1 Objectif

Comparer les implémentations des Règles d'Association chez Tanagra, R (package arules), Orange, RapidMiner, Knime et Weka.

Ce document reprend un précédent tutoriel dédié à la comparaison des implémentations libres des règles d'association[1]. Nous avions étudié Tanagra, Orange, et Weka. Nous étendons le comparatif aux logiciels R (package arules), RapidMiner et Knime.

Nos données se présentent sous la forme d'un tableau générique « attribut – valeur », avec les individus en ligne et les variables en colonne. Ce n'est pas le format usuel pour les règles d'association où l'on traite plutôt des bases transactionnelles : chaque ligne est une transaction, pour chaque transaction nous disposons de la liste des items observés.

Nous verrons dans ce didacticiel que certains logiciels savent traiter le format tableau en réalisant automatiquement en interne le recodage. Pour d'autres en revanche, il nous faudra procéder explicitement au recodage. Il importe alors de trouver les bons outils et la bonne séquence de traitements pour produire le format propice à l'extraction des règles d'association. Les manipulations ne sont pas toujours évidentes selon les logiciels.

2 Données et logiciels

Nous utilisons le fichier CREDIT-GERMAN.TXT[2]. Il décrit les caractéristiques de 1000 clients d'un organisme de crédit. Plutôt dédié à l'analyse supervisée, la variable « accord » ayant un rôle particulier, nous le mettons néanmoins en œuvre dans un schéma d'extraction de règles d'association, en accordant un statut identique à toutes les variables. Le fichier provient du serveur UCI[3]. Les variables continues ont été discrétisées. Dans la copie d'écran ci-dessous, nous pouvons voir une fraction des données.

credit-german.xls

	A	B	C	D	E	F	G	H
1	checking_status	disc_duration	credit_history	purpose	disc_amount	savings_status	employment	personal_status
2	<0	lo_1_year	critical/other existing	radio/tv	1000_2000	no known savings	>=7	male single
3	0<=X<200	up_2_years	existing paid	radio/tv	up_2000	<100	1<=X<4	female div/dep/mar
4	no checking	lo_1_year	critical/other existing	education	up_2000	<100	4<=X<7	male single
5	<0	up_2_years	existing paid	furniture/equipment	up_2000	<100	4<=X<7	male single
6	<0	1_2_years	delayed previously	new car	up_2000	<100	1<=X<4	male single
7	no checking	up_2_years	existing paid	education	up_2000	no known savings	1<=X<4	male single
8	no checking	1_2_years	existing paid	furniture/equipment	up_2000	500<=X<1000	>=7	male single
9	0<=X<200	up_2_years	existing paid	used car	up_2000	<100	1<=X<4	male single
10	no checking	lo_1_year	existing paid	radio/tv	up_2000	>=1000	4<=X<7	male div/sep
11	0<=X<200	up_2_years	critical/other existing	new car	up_2000	<100	unemployed	male mar/wid
12	0<=X<200	lo_1_year	existing paid	new car	1000_2000	<100	<1	female div/dep/mar
13	<0	up_2_years	existing paid	business	up_2000	<100	<1	female div/dep/mar
14	0<=X<200	lo_1_year	existing paid	radio/tv	1000_2000	<100	1<=X<4	female div/dep/mar
15	<0	1_2_years	critical/other existing	new car	1000_2000	<100	>=7	male single
16	<0	1_2_years	existing paid	new car	1000_2000	<100	1<=X<4	female div/dep/mar
17	<0	1_2_years	existing paid	radio/tv	1000_2000	100<=X<500	1<=X<4	female div/dep/mar
18	no checking	1_2_years	critical/other existing	radio/tv	up_2000	no known savings	>=7	male single
19	<0	up_2_years	no credits/all paid	business	up_2000	no known savings	<1	male single
20	0<=X<200	1_2_years	existing paid	used car	up_2000	<100	>=7	female div/dep/mar
21	no checking	1_2_years	existing paid	radio/tv	up_2000	500<=X<1000	>=7	male single
22	no checking	lo_1_year	critical/other existing	new car	up_2000	<100	1<=X<4	male single
23	<0	lo_1_year	existing paid	radio/tv	up_2000	500<=X<1000	1<=X<4	male single

credit-german

[1] http://tutoriels-data-mining.blogspot.com/2008/04/rgles-dassociation-orange-tanagra-et.html

[2] http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/credit-german.zip

[3] http://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

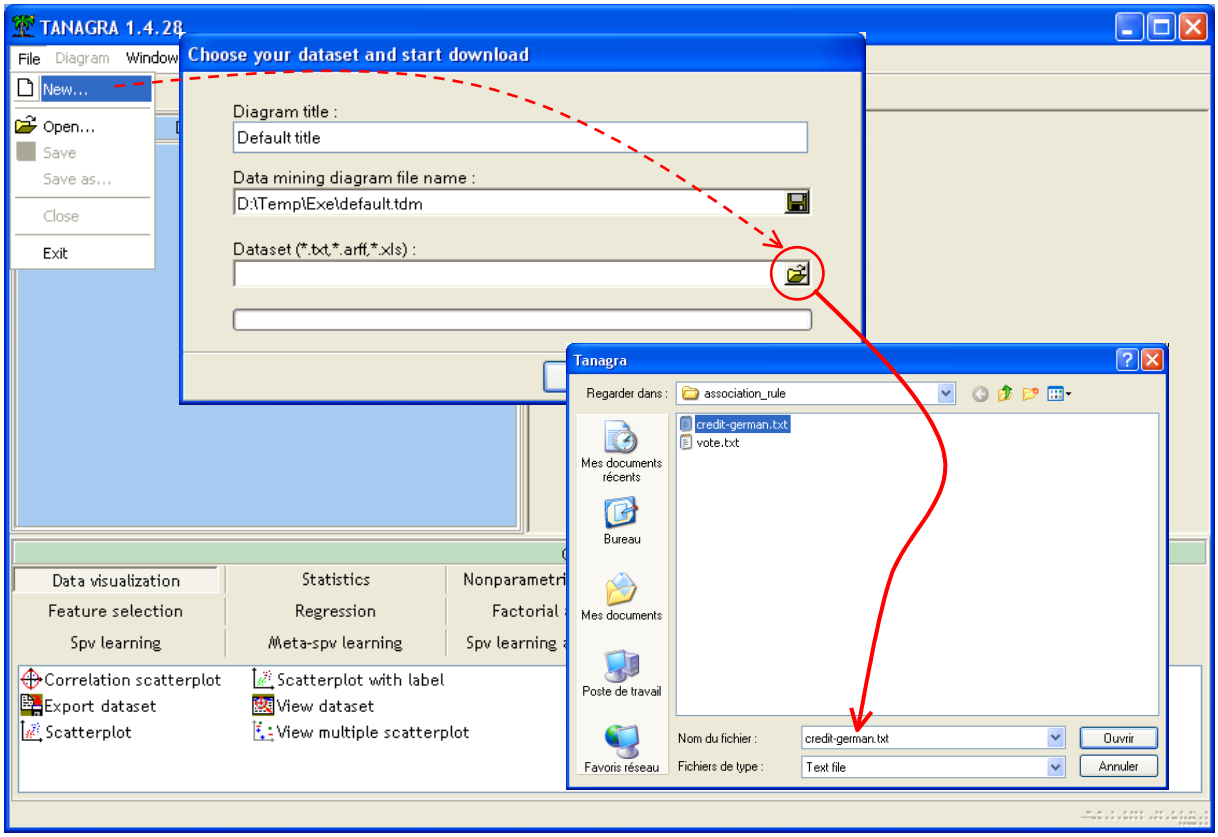
Tous les logiciels étudiés implémentent une version plus ou moins élaborée de l'algorithme A PRIORI (Agrawal et Srikant, 1994). Nous utiliserons systématiquement le même paramétrage afin que les résultats soient directement comparables : support minimum = 0.25 ; confiance minimale = 0.75 ; nombre maximal d'items dans une règle = 10.

Pour être tout à fait précis, et afin que tout un chacun puisse reproduire exactement les opérations, nous avons mis à contribution les versions suivantes dans ce comparatif : Tanagra 1.4.28 ; R 2.7.2 (package arules 0.6-6) ; Orange 1.0b2 ; RapidMiner Community Edition ; Knime 1.3.5 et Weka 3.5.6.

Tous ces logiciels chargent la totalité des données et effectuent les calculs en mémoire vive. Lorsque la taille de la base augmente, le véritable goulot d'étranglement est donc la mémoire disponible sur notre machine.

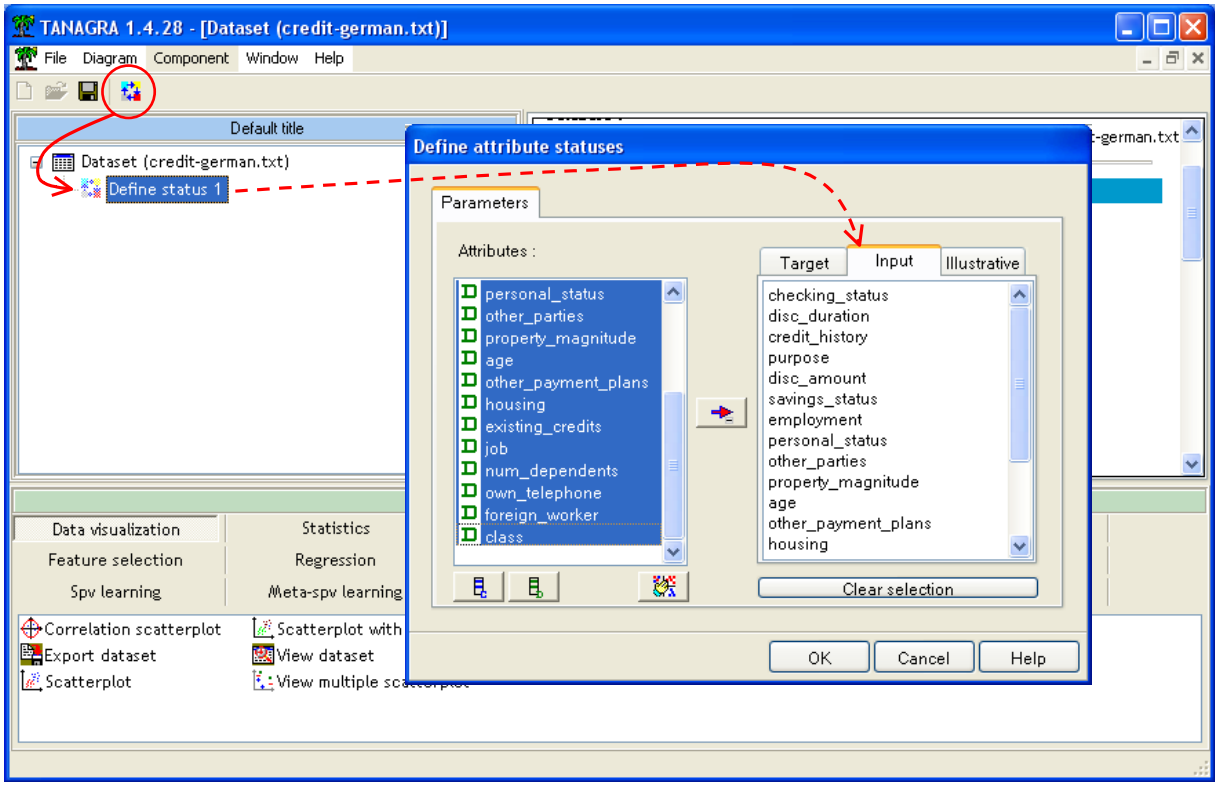
3 Tanagra (composant A Priori)

Création d'un diagramme et importation des données. Au démarrage de TANAGRA, nous créons un nouveau diagramme en actionnant le menu FILE / NEW. Nous sélectionnons le fichier CREDIT-GERMAN.TXT.

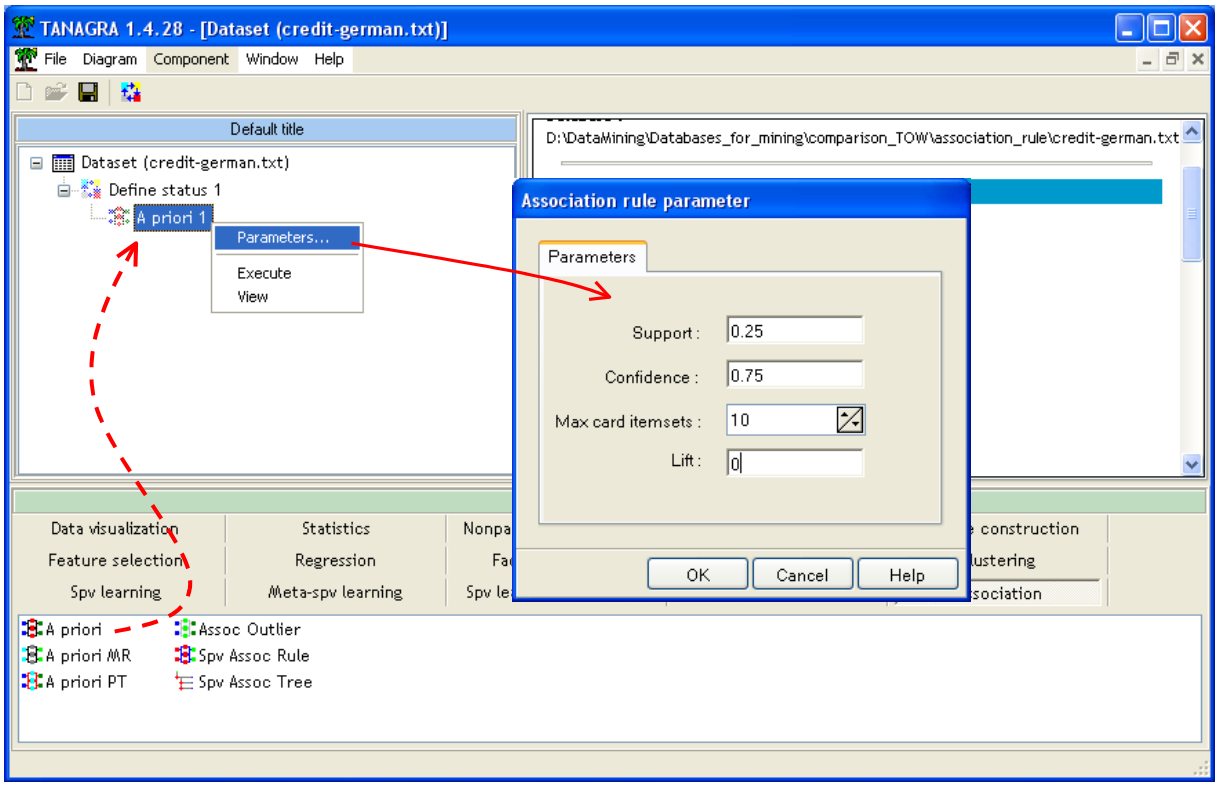


Tanagra nous indique que 19 variables et 1000 observations ont été chargées.

Définir le rôle des variables. Nous devons spécifier le rôle joué par chaque variable dans l'étude. Nous utilisons le composant DEFINE STATUS accessible dans la barre d'outils. Nous plaçons la totalité des variables en INPUT.

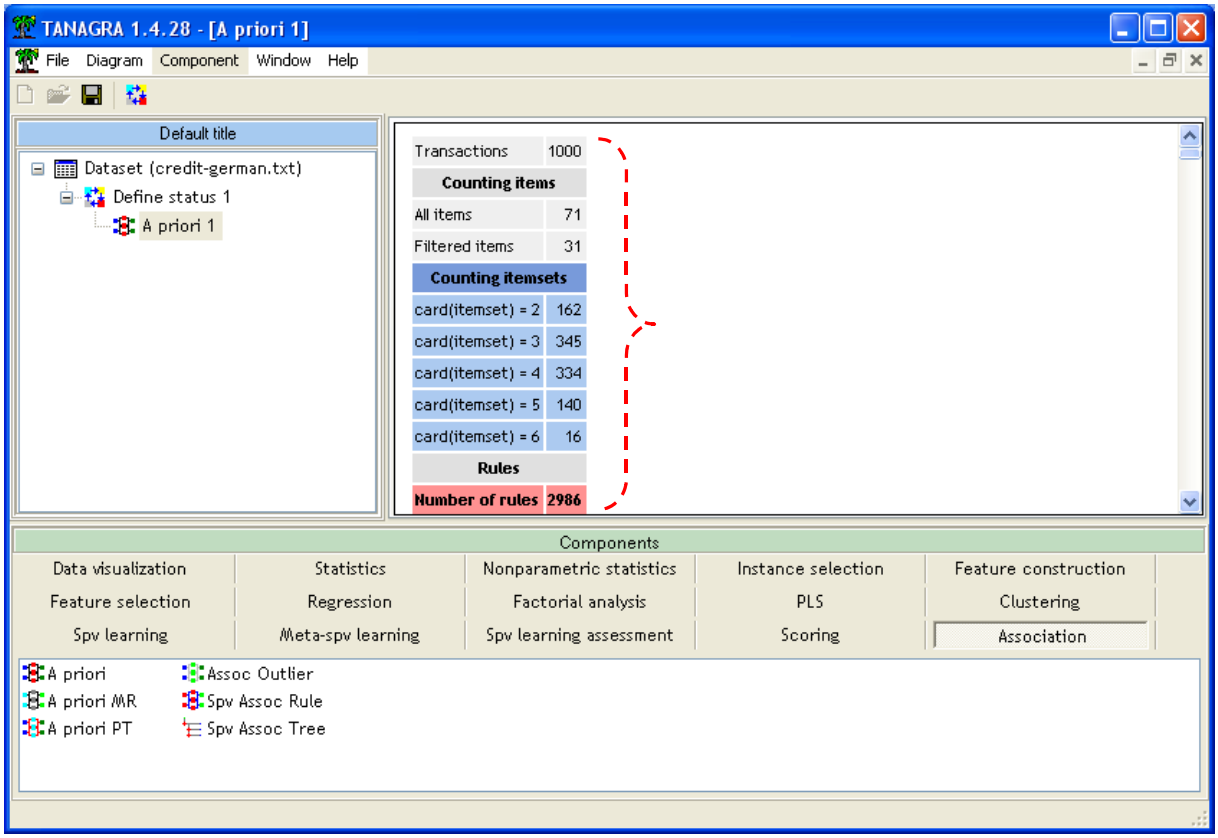


Extraire les règles. Nous pouvons insérer le composant A PRIORI (onglet ASSOCIATION) dans le diagramme. Nous actionnons le menu PARAMETERS pour définir les paramètres du traitement.



Le critère LIFT est mis à 0 afin qu'il n'intervienne pas dans le filtrage des règles.

Nous validons et nous actionnons le menu VIEW pour obtenir les résultats.



Tanagra fournit plusieurs indications : il y a 71 items (couple attribut = valeur) dans le fichier ; 31 ont passé le seuil de support c.-à-d. support >=0.25 ; nous disposons du nombre d'itemsets de cardinal équivalent c.-à-d. 162 itemsets de cardinal 2, etc. ; au final, 2986 règles ont été extraites.

Les règles sont énumérées dans la partie basse de la fenêtre. Elles sont triées selon un ordre décroissant du LIFT. Les règles les plus intéressantes sont affichées en premier (Note : ouh là, avec tous les bémols d'usage, je vois déjà les ayatollahs des mesures d'intérêt des règles lever les sourcils...).

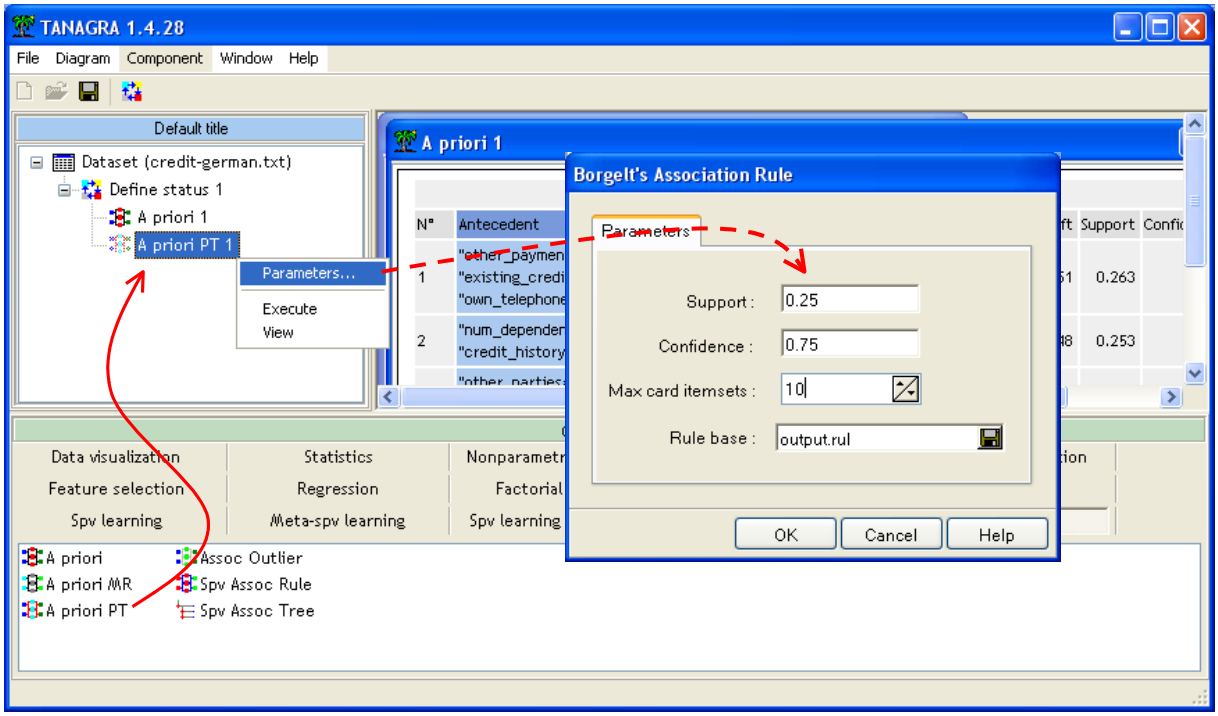
Number of rules : 2986

N°	Antecedent	Consequent	Lift	Support	Confidence
1	"other_payment_plans=none" - "existing_credits=one" - "own_telephone=none"	"credit_history=existing paid"	1.551	0.263	0.822
2	"num_dependents=one" - "class=good" - "credit_history=existing paid"	"other_payment_plans=none" - "existing_credits=one"	1.548	0.253	0.808
3	"other_parties=none" - "num_dependents=one" - "credit_history=existing paid"	"foreign_worker=yes" - "other_payment_plans=none" - "existing_credits=one"	1.534	0.319	0.769
4	"class=good" - "credit_history=existing paid"	"other_payment_plans=none" - "existing_credits=one"	1.534	0.289	0.801
5	"own_telephone=none" - "credit_history=existing paid"	"other_payment_plans=none" - "existing_credits=one"	1.527	0.263	0.797

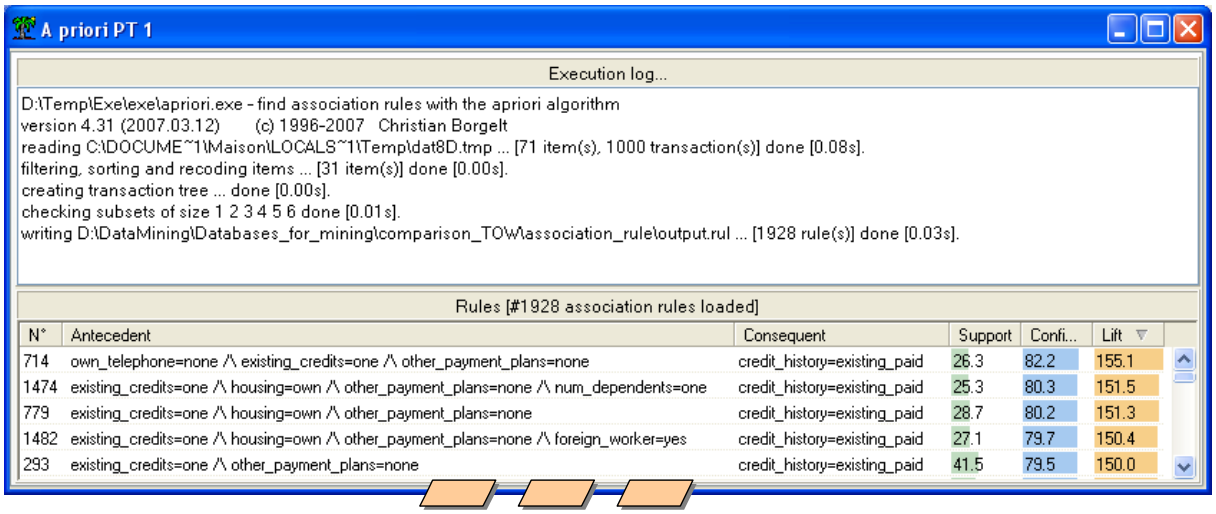
4 Tanagra (composant A Priori PT)

Tanagra intègre un second composant dédié à l'extraction des règles d'association. Il s'agit d'un programme externe « apriori.exe » de Christian BORGELT (http://www.borgelt.net/apriori.html). Nous avons eu l'occasion de le décrire de manière détaillée dans un de nos anciens tutoriels (http://tutoriels-data-mining.blogspot.com/2008/04/rgles-dassociation-avec-les-prefix-tree.html). Il présente deux particularités : il est très performant, capable d'appréhender des bases de taille assez conséquente (plus en tous les cas que le composant précédent) ; il introduit une limitation dans l'extraction, il ne produit que les règles comportant un seul conséquent. De fait, avec les paramètres ci-dessus, nous obtiendrons mécaniquement moins de règles.

Dans le diagramme de la section précédente, nous introduisons le composant A PRIORI PT (onglet ASSOCIATION). Nous actionnons le menu PARAMETERS.



Nous validons une fois les paramètres définis. Nous actionnons le menu VIEW.



Dans la fenêtre de résultats, la partie haute retranscrit les messages du programme de BORGELT (version 4.31) : 71 items ont été trouvés, 31 restent après filtrage, des itemsets de cardinal 1 à 6 ont été extraites. La partie basse énumère les règles. Nous disposons d'une fonctionnalité précieuse à ce stade, il est possible de trier les règles selon un des critères numériques d'appréciation. Il suffit de cliquer sur l'en-tête de la colonne.

Nous retrouvons ainsi la règle n°1 mise en évidence par le composant A PRIORI de la section précédente. En revanche, A PRIORI PT ne générant que les associations à un seul conséquent, les règles n° 2 à 8 du composant précédent ont disparu. Au total, une bonne partie passant à la trappe, « seules » 1928 règles sont extraites, bien moins que précédemment.

5 R (package arules)

Le package « arules » (http://cran.univ-lyon1.fr/web/packages/arules/index.html) permet d'extraire des règles d'association dans R (http://www.r-project.org/). Il s'agit aussi d'un portage du programme de BORGELT, avec ses avantages et ses inconvénients. Il produit notamment uniquement les règles à un seul item dans le conséquent. Par rapport au composant de Tanagra, nous devons introduire une étape supplémentaire dans le traitement, elle consiste à transformer explicitement les données « individus x variables » en données transactionnelles. L'opération est très facile... si on prend le temps de lire la documentation.

Chargement du package « arules ». La première étape consiste à installer le package. Puis, nous le chargeons en faisant appel à la commande **library(.)**

```
#charger le package
library(arules)
```

Importation et transformation des données. Nous chargeons le fichier au format texte avec **read.table(.)**, **summary(.)** donne un aperçu rapide des caractéristiques des variables.

```
#charger le fichier de données
setwd("D:/DataMining/Databases_for_mining/comparison_TOW/association_rule")
german <- read.table(file="credit-german.txt",header=T,dec=".",sep="\t")
summary(german)
```

Il n'est pas possible de produire directement les règles à partir d'un data.frame, nous devons transformer le format interne en données transactionnelles.

```
#transformer les données attributs-variables
#en données transactionnelles
german.trans <- as(german,"transactions")
summary(german.trans)
```

Nous avons toujours 71 items. Nous obtenons aussi une indication sur la densité des données (plus forte sera la densité, plus élevé sera le nombre de règles extraites par la méthode A Priori).

```
> summary(german.trans)
transactions as itemMatrix in sparse format with
 1000 rows (elements/itemsets/transactions) and
 71 columns (items) and a density of 0.2676056

most frequent items:
      foreign_worker=yes       other_parties=none       num_dependents=one other_payment_plans=none
                     963                      907                      845                      814
            housing=own                  (Other)
                     713                    14758

element (itemset/transaction) length distribution:
sizes
  19
1000

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     19      19      19      19      19      19

includes extended item information - examples:
                   labels       variables   levels
1       checking_status=<0 checking_status       <0
2    checking_status=>=200 checking_status    >=200
3 checking_status=0<=X<200 checking_status 0<=X<200

includes extended transaction information - examples:
  transactionID
1             1
2             2
3             3
```

Extraction des règles. L'étape suivante est l'extraction des règles.

```
#extraction des règles
german.regles <- apriori(german.trans,parameter=
                  list(supp=0.25,conf=0.75,minlen=2,maxlen=10,target="rules"))
summary(german.regles)
```

Nous retrouvons les sorties consoles du programme de BORGELT.

```
> #extraction des règles
> german.regles <- apriori(german.trans,parameter=
+                   list(supp=0.25,conf=0.75,minlen=2,maxlen=10,target="rules"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
       0.75    0.1    1 none FALSE            TRUE    0.25      2     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[71 item(s), 1000 transaction(s)] done [0.00s].
sorting and recoding items ... [31 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.01s].
writing ... [1928 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Nous disposons aussi du nombre d'itemsets selon les cardinalités, des statistiques sur les indicateurs de qualité des règles, etc.

```
> summary(german.regles)
set of 1928 rules

rule length distribution (lhs + rhs):sizes
  2   3   4   5   6
112 487 783 476  70

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   3.000   4.000   3.951   5.000   6.000

summary of quality measures:
    support         confidence          lift
 Min.   :0.2500   Min.   :0.7500   Min.   :0.8894
 1st Qu.:0.2700   1st Qu.:0.8397   1st Qu.:0.9972
 Median :0.2990   Median :0.8906   Median :1.0121
 Mean   :0.3294   Mean   :0.8859   Mean   :1.0437
 3rd Qu.:0.3580   3rd Qu.:0.9485   3rd Qu.:1.0475
 Max.   :0.8800   Max.   :0.9937   Max.   :1.5507

mining info:
         data ntransactions support confidence
 german.trans          1000    0.25       0.75
```

Inspection des règles. La commande inspect(.) permet de visualiser les règles, nous nous en tenons aux 10 premières ici.

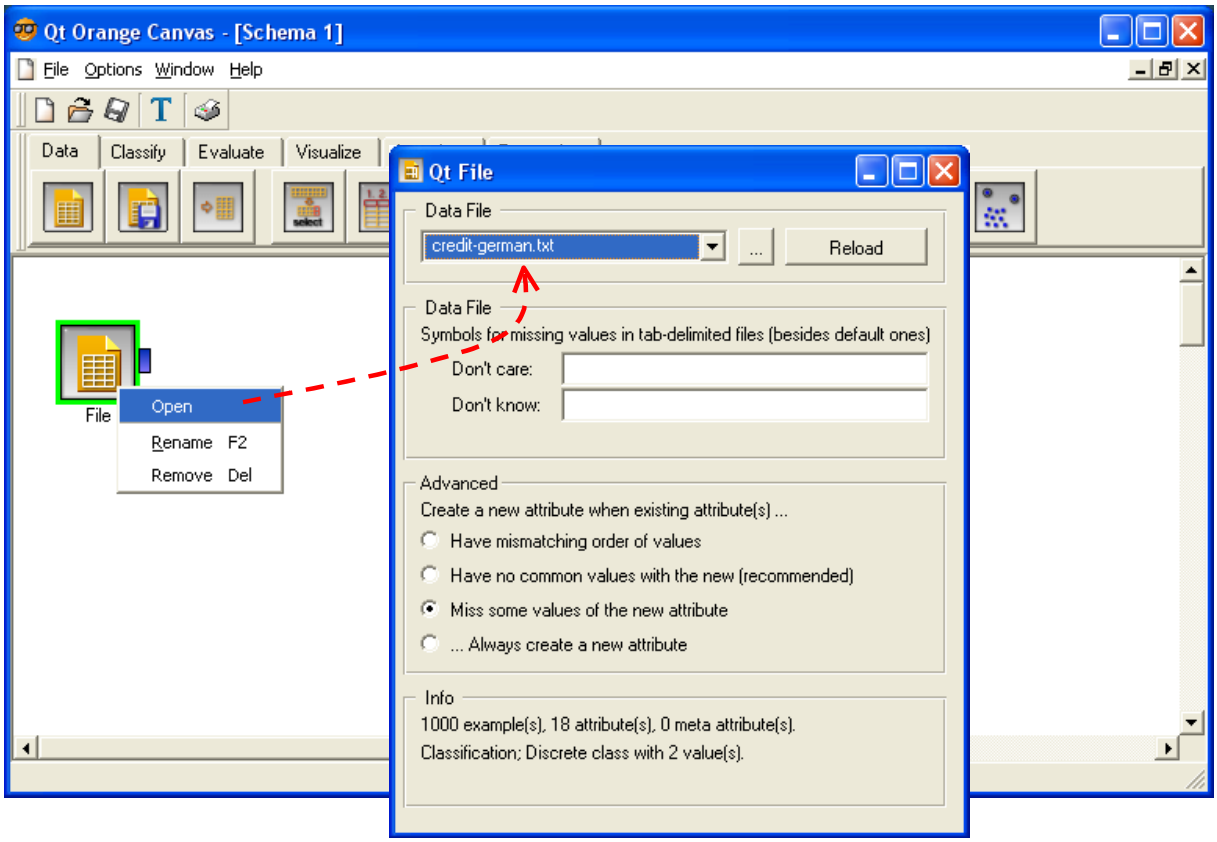
```
> #afficher les 10 premières règles trouvées
> inspect(german.regles[1:10])
   lhs                                          rhs                          support confidence      lift
1  {checking_status=0<=X<200}                => {foreign_worker=yes}           0.264  0.9814126 1.0191201
2  {checking_status=<0}                      => {foreign_worker=yes}           0.259  0.9452555 0.9815737
3  {purpose=radio/tv}                        => {num_dependents=one}           0.250  0.8928571 1.0566357
4  {purpose=radio/tv}                        => {foreign_worker=yes}           0.275  0.9821429 1.0198784
5  {property_magnitude=real estate}          => {foreign_worker=yes}           0.262  0.9290780 0.9647747
6  {credit_history=critical/other existing } => {other_payment_plans=none}     0.251  0.8566553 1.0524021
7  {credit_history=critical/other existing } => {other_parties=none}           0.268  0.9146758 1.0084628
8  {credit_history=critical/other existing } => {foreign_worker=yes}           0.279  0.9522184 0.9888042
9  {class=bad}                               => {num_dependents=one}           0.254  0.8466667 1.0019724
10 {class=bad}                               => {other_parties=none}           0.272  0.9066667 0.9996325
```

Nous pouvons aussi trier les règles selon un des indicateurs d'évaluation. Dans le cas du LIFT, nous obtiendrons les 5 premières règles suivantes, les mêmes que celles qui ont été mises en avant avec le composant A PRIORI PT de Tanagra.

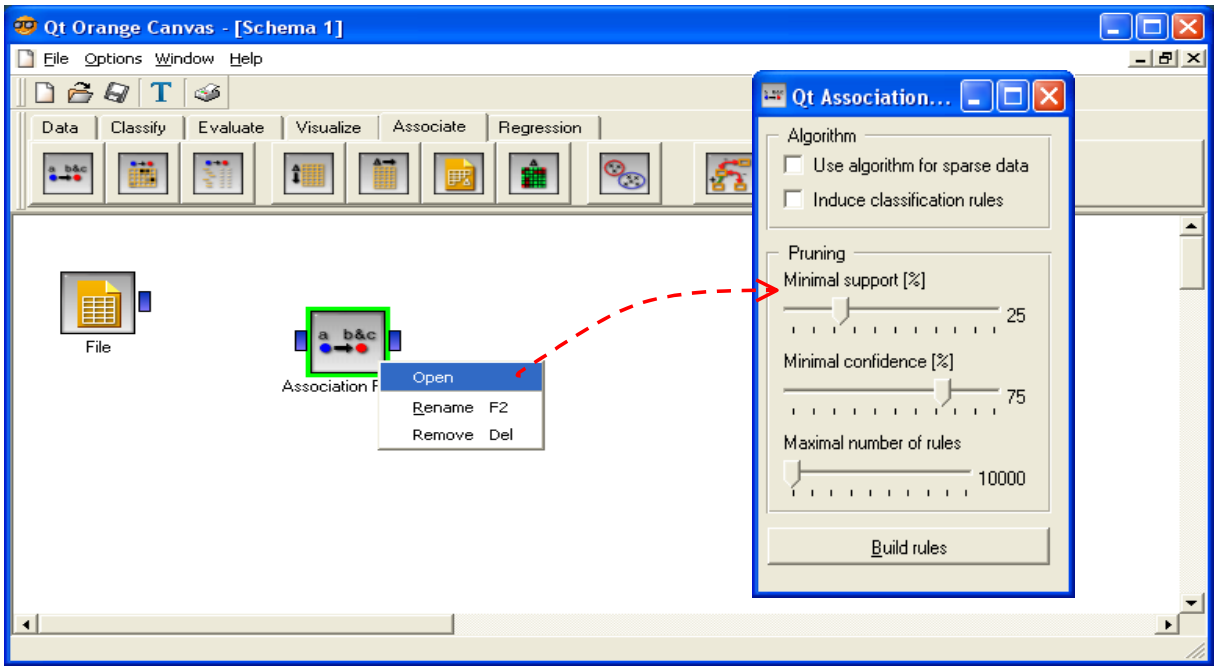
```
> #afficher les 5 règles avec le lift le + élevé
> regles.triees <- sort(german.regles,by="lift")
> inspect(regles.triees[1:5])
  lhs                                rhs                                support confidence     lift
1 {other_payment_plans=none,
   existing_credits=one,
   own_telephone=none}            => {credit_history=existing paid}    0.263  0.8218750 1.550708
2 {other_payment_plans=none,
   housing=own,
   existing_credits=one,
   num_dependents=one}            => {credit_history=existing paid}    0.253  0.8031746 1.515424
3 {other_payment_plans=none,
   housing=own,
   existing_credits=one}          => {credit_history=existing paid}    0.287  0.8016760 1.512596
4 {other_payment_plans=none,
   housing=own,
   existing_credits=one,
   foreign_worker=yes}            => {credit_history=existing paid}    0.271  0.7970588 1.503885
5 {other_payment_plans=none,
   existing_credits=one}          => {credit_history=existing paid}    0.415  0.7950192 1.500036
```

6 Orange

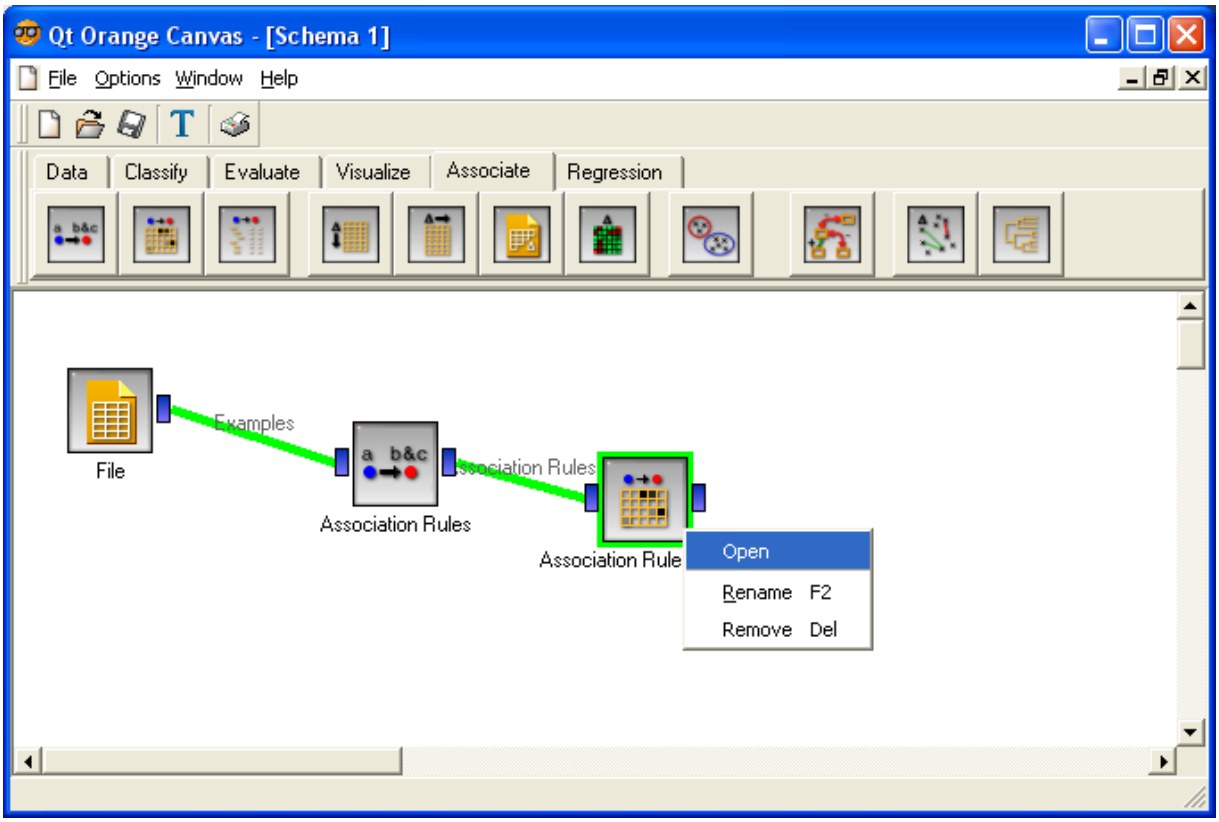
Création d'un schéma de traitements et importation des données. Au lancement de l'application, un schéma vierge est prêt. Le composant FILE (onglet DATA) sert à importer les données, nous actionnons le menu contextuel OPEN pour le paramétrer.



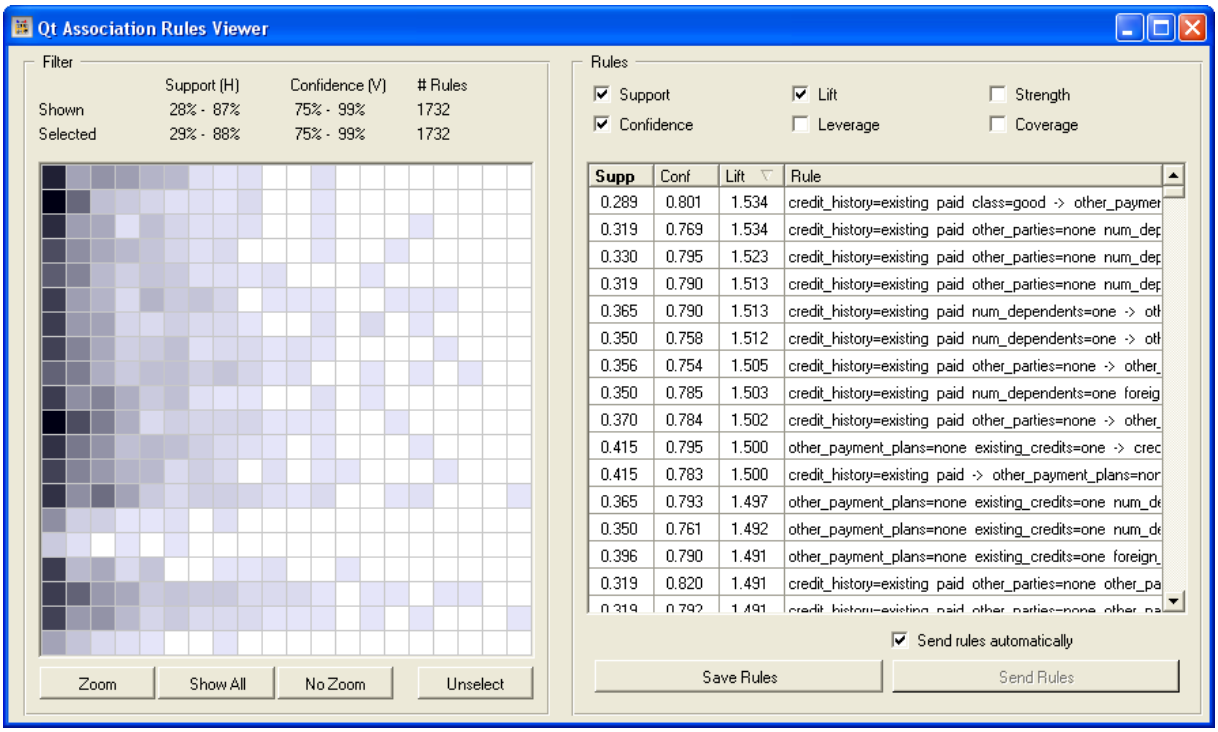
Extraction et visualisation des règles d'association. Nous insérons le composant ASSOCIATION RULES (onglet ASSOCIATE). Nous le paramétrons en actionnant le menu OPEN.



Nous insérons le composant ASSOCIATION RULES VIEWER, nous lui connectons le précédent. Nous pouvons dès lors connecter FILE à ASSOCIATION RULES. Le calcul est automatiquement démarré.



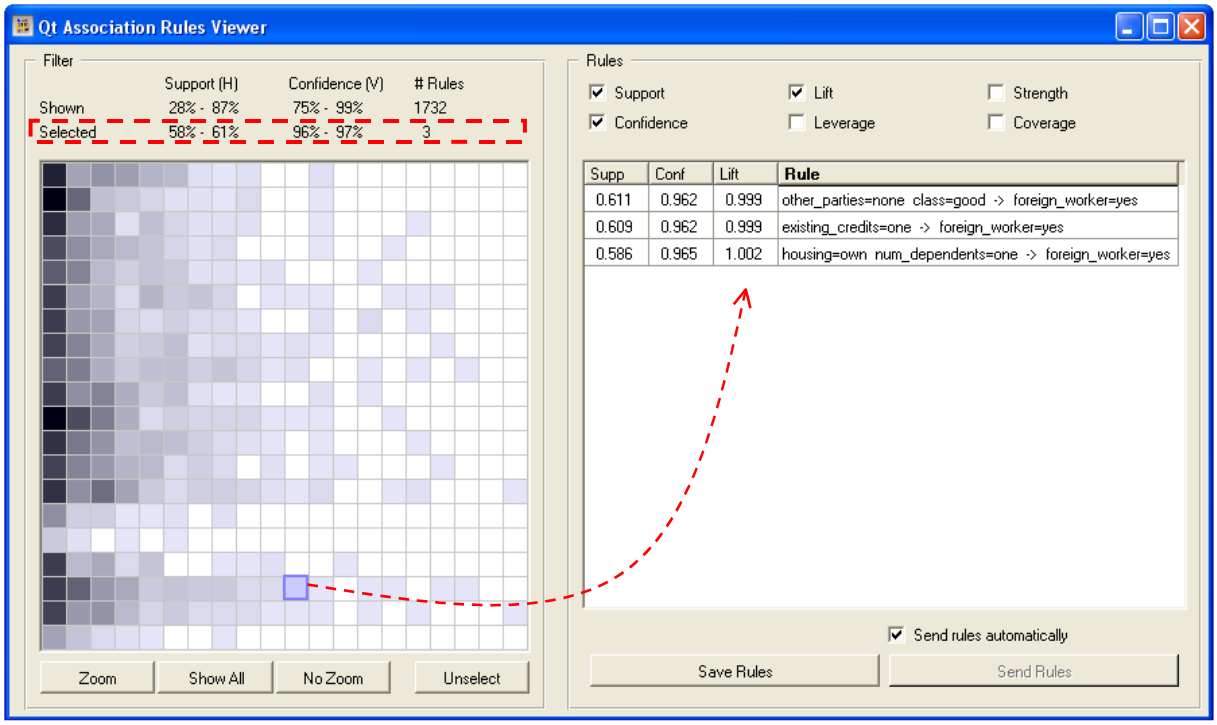
Pour accéder aux règles, nous cliquons sur le menu VIEW de ASSOCIATION RULES VIEWER.



La fenêtre de présentation est pour le moins originale. Nous apprenons que 1732 règles ont été extraites. Moins que pour les logiciels précédents, alors qu'il n'y a pas de limitation du nombre d'items dans le conséquent ici. Cette différence paraît assez mystérieuse. On constate par ailleurs

que le plus petit support des règles réellement extraite est 28.8%, il est de 75% en ce qui concerne la confiance. Lorsque nous relançons A PRIORI de Tanagra avec ces paramètres, nous obtenons effectivement 1732 règles. Je ne sais pas trop pourquoi malgré le paramétrage ci-dessus (25% support minimal et 75% confiance minimale), Orange ne sort que ces règles. Il faudrait analyser le détail de l'algorithme implémenté pour comprendre les différences avec les autres logiciels.

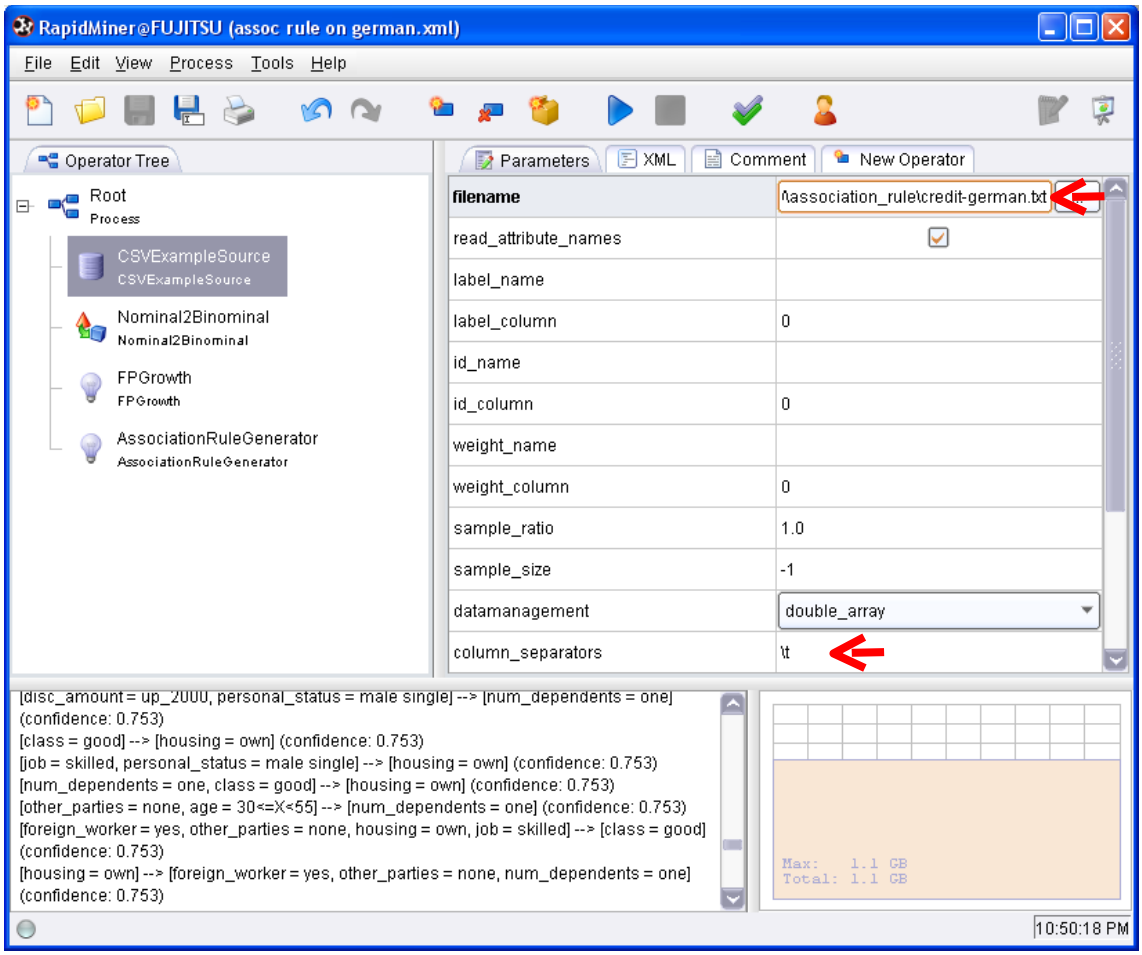
Pour en revenir à l'affichage, il est possible de trier les règles selon les critères numériques, il suffit de cliquer sur les en-têtes de colonnes. Nous pouvons rajouter d'autres critères. Option tout à fait intéressante, dans la partie gauche de la fenêtre, nous avons la densité des règles dans des zones définies par le couple de critères « support – confiance ». En cliquant sur un des rectangles, nous obtenons la liste des règles associées. Dans la copie d'écran ci-dessous, nous observons 3 règles dans la zone (58% < support < 61%) et (96% < confiance < 97%).



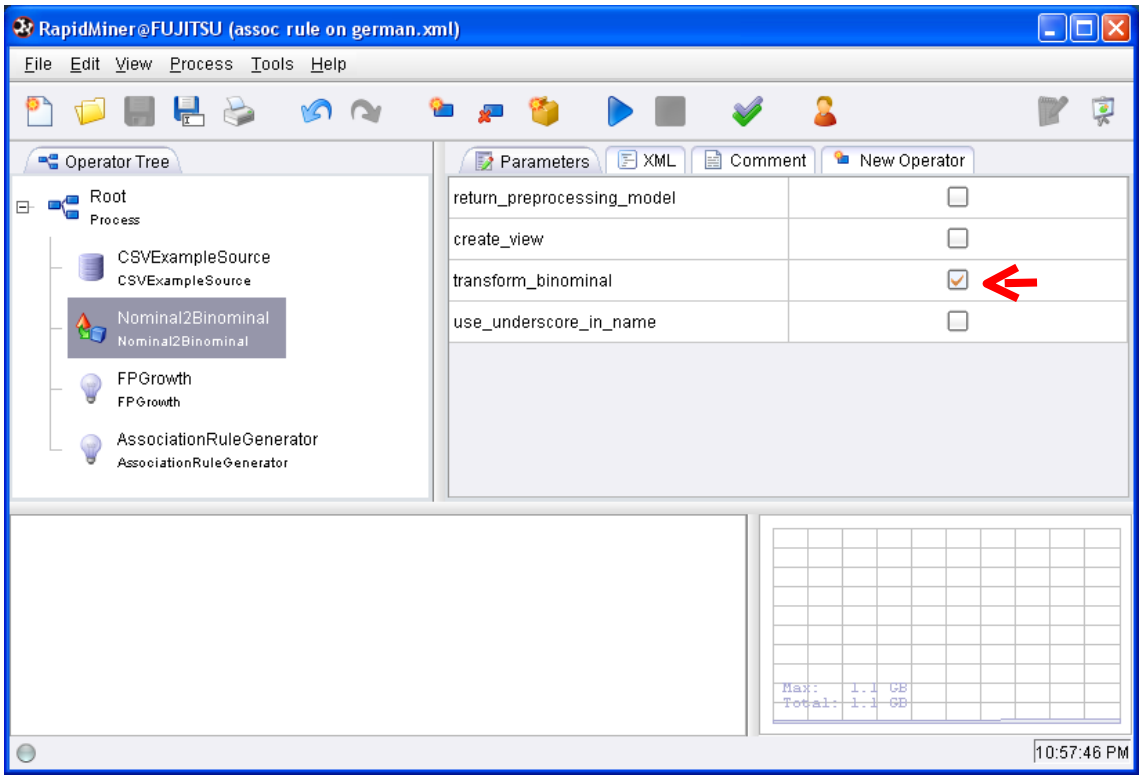
7 RapidMiner

Définition de l'arbre des opérations. Dans RapidMiner, il est plus judicieux de définir complètement la séquence de traitements avant de lancer les calculs. En effet, tous les opérateurs sont recalculés à chaque exécution.

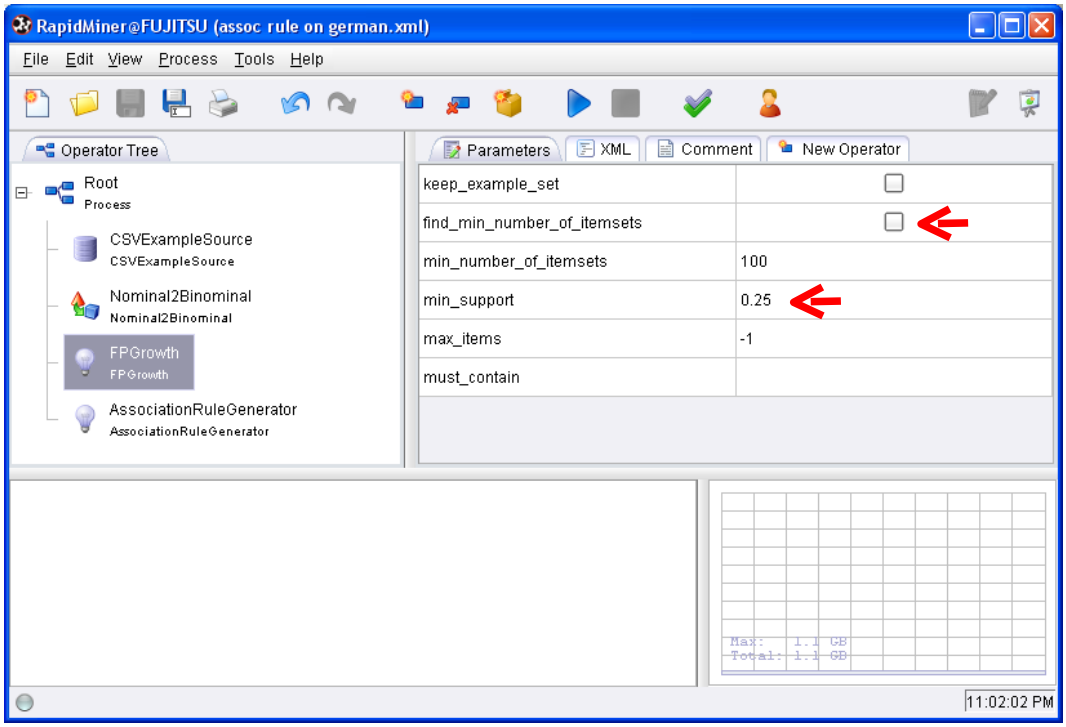
Le premier opérateur est **CSVEXAMPLESOURCE**, nous le paramétrons en désignant le nom du fichier et le séparateur de colonnes.



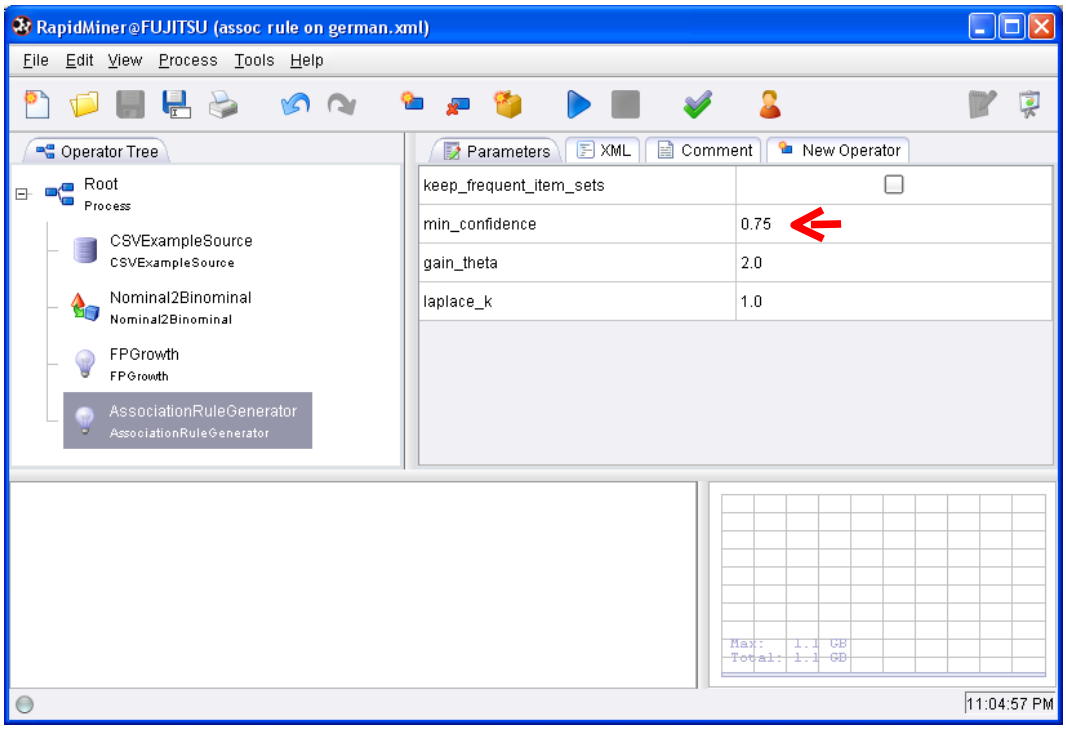
RapidMiner ne sait pas extraire les règles des données « individus x variables ». Nous réalisons un codage disjonctif complet à l'aide du composant **NOMINAL2BINOMIAL**.



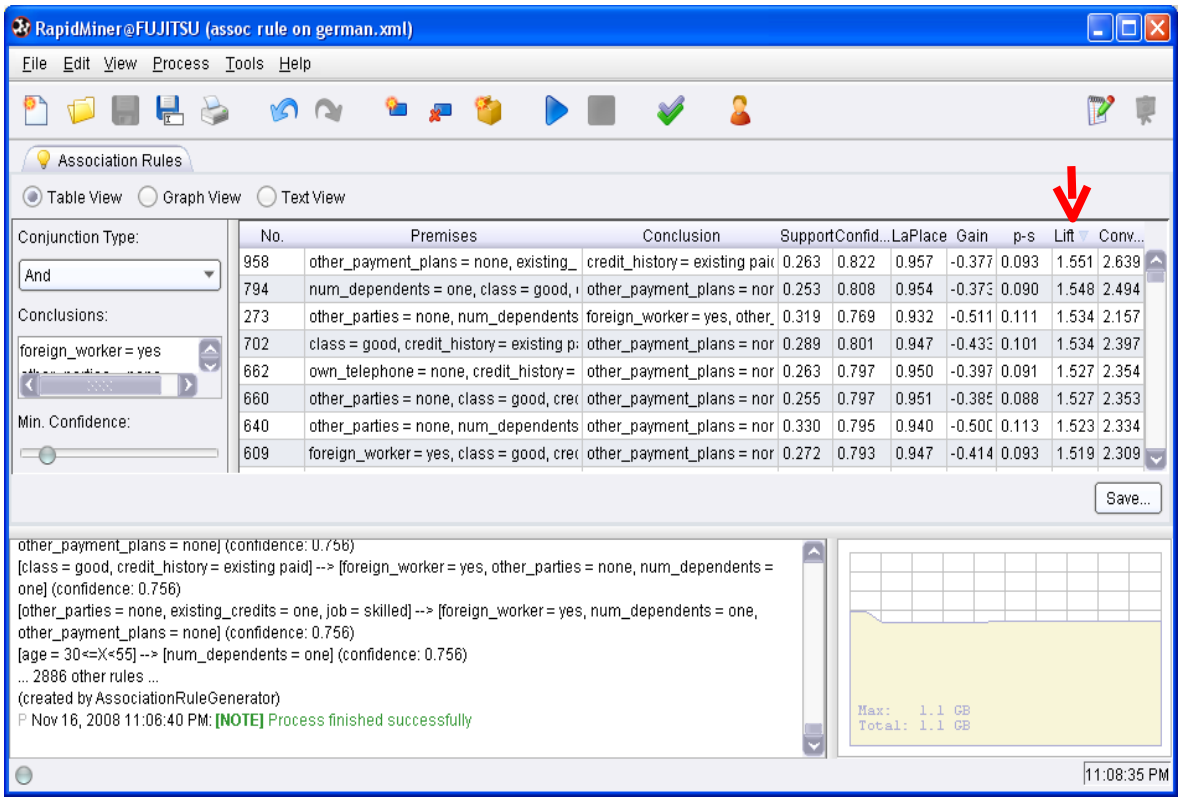
L'extraction se fait en deux étapes. Tout d'abord, le composant **FPGROWTH** extrait les itemsets fréquents. Des options non usuelles sont proposées par défaut (on peut définir le nombre d'itemsets à produire par exemple), nous devons faire attention pour obtenir un comportement similaire aux autres logiciels.



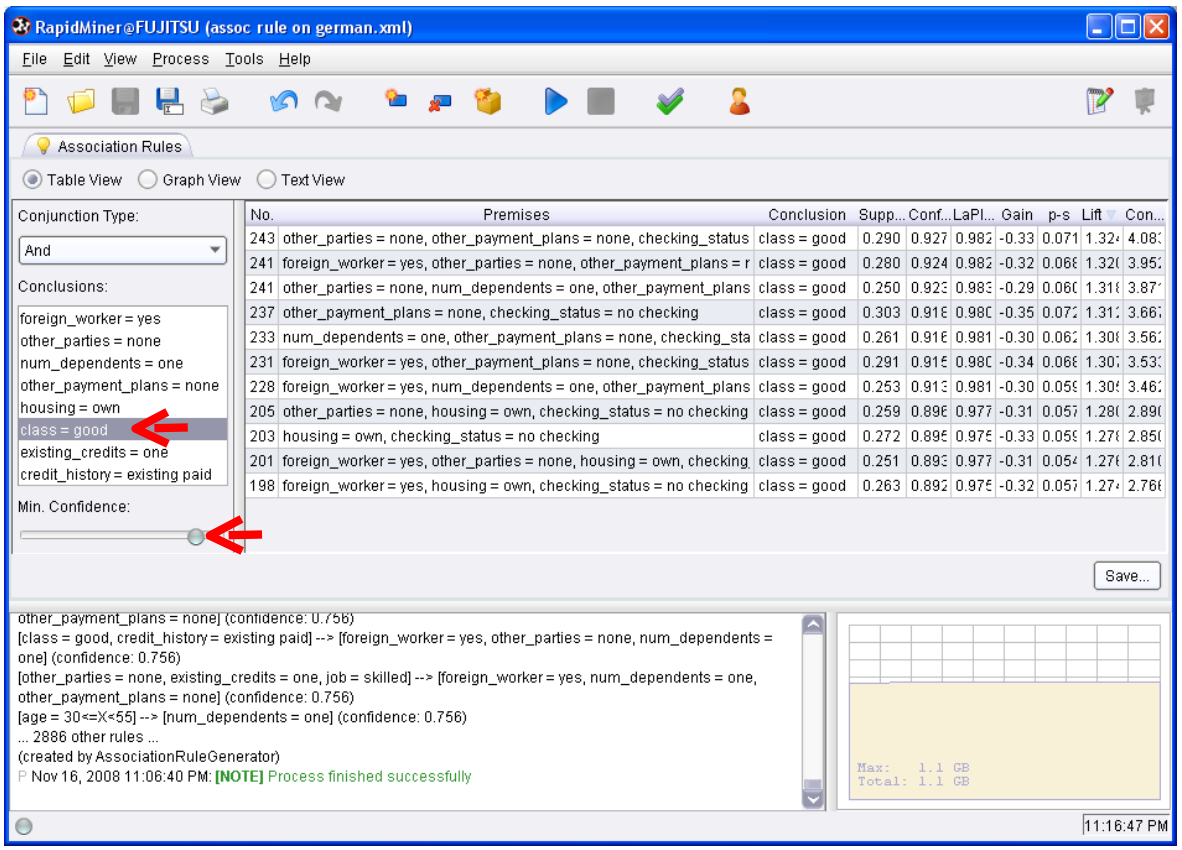
Ensuite, nous introduisons le composant **ASSOCIATIONRULEGENERATOR**. Nous fixons la confiance minimale.



Lancement des calculs et visualisation des résultats. Il ne nous reste plus qu'à lancer les calculs en cliquant sur le bouton PLAY dans la barre d'outils.



Il n'y a pas de limitation du nombre d'items dans le conséquent avec ce composant. Nous obtenons 2986 règles, les mêmes que ceux du composant A PRIORI de Tanagra. Plusieurs critères numériques d'évaluation sont proposés. Il est possible de trier les règles selon ces indicateurs. Dans notre copie d'écran, nous les avons triées selon le LIFT décroissant.

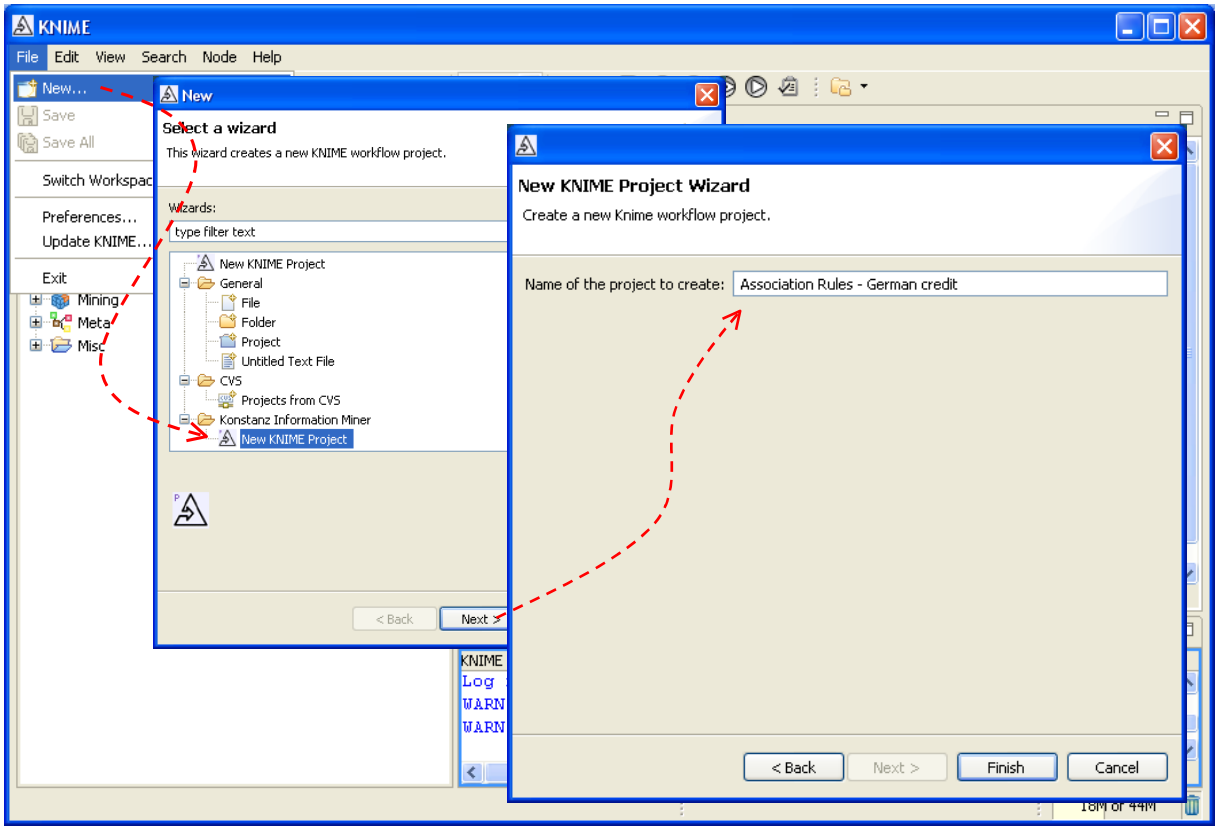


Option très intéressante, nous pouvons filtrer les règles selon le contenu du conséquent et la confiance. En choisissant un item dans la partie gauche de la fenêtre (CLASS = GOOD), et en élevant légèrement la confiance minimale, nous obtenons un sous ensemble de règles spécifique dans la copie d'écran ci-dessus. Nous pouvons aussi filtrer selon une conjonction (ET) ou une disjonction (OU) d'items. C'est assez remarquable.

8 Knime

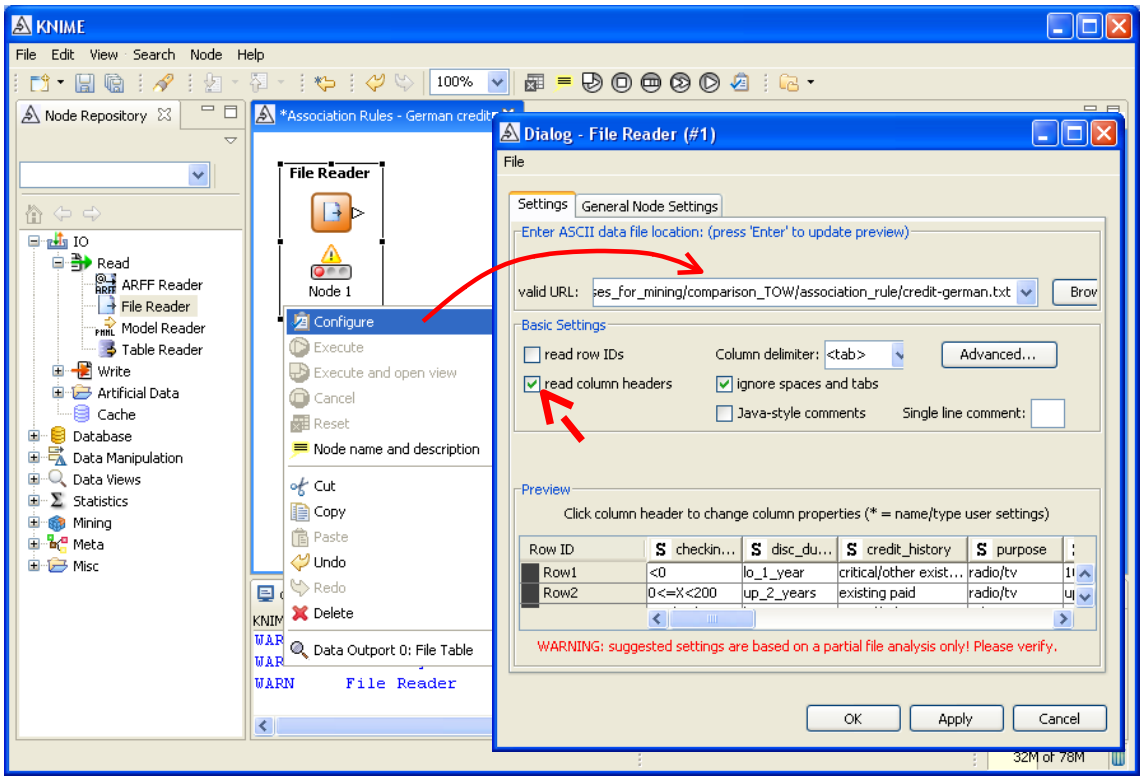
Une double préparation des données est nécessaire dans Knime avant de lancer l'algorithme d'apprentissage : un codage disjonctif complet, suivi d'une transformation en données transactionnelles. Voyons cela.

Création d'un workflow et importation des données. Nous créons un nouveau projet avec le menu FILE / NEW. Dans la boîte de dialogue qui apparaît, nous choisissons un projet de type Knime, puis nous spécifions son nom.

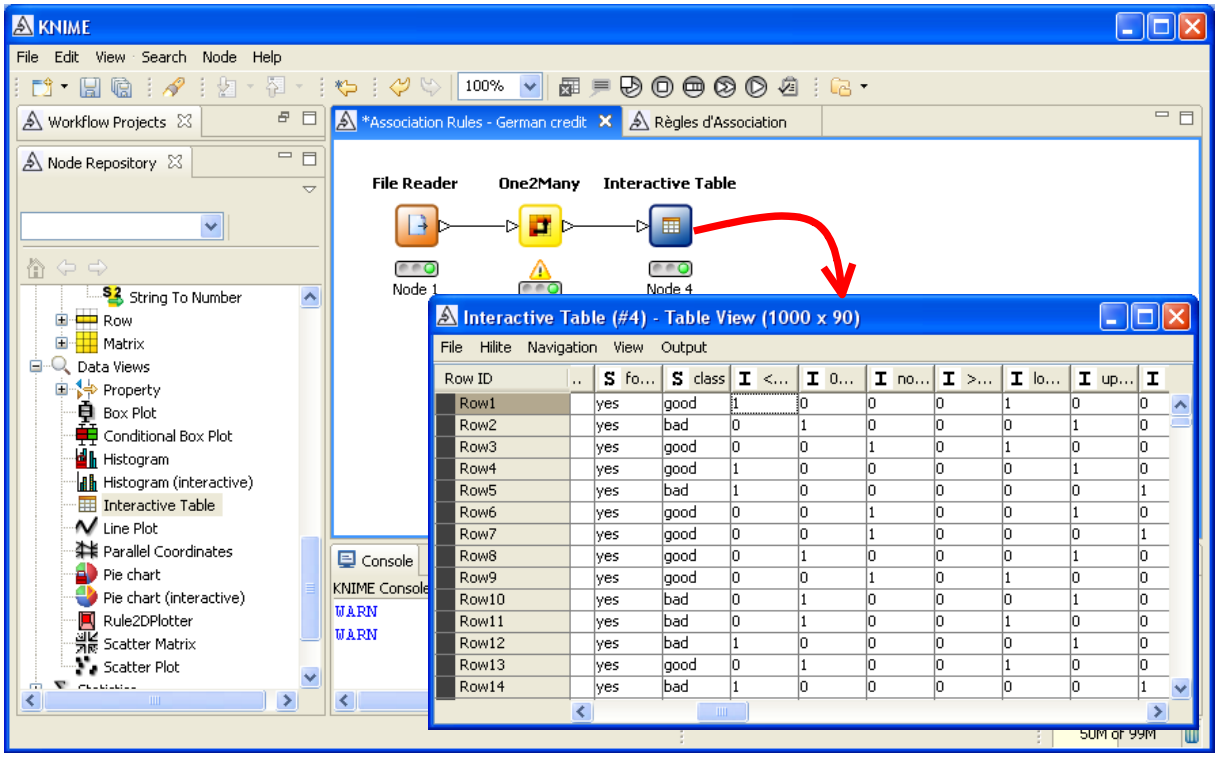


Un workflow vierge apparaît alors. La première étape consiste bien entendu à insérer le composant d'accès aux données. Nous utilisons l'outil FILE READER, nous le paramétrons en actionnant le menu CONFIGURE. Nous spécifions le nom du fichier, nous indiquons que la première ligne contient les noms de variables.

Nous validons puis nous cliquons sur EXECUTE pour charger effectivement les données. Le LED en dessous de composant doit passer au vert, indiquant le succès de l'opération.

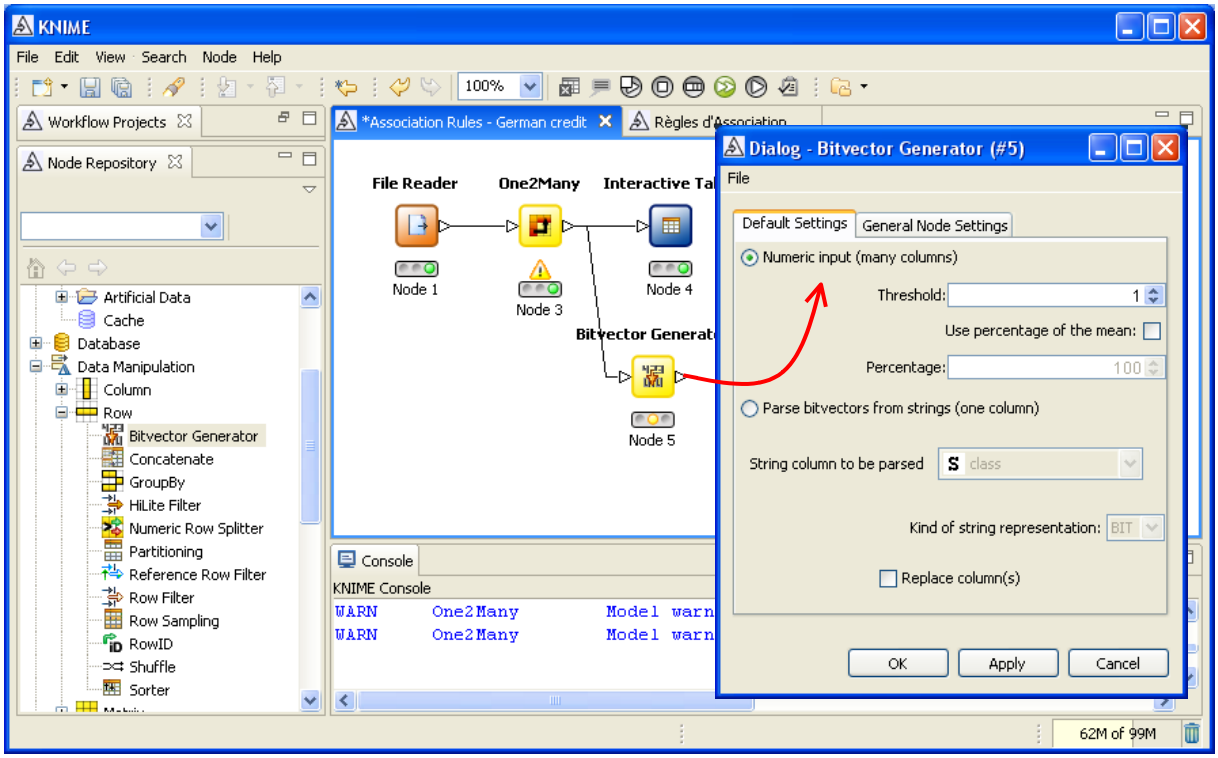


Codage disjonctif complet. Les variables catégorielles doivent être transformées en variables 0/1. Nous utilisons pour cela le composant ONE2MANY (DATA MANIPULATION / COLUMN). Nous lui connectons le composant précédent puis nous cliquons sur EXECUTE, il n'y a pas de paramétrage à faire ici. Pour visualiser le nouvel ensemble de données, nous ajoutons un composant de visualisation INTERACTIVE TABLE (DATA VIEWS) auquel nous connectons ONE2MANY. Les données recodées sont disponibles pour les opérateurs en aval.

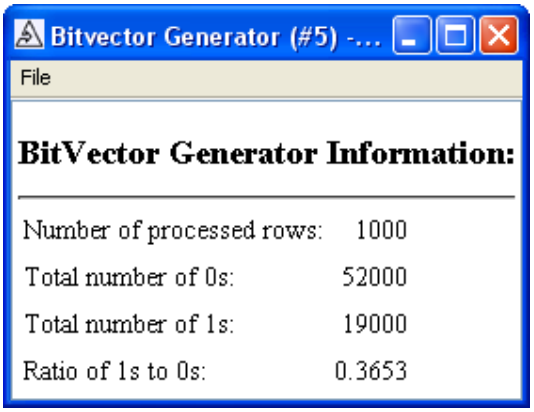


Il y a maintenant 90 colonnes dans l'ensemble de données, aux 19 variables initiales ont été ajoutées 71 items binaires 0/1 codées en variables numériques.

Transformation en données transactionnelles. Cette première transformation ne suffit pas. Il faut passer par un format interne spécifique, le format interne BITVECTOR, pour calculer les règles d'association. Nous introduisons le composant BITVECTOR GENERATOR (DATA MANIPULATION / COLUMN) dans le workflow. Nous lui connections le ONE2MANY puis nous actionnons le menu CONFIGURE pour le paramétrer. Nous nous assurons que seules les colonnes numériques, codées 0/1 précédemment, sont traitées.

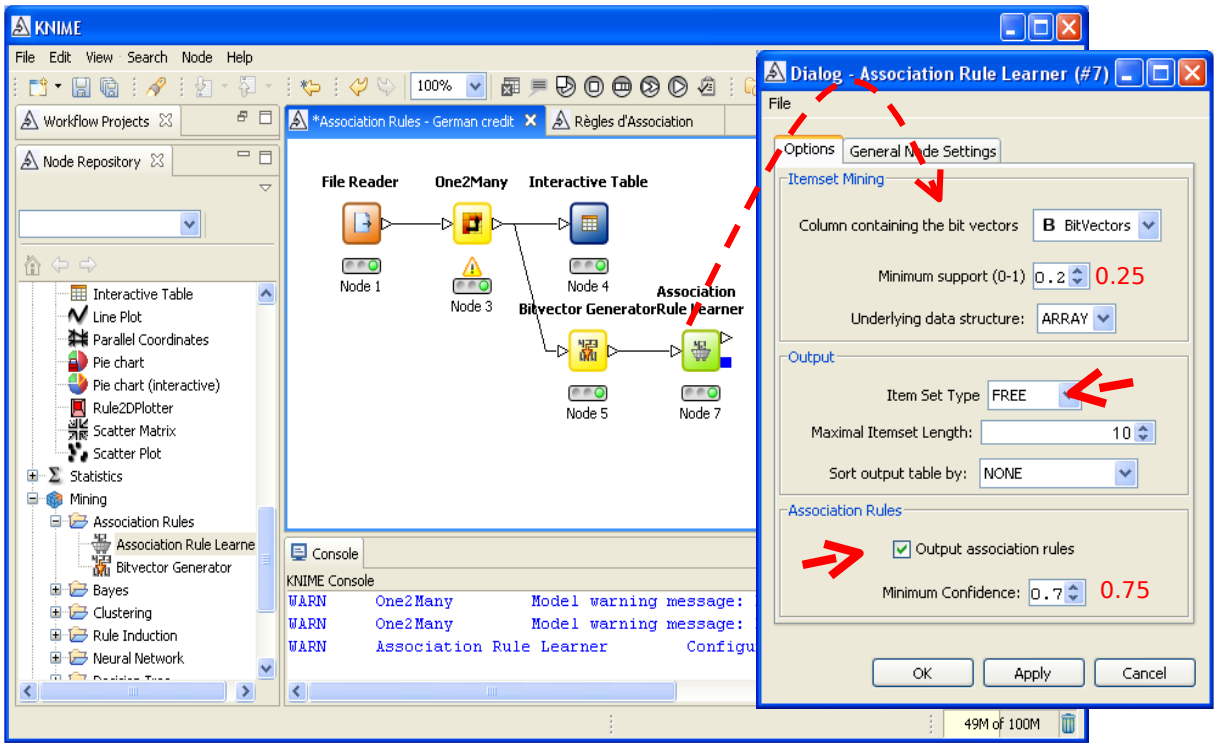


Nous actionnons le menu EXECUTE AND OPEN VIEW après le paramétrage, une fenêtre apparaît avec un rapport sur la structure des données transactionnelles.

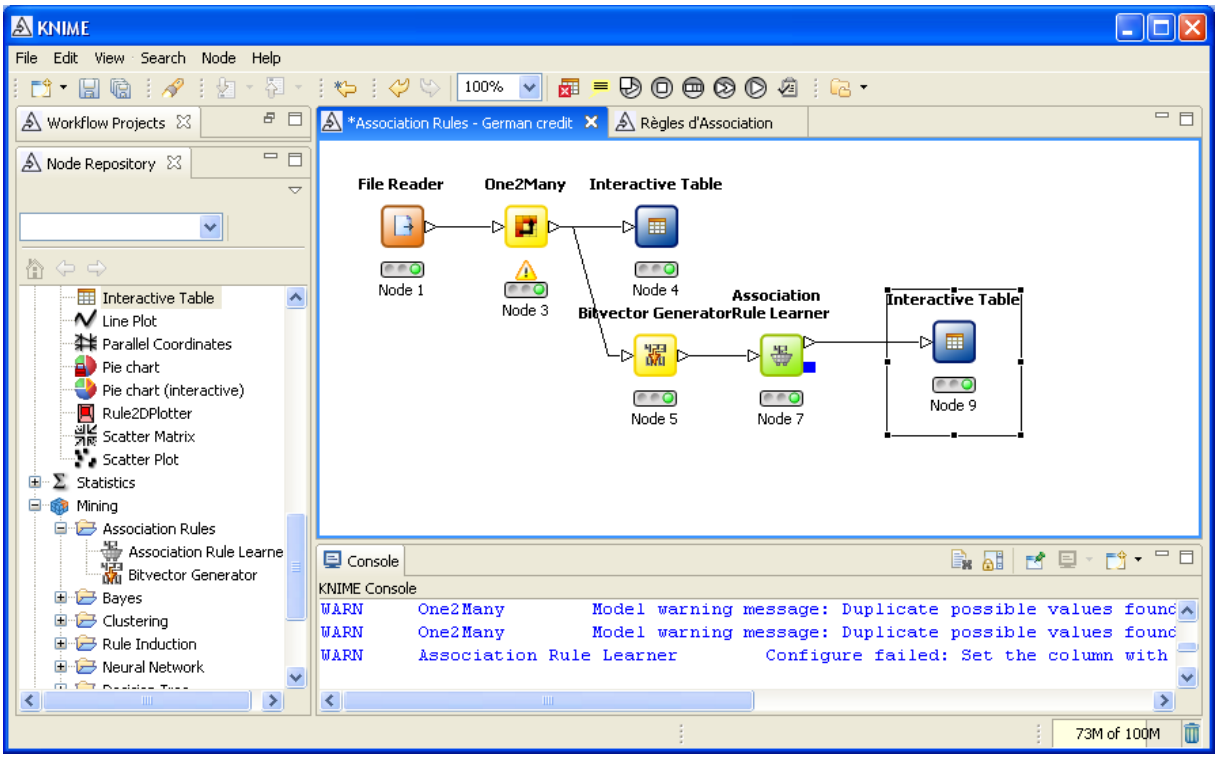


Pour retrouver l'indication de densité fournie par R, nous ferons le calcul 19000/(19000+52000) = 26.76%.

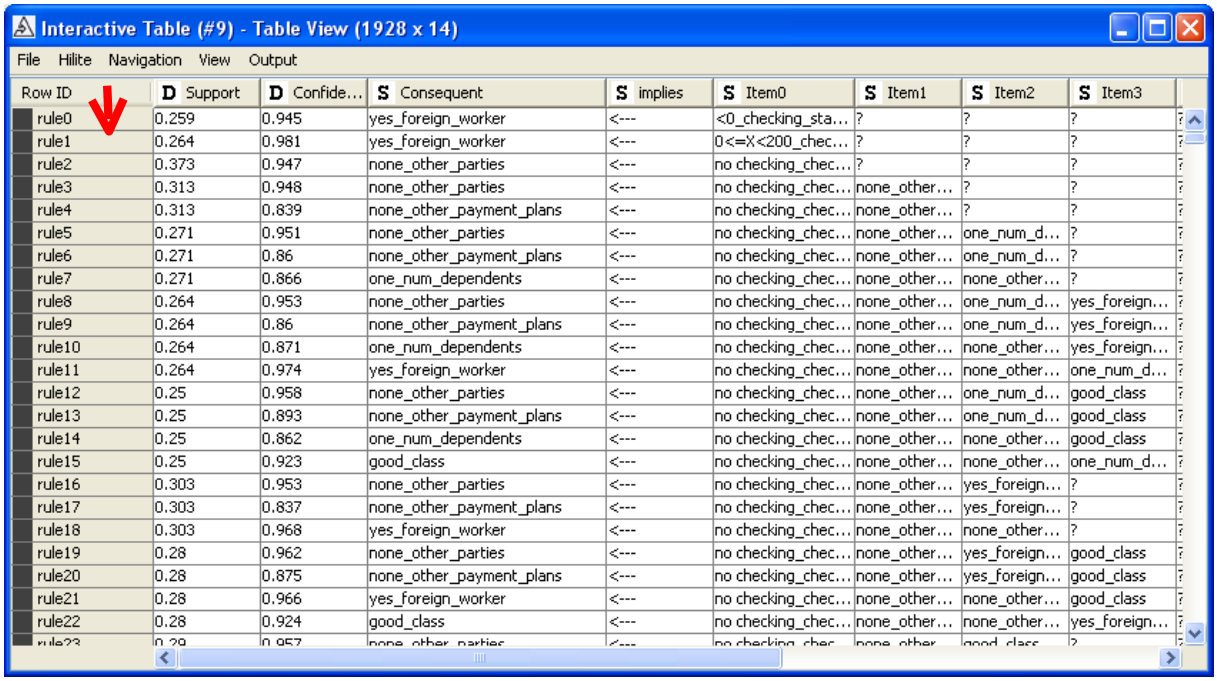
Extraction des règles. Nous pouvons maintenant initier l'extraction des règles, nous utilisons le composant ASSOCIATION RULE LEARNER. Nous le paramétrons de la manière suivante.



Nous utilisons le composant INTERACTIVE TABLE pour visualiser les règles.



KNIME ne produit que les règles à un seul conséquent. Il en a produit 1928 (numérotées de 0 à 1927 ! ça a failli m'induire en erreur), autant que le programme de BORGELT instancié dans R ou dans TANAGRA. L'outil de visualisation ne semble pas proposer de fonctionnalités interactives.

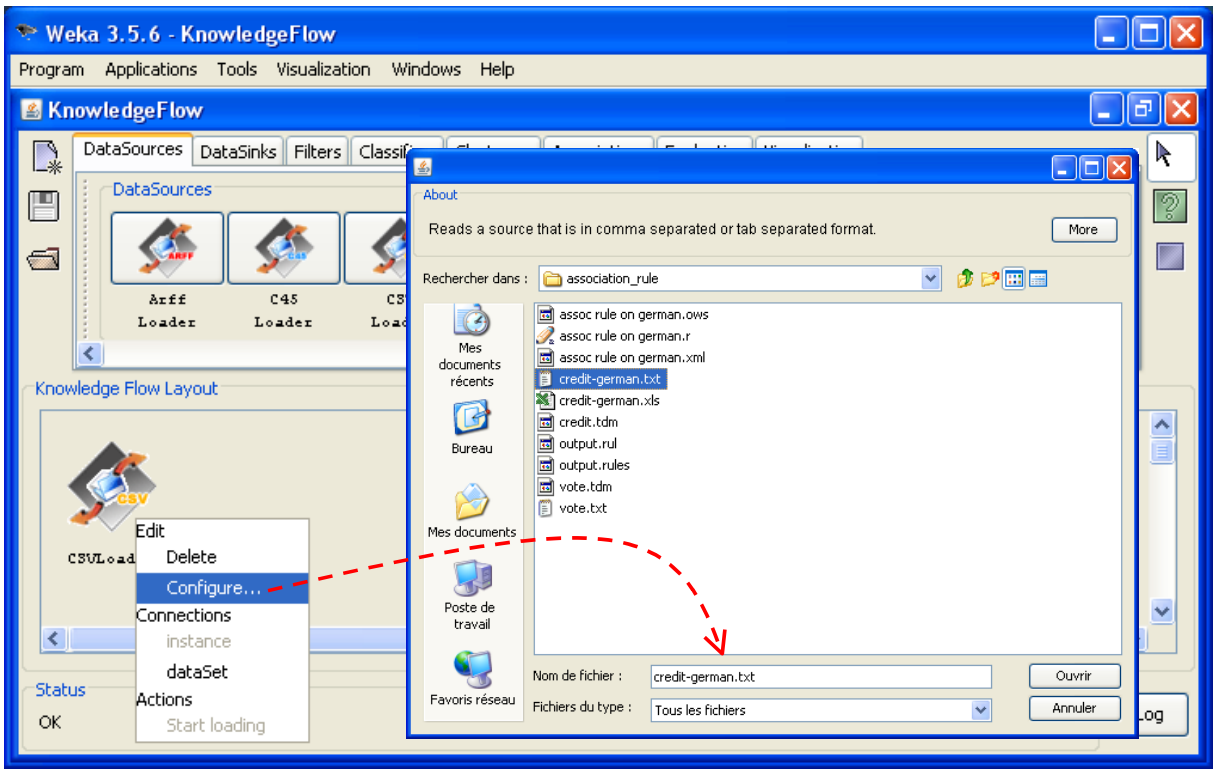


Row ID	Support	Confide...	Consequent	implies	Item0	Item1	Item2	Item3
rule0	0.259	0.945	yes_foreign_worker	<---	<0_checking_sta...	?	?	?
rule1	0.264	0.981	yes_foreign_worker	<---	0<=X<200_chec...	?	?	?
rule2	0.373	0.947	none_other_parties	<---	no checking_chec...	?	?	?
rule3	0.313	0.948	none_other_parties	<---	no checking_chec...	none_other...	?	?
rule4	0.313	0.839	none_other_payment_plans	<---	no checking_chec...	none_other...	?	?
rule5	0.271	0.951	none_other_parties	<---	no checking_chec...	none_other...	one_num_d...	?
rule6	0.271	0.86	none_other_payment_plans	<---	no checking_chec...	none_other...	one_num_d...	?
rule7	0.271	0.866	one_num_dependents	<---	no checking_chec...	none_other...	none_other...	?
rule8	0.264	0.953	none_other_parties	<---	no checking_chec...	none_other...	one_num_d...	yes_foreign...
rule9	0.264	0.86	none_other_payment_plans	<---	no checking_chec...	none_other...	one_num_d...	yes_foreign...
rule10	0.264	0.871	one_num_dependents	<---	no checking_chec...	none_other...	none_other...	yes_foreign...
rule11	0.264	0.974	yes_foreign_worker	<---	no checking_chec...	none_other...	none_other...	one_num_d...
rule12	0.25	0.958	none_other_parties	<---	no checking_chec...	none_other...	one_num_d...	good_class
rule13	0.25	0.893	none_other_payment_plans	<---	no checking_chec...	none_other...	one_num_d...	good_class
rule14	0.25	0.862	one_num_dependents	<---	no checking_chec...	none_other...	none_other...	good_class
rule15	0.25	0.923	good_class	<---	no checking_chec...	none_other...	none_other...	one_num_d...
rule16	0.303	0.953	none_other_parties	<---	no checking_chec...	none_other...	yes_foreign...	?
rule17	0.303	0.837	none_other_payment_plans	<---	no checking_chec...	none_other...	yes_foreign...	?
rule18	0.303	0.968	yes_foreign_worker	<---	no checking_chec...	none_other...	none_other...	?
rule19	0.28	0.962	none_other_parties	<---	no checking_chec...	none_other...	yes_foreign...	good_class
rule20	0.28	0.875	none_other_payment_plans	<---	no checking_chec...	none_other...	yes_foreign...	good_class
rule21	0.28	0.966	yes_foreign_worker	<---	no checking_chec...	none_other...	none_other...	good_class
rule22	0.28	0.924	good_class	<---	no checking_chec...	none_other...	none_other...	yes_foreign...

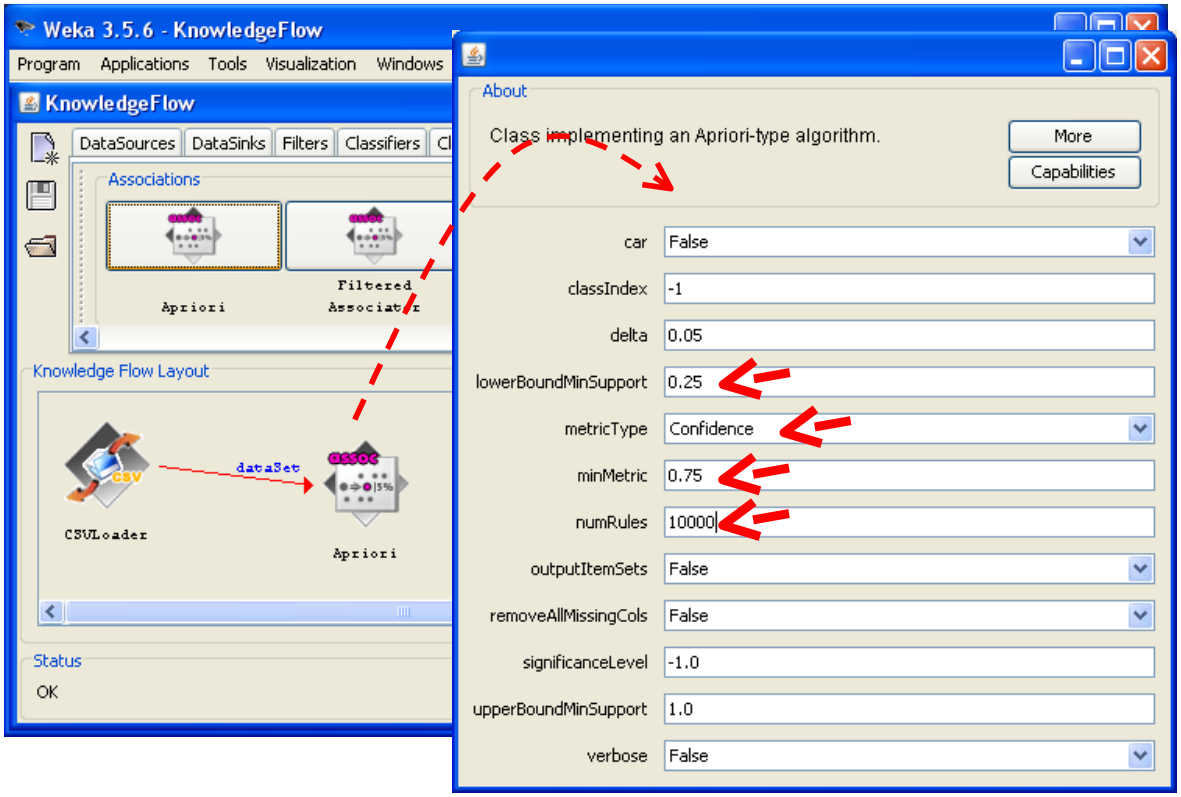
9 Weka

Dans Weka, tout comme dans RapidMiner, nous avons tout intérêt à définir complètement la séquence de traitements avant de lancer les calculs. Nous utilisons le mode KnowledgeFlow.

Définir les traitements. Au démarrage de Weka, nous actionnons le menu APPLICATIONS / KNOWLEDGEFLOW. Nous retrouvons un diagramme de traitements, assez similaire à celui d'Orange ou Knime. La première étape consiste tout naturellement à placer le composant d'accès aux données CSV LOADER (onglet DATASOURCES). Nous le configurons via le menu PARAMETERS.

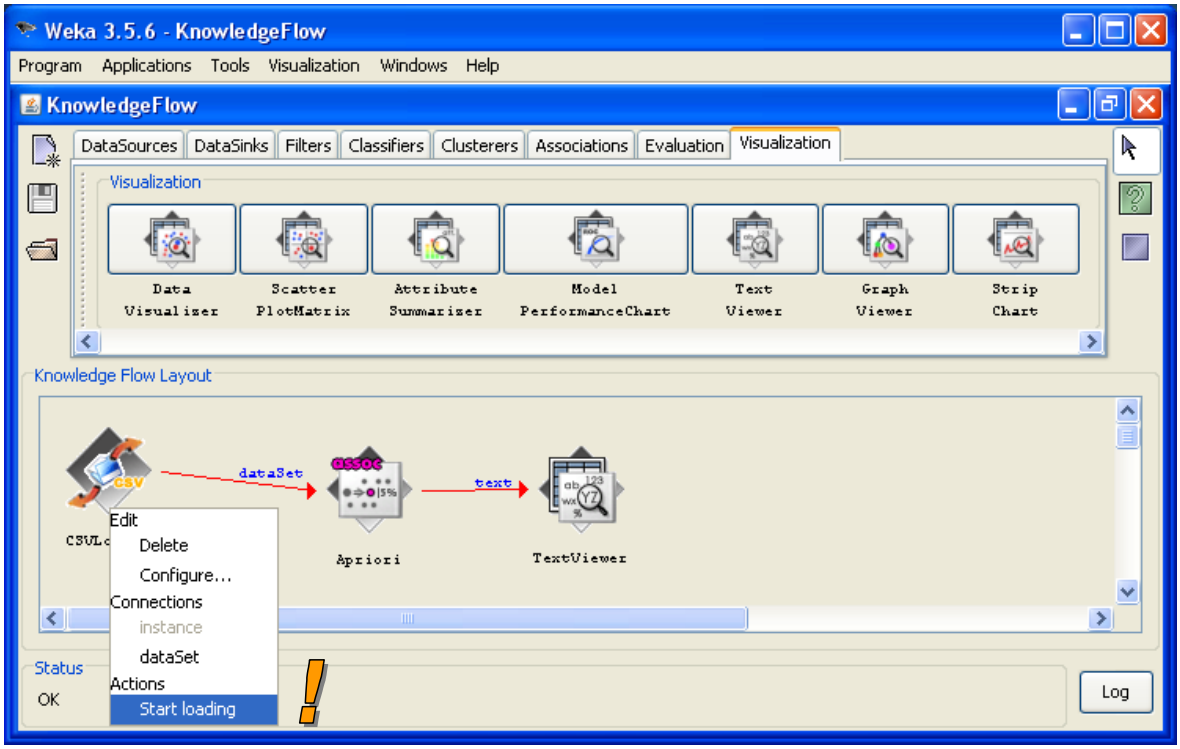


Nous insérons ensuite le composant APRIORI (onglet ASSOCIATIONS) auquel nous connectons le précédent en utilisant le flux DATASET. Le menu CONFIGURE permet de le paramétrer.

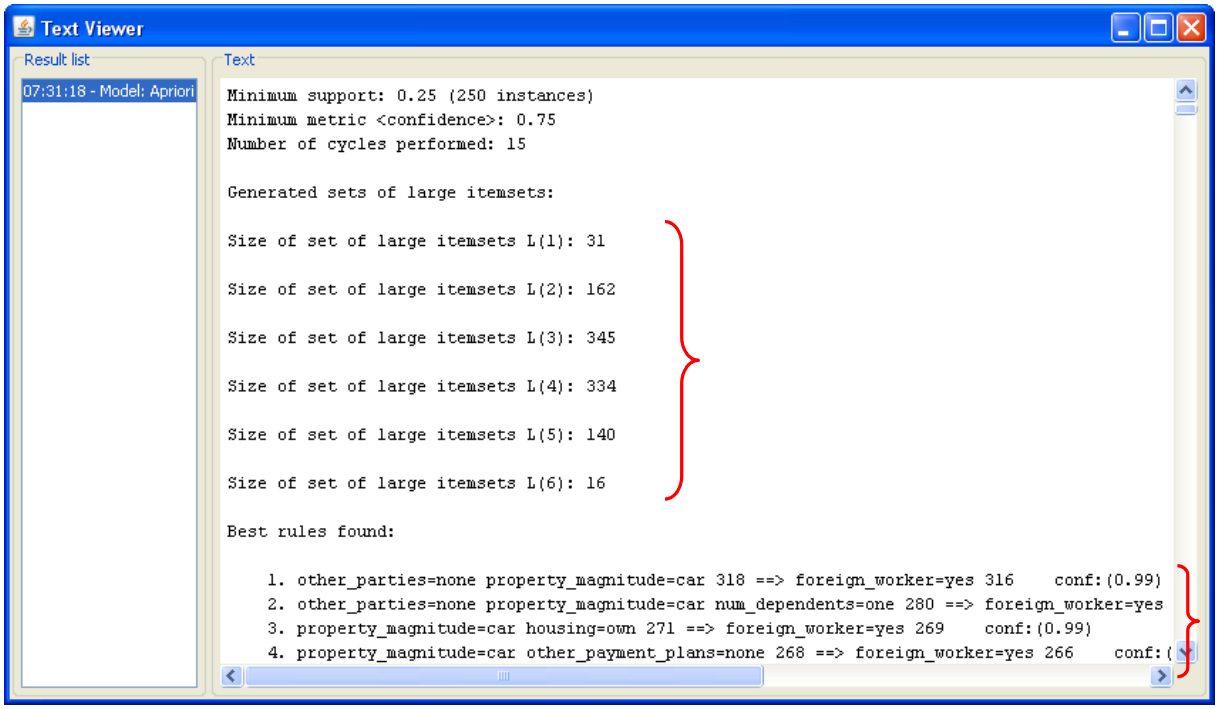


Le paramètre NUMRULES est fixé à 10000 uniquement pour que le nombre de règles générées ne soit pas indûment limité.

Il nous reste à insérer et à connecter l'outil de visualisation TEXT VIEWER. Nous lançons les calculs en actionnant le menu START LOADING... du premier composant CSVLOADER.



Pour accéder aux résultats, nous actionnons le menu SHOW RESULTS du composant TEXTVIEWER. Les résultats sont à rapprocher avec ceux du composant APRIORI de Tanagra. Dans la partie haute nous avons le nombre d'itemsets par cardinalité, dans la partie basse la liste des règles. Tout comme dans Tanagra, nous en obtenons 2986.



10 Conclusion

Tous les logiciels étudiés dans ce didacticiel savent extraire relativement simplement les règles d'association à partir d'un tableau « individus x variables ». Pour certains d'entre eux, un travail préparatoire préalable est nécessaire pour amener les données au format transactionnel. Il faut tout simplement comprendre de quoi il retourne… et introduire les bonnes séquences d'opérations. Ce n'est pas toujours évident, surtout en absence de documentation. J'avoue pour ma part avoir du tâtonné un peu. Mais finalement, une fois les données présentées correctement, les logiciels produisent des résultats similaires. C'est ce qui importe.

Enfin, comme nous avons pu le constater, la méthode produit un grand nombre de règles. La possibilité de les explorer interactivement est un atout certain. Certes, nous pouvons trier les règles selon les critères numériques. Orange permet même de combiner des critères. Mais il serait intéressant également de pouvoir les filtrer selon le contenu (ex. quelles sont les règles qui contiennent tel ou tel item dans le conséquent, dans l'antécédent, etc.). Seul RapidMiner propose ce type de fonctionnalités, que l'on retrouve pourtant quasiment systématiquement dans les bons logiciels commerciaux.