

1 Objectif

Comparer les performances des différentes implémentations libres (gratuites) de la régression logistique.

La régression logistique (http://fr.wikipedia.org/wiki/Régression_logistique) est une technique prédictive, très populaire dans la communauté statistique. Je ne sais pas si elle est très utilisée parce que très enseignée, ou très enseignée parce que largement utilisée. En tous les cas, on ne peut pas passer à côté si on s'intéresse un tant soit peu au Scoring c.-à-d. aux configurations où l'on souhaite prédire ou expliquer les valeurs d'une variable discrète (nominale ou ordinale) à partir d'une série de descripteurs (de type quelconque).

Les raisons de cet engouement sont nombreuses. La régression logistique s'intègre dans un cadre théorique parfaitement identifié, celui de la régression linéaire généralisée. On arrive aisément à la positionner parmi l'arsenal des techniques explicatives, à partir des caractéristiques de la variable à prédire et des variables prédictives. Ce n'est pas anodin. Nous pouvons ainsi garder des éléments de compréhension génériques (les tests basés sur le rapport de vraisemblance entre autres), tout en appliquant la méthode concrètement sur un problème spécifique.

C'est une technique semi paramétrique. Son champ d'application est large. Contrairement aux approches paramétriques, le cadre théorique reste valable même si les variables prédictives intègrent des variables numériques ou discrètes (recodées 0/1). Par rapport aux techniques issues de l'apprentissage automatique, elle intègre les outils de la statistique inférentielle. Il est possible d'évaluer statistiquement la significativité d'une variable, d'un groupe de variables, du modèle complet, en utilisant des critères en accord avec ceux qui ont guidé l'estimation des paramètres de l'équation de régression. La sélection automatique de variables peut faire partie intégrante du processus de modélisation.

Enfin, autre atout fort de la régression logistique, la lecture des coefficients sous forme de surcroît de risque (les fameux « odds ratio ») donne aux utilisateurs un outil de choix pour comprendre l'essence de la relation entre les descripteurs et la variable à prédire. En combinant les variables, nous pouvons évaluer finement l'influence des facteurs et des interactions entre les facteurs.

La régression logistique est implémentée dans tous les logiciels de statistique commerciaux. Elle est plus rare en revanche dans les logiciels libres. En partie parce que la méthode est peu connue des informaticiens, ceux qui sont les plus enclins à programmer des outils. Pour ma part, je n'ai quasiment jamais vu un article traitant ou utilisant la régression logistique dans les revues et conférences estampillées « machine learning » et apparentés (informatique). C'est assez extraordinaire quand on connaît sa popularité auprès des statisticiens. La situation change quand même un peu maintenant. Avec le label « data mining », il y a un certain brassage des cultures. On peut parler de « faire une régression » sans que certaines personnes ne s'imaginent que vous avez des problèmes psychiques.

Dans ce didacticiel, nous comparons la mise en œuvre de la régression logistique à l'aide de quelques logiciels libres : **Tanagra 1.4.27**, bien sûr, puisque je travaille dessus ; **R 2.7.2** (procédure GLM), qui est incontournable dès que l'on souhaite utiliser des techniques d'obédience statistique ; **Orange 1.0b2**, qui l'intègre dans sa panoplie ; **Weka 3.5.6**, qui l'aborde exclusivement sous l'angle de l'optimisation, en faisant l'impasse sur la partie inférentielle ; et enfin, toujours Weka mais via **le package RWeka 0.3-13 pour** le logiciel **R**.

Au delà de la comparaison, **ce didacticiel est aussi l'occasion de montrer la démarche à suivre pour réaliser la succession d'opérations suivantes sur ces différents logiciels** : importer un fichier au format ARFF ; fractionner les données en apprentissage et test ; lancer la

modélisation sur la fraction apprentissage ; évaluer les performances sur la partie test ; procéder à une sélection de variables en accord avec la régression logistique (et non pas basé sur des critères qui n'ont rien aucun rapport avec l'approche) ; évaluer de nouveau les performances du modèle simplifié.

Concernant la documentation, après une très longue période de disette, les références francophones de qualité commencent à abonder. Une description plus ou moins approfondie de la régression logistique est aujourd'hui incontournable dans tout ouvrage désireux d'aborder les problèmes de classement et de scoring (cf. une petite sélection de références dans la dernière section de ce didacticiel).

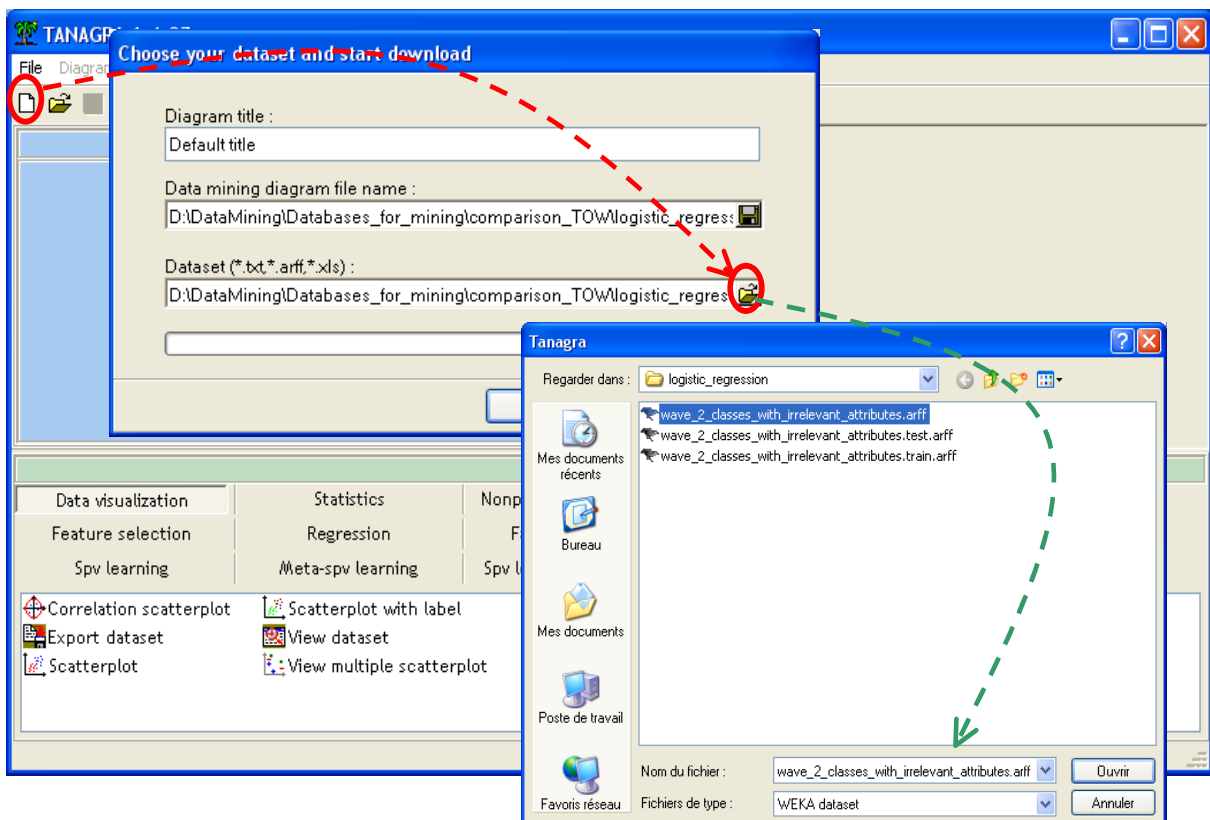
2 Données

Nous utilisons le fichier WAVE de Breiman et al. (1984) en la restreignant à 2 classes (A et B) (cf. http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/wave_2_classes_with_irrelevant_attributes.zip). Aux 21 descripteurs originaux (v1 à v21) ont été ajoutés 100 variables générées aléatoirement (alea1 à alea100). La sélection de variables sera cruciale dans ce contexte. Le fichier comporte 33.334 observations. La variable « SAMPLE » permet d'identifier les 10.000 individus réservés pour l'apprentissage, le reste étant dédié à l'évaluation des modèles. Comme nous utilisons les mêmes individus en test, les résultats seront comparables d'un logiciel à l'autre.

3 Régression logistique et sélection de variables

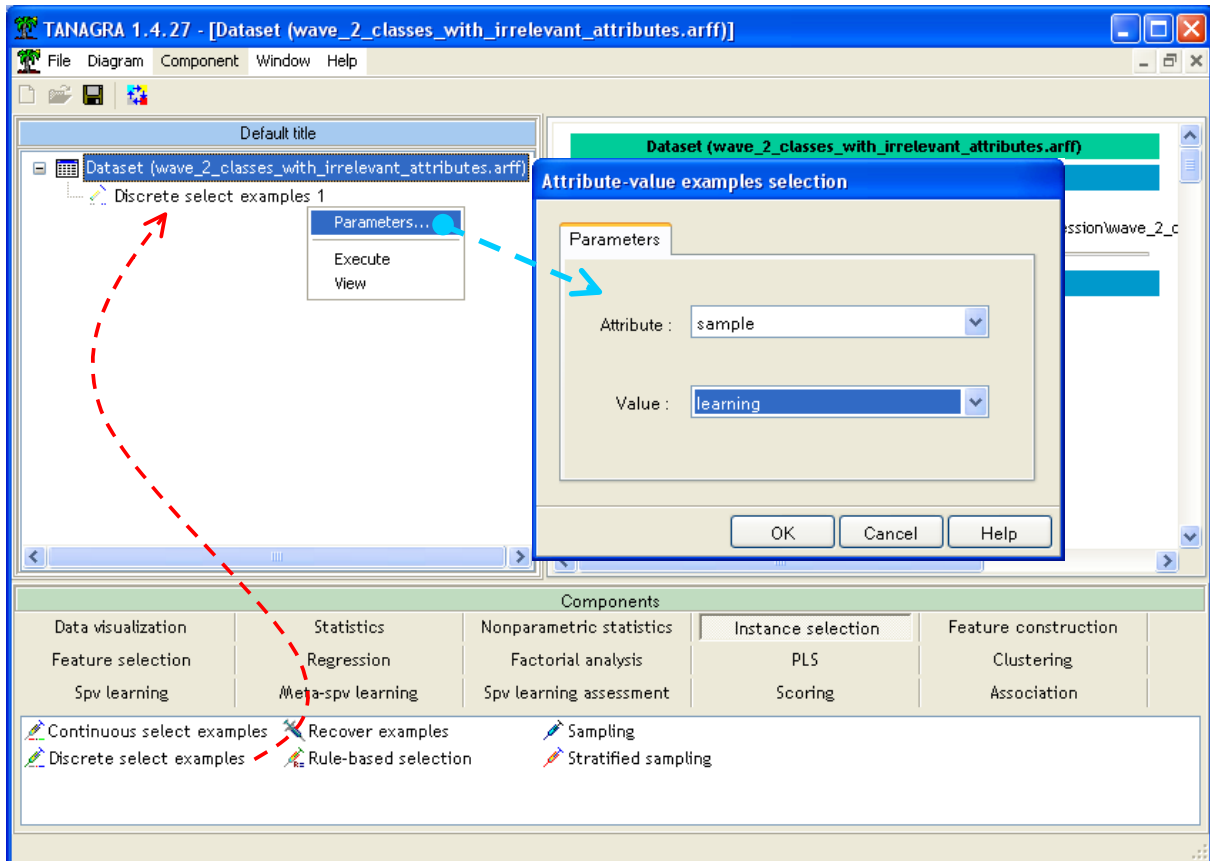
3.1 TANAGRA

Après lancé TANAGRA, nous activons le menu FILE / NEW pour créer un nouveau diagramme et importer les données. Dans la boîte de paramétrage, nous sélectionnons le fichier « wave_2_classes_with_irrelevant_attributes.arff ».



3.1.1 Partition des données en « apprentissage » - « test »

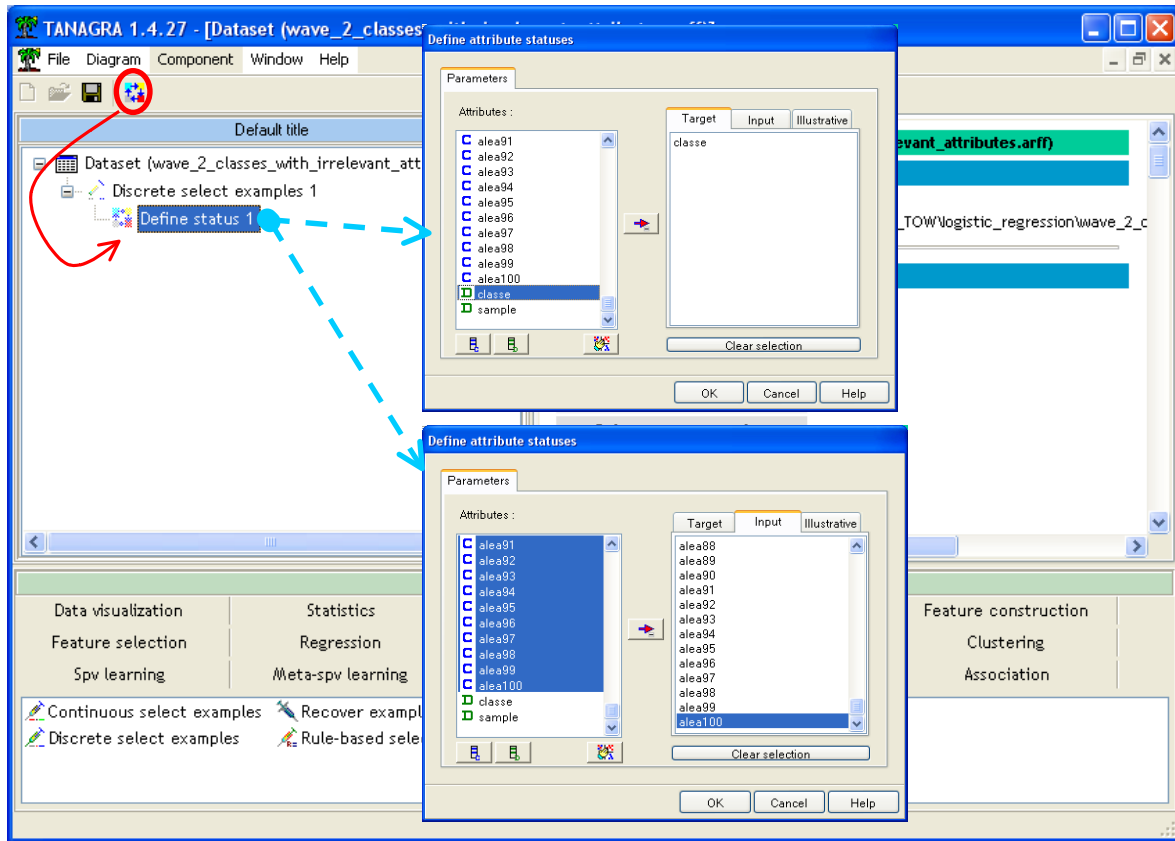
Nous devons isoler les observations dédiées à l'apprentissage. Nous insérons le composant DISCRETE SELECT EXAMPLES (onglet INSTANCE SELECTION) à cet effet. Nous cliquons sur le menu contextuel PARAMETERS, nous sélectionnons l'attribut SAMPLE avec la valeur LEARNING.



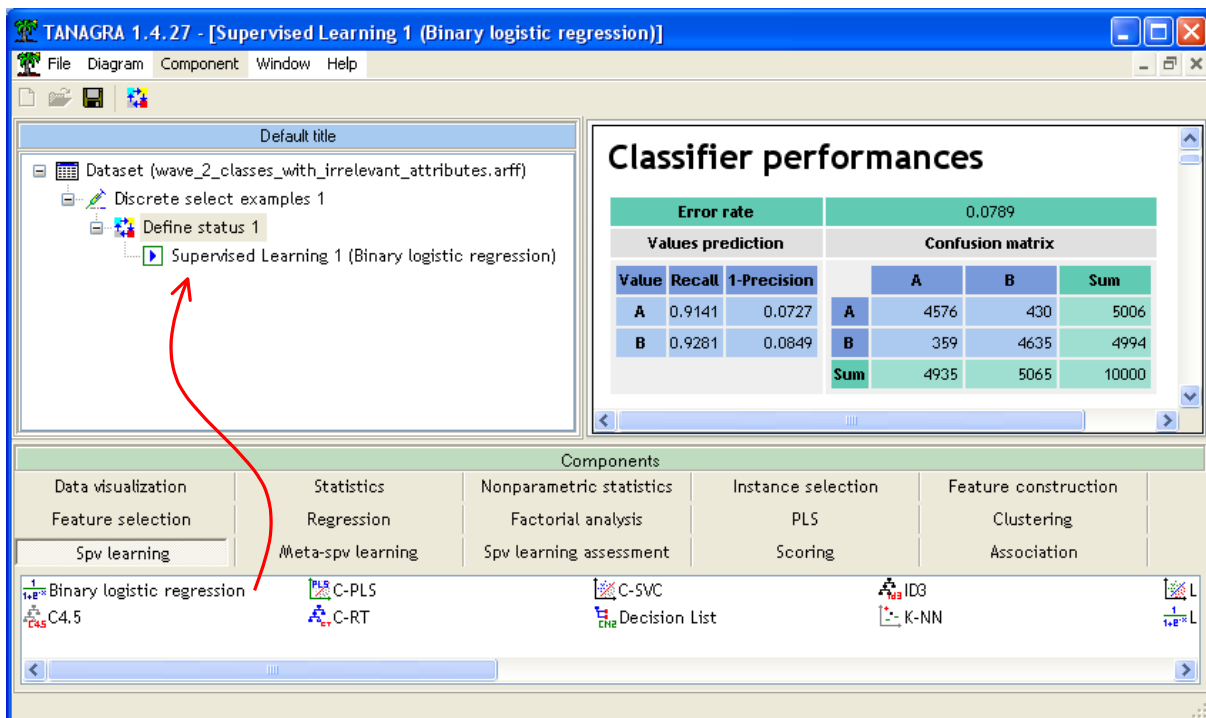
Après validation, le menu VIEW affiche le filtrage opéré, 10000 individus sont maintenant sélectionnés pour l'apprentissage.

3.1.2 Régression logistique binaire

Nous devons tout d'abord spécifier le rôle des variables. Pour ce faire, nous introduisons le composant DEFINE STATUS via le raccourci dans la barre d'outils. Nous plaçons CLASSE en TARGET, les variables v1 à alea100 en INPUT. Bien évidemment, la variable SAMPLE ne doit pas être utilisée.



Nous pouvons maintenant insérer le composant BINARY LOGISTIC REGRESSION (onglet SPV LEARNING). Nous actionnons le menu VIEW pour accéder aux résultats.



Au bout de 10 secondes, les résultats s'affichent. Le taux d'erreur en apprentissage est 7.89%. La déviance du modèle est $-2LL = 3586.688$. Ce chiffre est important, il permet de comparer la qualité de la minimisation d'un logiciel à l'autre. Le critère BIC (Schwartz) est $SC = 4710.350$, il nous sera utile pour comparer les modèles comportant un nombre de variables différent.

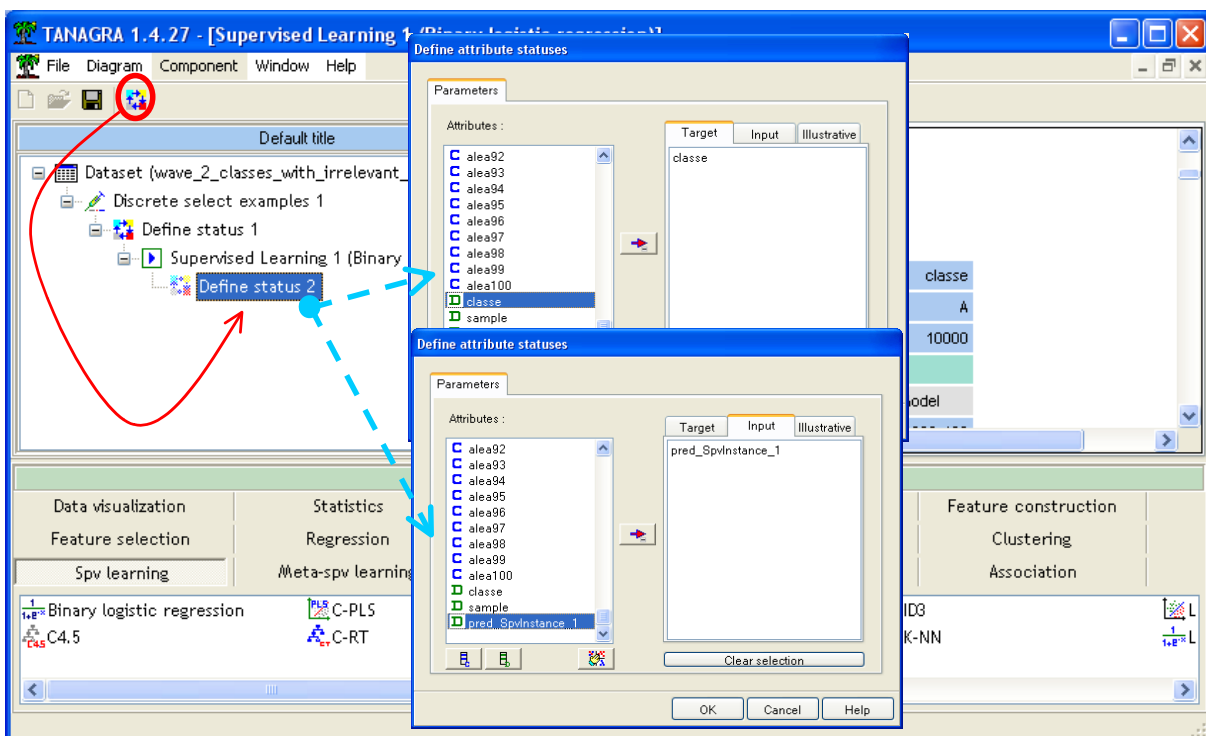
Adjustement quality

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 Chi ² test (LR)		
Chi-2	10276.2407	
d.f.	121	
P(>Chi-2)	0.0000	
R ² -like		
McFadden's R ²	0.7413	
Cox and Snell's R ²	0.6421	
Nagelkerke's R ²	0.8562	

Dans la partie basse de la fenêtre, nous disposons des coefficients et des indications sur leur significativité. Nous constatons (avec surprise) que certaines variables « alea » semblent significatives au risque 1%. Nous verrons plus loin si le processus de sélection de variables permettra de les évacuer.

3.1.3 Evaluation sur l'échantillon test

L'étape suivante consiste à évaluer les performances du classifieur. Nous insérons de nouveau le composant DEFINE STATUS. Nous plaçons en TARGET la variable à prédire CLASSE, en INPUT la variable prédite par le modèle PRED_SPV_INSTANCE_1.



Il ne nous reste plus qu'à insérer le composant TEST (onglet SPV LEARNING ASSESSMENT) pour obtenir la matrice de confusion. Il effectue automatiquement les calculs sur les individus non sélectionnés c.-à-d. les 23334 observations mises de côté initialement.

The screenshot shows the TANAGRA 1.4.27 interface. On the left, a diagram shows a workflow: Dataset (wave_2_classes_with_irrelevant_attributes.arff) -> Discrete select examples 1 -> Define status 1 -> Supervised Learning 1 (Binary logistic regression) -> Define status 2 -> Test 1. A red arrow points from the 'Test 1' component in the diagram to the 'Results' panel on the right. The 'Results' panel displays the following data:

Results						
pred_SpvInstance_1						
Error rate		0.0787				
Values prediction		Confusion matrix				
Value	Recall	1-Precision		A	B	Sum
A	0.9123	0.0699	A	10729	1031	11760
B	0.9304	0.0874	B	806	10768	11574
			Sum	11535	11799	23334

At the bottom, the 'Components' panel shows various options, with 'Spv learning assessment' selected. Below it, a list of components includes 'Test' and 'Train-test'.

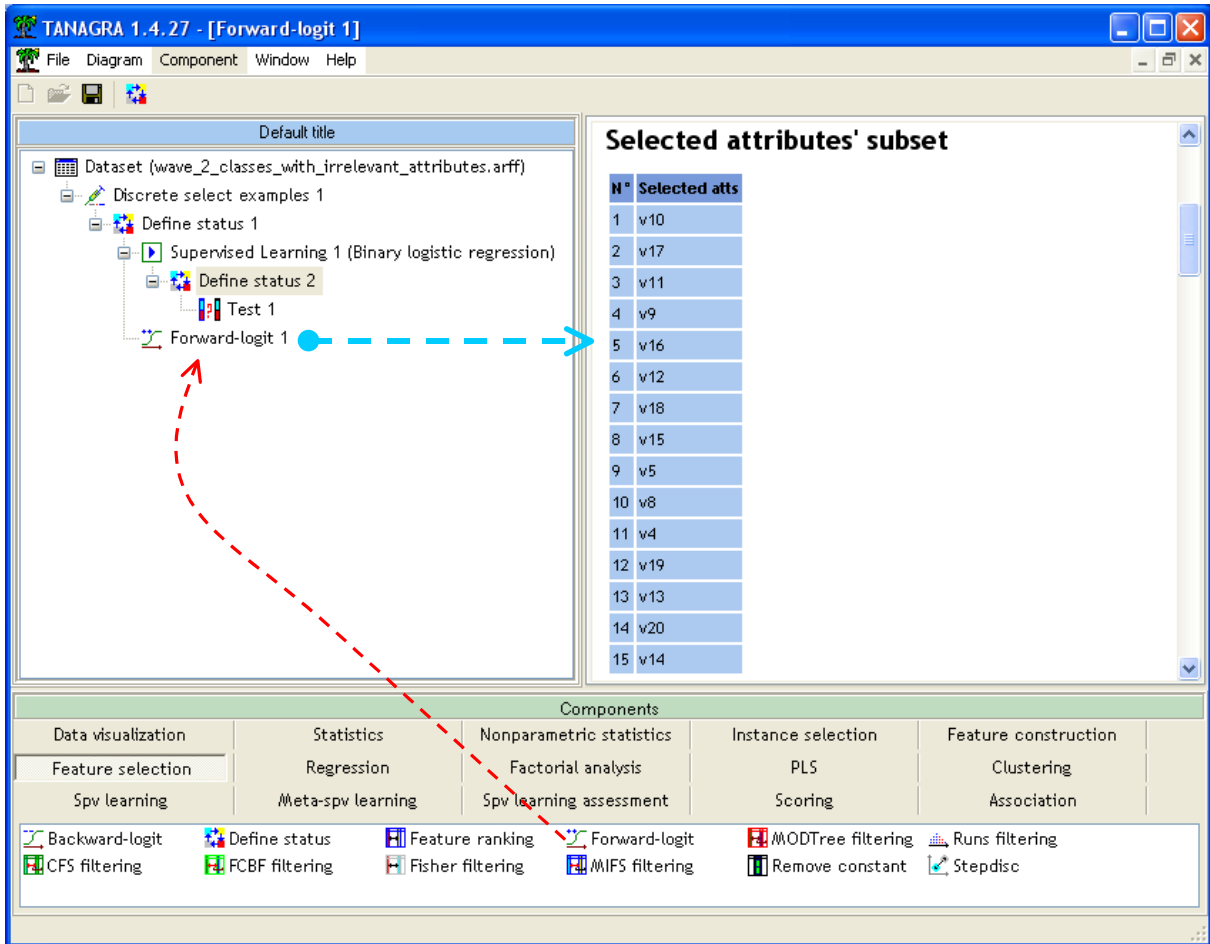
Le taux d'erreur est 7.87%, soit $(1031 + 806) = 1837$ individus mal classés sur 23334.

3.1.4 Sélection forward + évaluation

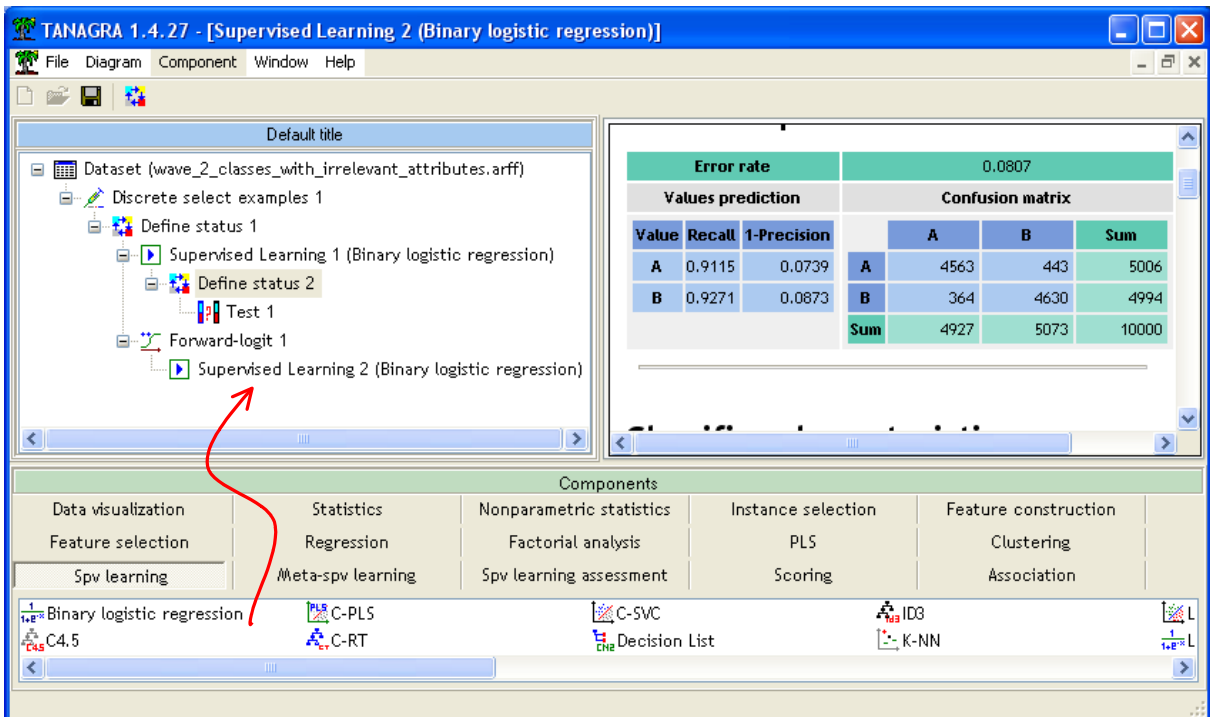
Bien classer est important. Bien classer en mettant en avant les variables les plus importantes c'est mieux. Surtout si l'on souhaite interpréter les résultats par la suite. Nous souhaitons donc procéder à une sélection de variables. Nous choisissons une sélection par avant (ascendante) basée sur le score (voir Tenenhaus, 2007 ; pages 447 à 453). Le principal intérêt de cette approche est le nombre limité de régressions à effectuer. Si p est le nombre de variables, dans le pire des cas c.-à-d. toutes les variables auront été sélectionné, nous aurons calculé p régressions.

Nous insérons le composant FORWARD LOGIT (onglet FEATURE SELECTION) dans le diagramme. Nous actionnons directement le menu VIEW pour accéder aux résultats. TANAGRA réalise une recherche ascendante, l'adjonction d'une variable se poursuit tant qu'elle est significative à 1% (paramètre modifiable). Nous avons également la possibilité de borner le nombre de variables à sélectionner.

Nous actionnons le menu contextuel VIEW. Au bout de 60 secondes, TANAGRA nous annonce la sélection de 15 variables. Aucune variable ALEA n'a été introduite. Dans la partie basse de la fenêtre, nous disposons du détail du processus de sélection. TANAGRA affiche les 5 meilleures variables à chaque étape.



Pour évaluer la qualité de la prédiction à l'aide de ces 15 descripteurs, nous définissons de nouveau la séquence apprentissage évaluation. Nous insérons tout d'abord la régression logistique (BINARY LOGISTIC REGRESSION, onglet SPV LEARNING)



Le taux d'erreur en apprentissage est de 8.07%. La déviance est $-2LL = 3678.061$, et surtout le critère BIC est $SC = 3825.427$. Nettement meilleur (plus petit) que celui du modèle intégrant toutes les variables.

Voyons ce qu'il en est pour la prédiction sur la partie test. Nous insérons un DEFINE STATUS en plaçant CLASSE en TARGET et la seconde prédiction, PRED_SPVINSTANCE_2, en INPUT. Puis le composant TEST (onglet SPV LEARNING ASSESSMENT).

The screenshot shows the TANAGRA 1.4.27 interface. On the left, a workflow diagram shows a sequence of components: Dataset, Discrete select examples, Define status 1, Supervised Learning 1 (Binary logistic regression), Define status 2, Test 1, Forward-logit 1, Supervised Learning 2 (Binary logistic regression), Define status 3, and Test 2. The 'Define status 3' and 'Test 2' components are circled in red. On the right, the 'Test 2' results panel is displayed. It shows the evaluation set as 'unselected examples'. The results for 'pred_SpvInstance_2' include an error rate of 0.0766. A confusion matrix is also shown, with a red dashed circle and an orange exclamation mark highlighting it.

Values prediction		Confusion matrix			
Value	Recall	1-Precision	A	B	Sum
A	0.9140	0.0674	10749	1011	11760
B	0.9329	0.0856	777	10797	11574
Sum			11526	11808	23334

Le taux d'erreur en test, calculé sur les 23334 observations mises de côté, est de 7.66% ($1011 + 777 = 1788$ mal classés). Nous avons un modèle plus simple, avec 15 variables contre 121 variables précédemment, et très légèrement plus performant, 7.66% d'erreur contre 7.87% (en tous les cas, la qualité de la prédiction n'a pas été dégradée).

3.2 R (avec la procédure GLM)

Le logiciel R n'est plus à présenter (<http://www.r-project.org/>). Très connu des statisticiens, il prend son essor dans la communauté du Data Mining. Il sait importer les fichiers ARFF via le package RWeka que nous étudierons plus en détail plus loin.

Nous introduisons les commandes suivantes pour charger les données et effectuer quelques vérifications.

```
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 nous fournit les résultats suivants.


```

> 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 Partition des données en « apprentissage » - « test »

Nous utilisons la colonne SAMPLE pour partitionner les données en 2 blocs distincts.

```

> #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 Régression logistique

Nous pouvons maintenant lancer la régression logistique sur les données « apprentissage ». Nous utilisons la procédure **glm(.)** (Régression linéaire généralisée).

```

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

```

Afficher la totalité des résultats n'a pas trop d'intérêt pour nous. On se bornera à visualiser la partie basse des sorties où nous pouvons noter que la déviance du modèle est $-2LL = 3586.7$, identique à celle de TANAGRA.

```

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

```

L'informaticien notera avec émerveillement la rapidité des calculs. R a mis moins de 2 secondes pour produire le modèle¹.

3.2.3 Evaluation sur la partie test

Pour évaluer les performances sur l'échantillon test, nous utilisons la fonction **predict(.)**. Elle nous fournit la probabilité d'affectation à la seconde classe PROBA. Il faut utiliser le seuil 0.5 pour décider si la prédiction doit être A ou B. Nous obtenons le vecteur PRED.

¹ J'utilise la commande **system.time(.)** pour mesurer le temps de traitement dans R

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

En croisant CLASSE et PRED, sur les données « test » toujours, nous obtenons la matrice de confusion. Nous en déduisons le taux d'erreur.

```
#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)
```

Voici les sorties de R correspondantes :

```
> #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
```

La matrice de confusion et le taux d'erreur de 7.87% sont exactement identiques à celle proposée par Tanagra : il y a $(1031 + 806) = 1837$ individus mal classés sur les 23334 de l'échantillon test.

3.2.4 Sélection de variables et nouvelle évaluation

Pour effectuer la sélection de variables, nous utilisons la procédure **stepAIC(.)** du package MASS. Elle cherche à trouver la combinaison de variables qui minimise un critère tenant compte à la fois de la capacité du modèle à intégrer les informations des données d'apprentissage et de sa complexité. Les indicateurs privilégiés sont le critère Akaike (AIC) ou BIC (Schwartz). Nous préférons ce dernier dans notre étude, les effectifs sont élevés et surtout il y a pléthore de variables candidates, dont une grosse majorité non pertinente.

Pour réaliser les calculs sous R, après avoir chargé le package MASS, nous calculons tout d'abord le modèle trivial, composé uniquement de la constante, puis nous lançons **stepAIC(.)** avec une recherche ascendante (forward). Le paramètre k permet de choisir le critère BIC [$k = \log(n)$] plutôt que AIC [$k = 2$].

```
#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)))
```

Cette approche est plus gourmande en ressources. Le calcul est assez long. C'est normal vu le nombre de régressions, donc d'optimisation, que doit réaliser R. Si p est le nombre de variables, R aura effectué $[p \times (p - 1) / 2]$ régressions dans le pire des cas c.-à-d. toutes les variables sont sélectionnées.

Voici le modèle final proposé par R :

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

Il a choisi un modèle à 15 variables, les mêmes que celles sélectionnées par Tanagra avec une approche basée sur les scores. Coïncidence amusante, les variables ont été de surcroît sélectionnées dans le même ordre.

Le modèle proposé par Tanagra et R étant le même, le critère BIC prend la même valeur, soit $SC = 3825.43^2$.

Ce modèle appliqué sur l'échantillon test propose également un taux d'erreur de 7.66%.

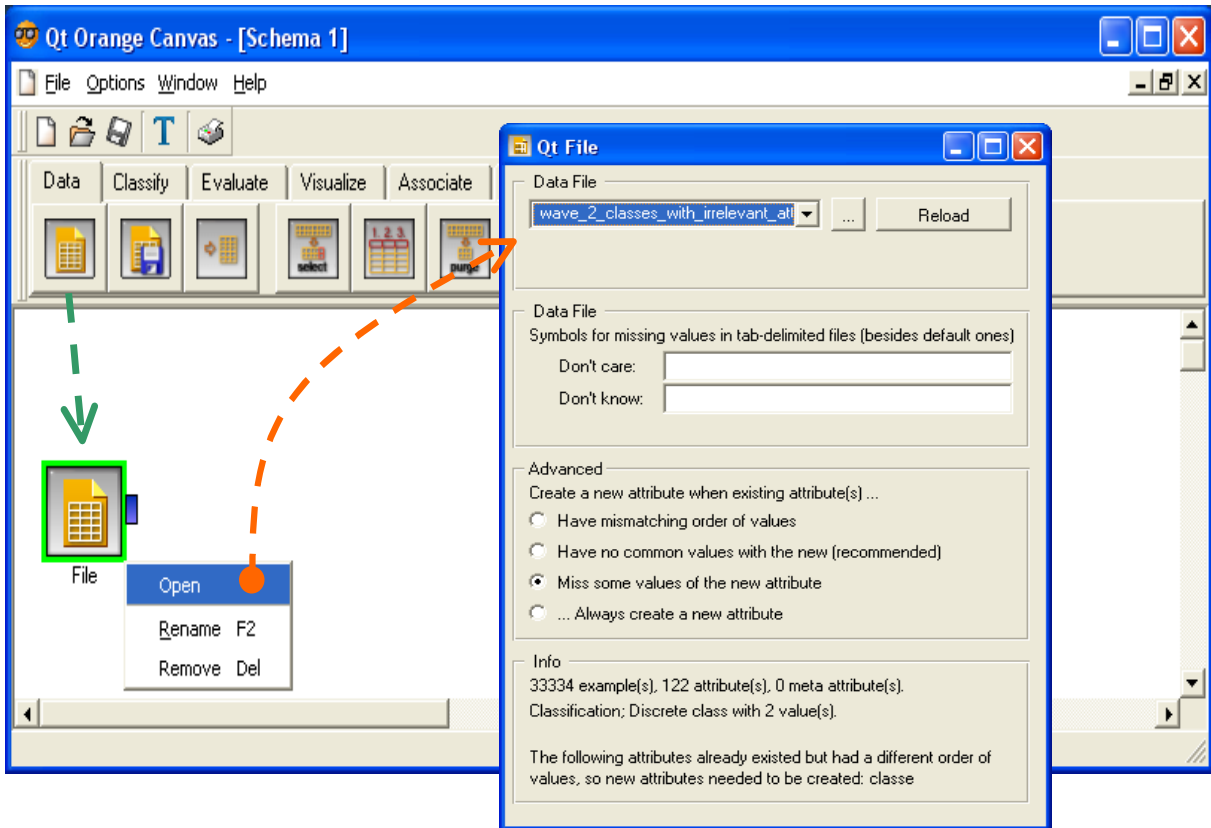
```
> #idem, prédiction sur fichier test + matrice de confusion + taux d'erreur
> proba <- predict(modele.forward, newdata = donnees.test,type="response")
> pred <- ifelse(proba < 0.5, 1, 2)
> pred <- factor(pred)
>
> mc <- table(donnees.test$classe,pred)
> print(mc)
  pred
  1    2
A 10749 1011
B  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

3.3 ORANGE

Orange est logiciel enthousiasmant (<http://www.ailab.si/orange/>), avec une interface agréable et des fonctionnalités très intéressantes pédagogiquement. La régression logistique est présente.

Nous utilisons le mode « schéma ». Pour charger les données, nous insérons le composant FILE (onglet DATA). Nous actionnons le menu OPEN pour le paramétrer. Nous choisissons le fichier wave_2_classes_with_irrelevant_attributes.arff.

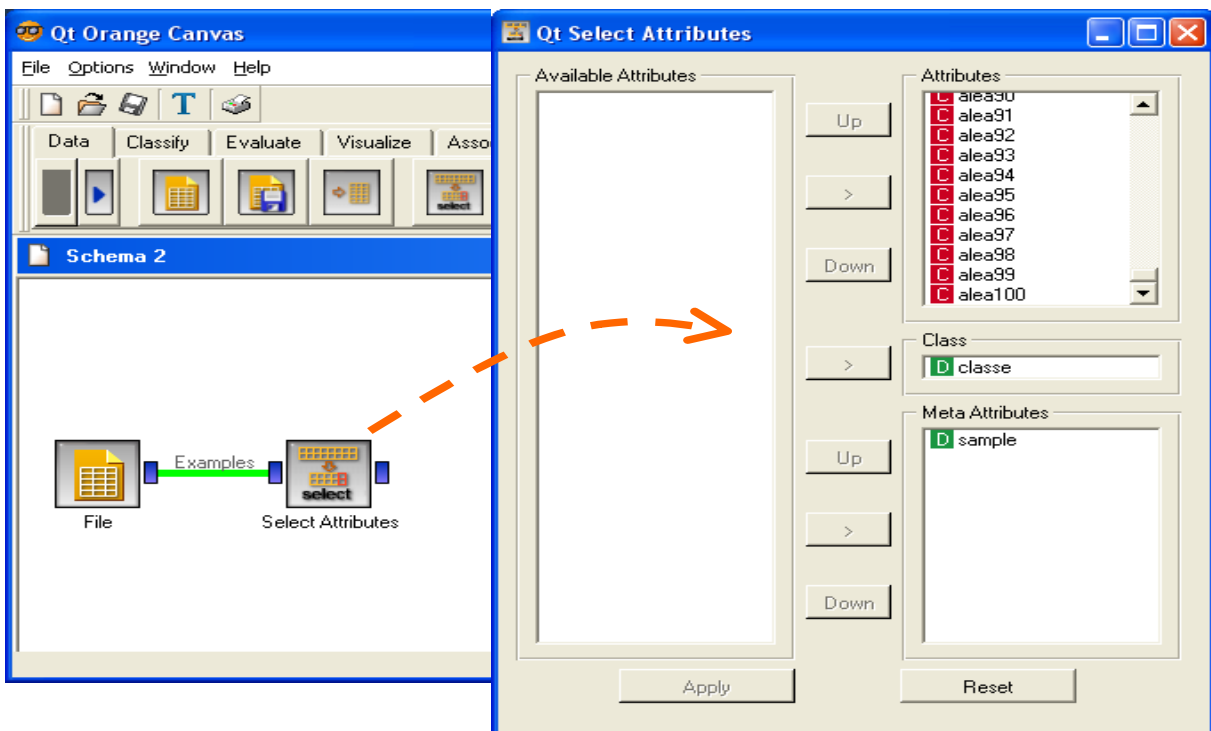
² R affiche AIC, mais comme nous avons modifié la valeur du paramètre k par $k = \log(n)$, où n est le nombre d'observations dans l'échantillon, il s'agit bien du critère BIC de Schwartz ici.



Orange indique que 122 descripteurs, 1 variable classe et 33334 observations ont été chargés.

3.3.1 Définition du rôle des variables

L'étape suivante consiste à préciser le rôle de chaque variable. Nous utilisons le composant SELECT ATTRIBUTES (onglet DATA). Après l'avoir connecté à l'icône source de données, nous la paramétrons de la manière suivante (via le menu OPEN toujours).



Attention, la variable SAMPLE doit être précisée en « Meta Attributes », sinon elle ne sera pas utilisable dans la suite du schéma.

3.3.2 Partition des données en « apprentissage » - « test »

Nous devons scinder maintenant l'échantillon en 2 parties à l'aide de la colonne SAMPLE. Nous utilisons le composant SELECT DATA (onglet DATA). Nous réalisons la connexion avec l'icône précédente, puis nous le paramétrons en cliquant sur le menu contextuel OPEN.

The screenshot shows the Orange Canvas interface with a workflow consisting of three components: 'File', 'Select Attributes', and 'Select Data'. The 'Select Data' component is highlighted with a green box, and a dashed orange arrow points to the 'Qt Select Data' dialog box. The dialog box is titled 'Qt Select Data' and contains the following elements:

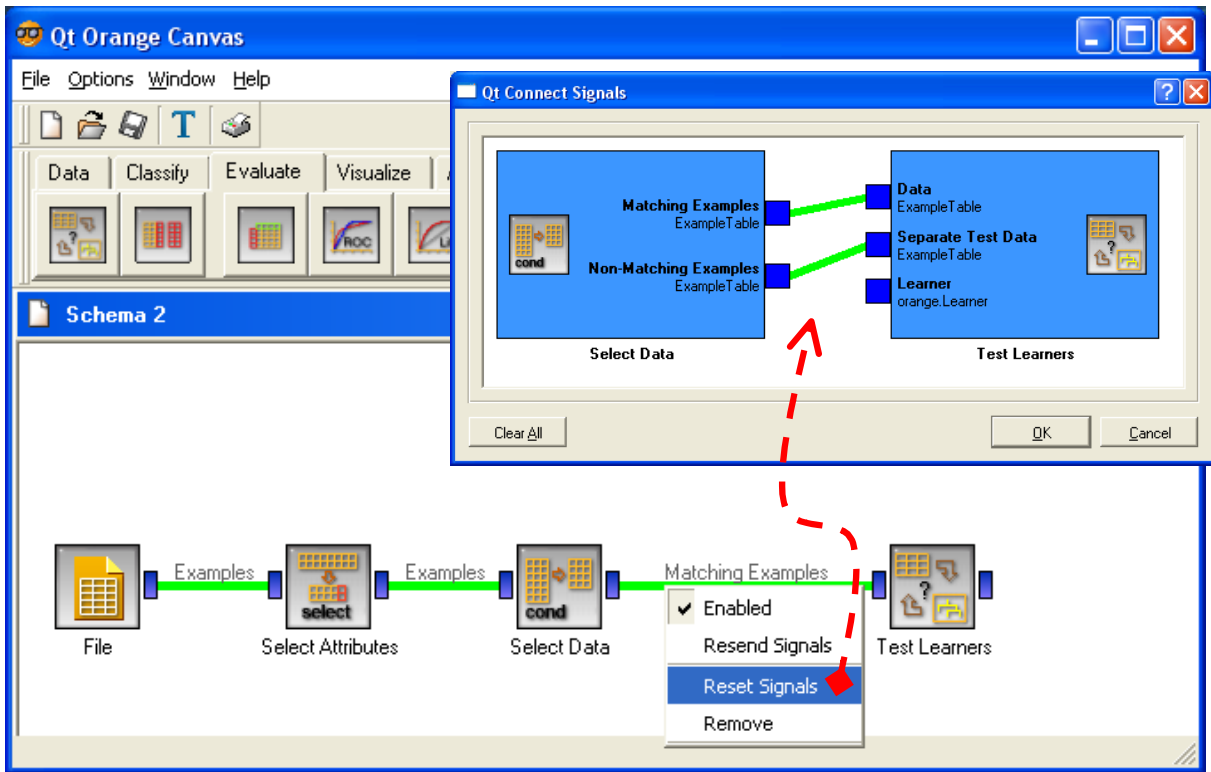
- Attribute Condition:** A list of attributes including 'alea93' through 'alea100', 'classe', and 'sample'. The 'sample' attribute is selected.
- Operator:** A dropdown menu with 'equals' selected.
- Values:** A list of values including 'learning' and 'test', with 'learning' selected.
- Data Selection Criteria:** A table with the following content:

Active	Condition
1	sample equals learning
- Data In:** 33334 examples, 123 attributes.
- Data Out:** 10000 examples, 122 attributes.
- Commit:** Checkboxes for 'Remove unused values/attributes', 'Remove unused classes', and 'Commit on change'.

Dans la boîte de dialogue, nous introduisons la condition « SAMPLE equals LEARNING » pour définir les individus actifs. Orange nous indique le bon effectif, soit 10000 individus.

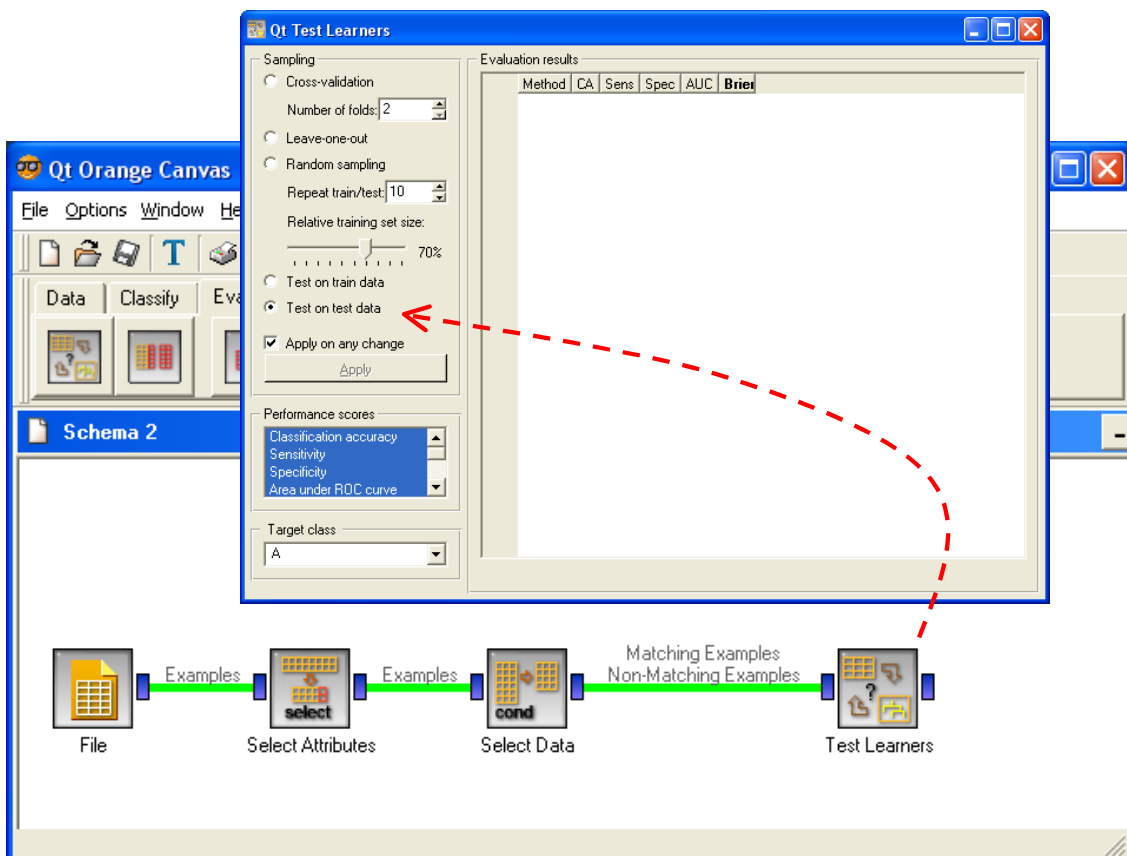
3.3.3 Préparation du composant d'évaluation

Assez curieusement, il est opportun à ce stade de placer et paramétrer le composant d'évaluation TEST LEARNERS (onglet EVALUATE). Nous lui connectons le composant précédent... et ça ne suffit pas, nous devons paramétrer plus finement la connexion en actionnant le menu contextuel RESET SIGNALS (clic droit sur la connexion).



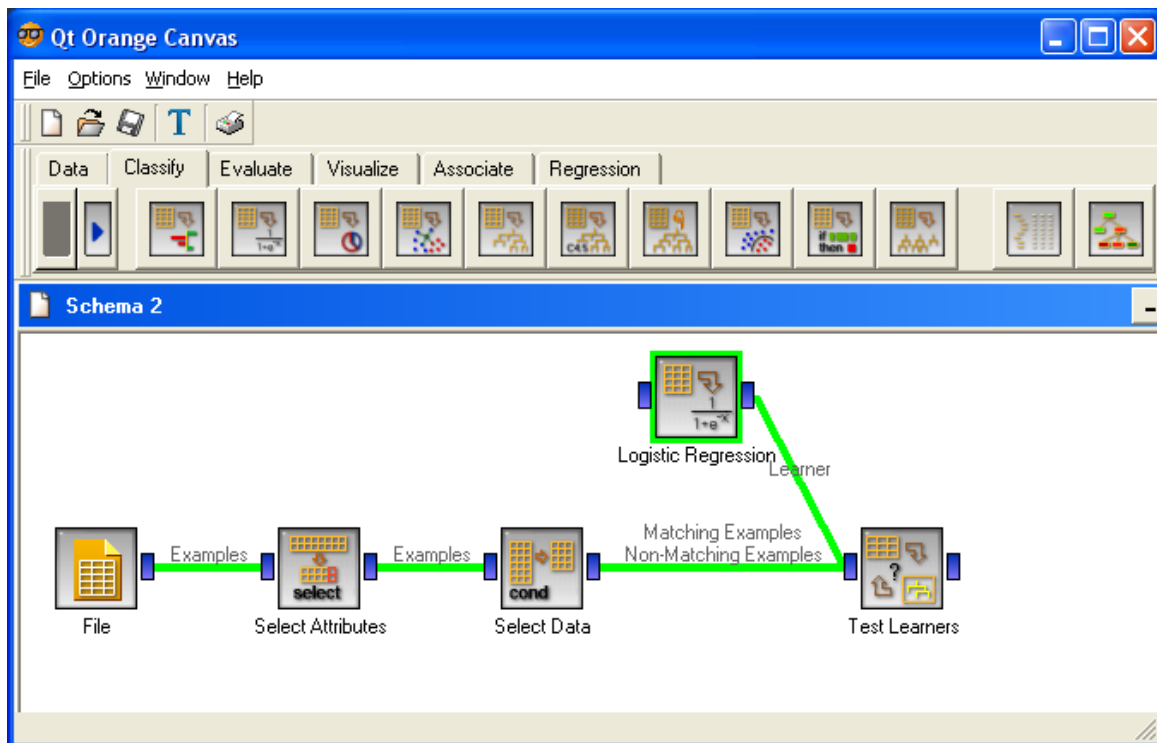
Dans la boîte de dialogue qui apparaît, nous précisons que les données en « apprentissage » **et** « test » doivent être envoyées au composant TEST LEARNERS. Après avoir validé, Orange indique dans le schéma les informations qui transitent d’une icône à l’autre.

Il ne nous reste plus qu’à paramétrer le composant TEST LEARNERS lui même via le menu contextuel OPEN. Nous indiquons que l’évaluation doit être réalisée sur l’échantillon test.



3.3.4 Régression logistique

Pour lancer la régression logistique, nous plaçons le composant LOGISTIC REGRESSION (onglet CLASSIFY) dans le schéma. Nous le connectons à l'icône TEST LEARNERS, d'où l'intérêt de l'avoir préalablement préparé convenablement ce dernier. Le calcul est relativement rapide.



Dans la fenêtre de TEST LEARNERS apparaissent maintenant les indicateurs de qualité du modèle.

The Qt Test Learners window displays the following settings and results:

Sampling:

- Cross-validation
 - Number of folds: 2
- Leave-one-out
- Random sampling
 - Repeat train/test: 10
 - Relative training set size: 70%
- Test on train data
- Test on test data
- Apply on any change

Performance scores:

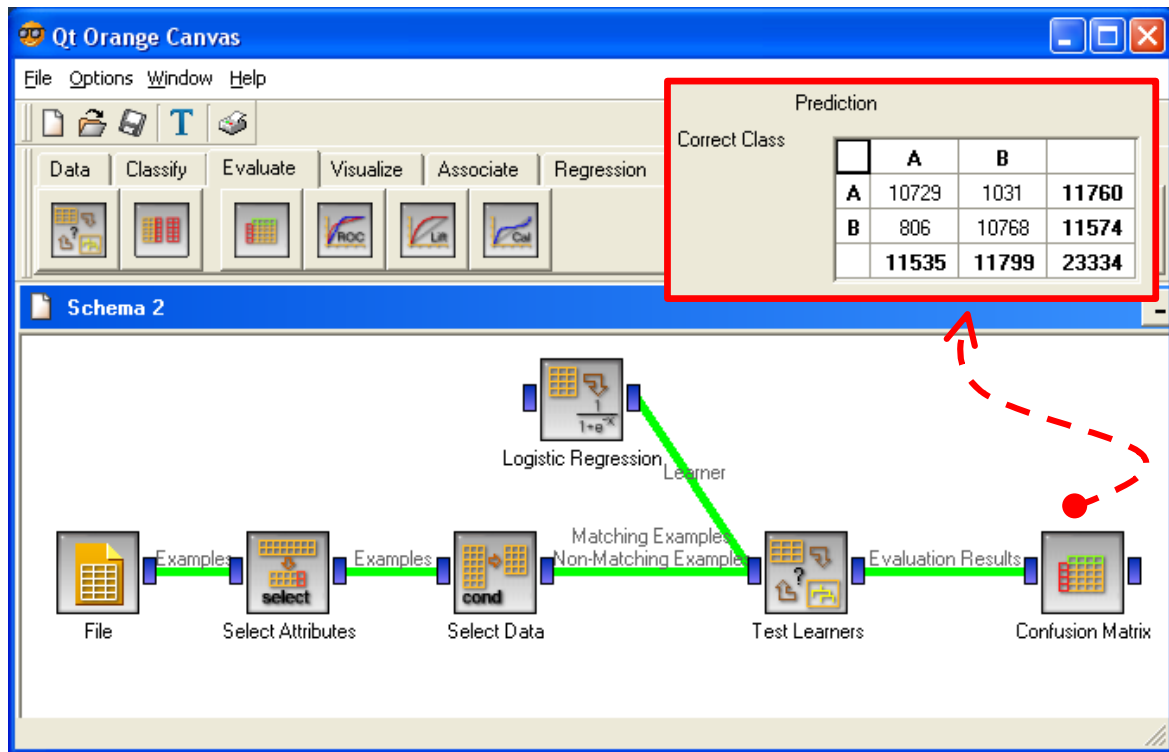
- Classification accuracy
- Sensitivity
- Specificity
- Area under ROC curve

Target class: A

Evaluation results:

Method	CA	Sens	Spec	AUC	Brier
1 Logistic regression	0.9213	0.9123	0.9304	0.9803	0.1115

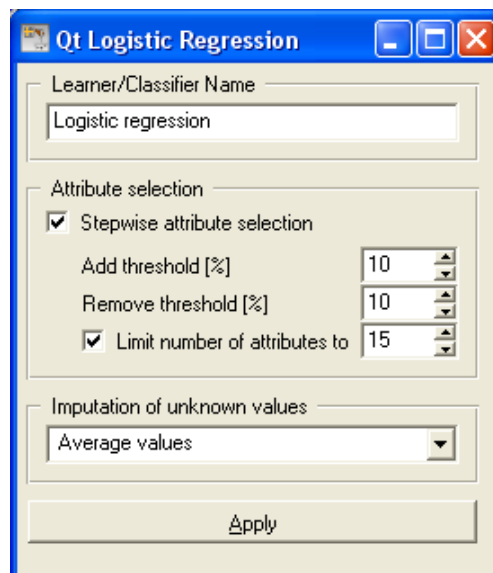
Le taux de succès en test est 92.13%, soit un taux d'erreur de 7.87%. Si nous souhaitons avoir le détail de la matrice de confusion, nous connectons le composant CONFUSION MATRIX (onglet EVALUATE) à la suite de TEST LEARNERS. Nous obtenons l'affichage suivant.



3.3.5 Sélection de variables

Il est possible de procéder à une sélection de variables avec la régression logistique. En actionnant le menu contextuel OPEN de LOGISTIC REGRESSION, nous pouvons demander une régression pas à pas. Les seuils de significativité sont en pourcentage. J'avoue ne pas très bien saisir le principe sous jacent. Est-ce en pourcentage de la déviance ? Le fichier d'aide n'est pas très clair à ce sujet.

Pour l'heure, nous demandons une recherche pas à pas (STEPWISE). Orange essaie itérativement d'ajouter et supprimer des variables. Après avoir constaté a posteriori que le calcul pouvait être très long, nous limitons la recherche à 15 variables. Nous cliquons sur le bouton APPLY.



Sur le composant CONFUSION MATRIX, nous obtenons une matrice de confusion identique à celle de GLM de R ou de Binary Logistic Regression de Tanagra, avec un taux d'erreur de $(1011+777)/23334 = 7.66\%$.

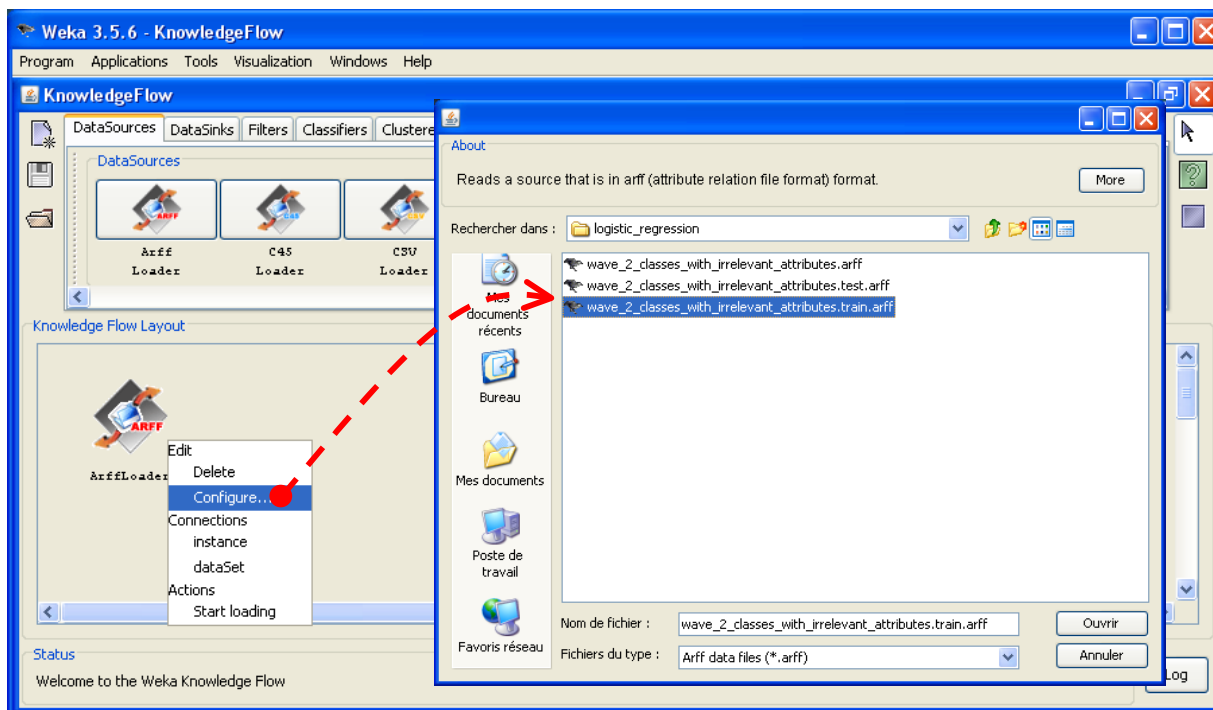
		Prediction		
Correct Class		A	B	
A	10749	1011	11760	
B	777	10797	11574	
	11526	11808	23334	

3.4 WEKA

WEKA n'est plus à présenter (<http://www.cs.waikato.ac.nz/ml/weka/>). C'est certainement une des références en matière de logiciel libre dans le data mining. Nous utilisons le mode KNOWLEDGE FLOW. N'ayant pas trouvé comment partitionner le fichier à l'aide de la variable SAMPLE, nous avons dû scinder le fichier manuellement en TRAIN et TEST et évacuer la variable SAMPLE. Nous avons bien sûr respecté l'affectation des individus aux échantillons d'apprentissage et de test. Ainsi, les résultats seront directement comparables avec ceux des autres logiciels.

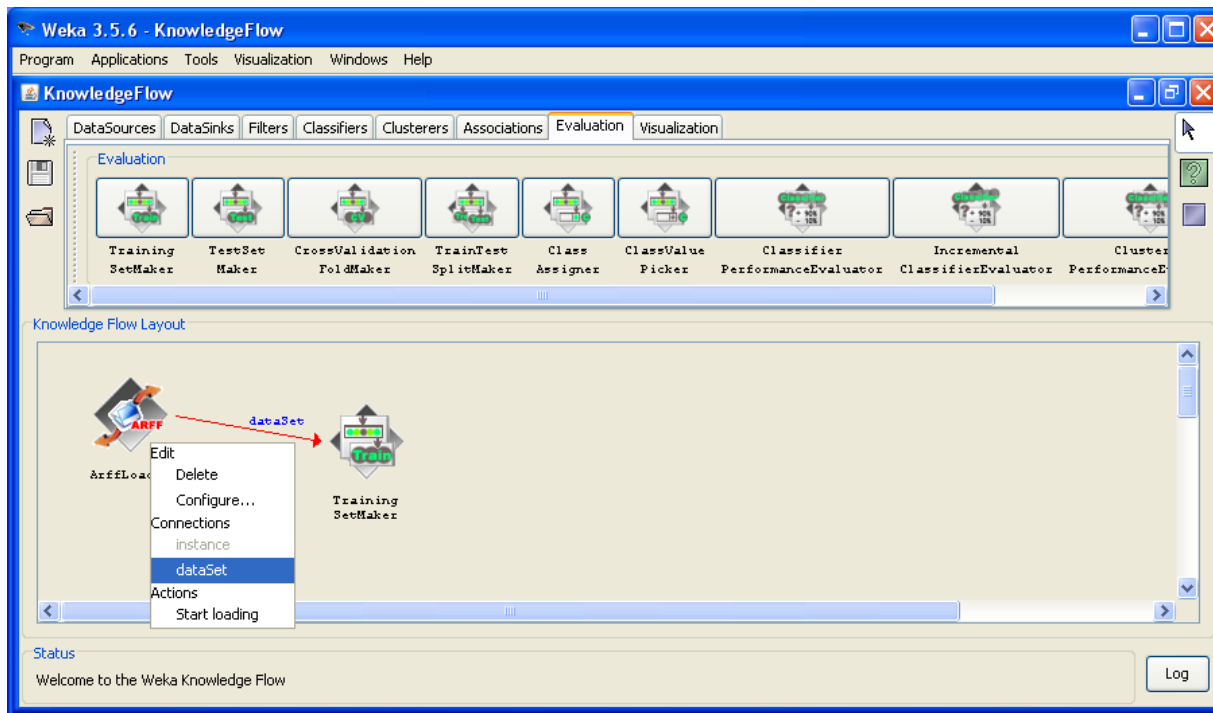
3.4.1 Régression logistique

Nous insérons le composant d'accès au fichier ARFF, ARFFLOADER (onglet DATASOURCES). Nous actionnons le menu contextuel CONFIGURE, nous sélectionnons le fichier wave_2_classes_with_irrelevant_attributes.train.arff.

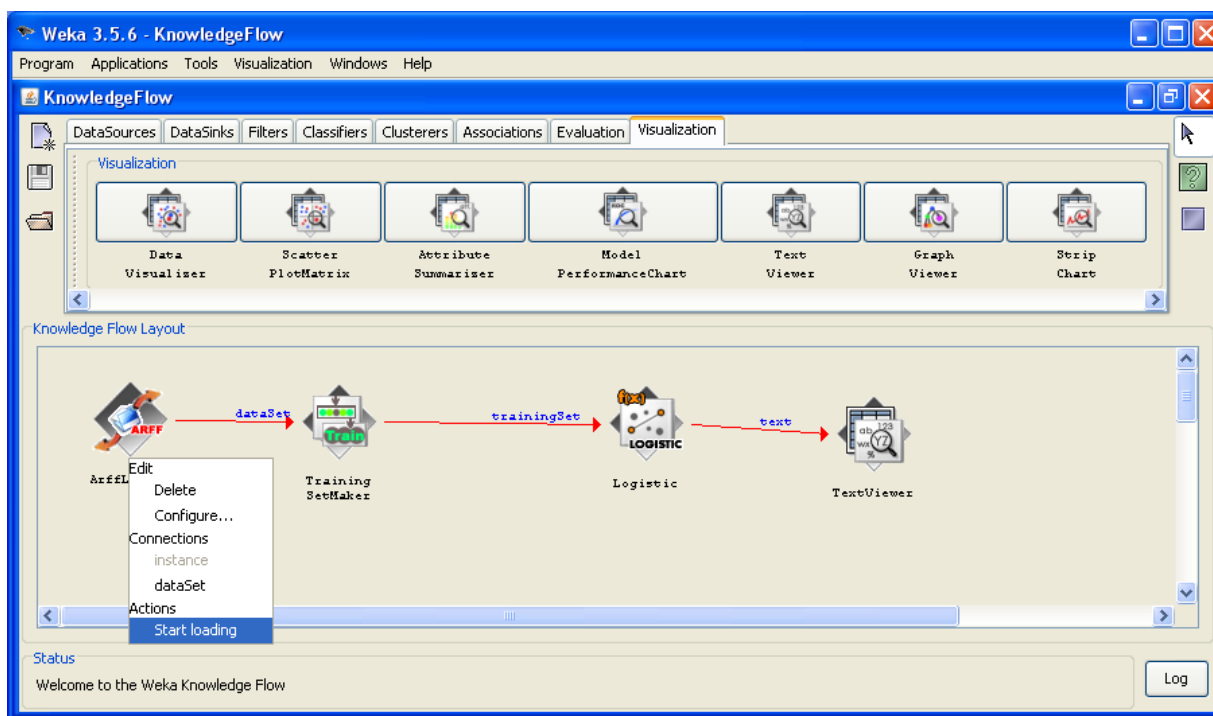


Weka considère que la dernière colonne des données correspond à la variable à prédire, les autres sont les variables prédictives. Ce qui est le cas chez nous. Il n'est donc pas nécessaire de préciser explicitement le rôle des variables. Dans le cas contraire, il aurait fallu utiliser le composant CLASS ASSIGNER (onglet EVALUATION).

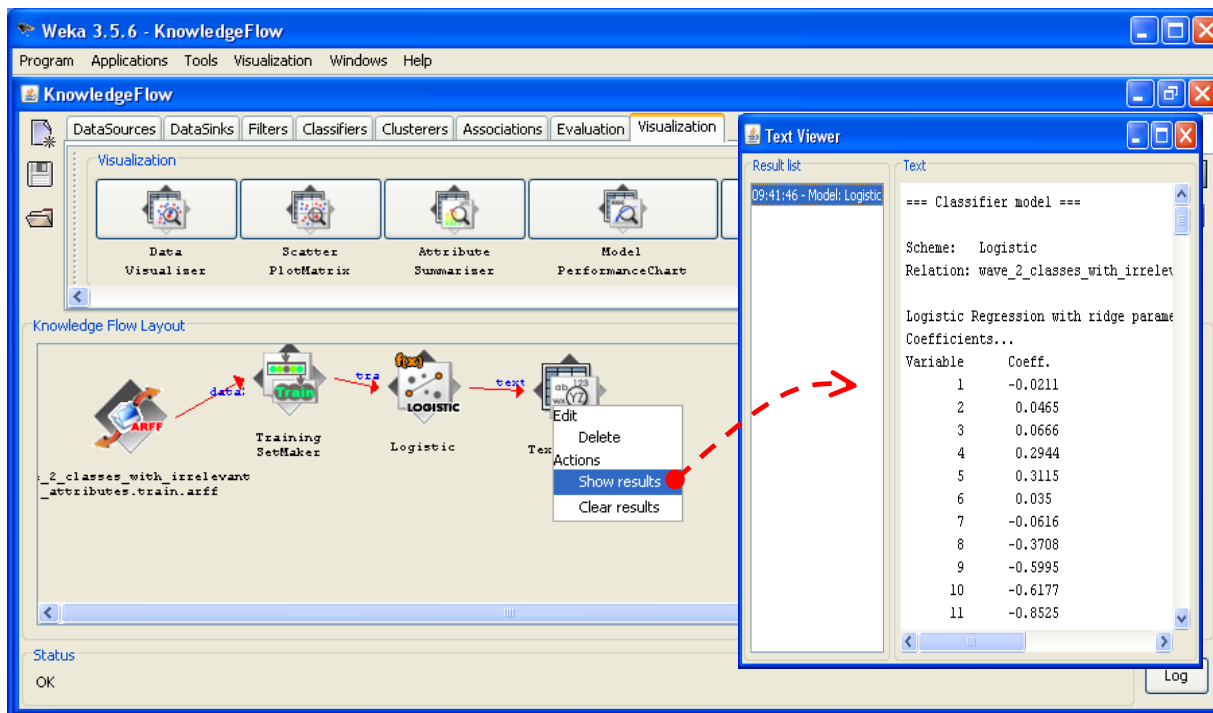
Pour indiquer que la totalité du fichier doit être utilisé en tant qu'échantillon d'apprentissage, nous introduisons le composant TRAINING SET MAKER (onglet EVALUATE). Nous établissons la connexion en faisant : un clic droit sur ARFFLOADER, nous choisissons l'option DATASET, puis la relierons à TRAINING SET MAKER.



Nous plaçons et relierons alors successivement le composant de calcul, LOGISTIC (onglet CLASSIFIERS), et celui de visualisation des résultats, TEXT VIEWER (onglet VISUALIZATION). Dans la copie d'écran, on remarquera entre chaque icône l'information qui est transmise. Pour lancer les calculs, nous actionnons le menu contextuel START LOADING... de l'icône ARFFLOADER.



Il ne nous reste plus qu'à actionner le menu SHOW RESULTS de TEXTVIEWER pour accéder aux résultats.



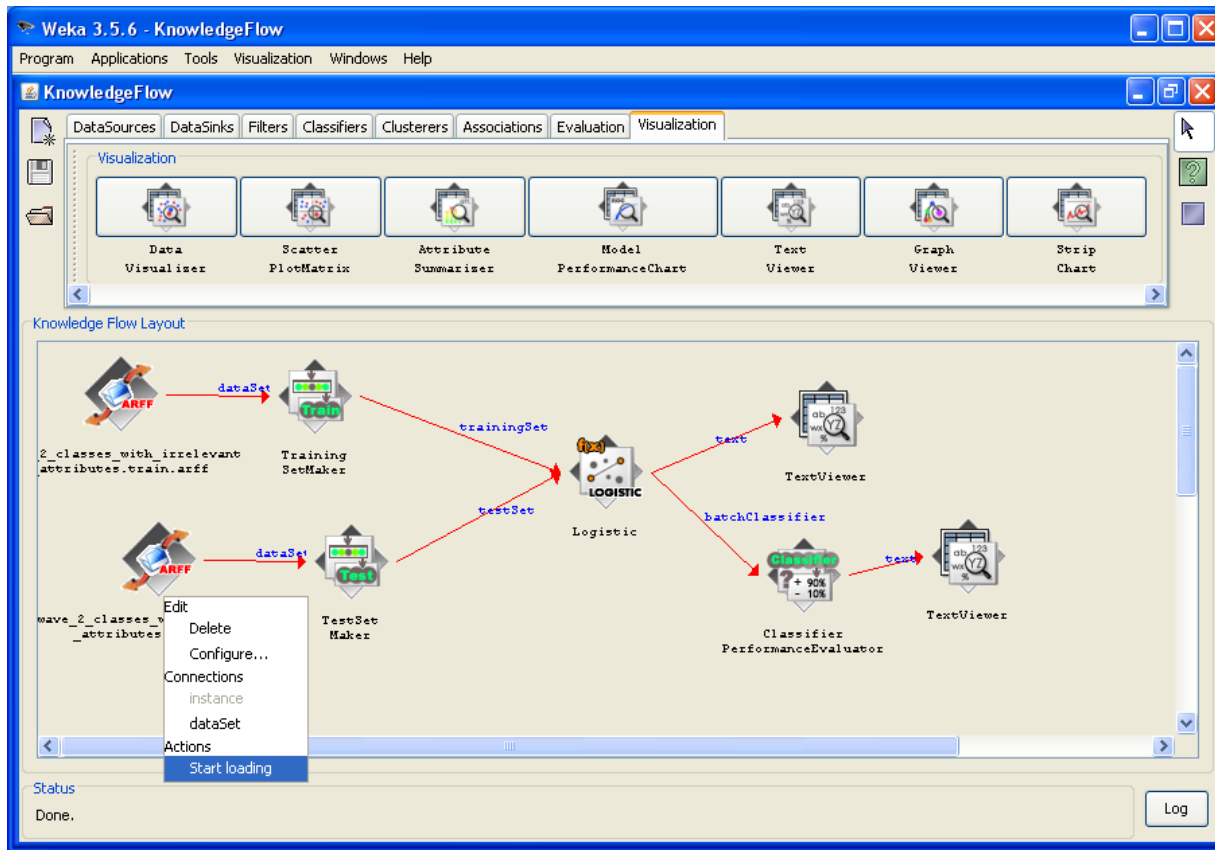
WEKA affiche directement les coefficients de l'équation de régression et leur exponentielle (odds-ratio). Nous ne disposons pas d'informations sur la qualité de l'apprentissage.

Remarque : Pourquoi Weka ne donne pas d'indications sur la significativité des coefficients ? Ça m'a beaucoup intrigué. La réponse nous l'avons à la lecture du code source des classes Logistic et Optimization. Weka utilise un quasi-newton pour optimiser la vraisemblance. La matrice Hessienne est approximée à l'aide d'un BFGS (<http://fr.wikipedia.org/wiki/BFGS>). Elle est suffisamment précise pour produire une optimisation de qualité. Elle ne l'est pas pour estimer correctement la matrice de variance covariance. Pour obtenir les statistiques de Wald, il faudrait compléter le code source en rajoutant un post traitement : calculer la matrice Hessienne à l'aide de l'expression analytique, son inverse est la matrice de variance covariance des coefficients.

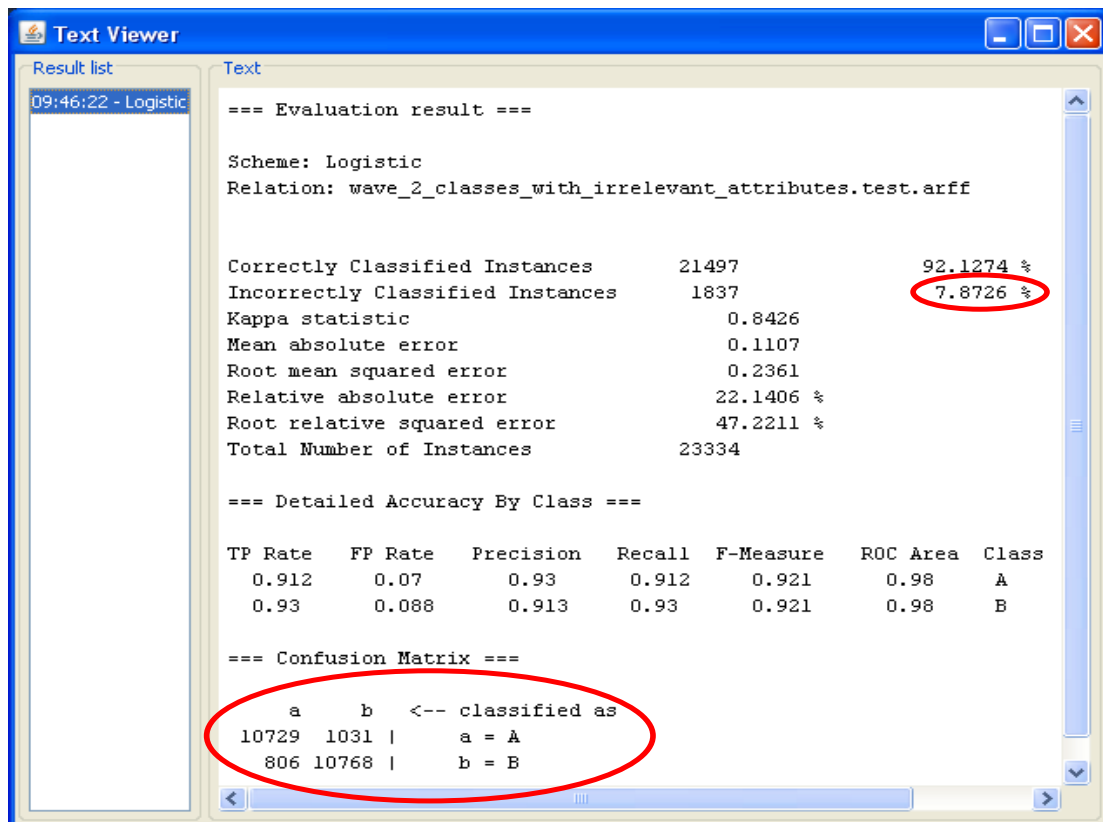
3.4.2 Evaluation sur les données test

Pour évaluer les performances sur l'échantillon test, nous devons : insérer un nouvel outil d'accès aux données sur « wave_2_classes_with_irrelevant_attributes.test.arff », spécifier qu'il s'agit d'un échantillon test via le composant TEST SET MAKER (onglet EVALUATION), puis utiliser le composant CLASSIFIER PERFORMANCE EVALUATOR (onglet EVALUATION). Un TEXT VIEWER complète le tout afin d'avoir accès aux résultats.

De nouveau, nous actionnons le menu START LOADING du ARFF LOADER pour lancer les calculs.



Les résultats sont visibles en cliquant sur le menu SHOW RESULTS de l'outil de visualisation.



La matrice de confusion est identique à celle des autres approches.

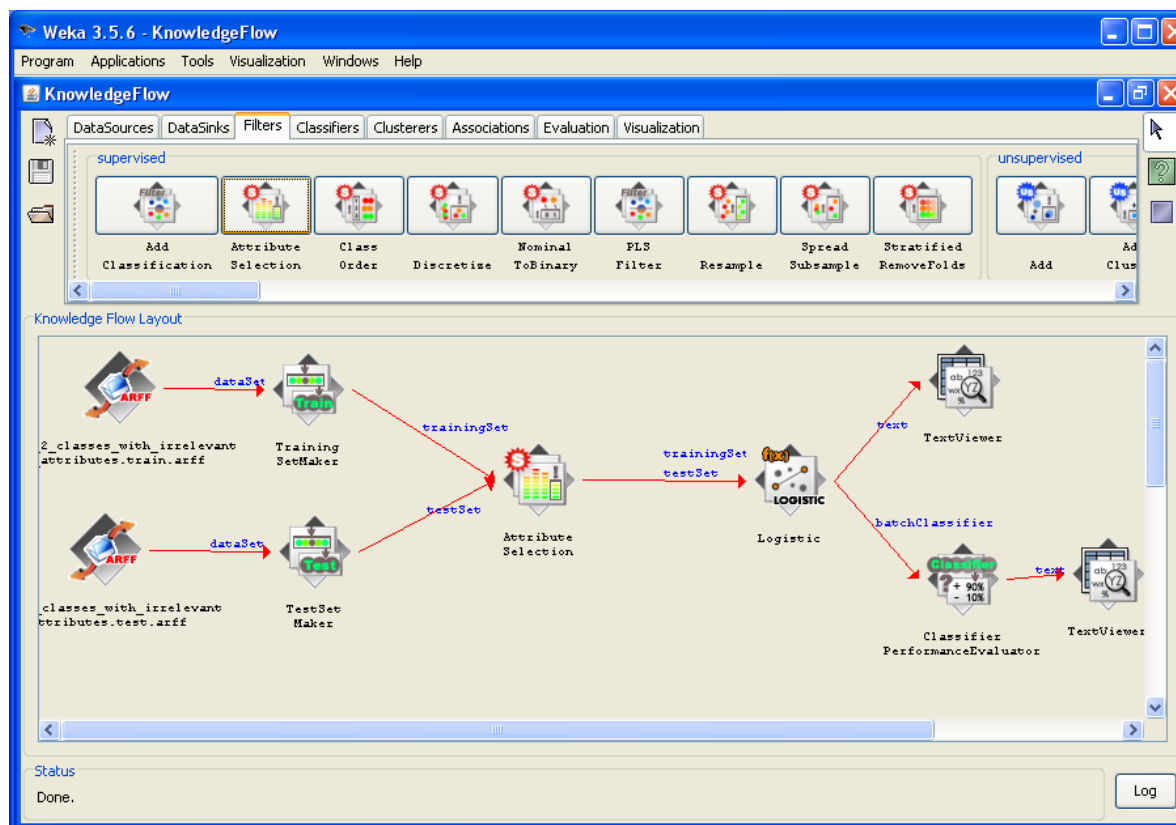
3.4.3 (Tentative de) Sélection de variables

L'affaire se corse lorsqu'il s'agit de procéder à une sélection de variables. En effet, il n'existe de technique en accord (direct) avec la régression logistique dans WEKA. Nous pouvons en revanche choisir une méthode de filtrage générique qui permet d'évacuer automatiquement les variables non pertinentes. Rien ne nous garantit néanmoins que les variables retenues sont celles qui sont « optimales » pour la régression logistique.

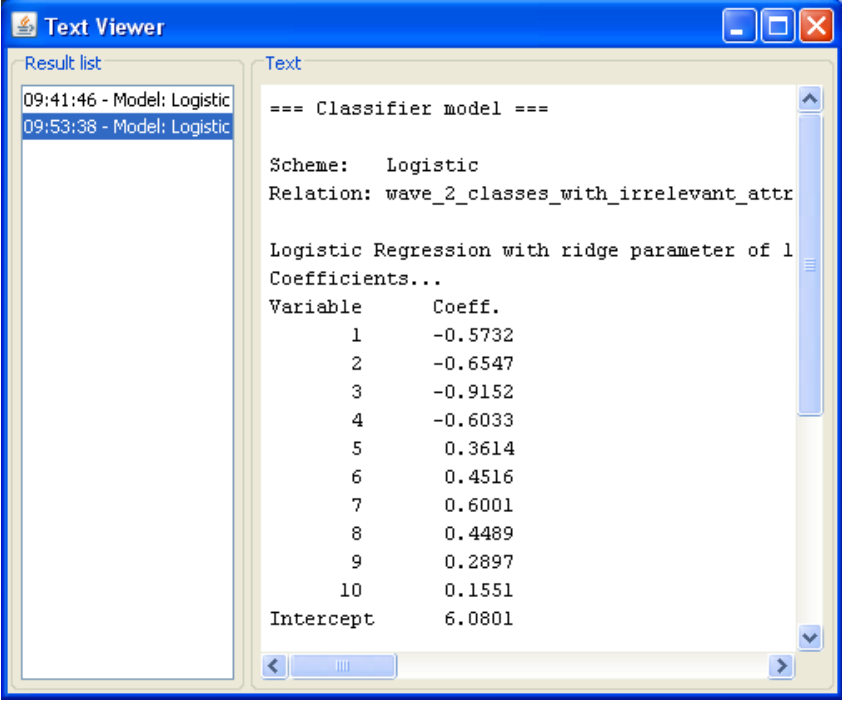
Ces réserves émises, essayons de voir comment introduire la sélection d'attribut dans le KNOWLEDGE FLOW. Nous utilisons le composant ATTRIBUTE SELECTION (onglet FILTERS). Notons en particulier la connexion établie entre ATTRIBUTE SELECTION et LOGISTIC. Elle est doublée. Les données en apprentissage **et** en test doivent être transférées.

Petite précision utile, ATTRIBUTE SELECTION utilise par défaut la CFSSUBSETEVAL comme critère d'évaluation et BESTFIRST comme stratégie de recherche. Si l'on se fie à la documentation, il semble que ce soit l'algorithme CFS qui soit utilisé (M. A. Hall (1998). Correlation-based Feature Subset Selection for Machine Learning. Hamilton, New Zealand.).

Le diagramme de calcul se présente comme suit.



Après avoir cliqué sur STARTLOADING du premier composant d'accès aux données, la filière est recalculée. Les résultats sont accessibles via le TEXTVIEWER associé.



Text Viewer

Result list

- 09:41:46 - Model: Logistic
- 09:53:38 - Model: Logistic

Text

```

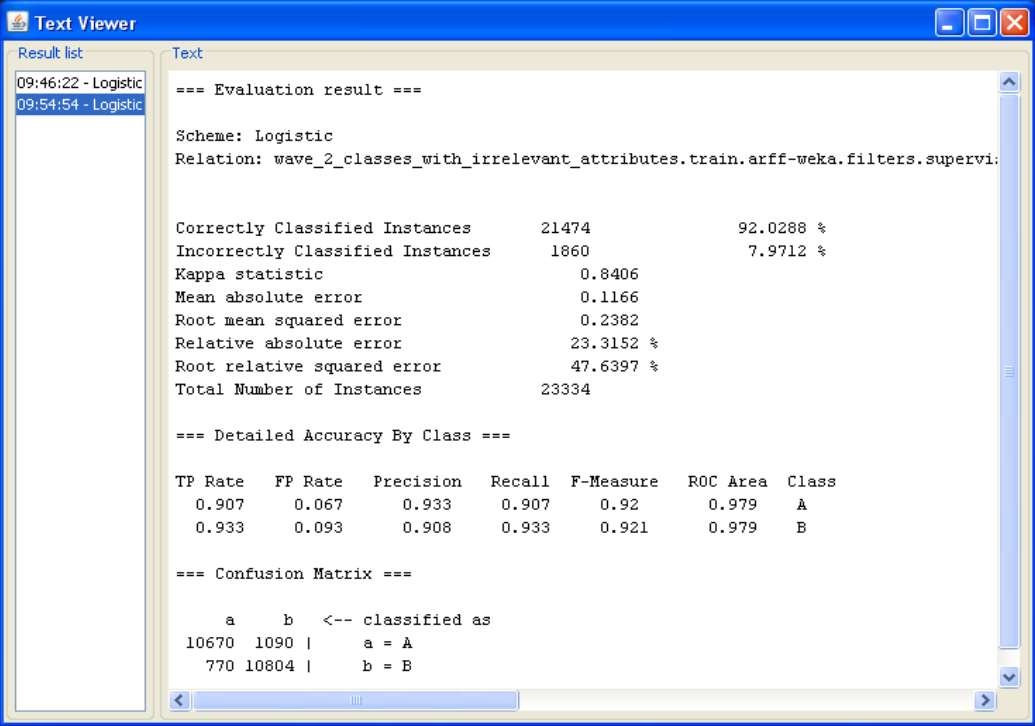
=== Classifier model ===

Scheme:   Logistic
Relation: wave_2_classes_with_irrelevant_attr

Logistic Regression with ridge parameter of 1
Coefficients...
Variable      Coeff.
      1      -0.5732
      2      -0.6547
      3      -0.9152
      4      -0.6033
      5       0.3614
      6       0.4516
      7       0.6001
      8       0.4489
      9       0.2897
     10       0.1551
Intercept    6.0801
  
```

10 variables ont été sélectionnées. Mais comme elles ont été re-numérotées, j'avoue n'avoir aucune idée sur leur identité. Nous verrons plus loin comment faire pour obtenir la liste adéquate.

Concernant les performances en test, nous actionnons STARTLOADING pour le second composant ARFF LOADER, nous obtenons la matrice de confusion suivante.



Text Viewer

Result list

- 09:46:22 - Logistic
- 09:54:54 - Logistic

Text

```

=== Evaluation result ===

Scheme: Logistic
Relation: wave_2_classes_with_irrelevant_attributes.train.arff-weka.filters.supervi:

Correctly Classified Instances      21474          92.0288 %
Incorrectly Classified Instances    1860           7.9712 %
Kappa statistic                    0.8406
Mean absolute error                 0.1166
Root mean squared error             0.2382
Relative absolute error             23.3152 %
Root relative squared error         47.6397 %
Total Number of Instances          23334

=== Detailed Accuracy By Class ===

TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
 0.907   0.067   0.933     0.907   0.92       0.979    A
 0.933   0.093   0.908     0.933   0.921     0.979    B

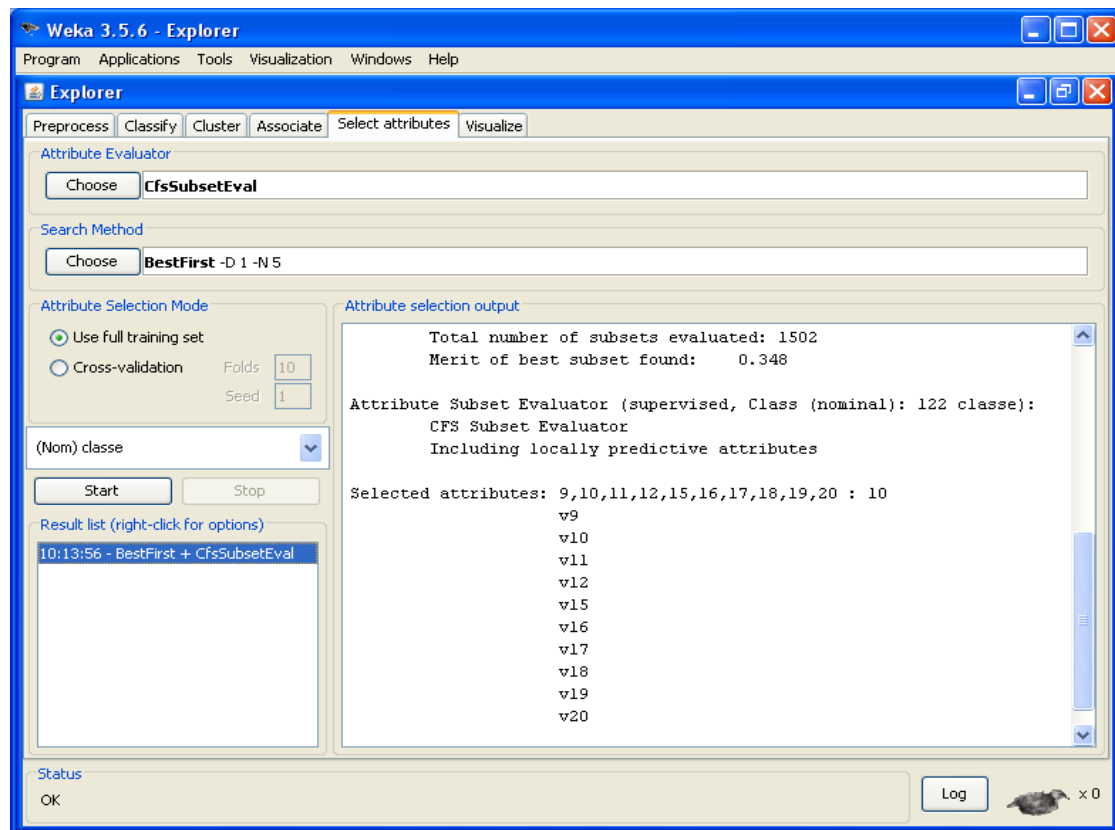
=== Confusion Matrix ===

  a   b  <-- classified as
10670 1090 |   a = A
 770 10804 |   b = B
  
```

Avec $(1090 + 770) = 1860$ mal classés, soit un taux d'erreur de 7.97%.

3.4.4 Liste des variables sélectionnées

10 variables ont été sélectionnées, mais on ne sait pas lesquelles. C'est quand même un peu ennuyeux. Pour les identifier, nous utilisons le module EXPLORER de WEKA. Nous réitérons la même opération en veillant à utiliser exactement les mêmes paramètres. WEKA nous propose :



Aucune des variables ALEA n'a été indûment retenue. Il semble néanmoins que la solution CFS soit trop restrictive pour la régression logistique. Nous avons 72 mal classés supplémentaires par rapport à la solution proposée par Tanagra, R et Orange (1788 mal classés). Ceci étant, 72 mal classés sur 23334 observations, on ne va pas non plus commencer à s'escrimer là dessus. D'autant plus que nous avons 5 variables en moins ici. Les solutions se tiennent. Tout dépend de nos priorités.

3.5 R (avec la procédure Logistic du package RWeka)

La procédure LOGISTIC de Weka (section 3.4) peut être appelée dans R via le package RWeka (version 0.3-13). L'avantage est de disposer de tout l'arsenal de R en matière : de gestion et de manipulation des données, de programmation, d'accès aux fonctionnalités issues des autres packages. Les possibilités deviennent tout simplement infinies. Nous pourrions par exemple, confronter en deux / trois commandes les prédictions de **glm(.)** et de **Logistic(.)** pour identifier les observations pour lesquelles les deux procédures sont en désaccord, etc, etc.

Pour l'heure, comme précédemment (section 3.2), nous importons de nouveau les données, puis nous les partitionnons en « apprentissage » - « test ».

```

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]

```

Nous pouvons alors lancer la procédure **Logistic(.)**

```

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

```

Nous obtenons les résultats suivants au bout de 8 secondes de calcul :

```

> #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

```

Le taux d'erreur en apprentissage est de 7.89%, exactement que pour le composant Logistic appelé directement dans Weka (heureusement). La fonction **print(.)**, elle, envoie directement la liste des coefficients, toujours sans aucune indication toujours sur leur pertinence.

La fonction **predict(.)** sait fournir directement les classes d'affectation pour chaque individu. Pour évaluer les performances sur l'échantillon test, nous faisons

```

#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)

```

Nous obtenons un taux d'erreur de 7.87%, avec (1031 + 806) 1837 individus mal classés.

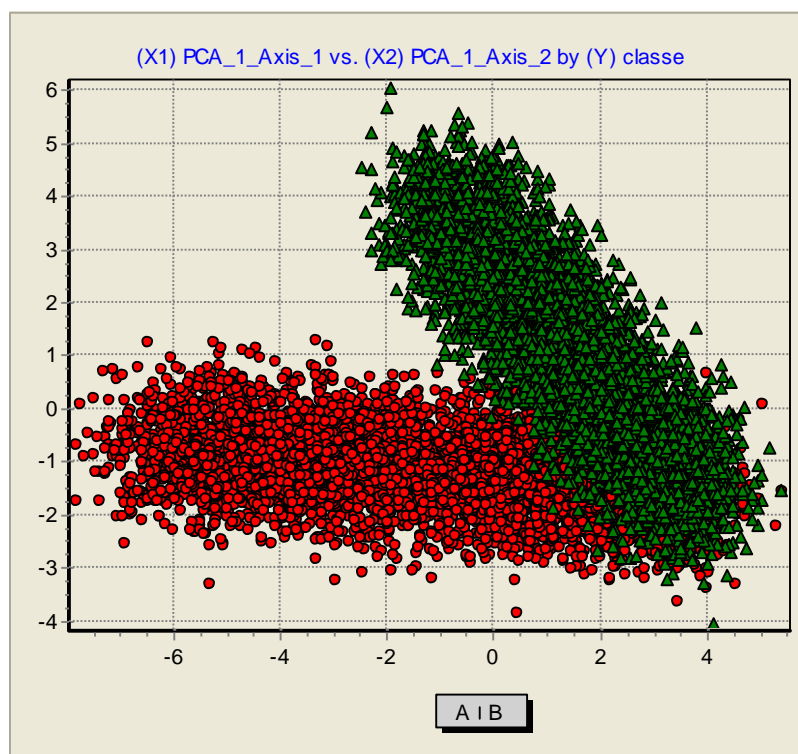

```
> #prediction sur la partie test
> pred <- predict(modele, newdata = donnees.test, type = "class")
>
> #matrice de confusion
> mc <- table(donnees.test$classe,pred)
> print(mc)
  pred
  A    B
A 10729 1031
B   806 10768
>
> #taux d'erreur en test
> erreur <- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
> print(erreur)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

Les fonctions de sélection de variables ne semblent pas disponibles dans le package RWeka (<http://cran.r-project.org/web/packages/RWeka/index.html>). Nous n'avons donc pas pu les mettre en œuvre dans cette section.

4 Récapitulatif

4.1 Comparaison des performances

Performances en classement. Pour être tout à fait honnête, le problème que nous traitons dans ce didacticiel est relativement simple. Projetés dans le premier axe factoriel, nous constatons que les classes A et B sont discernables, avec un recouvrement correspondant au taux d'erreur. Il n'est pas étonnant que les performances en classement soient plutôt bonnes. Et, ce n'était pas évident, les logiciels n'utilisant pas exactement les mêmes techniques d'optimisation en interne, ils fournissent exactement les mêmes modèles prédictifs c.-à-d. ils classent de manière identique les individus de l'échantillon test.



La véritable gageure est donc la sélection de variables. Nous constatons que les logiciels s'en tirent plutôt bien, avec des résultats qui se valent alors que les techniques utilisées sont différentes. Nous les récapitulons dans le tableau suivant :

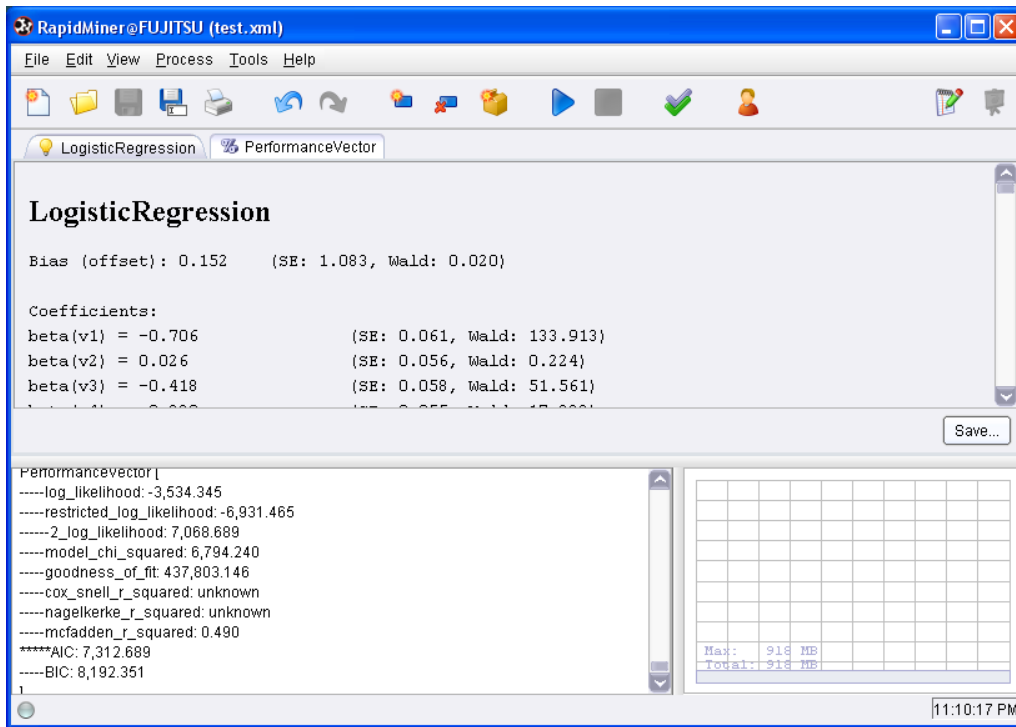
Logiciel	Taux d'erreur en test	Nombre de variables sélectionnées (variables « alea » incluses) [Méthode]	Taux d'erreur en test du modèle avec les variables sélectionnées
Tanagra	0.0787	15 (0) [Statistique - Méthode des scores]	0.0766
R (glm)	0.0787	15 (0) [Optimisation critère BIC]	0.0766
Orange	0.0787	15 (0) [Statistique - Limitée explicitement ici]	0.0766
Weka (Logistic)	0.0787	10 (0) [méthode CFS]	0.0797
R (RWeka package)	0.0787	-	-

Temps de calcul. Concernant les temps de calculs, mis à part la procédure **glm(.)** de R qui est réellement au dessus du lot (**Note de mise à jour. Les temps de calculs annoncés ont évolué. La version de Tanagra utilisée dans ce didacticiel est la 1.4.27. Avec la version 1.4.37, après un gros travail d'optimisation, la durée de construction du modèle est maintenant de 1.8 secondes, sur la même machine et dans les mêmes conditions ; la sélection « forward » passe à 12.5 secondes. Nous étudierons plus en détail les temps de calculs dans un prochain didacticiel, nous utiliserons un fichier de taille plus importante pour différencier de manière plus marquée les comportements.**). On constate que les logiciels se tiennent même s'ils utilisent par ailleurs des paradigmes d'optimisation différents. On imagine également, sans l'avoir vraiment testé dans nos expérimentations, que le paramétrage des techniques d'optimisation (tolérance, etc.) doit influencer sur la qualité et la durée de l'optimisation.

4.2 Les autres logiciels

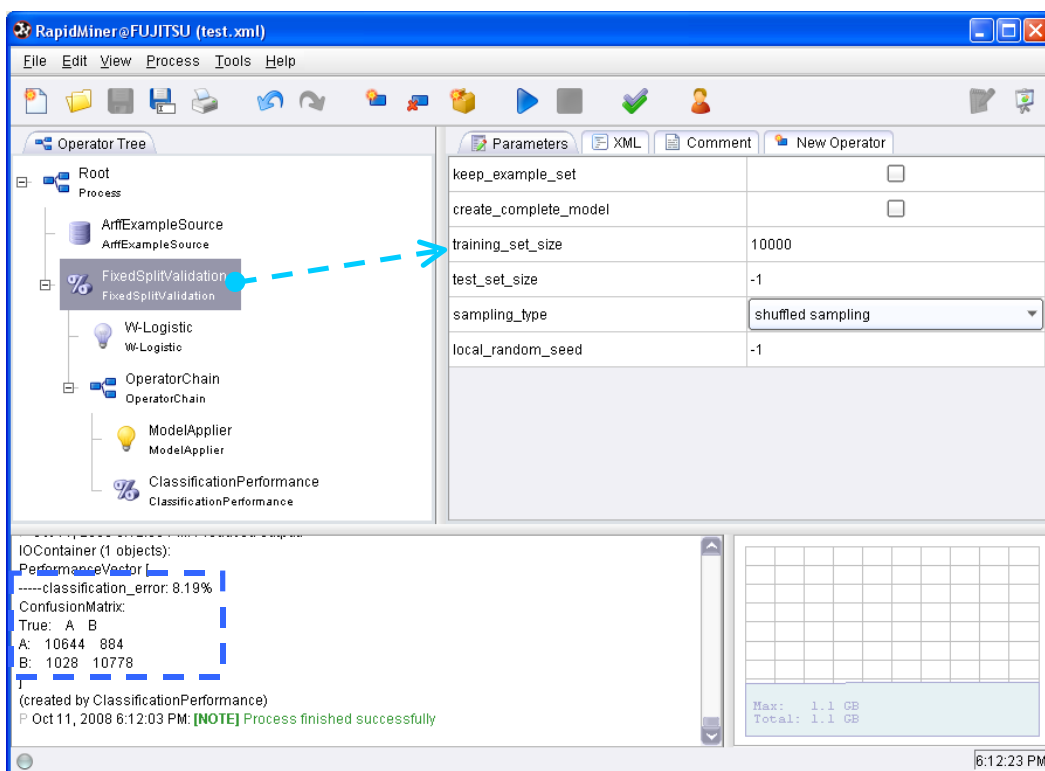
Parmi les autres logiciels libres de qualité, j'ai essayé de voir si la régression logistique était présente, et sous quelle forme :

- KNIME (version 1.3.5) : la régression logistique n'est pas proposée.
- RAPIDMINER (version 4.2) : deux composants de régression logistique sont proposés, Logistic Regression et Kernel Logistic Regression. La documentation n'est pas très disserte quant à ces outils. Il semble qu'ils s'appuient sur un algorithme génétique pour optimiser la log vraisemblance. J'ai essayé de lancer le premier en limitant le nombre de générations à 500 afin que les calculs ne soient pas trop dispendieux. Au bout de 308 secondes (plus de 5 minutes), RAPIDMINER fournit les coefficients et leurs écart-types, mais avec une qualité d'optimisation très en deçà des autres logiciels de ce comparatif (BIC = 8192.351 quand GLM de R obtient 4710.35 par exemple).



Nul doute qu'il faudrait se pencher un peu plus sur le contenu de ces composants, voir entre autres ce qu'il en est des différents paramètres, avant d'en tirer des conclusions péremptoires.

- RAPIDMINER (version 4.2 - Logistic de Weka) : Rapidminer intègre également la régression logistique de Weka. Il s'agit de la méthode Logistic toujours, nommée W-LOGISTIC ici. Je n'ai pas su subdiviser la base selon la variable SAMPLE. Les échantillons « apprentissage » et « test » ont donc été produits aléatoirement, en respectant néanmoins le rapport 10.000 / 23.334. Le taux d'erreur en test est 8.19%.



5 Conclusion

La régression logistique est une technique très largement utilisée. Par rapport aux anciens, nous avons la chance d'avoir à disposition des logiciels libres qui la propose sous différentes formes, avec des fonctionnalités plus ou moins avancées (sélection de variables, visualisation des résultats, application sur un échantillon test, déploiement sur de nouvelles observations non étiquetées, etc.).

Ces logiciels libres n'auront jamais les capacités des logiciels commerciaux en matière de reporting et d'industrialisation. Leur vocation est avant tout pédagogique. Plutôt que de les opposer, nous dirons plutôt qu'ils sont complémentaires. Je ne pense pas qu'il y a de « meilleur » logiciel dans l'absolu, je pense plutôt qu'il y a des logiciels qui répondent à des besoins. C'est notre rôle de définir précisément un cahier des charges adapté à nos objectifs et nos contraintes. Le choix de l'outil en découlera facilement.

6 Références

M. Bardos, « Analyse discriminante : Application au risque et au scoring financier », Dunod, 2001 ; chapitre 3.

P.L. Gonzalès, « Modèles linéaires généralisés » (chapitre 5) et « Modèles à réponses dichotomiques » (chapitre 6), in *Modèles Statistiques pour Données qualitatives*, Droesbeke, Lejeune et Saporta, éditeurs, Technip, 2005.

J.P. Nakache, J. Confais, « Statistique Explicative Appliquée », Technip, 2003 ; chapitre 4.

G. Saporta, « Probabilités, Analyse de données et Statistique », Technip, 2006 ; section 18.6.

M. Tenenhaus, « Statistique : Méthodes pour décrire, expliquer et prévoir », Dunod, 2007 ; chapitre 11.