MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Data Management

Lab 02 - Analyse de données

Robin Chappatte
Frédéric Montet
Brian Nydegger

Rendu le 28 novembre 2016
à Lausanne

Professeurs :
Dr. Laura Elena Raileanu
Fabien Dutoit

Table des matières

Introduction

Pour le second laboratoire du cours de Data Management, nous allons mettre en pratique les concepts d'analyse de donnée vus en cours :

— Arbre de décisions
— Clustering
— Règles d'associations

Pour faire cela, nous utiliserons le logiciel Weka car il dispose d'une grande collection d'outils pour l'analyse de donnée et est également libre et multi-plateforme, ce qui en fait un outil idéal pour le cadre académique dans lequel ce laboratoire est réalisé.

1 | Installation et configuration

Dans notre situation nous avons décidé d'installer Weka sur une machine virtuelle Ubuntu 16.10.

1.1 Installation

Nous avons installé :

— Java JDK 1.8_111
— Weka 3.8.0
— Xampp x64 7.0.13
— MySQL connector for Java 5.1.39

Un élément à mentionner est que nous avons utilisé le connecteur dans sa version 5.1.39 alors que sur le site de MySQL [1] il est disponible en 5.1.40. Ceci est dû au fait que nous avons utilisé la version disponible dans un repository ppa. L'installation à pu être faite avec la commande `sudo apt-get install libmysql-java`.

Hormis le point cité au paragraphe ci dessus, cette étape n'a pas posé de problème particulier qu'il serait nécessaire de mentionner. Il a suffit de suivre les différents tutoriaux relatifs à chacun des logiciels à installer.

1.2 Configuration

Pour configurer la machine virtuelle, un seul point a posé problème : la connexion à la base de donnée MySQL depuis Weka en utilisant le connecteur MySQL Java.

Le problème rencontré a été la prise en compte de la variable d'environnement `CLASSPATH` lors des tentatives de connexions. Sans cette variable, Weka ne trouve pas le connecteur.

Après avoir effectué une recherche, nous nous sommes rendu compte qu'avec Ubuntu, il fallait préciser le `CLASSPATH` dans la commande pour démarrer Weka à l'aide du paramètre `-cp`. Le listing Figure 1.1 présente le `.sh` qui a été utilisé pour démarrer Weka et le connecter à MySQL avec succès.

1. https://dev.mysql.com/downloads/connector/j/

```
# Variables
java8=/home/fredmontet/Desktop/lab02-weka/asset/java/jdk1.8.0_111/jre/bin/
    ↪ java
weka=/home/fredmontet/Desktop/lab02-weka/asset/weka-3-8-0/weka.jar
connector=/usr/share/java/mysql-connector-java.jar

# Command
export CLASSPATH=$connector:$CLASSPATH
$java8 -Xmx300m -cp "$weka:$connector" weka.gui.GUIChooser
```

FIGURE 1.1 – Contenu du fichier bash à exécuter pour lancer Weka

2 | Analyse de données

Les données à dispositions dans la base de donnée qui a été importée à partir du fichier *aventure2014.sql.gz* n'ont pas été utilisées complétement pour ce laboratoire. Uniquement les attributs suivants ont été utilisés :

— MaritalStatus
— Gender
— YearlyIncome
— TotalChildren ;
— EnglishEducation
— EnglishOccupation
— HouseOwnerFlag
— NumberCarsOwned

2.1 Arbre de décisions

L'algorithme choisi pour cette méthode de classification a été REPTree. Après avoir effectué plusieurs runs aux paramètres différents, nous avons choisi un des meilleurs de nos tests.

Sur le listing Figure 2.1, on observe les informations du run choisis ainsi que les attributs sur la base desquels on veut trouver les classes de l'attributs `EnglishOccupation` qui sont :

— Professional
— Management
— Skilled Manual
— Clerical
— Manual

```
Scheme:       weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 10 -S 42 -L -1
    ↪  -P -I 0.0 -batch-size 200
Relation:     QueryResult-weka.filters.unsupervised.attribute.Remove-R1
    ↪ -7,9,11,14,16,18,21-25
Instances:    18484
Attributes:   8
              MaritalStatus
              Gender
              YearlyIncome
              TotalChildren
              EnglishEducation
              EnglishOccupation
              HouseOwnerFlag
              NumberCarsOwned
Test mode:    split 80.0% train, remainder test
```

FIGURE 2.1 – Informations du run REPTree

Le résultat de ce run à montré que la moyenne pondérée des instances correctement classifiées est 91.29%. La valeur de ce résultat est bonne et indique que notre modèle est bon. Cependant, ce seul résultat ne permet pas d'affirmer que le modèle est bon. Il est nécessaire d'observer les différentes métriques dans la partie `Detailed Accuracy By Class` du listing Figure 2.2 pour en savoir d'avantage. Sur ce dernier, on constate que `0.913` est la moyenne des métriques TP Rate, Precision, Recall et F-Measure pour les différentes classes. Notre analyse est donc confortée par la stabilité du résultat moyen de ces différentes métriques.

Maintenant, il est intéressant de savoir quelles sont les forces et faiblesses de notre modèle. Pour cela, nous baserons nos constats sur la F-Measure. Cette dernière étant robuste car elle prend en compte precision et recall. On y voit que la classe au meilleur score est `Manual` avec 0.970 et la moins bonne est `Skilled Manual` avec 0.882.

Afin d'améliorer notre modèle, on se concentrera donc sur la classe `Skilled Manual`. La matrice de confusion nous donne le détail de classification des instances de `Skilled Manual`. On y voit que `Skilled Manual` est souvent pris pour `Professional` et vice-versa. Ceci peut nous donner une piste pour de plus amples analyse ou on se questionnera sur les similarités et différences entre 2 instances de chacune des classes pour comprendre quel sont les facteurs qui rentrent en jeux.

```
=== Summary ===

Correctly Classified Instances        3375               91.2902 %
Incorrectly Classified Instances       322                8.7098 %
Kappa statistic                          0.8883
Mean absolute error                      0.0434
Root mean squared error                  0.1549
Relative absolute error                 13.8991 %
Root relative squared error             39.2271 %
Total Number of Instances             3697

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area
    ↪   Class
0.904     0.043     0.899       0.904     0.902      0.860     0.982     0.963
    ↪     Professional
0.912     0.019     0.909       0.912     0.910      0.892     0.992     0.970
    ↪     Management
0.873     0.035     0.892       0.873     0.882      0.844     0.981     0.945
    ↪     Skilled Manual
0.946     0.013     0.930       0.946     0.938      0.927     0.991     0.971
    ↪     Clerical
0.972     0.005     0.968       0.972     0.970      0.966     0.997     0.981
    ↪     Manual

Weighted Avg.

0.913     0.027     0.913       0.913     0.913      0.885     0.987     0.963

=== Confusion Matrix ===

    a    b    c    d    e   <-- classified as
 1002   25   69   12    0 |    a = Professional
   46  568    9    0    0 |    b = Management
   62   32  802   20    3 |    c = Skilled Manual
    4    0   15  545   12 |    d = Clerical
    0    0    4    9  458 |    e = Manual
```

FIGURE 2.2 – Résultats du run REPTree

Concernant l'arbre de décision, son début est visible dans le listing Figure 2.3. Sur cet exemple, on voit que la division commence par l'apport financier, puis l'éducation, les enfants, l'état civil et la possession d'une maison.

Il est intéressant de noter que juste après la division homme-femme, l'état civil et la possession d'une maison ont été interverti dans la branche M et F. Ceci met en évidence que la meilleure manière de diviser les classes est prise en compte en fonction de chacune des instances.

```
YearlyIncome < 45000
|   YearlyIncome < 25000
|   |   EnglishEducation = Bachelors
|   |   |   YearlyIncome < 15000
|   |   |   |   TotalChildren < 1.5
|   |   |   |   |   Gender = M
|   |   |   |   |   |   MaritalStatus = M
|   |   |   |   |   |   |   HouseOwnerFlag = 1 : Manual (27/1) [0/0]
|   |   |   |   |   |   |   HouseOwnerFlag = 0 : Manual (3/0) [0/0]
|   |   |   |   |   |   MaritalStatus = S : Manual (11/0) [0/0]
|   |   |   |   |   Gender = F
|   |   |   |   |   |   HouseOwnerFlag = 1
|   |   |   |   |   |   |   MaritalStatus = M : Manual (21/1) [0/0]
|   |   |   |   |   |   |   MaritalStatus = S : Manual (7/0) [0/0]
|   |   |   |   |   |   HouseOwnerFlag = 0 : Manual (8/1) [0/0]
```

FIGURE 2.3 – Début du REPTree

2.2 Clustering

Première observation pour le clustering : Plusieurs des méthodes proposées par Weka reposent sur d'autres métodes.

Ainsi, FilteredClusterer et MakeDensityBasedClusterer demandent de choisir une méthode de clustering parmis celles mises à disposition par Weka. Petite particularité de Weka : Celui-ci n'empêche pas d'indiquer à (par exemple) FilteredClusterer d'utiliser FilteredClusterer pour fonctionner, permettant ainsi de "chaîner" indéfiniement ceux deux méthodes de clustering. Après plusieurs tests, ceci ne change rien aux valeurs de résultats (mais augmente le temps de génération de ceux-ci).

Deuxième observation : D'autres méthodes permettent de définir un algorithme de calcul de distance parmi ceux proposés par Weka.

Ainsi, HierarchicalClusterer et SimpleKMeans sont tous les deux configurable à ce niveau.

Dans notre cas nous avons choisi de détailler les résultats de la méthode FarthestFirst car sa simplicité permet d'en combiner l'usage avec les cluster FilteredClusterer et MakeDensityBasedClusterer et bien comprendre son fonctionnement facilite ainsi la compréhension d'autres méthodes que nous pourrions combiner avec lors de cas extra-académiques.

La méthode FarthestFirst est dite de clustering non-hiérarchique (ou Flat Clustering) car elle ne crée par d'arbre, et donc pas de descendance / hiérarchie entre les cluster qu'elle génère. Deux paramètre numériques sont définissables :

— Le nombre de cluster à générer
— La graine à utiliser pour tirer aléatoirement des exemples

Contrairement à d'autres méthodes essayées et évaluées, FarthestFirst ne tire par une graine aléatoirement quand elle est définie à zéro. Il est donc du devoir de l'utilisateur de Weka d'être attentif à cette particularité s'il souhaite lancer plusieurs générations affin de déterminer les meilleurs clusters possible.

Les options ne permettent pas de définir un nombre maximum d'itérations, on peut donc en déduire que la méthode évaluée repose sur la convergeance des données dans les clusters pour déterminer quand s'arrêter.

Pour nos tests nous avons essayez avec les nombres de clusters suivants :

— 2 : C'est le nombre minimum de clusters assignable et celà permet de voir quels données sont dissociées
— 8 : C'est le nombre de type de données différentes que nous traitons dans cet exercice

2.2.1 2 clusters

Les listes déroulantes du haut de l'interface permettent de définir les éléments à afficher sur l'axe des X et ceux à afficher sur l'axe des Y pour la visualisation. Si les types de données sont discrêtes, un paramêtre (jitter) permet de visualiser plus facilement l'étendue des répartitions en ajoutant une forme de bruit en X et en Y aux graphes. Dans les exemples si dessous, les graphes sont fait avec le revenu annuel en abssisse et la possession ou non de maison en ordonnée. En rouge et en bleu on voit les deux clusters déterminés par le run, dont le résultat est également ci-joint.

```
=== Run information ===

Scheme:       weka.clusterers.FarthestFirst -N 2 -S 0
Relation:     QueryResult-weka.filters.unsupervised.attribute.Remove-R2
    ↪ -7,9,11,14,16,18,21-25-weka.filters.unsupervised.attribute.Remove-R1
Instances:    18484
Attributes:   8
              MaritalStatus
              Gender
              YearlyIncome
              TotalChildren
              EnglishEducation
              EnglishOccupation
              HouseOwnerFlag
              NumberCarsOwned
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


FarthestFirst
==============

Cluster centroids:

Cluster 0
     M M 60000.0 2.0 Partial College Professional 1 1.0
Cluster 1
     S F 170000.0 4.0 Bachelors Management 0 4.0



Time taken to build model (full training data) : 0.07 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      14166 ( 77%)
1       4318 ( 23%)
```

FIGURE 2.4 – Output de l'éxécution avec 2 clusters

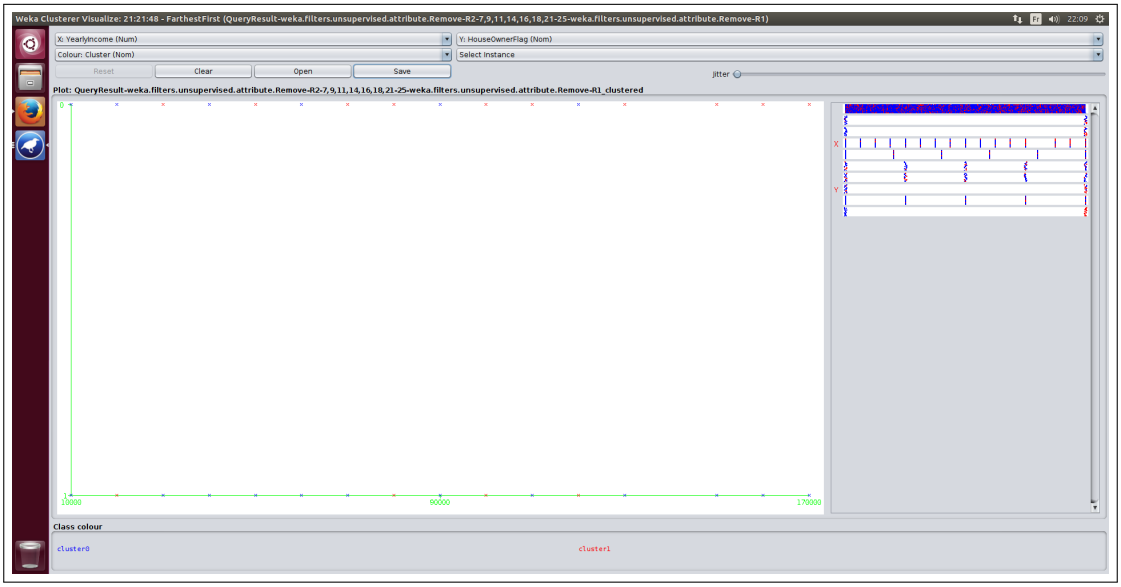

FIGURE 2.5 – Clustering sans jitter et avec deux clusters

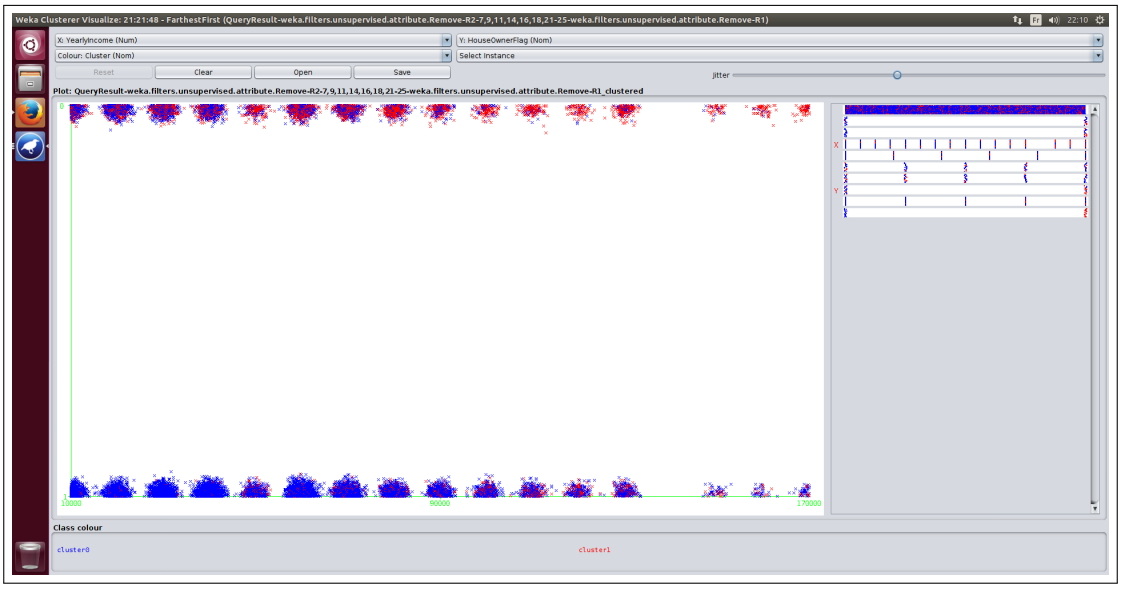

FIGURE 2.6 – Clustering avec jitter et avec deux clusters

2.2.2 8 clusters

Pour cette génération (le jitter a été volontairement poussé pour permettre une meilleure visualisation des résultats), on constates que les revenus influencent certains clusters quand à l'état civil (marié ou non). Il ne nous est pas en l'état possible de savoir pourquoi et de plus amples recherches seraient nécéssaires pour déterminer les facteurs

communs de ces résultats.

```
=== Run information ===

Scheme:       weka.clusterers.FarthestFirst -N 8 -S 1
Relation:     QueryResult-weka.filters.unsupervised.attribute.Remove-R2
    ↪ -7,9,11,14,16,18,21-25-weka.filters.unsupervised.attribute.Remove-R1
Instances:    18484
Attributes:   8
              MaritalStatus
              Gender
              YearlyIncome
              TotalChildren
              EnglishEducation
              EnglishOccupation
              HouseOwnerFlag
              NumberCarsOwned
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


FarthestFirst
==============

Cluster centroids:

Cluster 0
     M M 30000.0 0.0 Partial College Skilled Manual 0 1.0
Cluster 1
     S F 120000.0 5.0 Partial High School Professional 1 4.0
Cluster 2
     S F 160000.0 0.0 Graduate Degree Management 0 1.0
Cluster 3
     S M 10000.0 3.0 High School Manual 1 0.0
Cluster 4
     M M 170000.0 1.0 Bachelors Management 1 4.0
Cluster 5
     M F 30000.0 5.0 Graduate Degree Clerical 1 0.0
Cluster 6
     M F 120000.0 1.0 High School Professional 0 4.0
Cluster 7
     S M 130000.0 5.0 High School Management 0 4.0



Time taken to build model (full training data) : 0.07 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       3967 ( 21%)
1       1534 (  8%)
2       1825 ( 10%)
3       3031 ( 16%)
4       2374 ( 13%)
5       3405 ( 18%)
6       1736 (  9%)
7        612 (  3%)
```


FIGURE 2.7 – Output de l'éxécution avec 8 clusters

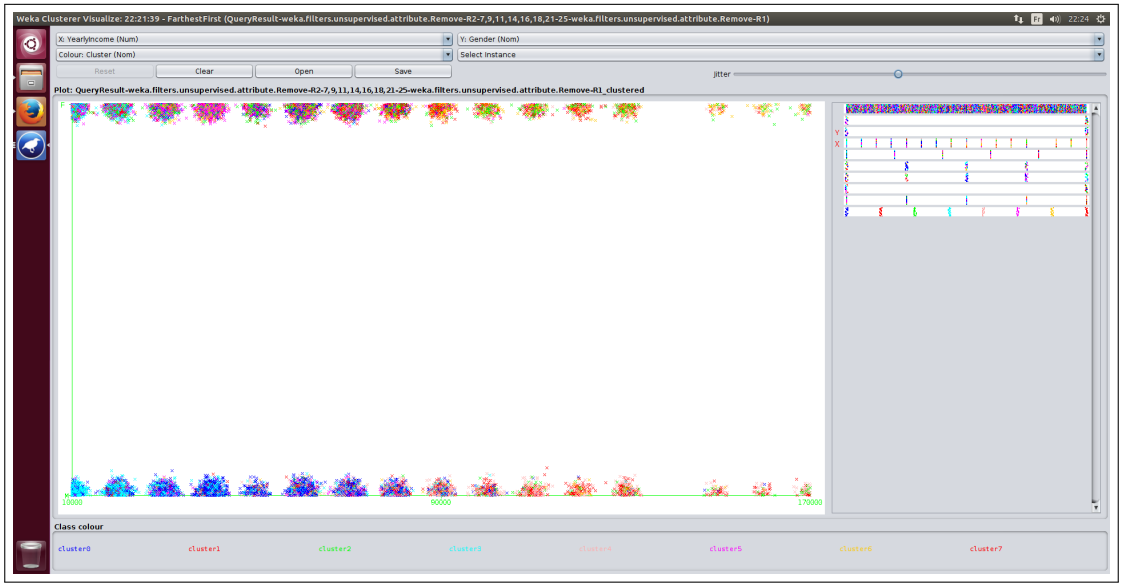

FIGURE 2.8 – Clustering avec jitter et avec huit clusters

2.3 Règles d'associations

Pour cette section, le filtre *NumericToNominal* a été appliqué aux attributs *NumberCarsOwned*, *YearlyIncome* et *TotalChildren*.

Seul les algorithmes *Apriori* et *FilteredAssociator* ont été testé. Seul ceux-ci étaient disponible (voir 2.9).

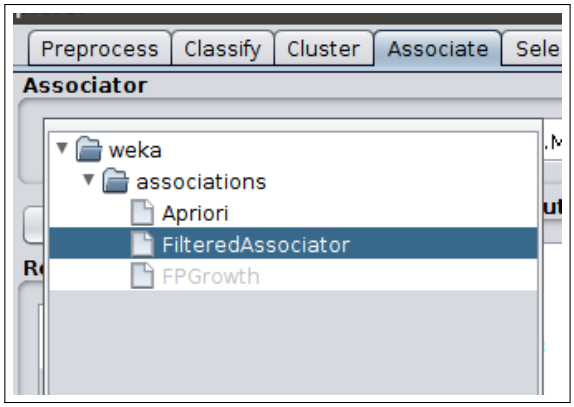

FIGURE 2.9 – Règles d'associations : Algos disponible

2.3.1 Apriori

Le seuil minimum de confiance a été modifié pour obtenir plus de règles.

Voici les paramètres :

```
1 Apriori -N 10 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c 1
```

Et le résultat (figure 2.10).

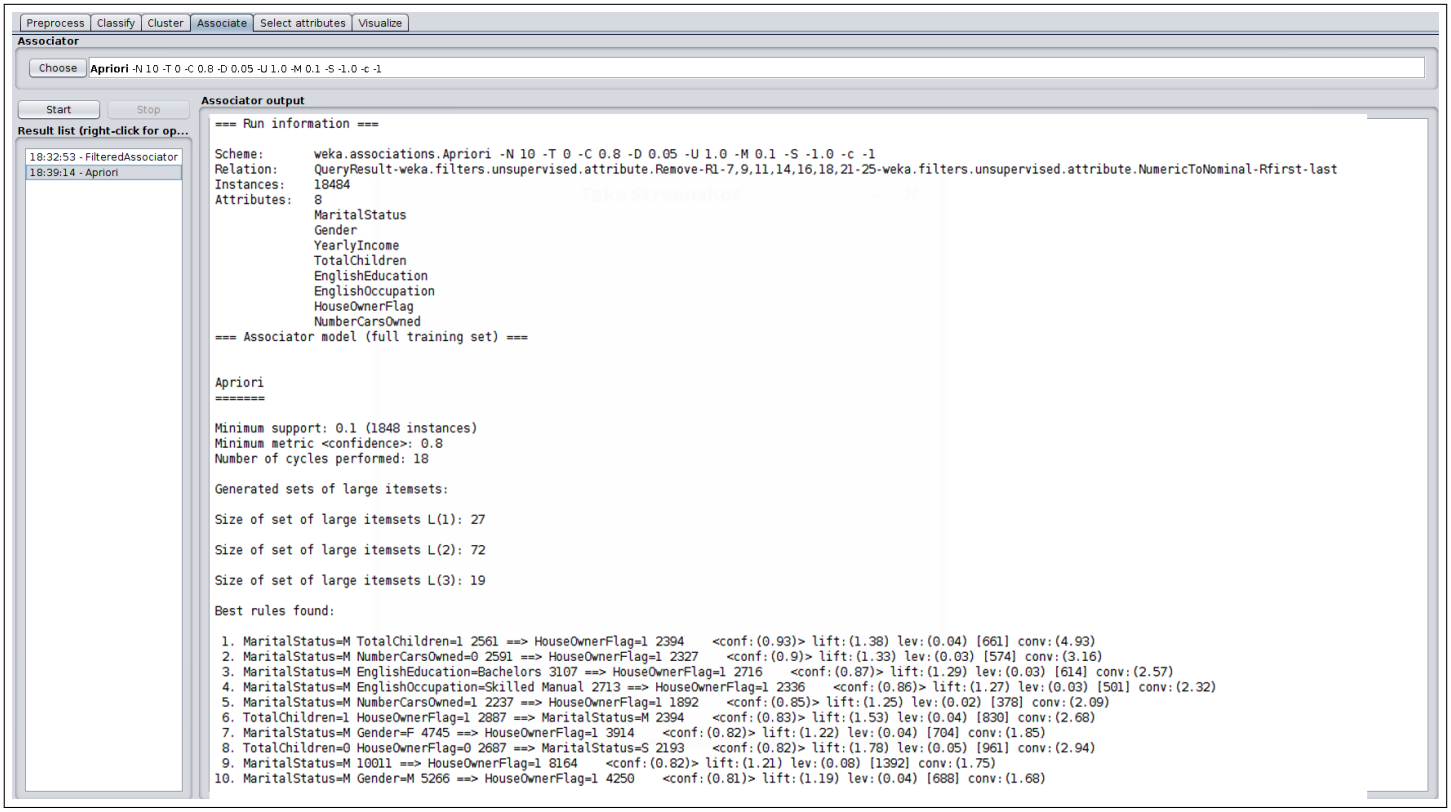

FIGURE 2.10 – Règles d'associations : Apriori résultat

2.3.2 FilteredAssociator

Cet algorithme permet de créer des règles d'association filtrées.

Les 10 règles obtenues (figure 2.11) sont identiques au résultat de l'*Apriori*.

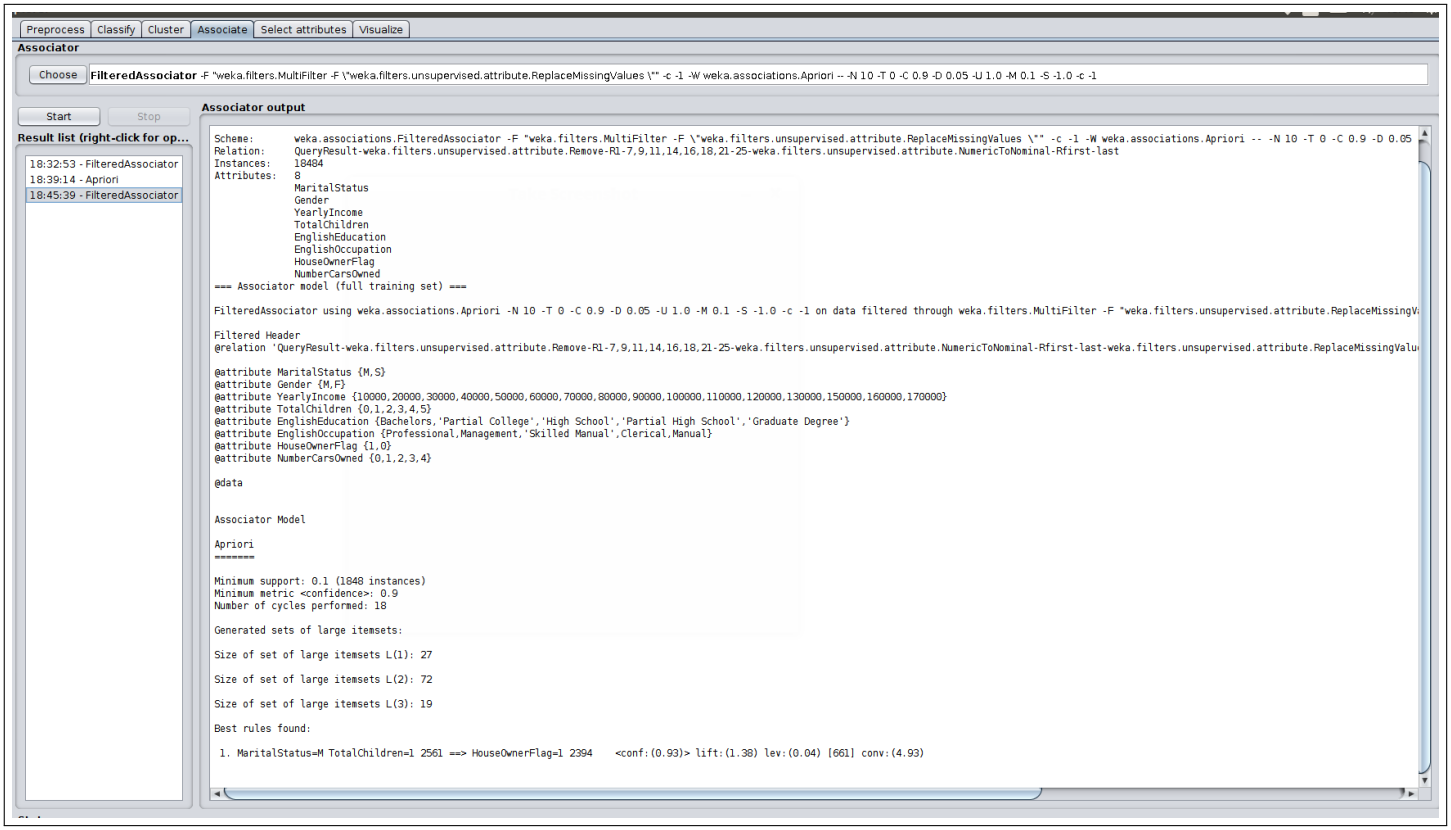

FIGURE 2.11 – Règles d'associations : FilteredAssociator résultat

3 | Conclusion

En faisant ce laboratoire, nous avons découvert la puissance du logiciel Weka. Malgré une interface qui pourrait être plus intuitive, la possibilité de se connecter directement à une base de donnée MySQL permet de faire des analyses très rapidement. Ainsi, Weka semble être un logiciel très utile lorsqu'il s'agit de faire une analyse de donnée préliminaire ou, lorsqu'une contrainte de coût et/ou délai est présente. Un exemple d'utilisation relativement réaliste serait celui ou Weka est connecté à une base de donnée en production pour ajuster les différents public cible d'une gamme de produit.

En testant les différents types d'algorithmes que nous avons vu en cours, notre compréhension de ces derniers s'est améliorée par le biais d'un exemple pratique. Nous avons compris que certains algorithmes nécessites un paramétrage particulier pour donner de bon résultats de classification. Désormais, notre vision critique vis-à-vis des algorithmes à dispotion est plus claire.