1 Objectif

Déploiement des modèles prédictifs avec « Pentaho Data Integration Community Edition » (PDI-CE), utilisation du standard de description PMML.

Le déploiement des modèles est une étape importante du processus Data Mining. Dans le cadre de l'apprentissage supervisé, il s'agit de réaliser des prédictions en appliquant les modèles sur des observations non étiquetées. Nous avons décrit à maintes reprises la procédure pour différents outils (ex. Tanagra, Sipina, Spad, ou encore R). Ils ont pour point commun d'utiliser le même logiciel pour la construction du modèle et son déploiement.

Ce nouveau didacticiel se démarque des précédents dans la mesure où nous utilisons un logiciel tiers pour le classement des nouvelles observations. Il fait suite à une remarque qui m'a été faite par **Loïc LUCEL** (merci infiniment Loïc pour tes précieuses indications) qui m'a fait prendre conscience de deux choses : le déploiement donne sa pleine mesure lorsqu'on le réalise avec un outil dédié au management des données, nous prendrons l'exemple de PDI-CE (Kettle) ; nous accédons à une certaine universalité lorsque nous décrivons les modèles à l'aide de standards reconnus/acceptés par la majorité des logiciels, en l'occurrence le standard de description PMML.

J'avais déjà parlé à plusieurs reprises de PMML. Mais jusqu'à présent, je ne voyais pas trop son intérêt si nous n'avons pas en aval un outil capable de l'appréhender de manière générique. Dans ce didacticiel, nous constaterons qu'il est possible d'élaborer un arbre de décision avec différents outils (SIPINA, KNIME et RAPIDMINER), de les exporter en respectant la norme PMML, puis de les déployer de manière indifférenciée sur des observations non étiquetées via PDI-CE. L'adoption d'un standard de description des modèles devient particulièrement intéressante dans ce cas.

Un peu à la marge de notre propos, nous décrirons des solutions de déploiement alternatives dans ce didacticiel. Nous verrons ainsi que Knime possède son propre interpréteur PMML. Il est capable d'appliquer un modèle sur de nouvelles données, quel que soit l'outil utilisé pour l'élaboration du modèle. L'essentiel est que le standard PMML soit respecté. En ce sens, Knime peut se substituer à PDI-CE. Autre piste possible, Weka, qui fait partie de la suite « Pentaho Community Edition », possède un format de description propriétaire directement reconnu par PDI-CE.

2 Données

Nous utilisons les données « heart », bien connues de la communauté de l'apprentissage automatique[1]. Nous souhaitons prédire la présence ou l'absence d'une maladie cardiaque (cœur) à partir des caractéristiques des patients (âge, sexe, etc.). Nous disposons de deux fichiers : **heart-train.arff** est le fichier d'apprentissage, il servira à la construction de l'arbre de décision ; **heart-unlabeled.txt** correspond aux observations non étiquetées sur lesquelles nous appliquerons le modèle prédictif. Au final, l'objectif est de produire un nouveau fichier comprenant les descripteurs repris de **heart-unlabeled.txt**, et une colonne supplémentaire « prédiction » indiquant la classe (présence ou absence de maladie cardiaque) affectée à chaque individu.

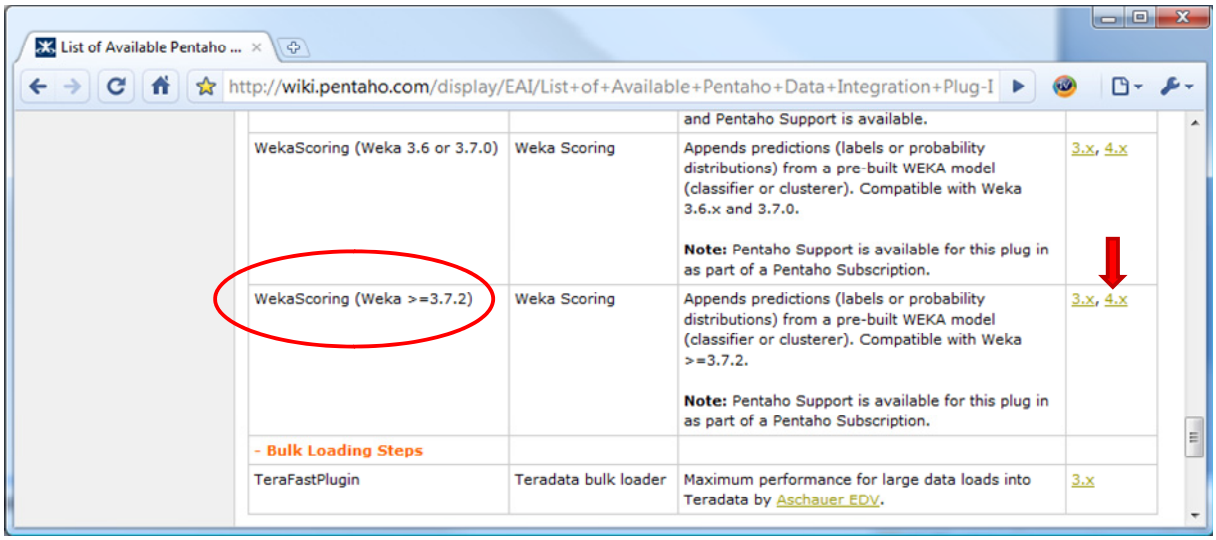
[1] http://archive.ics.uci.edu/ml/datasets/Heart+Disease

3 Déploiement avec PDI-CE

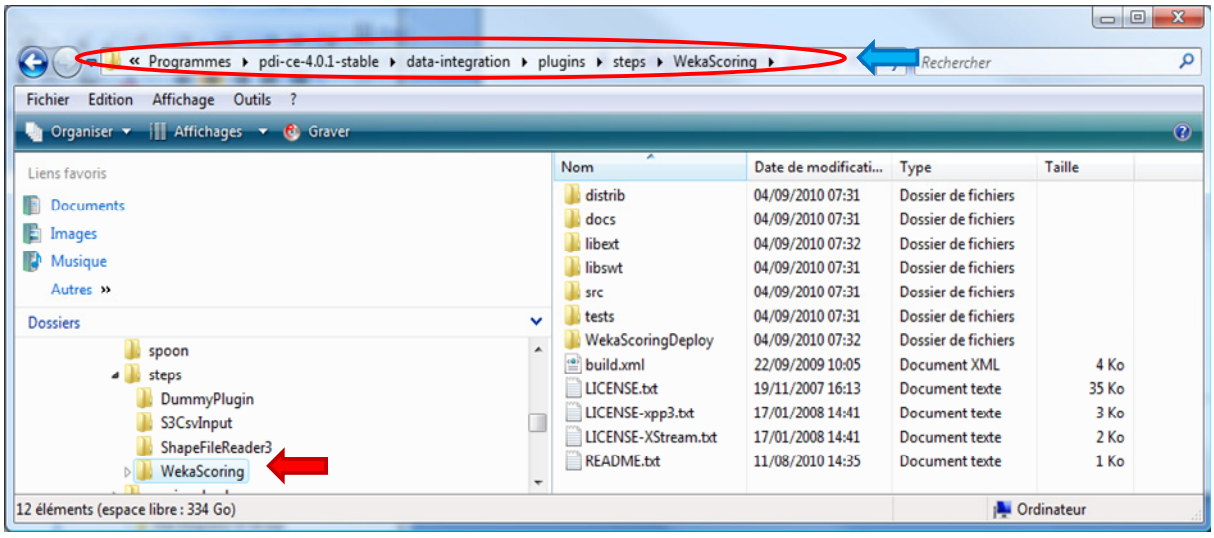
3.1 Installation de PDI-CE et du plug-in « WekaScoring »

Nous avons décrit l'installation de PDI-CE dans un précédent didacticiel[2]. Pour le déploiement des modèles prédictifs, nous devons intégrer le plug-in « WekaScoring ». Comme son nom l'indique, il sert à manipuler les modèles issus de Weka (avec un format propriétaire, nous y reviendrons plus loin – section 5) ; mais plus largement, il sait également gérer les modèles au format PMML, quelle qu'en soit la provenance.

Dans un premier temps, il faut récupérer le plug-in sur le site web de Pentaho. Attention, il ne faut pas mélanger les versions. Pour notre part, nous utilisons PDI-CE 4.0.1, nous avons chargé le plug-in WekaScoring version 4.x, adapté aux modèles produits à l'aide de Weka version 3.7.2 et ultérieures.

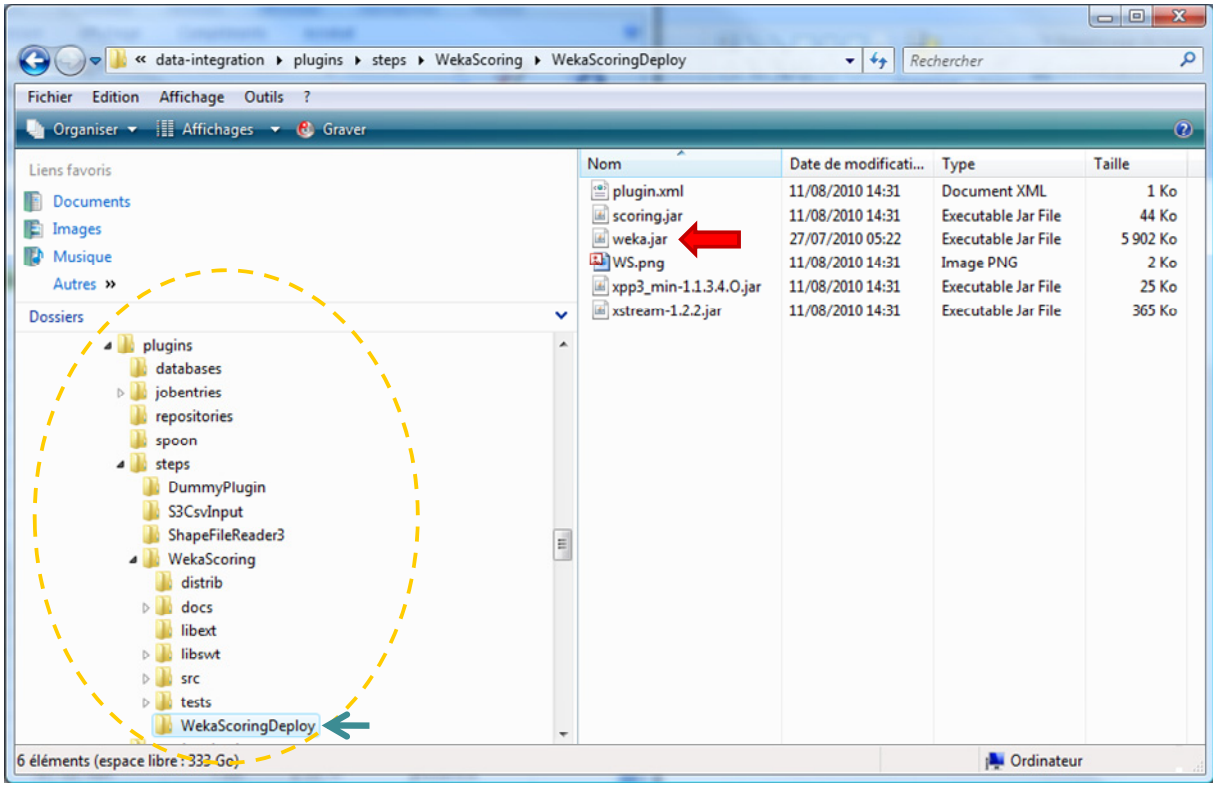


Puis, nous le désarchivons dans le répertoire d'installation de PDI-CE. Nous devrions avoir une configuration ressemblant à celle-ci sur le disque dur.

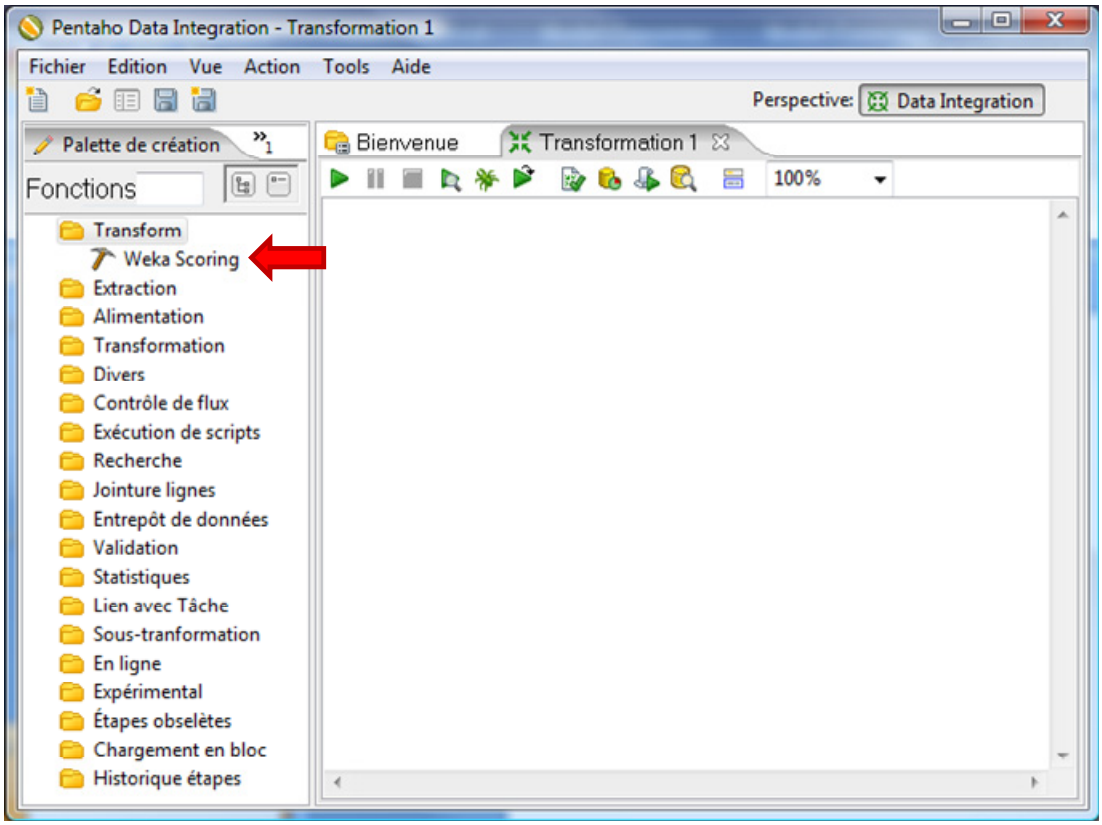


Ensuite, nous devons copier le fichier « **weka.jar** » issue de la distribution WEKA (la version 3.7.2 en ce qui me concerne) dans le sous-répertoire « WekaScoringDeploy ».

[2] http://tutoriels-data-mining.blogspot.com/2010/09/pentaho-data-integration.html



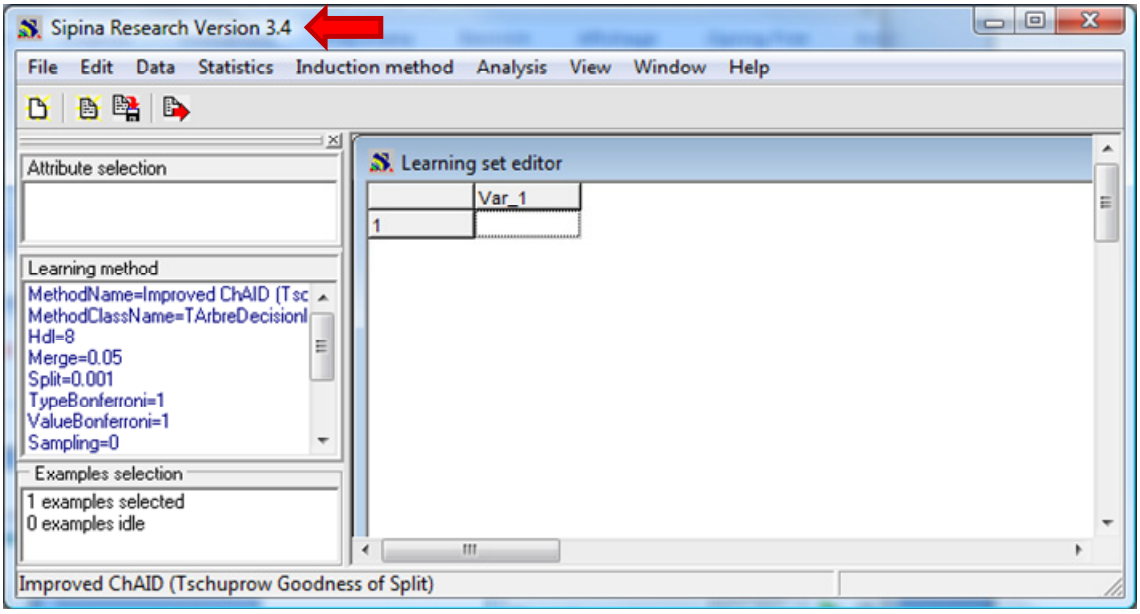
Enfin, pour vérifier le tout, le plus simple est de lancer PDI-CE. L'icône WekaScoring doit être disponible dans la branche TRANSFORM de la palette de création (créez un nouveau projet de transformation vide pour la vérification – Fichier / Nouveau / Transformation).



Pour pouvoir réaliser le déploiement via PDI-CE, il nous faut construire l'arbre de décision et le sauver au format PMML. Nous utilisons SIPINA pour cela.

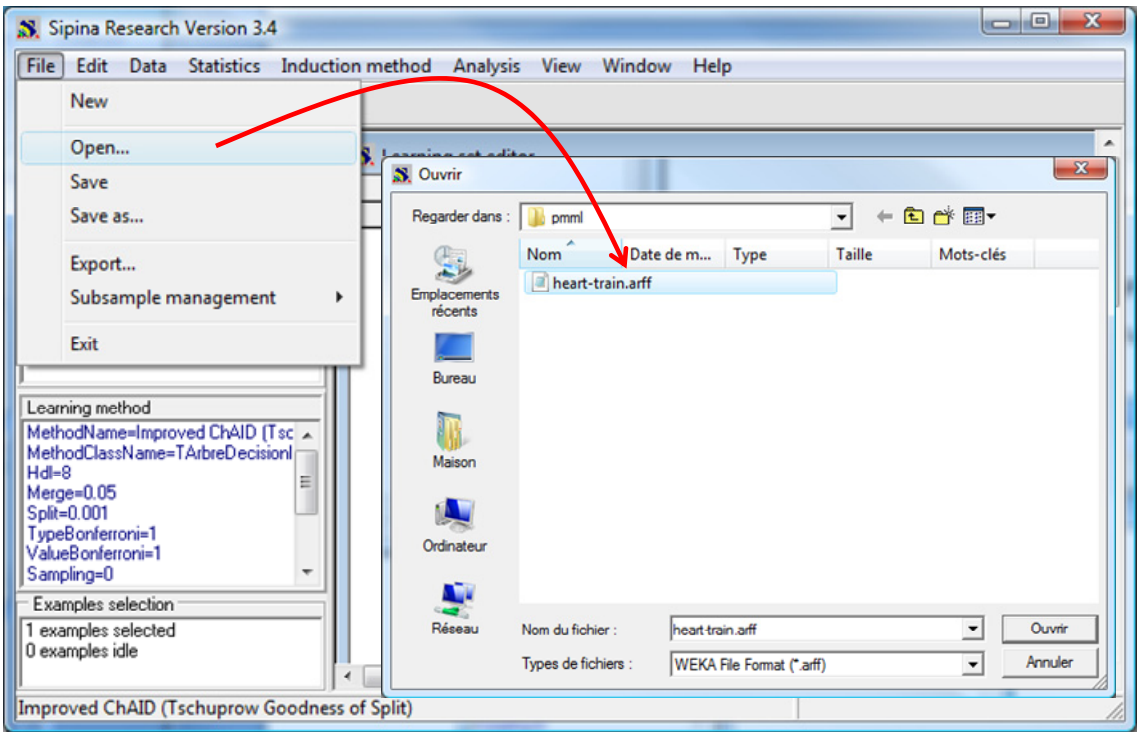
3.2 Création et sauvegarde d'un modèle au format PMML avec SIPINA

Pour cette partie, assurez vous de disposer de SIPINA 3.4. Vous pouvez vérifier le numéro de version dans la barre de titre du logiciel.

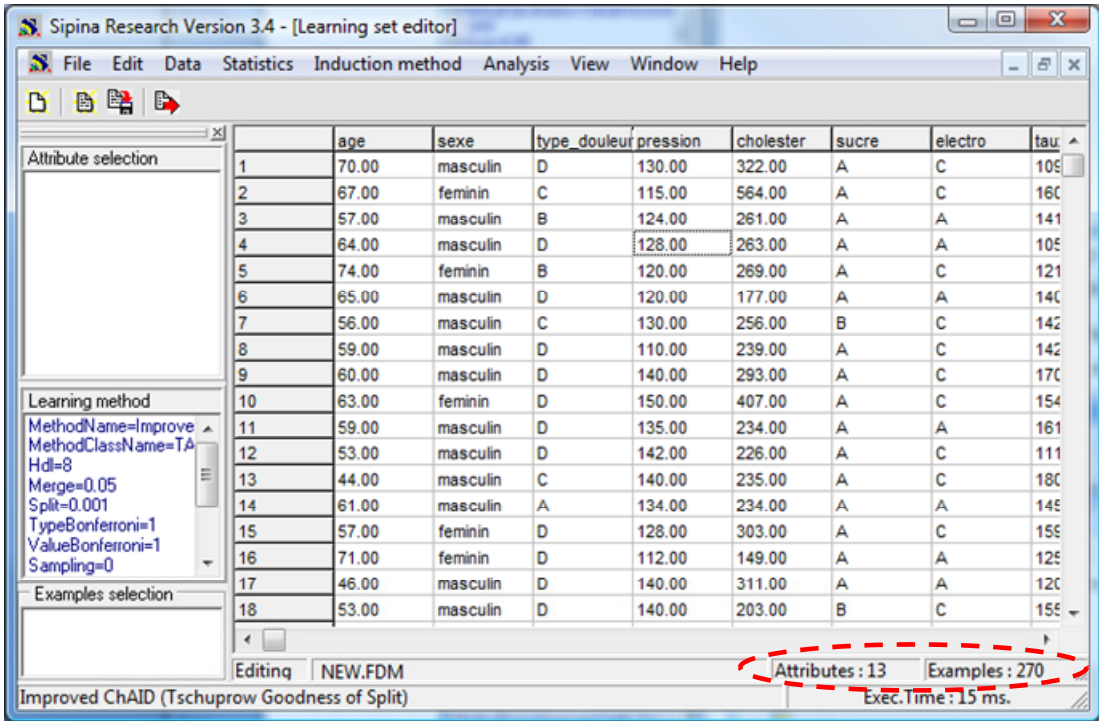


Si ce n'est pas le cas, le plus simple est de récupérer la dernière version sur le site de distribution du logiciel (http://sipina.over-blog.fr/), puis de l'installer par-dessus la précédente.

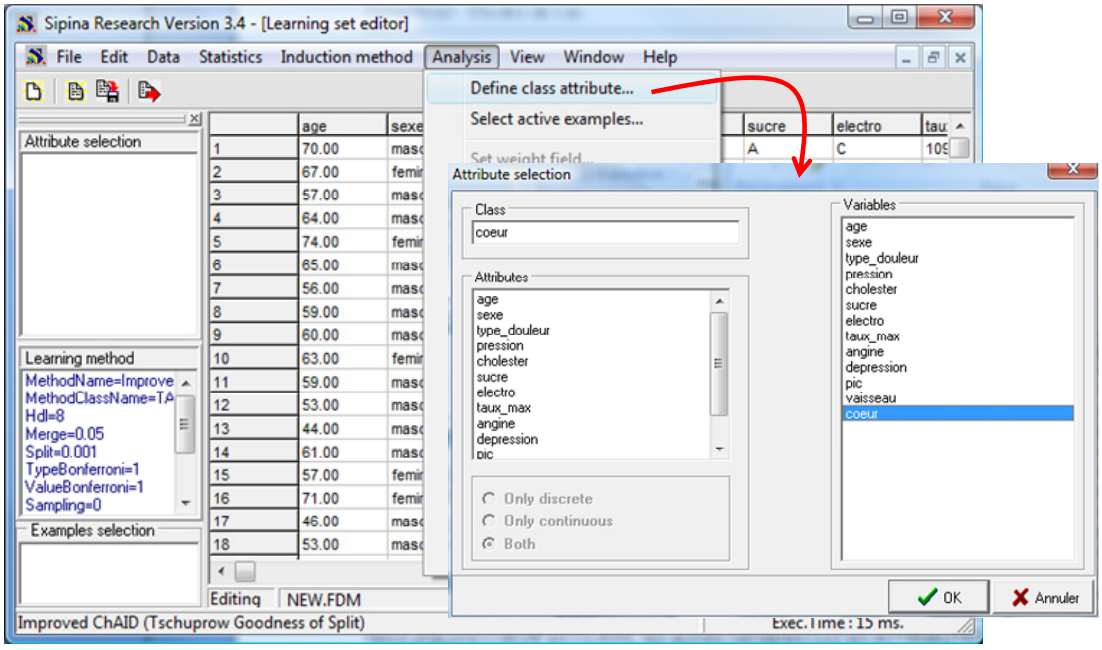
Chargement des données. Après avoir démarré SIPINA, nous chargeons les données en actionnant le menu FILE / OPEN… Nous sélectionnons le fichier « heart-train.arff » au format Weka.



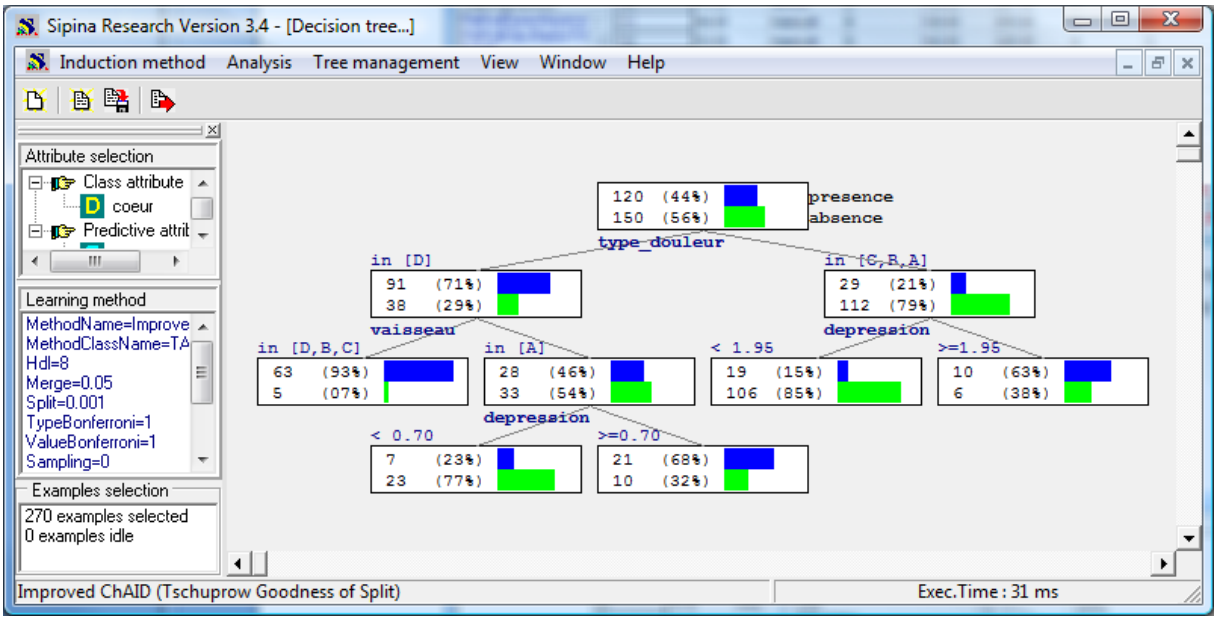
Nous disposons de 13 variables et 270 observations.



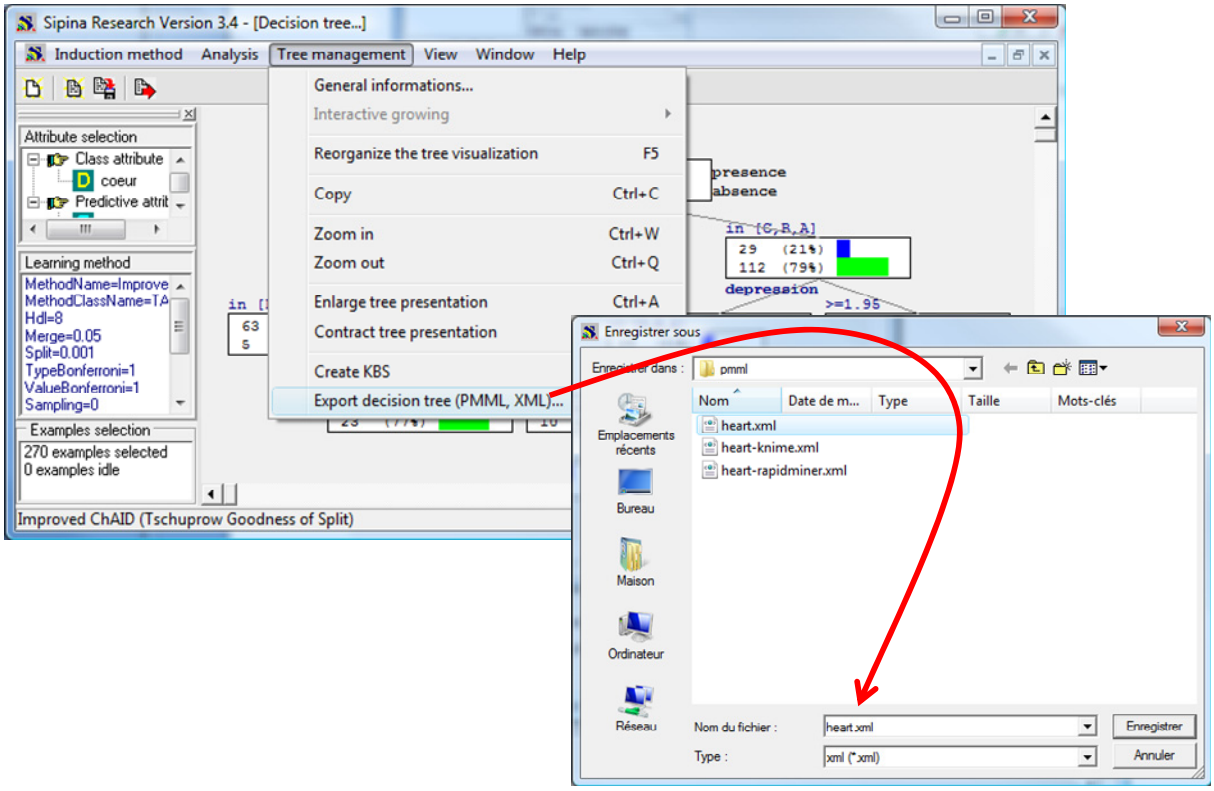
Construction de l'arbre. Nous devons spécifier le rôle des variables avant de démarrer la phase d'apprentissage. Pour ce faire, nous cliquons sur le menu ANALYSIS / DEFINE CLASS ATTRIBUTE. Nous plaçons CŒUR en CLASS, les autres variables (12) en ATTRIBUTES.



Il ne nous reste plus qu'à actionner le menu ANALYSIS / LEARNING… pour construire l'arbre de décision. Nous obtenons un modèle relativement simple, 3 variables parmi les 12 candidates interviennent dans la définition de l'arbre.



Exportation de l'arbre au format PMML. Il faut exporter l'arbre au format PMML. Nous actionnons le menu TREE MANAGEMENT / EXPORT DECISION TREE. Nous nommons le fichier « **heart.xml** ».



Remarque : *Choisir l'extension « .xml » semble plus judicieux pour PDI-CE. Mais en réalité, le contenu des fichiers « .xml » et « .pmml » est strictement le même.*

Comprendre le format PMML. Voyons justement le contenu du fichier « heart.xml ». On peut le subdiviser en plusieurs parties : (A) la définition des données (« DataDictionnary ») ; (B) la description de l'arbre avec la définition du problème d'apprentissage (« MiningSchema ») -- c.-à-d. spécifier la variable cible et les variables prédictives -- et l'énumération des nœuds successifs de l'arbre (« Node »).

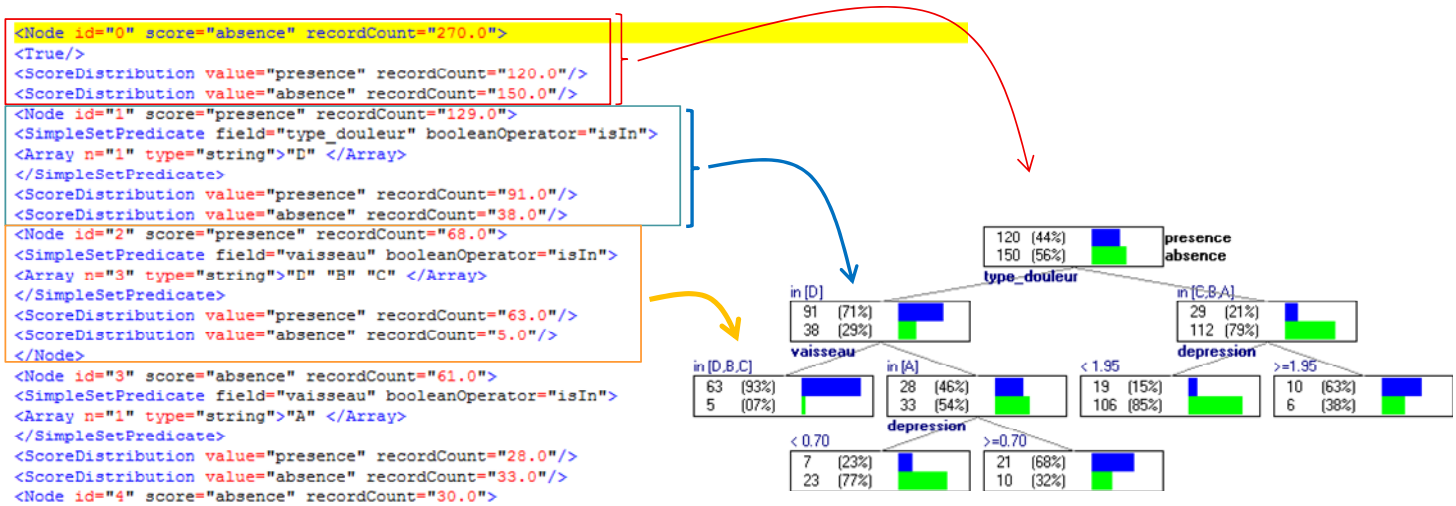
- Concernant le dictionnaire de données (Figure 1), nous observons par exemple que la variable « sexe » est catégorielle, avec deux valeurs (« value ») possibles : « masculin » et « féminin ». En revanche, la variable « pression » est quantitative, la valeur minimale « leftMargin » (resp. maximale, « rightMargin ») observée est 94 (resp. 200). Nous utilisons la notation scientifique.
- Le schéma d'apprentissage (Figure 2) spécifie le rôle particulier de « cœur » dans le modèle, il s'agit de la variable à prédire.
- Enfin, lors de la description de l'arbre (Figure 3), nous distinguons les correspondances entre le contenu du fichier PMML et la représentation graphique de la structure. Pour chaque nœud nous observons : l'effectif global, la proposition qui la défini, les effectifs par classe.

```
<DataDictionary numberOfFields="13">
<DataField name="age" optype="continuous" dataType="double">
<Interval closure="closedClosed" leftMargin="2.90000000000000E+001" rightMargin="7.70000000000000E+001"/>
</DataField>
<DataField name="sexe" optype="categorical" dataType="string">
<Value value="masculin"/>
<Value value="feminin"/>
</DataField>
<DataField name="type_douleur" optype="categorical" dataType="string">
<Value value="D"/>
<Value value="C"/>
<Value value="B"/>
<Value value="A"/>
</DataField>
<DataField name="pression" optype="continuous" dataType="double">
<Interval closure="closedClosed" leftMargin="9.40000000000000E+001" rightMargin="2.00000000000000E+002"/>
</DataField>
<DataField name="cholester" optype="continuous" dataType="double">
<Interval closure="closedClosed" leftMargin="1.26000000000000E+002" rightMargin="5.64000000000000E+002"/>
</DataField>
<DataField name="sucre" optype="categorical" dataType="string">
<Value value="A"/>
<Value value="B"/>
</DataField>
<DataField name="electro" optype="categorical" dataType="string">
<Value value="C"/>
<Value value="A"/>
<Value value="B"/>
</DataField>
<DataField name="taux_max" optype="continuous" dataType="double">
<Interval closure="closedClosed" leftMargin="7.10000000000000E+001" rightMargin="2.02000000000000E+002"/>
</DataField>
<DataField name="angine" optype="categorical" dataType="string">
<Value value="non"/>
<Value value="oui"/>
</DataField>
<DataField name="depression" optype="continuous" dataType="double">
<Interval closure="closedClosed" leftMargin="0.00000000000000E+000" rightMargin="6.19999980926514E+000"/>
</DataField>
<DataField name="pic" optype="continuous" dataType="double">
<Interval closure="closedClosed" leftMargin="1.00000000000000E+000" rightMargin="3.00000000000000E+000"/>
</DataField>
<DataField name="vaisseau" optype="categorical" dataType="string">
<Value value="D"/>
<Value value="A"/>
<Value value="B"/>
<Value value="C"/>
</DataField>
<DataField name="coeur" optype="categorical" dataType="string">
<Value value="presence"/>
<Value value="absence"/>
</DataField>
</DataDictionary>
```

Figure 1 - PMML - Description des données

```
<TreeModel modelName="DecisionTree" functionName="classification" s
<MiningSchema>
<MiningField name="age"/>
<MiningField name="sexe"/>
<MiningField name="type_douleur"/>
<MiningField name="pression"/>
<MiningField name="cholester"/>
<MiningField name="sucre"/>
<MiningField name="electro"/>
<MiningField name="taux_max"/>
<MiningField name="angine"/>
<MiningField name="depression"/>
<MiningField name="pic"/>
<MiningField name="vaisseau"/>
<MiningField name="coeur" usageType="predicted"/>
</MiningSchema>
```

Figure 2 - Mining Schema - Spécification du problème

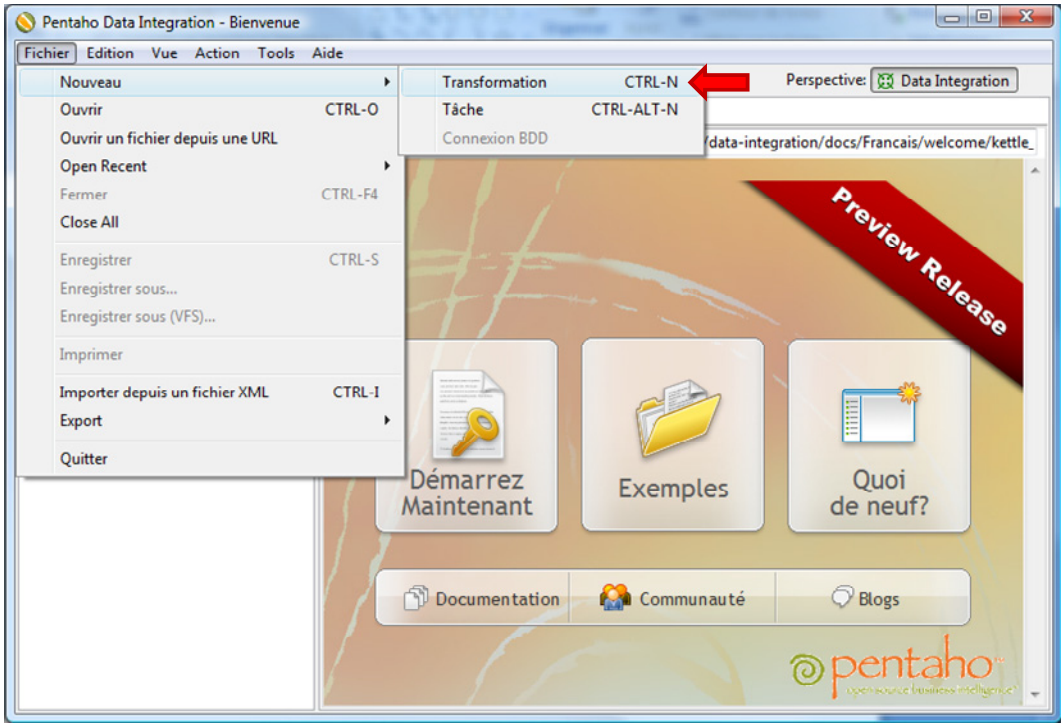



```
<Node id="0" score="absence" recordCount="270.0">
<True/>
<ScoreDistribution value="presence" recordCount="120.0"/>
<ScoreDistribution value="absence" recordCount="150.0"/>
<Node id="1" score="presence" recordCount="129.0">
<SimpleSetPredicate field="type_douleur" booleanOperator="isIn">
<Array n="1" type="string">"D" </Array>
</SimpleSetPredicate>
<ScoreDistribution value="presence" recordCount="91.0"/>
<ScoreDistribution value="absence" recordCount="38.0"/>
<Node id="2" score="presence" recordCount="68.0">
<SimpleSetPredicate field="vaisseau" booleanOperator="isIn">
<Array n="3" type="string">"D" "B" "C" </Array>
</SimpleSetPredicate>
<ScoreDistribution value="presence" recordCount="63.0"/>
<ScoreDistribution value="absence" recordCount="5.0"/>
</Node>
<Node id="3" score="absence" recordCount="61.0">
<SimpleSetPredicate field="vaisseau" booleanOperator="isIn">
<Array n="1" type="string">"A" </Array>
</SimpleSetPredicate>
<ScoreDistribution value="presence" recordCount="28.0"/>
<ScoreDistribution value="absence" recordCount="33.0"/>
<Node id="4" score="absence" recordCount="30.0">
<SimplePredicate field="depression" operator="lessThan" value="7.00000047683716E-001"/>
<ScoreDistribution value="presence" recordCount="7.0"/>
<ScoreDistribution value="absence" recordCount="23.0"/>
</Node>
<Node id="5" score="presence" recordCount="31.0">
<SimplePredicate field="depression" operator="greaterOrEqual" value="7.00000047683716E-001"/>
<ScoreDistribution value="presence" recordCount="21.0"/>
<ScoreDistribution value="absence" recordCount="10.0"/>
</Node>
</Node>
</Node>
<Node id="6" score="absence" recordCount="141.0">
<SimpleSetPredicate field="type_douleur" booleanOperator="isIn">
<Array n="3" type="string">"C" "B" "A" </Array>
</SimpleSetPredicate>
<ScoreDistribution value="presence" recordCount="29.0"/>
<ScoreDistribution value="absence" recordCount="112.0"/>
<Node id="7" score="absence" recordCount="125.0">
<SimplePredicate field="depression" operator="lessThan" value="1.95000004768372E+000"/>
<ScoreDistribution value="presence" recordCount="19.0"/>
<ScoreDistribution value="absence" recordCount="106.0"/>
</Node>
<Node id="8" score="presence" recordCount="16.0">
<SimplePredicate field="depression" operator="greaterOrEqual" value="1.95000004768372E+000"/>
<ScoreDistribution value="presence" recordCount="10.0"/>
<ScoreDistribution value="absence" recordCount="6.0"/>
</Node>
</Node>
</Node>
</TreeModel>
```

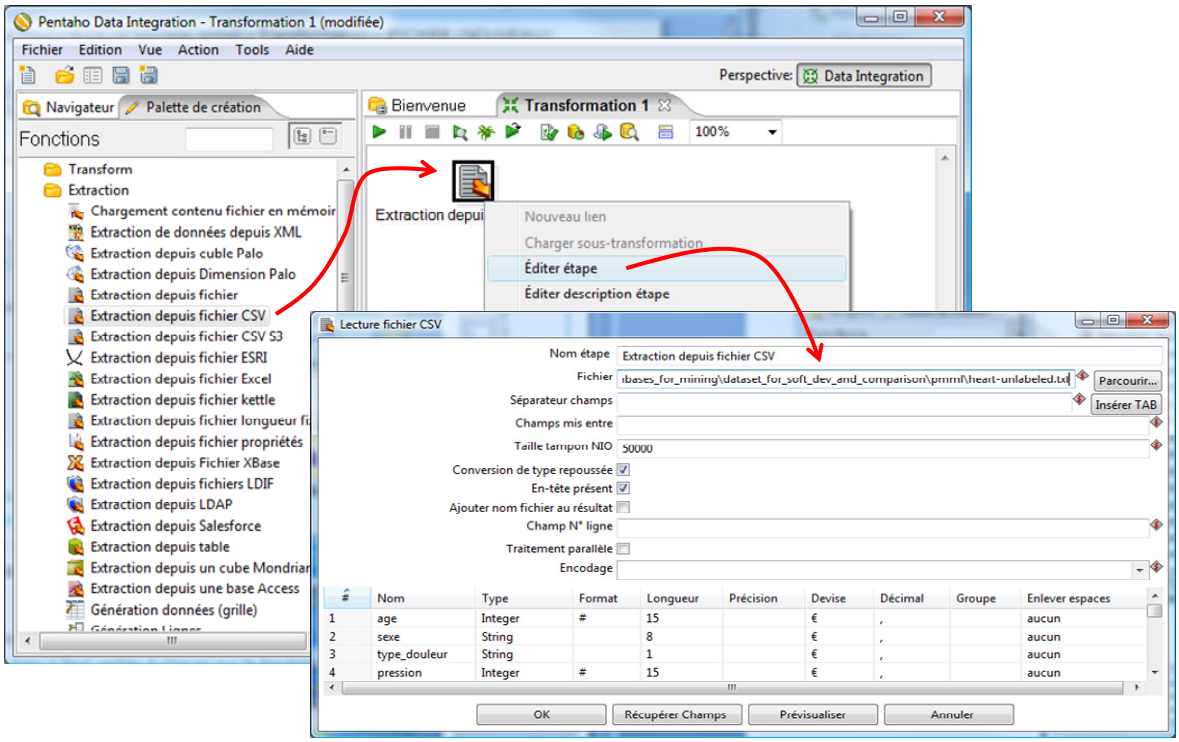
Figure 3- Description de l'arbre

3.3 Classement via PDI-CE et le plug-in WekaScoring

Nous souhaitons appliquer cet arbre sur les observations à classer « **heart-unlabeled.txt** ». Nous démarrons PDI-CE, puis nous créons un nouveau projet « Transformation » (FICHIER / NOUVEAU / TRANSFORMATION).

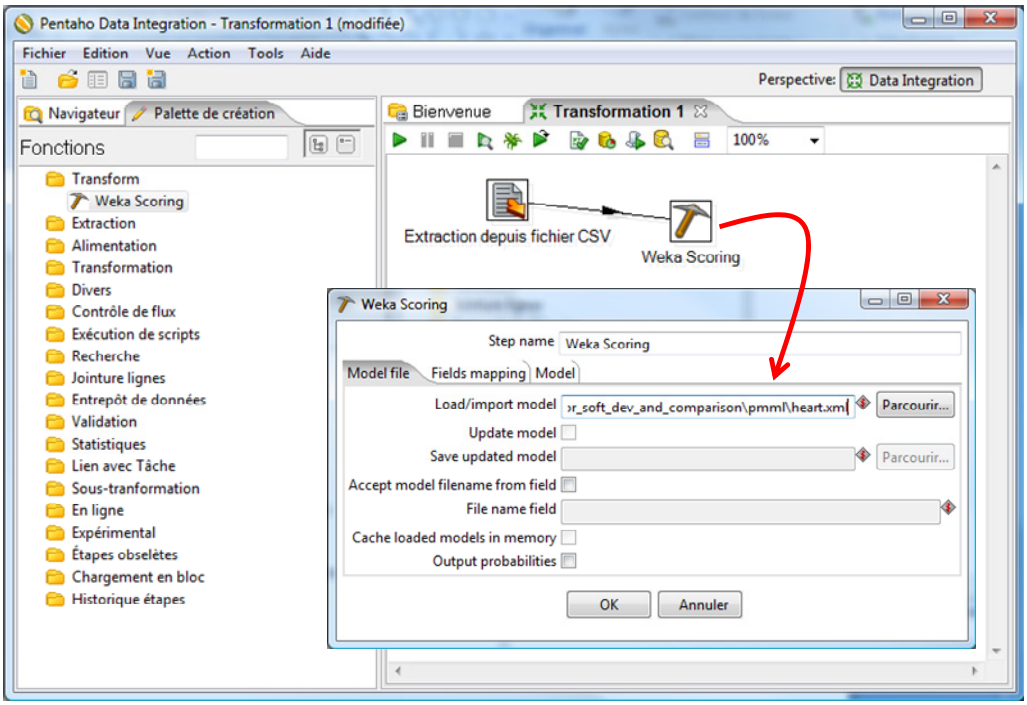


Importation des observations à classer. Nous chargeons le fichier avec le composant « **Extraction depuis fichier CSV** » (branche *EXTRACTION*). Nous le paramétrons en actionnant le menu contextuel « Editer étape ».

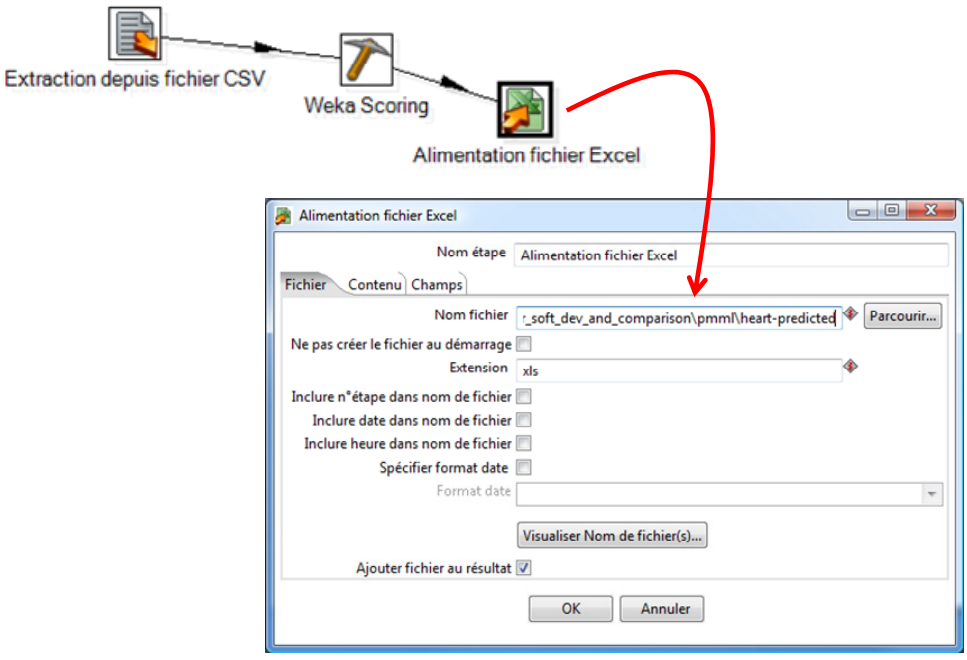


Attention, le caractère tabulation est le séparateur de colonne (« Séparateur champs »). Il n'est pas visible dans l'éditeur, mais il faut veiller à cliquer sur le bouton adéquat pour le spécifier. Nous cliquons sur le bouton « Récupérer Champs » pour obtenir la description des variables.

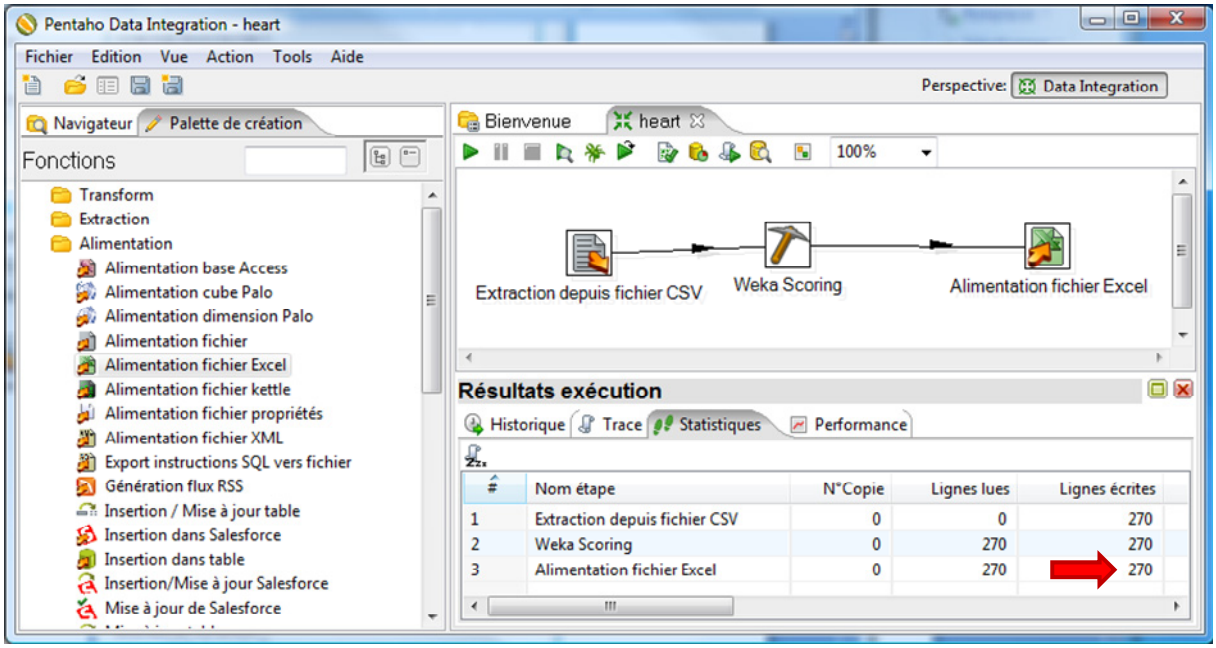
Classement avec le composant WekaScoring. Nous rajoutons le composant « WekaScoring », nous lui relions l'outil précédent. Nous paramétrons WekaScoring en actionnant le menu contextuel « Editer étape ». Nous importons le modèle « heart.xml » (bouton Load/Import model).



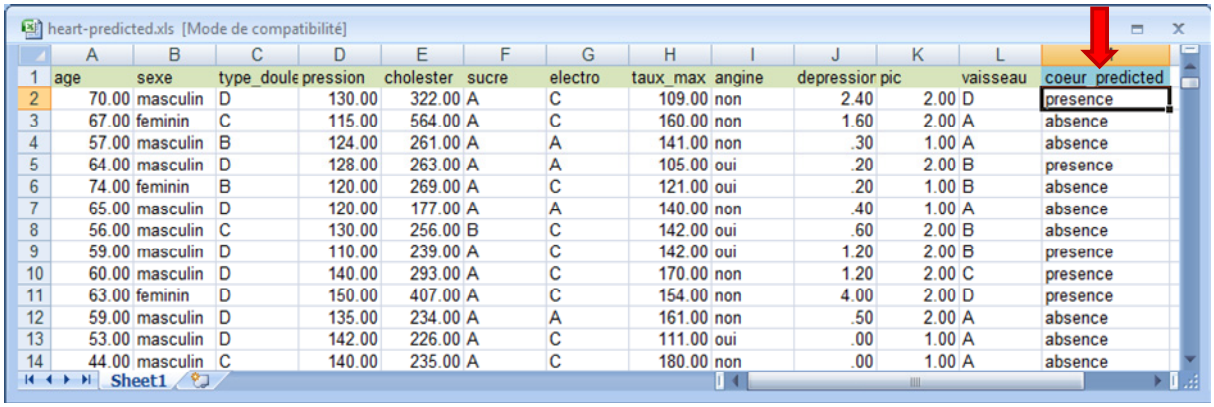
Création du fichier de sortie. Nous souhaitons créer un fichier de sortie au format Excel. Il doit contenir, pour chaque observation, les descripteurs et la classe prédite. Nous ajoutons le composant « Alimentation fichier Excel» (branche *ALIMENTATION*). Nous le paramétrons de manière à spécifier le fichier de sortie « heart-predicted.xls ».



Exécution de la transformation. Aboutissement de toutes ces manipulations, nous lançons les opérations en actionnant le menu ACTION / EXECUTER (F9).



Nous retrouvons le fichier des observations étiquetées « heart-predicted.xls » dans le répertoire de destination idoine. Nous l'ouvrons dans le tableur Excel, les prédictions se situent dans la dernière colonne « cœur_predicted ».



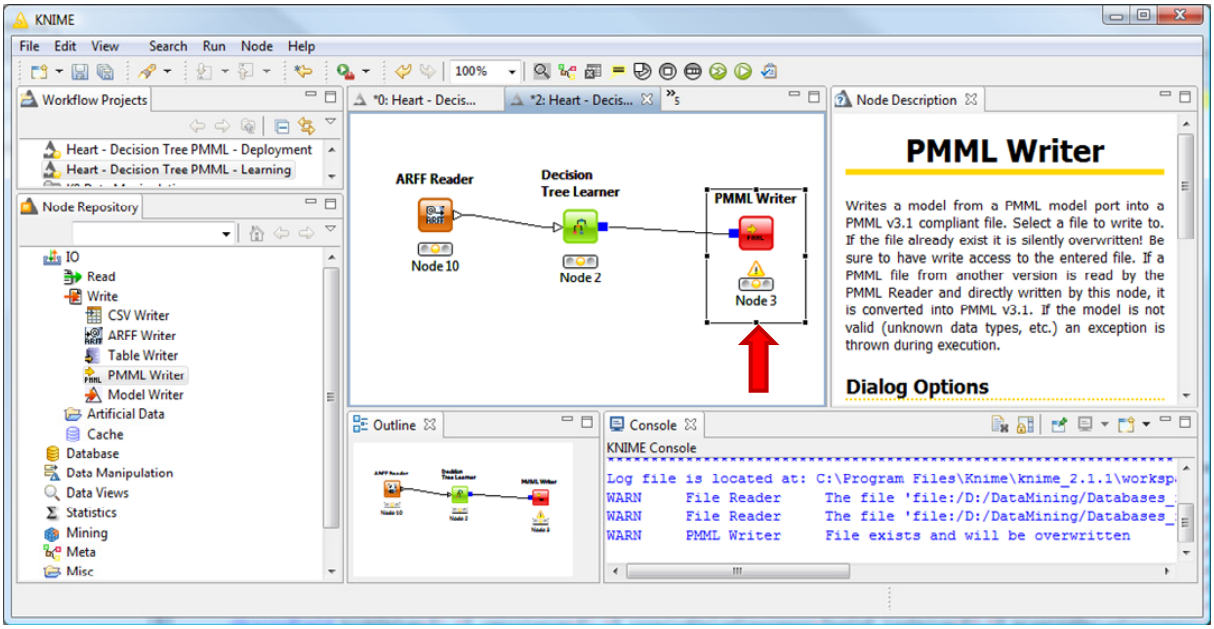
Le 1er individu par exemple est affecté à la classe « cœur = présence », le second à « absence », etc.

3.4 Exportation des modèles au format PMML avec d'autres logiciels

Nous avons construit notre arbre de décision avec SIPINA. Dans les deux sous-sections qui suivent, nous montrons qu'il est possible de reproduire la même démarche avec d'autres logiciels. L'essentiel est qu'ils sachent exporter correctement les modèles en respectant la norme PMML.

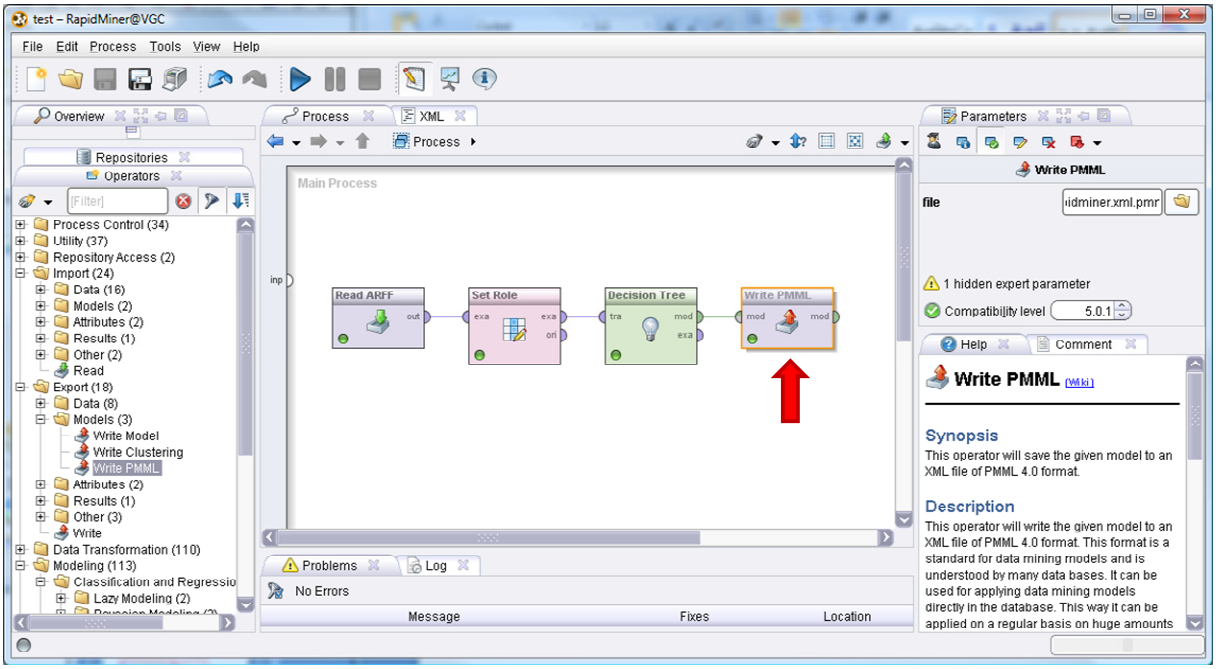
3.4.1 Elaboration et exportation des modèles avec Knime

Nous avons élaboré le diagramme suivant sous Knime. Le composant PMML WRITER sert à exporter le modèle qui lui est connecté. Dans notre cas, il s'agit d'un arbre de décision.



3.4.2 Elaboration et exportation des modèles avec RapidMiner

De la même manière, sous RapidMiner (**version 5.0.010** ; l'interface a beaucoup changé depuis nos précédents tutoriels). Nous pouvons construire le fichier PMML à partir d'un arbre de décision.

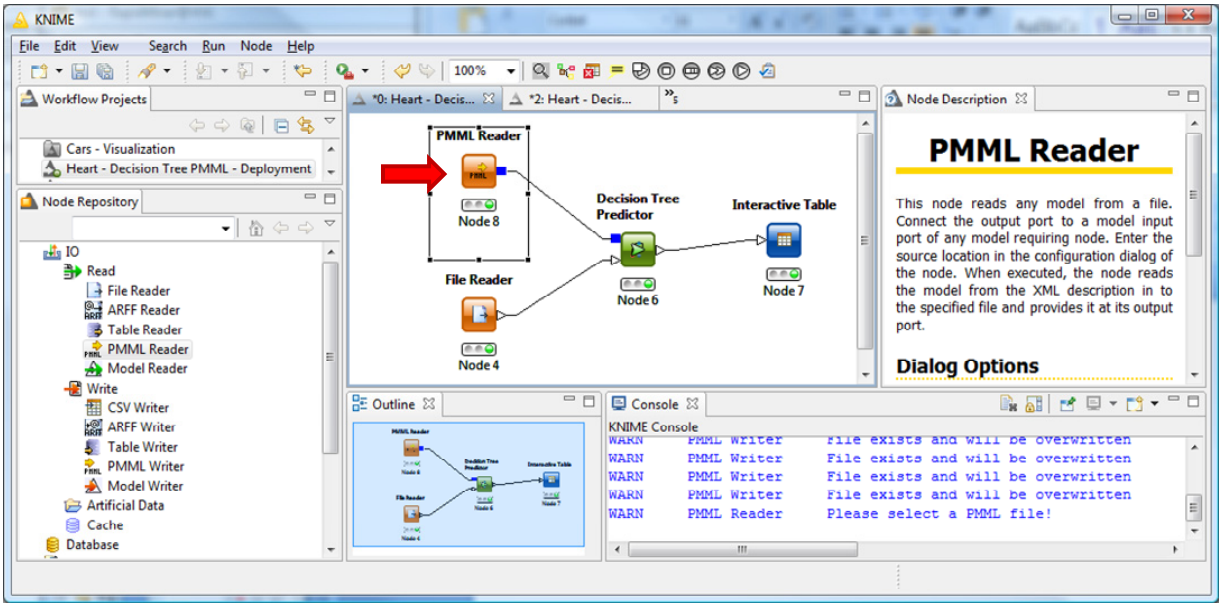


4 Déploiement avec d'autres logiciels (Knime)

PDI-CE est un outil destiné au management des données, son implication dans le déploiement de modèles est donc parfaitement naturel. Mais d'autres logiciels peuvent s'en charger, pourvu qu'ils sachent gérer les données et interpréter correctement les modèles. L'adoption d'un standard, le PMML, est certainement un atout indéniable dans ce cadre.

Nous montrons très succinctement comment déployer un modèle avec Knime. Nous avons utilisé le diagramme suivant. On notera la double source d'informations, données (File Reader) et

modèle (PMML Reader), en entrée de l'outil de prédiction (Decision Tree Predictor). Les observations sont directement visualisées avec le composant InteractiveTable.

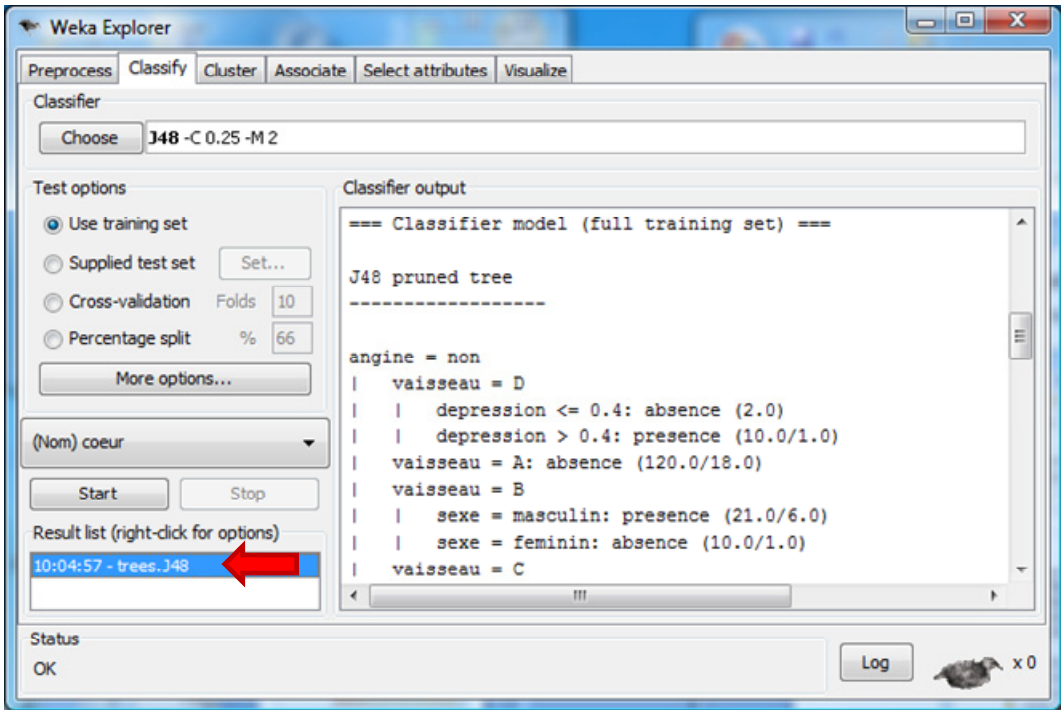


5 Format de description Weka et PDI-CE

Weka et PDI-CE appartiennent tous deux à la suite « Pentaho Community Edition ». Ils peuvent communiquer via un format de description natif (et binaire) des modèles.

5.1 Construction et sauvegarde du modèle sous Weka

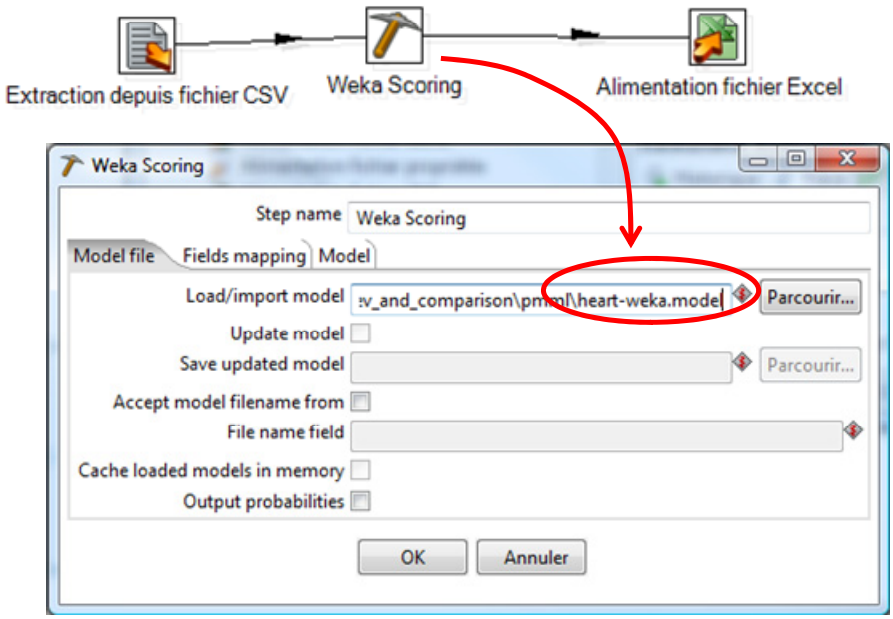
Il faut faire très attention aux versions des logiciels et des plugins dans cette partie. Nous avons utilisé la version 3.7.2 de Weka, la version de WekaScoring doit être à l'avenant. Après avoir démarré Weka en mode « Explorer » et chargé le fichier « heart-train.arff », nous activons l'onglet « Classify ». Nous sélectionnons la méthode J48, puis nous actionnons le bouton START.



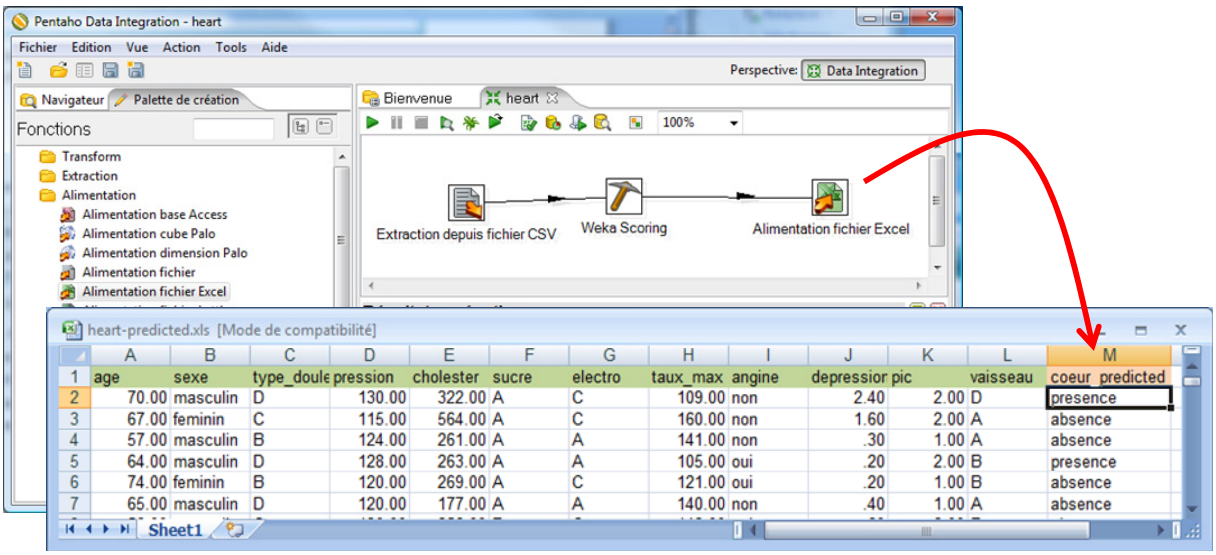
Pour sauver le modèle prédictif, nous réalisons un clic-droit sur l'arbre produit, et nous actionnons l'item SAVE MODEL du menu contextuel. Nous spécifions le nom « **heart-weka.model** ».

5.2 Exploitation du modèle sous PDI-CE

Nous revenons dans PDI-CE, nous utilisons le même diagramme que précédemment, sauf que nous modifions le modèle utilisé. Au lieu du fichier PMML issu de SIPINA, nous sélectionnons le fichier « .model » produit par Weka.



Il ne nous reste plus qu'à lancer le processus (menu ACTION / EXECUTER). Nous obtenons une nouvelle version du fichier « heart-predicted.xls ».



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	age	sexe	type_doule	pression	cholester	sucre	electro	taux_max	angine	depressior	pic	vaisseau	coeur_predicted
2	70.00	masculin	D	130.00	322.00	A	C	109.00	non	2.40	2.00	D	presence
3	67.00	feminin	C	115.00	564.00	A	C	160.00	non	1.60	2.00	A	absence
4	57.00	masculin	B	124.00	261.00	A	A	141.00	non	.30	1.00	A	absence
5	64.00	masculin	D	128.00	263.00	A	A	105.00	oui	.20	2.00	B	presence
6	74.00	feminin	B	120.00	269.00	A	C	121.00	oui	.20	1.00	B	absence
7	65.00	masculin	D	120.00	177.00	A	A	140.00	non	.40	1.00	A	absence

Remarque : A titre de curiosité, nous avons voulu confronter les prédictions de SIPINA avec celles de WEKA. Nous constatons qu'elles sont plus ou moins cohérentes : 227 (90+137) prédictions sur 270 sont concordantes. Nous ne savons pas en revanche, sans investigations supplémentaires (ex. évaluation des performances en validation croisée), lequel des deux est le plus précis.

		WEKA		
	Nombre de sipina	Étiquettes de		
	Étiquettes de	presence	absence	Total général
SIPINA	presence	90	25	115
	absence	18	137	155
	Total général	**108**	**162**	**270**

6 Conclusion

Le déploiement des modèles est (d'une certaine manière) l'aboutissement du Data Mining. Dans le cas de l'apprentissage supervisé, il s'agit de classer des individus avec un modèle préalablement construit. Nous avons mis l'accent sur le format de description PMML. Il cherche à s'imposer comme un standard de description. C'est sa principale force. S'il est adopté par tous, l'utilisation d'un outil ETL tel que PDI-CE pour le déploiement devient alors indépendant du logiciel de Data Mining utilisé pour produire le modèle de prédiction.

Bien entendu, PMML ne constitue certainement pas la panacée. C'est une solution comme une autre. Comme nous avons pu le constater, lorsque les logiciels se reconnaissent (ou appartiennent à la même suite, comme c'est le cas de Weka et PDI-CE), l'utilisation d'un format d'échange natif est possible. Il devrait en résulter des performances optimisées… *sinon on ne voit pas trop l'intérêt de produire un format propriétaire*.

Ce dernier point reste à vérifier. Une idée intéressante pour un prochain tutoriel serait justement de comparer la vitesse d'exécution du composant WekaScoring lors de l'appréhension de très gros fichiers, selon que le modèle est décrit avec le format binaire Weka ou avec la norme PMML.