1 Subject

Implementation of the logistic regression with various Data Mining Tools.

Logistic regression (http://en.wikipedia.org/wiki/Logistic_regression) is a popular supervised learning method.

There are several reasons for this. The theoretical foundation of the method is attractive. It is in line with the generalized regression. Thus the logistic regression is a well identified variant which one can implement according the kind of the dependent variable (class attribute). Their performance in prediction is comparable to the other approaches. Furthermore, it puts forward some indicators for the interpretation of the results. Among them, the famous odds-ratio enables to identify precisely the contribution of each predictor.

Logistic regression is available in many free tools. In this tutorial, we compare the implementation of this technique with Tanagra 1.4.27, R 2.7.2 (GLM command), Orange 1.0b2, Weka 3.5.6, and the package RWeka 0.3-13 for R. Beyond the comparison, this tutorial is also an opportunity to show how to achieve the succession of operations with these tools: importing an ARFF file (Weka file format); split the data into learning and test set; computing the predictive model on the learning set; testing the model on the test set; selecting the relevant variable using criterion in agreement with the logistic regression; evaluating again the performance of the simplified model.

2 Dataset

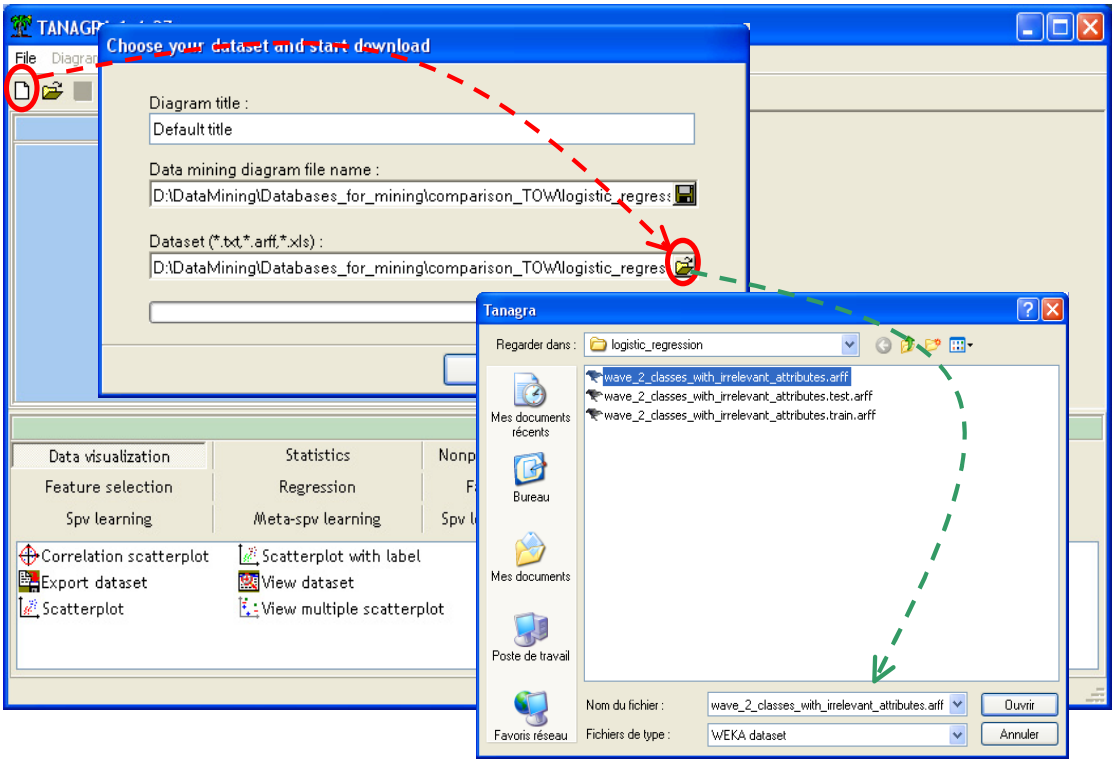
We use the WAVE dataset (Breiman and al., 1984), but we retain only 2 values of the class attribute (http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/wave_2_classes_with_irrelevant_attributes.zip). To the 21 original predictors (V1...V21), we add 100 "noise" variables (ALEA). They are completely independent to the problem. Thus, they should be removed during the variable selection process.

There are 33,334 examples. The column "SAMPLE" identifies the 10,000 examples for the learning part; the others are used for the testing phase. Because we use the same test sample for the various tools, the results will be comparable.

3 Logistic regression and feature selection

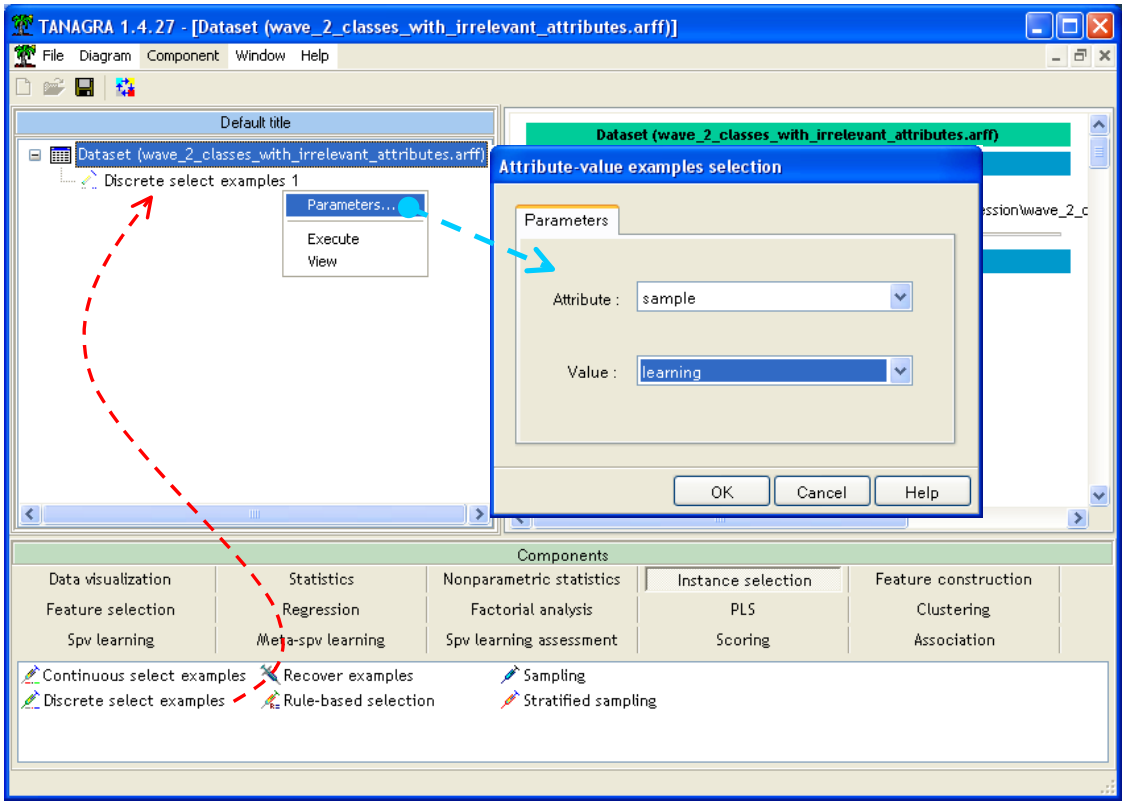
3.1 TANAGRA

After launching TANAGRA, we click on the FILE / NEW menu in order to create a diagram and import the data file. We select « wave_2_classes_with_irrelevant_attributes.arff ». ARFF is the WEKA file format. This is a text file with some specific indications about the variables and their type.



3.1.1 Splitting the dataset in a train and test sets

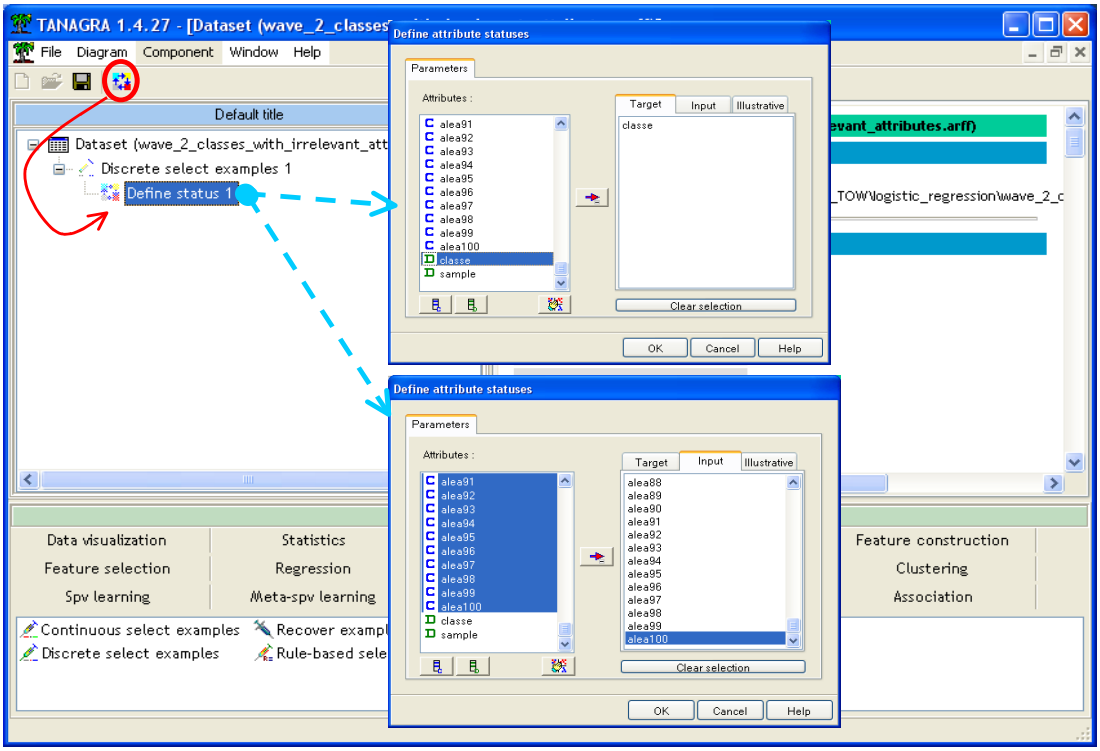
We insert the DISCRETE SELECT EXAMPLES (INSTANCE SELECTION tab) into the diagram. By clicking on the PARAMETERS menu, we select the SAMPLE column and the LEARNING value.



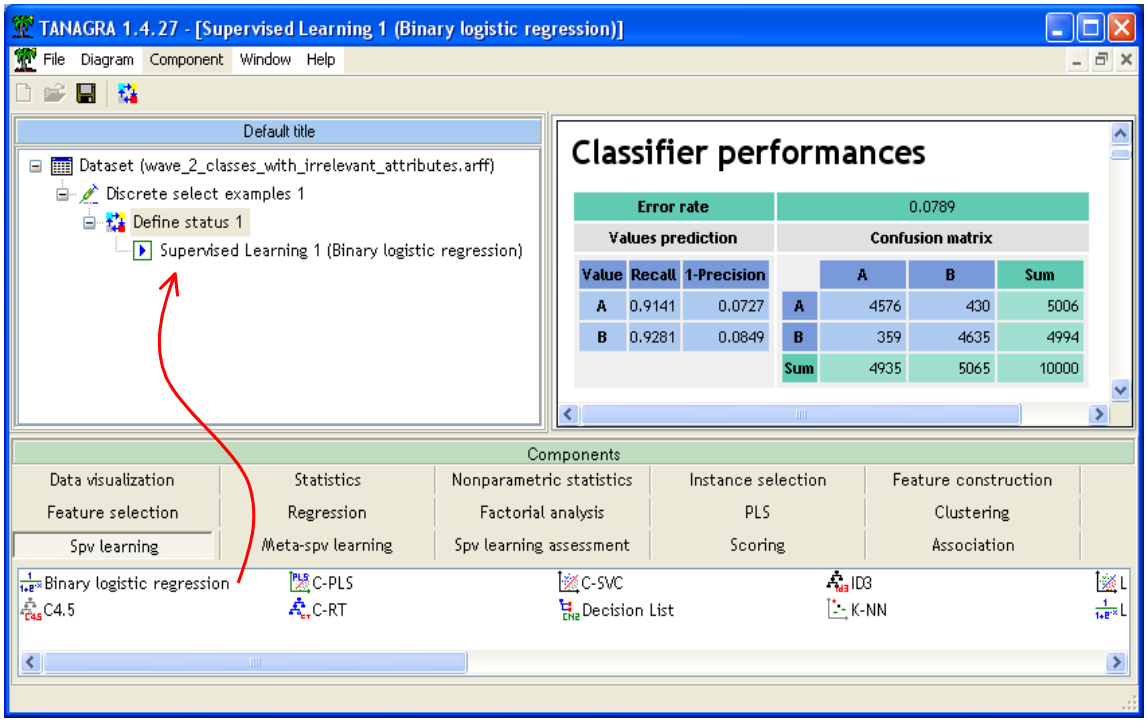
We click on the VIEW menu: 10,000 examples are now used for the learning process.

3.1.2 Binary Logistic Regression

The DEFINE STATUS component enables to define the TARGET (CLASSE) and the INPUT (V1...ALEA100) attributes. We must not select the SAMPLE column here.



We insert the BINARY LOGISTIC REGRESSION (SPV LEARINING tab) component into the diagram.



We click on VIEW in order to obtain the results.

The resubstitution error rate is 7.89%, the deviance -2LL = 3586.688. If we want to compare models with different number of parameters, it is better to minimize the SC criterion (BIC = 4710.35).

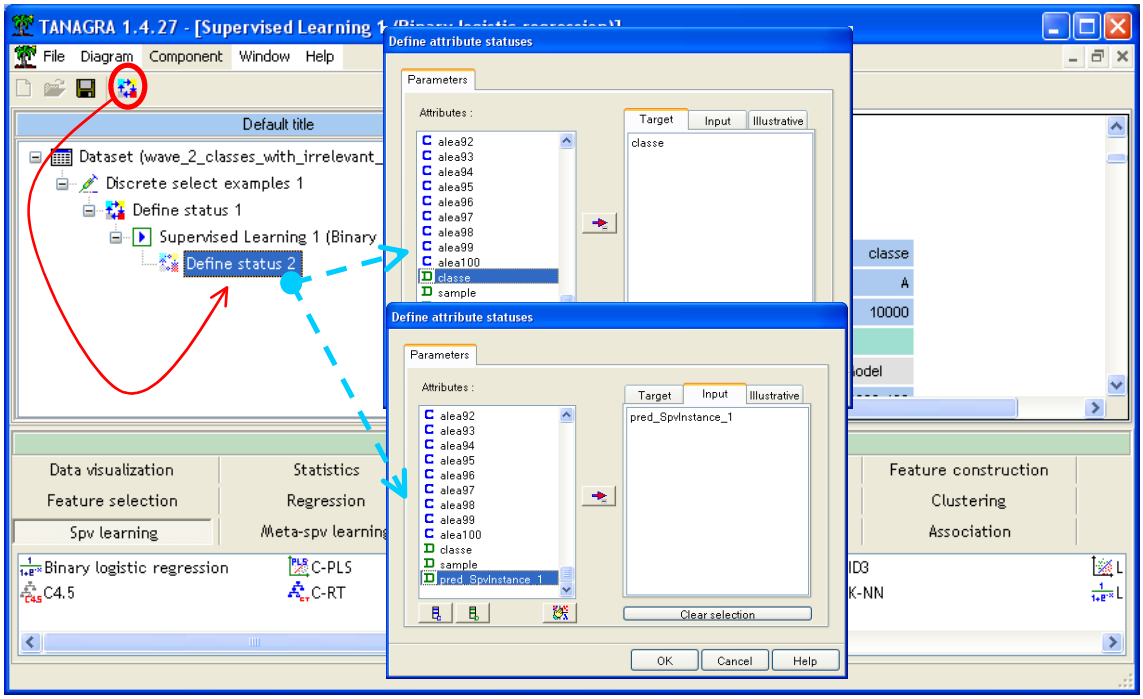
Adjustement quality

Predicted attribute		classe
Positive value		A
Number of examples		10000
Model Fit Statistics		
Criterion	Intercept	Model
AIC	13864.929	3830.688
SC	13872.140	4710.350
-2LL	13862.929	3586.688
Model Chi² test (LR)		
Chi-2		10276.2407
d.f.		121
P(>Chi-2)		0.0000
R²-like		
McFadden's R²		0.7413
Cox and Snell's R²		0.6421
Nagelkerke's R²		0.8562

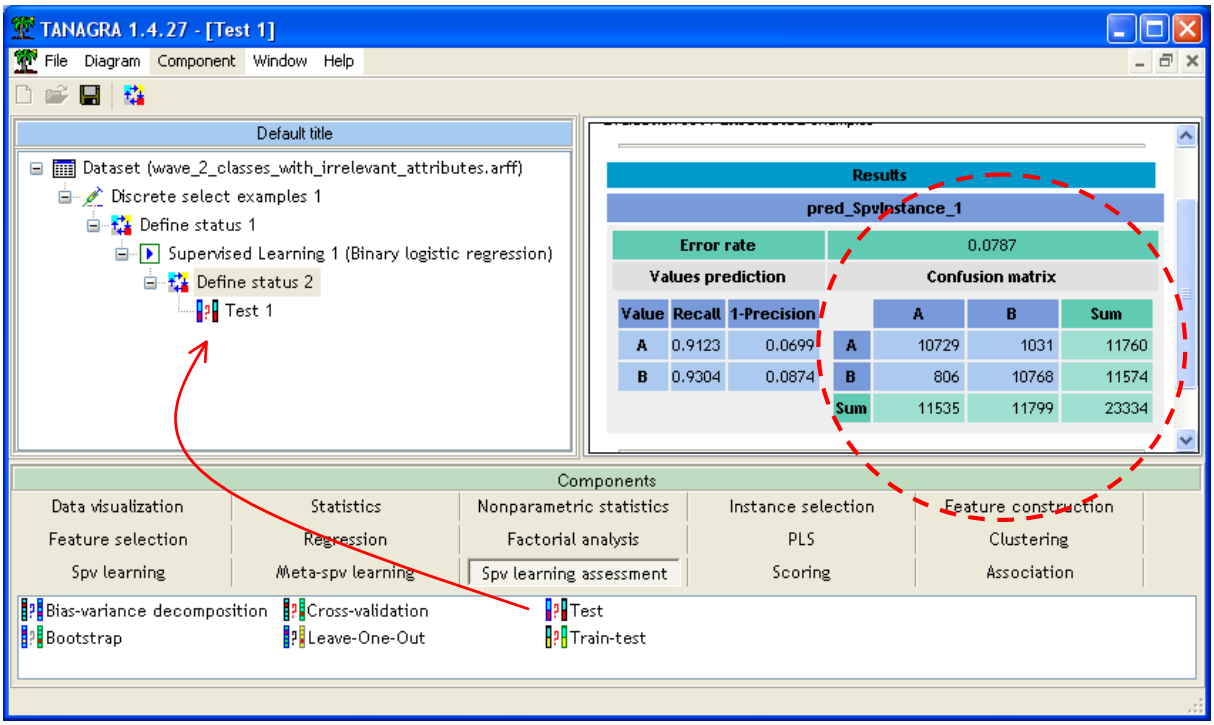
In the low part of the window, we see the estimated parameters. Some ALEA variables seem surprisingly significant. We will see if they will be removed during the selection process.

3.1.3 Measuring the test error rate

We want to evaluate the model on the test set. We add again a DEFINE STATUS component into the diagram. We set CLASSE as TARGET, the predicted values PRED_SPV_INSTANCE_1 as INPUT.



Then, we insert the TEST component (SPV LEARNING ASSESSMENT tab). The calculation is automatically made on the unselected examples i.e. on the test sample (23,334 examples).



The test error rate is 7.87%, there are (1,031 + 806) = 1,837 misclassified examples among 23,334.

3.1.4 Forward selection

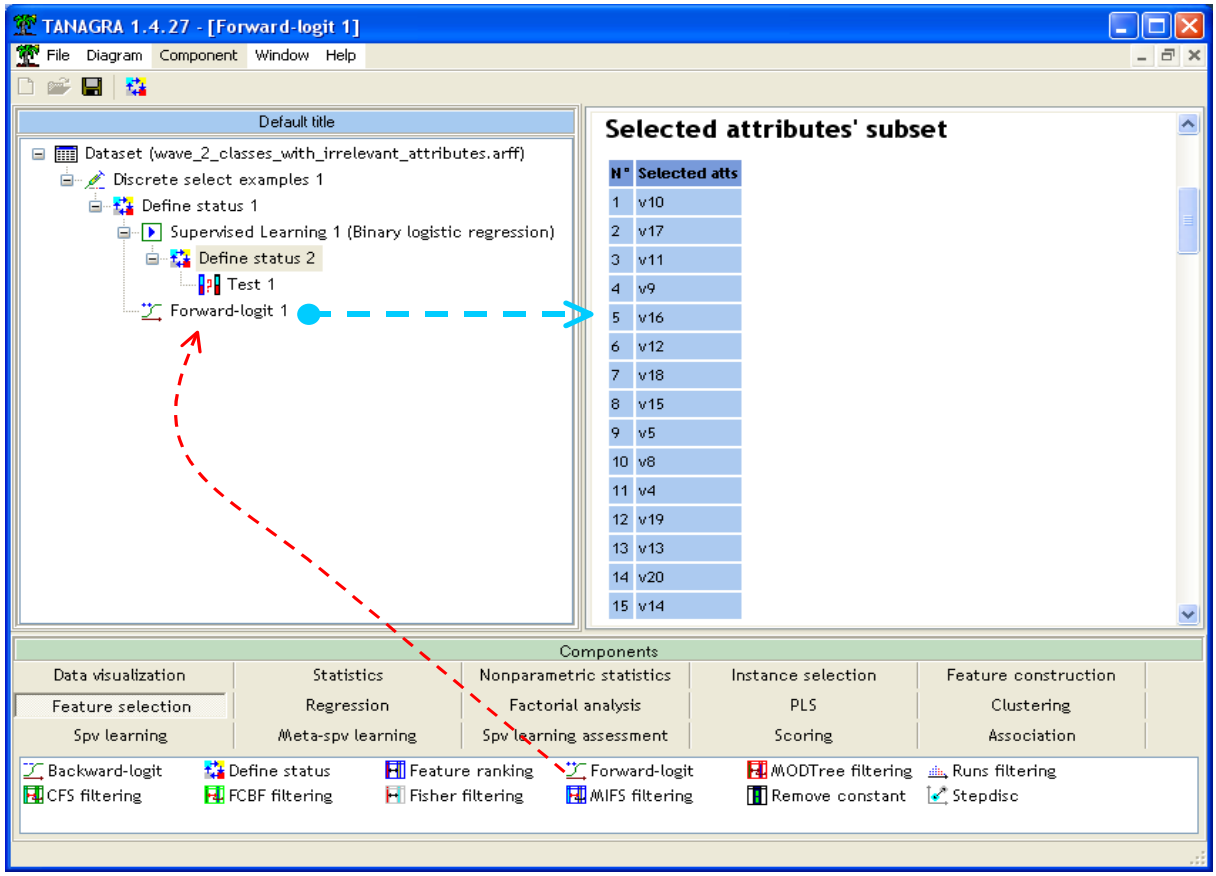
In a modeling process, the fewer the number of selected features, the easier is the model interpretation. In addition, according to the Occam Razor principle, among two models with the same performance, we will always prefer the simpler model because it is always more robust.

We would perform a forward selection on our dataset, according to the SCORE test (See http://faculty.chass.ncsu.edu/garson/PA765/logistic.htm#stepwise - Rao's efficient score statistic according other software). The main advantage of this approach is that, in the worst case i.e. all the "p" variables are selected, we perform only "p" logistic regression i.e. "p" optimizations of the likelihood criterion.

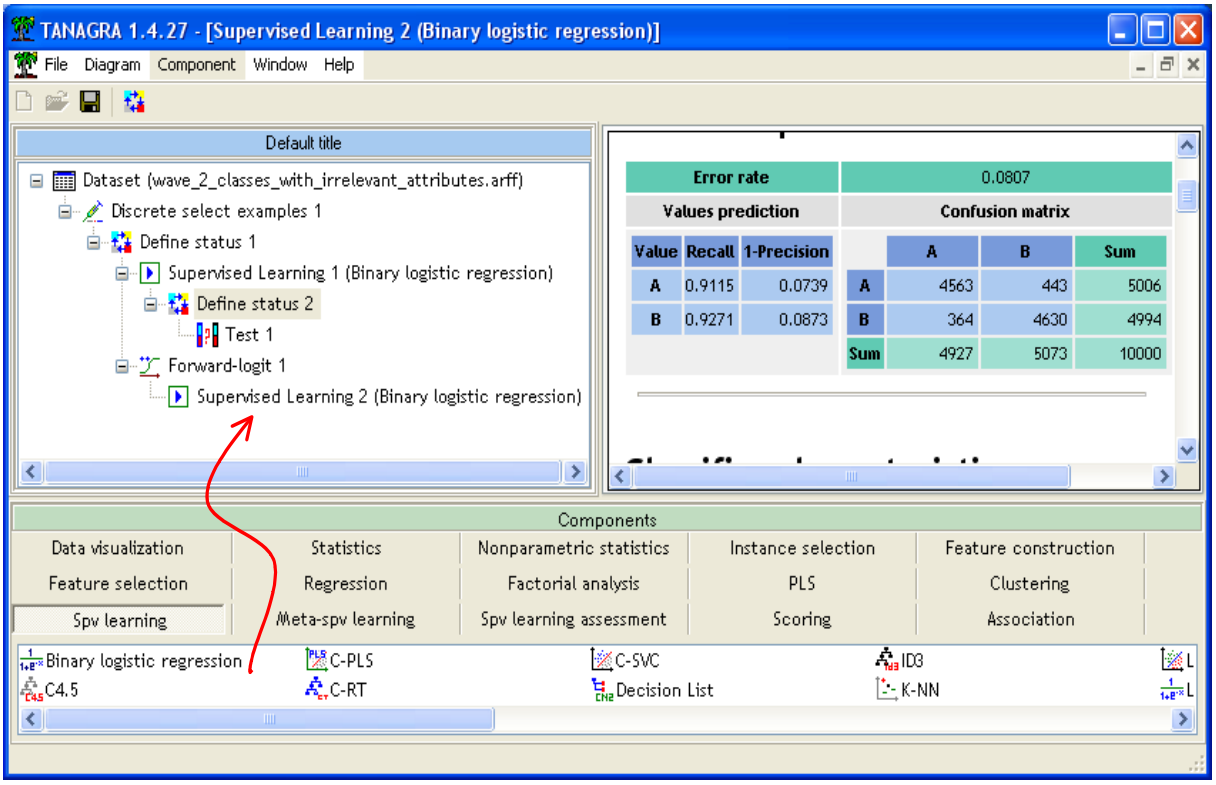
We insert the FORWARD LOGIT component (FEATURE SELECTION tab) into the diagram. We click on the VIEW menu. TANAGRA performs a forward feature selection based on the SCORE TEST; the significance level is 1%. We can modify this parameter. We can also specify the maximum number of selected variables.

After about 60 seconds on my computer (Dual Core 3.8 MHz under Windows XP), 15 variables are selected. Fortunately, none of the ALEA variables has been selected.

In the low part of the window, the details of the selection process are displayed. We can see the 5 best variables for each step.

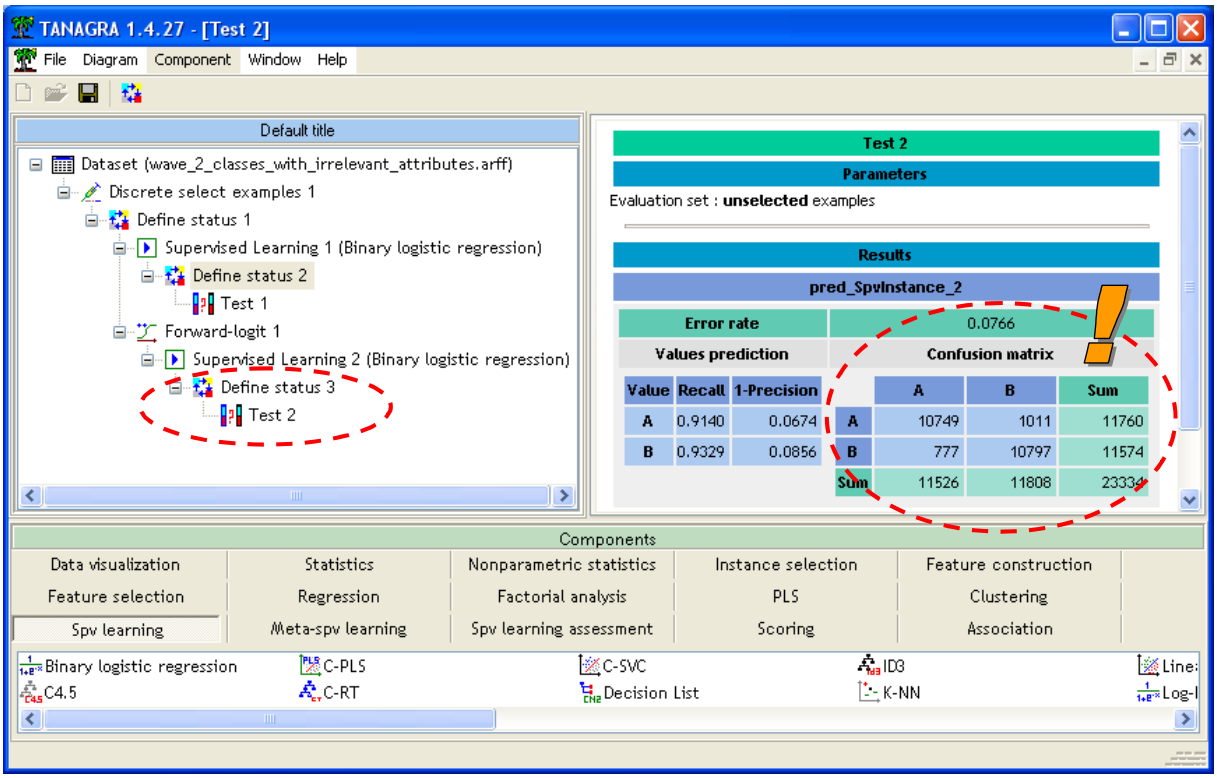


In order to measure the performance of the model using these variables, we perform again the logistic regression with the BINARY LOGISTIC REGRESSION component (SPV LEARNING tab)



The resubstitution error rate is 8.07%, le deviance -2LL = 3678.061. But these criteria are not useful. More interesting is the SC criterion. We obtain SC = 3825.427, much better than the previous model.

We would measure the test error rate. We insert the DEFINE STATUS component. We set CLASSE as TARGET, PRED_SPVINSTANCE_2 as INPUT. We then add the TEST (SPV LEARNING ASSESSMENT tab) component.



The test error rate, computed on the test set, is 7.66% (1011 + 777 = 1788 misclassified examples). In comparison with the previous model, we have much less variables (15 vs. 121) and about the same test error rate (7.66% vs. 7.87%). The new model is definitely preferable.

3.2 R (the GLM command)

R (http://www.r-project.org/) can handle the ARFF file format using the RWeka package. We set the following instructions in order to read and check the data file.

```
library(RWeka)

#charger les données
setwd("D:/DataMining/Databases_for_mining/comparison_TOW/logistic_regression")
donnees <- read.arff(file("wave_2_classes_with_irrelevant_attributes.arff","r"))

#petite vérification sur la classe et la variable de partition learning-test
summary(donnees[,c("classe","sample")])
```

R gives the following results.

```
> library(RWeka)
Le chargement a nécessité le package : rJava
Le chargement a nécessité le package : grid
>
> #charger les données
> setwd("D:/DataMining/Databases_for_mining/comparison_TOW/logistic_regression")
> donnees <- read.arff(file("wave_2_classes_with_irrelevant_attributes.arff","r"))
>
> #petite vérification sur la classe et la variable de partition learning-test
> summary(donnees[,c("classe","sample")])
 classe          sample
 A:16766   learning:10000
 B:16568   test    :23334
```

3.2.1 Partitioning the dataset in a train and test set

We utilize the SAMPLE column in order to subdivide the dataset. We create 2 "data.frame".

```
> #partitionner en apprentissage-test à l'aide de la colonne sample
> donnees.app <- donnees[donnees$sample=="learning",1:122]
> donnees.test <- donnees[donnees$sample=="test",1:122]
>
> nrow(donnees.app)
[1] 10000
> nrow(donnees.test)
[1] 23334
```

3.2.2 Logistic regression with R

We launch the logistic regression using the glm(.) command.

```
#régression logistique
modele <- glm(classe ~., data = donnees.app, family = binomial)
summary(modele)
```

The obtained deviance -2LL = 3586.7 is the same as Tanagra. We note that R is really fast.

```
    Null deviance: 13862.9  on 9999  degrees of freedom
Residual deviance:  3586.7  on 9878  degrees of freedom
```

3.2.3 Calculation of the test error rate

We want to apply the model on the test set. We use the **predict(.)** function. It computes the posterior probability for each example. The threshold 0.5 enables to specify the prediction PRED of the class attribute (A or B).

```
#prédiction sur les données test
proba <- predict(modele, newdata = donnees.test,type="response")
pred <- ifelse(proba < 0.5, 1, 2)
pred <- factor(pred)
```

By comparing the true class value and the predicted class value, we obtain the confusion matrix and the error rate.

```
#matrice de confusion
mc <- table(donnees.test$classe,pred)
print(mc)

#taux d'erreur en test
erreur <- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
print(erreur)
```

Here are the R outputs. The test error rate is the same as Tanagra 7.87%. There are 1,837 misclassified examples among 23,334.

```
> #matrice de confusion
> mc <- table(donnees.test$classe,pred)
> print(mc)
   pred
      1     2
  A 10729  1031
  B   806 10768
>
> #taux d'erreur en test
> erreur <- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
> print(erreur)
[1] 0.07872632
```

3.2.4 Feature selection with the stepAIC(.) command

The **stepAIC(.)** command from the **MASS** package enables to perform a feature selection. We want to minimize the BIC criterion using a forward selection strategy. The k parameter into the following screenshot enables to specify the utilization of the BIC criterion [k = log(n)] instead of the AIC criterion [k = 2] during the process.

```
#bibliothèque pour stepAIC
library(MASS)

#sélection forward basé sur l'optimisation du BIC
modele.initial <- glm(classe ~ 1, data = donnees.app, family = binomial)

modele.forward <- stepAIC(modele.initial,scope=list(lower="classe~1",
                  upper=as.formula(modele)),trace=TRUE,direction="forward",
                  k=log(nrow(donnees.app)))
```

In comparison with the approach based on the SCORE test, this approach needs much more computational resources. If all the p variables are selected, R performs [p x (p – 1) / 2] optimizations of the log-likelihood criterion.

We obtain the following model with 15 predictors.

```
Step:  AIC=3825.43
classe ~ v10 + v17 + v11 + v9 + v16 + v12 + v18 + v15 + v5 +
    v8 + v4 + v19 + v13 + v20 + v14
```

There are the same predictors as Tanagra. The BIC criterion is also the same one i.e. BIC = 3825.43 (Note: Even if R shows the AIC criterion, because we have modified the k parameter, this is indeed the BIC criterion).

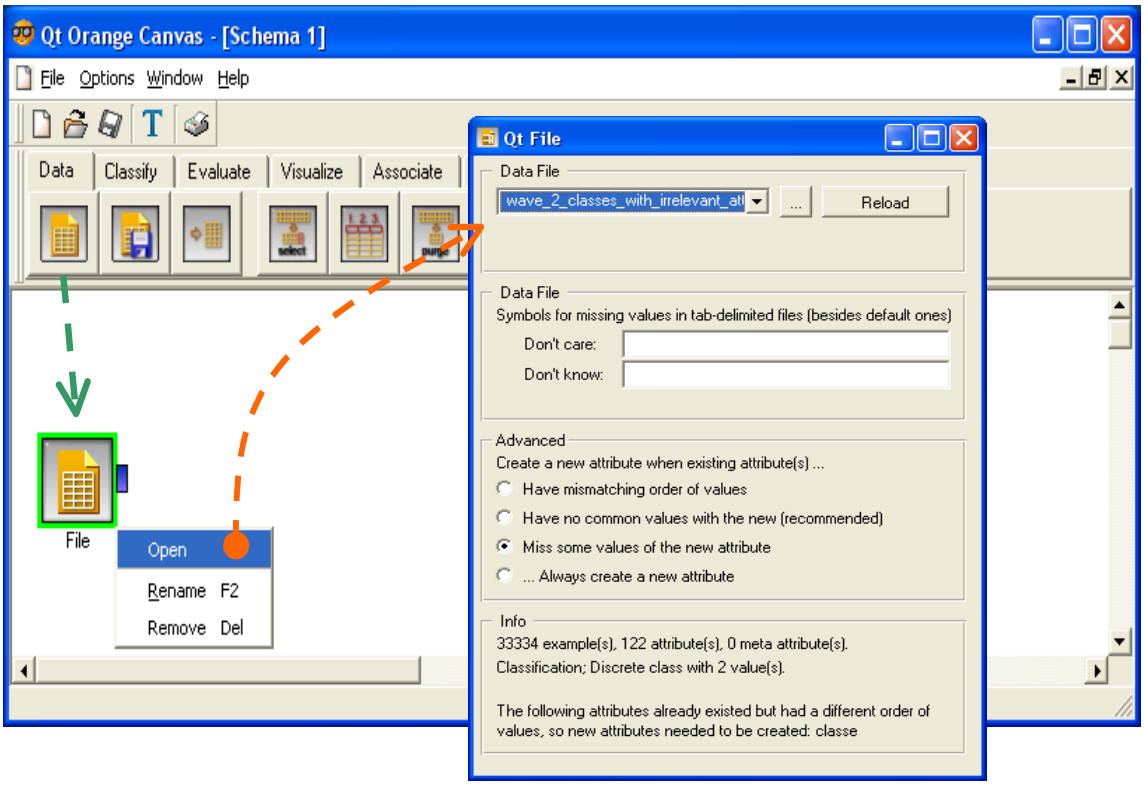
Applying the model on the test set gives the test error rate. We obtain 7.66%.

```
> #idem, prédiction sur fichier test + matrice de confusion + taux d'erreur
> proba <- predict(modele.forward, newdata = donnees.test,type="response")
> pred <- ifelse(proba < 0.5, 1, 2)
> pred <- factor(pred)
>
> mc <- table(donnees.test$classe,pred)
> print(mc)
   pred
       1     2
  A 10749  1011
  B   777 10797
>
> erreur <- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
> print(erreur)
[1] 0.07662638
```

3.3 ORANGE

Orange (http://www.ailab.si/orange/) is a nice software with very interesting functionalities. We use the "schema" execution mode in this tutorial.

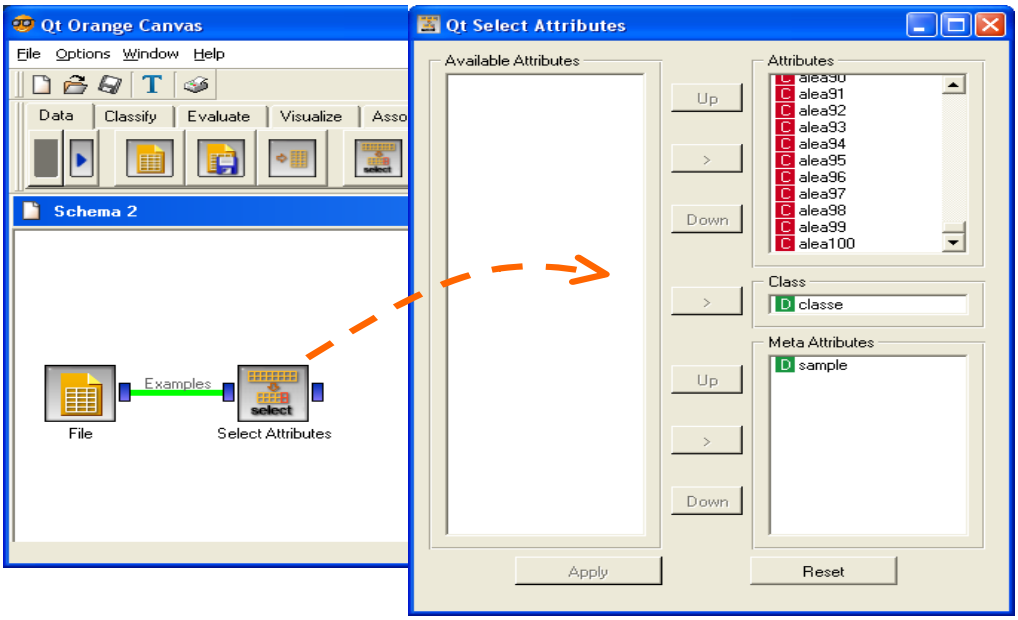
After launching Orange, we import the data file using the FILE (DATA tab) component. We select the "wave_2_classes_with_irrelevant_attributes.arff" file.



122 descriptors, 1 class attribute and 33,334 examples are loaded.

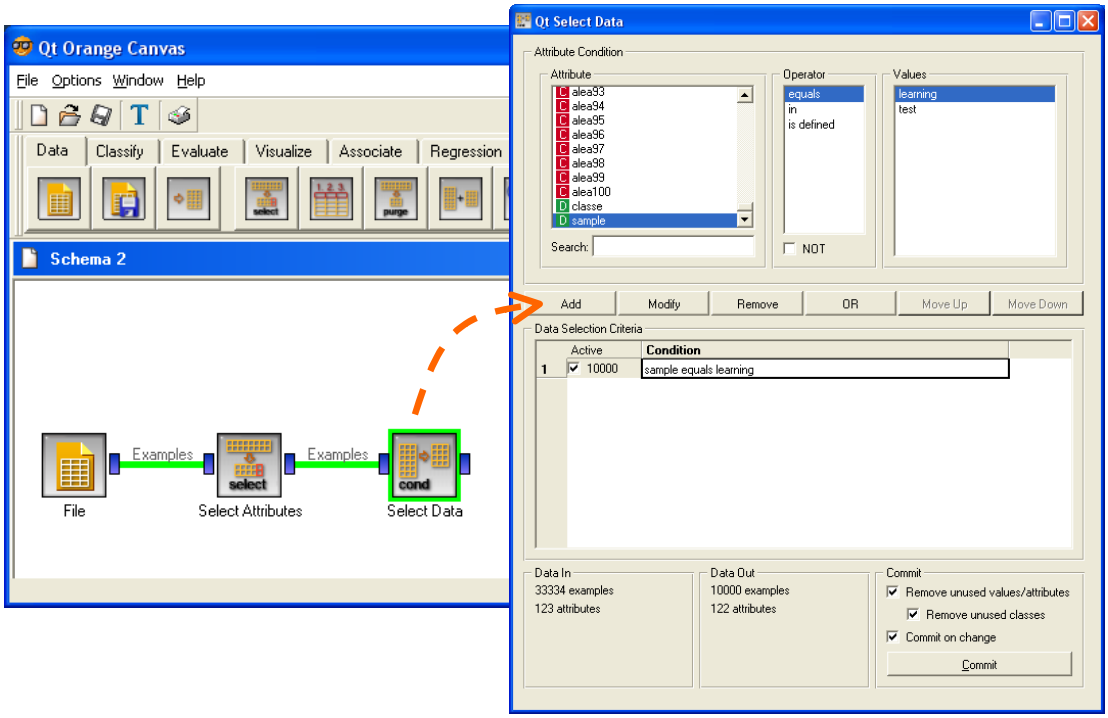
3.3.1 Specifying the status of the variables

We use the SELECT ATTRIBUTES component (DATA tab). After we connect the previous component to this last one, we click on the OPEN menu and set the following settings. Note: We must specify SAMPLE as a META ATTRIBUTES; otherwise we cannot use this column below.



3.3.2 Data partition

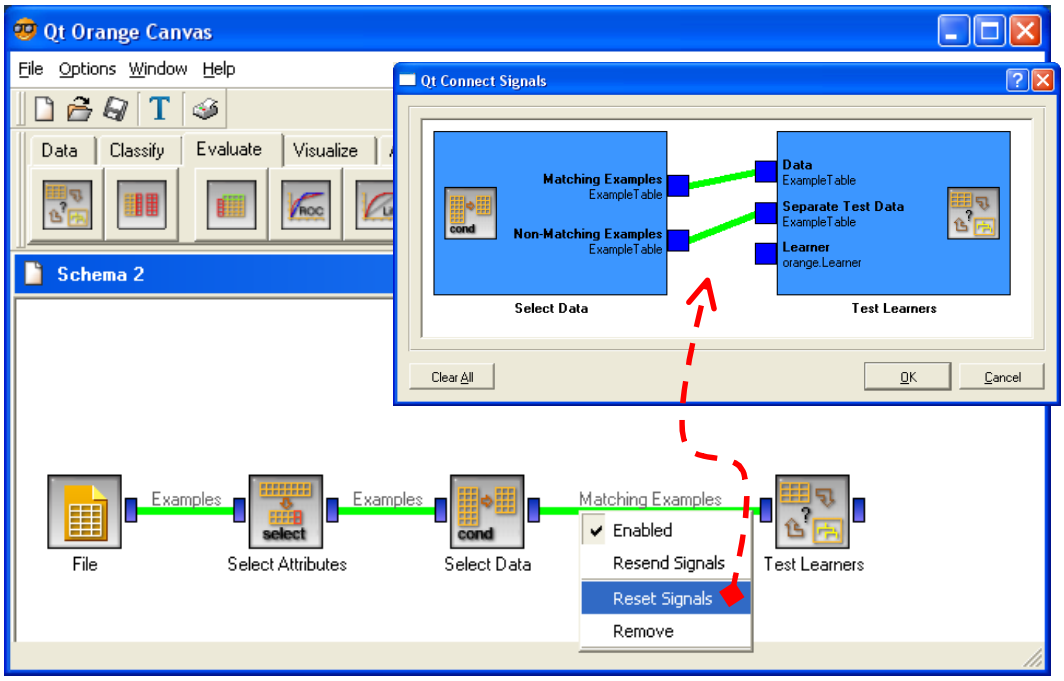
The SELECT DATA component (DATA tab) enables to select the learning and the test sample using the SAMPLE column. We connect the component to the previous one and click on the OPEN menu.



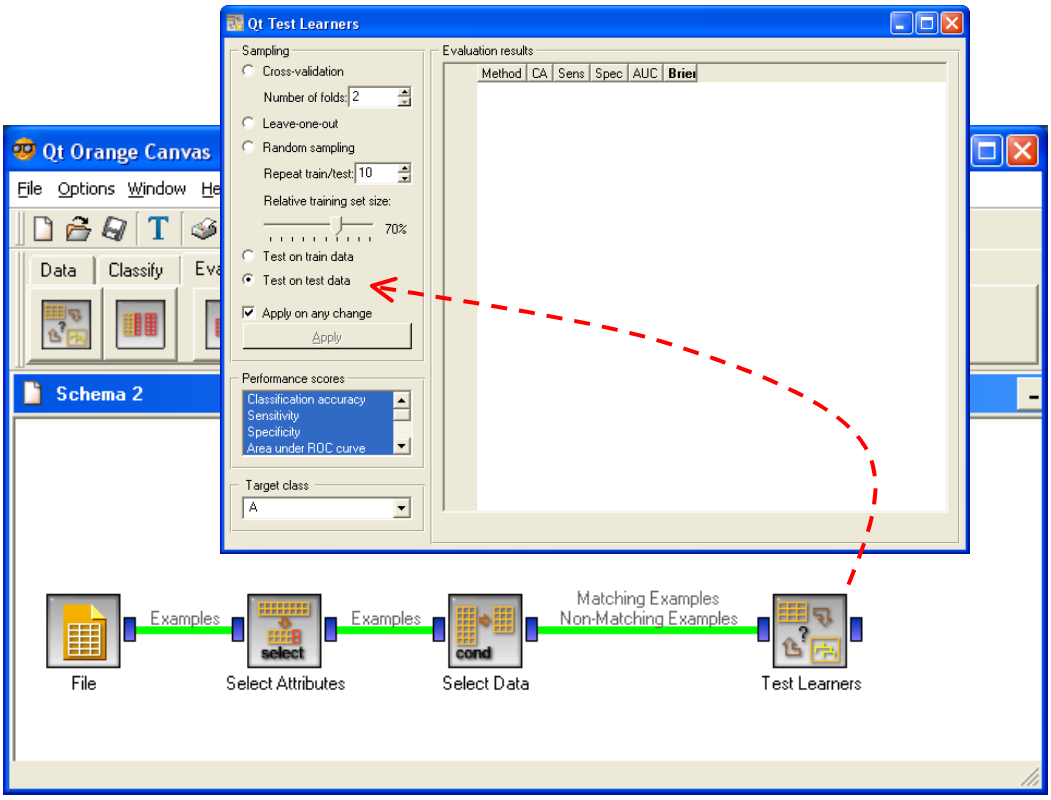
We set « SAMPLE equals LEARNING » in order to select the learning sample.

3.3.3 Preparing the test component

Curiously, it is appropriate at this step to insert the TEST LEARNERS component (EVALUATE tab). We connect the component previous … and it is not enough, we have to define more finely the connection by activating the RESET SIGNALS menu (right click on the connection).

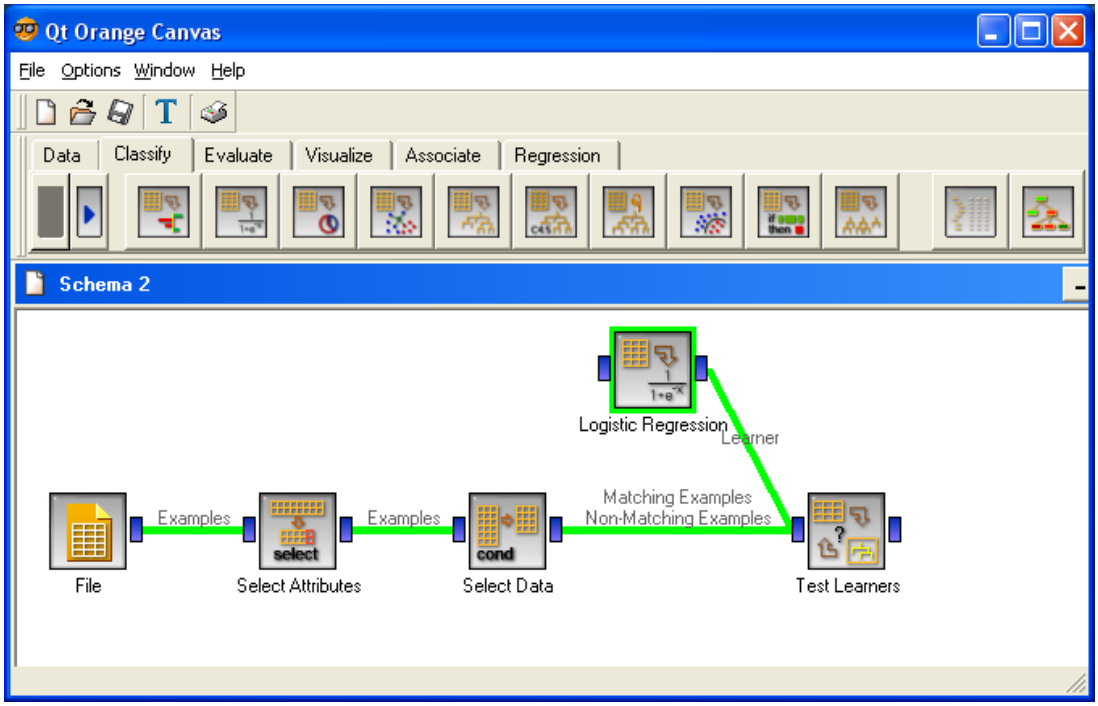


Then, we click on the OPEN menu of TEST LEARNERS. We specify that the error rate must be computed on the test set.

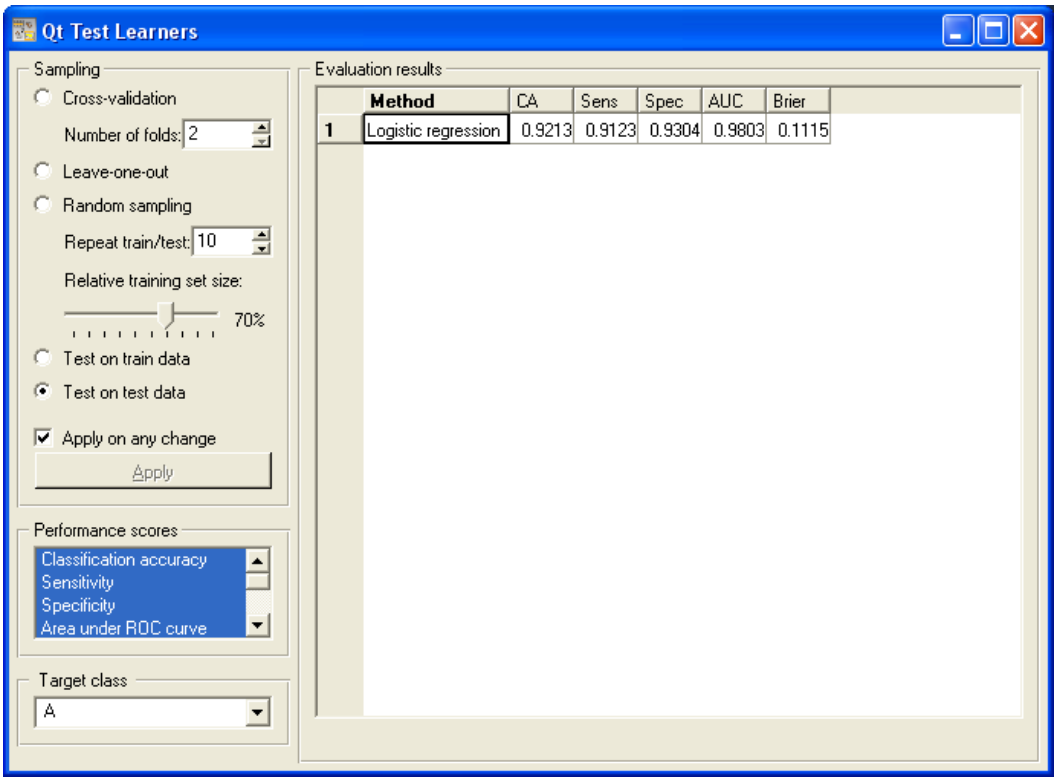


3.3.4 Logistic regression

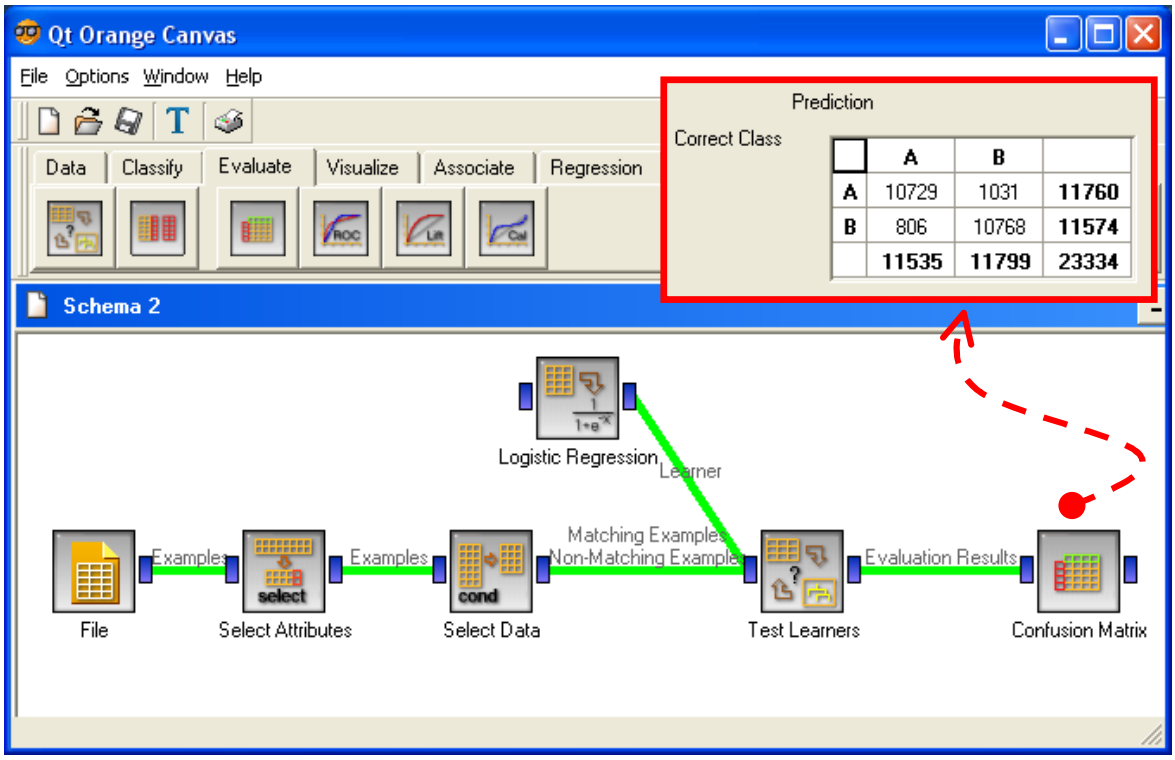
We can insert now the LOGISTIC REGRESSION component (CLASSIFY tab). We connect it to the TEST LEARNERS. The computation time is rather fast.



Into the TEST LEARNERS output window, we obtain the accuracy rate.

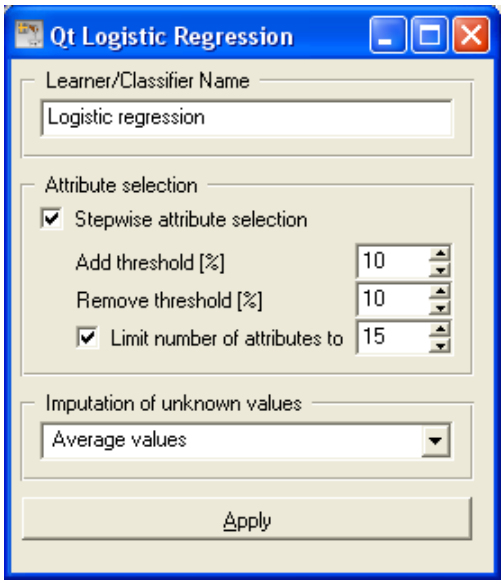


The accuracy rate is 92.13%, so the error rate is 7.87%. We can obtain the confusion matrix using the CONFUSION MATRIX component (EVALUATE tab).



3.3.5 Feature selection

We can perform a feature selection with the LOGISTIC REGRESSION component. We click on the OPEN menu. We can ask the STEPWISE regression. The thresholds are in percentage. Their significations are described into the documentation (press F1 into the dialog box). We limit the number of selected variables to 15. Then we click on the APPLY button.



Into the CONFUSION MATRIX component window, we obtain the confusion matrix. It is identical to those of R or Tanagra, with an error rate equal to (1011+777)/23334 = 7.66%.

Prediction

Correct Class

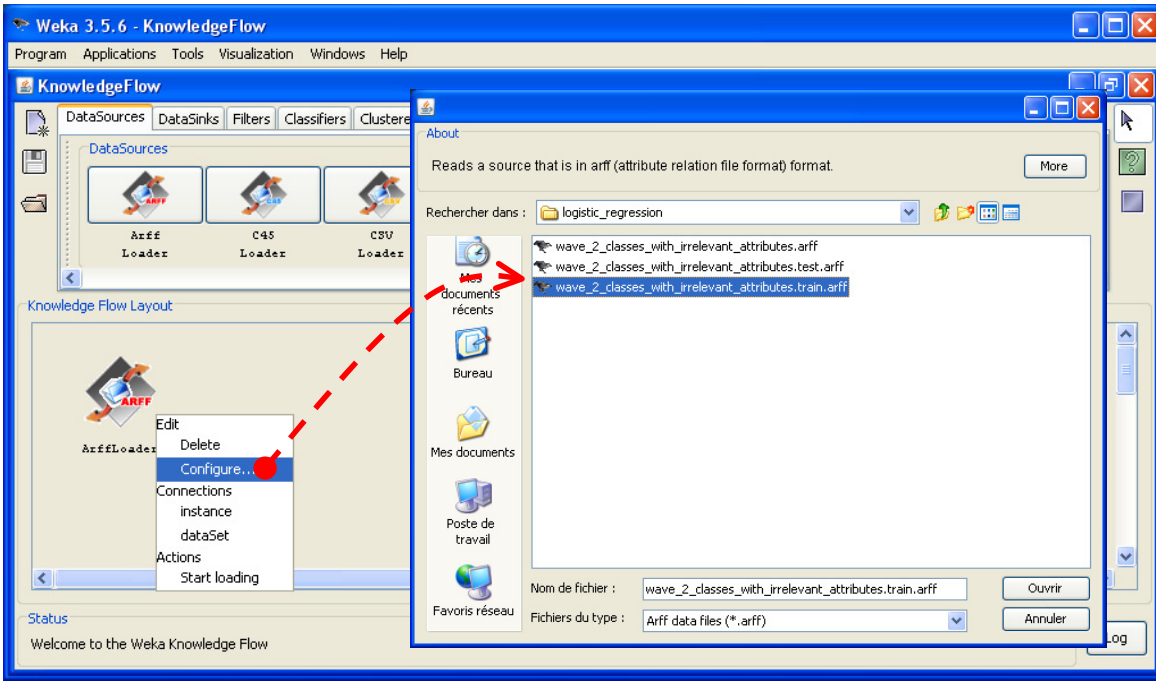
	A	B	
A	10749	1011	**11760**
B	777	10797	**11574**
	11526	**11808**	**23334**

3.4 WEKA

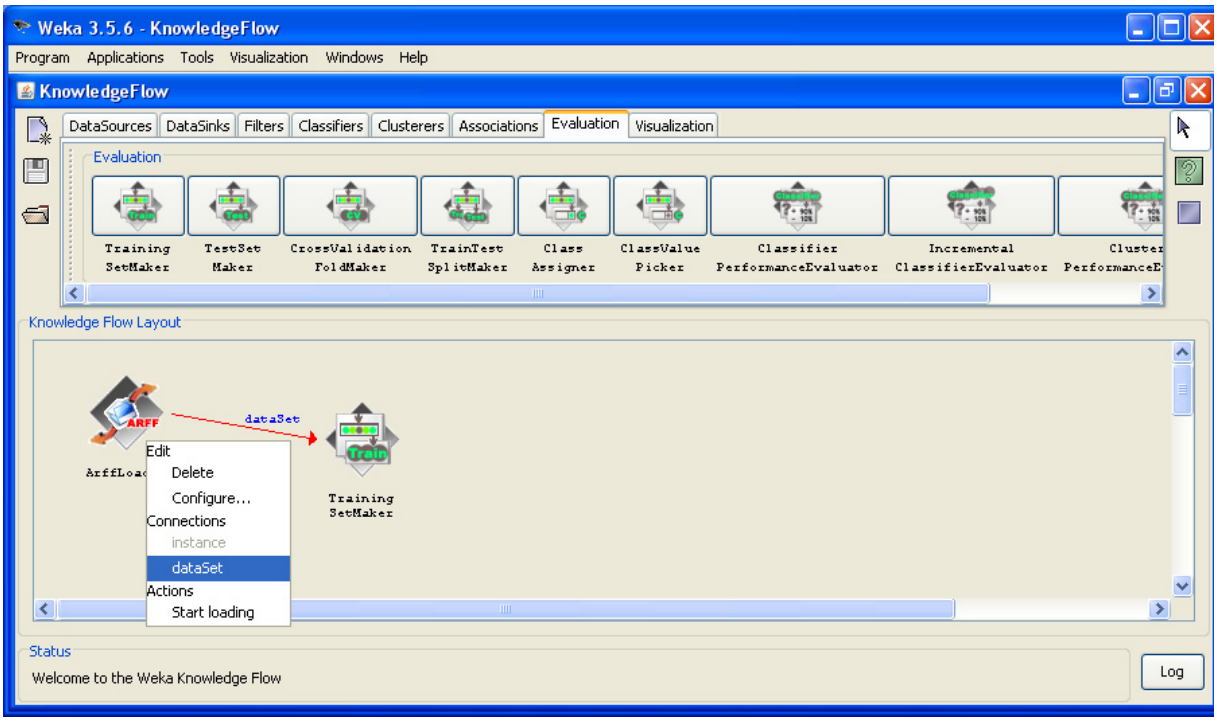
WEKA (http://www.cs.waikato.ac.nz/ml/weka/) is widely known. We use the KNOWLEDGE FLOW execution mode. I could not find what is the component which enables to split the dataset in a 2 part (train and test set). Thus, we had to manually subdivide the data file in a 2 parts before we launch Weka. It is not really convenient. But the utilization of the same train and test set for all the tools is really crucial in our tutorial. One of the goals of this experimentation is to compare precisely the results of the various tools.

3.4.1 Logistic regression

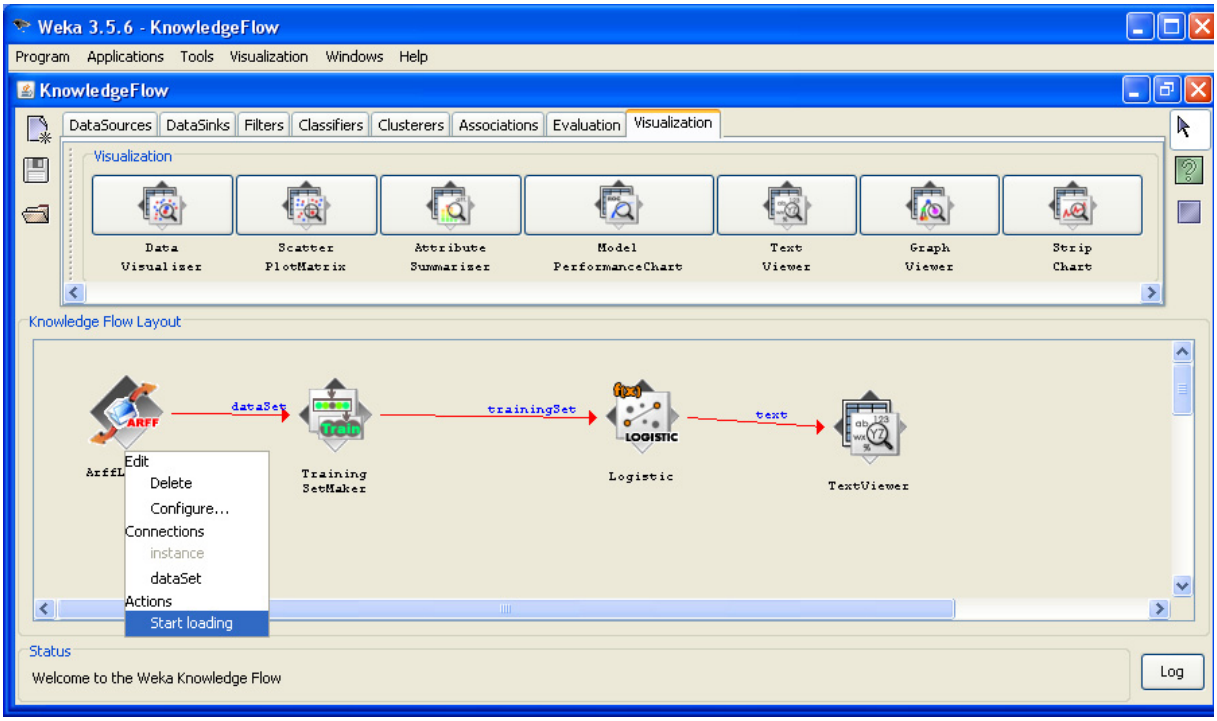
We insert the ARFFLOADER component (DATASOURCES tab) into the Knowledge Flow. We click on the CONFIGURE menu; we select "wave_2_classes_with_irrelevant_attributes.train.arff".



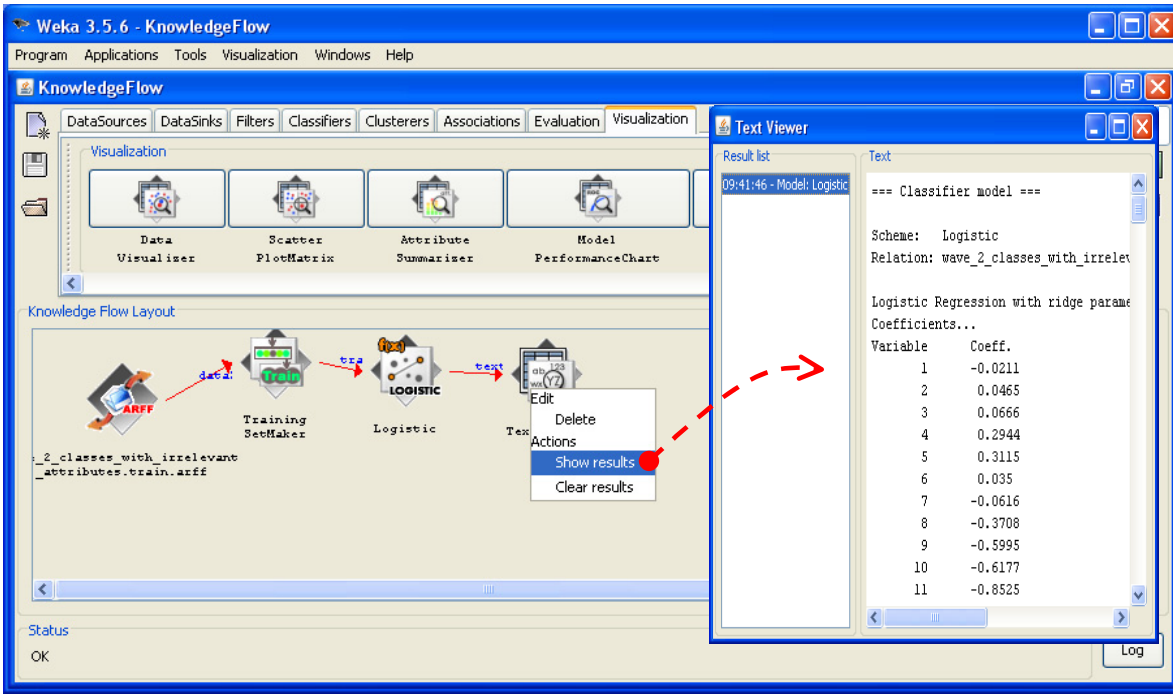
Weka defines automatically the last column as the class attribute. The others are the predictors. To specify that the whole file corresponds to the learning set, we insert the component TRAINING SET MAKER (EVALUATE tab). We define a DATASET connection between the previous component and this one.



Then we add the LOGISTIC component for the calculations, and the TEXT VIEWER component for the visualization of the results. We make the right connection between these components. We launch the calculation by clicking on the START LOADING menu of the ARFF LOADER component.

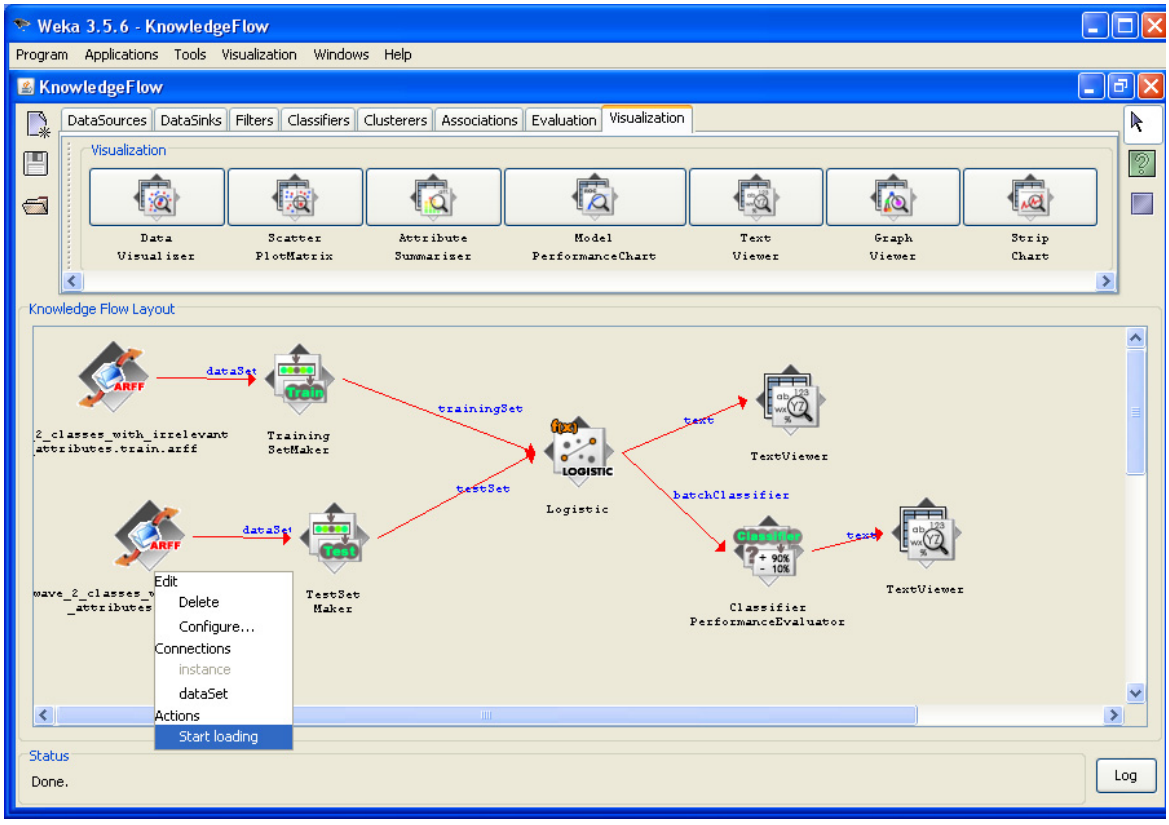


We click on the SHOW RESULTS menu of TEXT VIEWER in order to visualize the results.



WEKA gives the regression coefficients and the odds-ratio. We have no information about the significance of the whole regression and the significance of each variable.

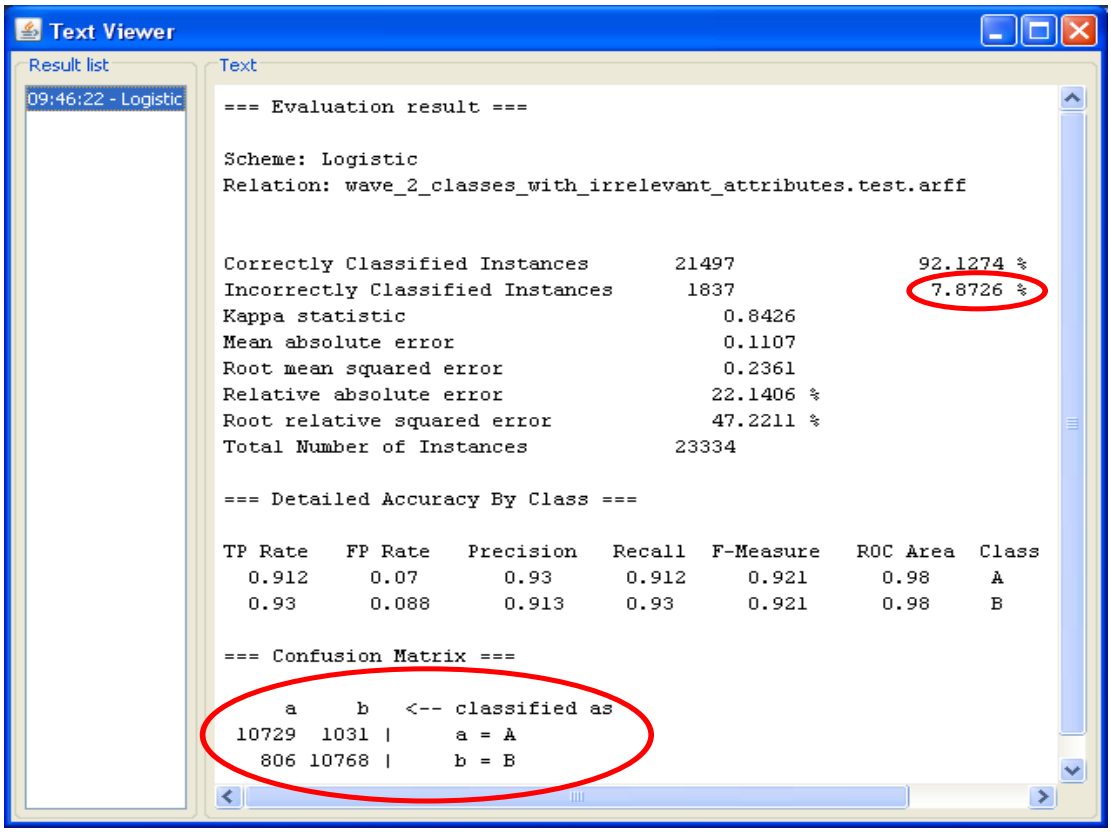
3.4.2 Evaluation on the test set



In order to evaluate the performances of the classifier on the test set, we must: load the test set "wave_2_classes_with_irrelevant_attributes.test.arff"with an ARFF LOADER (DATA SOURCES tab);

define this dataset as a test set using the TEST SET MAKER (EVALUATION tab); insert the CLASSIFIER PERFORMANCE EVALUATOR (EVALUATION tab), connected to the classifier to evaluate; a TEXT VIEWER enables so to visualize the results.

We click first on the START LOADING menu of the first component in order to launch the calculations. Then, we click on SHOW RESULTS menu of the TEXT VIEWER.



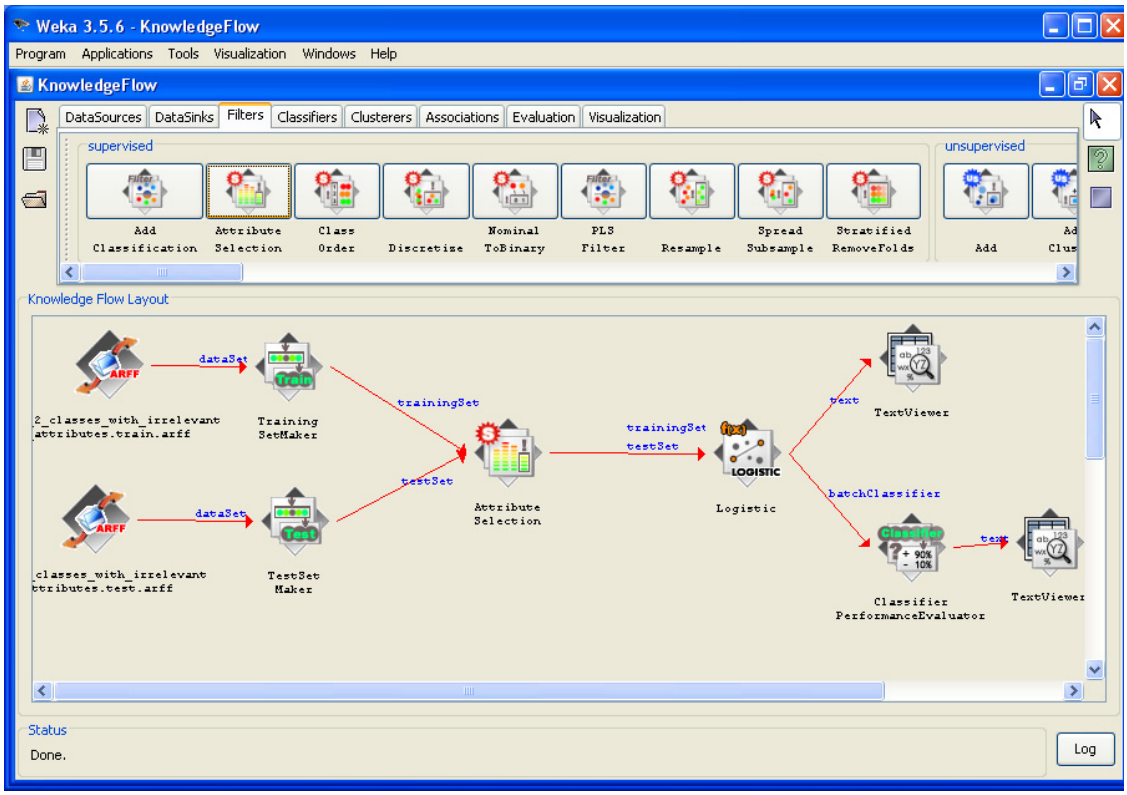
The confusion matrix is the same as the other tools.

3.4.3 (Trying the) Feature selection

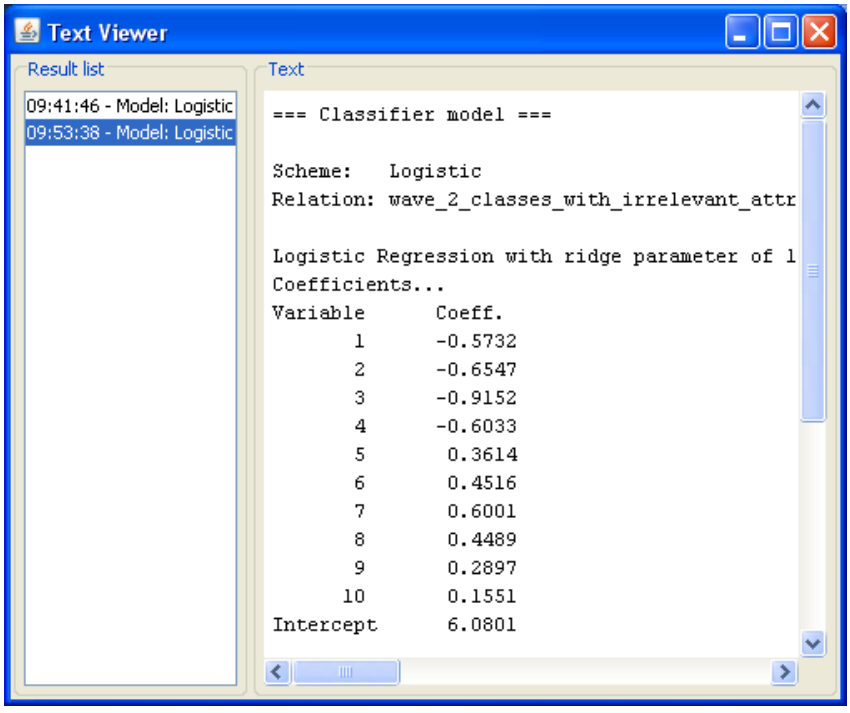
Weka does not propose a Feature Selection approach dedicated to the Logistic Regression. But we can use a generic filtering method. It enables to remove automatically the irrelevant variables. The drawback is that the approach is not especially in relation with the characteristics of the Logistic Regression.

We add the ATTRIBUTE SELECTION component (FILTERS tab) into the Knowledge Flow. We note the twofold connection between the ATTRIBUTE SELECTION and the LOGISTIC REGRESSION. According to the documentation, this component implements the CFS method [M. A. Hall (1998). Correlation-based Feature Subset Selection for Machine Learning. Hamilton, New Zealand.].

The new knowledge flow is the following

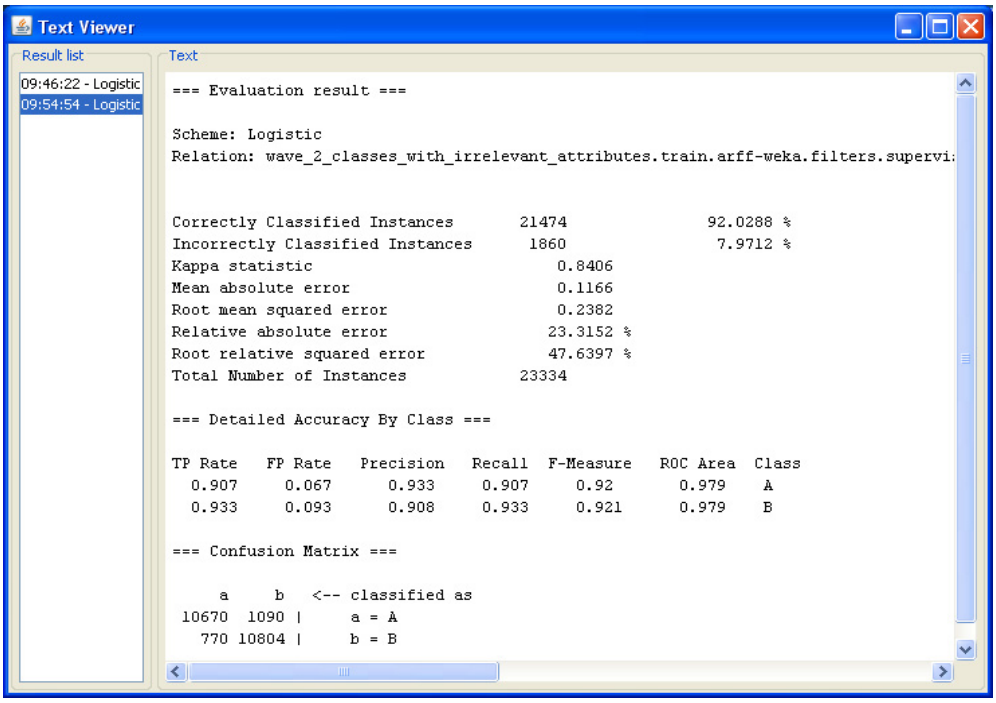


We click on the START LOADING menu of the data access component. The results are available into the TEXT VIEWER component.



It is seems that 10 descriptors are selected. But we don't know what these descriptors are!

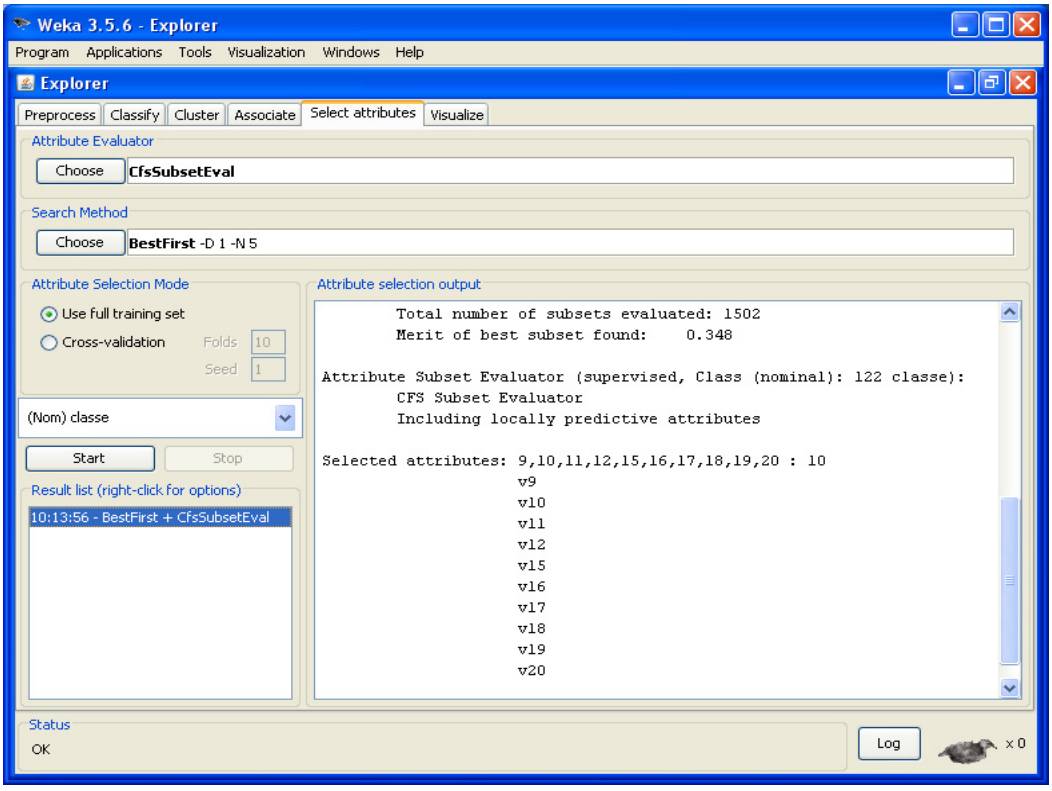
Regarding the error rate, we click on the START LOADING of the second data access component. The results are available into the TEXT VIEWER connected to the classifier performance evaluator.



There are (1090 + 770) = 1860 misclassified examples; the test error rate is 7.97%.

3.4.4 List of the selected variables

10 variables are selected among the candidate descriptors. But we don't know which these descriptors are. This is a problematic situation. Fortunately, we can find these descriptors using the EXPLORER module of Weka. We define the same analysis. We then obtain.



None of the ALEA variables are selected. It is a good result. But we obtain fewer variables compared with R or Tanagra. The consequence is that there are more misclassified examples on the test set (1860 vs. 1788). But is it really a problem? 72 supplementary misclassified examples among 23,334 seems not be a drawback. Especially, if in the same time, the model is described with 10 variables, and not 15. The answer depends on the priorities of the analysis.

3.5 R (using the RWeka package)

The LOGISTIC method of Weka can be used into the R environment using the RWeka package. The main advantage is that we can take advantage of the powerful data management functionalities of R. We can also directly compare the results of **Logistic(.)** and the **glm(.)** command e.g. on which examples the classification differs on the test set, etc.

In the screen shot below, we import the dataset. Then we create the learning part and the test part.

```
library(RWeka)

#charger les données
setwd("D:/DataMining/Databases_for_mining/comparison_TOW/logistic_regression")
donnees <- read.arff(file("wave_2_classes_with_irrelevant_attributes.arff","r"))

#petite vérification sur la classe et la variable de partition learning-test
summary(donnees[,c("classe","sample")])

#partitionner en apprentissage-test à l'aide de la colonne sample
donnees.app <- donnees[donnees$sample=="learning",1:122]
donnees.test <- donnees[donnees$sample=="test",1:122]
```

We can launch the **Logistic(.)** command

```
#régression logistique de Weka
modele <- Logistic(classe ~ ., data = donnees.app)
summary(modele)
```

We obtain the following results

```
> #régression logistique de Weka
> modele <- Logistic(classe ~ ., data = donnees.app)
> summary(modele)

=== Summary ===

Correctly Classified Instances        9211               92.11    %
Incorrectly Classified Instances       789                7.89    %
Kappa statistic                          0.8422
Mean absolute error                      0.1105
Root mean squared error                  0.2363
Relative absolute error                 22.1077 %
Root relative squared error             47.2571 %
Total Number of Instances            10000

=== Confusion Matrix ===

    a    b   <-- classified as
 4576  430 |    a = A
  359 4635 |    b = B
```

The resubstitution error rate is the same of Weka (7.89%). We can visualize the model with the **print(.)** command.

The **predict(.)** command enables to apply the classifier on the test set.

```
#prediction sur la partie test
pred <- predict(modele, newdata = donnees.test, type = "class")

#matrice de confusion
mc <- table(donnees.test$classe,pred)
print(mc)

#taux d'erreur en test
erreur <- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
print(erreur)
```

We can compute the confusion matrix and the test error rate: 7.87%, i.e. (1031 + 806) 1837 misclassified examples.

```
> #prediction sur la partie test
> pred <- predict(modele, newdata = donnees.test, type = "class")
>
> #matrice de confusion
> mc <- table(donnees.test$classe,pred)
> print(mc)
   pred
        A     B
  A 10729  1031
  B   806 10768
>
> #taux d'erreur en test
> erreur <- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
> print(erreur)
[1] 0.07872632
```

The filtering methods for variable selection seem not included in the package. We could not perform the variable selection (http://cran.r-project.org/web/packages/RWeka/index.html).

4 Conclusion

The logistic regression is a very popular supervised learning approach. It combines a good performance in prediction and a large possibility of the interpretation of the results.

This method is available in various free tools. It is interesting to compare them, especially because they do not use the same optimization algorithm and the same variable selection strategy.

In this tutorial, we note however that these tools give very similar results on our dataset.