1 Topic

Learning from imbalanced dataset with Logistic Regression.

In real problems, the classes are not equally represented in dataset. The instances corresponding to positive class, the one that we want to detect often, are few. For instance, in a fraud detection problem, there are a very few cases of fraud comparing to the large number of honest connections; in a medical problem, the ill persons are fortunately rare; etc. In these situations, using the standard learning process and assessing the classifier with the confusion matrix and the misclassification rate are not appropriate. We observe that the default classifier consisting to assign all the instances to the majority class is the one which minimizes the error rate.

For the dataset that we analyze in this tutorial, 1.77% of all the examples belong to the positive class. If we assign all the instances to the negative class – this is the default classifier – the misclassification rate is 1.77%. It is difficult to find a classifier which is able to do better. Even if we know that we have not a good classifier, especially because it does not supply a degree of membership to the classes (*Note: in fact, it assigns the same degree of membership to all the instances*).

A strategy enables to improve the behavior of the learning algorithms facing to the imbalance problem is to artificially balance the dataset. We can do this by eliminating some instances of the over-sized class (downsizing) or by duplicating some instances of the small class (over sampling). But few persons analyze the consequence of this solution on the performance of the classifier.

In this tutorial, we highlight the consequences of the downsizing on the behavior of the logistic regression. We implement the following experiment:

- We have a test set with 3000 instances. It is drawn randomly from the population. The proportion of the positive instances is an estimation of the probability to obtain a positive instance into the population.
- We use a learning sample which is also drawn randomly from the population. The positive instances are very few in comparison to the sample size (15 positives among 900 instances). We obtain the classifier **M1** that we evaluate on the test set. We use two criteria to assess the classifier: (1) the ability to assign higher scores to the positive instances, we use the ROC curve for that (and the AUC criterion); (2) the ability to correctly classify the new instances, we use the standard confusion matrix and we measure the error rate.
- Then we balance the learning sample by sampling among the negative instances. The new learning sample has 30 instances (15 positives vs. 15 negatives). We learn the classifier **M2** from this new learning sample. We use the test set to evaluate again this classifier.

The interest of this protocol is that we have a representative test sample for the evaluation i.e. a test sample where the proportions of positive and negative instances are the same as into the population. In practice, this is not always possible. The test sample is also not representative either. In this situation, we must also correct the performance criteria (such as error rate or accuracy) for an undistorted assessment of the true behavior of the classifier. Neglecting this aspect is often the source of the mistaken opinion that accompanies the process of balancing the data when we deal with the imbalance problem.

2 Dataset

We have a data file with 3900 instances. There are only 69 positive instances. The target attribute is "objective"; the predictive attributes are V1 to V6. We have randomly subdivided this dataset in train (900 instances) and test (3000) samples. They have the following characteristics.

Nombre de objective	status1		
objective	train	test	Total
positive	15	54	69
negative	885	2946	3831
Total	900	3000	3900

The STATUS1 column enables to identify the train and test samples. In the first one, we have 900 instances, of which 15 are positives and 885 negatives. Into the second one, the test sample, there are 3000 instances (54 positives and 2946 negatives). Both the train and the test are drawn randomly from the population.

A second column STATUS2 enables to specify the balanced learning set. There are 30 instances. We use this dataset for the construction of the second classifier M2 that we evaluate also on the test sample defined above.

Nombre de objective	status2		
objective	train	neglect	Total
positive	15	54	69
negative	15	3816	3831
Total	30	3870	3900

We note that the 30 instances which constitute this balanced learning sample are extracted from the representative training sample of 900 instances defined from STATUS1 as we can see from the following table.

Nombre de status1	status2		
status1	train	neglect	Total
train	30	870	900
test		3000	3000
Total	30	3870	3900

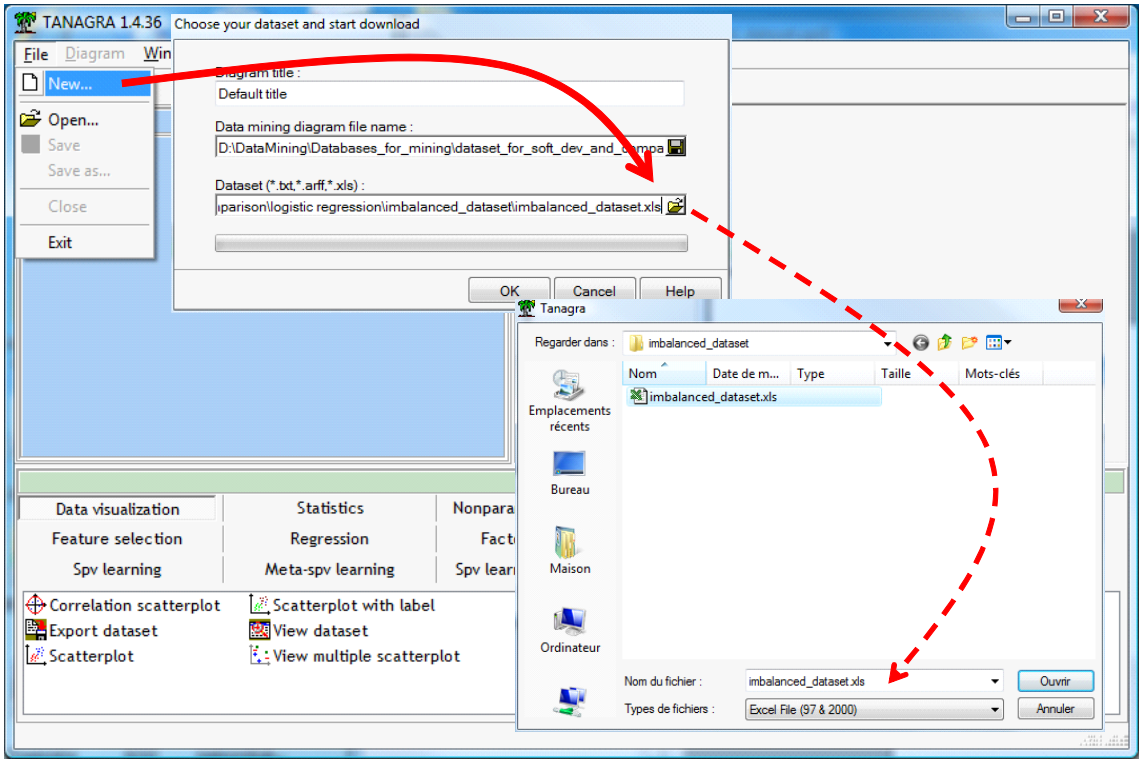
3 Learning from the original training set (M1 classifier)

3.1 Creating a diagram and importing the data file

We launch Tanagra. We create a new diagram (FILE / NEW). We select the « imbalanced_dataset.xls[1] » data file (XLS file format). Caution: the data file must not currently being edited in a spreadsheet[2]. We observe that 3900 instances and 9 columns are loaded

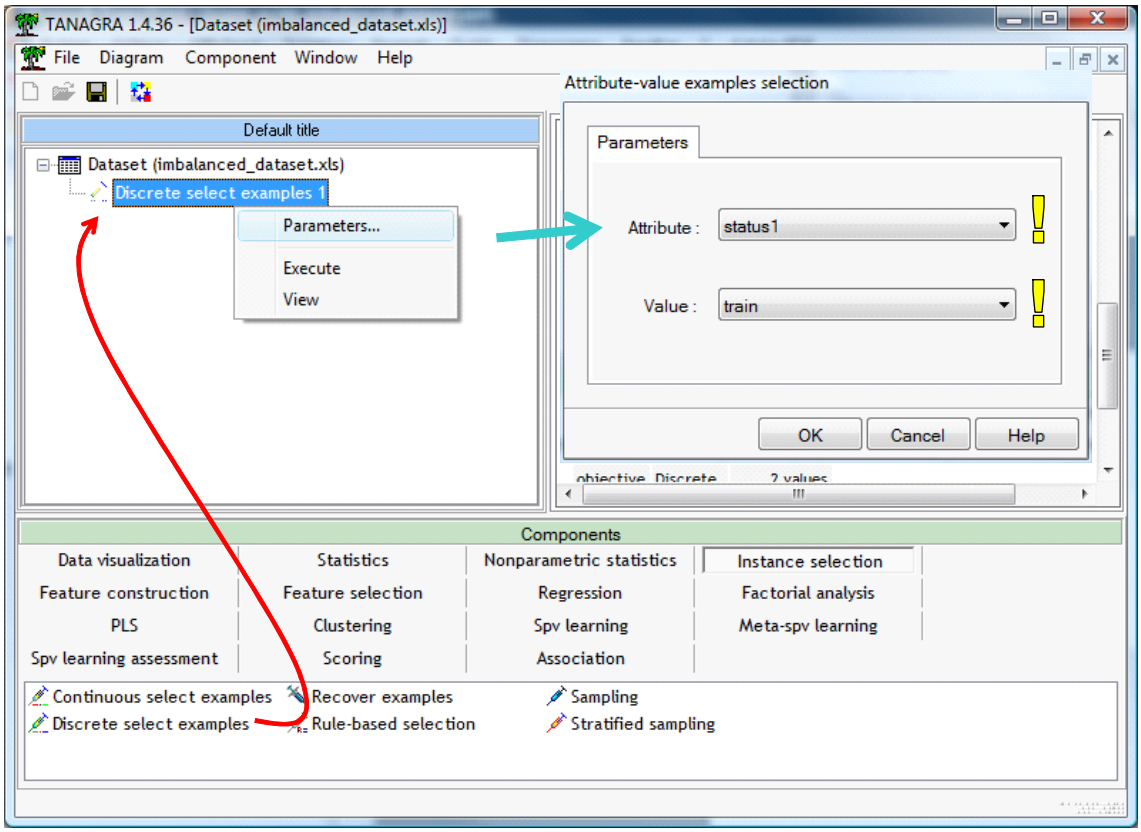
[1] http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/imbalanced_dataset.xls

[2] See http://data-mining-tutorials.blogspot.com/2008/10/excel-file-format-direct-importation.html; see also http://data-mining-tutorials.blogspot.com/2008/10/excel-file-handling-using-add-in.html

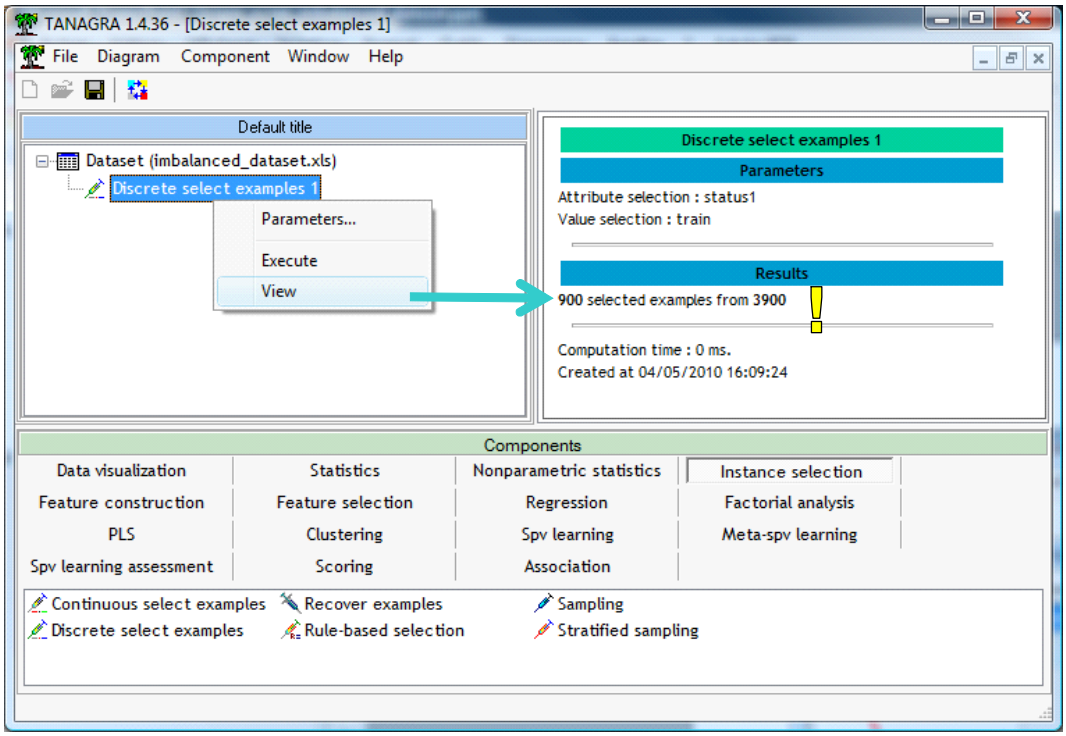


3.2 Specifying the learning sample

We want select the training sample. We insert the DISCRETE SELECT EXAMPLES (INSTANCE SELECTION tab) component into the diagram. We click on the PARAMETERS contextual menu. We set the following settings.

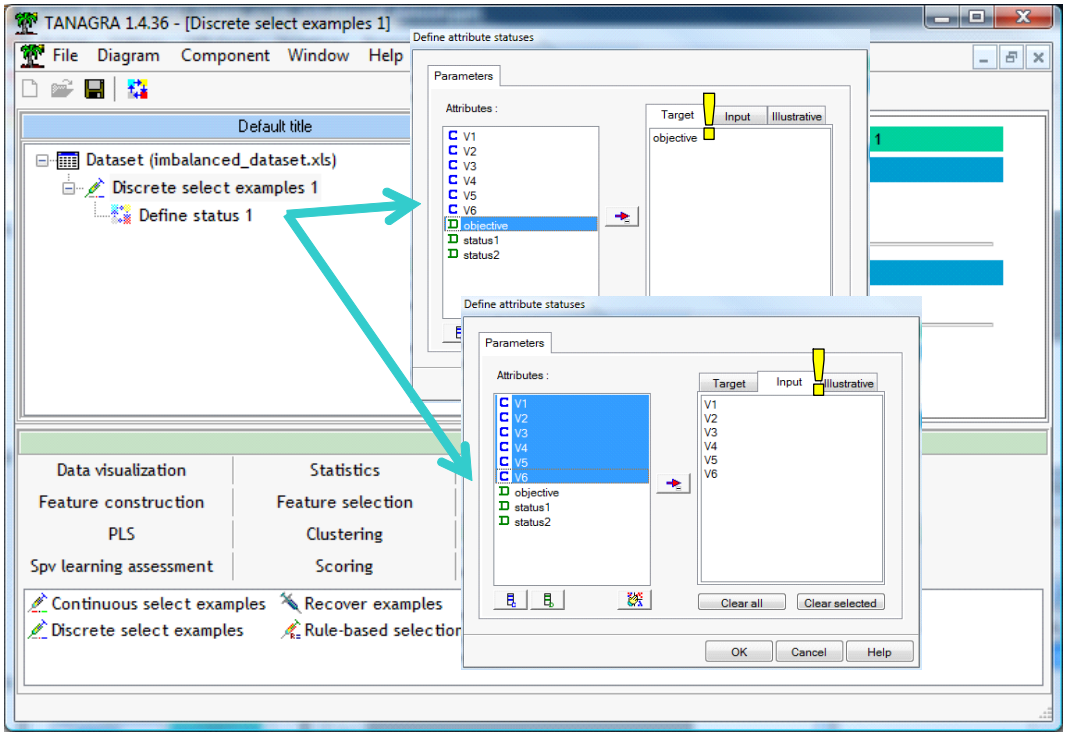


We click on the VIEW menu. Tanagra indicates that 900 instances among 3900 are selected for the subsequent part of the diagram.

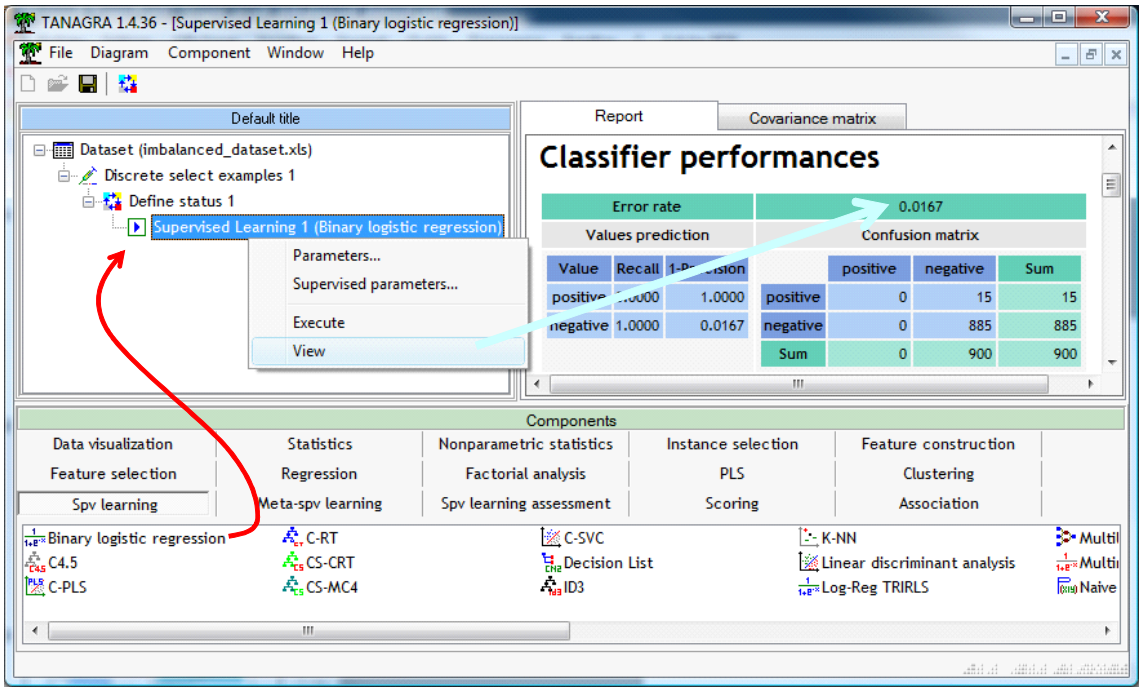


3.3 Constructing the classifier

We use the DEFINE STATUS component in order to specify the type of the variables for the analysis: OBJECTIVE is the target attribute; the variables V1 to V6 are the input attributes.



Then we add the BINARY LOGISTIC REGRESSION component (SPV LEARNING tab). We click on the VIEW menu to launch the calculations.



We observe from the confusion matrix that the classifier assigns all the instances to the negative class. The resubstitution error rate seems very small (1.67%). But it is not better than the one of the default classifier. Yet there was no error. The modeling process has run smoothly, we see below the coefficients of the logistic regression.

Attributes in the equation

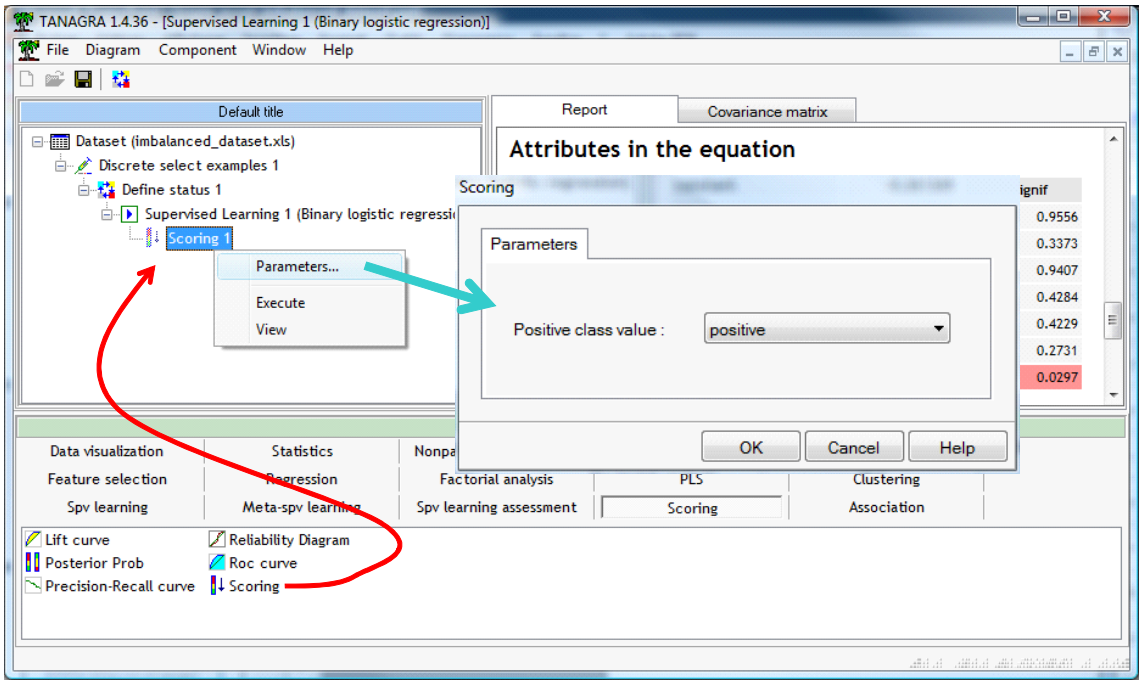
Attribute	Coef.	Std-dev	Wald	Signif
constant	-0.361369	6.4841	0.0031	0.9556
V1	-0.026503	0.0276	0.9205	0.3373
V2	-0.003101	0.0417	0.0055	0.9407
V3	-0.000127	0.0002	0.6271	0.4284
V4	0.000610	0.0008	0.6423	0.4229
V5	0.088404	0.0807	1.2011	0.2731
V6	0.000147	0.0001	4.7275	0.0297

This result may seem disappointing. The corrections introduced by the data miners are designed to correct this "abnormal" behavior. We would like to recognize the positive instances.

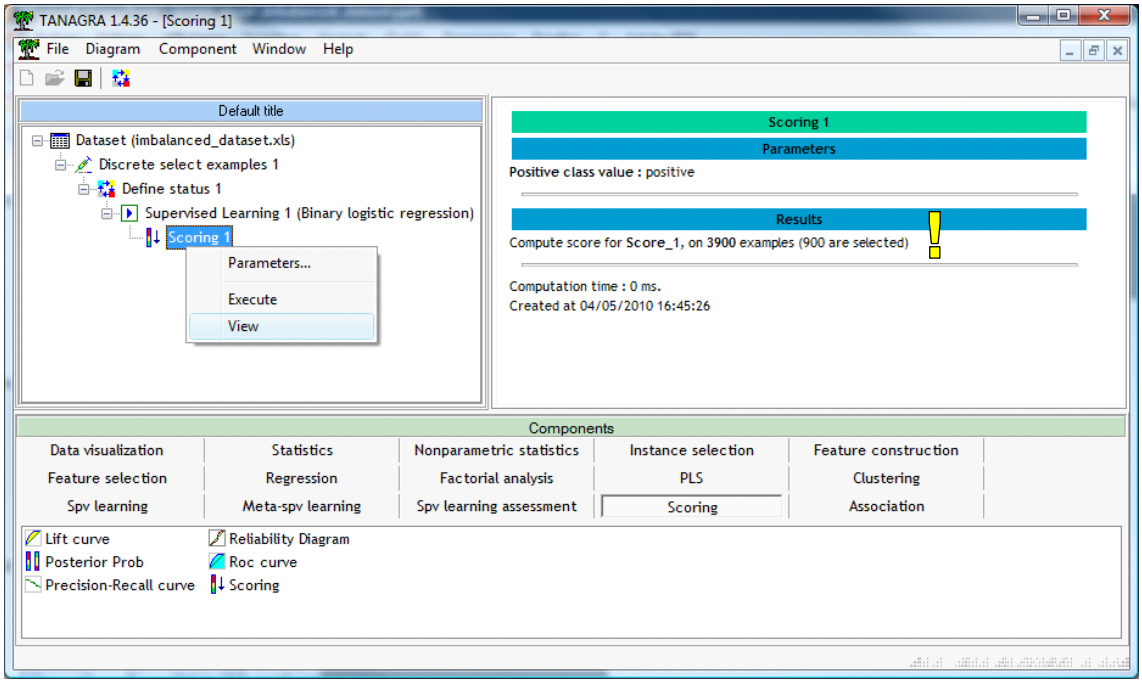
But is that our model as worse as the default classifier? One way to assess a classifier is its ability to assign higher scores (the propensity to be positive, the degree of membership to the positive class) to the positive instances than the negative instances. About the default classifier, we observe that it assigns the same score to all the instances, whether positive or negative. Rather than the confusion matrix and error rate, it is more appropriate, especially in the context of imbalanced dataset, to use the ROC curve[3] to evaluate the classifiers.

[3] http://en.wikipedia.org/wiki/Receiver_operating_characteristic

We inset the SCORING component (SCORING tab) to compute the score of each instances. We set the following parameter.



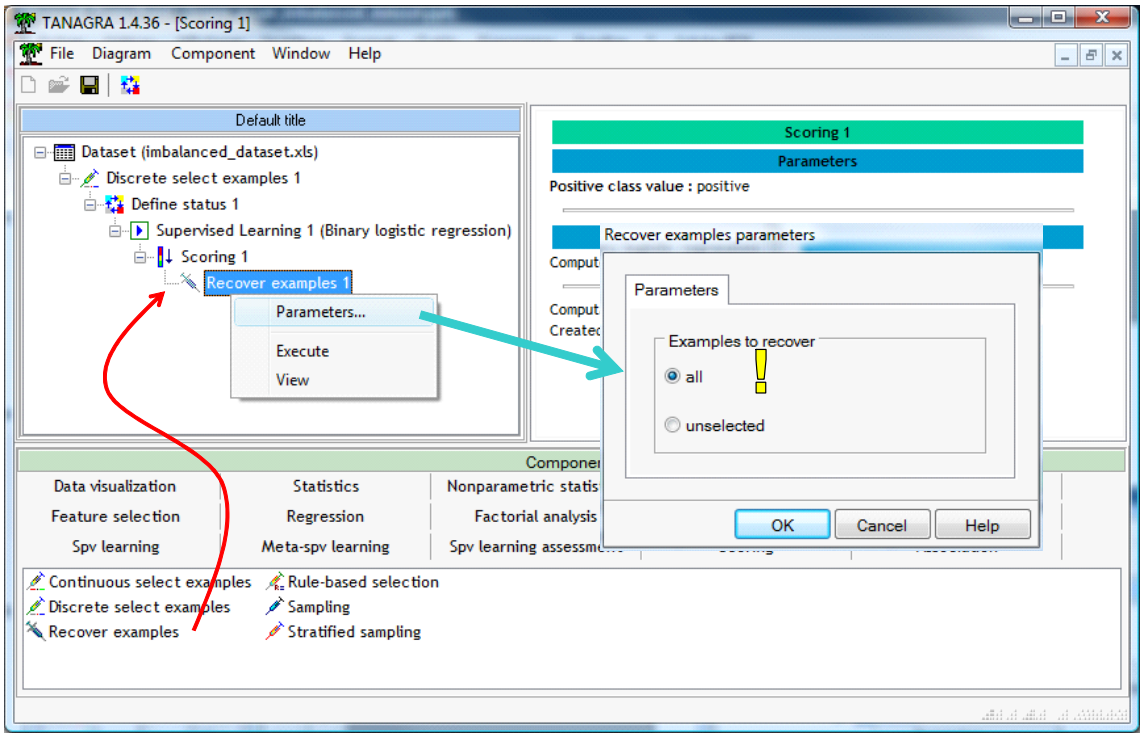
We note that the classifier is learned from the 900 instances of the learning sample. But the score is computed for all the instances of the database. A SCORE column is added to the current dataset.



3.4 Assessing the classifier on the test set

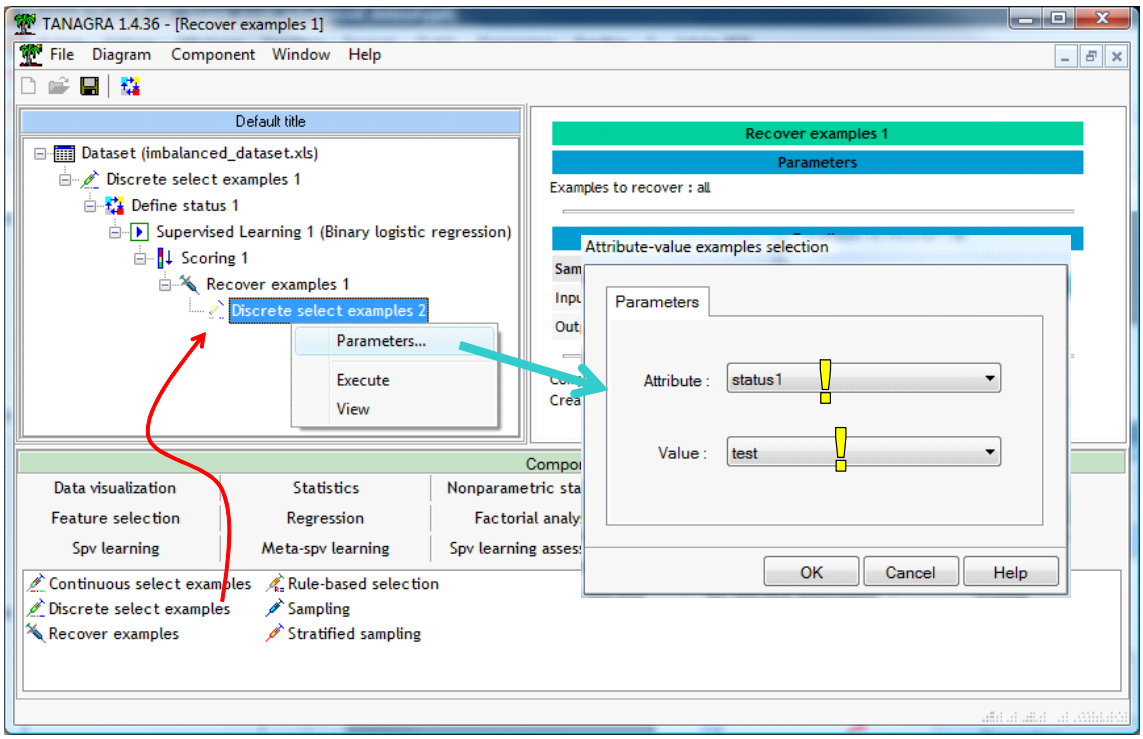
We use the test set (3000 instances) to evaluate the classifier. We activate this part of the dataset in the following way.

First, we activate all the instances. We insert the RECOVER EXAMPLES component (INSTANCE SELECTION tab) that we parameterize as follows.



All the instances (3900) are activated.

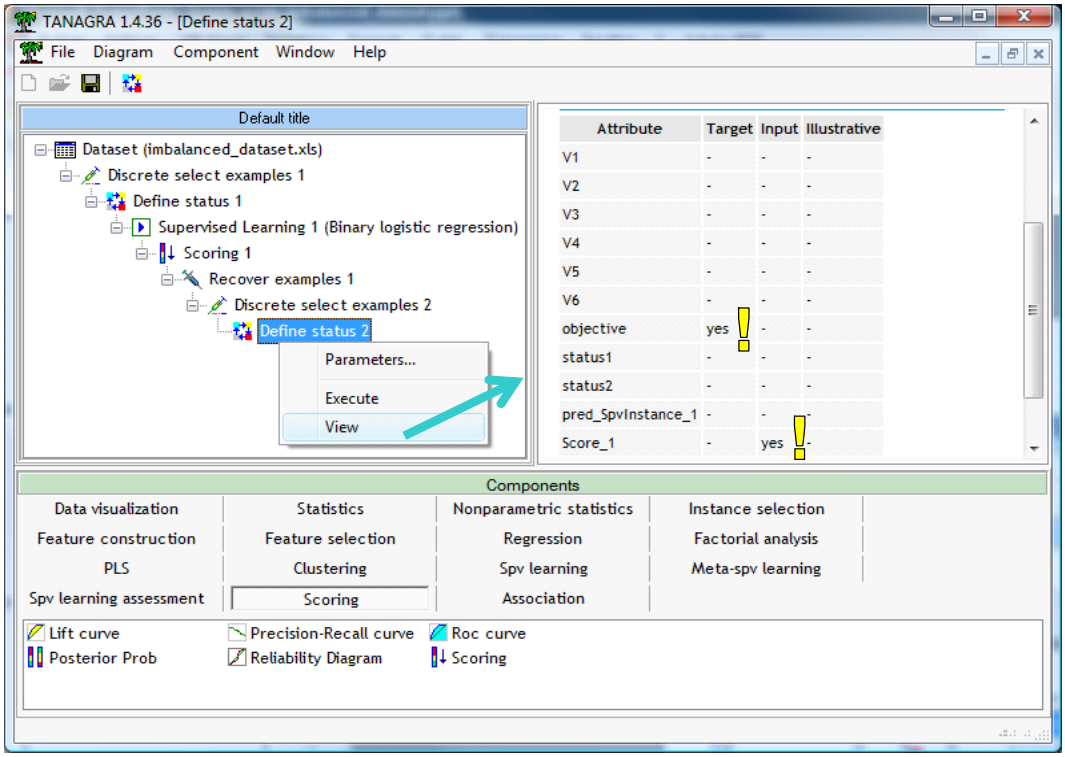
To select the test sample “STATUS1 = TEST”, we insert the DISCRETE SELECT EXAMPLES component (INSTANCE SELECTION tab). We set the following settings.



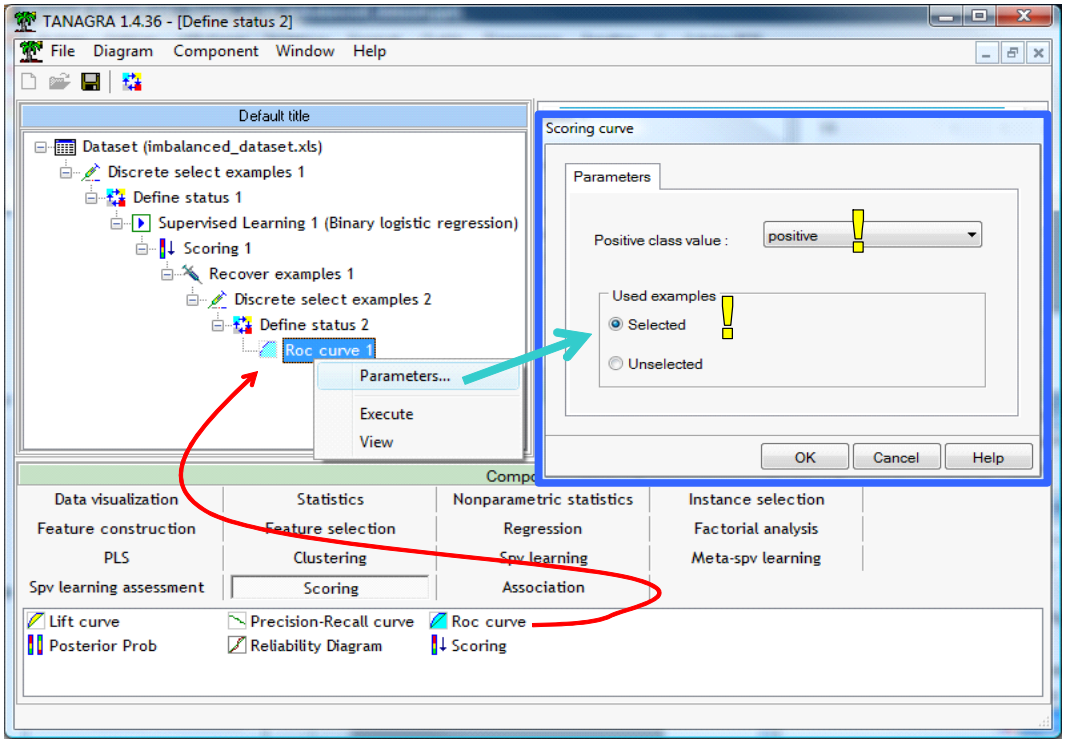
3000 instances (the test sample) are activated for the subsequent part of the diagram.

3.4.1 Constructing the ROC curve

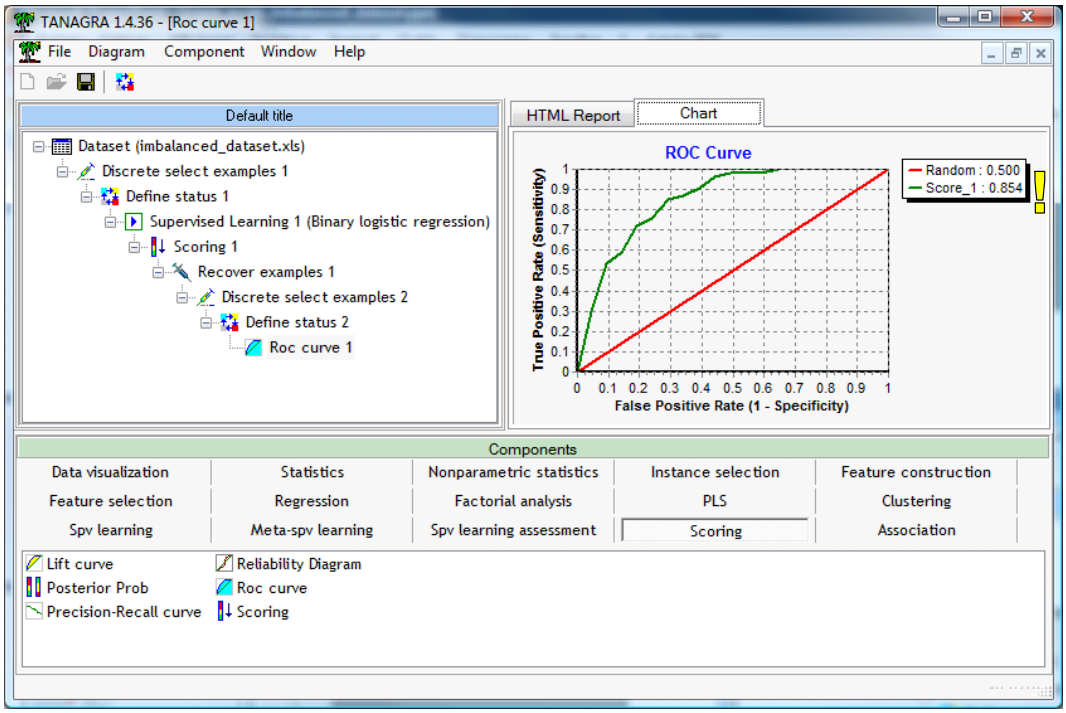
To elaborate the ROC curve, we need the target attribute and the score column. We insert the DEFINE STATUS component from the shortcut into the toolbar. We set OBJECTIVE as TARGET, SCORE_1 as INPUT.



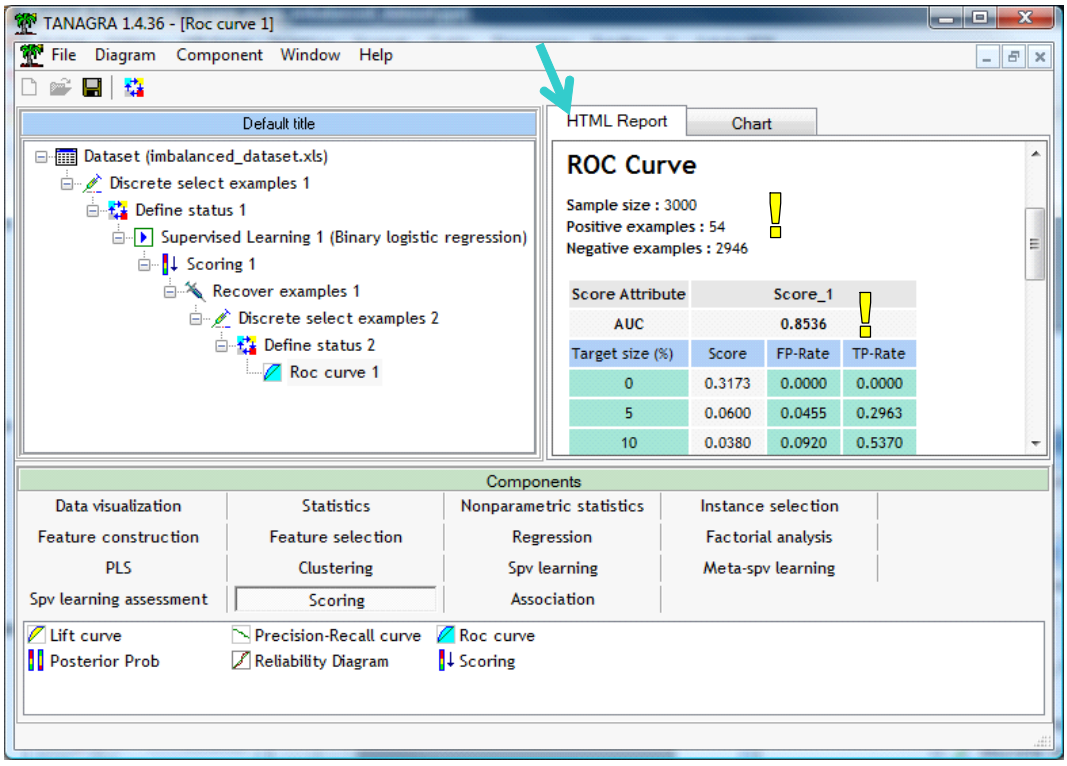
We add the ROC CURVE component (SCORING tab). We set the parameters which enable to identify the positive instances on the test set (3000 instances).



The ROC curve shows clearly that our classifier obtained from the logistic regression is not too bad. Tanagra supplies automatically the AUC criterion[4], we have 85.4% (the AUC of the default classifier would have been 50%).



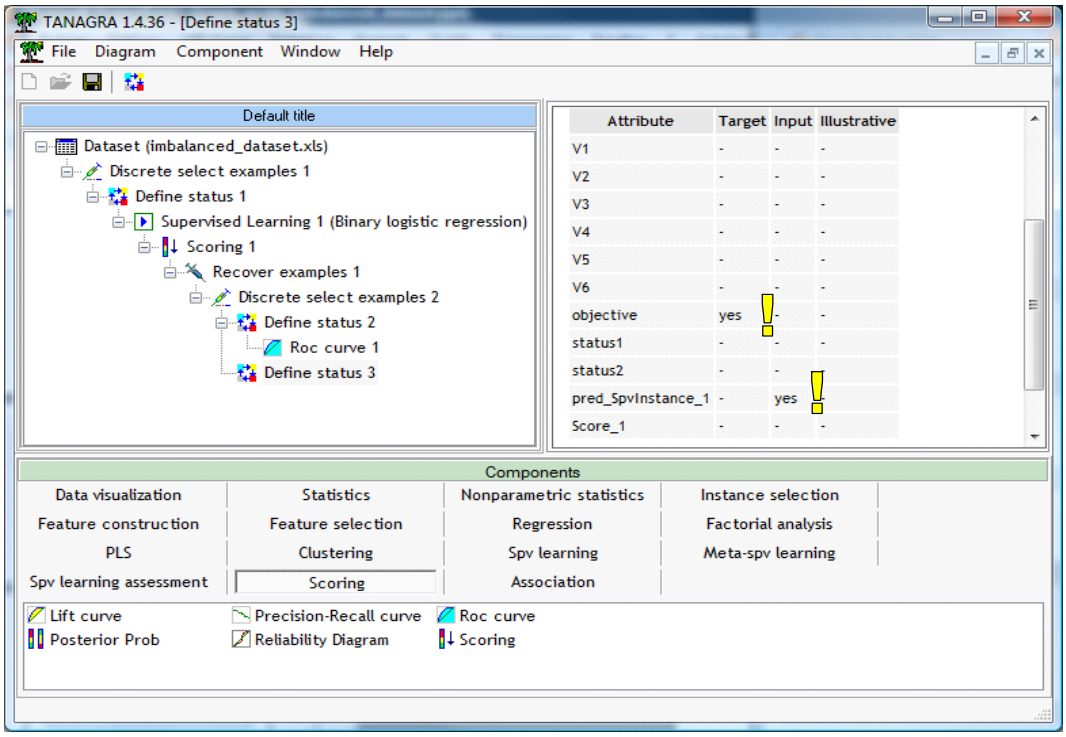
Into the "HTML REPORT", we have the details of calculations. The curve is actually computed on the test set. There were 54 positives among 3000 instances.



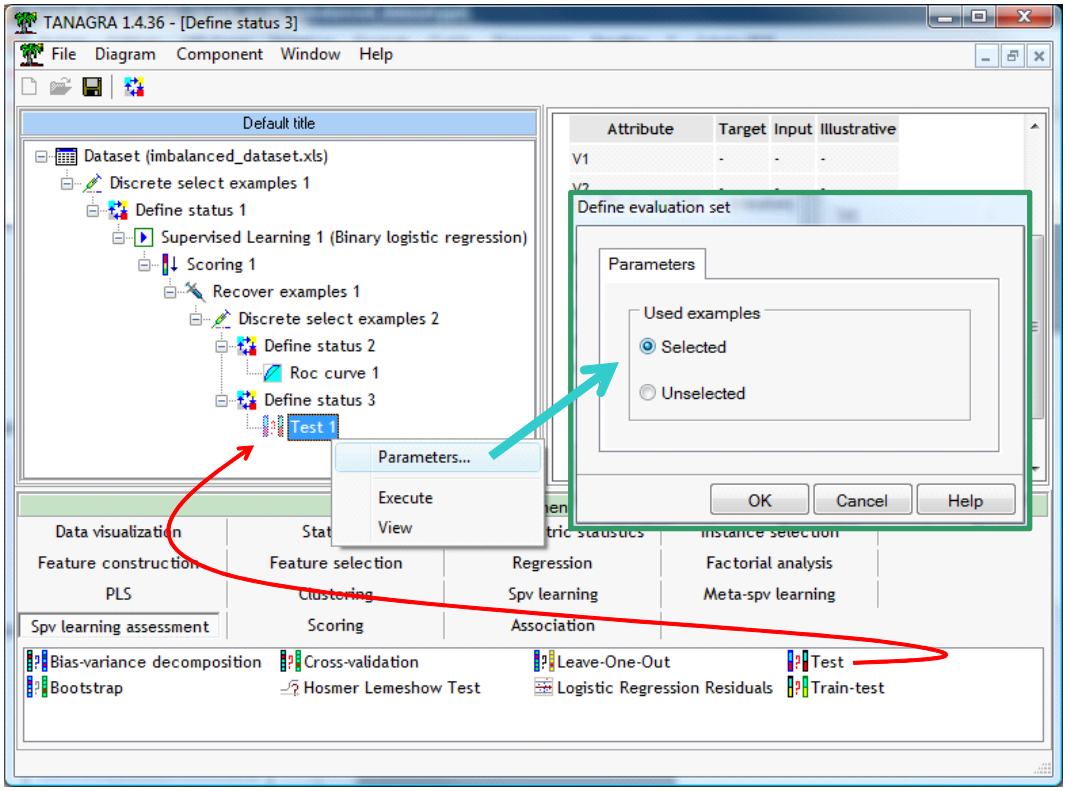
[4] About the AUC criterion, see http://en.wikipedia.org/wiki/Receiver_operating_characteristic

3.4.2 Test error rate

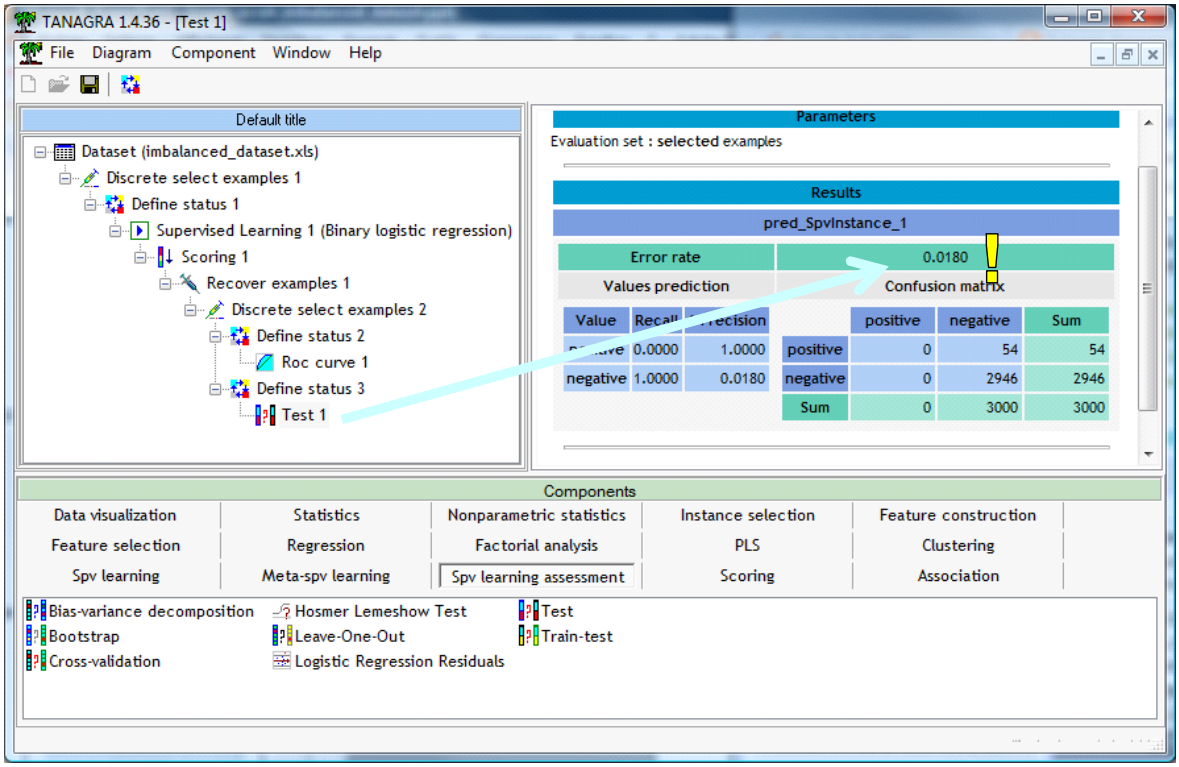
To compute the confusion matrix on the test set, we add again the DEFINE STATUS component. We set OBJECTIVE as TARGET attribute, and the prediction column PRED_SPV_INSTANCE_1 as INPUT.



Then, we add the TEST component (SPV LEARNING ASSESSMENT tab). We set the following settings in order to compute the matrix on the instances currently selected i.e. the test set of 3000 instances.



We click on the VIEW menu. Like on the learning set, all the instances are assigned to the negative class. The test error rate is 1.80%; the sensibility is 0% (0/54); the precision is undefined (0/0).



We note above all that the confusion matrix is not suitable to evaluate the performance of classifiers when we deal with imbalanced dataset. The ROC curve shows that we have an efficient model, better than the default classifier. The confusion matrix and the error rate are not able to show it in our very specific context.

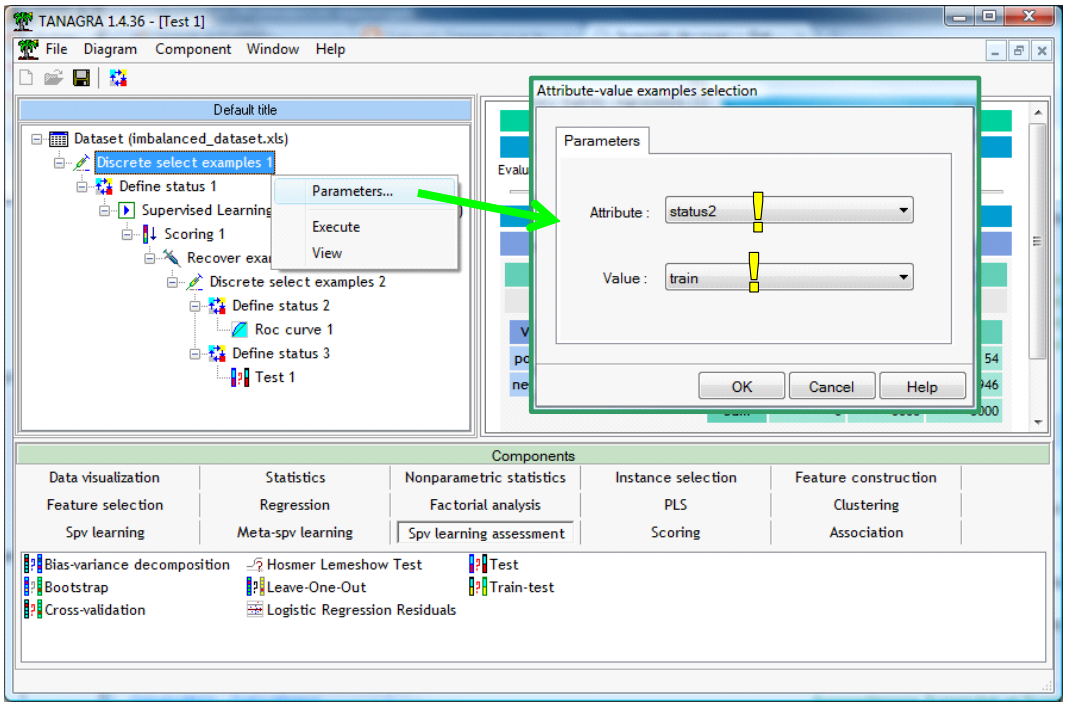
4 Learning from the balanced training set (M2)

In the context of imbalanced dataset, the majority of the data miners will tell you it is absolutely necessary to artificially balance the learning set. Given limited resources -- we can not use instances that do not exist -- a simple solution is to sample among the negative instances which are abundant. That's what we did in this section. Among the 900 individuals in the training sample, we have recovered the 15 positives (they are already scarce, we must use them all) and randomly extracted 15 instances within the negatives. Thus, the learning size is now 30 instances.

We come back at the root of the diagram. We must reproduce the sequence of operations defined above, while ensuring to work on the appropriate sample at each step.

4.1 Defining the new learning sample

We must specify the new learning set. We activate the PARAMETERS menu of DISCRETE SELECT EXAMPLES 1 into the diagram. We select now the instances corresponding to STATUS2 = TRAIN.

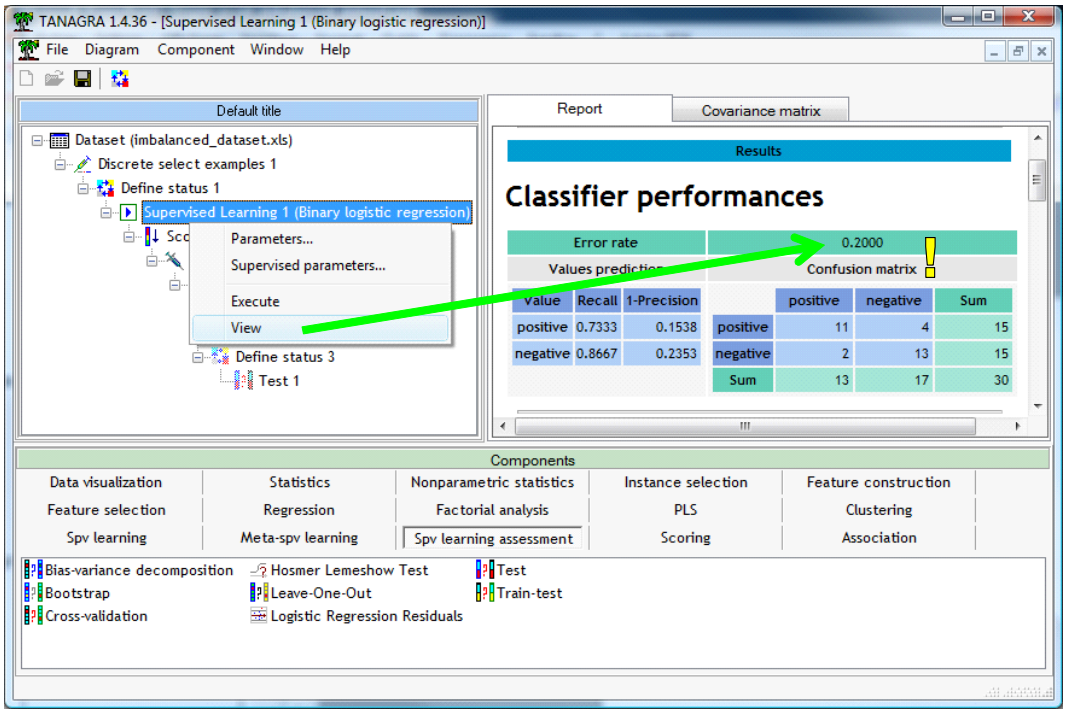


Tanagra shows that 30 instances (among 3900) are now selected for the learning process.

4.2 Training and assessing the classifier

4.2.1 Learning process

We select the SUPERVISED LEARNING 1 (BINARY LOGISTIC) component into the diagram. We click on the VIEW menu. The new classifier **M2**, created from the balanced learning set, is displayed.

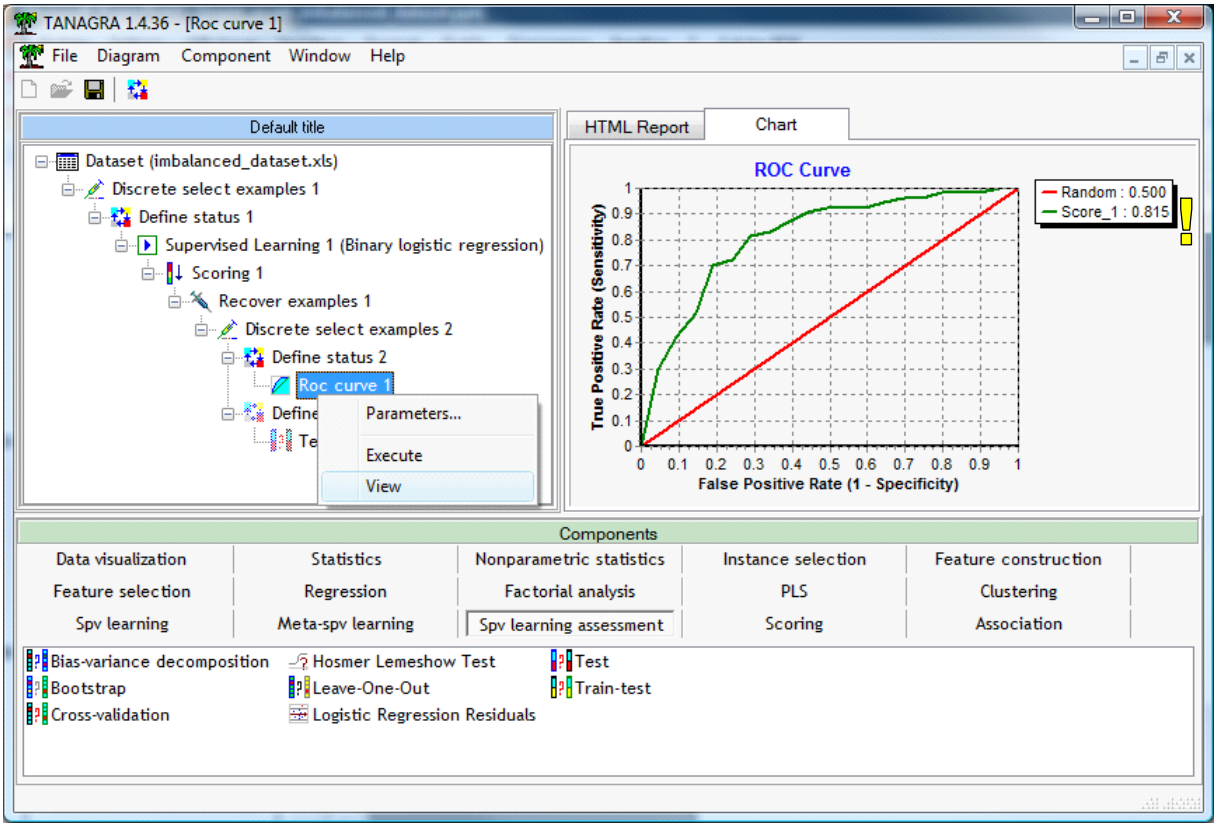


The resubstitution error rate is worse (20%) than the preceding classifier (M1). But at the same time, the sensibility is dramatically improved (73% = 11 / 15 vs. 0% previously). But we know that these values are questionable because we use the learning set that is moreover artificially balanced.

What is the behavior of the classifier on the test set?

4.2.2 ROC curve

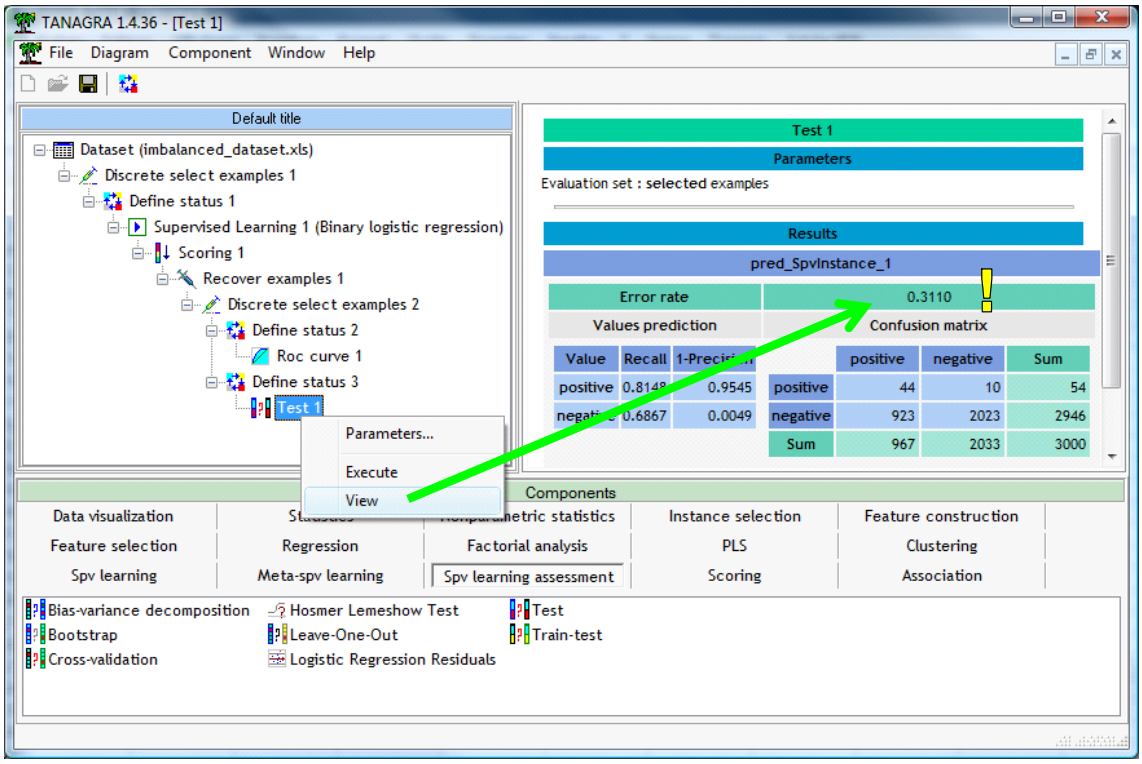
We click again on the VIEW menu for ROC CURVE 1.



In fact, M2 is not better than M1. The AUC criterion is 81.5%, it was previously 84.5%. The balancing of the learning set by downsizing has no effect on the ability of the classifier to assign higher score to positive instances. M2 is even slightly less efficient because we have reduced the training sample size.

4.2.3 Test error rate

We activate the VIEW menu of TEST 1. Like the confusion matrix on the training set, we note that the sensibility **(81.5% = 44/54)** of the classifier is mainly improved. But it was no consequence on the overall error rate **(31.1%)**, primarily because the precision is bad **(4.55% = 44/967)**.



5 Conclusion

Balancing a learning sample by under sampling among the majority class allows to improve the sensibility of a classifier. But it cannot improve the overall performance. We observe even that if we evaluate the classifier through its ability to assign higher scores to positive instances (using the ROC curve), the AUC criterion is decreased mainly because we use less instances (less information) for the construction of the classifier.

Last, we notice that we can improve the sensibility of the classifier M1 (learned from the imbalanced dataset) by changing the assignment threshold. Let $\pi(x)$ the posterior probability of the instance to be positive, the usual assignment rule is

$$\text{If } \pi(x) > 0.5 \text{ Then Y = + Else Y = -}$$

We can generalize the rule as follows

$$\text{If } \pi(x) > \theta \text{ Then Y = + Else Y = -}$$

θ is the threshold. When we increase the threshold, we increase mechanically the precision to the detriment of the sensibility; when we decrease θ, we improve the sensibility instead of the precision.

For our dataset, if we set $\theta = 0.017005$ for the classifier M1, we obtain the following confusion matrix on the test set.

Nombre de objective	pred.theta.corrected		
objective	positive	negative	Total
positive	44	10	54
negative	767	2179	2946
Total	811	2189	3000

For the same sensibility as M2 (81.5% = 44/54), the precision is better (5.42% = 44/8111 vs. 4.55%).