

# 1 Topic

## Induction of fuzzy rules using Knime.

This tutorial is the continuation of the one devoted to the induction of decision rules ([Supervised rule induction – Software comparison](#)). I have not included Knime in the comparison because it implements a method which is different compared with the other tools. Knime computes fuzzy rules. It wants that the target variable is continuous. That seems rather mysterious in the supervised learning context where the class attribute is usually discrete. I thought it was more appropriate to detail the implementation of the method in a tutorial that is exclusively devoted to the Knime rule learner (version 2.1.1). Especially, it is important to detail the reason of the data preparation and the reading of the results. To have a reference, we compare the results with those provided by the rule induction tool proposed by Tanagra.

Scientific papers about the method are available on line<sup>1,2</sup>.

## 2 Dataset

We use a version of the IRIS dataset with only two descriptors<sup>3</sup> (PET\_LENGTH, PET\_WIDTH). The interest is that we can represent graphically the data and the rules characteristics.

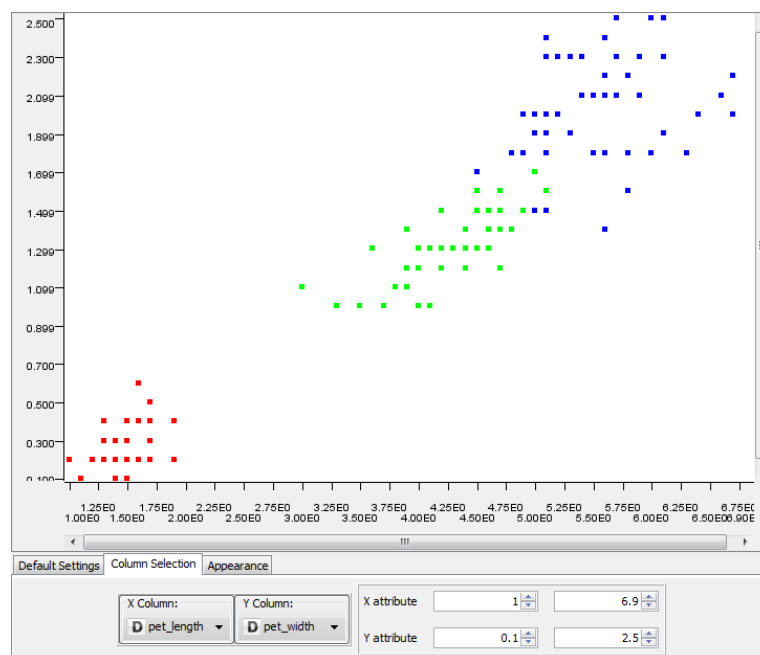


Figure 1 – Iris dataset - Positioning of the groups into a scatter plot (pet.length; pet.width)

<sup>1</sup> M.R. Berthold, « Mixed fuzzy rule formation », International Journal of Approximate Reasoning, 32, pp. 67-84, 2003.

[http://www.inf.uni-konstanz.de/bi/ml2/publications/Papers2003/Berto3\\_mixedFR\\_ijar.pdf](http://www.inf.uni-konstanz.de/bi/ml2/publications/Papers2003/Berto3_mixedFR_ijar.pdf)

<sup>2</sup> T.R. Gabriel, M.R. Berthold, « Influence of fuzzy norms and other heuristics on mixed fuzzy rule formation », International Journal of Approximate Reasoning, 35, pp.195-202, 2004.

[http://www.inf.uni-konstanz.de/bi/ml2/publications/Papers2004/GaBeo4\\_mixedFRappendix\\_ijar.pdf](http://www.inf.uni-konstanz.de/bi/ml2/publications/Papers2004/GaBeo4_mixedFRappendix_ijar.pdf)

<sup>3</sup> <http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/iris2D.txt>

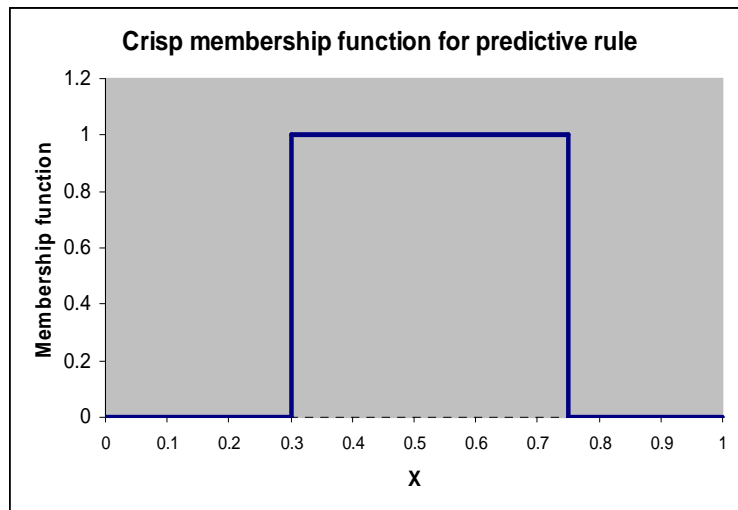
### 3 Induction of fuzzy rules using Knime

Roughly speaking, we say that the fuzzy rules incorporate a gradation in the definition of group membership regions in the representation space.

Let us consider the "crisp" decision rule:

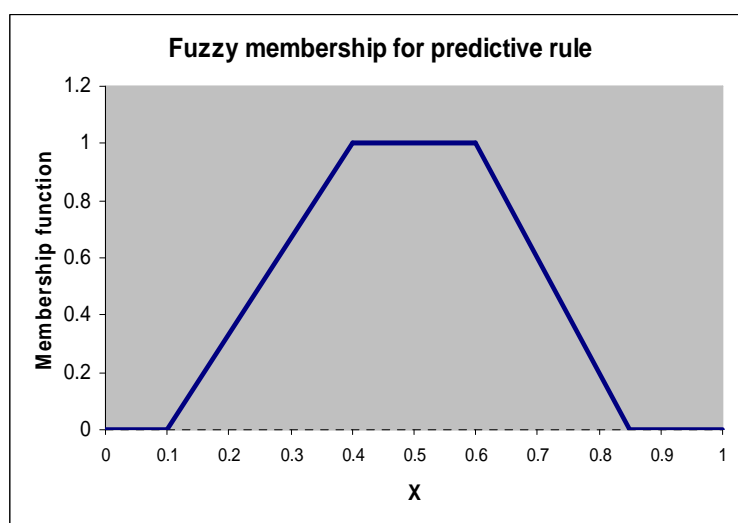
**If  $(X \geq 0.3)$  and  $(X < 0.5)$  Then  $Y = +$**

The (one-dimensional) region membership can be defined as follows



For an instance  $\omega$ , if  $X(\omega) \in [0.3 ; 0.5[$ , the membership degree is  $\mu(\omega) = 1$  ; if its value is outside of this interval, the membership degree is null [ $\mu(\omega) = 0$ ].

Without going into complicated discussions, it is clear that the thresholds "0.3" and "0.5" seem a bit arbitrary. They are going with some uncertainty because they have estimated using a sample. It is more convenient to include a certain gradation in the definition of the regions. For instance, instead a rectangle, we can use a trapezoid with the following coordinates  $\langle 0.1, 0.4, 0.6, 0.85 \rangle$



Thus:

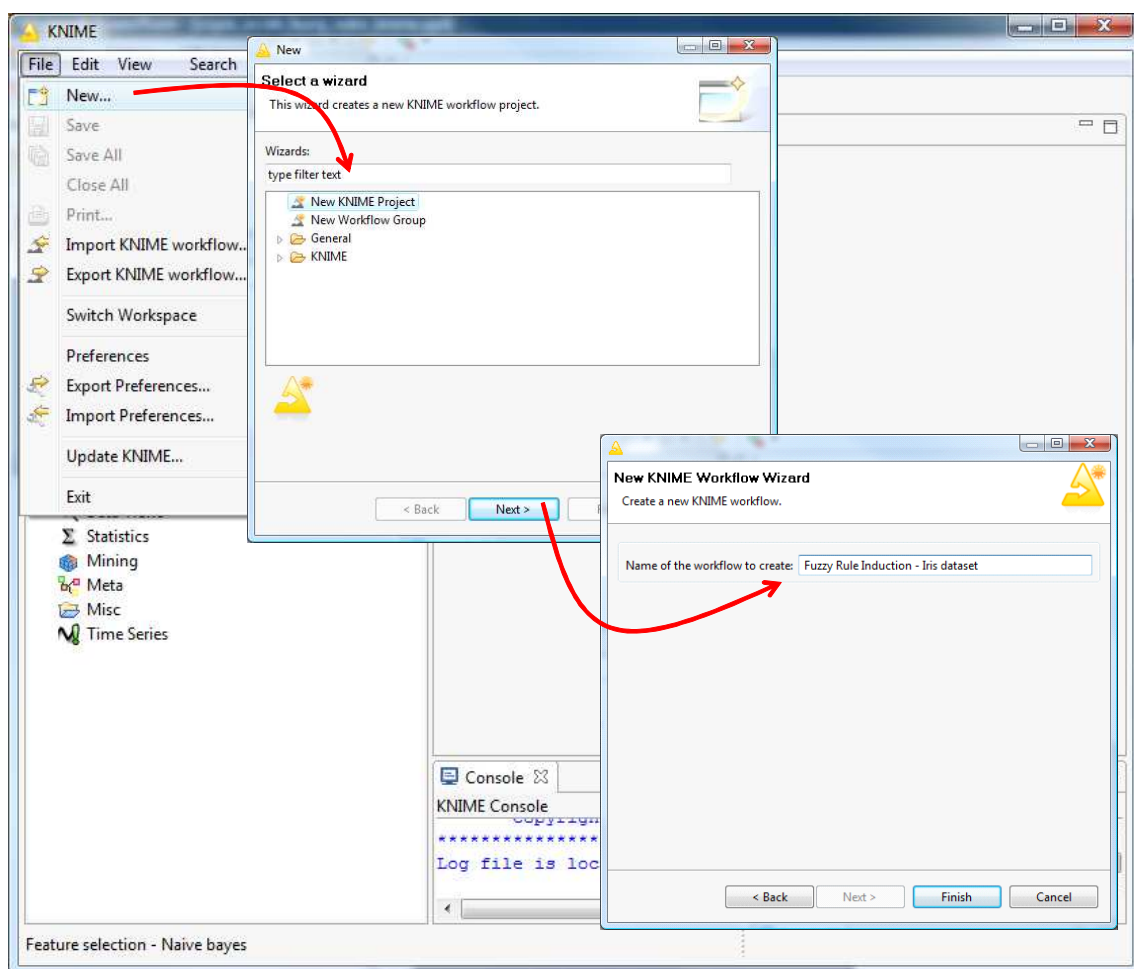
- For an individual with  $X = 0.5$ , it fully activates the rule with  $\mu = 1$ . The conclusion is "Y = +".
- For another instance with  $X = 0$ , it is not covered by the rule  $\mu = 0$ . The conclusion is "Y = -".
- For another instance  $X = 0.2$ , it partially activates the rule with  $\mu \approx 0.3$ . This reduces the scope of the conclusion.

This gradation is certainly more interesting for the comprehension of the decision. It is less arbitrary. But it is not without problems: (1) the integration of the decision rule into an information system for the deployment is not very easy, the rule cannot be translated directly in a SQL query; (2) many rules can be activated during the classification on an unseen instance, they have not the same weight, sometimes with a contradictory decision, we must implement a strategy for the management of incompatibilities.

Some authors think that fuzzy rules learners outperforms crisp decision rule learners because it reduces the variance of the classifiers.

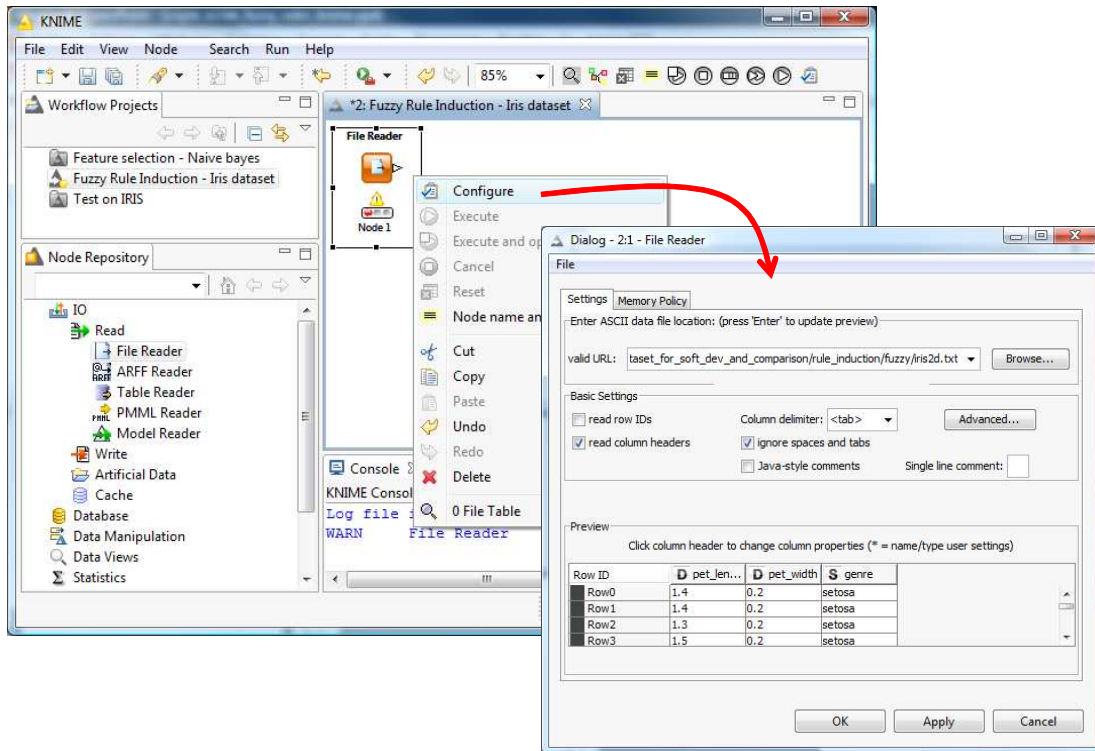
### 3.1 Creating a workflow and importing the dataset

We create a new project under Knime by clicking on the FILE / NEW menu. We set « Fuzzy Rule Induction - IRIS dataset » as project name.

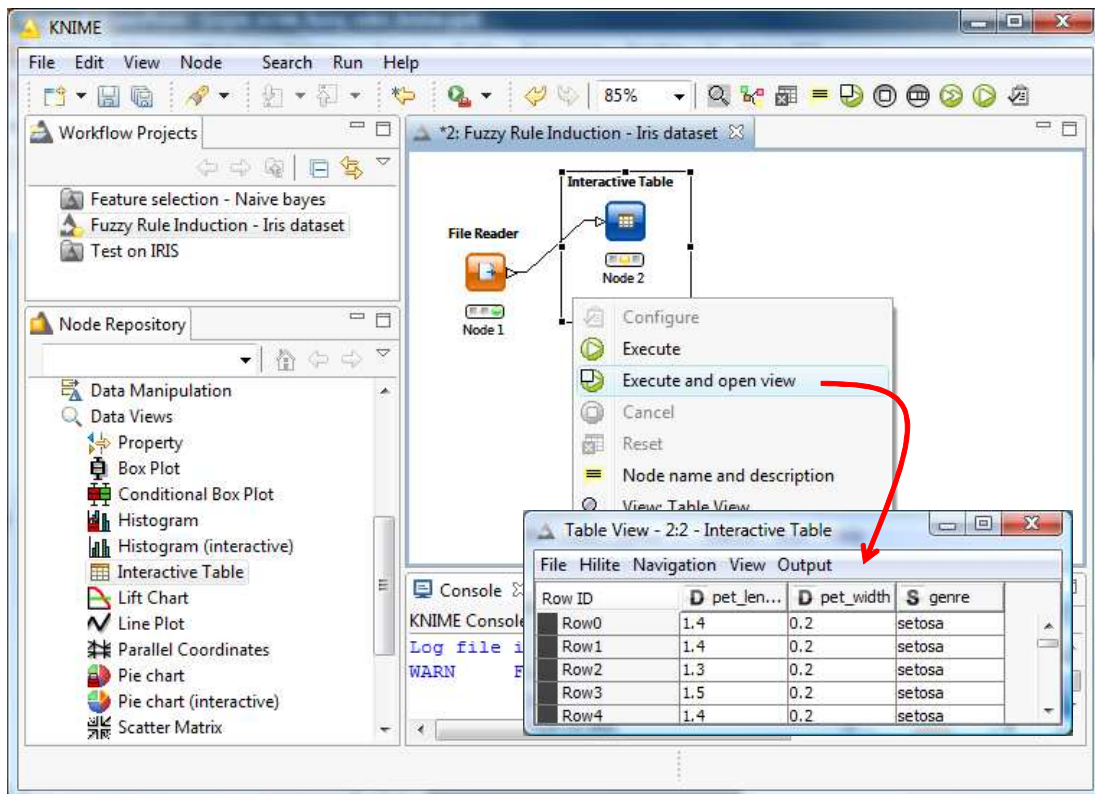


We obtain a new **Workflow Projects**.

To import the IRIS2D.TXT data file, we use the **FILE READER** component. We select the data file (CONFIGURE menu): the first row corresponds to the name of the variables; the column delimiter is the "tab" character.

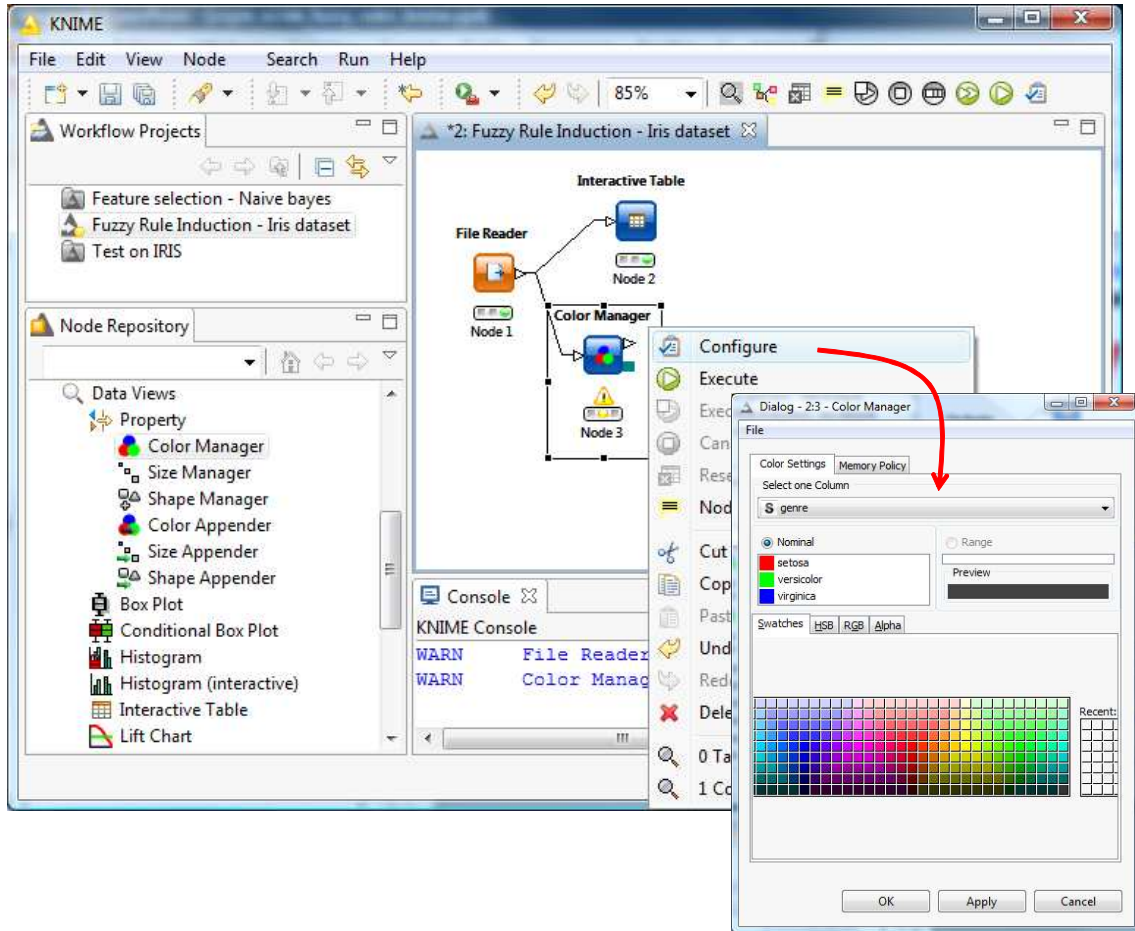


We click on the EXECUTE menu to perform the importation. We can visualize the dataset using **INTERACTIVE TABLE** component. We click on the EXECUTE and OPEN VIEW contextual menu.

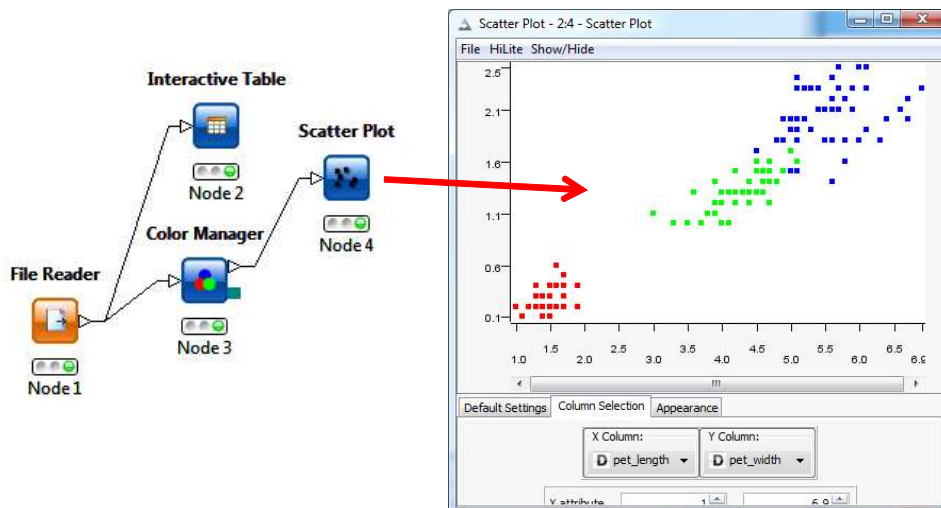


### 3.2 Graphical representation of the dataset

We want to make a scatter plot of the dataset by distinguishing the group memberships (Figure 1). We add the **COLOR MANAGER** component. We set the parameters as follows in order to colorize the points according to the target attribute **GENRE**.



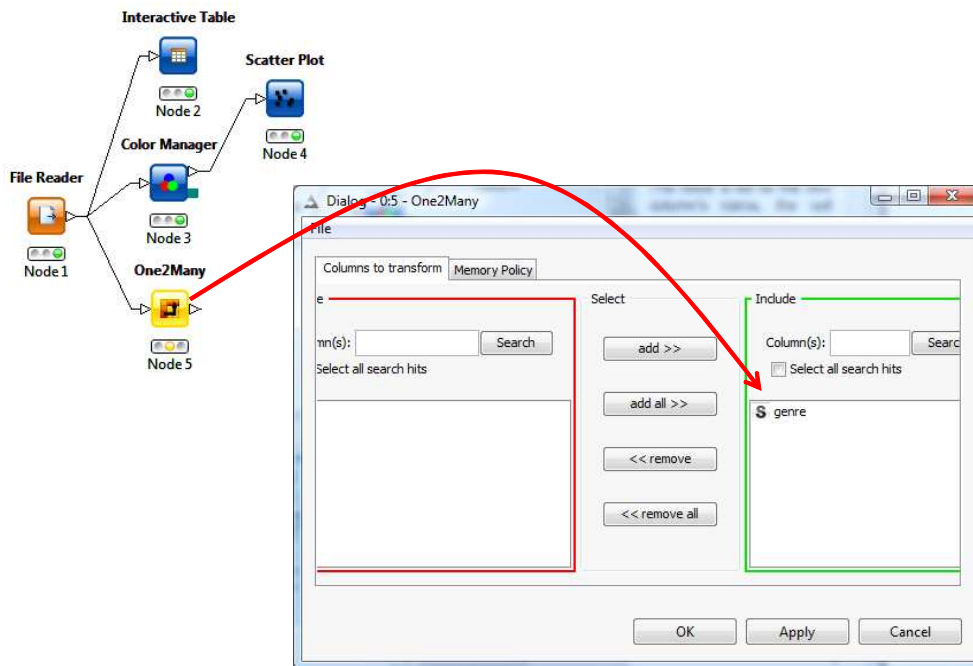
We add the **SCATTER PLOT** component (*Data Views*). We connect the COLOR MANAGER, then we click on the EXECUTE and OPEN VIEW menu. We can specify interactively the attributes on the horizontal and the vertical axes.



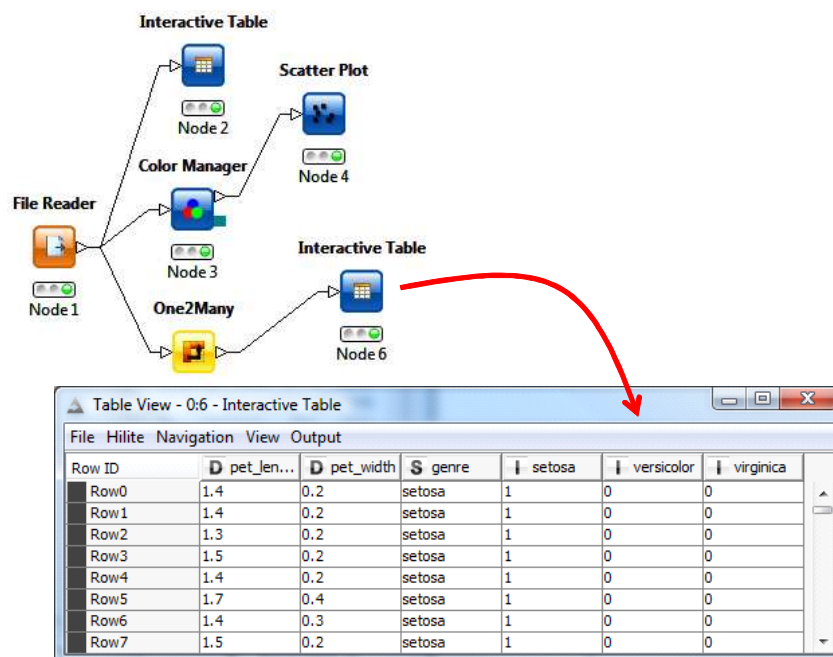
### 3.3 Transforming the target attribute for the induction process

In the supervised learning context, the target attribute is discrete. Yet, Knime asks one or more continuous variables. In reality, the target columns correspond to the degree of membership to each value of the original target attribute here.

The GENRE attribute consists of 3 values (setosa, versicolor and virginica), we must define 3 new binary columns 0/1. We use the **ONE2MANY** component (*Data Manipulation / Column / Transform*). We set the following parameters.

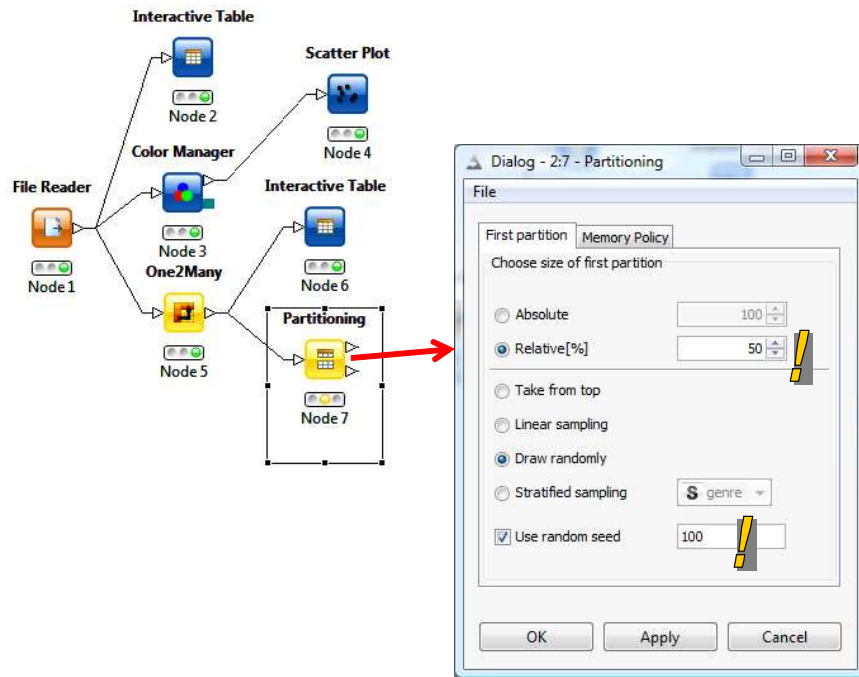


Again, we use the **INTERACTIVE TABLE** component (*Data Views*) to visualize the dataset.



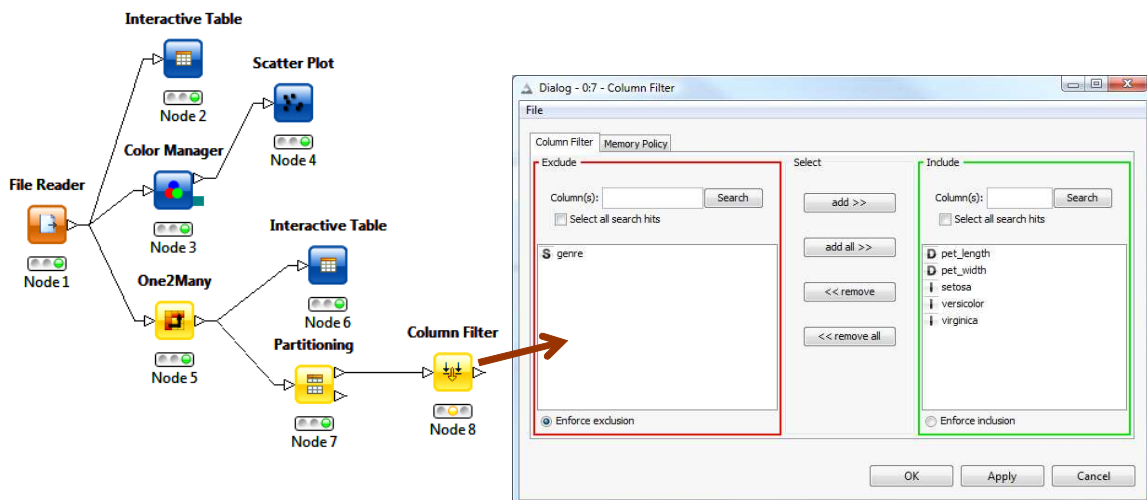
### 3.4 Partitioning the dataset into train and test samples

We use a test sample in order to assess the “true” performance of the classifier. Thus, first we must subdivide the dataset using the **PARTITIONNING** component (*DATA MANIPULATION / ROW / TRANSFORM*). We use the half (50%) for the training phase. We set USE RANDOM SEED = 100 to obtain the same results at each launching of the workflow.

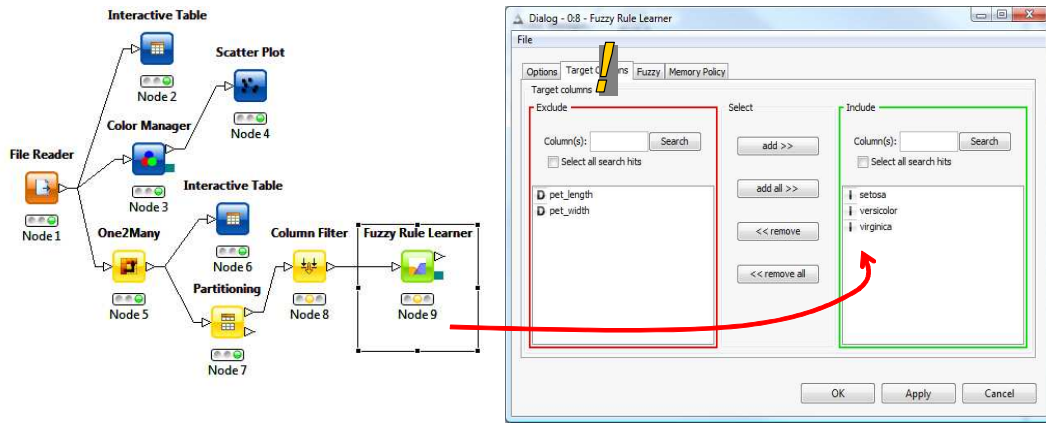


### 3.5 Induction of fuzzy rules

Now, we must specify the attribute to use during the analysis. The GENRE column is not useful here. We use **COLUMN FILTER** (*Data Manipulation / Column / Filter / Column Filter*).



Then, we add **FUZZY RULE LEARNER** (*Mining / Rule Induction / Fuzzy Rules*). We connect only the train set for the learning phase. We click on the CONFIGURE contextual menu. The most important here is to set appropriately the TARGET settings. We set the binary columns coming from the target attribute GENRE i.e. setosa, versicolor, virginica.



The other columns are the descriptors. We validate and we click on EXECUTE and OPEN VIEW.

**Learner Statistics**

- Number of epochs: 4
- Number of classes: 3
- Number of rules learned per class: (in total 9)
  - setosa: 1
  - versicolor: 3
  - virginica: 5
- Number of training instances per class: (in total 75)
  - setosa: 27
  - versicolor: 24
  - virginica: 24

The target attribute contains 3 values (number of classes); the learning algorithm supplies 9 rules, 1 for the "setosa" class value, 3 for "versicolor" and 5 for "virginica"; last, we have the number of instances for each class value.

We use INTERACTIVE TABLE to visualize the rules.

Row ID	[~] pet_length	[~] pet_width	S Class	I Weight	D Spread	I Features	D Variance
Rule_1	<1.1,1.1,1.7,3.3>	<0.1,0.1,0.6,0.6>	setosa	27	0.021	1	0.042
Rule_2	<3.3,3.3,4.9,5.0>	<0.6,1.0,1.5,1.7>	versicolor	22	0.056	2	0.268
Rule_5	<4.5,4.8,4.8,4.9>	<1.5,1.8,1.8,1.9>	versicolor	1	0	2	0
Rule_8	<5.1,5.1,5.1,5.8>	<1.5,1.6,1.6,1.7>	versicolor	1	0	2	0
Rule_3	<5.1,5.1,6.7,6.7>	<1.8,1.9,2.5,2.5>	virginica	15	0.068	1	0.294
Rule_4	<4.9,5.0,5.0,5.0>	<0.2,1.5,1.5,1.6>	virginica	1	0	2	0
Rule_6	<4.5,4.5,4.5,4.5>	<1.6,1.7,1.7,1.8>	virginica	1	0	1	0
Rule_7	<5.1,5.5,5.8,5.8>	<0.2,1.4,1.8,1.8>	virginica	5	0.008	2	0.084
Rule_9	<4.8,4.9,4.9,4.9>	<1.6,1.8,1.8,1.8>	virginica	2	0	2	0

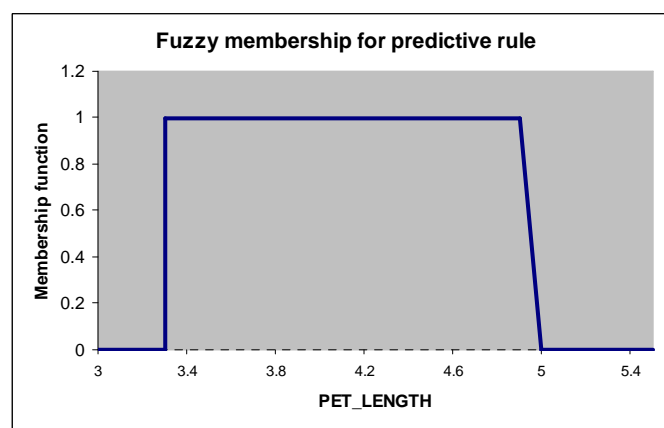
**Note 1:** FUZZY RULE LEARNER can handle only continuous descriptors. We must transform the discrete attributes, using ONE2MANY component for instance, if we want to use them.

**Note 2:** The algorithm supplies curiously 9 rules. We see graphically that 3 rules are sufficient. I think we must set carefully the settings of the algorithm before the launching of the learner.

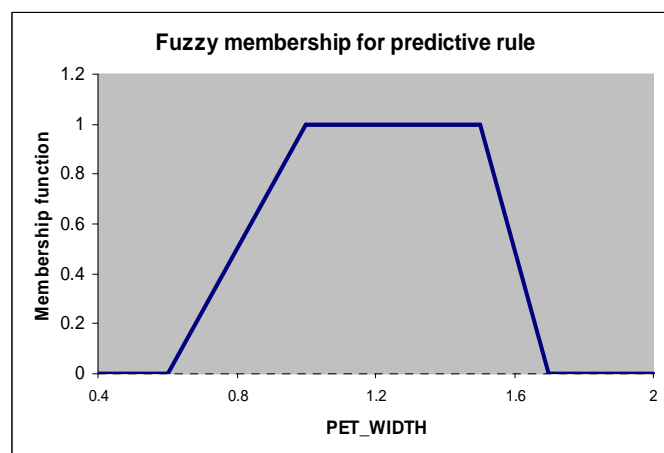
**Note 3:** The reading of the rules becomes tedious when the number of descriptors increases. We must interpret 4 values for each attribute.

### 3.6 Interpreting the rules

We study the rule n°2 here: the decision is "versicolor", its weight is 22 i.e. 22 instances are covered by the rule; other indicators are supplied. For each variable, we have the 4 coordinates of the trapezoid describing for the membership function. For PET\_LENGTH, we have <3.3, 3.3, 4.9, 5.0>

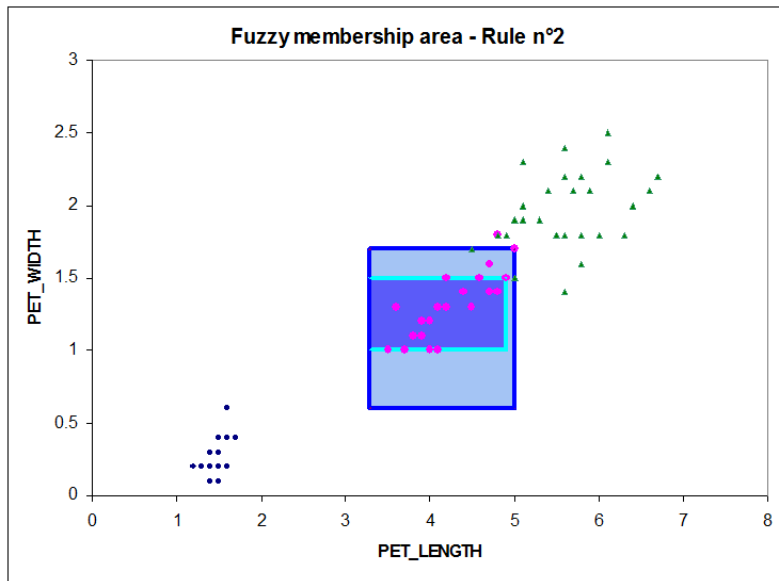


For PET\_WIDTH: <0.6, 1.0, 1.5, 1.7>



### 3.7 Representation of the rule in a scatter plot

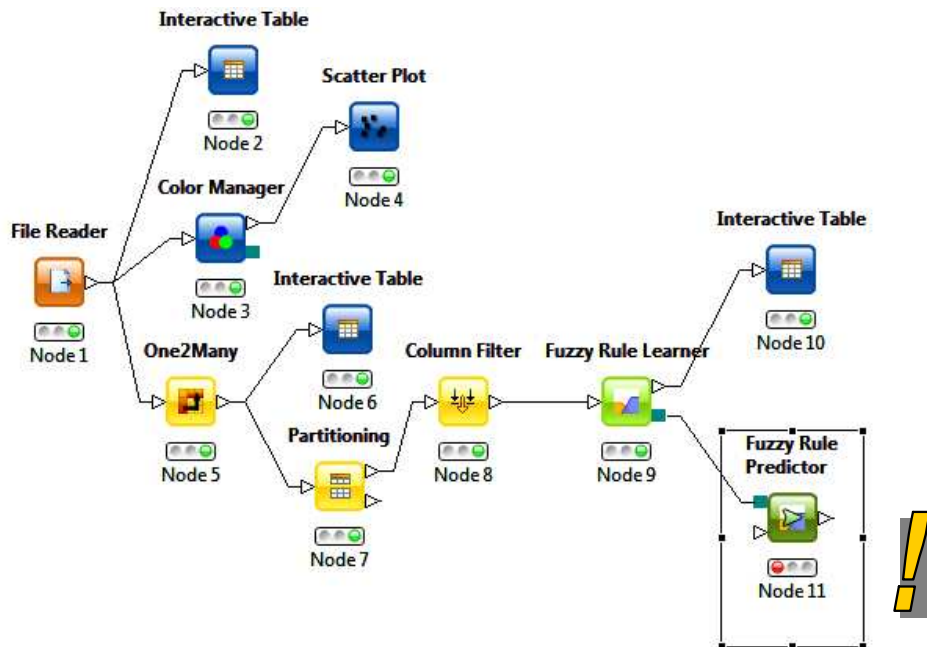
In a two dimensional diagram, we can represent the rule membership as follows.



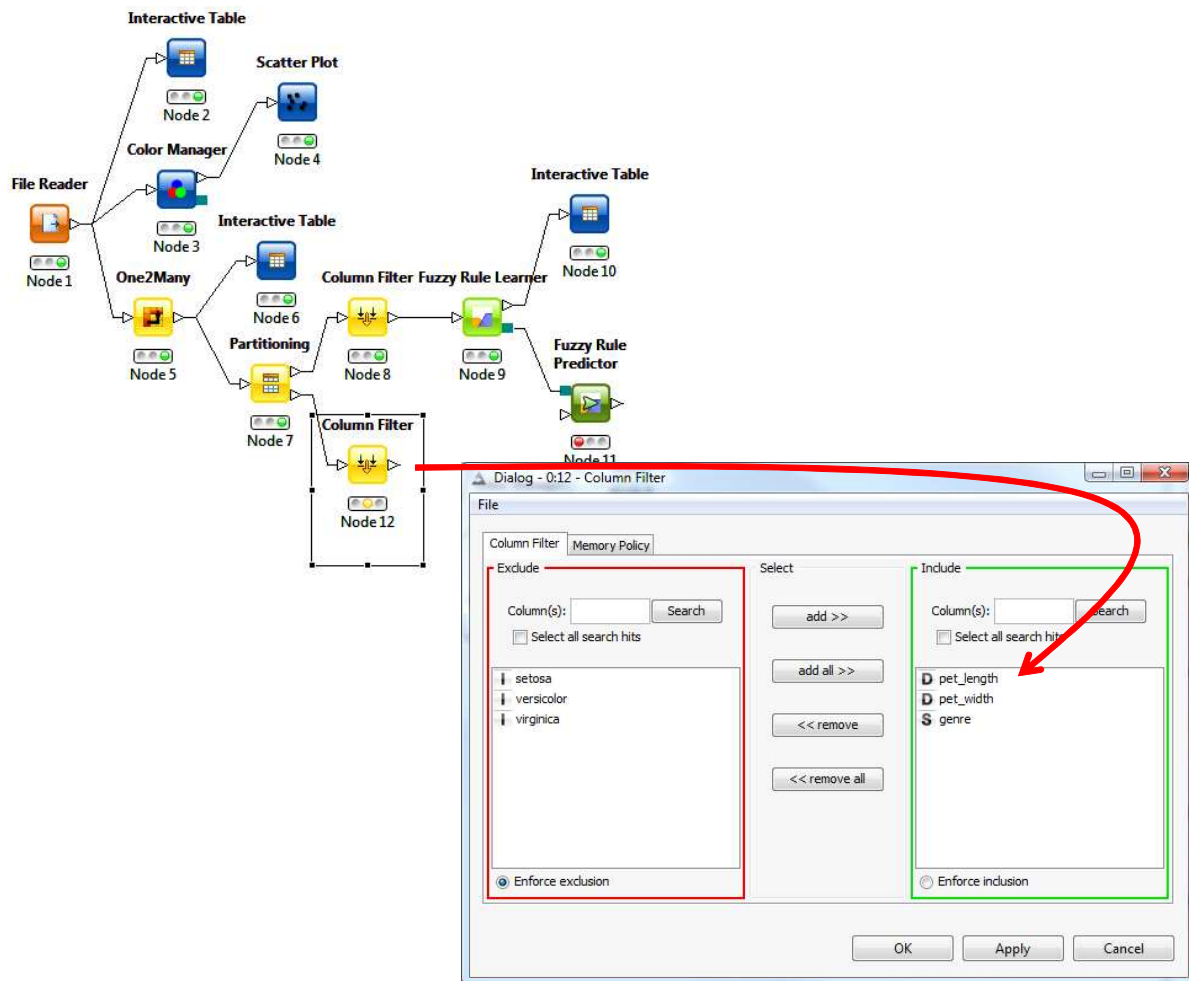
The inner rectangle (dark blue) designates an area of strong membership; the outer rectangle (blue) denotes the limit of not belonging (outside this area, the degree of membership of an individual to the rule is zero); from the inner rectangle to the outer rectangle, there is a gradation of membership. We note that when we consider all the rules of a classifier, some of them are overlapped. That makes the classification of a new individual complicated.

### 3.8 Generalization error rate – Confusion matrix

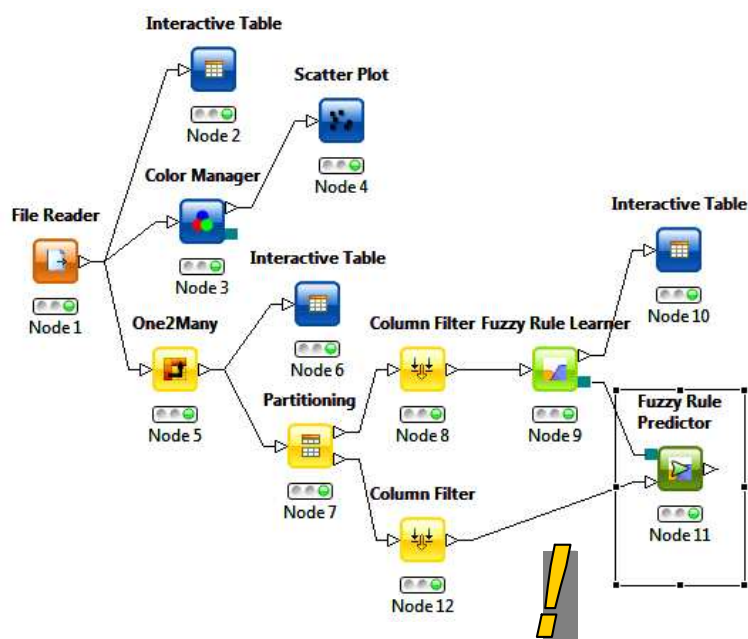
We want to evaluate the generalization capabilities of our classifier. For this, we use the **FUZZY RULE PREDICTOR** component (*Mining / Rule Induction / Fuzzy Rules*).



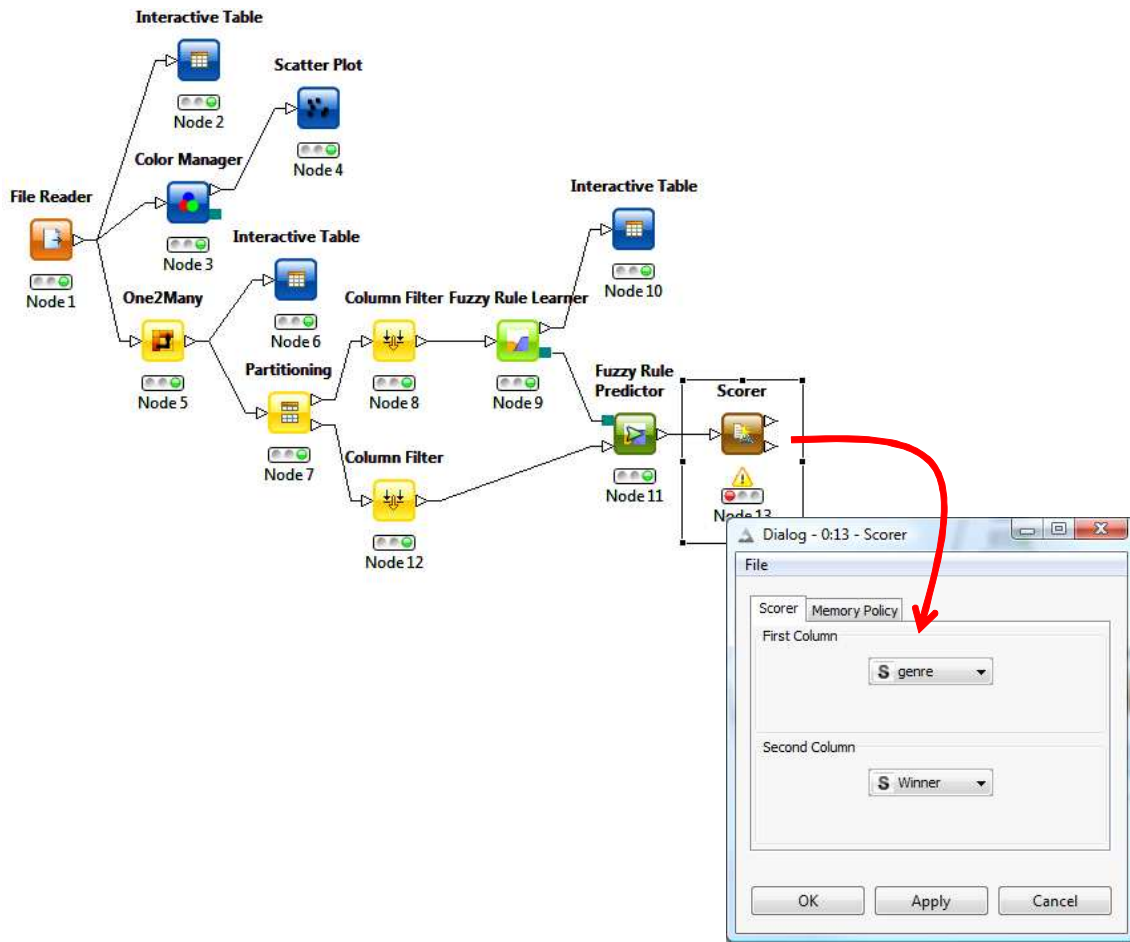
We filter the attributes coming from **PARTITIONING**. We use only the predictive variables and the target variable "GENRE" using **COLUMN FILTER**.



Then we use FUZZY RULE PREDICTOR to perform the prediction on the test set.



SCORER (*Mining / Scoring*) enables to compute the confusion matrix. We compare the target variable (GENRE) with the prediction of the classifier (WINNER).



We click on EXECUTE and OPEN VIEW. The test error rate is 4%.

genre \ Wi...	setosa	versicolor	virginica
setosa	23	0	0
versicolor	0	25	1
virginica	0	2	24

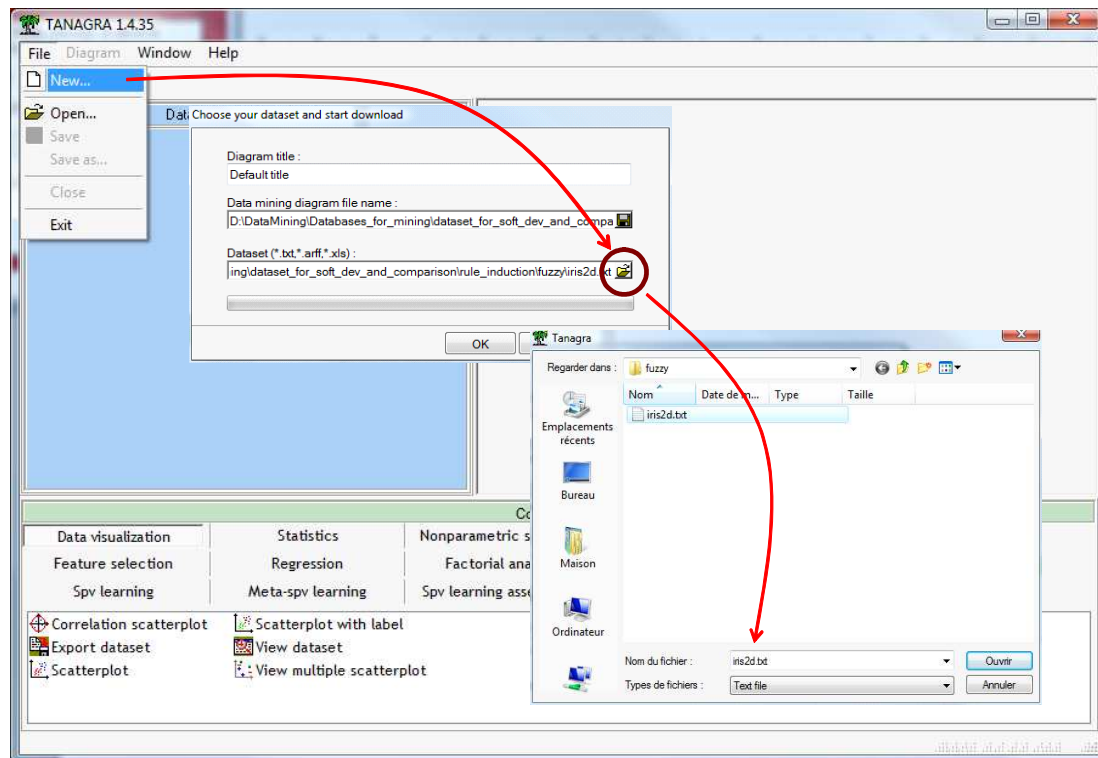
Correct classified: 72      Wrong classified: 3  
 Accuracy: 96 %      Error: 4 %

## 4 Induction of « crisp » rules using Tanagra

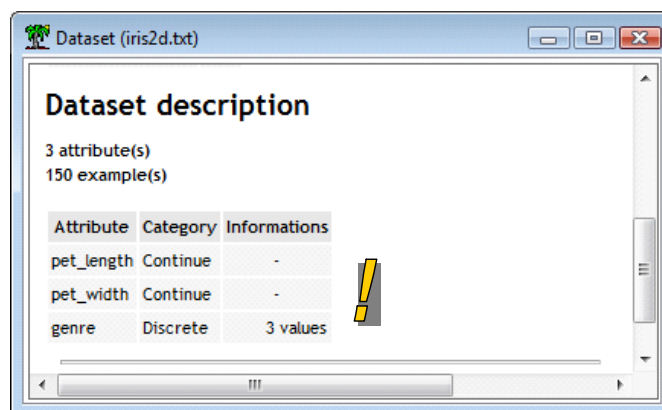
We want to analyze the behavior of a "crisp" rule learner on the same dataset, especially about the definition of the region membership. We use the RULE INDUCTION component of Tanagra. *About the comparison of the rule inducer in various tools (Weka, Orange, R), see "[Supervised rule induction – Software comparison](#)".*

### 4.1 Creating a diagram and importing the data file

We launch Tanagra, we click on the FILE / NEW menu to create a new diagram. We select the data file IRIS2D.TXT.



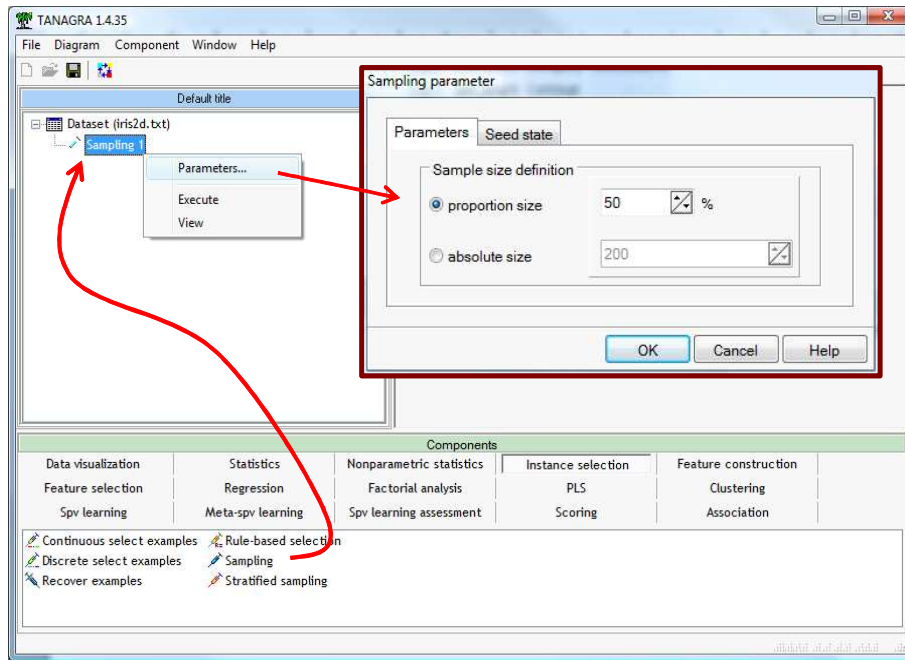
The dataset is loaded, we have 3 variables and 150 instances.



**Note:** The decimal separator is "." in the IRIS2D.TXT data file. If your computer does not handle this kind of decimal separator, you must modify the file using a text editor.

#### 4.2 Partitioning the dataset

We use the SAMPLING (*Instance Selection* tab) component to create the train and test samples. We set the following parameters (50% for the train sample).

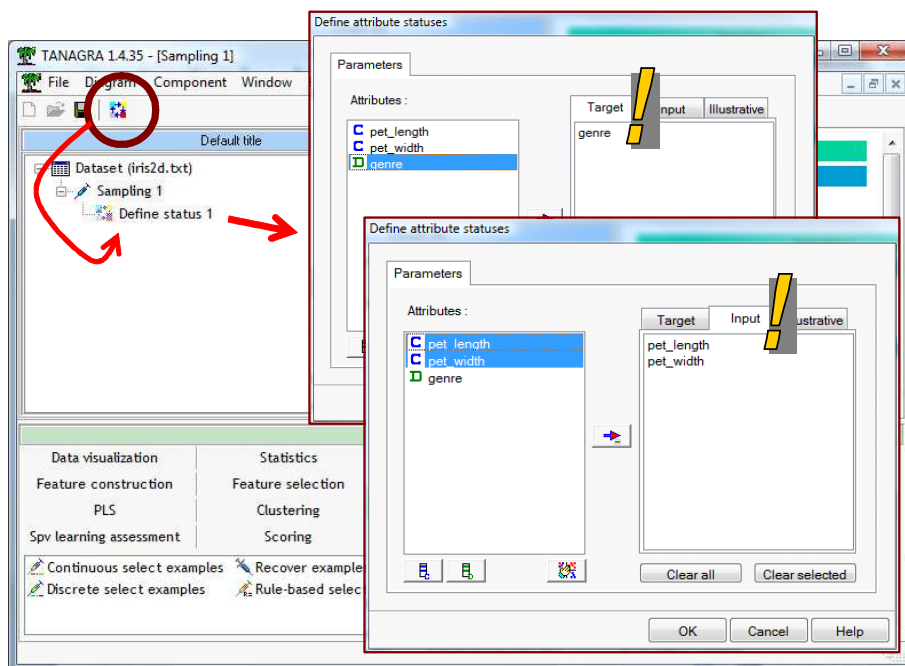


We validate and we click on the VIEW menu: 75 instances are incorporated into the train sample.

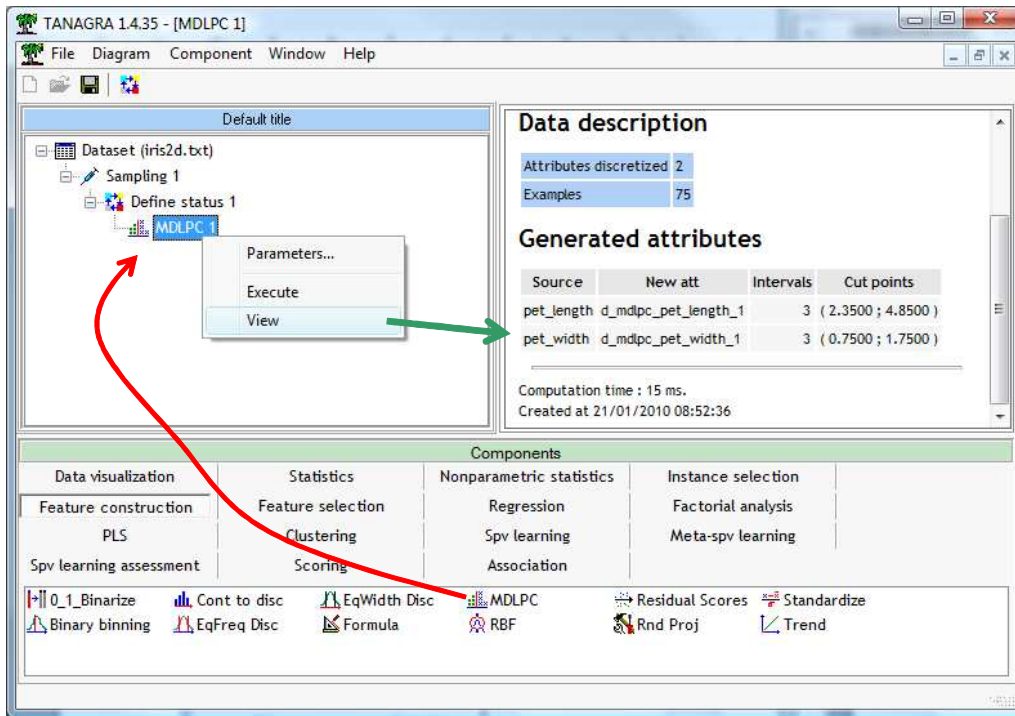
### 4.3 Discretization of continuous attributes

The RULE INDUCTION component can handle discrete descriptors only. We must discretize (subdividing into intervals) PET\_LENGTH and PET\_WIDTH before the learning process. We use a supervised approach.

We insert the DEFINE STATUS component into the diagram. We set PET\_LENGTH and PET\_WIDTH as INPUT, GENRE as target.



Then we add MDLPC (*Feature Construction*), a component for a supervised discretization process. We click on the VIEW menu to obtain the cut points. Two new variables are generated.

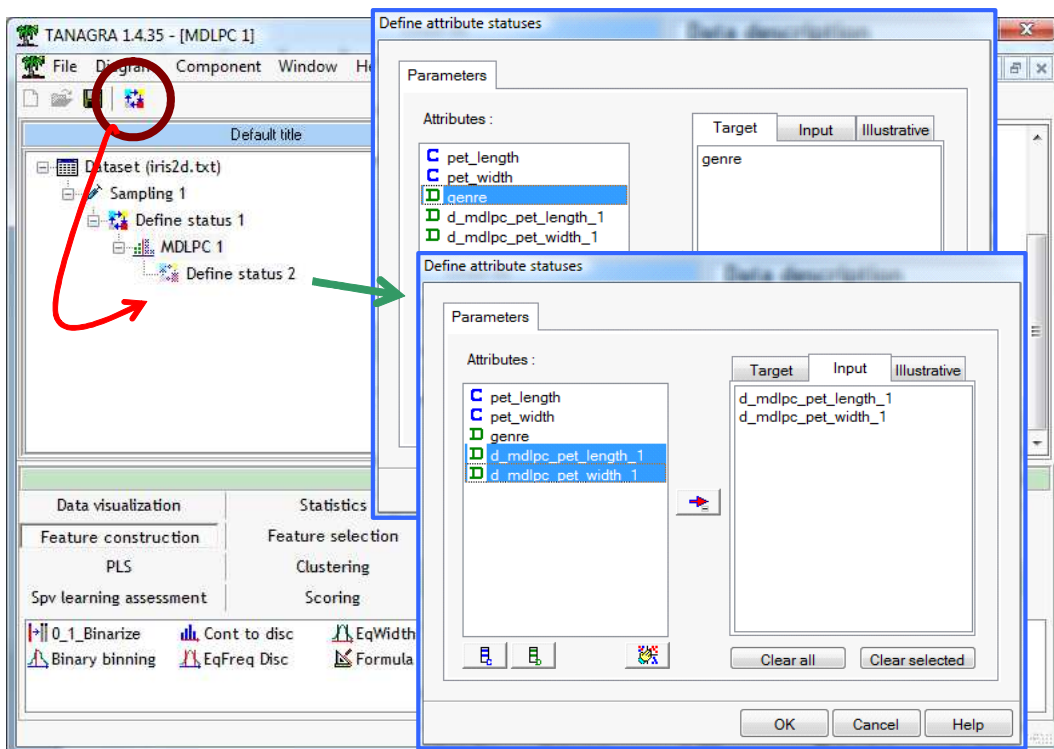


The cut points are (2.35; 4.85) for PET\_LENGTH; (0.75; 1.75) for PET\_WIDTH.

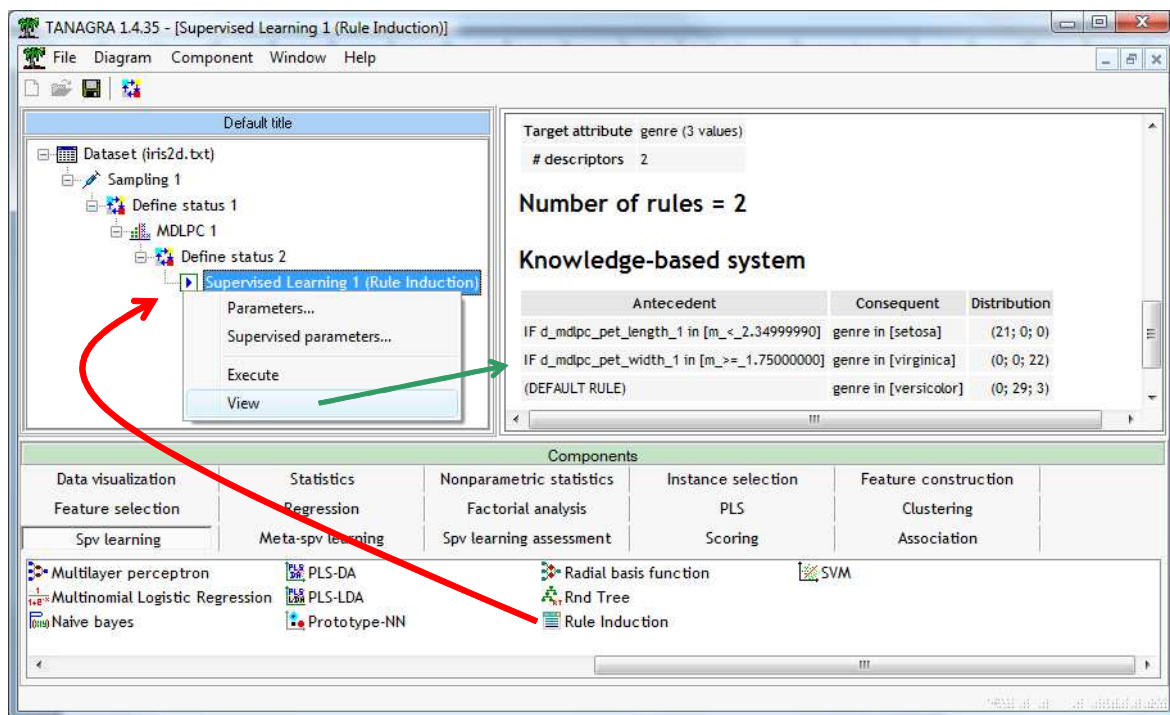
**Note:** It is very important to partitioning the dataset before the discretization process. Indeed, it is already a part of the learning process, the cut points must be determined using the training sample.

#### 4.4 Induction of rules

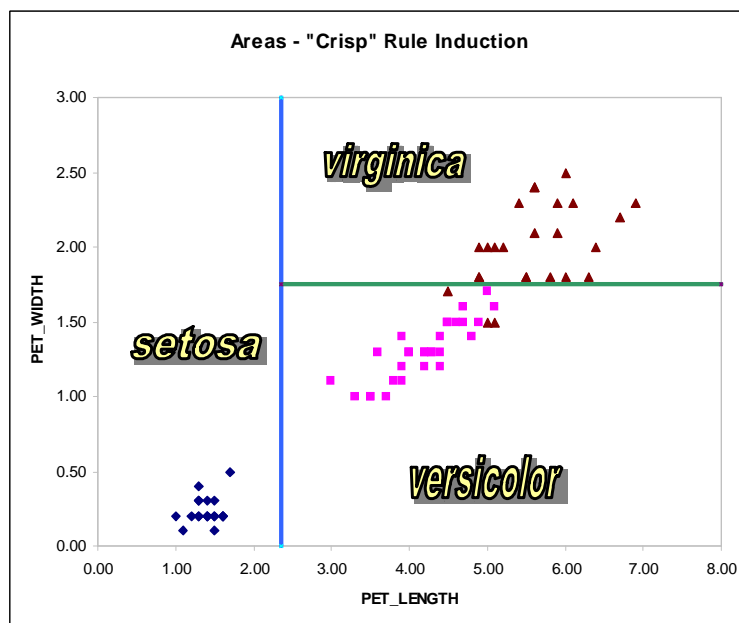
We can launch the rule induction process. We add again the DEFINE STATUS component. We set as INPUT the discretized variables; GENRE is always the target attribute.



We add the RULE INDUCTION component (*Spv Learning*). We click on the VIEW menu.

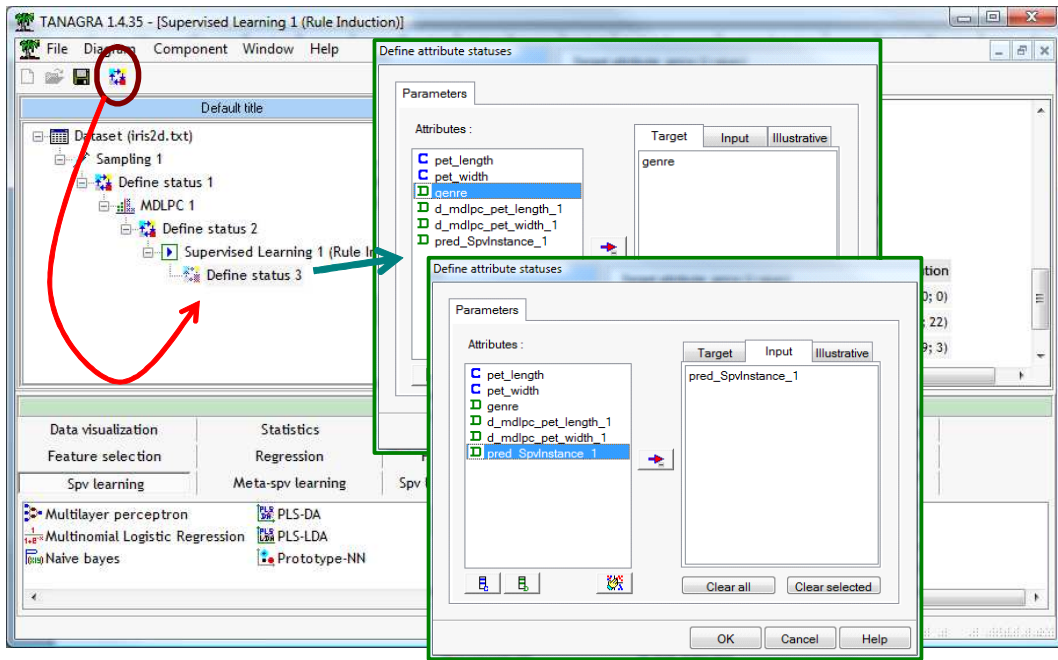


We obtain 3 rules. The last one is the "default rule". Here also, the rules define various regions in the representation space. But, the frontiers are "crisp". A new instance is assigned to one and only one region.

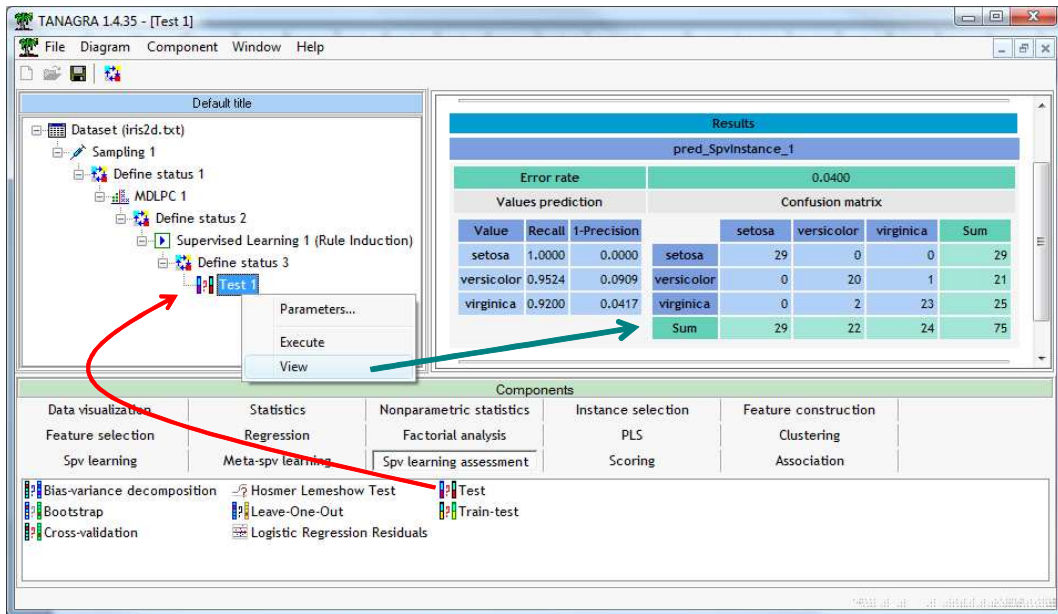


#### 4.5 Assessment on the test set

To assess the accuracy of the classifier, we insert a new DEFINE STATUS into the diagram. We set GENRE as target, PRED\_SPV\_INSTANCE\_1, the prediction column generated by the classifier, as input.



Then we add the TEST component (*Spv Learning Assessment*). It computes automatically the confusion matrix and some indicators on the test set. We click on the VIEW menu. Like for the fuzzy rules, the test error rate is 4%.



Actually, the error rate is not essential in this tutorial. We know that the IRIS problem is easy to learn. The challenge was to understand and to visualize the decision regions according to the rules.

## 5 Conclusion

The scientific interest of the predictive fuzzy rule induction is undeniable. However, it is little known outside a small circle of researchers. Mainly because it is not available into the popular tools, it restricts the diffusion of technology. Knime is one of the main tools which implement this approach. It seemed interesting to put forward the component to a specific tutorial.