1 Objectif

Comparer les performances des PLS-DA (Partial Least Squares Discriminant Analysis) avec celles des méthodes bien connues en apprentissage automatique.

La régression PLS est une technique de régression qui vise à prédire les valeurs prises par un groupe de variables Y (variables à prédire, variables cibles, variables expliquées) à partir d'une série de variables X (variables prédictives, les descripteurs, variables explicatives) (Tenenhaus[1], 1998 ; Garson, http://www2.chass.ncsu.edu/garson/PA765/pls.htm). Définie à l'origine au traitement des variables cibles continues, la Régression PLS peut être transposée à la prédiction d'une variable qualitative, de différentes manières[2], on parle « d'analyse discriminante PLS ». Elle fait alors preuve des qualités qu'on lui connaît habituellement, essentiellement la capacité à traiter un espace de représentation à très forte dimensionnalité, avec un grand nombre de descripteurs bruités et/ou redondants.

Ce document fait suite à un précédent didacticiel[3] où nous présentions différentes méthodes supervisées basées sur la Régression PLS. L'objectif est de montrer le comportement de l'une d'entre elles, PLS-LDA, dans un contexte où le nombre de descripteurs est élevé par rapport au nombre d'observations. Le ratio reste « raisonnable » (278 variables prédictives pour 232 observations en apprentissage). Nous pouvons néanmoins voir se dessiner dans cette expérimentation les principaux traits du traitement de ce type de données où, finalement, la maîtrise de la variance du classifieur est l'enjeu majeur. Pour confirmer cette idée, nous opposerons PLS-DA à des méthodes éprouvées telles que les SVM (Support Vector Machine, Librairie LIBSVM, Fan et al., 2005), les Random Forest (Breiman, 2001), ou... l'analyse discriminante linéaire[4] (Fisher, 1936 - *Ca peut paraître étrange de recourir à l'analyse discriminante dans notre contexte. Mais on verra que bien utilisée, elle se comportera tout à fait honorablement face aux autres méthodes*).

2 Données

Nous utilisons les données ARRHYTMIA.BDM[5]. Le fichier comporte 420 observations, 232 sont réservées pour l'apprentissage, 188 pour le test. La variable STATUS permet de distinguer les sous échantillons. La variable à prédire ARRHYTMIA est binaire. Elle indique la présence ou non d'arythmie cardiaque chez des patients. Les variables prédictives, toutes continues ou considérées comme telles, sont au nombre de 278. Attention, certaines sont composées d'une valeur unique, de ce fait totalement inutile pour la prédiction, elles ne doivent pas perturber le processus d'apprentissage.

Nous reproduisons dans ce didacticiel le schéma de comparaison de classifieurs déjà mis en avant dans un de nos didacticiels[6]. Nous forçons un peu le trait en intégrant un plus grand nombre de descripteurs. Les méthodes réputées stables devraient plus se démarquer. Nous

[1] L'ouvrage de M. Tenenhaus est certainement la référence francophone en matière de Régression PLS : M. Tenenhaus, « La régression PLS – Théorie et Pratique », Technip, 1998.

[2] Voir **S. Chevallier, D. Bertrand, A. Kohler, P. Courcoux, « Application of PLS-DA in multivariate image analysis », in J. Chemometrics, 20 : 221-229, 2006.**

[3] http://tutoriels-data-mining.blogspot.com/2008/05/analyse-discriminante-pls.html

[4] http://tutoriels-data-mining.blogspot.com/2008/04/analyse-discriminante-linaire.html.

[5] http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/arrhytmia.bdm ; au format binaire de TANAGRA.

[6] http://tutoriels-data-mining.blogspot.com/2008/03/comparaison-de-classifieurs.html

intégrons de plus de nouvelles méthodes dans le comparatif, notamment les méthodes dérivées de la Régression PLS, peu connues en apprentissage automatique.

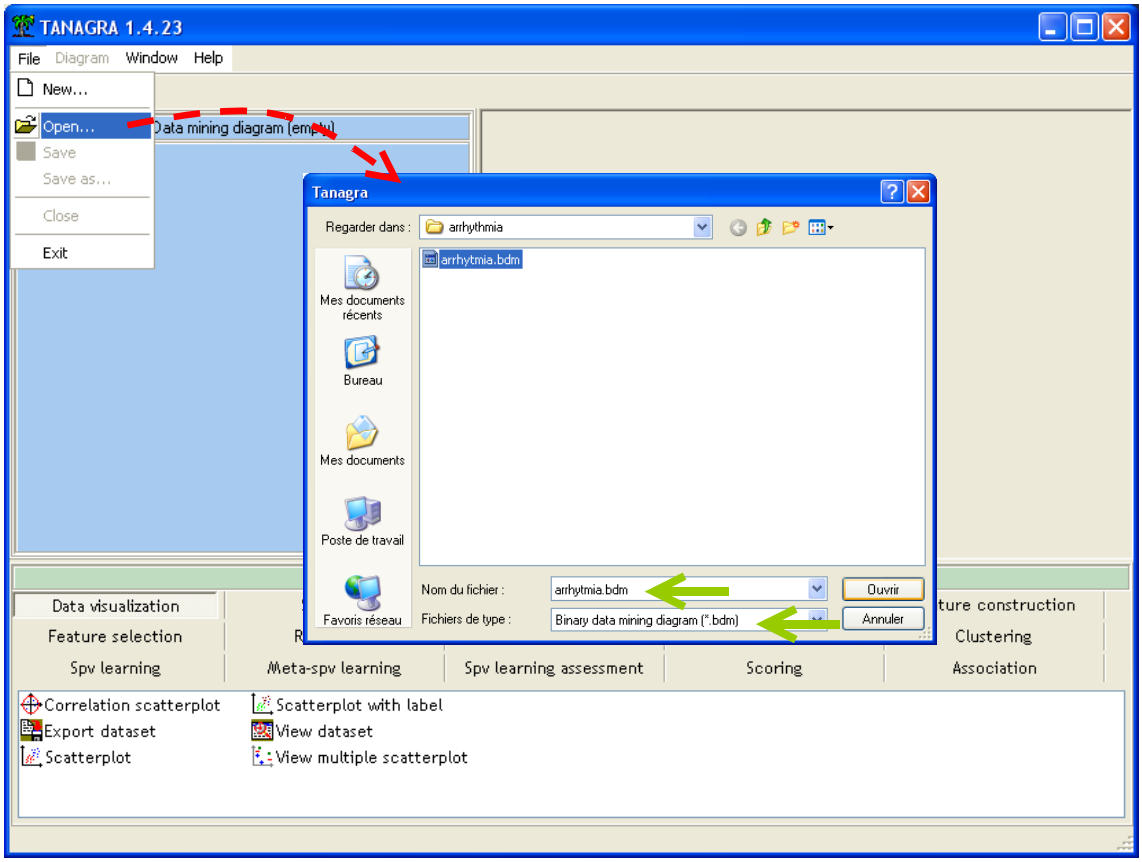
Concernant la comparaison sur l'échantillon test, il est évident que la faiblesse de l'effectif empêche toute conclusion définitive basée sur le taux d'erreur. Nous souhaitons avant tout positionner les méthodes selon leurs caractéristiques. Nous étudierons également le rôle du paramétrage, passé souvent sous silence dans les publications, mais qui jouent un rôle important dans le comportement des méthodes d'apprentissage.

Enfin, il est impossible de rappeler dans ce document les tenants et aboutissants des techniques abordées dans ce didacticiel. Le mieux est de se reporter à la documentation disponible sur le web, notamment les articles référencés sur notre portail Data Mining (voir le chapitre « Apprentissage Supervisé ») http://eric.univ-lyon2.fr/~ricco/data-mining/

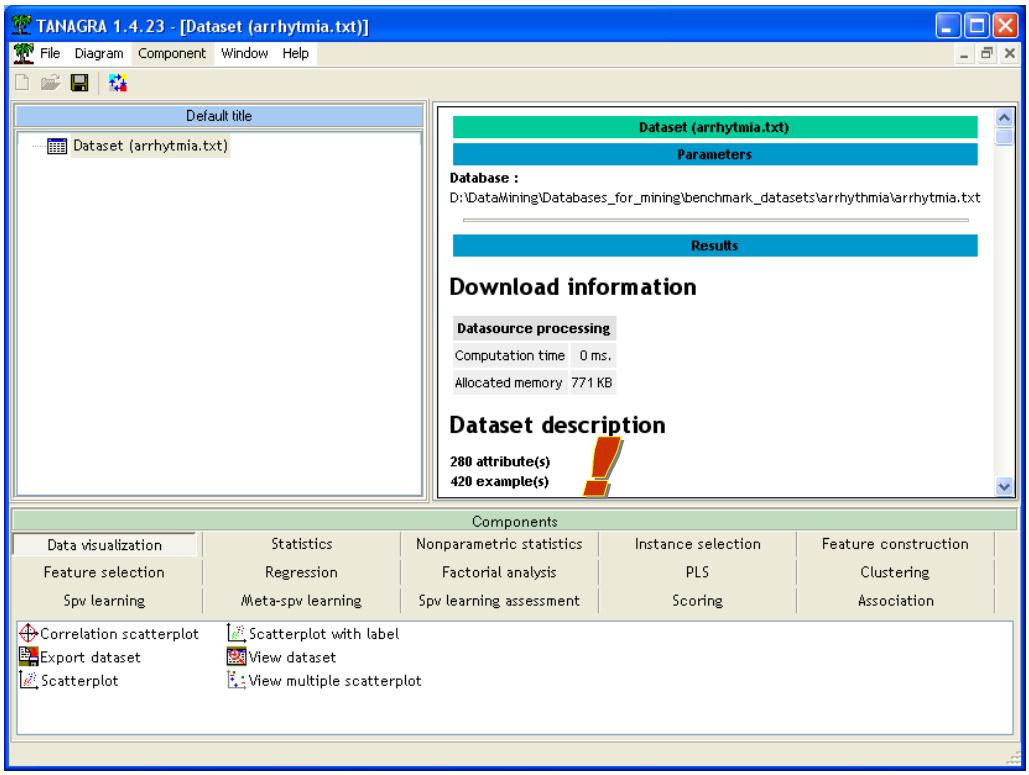
3 Comparaison des méthodes

3.1 Chargement des données et préparation des traitements

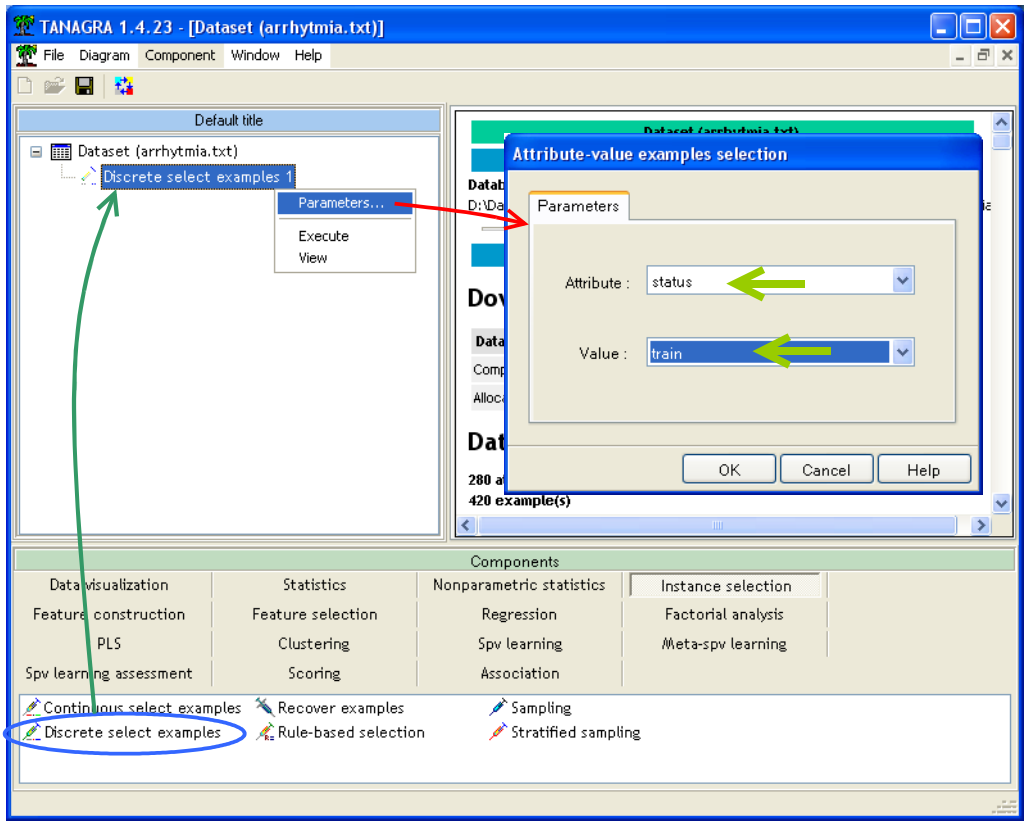
Chargement du fichier. Après avoir lancé TANAGRA (menu Démarrer de Windows), nous activons le menu FILE / OPEN dans la fenêtre principale. Une boîte de dialogue apparaît, nous choisissons le format binaire (*.BDM), nous sélectionnons le fichier ARRHYTMIA.BDM.



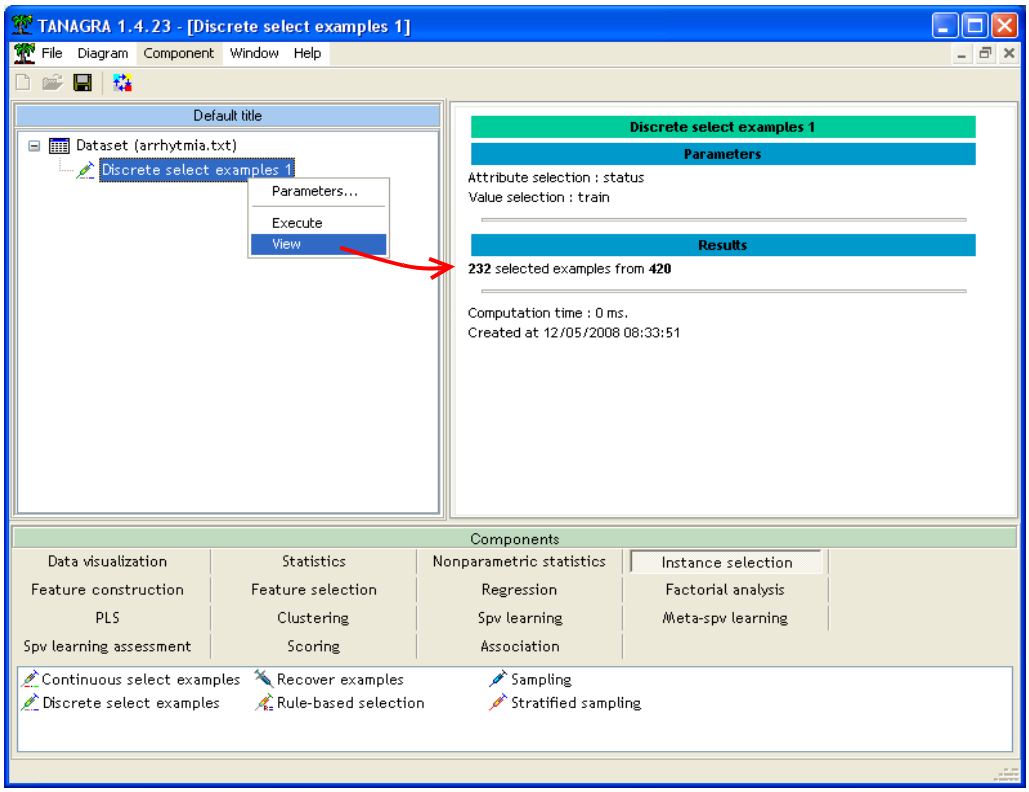
Les données constituent maintenant la racine d'un nouveau diagramme. Nous disposons bien de 420 observations et 280 attributs.



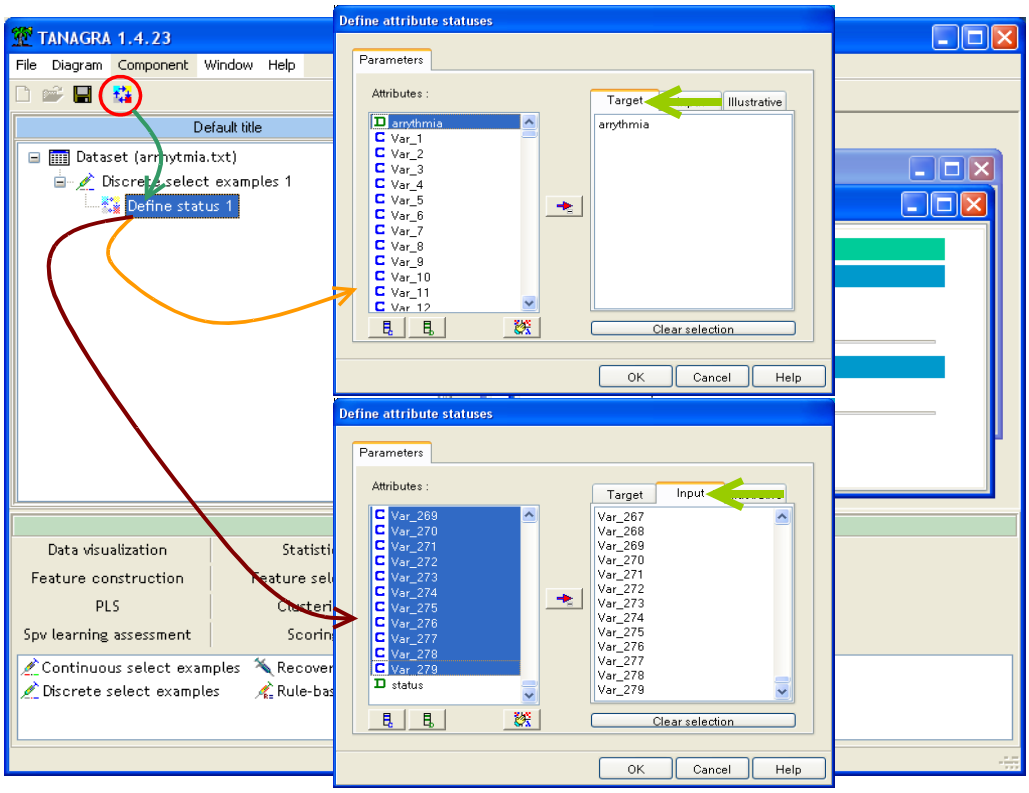
Subdivision apprentissage – test. Nous utilisons la variable STATUS pour définir les portions apprentissage et test de nos données. Nous insérons dans le diagramme le composant DISCRETE SELECT EXAMPLES (onglet INSTANCE SELECTION). Nous activons le menu contextuel PARAMETERS. Dans la boîte de paramétrage, nous plaçons STATUS en ATTRIBUTE et TRAIN en value. Nous indiquons ainsi que les observations étiquetées TRAIN vont correspondre à la partie apprentissage des données.



Après validation (OK), nous cliquons sur le menu contextuel VIEW, TANAGRA indique que 232 observations sont réservées pour la construction des modèles de prédiction.

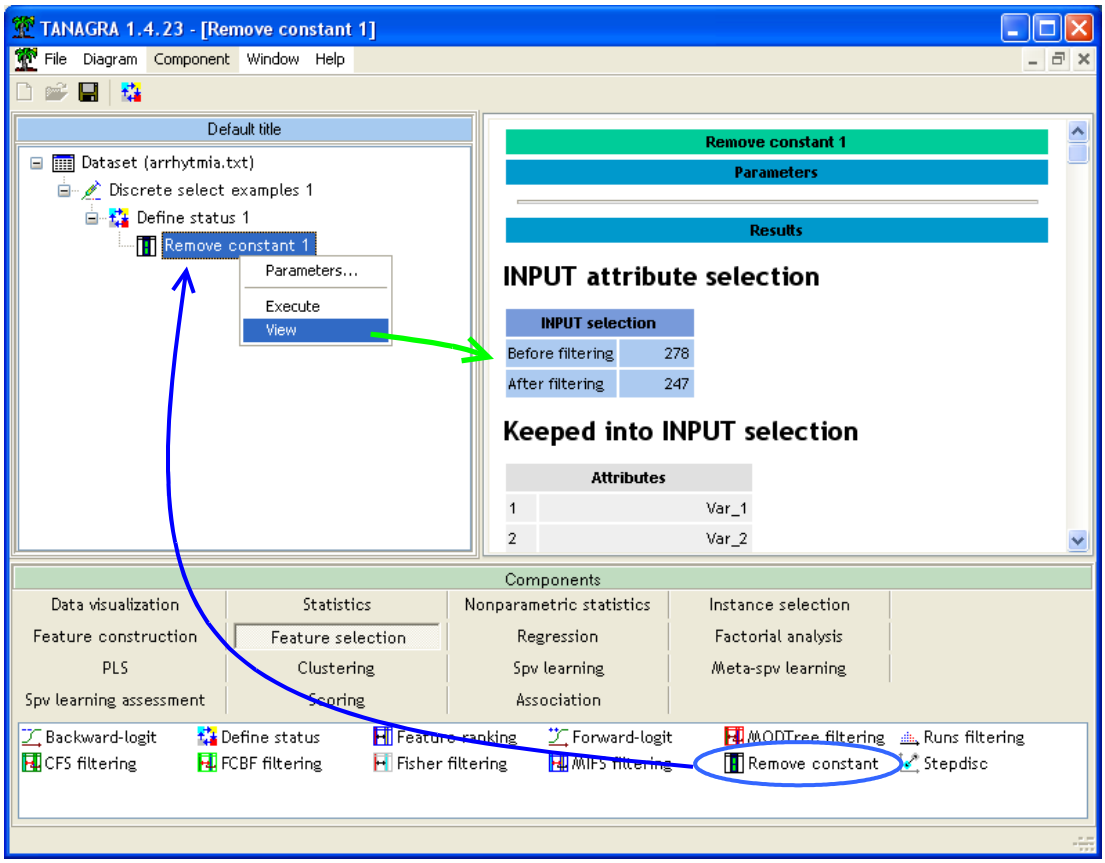


Variable cible et descripteurs. Troisième étape importante de cette phase préparatoire, nous devons indiquer la variable à prédire et les variables prédictives. Nous utilisons le composant DEFINE STATUS pour cela, accessible avec un raccourci dans la barre d'outils. Nous plaçons ARRYTHMIA en TARGET, les autres (VAR_1 à VAR_279) en INPUT.



Filtrage des descripteurs. Un préalable très important de manière générale, encore plus avec nos données, nous devons introduire un outil de filtrage des descripteurs. En effet, nous avions noté précédemment que certaines variables étaient en réalité des constantes, totalement inutiles dans la prédiction. Nous ne savons pas quelles sont ces variables. Nous introduisons donc dans le diagramme un outil qui retire automatiquement des variables INPUT celles qui correspondent à des constantes.

Nous insérons le composant REMOVE CONSTANT (onglet FEATURE SELECTION) dans notre diagramme. Nous lançons le processus de filtrage en activant le menu contextuel VIEW.

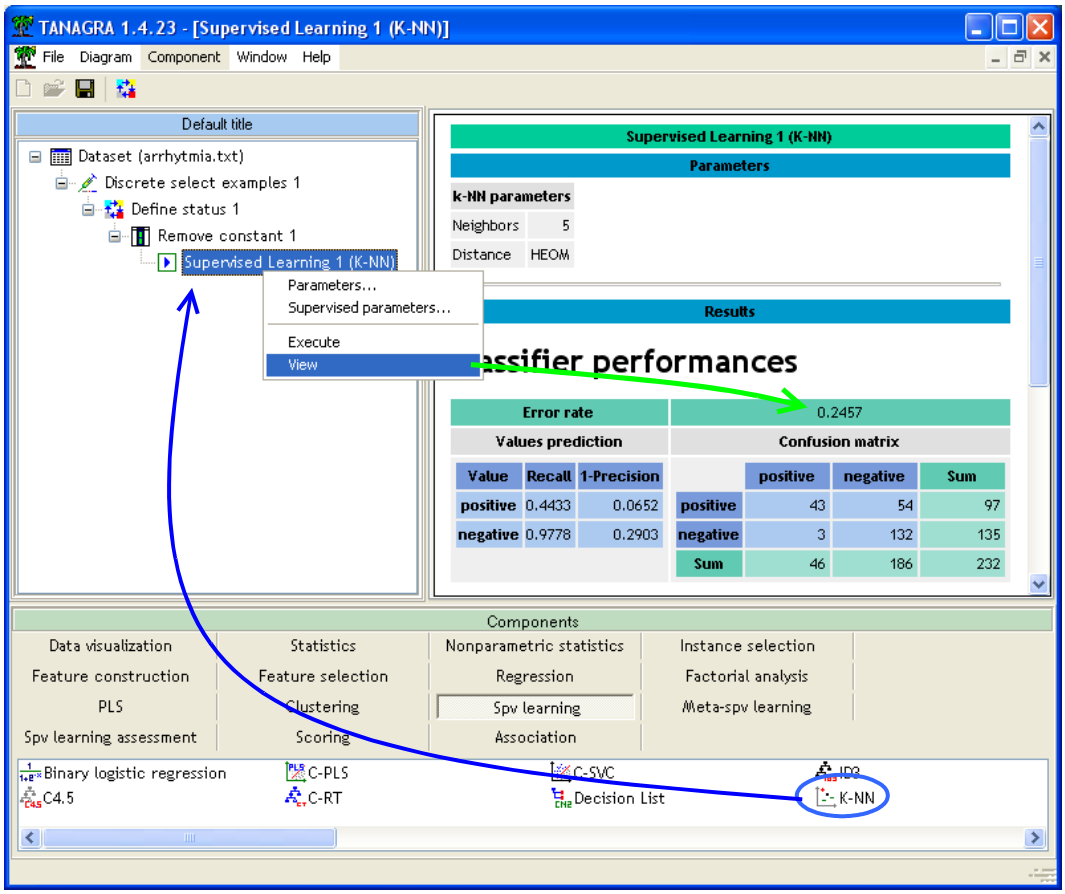


Le composant indique que parmi les 278 descripteurs proposées en entrées, 247 ont été retenus c.-à-d. la variable comporte au moins 2 valeurs différentes. En sortie du composant sont donc disponibles pour l'apprentissage : la variable TARGET qui est simplement transmise, et les 247 variables INPUT, utilisables pour la modélisation.

3.2 Méthode des plus proches voisins (K-NN)

C'est la pire des méthodes que l'on puisse employer dans ce contexte. La dimensionnalité élevée est un piège pour ce type d'approche. L'estimation locale des probabilités dans un voisinage restreint devient périlleuse. Cette méthode servira d'étalon, elle nous permettra de positionner les autres techniques. Faire pire devrait nous inquiéter.

Apprentissage. Nous insérons le composant K-NN (onglet SPV LEARNING) dans le diagramme. Nous lançons directement l'apprentissage via le menu contextuel VIEW. Bien entendu, pour la méthode des plus proches voisins, il n'y a pas vraiment de construction d'un modèle de prédiction dans cette phase, elle sert essentiellement à calculer les paramètres de normalisation des attributs.

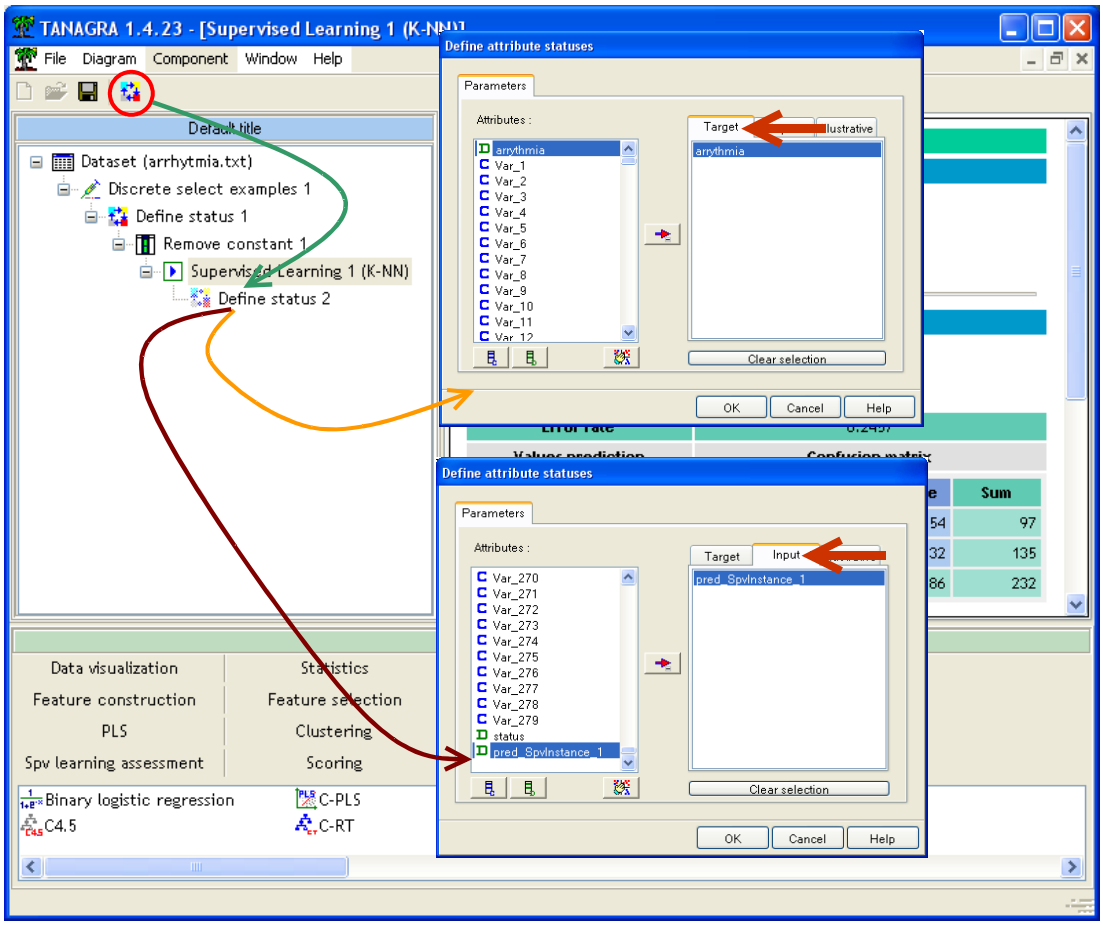


Le taux d'erreur en resubstitution est de 24.57%. On sait que ce ratio est souvent optimiste, le biais dépend de la méthode. Il nous faut un juge impartial pour évaluer les performances en prédiction, c'est le rôle de l'échantillon test.

Evaluation avec l'échantillon test. Sur la fraction des données qui ne sont pas intervenues dans la construction du classifieur, l'échantillon test, nous allons calculer la proportion des mal classés. On sait que l'estimateur est non biaisé dans ce cas.

Note : C'est la taille de l'échantillon test qui peut poser problème dans notre exemple. Etant relativement faible (188 observations), le taux d'erreur calculé sera imprécis, soumis à une certaine variabilité c.-à-d. si on change d'échantillon test, la possibilité d'obtenir un résultat sensiblement différent n'est pas négligeable.

Nous devons procéder en deux temps dans TANAGRA. Nous insérons tout d'abord le composant DEFINE STATUS dans le diagramme, toujours à l'aide du raccourci dans la barre d'outils. Nous plaçons en TARGET la variable cible ARRYTHMIA. En INPUT, nous sélectionnons la prédiction générée automatiquement par le classifieur PRED_SPVINSTANCE_1. Information très importante à noter, la prédiction a été réalisée sur la totalité des observations c.-à-d. à la fois sur les individus qui ont servi pour l'apprentissage du modèle, et sur les individus qui ont été mis de côté. Nous allons mettre à profit cette particularité pour former la matrice de confusion en test.

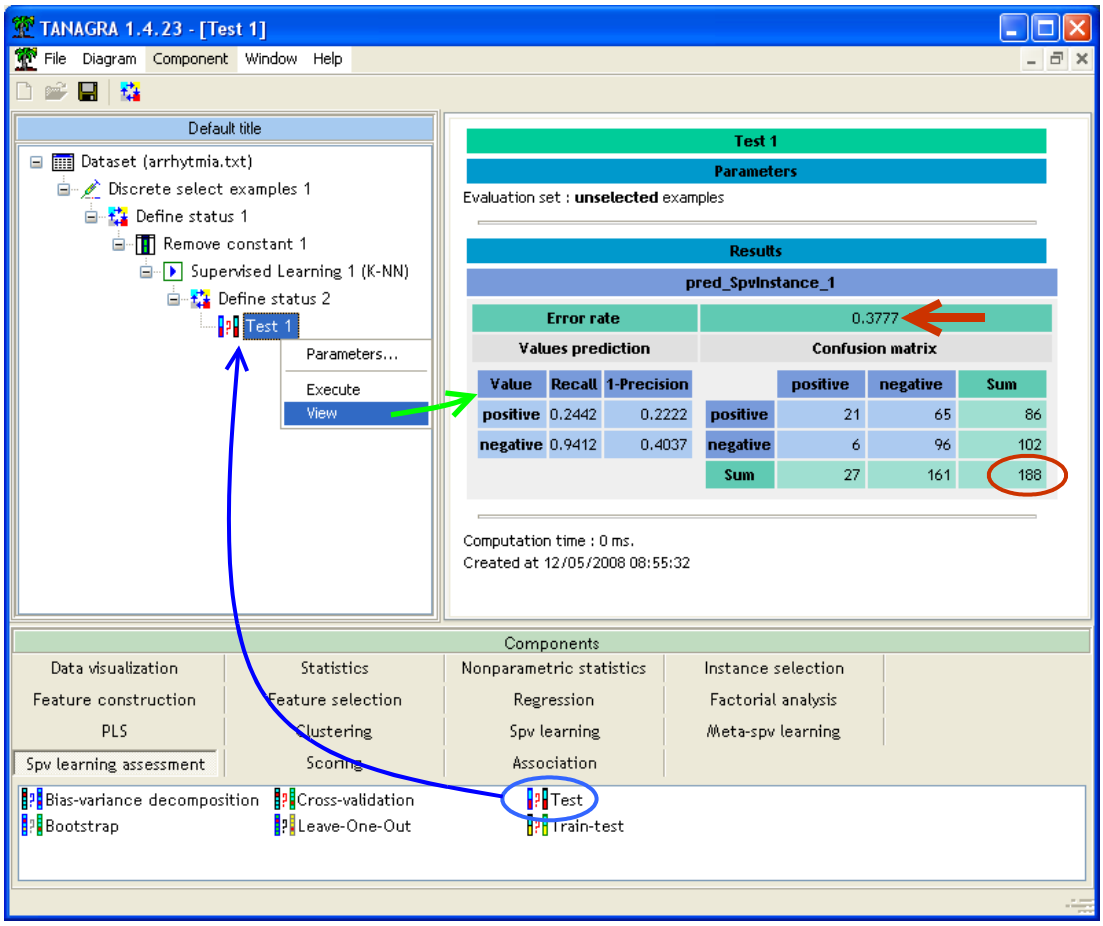


Nous insérons maintenant le composant TEST (onglet SPV LEARNING ASSESSMENT). Il est automatiquement paramétré pour réaliser les calculs sur les individus non sélectionnés, à savoir ceux que nous avons réservés pour l'évaluation du modèle. Nous activons le menu VIEW pour accéder aux résultats.

Le taux d'erreur en test, calculé sur 188 observations, est de 37.77%, bien au delà du 24.57% mesuré en resubstitution. C'est le taux de référence que nous utiliserons pour situer les autres méthodes.

La valeur 37.77% n'est pas très enthousiasmante. Le taux d'erreur du modèle par défaut (« default classifier » ou « baseline model » ou « null model »), qui consiste à prédire systématiquement ARRHYTMIA = NEGATIVE, est de $\frac{86}{188} = 45.74\%$. Ce qui nous donne un pseudo-R^2 de $1 - \frac{37.77\%}{45.74\%} = 17.43\%$[7] c.-à-d. nous réduisons de 17.43% la probabilité de mal classer du modèle par défaut.

[7] Voir « Pseudo R-Squared - Adjusted Count» parmi les différentes définitions du pseudo-R^2 disponible sur la référence suivante http://www.ats.ucla.edu/stat/mult_pkg/faq/general/Psuedo_RSquareds.htm

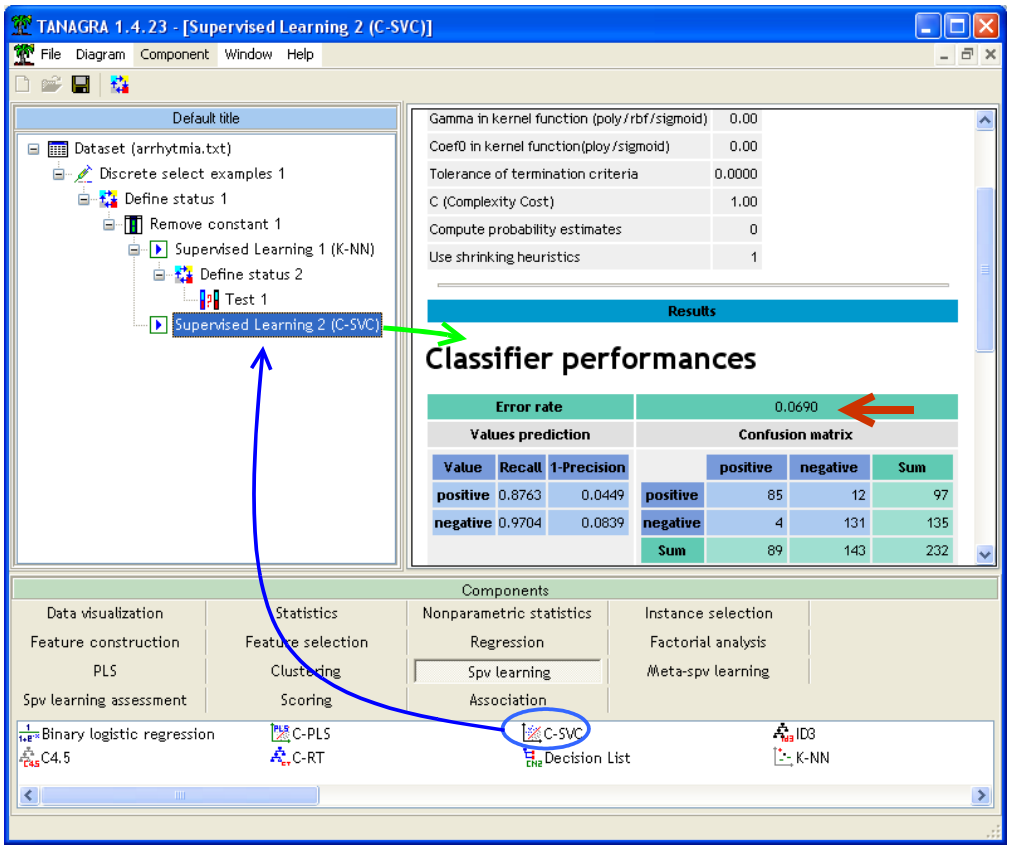


3.3 Support Vector Machine (SVM)

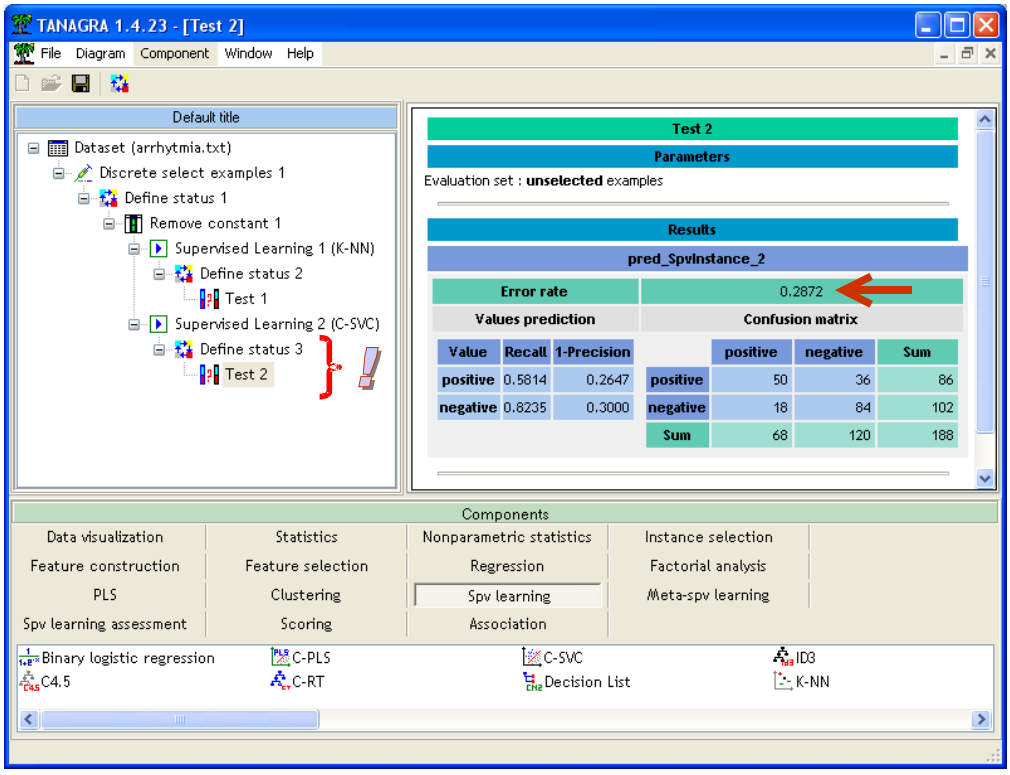
SVM Linéaire (C = 1.0). Voyons maintenant ce qu'il en est pour un SVM linéaire. Cette méthode est réputée très bien régularisée. Elle est à privilégier sur des fichiers avec relativement peu d'observations et un ratio nombre de variables – nombre d'observations particulièrement défavorable. Nous allons réitérer la même démarche apprentissage – test.

Nous insérons le composant C-SVC (onglet SPV LEARNING) dans le diagramme. Il implémente un SVM issu de la bibliothèque LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/). Par défaut, C-SVC instancie un SVM linéaire. Nous nous en contenterons dans un premier temps, sachant que nous avons la possibilité de paramétrer finement le composant. Nous actionnons le menu contextuel VIEW pour accéder aux résultats.

Le taux d'erreur en resubstitution est de 6.90%.



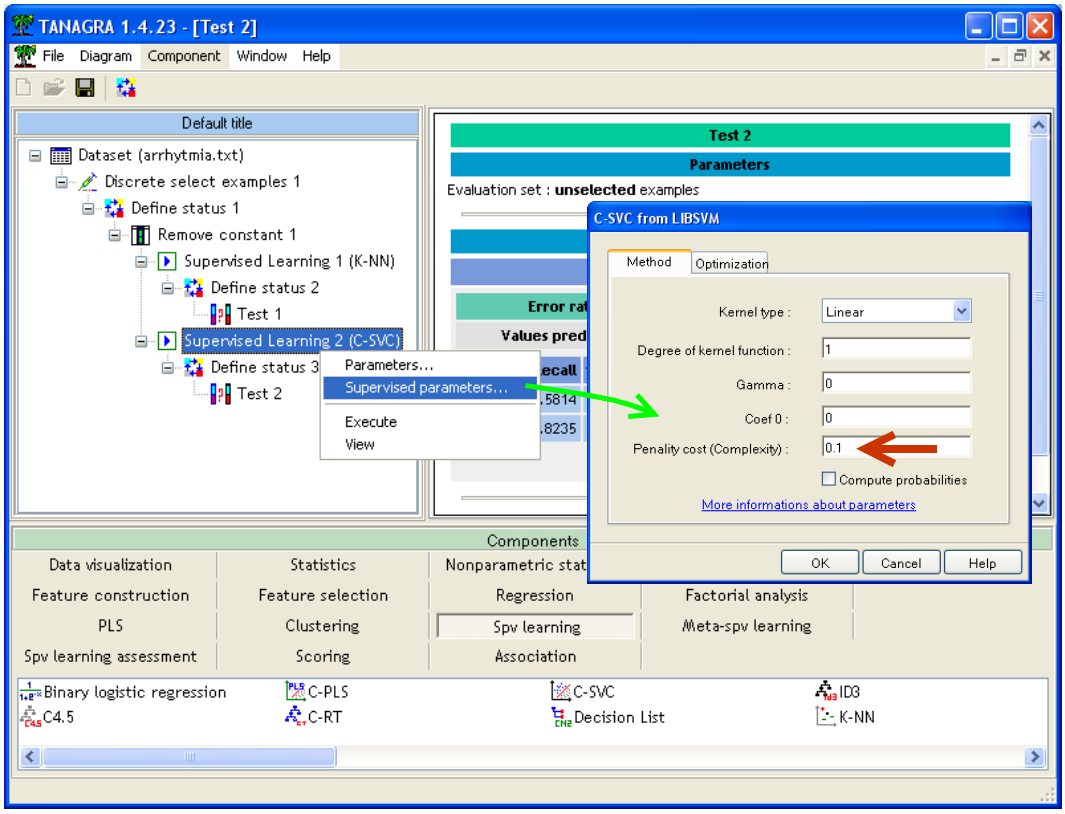
Encore une fois, il ne faut pas s'en émouvoir outre mesure, surtout dans notre contexte. Concentrons nous plutôt sur les performances en test. Nous renouvelons le même dispositif d'évaluation (DEFINE STATUS [TARGET : ARRYTHMIA, INPUT : PRED_SPVINSTANCE_2] + TEST – *On peut faire des copier/coller de branches du diagramme, voir http://tutoriels-data-mining.blogspot.com/2008/03/copier-coller-dans-le-diagramme.html*).



Le taux d'erreur en test est de 28.72%. Malgré les réserves émises sur la faible taille de l'échantillon test, l'écart avec les performances du K-NN n'est certainement pas uniquement imputable à l'échantillonnage. L'amélioration est manifeste.

SVM Linéaire fortement régularisée (C = 0.1). Toute méthode d'apprentissage dispose de sortes de « tournevis » qui permettent de guider l'apprentissage vers les solutions souhaitables compte tenu : du problème à traiter, des données manipulées et des objectifs du praticien. Souvent tout cela se résume à plus ou moins ingérer les informations portées par l'échantillon d'apprentissage. Si la méthode y est trop sensible, elle risque de sur ajuster les spécificités du fichier ; si elle y trop insensible, elle n'apprend pas les relations qui existent entre les descripteurs et la variable cible. Concernant les SVM, il y a un tournevis qui est trop souvent passé sous silence. Pourtant, il pèse significativement sur les résultats, il s'agit du « coût de régularisation[8] » (on parle aussi de « coût de complexité »). Il permet de définir la sensibilité du modèle aux individus mal classés lors de l'apprentissage : plus sa valeur est élevée, plus le classifieur s'adapte au fichier d'apprentissage, plus le risque de surajustement est élevé. Le problème bien entendu est de pouvoir fixer la bonne valeur du paramètre[9].

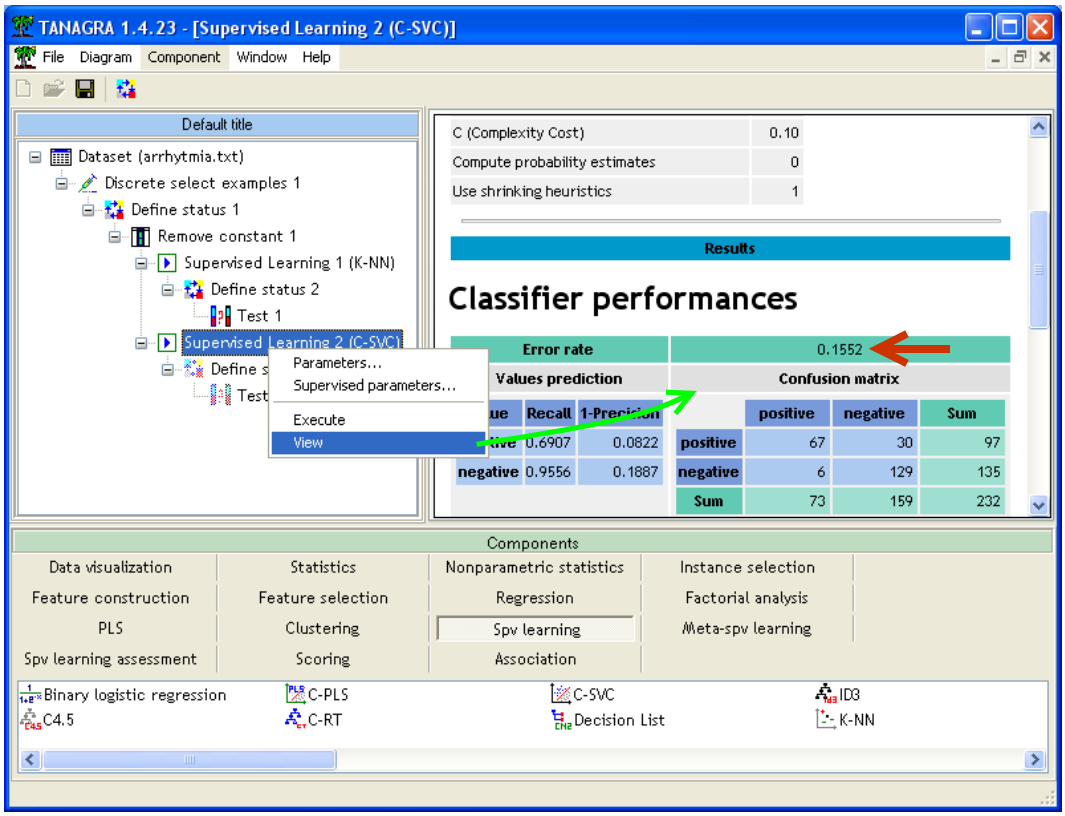
Dans notre cas, alertés par l'écart entre le taux d'erreur en resubstitution et le taux d'erreur en test, signe d'une tendance au sur apprentissage pour un modèle linéaire, nous souhaitons « raidir » l'apprentissage en diminuant la valeur du paramètre de coût. Nous actionnons le menu contextuel SUPERVISED PARAMETERS du composant C-SVC, nous remplaçons la valeur C = 1.0 par C = 0.1 (Important : attention au point décimal selon votre version de Windows).



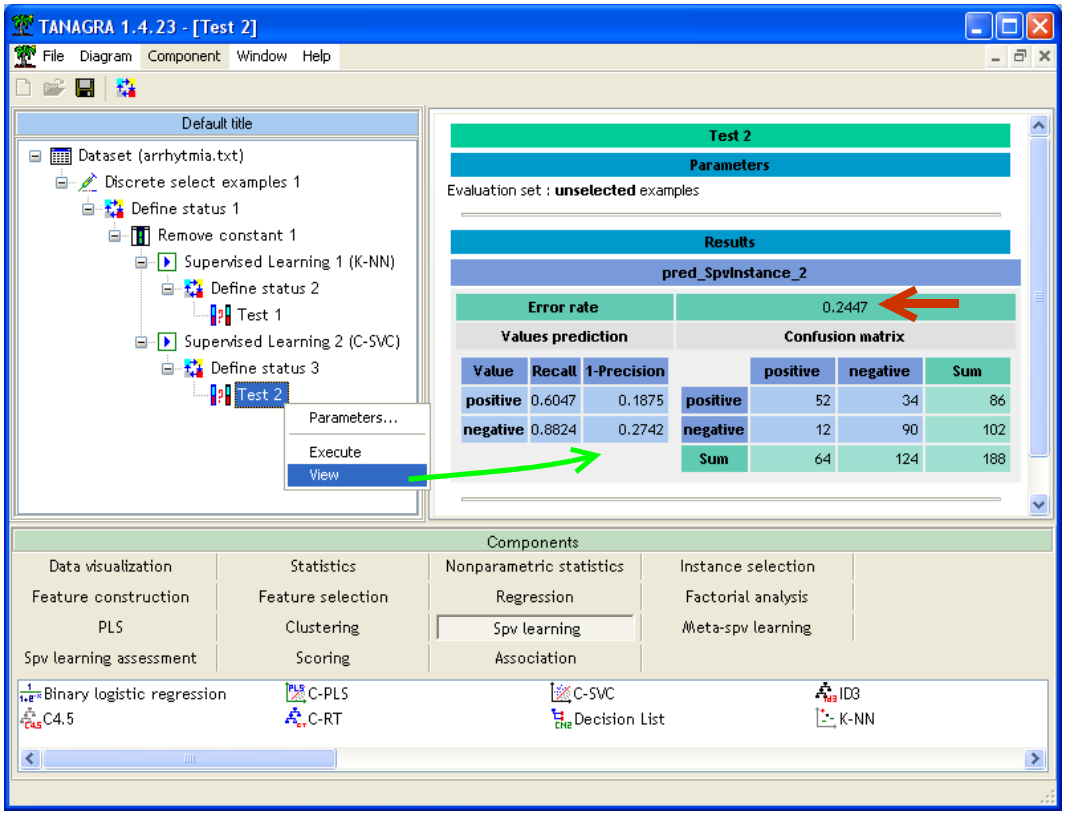
[8] http://www.research.ibm.com/dar/papers/pdf/svmpath_jmlr.pdf

[9] Souvent, on sait dans quel sens tourner le tournevis pour aiguiller la modélisation. En revanche, jusqu'à quel point il faut aller pour obtenir une solution optimale… on passe par le tâtonnement, avec des essais répétés apprentissage – test.

Nous validons, puis nous cliquons sur le menu VIEW pour obtenir les nouveaux résultats. Le taux d'erreur est de 15.52%. Le modèle colle moins aux données.



Pour avoir une évaluation sur l'échantillon test, nous actionnons le menu VIEW du composant TEST 2 dans le diagramme.



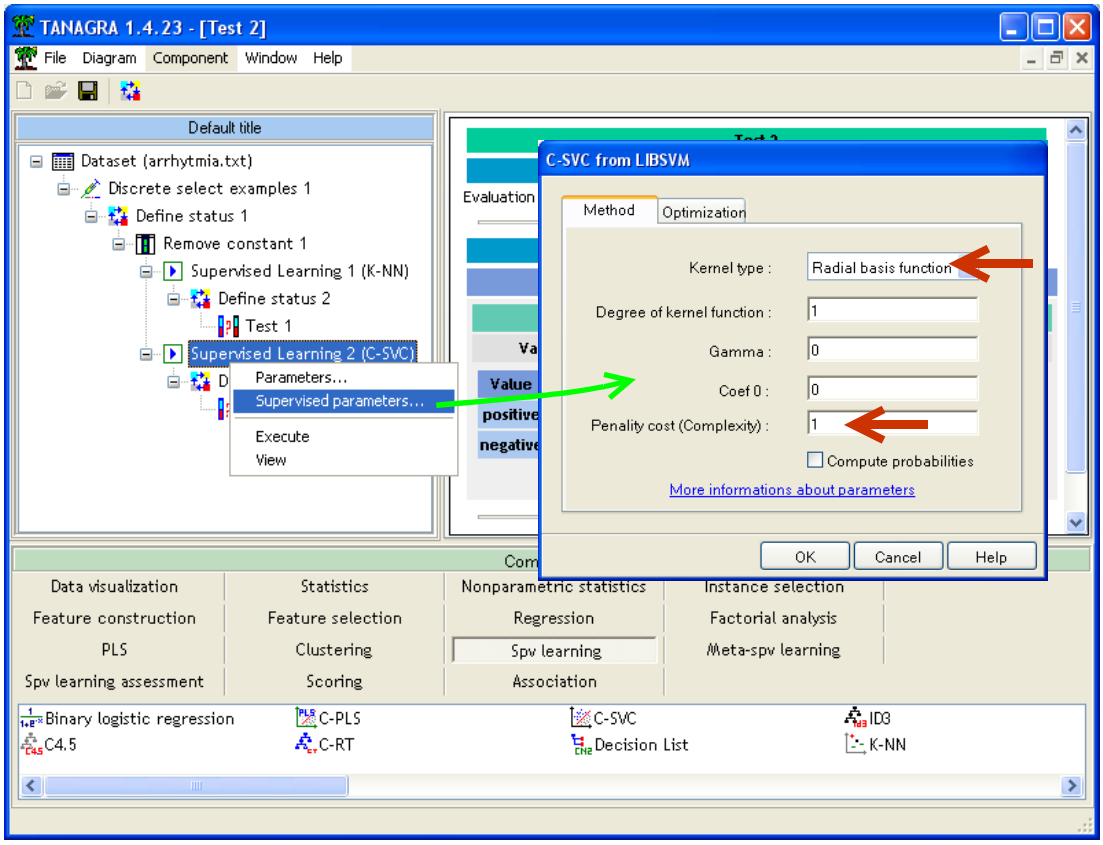
Le taux d'erreur en test est de 24.47%. Le nouveau modèle est légèrement meilleur que précédemment (8 individus supplémentaires ont été bien classés sur 188).

Remarque : Il est malheureux que ce paramètre soit souvent passé sous silence dans les publications qui recourent aux SVM. Comme on vient de le voir, son influence est réelle. La principale difficulté est de savoir comment la manipuler, ce n'est pas très simple. Dans notre contexte, le ratio nombre de variables – nombre d'observations défavorable, l'écart taux d'erreur en resubstitution – taux d'erreur en test, laissent à penser que réduire la sensibilité de la modélisation aux données d'apprentissage est la bonne piste. La situation peut être plus contrastée dans d'autres contextes.

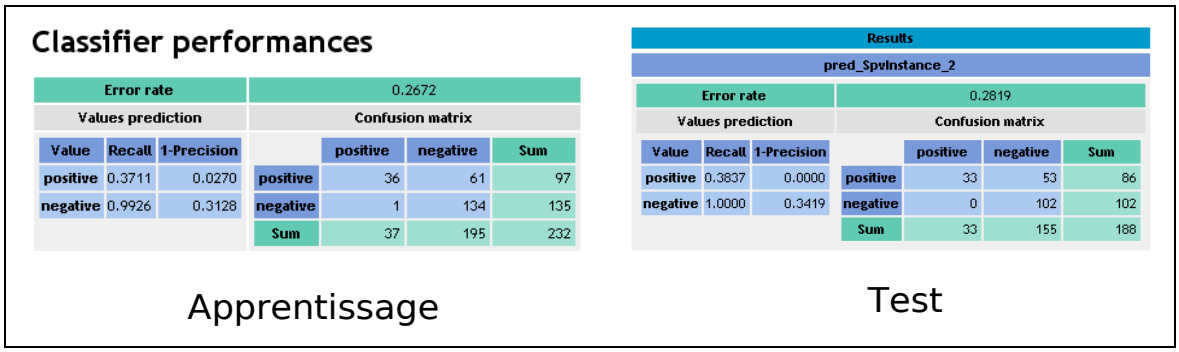
Pour confirmer notre idée, nous avons tenté d'augmenter le paramètre de coût pour produire un classifieur qui s'adapte plus aux données d'apprentissage (C = 10), le taux d'erreur en resubstitution passe à 0.43% dans ce cas. Mais, avant de nous laisser aller à une fausse joie, nous constatons que le taux d'erreur en test est 39.36%, nettement dégradé par rapport au SVM précédent, signe que nous sommes manifestement dans une situation de surajustement.

SVM-RBF (C = 1.0). Le choix du noyau est le second paramètre qui oriente significativement le comportement des SVM. Rappelons l'idée très brièvement : nous avons la possibilité, sans avoir à le calculer explicitement, de construire la frontière de discrimination dans un espace différent. Une frontière linéaire dans cet espace correspond à une frontière non linéaire dans l'espace originel.

Il est tentant de penser qu'un modèle non linéaire est forcément plus performant qu'un modèle linéaire. Mais ce n'est pas (toujours) vrai. Certes, on réduit ainsi le biais, mais ce n'est pas sans conséquences sur la variance du classifieur. Dans notre cas, le mieux est de voir ce qu'il en est en testant la configuration. Nous revenons sur la boîte de paramétrage de C-SVC (menu SUPERVISED PARAMETERS), nous spécifions KERNEL TYPE = RADIAL BASIS FUNCTION (noyau RBF), puis nous ramenons le coût de pénalisation à sa valeur initiale PENALITY COST = 1.0.



Le taux d'erreur en apprentissage est de 26.72%, en test il passe à 28.19%.



Classifier performances

Error rate	0.2672				
Values prediction			Confusion matrix		
Value	Recall	1-Precision		positive	negative	Sum
positive	0.3711	0.0270	positive	36	61	97
negative	0.9926	0.3128	negative	1	134	135
			Sum	37	195	232

Apprentissage

Results — pred_SpvInstance_2

Error rate	0.2819				
Values prediction			Confusion matrix		
Value	Recall	1-Precision		positive	negative	Sum
positive	0.3837	0.0000	positive	33	53	86
negative	1.0000	0.3419	negative	0	102	102
			Sum	33	155	188

Test

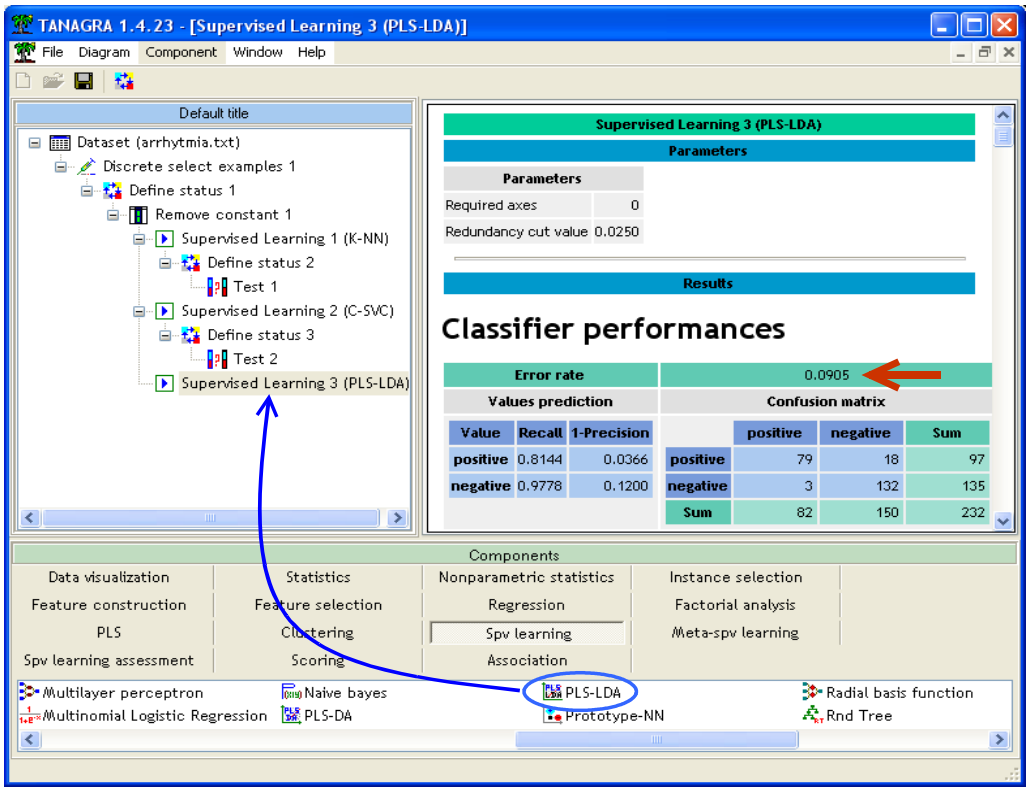
Manifestement, c'est bien la maîtrise de la variance qui est primordiale dans le contexte de notre étude. Il en est souvent ainsi d'ailleurs dès que nous traitons des données avec un ratio nombre de variables – nombre d'observations défavorable.

3.4 Régression PLS (PLS-LDA)

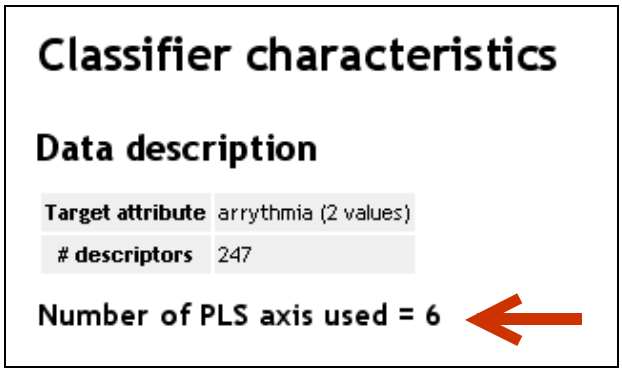
PLS-LDA (6 axes). La méthode PLS-LDA est présentée dans un de nos précédents didacticiels (http://tutoriels-data-mining.blogspot.com/2008/05/analyse-discriminante-pls.html). Il s'agit d'un apprentissage en deux phases : (1) une Régression PLS sur les indicatrices de la variable cible ; (2) une analyse discriminante linéaire sur les facteurs de la régression PLS.

Le contrôle de la variance, la dépendance aux données d'apprentissage, repose sur le choix du nombre de facteurs que nous présentons à l'analyse discriminante linéaire dans la deuxième phase. TANAGRA intègre un mécanisme basé sur l'examen de la redondance. Si la variabilité expliquée par un facteur supplémentaire est inférieure à un seuil, le processus est stoppé.

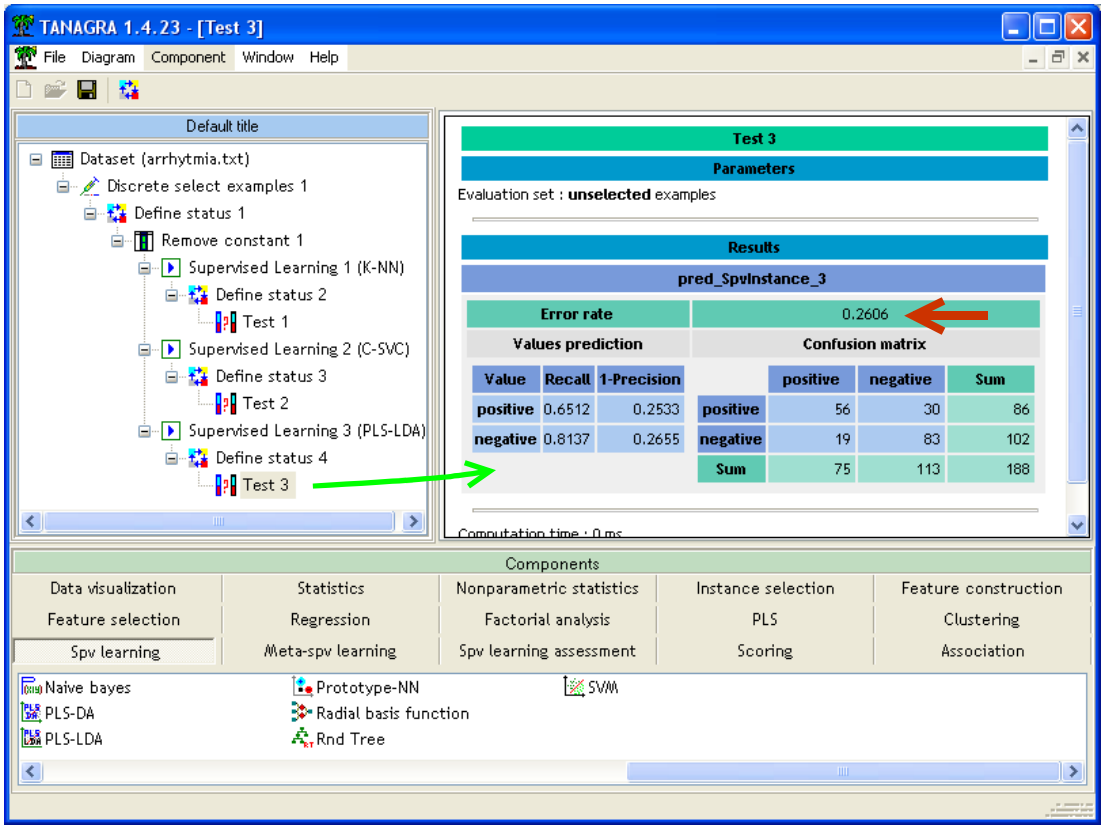
Nous plaçons le composant PLS-LDA (onglet SPV LEARNING) dans le diagramme. Nous activons directement le menu VIEW, le taux d'erreur en resubstitution est 9.05%.



La méthode a automatiquement sélectionné 6 axes.



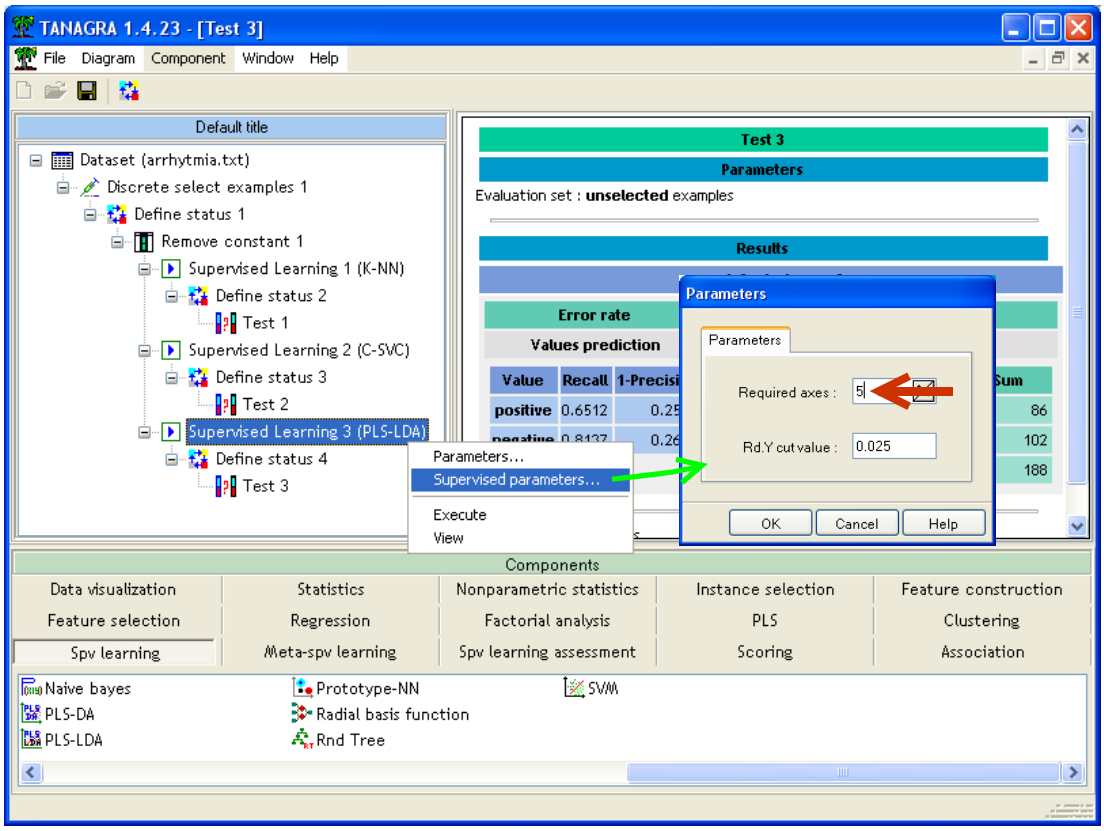
Nous complétons le diagramme pour l'évaluation en test. Le taux d'erreur est dans ce cas 26.06%.



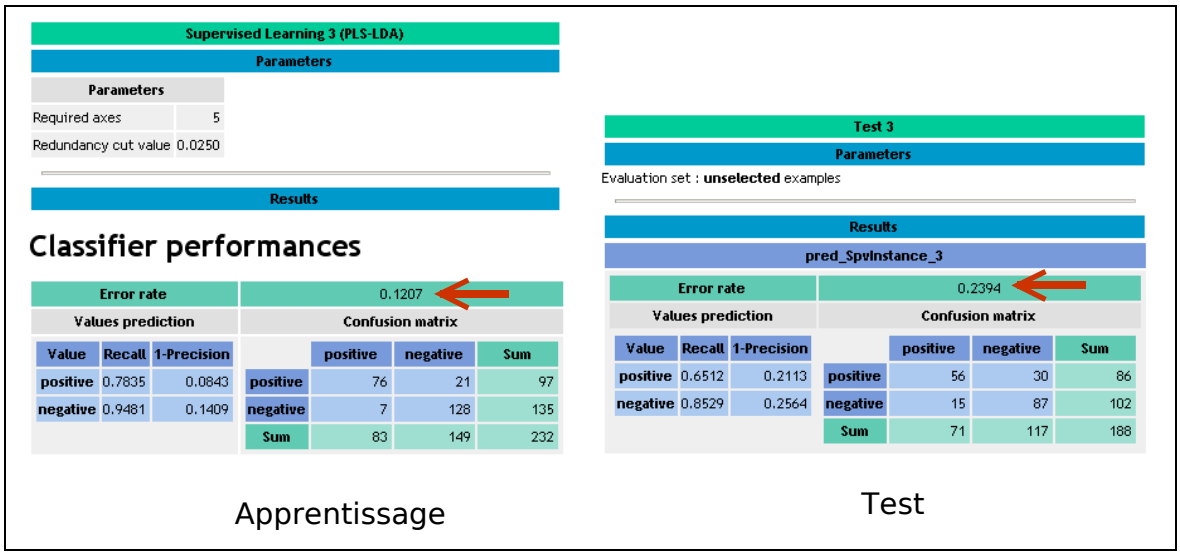
Remarque : La méthode tient la route face aux SVM linéaires étudiés ci-dessus. Dans la grande majorité des cas, dans un espace très bruité avec des descripteurs redondants, la régression PLS et les SVM linéaires présentent souvent des performances comparables. Les propriétés de régularisation de la Régression PLS sont remarquables.

PLS-LDA mieux régularisée (5 axes). Ici aussi, nous avons la possibilité de maîtriser la variance du modèle, en fixant explicitement le nombre d'axes factoriels, par tâtonnement essentiellement. Compte tenu des commentaires émis ci-dessus, nous décidons de « raidir » l'apprentissage, cela passe par une réduction du nombre d'axes. Pour ce faire, nous actionnons le menu SUPERVISED PARAMETERS du composant PLS-LDA, nous plaçons la valeur REQUIRED AXES[10] = 5.

[10] La valeur précédente « REQUIRED AXES = 0 » indiquait que le nombre de facteurs était déterminé automatiquement.



Le taux d'erreur en apprentissage passe à 12.7%, le taux d'erreur en test est maintenant égal à 23.94%, meilleur que n'importe lequel des SVM étudiés plus haut (*1 individu bien classé supplémentaire par rapport au « SVM Linéaire - C = 0.1 », ce n'est pas la gloire non plus*).



Remarque : Compromis et tâtonnements sont également de mise ici. Si on serre trop la vis (ex. 2 facteurs seulement), on est moins sensible aux données d'apprentissage (taux d'erreur en resubstitution = 19.4%), on n'en extrait pas suffisamment d'informations pour le classement (taux d'erreur en test = 31.38%).

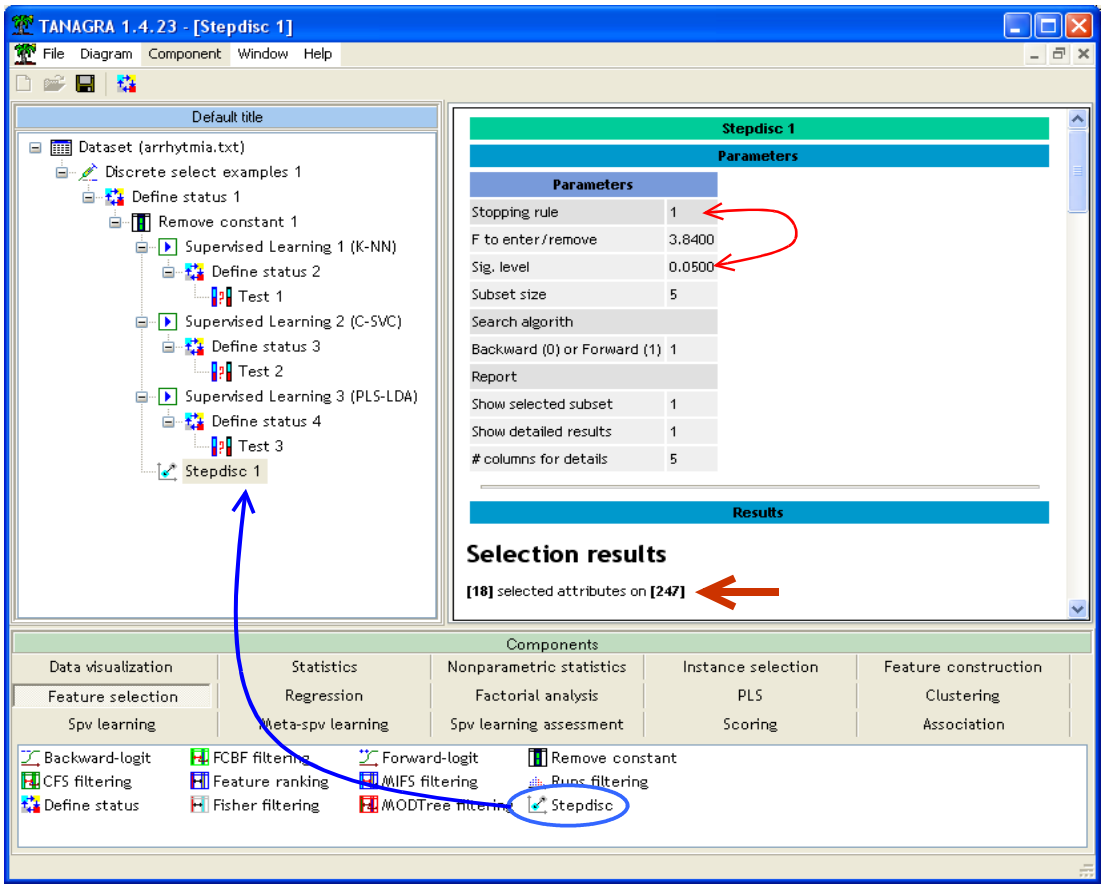
3.5 Analyse discriminante linéaire (LDA)

Analyse discriminante linéaire (18 variables). Lancer directement une analyse discriminante sur ce type de problème est une hérésie. Procéder à l'inversion d'une matrice 247 x 247 n'est pas une opération simple. De plus, il est fort à parier que l'apprentissage sera peu efficace. Les effectifs étant faibles au regard de la dimensionnalité, l'estimation de la matrice de variance co-variance sera très instable. De quel tournevis dispose-t-on alors pour mieux contrôler l'apprentissage ?

Il existe des techniques de régularisation de l'analyse discriminante, on pense notamment aux techniques « ridge », ou encore à l'analyse discriminante sur facteurs (la méthode PLS-DA en est une d'ailleurs). Mais dans ce didacticiel, nous allons nous tourner vers une approche très simple : la sélection de variables. Moins nous sélectionnons de variables, plus le classifieur sera stable, mais moins il tirera profit des informations proposées par le fichier d'apprentissage. A l'inverse, trop de variables produit un modèle instable, fortement dépendant des données d'apprentissage.

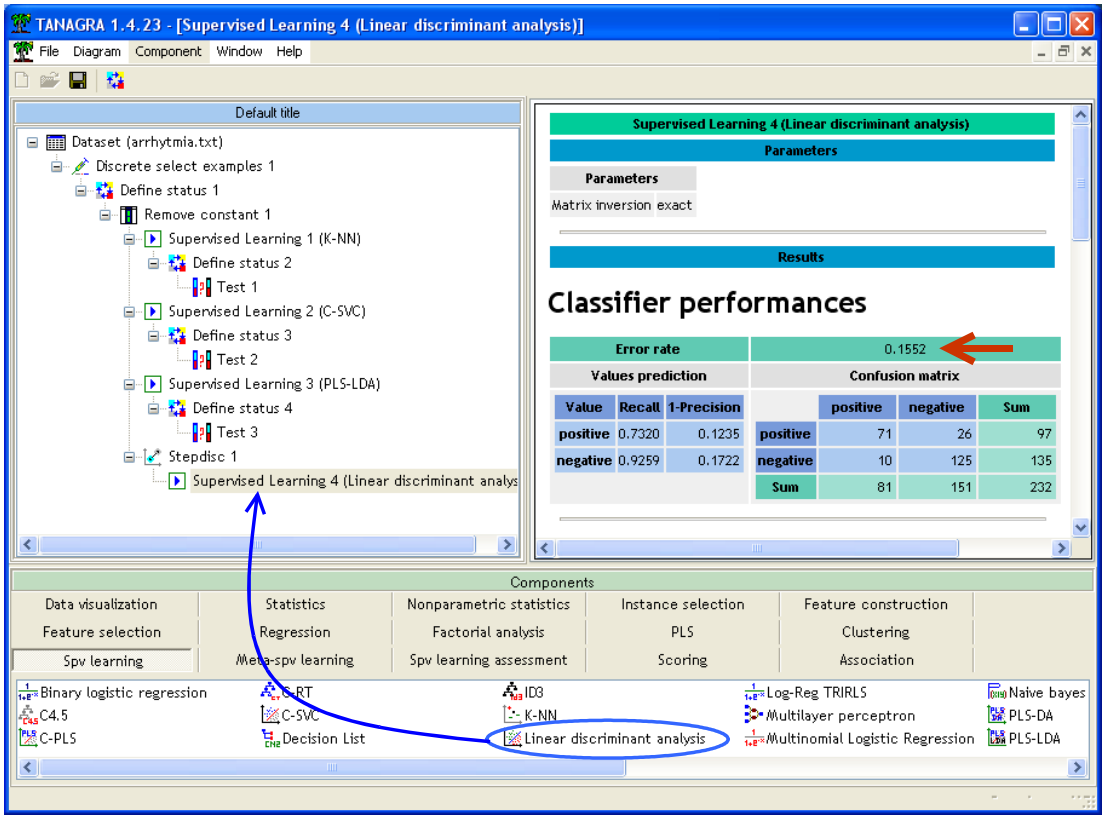
Par rapport aux autres techniques, la stratégie de sélection est naturelle concernant la LDA. En effet, l'algorithme de réduction, la méthode STEPDISC, est en accord avec le critère sur lequel s'appuie la LDA pour évaluer la séparabilité des groupes, en l'occurrence le LAMBDA de WILKS[11].

Nous insérons le composant STEPDISC (onglet FEATURE SELECTION) dans le diagramme. Nous laissons les paramètres par défaut. Il procédera à une sélection FORWARD (ajout au fur et à mesure des variables), le processus est stoppé lorsque la probabilité critique (p-value) associée à la variable additionnelle est inférieure au seuil de signification que l'on s'est choisi (0.05 par défaut). Nous cliquons sur VIEW pour accéder aux résultats.

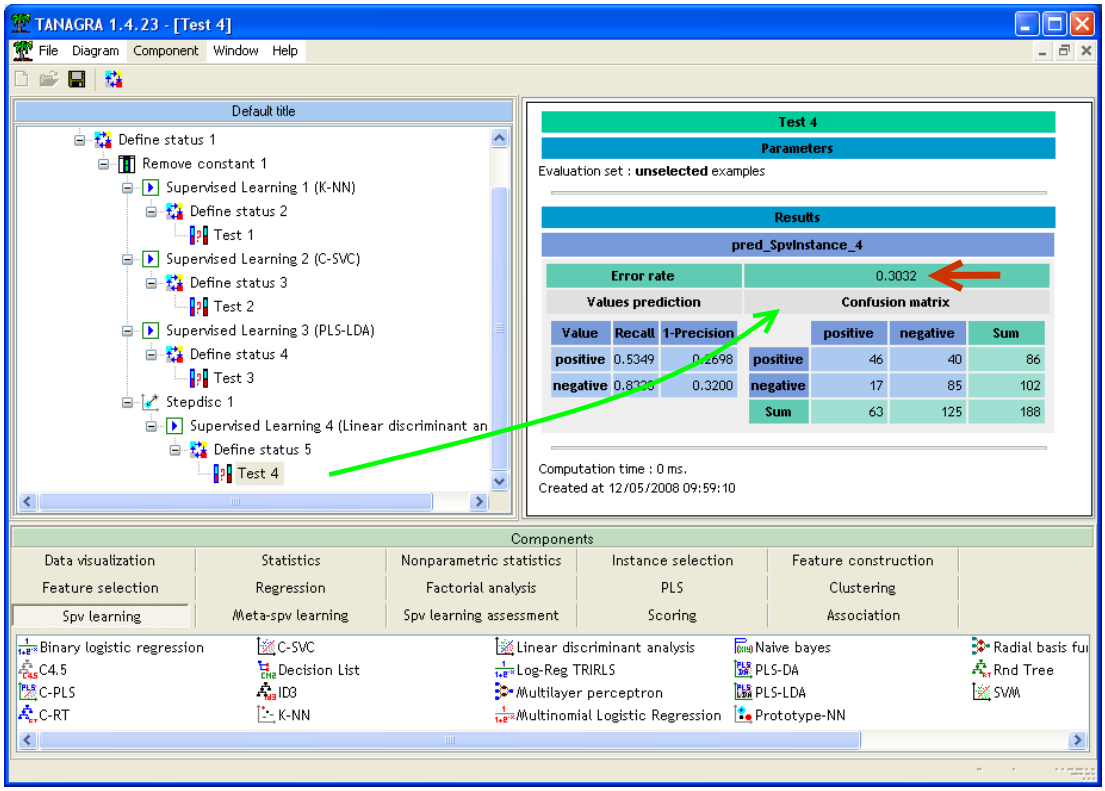


[11] http://eric.univ-lyon2.fr/~ricco/cours/slides/analyse_discriminante.pdf ; page 9.

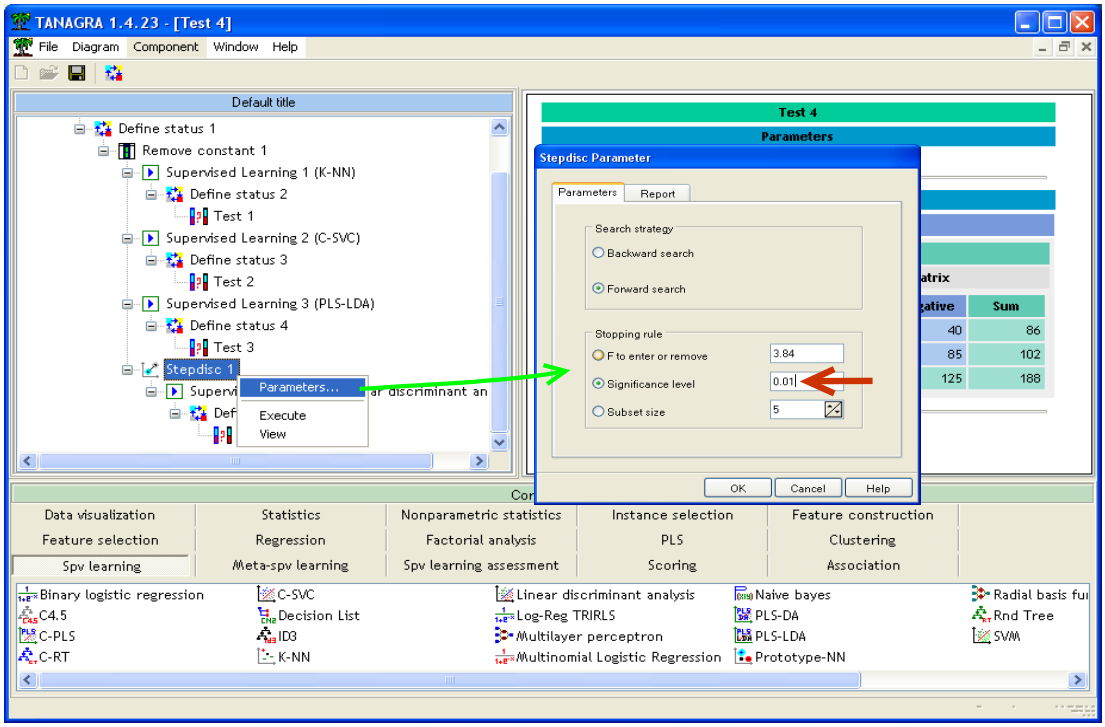
18 variables ont été automatiquement sélectionnées. La liste est disponible. Nous pouvons maintenant insérer le composant LINEAR DISCRIMINANT ANALYSIS (onglet SPV LEARNING). Le taux d'erreur en apprentissage est de 15.52%.



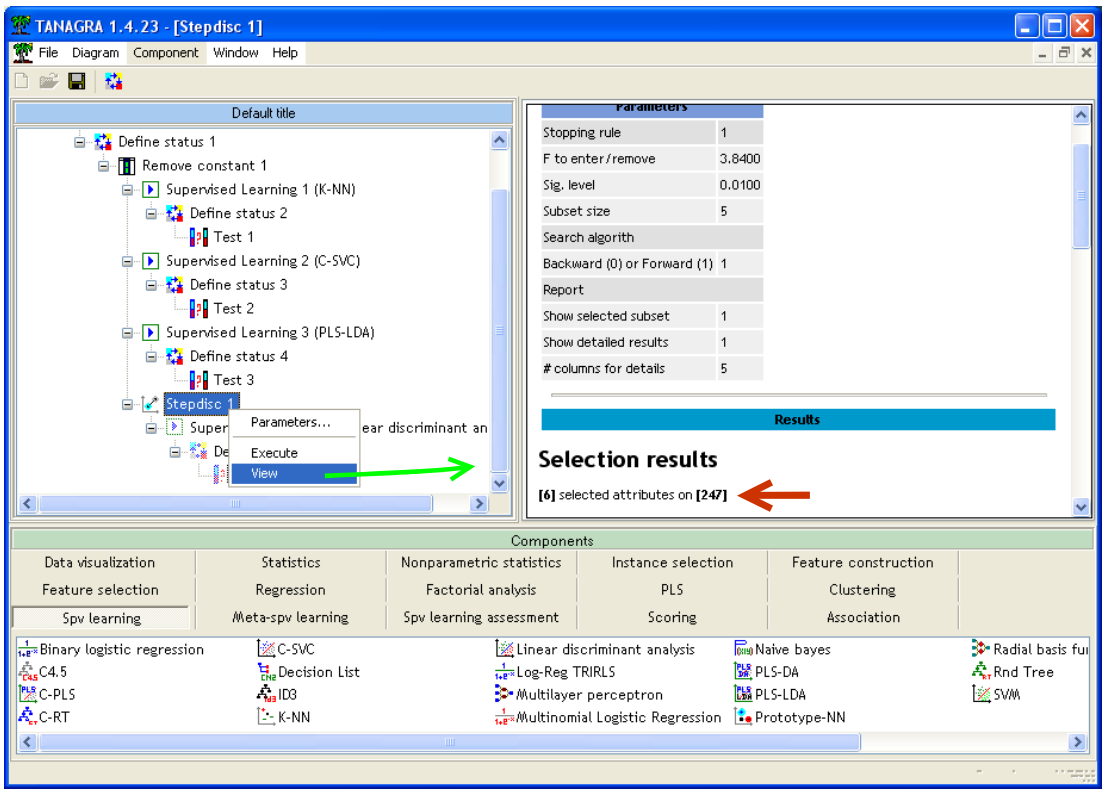
En test, après avoir complété le diagramme, il passe à 30.32%, très loin des méthodes SVM et PLS-DA, proche plutôt de la méthode des K-NN. Il y a un problème de sur dimensionnalité.



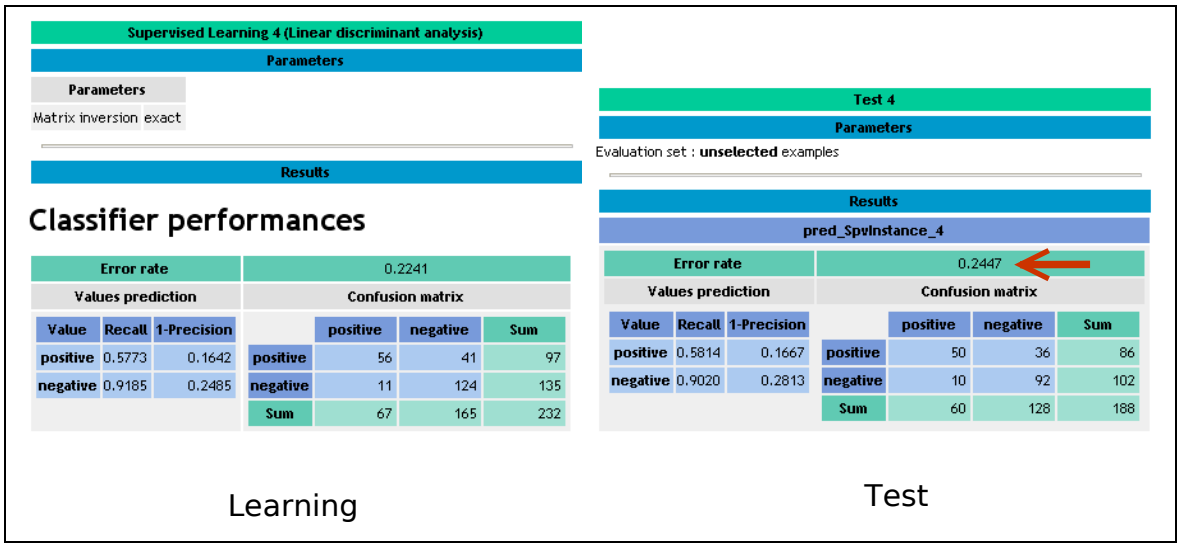
Analyse discriminante linéaire (6 variables). Manifestement, nous n'avons pas suffisamment serré la vis. Il y a sur ajustement sur les données. il faut réduire le nombre de variables. Nous revenons sur le composant STEPDISC, nous activons le menu PARAMETERS. Nous modifions le seuil de signification, nous le passons à 0.01.



Nous lançons les calculs (menu VIEW), 6 variables sont sélectionnées maintenant.



Pour ce qui est du classifieur, le taux d'erreur en resubstitution est de 22.41%, en test il est à 24.47%, un niveau tout à fait comparable aux SVM linéaires et PLS-LDA ci-dessus.



On se demande parfois pourquoi on se complique la vie avec des méthodes sophistiquées lorsqu'une technique aussi simple et répandue que l'analyse discriminante linéaire, bien paramétrée, peut produire des résultats aussi bons.

Remarque : Ici également, le tâtonnement est de mise. Trop ou trop peu de variables dégradent fortement les performances de l'analyse discriminante. La plage des bonnes valeurs est assez mince dans notre exemple.

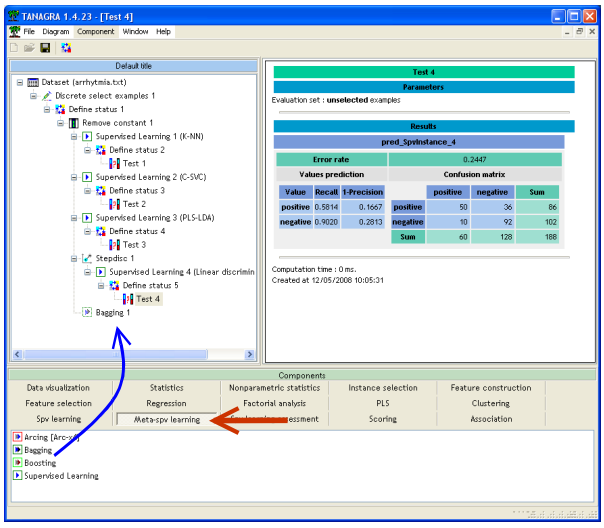
3.6 Random Forest

Random Forest. C'est une méthode due à Breiman (2001)[12]. L'idée repose sur la combinaison BAGGING + ARBRE de Décision. L'agrégation de plusieurs arbres construits sur des réplications (tirage avec remise) de l'échantillon d'apprentissage produit un classifieur efficace, plus efficace en tous les cas que l'utilisation d'un seul arbre. Cette technique est très performante. Elle repose, tout comme les SVM, sur la maximisation de la marge, mais dans une métrique différente. Marge ici s'entend : écart entre le mode de la probabilité d'affectation et la probabilité suivante.

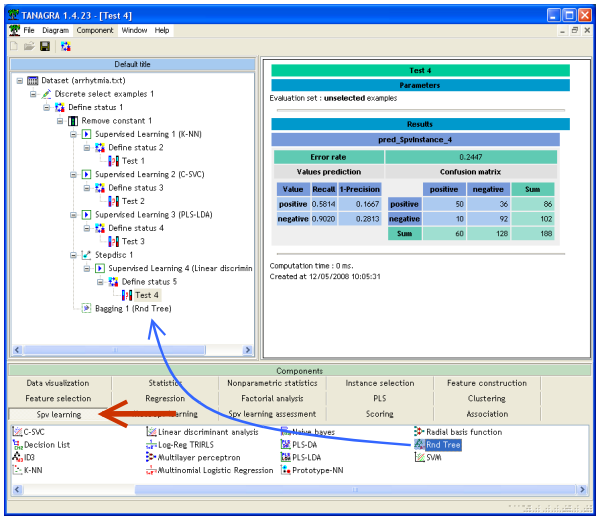
Pour insérer la méthode RANDOM FOREST dans le diagramme[13], nous devons procéder en deux temps : (1) tout d'abord, introduire le composant BAGGING (onglet META-SPV LEARNING) ; (2) puis, y intégrer la méthode spécifique d'induction d'arbre RND TREE (onglet SPV LEARNING).

[12] http://www.stat.berkeley.edu/~breiman/RandomForests/ ; http://en.wikipedia.org/wiki/Random_forest

[13] Voir parmi les didacticiels TANAGRA : http://tutoriels-data-mining.blogspot.com/2008/03/random-forests.html

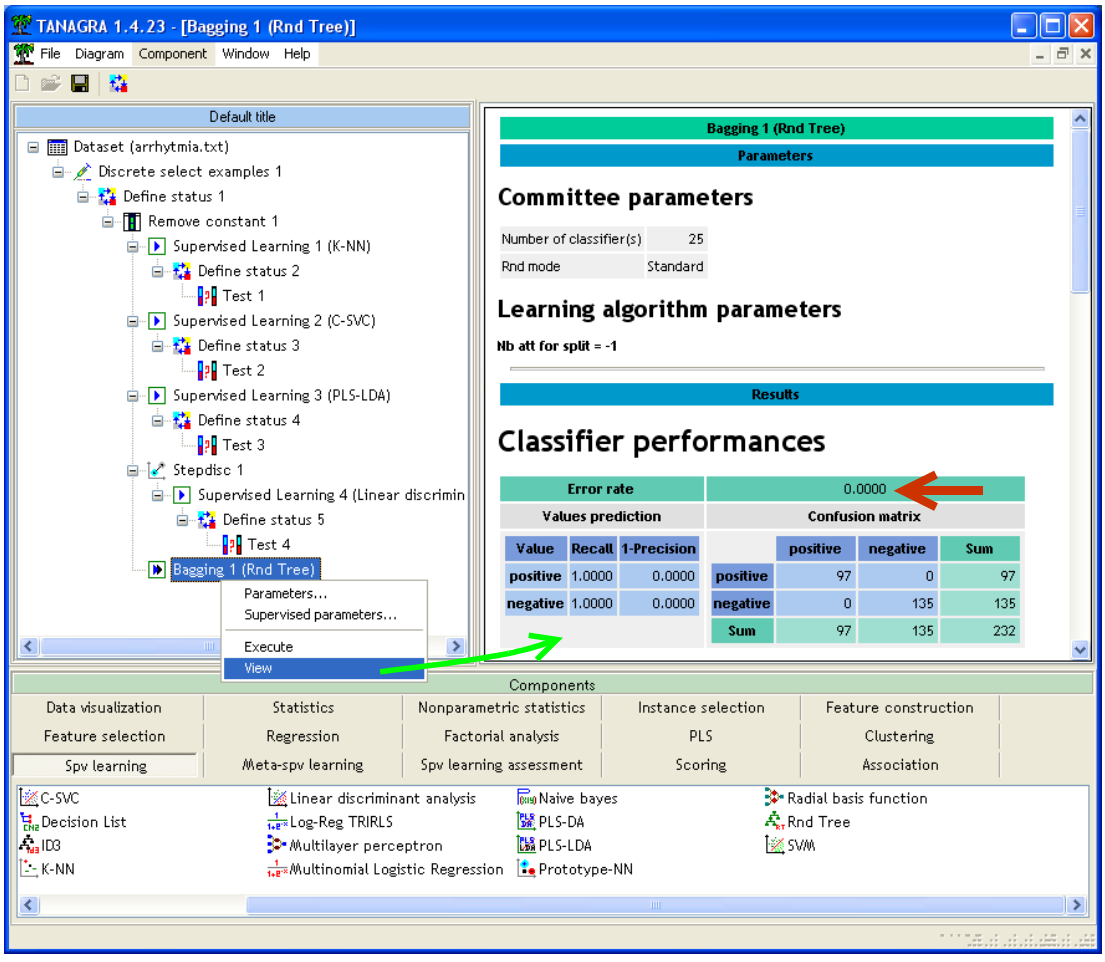


(1)

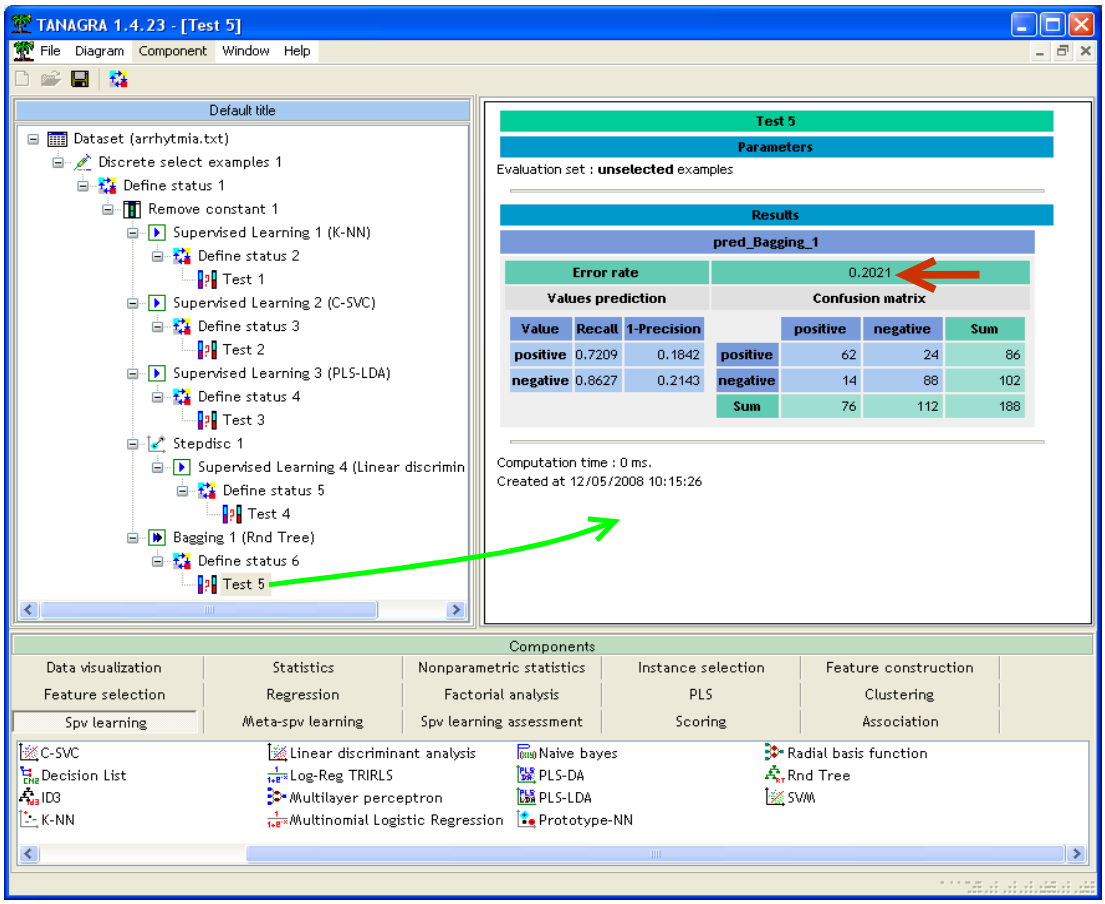


(2)

Nous lançons les calculs (menu VIEW). Le taux d'erreur en apprentissage est de 0%, c'est un résultat normal pour ce type de méthode.



Voyons ce qu'il en est sur la partie test.



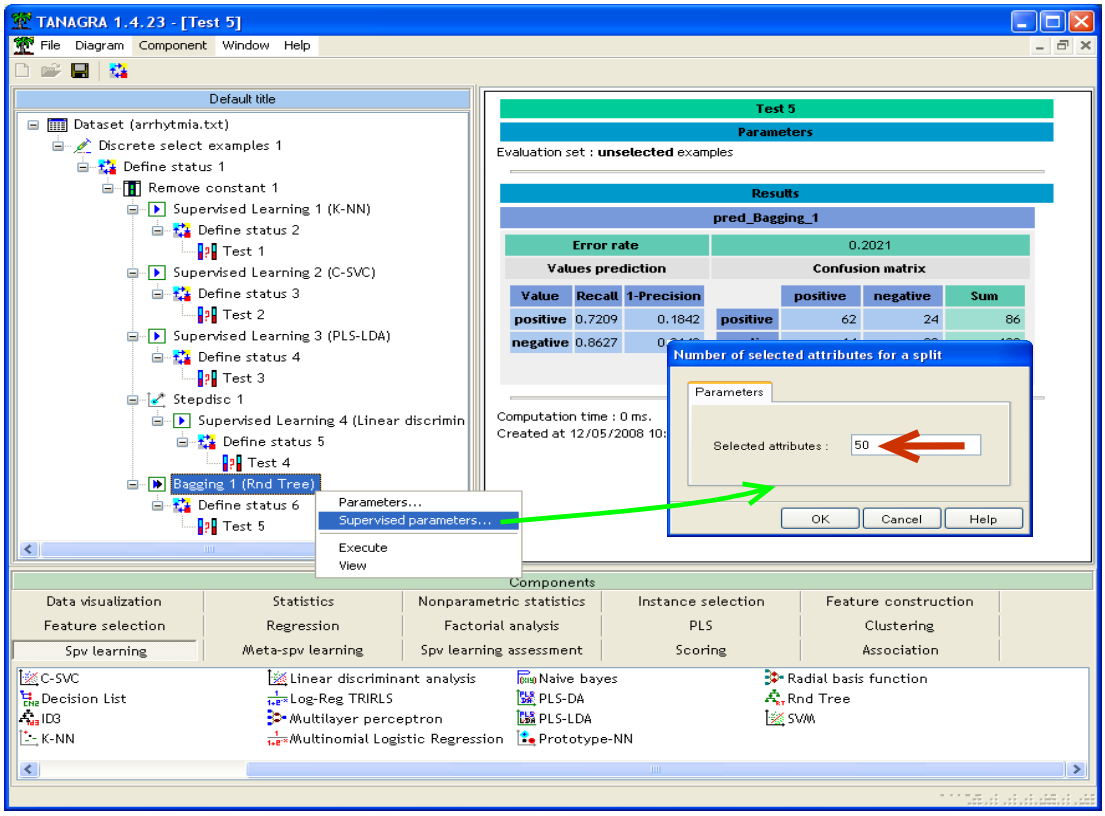
Le taux d'erreur en test est de 20.21%. C'est le meilleur taux d'erreur de toutes les approches analysées jusqu'à présent. Ce résultat corrobore les innombrables publications qui montrent la très bonne tenue des « Random Forest » dans de très nombreux domaines.

Paramétrer Random Forest (Split Variables = 50). Où sont les tournevis ici ? Le premier paramètre, évident, est le nombre de réplications. On augmente la puissance globale du classifieur lorsqu'on l'augmente, de manière marginale néanmoins arrivé à un certain stade. TANAGRA, par défaut, produit 25 arbres. Nous avons tenté d'augmenter ce chiffre (50, puis 100...), aucune amélioration n'a été constatée. Piste close donc, du moins pour nos données.

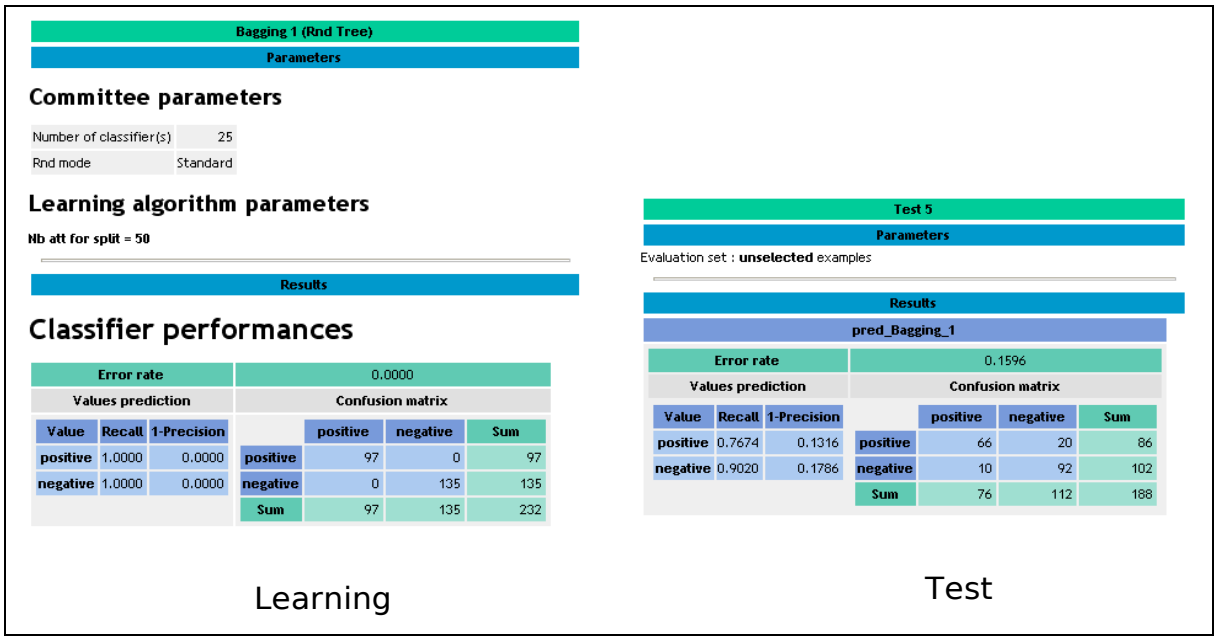
Une autre voie semble plus intéressante. Pour l'explorer, revenons sur ce qui fait le succès des « Random Forest ». Elle construit une forêt d'arbres qui, lorsqu'elles sont agrégées, prédisent mieux que les modèles individuels. Le système est d'autant plus performant que : les arbres sont décorrélés les uns des autres c.-à-d. ils ne classent pas de la même manière ; les arbres individuels sont performants. Comment agir sur ces deux propriétés ?

Le seul paramètre que l'on peut réellement manipuler est le nombre de variables que l'on étudie lors de la segmentation d'un nœud pendant la construction des arbres. Rappelons que l'opération est réalisée en deux temps : tout d'abord, on effectue un tirage aléatoire de P variables parmi les J descripteurs ; puis, on cherche la variable de segmentation parmi ces P variables. Si P est élevé c.-à-d. P # J, nous retrouvons l'algorithme usuel de construction des arbres de décision. Les arbres individuels seront performants, mais nous courrons le risque de produire des arbres similaires, qui classent de la même manière. A l'inverse, si P est petit, les arbres qui forment la forêt seront certainement très décorrélés. Mais on risque de dégrader l'apprentissage, au point de détériorer les performances des modèles individuels. Voilà. Maintenant, il faut trouver un compromis au milieu de tout ça. L'ennui est que nous avons du mal à discerner dans quel sens tourner ce « tournevis » pour produire un modèle global performant.

Nous sélectionnons le composant BAGGING 1 (RND TREE) dans le diagramme. Nous actionnons le menu SUPERVISED PARAMETERS. Dans la boîte de paramétrage, la valeur par défaut de SELECTED ATTRIBUTES est « -1 », la formule utilisée dans ce cas est « P = ROUND(LOG2(J)) + 1 ». Nous plaçons la valeur « 50 », nous validons en cliquant sur OK.



L'apprentissage est lancé avec le menu VIEW, le taux d'erreur en resubstitution est toujours égal à 0%. Voyons ce qu'il en est sur l'échantillon test en activant le menu VIEW du composant TEST 5.



Nous obtenons un taux d'erreur de **15.96%**. Le meilleur résultat que nous ayons atteint sur ce fichier de données.

3.7 Récapitulatif

Un petit tableau récapitulatif résume tous les résultats. Nous avons mis des étoiles « * » pour regrouper approximativement les méthodes selon leurs performances.

Méthode	Taux d'erreur en resubstitution	Taux d'erreur en test
K-NN (5 voisins)	24.57%	37.77%
SVM Linéaire (C = 1.0)	6.90%	28.72%
SVM Linéaire (C = 0.1)	15.52%	24.47%**
SVM – RBF (C = 1.0)	26.72%	28.19%
PLS-LDA (6 axes)	9.05%	26.06%
PLS-LDA (5 axes)	12.70%	23.94%**
LDA (18 variables)	15.52%	30.32%
LDA (6 variables)	22.41%	24.47%**
Random Forest (25 arbres, Split = -1)	0.00%	20.21***
Random Forest (25 arbres, Split = 50)	0.00%	15.96%****

Manifestement, « Random Forest » est au dessus du lot. L'écart augmente avec un paramétrage approprié. Viennent ensuite les méthodes PLS-LDA et SVM linéaire.

Autre lecture de ce tableau. Les performances des méthodes, quelles qu'elles soient, sont fortement influencés par le choix des paramètres.

4 Conclusion

Voilà un didacticiel que j'ai eu beaucoup de plaisir à écrire, car il va largement au delà du simple « Comment faire ... ». L'idée initiale était de montrer la mise en œuvre de la méthode PLS-LDA dans un contexte de surabondance de descripteurs. Contexte où la maîtrise de l'instabilité est l'enjeu principal. Du pain béni pour les méthodes dérivées de la Régression PLS.

Au fil de l'écriture, d'autres résultats très intéressants m'ont intrigué, m'emmenant à ré-orienter un peu mon propos. Voici quelques éléments que l'on peut souligner en conclusion.

La malédiction de la dimensionnalité n'est pas un mythe. Lorsque le ratio « nombre de variables / nombre d'observations » est défavorable, certaines méthodes peuvent totalement s'effondrer. C'est le cas de la méthode des plus proches voisins pour nos données. Le classifieur se démarque peu du classifieur par défaut (prédire systématiquement NEGATIVE).

La Régression PLS, tout comme les Support Vector Machine, parce qu'elles sont excellemment régularisées, se comportent très bien dans ce contexte. Les mécanismes sont différents, mais à l'arrivée, l'efficacité est similaire.

Le rôle des paramètres est mis en lumière dans ce didacticiel. Pour un problème à traiter, compte tenu des caractéristiques des données et des solutions que l'on souhaite privilégier, il importe

d'identifier les bons leviers (les fameux tournevis) et, ensuite, savoir dans quel sens les orienter. Quant à la détermination des bonnes valeurs, c'est très souvent affaire de tâtonnement. En tous les cas, bien les spécifier pèse significativement sur les performances.

Le comportement de l'analyse discriminante linéaire sur nos données est assez édifiant. Il y a plein de raisons de vouloir opérer une sélection de variables. Dans notre cas, ce levier a permis de réduire l'instabilité. Au final, nous obtenons des performances comparables à celles des SVM ou de la Régression PLS. C'est un résultat qui donne à réfléchir.

Dans notre étude, la méthode « Random Forest » est manifestement la plus performante. Bien sûr, ce résultat n'est valable que pour les données étudiées, n'allons pas généraliser cela de manière intempestive pour tous les problèmes d'apprentissage supervisé. Il reste néanmoins un sentiment mitigé. Certes, on voit à peu près pourquoi cette technique marche. De nombreux articles, qui ont analysé en détail le comportement de la méthode, l'ont suffisamment démontré. Il existe également des publications qui donnent des pistes pour l'améliorer encore. Mais l'opacité du paramétrage, tout simplement parce que l'on ne voit pas très bien comment les mettre en relation avec les caractéristiques de l'étude, rend encore son utilisation pratique difficile. Dans notre cas, nous avons testé 10, 20, 30,..., 100. La valeur « 50 » donne un bon résultat sans qu'on puisse très bien se l'expliquer par ailleurs. Dans une exploration purement empirique basé sur le schéma « je bidouille, je regarde ce que ça donne », ça peut passer. Dans un contexte de recherche où la définition des conditions de reproductibilité des stratégies est au moins aussi importante que la performance ponctuelle, ce n'est pas tenable.