Subject

A Multilayer Perceptron for a classification task (neural network): comparison of TANAGRA, SIPINA and WEKA.

When we want to train a neural network, we have to follow these steps:

- Import the dataset;
- Select the discrete target attribute and the continuous input attributes;
- Split the dataset into learning and test set;
- Choose and parameterize the learning algorithm;
- Execute the learning process;
- Evaluate the performance of the model on the test set.
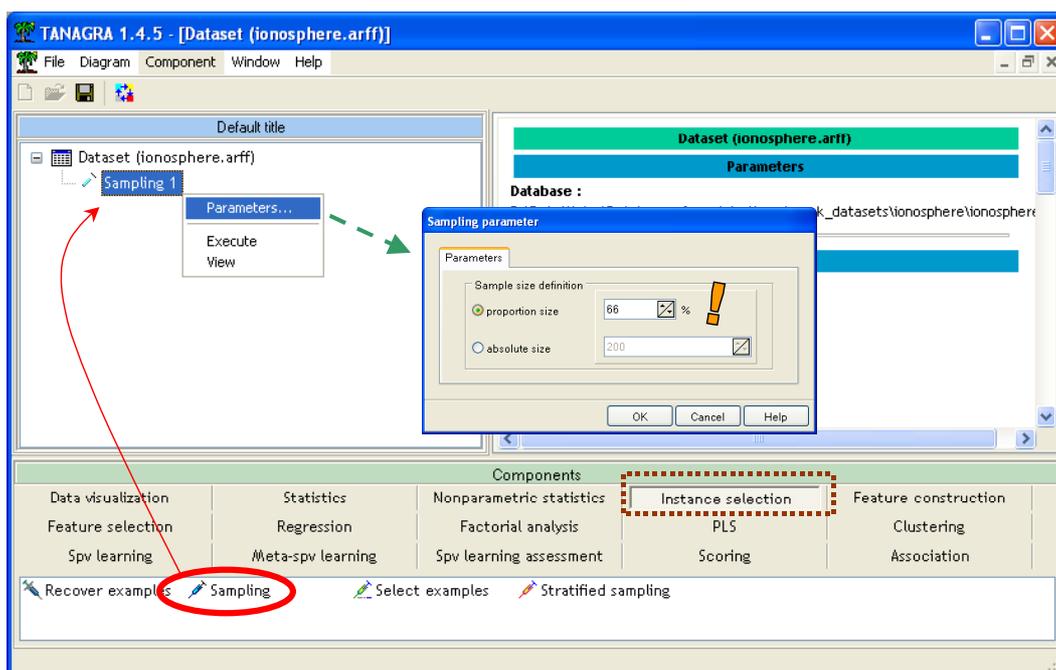
Dataset

We use the IONOSPHERE.ARFF from UCI IRVINE (ARFF is the WEKA file format). The attributes are standardized. There are 351 examples, 33 continuous descriptors, and a binary class attribute.

# Training a neural network with TANAGRA

Dataset importation

We click on the FILE/NEW menu in order to create a new diagram and import the dataset.

## Splitting the dataset into learning and test set

In the next step, we have to split the dataset into a learning set, which is used for the computation of the neural network weights, and a test set, which is used for the model performance evaluation.

We add the SAMPLING component; we use 66% of examples for the learning phase.
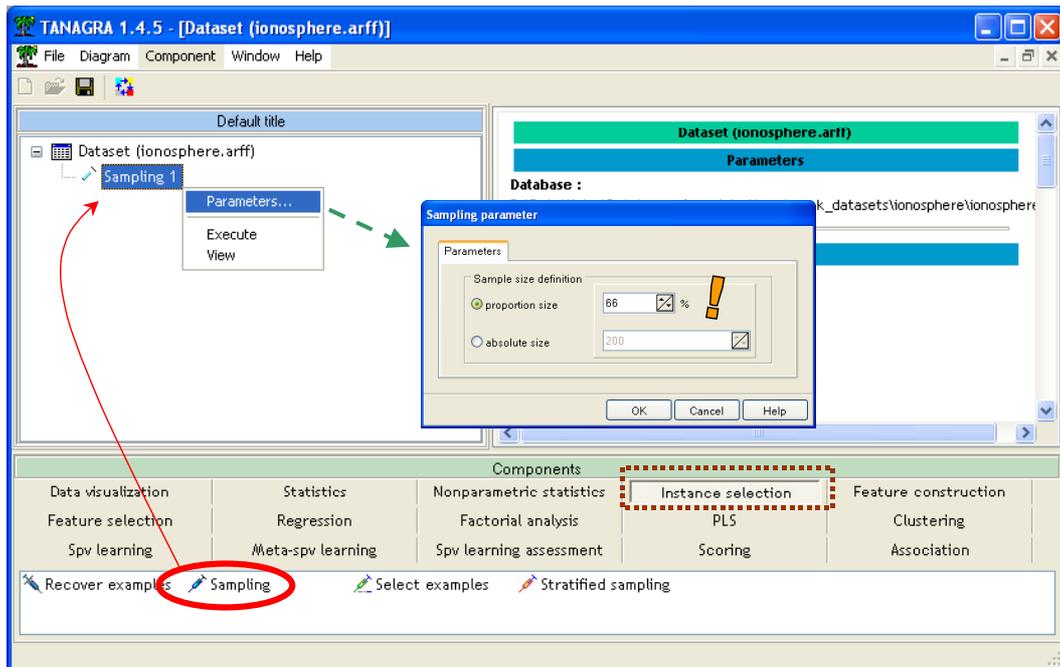
Select the class and the predictive attributes

We add the DEFINE STATUS in the diagram, we use the shortcut in the toolbar, we set CLASS as TARGET, and all continuous attributes as INPUT.
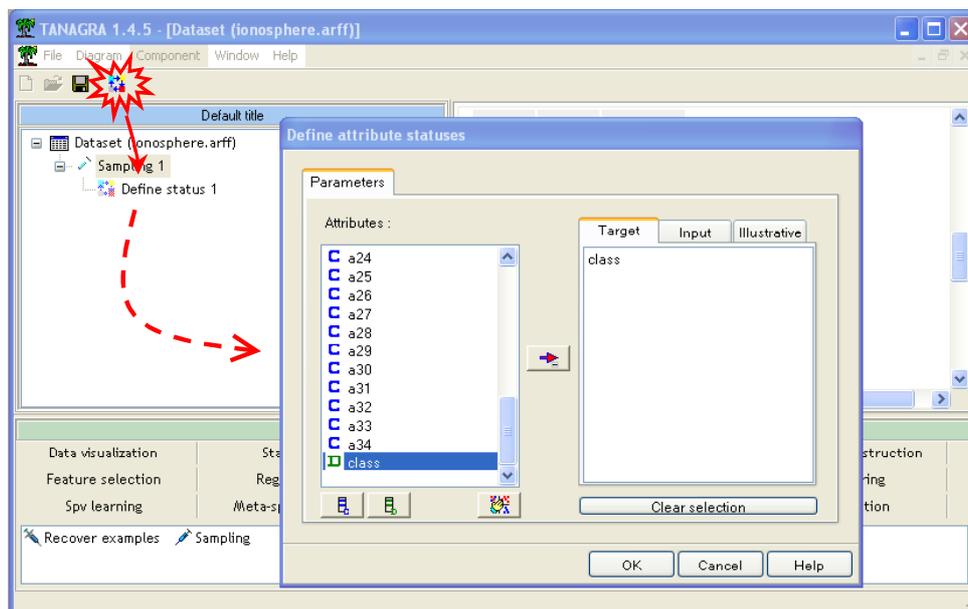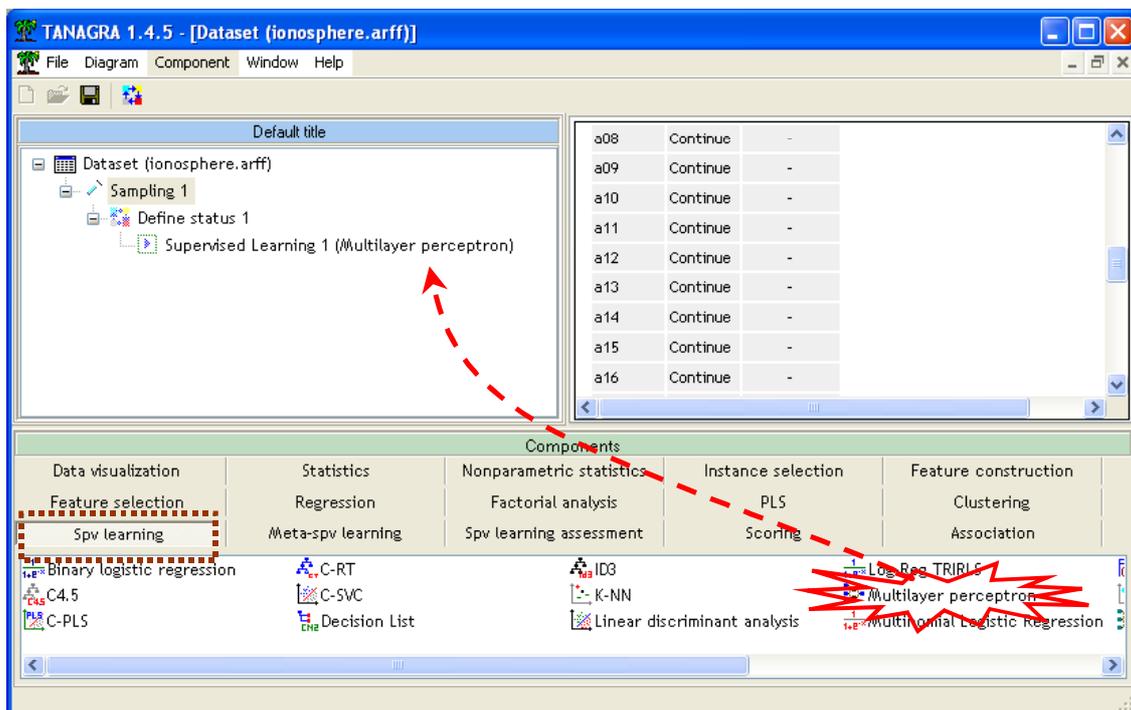
Learning algorithm

We want to add a Multiplayer Perceptron in the diagram. In the first step, we add a learning implementation algorithm (SUPERVISED LEARNING from the META-SPV LEARNING TAB).
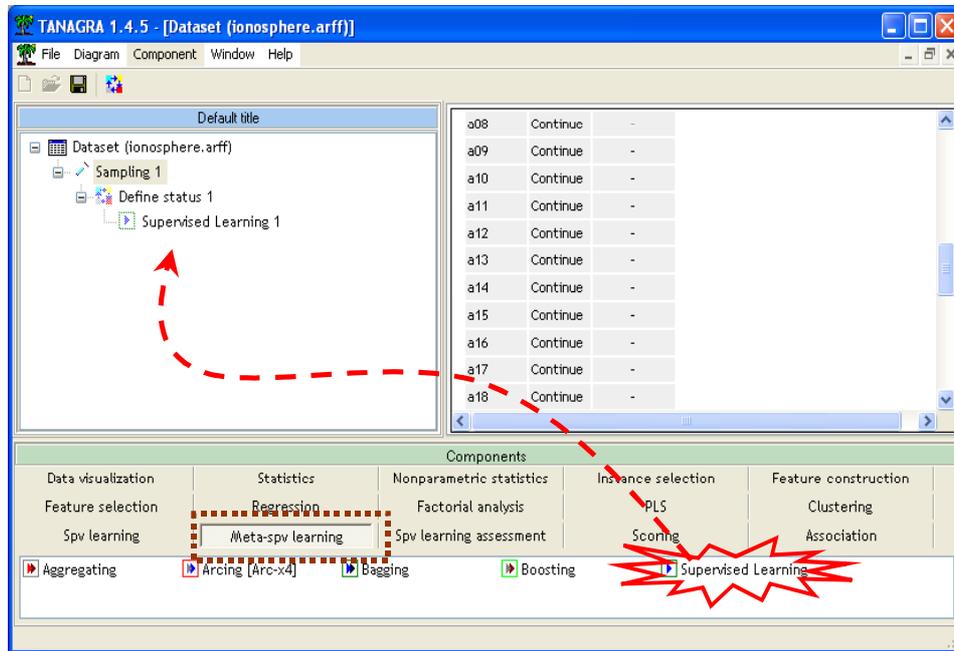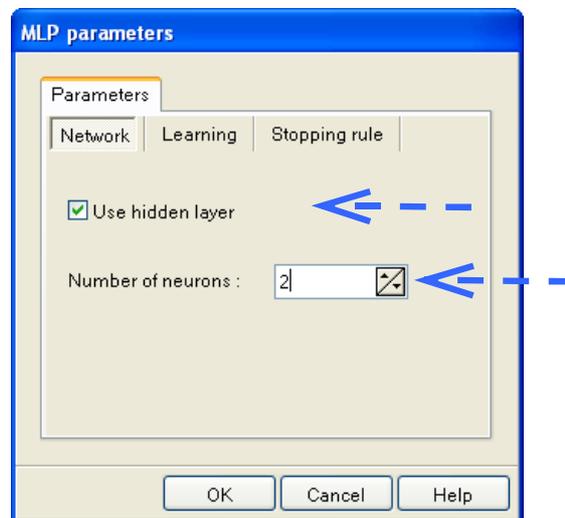

In the second step, we embed in the first one, a learning method algorithm i.e. the MULTILAYER PERCEPTRON from the SPV LEARNING tab.
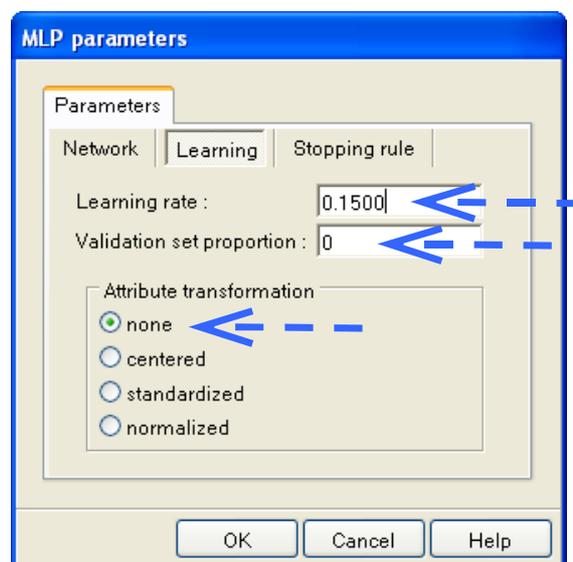
Setting the parameters

There are several kinds of parameters. The first ones are the neural architecture parameters (NETWORK tab). We use a hidden layer with two neurons.

The next ones are the learning parameters (LEARNING tab). We set the LEARNING RATE to 0.15.

We can define a validation set. This sub-sample enables to compute the error rate on a part of the learning set which is not used for the computation of weights. In this analysis, we do not use a validation set – VALIDATION SET PROPORTION = 0.15.

Last, because we have already standardized descriptors, we set ATTRIBUTE TRANSFORMATION to NONE.

In the last tab, STOPPING RULE, we set the parameters which enables to stop the learning process: MAX ITERATION is the max number of epochs; ERROR RATE THRESHOLD enables to stop the learning process if the resubstitution error rate is lower than this threshold.

It is possible to stop the learning phase when we note a stagnation of the validation error rate on GAP TEST STAGNATION epochs. But it is not a very efficient option, check only this option if you are confident about the behavior of neural network.
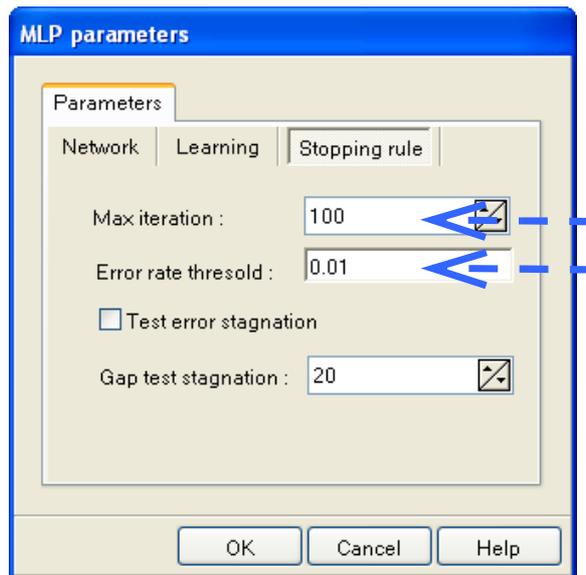
Reading the results

We select the VIEW menu, the weights are computed and a new window appears.

Supervised Learning 1 (Multilayer perceptron)

Parameters

MLP architecture	
Use hidden layer	yes
Neurons in the hidden layer	2
Learning parameters	
Validation set proportion	0.00
Learning rate	0.15
Attribute transformation	none
Stopping rule	
Max iteration	100
Error rate thresold	0.0100
Verifiy error stagnation	no

Results

Classifier performances

Error rate		0.0260		
Values prediction		**Confusion matrix**		
Value	Recall	1-Precision		good	bad	Sum

Value	Recall	1-Precision		good	bad	Sum
good	1.0000	0.0403	good	143	0	143
bad	0.9318	0.0000	bad	6	82	88
			Sum	149	82	231

In the first part of the window, we can see a summary of the network parameters and the resubstitution confusion matrix (0.026). We know that this estimation of the error rate is often highly optimistic.

In the second part of the window, the weights of the network are displayed. We can copy and paste theses values in a spreadsheet.

The ATTRIBUTE CONTRIBUTION part computes the error rate of the Perceptron when we remove one attribute. TANAGRA compares this error rate with the error rate of the whole model in order to evaluate the importance of each attribute in the prediction.

Attribute contribution

Excluded attribute	Error rate	Difference	Statistics
none	0.0260	-	-
a01	0.1169	0.0909	8.6868
a03	0.0433	0.0173	1.6546
a04	0.0519	0.0260	2.4819
a05	0.0563	0.0303	2.8956
a06	0.0563	0.0303	2.8956
a07	0.0260	0.0000	0.0000

For instance, we see in this table that the error rate of the whole Perceptron is 0.026. If we remove the "a01" attribute -- i.e. we use the average of the attribute instead of the true values -- the error rate becomes 0.1169. The difference is 0.0909, if we use a statistical comparison between these two proportions; the t-value is 8.6868, it seems highly significant.

In the last part of the window, the ERROR RATE DECREASING shows the error rate during the learning process. We can copy and paste this table in a spreadsheet and create a graphical representation of the error rate progression.

Error rate decreasing

Error evaluation			
Epoch	Train err	Val. err	MSE
1	0.3810	1.0000	54.4613
6	0.0823	1.0000	23.2339
11	0.0736	1.0000	16.4484
16	0.0519	1.0000	13.4766
21	0.0390	1.0000	11.7270
26	0.0346	1.0000	10.5781
31	0.0346	1.0000	9.8356
36	0.0303	1.0000	9.3085
41	0.0303	1.0000	8.9049
46	0.0303	1.0000	8.5669
51	0.0303	1.0000	8.1741
56	0.0260	1.0000	7.6697
61	0.0260	1.0000	7.4119
66	0.0260	1.0000	7.2302
71	0.0260	1.0000	7.0859
76	0.0260	1.0000	6.9666
81	0.0260	1.0000	6.8660
86	0.0260	1.0000	6.7802
91	0.0260	1.0000	6.7064
96	0.0260	1.0000	6.6424
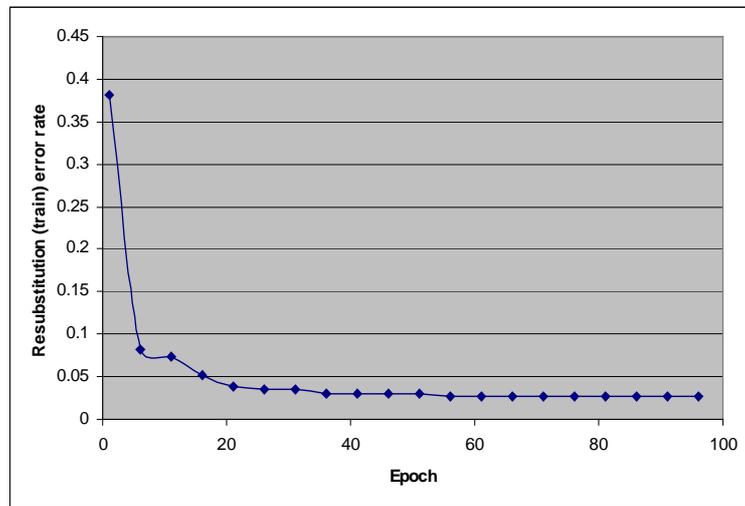
Evaluate the network on a test set

We want to compute the test error rate on the 120 remaining examples. We add again the DEFINE STATUS component in the diagram. We set CLASS as TARGET, the prediction of the neural network (PRED_SPVINSTANCE_1) as INPUT.
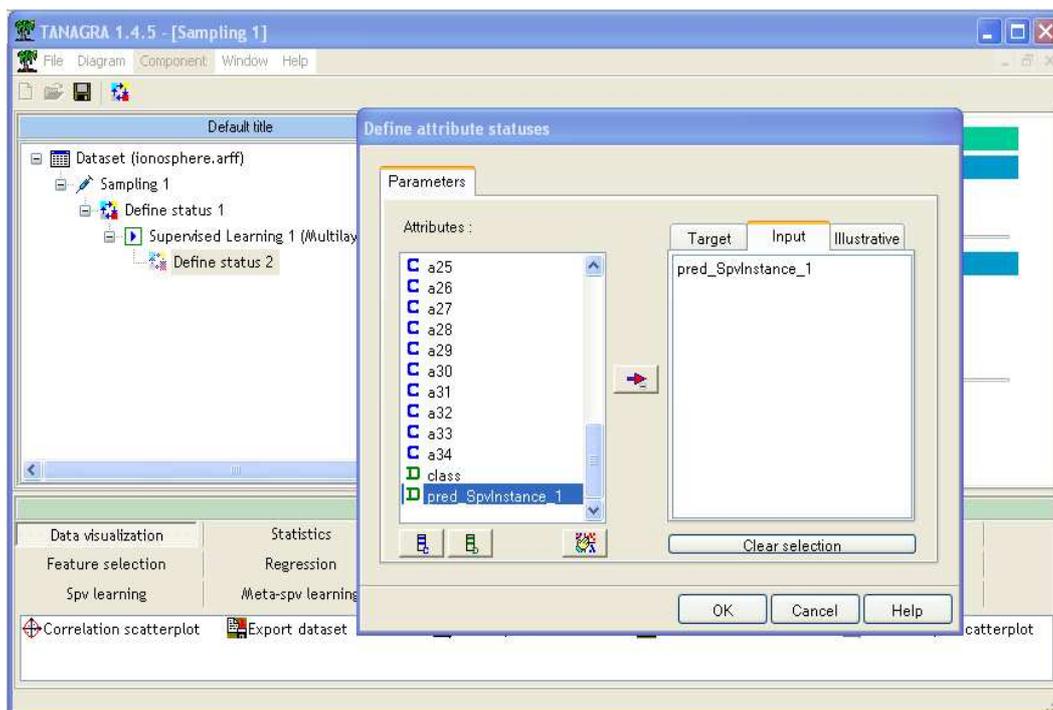
Then, we add the TEST component (SPV LEARNING ASSESMENT tab). The error rate must be computed on the test (unselected examples) set.
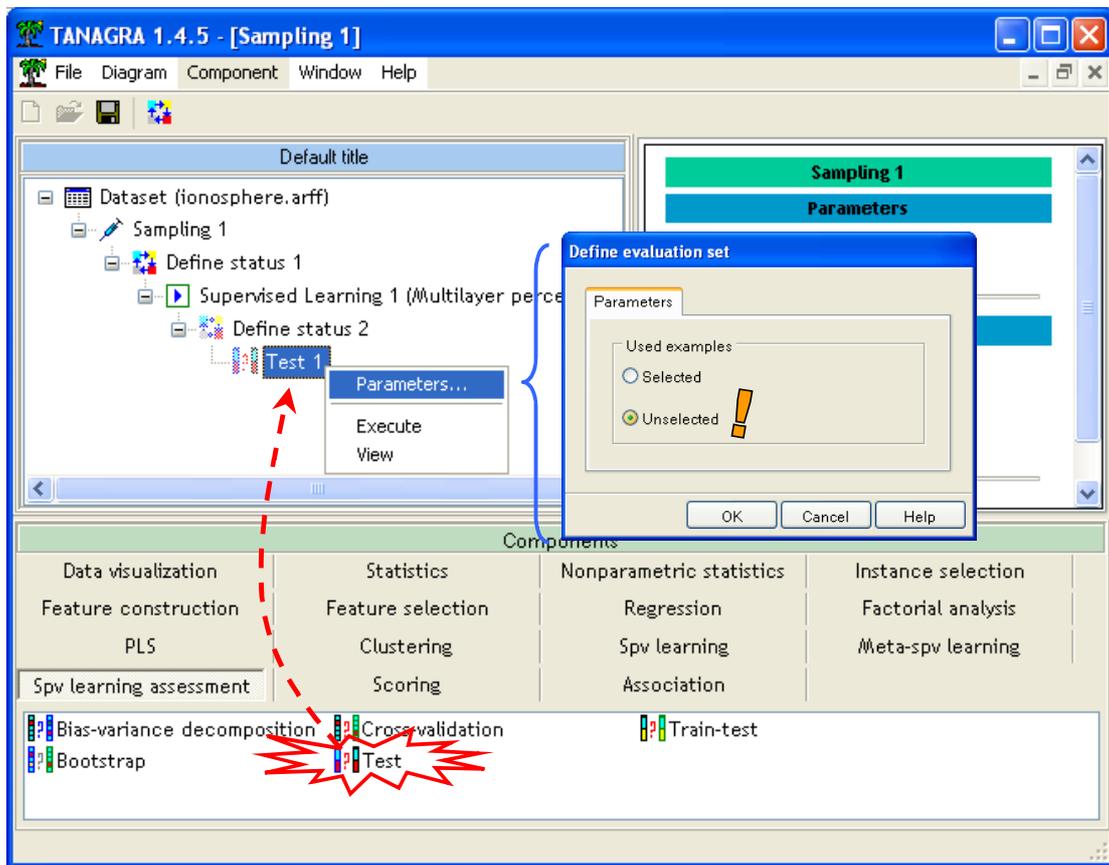


We click on the VIEW menu; the test error rate is 0.125.

Modifying the network parameters

We can improve the power of the neural network when we modify the number of neurons in the hidden layer. We set this parameter to 10. A priori, we should obtain a more efficient network.

We click again on the VIEW menu. The resubstitution error rate is 0.0206.

When we click on the VIEW menu of the TEST component, the test error rate is 0.1083.
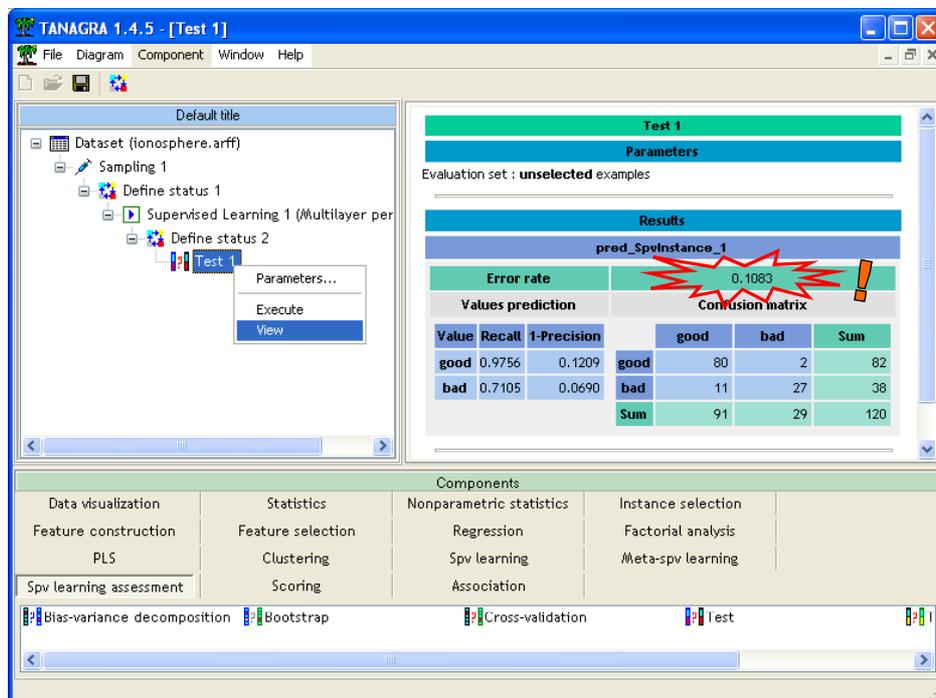
We have a small test set, the results suffers of a strong variability. This difference is not really significant. We have tried some other algorithms such as Linear Support Vector Machine or Linear Discriminant Analysis. We see in the following screenshot the accuracy on the same test set.
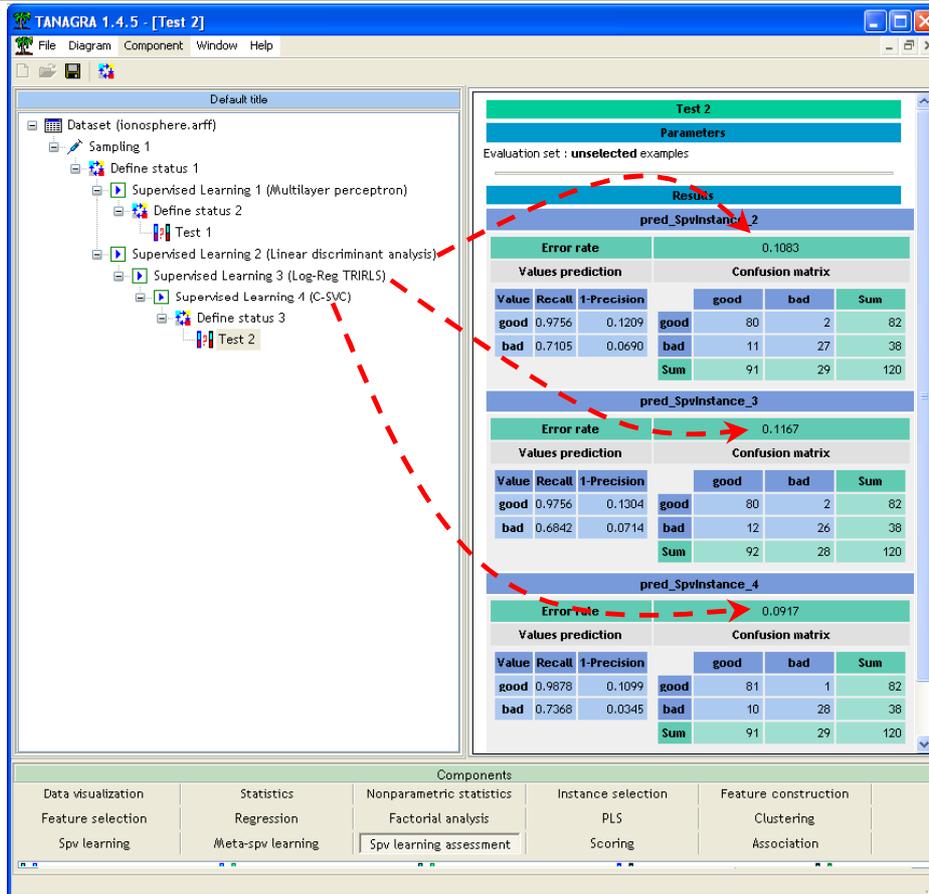
Training a neural network with SIPINA

Importing the dataset

In order to import the dataset, we click on the FILE/OPEN menu.
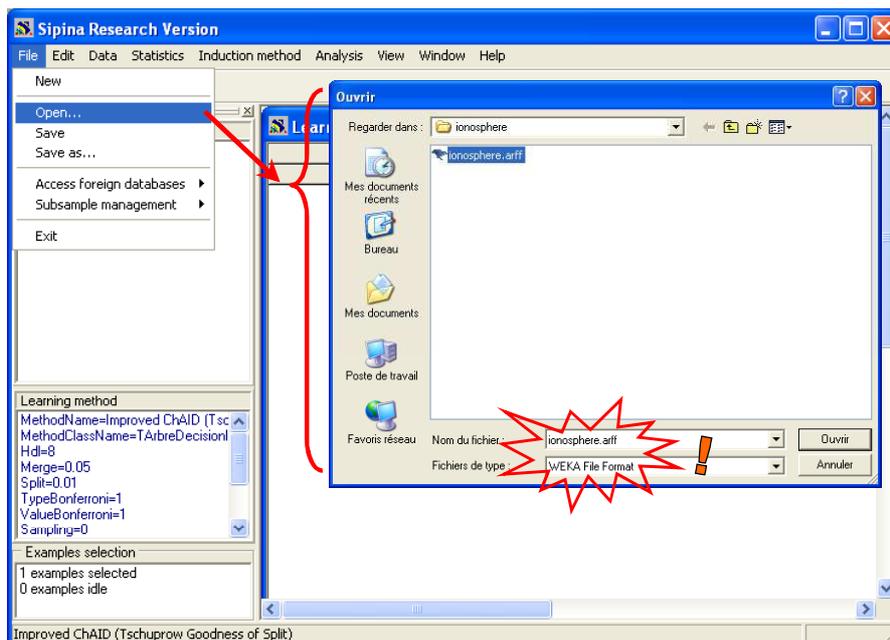
Splitting the dataset

We want to use the 66% of the dataset as learning set. We select the ANALYSIS / SELECT ACTIVE EXAMPLES menu, we select the RANDOM SAMPLING option.
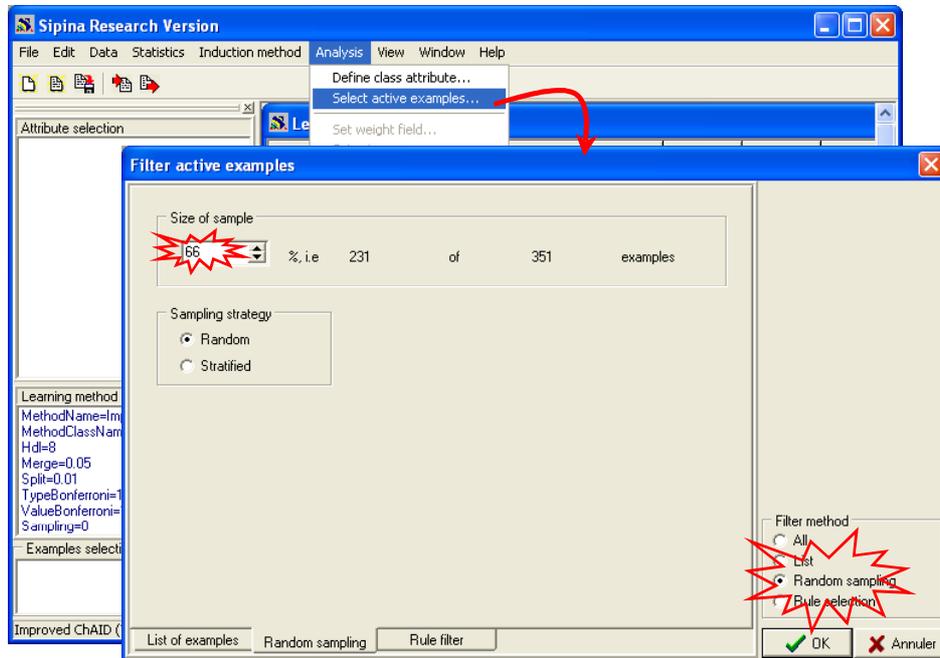
The subsets size appears in a window.
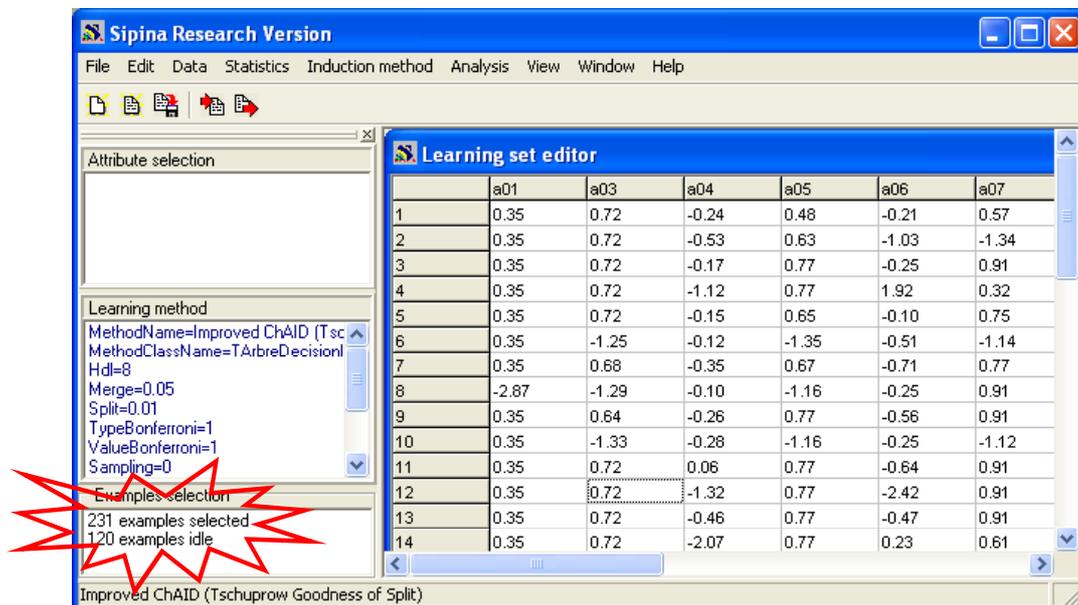
Defining the class and the predictive attributes

We click on the ANALYSIS / DEFINE CLASS ATTRIBUTE menu in order to define the role of attributes. We use drag-and-drop in order to define the TARGET and the INPUT attributes.
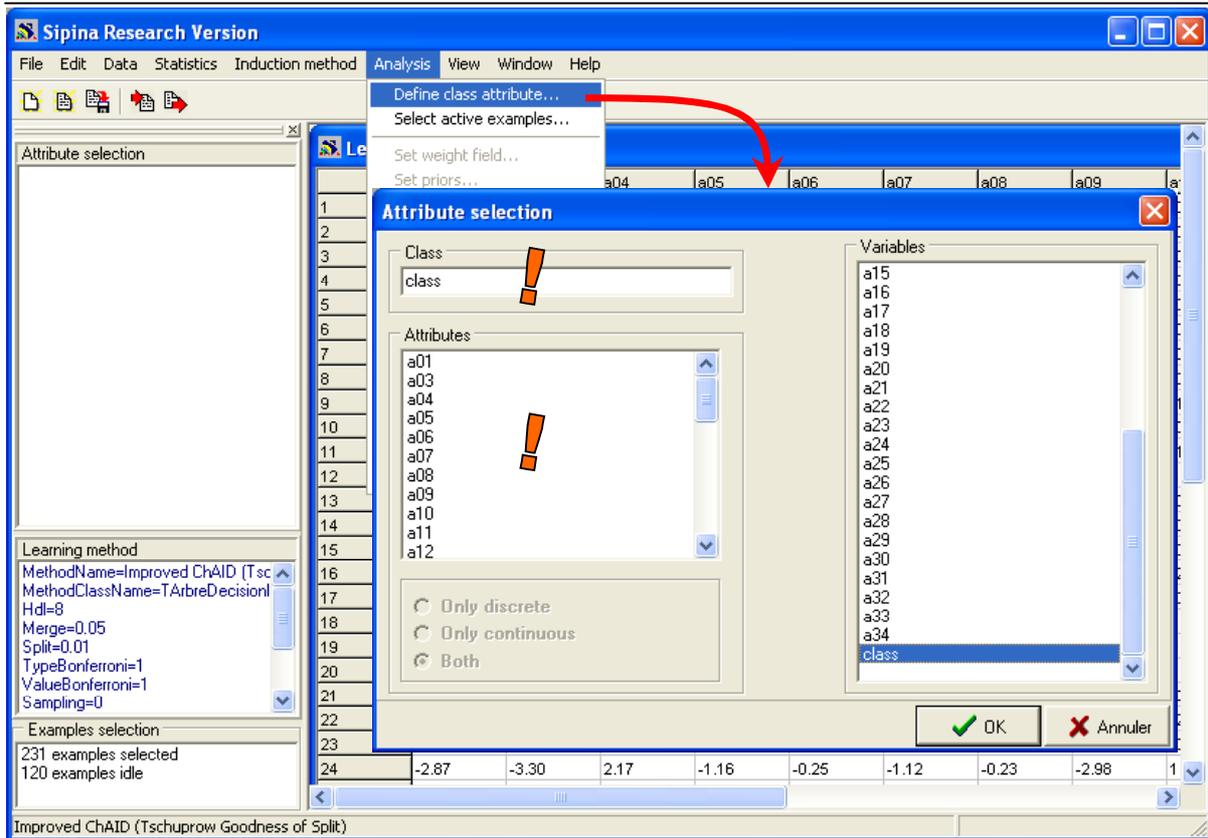
The attributes selection appears on the left part of the window. The type of the attributes is displayed.
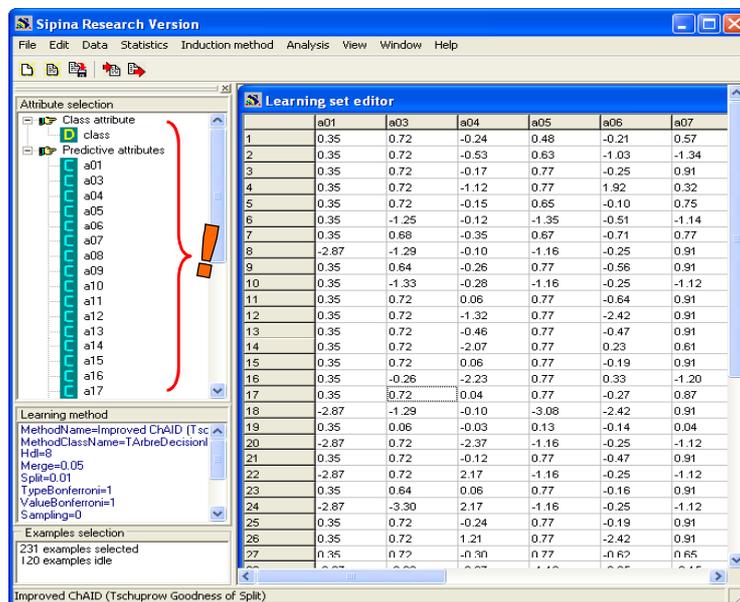
Learning algorithm and parameters settings

The INDUCTION METHOD / STANDARD ALGORITHM menu enables us to choose the learning algorithm. We select the NEURAL NETWORK tab and click on MULTILAYER PERCEPTRON method.
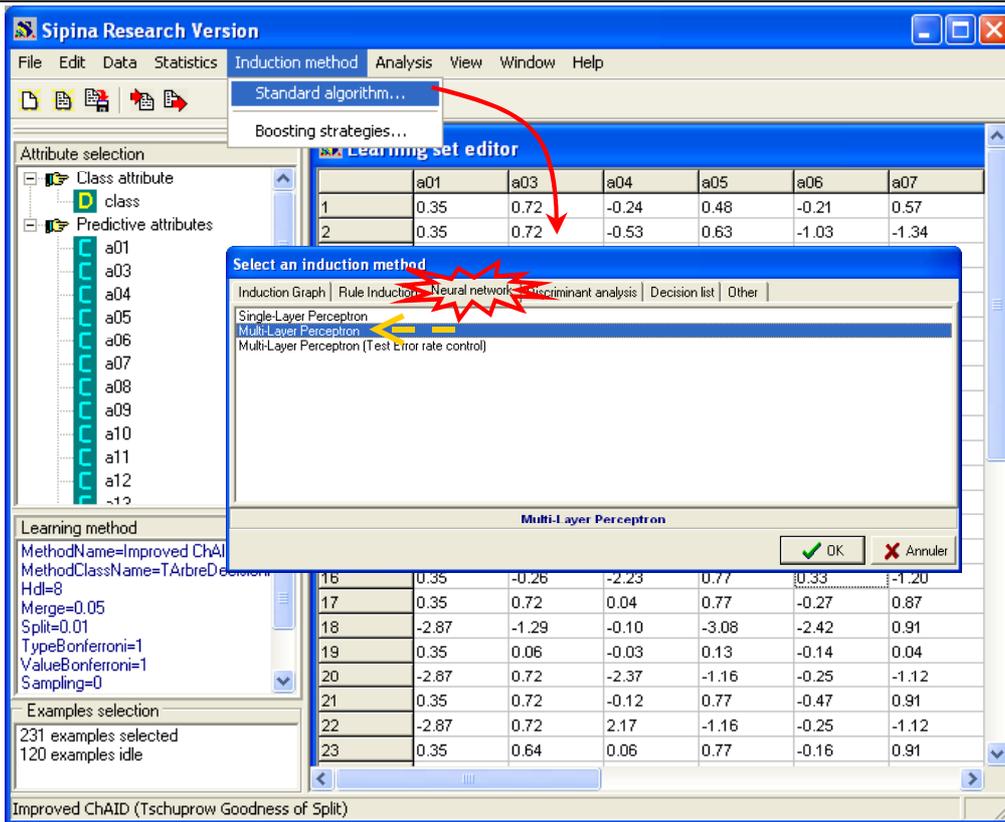
When we click on the OK button, a new dialog box appears. We can set the architecture of the perceptron and the training parameters.
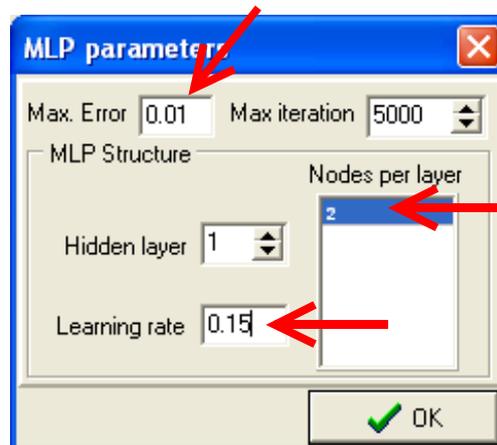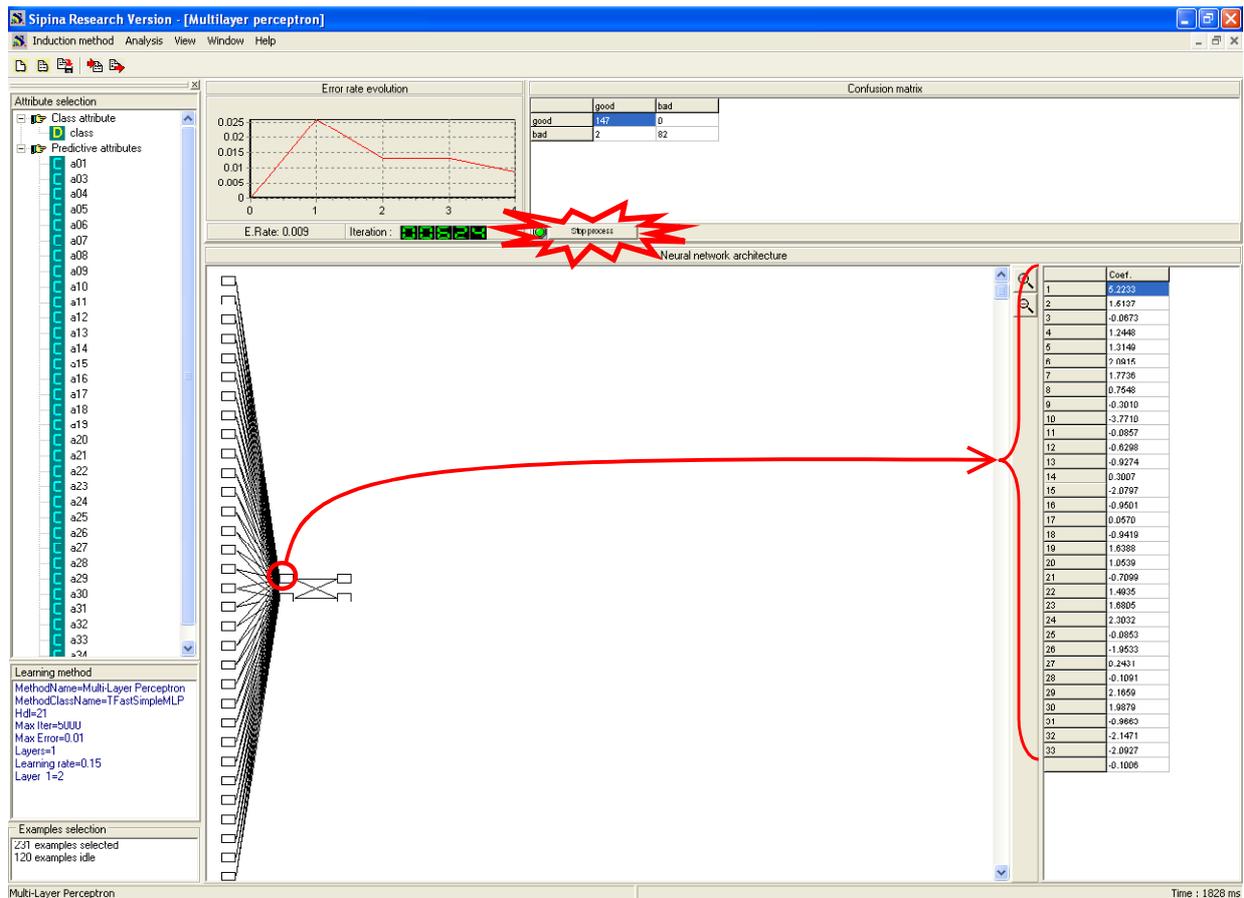

We note that we choose a high MAX ITERATION (5000). That does not matter because we can view the error rate decreasing and interactively stop the learning process in SIPINA.
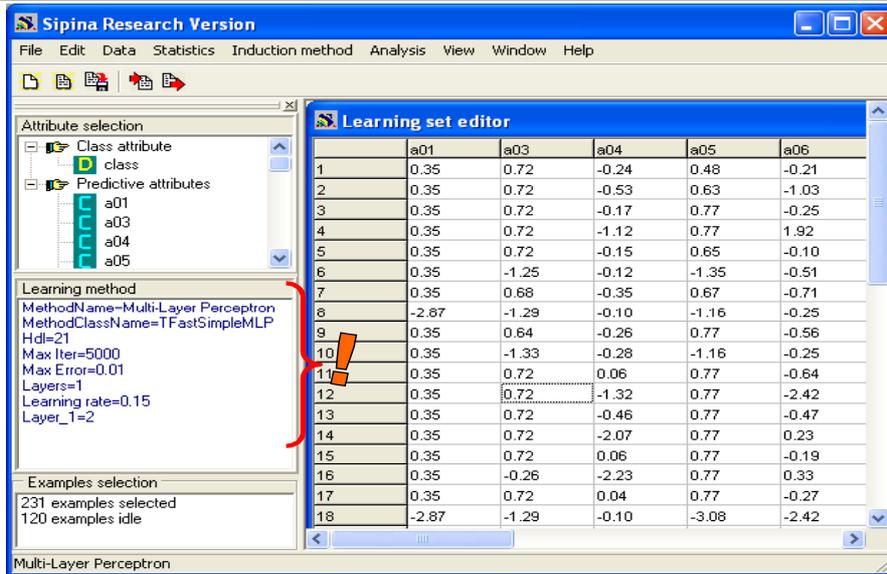
Learning process

We select the ANALYSIS / LEARNING menu. A new window appears, we can follow the error rate progression. A STOP button enables us to stop the processing.

Error rate evolution shows the error rate progression, we see that we obtain an error rate of 0.009 at the iteration 624 . The confusion matrix is at the right part of the window.

The **STOP PROCESS** button is very important. We can stop the processing when we think that we cannot obtain a significant improvement in the remaining iterations.

In the bottom part of the window, when we select a neuron, the associated weights are displayed.

Test error rate

In order to apply the prediction model on the test set, we click on the ANALYSIS / TEST menu.
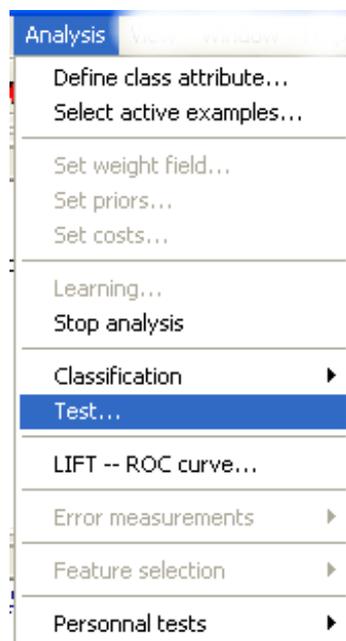
In the subsequent dialog box, we set the following option.
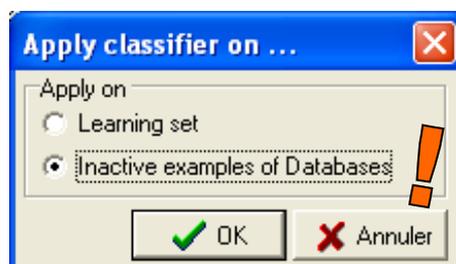
The confusion matrix appears in a new window. The test error rate is 0,0917.

Using a validation set

We can follow the learning process in SIPINA; the utilization of the validation set is more interesting. We can stop the learning process when the validation error rate does not decrease. The learning set is thus split into two parts: the first, says "training set", is used for the computation of the weights of the network; the second, says "validation set", is used for a "honest" evaluation of the error rate.

We close all the windows the WINDOW / CLOSE ALL menu. In order to include the utilization of a validation set in the learning process, we select a new algorithm: INDUCTION METHOD / STANDARD ALGORITHM menu, MULTILAYER PERCEPTRON (TEST ERROR RATE CONTROL) option.
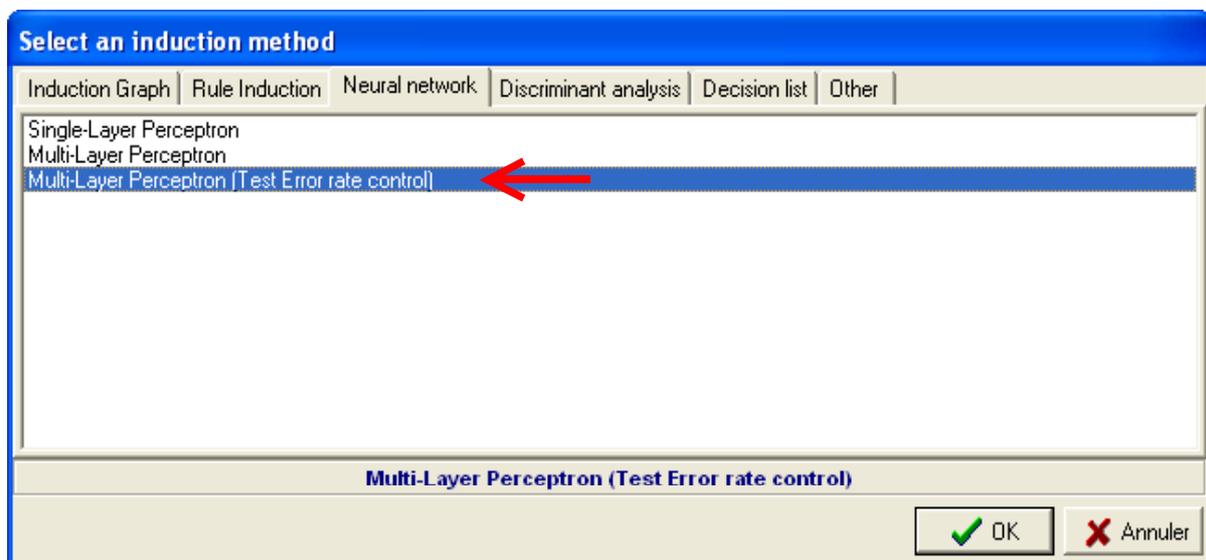
The learning set size is 231. We set 70% of them as a training set (70% of 231 = 161 examples), and the remaining as validation set (231 – 161 = 70 examples).
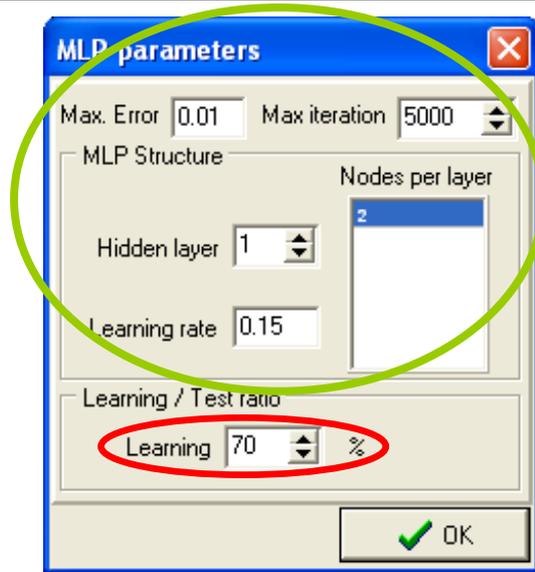
We click on the ANALYSIS / LEARNING menu in order to execute the learning process. Two curves appear now in the chart. In some cases, the validation error rate may increase when we have overfitting.

Validation error rate

Training error rate
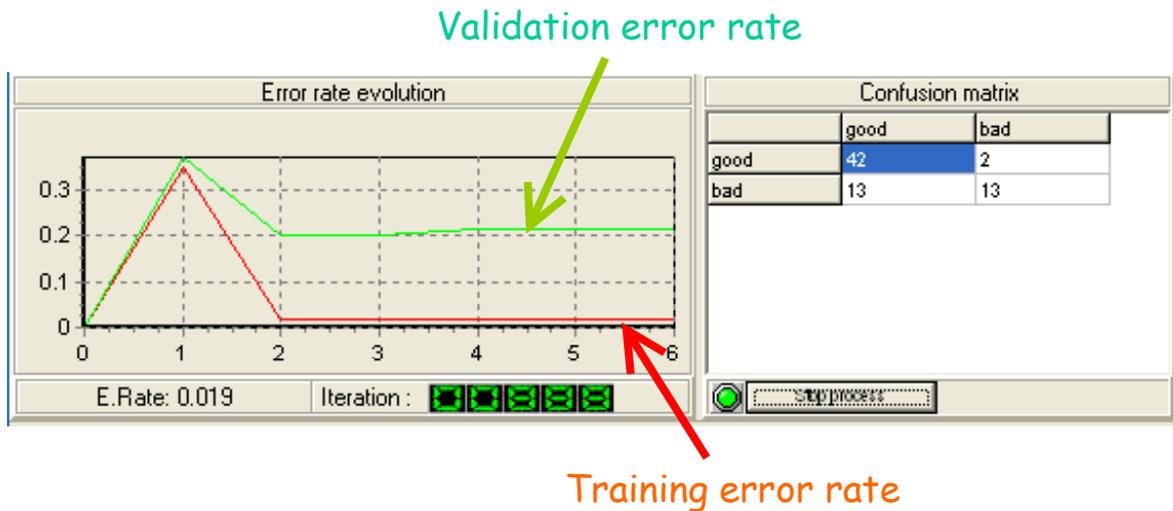
The confusion matrix on the test set gives the following results (ANALYSIS / TEST menu).
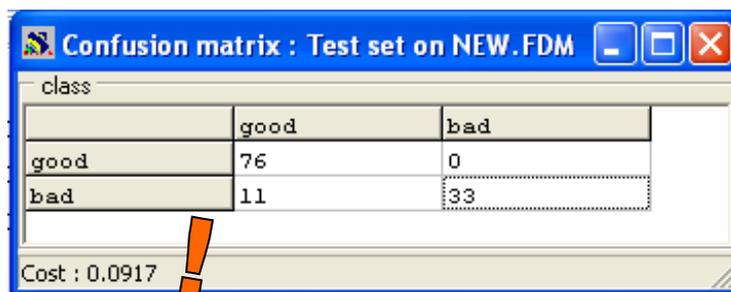
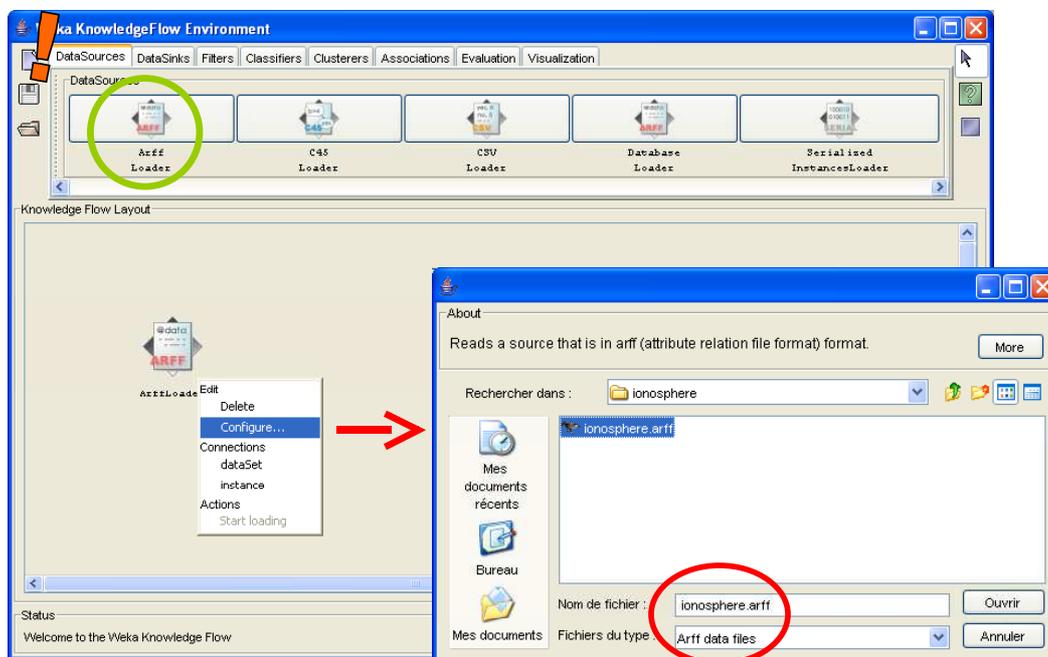# Training a neural network with WEKA

When we execute WEKA (http://www.cs.waikato.ac.nz/ml/weka/), a dialog bow appears, which allows us to choose the execution mode of the software. We select the **KNOWLEDGE FLOW** mode. We have used the **3.5.1** version in this tutorial.
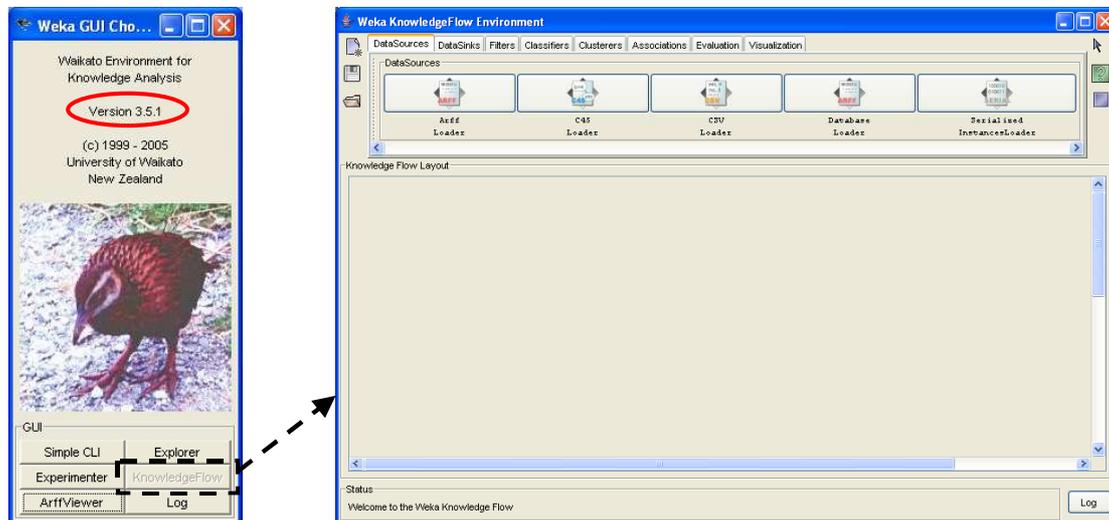
Importing the dataset

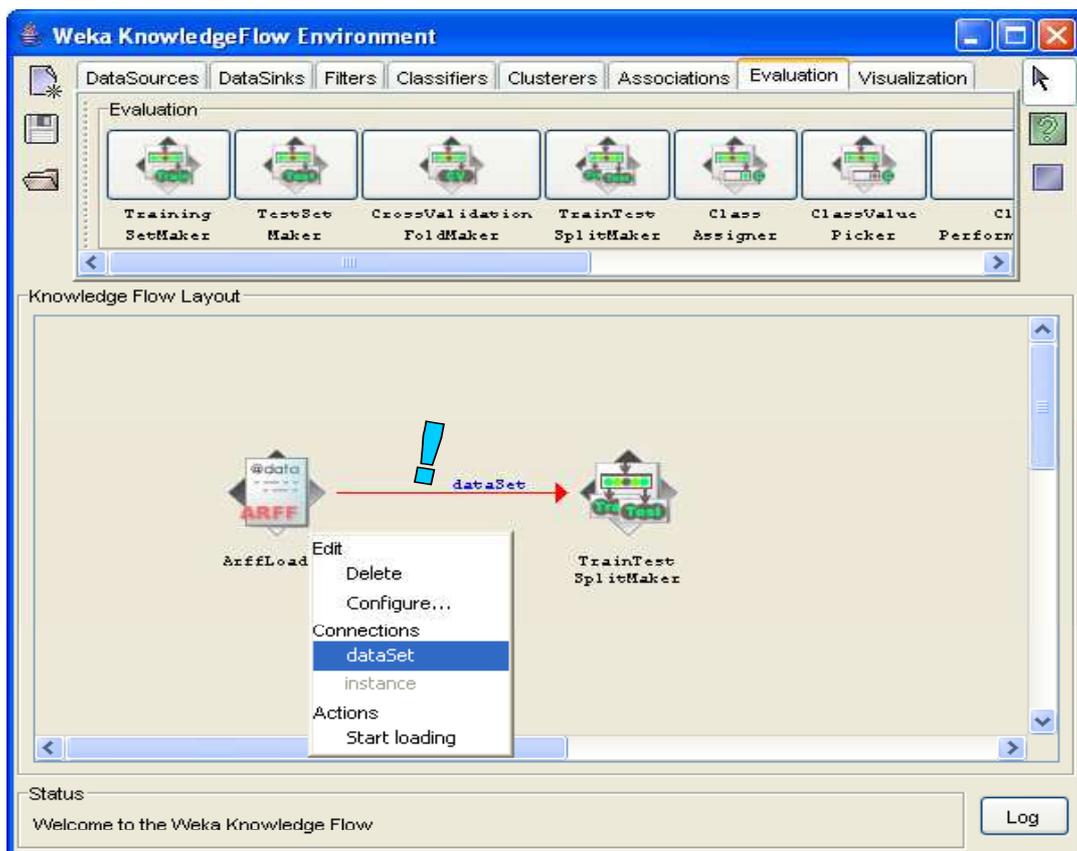The ARFF LOADER component enables to import the dataset.

Splitting the dataset

The TRAINTEST SPLITMAKER (EVALUATION tab) enables to split the dataset into learning and test set.
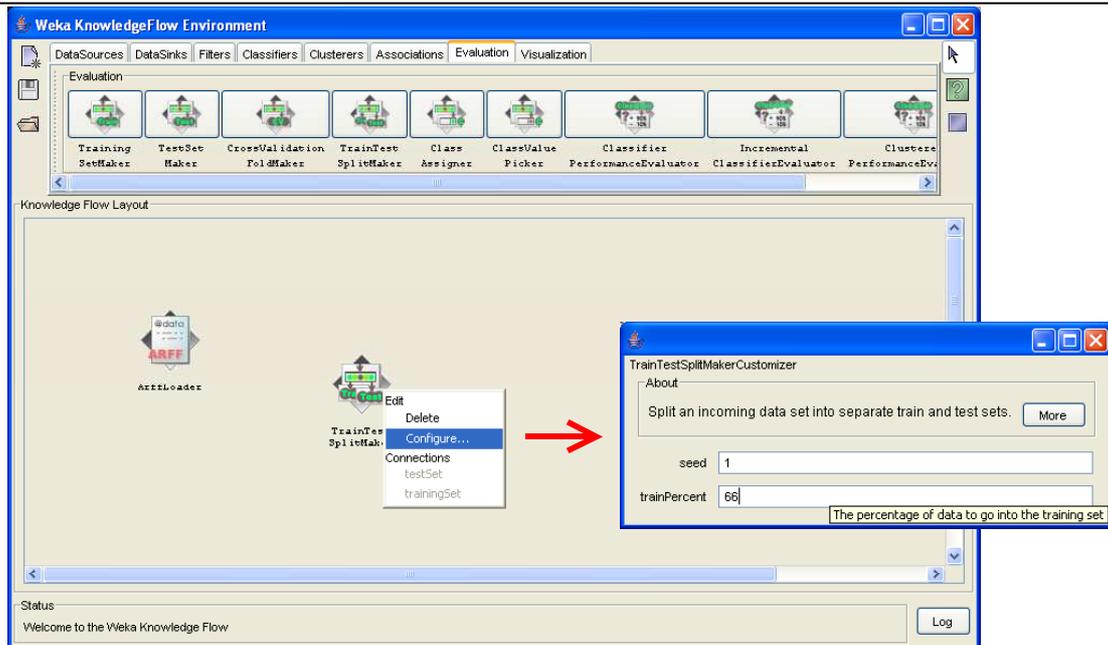
We connect the ARFF LOADER component to this new component; we use the DATASET connection.

**Learning algorithm and parameters**

In WEKA, the last column of the dataset is the default class attribute; the other columns are the predictive attributes. If we have not this configuration, we must use the CLASS ASSIGNER component.

The supervised learning methods are in the CLASSIFIERS tab. We add the MULTILAYER PERCEPTRON component in the diagram. We click on the CONFIGURE menu in order to set the right parameters.
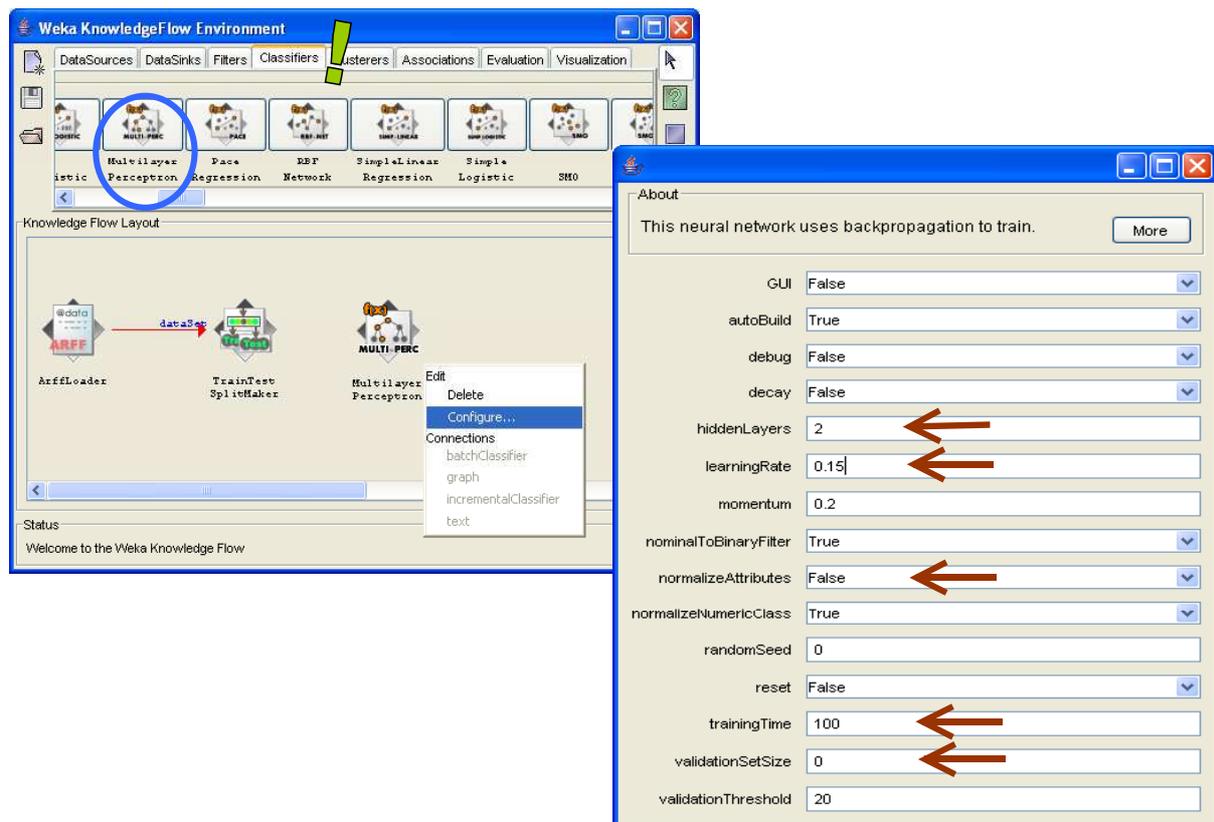

We set two neurons in the hidden layer (HIDDENLAYERS); the learning rate is 0.15 (LEARNING RATE); we do not use attribute transformation (NORMALIZE ATTRIBUTES = FALSE); the max number of iteration is set to 100 (TRAINING TIME); and we do not use a validation set (VALIDATION SET SIZE = 0).

We connect twice the TRAINTEST SPLITMAKER to this new component; we use the "training set" (1) and the "test set" (2) connections.
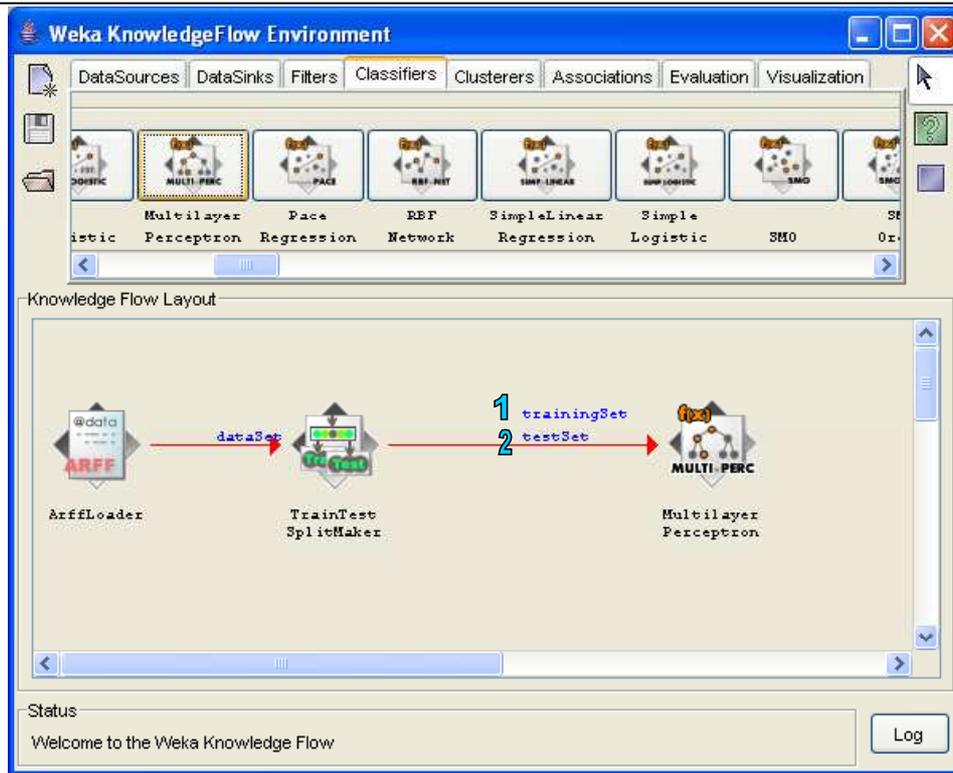
In order to visualize the weights of the network, we add the TEXT VIEWER component from the VISUALIZATION tab. We use the TEXT connection.
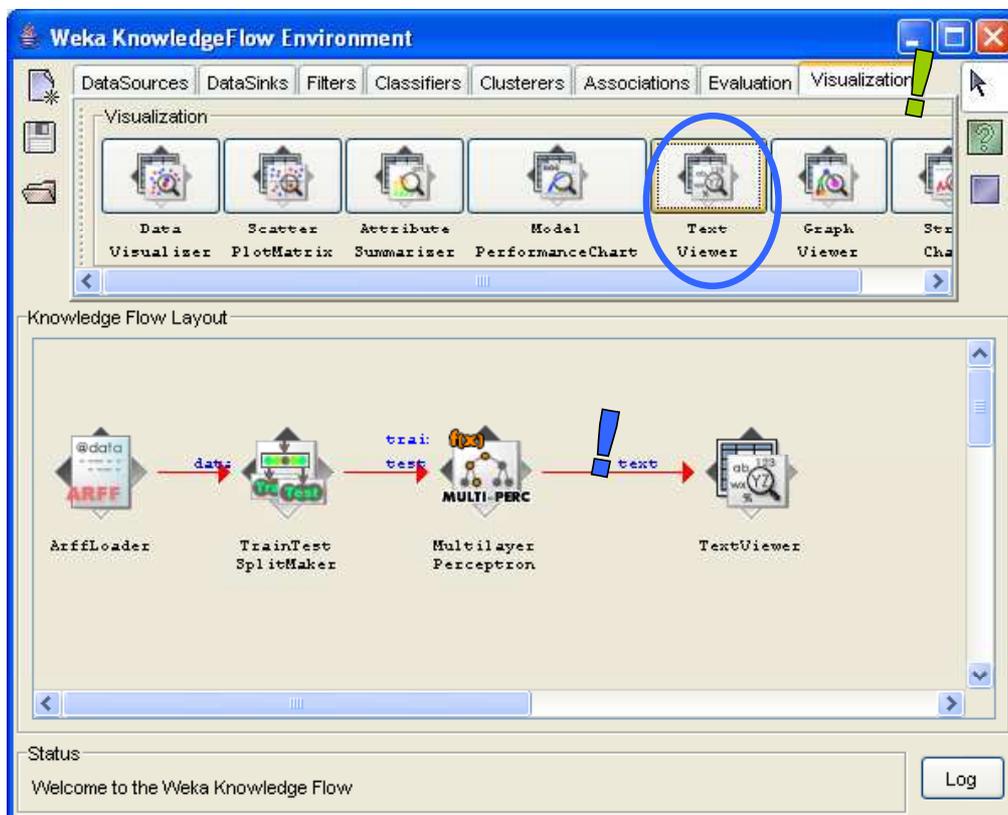
To launch the learning process, we click on the START LOADING menu of the ARFF LOADER component (1). And to display the results, we click on the SHOW RESULTS menu of TEXT VIEWER (2).
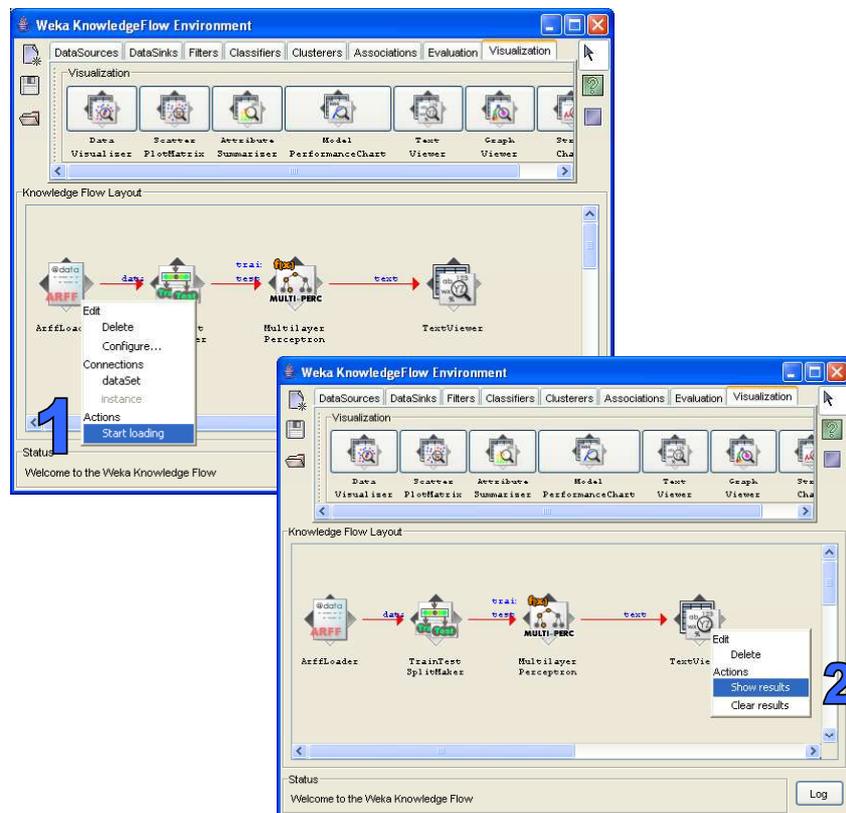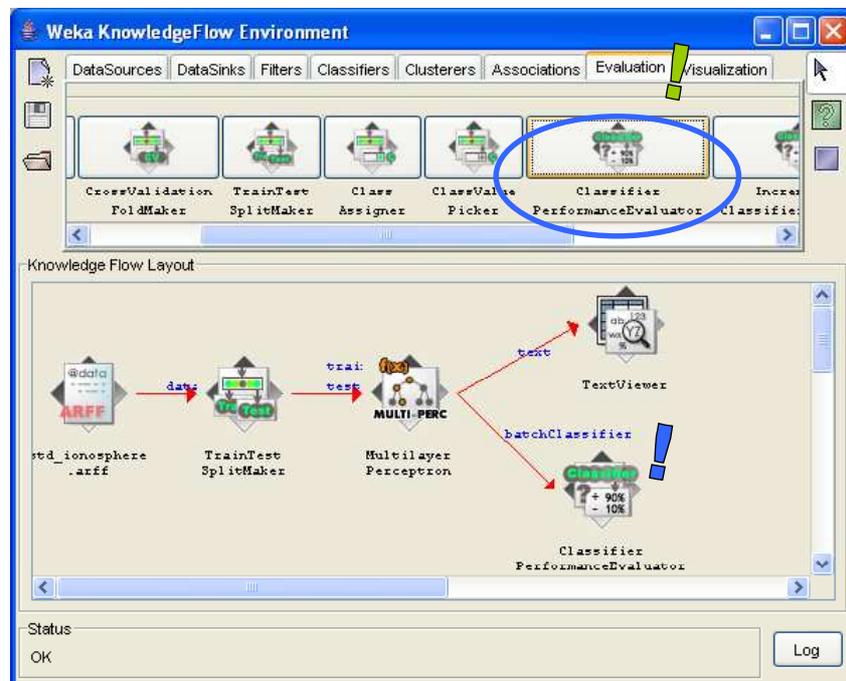

The weights of the Multilayer Perceptron appear in a new window.
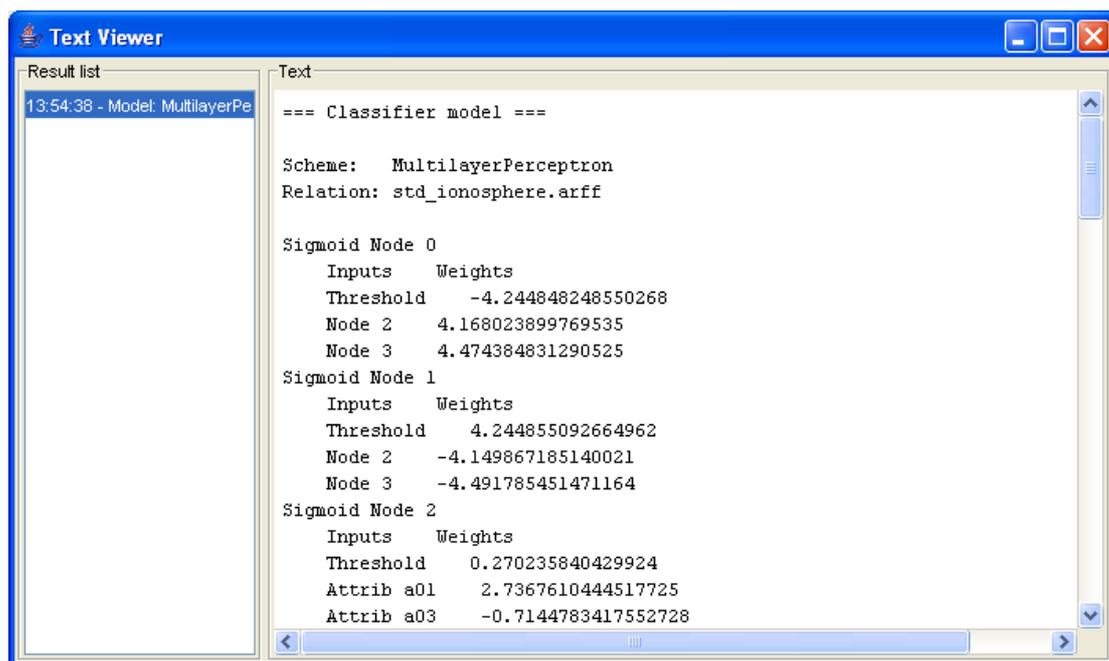
Test error rate

We must add two new components in the diagram in order to apply the network on the test set and visualize the confusion matrix. First, we add the CLASSIFIER PERFORMANCE EVALUATOR (EVALUATION tab) in the diagram and we use the BATCH CLASSIFIER connection.

Second, we add a new TEXT VIEWER component to visualize the results (TEXT connection).
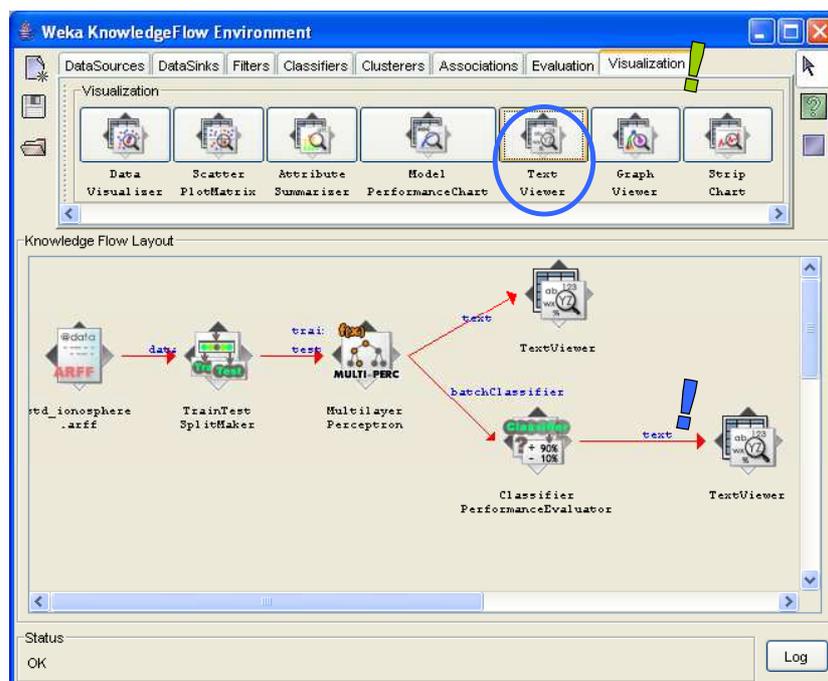
We must execute again the learning process (START LOADING of the ARFF LOADER component). We click on the SHOW RESULTS menu of TEXT in order to display the results.

There are 120 examples in the test set. The test error rate is 18.33%.

Conclusion

We note in this tutorial that the logic of the training and the evaluation of a neural network is the same one, whatever the software used.

The implementation of a perceptron is finally rather simple. The interpretation of the results, in particular the comprehension of the weights of the network, is definitely more complicated.