Diplomado en Minería de Datos

PEUVI, Facultad de Ciencias, UNAM

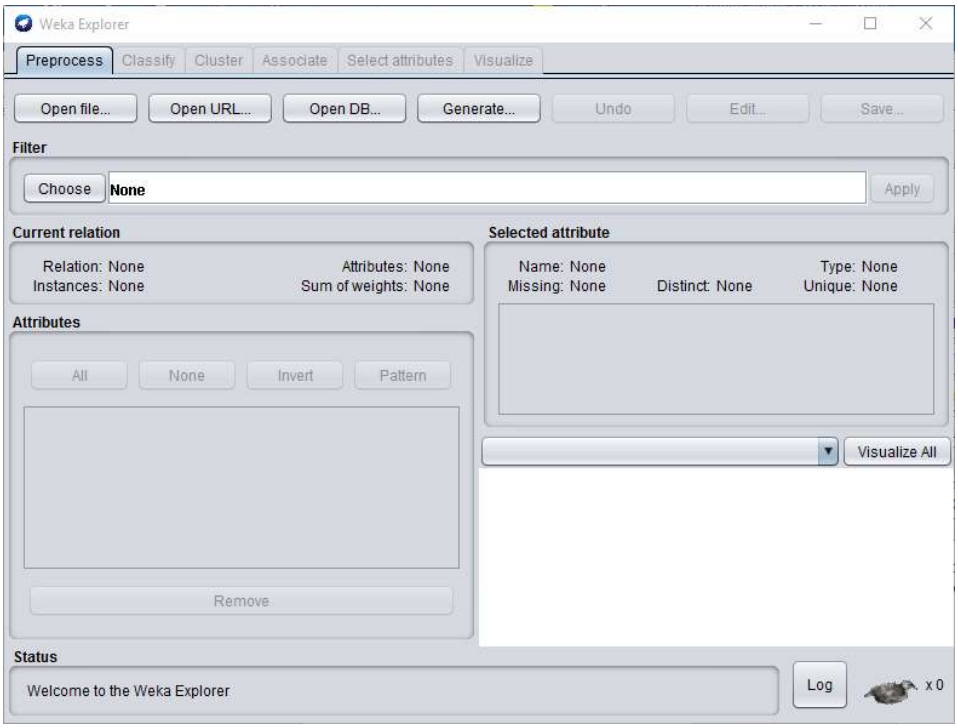
M.I. Gerardo Avilés Rosas

gar@ciencias.unam.mx

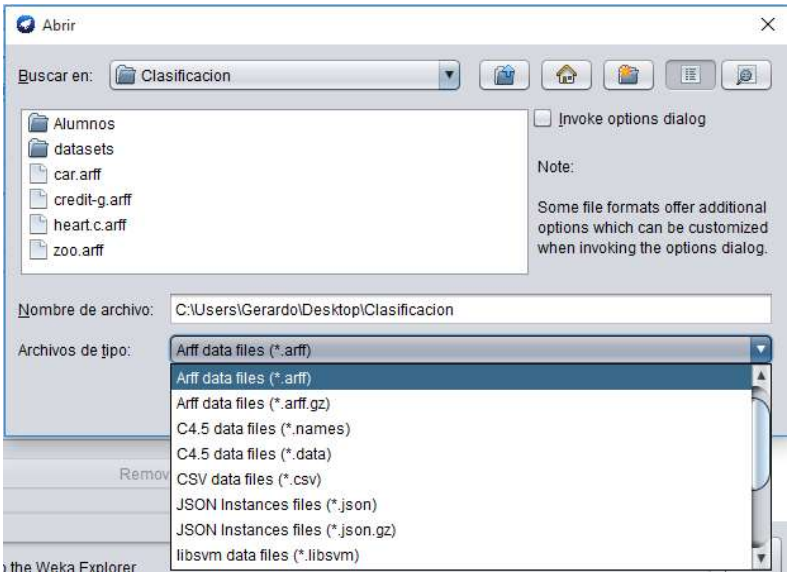
Módulo 6 **Minería de Datos**
Ejemplo de redes neuronales con Weka

1. Iniciamos **Weka**. Una vez que arrancó, vamos a ejecutar la aplicación **Explorer**:

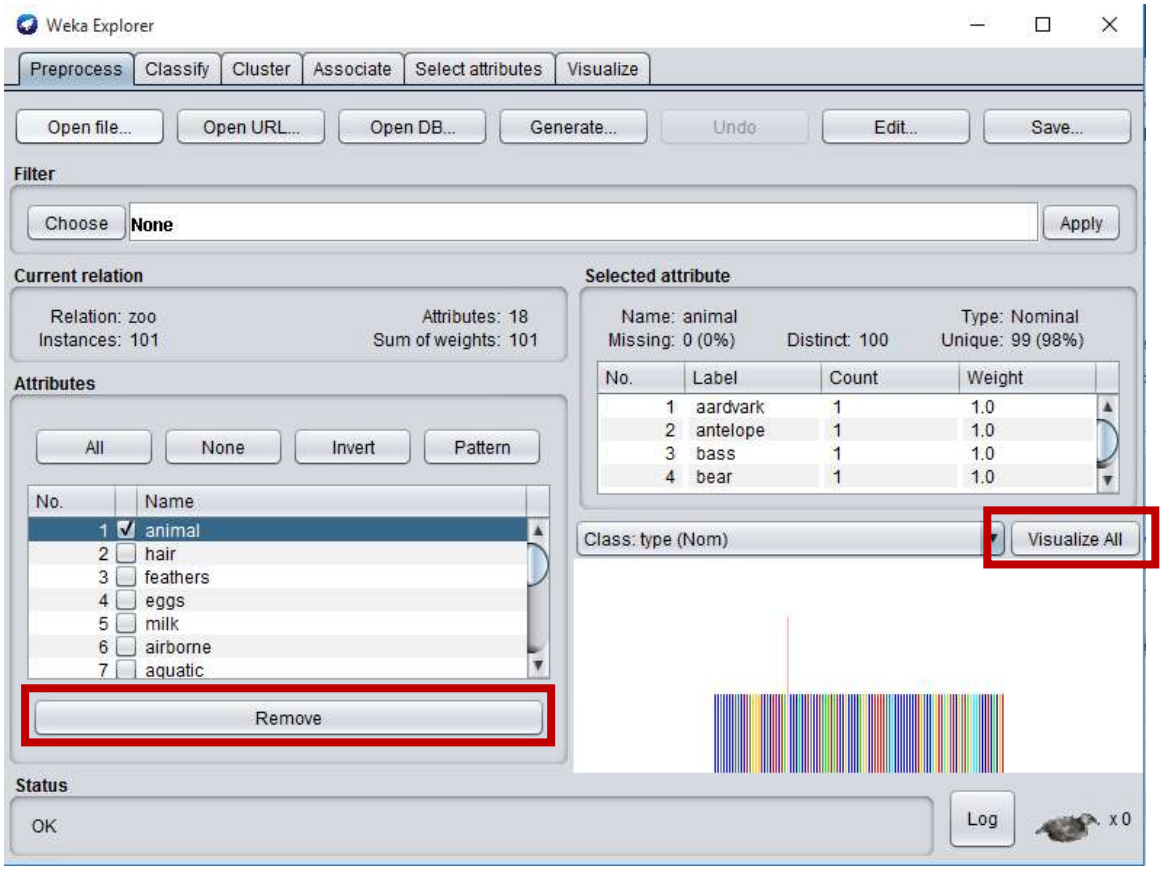
2. Se muestra la siguiente ventana. Como se puede observar, la mayoría de las opciones se encuentran desactivadas ya que no se ha cargado ningún conjunto de datos:



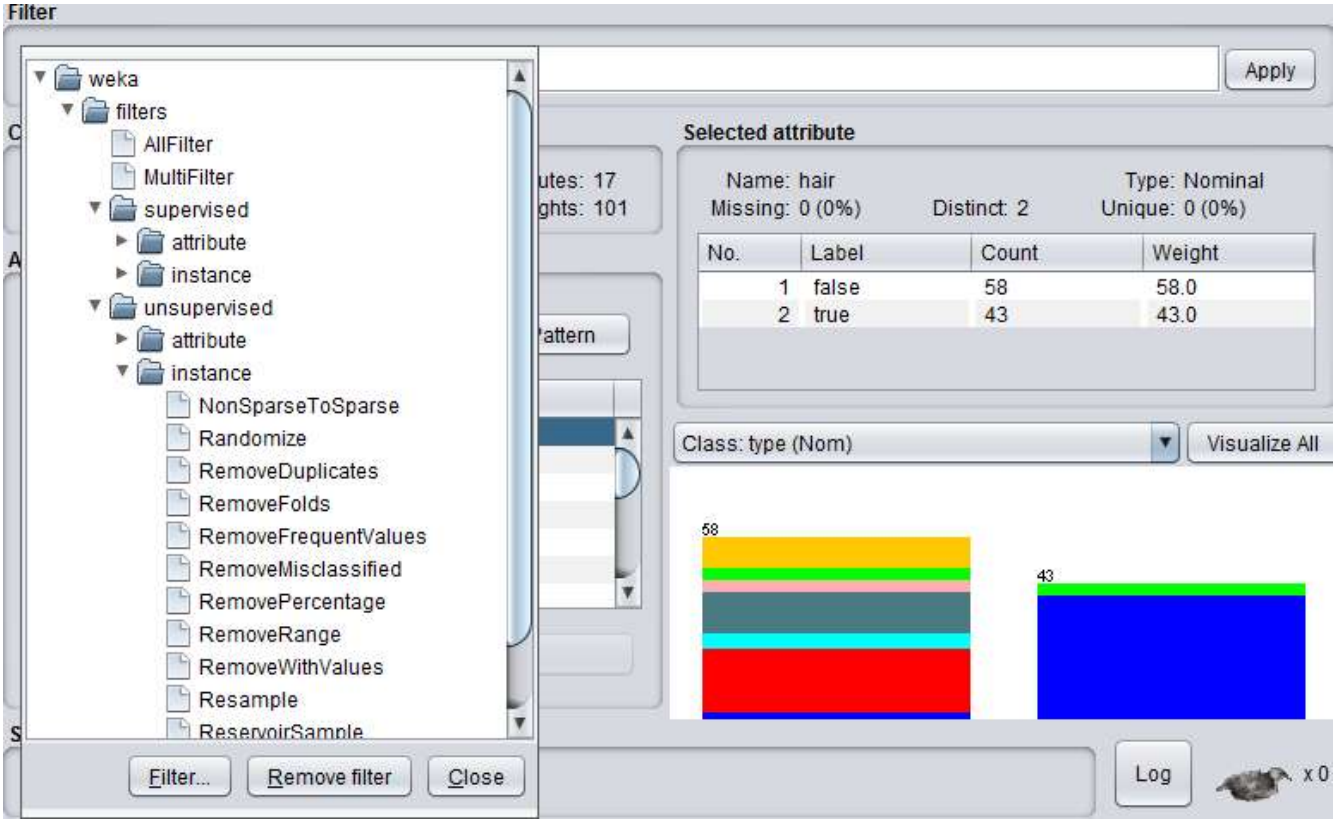
3. Vamos a dar clic en la opción Open file y buscamos el dataset **zoo.arff**, que es un formato nativo de **Weka**:



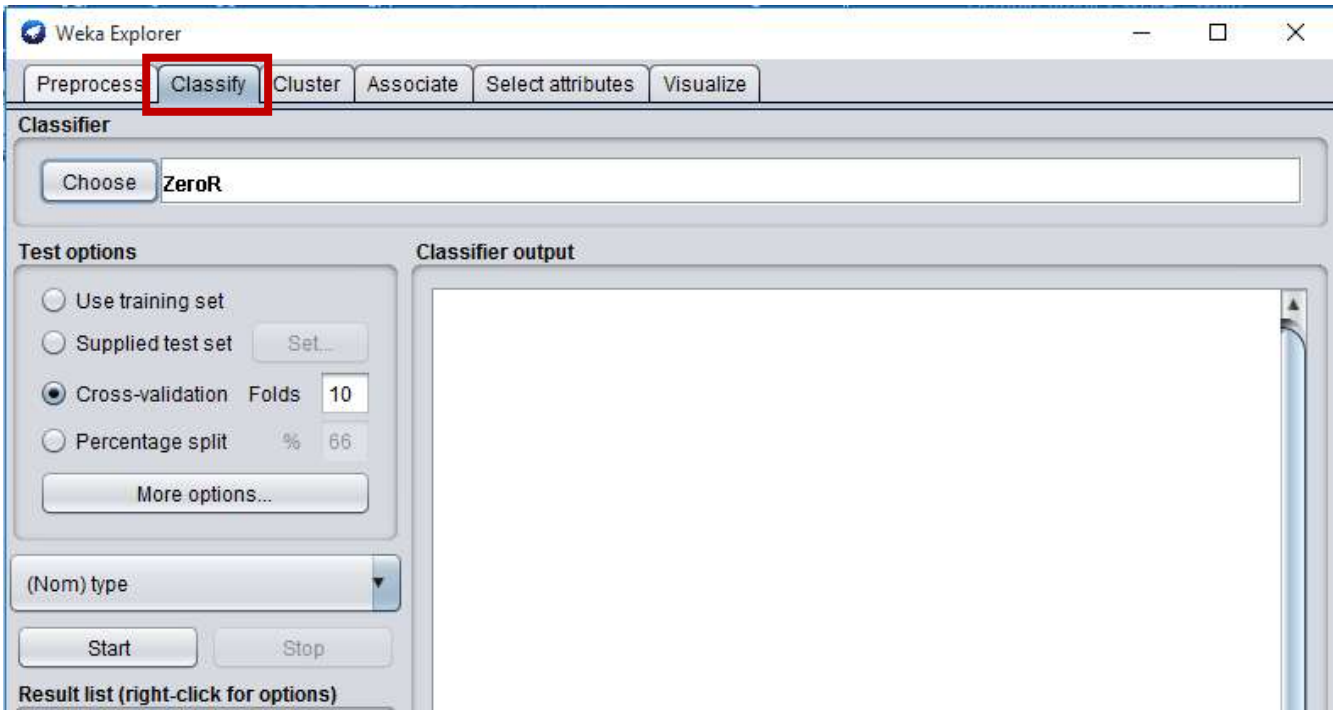
4. Una vez cargado el dataset, vamos a seleccionar el **atributo Animal** y damos clic en la opción **Remove** (se encuentra debajo de la lista de atributos). Esto es para evitar que el árbol de sobreajuste por la cantidad de valores único que tiene (100). No se recomienda agregar al modelo de clasificación atributos que tenga valores del tipo llave primaria, para no crear este efecto:



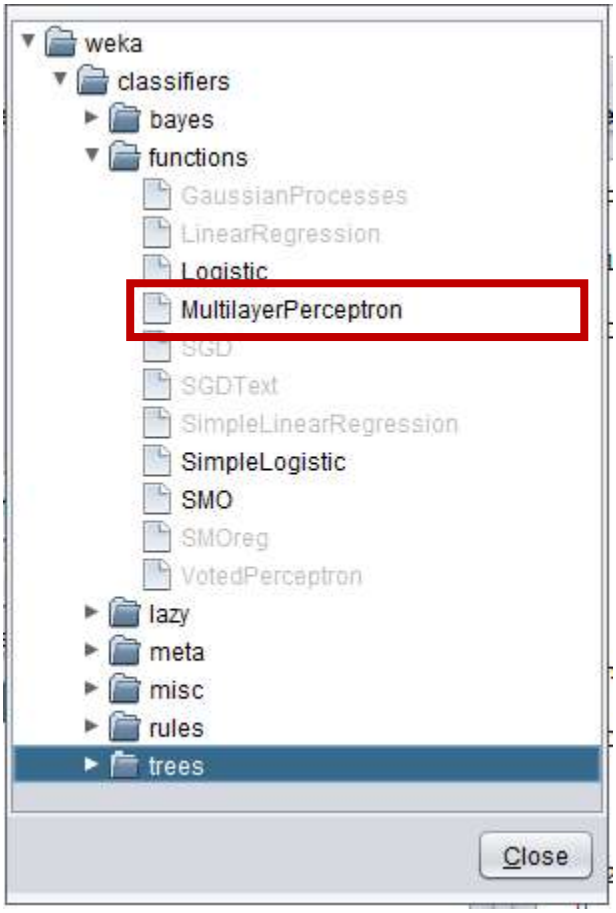
5. Vamos a dejar las variables sin modificar, sin embargo, desde esta perspectiva, tenemos acceso a la sección de **Filtros (Filter)**, que se utilizan para aplicar preprocesamiento de los datos que se tengan actualmente cargados en **Weka**. Los vamos a encontrar en sus versiones supervisados y no supervisados:



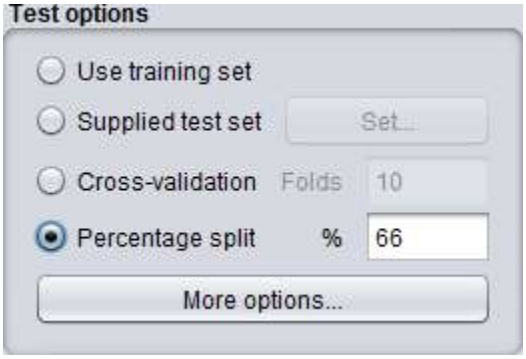
6. Nos vamos a cambiar a la pestaña **Clasificación**:



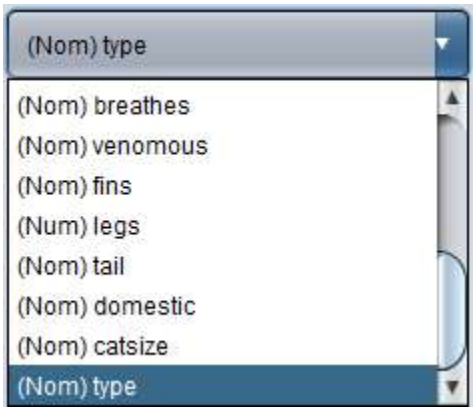
7. Vamos a dar clic en **Choose**, para tener acceso a todos los algoritmos de clasificación. En este caso, iremos a la categoría **functions** y seleccionamos **MultilayerPerceptron**, que es una implementación de las **redes neuronales de alimentación hacia adelante con retropropagación del error** que revisamos en clase:



8. Una vez seleccionado, en la sección de **Test options**, vamos a utilizar la primera opción (**Percentage split**), que corresponde con una **evaluación del modelo**, utilizando la técnica de **Hold – out**. En esta técnica, se hace una evaluación del modelo de clasificación seleccionado, dividiendo el conjunta de tuplas de entrenamiento en dos bloques: para conjunto de entrenamiento se selecciona 2/3 del conjunto original, las tuplas restantes se dejan para la etapa de prueba:



9. Con esto, estamos listos para poder generar el modelo, para esto, vamos a dar clic en el botón **Start**. Debemos asegurarnos que se tenga selecciona la variable objetivo adecuada, es decir, aquella que tiene las etiquetas de clase. Para el ejemplo que vamos a utilizar, tenemos un dataset que tiene un conjunto de características sobre **100 animales** distintos: ***si tienen pelo, plumas, si ponen huevos, si dan leche, si son acuáticos, si son depredadores, si tienen columna vertebral,*** entre otros. Se desea saber, con base en estas características, **¿qué tipo de animal es?**, las opciones son: ***pez, ave, mamífero, réptil, anfibio, insecto o invertebrado***:



10. Una vez que termina el entrenamiento, podemos ver los resultados arrojados por **Weka**. En primer lugar, encontramos información sobre el modelo utilizado, las variables involucradas, sí se aplicó o no un preprocesamiento y la forma de evaluar el modelo:

```
=== Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation:     zoo-weka.filters.unsupervised.attribute.Remove-R1
Instances:    101
Attributes:   17
              hair
              feathers
              eggs
              milk
              airborne
              aquatic
              predator
              toothed
              backbone
              breathes
              venomous
              fins
              legs
              tail
              domestic
              catsize
              type
Test mode:    split 66.0% train, remainder test
```

En segundo lugar, se muestran los detalles de ajustes de pesos y sesgos en cada uno de los nodos que componen la red neuronal:

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

Sigmoid Node 0
    Inputs    Weights
    Threshold    -2.4102443062906773
    Node 7    0.8933250230596422
    Node 8    3.290140161109739
    Node 9    1.1590279038957556
    Node 10    -0.14173202035514082
    Node 11    -3.4215294874941913
    Node 12    -1.3799828562926024
    Node 13    -0.17044746974344052
    Node 14    -3.1311651733871035
    Node 15    1.815857380028766
    Node 16    0.9191211959085643
    Node 17    0.5354590060856959
```

Enseguida encontramos información de la tasa de la precisión, el error de clasificación y estadísticas detalladas sobre la precisión en cada una de las clases:

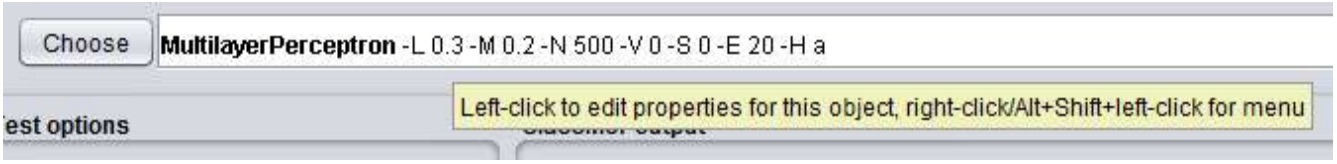
```
Time taken to build model: 1.17 seconds

=== Evaluation on test split ===

Time taken to test model on training split: 0 seconds

=== Summary ===

Correctly Classified Instances          32               94.1176 %
Incorrectly Classified Instances         2                5.8824 %
Kappa statistic                          0.9233
Mean absolute error                      0.0309
Root mean squared error                  0.1207
Relative absolute error                 13.9077 %
Root relative squared error             35.981  %
Total Number of Instances               34

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     mammal
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     bird
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     reptile
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     fish
                 0.000    0.000    0.000      0.000    0.000      0.000    ?         ?         amphibian
                 0.667    0.000    1.000      0.667    0.800      0.789    1.000     1.000     insect
                 1.000    0.065    0.600      1.000    0.750      0.749    1.000     1.000     invertebrate
Weighted Avg.    0.941    0.006    0.965      0.941    0.943      0.941    1.000     1.000
```

Finalmente encontramos la matriz de confusión, que nos da información sobre las tuplas correctamente clasificadas y en el caso de los errores, podemos saber dónde se confundió el modelo:

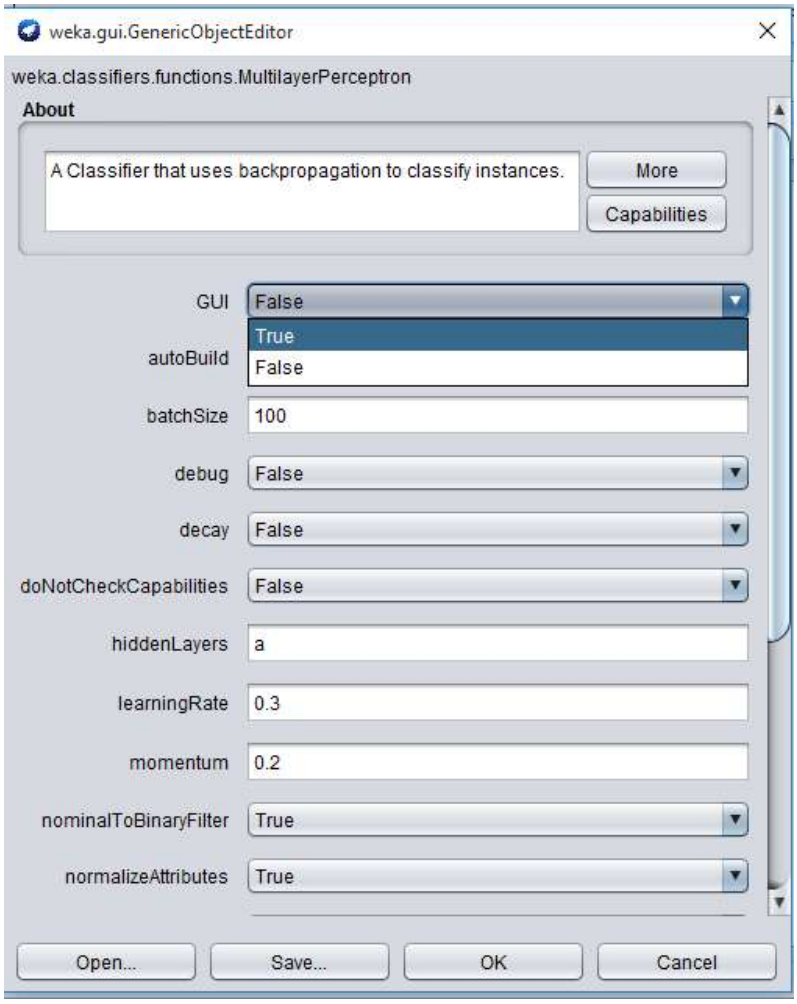
```
=== Confusion Matrix ===

  a  b  c  d  e  f  g   <-- classified as
 13  0  0  0  0  0  0 |  a = mammal
  0  6  0  0  0  0  0 |  b = bird
  0  0  1  0  0  0  0 |  c = reptile
  0  0  0  5  0  0  0 |  d = fish
  0  0  0  0  0  0  0 |  e = amphibian
  0  0  0  0  0  4  2 |  f = insect
  0  0  0  0  0  0  3 |  g = invertebrate
```

Más adelante estudiaremos lo aspectos en cuanto evaluación.

11. Finalmente, podemos ver la topología de la red generada. Para activa la visualización por medio de interfaz gráfica, debemos dar clic sobre el nombre del algoritmo:



En la ventana que se muestra, cambiamos el valor de la primera propiedad, GUI, de FALSE a TRUE:



12. Volvemos a generar el modelo y se mostrará la red neuronal: