Cluster# 0 (24)	1 (26)	2 (50)	3 (50)
sepallength	5.8433	6.3292	5.5731	6.588	5.006
sepalwidth	3.054	2.9792	2.5769	2.974	3.418
petallength	3.7587	4.6	3.9462	5.552	1.464
petalwidth	1.1987	1.4625	1.2	2.026	0.244
class	Iris-setosa Iris-versicolor Iris-versicolor		Iris-virginica	Iris-setosa	





● 5 clusters.

Attribute	Full Data (150)	Cluster#				
		0 (19)	1 (19)	2 (50)	3 (50)	4 (12)
sepallength	5.8433	5.8789	5.5526	6.588	5.006	6.6333
sepalwidth	3.054	2.9211	2.4526	2.974	3.418	3.0333
petallength	3.7587	4.4211	3.8632	5.552	1.464	4.6333
petalwidth	1.1987	1.4105	1.1579	2.026	0.244	1.4583
class		Iris-setosa	Iris-versicolor	Iris-versicolor	Iris-virginica	Iris-setosa



Podemos observar como la longitud y la anchura del pétalo son los atributos en los que mayor diferencia existe entre un cluster y otro puesto que son los atributos más significativos.

Como podemos ver siempre una de las tres clases de la base de datos siempre se agrupa correctamente abajo a la izquierda. Las otras dos clases tienen mayor dificultad para ser diferenciadas entre sí con los atributos que han sido analizados para esta base de datos en concreto.



#### 4.1.2. Ejercicio 2.

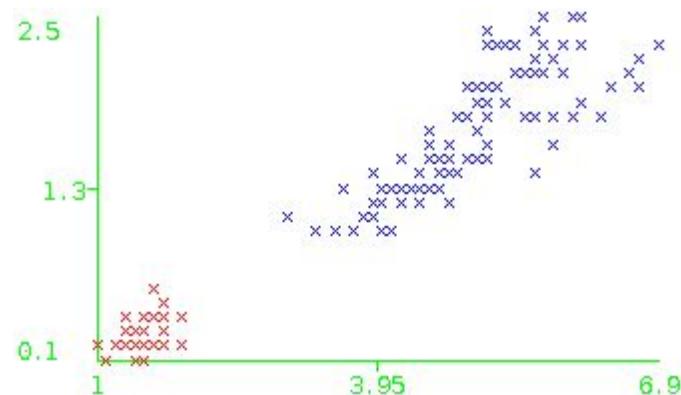
*Repite el mismo proceso anterior ignorando el atributo de clase y analiza si existen diferencias entre los clusters creados.*

Ignoraremos el atributo clase pulsando ‘Ignore attributes’ y seleccionando el atributo ‘class’.

- 2 clusters.

Cluster centroids:

Attribute	Full Data (150)	Cluster# 0 (100)	1 (50)
=====			
sepallength	5.8433	6.262	5.006
sepalwidth	3.054	2.872	3.418
petallength	3.7587	4.906	1.464
petalwidth	1.1987	1.676	0.244

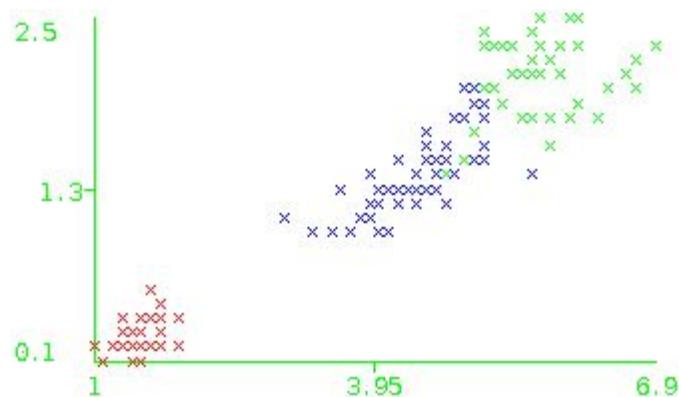




● 3 clusters.

Cluster centroids:

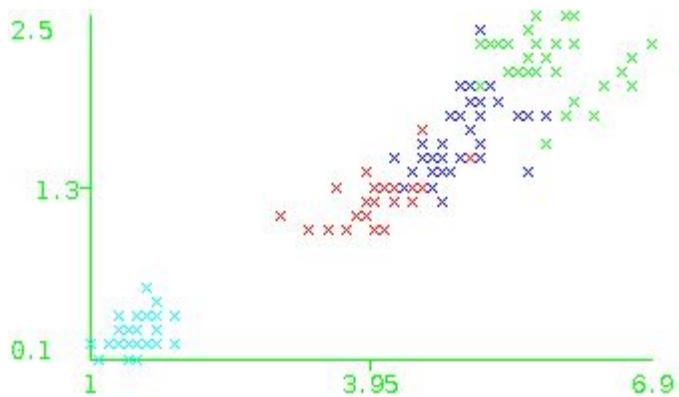
Attribute	Full Data (150)	Cluster#		
		0 (61)	1 (50)	2 (39)
<hr/>				
sepallength	5.8433	5.8885	5.006	6.8462
sepalwidth	3.054	2.7377	3.418	3.0821
petallength	3.7587	4.3967	1.464	5.7026
petalwidth	1.1987	1.418	0.244	2.0795



● 4 clusters.

Cluster centroids:

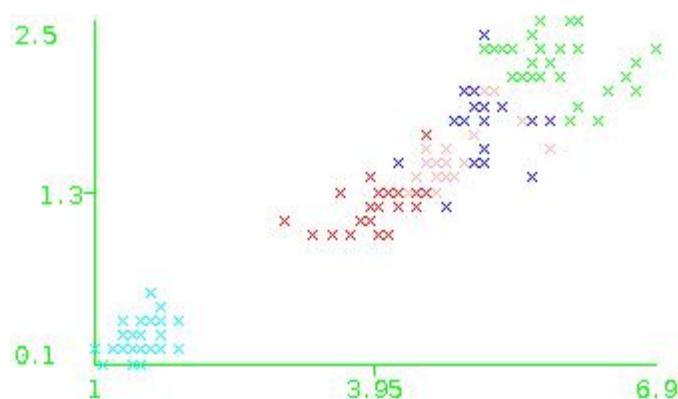
Attribute	Full Data (150)	Cluster#			
		0 (42)	1 (29)	2 (29)	3 (50)
<hr/>					
sepallength	5.8433	6.25	5.5828	6.9586	5.006
sepalwidth	3.054	2.9	2.569	3.1345	3.418
petallength	3.7587	4.8738	4.0034	5.8552	1.464
petalwidth	1.1987	1.6405	1.231	2.1724	0.244





- 5 clusters.

Attribute	Full Data (150)	Cluster #				
		0 (27)	1 (26)	2 (27)	3 (50)	4 (20)
sepallength	5.8433	6.0296	5.55	6.9667	5.006	6.55
sepalwidth	3.054	2.7556	2.5808	3.137	3.418	3.05
petallength	3.7587	4.9444	3.9269	5.8852	1.464	4.805
petalwidth	1.1987	1.7037	1.2	2.2	0.244	1.55



Se observan diferencias significativas respecto a los clusters con el atributo 'class', la suma cuadrática de los errores en los clusters se reduce si no tenemos en cuenta este atributo y por supuesto cambian tanto los valores como las instancias en cada cluster.

Esto se produce porque al ignorar el atributo 'class' estamos denegando mucha información al algoritmo, tenía la información de la clase a la que pertenecía realmente cada instancia. Al ignorar este atributo, el algoritmo deberá intentar clasificar de forma correcta cada instancia sin tener conocimiento de la clase a la que pertenece, que es la información más determinante.

Se observan mayores diferencias a medida que aumentamos el número de clusters y vemos que se obtiene una clasificación peor.



#### 4.1.3. Ejercicio 3.

*De la misma forma, analiza qué ocurre al aplicar K-means en tu base de datos con la etiqueta de clase y sin ella, fijando k=número de clases. Saca alguna conclusión acerca de los clusters creados.*

Como en nuestra base de datos existen cuatro clases, usaremos k=4 para aplicar K-means.

- Con el atributo de clase.

Attribute	Full Data (846)	Cluster#			
		0 (275)	1 (194)	2 (191)	3 (186)
<hr/>					
compactness	93.6785	89.0691	98.7371	89.4084	99.6022
circularity	44.8617	40.8	46.6237	43.6387	50.2849
distance_circularity	82.0887	71.0509	92.4485	73.0733	96.8602
radius_ratio	168.9409	143.7891	187.5103	162.6859	193.1828
pr_axis_aspect_ratio	61.6939	60.0909	61.7165	64.4974	61.1613
max_length_aspect_ratio	8.5674	8.7636	9.201	7.1885	9.0323
scatter_ratio	168.8392	140.0073	185.6701	159.3037	203.7043
elongatedness	40.9338	48.4909	36.6237	41.9476	33.2151
pr_axis_rectangularity	20.5827	18.4727	21.8763	19.6911	23.2688
max_length_rectangularity	147.9988	140.6364	151.3763	143.7644	159.7097
scaled_major_variance	188.6253	161.5018	203.0876	182.1518	220.2903
scaled_minor_variance	439.9113	291.68	523.201	386.1832	627.371
scaled_radius_of_gyration	174.7033	152.4545	184.2887	172.0366	200.3387
major_skewness	72.4622	73.0836	68.7887	76.2513	71.4839
minor_skewness	6.3771	6.6036	7.5928	4.6178	6.5806
minor_kurtosis	12.5993	11.0727	15.5515	9.5654	14.8925
major_kurtosis	188.9326	188.0364	190.5567	188.4607	189.0484
hollows_ratio	195.6891	194.8691	199.6186	192.3613	196.2204
Vehicle		Bus	Van	Saab	Bus
					Opel



● Ignorando el atributo de clase.

Attribute	Full Data (846)	Cluster#			
		0 (178)	1 (217)	2 (214)	3 (237)
<hr/>					
compactness	93.6785	90.9888	93.1244	85.243	103.8228
circularity	44.8617	38.764	45.2028	40.8364	52.7637
distance_circularity	82.0887	72.8989	83.6959	65.7196	102.2996
radius_ratio	168.9409	160.5787	176.0553	132.5981	201.5232
pr_axis_aspect_ratio	61.6939	62.1461	64.53	58	62.0928
max_length_aspect_ratio	8.5674	7.4831	9.6728	7.0748	9.7173
scatter_ratio	168.8392	144.8933	165.8249	140.6916	215
elongatedness	40.9338	45.9888	40.2903	48.3598	31.0211
pr_axis_rectangularity	20.5827	18.691	20.2535	18.4626	24.2194
max_length_rectangularity	147.9988	134.9551	149.4055	137.9813	165.5527
scaled_major_variance	188.6253	167.1292	188.0415	162.7757	228.6456
scaled_minor_variance	439.9113	318.0674	414.0092	292.5794	688.173
scaled_radius_of_gyration	174.7033	140.4157	176.1843	159.0654	213.2194
major_skewness	72.4622	66.3933	71.129	79.2617	72.1013
minor_skewness	6.3771	5.3596	6.8065	5.7383	7.3249
minor_kurtosis	12.5993	14.0225	11.1106	10.0981	15.1519
major_kurtosis	188.9326	195.6685	190.553	182.2477	188.4262
hollows_ratio	195.6891	202.4213	198.0323	186.5	196.7848

Observamos que el número de iteraciones y el tiempo de ejecución ha aumentado aunque la suma cuadrática de los errores en los clusters se reduce. En conclusión, se puede ver que el atributo de clase aporta una gran información de la instancia por lo que tarda menos en clasificarlo en un cluster u otro.



4.1.4. Ejercicio 4.

*¿Podrías saber con anterioridad qué ocurriría si fijásemos el número de clusters igual al número de patrones de una base de datos?*

Si fijamos el número de clusters igual al número de patrones de una base de datos, se asignará un cluster a cada patrón, teniendo cada uno de los cluster un sólo elemento.

Podría ocurrir que existan varios clusters para un mismo patrón pero Weka no permite varios centroides sobre el mismo patrón.

4.1.5. Ejercicio 5.

*El algoritmo K-means es estocástico, ¿por qué? ¿De qué depende que se encuentren unos clusters u otros?*

Se trata de un algoritmo estocástico porque su comportamiento es no determinista, en la medida que el subsiguiente estado del sistema está determinado tanto por las acciones predecibles del proceso como por elementos aleatorios.

Que se encuentren unos cluster u otros depende de la inicialización de los centroides y la métrica de distancia usada.

4.1.6. Ejercicio 6.

*Utilizando tu base de datos y seleccionando la variable de clase en la opción “Classes to clusters evaluation”, ¿con qué número de clusters y con qué métrica de distancia obtienes mejores resultados?. Para obtener mejores resultados realiza 3 ejecuciones con semillas distintas para cada configuración y emplea la media de la métrica SSE.*

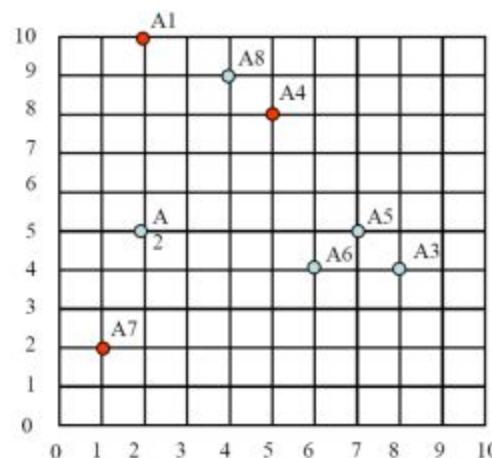
Tras varias ejecuciones con distintas métricas de distancias se observa que se obtienen mejores resultados con la distancia Euclídea que con la distancia Manhattan.



Respecto al número de clusters, se obtienen mejores resultados cuanto mayor es el número de clusters pero a medida que se aumenta el número se reduce esta mejora.

#### 4.1.7. Ejercicio 7.

Agrupa los 8 puntos de la figura en 3 clusters usando el algoritmo K-means. Los centroides iniciales se encuentran sobre los puntos A1, A4 y A7. Emplea la distancia Euclídea para realizar los cálculos (opcionalmente puedes repetir el ejercicio con otras métricas de distancias). Representa los clusters creados y la posición de los centroides después de cada iteración.



	Cluster 0		Cluster 1		Cluster 2	
	Nodos	Centroide	Nodos	Centroide	Nodos	Centroide
Iteración 0	A1	(2, 10)	A4	(5, 8)	A7	(1, 2)

Distancias	A1	A2	A3	A4	A5	A6	A7	A8
Cluster 0	0	5	8.4853	3.6056	7.0711	7.2111	8.0623	2.2361
Cluster 1	3.6056	4.2426	5	0	3.6056	4.1231	7.2111	1.4142
Cluster 2	8.0623	3.1623	7.2801	7.2111	6.7082	5.3852	0	7.6158



	Cluster 0		Cluster 1		Cluster 2	
	Nodos	Centroide	Nodos	Centroide	Nodos	Centroide
Iteración 0	A1	(2, 10)	A4	(5, 8)	A7	(1, 2)
Iteración 1	A1	(2, 10)	A3 A4 A5 A6 A8	(6, 6)	A2 A7	(1.5, 3.5)

Distancias	A1	A2	A3	A4	A5	A6	A7	A8
Cluster 0	0	5	8.4853	3.6056	7.0711	7.2111	8.0623	2.2361
Cluster 1	5.6569	4.1231	2.8284	2.2361	1.4142	2	6.4031	3.6056
Cluster 2	6.5192	1.5811	6.5192	5.7009	5.7009	4.5277	1.5811	6.0415

	Cluster 0		Cluster 1		Cluster 2	
	Nodos	Centroide	Nodos	Centroide	Nodos	Centroide
Iteración 0	A1	(2, 10)	A4	(5, 8)	A7	(1, 2)
Iteración 1	A1	(2, 10)	A3 A4 A5 A6 A8	(6, 6)	A2 A7	(1.5, 3.5)
Iteración 2	A1 A8	(3, 9.5)	A3 A4 A5 A6	(6.5, 5.25)	A2 A7	(1.5, 3.5)

Distancias	A1	A2	A3	A4	A5	A6	A7	A8
Cluster 0	1.118	4.6098	7.433	2.5	6.0208	6.265	7.7621	1.118
Cluster 1	6.5431	4.5069	1.9526	3.1325	0.559	1.3463	6.3885	4.5069
Cluster 2	6.5192	1.5811	6.5192	5.7009	5.7009	4.5277	1.5811	6.0415



	Cluster 0		Cluster 1		Cluster 2	
	Nodos	Centroide	Nodos	Centroide	Nodos	Centroide
Iteración 0	A1	(2, 10)	A4	(5, 8)	A7	(1, 2)
Iteración 1	A1	(2, 10)	A3 A4 A5 A6 A8	(6, 6)	A2 A7	(1.5, 3.5)
Iteración 2	A1 A8	(3, 9.5)	A3 A4 A5 A6	(6.5, 5.25)	A2 A7	(1.5, 3.5)
Iteración 3	A1 A4 A8	(3.67, 9)	A3 A5 A6	(7, 4.33)	A2 A7	(1.5, 3.5)

Distancias	A1	A2	A3	A4	A5	A6	A7	A8
Cluster 0	1.9465	4.3346	6.6143	1.664	5.2047	5.5162	7.4919	0.33
Cluster 1	7.5597	5.0447	1.053	4.1796	0.67	1.053	6.4365	5.5506
Cluster 2	6.5192	1.5811	6.5192	5.7009	5.7009	4.5277	1.5811	6.0415

Puesto que se dejan de producir cambios, se acaba el algoritmo y el resultado es el que se representa en la siguiente imagen.

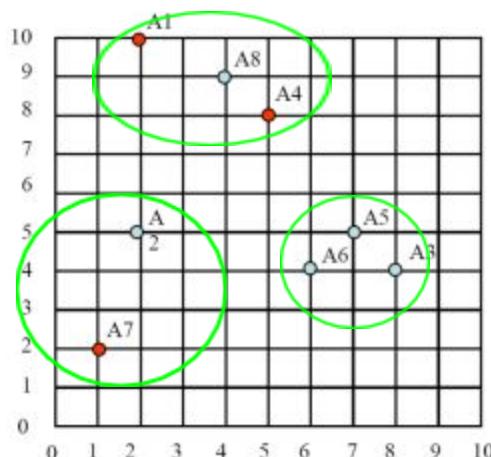


Imagen 30. Clusters.



## 4.2. Clustering jerárquico.

### 4.2.1. Ejercicio 1.

Ejecuta el algoritmo *HierarchicalClusterer* con un número de clusters igual al número de clases de tu problema y sin emplear la etiqueta de clase. ¿Con qué *linkType* obtienes los mejores resultados? (Utiliza la etiqueta de clase para ver la bondad de los clusters obtenidos).

- SINGLE.

Incorrectly clustered instances :	622.0	73.5225 %
-----------------------------------	-------	-----------

- COMPLETE.

Incorrectly clustered instances :	542.0	64.0662 %
-----------------------------------	-------	-----------

- AVERAGE.

Incorrectly clustered instances :	538.0	63.5934 %
-----------------------------------	-------	-----------

- MEAN.

Incorrectly clustered instances :	540.0	63.8298 %
-----------------------------------	-------	-----------

- CENTROID.

Incorrectly clustered instances :	511.0	60.4019 %
-----------------------------------	-------	-----------

- WARD.

Incorrectly clustered instances :	509.0	60.1655 %
-----------------------------------	-------	-----------

- ADJCOMPLPETE.

Incorrectly clustered instances :	628.0	74.2317 %
-----------------------------------	-------	-----------

- NEIGHBOR\_JOINNING.

Incorrectly clustered instances :	628.0	74.2317 %
-----------------------------------	-------	-----------

Podemos comprobar que con WARD se obtienen mejores resultados.



#### 4.2.2. Ejercicio 2.

Utiliza un algoritmo de clustering jerárquico aglomerativo para agrupar los datos descritos por la siguiente matriz de distancias. Emplea el método Simple linkage. Resuelve también el ejercicio con el método Complete linkage y compara los resultados. Representa los dendrogramas.

	A	B	C	D
A	0	1	4	5
B		0	2	6
C			0	3
D				0

- Simple linkage.

Iteración 0	A	B	C	D
A	0	1	4	5
B		0	2	6
C			0	3
D				0

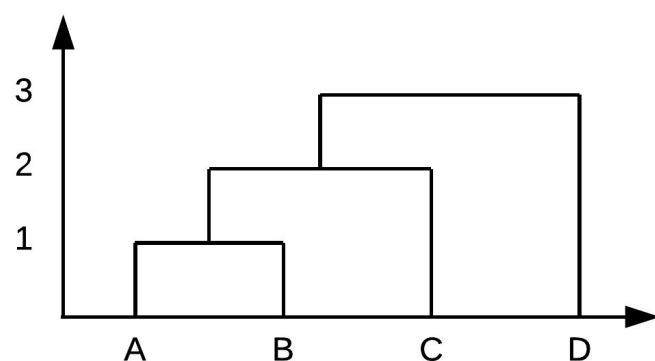
Iteración 1	AB	C	D
AB	0	2	5
C		0	3
D			0

Iteración 2	ABC	D
ABC	0	3
D		0

Por último se unen los dos cluster restantes.



El dendograma resultante es el representado en la siguiente imagen.



**Imagen 31.** Dendograma con Simple linkage.

- Complete linkage.

Iteración 0	A	B	C	D
A	0	1	4	5
B		0	2	6
C			0	3
D				0

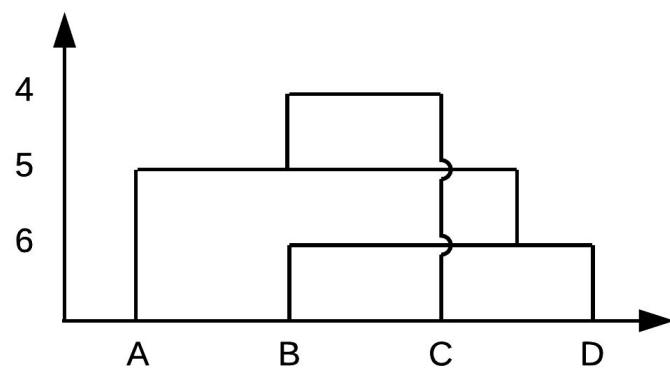
Iteración 1	A	BD	C
A	0	5	4
BD		0	3
C			0

Iteración 2	ABD	C
ABD	0	4
C		0



Por último se unen los dos cluster restantes.

El dendograma resultante es el representado en la siguiente imagen.



**Imagen 31.** Dendograma con Complete linkage.



#### 4.3. Algoritmo DBScan.

##### 4.3.1. Ejercicio 1.

Ejecuta el algoritmo DBScan (sin emplear la etiqueta de clase) cambiando los valores de los parámetros *epsilon* y *minPoints*. ¿Qué número de clusters obtienes y qué precisión presenta cada una de las combinaciones de parámetros? (Utiliza la etiqueta de clase para ver la bondad de los clusters obtenidos). Busca la mejor combinación.

- $\text{epsilon}=0.9, \text{minPoints}=6$ .

```
Number of generated clusters: 1
Incorrectly clustered instances : 628.0 74.2317 %
```

- $\text{epsilon}=0.5, \text{minPoints}=6$ .

```
Number of generated clusters: 1
Incorrectly clustered instances : 621.0 73.4043 %
```

- $\text{epsilon}=0.2, \text{minPoints}=6$ .

```
Number of generated clusters: 4
Incorrectly clustered instances : 11.0 1.3002 %
```

- $\text{epsilon}=0.2, \text{minPoints}=5$ .

```
Number of generated clusters: 7
Incorrectly clustered instances : 26.0 3.0733 %
```

- $\text{epsilon}=0.2, \text{minPoints}=10$ .

```
Number of generated clusters: 1
Incorrectly clustered instances : 0.0 0 %
```

Aunque mejore la precisión, se reduce el número de instancias agrupadas.



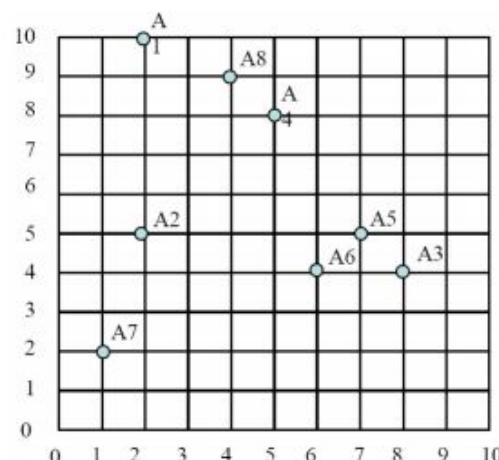
#### 4.3.2. Ejercicio 2.

*¿Con qué algoritmo de clustering has obtenido los clusters más precisos (con respecto a la etiqueta de clase)?*

Se ha obtenido clusters más precisos con el algoritmo ‘DBScan’, llegando a obtener un 100% de precisión porque con este algoritmo no siempre se clasifican todas las instancias.

#### 4.3.3. Ejercicio 3.

*Agrupa los 8 puntos de la figura utilizando el algoritmo DBSCAN. Considera el número mínimo de puntos en el vecindario igual a 2 y el radio del vecindario igual a  $\sqrt{2}$  y  $\sqrt{10}$ .*



Comenzaremos calculando la matriz de distancias.

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	5	8.485	3.605	7.071	7.211	8.062	2.236
A2		0	6.082	4.242	5	4.123	3.162	4.472
A3			0	5	1.414	2	7.280	6.403
A4				0	3.605	4.123	7.211	1.414
A5					0	1.414	6.708	5
A6						0	5.385	5.385
A7							0	7.615



UNIVERSIDAD DE CÓRDOBA  
Escuela Politécnica Superior

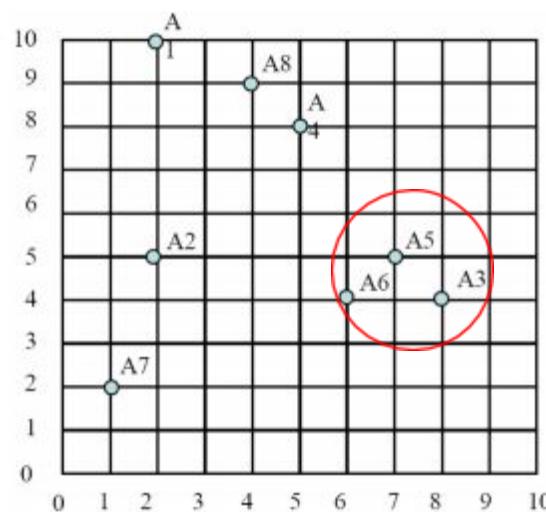
Ingeniería Informática  
Sistemas Interactivos  
Víctor Monserrat Villatoro  
i32moviv@uco.es



A8									0
----	--	--	--	--	--	--	--	--	---

- Radio del vecindario igual a  $\sqrt{2} = 1.414$

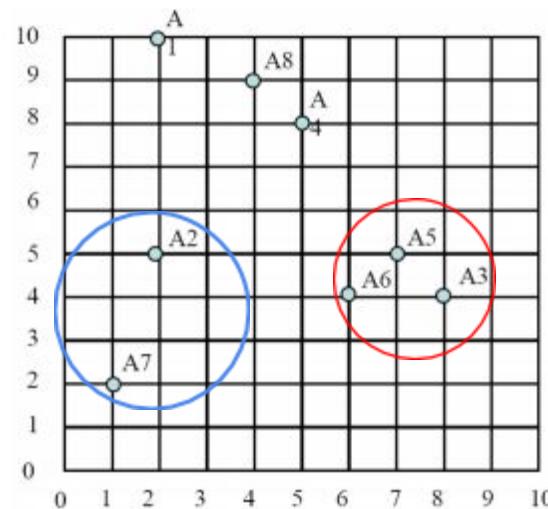
	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	5	8.485	3.605	7.071	7.211	8.062	2.236
A2		0	6.082	4.242	5	4.123	3.162	4.472
A3			0	5	1.414	2	7.280	6.403
A4				0	3.605	4.123	7.211	1.414
A5					0	1.414	6.708	5
A6						0	5.385	5.385
A7							0	7.615
A8								0





- Radio del vecindario igual a  $\sqrt{10} = 3.162$

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	5	8.485	3.605	7.071	7.211	8.062	2.236
A2		0	6.082	4.242	5	4.123	3.162	4.472
A3			0	5	1.414	2	7.280	6.403
A4				0	3.605	4.123	7.211	1.414
A5					0	1.414	6.708	5
A6						0	5.385	5.385
A7							0	7.615
A8								0





## 5. Práctica 5: Redes Neuronales Evolutivas (NNEP).

Antes de comenzar configuraremos nuestra base de datos para poderla usar con este software. Para ello, crearemos un fichero ‘vehicle.dat’ con el siguiente formato y lo situaremos en ~/NNEP/data/complete datasets.

```
<número_patrones> <número_atributos> <número_clases>
< código_columnas>
Patrón1
Patrón2
Patrón3
...
PatrónN
```

Nuestro fichero, por lo tanto, quedará con el siguiente formato.

```
1 846 18 4
2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
3 88.0 45.0 82.0 155.0 56.0 8.0 154.0 43.0 19.0 149.0 180.0 357.0 170.0 69.0 3.0 0.0 188.0 193.0 0 0 1 0
4 94.0 46.0 77.0 169.0 60.0 8.0 158.0 42.0 20.0 148.0 181.0 373.0 181.0 67.0 12.0 2.0 193.0 199.0 0 0 1 0
5 91.0 46.0 78.0 148.0 61.0 9.0 147.0 45.0 19.0 152.0 168.0 323.0 199.0 70.0 13.0 11.0 189.0 200.0 1 0 0 0
6 101.0 52.0 105.0 162.0 53.0 10.0 212.0 31.0 24.0 163.0 226.0 669.0 204.0 74.0 12.0 11.0 186.0 194.0 0 0 0 1
7 85.0 45.0 80.0 154.0 64.0 9.0 147.0 45.0 19.0 148.0 169.0 324.0 174.0 71.0 1.0 4.0 188.0 199.0 1 0 0 0
8 85.0 45.0 70.0 130.0 58.0 8.0 151.0 45.0 19.0 146.0 171.0 334.0 187.0 79.0 2.0 5.0 181.0 186.0 0 1 0 0
9 88.0 46.0 74.0 171.0 68.0 6.0 152.0 43.0 19.0 148.0 180.0 349.0 192.0 71.0 5.0 11.0 189.0 195.0 0 1 0 0
10 90.0 37.0 80.0 171.0 58.0 9.0 157.0 42.0 20.0 132.0 172.0 373.0 115.0 60.0 3.0 18.0 201.0 209.0 0 0 1 0
```

Ahora podremos cargar la base de datos y dividirla en un 75% para entrenar y un 25% para generalizar.

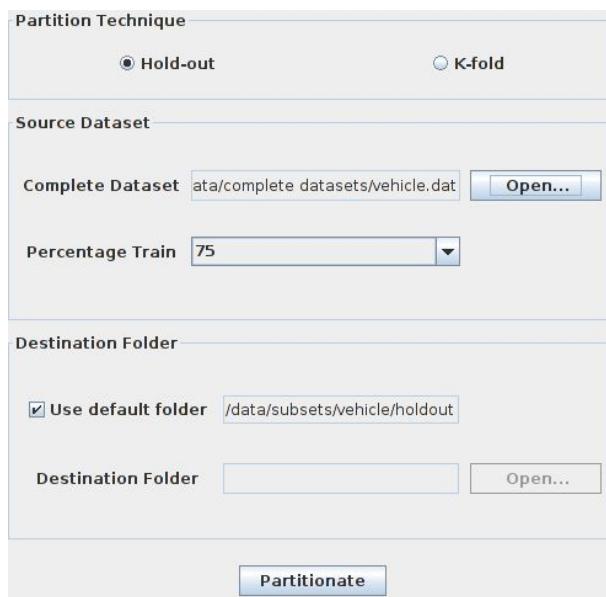


En primer lugar, ejecutamos el programa y se nos mostrará la siguiente interfaz inicial.



**Imagen 32.** Interfaz de NNEP.

Pulsamos en ‘Data Management’ y se nos abrirá una nueva ventana, la cual configuraremos de la siguiente forma y pulsaremos ‘Partitionate’.



**Imagen 33.** Dendograma con Complete linkage.

Y se crearán dos subconjuntos, cada uno en un fichero, en el directorio seleccionado.



### 5.1. Ejercicio 1.

*Descripción de los valores utilizados para los distintos parámetros y del razonamiento que se ha seguido para su elección.*

Pulsamos ahora en ‘Mono-Objective Experiments’, se abrirá una nueva ventana, pulsaremos para crear uno nuevo y lo configuraremos de la siguiente forma, cambiando el nombre en cada caso. Usaremos ‘DoubleElitistNeuralNetAlgorithm’.

Elegiremos una configuración de 100 generaciones porque aumentando el número de generaciones no se consiguen mejoras notables y aumenta bastante el tiempo de ejecución.

Experiment

Name of the Experiment: VehicleSUs/VehiclePUs

Number of Executions: 30

Algorithm

Type of Algorithm: DoubleElitistNeuralNetAlgorithm

Type of Problem: Classification

Specie

Type of Specie: NeuralNetIndividualSpecies

Evaluator

Error Function: LogisticErrorFunction

Normalize Data

Evaluation Method: Hold-out

Train Data File: /ata/subsets/vehicle/holdout/train\_vehicle.dat

Test Data File: /ata/subsets/vehicle/holdout/test\_vehicle.dat

Mutators

Structural Mutator: StructuralMutator

Parametric Mutator: ParametricSRMutator

Listeners

Statistical Listener

Statistical ... Number of Decimals: 7

Report Directory Name: report Extended Report Frequency: 20

Chart Listener

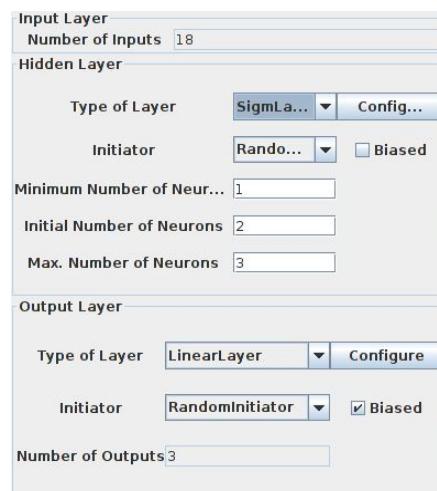
Chart Repo... Options:  Best Fitness  CCR/MSE  Means Fitness

Imagen 34. Configuración del experimento.



- NNEP solo con SUs.

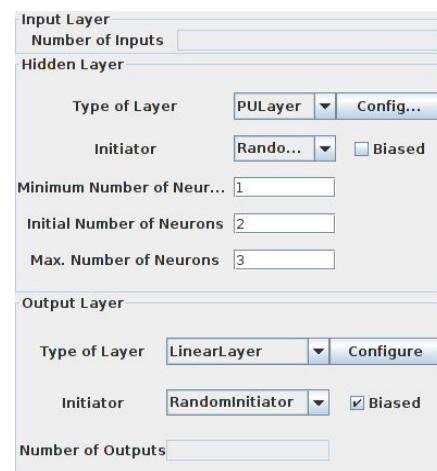
Configuramos las propiedades de la especie.



**Imagen 35.** Configuración de la especie para SUs.

- NNEP solo con PUs.

Configuramos las propiedades de la especie.



**Imagen 36.** Configuración de la especie para PUs.

Los valores quedan por defecto, se consiguen buenos resultados. Si aumentamos el número de neuronas, se sobreentrena el modelo.



## 5.2. Ejercicio 2.

Resultados obtenidos según el formato de las hojas de cálculo obtenidas por el programa. Es obligatorio incluir la media y la desviación típica en CCR de los 30 resultados para el conjunto de generalización.

Tras pulsar en por cada experimento, esperamos la ejecución del algoritmo y abrimos los ficheros generados por este, los cuales se encuentran en ~/NNEP/reports. Evaluaremos el subconjunto para generalizar, es decir, el de test.

- NNEP solo con SUs.

Generación	CCR			
	Media		Desviación típica	
	Entropía	Precisión	Entropía	Precisión
20	37.34	44.95	6.09	4.19
40	42.31	48.39	4.71	4.55
60	46.79	50.86	5.37	5.25
80	49.15	53.66	6.15	5.22
Final	51.72	55.33	5.45	5.27

- NNEP solo con PUs.

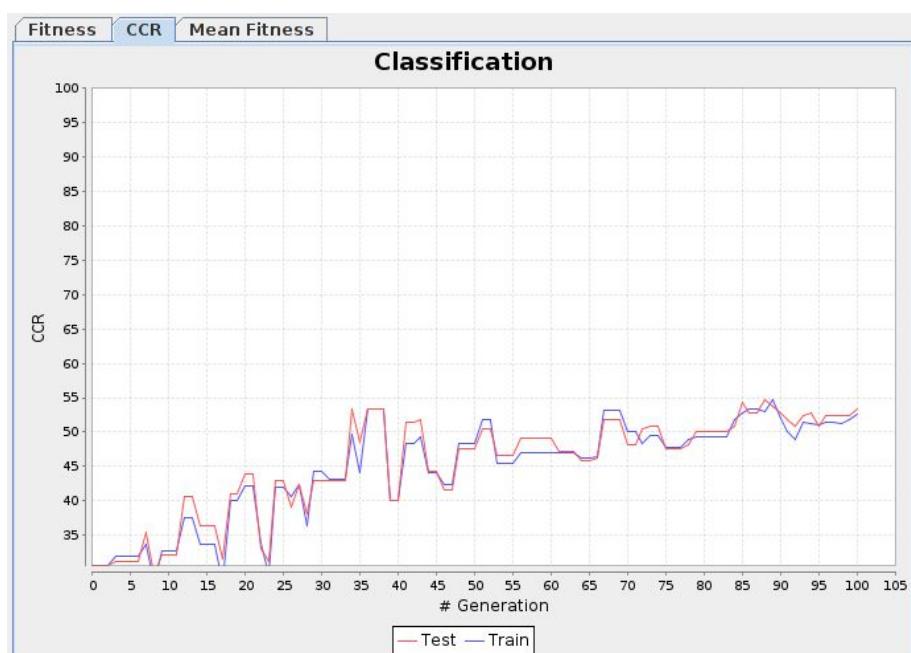
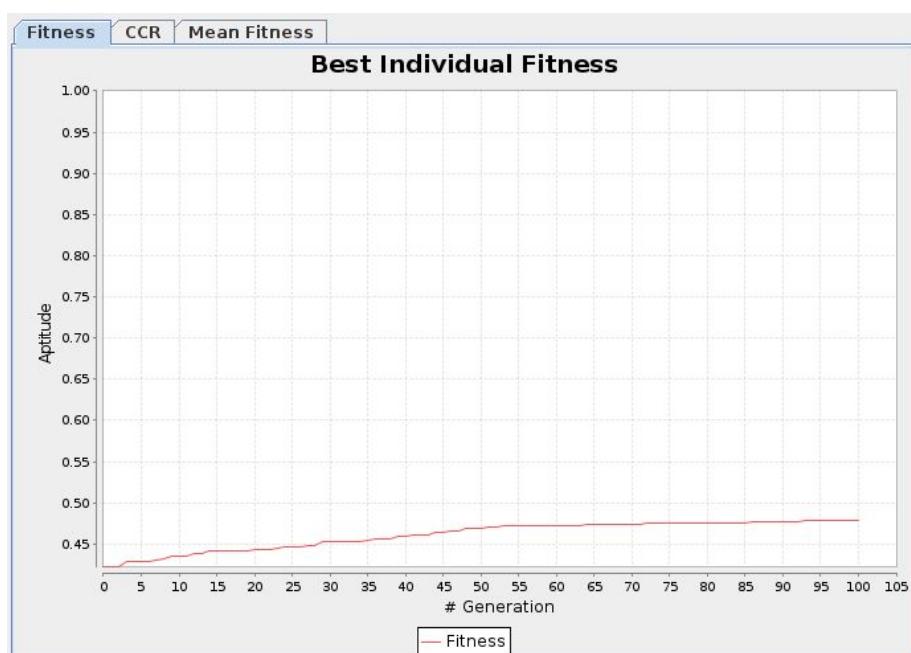
Generación	CCR			
	Media		Desviación típica	
	Entropía	Precisión	Entropía	Precisión
20	43.56	47.67	6.29	4.23
40	46.21	52.83	6.01	4.12
60	51.21	54.33	5.73	5.03
80	53.93	56.33	6.14	5.24
Final	56.22	57.56	6.00	4.89

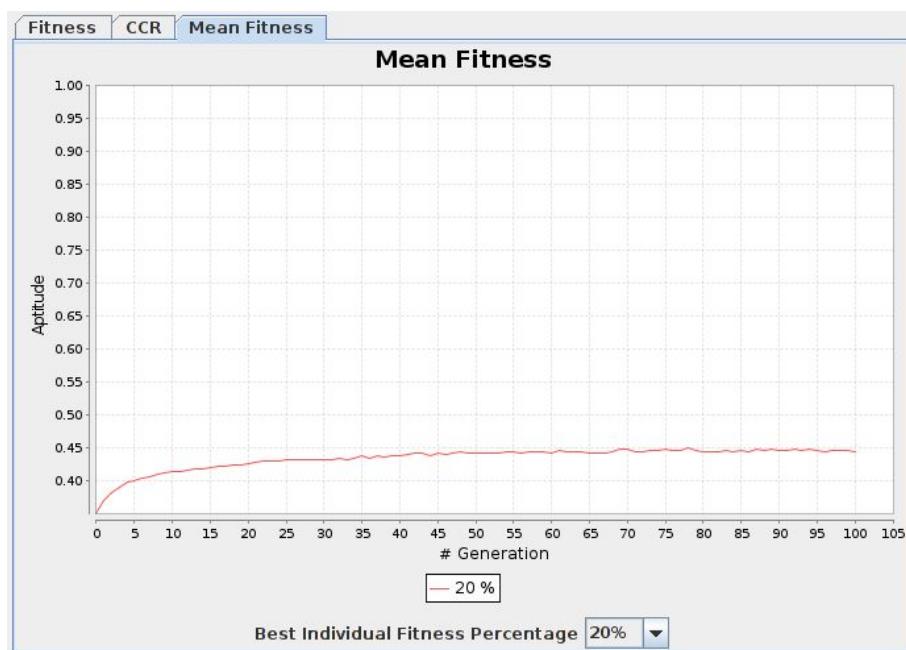


### 5.3. Ejercicio 3.

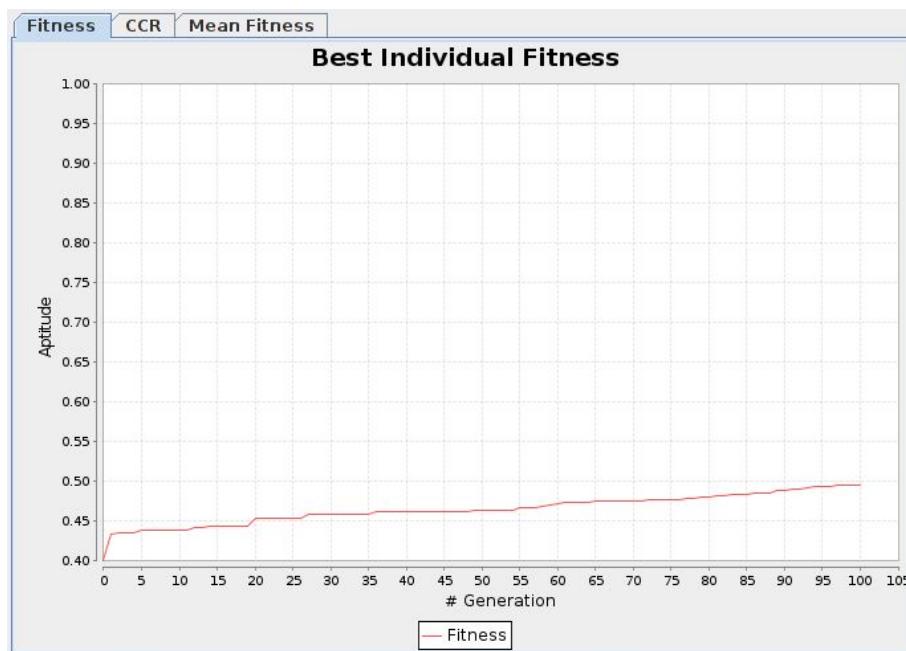
Análisis del comportamiento del algoritmo utilizando las gráficas de convergencia para entrenamiento y generalización que genera NNEP.

- NNEP solo con SU's.





- NNEP solo con PUs.





Podemos observar como la aptitud mejora a medida que se realizan iteraciones. Sin embargo la clasificación y la media de la aptitud mejora en las primeras iteraciones pero llega un momento que se mantiene e incluso puede bajar.

Los valores de train y test son parecidos, aunque un poco mejores los de test excepto algunas iteraciones. Que se consigan mejores valores en el conjunto de entrenamiento puede significar un sobreentrenamiento, pero al conseguirse en tan pocas iteraciones no lo consideraremos.



#### 5.4. Ejercicio 4.

*Análisis del mejor modelo obtenido. Del conjunto de los 30 modelos obtenidos, seleccionar el que resulte en el menor error de generalización. Analizar la fórmula matemática correspondiente de forma que sepa discernir entre capas, entradas, sesgos y salidas. Analizar la matriz de confusión para el conjunto de generalización.*

Analizaremos solamente la entropía del doble elitismo, la precisión se podría analizar de la misma forma.

- NNEP solo con SUs.

El mejor modelo se ha obtenido en la iteración 25 de la generación final cuyo **Test CCR: 58.01886792452831**.

La fórmula matemática es la siguiente:

```
13.848330101553858 * ( 1/(1+e^-(-  
+1.2504151476476704 * ( x1 )  
-1.1389720411293858 * ( x4 )  
-0.8306846703274283 * ( x7 )  
+0.8040817397110928 * ( x8 )  
-3.498822608264017 * ( x9 )  
+2.697249225406501 * ( x10 ) )) )  
-6.694590100515767 * (1)  
3.932276835105559 * ( 1/(1+e^-(-  
-0.2858407354210381 * ( x1 )  
+0.20858101341639373 * ( x2 )  
-2.519722789953512 * ( x3 )  
+0.25705665797033894 * ( x4 )  
+0.3989233875772832 * ( x7 )  
+0.6797469361108147 * ( x10 )  
-0.27808459213737663 * ( x11 )  
+1.4427057981757319 * ( x13 )  
+1.4477610475417233 * ( x14 )  
-1.8362400586152985 * ( x15 )  
+0.5243870995935597 * ( x17 ) )) )  
-1.331346179194805 * (1)  
  
0.2744578104540334 * ( 1/(1+e^-(-  
+1.2504151476476704 * ( x1 )  
-1.1389720411293858 * ( x4 )  
-0.8306846703274283 * ( x7 )  
+0.8040817397110928 * ( x8 )  
-3.498822608264017 * ( x9 )  
+2.697249225406501 * ( x10 ) )) )  
-0.07353627199925412 * ( 1/(1+e^-(-  
+1.0486709439978008 * ( x2 )  
-0.06344113817960621 * ( x6 )  
+0.39461572447912674 * ( x8 )  
+1.083524466153341 * ( x9 )  
+0.5234304403848442 * ( x16 )  
+0.6556922422777467 * ( x18 ) )) )  
+0.09212196893499643 * (1)
```

Los nodos de la capa de entrada se representan por x seguido del número de atributo correspondiente, los valores que multiplican son los pesos que tienen en la unión entre el nodo de entrada y el de la capa oculta.



Los valores multiplicados por (1) son los sesgos desde la capa de entrada hasta la capa oculta, uno por cada neurona.

Como no se representan ni el atributo 5 ni el 12 podemos ver que existen 16 neuronas en la capa de entrada y por la salida sabemos que existen 3 neuronas en la capa oculta.

Como se puede ver el primer nodo de la capa oculta tiene conexión con el primer nodo de la capa de salida. El segundo nodo de la capa oculta tiene conexión con el segundo de la capa de salida. Por último, el tercero de la capa oculta tiene conexión con el primero y el tercero de la capa de salida.

La matriz de confusión es la siguiente para el conjunto de generalización:

Test Confusion Matrix				
-----				
Predicted	0	1	2	3
Target	0	47	1	3
0	47	1	3	0
1	11	43	1	0
2	5	16	33	0
3	5	21	26	0
CCR	0.9215686274509803	0.7818181818181819	0.6111111111111112	0.0

Observamos que la primera clase ha sido la mejor clasificada y que existen dos clases que se clasifican muy bien y otras dos que no tanto.



- NNEP solo con PUs.

El mejor modelo se ha obtenido en la iteración 9 de la generación final cuyo **Test CCR: 65.09433962264151**.

La fórmula matemática es la siguiente:

$$\begin{aligned} & 0.5459212006731411 * (x5^{-0.7227722440379648} * x6^{1.5650071698360446} * x7^{0.4627537337672522} * x8^{-1.6665660066539094} * \\ & x13^{0.42757053371530573} * x14^{-0.9807976859848605} * x15^{0.4690906847963664} * x17^{0.7186500446418389} * \\ & x18^{0.9658898601247177}) \\ & -6.1127039407879975 * (x1^{-1.6867104599925047} * x4^{1.4843975704011796} * x5^{-0.7486097901121554} * x6^{-0.620787360521097} * \\ & x7^{3.3851537120368267} * x8^{-0.41197398056687345} * x9^{0.8843204325578364} * x10^{-2.6790058566994617} * \\ & x13^{0.5500666951687648} * x14^{0.41557388335931145} * x16^{0.09323690051530299} * x17^{0.3796217572918034} * \\ & x18^{0.06015285388945782}) \\ & +0.7780834059659956 * (x4^{0.31816956367187166} * x5^{0.910654023372951} * x6^{-1.1026301390268012} * x8^{-0.4691402848795656} * \\ & x9^{0.30743613388341123} * x11^{0.28197739748009837} * x13^{0.6211160987147134} * x15^{-0.10351365329661728} * \\ & x17^{1.1558704114263603} * x18^{-0.8551954137154745}) \\ & +4.126359311758535 * (1) \\ & -1.8829763954403804 * (x5^{-0.7227722440379648} * x6^{1.5650071698360446} * x7^{0.4627537337672522} * x8^{-1.6665660066539094} * \\ & x13^{0.42757053371530573} * x14^{-0.9807976859848605} * x15^{0.4690906847963664} * x17^{0.7186500446418389} * \\ & x18^{0.9658898601247177}) \\ & +2.1582652984481445 * (x4^{0.31816956367187166} * x5^{0.910654023372951} * x6^{-1.1026301390268012} * x8^{-0.4691402848795656} * \\ & x9^{0.30743613388341123} * x11^{0.28197739748009837} * x13^{0.6211160987147134} * x15^{-0.10351365329661728} * \\ & x17^{1.1558704114263603} * x18^{-0.8551954137154745}) \\ & -0.3188163704473376 * (1) \\ & 0.4256826387932924 * (x5^{-0.7227722440379648} * x6^{1.5650071698360446} * x7^{0.4627537337672522} * x8^{-1.6665660066539094} * \\ & x13^{0.42757053371530573} * x14^{-0.9807976859848605} * x15^{0.4690906847963664} * x17^{0.7186500446418389} * \\ & x18^{0.9658898601247177}) \\ & -0.8692467539629594 * (1) \end{aligned}$$

Los nodos de la capa de entrada se representan por x seguido del número de atributo correspondiente, los valores que multiplican son los pesos que tienen en la unión entre el nodo de entrada y el de la capa oculta.

Los valores multiplicados por (1) son los sesgos desde la capa de entrada hasta la capa oculta, uno por cada neurona.

Como no se representan ni el atributo 2, ni el 3 ni el 12 podemos ver que existen 15 neuronas en la capa de entrada y por la salida sabemos que existen 3 neuronas en la capa oculta.



Como se puede ver el primer nodo de la capa oculta tiene conexión con los tres nodos de la capa de salida. El segundo nodo de la capa oculta tiene conexión con el primero y el tercero de la capa de salida. Por último, el tercero de la capa oculta tiene conexión con el primero de la capa de salida.

La matriz de confusión es la siguiente para el conjunto de generalización:

Test Confussion Matrix					
Predicted				CCR	
Target	0	1	2	3	
0	48	1	1	1	0.9411764705882353
1	1	52	0	2	0.9454545454545454
2	6	17	28	3	0.5185185185185185
3	2	15	25	10	0.19230769230769232

Observamos que la segunda clase ha sido la mejor clasificada y que existen dos clases que se clasifican muy bien y otras dos que no tanto.