REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

Mémoire de fin d'étude

En vue de l'obtention du diplôme de Master en informatique

Option : *Conduite de projet informatique.*

Thème

Extraction de connaissances à partir de résultats d'évaluation d'apprenants en utilisant le data mining

Dirigé par:

M^me^ BOUARAB FARIDA

Réalisé par :

M^elle^ ATMANI SALIHA

&

M^elle^ BELAIDI LYNDA

Promotion: 2012/2013

REMERCIEMENTS

D'abord nous remercions le bon dieu de nous avoir donné santé, courage, volonté et foi pour réaliser ce travail.

Nous tenons à exprimer notre profonde gratitude à notre promotrice Mme BOUARAB Farida pour tout ce qu'elle nous a apportée comme aide, connaissances et conseils.

Nous remercions vivement les membres du jury pour avoir accepté d'évaluer notre travail

Nos remerciements vont aussi à toutes les personnes qui nous ont aidés de près ou de loin à la réalisation de ce travail.

DEDICACES

Je dédie ce modeste travail à mes chers parents, ma sœur Selia, mon frère Arezki, mes proches, à mon binôme Saliha et à sa famille, à tous mes amis (es) et enfin à toutes les personnes ayant contribué chacune à sa manière au bon accomplissement de notre projet.

Lynda

DEDICACES

Je dédie ce modeste travail à mes chers parents, ma sœur, mon frère, ma grand-mère, qui m'ont soutenu tout au long de mes études, toutes la famille et mes amis, et enfin à toutes les personnes ayant contribué chacune à sa manière au bon accomplissement de notre projet.

Lynda

DEDICACES

Je dédie ce modeste travail à la mémoire de ma mère, à mon père, à mes frères Belaid, Mohamed, Mouloud et sa famille, mes sœurs Ouardia et sa famille, Louiza et son mari, mes proches, à tous mes amis (es) et enfin à toutes les personnes ayant contribué chacune à sa manière au bon accomplissement de notre projet.

Saliha

Table des matières

Introduction générale..... 01

Chapitre 1 : Data mining pour l'éducation

Introduction..... 03

1. Extraction des Connaissances à partir de Données (ECD)..... 03

1.1 Définition..... 03

1.2 Processus d'ECD..... 03

2. Datamining..... 07

2.1 Définition..... 07

2.2 Taches du datamining..... 08

2.2.1 Classification automatique supervisée..... 08

2.2.2 Classification automatique non supervisée..... 09

2.2.3 Les règles d'association..... 09

2.3 Domaines d'application..... 09

3. Datamining pour l'éducation (EDM)..... 10

3.1 Définitions..... 10

3.2 Méthodes du data mining pour l'éducation..... 11

3.2.1 Prédiction..... 11

3.2.2 Clustering..... 12

3.2.3 L'exploration minière de la relation..... 12

3.2.4 Distillation de données pour le jugement humain..... 13

3.2.5 Découverte des modèles..... 13

3.3 Principales applications des méthodes EDM..... 14

3.4 Exemples du datamining pour l'éducation..... 15

Conclusion..... 19

Chapitre 2 : Conception

Introduction..... 20

1. Le preprocessing..... 20

1.1 La sélection des données..... 25
1.2 Le calcul algébrique..... 25
1.3 La transformation des données d'apprentissage..... 26
1.3.1 Le format SQL..... 26
1.3.2 Le format ARFF..... 26
1.3.3 Le processus de transformation..... 27
2. Application des algorithmes de data mining..... 28
2.1 Présentation des algorithmes du data mining..... 28
2.1.1 La classification..... 28
2.1.1.1 Les arbres de décision..... 28
2.1.2 Le clustering..... 30
2.1.2.1 L'algorithme kMeans (Les centres mobiles)..... 30
2.1.3 Les règles d'associations..... 32
2.1.3.1 L'algorithme Apriori..... 32
2.2 Etapes d'application des algorithmes de data mining..... 35
3. Architecture globale de l'application..... 38
Conclusion..... 38

Chapitre 3 : Implémentation et test

Introduction..... 39
Partie 1 : Implémentation..... 39
1. L'environnement de développement..... 39
1.1 Les langages de programmation..... 39
1.1.1 Le langage Java..... 39
1.1.2 Le langage HTML (Hyper Text Markup Language)..... 40
1.1.3 Javascript..... 40
1.2 Les outils..... 40
1.2.1 NetBeans..... 40
1.2.2 EasyPHP..... 41
1.2.3 GlassFish..... 41

2. L'utilisation de l'API weka sous NetBeans..... 41

2.1 Chargement de l'API..... 41

2.2 Utilisation de l'API..... 42

3. Exécution de l'application..... 44

3.1 L'environnement d'exécution..... 44

3.2 Présentation des interfaces de l'application..... 46

Partie 2 : tests..... 51

Conclusion..... 71

Conclusion générale..... 72

Bibliographie..... 73

Liste des figures

Figure1 : Processus d'extraction de connaissances..... 4

Figure2 : Datamining vers la connaissance..... 8

Figure 3 : Avantages que le datamining peut apporter aux campagnes de promesses de dons adressées aux anciens élèves..... 19

Figure4 : Schéma global du preprocessing..... 20

Figure5 : Diagramme entités/associations..... 22

Figure 6 : Table relationnelle pour la session d'apprentissage..... 24

Figure 7 : Exemple de sélection de données à partir de la base de données du domaine du système web siela..... 25

Figure 8 : Schéma de sauvegarde de données..... 27

Figure 9 : Processus de transformation de données..... 28

Figure10: L'algorithme kMeans..... 31

Figure 11 : Partitionnement basé sur k-Means..... 32

Figure 12 : L'algorithme Apriori..... 33

Figure 13 : L'algorithme de génération des itemsets candidats..... 34

Figure 14 : Etapes d'application des algorithmes de data mining..... 36

Figure15: Architecture globale de l'application..... 38

Figure 16 : L'ajout du fichier weka.jar..... 42

Figure 17 : Les packages concernant l'algorithme J48..... 42

Figure 18: Le code java (méthode) permettant l'affichage de l'arbre de décision..... 43

Figure 19 : Les packages concernant l'algorithme KMeans..... 43

Figure 20 : Quelques instructions essentielles de l'algorithme KMeans..... 43

Figure 21 : Les packages concernant l'algorithme Apriori..... 43

Figure 22: Quelques instructions essentielles de l'algorithme Apriori..... 44

Figure 23: Génération du fichier WAR..... 44

Figure 24 : Déploiement du fichier WAR..... 45

Figure 25 : Espace Data mining..... 46

Figure 26: Calcul algébrique..... 47

Figure 27: Conversion du fichier SQL..... 48

Figure 28 : Les trois catégories d'algorithmes de data mining..... 48

Figure 29 : Interface de l'algorithme J48..... 49

Figure 30: Algorithme kmeans..... 50

Figure 31 : Interface de l'algorithme Apriori..... 50

Figure 32 : Exemple d'exécution de l'algorithme J48..... 51

Figure 33 : Exemple d'exécution de KMeans..... 56

Figure 34 : Résultat de l'algorithme KMeans (test 2)..... 60

Figure 35: Exécution de l'algorithme Apriori..... 63

Introduction générale

L'objet des recherches sur les EIAH est la recherche des principes de conception, de développement et d'évaluation d'environnements informatiques qui ont pour objectifs de favoriser ou susciter des apprentissages, de les accompagner et de les valider. Ainsi, l'apprenant, peut apprendre à sa propre allure, en suivant un parcours personnalisé et individualisé, et l'enseignant peut l'aider et le suivre durant son apprentissage, en évaluant ses connaissances, ses aptitudes et en détectant et analysant ses erreurs afin de suivre la progression de son niveau d'apprentissage et ses capacités d'acquisition.

Cependant, on se basant sur le modèle de l'apprenant, le data mining (particulièrement le data mining pour l'éducation) offre un ensemble d'algorithmes et de techniques qui permettent d'extraire des connaissances à partir de données, qui sont utiles pour l'enseignant afin de prendre les bonnes décisions concernant les apprenants.

Plus large que la notion de data mining, nous avons le domaine d'Extraction de Connaissances à partir de Données (ECD) (en anglais Knowledge Discovery in Database (KDD)). Le processus d'ECD englobe toutes les étapes de stockage et de préparation de données, l'analyse de celles-ci par différentes techniques, et enfin, l'interprétation et l'évaluation des connaissances acquises. Certes le Data Mining n'est qu'un rouage dans la formidable machine qu'est le KDD, mais il est sans conteste le cœur et le moteur de tous ce processus.

Notre travail dans ce domaine, consiste en l'implémentation de trois algorithmes de data mining parmi les catégories suivantes : classification, clustering, et règles d'association. Pour chacun de ces algorithmes, il ya une phase de préparation des données en entrées, c'est la partie qui se nomme preprocessing, et qui permet à partir d'une base de données concernant l'apprenant et son évaluation de faire des calculs algébriques, suivi d'une transformation de format pour avoir le format appropriée à l'algorithme de data mining.

Nous avons organisé notre mémoire de la manière suivante :

- ✓ Le chapitre 1, présente le processus d'extraction de connaissances (ECD), le data mining, ainsi que ses tâches et ses domaines d'application, et le data mining pour l'éducation.

- ✓ Le chapitre 2, décrit les différentes étapes de la conception de notre application englobant le module de preprocessing et les différents modules d'utilisation des algorithmes de data mining choisis.
- ✓ Le chapitre 3, est consacré à l'implémentation de notre application en utilisant l'API Weka, suivi de tests et d'interprétation des résultats relativement au contexte d'utilisation.
- ✓ Enfin, nous présentons une conclusion générale et quelques perspectives pour notre travail.

Chapitre 1

Data mining pour l'éducation

Introduction:

Ces dernières années, les techniques de datamining ont été utilisées dans plusieurs domaines comme le e-commerce, la bioinformatique, la médecine… etc. L'utilisation des techniques de datamining dans l'exploration de données éducatives est également devenue populaire, en témoigne la conférence EDM (Educational Data Mining) (Juin 20-21, 2008, Canada) qui est organisée pour discuter des travaux dans cette discipline. Plusieurs travaux sont publiés autour de l'utilisation du datamining pour découvrir des informations utiles assistant les chercheurs, les apprenants, les autres acteurs concernés dans les EIAH et notamment les enseignants/tuteurs dans le processus d'enseignement.

Dans ce chapitre nous allons traiter ce domaine émergent (Datamining pour l'éducation) en présentant en premier lieu le processus d'ECD, puis le datamining, ensuite nous décrivant le datamining pour l'éducation à travers une définition, ses méthodes, ses principales applications, et on termine par des exemples d'EDM.

1. Extraction des Connaissances à partir de Données (ECD):

Le datamining est une étape d'une chaine de traitement plus complète, le processus d'extraction de connaissances à partir de données (ECD). Qu'est ce que l'ECD ? Quels sont ses étapes? [2]

1.1 Définition:

Au vue de l'émergence de nouveaux champs d'application grâce au Datamining, les experts dans le domaine travaillèrent à réunir plusieurs techniques afin de créer une méthode de recherche de connaissance intégrant toutes les étapes, du recueil des données à l'évaluation de la connaissance acquise. C'est ainsi qu'est né le terme d'Extraction des Connaissances à partir des Données (ECD), ou en anglais Knowledge Discovery in Database (KDD).

*L'ECD dé*signe tout le cycle de découverte d'informations ou de connaissances dans les bases de données. Il regroupe donc toutes les opérations à effectuer pour extraire de l'information de ces données [10].

1.2 Processus d'ECD :

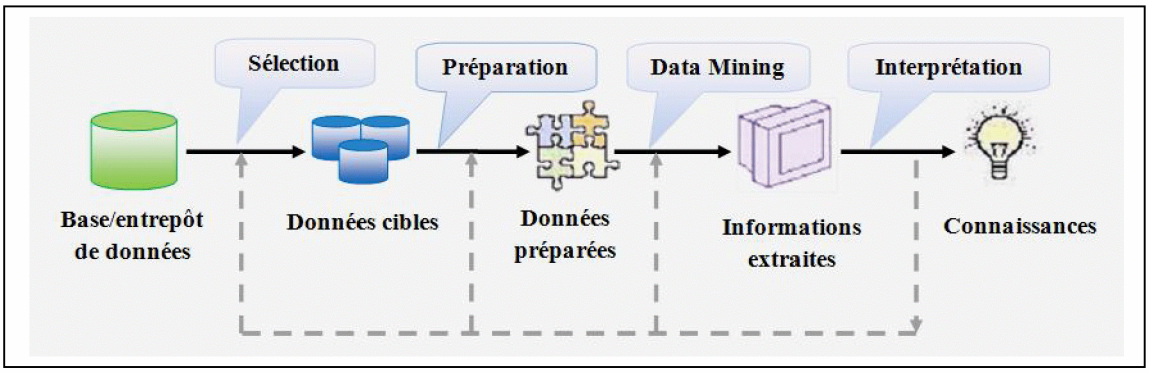


Figure 1 : *Processus d'extraction de connaissances [21].*

L'ECD est un processus complexe qui se déroule suivant une suite d'opérations. Les phases de l'ECD sont les suivantes:

- **Phase 1 : Positionnement du problème (fixation des objectifs) :**

Les connaissances à découvrir sont plus ou moins indiquées par l'utilisateur. Avant la collecte des données, on pose le problème et ou l'on définit les objectifs. Il s'agit de comprendre le contexte de la recherche en vue de donner une signification logique aux variables.

La pose du problème se décompose en deux étapes: *la formulation du problème* et *la typologie du problème* [4].

- **La formulation du problème :**

La première étape consiste à formuler le problème réel sous une forme qui peut être traitée par les techniques et les outils de modélisation.

Une des approches les plus communes consiste à découper le problème complexe en sous - problèmes de complexité moindre et à collecter les données nécessaires au traitement de chacun des sous – problèmes [12].

- **La typologie du problème :**

- **Affectation:** Lorsque l'on connaît l'appartenance des éléments à une ou plusieurs classes, il s'agit d'identifier des facteurs d'affectation.
- **Structuration:** Si l'objectif est de mettre en évidence des classes ou des facteurs de différenciation, la démarche relève alors d'une action d'identification des facteurs de structuration.

- **Les résultats attendus :** Avant de se lancer dans la collecte de données, il faut savoir ce que l'on attend et ce que l'on compte faire de la connaissance [12].

➢ **Phase 2 : La sélection (collecte de données) :**

On procède d'abord par une collecte de données. Ces dernières se trouvant généralement dans des bases de données ou dans des data warehouses. Les données collectées, ne faisant état d'aucun critère de sélection préalable, nécessitent habituellement une phase de sélection pour n'en garder que les données dont on a besoin.

Une sélection optimale des données nécessite souvent l'aide d'experts du domaine pour déterminer les facteurs, ou variables, les plus aptes à décrire la problématique.

Dans le cas où les experts ne sont pas disponibles, une recherche des variables les plus déterminantes est effectuée par des techniques d'analyse (comme la régression, les réseaux de neurones…) [2].

➢ **Phase 3 : Le nettoyage des données :**

Une faible qualité des données (erreurs de saisie, champs nuls, valeurs aberrantes) impose, généralement, une phase de nettoyage des données. Celle-ci a pour objectif de corriger ou de contourner les inexactitudes ou les erreurs de données [3].

Parmi les types de valeurs pouvant créer des ambiguïtés:

- ***Valeurs aberrantes*:** ce sont les données qui ont des valeurs anormales par rapport à leurs natures (leurs types). Par exemple, un client qui a une date de naissance égale à l'année en cours. Un contrôle sur les domaines des valeurs permet de retrouver ces valeurs.
- ***Les valeurs manquantes :***

Certaines données peuvent être absentes et gêner ainsi l'analyse. Il faut donc définir des règles pour gérer ou pour remplacer ces données manquantes.

Il faut gérer ces valeurs manquantes selon l'une des méthodes suivantes:

✓ **Remplacer les données manquantes:** La première méthode, supportée par certains logiciels, remplace la donnée absente par une valeur qui est soit choisie par l'utilisateur (remplacée par la moyenne ou la médiane, par exemple), soit calculée (remplacée par le résultat d'une formule de score), soit héritée.

- ✓ **Exclure les enregistrements incomplets:** Cette deuxième méthode, très restrictive, consiste à exclure tous les enregistrements dont une valeur manque. ce choix est pénalisant car il réduit la base d'apprentissage.
- ✓ **Gérer les valeurs manquantes:** Lorsque l'absence de données est acceptable du point de vue de la performance du modèle, les algorithmes offrent généralement la possibilité de gérer à part la valeur manquante en la distinguant des valeurs renseignées, ou celle considérer la valeur manquante comme un facteur d'indécision.

- **Les valeurs nulles :** le nettoyage des données doit intégrer une analyse spécifique des exemples à zéro. L'analyse de l'existence de ces enregistrements totalement nuls doit être menée afin d'en identifier les causes externes, avant de lancer les algorithmes d'apprentissage [12].

➢ **Phase 4 : Action sur les attributs :**

Il s'agit de transformer un attribut A en une autre variable A' qui serait, selon les objectifs de l'étude, plus appropriée. Différentes méthodes sont pratiquées comme la discrétisation qui consiste à transformer des attributs continus en découpant le domaine de valeurs de ces attributs en intervalles afin d'obtenir des attributs qualitatifs. Il existe à cet effet pléthore de méthodes de discrétisation : supervisées ou non, à intervalles de tailles identiques, ou à intervalles à effectifs constants. On peut également centrer par rapport à la moyenne et réduire par l'écart type les valeurs des variables continues. Ce traitement leur confère certaines propriétés mathématiques intéressantes lors de la mise en œuvre de méthodes d'analyse des données multidimensionnelles [11].

➢ **Phase 5 : L'exploitation (modélisation):**

Cette phase consiste à extraire la connaissance utile à partir d'un large volume de données et à la présenter sous une forme synthétique. Elle repose sur une recherche exploratoire, c'est à dire dépourvue de préjugés concernant les relations entre les données [3].

Les techniques de fouille de données cherchent à construire un « modèle ». Ce modèle doit rendre compte des relations liant la description d'une situation à un résultat. La formulation d'un modèle peut prendre plusieurs formes :

- Une formulation à base d'équations (réseaux de neurones, techniques de régression).

- Une formulation à base d'analyse logique de la forme ''Si A et B alors C'' (arbres de décision, règles d'associations).
- Une formulation par les techniques de projection des données dans un espace plus ou moins structuré (analyse factorielle).

La construction du modèle se fait sur les données d'apprentissage qui doivent être distinctes des données de test. Les données d'apprentissage et de test sont généralement issues à partir de la même base de données mais elles ne comprennent pas les mêmes données.

L'apprentissage prend généralement 70% à 80% des enregistrements, la base de test étant constituée du reste.

Les modèles construits de manière totalement automatique, sont particulièrement sensibles à la qualité des données. Aussi, une interaction avec l'utilisateur est parfois requise pour guider et améliorer le raisonnement au fur et à mesure de la construction du modèle.

Cette interactivité contribue, également, à bâtir des modèles parfois moins performants mais souvent plus réalistes [2].

- **Phase 6 : Evaluation des résultats :**

L'évaluation permet d'estimer la qualité du modèle qui a été construit, ainsi qu'à déterminer sa capacité à prévoir correctement les résultats de nouvelles situations.

A l'issue de la construction du modèle, il est possible d'en tester la pertinence sur l'ensemble des tests. En général, la performance d'un modèle s'apprécie au travers une matrice qui compare la situation réelle et la situation prévue par le modèle. La qualité globale du modèle est la portion de prédictions exactes par rapport aux nombre total de prédictions. Un niveau de prédiction de 80% est jugé moyen [1].

- **Phase 7 : utilisation des connaissances découvertes :**

On intègre le modèle dans le processus de l'entreprise, c'est l'étape de la transition du domaine des études au domaine opérationnel [4].

2. Datamining:

Après avoir présenté les différentes étapes du processus d'ECD, nous nous intéressons dans ce qui suit particulièrement à l'étape de datamining qui constitue le cœur du processus ECD. Qu'est ce que le datamining ? A quoi sert-il ?

2.1 Définition:

Le datamining ou la fouille de données, est un processus d'exploration et d'analyse de (souvent) grands volumes de données (stockées dans des entrepôts de données), en vue d'une part de les rendre plus compréhensibles et d'autre part de détecter dans ces données des corrélations significatives (règles de classement et de prédiction, associations, classifications, relations inconnues ou cachées…), des structures particulières restituant l'essentiel de l'information utile tout en réduisant la quantité de données permettant d'étayer les prises de décision. En bref, le Datamining est l'art d'extraire des informations (ou même des connaissances) à partir des données [5].

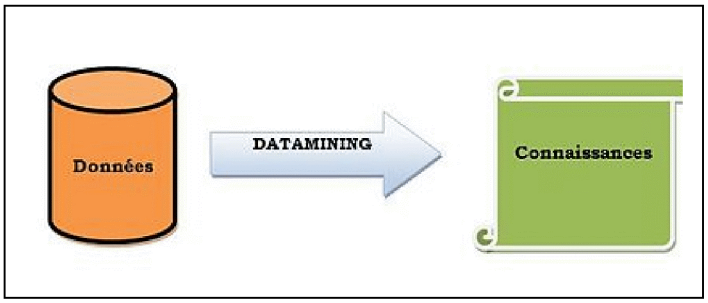


Figure 2 : *Datamining vers la connaissance [6].*

D'autres définitions ont été données par des experts dans le domaine parmi elles:

Le datamining est la candidature d'algorithmes spécifiques pour extraire des modèles des données [10].

La découverte de nouvelles corrélations et liens par le traitement d'un large volume de données.

Une confusion subsiste encore entre le datamining, que nous appelons en français « fouille de données », et knowledge discovery in data bases (KDD), que nous appelons en français « extraction des connaissances à partir des données » (ECD). Le Datamining est l'un des maillons de la chaîne de traitement pour la découverte des connaissances à partir des données. Sous forme imagée, nous pourrions dire que l'ECD est un véhicule dont le Datamining est le moteur [20].

2.2 Tâches du datamining:

2.2.1 Classification automatique supervisée :

La classification automatique supervisée consiste à examiner les caractéristiques d'un objet nouvellement présenté afin de l'affecter à une classe d'un ensemble prédéfini. Le

modèle généré permet, en outre de prédire ou estimer la valeur manquante ou erronée en utilisant le modèle de classification comme référence [2].

2.2.2 Classification automatique non supervisée :

C'est une tâche qui vise à identifier des ensembles d'éléments qui partagent certaines similarités. Elle se distingue de la classification automatique supervisée par le fait qu'elle ne se base pas sur des classes prédéfinies.

La classification non supervisée, ou encore clustering, permet de créer plusieurs clusters. Un cluster est un ensemble de données dites semblables. Chacun d'entre eux est séparé des autres et regroupe des données différentes. Plus la distance entre deux clusters est importante, plus les données qu'ils renferment sont différentes.

L'algorithme de clustering requiert de savoir en combien de clusters les données sont réparties, ce nombre étant dépendant des besoins propres à la recherche [2].

2.2.3 Les règles d'association :

C'est une tâche qui permet de découvrir les rapports de lien qui peuvent exister dans une base de données. Ces liens sont généralement exprimés sous la forme '' A --> B '' qui signifie que la présence de A implique la présence de B (avec une certaine probabilité).

Exemple : Un étudiant qui réussi en mathématiques réussira en algorithmique dans 80% des cas [2].

2.3 Domaines d'application :

Dans nos jours le Datamining est devenu plus en plus applicable dans différents domaines et secteurs d'activités parmi lesquels :

- *Sociétés commerciales :* L'étude d'appétence dans les sociétés commerciales vise à concentrer les mailings et le phoning sur les clients susceptibles de répondre favorablement.
- *Grandes surfaces :* L'analyse du ticket de la supérette permet de déterminer les articles souvent achetés simultanément et organiser les promotions en conséquence.
- *Secteur bancaire :* L'objectif premier des banques est de réduire le risque de prêts bancaires ; la création de modèles à partir des caractéristiques des clients permet de discriminer les clients à risque élevé.

- *Secteur d'assurances :* Dans le secteur d'assurances, le datamining est utilisé pour la détection de fraudes et leur anticipation.
- *Secteur médical :* Dans le secteur médical, le datamining est naturellement répandu. On trouve par exemple la détermination de groupes de patients susceptibles d'être soumis à des protocoles thérapeutiques déterminés. Les études sur les associations de médicament permettent, entre autres, de révéler les anomalies de prescription.
- *Secteur de l'éducation :* Le datamining est employé dans les établissements scolaires pour améliorer la qualité d'enseignement. Par exemple, répartir les élèves ayant une grande capacité d'assimilation dans la même classe [2]. On essayera de détailler ce domaine dans ce qui suit :

3. Datamining pour l'éducation (EDM) :

3.1 Définitions :

- **EIAH (Environnements Informatiques pour l'Apprentissage Humain):**

Les EIAH sont des environnements informatiques conçus dans le but de favoriser l'apprentissage humain, c'est-à-dire la construction de connaissances chez un apprenant.
Ces environnements relèvent de l'informatique, mais aussi d'autres disciplines comme la pédagogie, la psychologie, etc. La qualité d'un EIAH peut être définie en mettant le point sur deux entités. La première c'est la qualité de ce qu'il offre comme connaissances, activités, supports, outils, contenus, etc. la seconde est la qualité et la pertinence du contrôle pédagogique (s'il existe) proposé pour assurer les objectifs d'apprentissage [1].

- **Datamining pour l'éducation :**

C'est un processus conçu pour l'analyse de données issues uniquement de milieux éducatifs, pour mieux comprendre les apprenants et les situations dans lesquelles ils apprennent.
Le domaine du datamining pour l'éducation (EDM) s'appuie sur des méthodes et théories issues de nombreuses disciplines telles que la fouille de donnée, la découverte de connaissance, la psychométrie et l'apprentissage statistique. Il a pour but de construire des modèles et produire des résultats qui puissent aider à la conception et la réalisation d'applications et d'environnements innovants pour l'apprentissage, ainsi que d'apporter une

contribution théorique à la psychologie de l'éducation ou d'autres domaines en éducation [7].

3.2 Méthodes du datamining pour l'éducation:

Les types de méthode EDM sont liées à celles de l'extraction de données en général, mais avec quelques différences basées sur les caractéristiques uniques de données sur l'éducation.

Ryan Baker classifie les méthodes de l'EDM comme suit:

3.2.1 Prédiction :

- **Classe :**

Afin de comprendre notre vie quotidienne, nous sommes constamment obligés à classer, catégoriser et évaluer. La classification consiste à étudier les caractéristiques d'un nouvel objet pour l'attribuer à une classe prédéfinie. Les objets à classifier sont généralement des enregistrements d'une base de données, la classification consiste à mettre à jours chaque enregistrement en déterminant la valeur d'un champ de classe. Le fonctionnement de la classification se décompose en deux phases. La première étant la phase d'apprentissage. Dans cette phase, les approches de classification utilisent un jeu d'apprentissage dans lequel tous les objets sont déjà associés aux classes de références connues. L'algorithme de classification apprend du jeu d'apprentissage et construit un modèle. La seconde phase est la phase de classification proprement dite, dans laquelle le modèle appris est employé pour classifier de nouveaux objets [13].

- **Régression** pourrait se référer à:

- Régression (psychologie) : une réaction défensive à quelques impulsions non acceptées.
- L'analyse de régression , une technique statistique pour estimer les relations entre les variables. Il existe plusieurs types de régression:
    1. Modèle de régression linéaire.
    2. La régression linéaire simple.
    3. La régression logistique.
    4. La régression non linéaire.
    5. Régression non paramétrique.

6. Régression robuste.
7. Régression pas à pas.

- **Estimation de la densité :**

Dans les probabilités et les statistiques , l'estimation de densité est la construction d'une estimation, basée sur une observation donné, d'un sous-jacent inobservable fonction de densité de probabilité. La fonction de densité inobservable est considérée comme la densité selon laquelle une grande population est distribuée, les données sont généralement considérées comme un échantillon aléatoire de cette population.

3.2.2 Clustering :

L'analyse typologique ou le regroupement est la tâche de regrouper un ensemble d'objets (cluster), tel que chacun d'entre eux est séparé des autres et regroupe des données différentes. Plus la distance entre deux clusters est importante, plus les données qu'ils renferment sont différentes.

Il s'agit d'une tâche principale d'exploration minière de données , et une technique courante pour les statistiques d'analyse de données utilisé dans de nombreux domaines, y compris l'apprentissage automatique, la reconnaissance de formes, analyse d'images, la recherche d'information, et la bioinformatique .

3.2.3 L'exploration minière de la relation :

Implique de découvrir des rapports entre les variables dans un ensemble de données et de les coder comme des règles pour l'usage postérieur. Par exemple, l'exploitation de rapport peut identifier les rapports parmi des produits achetés dans des achats en ligne.

- **Règle d'association:**

Peut être employée pour trouver les erreurs d'étudiant qui se produisent, associant le contenu aux types d'utilisateur pour établir des recommandations pour le contenu qui est susceptible d'être intéressant, ou pour faire des changements aux approches d'enseignement. Ces techniques peuvent être employées pour associer l'activité d'étudiant, dans un système de gestion d'étude ou des forums de discussion, aux catégories d'étudiant ou pour étudier des questions telles que pourquoi l'utilisation des essais de pratique par des étudiants diminue au-dessus d'un semestre d'étude.

- **L'exploration minière de la corrélation**

- **Extraction de motifs séquentiels:**

Etablit les règles qui capturent les raccordements entre les occurrences des événements séquentiels, par exemple, trouvant des ordres temporels, tels que des erreurs d'étudiant suivies de chercher l'aide. Ceci pourrait être employé pour détecter des événements, tels que des étudiants régressant à faire des erreurs en mécanique quand ils écrivent avec des techniques de pensée plus complexes et plus critiques, et pour analyser des interactions dans les forums en ligne de discussion.

- **L'extraction de données de causalité**

3.2.4 Distillation de données pour le jugement humain :

Est une technique qui implique de représenter des données d'une manière qui permet à un humain d'identifier ou de classifier rapidement des dispositifs des données. Ce secteur de l'exploitation éducative de données améliore les modèles de machine-learning parce que les humains peuvent y identifier des modèles, ou les dispositifs de, les actions d'étudiant, les comportements d'étudiant, ou les données d'étude comportant la collaboration parmi des étudiants.

3.2.5 Découverte des modèles:

Est une technique qui implique d'employer un modèle validé d'un phénomène (développé la prévision, clustering, ou l'ingénierie cognitive manuelle) comme composant dans plusieurs analyses. Par exemple, Jeong et Biswas (2008) ont établi les modèles qui ont classé l'activité par catégorie d'étudiant des données de base de comportement : les interactions des étudiants avec un game-like d'environnement d'étude qui emploie l'apprentissage par l'enseignement.

En utilisant ces techniques, les chercheurs d'extraction de données éducatives peuvent établir des modèles pour répondre à des questions telles que :

- Quel ordre des matières est le plus efficace pour un étudiant spécifique ?
- Quelles actions d'étudiant sont associées à plus d'étude (par exemple, des catégories plus élevées de cours) ?
- Quelles actions d'étudiant indiquent la satisfaction, l'enclenchement, le progrès d'étude, etc.?
- Quels dispositifs d'un environnement d'étude en ligne mènent à améliorer l'étude ?

- Ce qui prévoira le succès d'étudiant ? [13]

3.3 Principales applications des méthodes EDM :

Les chercheurs d'extraction de données éducatives étudient une variété de secteurs, y compris l'apprentissage de l'individu du logiciel éducatif. L'ordinateur a soutenu l'apprentissage collaborative, le test Computer-Adaptive (et de tester largement), et les facteurs qui sont associés à l'échec d'étudiant ou le non-retentissement dans des cours.

Au travers de ces domaines, un domaine clé de l'application est dans l'amélioration de modèles d'études. Les modèles d'étudiants représentent des informations sur les caractéristiques d'un étudiant ou état, comme les connaissances actuelles, la motivation, la méta-connaissance, et les attitudes de l'étudiant.

La modélisation des différences individuelles d'étudiant dans ces domaines, permet au logiciel de répondre à ces différences, en améliorant de manière significative l'apprentissage des étudiants.

Les méthodes d'extraction de données éducatives ont permet à des chercheurs de modeler une plus large gamme d'attribut d'étudiant potentiellement appropriés en temps réel, y compris les constructions de plus haut niveau qu'était auparavant possible. Par exemple, ces dernières années, les chercheurs ont utilisé des méthodes EDM pour déduire si un étudiant est en jeu du système, l'expérience faible auto-efficacité, ou encore si un élève est ennuyé ou frustré. Les chercheurs ont également pu étendre la modélisation de l'élève au-delà même des logiciels éducatifs, à déterminer quels sont les facteurs prédictifs de l'échec scolaire ou le non-retentissement en cours de niveau collégial ou entièrement l'université.

Un deuxième domaine d'application des méthodes EDM a été dans la découverte ou l'amélioration des modèles de la structure de la connaissance d'un domaine. Par la combinaison de cadres de modélisation psychométriques avec des algorithmes recherchant dans l'espace de la littérature d'apprentissage automatique, un certain nombre de chercheur ont pu développer les approches automatisées qui peuvent découvrir des modèles de domaine précis, directement de données. Par exemple, Barnes a développé les algorithmes qui peuvent automatiquement découvrir un QMATRIX de données, Desmarais et Pavlik ont développé des algorithmes pour trouver la structure de connaissance d'ordre partiel (POKS), les modèles qui expliquent les corrélations de connaissance dans un domaine.

Un troisième domaine d'application de méthodes EDM a été dans l'étude du support pédagogique (tant dans l'étude du logiciel, que dans d'autres domaines, comme des comportements d'étude collaboratifs), vers la découverte de quels types de support pédagogique sont plus efficaces, en général ou pour les différents groupes d'étudiants ou dans différentes situations. Une méthode populaire pour étudier le support pédagogique apprend la décomposition. L'étude de la décomposition adapte des courbes d'apprentissage exponentielles aux données d'exécution, reliant le succès postérieur d'un étudiant au montant de chaque type de support pédagogique que l'étudiant a reçu jusqu'à ce point.

Un quatrième domaine clé d'application de méthodes EDM a été dans la recherche de preuve empirique pour raffiner et prolonger des théories éducatives et des phénomènes éducatifs célèbres, vers une compréhension plus profonde des facteurs clés affectant l'apprentissage, souvent en vue de mieux concevoir les systèmes d'apprentissage. Par exemple Gong, Rai et Heffernan ont examiné l'impact d'autodiscipline en éducation et ont constaté que, pendant que cela corrélé à la connaissance entrante plus haute et moins d'erreurs, l'impact réel sur l'éducation était marginal [9].

3.4 Exemples du datamining pour l'éducation :

Exemples1 : *L'analyse de traces générées par les EIAH pour produire des indicateurs.*

L'application de techniques de datamining sur les traces générées par des EIAH permet :

(1) d'identifier des motifs utiles concernant le comportement des étudiants.

(2) de savoir de quelle façon les étudiants apprennent dans les EIAH.

(3) de découvrir les associations entre les besoins des étudiants et une liste de matériels d'apprentissage.

(4) de connaître la cause des problèmes dans le système d'apprentissage.

(5) pour proposer des expériences et des activités d'apprentissage personnalisables aux étudiants.

(6) de classifier les étudiants en se basant sur les caractéristiques extraites à partir des données enregistrées afin de prévoir leurs notes finales.

(7) de décrire le processus d'apprentissage de l'étudiant avec l'objectif de classifier les étudiants pour leur donner différents conseils en se basant sur leurs compétences et sur d'autres caractéristiques…etc [1].

Exemple 2 : *Application du datamining dans l'enseignement supérieur.*

Les trois études de cas suivantes illustrent des applications clés du datamining dans l'enseignement supérieur.

Étude de cas 1 : *Créer des typologies de résultats d'apprentissage significatives.*

Défi :

La question posée est : *Que savent les établissements sur leurs étudiants ?*

Cette étude de cas démontre comment les universités publiques de banlieue peuvent établir des typologies de résultats d'apprentissage pour leurs étudiants en utilisant le datamining non supervisé.

Une université publique type de banlieue avec 15 000 inscrits en moyenne classe ses étudiants en trois catégories : « transfert », « formation professionnelle » ou « mise à niveau des compétences de base ». Ces identifications, toutefois, sont basées sur les objectifs éducatifs initialement déclarés par les étudiants lors de leur inscription. Bien qu'il s'agisse de classifications inclusives, elles n'aident pas à illustrer les différences entre chaque type d'étudiant.

Solution :

A fin d'établir des typologies appropriées pour les 15 000 étudiants, les chercheurs ont utilisé à la fois TwoStep et K-means, deux puissants algorithmes de classification. Mais l'application de ces algorithmes a donné des résultats mitigés.

Les chercheurs ont ensuite utilisé une autre méthode qui a examiné les résultats pédagogiques conjointement avec la longueur des études. Il est plus facile de définir des catégories de résultats que d'y classer les étudiants. Dans ce cas, l'algorithme TwoStep a produit les classes suivantes : « transferts », « formation professionnelle », « compétences de base », « résultats mitigés » et « décrocheurs ». K-means a validé ces classes.

Résultats :

Le datamining, combiné aux caractéristiques démographiques des étudiants et à d'autres informations, a permis à l'université de mieux comprendre ses types d'étudiants. Des conclusions plus évidentes sont apparues après analyse croisée des données démographiques et des classes.

Les typologies ont aussi leur importance parce qu'elles vont au-delà du profilage conventionnel des étudiants pour identifier des groupes homogènes, augmentant ainsi l'exactitude des algorithmes de modélisation prédictive.

Même si un projet de datamining ne débouche que sur la découverte de typologies appropriées sans aller jusqu'à la modélisation prédictive, les schémas et liens nouvellement découverts aident les éducateurs et les administrateurs à mieux répondre aux besoins de divers groupes d'étudiants.

Enfin, il n'y a pas de bonnes ou de mauvaises typologies. Par essence, une typologie est bonne si elle sert un objectif de recherche particulier [8].

Etude de cas 2 : *Planification universitaire et interventions – prédiction des transferts.*

Défi :

Cette étude de cas expose une solution à un épineux problème de l'enseignement supérieur : *comment prédire avec précision les résultats scolaires pour faciliter une intervention en temps opportun ?* Lorsque les établissements utilisent le datamining pour prédire quels étudiants sont les plus à risques, ils peuvent éviter qu'un étudiant échoue avant même que celui-ci sache qu'il court un risque d'échec. Bien qu'il ait toujours été difficile de découvrir quels étudiants font l'objet d'un transfert, la National Student Clearing House permet désormais aux universités publiques et aux grandes universités de recouper leurs données. Ainsi, data miners et décideurs peuvent lier le comportement scolaire des étudiants des universités publiques à leurs résultats en matière de transfert.

Solution :

Créer un modèle efficace de datamining avec ces données nécessite de combiner typologies et connaissance du domaine. La connaissance du domaine éducatif des transferts souligne que le moyen le plus efficace d'augmenter les transferts d'étudiants est d'identifier le plus tôt possible les étudiants orientés vers un transfert. Encadrer ceux qui sont les plus susceptibles de faire l'objet d'un transfert est bien plus significatif que compter le nombre d'étudiants qui ont accumulé suffisamment d'unités pour être transférés. Les analystes ont exécuté simultanément des algorithmes de réseau neuronal et de généralisation de règle afin de comparer la précision des prédictions.

Résultats :

Le datamining a permis au collège d'identifier avec précision les bons candidats au transfert. Après un vaste apprentissage automatique, l'algorithme de réseau neuronal a produit une précision de prédiction de 72%, et les algorithmes de généralisation de règle, C5.0 et C&RT, ont donné la même précision de prédiction de 80% [8].

Étude de cas 3 : *Prédire les contributions des anciens élèves.*

Défi :

Pour une université urbaine type de 25 000 étudiants, la population d'anciens élèves peut être dix fois supérieure au nombre d'inscrits. La plupart des universités envoient régulièrement des courriers aux anciens élèves, qu'ils répondent ou pas. Des courriers de cette envergure coûtent généralement plus de 100 000 dollars par an. Cette étude de cas montre comment le datamining aide les universités à cibler les anciens élèves les plus susceptibles d'apporter des contributions.

Solution :

Il est souvent difficile de déterminer si les courriers affectent directement le volume et la somme des contributions des anciens élèves. Face au même type de courrier, un ancien élève pourra contribuer régulièrement, un autre pas. La présence d'aberrations, comme ces anciens élèves qui donnent inopinément de grosses sommes, ne fait que rajouter à la confusion.

Comment les établissements identifient-ils et cultivent-ils les relations avec ces « aberrations » ?

La figure 3 montre l'intérêt d'utiliser le datamining pour déterminer quels anciens cibler que d'envoyer des courriers à tous les anciens élèves. La ligne courbe est le taux de retour optimal (contributions des anciens élèves) tel que prédit par le datamining. La ligne droite à 45 degrés est le résultat supposé si toute la population recevait ce courrier. Dans ce cas, le graphique indique que lorsque le courrier atteint le 30ème centile de la population qui, selon le datamining, devrait y répondre, 80% répondent par une contribution. Sans le datamining, l'université serait obligée d'envoyer ce courrier à 70% ou plus des anciens élèves (à peu près tout le monde) afin d'atteindre ceux qui y répondraient.

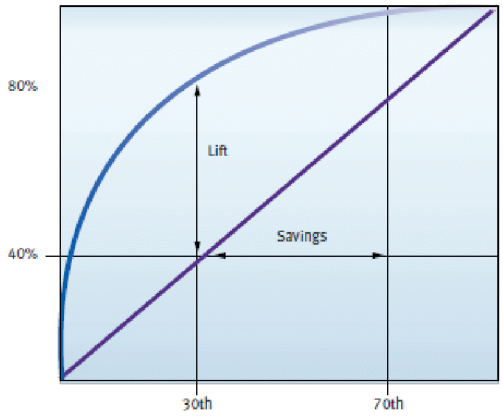


Figure 3 : *Avantages que le datamining peut apporter aux campagnes de promesses de dons adressées aux anciens élèves.*

Résultats :

Grâce au datamining, l'université a découvert un moyen de rendre son courrier plus efficace et a augmenté les contributions des anciens élèves, tout en réduisant les frais d'expédition. La meilleure description en est le concept appelé « lift ». Si le datamining peut identifier rapidement des donateurs potentiels par un rapport de deux sur quatre (prédisant correctement deux sur quatre qui feront un don), alors l'université peut obtenir des résultats en n'envoyant des courriers qu'aux 40% indiqués de la population des anciens élèves, gagnant ainsi un temps et un argent considérables [8].

Conclusion :

L'expression Educational Datamining est apparue vers les années 2000. Depuis, plusieurs conférences internationales ont été organisées dans le but d'améliorer cette discipline.

Le datamining pour l'éducation est un puissant outil d'analyse et d'exploration de grandes bases de données en vue d'extraire des connaissances importantes pour les personnes concernées dans les environnements EIAH.

Chapitre 2

Conception

Introduction:

Notre travail s'inscrit dans le cadre de l'évaluation des connaissances des apprenants, il consiste à la conception et la réalisation d'une application, qui permet à partir des tables relationnelles concernant les résultats des apprenants, de faire des calculs algébriques et la transformation de données pour pouvoir appliquer ensuite les algorithmes du datamining implémentés dans la plateforme Weka. Pour concevoir notre application, il est nécessaire de la divisée en deux parties principales :

- *Preprocessing* : (Sélection des données + calculs algébriques + transformation de données).
- *Application des algorithmes du datamining*.

1. Le preprocessing :

Il consiste à la préparation des données pour l'étape suivante qui est l'application des algorithmes du datamining, voici le schéma global :

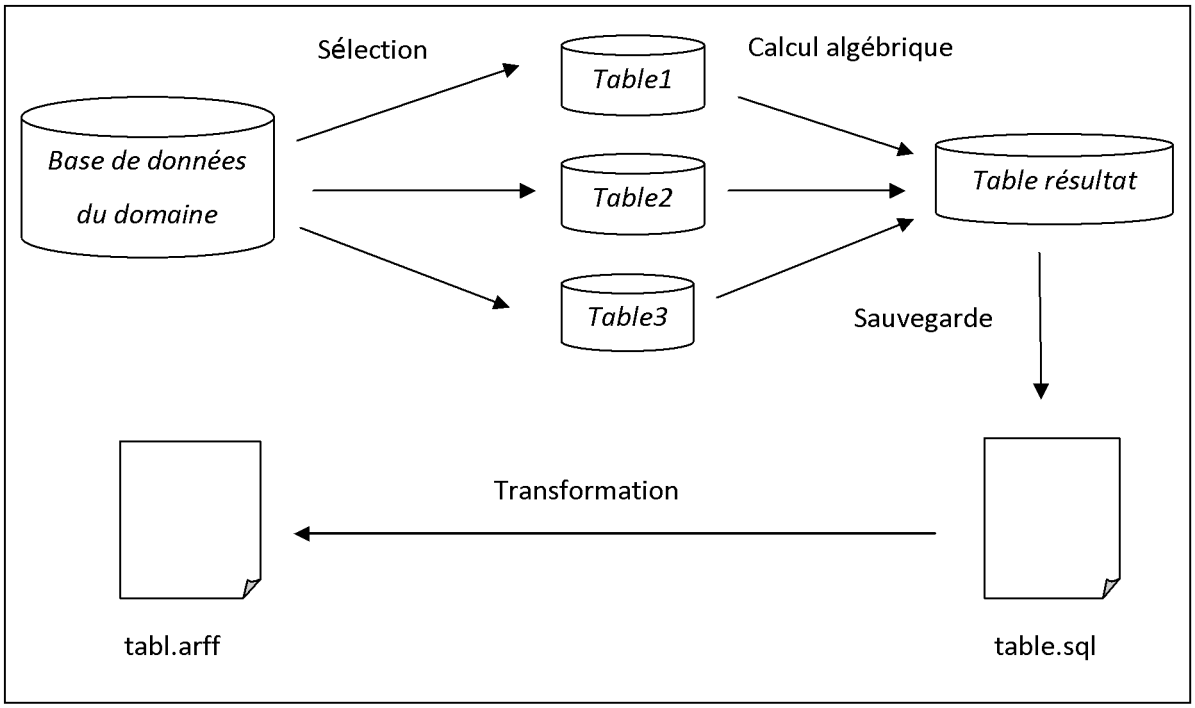


Figure 4 : *Schéma global du preprocessing.*

Ce module est divisé en trois parties, mais avant cela on doit rappeler d'abord le modèle du domaine:

- ***Modélisation des résultats des apprenants :***

D'après le modèle entité/association et le schéma relationnel décrits dans [3], sur lesquels on a apporté des améliorations pour des besoins d'analyse (application du clustering), nous proposons le modèle entité/association et relationnel suivants :

- ***Le modèle entité/association :***

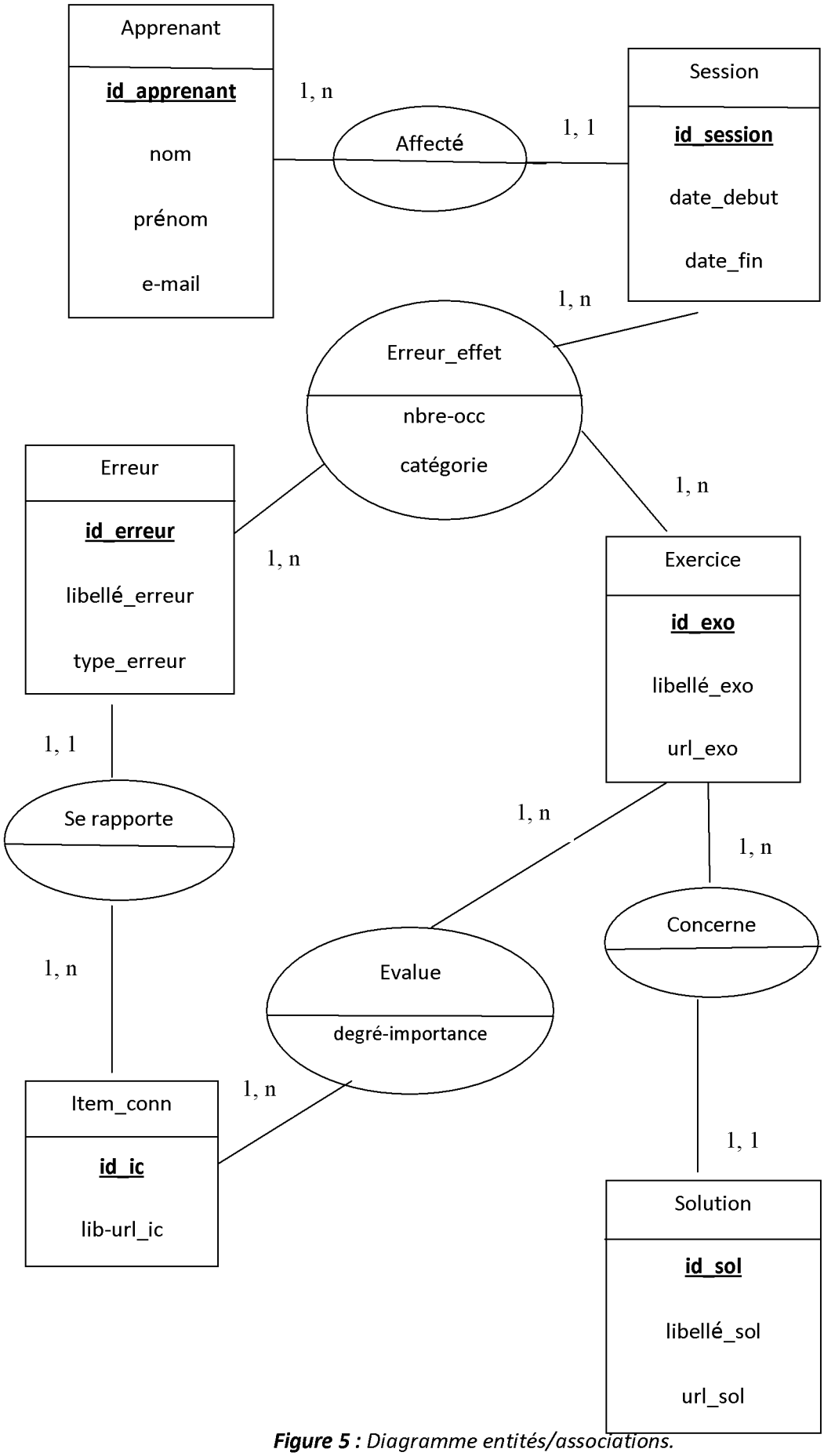


Figure 5 : *Diagramme entités/associations.*

Selon la réalité étudiée on peut établir les règles suivantes :

- Un apprenant peut effectuer une ou plusieurs sessions.
- Une session est propre à un seul apprenant.
- Un exercice peut avoir une ou plusieurs solutions.
- Un exercice peut traiter plusieurs items de connaissances.
- Une erreur se rapporte à un item de connaissances.
- Un apprenant peut commettre une ou plusieurs erreurs en résolvant un exercice d'une ou plusieurs sessions.

- ***Le modèle relationnel du domaine :***

table	commentaire
Apprenant (**id_apprenant**, nom, prénom, e-mail)	Appartient au modèle de l'apprenant
Session (**id_session**, date_début, date_fin, *id_apprenant*)*	C'est l'intersection de l'apprenant avec le domaine
Exercice (**id_exercice**, libellé_exercice, url_exo)	Table contenant les exercices correspondants à une session d'apprentissage
Erreur (**id_erreur**, libellé_erreur, type_erreur*, id_ic*)*	Contient toutes les erreurs connus sémantiques ou lexicaux-syntaxiques
Solution (**id_sol**, libellé_sol, url_sol, *id_exercice*)*	Toutes les solutions possibles pour chaque exercice
Item_conn (**id_ic**, lib-url_ic)	Contient tous les items de connaissances qu'on peut attribuer à des notions d'apprentissage
Erreur_effect (**id_session, id_erreur, id_exercice**, nbre_occ, catégorie)	Table contenant toutes les erreurs effectuées par un apprenant lors d'une session donnée en résolvant un exercice donné. Nbre_occ : précise combien de fois une erreur donnée s'est répétée dans un exercice lors d'une session donnée. Catégorie : permet de classer un apprenant selon le nbre_occ.
Evalue (**id_exercice**, **id_ic**, degré-importance)	Table qui contient les items de connaissances évalués dans un exercice donné

Figure 6 : *Table relationnelle pour la session d'apprentissage.*

1.1 La sélection des données :

Cette étape permet à partir d'une base donnée relationnelle de sélectionner une ou plusieurs tables pour les affichées et pouvoir visualiser l'ensemble des attributs et des tuples.

Exemple : Sélection de deux tables (Apprenant et Session) :

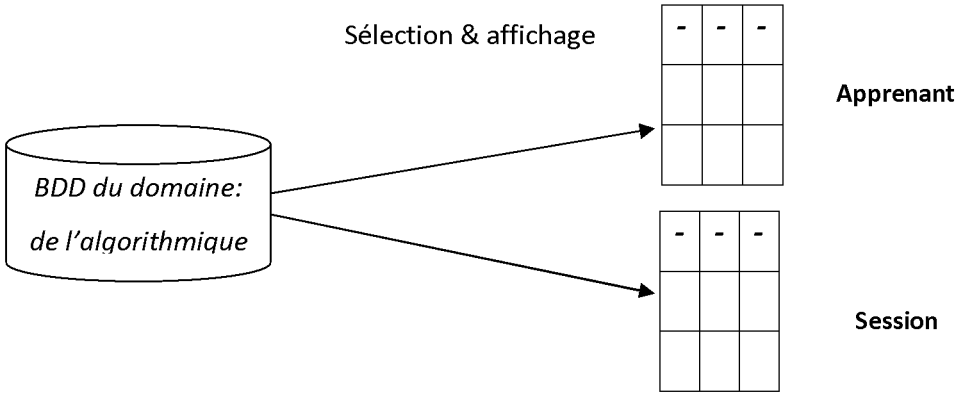


***Figure 7** : Exemple de sélection de données à partir de la base de données du domaine du système web siela.*

La requête SQL qui permet la sélection d'une table est la suivante :

SELECT * FROM nomTable ;

1.2 Le calcul algébrique :

Cette étape consiste à effectuer des opérations algébriques sur les tables relationnelles. Les opérateurs algébriques les plus utiles et les plus courants sont :

- ***La jointure :*** Désignée par le symbole (|><|), combine deux tables en une nouvelle table. Ainsi, l'opération R|><|PS permet de joindre deux tables R et S d'après la condition de jointure (prédicat de jointure) P.
  Requête SQL: SELECT <attributR1>, <attributR2> FROM R1, R2 WHERE <condition>;
- ***La projection :*** L'opérateur de projection π M(R) construit à partir de la table R, une sous table dont les noms d'attributs sont définis dans M.
  Requête SQL: SELECT [distinct] <attribut> FROM R1;
- ***La restriction (sélection):*** Désignée par le symbole σ F(R) permet la sélection des tuples d'une table R d'après une formule F.
  Requête SQL: SELECT [distinct] <attribut> FROM R1 WHERE <condition>;

1.3 La transformation des données d'apprentissage:

Les outils du datamining exigent un format de données spécifiques pour pouvoir implémenté les algorithmes qu'ils proposent. Dans notre cas le format accepté par Weka est le format arff, et en raison de la non disponibilité d'un outil permettant directement la transformation d'une table relationnelle en arff, a part l'outil en ligne qui permet le transfert de données du format CSV au format arff et les modifications manuelles, on propose d'implémenter un module sous java, qui prend en entrée un fichier au format sql et le transforme pour produit en sortie un fichier au format arff.

1.3.1 Le format SQL :

C'est le format de base de données, il permet de décrire la table relationnelle sous forme de texte. Voici la structure de base d'un fichier sql :

CREATE TABLE *nom-table*
(attribut1 type1,
attribut2 type2,
.
.
attributN typeN
) ;

} *La création de la table*

INSERT INTO *nom-table (attribut1, attribut2, ... attributN)*
VALUES *(val-attribut1, val-attribut2, ... val-attributN) ;*

} *L'insertion d'enregistrements*

1.3.2 Le format ARFF :

Un fichier ARFF (Attribute-Relation File Format) est un fichier de texte ASCII qui décrit une liste d'exemples ou d'instances partageant un ensemble d'attributs, voici sa structure:

@RELATION *<nom-relation>*
@ATTRIBUTE *<nom-attribut1> <type donnée1>*
@ATTRIBUTE *<nom-attribut2> <type donnée2>*
.
.
@ATTRIBUTE *<nom-attributN> <type donnéeN>*

} L'en-tête

@DATA

val-attribut1, val-attribut2, … val-attributN

.

.

} Les données ARFF

- Le <nom-attribut> doit commencer avec un caractère alphabétique, et s'il contient des espaces alors le nom de l'attribut entier doit être mis entre quottes.
- Le <type donnée> peut être un parmi les quatre types supporté par weka :
  - numeric : peut être un nombre réel ou entier.
  - string.
  - date [<date format>].
  - <spécification nominale> : définit ainsi {<nom-nominal1>, <nom-nominal2>, <nom-nominal3>, …}
- Dans la section de données ARFF :
  - Chaque instance est représentée sur une seule ligne.
  - Les valeurs de l'attribut pour chaque instance sont délimitées par des virgules, et doivent apparaître dans l'ordre dans lequel ils ont été déclarés dans la section de l'en-tête.
  - Une valeur manquante est représentée par un point d'interrogation.

1.3.3 Le processus de transformation :

Le module de transformation contient deux étapes :

Etape 1 : *Sauvegarder la table relationnelle issue du calcul algébrique sous format sql :*

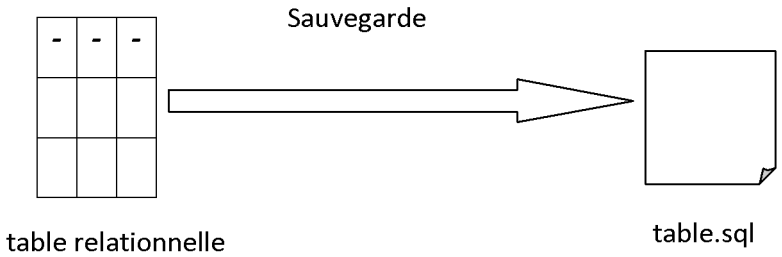


Figure 8 : *Schéma de sauvegarde de données.*

Etape 2 : *Transformer le fichier sql en un fichier arff.*

Figure 9 : *Processus de transformation de données.*

2. Application des algorithmes de data mining :

2.1 Présentation des algorithmes du data mining :

Le data mining offre plusieurs algorithmes, qui sont implémentés dans des plates formes différentes comme **weka**, que nous allons utiliser pour effectuer quelques tests.

Dans notre cas on va s'intéresser essentiellement à 3 algorithmes, classés sous trois catégories comme suit :

2.1.1 La classification :

La plupart des algorithmes de classification tentent de trouver un modèle (une fonction mathématique) qui explique le lien entre les données d'entrée et les classes de sortie. Un ensemble d'apprentissage (ensemble de classes) est donc utilisé par l'algorithme. Cette méthode de raisonnement est appelée inductive car on induit la connaissance (le modèle) à partir des données d'entrées (objets à classifier) et des sorties (leurs classes). Grâce à ce modèle, on peut alors prédire les classes de nouvelles données. Le modèle est bon s'il permet de bien prédire. On cite 3 catégories d'algorithmes : les réseaux de neurones, arbre de décision et K-NN. Dans se qui suit on va présenter les arbres de décision [2].

2.1.1.1 Les arbres de décision :

Un arbre de décision est, comme son nom l'indique, un outil d'aide à la décision qui permet de classifier une population d'individus selon les valeurs de leurs attributs. C'est une représentation graphique de la procédure de classification où :

- Une feuille indique une classe.
- Un nœud spécifie un test que doit subir un certain attribut.
- Chaque branche sortant de ce nœud correspond à une valeur possible de l'attribut en question.

Pour classifier un nouvel objet, on suit le chemin partant de la racine (nœud initial) à une feuille en effectuant les différents tests d'attributs à chaque nœud. L'arbre permet d'émettre des prédictions sur les données par réduction, niveau par niveau, du domaine de solutions.

La démarche générale de construction de l'arbre de décision consiste en deux étapes :

- Construction de l'arbre à partir des données (apprentissage).
- Elagage de l'arbre dans le but d'alléger l'arbre résultant souvent volumineux.

Il existe une grande variété d'algorithmes pour construire des arbres de décision ; quelques uns des plus répandus portent les noms de ID3 (*Inductive Decision-tree*) introduit par Quinlan et amélioré pour devenir C4.5, CART (*Classification And Regression Trees*, introduit par Breiman et CHAID (*Chi-squared Automatic Interaction Detection*). Dans ce qui suit on présente la démarche de C4.5 pour la construction de l'arbre de décision.

- **L'algorithme C4.5**

Construction de l'arbre :

Le principe général démarre d'un arbre vide et procède à la construction de l'arbre de manière inductive (à partir des données) et récursive en commençant par l'ensemble des objets tout entier. Si tous les objets sont de même classes, une feuille est créée avec le nom de la classe. Sinon, l'ensemble d'objets est partagé en sous-ensembles selon la valeur d'un certain attribut qui subissent le même traitement.

Afin d'avoir un arbre de décision concis et suffisant, il ne suffit pas de traiter les attributs séquentiellement. Au contraire, toute la richesse des arbres de décision consiste à choisir judicieusement les attributs d'éclatement pour aboutir, par le chemin le plus court et nécessaire, au plus grand nombre d'objets de la même classe.

Elagage de l'arbre :

L'opération d'élagage de l'arbre se fait en deux phases : Le pré-élagage et le post-élagage.

Le pré-élagage consiste à fixer un critère d'arrêt qui permet de stopper la construction de l'arbre lors de la phase de construction.

Le post-élagage est un traitement qui intervient après la construction entière de l'arbre. Il consiste à supprimer les sous-arbres qui n'améliorent pas l'erreur de classification [2].

Dans notre cas on va implémenter l'algorithme J48, qu'est une implémentation open source de l'algorithme C4.5.

2.1.2 Le clustering :

La majorité des algorithmes de clustering peuvent être répartis en trois grandes familles :

- ***Les algorithmes de partitionnement :*** Ils adoptent une recherche itérative des classes jusqu'à l'optimisation d'un critère d'arrêt (par exemple atteindre une certaine inertie). Les algorithmes de partitionnement sont k-Means, k-Medoïds, PAM, CLARA et CLARANS.

- ***Les algorithmes hiérarchiques :*** Sont une décomposition hiérarchique d'un tableau, ou ensemble de données. Deux stratégies sont envisageables dans ce type de classification, la première ***ascendante*** (Agglomératifs***,*** consiste à partir d'une classe par objet et de fusionner ces classe deux à deux jusqu'à obtention d'une classe englobant la totalité des objets ou atteinte d'une condition d'arrêt, on cite l'algorithme DIANA. L'autre stratégie est dite ***descendante*** (divisifs), à l'inverse de la précédente, elle part d'une classe englobant tous les objets puis divise a chaque itération une classe en classes plus petites jusqu'à obtention d'un objet par classe ou atteinte d'une condition d'arrêt, l'algorithme BIRCH est proposé [10].

- ***Les algorithmes à base de densité :*** Les classes sont formées selon le voisinage des objets et le niveau de densité de chaque objet. Parmi les algorithmes les plus connus dans cette catégorie, nous citons DBSCAN et OPTICS [2].

2.1.2.1 L'algorithme kMeans (Les centres mobiles) :

L'algorithme kMeans qui est un algorithme de partitionnement (ou l'algorithme des centres mobiles) a été introduit par J.MacQuenn et mis en œuvre sous sa forme actuelle par E.Forgy. Il est le plus utilisé dans les applications scientifiques et industrielles car il est le plus simple et le plus rapide.

Dans cet algorithme, chaque classe est représentée par la moyenne (centroide). kMeans est un algorithme itératif. Il commence avec un ensemble de *k* individus de référence choisis de façon aléatoire. Les individus de données sont ainsi partitionnés dans *k* classes, un individu appartient à une classe si le centre de cette classe est le plus proche de lui (en terme de distance). La mise à jour des centroides et l'affectation des individus de données aux classes sont réalisées pendant les itérations successives.

L'application de l'algorithme kMeans se fait suivant les étapes suivantes :

1) Choix de k, le nombre de cluster à créer.

2) Choix des centres des clusters de manière aléatoire à partir des objets en entrée.

La procédure adoptée pour le choix des centres des Clusters initiaux est extrêmement importante car elle a un impact direct sur le résultat final du Clustering. Il est donc très important de choisir des clusters bien séparés.

L'algorithme kMeans basique se base sur une initialisation aléatoire, après plusieurs utilisation du résultat de l'algorithme kMeans, des chercheurs proposent des améliorations au niveau de cette étape puisqu'elle impact le résultat final [14].

- **Algorithme de kMeans :**

L'algorithme prend en entrée:

K: le nombre de classes voulu.

M: le nombre d'individu

Entrée : Population de *m* individus, *k*

Choisir aléatoirement *k* centres initiaux *c1*, ..., *ck*

Répéter

- Répartir chacun des *m* individus dans une classe *i* dont le centre *ci* est le plus proche en termes de distance.
- Calculer les nouveaux centres : pour tout *i*, *ci* est la moyenne des individus de la classe *i* (centroide).

Jusqu'à ce qu'il n'y ait plus de changement.

Figure 10 : *L'algorithme kMeans [14].*

La figure 11 montre un exemple de déroulement de l'algorithme k-Means sur un nuage d'objets bidimensionnels, avec k = 3.

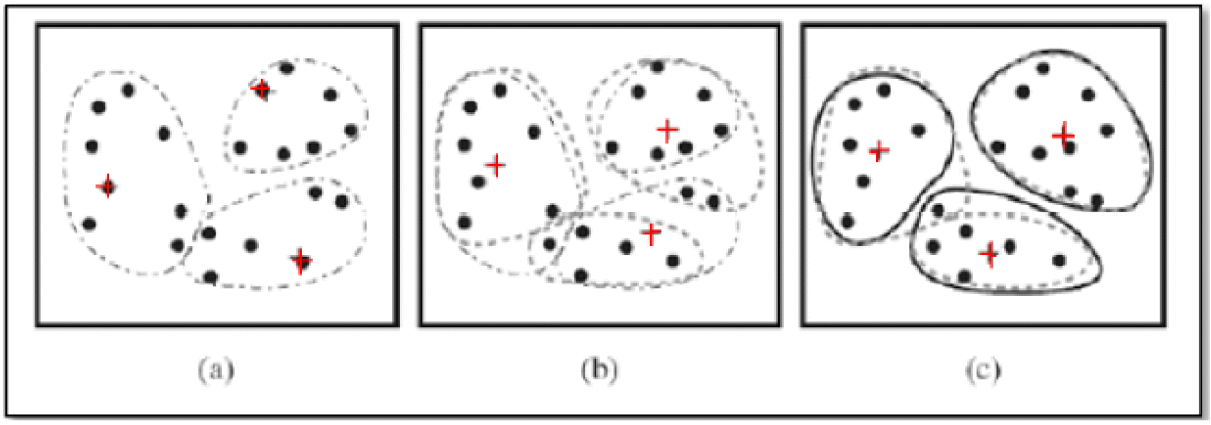


Figure 11 : *Partitionnement basé sur k-Means [2].*

Figure 11.a : Choix aléatoire de trois centre initiaux (objets) marqués par (+) et affectation de chaque objet restant au centre le plus proche.

Figure 11.b : Calcul du nouveau centre pour chaque classe (moyenne des objets de la classe) et redistribution des objets selon les nouveaux centres.

Figure 11.c : Le processus est répété jusqu'à stabilité des classes.

Les avantages de cet algorithme :

- Il s'adapte bien pour des populations de tailles importantes.
- Il est relativement efficace ; si n le nombre d'objets, k le nombre de classes et t le nombre d'itération, l'algorithme converge, généralement, avec k et t suffisamment inférieur à n ($k, t << n$).
- Il est indépendant de l'ordre d'arrivée des données.

Parmi les inconvénients de cet algorithme, on cite :

- Il est applicable seulement dans le cas où la moyenne des objets est définie ;
- Le nombre de classes k, doit être spécifié à priori ;
- Il n'est pas adapté pour découvrir des classes avec des structures non-convexes, et des classes de tailles et formes différentes [14].

2.1.3 Les règles d'associations :

Les règles d'association, étant une méthode d'apprentissage non supervisé, permettent de découvrir à partir d'un ensemble de transactions, un ensemble de règles qui expriment une possibilité d'association entre différents attributs.

2.1.3.1 L'algorithme Apriori :

Développé par Agrawal R. and Srikant R. L'algorithme Apriori est un algorithme par niveaux qui permet de découvrir les sous-ensembles d'items fréquents en partant de ceux dont la longueur est 1 et en augmentant la longueur au fur et à mesure. Cet algorithme est fondé sur la propriété des sous ensembles d'items fréquents. Chaque niveau comprend une phase de génération de tous les itemsets candidats et une phase d'évaluation pour en éliminer les non fréquents.

Apriori est un algorithme qui a été conçu pour rechercher les itemsets fréquents. Il est donc à la charge de l'utilisateur d'exploiter ces itemsets pour la génération des règles intéressantes.

Pour cela, de simples algorithmes de recherche sont utilisés [8].

- **Algorithme de Apriori :**

L'algorithme Apriori prend en entrée:

D: la population d'individus.

MinSup : le seuil minimum de support.

Entrée : Une population d'individus *D*, *MinSup*.

k=0, *Lk* = ∅.

C1 = l'ensemble des *1*-itemsets dans *D*.

L1 = l'ensemble des *1*-itemsets fréquents de *C1*.

Tant que (*Lk+1*) est non vide

Ck+1 = (*k+1*)-itemsets candidats générés à partir de (*Lk*)

Lk+1 = ensemble des (*k+1*)-itemsets fréquents de *Ck+1* (support > SupMin)

k = *k*+1 ;

Retourner : È ∪ Lk

Figure 12 : *L'algorithme Apriori [15].*

Les (*k+1*)-itemsets candidats sont générés à partir d'un ensemble de *k*-itemsets *Lk* par le biais de la procédure suivante :

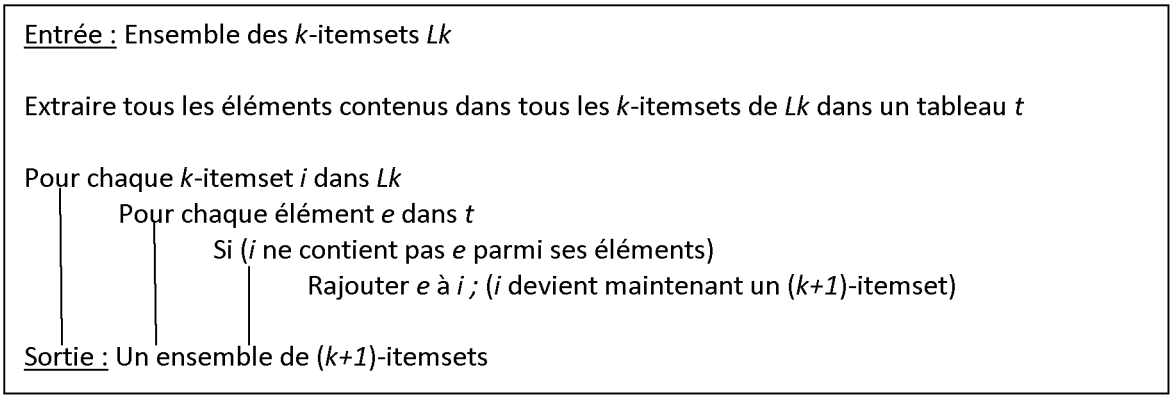


Figure 13 : *L'algorithme de génération des itemsets candidats [15].*

Exemple :

Soit les *3*-itemsets *i1* = {1, 2, 3}, *i2* = {1, 2, 4}.

Les éléments extraits à partir de *i1 et i2* sont : {1}, {2}, {3}, {4}.

Les 4-itemsets candidats générés sont : {1, 2, 3, 4}.

La génération des règles intéressantes à partir de chaque *k*-itemset fréquent *i* se fait en trois étapes :

1. Générer tous les sous-itemsets, de tailles inférieures à *k*, à partir de *i* dans un tableau *t.*

2. Générer toutes les implications possibles entre les éléments de *t* (règles potentielles);

3. Eliminer les règles qui ne respectent pas la contrainte sur la confiance.

Exemple :

Soit le 3-itemset *i* = {1, 2}.

Les sous-items générés à partir de *i* sont : {1}, {2}.

Les implications possibles : {1} → {2} (confiance >ConfMin), {2} → {1} (confiance < ConfMin).

La règle {2} → {1} est éliminée car elle ne respecte pas la contrainte sur la confiance.

Les avantages de « Apriori » :

- Il fournit des résultats clairs: règles faciles à interpréter.
- Il est relativement simple.

Les inconvénients de « Apriori »:

- Il est très coûteux en temps de calcul.

- Il produit un nombre important de règles triviales ou inutiles (qui seront détruites par la suite) [14].

2.2 Etapes d'application des algorithmes de data mining:

Il consiste à décrire les différentes étapes d'application des algorithmes du data mining, en commençant par le choix de l'algorithme de data mining jusqu'à l'affichage et la sauvegarde des résultats. Voici un schéma détaillé:

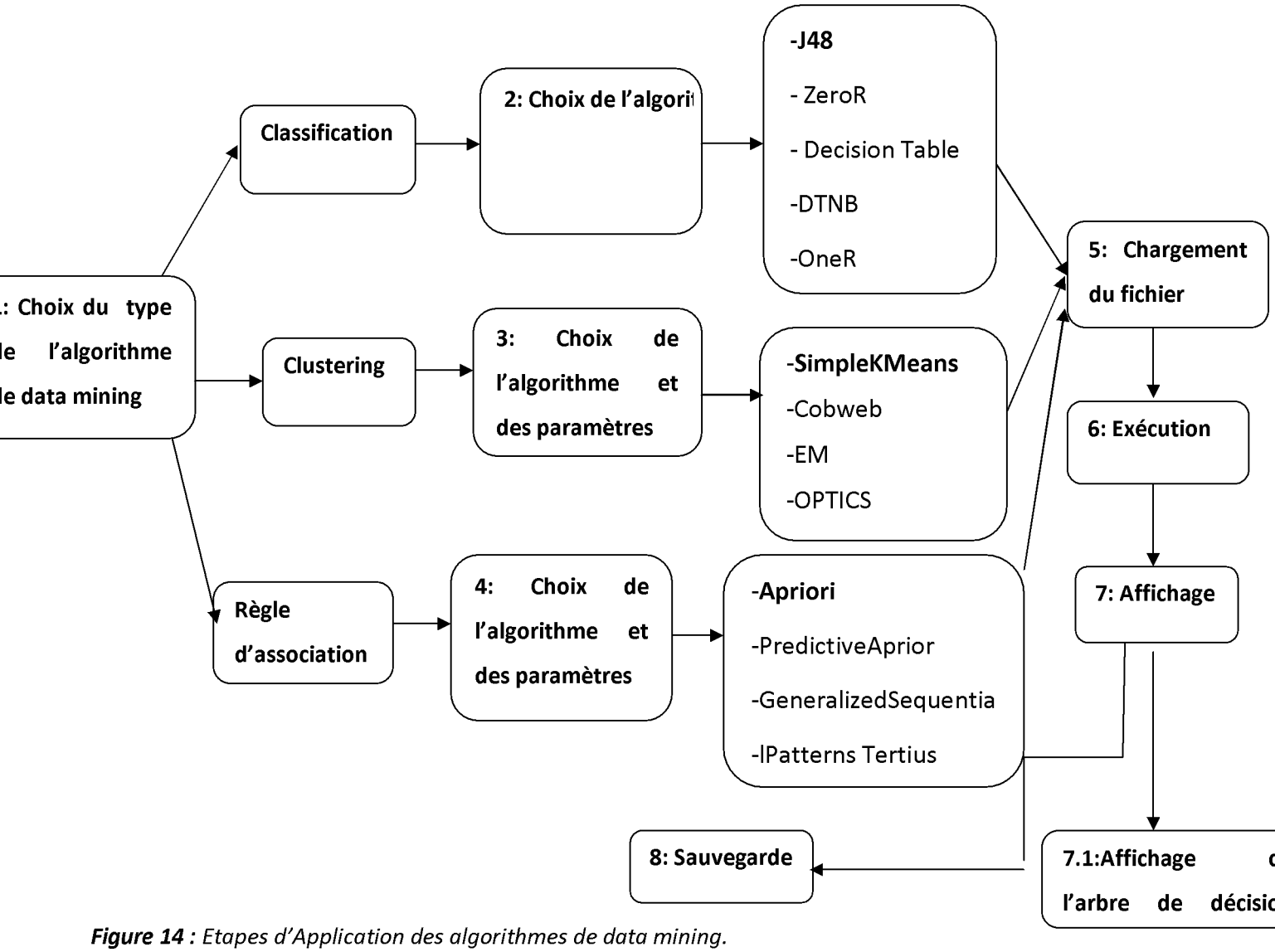


Figure 14 : *Etapes d'Application des algorithmes de data mining.*

1. *Choix du type de l'algorithme de data mining* : Cette étape permet de choisir un algorithme du data mining parmi les trois catégories d'algorithmes suivantes:
   - ***Classification*** .
   - ***Clustering.***
   - ***Règle d'association.***
2. *Choix de l'algorithme et des paramètres:* Il existe plusieurs algorithmes de classification: J48, ZeroR, Decision Table, DTNB, OneR…etc, dans ce cas on a décidé d'implémenter l'algorithme J48.

   On peut choisir les 2 modes d'évaluation suivant:
   - *Use training set* : Toutes les données servent à la fois à apprendre et à tester les modèles.
   - *Cross validation* : Pour une valeur K, le jeu est divisé en K partitions. L'une constitue le jeu de test, les autres forment le jeu d'apprentissage. Ce processus est répété K fois, chaque partition étant utilisée une fois comme jeu de test. Une valeur de 10 pour K en général conseillée.
3. *Choix de l'algorithme et des paramètres:* Parmi les algorithmes du clustering (Cobweb, EM, SimpleKMeans,…etc) on a choisit d'implémenter Kmeans. Et les paramètres à choisir sont les suivant :

   -*Nombre de cluster K* : C'est le nombre de partitions à crées.

   -*Ignorer un attribut* : Pour ignorer un attribut sur lequel on ne veut pas appliquer le clustering.
4. *Choix de l'algorithme et des paramètres:* Les algorithmes de Règle d'association sont : FiltredAssociator, PredictiveApriori, GeneralizedSequentialPatterns, Tertius. On implémente l'algorithme Apriori. Et pour paramètre on a :

   -*Nombre de règles* : Le nombre de règles que l'algorithme doit produire.

5. Chargement du fichier : Regroupe les opérations nécessaires pour la lecture de données stockées dans le fichier ARFF.

6. *Exécution* : consiste en l'exécuter de l'algorithme avec le mode choisit et les paramètres sélectionnés.

7. *Affichage* : Concerne la mise en forme et la visualisation du résultat de l'exécution de l'algorithme.

7.1 *Affichage de l'arbre de décision (J48): Visualisation de l'arbre de décision.*

8. *Sauvegarde* : Cette étape permet de sauvegarder les informations essentielles du résultat de l'exécution pout l'interprétation.

3. Architecture globale de l'application :

L'application qu'on propose de faire est divisée essentiellement en 2 parties (ou modules), la première partie est le preprocessing détaillé plus haut, et la deuxième concerne l'implémentation des algorithmes de data mining. Le schéma suivant présente son architecture globale :

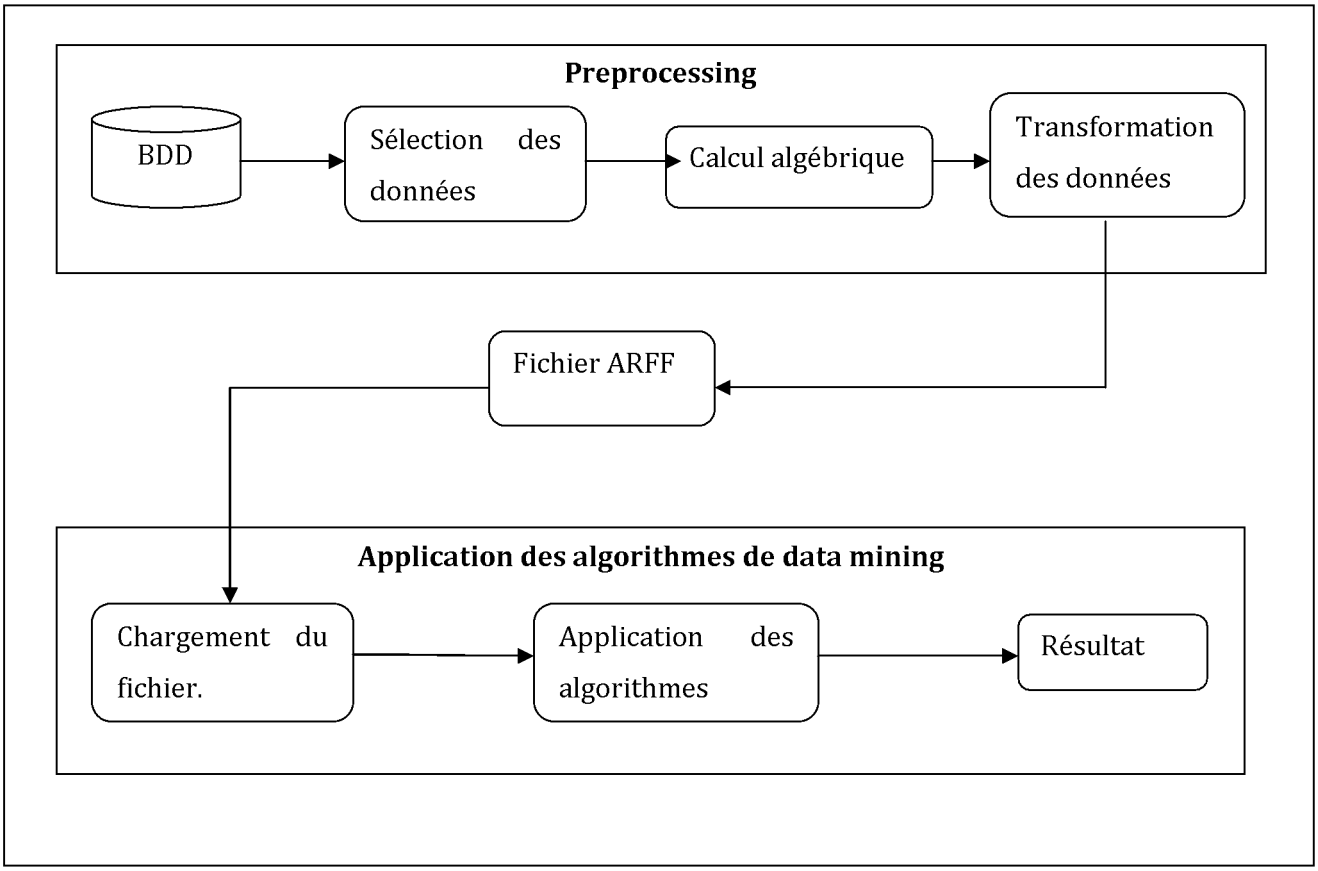


Figure 15 : *Architecture globale de l'application*

Conclusion :

Dans ce chapitre on a détaillé chacune des parties qui constitue notre application, en commençant tout d'abord par le preprocessing, ensuite l'application des algorithmes du data mining. Et nous avons conclus le chapitre par l'architecture globale de toute l'application, ce qui va nous servir d'appui dans le chapitre suivant pour la réalisation de cette dernière.

Chapitre 3

Implémentation et test

Introduction:

Ce dernier chapitre a pour objectif, d'implémenter l'API Weka sous l'environnement Java, ce qui nous pousse a présenté tout d'abord les langages de programmation et les outils utilisés, puis les étapes du chargement de l'API Weka et son utilisation sous NetBeans, ensuite nous allons décrire l'environnement d'exécution, ainsi que quelques interfaces illustrant le déroulement de notre application. Et on termine avec quelques tests de l'API.

Partie 1 : Implémentation

1. L'environnement de développement :

1.1 Les langages de programmation :

Pour la réalisation de notre application nous avons utilisés le langage Java et plus précisément les JSP, Servlets et classes Java pour l'implémentation de l'API Weka et le preprocessing, le langage HTML et Javascript pour rendre les pages web plus dynamiques. Voici une petite description de ces langages:

1.1.1 Le langage Java [18]:

Java est un langage de programmation récent (les premières versions datent de 1995) développé par Sun MicroSystems. Il a rapidement intéressé les développeurs et il a obtenu un très grand succès sur la scène d'Internet pour deux raisons principales :

- C'est un langage orienté objet dérivé du C, mais plus simple à utiliser et plus « pur » que le C++. On entend par « pur » le fait qu'en Java, on ne peut faire que de la programmation orienté objets, cette dernière signifie que vous pouvez capitaliser toutes les classes et programmes qui ont était développés et tester correctement, et vous pouvez les exploités directement sans avoir à concevoir d'autres programmes.
- Il est multi plates-formes : Java est un langage qui peut travailler sur plusieurs plates-formes c.-à-d. que les programmes tournent sans modification sur tous les environnements (systèmes d'exploitations) où Java existe (Windows, Unix, Linux, Solaris, Mac…), et ceci grâce à la machine virtuelle Java (JVM) qui n'est pas physiquement présente (machine Software), et qui sert a compilé le programme indépendamment du système d'exploitation sur lequel il est installé.

- ***Les Servlets :***

Une servlet est un programme qui s'exécute côté serveur en tant qu'extension du serveur. Elle reçoit une requête du client, elle effectue des traitements et renvoie le résultat. La liaison entre la servlet et le client peut être directe ou passer par un intermédiaire comme par exemple un serveur http.

Ecrite en java, une servlet en retire ses avantages : la portabilité, l'accès à toutes les API de java dont JDBC pour l'accès aux bases de données, ...

Une servlet peut être invoquée plusieurs fois en même temps pour répondre à plusieurs requêtes simultanées.

- ***Les JSP (Java Server Pages) :***

Les JSP sont une technologie Java qui permettent la génération de pages web dynamiques. Elles permettent de séparer la présentation sous forme de code HTML et les traitements sous formes de classes Java définissant un bean ou une servlet. Une JSP est habituellement constituée :

- de données et de tags HTML.
- de tags JSP.
- de scriptlets (code Java intégré à la JSP).

1.1.2 Le langage HTML (Hyper Text Markup Language*):*

HTML est un langage de description (et non pas un langage de programmation) qui permet de décrire l'aspect d'un document, d'y inclure des informations variées (textes, images, sons, animations ….) et d'établir des relations cohérentes entre ces informations grâce aux liens hypertextes [18].

1.1.3 Javascript :

Le Java Script est le langage favori des créateurs de sites web. Il permet en quelques lignes de code de dynamiser les pages web. C'est un langage interprété par les navigateurs qui n'exige aucune configuration spéciale des serveurs web [16].

1.2 Les outils :

Comme outils nous avons utilisés : NetBeans 6.7.1, EasyPHP 5.3.3.1, et GlassFish v2.1

1.2.1 NetBeans :

C'est l'un des outils de développement en Java (IDE Java), créé par Sun Microsystems. Il contient toutes les fonctionnalités d'un bon IDE :

- Un debugger.
- Un outil de modélisation UML.
- Outils nécessaire pour faires des applications J2EE.
- Il est gratuit [18].

1.2.2 EasyPHP :

L'outil le plus indispensable pour développer et implémenter une application web dynamique est Easy PHP. En effet, Easy PHP est un « package » qui contient tous les programmes nécessaires pour pouvoir traiter du PHP. Il regroupe un serveur web Apache, un SGBD de base de données (PHPMyAdmin) et un serveur SQL de base de données [19].

1.2.3 GlassFish :

GlassFish est un serveur d'applications facile et rapide, compatible avec la plate-forme Java entreprise Edition (Java EE) 5 permettant le développement et la distribution d'application Web et de services Web [17].

2. L'utilisation de l'API Weka sous NetBeans :

2.1 Chargement de l'API :

Pour le chargement de l'API Weka sous NetBeans, on procéde ainsi :

- Installer le logiciel Weka 3.6.
- Copier le fichier weka.jar qui se trouve dans le répertoire de Weka : *« C:\Program Files\Weka-3-6 »*, et le mettre dans le répertoire de notre projet API Weka : *« C:\Users\HOME\Documents\NetBeansProje\API Weka».*
- A partir de NetBeans et dans l'arborescence du projet API Weka, au niveau de Libraries on clique avec le bouton droit et on choisit « *Add JAR/Folder... »*, dans la fenêtre qui apparaît on sélectionne notre fichier weka.jar comme montrer dans la figure ci-dessous :

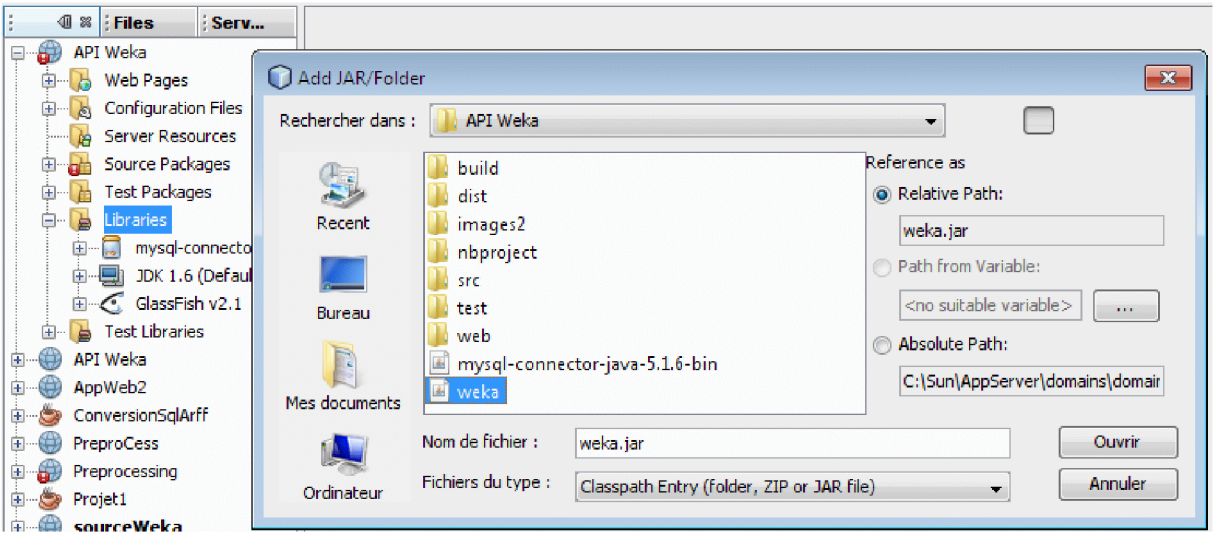


Figure 16 *: L'ajout du fichier weka.jar.*

2.2 Utilisation de l'API :

Une fois le fichier weka.jar est ajouté à la librairie, on peut utiliser l'API Weka pour implémenter les algorithmes, en commençant tout d'abord, par ajouter (importer) les packages spécifiques à chaque algorithme, ensuite toutes les instructions nécessaires pour la programmation des options, du choix de l'algorithme, de l'affichage de l'arbre de décision…etc. Voici quelques fragments de code :

- ***Algorithme J48 :***

```
import weka.classifiers.Evaluation;
import weka.classifiers.trees.J48;
import weka.core.Attribute;
import weka.core.Instances;
import weka.gui.explorer.ExplorerDefaults;
import weka.gui.treevisualizer.PlaceNode2;
import weka.gui.treevisualizer.TreeVisualizer;
```

Figure 17 : *Les packages concernant l'algorithme J48.*

```
     public void actionPerformed (ActionEvent e){
       try{
    TreeVisualizer tv;
     final javax.swing.JFrame jf =new javax.swing.JFrame("Weka Classifier Tree Visualizer
     jf.setSize(800,600);
     jf.getContentPane().setLayout(new BorderLayout());
     tv =new TreeVisualizer(null,tree.graph(),new PlaceNode2());
     jf.getContentPane().add(tv, BorderLayout.CENTER);
     jf.addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent e) {
        jf.dispose();
       }
     });
     jf.setVisible(true);
     tv.fitToScreen();
       } catch(Exception ex){
ex.printStackTrace();
      }}
    });
```

Figure 18 *: Le code java (méthode) permettant l'affichage de l'arbre de décision.*

- ***Algorithme KMeans :***

```
import weka.clusterers.ClusterEvaluation;
import weka.clusterers.SimpleKMeans;
import weka.core.Instances;
import weka.gui.explorer.ExplorerDefaults;
```

Figure 19 : *Les packages concernant l'algorithme KMeans.*

```
ClusterEvaluation eval= new ClusterEvaluation();
SimpleKMeans kmeans= new SimpleKMeans() ;

kmeans.buildClusterer(instances);
N = instances.numInstances () ;
n = instances.numAttributes () ;

out_k.append(kmeans.toString());
```

Figure 20 : *Quelques instructions essentielles de l'algorithme KMeans.*

- ***Algorithme Apriori :***

```
import weka.associations.Apriori;
import weka.associations.AssociatorEvaluation;
import weka.core.Instances;
```

Figure 21 : *Les packages concernant l'algorithme Apriori.*

```
Apriori apr = new Apriori();
AssociatorEvaluation eval= new AssociatorEvaluation();
int n, N;
int nbRules;

N = instances.numInstances () ;
n = instances.numAttributes () ;

apr.buildAssociations(instances);
apriori.append(eval.evaluate(apr, instances));

apr.setNumRules(nbRules);
```

Figure 22 : *quelques instructions essentielles de l'algorithme Apriori.*

3. Exécution de l'application:

3.1 L'environnement d'exécution :

Pour l'exécution de l'application, on a besoin du fichier WAR, GlassFish v2.1, et de EasyPHP 5.3.3.1 :

- **Le fichier WAR (Web Application ARchive) :**
  - **Définition :** Un fichier WAR est un fichier JAR () utilisé pour contenir un ensemble de JSP, servlets, classes Java, fichiers XML, et des pages web statiques (HTML, JavaScript…), le tout constituant une application web. Cette archive est utilisé pour déployer une application web sur un serveur d'application [22].
  - **Génération :** On génère le fichier WAR à partir de l'outil NetBeans comme schématisé dans la figure ci-dessous :

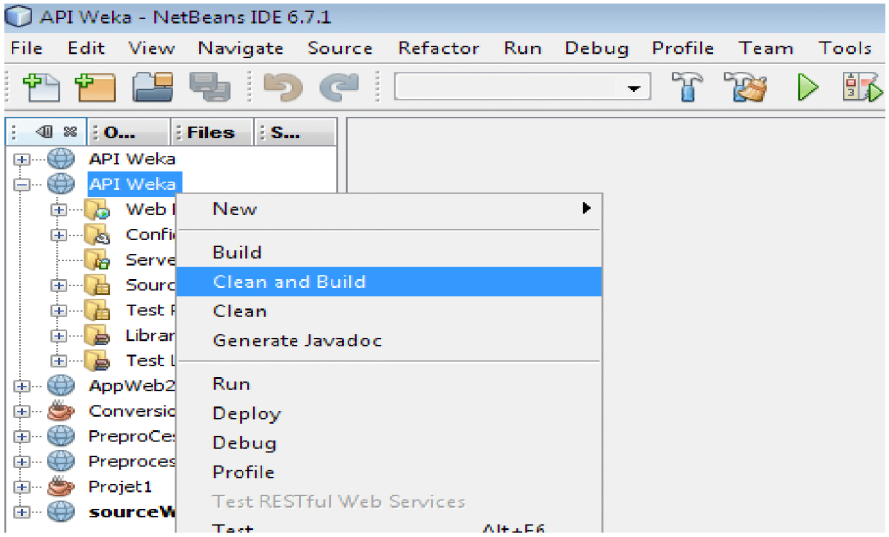


Figure 23: *Génération du fichier WAR.*

On trouve le fichier dans le dossier **dist** du répertoire du projet API Weka.

➢ **Déploiement de l'application sur GlassFish :**

Voici les étapes à suivre :

- Démarrer le serveur GlassFish.
- Se connecter à la console d'administration du serveur, avec l'adresse suivante : ***« localhost:4848 »***, en saisissant tout d'abord le login et le mot de passe.
- Au niveau de la console d'administration, dans la partition ***Applications -> Web Applications***, on clique sur le bouton ***Deploy…,*** pour choisir le fichier WAR au niveau de la nouvelle fenêtre qui s'affiche, et on valide avec le bouton ***OK***, comme détailler dans la figure suivante :

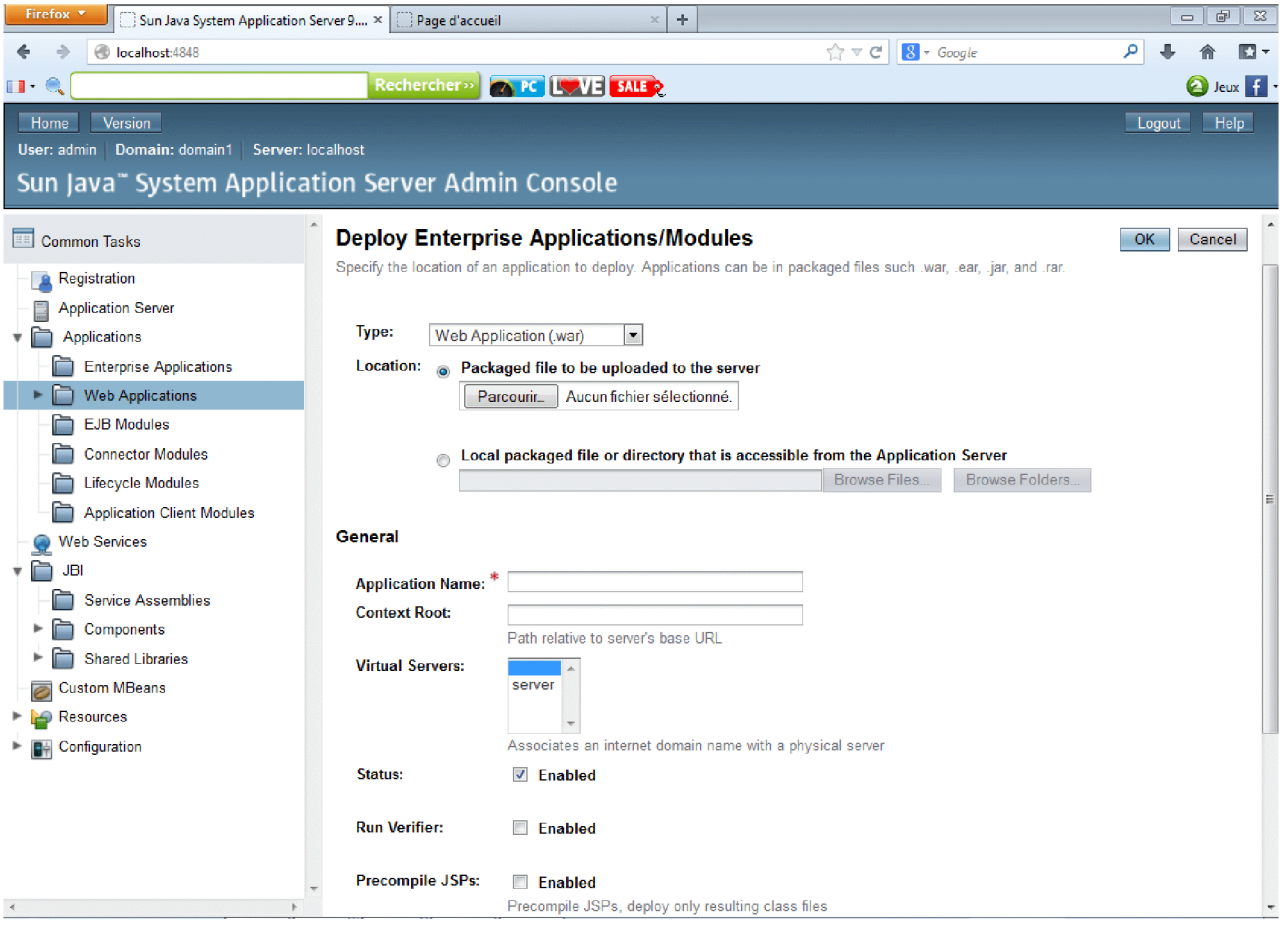


Figure 24 *: Déploiement du fichier WAR.*

- Enfin, on saisit l'adresse : ***« localhost:8080/API_Weka/accueil.jsp »***, pour accéder à la page d'accueil.

➢ **La connexion à la base de données :**

Pour la connexion à la base se données, il suffit simplement de démarrer EasyPHP.

3.2 Présentation des interfaces de l'application:

Dans ce qui suit on présentera quelques interfaces essentielles illustrant les étapes d'exécution de notre application :

A partir de la page d'accueil, on accède à l'espace *data mining*, en cliquant sur le lien « **Data mining** ». Une fois à l'intérieur (voir la figure suivante), on remarque les deux liens suivants:

- ***Preprocessing*** : Pour préparer le fichier d'entrée.
- ***Utiliser Weka*** : Pour appliquer les algorithmes de data mining.

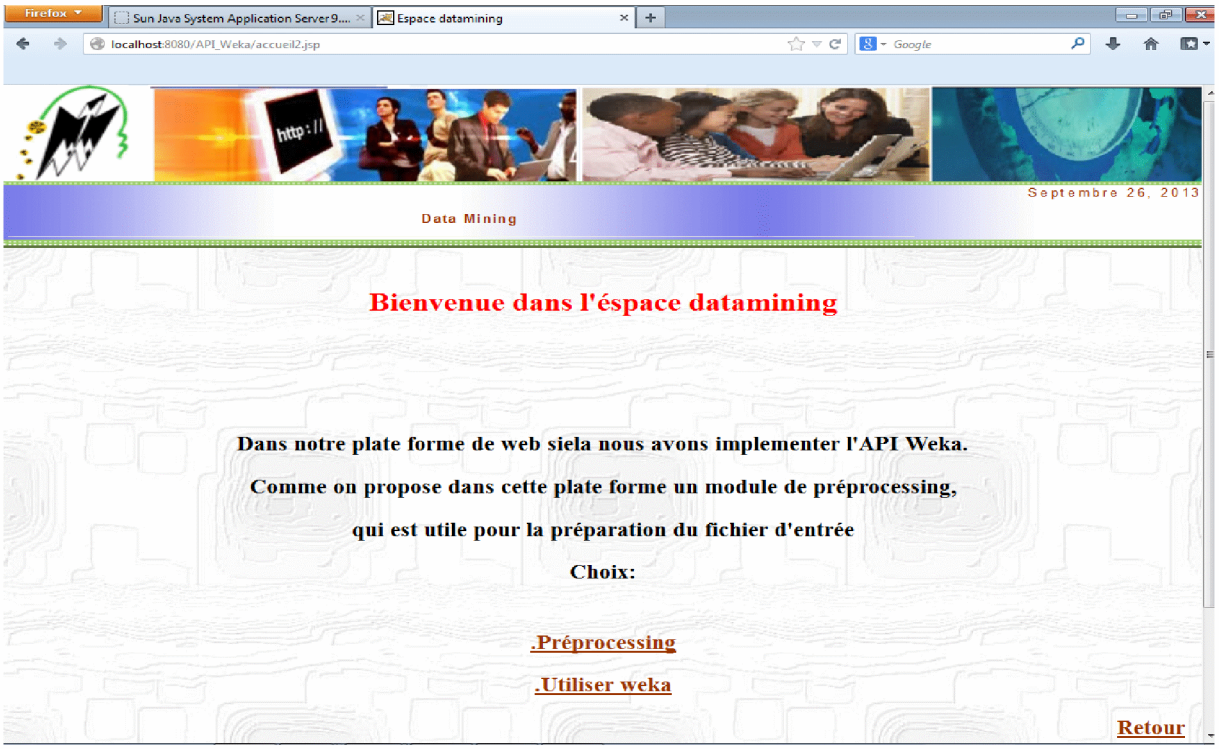


Figure 25 *: Espace Data mining.*

Si on clique sur le lien ***Preprocessing,*** une fenêtre apparaît, permettant de choisir une base de données et cliquer sur ***Afficher*** pour passer à la fenêtre qui suit, où on sélectionne les tables qu'on veut visualiser et valider avec le bouton ***Afficher***, puis une nouvelle fenêtre s'affiche, où on peut voir le contenu des tables ainsi que leurs attributs, où il est également possible de faire un calcul algébrique, en saisissant une requête dans le champ ***saisir requête,*** comme on peut le voir dans la figure ci-dessous.

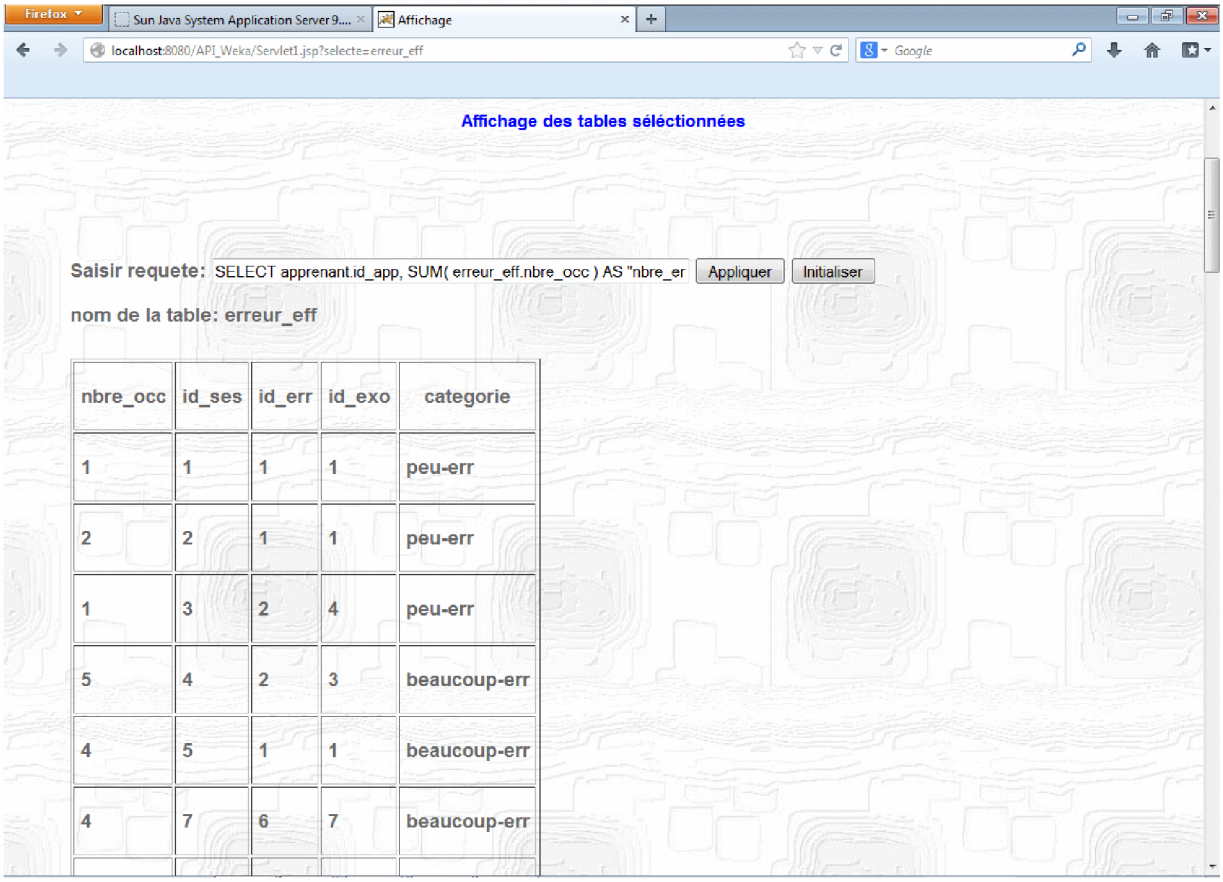


Figure 26 *: Calcul algébrique.*

Une fois la requête est validée avec le bouton ***Appliquer***, on peut enregistrer la table résultante sous format *SQL*, à travers la fenêtre de sauvegarde qui s'ouvre en saisissant le nom du fichier. Ensuite, on convertit le fichier SQL en un fichier ARFF, on parcoure tout d'abord le nom du fichier SQL à convertir, en cliquant sur le bouton ***OpenFileSQL*** pour afficher la fenêtre d'ouverture, puis on clique sur le bouton ***SaveFileARFF***, et dans la fenêtre qui s'ouvre, on choisit l'emplacement où on veut sauvegarder et on saisit le nom du fichier ARFF, et enfin on clique sur le bouton ***Convertir***, voir figure suivante :

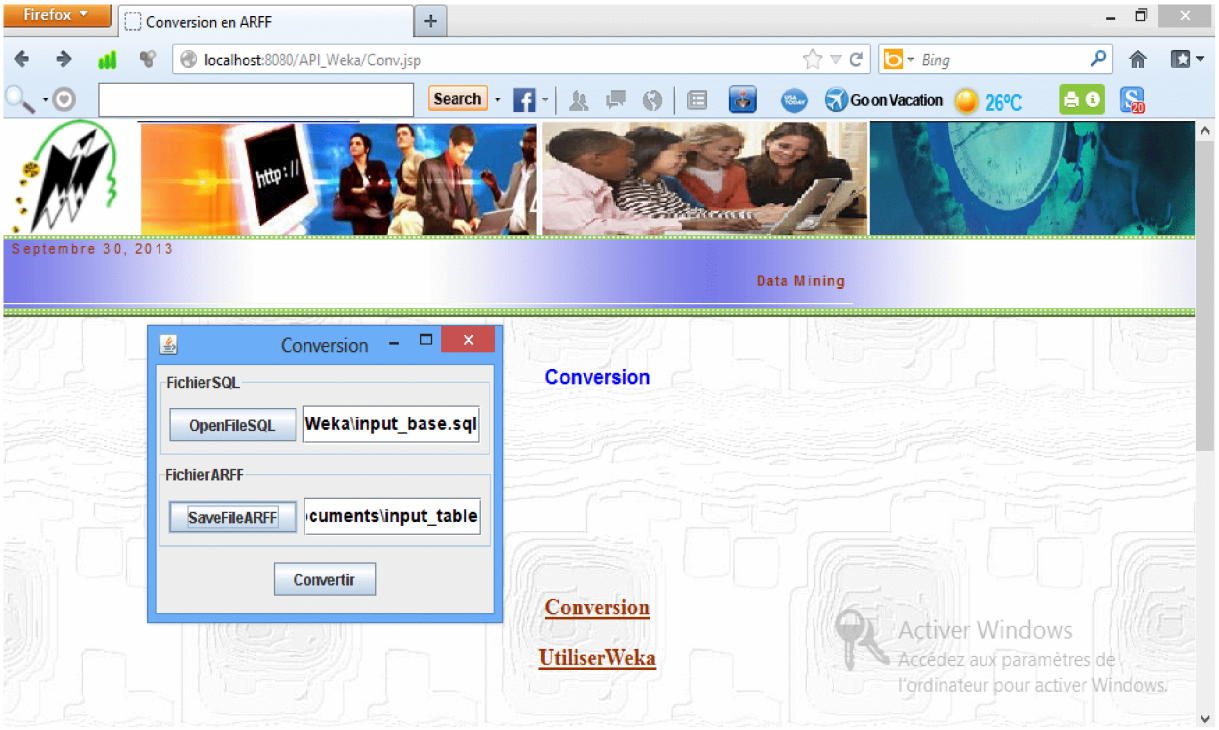


Figure 27 : *Conversion du fichier SQL.*

Il est temps maintenant d'appliquer les algorithmes du data mining sur le fichier ARFF. Pour cela on clique sur le lien ***Utiliser weka***, la figure qui apparaît nous permet de choisir un algorithme parmi les 3 catégories d'algorithmes du data mining :

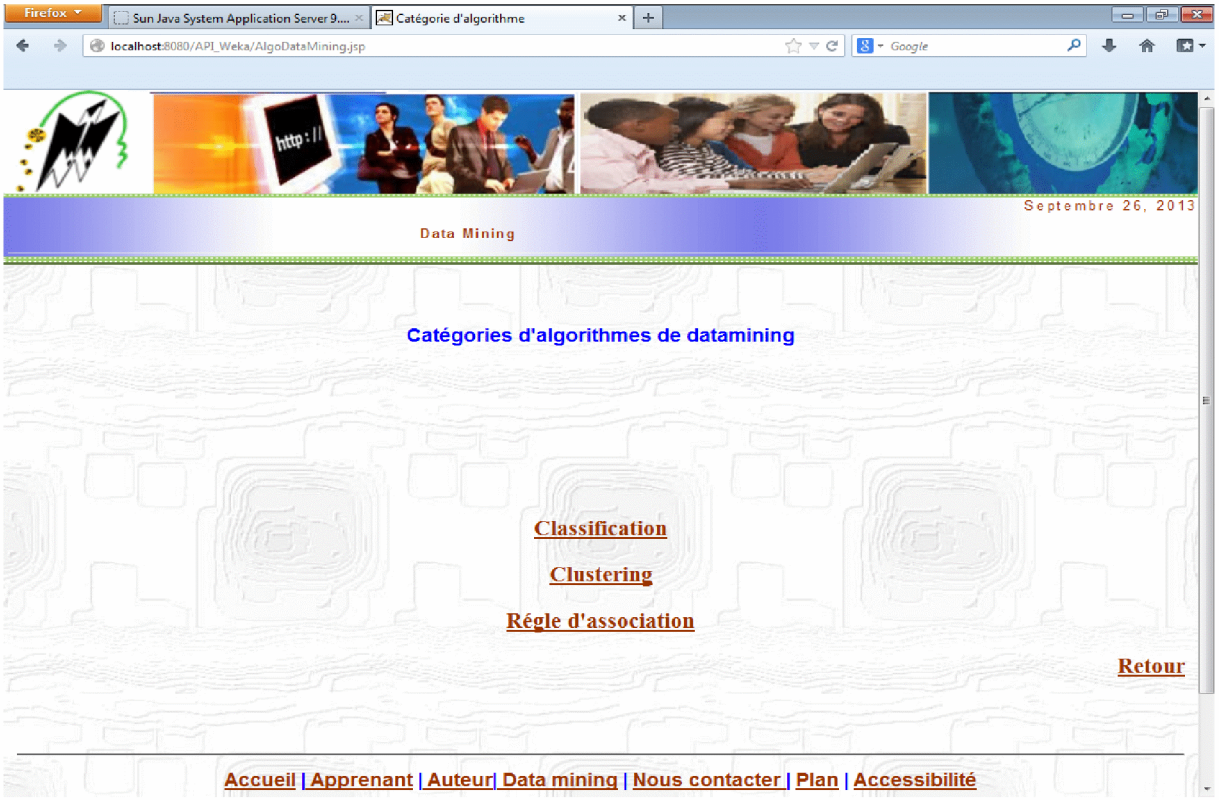


Figure 28 : *Les trois catégories d'algorithmes de data mining.*

On cliquant sur « ***Classification »*** dans la figure 28, on aura l'interface suivante de l'algorithme J48 :

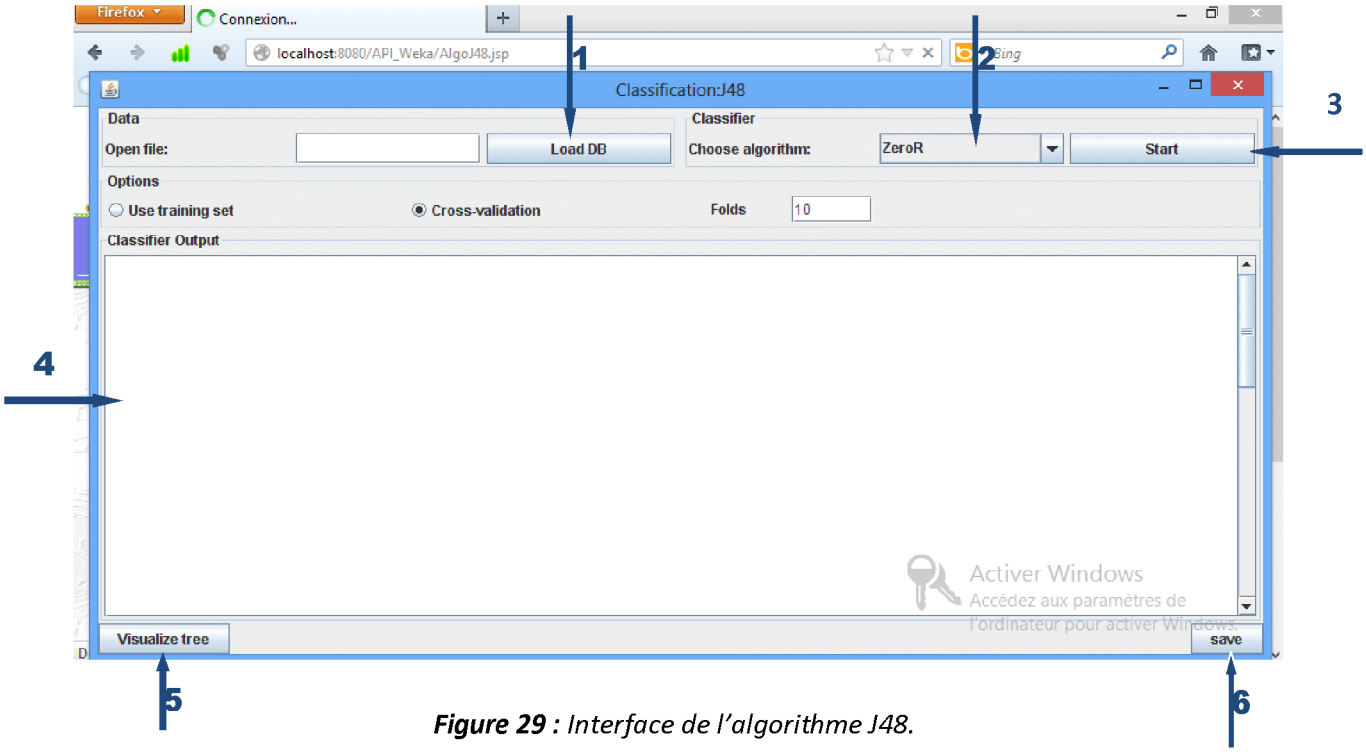


Figure 29 : *Interface de l'algorithme J48.*

Les flèches sur la figure montrent les principales étapes de cet algorithme :

1 : cliquer sur le bouton « *Load DB* », une fenêtre s'ouvre pour choisir le fichier arff.

2 : ce champ nous donne la possibilité de choisir un algorithme.

3 : lancer l'exécution de l'algorithme avec le bouton « *start* ».

4 : cette zone, permet la visualisation du résultat.

5 : cliquer sur le bouton « *Visualize tree* », et une fenêtre s'ouvre pour visualiser l'arbre de décision.

6 : le bouton « save » permet la sauvegarde du résultat.

Dans la figure 28, cliquer sur le lien « ***Clustering*** » pour avoir l'interface de kmeans suivante :

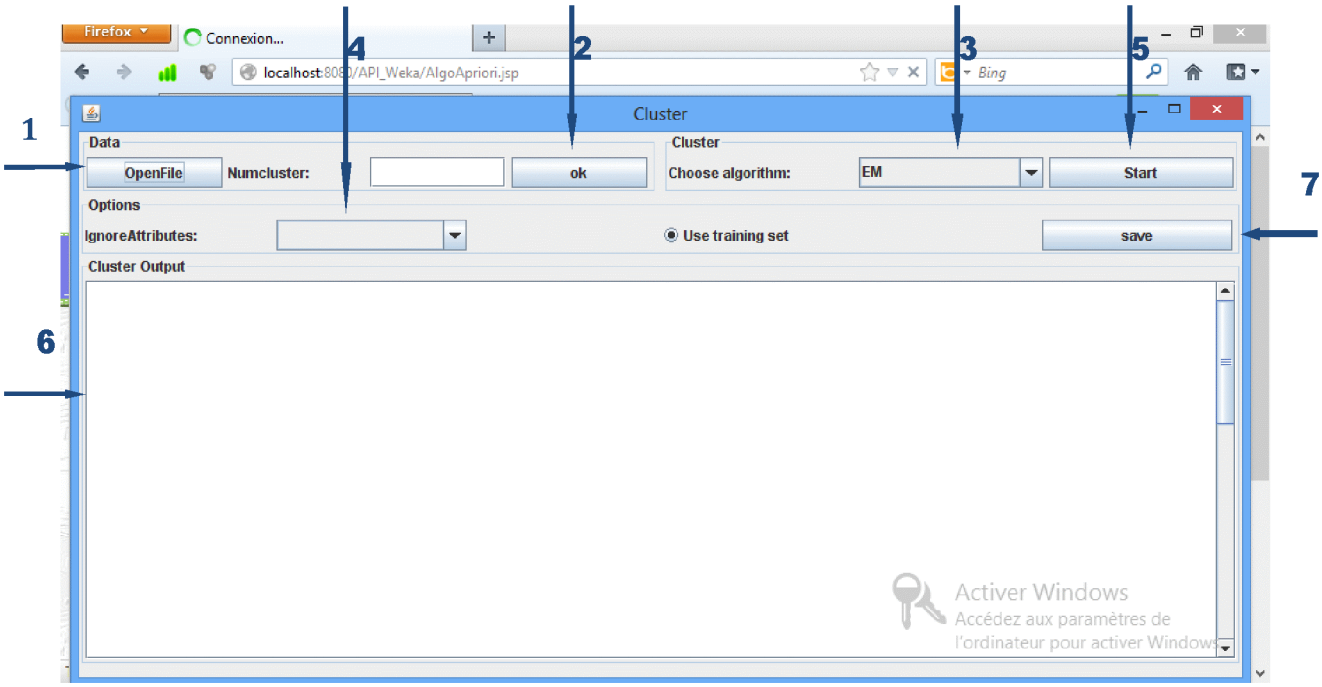


Figure 30 : *Interface de l'algorithme kmeans.*

Les flèches 1, 3, 5, 6 et 7 ont les mêmes fonctionnalités que celles indiqué dans J48.

2 : saisir le nombre de cluster.

4 : ignorer un attribut.

Et enfin pour utiliser l'algorithme Apriori, cliquer sur « ***Règle d'association*** » dans la figure 28, et on aura l'interface suivante :

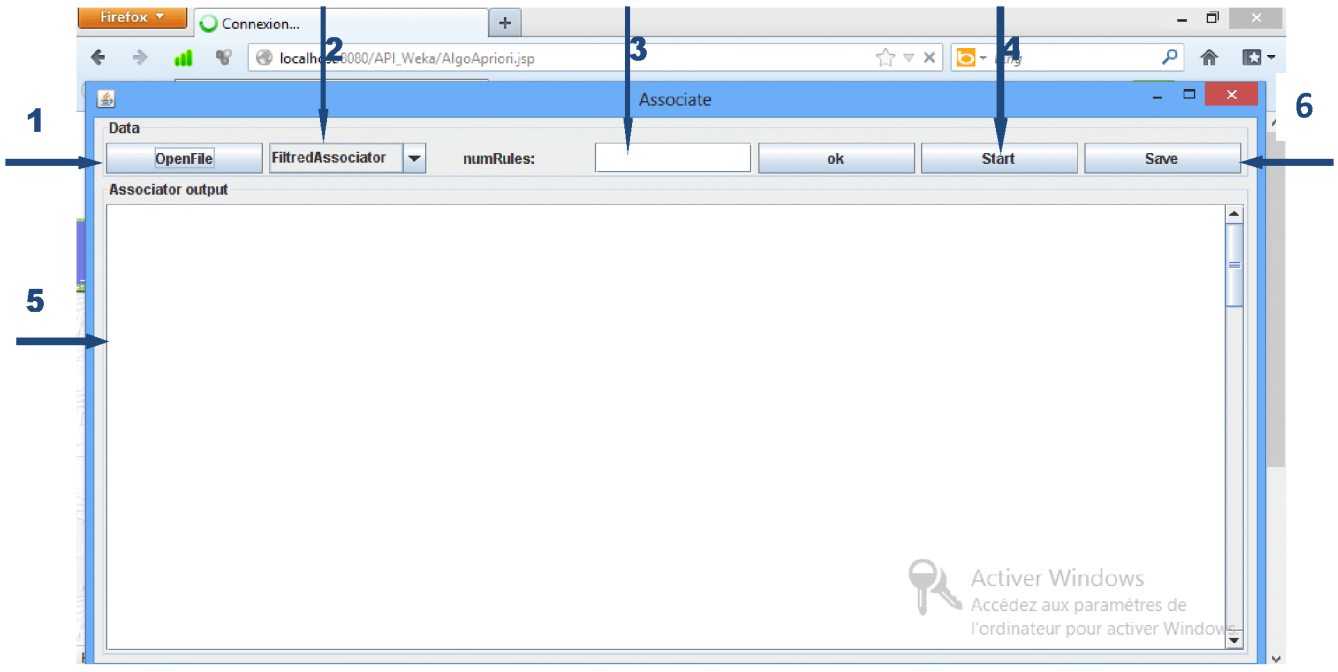


Figure 31 : *Interface de l'algorithme Apriori.*

Les flèches 1, 2, 4, 5 et 6 leurs fonctionnalités sont pareilles que celle de J48.

3 : saisir le nombre de règle à généré.

Partie 2 : Tests

- **L'algorithme J48 :**

Pour le cas de l'algorithme J48, on effectuera le test suivant :

Test1 : Evaluation des apprenants selon le nombre d'erreurs commises, et la catégorie à laquelle ils appartiennent.

La fenêtre ci-dessous montre un exemple d'exécution de l'algorithme J48 :

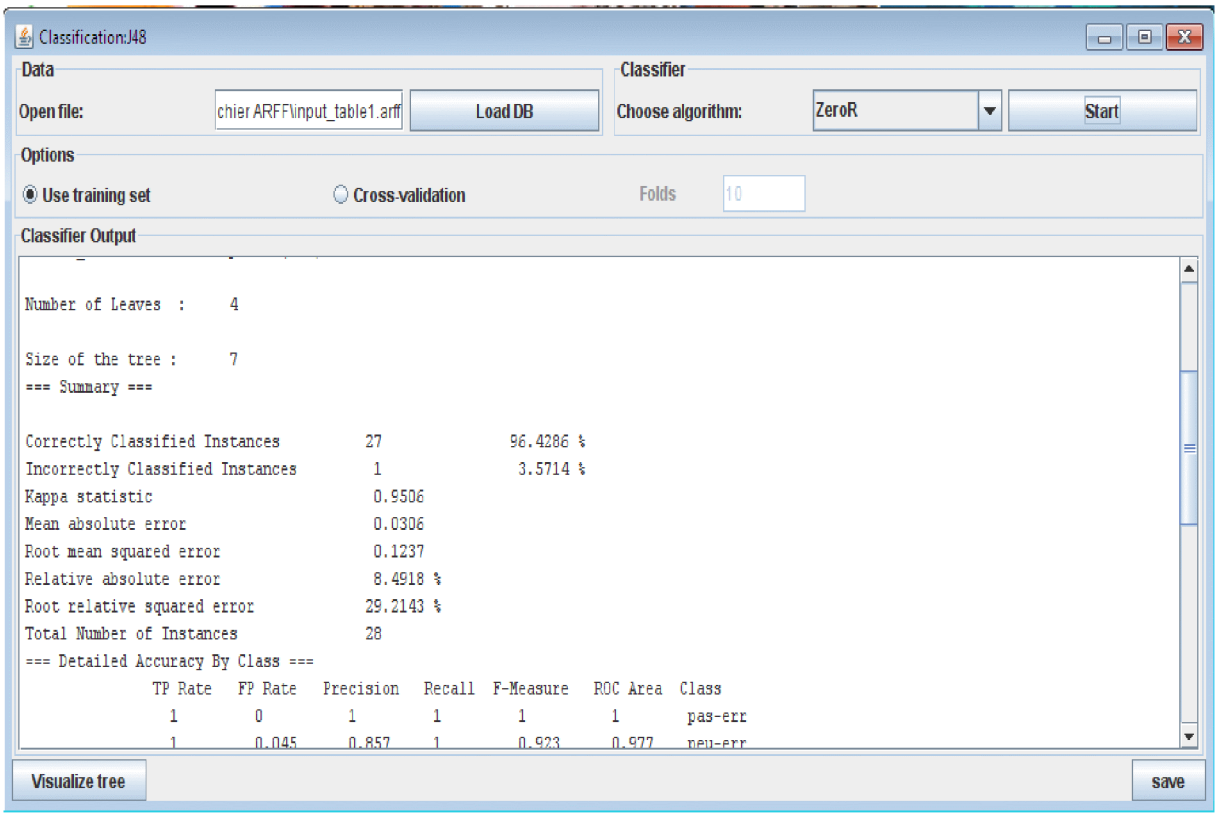


Figure 32 *: Exemple d'exécution de l'algorithme J48.*

Remarque :

On peut classer les apprenants selon quatre catégories :

Pas-err : s'il n'a pas commis d'erreurs.

Peu-err : s'il a commis une ou 4 erreurs.

Pas-mal-err : s'il a commis entre 5 et 10 erreurs.

Beaucoup-err : s'il a commis entre 11 et 15 erreurs.

Pour cela, on prend en entrée le fichier **input_table1** issu de la requête suivante :

```
SELECT apprenant.id_app, SUM(nbre_occ ) AS "nombre_err", categorie
FROM apprenant, session , erreur_eff
WHERE apprenant.id_app = session.id_app
AND session.id_ses = erreur_eff.id_ses
GROUP BY apprenant.id_app
ORDER BY apprenant.id_app ;
```

Fichier input_table1

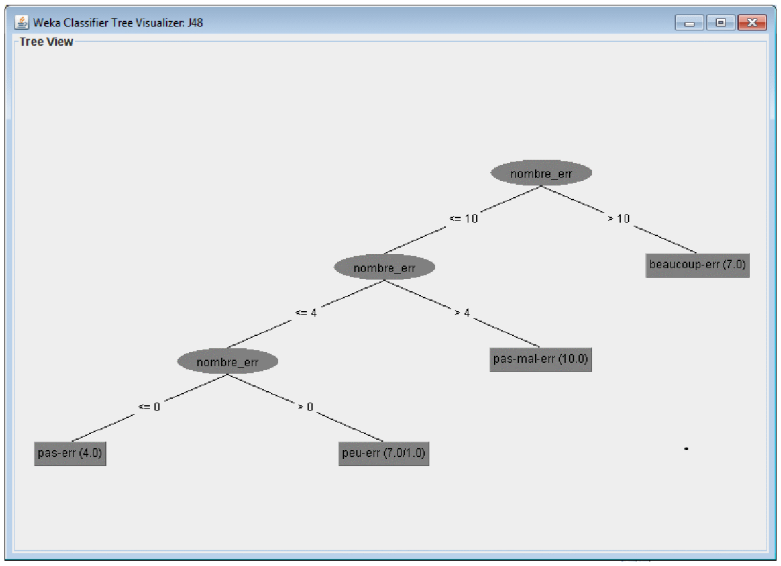
Test 1

```
@RELATION table1
@ATTRIBUTE id_app
NUMERIC
@ATTRIBUTE nombre_err
NUMERIC
@ATTRIBUTE categorie
{pas-err,peu-err,pas-
mal-err,beaucoup-err}

@DATA
3,'1','peu-err',
4,'13','beaucoup-err',
5,'0','pas-err',
6,'0','pas-err',
7,'4','pas-mal-err',
8,'0','pas-err',
9,'6','pas-mal-err',
10,'11','beaucoup-err',
11,'12','beaucoup-err',
12,'11','beaucoup-err',
13,'6','pas-mal-err',
14,'3','peu-err',
15,'2','peu-err',
16,'9','pas-mal-err',
17,'6','pas-mal-err',
18,'8','pas-mal-err',
19,'10','pas-mal-err',
20,'12','beaucoup-err',
21,'11','beaucoup-err',
22,'10','pas-mal-err',
23,'15','beaucoup-err',
24,'0','pas-err',
25,'3','peu-err',
26,'4','peu-err',
27,'4','peu-err',
28,'5','pas-mal-err',
29,'6','pas-mal-err',
30,'10','pas-mal-err',
```

L'algorithme J48

Mode de test : *Use training set*

```
Test mode: evaluate on training data
J48 pruned tree
------------------

nombre_err <= 10
|   nombre_err <= 4
|   |   nombre_err <= 0: pas-err (4.0)
|   |   nombre_err > 0: peu-err (7.0/1.0)
|   nombre_err > 4: pas-mal-err (10.0)
nombre_err > 10: beaucoup-err (7.0)

Number of Leaves  :     4

Size of the tree :      7
=== Summary ===

Correctly Classified Instances          27               96.4286 %
Incorrectly Classified Instances         1                3.5714 %
Kappa statistic                          0.9506
Mean absolute error                      0.0306
Root mean squared error                  0.1237
Relative absolute error                  8.4918 %
Root relative squared error             29.2143 %
Total Number of Instances               28
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1         1         1          1        pas-err
                 1         0.045      0.857     1         0.923      0.977    peu-err
                 0.909     0          1         0.909     0.952      0.984    pas-mal-err
                 1         0          1         1         1          1        beaucoup-err
Weighted Avg.    0.964     0.01       0.969     0.964     0.965      0.989
=== Confusion Matrix ===

      a  b  c  d   <-- classified as
      4  0  0  0 |  a = pas-err
      0  6  0  0 |  b = peu-err
      0  1 10  0 |  c = pas-mal-err
      0  0  0  7 |  d = beaucoup-err
```



Interprétation des résultats :

Test 1 : *Use training set.*

On remarque que :

Le nombre d'instances correctement classifiées est : 27, et la moyenne c'est : 96.4286 %

Le nombre d'instances incorrectement classifiées est : 1, et la moyenne c'est : 3.5714 %

Le coefficient Kappa est : 0,9506.

Le nombre total d'instance est : 28

D'après la matrice de confusion, on trouve que :

4 apprenants n'ont pas faits d'erreurs, 6 ont fait peu d'erreurs, 11 ont fait pas mal d'erreurs, avec un apprenant mal classée qui a fait 4 erreurs, donc, il doit être affecté à la catégorie peu-err. Enfin, on a 7 apprenants on fait beaucoup d'erreurs.

L'arbre de décision, quant à elle, nous permet de déduire les informations suivantes :

Quand :

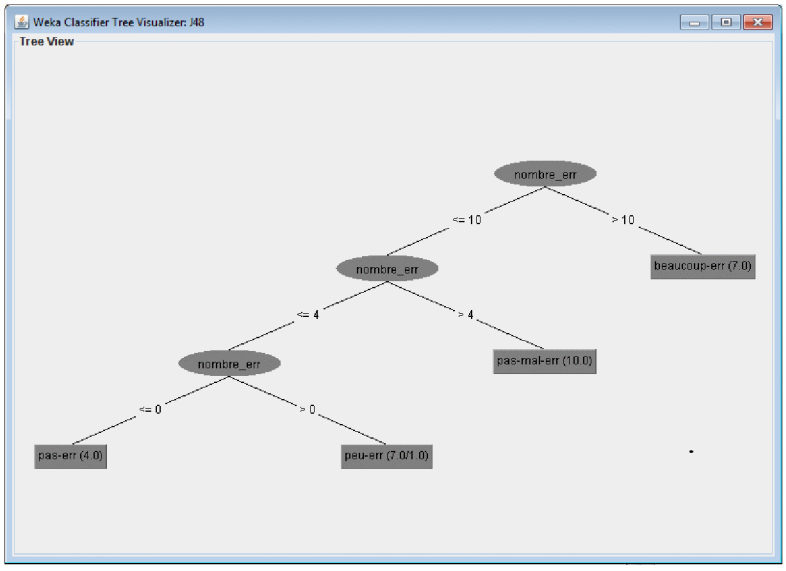
- Nombre-err <= 0 : on est dans la catégorie pas d'erreurs.
- Nombre-err > 0 & <= 4 : on est dans la catégorie peu d'erreurs.
- Nombre-err > 4 & <= 10 : on est dans la catégorie pas-mal-err.
- Nombre-err > 10 : on est dans la catégorie beaucoup-err.

Test 1

```
@RELATION table1
@ATTRIBUTE id_app
NUMERIC
@ATTRIBUTE nombre_err
NUMERIC
@ATTRIBUTE categorie
{pas-err,peu-err,pas-
mal-err,beaucoup-err}

@DATA
3,'1','peu-err',
4,'13','beaucoup-err',
5,'0','pas-err',
6,'0','pas-err',
7,'4','pas-mal-err',
8,'0','pas-err',
9,'6','pas-mal-err',
10,'11','beaucoup-err',
11,'12','beaucoup-err',
12,'11','beaucoup-err',
13,'6','pas-mal-err',
14,'3','peu-err',
15,'2','peu-err',
16,'9','pas-mal-err',
17,'6','pas-mal-err',
18,'8','pas-mal-err',
19,'10','pas-mal-err',
20,'12','beaucoup-err',
21,'11','beaucoup-err',
22,'10','pas-mal-err',
23,'15','beaucoup-err',
24,'0','pas-err',
25,'3','peu-err',
26,'4','peu-err',
27,'4','peu-err',
28,'5','pas-mal-err',
29,'6','pas-mal-err',
30,'10','pas-mal-err',
```

Mode de test : *Cross validation*

```
Test mode: 10-fold cross-validation
J48 pruned tree
------------------

nombre_err <= 10
|   nombre_err <= 4
|   |   nombre_err <= 0: pas-err (4.0)
|   |   nombre_err > 0: peu-err (7.0/1.0)
|   nombre_err > 4: pas-mal-err (10.0)
nombre_err > 10: beaucoup-err (7.0)

Number of Leaves  :     4

Size of the tree :      7
=== Summary ===

Correctly Classified Instances          25               89.2857 %
Incorrectly Classified Instances         3               10.7143 %
Kappa statistic                          0.8492
Mean absolute error                      0.0655
Root mean squared error                  0.2228
Relative absolute error                 17.9314 %
Root relative squared error             51.9484 %
Total Number of Instances               28
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1         1         1          1        pas-err
                 0.667     0.045      0.8       0.667     0.727      0.939    peu-err
                 0.909     0.118      0.833     0.909     0.87       0.928    pas-mal-err
                 1         0          1         1         1          1        beaucoup-err
Weighted Avg.    0.893     0.056      0.892     0.893     0.89       0.959
=== Confusion Matrix ===
      a  b  c  d   <-- classified as
      4  0  0  0 |  a = pas-err
      0  4  2  0 |  b = peu-err
      0  1 10  0 |  c = pas-mal-err
      0  0  0  7 |  d = beaucoup-err
```



Interprétation des résultats :

Test1 : *Cross validation (avec k = 10).*

On remarque que :

Le nombre d'instances correctement classifiées est : 25, et la moyenne c'est : 89.2857 %

Le nombre d'instances incorrectement classifiées est : 3, et la moyenne c'est : 10.7143 %

Le coefficient Kappa est : 0,8492.

Le nombre total d'instance est : 28.

D'après la matrice de confusion, nous avons :

4 apprenants n'ont pas faits d'erreurs, 6 ont fait peu d'erreurs avec 2 d'entre eux mal classés, qui probablement doivent être affectés à la catégorie pas-mal-err, 11 ont fait pas mal d'erreurs, avec un apprenant mal classée qui a fait 4 erreurs, donc, il doit être affecté à la catégorie peu-err. Enfin, on a 7 apprenants on fait beaucoup d'erreurs.

Pour l'arbre de décision, c'est la même chose que l'évaluation en mode *Use training set*.

On peut constater que l'évaluation en mode *use training set* est plus correcte que l'évaluation en mode *cross validation*, car le premier nous donne, sur 28 instances une seule seulement est mal classifié, alors que le deuxième il y'avait 3 instances sur 28 mal classifiées, puisque deux apprenants parmi ceux qui ont fait peu d'erreurs (c.à.d. ils ont commit entre 1 et 4 erreurs), ont été affectés à la catégorie pas-mal-err, donc incorrectement classifiés.

- **L'algorithme KMeans :**

Test 1 : Evaluation des apprenants selon le nombre d'erreurs commises, et la catégorie à laquelle ils appartiennent.

Pour effectué le **test 1**, on prend en entrée le même fichier **input_table1**, que celui utiliser dans l'algorithme J48.

L'interface qui suit illustre un exemple d'exécution de KMeans :

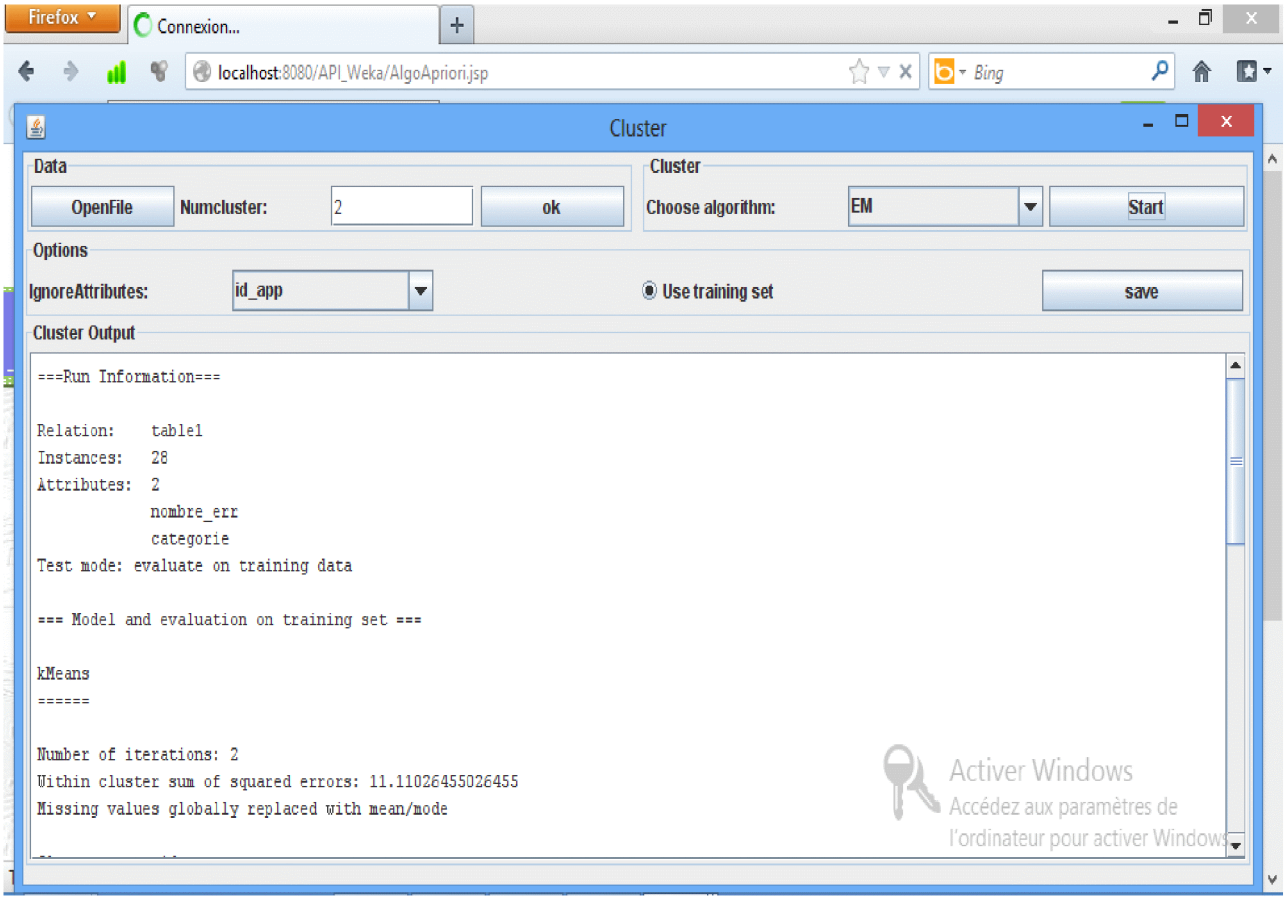


***Figure 33** : Exemple d'exécution de KMeans.*

-La partie « Run Information » : décrit les informations du fichier arff.

Relation : table1 → le nom de la table.

Instances : 28 → le nombre d'instances que contient le fichier.

Attributes : 2

nombre_err
Categorie
} le nombre d'attribut qu'on va tester ainsi que leur nom.

-« kMeans » : on trouve :

Number of iterations : 2 → le nombre d'itérations faites par l'algorithme pour la génération du résultat.

Within cluster sum of squared errors: 11,11 → fonction de coût.

-Cluster centroids : on trouve la moyenne générale des données, ainsi que les partitions k (cluster) et la moyenne affectée à chaque cluster.

Fichier input_table1	**Algorithme KMeans**
Test 1	**Résultat pour k=2.**

```
@RELATION table1
@ATTRIBUTE id_app
NUMERIC
@ATTRIBUTE nombre_err
NUMERIC
@ATTRIBUTE categorie
{pas-err,peu-err,pas-
mal-err,beaucoup-err}

@DATA
3,'1','peu-err',
4,'13','beaucoup-err',
5,'0','pas-err',
6,'0','pas-err',
7,'4','pas-mal-err',
8,'0','pas-err',
9,'6','pas-mal-err',
10,'11','beaucoup-err',
11,'12','beaucoup-err',
12,'11','beaucoup-err',
13,'6','pas-mal-err',
14,'3','peu-err',
15,'2','peu-err',
16,'9','pas-mal-err',
17,'6','pas-mal-err',
18,'8','pas-mal-err',
19,'10','pas-mal-err',
20,'12','beaucoup-err',
21,'11','beaucoup-err',
22,'10','pas-mal-err',
23,'15','beaucoup-err',
24,'0','pas-err',
```

```
===Run Information===

Relation:     table1
Instances:    28
Attributes:   2
              nombre_err
              categorie
Test mode: evaluate on training data

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

kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 11.11026455026455
Missing values globally replaced with mean/mode

Cluster centroids:
                                 Cluster#
Attribute          Full Data            0            1
                        (28)          (7)         (21)
=========================================================
nombre_err               6.5      12.1429        4.619
categorie        pas-mal-err beaucoup-err  pas-mal-err
```

Résultat pour k=4.

```
25,'3','peu-err',
26,'4','peu-err',
27,'4','peu-err',
28,'5','pas-mal-err',
29,'6','pas-mal-err',
30,'10','pas-mal-err',
```

```
===Run Information===

Relation:     table1
Instances:    28
Attributes:   2
              nombre_err
              categorie
Test mode: evaluate on training data

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

kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 4.202920634920636
Missing values globally replaced with mean/mode

Cluster centroids:
                                Cluster#
Attribute          Full Data          0          1          2          3
                        (28)         (7)        (6)       (10)        (5)
=============================================================================
nombre_err               6.5    12.1429        5.5        1.7        9.4
categorie        pas-mal-err beaucoup-err pas-mal-err    peu-err pas-mal-err
```

Résultat pour k=5.

```
===Run Information===

Relation:     table1
Instances:    28
Attributes:   2
              nombre_err
              categorie
Test mode: evaluate on training data

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

kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 0.11729100529100532
Missing values globally replaced with mean/mode

Cluster centroids:
                               Cluster#
Attribute          Full Data          0          1          2          3          4
                        (28)         (7)        (6)        (4)        (5)        (6)
=============================================================================================
nombre_err               6.5    12.1429        5.5          0        9.4     2.8333
categorie        pas-mal-err beaucoup-err pas-mal-err    pas-err pas-mal-err    peu-err
```

Interprétation des résultats :

- **Pour k=2 :**

-Dans la colonne «***Full Data***» l'algorithme calcule la moyenne (moyenne générale) de toutes les instances (28), les informations qui suivent seront pareilles, même si en change le **K**.

nombre_err : **6,5** est la moyenne du nombre d'erreurs commises par les apprenants.

categorie : **pas-mal-err**, puisque la moyenne de nombre_err est **6,5**, cette valeur appartient à l'intervalle [5 - 10] qui représente la classe **pas-mal-err.**

Les colonnes ***Cluster #***, sont interprété comme suit :

Colonne 1 :

0 → le numéro du cluster.

(7) → indique le nombre d'apprenants qui sont affectées au cluster 0.

nombre_err : **12,1429** → la moyenne du nombre d'erreurs, c'est-à-dire la moyenne des valeurs qui se rapproche, qui sont dans l'intervalle [11 - 15].

categorie : **beaucoup-err** → pour la moyenne **12,1429** l'algorithme lui affecte la classe **beaucoup-err**.

En résumé, ***7*** apprenants classé dans le **cluster 0**, ont commit le nombre d'erreurs dont la moyenne est **12,1429**, qui sont classés dans la catégorie **beaucoup-err.**

Colonne 2 :

Le **cluster 1** contient **21** apprenants, qui ont commit le nombre d'erreurs dont la moyenne est **4,619**, qui sont affectés à la catégorie **pas-mal-err**. C'est-à-dire la moyenne de toutes les valeurs comprises entre [0-10].

- **Pour k=4 :**

Il n'y a pas de changement pour le **cluster 0** qui est situé à la **colonne 1.**

Pour le **cluster 1** précédent, l'algorithme a divisé l'intervalle [0 - 10] en trois clusters, qui veut dire que kMeans a cherché de nouveaux centres pour les valeurs qui se rapproche dans cet intervalle.

Dans le **cluster 2** l'algorithme calcule la moyenne du nombre d'erreurs dont les valeurs sont entre [0 - 4] qui sont affecté à la classe **peu-err**.

Cluster 1 : c'est la moyenne des valeurs entre [4 - 6]
Cluster 3: c'est la moyenne des valeurs entre [8 - 10]
} categorie : **pas-mal-err**

On voit bien que la valeur 4 est dans deux intervalles, car l'apprenant est mal classé au lieu de le classé dans catégorie **peu-err**, on la classé dans **pas-mal-err**, alors l'algorithme la ajouté à la moyenne des valeurs dont la catégorie est **pas-mal-err**.

- **Pour k=5 :**

Pour les clusters 0, 1 et 3, il n'y a aucun changement par rapport à **k=4**.

Par contre on remarque que l'algorithme a divisé l'intervalle [0 - 4] en deux clusters, le **cluster 2** qui a **4** apprenants dont le nombre d'erreur est null, et le **cluster 4** qui contient **6** apprenants, dont la moyenne du nombre d'erreurs commises est **2,8333** qui sont affectés à la classe **peu-err**.

Remarque :

A chaque fois qu'on augmente le nombre de cluster **K,** kMeans cherche de nouveaux centres, et calcule la moyenne des valeurs qui se rapproche.

Test 2 : Evaluation des apprenants selon le nombre d'erreurs.

On prend le même fichier que **test 1,** et on ignore les attributs **id_app** et **categorie.**

L'interface suivante montre le résultat de l'algorithme :

```
===Run Information===

Relation:     table1
Instances:    28
Attributes:   1
              nombre_err
Test mode: evaluate on training data

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

kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 0.5411111111111112
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute     Full Data          0          1
                   (28)       (12)       (16)
=============================================
nombre_err          6.5         11      3.125
```

***Figure 34** : Résultat de l'algorithme KMeans (test 2).*

Interprétation du résultat pour k=2:

-**Full Data :** indique la moyenne générale de toutes les erreurs commises.

Cluster 0 : contient **12** instances qui sont dans l'intervalle [8 - 15], dont la moyenne est **11.**

Cluster 1 : contient **16** instances qui sont dans l'intervalle [0 - 7], dont la moyenne est **3,125**.

Remarque :

Dans le **test 1** pour **k=2**, l'algorithme a divisé les valeurs en deux intervalles qui sont [11 –15] pour le **cluster 0** et [0 - 10] pour le **cluster 1,** alors dans le premier cas l'algorithme calcule les moyennes selon l'attribut categorie, tant dit que dans le deuxième cas, il calcule seulement les moyennes des valeurs qui se rapproche**.**

Test 3 : Evaluation des apprenants selon le nombre d'erreurs et le nombre d'exercice.

Le fichier **input_table2** est obtenu avec la requête suivante:

```
SELECT apprenant.id_app, SUM(nbre_occ ) AS "nombre_err", COUNT(id_exo ) AS "nombre_exo"
FROM apprenant, session, erreur_eff
WHERE apprenant.id_app = session.id_app
AND session.id_ses = erreur_eff.id_ses
GROUP BY apprenant.id_app
ORDER BY apprenant.id_app
```

Fichier input_table2	**Algorithme kmeans**
Test 3	***Résultat pour k=2***

```
@RELATION table1
@ATTRIBUTE id_app
NUMERIC
@ATTRIBUTE nombre_err
NUMERIC
@ATTRIBUTE nombre_exo
NUMERIC

@DATA
1,'1','1',
2,'2','1',
3,'1','1',
4,'13','3',
5,'0','1',
6,'0','1',
7,'4','2',
8,'0','1',
9,'6','2',
10,'11','3',
11,'12','3',
12,'11','2',
13,'6','2',
14,'3','1',
15,'2','1',
16,'9','2',
17,'6','2',
18,'8','2',
19,'10','3',
20,'12','3',
21,'11','2',
22,'10','2',
23,'15','3',
24,'0','1',
25,'3','1',
26,'4','1',
27,'4','1',
28,'5','1',
29,'6','2',
30,'10','2',
```

```
===Run Information===

Relation:     table1
Instances:    30
Attributes:   3
              id_app
              nombre_err
              nombre_exo
Test mode: evaluate on training data

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

kMeans
======

Number of iterations: 4
Within cluster sum of squared errors: 4.418531179812392
Missing values globally replaced with mean/mode

Cluster centroids:
                           Cluster#
Attribute     Full Data          0          1
                   (30)       (16)       (14)
=============================================
id_app             15.5     17.125    13.6429
nombre_err       6.1667       9.75     2.0714
nombre_exo       1.7667      2.375     1.0714
```

Interprétation des résultats :

-full data : la moyenne générale de toutes les instances de chaque attribut.

Cluster 0 : 9,75 est la moyenne des erreurs commises, qui correspond à la moyenne **2,375** du nombre d'exercices traiter par **16** apprenants.

Cluster 1 : contient **14** apprenants, dont la moyenne des erreurs commises est **2,0714, 1,0714** est la moyenne du nombre d'exercices, qui sont traités par ces apprenants.

L'algorithme a divisé les apprenant en deux clusters, les apprenants ayant fait 1 exercice dont le nombre d'erreurs commises varies entre [0 - 5], et ceux qui ont traités 2 ou 3 exercices dont le nombre d'erreurs varies entre [6 - 15].

- **L'algorithme Apriori :**

L'interface qui suit illustre un exemple d'exécution de l'algorithme **Apriori** :

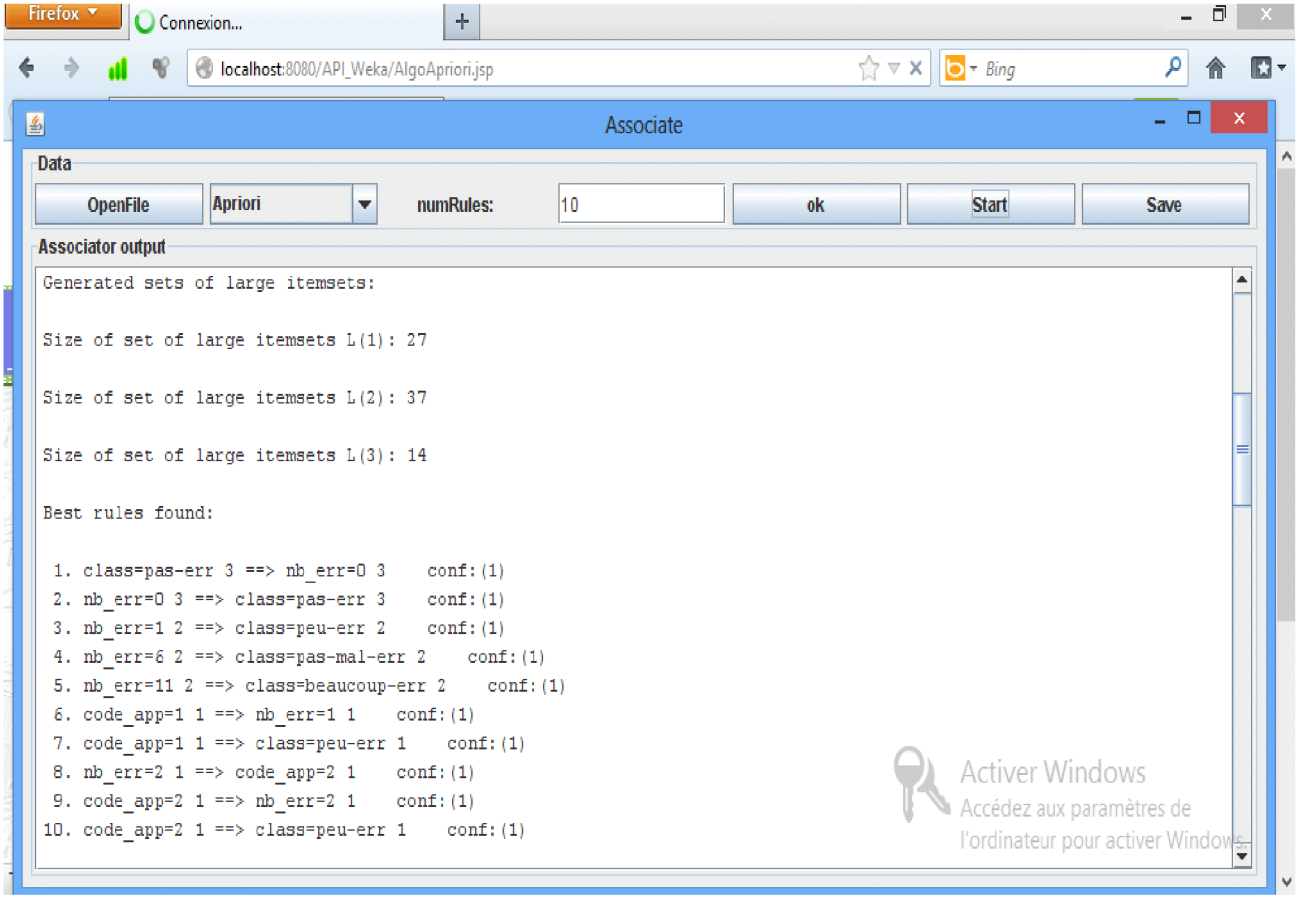


Figure 35 : *Exécution de l'algorithme Apriori.:*

<table>
<tr><th>Fichier input_table3</th><th>Algorithme Apriori</th></tr>
<tr><td>Test 1</td><td>Nombre de règle = 10</td></tr>
<tr><td rowspan="4">

```
@RELATION apprenant

@ATTRIBUTE code_app
{1,2,3,4,5,6,7,8,9,10,1
1,12,13,14}
@ATTRIBUTE nb_err
{1,2,13,0,4,6,11,12,3}
@ATTRIBUTE class  {pas-
err, peu-err, pas-mal-
err, beaucoup-err}

@DATA
1,'1','peu-err',
2,'2','peu-err',
3,'1','peu-err',
4,'13','beaucoup-err',
5,'0','pas-err',
6,'0','pas-err',
7,'4','pas-mal-err',
8,'0','pas-err',
9,'6','pas-mal-err',
10,'11','beaucoup-err',
11,'12','beaucoup-err',
12,'11','beaucoup-err',
13,'6','pas-mal-err',
14,'3','peu-err',
```

</td><td>

```
Best rules found:

 1. class=pas-err 3 ==> nb_err=0 3    conf:(1)
 2. nb_err=0 3 ==> class=pas-err 3    conf:(1)
 3. nb_err=1 2 ==> class=peu-err 2    conf:(1)
 4. nb_err=6 2 ==> class=pas-mal-err 2    conf:(1)
 5. nb_err=11 2 ==> class=beaucoup-err 2    conf:(1)
 6. code_app=1 1 ==> nb_err=1 1    conf:(1)
 7. code_app=1 1 ==> class=peu-err 1    conf:(1)
 8. nb_err=2 1 ==> code_app=2 1    conf:(1)
 9. code_app=2 1 ==> nb_err=2 1    conf:(1)
10. code_app=2 1 ==> class=peu-err 1    conf:(1)
```

</td></tr>
<tr><td>Nombre de règle = 14</td></tr>
<tr><td>

```
 1. class=pas-err 3 ==> nb_err=0 3    conf:(1)
 2. nb_err=0 3 ==> class=pas-err 3    conf:(1)
 3. nb_err=1 2 ==> class=peu-err 2    conf:(1)
 4. nb_err=6 2 ==> class=pas-mal-err 2    conf:(1)
 5. nb_err=11 2 ==> class=beaucoup-err 2    conf:(1)
 6. code_app=1 1 ==> nb_err=1 1    conf:(1)
 7. code_app=1 1 ==> class=peu-err 1    conf:(1)
 8. nb_err=2 1 ==> code_app=2 1    conf:(1)
 9. code_app=2 1 ==> nb_err=2 1    conf:(1)
10. code_app=2 1 ==> class=peu-err 1    conf:(1)
11. code_app=3 1 ==> nb_err=1 1    conf:(1)
12. code_app=3 1 ==> class=peu-err 1    conf:(1)
13. nb_err=13 1 ==> code_app=4 1    conf:(1)
14. code_app=4 1 ==> nb_err=13 1    conf:(1)
```

</td></tr>
<tr><td>Nombre de règle = 28</td></tr>
<tr><td>

```
15. code_app=4 1 ==> class=beaucoup-err 1    conf:(1)
16. code_app=5 1 ==> nb_err=0 1    conf:(1)
17. code_app=5 1 ==> class=pas-err 1    conf:(1)
18. code_app=6 1 ==> nb_err=0 1    conf:(1)
19. code_app=6 1 ==> class=pas-err 1    conf:(1)
20. nb_err=4 1 ==> code_app=7 1    conf:(1)
21. code_app=7 1 ==> nb_err=4 1    conf:(1)
22. code_app=7 1 ==> class=pas-mal-err 1    conf:(1)
23. code_app=8 1 ==> nb_err=0 1    conf:(1)
24. code_app=8 1 ==> class=pas-err 1    conf:(1)
25. code_app=9 1 ==> nb_err=6 1    conf:(1)
26. code_app=9 1 ==> class=pas-mal-err 1    conf:(1)
27. code_app=10 1 ==> nb_err=11 1    conf:(1)
28. code_app=10 1 ==> class=beaucoup-err 1    conf:(1)
```

</td></tr>
<tr><td>Nombre de règle = 44</td></tr>
</table>

```
29. nb_err=12 1 ==> code_app=11 1    conf:(1)
30. code_app=11 1 ==> nb_err=12 1    conf:(1)
31. code_app=11 1 ==> class=beaucoup-err 1    conf:(1)
32. code_app=12 1 ==> nb_err=11 1    conf:(1)
33. code_app=12 1 ==> class=beaucoup-err 1    conf:(1)
34. code_app=13 1 ==> nb_err=6 1    conf:(1)
35. code_app=13 1 ==> class=pas-mal-err 1    conf:(1)
36. nb_err=3 1 ==> code_app=14 1    conf:(1)
37. code_app=14 1 ==> nb_err=3 1    conf:(1)
38. code_app=14 1 ==> class=peu-err 1    conf:(1)
39. nb_err=2 1 ==> class=peu-err 1    conf:(1)
40. nb_err=13 1 ==> class=beaucoup-err 1    conf:(1)
41. nb_err=4 1 ==> class=pas-mal-err 1    conf:(1)
42. nb_err=12 1 ==> class=beaucoup-err 1    conf:(1)
43. nb_err=3 1 ==> class=peu-err 1    conf:(1)
44. code_app=1 class=peu-err 1 ==> nb_err=1 1    conf:(1)
```

Nombre de règle = 100

```
45. code_app=1 nb_err=1 1 ==> class=peu-err 1    conf:(1)
46. code_app=1 1 ==> nb_err=1 class=peu-err 1    conf:(1)
47. nb_err=2 class=peu-err 1 ==> code_app=2 1    conf:(1)
48. code_app=2 class=peu-err 1 ==> nb_err=2 1    conf:(1)
49. code_app=2 nb_err=2 1 ==> class=peu-err 1    conf:(1)
50. nb_err=2 1 ==> code_app=2 class=peu-err 1    conf:(1)
51. code_app=2 1 ==> nb_err=2 class=peu-err 1    conf:(1)
52. code_app=3 class=peu-err 1 ==> nb_err=1 1    conf:(1)
53. code_app=3 nb_err=1 1 ==> class=peu-err 1    conf:(1)
54. code_app=3 1 ==> nb_err=1 class=peu-err 1    conf:(1)
55. nb_err=13 class=beaucoup-err 1 ==> code_app=4 1    conf:(1)
56. code_app=4 class=beaucoup-err 1 ==> nb_err=13 1    conf:(1)
57. code_app=4 nb_err=13 1 ==> class=beaucoup-err 1    conf:(1)
58. nb_err=13 1 ==> code_app=4 class=beaucoup-err 1    conf:(1)
59. code_app=4 1 ==> nb_err=13 class=beaucoup-err 1    conf:(1)
60. code_app=5 class=pas-err 1 ==> nb_err=0 1    conf:(1)
61. code_app=5 nb_err=0 1 ==> class=pas-err 1    conf:(1)
62. code_app=5 1 ==> nb_err=0 class=pas-err 1    conf:(1)
63. code_app=6 class=pas-err 1 ==> nb_err=0 1    conf:(1)
64. code_app=6 nb_err=0 1 ==> class=pas-err 1    conf:(1)
65. code_app=6 1 ==> nb_err=0 class=pas-err 1    conf:(1)
66. nb_err=4 class=pas-mal-err 1 ==> code_app=7 1    conf:(1)
67. code_app=7 class=pas-mal-err 1 ==> nb_err=4 1    conf:(1)
68. code_app=7 nb_err=4 1 ==> class=pas-mal-err 1    conf:(1)
69. nb_err=4 1 ==> code_app=7 class=pas-mal-err 1    conf:(1)
70. code_app=7 1 ==> nb_err=4 class=pas-mal-err 1    conf:(1)
71. code_app=8 class=pas-err 1 ==> nb_err=0 1    conf:(1)
72. code_app=8 nb_err=0 1 ==> class=pas-err 1    conf:(1)
73. code_app=8 1 ==> nb_err=0 class=pas-err 1    conf:(1)
74. code_app=9 class=pas-mal-err 1 ==> nb_err=6 1    conf:(1)
75. code_app=9 nb_err=6 1 ==> class=pas-mal-err 1    conf:(1)
76. code_app=9 1 ==> nb_err=6 class=pas-mal-err 1    conf:(1)
77. code_app=10 class=beaucoup-err 1 ==> nb_err=11 1    conf:(1)
78. code_app=10 nb_err=11 1 ==> class=beaucoup-err 1    conf:(1)
79. code_app=10 1 ==> nb_err=11 class=beaucoup-err 1    conf:(1)
80. nb_err=12 class=beaucoup-err 1 ==> code_app=11 1    conf:(1)
81. code_app=11 class=beaucoup-err 1 ==> nb_err=12 1    conf:(1)
```

```
82. code_app=11 nb_err=12 1 ==> class=beaucoup-err 1    conf:(1)
83. nb_err=12 1 ==> code_app=11 class=beaucoup-err 1    conf:(1)
84. code_app=11 1 ==> nb_err=12 class=beaucoup-err 1    conf:(1)
85. code_app=12 class=beaucoup-err 1 ==> nb_err=11 1    conf:(1)
86. code_app=12 nb_err=11 1 ==> class=beaucoup-err 1    conf:(1)
87. code_app=12 1 ==> nb_err=11 class=beaucoup-err 1    conf:(1)
88. code_app=13 class=pas-mal-err 1 ==> nb_err=6 1    conf:(1)
89. code_app=13 nb_err=6 1 ==> class=pas-mal-err 1    conf:(1)
90. code_app=13 1 ==> nb_err=6 class=pas-mal-err 1    conf:(1)
91. nb_err=3 class=peu-err 1 ==> code_app=14 1    conf:(1)
92. code_app=14 class=peu-err 1 ==> nb_err=3 1    conf:(1)
93. code_app=14 nb_err=3 1 ==> class=peu-err 1    conf:(1)
94. nb_err=3 1 ==> code_app=14 class=peu-err 1    conf:(1)
95. code_app=14 1 ==> nb_err=3 class=peu-err 1    conf:(1)
```

Interprétation des Résultats :

Pour le nombre de règle=10 :

Le terme **conf (la confiance d'une règle)** : si condition alors conclusion est le rapport : Nombre de données ou les items de la condition et de la conclusion apparaissent simultanément / nombre de données ou les items de la condition apparaissent simultanément.

La règle **1** : on 3 classes **pas_err** se qui implique **nb_err=0**. Et la règle 2 est la réciproque de la règle 1.

La règle **3** : il existe 2 valeurs pour le **nb_err=2**, qui sont classé dans **peu-err.**

La règle **4** : il y a 2 valeurs pour le **nb_err=6**, qui sont classé dans **pas-mal-err.**

La règle **5** : il y a 2 valeurs pour le **nb_err=11**, qui sont classé dans **beaucoup-err.**

Les règles **6** et **7 :** le **code_app=1,** il a commit **nb_err=1** qui est affecté à la classe **peu_err.**

Les règles **8**, **9** et **10** : pour le **nb_err=2** sont commises par l'apprenant dans le **code_app=2,** qui est classé dan **peu-err.**

Remarque :

Les règles générées ne prennent en considération que deux attributs parmi les trois (un représente la condition, et l'autre représente la conclusion), et on voit bien qu'il reste beaucoup d'instances (les codes apprenants allant de 3 jusqu'à 14) qui ne sont pas traitées.

Pour le nombre de règle=14 :

Les nouvelles règles générées sont **11**, **12, 13** et **14,** pour les premières sont les mêmes.

Les règles **11** et **12** signifient, que le **code_app=3** a commit 1 erreur (**nb_err=1**), et il est classé dans **peu_err**.

La règle **13** : **nb_err=13,** ces erreurs sont faites par l'apprenant dont le **cod_app=4.** La règle 14 est la réciproque de 13.

Remarque :

La encore, il reste des règles qui ne sont pas traitées. Même si on a donnée 14 règles qui est égale au nombre d'instances du fichier.

Pour le nombre de règle=28 :

La règle **15** signifie que l'apprenant **4** est affecté à la classe **beaucoup-err**.

Les règles **16**, **17, 18 et 19 :** signifient que les apprenants **5** et **6** n'ont pas commit d'erreur, ils sont classés dans **pas-err**.

La règle **20** : on a l'apprenant **7**, qui a commit **4** erreurs.

Les règles **21** et **22** : signifient que l'apprenant **7** a commit **4** erreurs, donc il est classé dans **pas-mal-err**.

Les règles **23 et 24 :** signifient que l'apprenant **8** n'a pas commit d'erreur, il est classé dans **pas-err**.

Les règles **25** et **26** : signifient que l'apprenant **9** a commit **6** erreurs, donc il est classé dans **pas-mal-err**.

Les règles **27** et **28** : signifient que l'apprenant **10** a commit **11** erreurs, donc il est classé dans **beaucoup-err**.

Remarque :

Même avec le nombre de règles = 28, il reste des instances (les codes apprenants 11, 12, 13 et 14) qui ne sont pas encore traitées.

Pour arriver à générer toutes les règles pour toutes les instances on a augmenté le nombre de règles à fur et à mesure jusqu'à 44.

Pour le nombre de règle = 44 :

La règle **44** signifie que l'apprenant **1** qui est affecté à la classe **peu-err**, a commis une **1** erreur.

Remarque :

La, toutes les instances sont épuisées, mais seulement avec des règles qui prennent deux attributs, puisque, le processus recommence au niveau de la règle 44 avec des règles qui prennent en considération les trois attributs.

Pour le nombre de règle = 100 :

Dans ce cas, il donne seulement 95 règles, donc la génération de règles est terminée, parce qu'il n'y a pas d'autres instances et attributs à prendre en charge.

Pour confirmé que la génération de règles est terminée, on a effectué un test avec le **nombre de règles = 120**, et le résultat est toujours 95 règles.

Conclusion :

D'après ce test, on peut conclure qu'Apriori prend au début deux attributs pour générer les règles jusqu'à épuisement de toutes ces dernières, puis il recommence le même processus en ajoutant un autre attribut et ainsi de suite jusqu'à la prise en charge de tous les attributs.

<table>
<tr><th>Fichier input_table3</th><th>Algorithme Apriori</th></tr>
<tr><td>Test 2</td><td>Nombre de règle = 10</td></tr>
<tr><td>

```
@RELATION apprenant

@ATTRIBUTE code_app
{1,2,3,4,5,6,7,8,9,10,11,1
2,13,14,15,16}
@ATTRIBUTE nb_err
{1,2,13,0,4,6,11,12,3}
@ATTRIBUTE class  {pas-
err, peu-err, pas-mal-err,
beaucoup-err}

@DATA
1,'1','peu-err',
2,'2','peu-err',
3,'1','peu-err',
4,'13','beaucoup-err',
5,'0','pas-err',
6,'0','pas-err',
7,'4','pas-mal-err',
8,'0','pas-err',
9,'6','pas-mal-err',
10,'11','beaucoup-err',
11,'12','beaucoup-err',
12,'11','beaucoup-err',
13,'6','pas-mal-err',
14,'3','peu-err',
15,'13','beaucoup-err',
16,'13','beaucoup-err'
```

</td><td>

```
Best rules found:

 1. nb_err=13 3 ==> class=beaucoup-err 3    conf:(1)
 2. class=pas-err 3 ==> nb_err=0 3    conf:(1)
 3. nb_err=0 3 ==> class=pas-err 3    conf:(1)
 4. nb_err=1 2 ==> class=peu-err 2    conf:(1)
 5. nb_err=6 2 ==> class=pas-mal-err 2    conf:(1)
 6. nb_err=11 2 ==> class=beaucoup-err 2    conf:(1)
```

</td></tr>
<tr><td>Test 3</td><td>Nombre de règle = 10</td></tr>
<tr><td>

```
@RELATION apprenant

@ATTRIBUTE code_app
{1,2,3,4,5,6,7,8,9,10,11,1
2,13,14,15,16,17,18,19,20}
@ATTRIBUTE nb_err
{1,2,13,0,4,6,11,12,3}
@ATTRIBUTE class  {pas-
err, peu-err, pas-mal-err,
beaucoup-err}

@DATA
1,'1','peu-err',
2,'2','peu-err',
3,'1','peu-err',
4,'13','beaucoup-err',
5,'0','pas-err',
6,'0','pas-err',
7,'4','pas-mal-err',
8,'0','pas-err',
9,'6','pas-mal-err',
10,'11','beaucoup-err',
11,'12','beaucoup-err',
12,'11','beaucoup-err',
```

</td><td>

```
Best rules found:

 1. nb_err=13 5 ==> class=beaucoup-err 5    conf:(1)
 2. class=pas-err 4 ==> nb_err=0 4    conf:(1)
 3. nb_err=0 4 ==> class=pas-err 4    conf:(1)
 4. nb_err=6 3 ==> class=pas-mal-err 3    conf:(1)
 5. nb_err=1 2 ==> class=peu-err 2    conf:(1)
 6. nb_err=11 2 ==> class=beaucoup-err 2    conf:(1)
```

</td></tr>
</table>

13,'6','pas-mal-err', 14,'3','peu-err', 15, '13','beaucoup-err', 16,'13','beaucoup-err' 17,'13','beaucoup-err' 18,'0','pas-err' 19,'6','pas-mal-err' 20,'13','beaucoup-err'	
Test 4	**Nombre de règle = 10**
@RELATION apprenant @ATTRIBUTE code_app {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20} @ATTRIBUTE nb_err {1,2,13,0,4,6,11,12,3} @ATTRIBUTE class {pas-err, peu-err, pas-mal-err, beaucoup-err} @DATA 1,'1','peu-err', 2,'2','peu-err', 3,'1','peu-err', 4,'13','beaucoup-err', 5,'0','pas-err', 6,'0','pas-err', 7,'4','pas-mal-err', 8,'0','pas-err', 9,'6','pas-mal-err', 10,'11','beaucoup-err', 11,'12','beaucoup-err', 12,'11','beaucoup-err', 13,'6','pas-mal-err', 14,'3','peu-err', 15, '13','beaucoup-err', 16,'13','beaucoup-err' 17,'1','peu-err' 18,'12','beaucoup-err' 19,'12','beaucoup-err' 20,'1','peu-err'	Best rules found: 1. nb_err=1 4 ==> class=peu-err 4 conf:(1) 2. nb_err=13 3 ==> class=beaucoup-err 3 conf:(1) 3. class=pas-err 3 ==> nb_err=0 3 conf:(1) 4. nb_err=0 3 ==> class=pas-err 3 conf:(1) 5. nb_err=12 3 ==> class=beaucoup-err 3 conf:(1) 6. nb_err=6 2 ==> class=pas-mal-err 2 conf:(1) 7. nb_err=11 2 ==> class=beaucoup-err 2 conf:(1)

Interprétation des résultats :

On prend le même fichier que **Test 1**, mais on ajoute deux instances (15,16).

Comparaison avec Test1 :

On remarque que dans **Test1**, quand on a générer les 10 règles, le **nb-err =13** n'a pas encore été traité, mais dans **Test2**, après avoir ajouté les deux instances (15, 16), qui représentent deux occurrences du **nb-err=13**, on voit bien que **nb-err=13** a été traité et figure en premier avec nombre occurrence=3. Autre chose qu'on peut déduire, c'est qu'au niveau de ce test, le nombre de règles générées est 6, et d'après le test 3 ou on a augmenté le nombre

d'instances jusqu'à 20, l'algorithme a générée seulement 6 règles, puis on a effectué le test 4 qui consiste a changer les 4 dernières instances, de manière a augmenté le nombre d'occurrence de **nb-err=12** et dans ce cas, l'algorithme a généré 7 règles.

Conclusion :

On peut conclure que :

- L'ordre d'apparition des règles dépend du nombre d'occurrence des instances. Cependant, Apriori affiche les règles selon l'ordre décroissant du nombre d'occurrence des instances.
- Le nombre de règles générées, ne dépend pas du nombre d'instances (la taille du fichier) mais du contenu des instances.

Problématique : vu les inconvénients de l'algorithme Apriori, on peut poser le problème qui consiste à la recherche de filtre pour les règles générées par ce dernier, ainsi éliminer les règles redondantes inutile, et d'exploiter toutes les instances de la table pour générer toutes les règles d'association possible, qui seront intéressantes dans les cadre de l'évaluation des apprenants.

Conclusion :

Dans ce chapitre on s'est intéressé à l'implémentation de trois algorithmes de data mining (J48, KMeans et Apriori), comme on a effectué quelques tests sur ces derniers, pour mettre en évidences leur intérêts dans l'évaluation des apprenants.

Conclusion générale

Notre objectif était d'utiliser les outils du data mining dans le cadre de l'analyse des résultats d'évaluation des apprenants par extraction des connaissances utiles pour la suite de l'apprentissage. De ce fait, notre travail va consister, a intégrer un module de preprocessing, et un module d'application des algorithmes de data mining, encapsulé dans un service web et puisant ses données à partir du système WebSeila dédié pour l'auto-apprentissage de algorithmique aux débutants.

Au terme de notre travail nous avons pu :

- Enrichir la version Websiela4 par l'amélioration de l'option *data mining*.
- Implémenter un module de *preprocessing* (Sélection des données, calcul algébrique)
- Implémenter un module de conversion en ARFF (transformation du format SQL au format ARFF).
- Utiliser l'API Weka via l'environnement java.
- Proposer quelques tests en utilisant les algorithmes (J48, KMeans et Apriori), pour mieux voir leurs utilités.

En guise de perspectives, nous proposons :

- L'interprétation automatique notamment un filtre des résultats de l'algorithme Apriori, pour générer que les règles utiles, et d'éliminer les règles redondantes.

Bibliographie

[1] PHAM THI NGOC Diem : « *Spécification et conception de services d'analyse de l'utilisation d'un environnement informatique pour l'apprentissage humain* » Doctorat de l'Université du Maine, Novembre 2011

[2] MENOUER Tarek & DERMOUCHE Mohamed : « *Application de techniques de data mining pour la classification automatique des données et la recherche d'associations* » Mémoire de fin d'études ESi, 2009/2010

[3] Medjnoun Chabha & Messad Djamila : « *Application des outils du datamining pour l'analyse des erreurs des apprenants en Learning by doing* » Mémoire de fin d'étude (Master), 2011/2012

[4] Mr.RAHMANI Rabah : « *Découverte d'associations sémantiques dans les bases de données relationnelles par des méthodes de Data Mining* » Mémoire de fin d'étude (Magister)

[10] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth et R. Uthurusamy (Eds.) : « *Advances in Knowledge Discovery and Data Mining* » MIT Press, 1996

[18] Jean-Michel DOUDOUX : *« Développons en Java »*

[19] Roger Engel. Patrice Silverio : « *Application Cobol sur le Web, avec une étude de cas détaillée* ». Edition Eyrolles, 2000.

[21] BOUZIDI EL Hocine & SADKI Omar : « *Conception et implémentation d'un entrepôt de données et application des techniques du Data Mining pour l'analyse des échecs/succès des étudiants de l'ESI* » Mémoire de fin d'études ESi, 2009/2010

Webographie

[5] http://bliaudet.free.fr/IMG/pdf/Cours_de_data_mining_1-Introduction-EPF.pdf

[6] http://fr.wikiversity.org/wiki/Datamining/Qu27est_ce_que_le_Datamining_%3F

[7] http://www.educationaldatamining.org/

[8] http://www.findthatpdf.com/search-34612955-fPDF/download-documents imw14303frfr.pdf.htm

[9] http://www.educationaldatamining.org/JEDM/images/articles/vol1/issue1/ JEDMVol1Issue1_BakerYacef.pdf

[11] http://eric.univ-lyon2.fr ~ricco/cours/slides/IntroDMDraft2002.pdf

[12] http://depot-e.uqtr.ca/1423/1/030000495.pdf

[13] http://www.educationaldatamining.org/

[14] http://www2.lifl.fr/~talbi/Cours-Data-Mining.pdf

[15] http://www.asso-aria.org/coria/2007/137.pdf

[16] http://www.ccim.be/ccim328/js/index.htm

[17] http://docs.oracle.com/cd/E19879-01/821-1041/relnotessges/index.html

[20] http: //morgon.univ-lyon2.fr/Introduction_au_datamining_cours.htm

[22] http://fr.wikipedia.org/wiki/WAR_(format_de_fichier)