

1 Objectif

Stratégie « wrapper » pour la sélection de variables. Suite.

Ce didacticiel fait suite à celui consacré à la stratégie wrapper pour la sélection de variables en apprentissage supervisé (<http://tutoriels-data-mining.blogspot.com/2009/05/strategie-wrapper-pour-la-selection-de.html>). Nous y analysons le comportement de Sipina, puis nous avons programmé une procédure ad hoc dans R. Dans ce didacticiel, nous étudions la mise en oeuvre de la méthode dans les logiciels **Knime 2.1.1**, **Weka 3.6.0** et **RapidMiner 4.6**.

La démarche est la suivante : (1) utilisation du fichier d'apprentissage pour la sélection des variables les plus performantes pour le classement ; (2) création du modèle sur les descripteurs sélectionnés ; (3) évaluation des performances sur un fichier test contenant toutes les variables candidates.

Ce troisième point est très important. Nous ne pouvons pas connaître initialement les variables prédictives qui seront finalement retenues. Il ne faut pas que nous ayons à préparer manuellement le fichier test en y intégrant uniquement celles qui auront été choisies par la procédure wrapper. C'est une condition essentielle pour que la démarche soit automatisable. En effet, dans le cas contraire, chaque modification de paramétrage dans la procédure wrapper aboutissant à autre sous-ensemble de descripteurs nous obligerait à modifier manuellement le fichier test. Ce qui s'avère très rapidement fastidieux.

A la lumière de ce cahier des charges, il est apparu que seul Knime a permis de mettre en place le dispositif complet. Avec les autres logiciels, il est certes possible de sélectionner les variables pertinentes sur le fichier d'apprentissage. Je n'ai pas pu en revanche (ou je n'ai pas su) réaliser simplement le déploiement sur un fichier test comprenant la totalité des variables candidates.

Le modèle d'indépendance conditionnel est la méthode d'apprentissage supervisé utilisée, le modèle bayésien naïf selon la terminologie de l'apprentissage automatique¹.

2 Données

Nous avons subdivisé les données en 2 fichiers distincts². Ils sont au format **ARFF** (Weka). Le premier, **mushroom-train.arff**, correspond à l'échantillon d'apprentissage (2000 observations). Nous utiliserons d'une part pour sélectionner les variables avec l'approche wrapper, d'autre part pour construire le modèle définitif avec les descripteurs retenus. Le second, **mushroom-test.arff**, correspond à l'échantillon test (6124 observations). Nous l'utiliserons pour évaluer les performances en généralisation. Comme nous le soulignons plus haut, il est (et il doit être) structurellement identique au premier fichier. Nous ne connaissons pas, a priori, les descripteurs qui seront finalement sélectionnés.

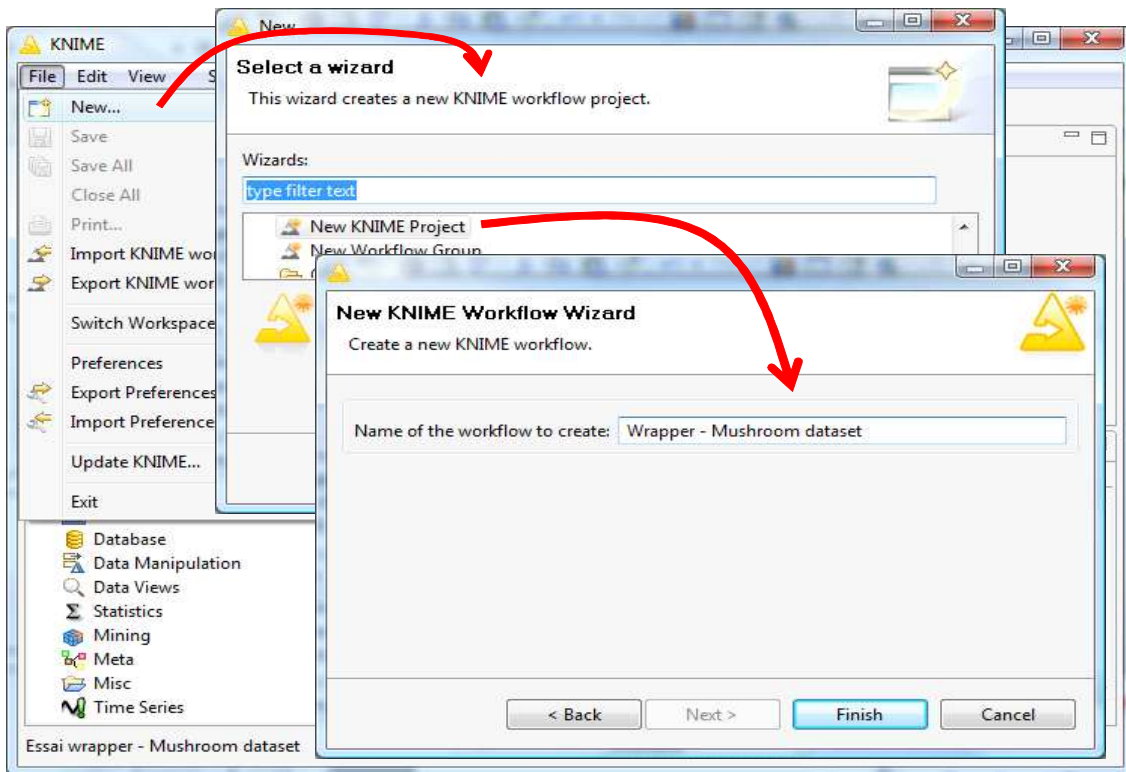
¹ http://en.wikipedia.org/wiki/Naive_Bayes_classifier

² <http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/mushroom.wrapper.arff.zip>

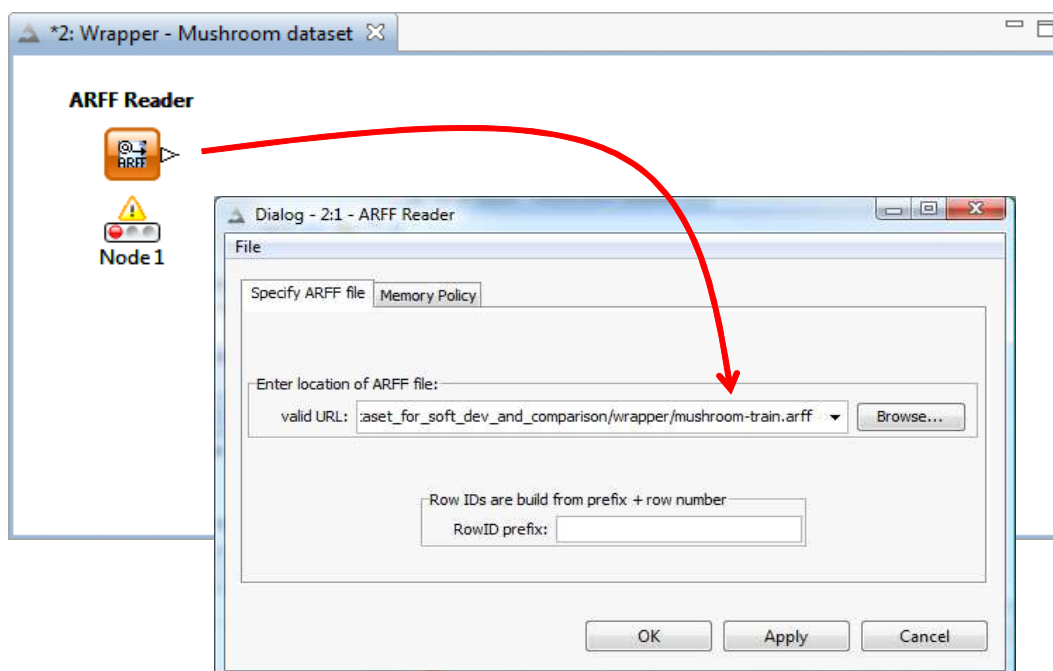
3 Stratégie wrapper avec Knime

3.1 Importation des données

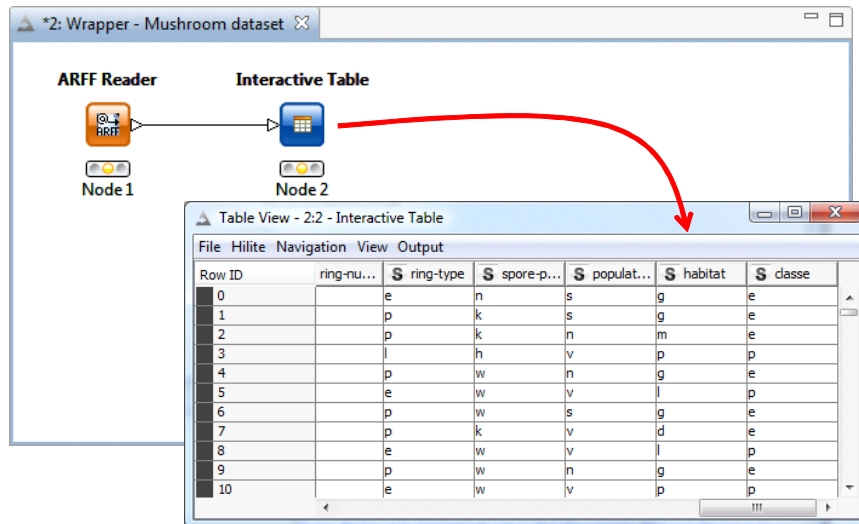
Après avoir démarré le logiciel, nous créons un nouveau projet en actionnant le menu FILE / NEW. Nous choisissons un projet Knime, nous le nommons « Wrapper – Mushroom dataset ».



Nous insérons le composant ARFF READER (branche IO / READ dans le « Node Repository ») dans l'espace de travail. Avec le menu CONFIGURE, nous sélectionnons le fichier mushroom-train.arff.



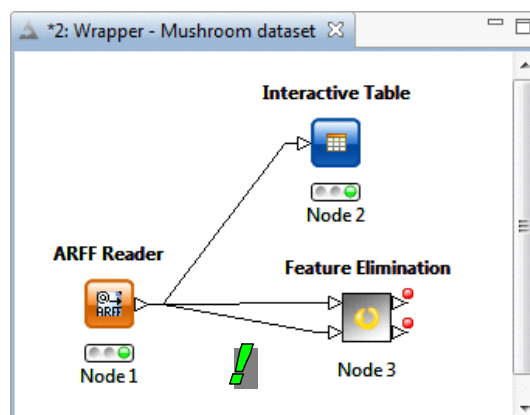
Pour vérifier si les données ne présentent pas d'erreurs, nous introduisons le composant INTERACTIVE TABLE (*DATAVIEWS*) auquel nous branchons le précédent. Via le menu contextuel EXECUTE AND OPEN VIEW, nous obtenons l'affichage des 2000 observations. « Classe » est la variable à prédire.



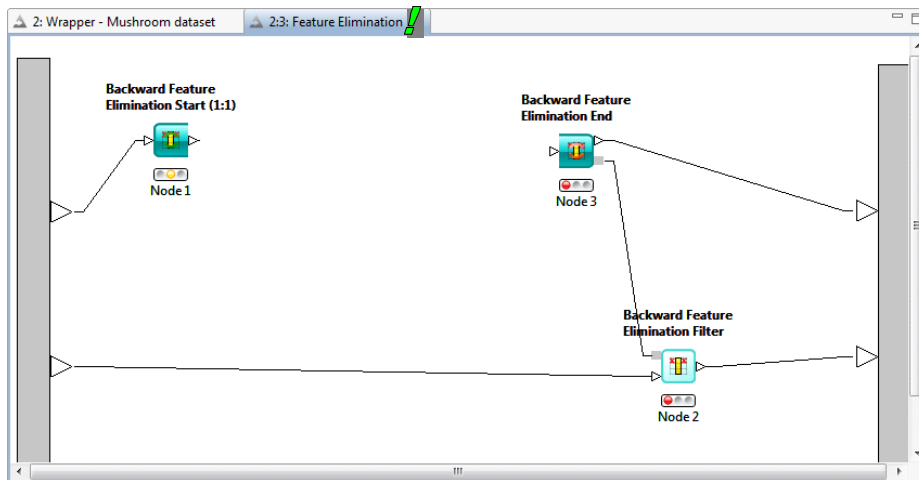
3.2 Le méta nœud « Feature Elimination »

Un des aspects enthousiasmants de Knime est la possibilité d'utiliser (et de définir semble-t-il) des méta nœuds qui représentent des enchaînements typiques d'opérations. Dans notre contexte, nous nous intéressons au composant FEATURE ELIMINATION (*META*). Il implémente la stratégie wrapper sous la forme d'un algorithme glouton d'élimination des descripteurs. Au départ, le modèle est construit sur la totalité des variables prédictives. On enlève tour à tour chacune d'elles. Celle qui induit la plus faible dégradation du taux d'erreur est éliminée. Le retrait est définitif, il ne sera plus remis en cause. Et on continue ainsi jusqu'à ce que l'on ait retiré toutes les variables. Au final, nous conserverons le plus petit sous-ensemble de descripteurs qui présente le taux d'erreur le plus faible c.-à-d. à performances égales, nous choisirons la solution comportant le moins de variables en vertu du principe de parcimonie.

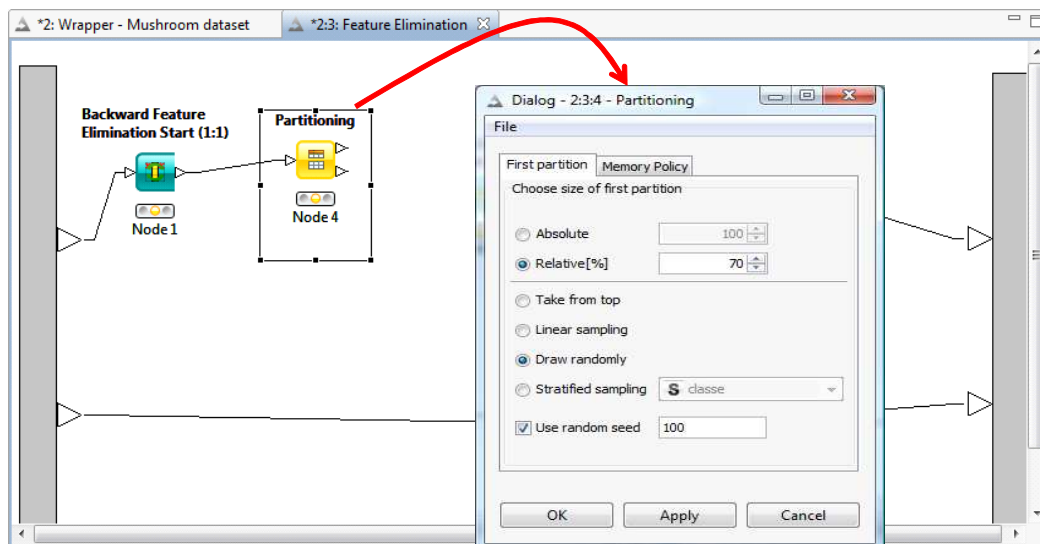
Nous insérons FEATURE ELIMINATION dans l'espace de travail et nous lui connectons deux fois (!) le composant ARFF READER. Deux fois parce que nous utilisons l'échantillon d'apprentissage pour, d'une part, sélectionner les variables pertinentes et, d'autre part, construire le modèle définitif.



Nous double-cliquons sur le composant FEATURE ELIMINATION pour l'éditer.



L'idée est de programmer sous forme d'enchaînement d'opérateurs une boucle de recherche. Ce n'est pas facile. Dans notre cas, le composant BACKWARD FEATURE ELIMINATION START illustre le début de la boucle, BACKWARD FEATURE ELIMINATION END, la fin. Dans le corps de la boucle, nous devons implémenter l'apprentissage (construction du modèle) et l'évaluation de chaque configuration candidate (mesurer le taux d'erreur en généralisation). Pour obtenir une estimation non biaisée du taux d'erreur, et ne pas favoriser les modèles complexes, il faut bien évidemment que les données servant au test ne soient pas mises à contribution dans la phase d'apprentissage³. Ainsi nous devons partitionner les données en deux parties à l'intérieur de la boucle. Pour ce faire, nous insérons le composant PARTITIONNING (DATA MANIPULATION / ROW / TRANSFORM) auquel nous connectons le premier composant.

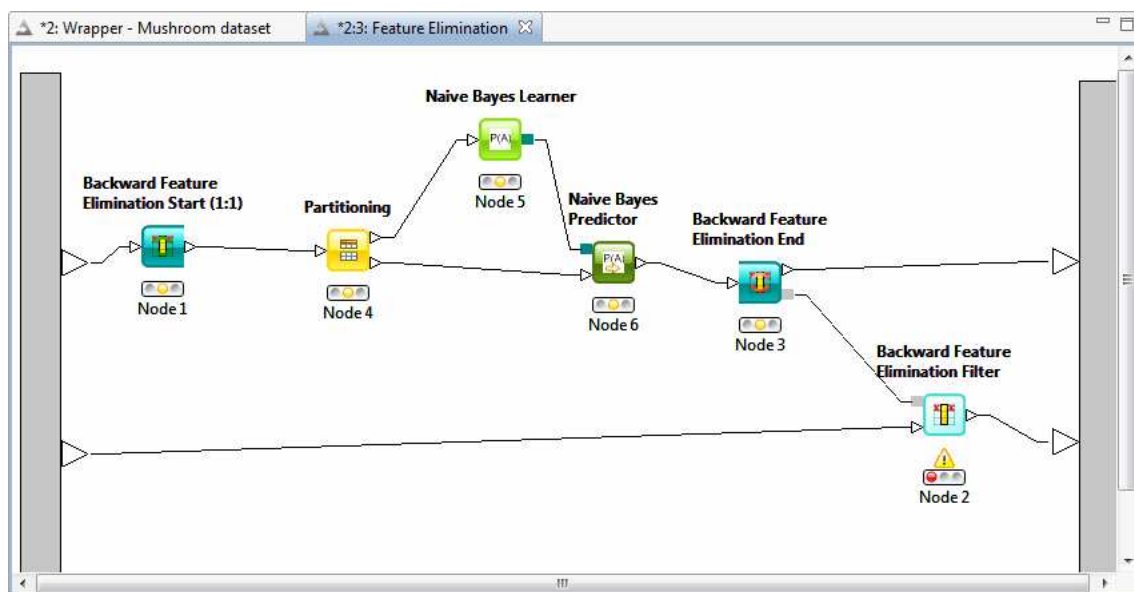


³ Dans l'idéal, la validation croisée est utilisée pour évaluer les configurations (R. Kohavi, G. John, « Wrappers for feature subset selection », <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.720>). Il est certainement possible d'intégrer le meta nœud dédié (X-VALIDATION) dans l'algorithme de recherche. Mais cela complexifierait trop notre dispositif, nous éloignant de notre objectif initial qui est de montrer, de la manière la plus simple possible, l'implémentation de la stratégie « wrapper » dans Knime. Nous nous en tiendrons au schéma « hold-out » (construction du modèle et évaluation sur échantillons séparés) dans ce didacticiel.

Nous le paramétrons (menu CONFIGURE) pour utiliser 70% des données pour la modélisation, et donc 30% pour mesurer les performances. Par ailleurs, nous fixons le paramètre SEED du générateur de nombre aléatoire à 100 pour que nous aillions les mêmes résultats vous et moi.

Remarque : Attention, nous travaillons exclusivement sur le premier fichier de données comportant 2000 observations ici. Il n'est pas question d'utiliser, de quelque manière que ce soit, l'échantillon test de 6124 individus (mushroom-test.arff) qui servira uniquement à juger des performances du sous-ensemble de variables finalement retenu. Quand on parle de 70% des observations, il s'agit bien de 70% de 2000, soit 1400 individus.

Il nous reste à introduire les composants de modélisation NAIVE BAYES LEARNER⁴ (MINING / BAYES) et de prédiction NAIVE BAYES PREDICTOR (MINING / BAYES) que nous connectons aux différents composant déjà présents comme suit.

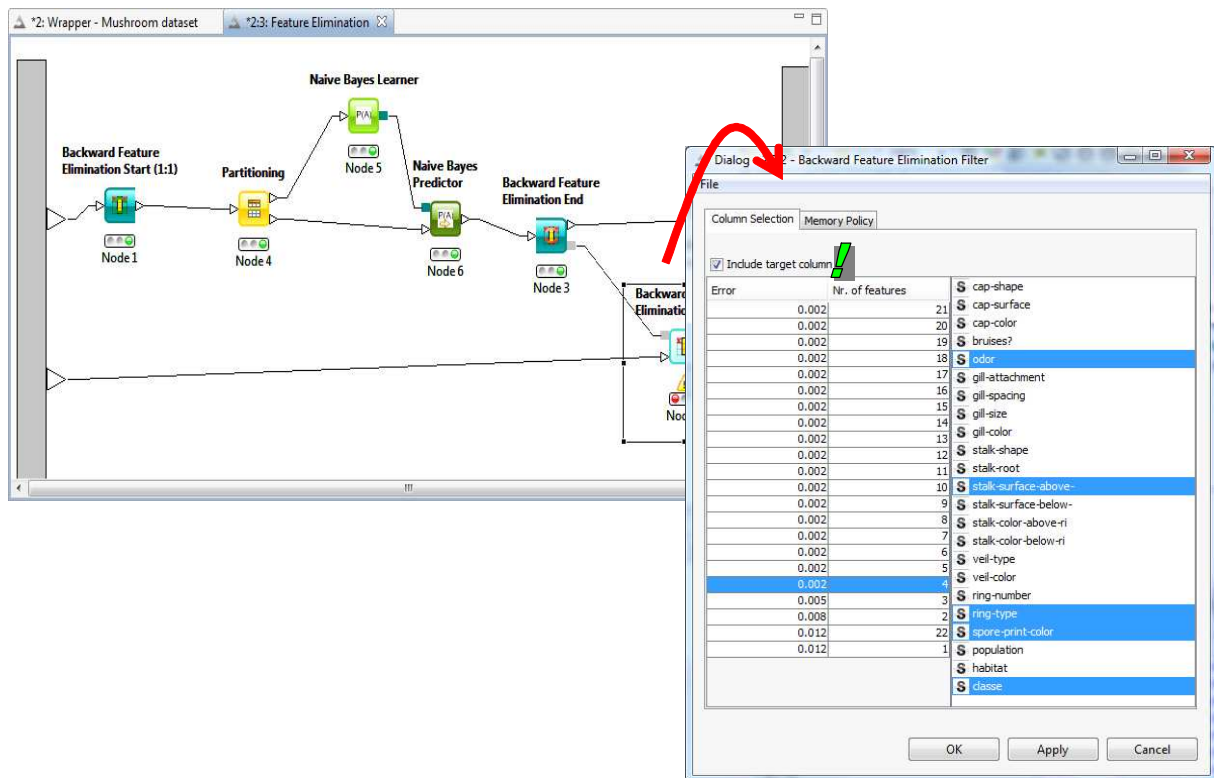


Nous sommes prêts pour lancer la procédure de sélection. Le plus simple est d'activer le menu EXECUTE de la fin de la boucle de recherche BACKWARD FEATURE ELIMINATION END.

Nous pouvons suivre le déroulement des opérations dans la fenêtre « Console ». Après un certain temps, tous les composants sont passés au vert. Il ne reste plus qu'à paramétrer (menu CONFIGURE) le composant BACKWARD FEATURE ELIMINATION FILTER qui permet de choisir le sous-ensemble « optimal » de variables.

Dans la boîte de dialogue qui apparaît, nous constatons que le plus petit taux d'erreur que l'on peut obtenir est de 0.002, le sous-ensemble minimal de variables correspondant en comporte 4 (odor, stalk-surface-above, ring-type et spore-print-color).

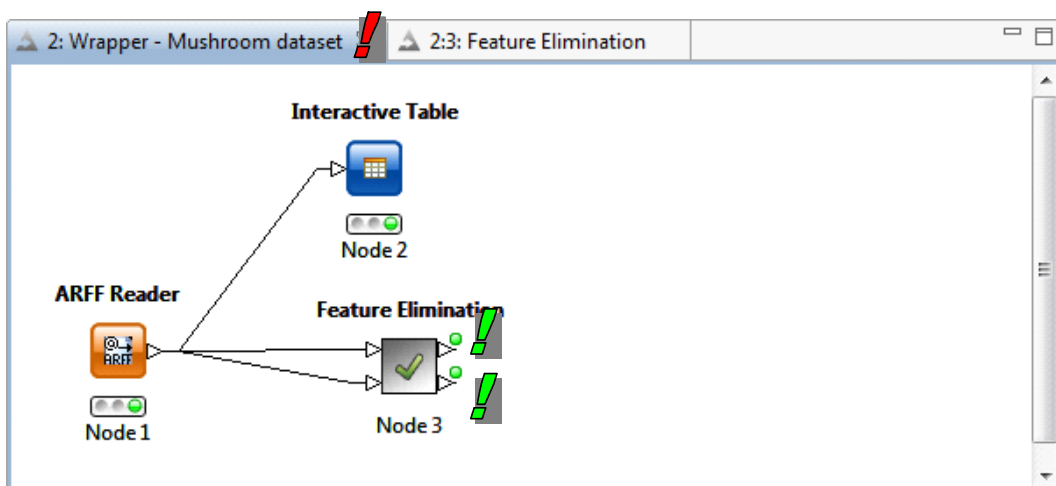
⁴ Assurez vous que la variable cible est bien CLASSE en actionnant le menu CONFIGURE.



Nous sélectionnons cette solution, nous cochons également l'option « Include Target Column » et nous validons en cliquant sur le bouton OK. Puis nous actionnons le menu EXECUTE du composant. Ainsi, en sortie du méta nœud de sélection de variables, ce seront bien ces 4 variables susmentionnées, avec la variable cible, qui seront utilisées pour la construction du modèle définitif.

3.3 Construction du modèle définitif

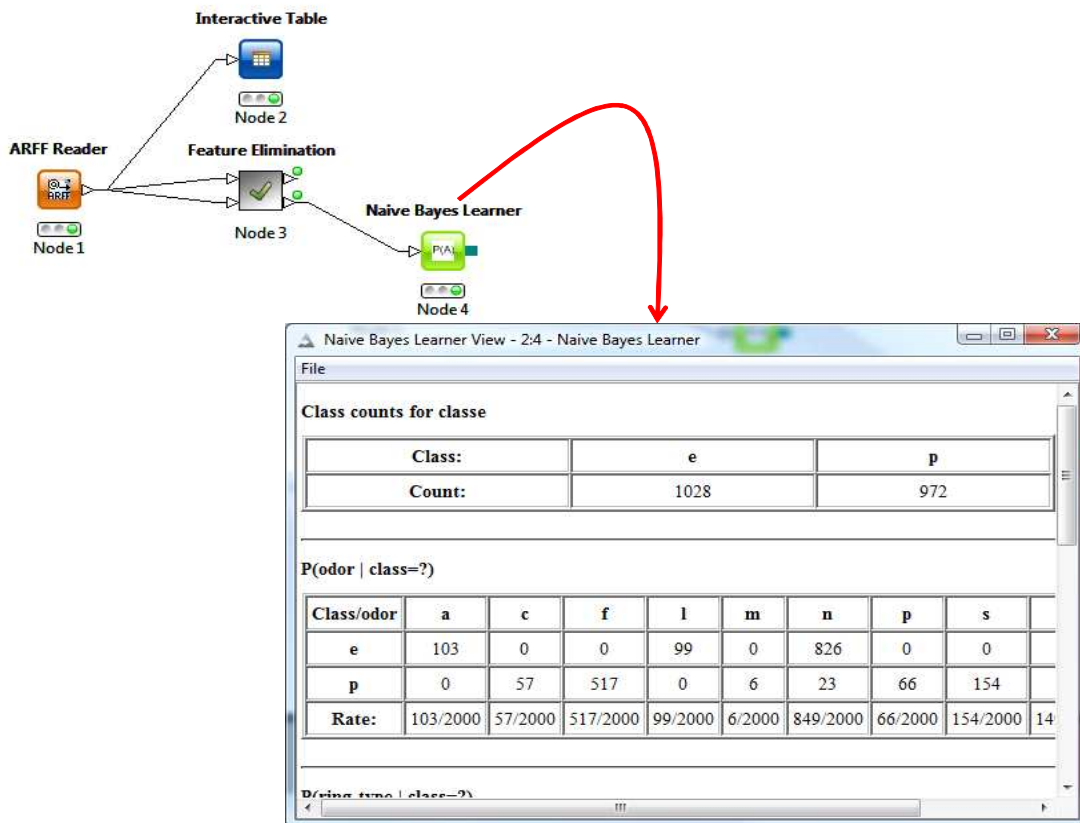
Nous revenons dans l'onglet initial de l'espace de travail. Nous notons que les deux sorties de FEATURE ELIMINATION sont passés au vert.



La seconde sortie du composant FEATURE ELIMINATION nous intéresse particulièrement : elle met à notre disposition le fichier d'apprentissage (mushroom-train.arff) réduit aux 4 variables (plus la variable cible) sélectionnées par la procédure wrapper.

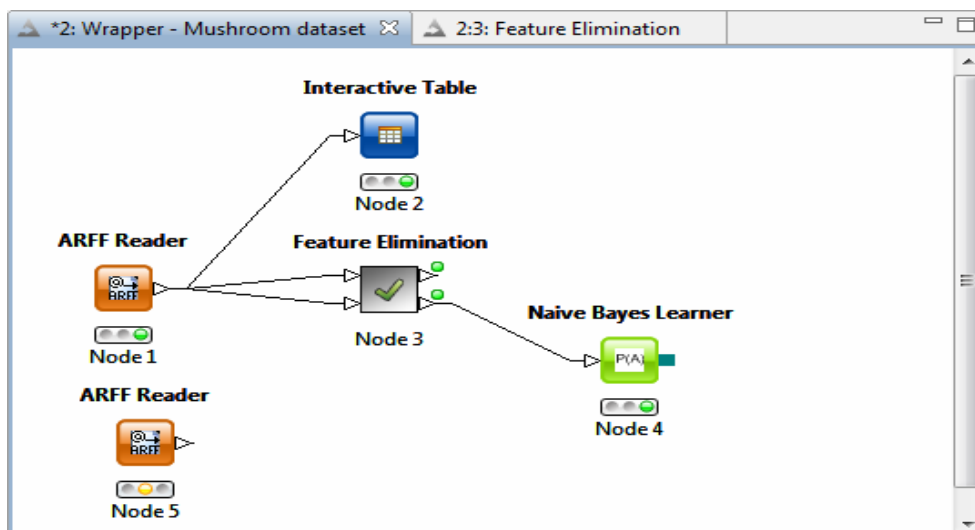
Nous lui associons de nouveau le composant NAIVE BAYES LEARNER (*MINING*) en spécifiant la variable cible CLASSE (menu CONFIGURE). Nous actionnons le menu EXECUTE AND OPEN VIEW.

Nous obtenons une série de tableaux de contingence croisant chaque variable prédictive à la cible. Ils servent à calculer les probabilités conditionnelles nécessaires au modèle d'indépendance conditionnelle. Nous n'avons plus que 4 variables prédictives à ce stade.

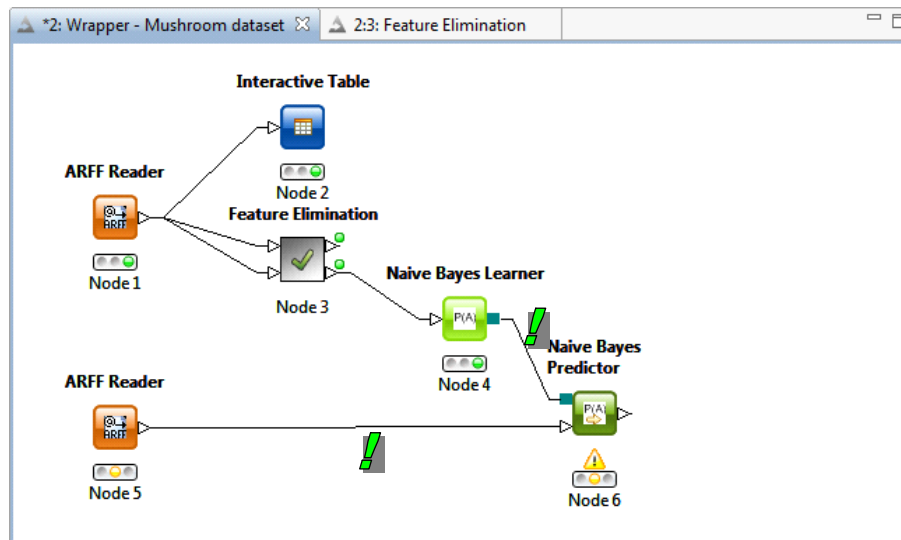


3.4 Evaluation sur l'échantillon test

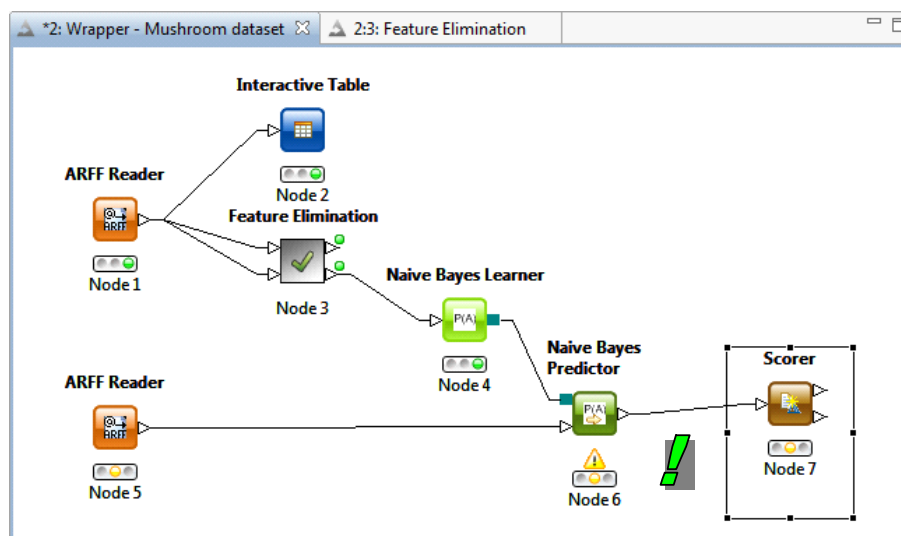
Pour évaluer le modèle à quatre variables, nous chargeons le fichier test (mushroom-test.arff) à l'aide d'un nouveau composant ARFF READER (*IO / READ*).



Puis nous insérons le prédicteur NAIVE BAYES PREDICTOR (*MINING / BAYES*). Il prend en entrée l'échantillon test et le modèle en provenance du « learner ».



Pour construire la matrice de confusion et calculer le taux d'erreur en généralisation, nous utilisons le composant SCORER (*MINING / SCORING*). Nous actionnons le menu CONFIGURE pour nous assurer que nous confrontons bien la variable cible CLASSE avec la prédiction du modèle WINNER.



Nous cliquons enfin sur le menu EXECUTE AND OPEN VIEW pour obtenir la matrice de confusion. Le taux d'erreur est de 0.212% (13 individus mal classés sur 6124).

Confusion Matrix - 2:7 - Scorer

classe \ Wi...	e	p
e	3180	0
p	13	2931

Correct classified: 6,111 Wrong classified: 13
 Accuracy: 99.788 % Error: 0.212 %

4 Stratégie wrapper avec Weka

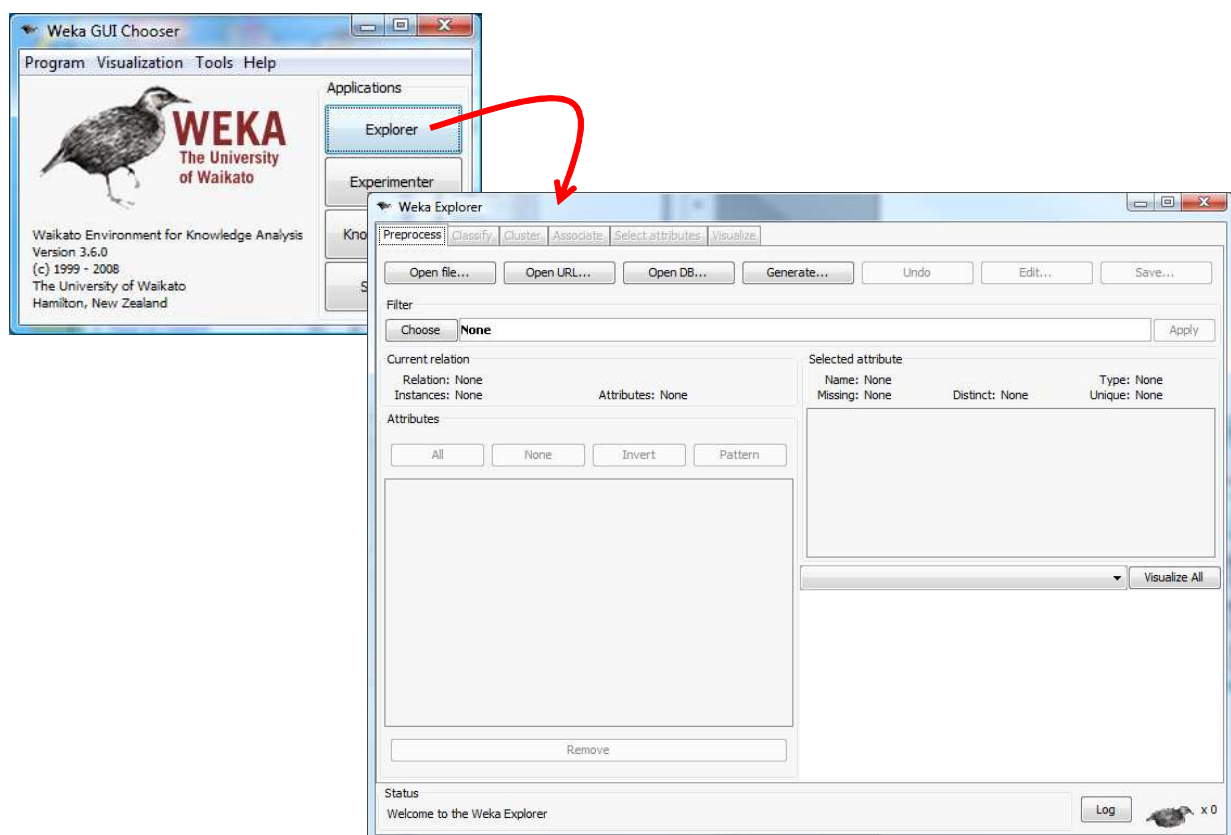
Dans le mode EXPLORER, Weka propose un dispositif performant pour sélectionner les meilleures variables prédictives avec la stratégie « wrapper »⁵.

Contrairement à Knime, il est possible de mettre en place facilement la validation croisée lors du processus de sélection. En revanche, il n'est pas aisé de déployer le modèle avec les variables choisies sur le fichier test comportant la totalité des variables initiales. En effet, pour construire le modèle final sur les variables sélectionnées, nous devons les retirer explicitement avec un outil spécifique (bouton REMOVE dans l'onglet PREPROCESS). De fait, lorsque nous souhaitons évaluer le modèle sur un échantillon externe (SUPPLIED TEST SET, onglet CLASSIFY), Weka, qui réalise un test de cohérence, nous annonce que le fichier test n'est pas compatible avec les données ayant servi à construire le modèle.

Dans ce chapitre donc, nous nous bornerons à montrer comment réaliser la sélection de variables avec la stratégie wrapper dans Weka. Nous ne réaliserons pas l'évaluation du modèle final sur l'échantillon test.

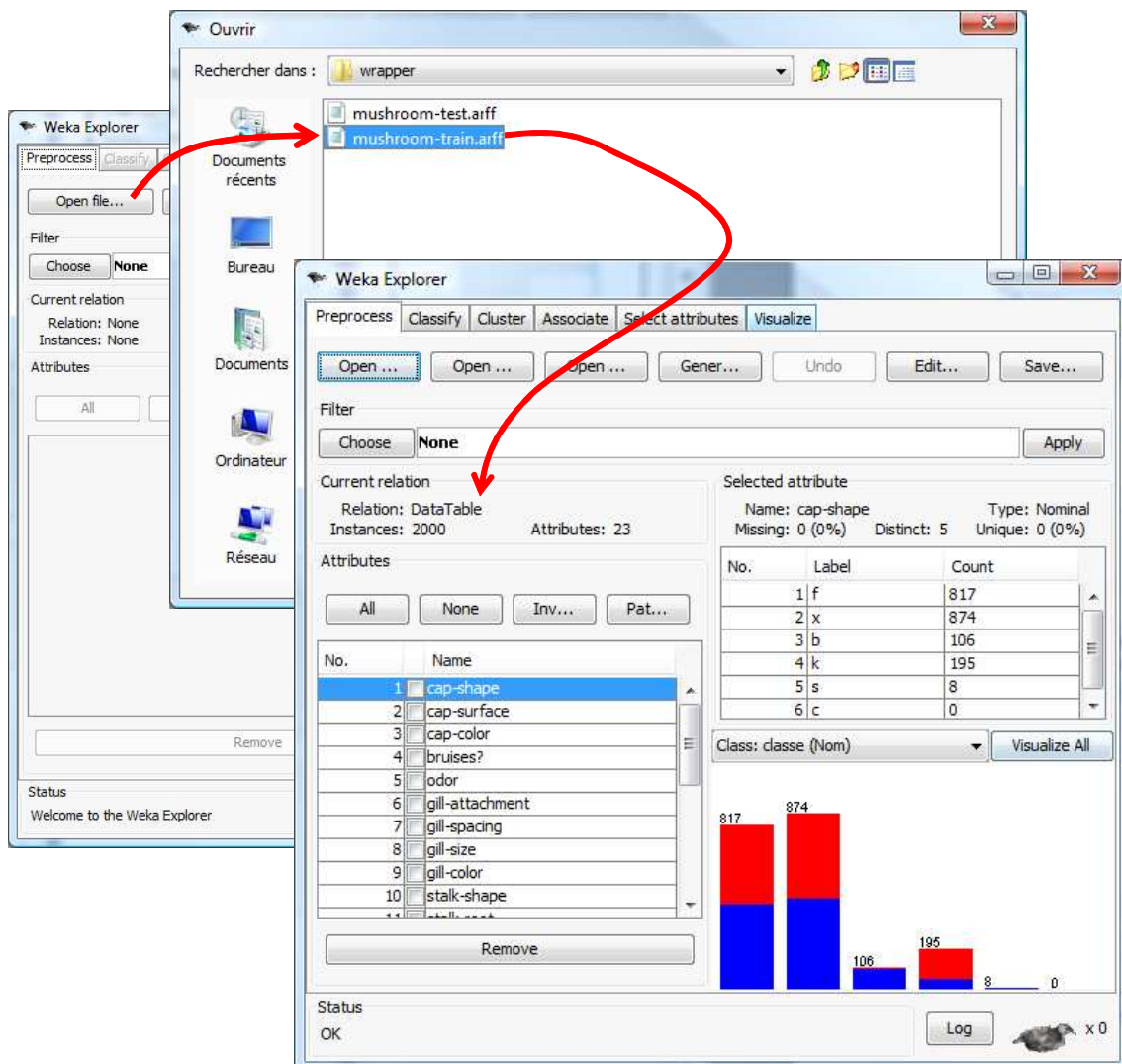
4.1 Importation des données

Après avoir démarré Weka, nous demandons le mode EXPLORER. La fenêtre principale de l'application est affichée.



⁵ Il ne semble pas opérationnel dans le mode KNOWLEDGE FLOW. J'ai bien cherché, mais je peux me tromper bien sûr.

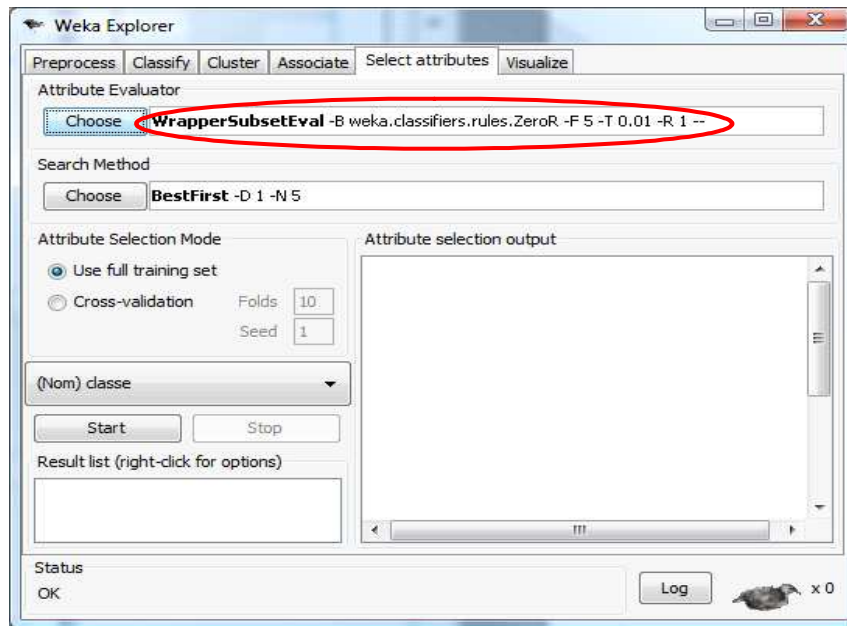
Dans l'onglet PREPROCESS, nous cliquons sur le bouton OPEN FILE et nous sélectionnons notre fichier mushroom-train.arff.



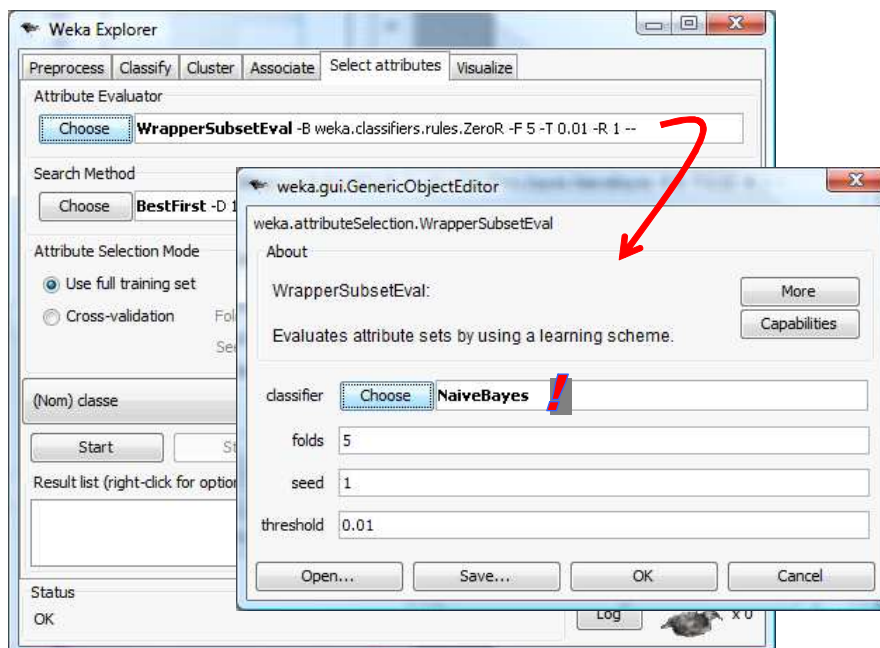
4.2 Procédure wrapper dans Weka

Nous passons à l'onglet SELECT ATTRIBUTES.

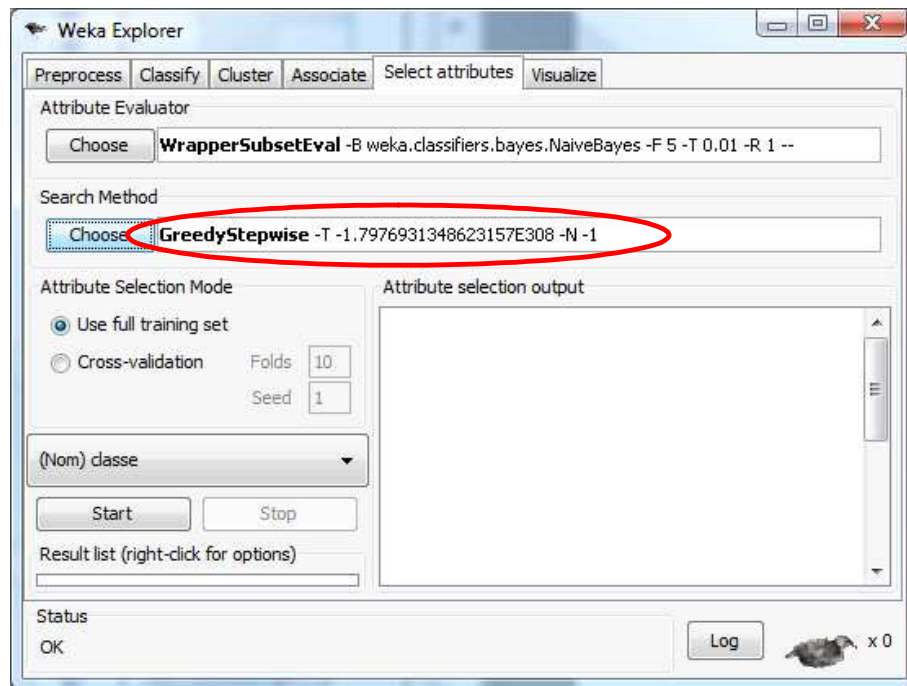
Dans ATTRIBUTE EVALUATOR, nous devons spécifier le mode d'évaluation de la pertinence des descripteurs. Nous choisissons (bouton CHOOSE) l'approche WRAPPERSUBSETEVAL.



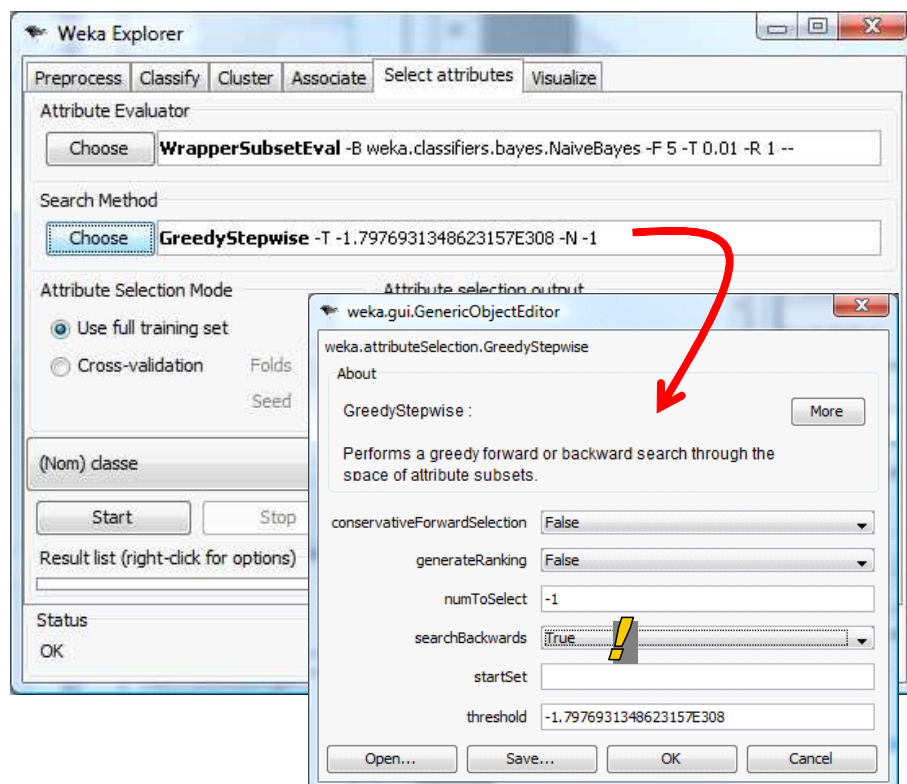
Nous le paramétrons en cliquant dans la barre de description. Dans la boîte de dialogue, nous sélectionnons l'algorithme d'apprentissage supervisé, NAIVE BAYES, et le nombre de portions de la validation croisée (par défaut FOLDS = 5). Ce dernier point est important. Cela veut dire que la procédure utilise la validation croisée pour mesurer les performances en généralisation de chaque sous-ensemble de variables à tester. C'est une alternative tout à fait avantageuse du schéma « hold-out » (apprentissage – test) que nous avons mis en place dans Knime.



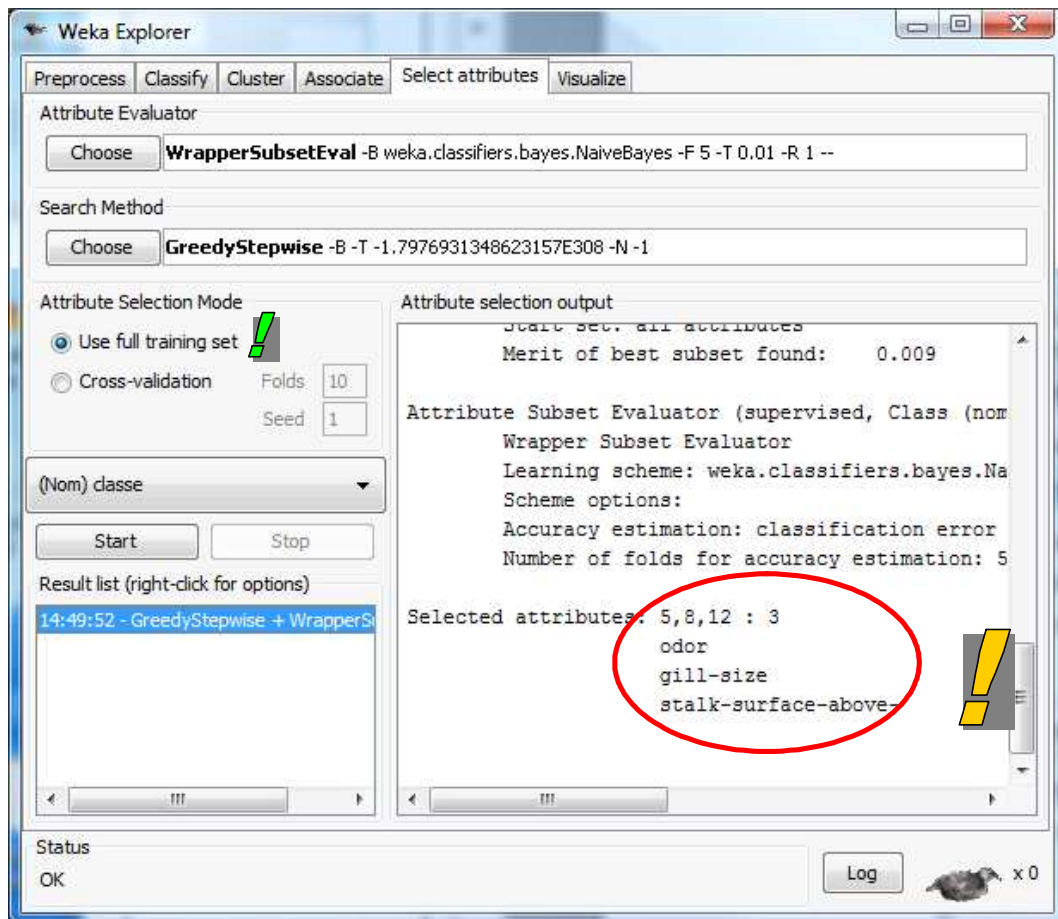
Passons maintenant au mode d'exploration des solutions SEARCH METHOD. Nous souhaitons implémenter la recherche gloutonne BACKWARD. Pour cela, nous choisissons GREEDY STEPWISE.



Nous le paramétrons, toujours en cliquant sur la barre de description, de manière à ce qu'une recherche BACKWARD soit effectuée (SEARCHBACKWARDS = TRUE).



Reste à lancer la procédure. Pour cela, nous actionnons le bouton START, non sans avoir remarqué qu'elle utilise la totalité des données pour réaliser les opérations (ATTRIBUTE SELECTION MODE = USE FULLE TRAINING SET).



Au final, les variables `odor`, `gill-size` et `stalk-surface-above` sont sélectionnées pour la prédiction.

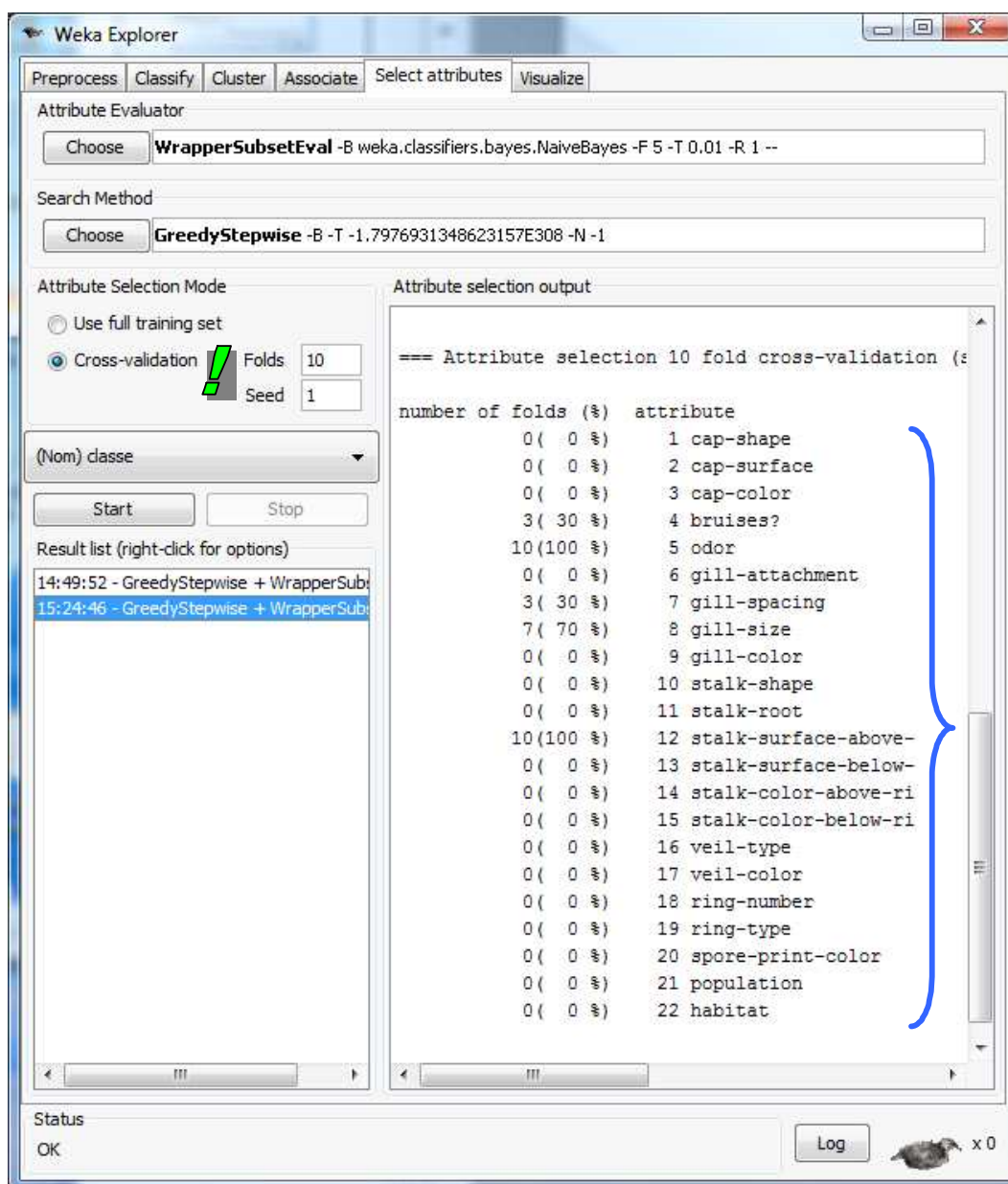
4.3 Pondération des attributs par validation croisée

Nous avons mis en place avec Weka l'équivalent de la procédure implémentée dans Knime, la même que nous avons décrite précédemment dans SIPINA et R (<http://tutoriels-data-mining.blogspot.com/2009/05/strategie-wrapper-pour-la-selection-de.html>).

Mais Weka intègre une fonctionnalité supplémentaire intéressante.

Revenons un peu sur l'algorithme de recherche. Il repose sur des optimisations qui s'enchaînent à qui mieux-mieux. La solution obtenue est très fortement dépendante de l'échantillon utilisé. Prenons un exemple simple pour appréhender l'idée. X_1 et X_2 sont deux variables candidates qui ont un pouvoir prédictif équivalent, elles sont redondantes. X_1 peut être choisi, et masquer X_2 , à la faveur d'un échantillon favorable. Sur un autre échantillon, il peut en être autrement. Pour dépasser cet écueil, Weka propose d'encapsuler la sélection dans une validation croisée. Si vraiment X_1 domine X_2 , il devrait toujours être sélectionné quel que soit l'échantillon utilisé.

Nous choisissons l'option ATTRIBUTE SELECTION MODE = CROSS-VALIDATION (FOLDS = 10). Et nous cliquons de nouveau sur START.

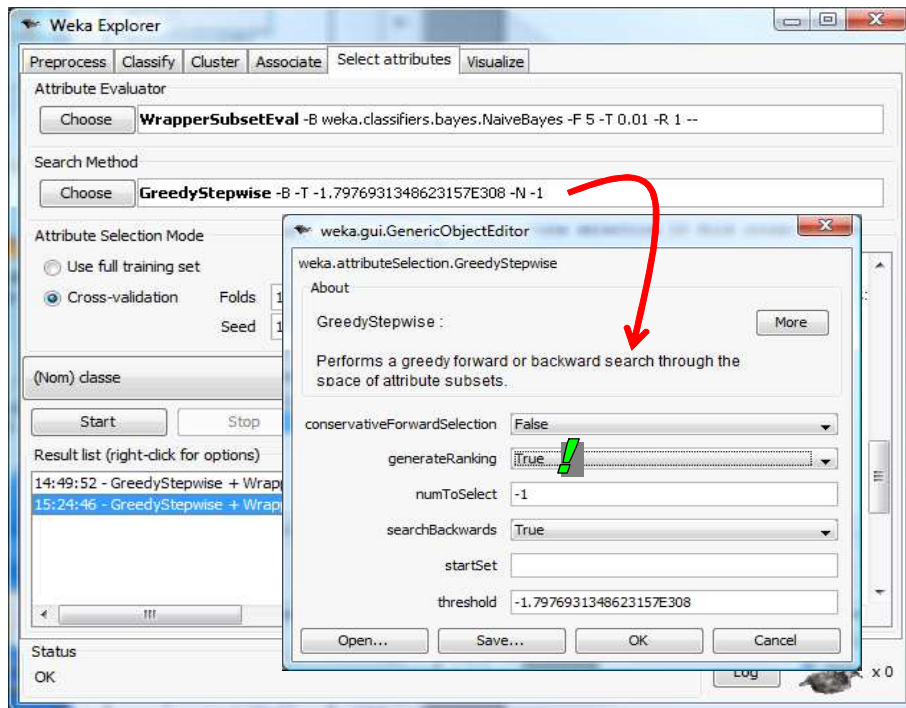


5 variables prédictives se démarquent cette fois. Sur les 10 sessions réalisées, 5 variables ont été incluses au moins une fois dans les sous-ensembles « optimaux » : **bruises** (3 fois), **odor** (10), **gill-spacing** (3), **gill-size** (7) et **stalk-surface-above** (10). Cela relativise le résultat à 3 seules variables mis en avant dans la section précédente.

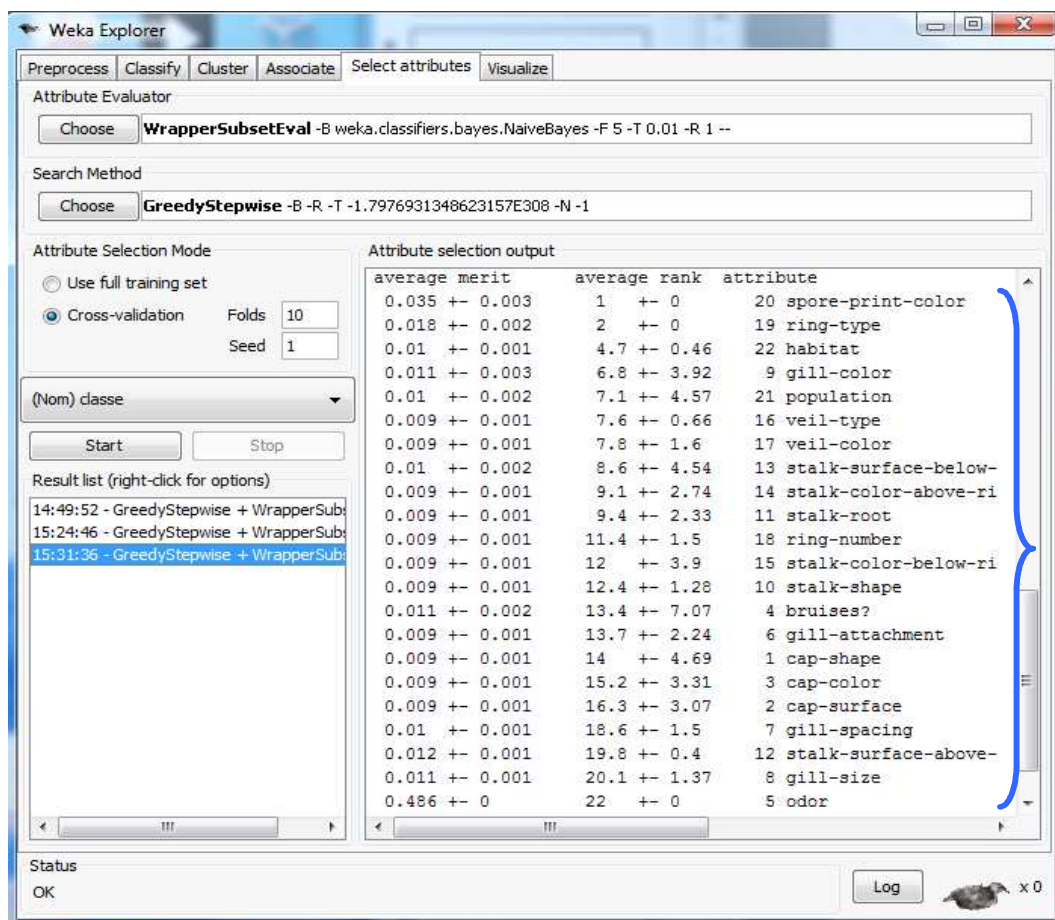
4.4 « Ranking » des attributs

Weka, qui décidément nous cache des trésors insoupçonnés, nous propose une fonctionnalité supplémentaire : le « ranking » des descripteurs. Si l'on s'en tient à la procédure BACKWARD, la première variable retirée de l'ensemble des descripteurs candidats est celle qui apporte le moins d'information ; puis la seconde variable retirée est la deuxième moins importante ; etc. Weka sait tenir compte de cela pour mesurer l'impact de chaque attribut. En encapsulant le dispositif dans une validation croisée, nous obtenons une pondération des attributs plus riche que la simple présence/absence dans la solution finalement obtenue.

Nous revenons sur SEARCH METHOD, nous cliquons sur la barre de description. Dans la boîte de paramétrage, nous choisissons l'option GENERATE RANKING = TRUE.



Il ne nous reste plus qu'à cliquer de nouveau sur le bouton START.



Average rank. Nous nous focalisons sur la colonne « average rank ». Comme nous avons réalisé une recherche backward, nous devons lire les résultats de bas en haut, les variables les plus intéressantes ont été retirées en dernier dans le processus d'exploration.

Nous constatons que la variable la plus importante est **odor**. Elle a toujours été retirée en dernier, l'écart type du rang moyen est nul. Ensuite viennent, si l'on s'en tient aux 3 autres meilleures variables par ordre d'importance : **gill-size**, **stalk-surface-above** et **gill-spacing**. A l'opposé, on notera que **spore-print-color** et **ring-type** sont les deux premières systématiquement exclues.

Average Merit. La colonne « average merit » semble indiquer le taux d'erreur en généralisation du modèle lorsque nous retirons la variable. Si l'on considère « odor », lorsqu'elle est retirée, puisque c'est la dernière variable présente, nous obtenons le taux d'erreur du classifieur par défaut qui est de 48.6%. Si nous retirons « gill-size », le taux d'erreur du sous-ensemble de variables restantes serait de 1.1%.

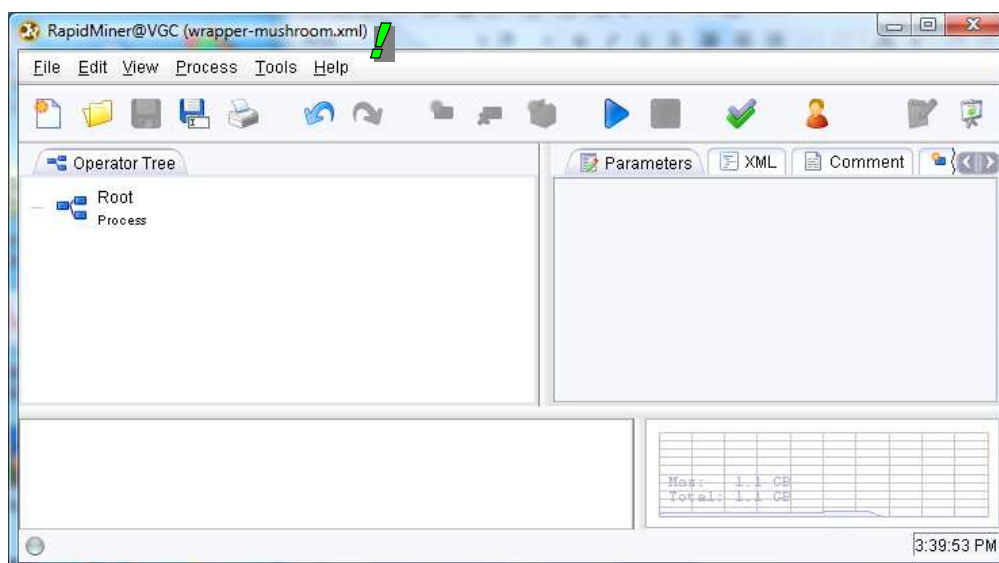
C'est une indication comme une autre. Il reste que le mode de calcul du « merit » paraît quand même assez étrange. En effet, lors du retrait de « gill-size », le nombre de variables restantes dans le sous-ensemble solution peut varier d'une session à l'autre. Mesurer un taux d'erreur moyen n'est pas aisé.

5 Stratégie wrapper avec RapidMiner

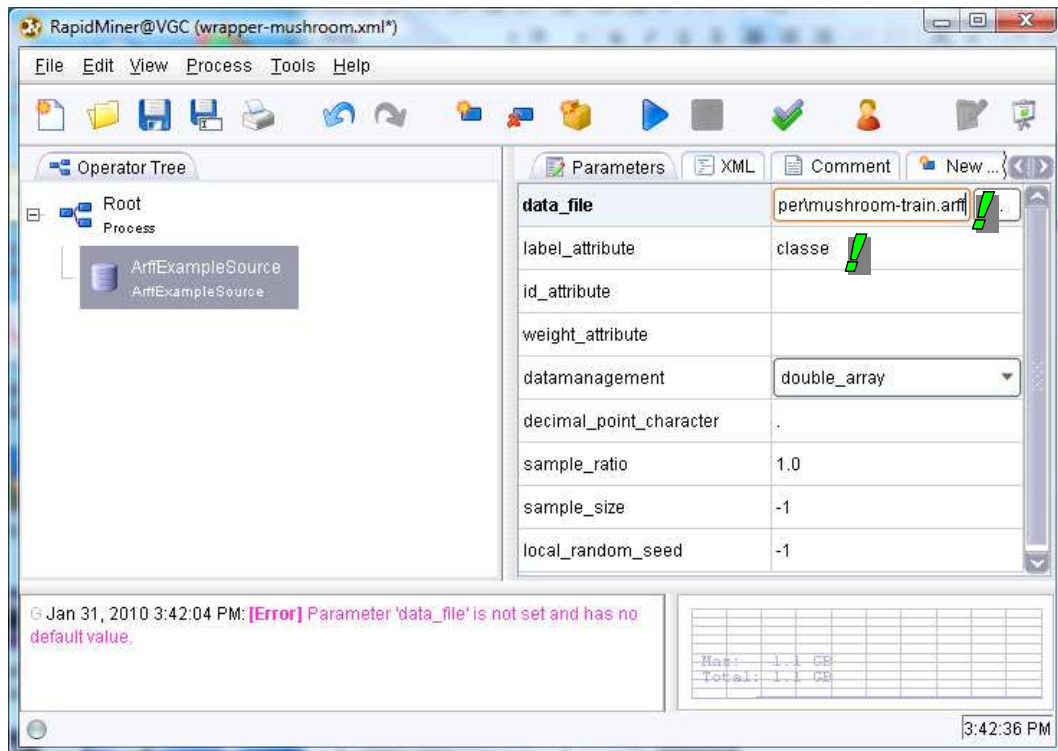
Tout comme Weka, RapidMiner ne permet pas de déployer le modèle avec les variables sélectionnées sur un fichier test distinct contenant la totalité des variables initiales. Tout comme Weka toujours, il sait encapsuler le processus de sélection wrapper dans une validation croisée pour obtenir, non pas une solution unique, mais plutôt un système de pondération des descripteurs permettant de distinguer ceux qui peuvent jouer un rôle important dans la prédiction.

5.1 Wrapper avec RapidMiner

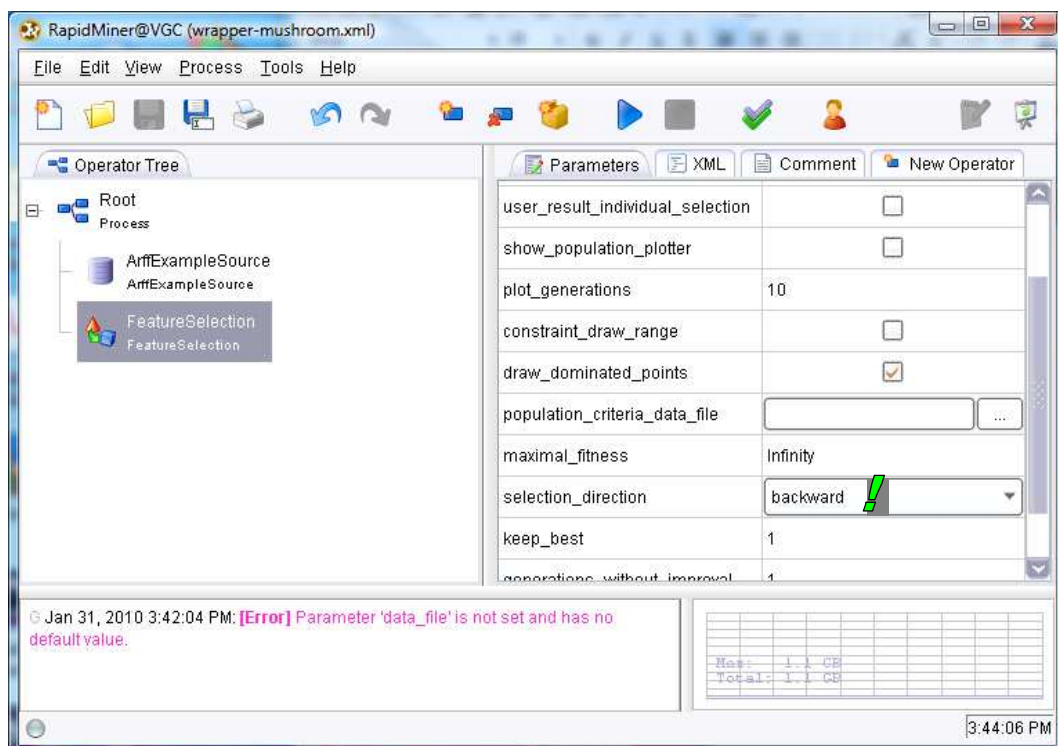
Après avoir démarré RapidMiner, nous créons un nouveau diagramme en actionnant le menu FILE / NEW. Nous le sauvegardons immédiatement sous le nom « wrapper-mushroom.xml ».



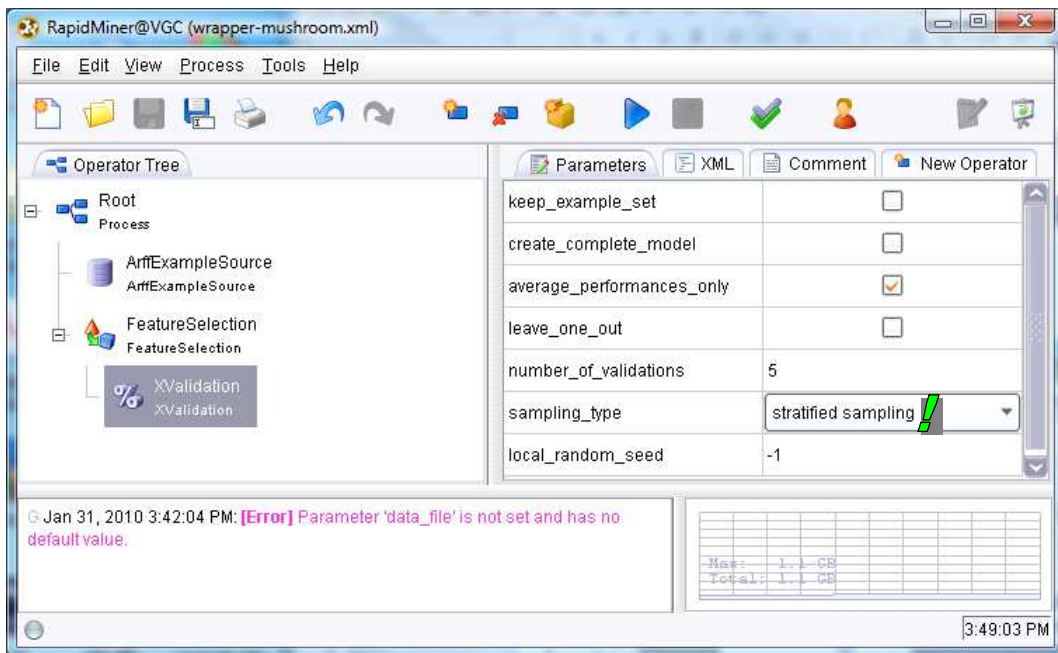
Première étape toujours dans ce type de logiciel, nous devons charger les données. Nous insérons le composant ARFFEXAMPLESOURCE (menu *NEW OPERATOR / IO / EXAMPLES*). Nous le paramétrons en spécifiant le nom du fichier « mushroom-train.arff » et en indiquant la variable cible (*LABEL_ATTRIBUTE = CLASSE*).



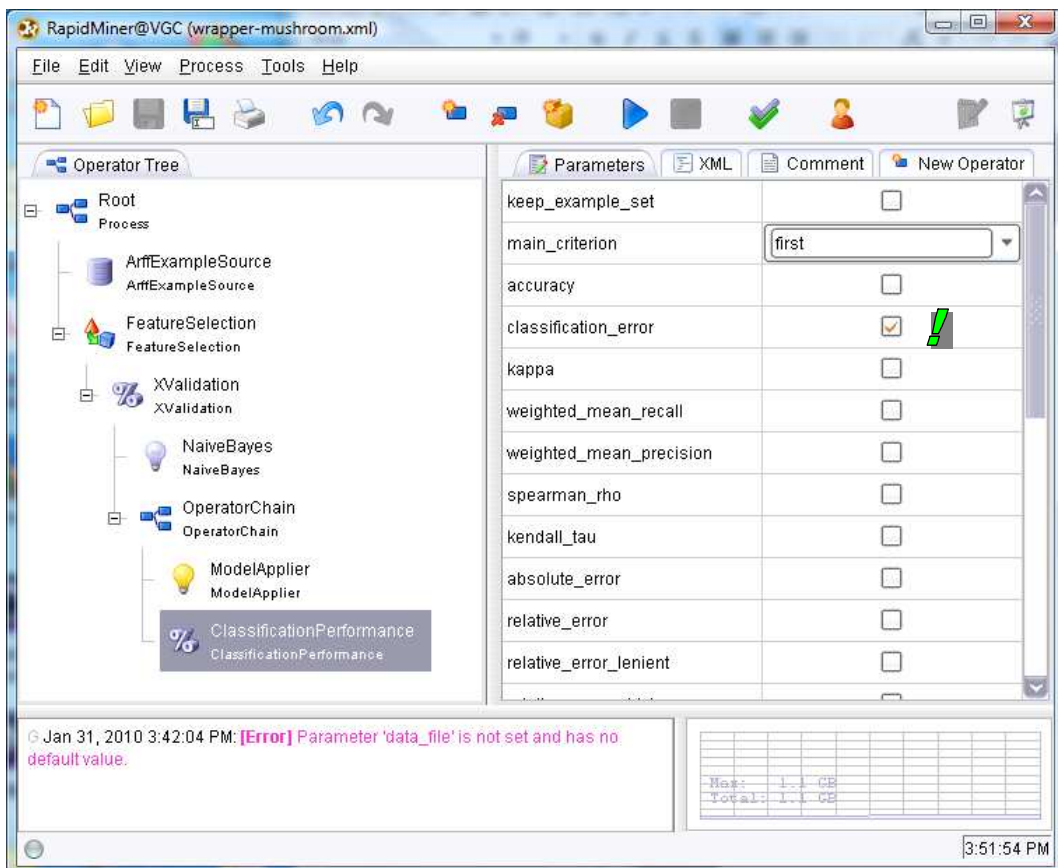
Puis nous insérons le composant FEATURE SELECTION (*PREPROCESSING / ATTRIBUTES / SELECTION*). Nous indiquons la stratégie de recherche (*SELECTION_DIRECTION = BACKWARD*).



Le mode d'évaluation des performances est la validation croisée. Nous insérons le composant XVALIDATION (VALIDATION). Nous demandons une subdivision en 5 blocs, lesquels sont constitués par échantillonnage stratifié c.-à-d. les proportions des classes sont respectées dans les blocs).

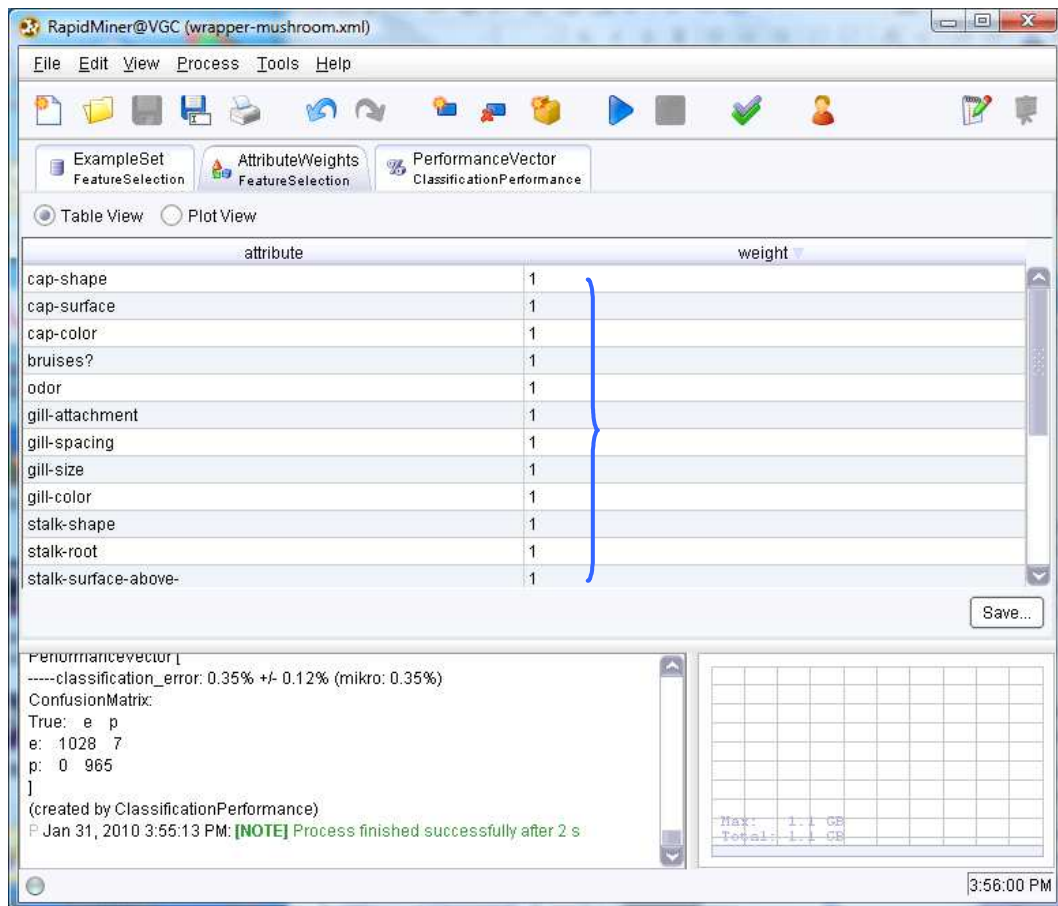


Ici intervient l'algorithme d'apprentissage NAIVE BAYES (LEARNER / SUPERVISED / BAYES). Puis nous définissons l'enchaînement prédiction (MODEL APPLIER) et mesure du taux d'erreur (CLASSIFICATION PERFORMANCE) à l'aide d'un OPERATOR CHAIN.



CLASSIFICATION ERROR est la mesure d'évaluation utilisée.

Il ne reste plus qu'à lancer le processus en cliquant sur le bouton RUN (bleu) dans la barre d'outils. Nous obtenons la matrice de confusion de la solution retenue (onglet PERFORMANCE VECTOR) et surtout la liste des variables présentes dans ATTRIBUTE WEIGHTS.



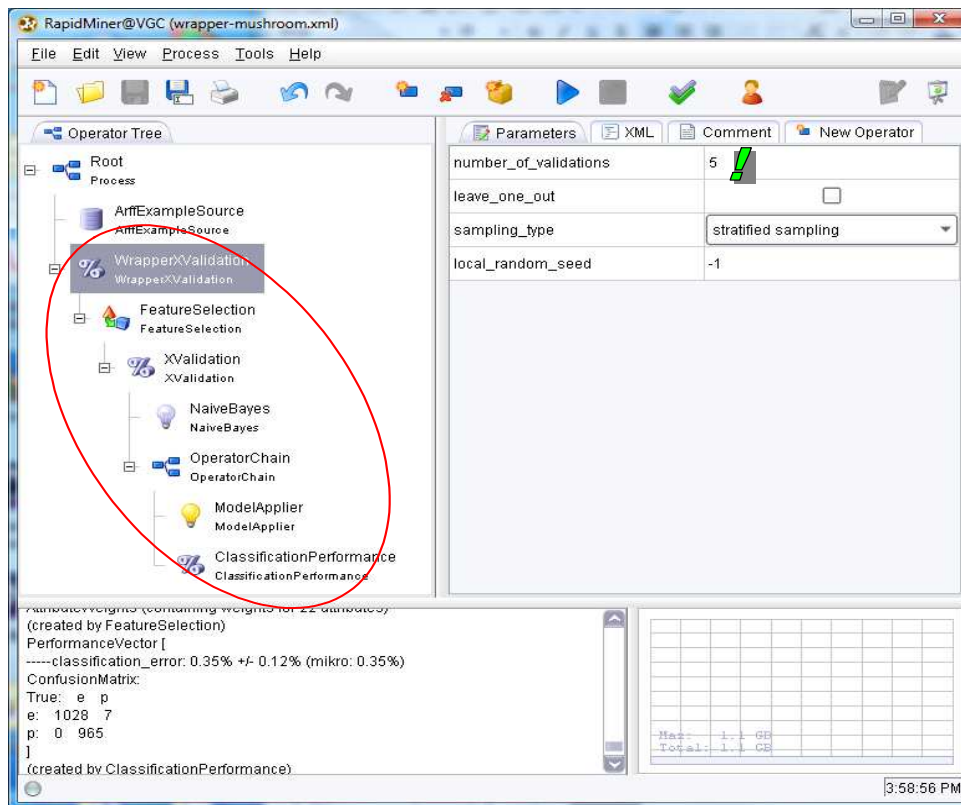
Manifestement, RapidMiner n'applique pas le principe de parcimonie puisque très peu de variables seulement sont exclues de l'ensemble des variables candidates. Il semble sélectionner la première solution trouvée minimisant le taux d'erreur, la plus complexe en l'occurrence puisque nous sommes dans une recherche backward.

5.2 Pondération des attributs selon leur pertinence

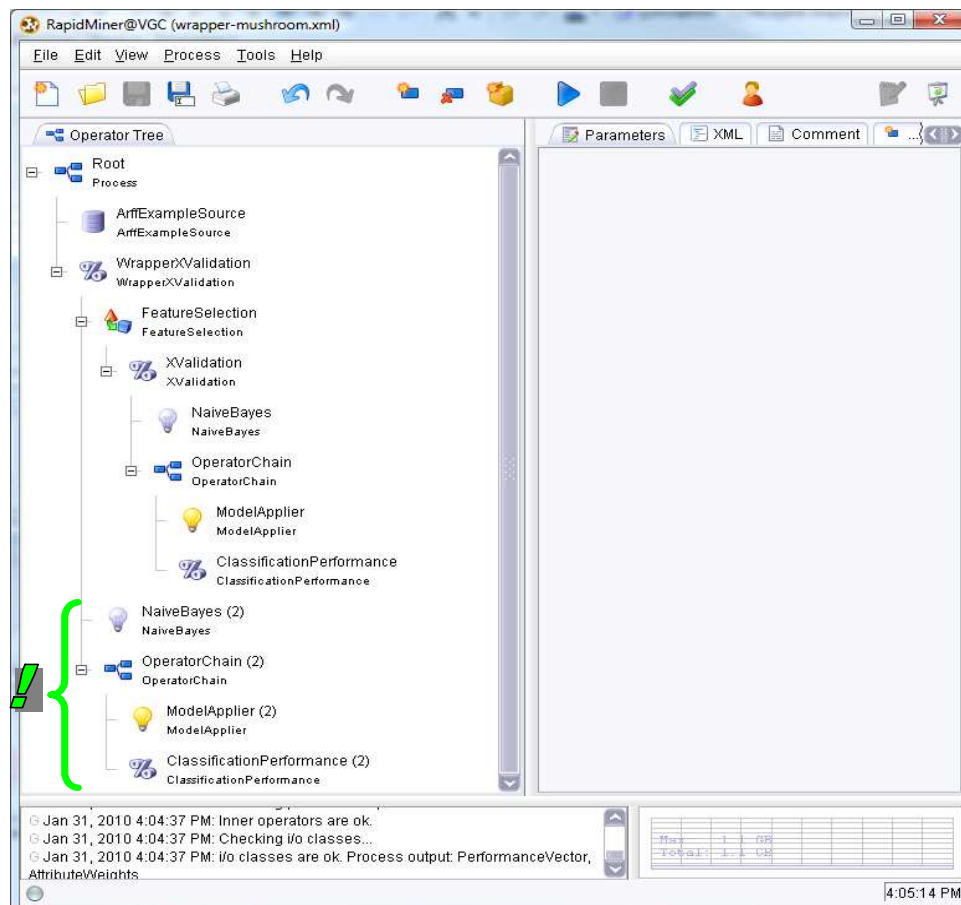
Nous le savons maintenant, pour donner une meilleure assise aux résultats. Il est plus intéressant d'aboutir à un système de pondération des variables plutôt qu'à une solution unique parfois exagérément dépendante de l'échantillon utilisé.

Nous revenons dans la fenêtre d'édition du diagramme. Nous insérons à la suite de ARFF EXAMPLE SOURCE le composant WRAPPER XVALIDATION (VALIDATION). Il s'agit d'une validation croisée qui encapsule le processus de sélection, nous demandons une subdivision en 5 blocs.

Par glisser – déposer, nous y incorporons la branche FEATURE SELECTION.



Enfin, nous complétons de nouveau le diagramme avec l'enchaînement apprentissage avec le modèle bayésien naïf + prédiction + évaluation.



Nous cliquons sur le bouton RUN. Les calculs sont forcément plus longs. Dans la validation croisée globale pour évaluer les solutions proposées est intégrée une validation croisée pour évaluer le rôle de chaque variable prédictive. Nous activons l'onglet ATTRIBUTE WEIGHTS.

attribute	weight
cap-surface	1
cap-color	1
odor	1
gill-attachment	1
gill-spacing	1
gill-size	1
gill-color	1
stalk-shape	1
stalk-root	1
stalk-surface-above-	1
veil-type	1
veil-color	1
spore-print-color	1
population	1
stalk-surface-below-	0.800
stalk-color-above-ri	0.800
ring-type	0.800
...	0.800

AttributeWeights (containing weights for 22 attributes)
(created by WrapperXValidation)
Jan 31, 2010 4:07:52 PM: [NOTE] Process finished successfully after 16 s

La solution précédente est relativisée. Mais, manifestement, le nombre de variables sélectionnées est surestimé. **Compte tenu de son comportement, il semble plus approprié d'utiliser la recherche FORWARD dans RapidMiner car il privilégie la première solution trouvée.**

6 Conclusion

La stratégie wrapper utilise explicitement un critère de précision pour déterminer le sous-ensemble de variables prédictives optimal. De plus, il ne s'appuie pas sur les particularités de l'algorithme d'apprentissage lors de l'exploration des solutions. Voilà deux caractéristiques qui en font un outil privilégié dans un processus de sélection de variables.

En revanche, comme nous avons pu le constater dans ce didacticiel, elle est très gourmande en ressources de calcul, avec des validations croisées qui s'enchaînent et se superposent. Sa mise en œuvre lorsque nous traitons de très grandes bases de données, tant en nombre de variables que d'observations, est coûteuse. Surtout lorsque nous utilisons une méthode d'apprentissage peu adaptée au dispositif : coupler la stratégie wrapper avec un réseau de neurones (perceptron multi-couches) par exemple n'est pas très approprié, le faire sur une base comportant des centaines de descripteurs et des centaines de milliers d'observations serait carrément masochiste.