Objectif

Montrer l'utilisation des RANDOM FOREST dans TANAGRA.

RANDOM FOREST est une technique d'apprentissage supervisée qui combine une technique d'agrégation, le BAGGING, et une technique particulière d'induction d'arbres de décision. Lors de la construction de l'arbre, pour initier la segmentation d'un nœud, la méthode effectue dans un premier temps une sélection aléatoire parmi les variables candidates ; sélection à partir de laquelle, dans un deuxième temps, elle cherche la variable de segmentation. La taille de la sélection est un paramètre de l'algorithme. Si elle n'est pas spécifiée, on propose généralement la formule « partie entière de $Log_2(J)+1$ », où J est le nombre total de variables.

L'idée est assez simple, et déjà présente dans les premiers articles de BREIMAN (1996) sur le BAGGING : en exacerbant la variabilité de la technique d'apprentissage, nous augmentons l'efficacité de la technique d'agrégation.

Fichier

Nous traitons le fichier HEART de l'UCI (http://www.ics.uci.edu/~mlearn/MLRepository.html).

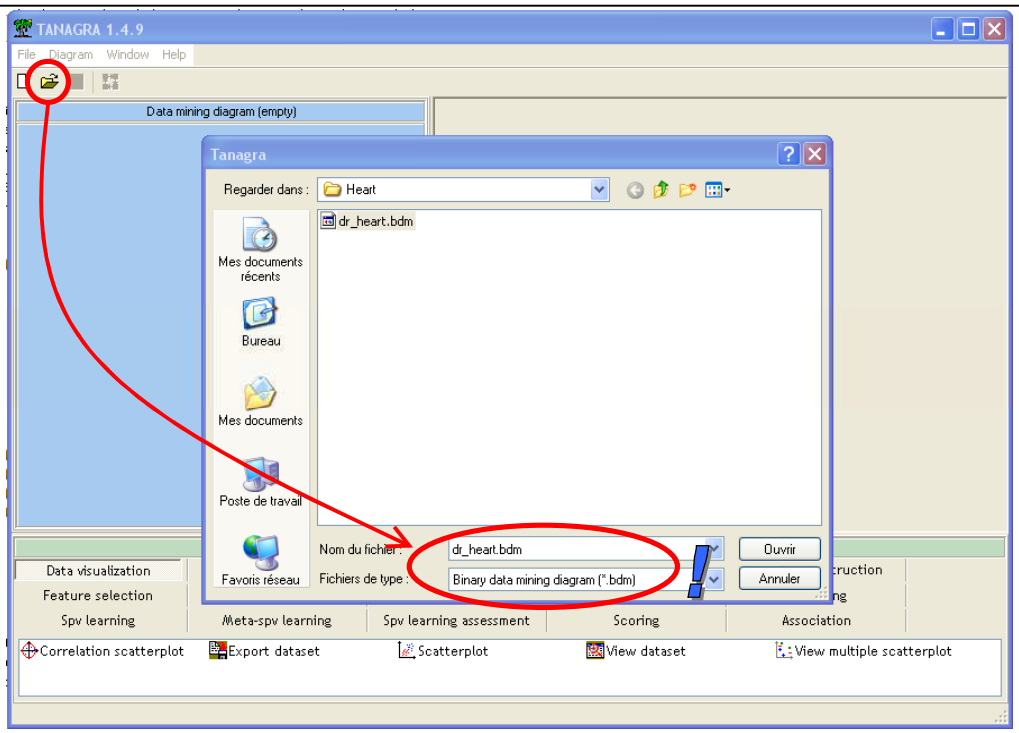
L'objectif est de prédire la présence ou l'absence de maladie cardio-vasculaire chez des patients. Ce fichier a déjà été utilisé dans nos didacticiels (http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/dr_comparer_spv_learning.pdf).

RANDOM FOREST

Charger les données

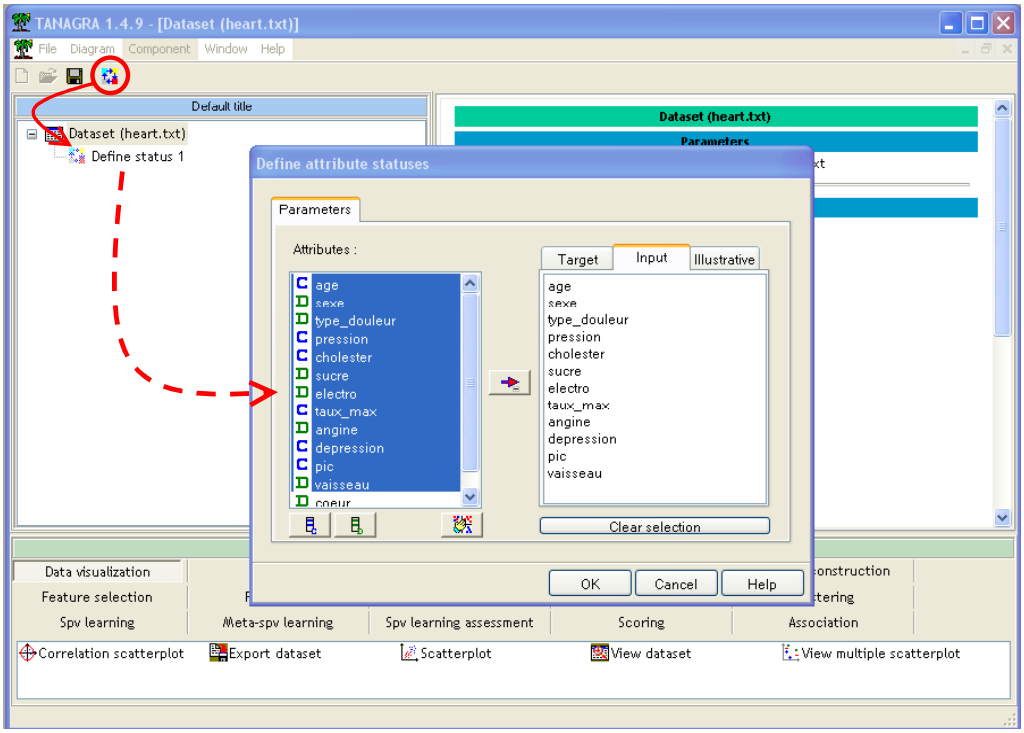
Nous chargeons le fichier DR_HEART.BDM, déjà au format TANAGRA[1], avec le menu FILE/OPEN.

[1] Le format BDM est un format binaire qui contient à la fois les données et les traitements associés. Il ne peut pas être manipulé en dehors du logiciel. A la différence du format TDM que l'on peut ouvrir dans un éditeur de texte par exemple.



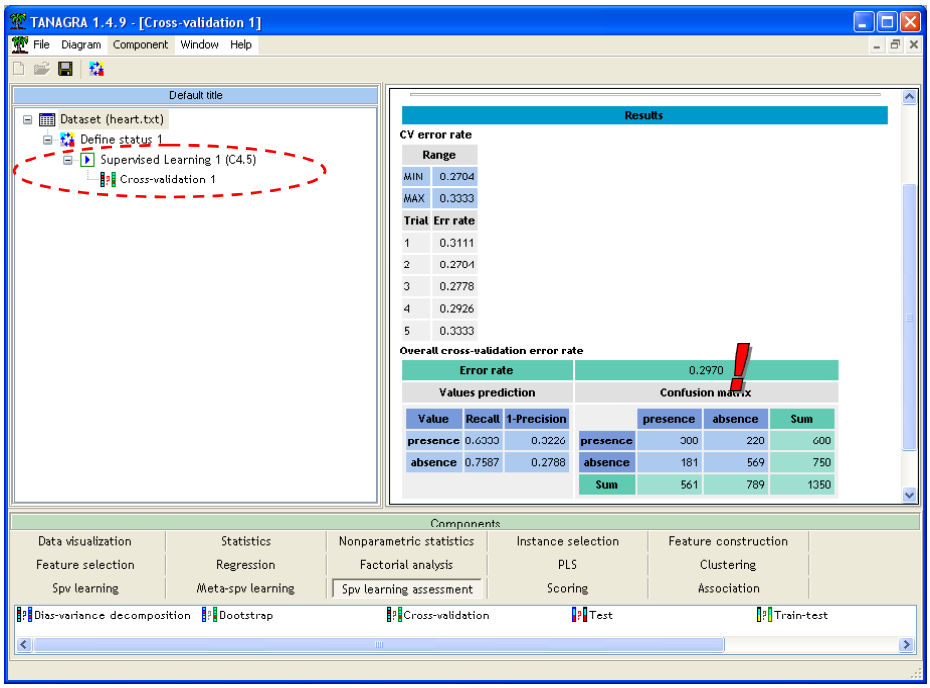
Définir le problème

L'étape suivante consiste à désigner, à l'aide du composant DEFINE STATUS, la variable à prédire CŒUR (TARGET), et les variables prédictives (INPUT), toutes les autres.



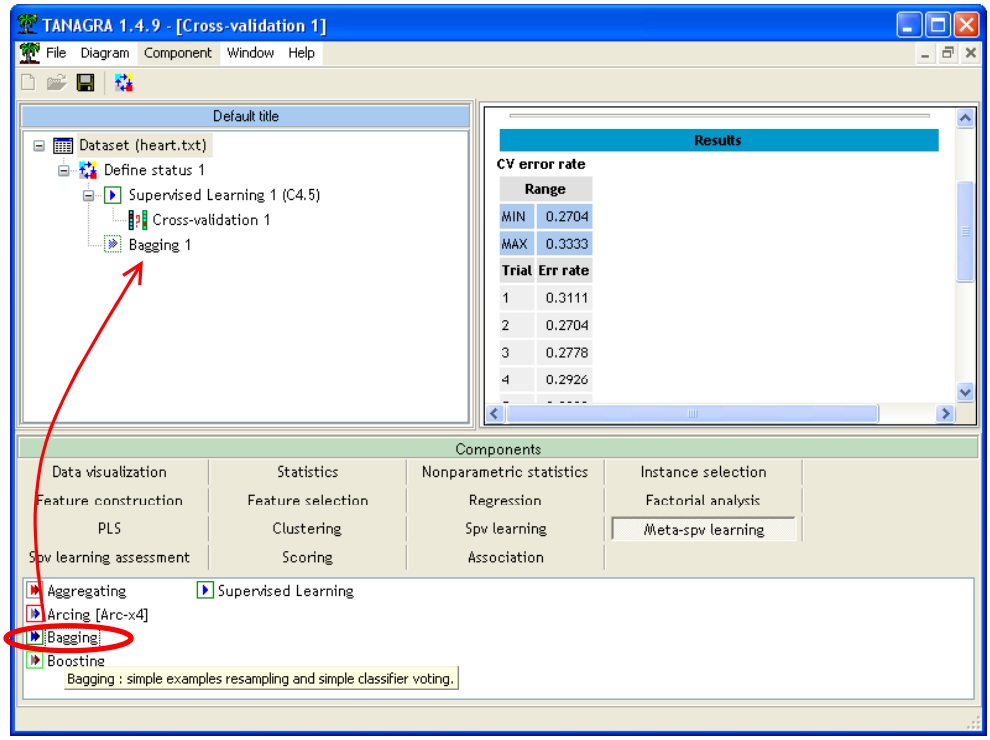
Apprentissage et évaluation de C4.5

Pour avoir un point de comparaison, évaluons tout d'abord les performances de C4.5. Pour ce faire, nous ajoutons le composant d'apprentissage correspondant et la validation croisée. L'exécution indique un taux d'erreur de 29.70%.

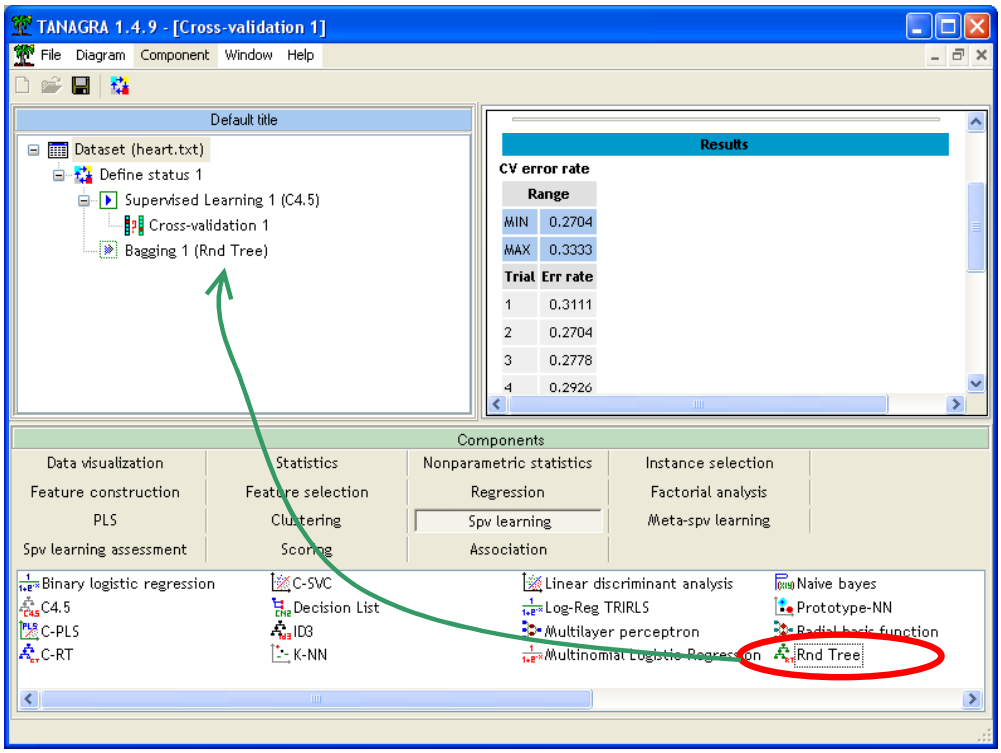


RANDOM FOREST

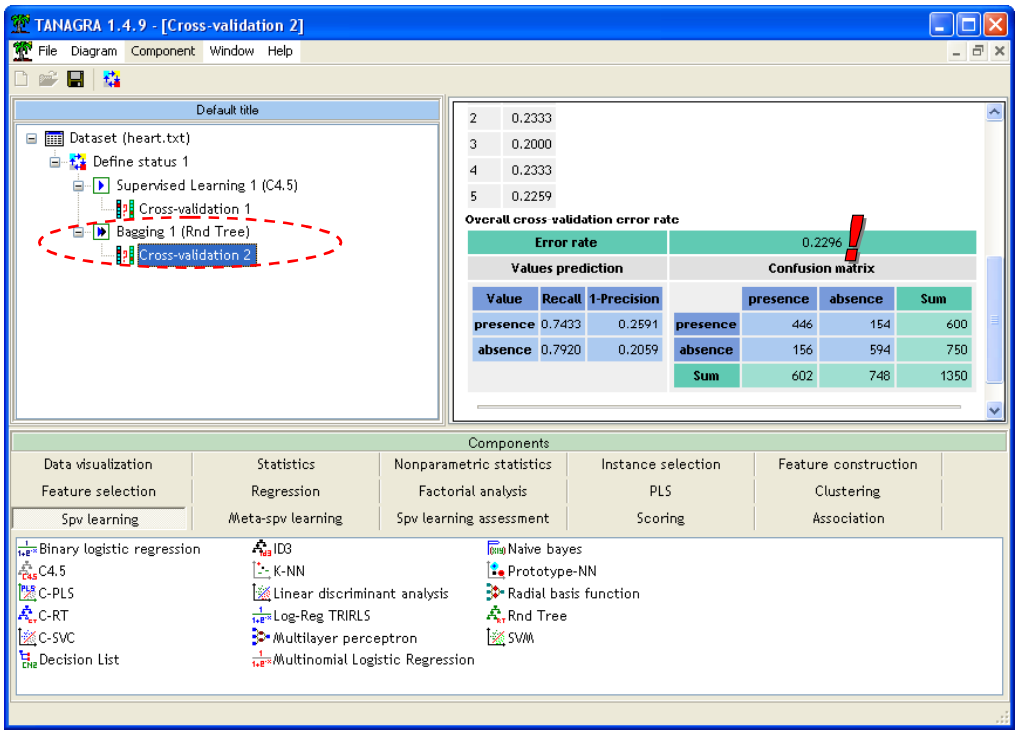
Plaçons maintenant l'apprentissage avec les « Random Forest », et évaluons-le avec le même canevas. L'insertion des Random Forest dans le diagramme se fait en deux temps. Nous devons placer le composant meta-apprentissage BAGGING (onglet Meta-spv learning).



Puis nous y insérons la méthode d'apprentissage RANDOM TREE. Il est évident qu'utiliser cette méthode en dehors du schéma d'agrégation n'a aucun sens, elle présenterait des performances particulièrement médiocres par rapport à un arbre classique.



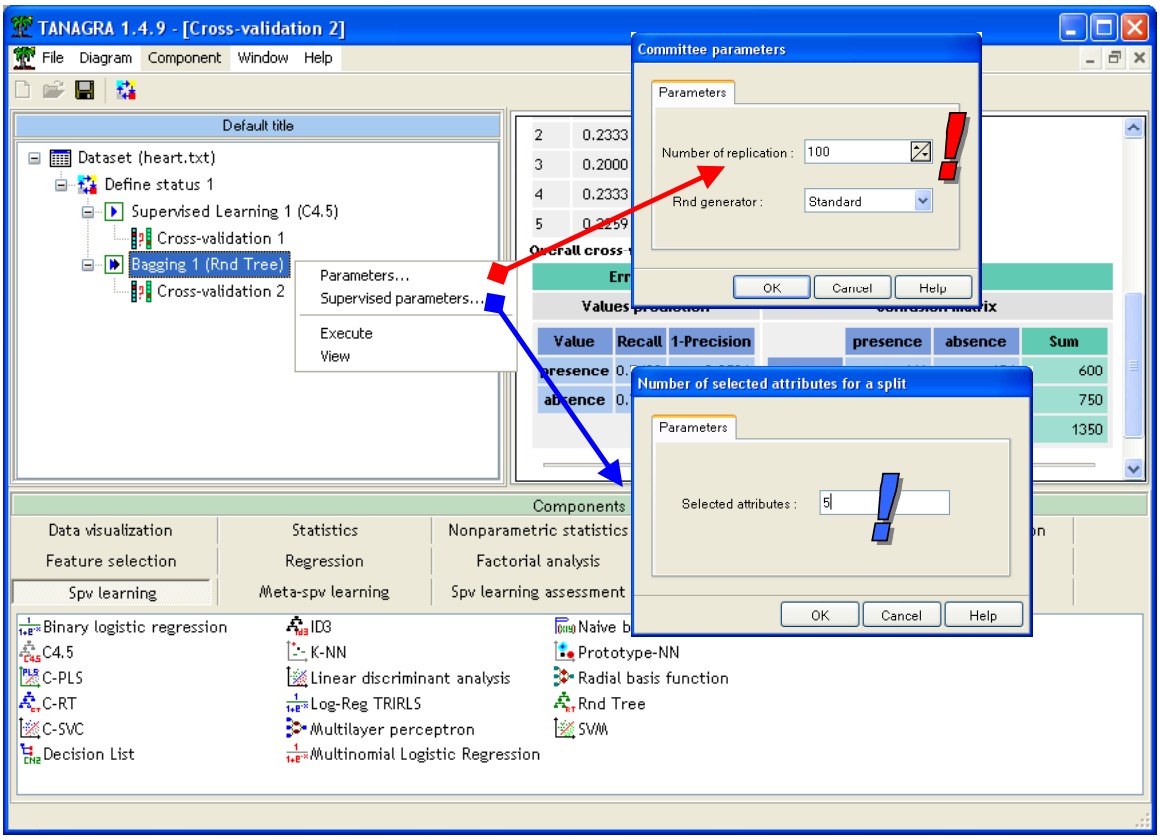
Reste alors à placer la validation croisée pour en évaluer les performances.



Nous constatons que le taux d'erreur a fortement diminué, il est passé à 22.9%

Modifier les paramètres

Les modifications des paramètres peuvent porter sur le nombre d'instances d'apprentissage (PARAMETERS) et la taille de la pré-sélection lors de la construction de l'arbre (SUPERVISED PARAMETERS). Fixons-les respectivement à 100 et 5 (si nous spécifions une valeur négative dans le second cas, TANAGRA utilise la formule décrite plus haut).



Le taux d'erreur n'évolue guère sur notre application.

Error rate		0.2281	
Values prediction		**Confusion matrix**	
Value	**Recall**	**1-Precision**	

	presence	absence	Sum
presence	0.7417	0.2559	445	155	600
absence	0.7960	0.2061	153	597	750
			Sum	598	752	1350

Souvent, augmenter le nombre de réplications améliore les performances, mais de manière très marginale néanmoins lorsque nous arrivons à un certain stade (une centaine).