1 Introduction

Description des formats de fichiers manipulés par SIPINA.

L'accès aux données est la première étape du processus Data Mining. Lorsque nous souhaitons initier un traitement à l'aide d'un logiciel quelconque, la première question que nous nous posons est systématiquement « comment dois-je procéder pour importer mes données ? ». C'est donc un critère important pour juger de la qualité d'un logiciel. Nous pourrons fatalement moins consacrer de temps à l'exploration et l'interprétation lorsque la lecture et la manipulation des données deviennent des opérations difficiles et fastidieuses.

Deux points de vue permettent de positionner les formats de fichier : la souplesse et la performance.

On entend par souplesse la capacité à manipuler facilement le fichier, même en dehors du logiciel spécialisé. Le fichier texte est le format à privilégier dans ce contexte. Nous pouvons l'ouvrir, le modifier et l'enregistrer dans n'importe quel éditeur de texte. De plus, tout logiciel destiné à la manipulation de données (tableur, système de gestion de base de données entre autres) sait appréhender ce type de fichier.

La performance revient surtout à évaluer la rapidité des accès et, dans une moindre mesure, l'occupation disque. Ce critère est surtout important lorsque nous avons à manipuler de très grands fichiers. En effet, Sipina réalisant les traitements en mémoire centrale, comme la majorité des logiciels de Data Mining libres d'ailleurs, il n'est pas nécessaire de répéter fréquemment les opérations de chargement et de sauvegarde.

Lorsque la base est d'une taille raisonnable, force est de constater que le tableur tient une place à part. Des sondages ont montré qu'un grand nombre de data miner couplent l'utilisation d'Excel avec un logiciel spécialisé[1]. Ce n'est guère étonnant. Excel est un outil très répandu et simple d'utilisation. Et si l'on souhaite se tourner vers les solutions libres, Calc de Open Office peut très bien prendre le relais (http://fr.openoffice.org/). Le tableur répond parfaitement au critère de souplesse. Il présente de plus de bonnes aptitudes : 65535 observations (la première ligne étant réservée aux noms de variables) et 256 colonnes (variables) jusqu'à Office XP, plus encore avec Office 2007. Pourvoir établir une passerelle simple entre un tableur et un logiciel de Data Mining est donc une piste que l'on se doit d'explorer.

Dans ce document, notre premier objectif est de **faire le point sur les différents formats de fichier que gère Sipina**. Il y a les fichiers textes au format simplifié (texte avec séparateur tabulation) ou spécialisé (ARFF de Weka) ; il y a les formats binaires que seul Sipina sait lire, mais qui sont très performants. Nous décrirons également la solution originale que nous avons mis en place pour faciliter le transfert d'Excel

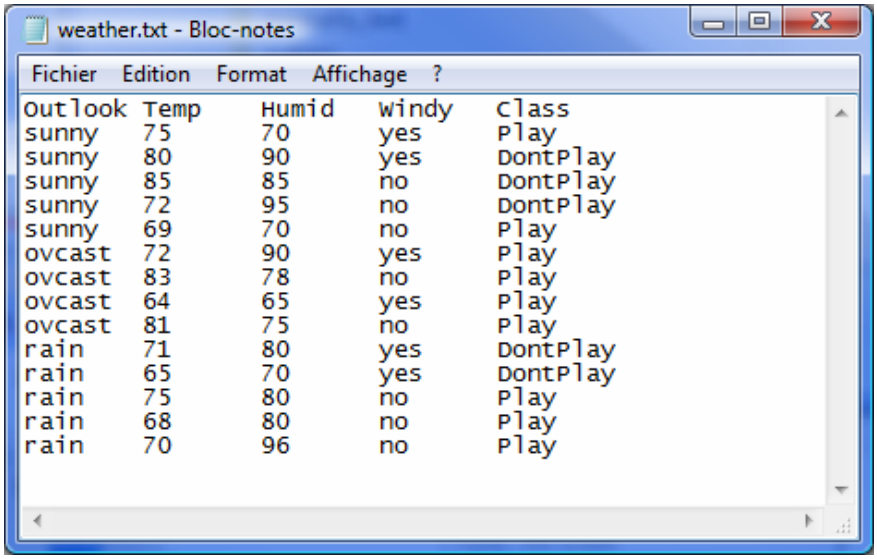
[1] Voir http://www.kdnuggets.com/polls/2008/tools-languages-used-data-cleaning.htm et http://www.kdnuggets.com/polls/2008/data-mining-software-tools-used.htm

vers Sipina. Certaines solutions sont décrites en détail dans des didacticiels accessibles par ailleurs, nous indiquerons les pointeurs adéquats au fil du texte.

L'autre objet de ce didacticiel est de **comparer les performances de Sipina selon ces différents formats, lorsque l'on traite un fichier de grande taille, comportant 4.817.099 observations et 42 variables.**

2 Données

Pour illustrer notre propos, nous utilisons le fichier WEATHER.TXT (Quinlan, 1993 -- http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/weather.txt). Il est au format texte, les colonnes sont séparées par des tabulations.



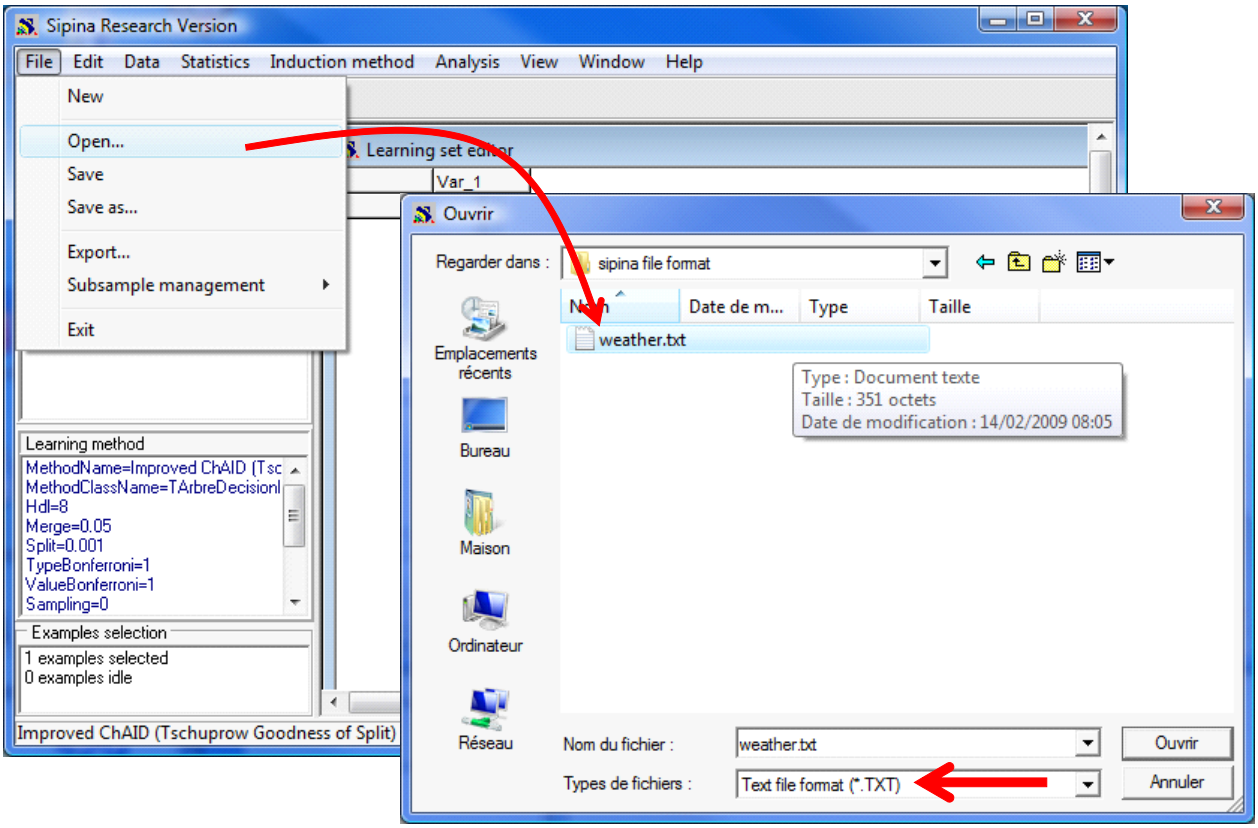
Ce type de format est très fréquemment utilisé, entre autres parce que les tableurs savent le produire facilement. Il s'agit d'une variante du format CSV (Comma-separated values -- http://fr.wikipedia.org/wiki/Comma-separated_values) où, à la place de la virgule, les colonnes sont délimitées par le caractère tabulation (code ASCII 9). Dans la très grande majorité des cas, on considère que la première ligne correspond aux noms des variables, les suivantes sont les observations.

3 Les fichiers au format texte

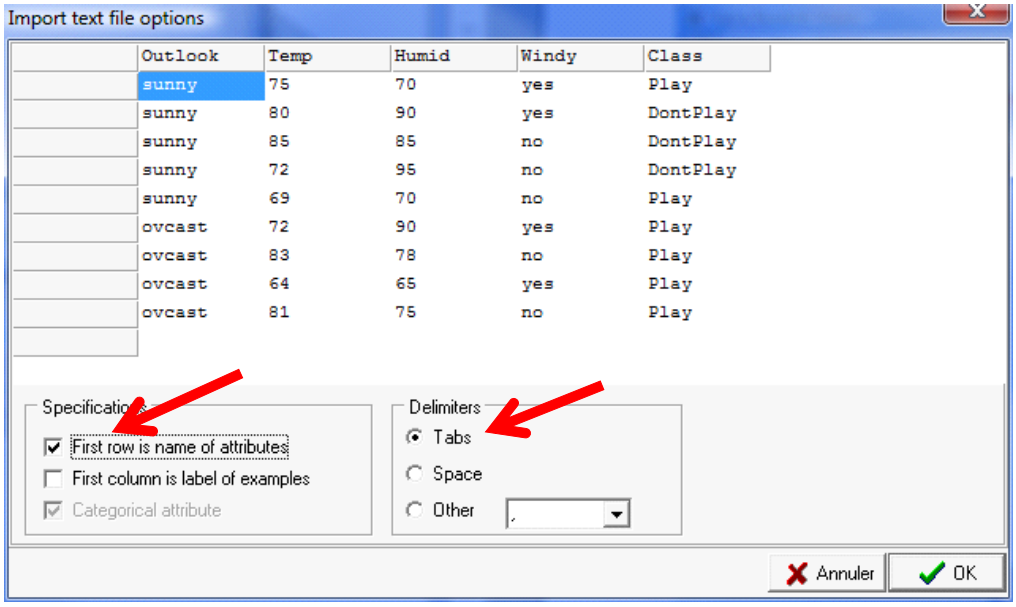
Notre point de départ est le fichier WEATHER.TXT ci-dessus. Nous le chargeons pour le visualiser. Par la suite, nous procéderons de la manière suivante pour chaque format de fichier géré par Sipina : (1) nous exportons le fichier dans un nouveau format ; (2) nous vidons la grille d'affichage ; (3) enfin, nous essayons de relire le fichier pour vérifier son intégrité.

3.1 Fichier texte CSV (*.TXT)

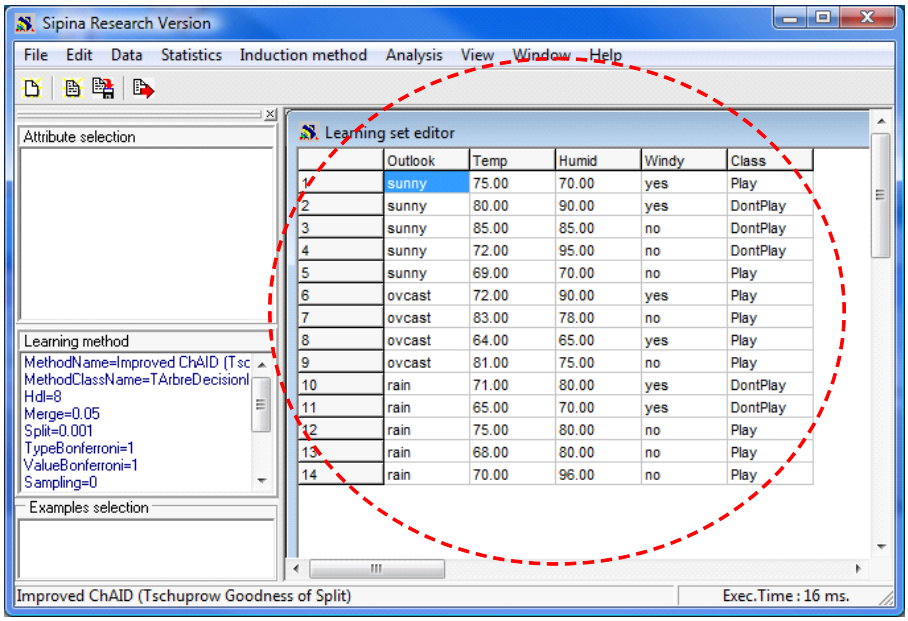
Importer le format TXT. Après avoir démarré Sipina, nous cliquons sur le menu FILE / OPEN. Une boîte de dialogue apparaît, nous choisissons le format TEXT FILE FORMAT (*.TXT) et nous sélectionnons le fichier WEATHER.TXT.



Une boîte de dialogue apparaît. Nous indiquons 2 nouvelles informations : le séparateur de colonnes est le caractère tabulation, la première ligne correspond au nom des variables.



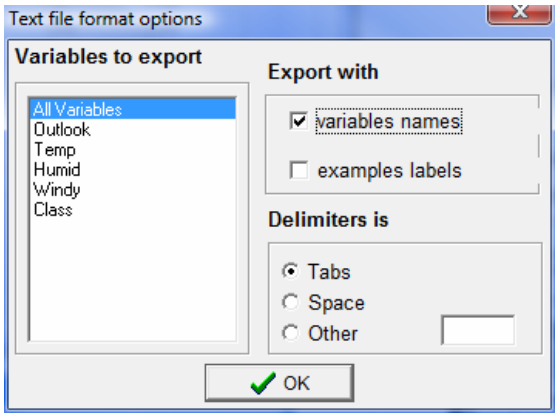
Nous validons, les données sont chargées dans la grille de la fenêtre principale de l'application.



Remarque : règle de typage des variables. Les colonnes peuvent contenir des valeurs numériques ou alphanumériques. Sipina s'appuie sur la première ligne des données (sur la seconde ligne du fichier lorsque la première ligne correspond aux noms de variables) pour détecter le type des variables. Il utilise la règle suivante : si la chaîne représente une valeur numérique, c.-à-d. il peut la transtyper en une valeur réelle, il considère que la variable est quantitative ; si elle est alphabétique, c.-à-d. il ne peut pas la transtyper en valeur numérique, la variable est considérée qualitative. Le type n'est pas immuable, il est possible de le modifier de manière interactive dans le logiciel, après l'importation. Il suffit pour cela d'actionner le menu DATA / DEFINE SPECIFICATIONS.

Remarque : point décimal. Attention, pour Sipina, le point décimal est toujours le caractère « . » quelle que soit la configuration de votre système.

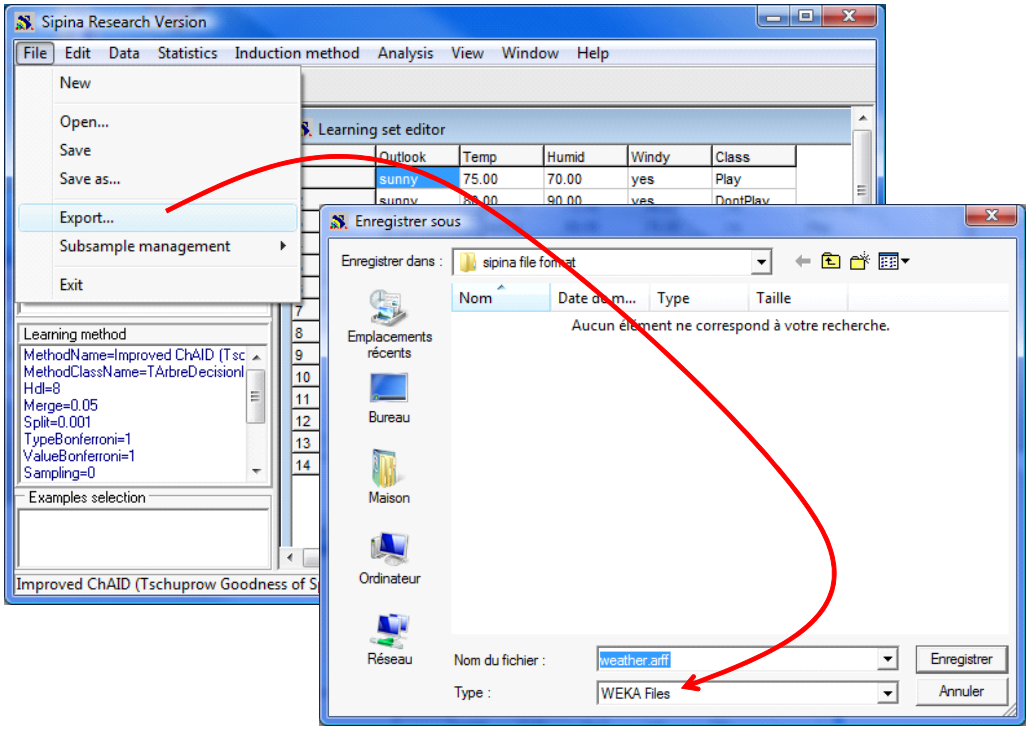
Exporter le format TXT. Pour exporter un ensemble de données au format texte avec séparateur tabulation, nous actionnons la commande FILE / EXPORT. Dans la boîte de saisie du nom de fichier, nous choisissons le format TEXT FILE. Par la suite, une boîte de dialogue apparaît nous demandant d'indiquer la liste des variables à exporter et les spécifications du fichier (intégration du nom des variables, intégration de la colonne des identifiants, le type de séparateur).



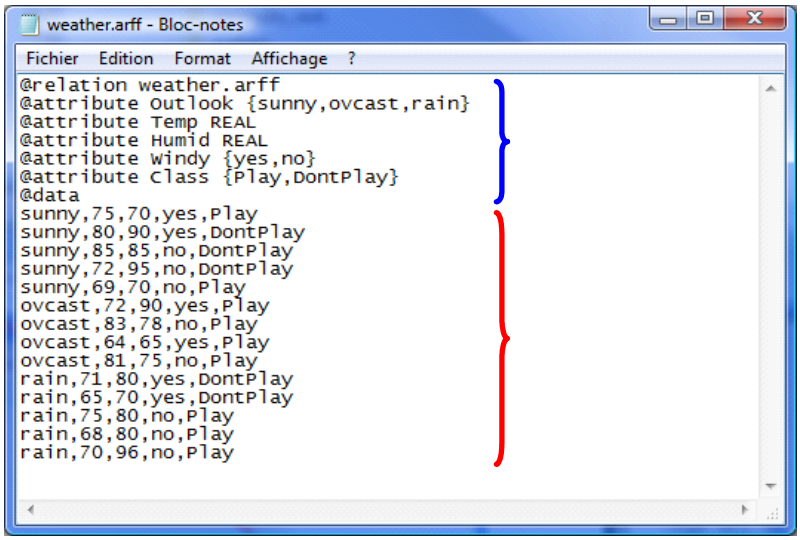
3.2 Fichier texte Weka (*.ARFF)

Le format ARFF de Weka (http://www.cs.waikato.ac.nz/ml/weka/) correspond à un fichier texte. Il est subdivisé en deux parties : la première correspond au dictionnaire de données, la seconde à la description des valeurs.

Exporter au format ARFF. Pour exporter un ensemble de données au format ARFF, nous cliquons sur FILE / EXPORT. Nous choisissons le format WEKA FILES dans la boîte de définition du nom de fichier. Nous introduisons le nom de fichier WEATHER.ARFF.



Le fichier est présent sur le disque maintenant. Nous pouvons l'éditer avec un éditeur de texte, nous observons les éléments suivants.



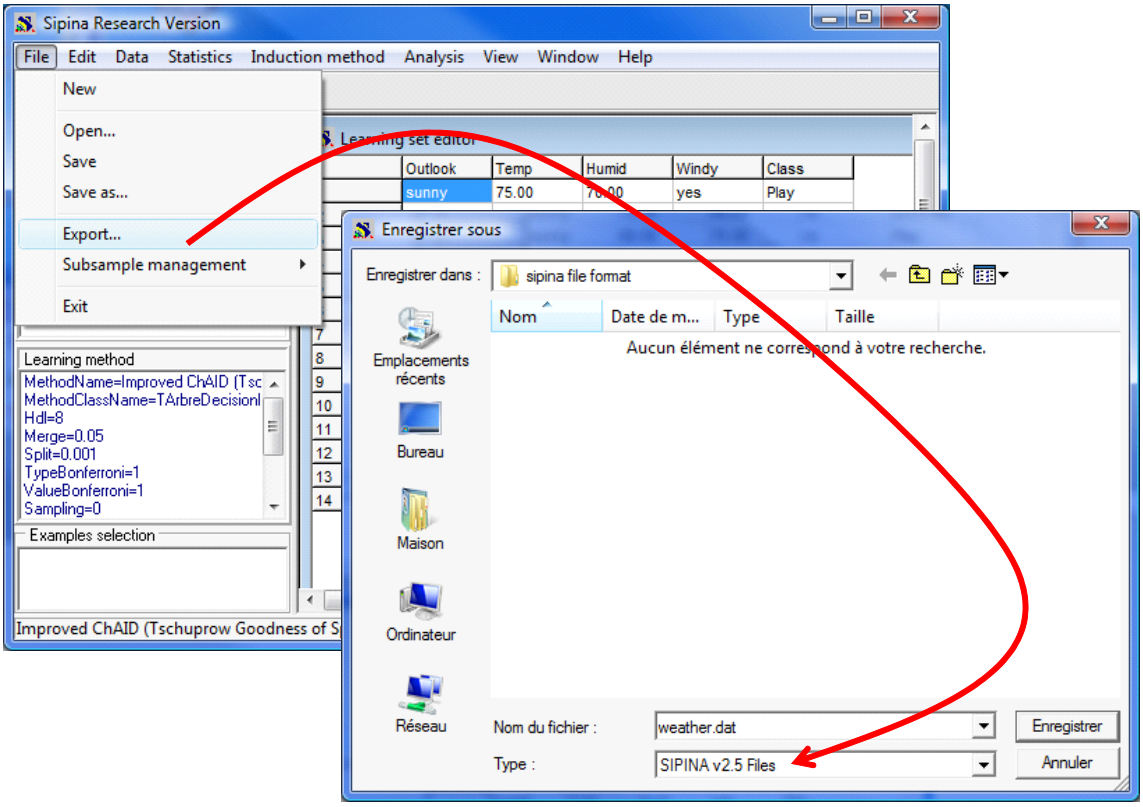
Dans la partie dictionnaire de données, une variable quantitative est définie par le mot clé REAL. Lorsqu'elle est discrète, ses valeurs sont énumérées. La partie données (DATA) est identique au format CSV.

Importer le format ARFF. Sipina sait lire correctement le format ARFF. Il nous faut simplement actionner le menu FILE / OPEN et choisir l'option WEKA FILE FORMAT (*.ARFF). Il n'y aucun paramétrage à faire, les données sont directement chargées et les variables correctement typées.

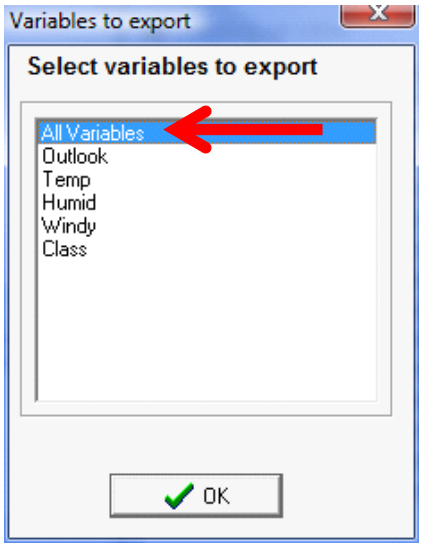
3.3 Fichier texte de Sipina version 2.5

La version précédente de Sipina, la version 2.5, est un programme 16 bits qui n'a jamais pu évoluer, entre autres, à cause de l'utilisation intensive d'une bibliothèque externe propriétaire. Sipina 2.5 s'appuie sur une architecture très similaire à celle de Weka. Mis à part qu'elle utilise deux fichiers séparés pour décrire le dictionnaire (fichier .PAR) et les valeurs (fichier .DAT). Avec le recul, on se rend compte que c'était une très mauvaise idée, les utilisateurs avaient du mal à comprendre la manipulation simultanée de 2 fichiers pour décrire un seul ensemble de données.

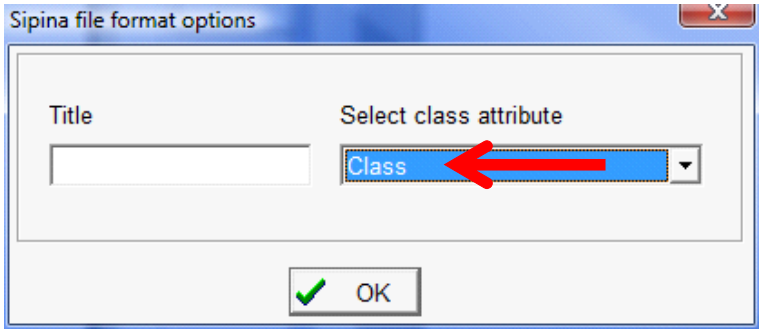
Exporter au format Sipina version 2.5. Pour exporter les données dans ce format, nous actionnons le menu FILE / EXPORT. Dans la boîte de paramétrage, nous choisissons le format SIPINA V2.5 FILES. Nous saisissons le nom de fichier WEATHER.DAT.



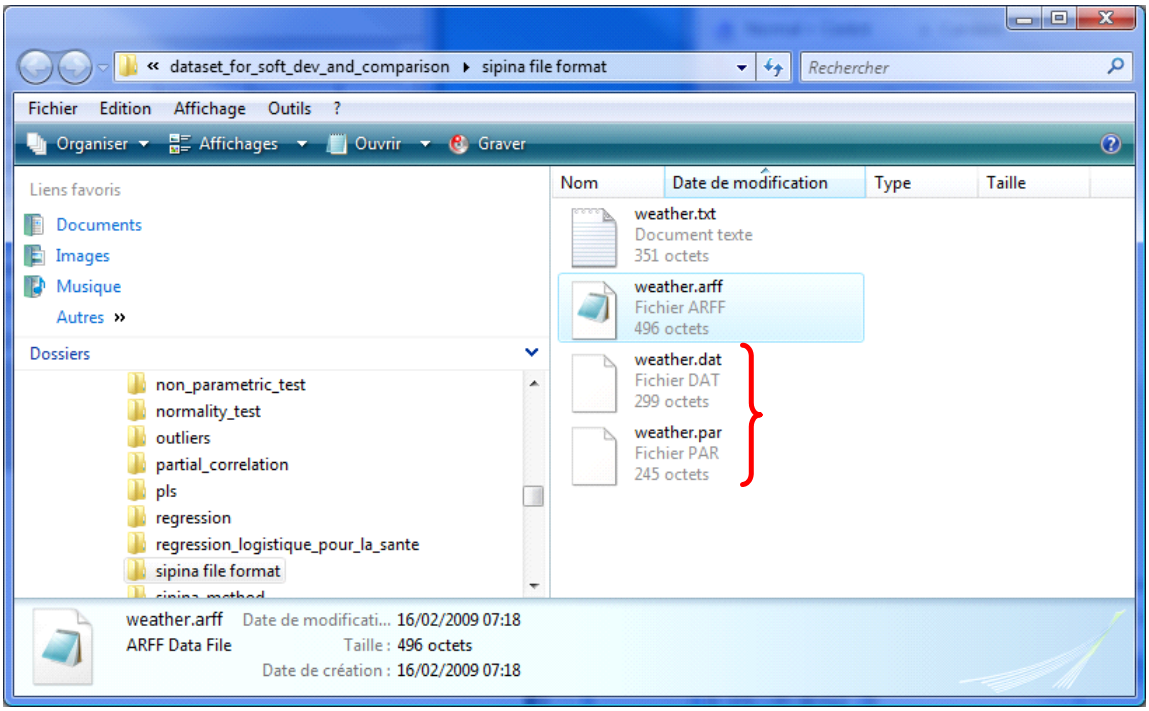
Une seconde boîte apparaît alors. Elle nous demande de spécifier la liste des variables à exporter. Nous les sélectionnons toutes, soit ALL VARIABLES.



Une autre boîte surgit encore. Elle nous demande cette fois-ci de choisir la variable à prédire lors de l'induction. Dans notre cas, il s'agit de la variable CLASS.

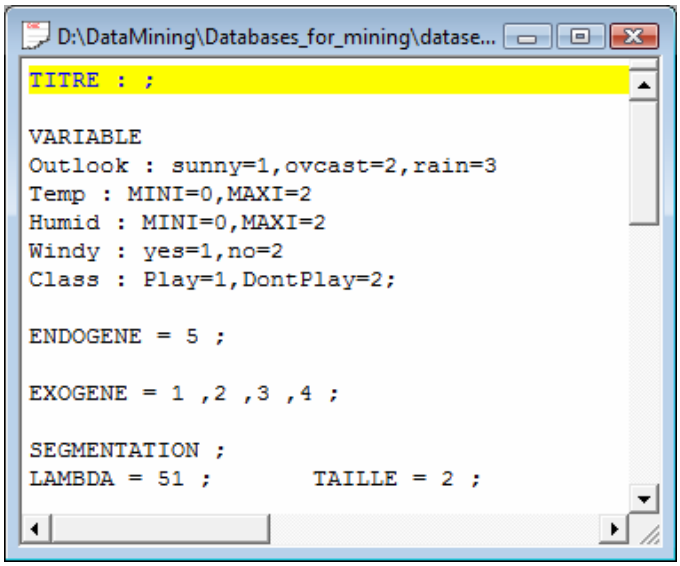


Après avoir validé. Nous observons que 2 nouveaux fichiers ont été générés.

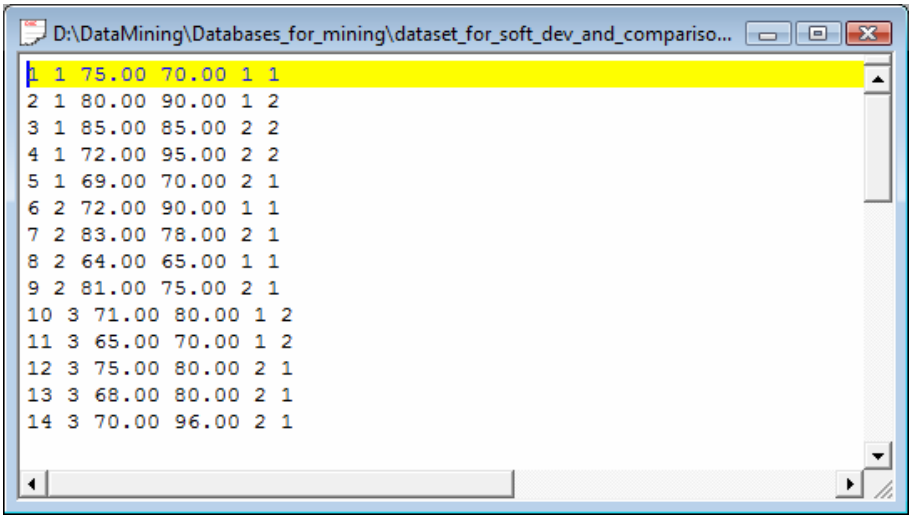


Ouvrons-les tour à tour dans un éditeur de texte.

Dans WEATHER.PAR, nous observons la description des variables. Notons que des codes sont associés aux modalités des variables discrètes. Pour les variables continues, les valeurs associées à MINI et MAXI ne sont pas utilisées. D'autres paramètres relatifs à l'induction sont également décrits.



Dans WEATHER.DAT, nous avons la description des valeurs. A la différence de Weka, Sipina utilise directement les codes décrits dans le dictionnaire de données pour les variables discrètes.



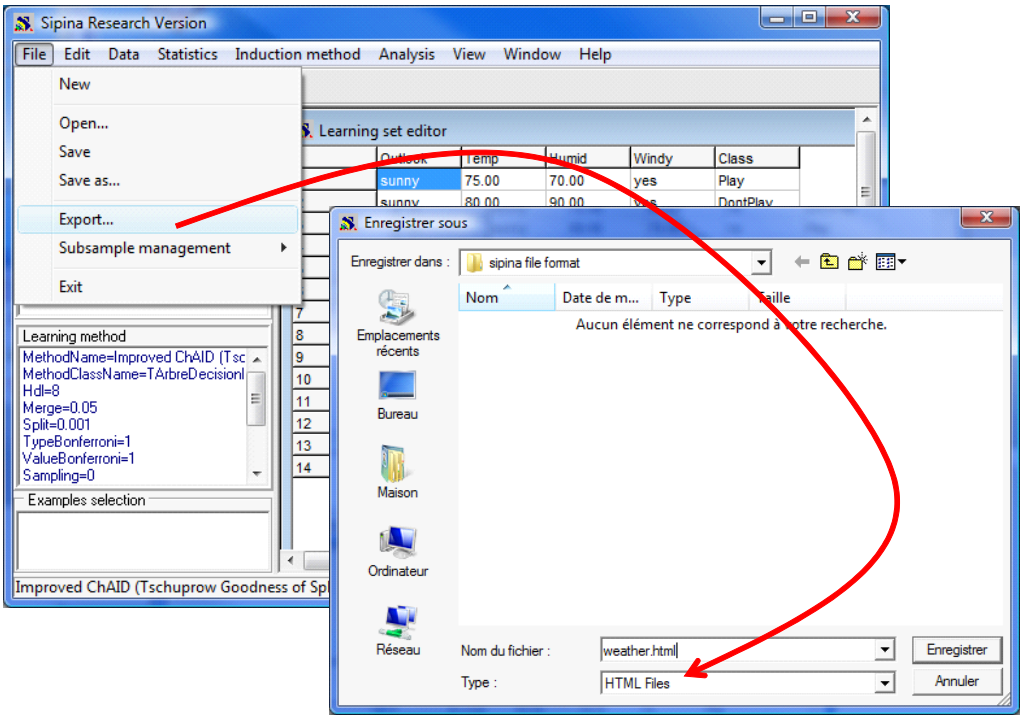
3.4 Fichier HTML

Nous disposons de toutes les commandes nécessaires pour décrire un tableau dans le format HTML, qui est un fichier texte, ne l'oublions pas. L'avantage est double. L'ensemble de données peut être visualisé dans n'importe quel navigateur web. De plus, les tableurs savent lire ce format (les tests ont été réalisés sur EXCEL et OPEN OFFICE CALC).

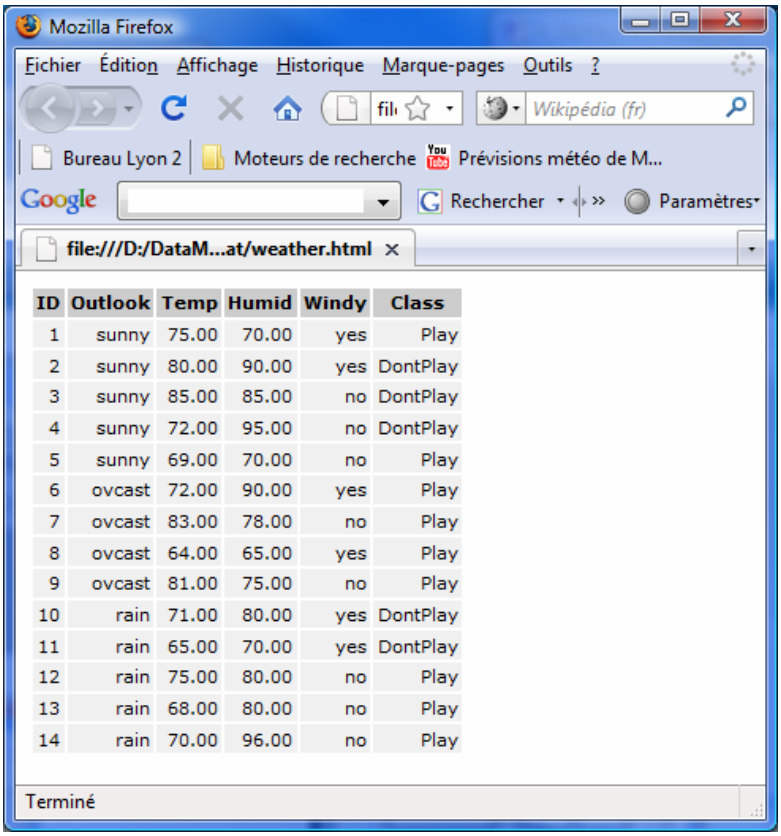
Attention néanmoins, ce format étant assez verbeux, la taille du fichier peut rapidement enfler. En

effet, il faut encadrer chaque cellule des balises idoines.

Exporter au format HTML. Pour exporter au format HTML, nous cliquons sur le menu FILE / EXPORT et nous choisissons le format HTML dans la boîte de paramétrage. Le nom de fichier sera WEATHER.HTML.

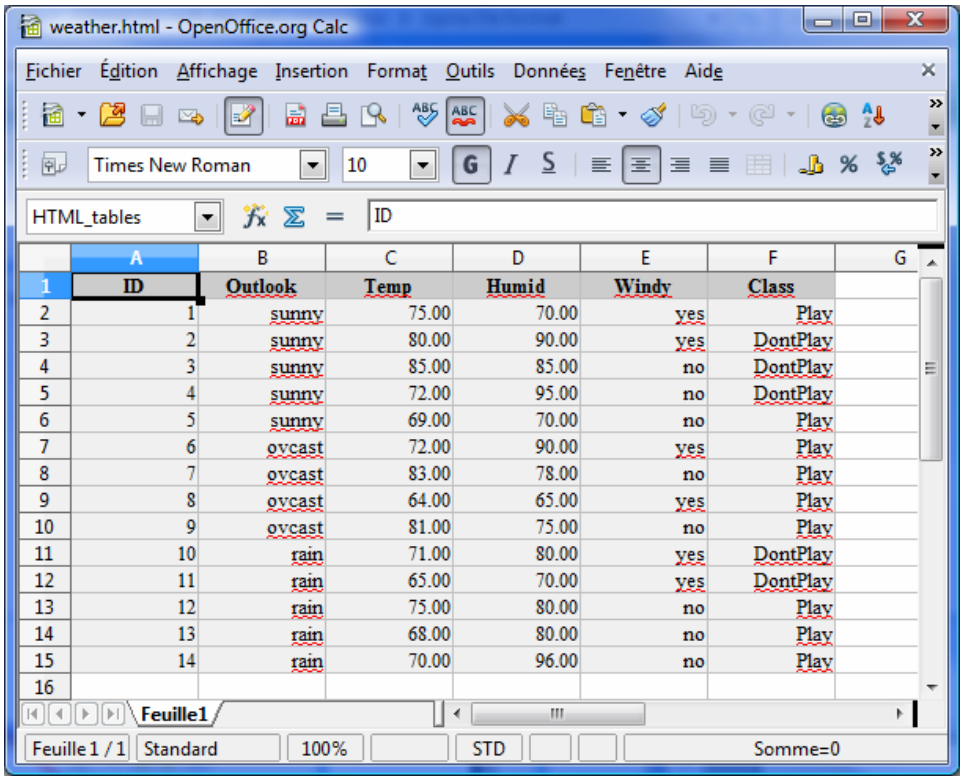


Le fichier peut être visualisé dans un navigateur Internet, ici Firefox.



ID	Outlook	Temp	Humid	Windy	Class
1	sunny	75.00	70.00	yes	Play
2	sunny	80.00	90.00	yes	DontPlay
3	sunny	85.00	85.00	no	DontPlay
4	sunny	72.00	95.00	no	DontPlay
5	sunny	69.00	70.00	no	Play
6	ovcast	72.00	90.00	yes	Play
7	ovcast	83.00	78.00	no	Play
8	ovcast	64.00	65.00	yes	Play
9	ovcast	81.00	75.00	no	Play
10	rain	71.00	80.00	yes	DontPlay
11	rain	65.00	70.00	yes	DontPlay
12	rain	75.00	80.00	no	Play
13	rain	68.00	80.00	no	Play
14	rain	70.00	96.00	no	Play

Il peut être aussi importé directement dans un tableur, ici dans CALC de OPEN OFFICE.



	A	B	C	D	E	F	G
1	ID	Outlook	Temp	Humid	Windy	Class	
2	1	sunny	75.00	70.00	yes	Play	
3	2	sunny	80.00	90.00	yes	DontPlay	
4	3	sunny	85.00	85.00	no	DontPlay	
5	4	sunny	72.00	95.00	no	DontPlay	
6	5	sunny	69.00	70.00	no	Play	
7	6	overcast	72.00	90.00	yes	Play	
8	7	overcast	83.00	78.00	no	Play	
9	8	overcast	64.00	65.00	yes	Play	
10	9	overcast	81.00	75.00	no	Play	
11	10	rain	71.00	80.00	yes	DontPlay	
12	11	rain	65.00	70.00	yes	DontPlay	
13	12	rain	75.00	80.00	no	Play	
14	13	rain	68.00	80.00	no	Play	
15	14	rain	70.00	96.00	no	Play	
16							

4 Les fichiers binaires

Les formats binaires sont propres à Sipina. Dans sa forme la plus fruste (FDM), il s'agit simplement d'un « dump » des vecteurs de valeurs. Il est donc très performant lorsque le fichier comporte un grand nombre d'observations (jusqu'à plusieurs millions) et un nombre moyennement élevé de variables (de l'ordre de quelques centaines). Il est moins adapté en revanche lorsqu'il s'agit de traiter des fichiers atypiques comportant peu d'observations (centaines) et un très grand nombre de variables (dizaines de milliers).

Pour ce qui est de la taille du fichier, elle est exactement identique à l'espace occupée en mémoire. Nous pouvons la résumer succinctement de la manière suivante :

- Chaque valeur occupe 4 octets ;
- A chaque observation est associé un label décrit sur 25 octets ;
- A chaque variable est associé un nom de variable encodé sur 25 octets ;
- Nous devons y ajouter éventuellement la description des modalités des variables discrètes.

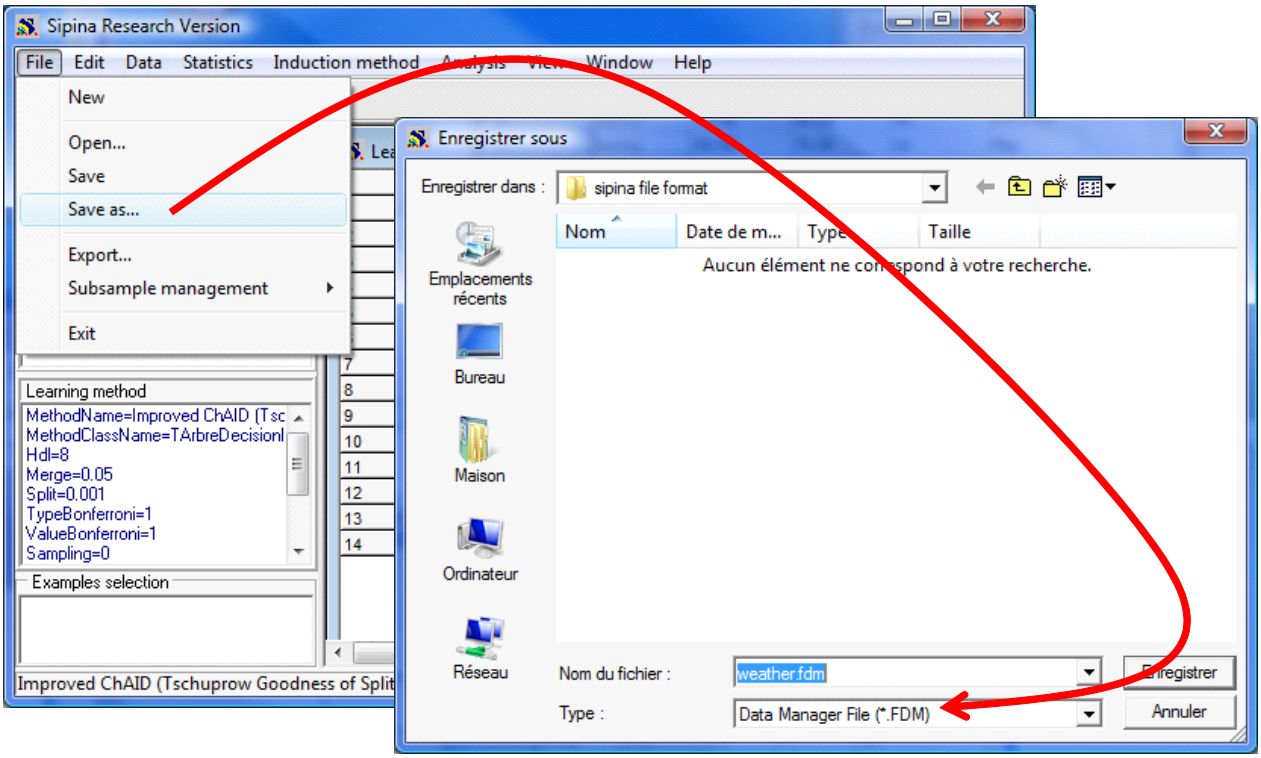
Pour un fichier comportant 100.000 observations et 10 variables continues, la taille du fichier sera :

100.000 x 10 x 4 + 100.000 x 25 + 10 x 25 = 6.500.250 octets

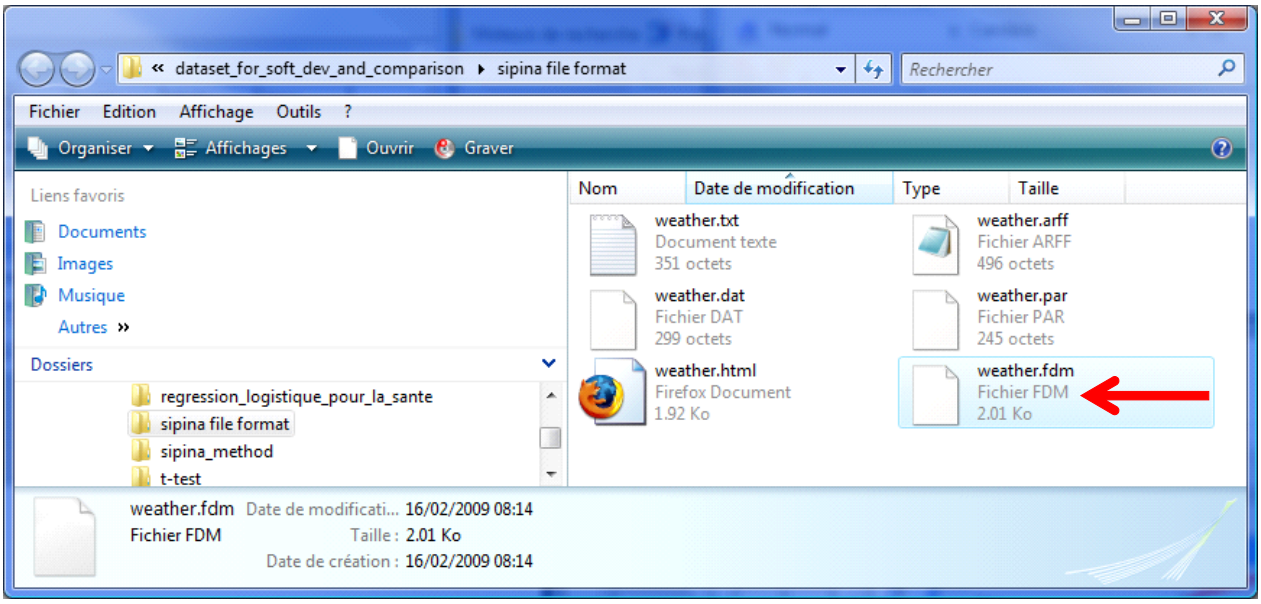
4.1 Le format natif FDM

Exporter un fichier FDM. Pour créer un fichier FDM, nous actionnons le menu FILE / SAVE AS. C'est le

format par défaut (DATA MANAGER FILE), nous spécifions le nom de fichier WEATHER.FDM.



Comme nous pouvons le constater dans notre répertoire de travail, avec 2.01 Ko, la taille du fichier FDM est plus importante que les autres.



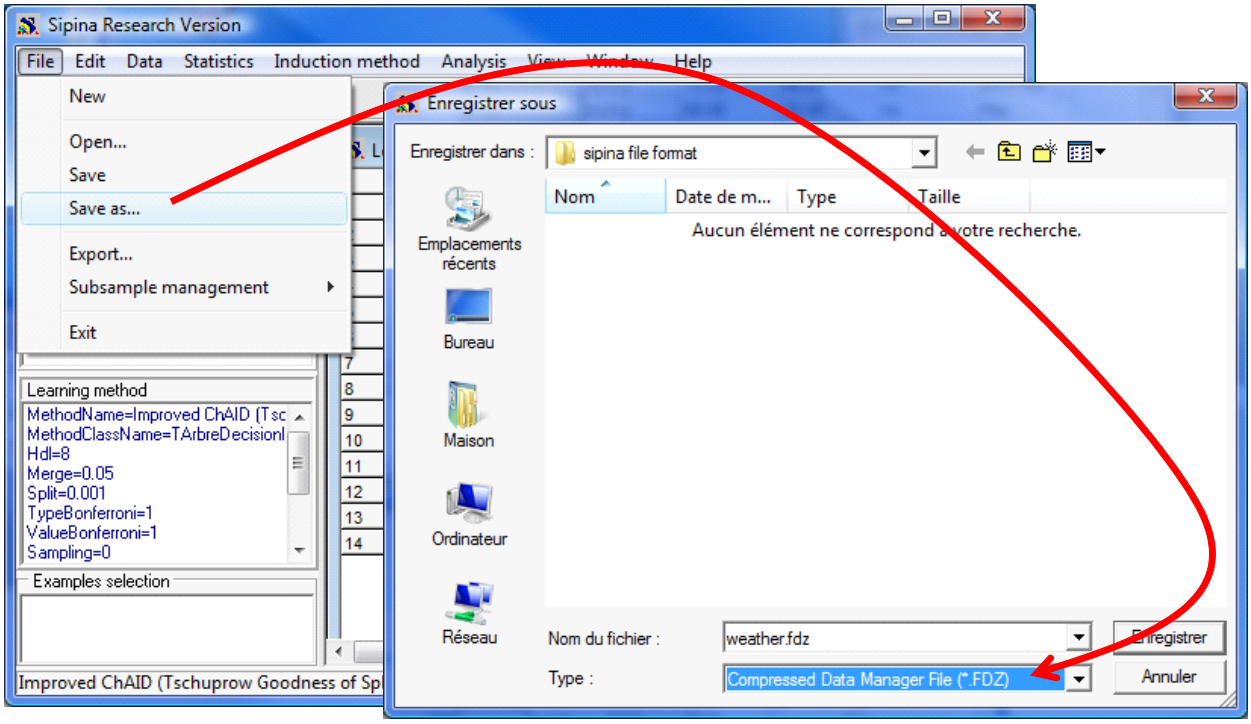
Lire un fichier FDM. Seul Sipina sait lire un fichier FDM, il suffit pour cela d'actionner le menu FILE / OPEN et sélectionner ce format.

Le principal argument du format FDM est sa rapidité, son principal défaut son volume. Pour palier cet inconvénient, nous proposons deux formats binaires compressés dans Sipina.

4.2 Le format compressé FDZ

Le format FDZ est aussi un format binaire. Il s'agit du fichier FDM compressé à l'aide de l'algorithme LZW (Lempel Ziv Welch - http://fr.wikipedia.org/wiki/Lempel-Ziv-Welch). Plutôt que la qualité de la compression, nous avons privilégié la rapidité. La taille du fichier qui en résulte est un peu plus élevée que si nous avions compressé manuellement le fichier à l'aide des outils dédiés tel que 7-ZIP.

Créer un fichier FDZ. Pour créer un fichier au format FDZ, il suffit de cliquer sur le menu FILE /SAVE AS et de choisir ce format. Nous spécifions le nom WEATHER.FDZ.

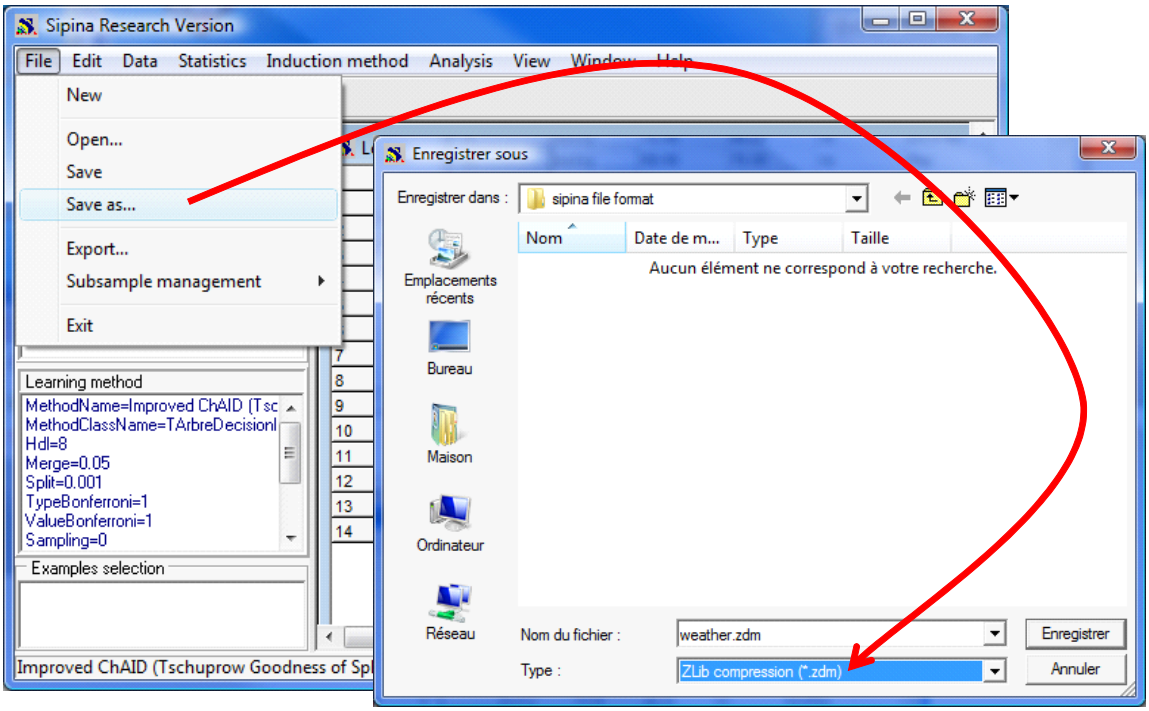


Lire un fichier FDZ. Nous passons par le menu FILE / OPEN pour lire un fichier FDZ.

4.3 Le format compressé ZDM

Le format ZDM est aussi issu de la compression du fichier FDM. Il s'appuie sur la bibliothèque ZLIB de Delphi. A la différence de FDZ, il agit directement au fur et à mesure en mémoire, il ne crée pas de fichier intermédiaire sur le disque durant l'opération. C'est le format qu'il faut privilégier si nous travaillons sur un support aux performances médiocres (une clé USB par exemple).

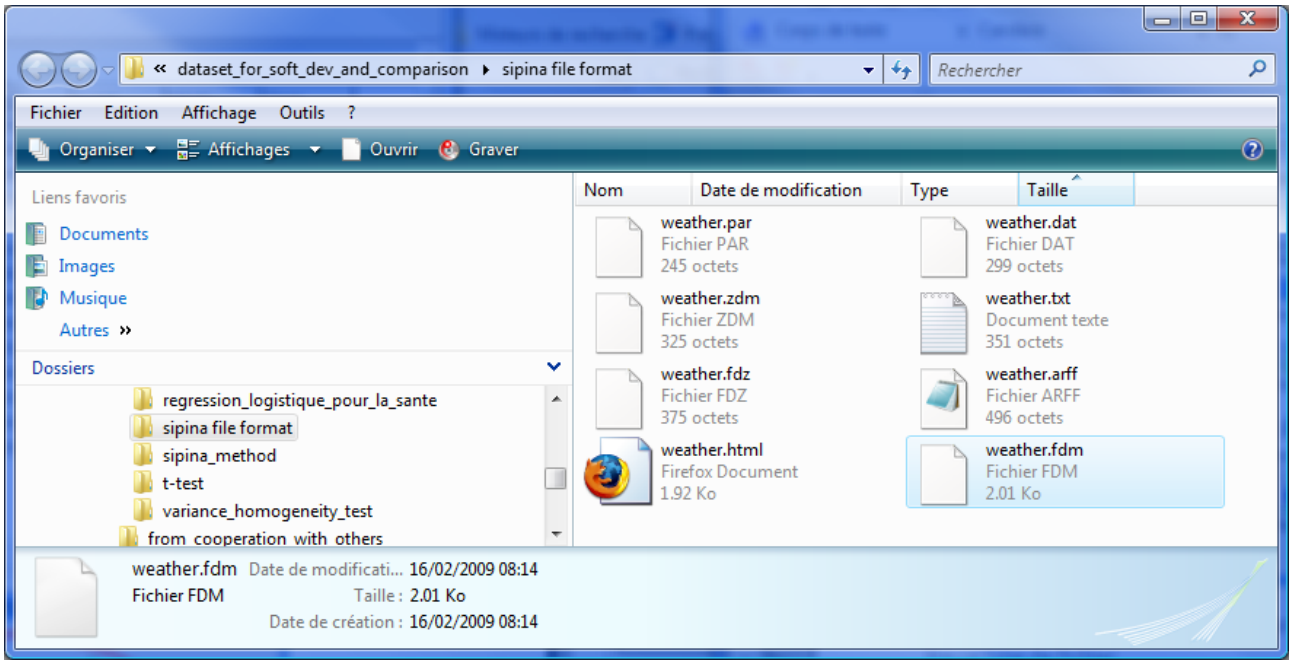
Créer un fichier ZDM. Pour créer un fichier ZDM, nous actionnons le menu FILE / SAVE AS et nous le sélectionnons ce format.



Lire un fichier ZDM. De même que pour les autres formats binaires, nous passons par FILE / OPEN pour lire ce type de fichier.

5 Récapitulatif – Taille des fichiers

Nous récapitulons ici la taille des fichiers selon le format sélectionné. Pour le format de l'ancienne version de Sipina (version 2.5), il faudrait additionner le volume des fichiers .PAR et .DAT.



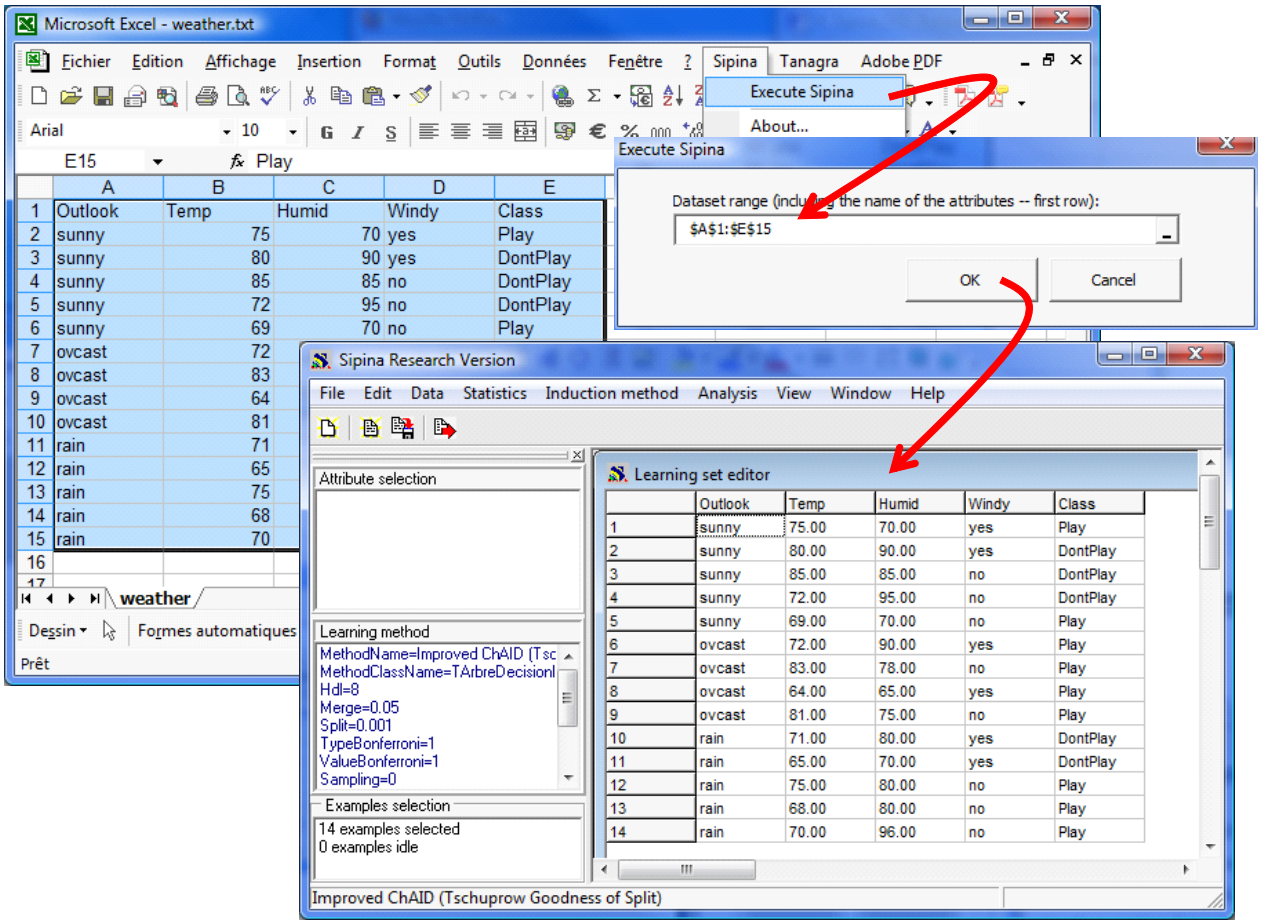
On notera principalement que le format ZDM est le moins gourmand en espace disque, à l'inverse de FDM. C'est assez représentatif des situations que nous rencontrons dans la pratique.

6 Une passerelle entre Excel et Sipina

Quoiqu'en dise, la majorité des data miner passent par Excel pour manipuler leurs données. Parce qu'il est souple, parce qu'il est relativement performant. Il paraît donc indispensable de pouvoir importer le format XLS, ce que font tous les outils commerciaux ; ou plus simple encore, de pouvoir transférer les données d'EXCEL vers le logiciel spécialisé, ce que fait Sipina.

Sipina utilise une macro complémentaire (SIPINA.XLA) pour établir une connexion avec Excel. Elle est systématiquement installée avec la distribution. Son intégration dans Excel et son utilisation sont décrites dans deux didacticiels animés accessibles en ligne : http://tutoriels-data-mining.blogspot.com/2008/03/connexion-excel-sipina.html ou http://sipina.over-blog.fr/article-17592277.html

Pour notre fichier, nous aurons la succession d'opérations suivante



7 Comparaison des performances

Nous disposons de différentes manières d'appréhender les fichiers de données. La question qui se pose est la suivante : « quel format privilégier ? ». Il n'y a pas de réponse définitive. Tout dépend des caractéristiques du fichier.

Si le fichier est de taille modérée, en clair s'il tient dans un tableur, nous avons intérêt à privilégier la connexion Excel – Sipina. Nous y trouvons un très bon compromis entre la souplesse et la puissance. Nous disposons de toutes les fonctionnalités d'édition du tableur et le transfert est rapide.

Lorsque le fichier ne tient plus dans un tableur, il faut trouver le bon arbitrage entre performances et souplesse. Ici **s'opposent vraiment les formats textes et binaires**.

Pour comparer les performances, nous travaillons sur un second fichier comportant **4.817.099 observations et 42 variables** (http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/kdd-cup-discretized-descriptors.txt.zip). Il s'agit d'une variante du fichier « KDD Cup 1999 Data[2] ». Nous utiliserons deux critères pour comparer les formats : l'occupation disque et le temps de traitement. Afin que tout un chacun puisse reproduire l'expérimentation et situer leurs propres résultats, voici les caractéristiques de notre machine : Processeur Quad Core Q9400 à 2.66 Ghz, 4 Go de RAM, fonctionnant sous Windows Vista.

	Ficher texte (CSV)	Fichier ARFF (Weka)	Fichier FDM (binaire natif)	Fichier FDZ (FDM compressé)	Fichier ZDM (compression à la volée)
Taille sur le disque (Ko)	447.917	447.919	808.055	107.846	**15.037**
Durée lecture (secondes)	107	105	8	57	**7**
Durée écriture (secondes)	188	120	**8**	15	15

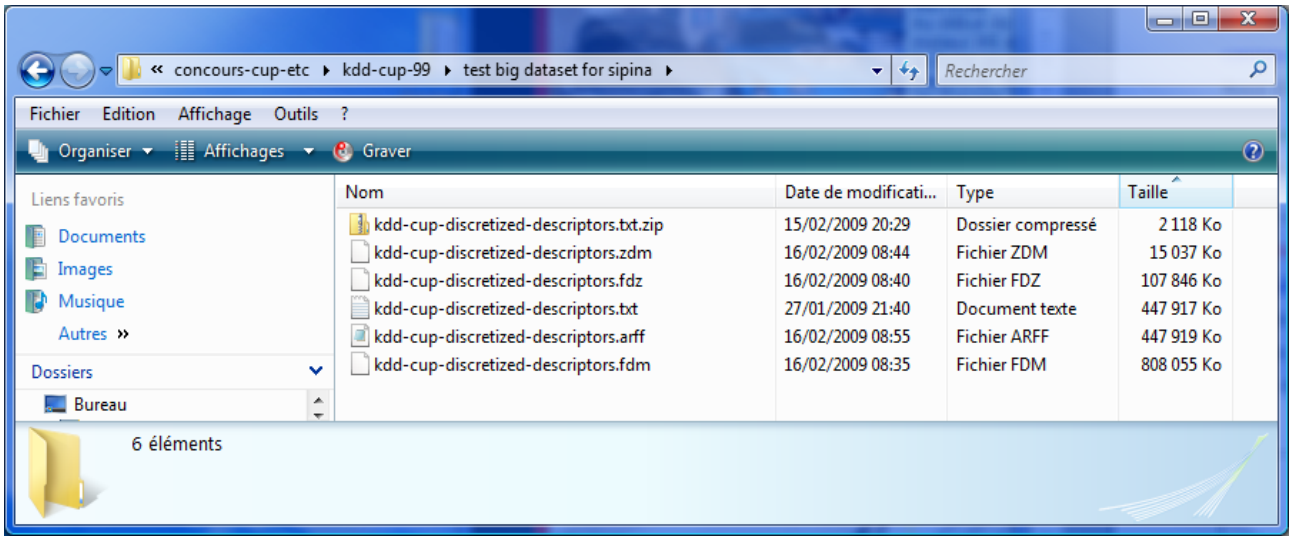
Il y a quelques remarques à émettre et des résultats à commenter :

- Il est évident que l'occupation mémoire des données, une fois chargées, est exactement le même quel que soit le format utilisé.
- Concernant la taille des fichiers, les formats CSV et ARFF se valent, ce n'est pas une surprise.
- L'occupation disque du format FDM est très différente. La raison est que le format binaire encode uniformément chaque valeur sur 4 octets. Pour un fichier texte, une valeur entière n'occupe que le nombre de chiffres qu'elle comporte (ex. la valeur 5 n'occupe que 1 octet sur le disque pour un fichier texte, elle occupe 4 octets pour un fichier binaire). Nos données comportant un grand nombre de valeurs entières, le fichier FDM est nettement plus volumineux que CSV ou ARFF.

[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

- On s'y attendait, la durée d'importation des formats texte (CSV et ARFF) est largement plus élevée par rapport au format FDM. D'un côté, la nécessité de parser le fichier est très pénalisant ; de l'autre, la lecture par blocs des vecteurs de valeurs est extrêmement profitable.
- La durée de l'exportation est aussi très différente entre CSV/ARFF et FDM, même s'il n'y a pas d'analyse à faire lors de l'exportation au format texte. Il est vrai que nous n'avons pas vraiment optimisé les procédures d'écriture des fichiers CSV et ARFF.
- L'inflation de la taille des fichiers FDM nous a poussé à développer des formats compressés. Par rapport au format natif FDM, la taille des fichiers est considérablement réduite. On note que ZDM (15.037 Ko) est nettement plus petit que FDZ (107.846 Ko).
- Le format FDZ passe par un fichier intermédiaire FDM avant de procéder à la compression. Il faut donc, au moins temporairement, comptabiliser la taille du fichier FDM sur le support utilisé pour que l'opération arrive correctement à son terme. Sur un disque dur, cela ne devrait pas trop poser problème, si on entend écrire directement le fichier sur un support amovible (clé USB par exemple), il faut en tenir compte.
- La compression des données entraîne un surcroît de calculs. On le voit très bien lors de l'écriture pour FDZ et ZDM, lors de la lecture pour FDZ.
- Mais pas pour ZDM lors du chargement. En effet le faible volume de données à transférer contrebalance le surcroît de calculs. Ainsi, son temps de chargement est équivalent à celui de FDM.
- En écriture, malgré la création d'un fichier intermédiaire, FDZ reste très rapide, au même niveau que le format ZDM.

Voici un récapitulatif des fichiers sur le disque. On remarquera avec un certain amusement que le fichier texte (CSV) zippé à l'aide d'un programme externe est plus petit, de très loin (2.118 Ko), que tous les formats que nous utilisons. Cela relativise un peu nos solutions. Peut est-ce là finalement la meilleure piste à explorer, sachant quand même que nous avons choisi l'option *ultra* de 7-ZIP (http://www.7-zip.org/) lors de la compression, le temps de traitement a été très long (plus de 120 secondes).

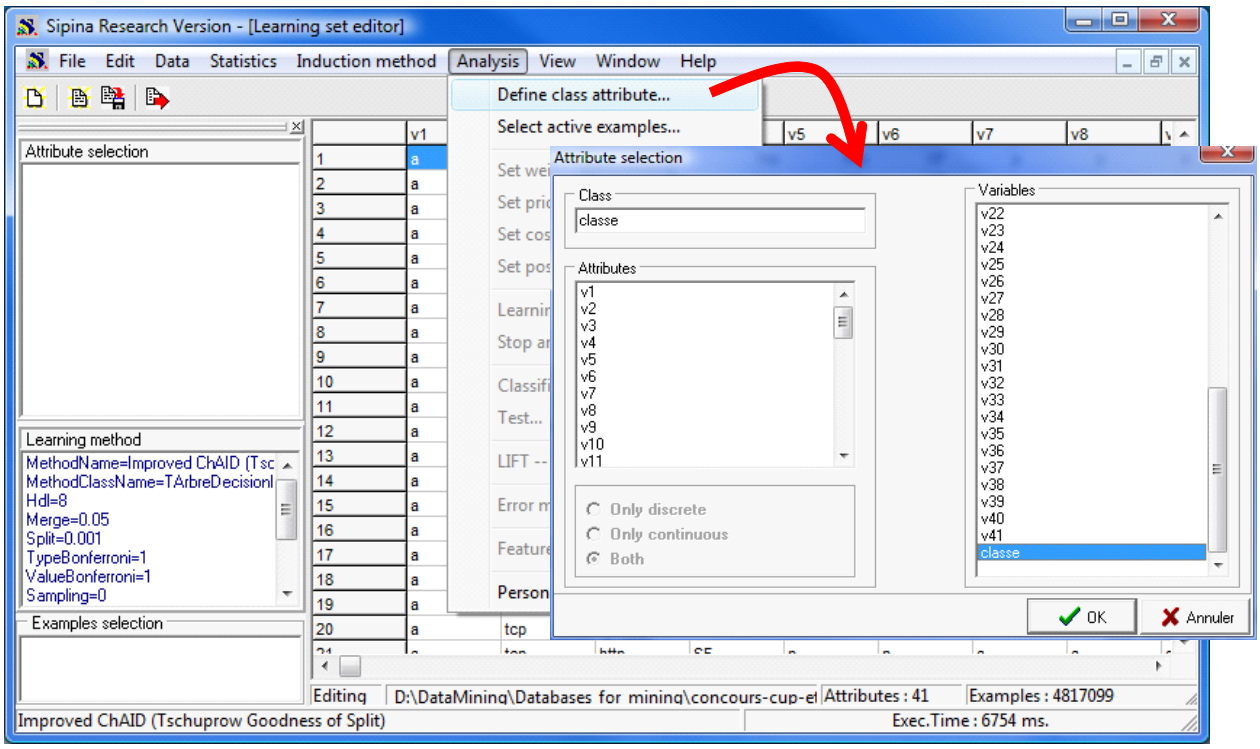


8 Traitement du fichier KDD-CUP 1999

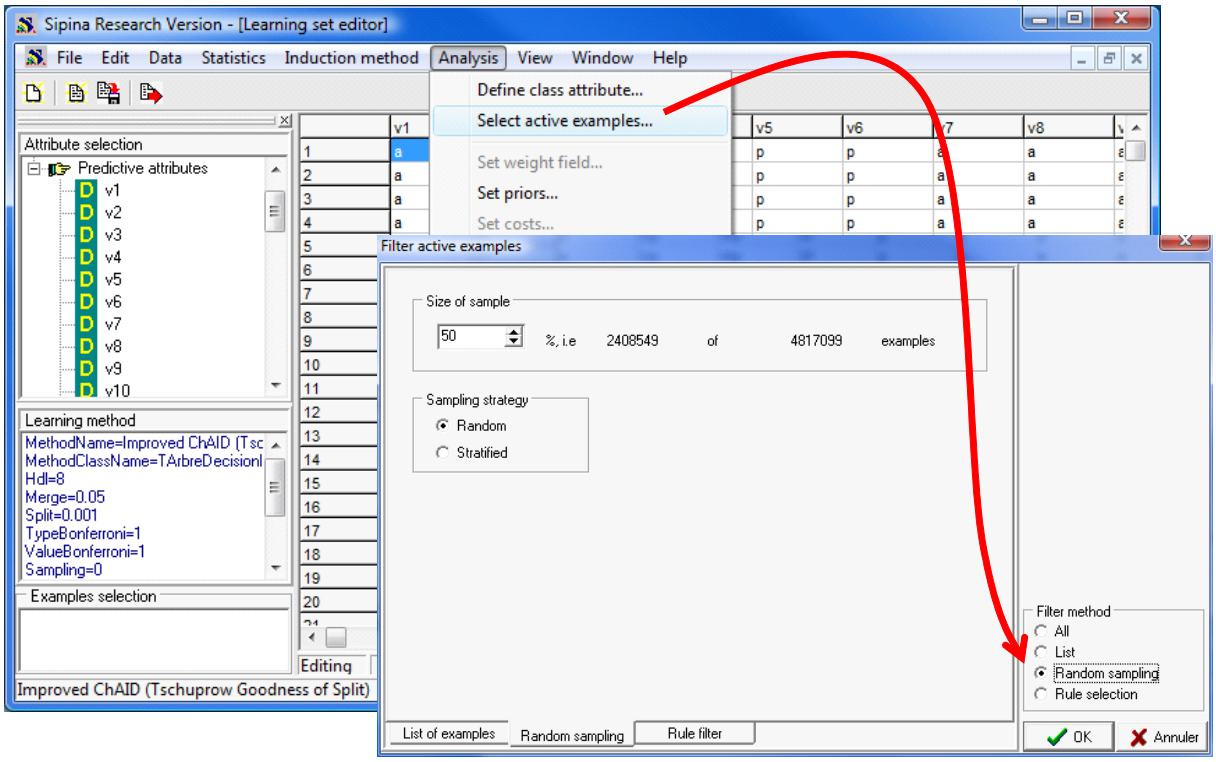
Bien que ce ne soit pas notre propos dans ce didacticiel, nous ne résistons pas à l'envie de tester le comportement de Sipina lors du traitement d'un aussi gros fichier. On sait que théoriquement, nous pouvons allouer 2 Go en mémoire centrale. Mais nous ne pouvons pas exploiter la totalité de cet espace pour les données et les traitements, l'application a besoin également d'une certaine quantité de mémoire pour son propre fonctionnement (interface, etc.).

Avant même de commencer le processus d'induction, le gestionnaire de programme de Windows nous indique que le logiciel avec les données chargées occupe 882 Mo en mémoire.

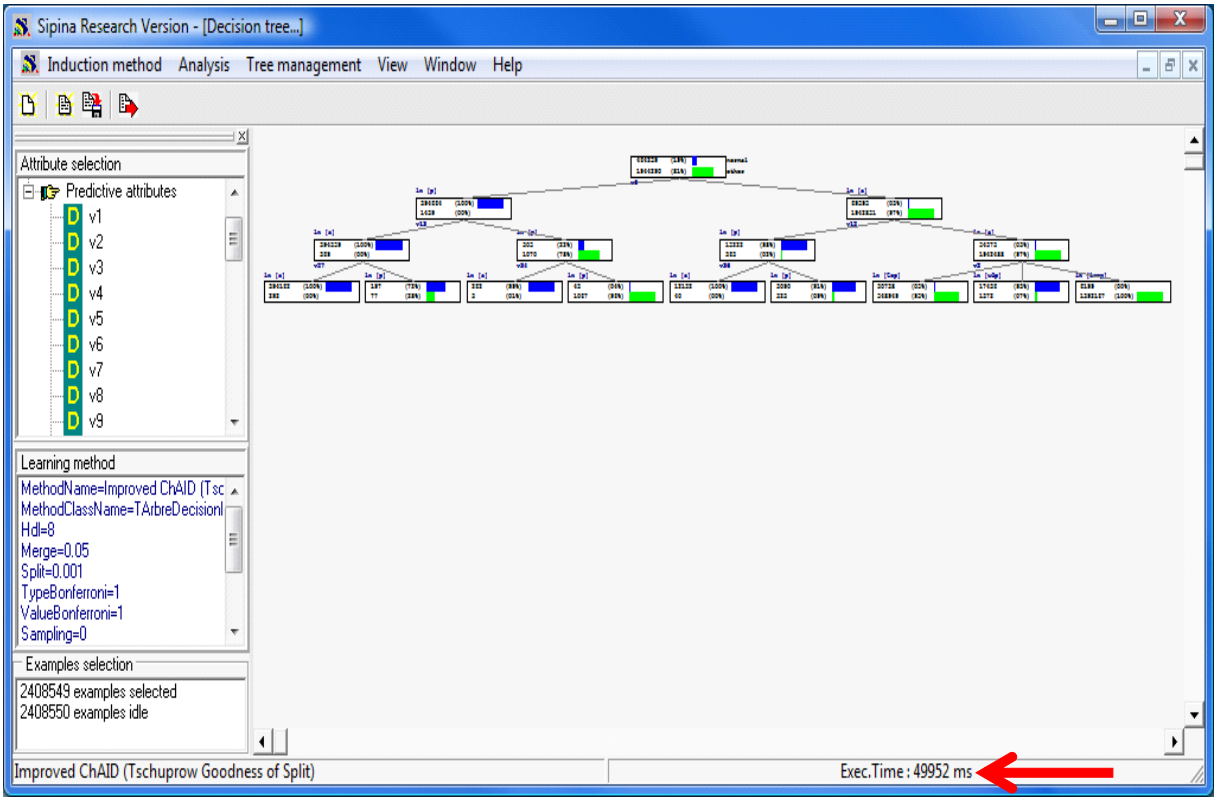
Voyons s'il est possible de construire un arbre de décision. On souhaite prédire CLASSE à partir des autres variables. Nous actionnons le menu ANALYSIS / DEFINE CLASS ATTRIBUTE. Nous réalisons la sélection suivante.



Nous souhaitons utiliser la moitié des observations pour la construction de l'arbre, l'autre moitié pour son évaluation. Nous passons par le menu ANALYSIS / SELECT ACTIVE EXAMPLES. Nous choisissons l'option RANDOM SAMPLING à 50% c.-à-d. nous utilisons 2.408.549 observations pour la construction de l'arbre.

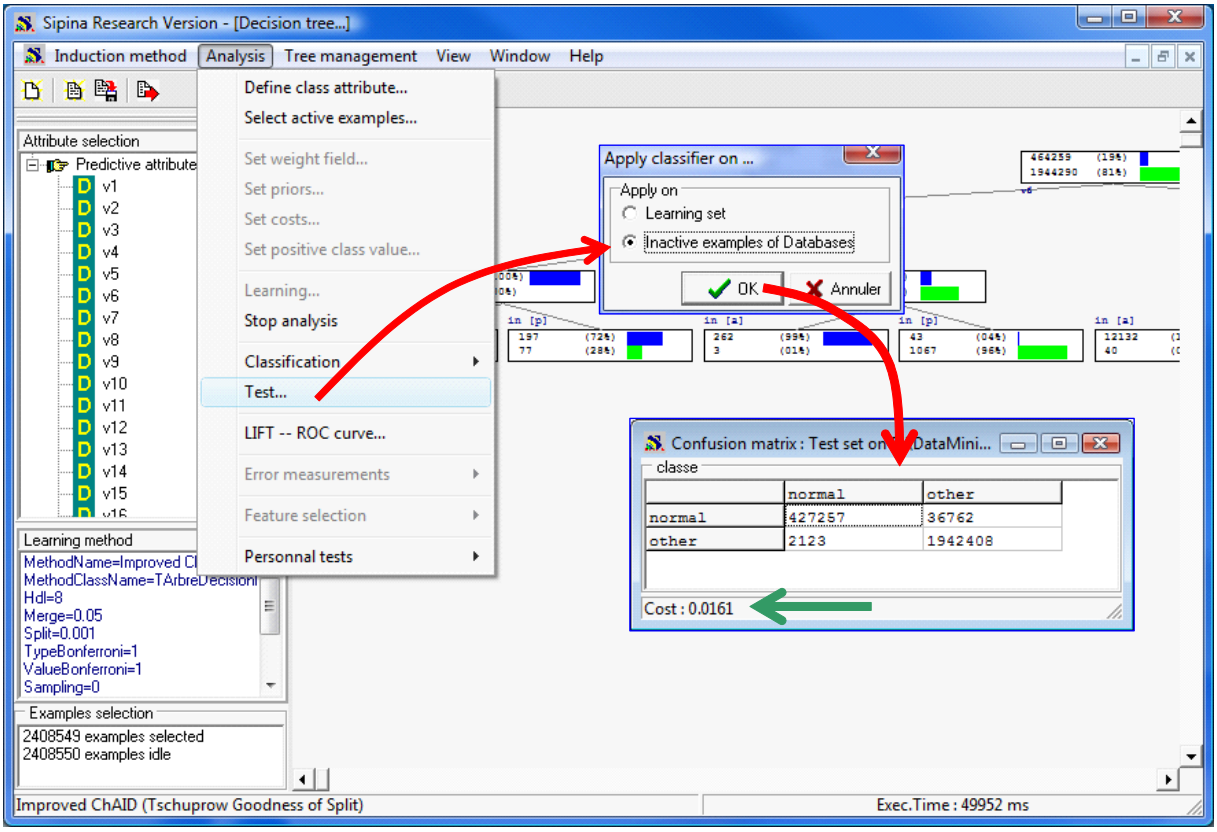


Nous pouvons lancer les traitements, nous utilisons la méthode par défaut IMPROVED CHAID. Nous cliquons sur le menu ANALYSIS / LEARNING. Au bout de 50 secondes, nous obtenons l'arbre suivant.



L'occupation mémoire de Sipina est de 920 Mo à ce stade. Pour évaluer l'arbre sur l'échantillon test, nous passons par le menu ANALYSIS / TEST et nous sélectionnons l'option INACTIVE EXAMPLES OF

DATABASES (c.-à-d. les 2.408.550 observations qui ont été mises de côté). Nous obtenons la matrice de confusion suivante avec un taux d'erreur de 1.61 %.



Remarque : *Sur mon antique Celeron à 2.53 Ghz tournant sous Windows XP avec 1 Go de RAM, le calcul a également été possible. Le temps de chargement du fichier ZDM a été de 30 secondes, la construction de l'arbre a pris 125 secondes*.

9 Conclusion

Notre idée dans ce didacticiel était de faire un rapide tour d'horizon des formats de fichiers que l'on pouvait traiter avec Sipina. On peut les scinder en deux catégories : les formats texte, avec comme principal atout la souplesse ; et les formats binaires, décisifs en matière de temps de traitements et d'occupation disque lorsqu'ils sont compressés.

Bien sûr, comme dans la majorité des cas, lorsque nous avons à choisir parmi plusieurs alternatives, penser que telle ou telle solution serait uniformément meilleure aux autres serait une grosse erreur. Tout dépend de nos priorités. Si nous devons multiplier les opérations de chargement et sauvegarde, il faut privilégier le format binaire. Si nous sommes emmenés à interagir avec d'autres logiciels, utiliser un format texte paraît être la meilleure stratégie.