

Mémoire de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'Etat en Informatique

Option : Systèmes informatiques

Thème

**Application de techniques de data mining pour la classification
automatique des données et la recherche d'associations**

Réalisé par

- **MENOUER Tarek**
- **DERMOUCHE Mohamed**

Encadré par

- **M^{lle} K. BENATCHBA**
- **M^{er} F. DAHAMNI**

Promotion : 2009 / 2010

Dédicaces

A mes parents qui m'ont donné le jour.

A toi **ma mère**, dont l'amour et le soutien à chaque moment de ma vie m'ont donné courage, force et m'ont permis d'arriver à ce niveau. Mille fois merci pour tous tes sacrifices. Les mots sont très faibles pour te l'affirmer : tu es et resteras la personne la plus importante pour moi.

A toi **mon père** qui a guidé mes pas. Merci pour tes conseils, ta présence constante et toute l'attention que tu as toujours portée à mon évolution. Mille fois merci pour ce grand amour que tu as pour nous, tes enfants, dont je suis heureux et fier de faire partie.

Mohamed.

A mes très chers parents qui ont toujours été là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour.

Merci **maman**, Merci **papa**.

Tarek.

Remerciements

Cette œuvre a vu le jour grâce au concours de plusieurs personnes. Je saisis cette occasion pour exprimer mes sincères remerciements à tous ceux qui, de près comme de loin, ont contribué à la mise en œuvre de ce mémoire.

Je remercie mes promoteurs, **M^{lle} BENATCHBA** et **M^{er} DAHAMNI**, qui ont bien voulu sacrifier leurs préoccupations en ne ménageant aucun effort pour suivre ce travail.

Je remercie **M^{lle}. BOUSBIA** pour l'attention et tous les conseils édifiants qu'elle nous a prodigués tout au long de ce travail.

Je remercie **les membres du jury** qui ont bien voulu sanctionner ce travail.

Merci à **M^{er} BERRACHEDI Abdelkrim** pour toute l'aide qu'il a consentie pour le bon déroulement de ma formation.

Je dis merci à toutes mes frères et sœurs **Samia, Amel, Said, Ahmed, Hayet** et à tous les membres de ma famille, notamment ma grand-mère **Mama** et mon oncle **Athmane**.

Dans le même ordre d'idées, je remercie **tous mes amis** et ceux dont les noms ne figurent pas dans cette page, mais qui ont tous constitué pour moi un soutien certain, je pense notamment à **Soumia, Amine, Tarek** et **Abdeldjalil**.

Mohamed.

En premier lieu, je veux remercier **M^{lle} BENATCHBA** qui nous a encadrés durant cette année, pour ses conseils, son dévouement constant et son entière disponibilité.

Je tiens aussi à remercier **M^{er} DAHAMNI**, notre encadreur, pour son aide et ces remarques précieuses.

Mes vifs remerciements vont à tous les membres du Laboratoire des Méthodes de Conception des Systèmes (LMCS) pour leur accueil et leur sympathie.

J'exprime ma profonde gratitude à l'ensemble des enseignants de l'ESI. En particulier **M^{er} AIT AOUDIA** et tout le personnel de l'ESI.

Que les membres du jury trouvent ici le témoignage de notre reconnaissance pour avoir bien voulu juger notre travail.

A mes chers frères et sœurs : **Ouassila, Anissa, Merouane, Meriem** et **Mouni**.

Je remercie **toutes les personnes** qui ont constitué, pour moi, un soutien certain, spécialement **Kenza** et notamment **Mohamed** et **Abdeldjalil**.

Tarek.

Table des matières

Résumé	VIII
Liste des figures	X
Liste des tableaux	XII
Liste des algorithmes	XIII
 Introduction générale.....	 1
 Chapitre I : Concepts et généralités sur le data mining et le processus ECD	 3
1. Introduction	4
2. Le processus ECD	4
2.1. Définition	4
2.2. Etapes du processus ECD	4
3. Le data mining.....	8
3.1. Définition	8
3.2. Principe.....	8
3.3. Tâches de data mining	8
3.4. Techniques de data mining	9
3.5. Domaines d'application	10
3.6. Difficultés de data mining	10
4. Conclusion	11
 Chapitre II : Etat de l'art sur les techniques de data mining.....	 12
1. Introduction	13
2. La classification automatique non supervisée.....	13
2.1. Introduction	13
2.2. Définition	14
2.3. Terminologie	14
2.4. Les algorithmes de clustering.....	17
2.4.1. Les algorithmes de partitionnement.....	17
2.4.1.1. L'algorithme k-Means.....	17
2.4.1.2. L'algorithme k-Medoids	19
2.4.1.3. L'algorithme PAM.....	20
2.4.1.4. L'algorithme CLARA.....	20
2.4.1.5. L'algorithme CLARANS.....	20

2.4.2.	Les algorithmes hiérarchiques.....	21
2.4.2.1.	L'algorithme BIRCH	22
2.4.2.2.	L'algorithme DIANA.....	22
2.4.3.	Les algorithmes basés sur la densité	23
2.4.3.1.	L'algorithme DBSCAN.....	24
2.4.3.2.	L'algorithme OPTICS	25
2.4.4.	Autres algorithmes	25
2.4.4.1.	Les algorithmes basés sur la grille	25
2.4.4.2.	Les algorithmes basés sur les statistiques	25
2.5.	Conclusion	26
3.	La classification automatique supervisée	26
3.1.	Introduction	26
3.2.	Problématique	27
3.3.	Les algorithmes de classification	27
3.3.1.	Les arbres de décision	27
3.3.2.	L'algorithme k-NN	29
3.3.3.	Les réseaux de neurones.....	30
3.3.4.	Autres méthodes de classification	31
3.3.4.1.	Les réseaux bayésiens.....	31
3.4.	Conclusion	32
4.	Les règles d'association	32
4.1.	Introduction	32
4.2.	Problématique	33
4.3.	Définitions	33
4.4.	Les algorithmes de recherche d'associations	34
4.4.1.	L'algorithme Apriori	34
4.4.2.	L'algorithme FP-Growth	35
4.5.	Conclusion	35
5.	Conclusion	36
Chapitre III : Conception et mise en Œuvre		37
1.	Introduction	38
2.	Architecture générale de l'application.....	38
2.1.	Module de lecture.....	38
2.1.1.	Accès aux données.....	39
2.1.2.	Nettoyage des données	39

2.2.	Module de calcul	39
2.3.	Module statistique	40
2.3.1.	Distance de Minkowski	40
2.3.2.	Distance entre deux classes.....	41
2.3.3.	Inertie	41
2.3.4.	Erreur de classification	43
2.4.	Module d'affichage	43
2.4.1.	Visualisation des données.....	43
2.4.2.	Visualisation des partitions.....	44
2.4.3.	Visualisation de l'arbre de décision	44
2.5.	Module de sauvegarde	45
3.	Mise en œuvre des algorithmes	45
3.1.	Le clustering.....	45
3.1.1.	L'algorithme k-Means	45
3.1.2.	L'algorithme PAM.....	46
3.1.3.	L'algorithme CLARA.....	46
3.1.4.	L'algorithme hiérarchique ascendant.....	47
3.1.5.	L'algorithme génétique	48
3.1.5.1.	L'initialisation.....	49
3.1.5.2.	L'évaluation.....	49
3.1.5.3.	La sélection.....	50
3.1.5.4.	Le croisement	51
3.1.5.5.	La mutation	53
3.1.5.6.	Le critère d'arrêt.....	53
3.2.	La classification	54
3.2.1.	L'algorithme k-NN	54
3.2.2.	L'arbre de décision	54
3.3.	Les règles d'association	56
4.	Environnement de mise en œuvre.....	57
4.1.	Les classes d'objets	58
4.2.	L'interface utilisateur	62
5.	Conclusion	63

Chapitre IV : Tests et résultats	64
1. Introduction	65
2. Présentation des benchmarks	65
2.1. Benchmarks ESI	65
2.2. Benchmark IRIS [38]	69
2.3. Benchmarks IMAGE	69
2.4. Benchmark SUPERETTE	70
3. Tests de clustering	71
3.1. Tests sur les algorithmes	72
3.1.1. L'algorithme CLARA	72
3.1.2. L'algorithme hiérarchique ascendant	74
3.1.3. L'algorithme génétique	75
3.1.4. Comparatif des algorithmes de clustering	79
3.2. Tests sur les benchmarks	84
3.2.1. Benchmark ESI-4SIQ	84
3.2.2. Benchmark IMAGE	86
4. Tests de classification	87
4.1. Tests sur les algorithmes	87
4.1.1. L'algorithme k-NN	87
4.1.2. L'arbre de décision	89
4.1.3. Comparatif des algorithmes de classification	89
4.2. Tests sur les benchmarks	90
5. Tests des règles d'association	91
5.1. Tests sur l'algorithme Apriori	91
1.1. Tests sur les benchmarks	93
1.1.1. Benchmark SUPERETTE	93
1.1.2. Benchmark ESI-3SIQ	94
6. Conclusion	94
 Conclusions et perspectives	 96
 Bibliographie	 98
Annexe : Benchmarks	101

Résumé

Le data mining, ou fouille de données, est un domaine pluridisciplinaire permettant, à partir d'une très importante quantité de données brutes, d'en extraire des informations cachées, pertinentes et inconnues auparavant.

Nous nous proposons, dans le présent travail, de développer une application de data mining dont l'approche générale procède par trois étapes :

D'abord, nous employons la technique de classification non supervisée (clustering) pour structurer les données de départ en groupes de données homogènes (classes). Pour ce faire, nous mettons en œuvre des algorithmes dédiés à la classification non supervisée en mettant l'accent sur l'algorithme génétique.

Une fois les classes construites, nous utilisons la technique de classification supervisée pour prédire la classe de toute nouvelle donnée arrivée. Pour cela, nous mettons en œuvre des algorithmes de classification supervisée.

Enfin, les données sont soumises à un algorithme de recherche d'associations pour en extraire les éventuelles règles d'associations (corrélations) qui s'y cachent.

Mots clés :

data mining, fouille de données, ECD, clustering, algorithmes génétiques, classification, règles d'association.

Abstract

Data mining is a multidisciplinary field that is used to dig through vast quantities of data to discover hidden patterns and relationships previously unknown.

We propose, in this paper, to develop an application of data mining whose approach proceeds by three stages:

First, we use the technique of clustering in order to structure the original data in groups of homogenous data (classes). To do this, we propose algorithms of clustering with emphasis on the genetic algorithm.

Once classes constructed, we use the technique of scoring to predict the class of any new data. For this, we propose scoring algorithms.

Finally, the data are subject to a search algorithm in order to extract association rules (correlations) that may be hiding there.

Keywords :

data mining, KDD, clustering, genetic algorithms, scoring, association rules.

Liste des figures

Figure 1 : Etapes du processus ECD [7].....	8
Figure 2 : Représentation d'un centroïde.....	15
Figure 3 : Représentation d'un médoïde ($k=1$).....	15
Figure 4 : Partitionnement basé sur k-Means [19].....	18
Figure 5 : Partitionnement basé sur k-Medoids [12].....	19
Figure 6 : Dendrogramme [24].....	22
Figure 7 : Illustration du concept μ -directement densité-accessible [27].....	23
Figure 8 : Illustration du concept μ -accessible [27].....	24
Figure 9 : Illustration du concept μ -connecté [27].....	24
Figure 10 : Arbre de décision [28].....	28
Figure 11 : Exemple d'application de l'algorithme k-NN.....	30
Figure 12 : Perceptron à trois couches (schéma type) [28].....	31
Figure 13 : Architecture générale de l'application.....	38
Figure 14 : Visualisation des données dans un tableau.....	43
Figure 15 : Visualisation graphique d'une partition par l'ACP.....	44
Figure 16 : Représentation graphique d'un arbre de décision.....	44
Figure 17 : Déroulement de l'algorithme génétique [35].....	48
Figure 18 : Codage de partitions par classe.....	49
Figure 19 : Roulette de sélection.....	50
Figure 20 : Intervalles de sélection.....	51
Figure 21 : Croisement en un point de coupure.....	52
Figure 22 : Croisement en deux points de coupure.....	52
Figure 23 : Exemple de mutation.....	53
Figure 24 : Fenêtre principale de l'interface utilisateur.....	62
Figure 25 : Exemple d'un benchmark IMAGE.....	70
Figure 26 : Exemple d'un individu du benchmark SUPERETTE.....	71
Figure 27 : Qualité de clustering de CLARA en fonction de la taille de l'échantillon.....	73
Figure 28 : Qualité de clustering de CLARA en fonction du nombre d'essais.....	73
Figure 29 : Qualité de clustering de l'AG en fonction du mode d'initialisation.....	76
Figure 30 : Qualité de clustering de l'AG en fonction du nombre d'itérations.....	77
Figure 31 : Temps d'exécution de l'AG en fonction du nombre d'itérations.....	77
Figure 32 : Qualité de clustering de l'AG en fonction de la taille de la population de solutions.....	78
Figure 33 : Temps d'exécution de l'AG en fonction de la taille de la population de solutions.....	79
Figure 34 : Temps d'exécution des algorithmes de clustering en fonction de la taille de la population.....	80
Figure 35 : Temps d'exécution des algorithmes de clustering en fonction de la dimension de la population.....	81
Figure 36 : Qualité des résultats des algorithmes de clustering.....	82
Figure 37 : Qualité des résultats des algorithmes de clustering en fonction du nombre de classes.....	83
Figure 38 : Résultat de clustering sur le benchmark ESI-4SIQ (projection sur 2 attributs).....	84
Figure 39 : Erreur de classification de k-NN en fonction de la taille de l'ensemble d'apprentissage.....	87
Figure 40 : Erreur de classification de k-NN en fonction du nombre de voisins k	88

Figure 41 : Erreur de classification de l'arbre de décision en fonction de la taille de l'ensemble d'apprentissage	89
Figure 42 : Comparaison du taux d'erreur des algorithmes de classification.....	90
Figure 43 : Nombre de règles de l'algorithme Apriori en fonction du support minimum.....	91
Figure 44 : Nombre de règles de l'algorithme Apriori en fonction du support minimum.....	92
Figure 45 : Temps d'exécution de l'algorithme Apriori en fonction de la dimension	93

Liste des tableaux

Tableau 1 : La famille des benchmarks ESI.....	66
Tableau 2 : Attributs communs aux benchmarks ESI.....	66
Tableau 3 : Attributs du benchmark ESI-1TRC.....	67
Tableau 4 : Attributs du benchmark ESI-2TRC.....	67
Tableau 5 : Attributs du benchmark ESI-3SI.....	67
Tableau 6 : Attributs du benchmark ESI-3SIQ.....	68
Tableau 7 : Attributs du benchmark ESI-4SI.....	68
Tableau 8 : Attributs du benchmark ESI-4SIQ.....	68
Tableau 9 : Attributs du benchmark SUPERETTE.....	70
Tableau 10 : Premiers tests sur les algorithmes de clustering.....	72
Tableau 11 : Temps d'exécution de CLARA en fonction du nombre d'essais.....	74
Tableau 12 : Résultats de l'algorithme hiérarchique ascendant en fonction de la métrique inter-classe.....	75
Tableau 13 : Meilleurs paramètres des algorithmes de clustering.....	79
Tableau 14 : Classement des algorithmes de clustering selon le temps d'exécution.....	82
Tableau 15 : Classement des algorithmes de clustering selon la qualité des résultats.....	83
Tableau 16 : Résultats de clustering appliqué sur le benchmark ESI-4SIQ.....	85
Tableau 17 : Résultats de clustering appliqué sur le benchmark IMAGE.....	86
Tableau 18 : Tests de classification sur le benchmark ESI-SIQ-SI.....	91
Tableau 19 : Résultats d'application de l'algorithme Apriori sur le benchmark SUPERETTE.....	93
Tableau 20 : Résultats d'application de l'algorithme Apriori sur le benchmark ESI-3SIQ.....	94

Liste des algorithmes

Algorithme 1 : Distance de Minkowski	40
Algorithme 2 : Distance entre deux classes.....	41
Algorithme 3 : Inertie d'une population.....	42
Algorithme 4 : Inertie intra-classe	42
Algorithme 5 : Inertie inter-classe	42
Algorithme 6 : L'Algorithme k-Means [24]	45
Algorithme 7 : L'Algorithme PAM [25].....	46
Algorithme 8 : L'Algorithme CLARA [17].....	47
Algorithme 9 : L'Algorithme hiérarchique ascendant.....	47
Algorithme 10 : L'Algorithme de sélection par la roulette.....	51
Algorithme 11 : L'Algorithme k-NN [5].....	54
Algorithme 12 : L'Algorithme ID3 [37].....	55
Algorithme 13 : L'Algorithme Apriori [32].....	56
Algorithme 14 : L'Algorithme de génération des itemsets candidats	57

Introduction générale

Les quantités d'informations collectées chaque jour dans le monde ne cessent d'augmenter. En effet, le développement des capacités de stockage, particulièrement les entrepôts de données (*datawarehouse*¹) et les vitesses de transmission des réseaux ont conduit à leur croissance exponentielle. Certains experts estiment que le volume de données double tous les ans [1]. A titre d'exemple, la société Wal-Mart (chaîne de supermarchés aux états unis) effectue environ 21 millions de transactions par jour dans ses bases de données [2]. Ces quantités gigantesques de données ne pouvant pas être traitées et analysées humainement, les bases de données risquent de devenir des "cimetières" de données inutiles (*data tombs*).

Le data mining est l'ensemble des algorithmes et techniques destinés à l'exploration et l'analyse des grandes bases de données en vue de détecter des règles, des associations, des tendances inconnues non fixés à priori, des structures particulières restituant de façon concise l'ensemble de l'information utile (connaissance).

Le data mining se répand particulièrement dans les secteurs qui sont, par leur activité, susceptibles de drainer des flux de données économiques et comportementales individualisées : grandes distributions, téléphonie, banques, assurances... Selon [3], le data mining est l'une des dix technologies qui "changeront" le monde dans le 21^{ème} siècle.

Notre travail consiste à développer une application qui permet d'appliquer des algorithmes de data mining. Il s'agit d'un outil d'aide à la décision dont la démarche générale s'articule autour de trois étapes principales :

Dans la première étape, nous utilisons la technique de classification non supervisée (clustering) dans le but de structurer les données en un certain nombre de clusters (classes). Chaque classe représente une tranche de la population d'étude ayant un certain degré de similarité.

Dans la seconde étape, nous utilisons la technique de classification supervisée dans le but de prédire la classe de nouvelles données.

La dernière étape est consacrée aux règles d'association. Elle consiste en l'étude des caractéristiques des données dans le but d'identifier d'éventuelles corrélations (associations) entre ces caractéristiques.

Après avoir introduit les éléments nécessaires pour la lecture du document, la suite s'organise de la manière suivante :

¹ datawarehouse : Base de données dans laquelle est centralisé un volume important de données. L'organisation des données est conçue pour que les personnes intéressées aient accès rapidement et sous forme synthétique à l'information dont elles ont besoin.

Le premier chapitre est consacré à une présentation générale du concept de data mining. Nous y introduisons le processus ECD, la place du data mining dans ce processus et les différents types de problèmes que l'on peut résoudre à l'aide de data mining.

Le deuxième chapitre fait un état de l'art de différentes techniques et algorithmes de data mining utilisés, notamment, pour la classification automatique de données (supervisée et non supervisée) et l'extraction des règles d'association.

Le troisième chapitre décrit la partie conceptuelle. Nous présentons, entre autres, la mise en œuvre des algorithmes. A ce niveau, un accent est porté sur l'implémentation de l'algorithme génétique pour résoudre le problème de clustering.

Enfin le quatrième et dernier chapitre est consacré aux tests et résultats où les algorithmes sont testés face à différents jeux de données réels.

Chapitre I : Concepts et généralités sur le data mining et le processus ECD

1. Introduction

Les technologies de data mining permettent, grâce aux processus d'intelligence artificielle, de traiter des masses gigantesques de données afin d'en extraire l'information cruciale (connaissance), celle qui sera déterminante pour une prise de décision efficace. Une connaissance est définie par un ensemble de relations (règles, phénomènes, exceptions, tendances...) entre les données.

Le data mining est apparu au début des années 1990. Cette émergence est le résultat de la combinaison de plusieurs facteurs à la fois technologiques, économiques et même sociopolitiques. Les volumes gigantesques de données constituent, dès lors, des mines d'informations stratégiques aussi bien pour les décideurs que pour les utilisateurs.

Le data mining est l'un des maillons de la chaîne de traitement pour la découverte de connaissances à partir de données (ECD). Sous forme imagée, on pourrait dire que l'ECD est un véhicule dont le data mining est le moteur [4].

Dans ce chapitre introductif, nous allons donner les concepts fondamentaux de data mining et du processus ECD. Il s'agit d'une première introduction du sujet qui sera affinée et approfondie tout au long du reste de ce document.

2. Le processus ECD

Le data mining est une étape d'une chaîne 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. Définition

L'*extraction de Connaissances à partir de Données (ECD)* désigne 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 [5].

2.2. Etapes du processus ECD

L'ECD est un processus complexe qui se déroule suivant une suite d'opérations. Selon [6], l'ECD peut être vu comme un processus en sept étapes :

1. Positionnement du problème ;
2. Collecte et sélection des données ;
3. Nettoyage des données ;
4. Actions sur les attributs ;
5. Construction du modèle ;
6. Evaluation des résultats ;
7. Intégration de la connaissance.

2.2.1. Positionnement du problème

Il s'agit de comprendre le contexte de la recherche pour donner une signification logique aux variables de sa problématique. Dans cette étape, toutes les variables susceptibles d'expliquer le phénomène étudié sont identifiées.

Une première approche consiste à reformuler le problème pour qu'il puisse être traité par les techniques et les outils de modélisation (par exemple modéliser le problème sous forme d'une fonction mathématique).

Une autre approche, des plus efficaces, consiste à découper le problème complexe en sous-problèmes plus simples et à traiter, indépendamment, chacun de ces sous-problèmes.

2.2.2. Collecte et sélection des données

On procède d'abord par une collecte de données. Ces dernières se trouvant généralement dans des bases de données ou dans des datawarehouses. 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...).

2.2.3. Nettoyage des données

Une faible qualité de données (erreurs de saisie, champs nuls, valeurs aberrantes...) impose généralement une phase de nettoyage de celles-ci. Cette étape a pour objectif de corriger ou de contourner les inexactitudes ou les erreurs de données.

On commence par rechercher et éliminer les valeurs aberrantes. Ensuite, on vise à gérer les données manquantes par l'une des méthodes suivantes :

- Exclure les enregistrements incomplets ;
- Remplacer les valeurs manquantes (généralement par la moyenne) ;
- Gérer les valeurs manquantes (lorsque l'absence de donnée est acceptable du point de vue de la performance).

2.2.4. Actions sur les attributs

Une fois les variables sont pertinentes et les données sont fiables, une transformation éventuelle s'impose pour les préparer au travail d'analyse. Il s'agit d'intervenir sur les valeurs des variables pour qu'elles soient mieux exploitables par les algorithmes de traitement. Voici des exemples d'actions sur les variables parmi les plus usuelles :

Modification de l'unité de mesure :

Afin d'éviter certaines disproportions, il est recommandé de procéder à une "normalisation" des distributions. Dont l'objectif est d'obtenir des ordres de grandeurs comparables pour chaque attribut. Pour un échantillon de données numériques, la normalisation consiste à soustraire, à chaque valeur, la valeur moyenne de l'échantillon et à diviser cette différence par l'écart-type de l'échantillon.

$$\text{valeur normalisée} = \frac{\text{valeur} - \text{moyenne}}{\text{écart-type}} \quad [6]$$

Transformation des dates :

Les dates absolues ont généralement beaucoup moins de "valeur", du point de vue d'un travail de modélisation, que des fréquences ou des écarts entre dates. Ainsi, on calculera par exemple l'ancienneté d'un étudiant par la différence entre la date d'entrée et la date actuelle.

Modification des données géographiques en coordonnées :

Les techniques de data mining ont généralement des difficultés à appréhender des codes postaux ou des adresses. Une approche habile consiste à joindre des coordonnées de longitude et de latitude (géocodage).

2.2.5. Construction du modèle (data mining)

Il s'agit de la phase la plus souvent décrite sous le terme de "data mining" et qui repose, pour partie, sur une recherche exploratoire, dépourvue de préjugés concernant les relations entre les données.

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 [6] :

- 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 du type "x, y et A" (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 [6].

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.

2.2.6. Evaluation du résultat

Cette étape permet d'estimer la fiabilité du modèle c'est-à-dire sa qualité à déterminer correctement les valeurs, qu'il est censé avoir apprises, sur des cas nouveaux.

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 au nombre total de prédictions. Un niveau de prédiction de 80% est jugé moyen [6].

2.2.7. Intégration de la connaissance

La connaissance n'est rien tant qu'elle n'est pas convertie en décision puis en action. Cette étape est essentielle puisque il s'agit de la transition du domaine des études au domaine opérationnel.

Dans cette phase finale, il est opportun de dresser un bilan du déroulement des étapes précédentes. Ce bilan sert à améliorer l'existant en termes de données et de collecte de ces données ; par exemple, la faible qualité constatée de données conduit à revoir les processus d'alimentation de la base.

Les étapes du processus ECD peuvent être intégrées dans trois grandes étapes :

- L'étape de prétraitement : Elle regroupe la collecte, la sélection, le nettoyage des données et les actions sur les variables ;
- L'étape de data mining proprement dit ;
- L'étape de post-traitement : Elle regroupe l'évaluation des résultats et l'intégration de la connaissance. (Figure 1 *Figure 1*)

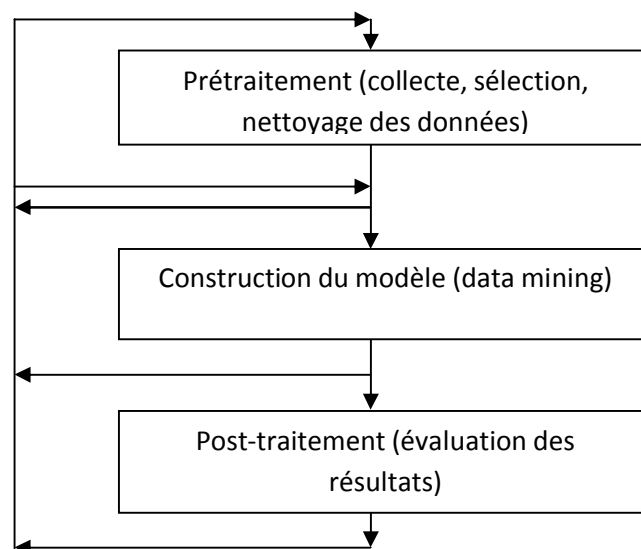


Figure 1 : Etapes du processus ECD [7]

3. Le data mining

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

3.1. Définition

Le data mining, ou fouille de données, est l'une des étapes du processus ECD (voir 2.2). Cette étape regroupe l'ensemble des méthodes et techniques destinées à l'exploration des bases de données de façon automatique, ou semi-automatique, dans l'optique de détecter des règles, des associations, des tendances nouvelles et imprévisibles, des structures particulières restituant l'essentiel de l'information utile [8].

Selon [9], il s'agit d'un processus de sélection, exploration, modification et modélisation de grandes bases de données afin de découvrir des relations entre les données jusqu'alors inconnues.

3.2. Principe

Le data mining adopte une démarche sans a priori (approche pragmatique) et essaie ainsi de faire "émerger, à partir de données brutes, des inférences que l'expérimentateur peut ne pas soupçonner, et dont il aura à valider la pertinence à l'aide des techniques et des algorithmes" [10].

3.3. Tâches de data mining

Le data mining est un ensemble de techniques complémentaires dédiées à différentes tâches. Selon [11], ces techniques sont partagées, principalement, entre la classification automatique (supervisée et non supervisée) et la recherche d'associations.

La 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 [11]. Le modèle généré permet, en outre, de prédire ou estimer la valeur manquante ou erronée en utilisant le modèle de classification comme référence [12].

La 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 [11].

Le principe général de la classification automatique supervisée, également appelée clustering, repose sur le regroupement des objets de telle manière que ceux qui appartiennent à la même classe soient fortement similaires entre eux et fortement dissimilaires avec les objets qui appartiennent aux autres classes. Cette tâche précède, généralement, la phase de clustering.

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 " $A \rightarrow B$ " qui signifie que la présence de A implique la présence de B (avec une certaine probabilité).

Exemple : Un étudiant qui réussit en mathématiques réussira en algorithmique dans 80% des cas.

3.4. Techniques de data mining

Les techniques de data mining diffèrent en fonction des besoins de l'utilisateur, entre autres la tâche à effectuer. Chacune des tâches citées ci-dessus regroupe une multitude d'algorithmes pour construire le modèle auquel elle est associée.

Selon [13], les dix algorithmes les plus populaires dans le domaine de data mining sont, dans l'ordre: C4.5, k-Means, SVM, Apriori, EM, PageRank, AdaBoost, k-NN, Naive Bayes et CART. Ces algorithmes sont classifiés, selon la tâche associée, comme suit :

- Algorithmes de construction des arbres de décision : CART, C4.5 ;
- Algorithmes de classification non supervisée : k-Means, EM ;
- Algorithmes de classification supervisée : k-NN, SVM, Naive Bayes et AdaBoost ;
- Algorithme d'extraction des règles d'association : Apriori ;
- Algorithme de classification automatique des pages Web : PageRank.

Un état de l'art des principaux algorithmes de data mining fait l'objet du chapitre II.

3.5. Domaines d'application

Le data mining est une spécialité transverse, elle regroupe un ensemble de théories et d'algorithmes ouverts à tout domaine susceptible de drainer une masse importante de données. Parmi ces domaines, on cite :

- 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 data mining est utilisé pour la détection de fraudes et leur anticipation ;
- Secteur médical : Dans le secteur médical, le data mining 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 data mining 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.

3.6. Difficultés de data mining

La mise en œuvre de data mining rencontre trois difficultés principales [14] et [15] :

Qualité des données :

Les statistiques ont montré que 60% à 70% du temps de travail dans un projet de data mining est consacré au prétraitement des données (sélection, correction, transcodage, chargement...), ce qui montre que le temps de préparation est un inconvénient majeur qui influe sur le temps global du projet.

Choix des algorithmes et de l'itinéraire du travail :

Pour pouvoir répondre aux questions qui se posent, les algorithmes doivent être choisis en fonction du problème traité. Il faut que l'expert en datamining soit aussi un animateur et possède des qualités que l'on trouve rarement ensemble chez la même personne : rigueur dans la méthode, ouverture et chaleur humaine dans la communication.

Evaluation des résultats :

Avant de procéder au déploiement final du modèle, il est important de l'évaluer plus complètement et de passer en revue toutes les différentes étapes exécutées pour construire ce modèle. Ceci permettra d'être certain qu'il permet d'atteindre les objectifs fixés. Lors de la définition du problème, un objectif principal est de déterminer s'il y a un aspect important du problème à résoudre qui n'a pas été suffisamment considéré. A la fin de cette phase, une décision sur l'utilisation des résultats fournis par les outils de data mining devrait être prise.

4. Conclusion

Nous avons exposé, à travers ce chapitre introductif, les principales notions évoquées sur le data mining qui n'est qu'une étape de traitement au sein du processus ECD. Ce dernier est né de la nécessité d'englober et de compléter le data mining pour que le traitement soit le plus automatisé possible.

Nous avons vu que les techniques de data mining ne font pas état des hypothèses fixées à priori, comme le font les statistiques traditionnelles, mais cherchent à "établir" un modèle par l'exploration des bases de données. Le data mining fait passer de l'analyse confirmatoire à l'analyse exploratoire.

Nous avons également montré que les tâches de data mining sont partagées entre la classification automatique supervisée, la classification automatique non supervisée et les règles d'association. Ces tâches se distinguent, chacune, par ses objectifs, ses algorithmes et ses résultats.

Le chapitre suivant sera consacré à l'étude des principaux algorithmes de data mining.

Chapitre II : Etat de l'art sur les techniques de data mining

1. Introduction

Le data mining offre une très grande variété de techniques et d'algorithmes de fouille de données. Ces algorithmes ont des origines diverses ; Certains sont issus des statistiques (régression...), d'autres de l'intelligence artificielle (réseaux de neurones, arbres de décision...), certains encore s'inspirent de la théorie de l'évolution (algorithmes génétiques...) ou de l'éthologie (colonies d'abeilles...). Cette combinaison de technologies facilite la résolution, la compréhension, la modélisation et l'anticipation des problèmes.

Dans ce deuxième chapitre, on se propose de donner les bases mathématiques, les algorithmes et les programmes de calcul pour les principales tâches de data mining : la classification non supervisée, la classification supervisée et l'extraction des règles d'association.

Chaque tâche comporte un recueil d'algorithmes parmi les plus courants. Pour chaque algorithme, on donne le principe de fonctionnement, les caractéristiques, les avantages, les inconvénients et éventuellement un exemple d'application.

Les données sont considérées comme étant l'ensemble des enregistrements de la base de données. Chaque enregistrement représente un objet et les champs de l'enregistrement définissent les attributs de l'objet.

2. La classification automatique non supervisée

2.1. Introduction

La classification est une tâche appliquée dans la vie courante. Elle est utilisée pour expliquer les nouveaux phénomènes rencontrés en les comparant avec des concepts et des phénomènes connus et en essayant de rapprocher les caractéristiques le plus possible. C'est un sujet de recherche actif qui se place au cœur de data mining, ce qui justifie pleinement l'intérêt qui lui est porté.

Les techniques de classification sont réparties essentiellement en deux approches : la classification supervisée et la classification non supervisée (appelée aussi "clustering"). L'objectif le plus simple de clustering est de répartir une population d'objets en groupes d'observations homogènes (classes), chaque classe étant bien différenciée des autres, pour former une partition.

Le clustering est un sujet abordé par plusieurs métiers. Ce fait lui donne l'avantage d'être utilisé par divers domaines d'application qui peuvent être catalogués suivant le but recherché selon trois grands types [16] :

- Extraction des connaissances : Dans ce type d'applications, on cherche généralement à donner une description sémantique aux groupes constitués afin de donner un sens à l'information dont on dispose ;

- Réduction des bases de données : L'objectif visé est de réduire l'espace des données sur lequel on travaille en espérant que le problème qu'on cherche à résoudre devient beaucoup plus simple ;
- Etude de comportement (*profiling*) : Le but est de détecter des sous populations qui partagent des caractéristiques proches, ce qui leur donne un certain comportement commun. Les systèmes de décision se basent souvent sur l'étude de comportement afin d'entreprendre des actions convenables à chacune des sous population.

Dans ce qui suit, nous allons présenter les principaux algorithmes de clustering parmi les plus représentatifs tout en essayant de bien cerner les points forts et les faiblesses de chaque algorithme.

2.2. Définition

Le clustering est une approche de classification dont les classes existent a posteriori. Au départ, on dispose d'un ensemble d'objets non étiquetés (dont la classe est inconnue). A partir de ces objets, l'idée est de parvenir à détecter des objets "similaires" afin de les regrouper dans des classes [16].

Selon [17], le clustering consiste à diviser une population d'objets en sous-ensembles d'objets appelés classes pour que tous les objets dans une même classe soient similaires et les objets de classes distinctes soient dissimilaires.

Parmi les objectifs que le clustering doit permettre d'atteindre, citons la réduction des données et la prédiction basée sur les groupes.

2.3. Terminologie

Avant d'entamer le processus de clustering, il est indispensable de définir quelques concepts :

2.3.1. Représentation d'une classe

C'est la description de l'ensemble des objets constituant la classe. Cette description, caractérisée par un ensemble de paramètres, permet de définir la forme et la taille de la classe dans un espace de données.

Dans la plupart des algorithmes de classification, la description d'une classe est simplement une combinaison linéaire des objets (centroïde), ou par un axe médian (médoïde) [18].

Le centroïde :

Un centroïde d'une classe C est un vecteur où chacune de ses composantes correspond à la moyenne arithmétique (quand elle existe) des objets de la classe C [18].

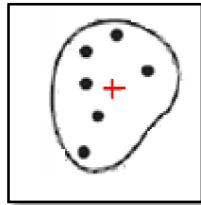


Figure 2 : Représentation d'un centroïde

Le médoïde :

On prend k objets parmi les objets de la classe. Ces k éléments sont centraux vis-à-vis de la classe, c'est-à-dire qu'ils sont proches du centre géométrique de la classe [18]. Ils permettent de représenter la classe non pas par un objet unique, qui n'a pas toujours d'existence réelle, mais plutôt par un noyau médian censé définir au mieux la classe.

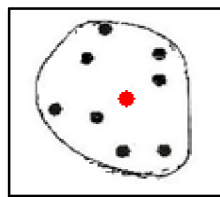


Figure 3 : Représentation d'un médoïde ($k=1$)

2.3.2. La notion de proximité (distance)

Une mesure de distance (appelée aussi mesure de similarité ou de proximité) exprime une similarité ou une dis-similarité : plus deux objets se rassemblent, plus leur similarité est grande et plus deux objets s'éloignent, plus leur similarité est petite.

La mesure de distance peut s'appliquer entre deux objets ou entre deux classes.

Distance entre deux objets :

Cette mesure peut être exprimée par une formule qui génère un nombre positif reflétant la proximité ou l'éloignement entre deux objets.

Selon [11], pour qu'une fonction de deux objets $d(x, y)$ soit une vraie métrique de distance, elle doit remplir les conditions suivantes :

- $d(x, y) = 0$ si et seulement si $x = y$
- $d(x, y) \geq 0$ quels que soient x et y

- $d(x, y) = d(y, x)$. (symétrie)
- $d(x, y) \leq d(x, z) + d(z, y)$. (inégalité triangulaire)

Les mesures de distance entre objets dépendent notamment du type de données. Parmi les mesures qui traitent des données numériques très populaires, on trouve la distance de Minkowski [19], notée dm , entre deux objets x, y définis chacun par ses attributs ;

$x = (a_1, a_2, \dots, a_n), y = (b_1, b_2, \dots, b_n)$:

$$d_m(x, y) = \sqrt[q]{|a_1 - b_1|^q + |a_2 - b_2|^q + \dots + |a_n - b_n|^q} \quad [19]$$

Si $q = 1$, il s'agit de la distance de Manhattan.

Si $q = 2$, c'est la distance euclidienne.

Distance entre deux classes :

Les mesures inter-classe les plus populaires dans la littérature sont : le lien simple (*single link*), le lien moyen (*average link*) et le lien complet (*complete link*) [20].

La distance D entre 2 classes C_1, C_2 notée $D(C_1, C_2)$ est calculée pour toute paire d'objets, un objet de chaque classe. Une opération (Minimum pour le lien simple, Moyenne pour le lien moyen et Maximum pour le lien complet) est ensuite appliquée pour chaque paire d'objets.

$$D(C_1, C_2) = \text{Opération} \{ d(x, y) \mid x \in C_1, y \in C_2 \} \quad [20].$$

Tel que : $d(x, y)$ = distance entre les deux objet x et y .

2.3.3. L'inertie :

L'inertie d'une population d'objets est la moyenne des carrés des distances des individus à leur centre de gravité. Cette entité mesure l'homogénéité des objets de la population.

L'inertie intra-classe est la somme des inerties des classes. Plus cette entité est petite, plus les objets de la même classe sont homogènes.

L'inertie inter-classe est la moyenne (pondérée par le nombre de classes) des carrées des distances des centres des classes au centre global. Plus cette entité est importante, plus les classes sont éloignées [21].

Un bon clustering possède deux propriétés :

- Inertie intra-classe (variance des objets dans la même classe) minimale ;
- Inertie inter-classe (variance des centres des classes) maximale.

2.4. Les algorithmes de clustering

La majorité des algorithmes de clustering peuvent être répartis en trois grandes familles [12], [22] et [23] :

- 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 hiérarchiques : Ils sont descendants si, à partir d'une seule classe, ils cherchent à établir une partition par division ; ou ascendants s'ils cherchent à former des classes plus grandes par fusion de classes jusqu'à la satisfaction d'un critère d'arrêt.
- Les algorithmes à base de densité : Les classes sont formées selon le voisinage des objets et le niveau de densité de chaque objet.

2.4.1. Les algorithmes de partitionnement

Leur principe général est de démarrer à partir d'un certain nombre de classes qui sont partitionnés d'une manière aléatoire en effectuant une "redistribution" des objets ou en essayant d'identifier les classes comme étant des régions très peuplées jusqu'à la rencontre d'un critère d'arrêt.

Les algorithmes de partitionnement que nous allons développer sont k-Means, k-Medoids, PAM, CLARA et CLARANS.

2.4.1.1. L'Algorithme k-Means

L'algorithme k-Means (ou l'algorithme des centres mobiles) a été introduit par J. MacQueen et mis en œuvre sous sa forme actuelle par E. Forgy [12]. 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 (centroïde). k-Means 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 centroïdes et l'affectation des individus de données aux classes sont réalisées pendant les itérations successives [24].

La Figure 4 *Figure 4* montre un exemple de déroulement de l'algorithme k-Means sur un nuage d'objets bidimensionnels, avec $k = 3$.

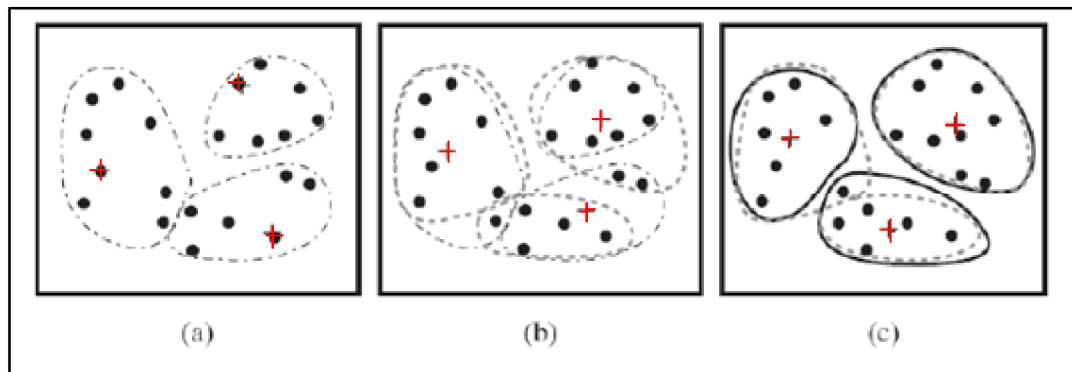


Figure 4 : Partitionnement basé sur k-Means [19].

Figure 4.a : Choix aléatoire de trois centre initiaux (objets) marqués par (+) et affectation de chaque objet restant au centre le plus proche ;

Figure 4.b : Calcul du nouveau centre pour chaque classe (moyenne des objets de la classe) et redistribution des objets selon les nouveaux centres ;

Figure 4.c : Le processus est répété jusqu'à stabilité des classes.

k-Means vise donc à minimiser la variance intra-classe qui se traduit par la minimisation de l'énergie suivante [11] :

$$E = \frac{1}{2} \sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \|x_i - \mu_j\|^2 ; \quad \text{à } \mu_j \in \mathbb{R}^D$$

Comme avantages de cet algorithme, on cite [24] :

- ✓ 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 ($k, t \ll n$) ;
- ✓ Il est indépendant de l'ordre d'arrivée des données.

Parmi les inconvénients de cet algorithme, on cite [24] :

- 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 converge souvent vers un optimum local ;
- Il est sensible aux objets isolés (bruits) ;
- Il n'est pas adapté pour découvrir des classes avec des structures non-convexes, et des classes de tailles et formes différentes.

2.4.1.2. L'algorithme k-Medoids

Il est introduit par Kaufman et Rousseeuw [12]. L'esquisse de l'algorithme k-Medoids ressemble à celle de k-Means sauf que, contrairement à l'algorithme k-Means où la classe est représentée par une valeur moyenne, le centroïde, dans l'algorithme k-Medoids, une classe est représentée par un de ses objets prédominants, le médoïde.

L'algorithme k-Medoids utilise une fonction objectif qui définit la distance moyenne entre un objet et le médoïde.

Soient :

k : le nombre de classes défini à priori,

$P = \{p_1, p_2, \dots, p_n\}$, population de n objets,

$M = \{m_1, m_2, \dots, m_k\}$, l'ensemble des médoides des k classes,

$C = \{C_1, C_2, \dots, C_k\}$, l'ensemble des k classes,

$d(p_{ij}, m_i)$: distance d'un objet p_{ij} à son médoïde m_i , avec $p_{ij}, m_i \in C_i$ pour $j=1, |C_i|$

$|C_i|$ étant le nombre d'objets de la classe C_i .

La fonction objectif est définie de la manière suivante [12] :

$$J = \sum_{i=1}^k \sum_{p \in C_i} d(p, m_i)$$

La Figure 5 est une illustration du déroulement de l'algorithme k-Medoids sur un nuage d'objets bidimensionnels avec $k = 3$.

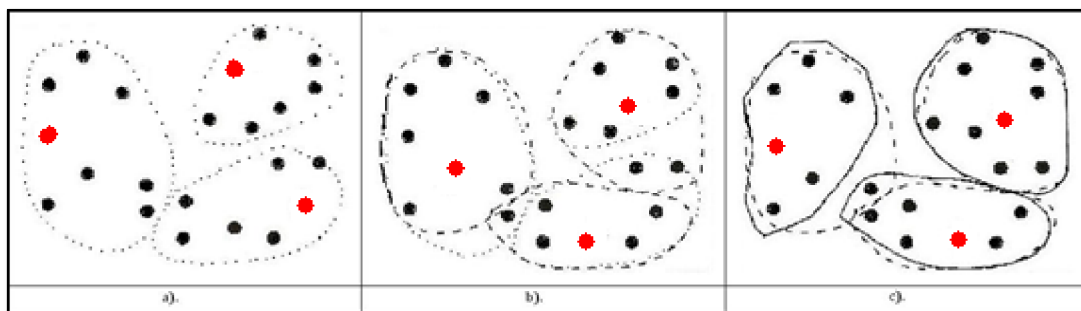


Figure 5 : Partitionnement basé sur k-Medoids [12].

Figure 5.a : Choix des trois centres initiaux marqués par (●) et affectation de chaque objet restant au centre le plus proche ;

Figure 5.b : Calcul des nouveaux médoides pour chaque classe et redistribution des objets selon les nouveaux médoides (Le médoïde étant l'objet le plus proche du centroïde de la classe) ;

Figure 5.c : Le processus est répété jusqu'à stabilité des classes.

Comme avantages de cet algorithme par rapport à k-Means, on cite :

- ✓ Il s'adapte à n'importe quel type de données ;
- ✓ Il est insensible aux objets isolés.

Pour améliorer la qualité de clustering de L'algorithme k-Medoids, de nombreuses variantes s'y sont succédées, entre autres les algorithmes PAM [25], CLARA [12] et CLARANS [17]. Ces algorithmes adoptent le même principe de représentation par médoïde et utilisent la même fonction objectif donnée ci-dessus.

2.4.1.3. L'Algorithme PAM

PAM (*Partitioning around Medoids*) a été développé par Kaufman et Rousseeuw [16]. L'idée de cet algorithme consiste à commencer avec un ensemble de k médoïdes puis échanger le rôle entre un objet médoïde et un non-médoïde si cela permet de réduire la distance globale, ce qui revient à minimiser la fonction objectif (voir 2.4.1.2) [25].

L'inconvénient majeur de cet algorithme est son coût total de calcul, il est d'une complexité quadratique de l'ordre de $O(k.(n-k)^2)$ avec n le nombre d'objets et k le nombre de classes pour chaque itération, ceci le rend non adaptable pour une population importante d'objets.

2.4.1.4. L'Algorithme CLARA

L'algorithme CLARA (*Clustering LARge Application*) a été mise en œuvre par Kaufman et Rousseeuw dans le but de réduire le coût de calcul de PAM [12] et [17].

Cet algorithme travaille sur des échantillons. On prend une petite partie d'objets (échantillon). Ensuite, les k médoïdes sont déterminés en appliquant PAM sur cet échantillon. Si cet échantillon est choisi d'une manière aléatoire, alors il représente bien tous les objets, donc les médoïdes sont similaires à ceux qui sont créés à partir de tous les objets [17].

L'algorithme est, généralement, exécuté sur plusieurs échantillons pour obtenir le meilleur résultat. Les auteurs ont indiqué, suite à leurs expérimentations, que la taille d'un échantillon de $(40+2k)$, k étant le nombre de classes, donne un bon résultat [16].

Le principal inconvénient de l'algorithme CLARA, outre les difficultés d'échantillonnage, est que si un objet qui serait le meilleur médoïde n'apparaît dans aucun échantillon, alors la meilleure solution ne pourra jamais être atteinte.

2.4.1.5. L'Algorithme CLARANS

CLARANS (*Clustering Large Application based on RANdomized Search*) a été mis en place par Raymon Ng et Jiawei Han [12] et [17].

Les auteurs de cet algorithme ont utilisé un graphe pour résoudre le problème de recherche des k médoïdes. Chaque nœud dans le graphe est un ensemble de k objets médoïdes et

possède $k.(n-k)$ voisins. Un nœud voisin s'obtient en échangeant les rôles entre le médoïde et un non-médoïde.

Une fois le graphe construit, PAM parcourt tous les nœuds voisins à la recherche de la solution, ce qui rend le coût de traitement trop cher. CLARA, à son tour, ne parcourt qu'un sous-graphe de l'ensemble des voisins, il travaille donc sur une région localisée.

CLARANS est une combinaison des deux concepts, il choisit aléatoirement un nombre de nœuds voisins (ce nombre est donné en paramètre) et se déplace d'un nœud au nœud adjacent qui minimise la fonction objectif (voir 2.4.1.2).

L'algorithme s'arrête quand aucun nœud adjacent ne peut optimiser la fonction objectif.

Cet algorithme de clustering a tous les avantages de PAM et de CLARA avec une complexité linéaire par rapport au nombre d'objets. D'ailleurs, l'expérimentation des auteurs a montré qu'il fournit toujours les meilleurs résultats par rapport au PAM et CLARA dans tous les cas [26].

2.4.2. Les algorithmes hiérarchiques

Ces algorithmes construisent les classes graduellement sous une forme hiérarchique, autrement dit, un arbre des classes appelé "dendrogramme" (Figure 6).

Ils sont divisés en 2 types : Agglomératifs (ascendants) et divisifs (descendants).

En agglomération, on commence par considérer chaque objet comme une classe et on essaye de fusionner deux ou plusieurs classes appropriées (selon une similarité) pour former une nouvelle classe. Le processus est répété jusqu'à atteindre un critère d'arrêt.

En division, tous les objets sont au début considérés comme une seule classe, on divise successivement les classes en classes plus raffinées selon une similarité. Le processus est répété jusqu'à atteindre un critère d'arrêt.

Le critère d'arrêt peut être le nombre de classes désiré, le nombre minimum (ou maximum) d'objets dans chaque classe, le nombre d'itérations, l'inertie...etc.

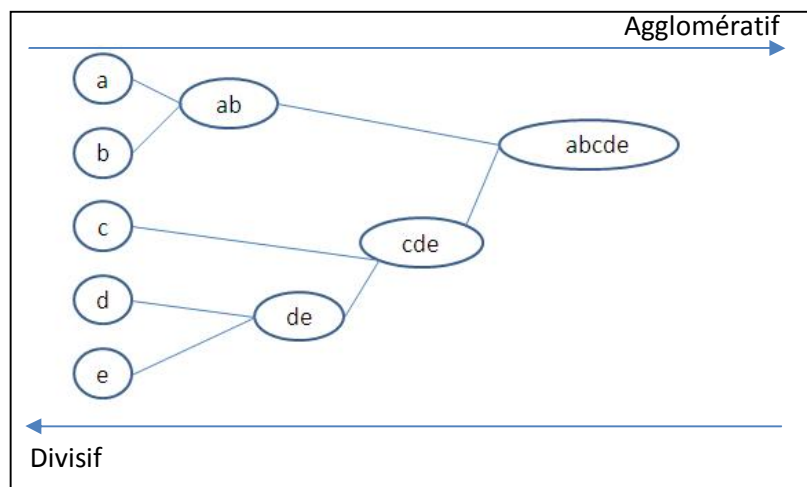


Figure 6 : Dendrogramme [24].

Il y a de nombreux algorithmes hiérarchiques qui sont proposés dans la littérature, les plus connus sont : BIRCH pour le groupement agglomératif et DIANA pour le groupement divisif.

2.4.2.1. L'Algorithme BIRCH

BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) est un algorithme de clustering hiérarchique agglomératif. Il a été Introduit par Zhang, Ramakrishan et Livny [12] et [17].

La communauté du domaine de classification trouve que BIRCH est l'un des meilleurs algorithmes pouvant traiter de gros jeux de données [17].

L'idée principale est que la classification est effectuée sur un ensemble d'objets compactée et organisée en une structure d'arbre équilibré appelé CF_Tree (*Clustering Feature Tree*) de taille limitée, au lieu qu'elle s'effectue sur l'ensemble des objets original. CF_Tree est une structure hiérarchique où chaque niveau représente une phase de clustering.

Parmi les avantages de BIRCH cités dans [7], on trouve :

- ✓ La capacité de traiter de grands volumes de données ;
- ✓ La structure CF_Tree peut être ajustée à l'espace mémoire disponible ;

Cependant, certaines faiblesses ont été constatées telles que :

- L'algorithme ne considère que les données numériques (dans sa version originale) ;
- Il est fortement sensible à l'ordre d'arrivée des données.

2.4.2.2. L'Algorithme DIANA

DIANA est un algorithme hiérarchique qui procède par division. Son fonctionnement repose sur un principe simple. Au départ, tous les objets appartiennent à un seul et unique groupe.

Un critère de partitionnement est, ensuite, utilisé pour diviser le groupe en deux sous-groupes et ensuite pour diviser chaque sous-groupe récursivement jusqu'à ce que chaque groupe contienne un seul élément.

2.4.3. Les algorithmes basés sur la densité

Ces algorithmes considèrent les classes comme étant des régions denses dans l'espace d'objets. Un objet de l'espace est dense si le nombre de ses voisins dépasse un certain seuil fixé à priori. Ils essaient alors d'identifier les classes en se basant sur la densité des objets dans une région. Les objets sont alors groupés non pas sur la base d'une distance mais sur la base de la densité de voisinage excède une certaine limite.

Parmi les algorithmes les plus connus dans cette catégorie, nous citons DBSCAN et OPTICS [12], [23].

Avant de développer les algorithmes, il faut d'abord définir quelques concepts [27] :

Soient :

X : ensemble d'objets ;

ϵ : rayon maximum de voisinage ;

MinPts : nombre minimum d'objets dans le voisinage d'un point.

Le voisinage d'un objet p par rapport à ϵ : $V_\epsilon(p) = \{q \in X / \text{distance}(p, q) \leq \epsilon\}$.

Un objet p est "directement densité-accessible" à partir de q (Figure 7) si :

- $p \in V_\epsilon(q)$
- $|V_\epsilon(q)| > \text{MinPts}$ (q est dit noyau)

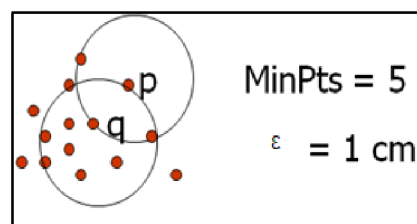


Figure 7 : Illustration du concept de directement densité-accessible [27]

Un objet p est "accessible" à partir de q s'il existe $p_1, \dots, p_n, p_1 = q, p_n = p$ tel que p_{i+1} est directement densité-accessible à partir de p_i pour $1 < i < n$. (Figure 8)

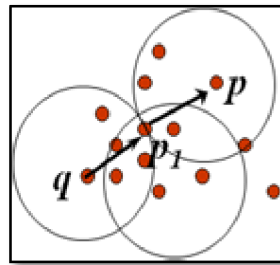


Figure 8 : Illustration du concept d'accessibilité [27]

Un objet p est "connecté" à q s'il existe un objet o tel que p et q soient accessibles à partir de o (Figure 9).

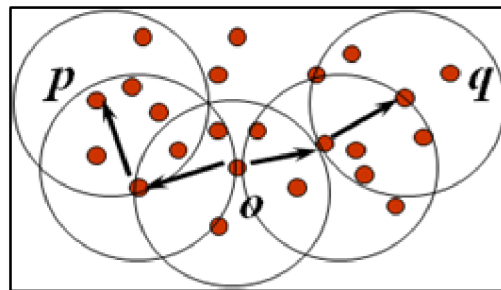


Figure 9 : Illustration du concept de connectivité [27]

Une classe est l'ensemble maximal des objets connectés [27].

2.4.3.1. L'algorithme DBSCAN

DBSCAN (*Density Based Spatial Clustering of Application with Noise*) est introduit par Ester, Kriegel, Sander et Xu [12].

La démarche générale de DBSCAN se résume comme suit [27] :

1. Sélectionner aléatoirement un objet p de l'ensemble d'objets X .
2. Déterminer l'ensemble E des objets accessibles depuis p dans X .
3. Si p est un noyau, alors E est une classe.
4. Sinon sélectionner un autre objet et aller en (2).
5. S'arrêter lorsque tous les objets ont été sélectionnés.

Comme avantages de DBSCAN, on cite [17] :

- ✓ Il peut chercher les classes de forme arbitraire ;
- ✓ Il est insensible à l'ordre d'arrivée des objets ;
- ✓ Il est incrémental car un nouvel objet qui arrive peut affecter certains voisinages.

Parmi les inconvénients de cet algorithme, on cite [17] :

- Il est difficile à préciser le rayon de voisinage ε ;
- La performance peut être diminuée pour les données de haute dimension.

2.4.3.2. L'algorithme OPTICS

L'algorithme OPTICS a été introduit par Ankerst, Breunig, Kriegel et Sandar pour pallier au premier inconvénient de l'algorithme DBSCAN. Il consiste à effectuer un clustering DBSCAN pour plusieurs valeurs du rayon de voisinage de manière simultanée [12].

L'algorithme procède de la manière suivante :

1. Commencer par créer une liste ordonnée de différents rayons ;
2. Appliquer ensuite DBSCAN, en parallèle, à cette liste ;
3. Choisir la plus petite distance qui assure la plus forte densité.

2.4.4. Autres algorithmes

2.4.4.1. Les algorithmes basés sur la grille

Le principe de cette famille d'algorithmes est qu'on divise l'espace de données en cellules (fusion des régions jugées denses, selon la valeur d'un seuil fixé à priori, les régions jugées peu denses permettent d'établir des frontières.). Donc ce type d'algorithme est conçu pour des données spatiales. Une cellule peut être un cube, une région ou un hyper rectangle. En fait, elle est un produit cartésien de sous intervalles d'attributs de données.

Avec une telle représentation des données, au lieu de faire la classification dans l'espace de données, on la fait dans l'espace spatial en utilisant des informations statistiques des objets dans la cellule. Les algorithmes les plus connus dans cette catégorie sont STING, CLIQUE et WaveCluster [23] et [17].

2.4.4.2. Les algorithmes basés sur les statistiques

Ces algorithmes démarrent du principe que les données de départ ont été générées suivant une certaine loi de distribution (normale, uniforme, exponentielle,...)

Le principe général est d'utiliser le modèle statistique permettant d'approcher au mieux cette distribution pour réaliser le groupement.

Parmi les systèmes de classification basés sur des modèles statistiques, on peut trouver COBWEB (Fisher), CLASSIT (Gennari, Langley et Fisher) et AutoCass (Cheeseman et Stutz) [12].

2.5. Conclusion

Nous avons constaté à travers cette étude des algorithmes de clustering qu'il en existe plusieurs. Ces algorithmes se distinguent principalement par :

- Le type des attributs manipulés (k-Means par exemple ne peut traiter que des attributs numériques) ;
- La complexité de calcul : un algorithme de grande complexité (PAM par exemple) a un coût de calcul élevé et s'adapte mal aux cas d'un jeu de données important ;
- Le critère d'arrêt : certains algorithmes exigent un critère de qualité pour l'arrêt, comme le cas des algorithmes de partitionnement.

Le choix d'un algorithme approprié dépend fortement du contexte de son application, la nature des données et les ressources disponibles. Une analyse attentive des données aide à bien choisir le meilleur algorithme à partir du moment qu'il n'existe pas un algorithme qui peut répondre à toutes les demandes.

3. La classification automatique supervisée

3.1. Introduction

La classification supervisée est une tâche largement appliquée dans la vie courante. En effet, il existe une multitude de problèmes qui entrent dans ce cadre, parmi lesquels on trouve la reconnaissance des caractères manuscrits, la reconnaissance des paroles, la catégorisation des textes, la détection des spams, l'aide au diagnostic médical, la bioinformatique...etc.

A la différence de la classification non supervisée où les groupes d'objets sont découverts à posteriori, les techniques de classification supervisée s'appliquent lorsqu'on veut rattacher un nouvel objet (observation) à une classe qui est choisie parmi un ensemble de classes connues. Cette étape suit, généralement, l'étape de clustering.

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'entrée (objets à classer) 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. Un exemple de cette catégorie d'algorithmes est le réseau de neurones.

Une autre approche qui n'est pas moins intéressante, c'est le raisonnement à partir des cas. Ces algorithmes ne cherchent pas à calculer le modèle mais à trouver, pour l'objet à classer, un ou plusieurs cas similaires déjà résolus pour en déduire la classe. Les arbres de décision et l'algorithme k-NN adoptent ce principe.

Dans ce qui suit, nous allons décrire et détailler les principaux algorithmes de classification automatique supervisée.

3.2. Problématique

On dispose d'un ensemble X , comportant N données étiquetées (dont la classe est connue). Chaque donnée x est caractérisée, par P attributs et par sa classe $C_i \in C$, C étant l'ensemble des classes.

Le problème consiste alors, en s'appuyant sur l'ensemble $X = \{ (x_i; C_i); i \in \{1..N\} \}$, à prédire la classe de toute nouvelle donnée x .

3.3. Les algorithmes de classification

Parmi les algorithmes de classification supervisée les plus populaires dans la littérature [24], on trouve : les arbres de décision, l'algorithme k-NN et les réseaux de neurones.

3.3.1. Les arbres de décision

Un arbre de décision est, comme son nom l'indique, un outil d'aide à la décision qui permet de classer 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 (Figure 10).

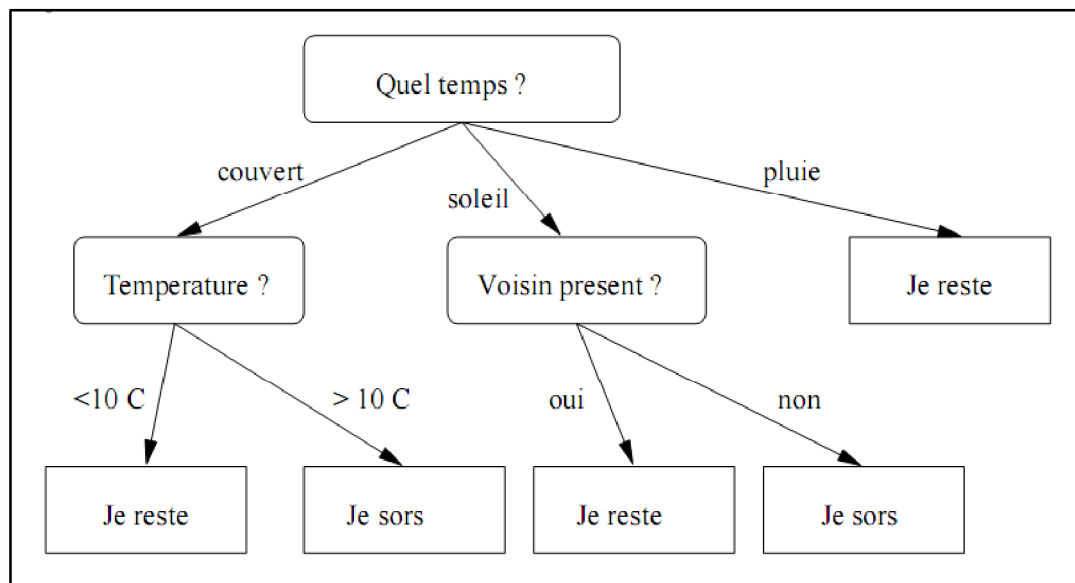


Figure 10 : Arbre de décision [28]

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.

Construction de l'arbre :

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 [29], CART (*Classification And Regression Trees*, introduit par Breiman et al [30] et CHAID (*Chi-squared Automatic Interaction Detection*) [11].

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.

Le choix des attributs peut se faire par plusieurs techniques, entre autres :

- Entropie (ID3, C4.5) [29] ;
- Indice de Gini (CART) [29] ;
- Table de Khi-2 (CHAID) [24].

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.

Les arbres de décision constituent un moyen très efficace de classification, et ce pour les avantages qu'elles présentent. Parmi ses avantages, on peut citer [30] [5] :

- ✓ Facilité à manipuler des données catégoriques ;
- ✓ Traitement facile des variables d'amplitudes très différentes ;
- ✓ La classe associée à chaque individu peut être justifiée ;
- ✓ Les attributs apparaissant dans l'arbre sont des attributs pertinents pour le problème de classification considéré ;

Ces méthodes présentent tout de même des inconvénients dont les plus importants sont :

- La sensibilité au bruit et aux points aberrants ;
- La sensibilité au nombre de classes (plus le nombre de classes est grand plus les performances diminuent) ;
- Le besoin de refaire l'apprentissage si les données évoluent dans le temps.

3.3.2. L'Algorithme k-NN

k-NN (*k Nearest Neighbours*) est un algorithme de raisonnement à partir de cas c'est-à-dire prendre des décisions en recherchant un ou plusieurs cas similaires déjà résolus. La décision consiste à chercher les k échantillons les plus voisins de l'objet et de l'affecter à la classe qui est la plus représentative dans ces k échantillons ("*dis-moi qui sont tes amis, et je te dirais qui tu es*").

L'approche la plus simple est de rechercher le cas le plus similaire et de prendre la même décision, on parle de 1-NN. Si cette approche peut fournir des résultats acceptables sur des problèmes simples pour lesquels les objets sont bien répartis en groupes denses de même classe, en règle générale, il faut considérer un nombre de voisin plus important pour obtenir de bons résultats [5].

Exemple :

La Figure 11 illustre un exemple d'application de k-NN pour classer les deux objets A, B avec $k = 4$; La classe (A) est () et la classe (B) est ().

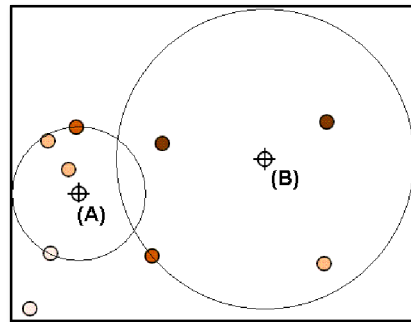


Figure 11 : Exemple d'application de l'algorithme k-NN

L'algorithme k-NN a les avantages de :

- ✓ Ne pas refaire l'apprentissage lors de l'introduction de nouveaux attributs ;
- ✓ Traiter tout type de données avec un nombre d'attributs élevé ;
- ✓ Fournir des résultats clairs.

Cependant, le coût de classification peut devenir cher car :

- D'une part, le temps pour calculer les voisins peut être prohibitif ;
- D'autre part, il faut toujours stocker le modèle durant toute l'opération de classification.

3.3.3. Les réseaux de neurones

Un réseau de neurones est un système composé de plusieurs unités de calcul simples (nœuds) fonctionnant en parallèle, dont la fonction est déterminée par la structure du réseau et l'opération effectuée par les nœuds.

Le principe de fonctionnement est le suivant : On dispose initialement d'une base de connaissances constituée de couples de données (entrées / sorties) et on souhaite utiliser cette base de données pour entraîner un algorithme à reproduire les associations constatées entre les entrées et les sorties de l'échantillon.

L'exemple le plus simple de réseau de neurones est souvent donné par le "perceptron multicouches" qui est un cas particulier de réseau de neurones (Figure 12).

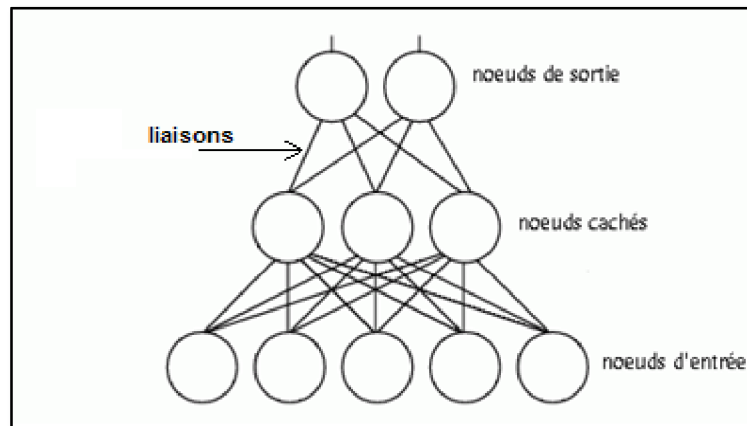


Figure 12 : Perceptron à trois couches (schéma type) [28]

Pour un réseau de neurones avec N nœuds d'entrée, notées $C(1), \dots, C(N)$, et N poids affectés aux liaisons et notés $w(1), \dots, w(N)$ l'entrée d'un nœud de la couche suivante sera généralement une somme pondérée des valeurs de sortie des neurones précédents :

$$X = w(1)*C(1) + w(2)*C(2) + w(3)*C(3) + \dots + w(N)*C(N)$$

Les poids sont des paramètres adaptatifs, dont la valeur est à déterminer en fonction du problème via un algorithme d'apprentissage (propagation, rétro-propagation...) [30].

Les réseaux de neurones peuvent être utilisés pour effectuer une classification supervisée floue de la manière suivante : chaque nœud d'entrée correspond à un attribut de l'objet (autant de nœuds d'entrée que d'attributs). On peut prendre un neurone de sortie par classe ; la valeur de sortie est la valeur de la fonction d'appartenance (probabilité que l'objet appartienne à cette classe) [30].

3.3.4. Autres méthodes de classification

3.3.4.1. Les réseaux bayésiens

Les réseaux bayésiens sont à la fois des modèles de représentation des connaissances et des machines à calculer les probabilités conditionnelles [16].

Le problème de classification peut être formulé en utilisant les probabilités a-posteriori [24] :

$P(C|X)$ = probabilité que l'objet $X = (x_1, \dots, x_k)$ soit dans la classe C .

L'idée est d'affecter à une instance X la classe C telle que $P(C|X)$ est maximale.

Théorème de Bayes [24] :

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

$P(X)$ est une constante pour toutes les classes.

$P(C)$ = fréquence relative des instances de la classe C .

$P(C/X)$ (probabilité que l'objet $X = (x_1, \dots, x_k)$ soit dans la classe C est maximal $= P(X/C) \cdot P(C)$ (probabilité que la classe C est dans le tuple (instance) $X = (x_1, \dots, x_k)$ fois la fréquence relative des objets de la classe C).

Le problème est que le calcul de $P(X/C)$ est non faisable !

Le remède est d'adopter "l'hypothèse naïve" de Bayes qui est basée sur l'indépendance des attributs (les attributs sont supposés indépendants les uns des autres), ce qui donne :

$$P(x_1, \dots, x_k | C) = P(x_1 | C) \cdot \dots \cdot P(x_k | C)$$

$P(x_i | C)$ est estimée comme la fréquence relative des instances possédant la valeur x_i (i-ème attribut) dans la classe C .

Le calcul des deux entités n'est pas coûteux.

3.4. Conclusion

Nous avons constaté à travers cette étude des algorithmes de classification supervisée qu'il en existe plusieurs ; certains sont issus des recherches sur l'apprentissage (*machine learning*) comme les arbres de décision et l'algorithme k-NN, d'autres de l'intelligence artificielle comme les réseaux de neurones. Ces algorithmes diffèrent principalement par :

- La capacité de traiter n'importe quel type de données (comme k-NN, les arbres de décision) ;
- Le temps de traitement (l'algorithme k-NN par exemple présente un temps de traitement relativement coûteux dû au calcul des voisins) ;
- La sensibilité aux bruits (les arbres de décision sont très sensibles aux bruits par rapport aux autres algorithmes) ;

La classification supervisée peut mesurer l'importance de chaque attribut pour classer de nouveaux objets. Par exemple le gain d'information permet de mesurer la typicité d'un attribut.

Enfin, à l'inverse de la classification non supervisée, il est ici simple d'évaluer les résultats d'une classification. Un ensemble de test (ensemble d'objets étiquetés) est prévu pour cela. On soumet chaque objet de l'ensemble de test à l'algorithme de classification et on vérifie si l'algorithme trouve la bonne classe.

4. Les règles d'association

4.1. Introduction

Nous présentons dans cette partie la troisième tâche de data mining qui est fondée sur la découverte des règles d'association à partir d'un ensemble de données. Ce thème est souvent considéré comme faisant parti des approches d'apprentissage non supervisé, utilisé dans le domaine de data mining et d'extraction des connaissances.

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.

Un exemple classique de l'utilité de cette approche est le panier de la ménagère [31] qui décrit un ensemble d'achats effectués au supermarché ; les règles d'association permettent de découvrir des règles dans l'ensemble de transactions (comme par exemple : Si "poisson" alors "riz"). Ces règles permettent par exemple au gérant de proposer des bons de réductions significatifs sur les achats futurs de ses clients.

Dans cette partie, nous nous intéressons aux algorithmes couramment utilisés pour découvrir les règles d'association dans les bases de données.

4.2. Problématique

Les règles d'associations ont été introduites par Agrawal R., Imielinski T. et Swami A. pour analyser le panier de la ménagère [32] (découvrir les articles qui figurent ensemble sur le ticket de supérette).

Le problème peut se présenter comme suit : Soit $I = i_1, i_2, \dots, i_n$ un ensemble d'articles et D un ensemble de transactions, où chaque transaction T est un ensemble d'articles $\subseteq I$.

Chaque transaction T est composée d'un ou de plusieurs articles (items).

Quelles sont les articles qui ont été achetés ensemble (qui apparaissent dans la même transaction) ?

4.3. Définitions

Dans un contexte plus général, la recherche d'associations se fait entre les attributs d'une population d'objets. Pour garder la même appellation, les attributs des objets sont appelés "items" ; un ensemble d'items est alors dit "itemset".

Une règle d'association est une implication entre deux itemsets (X, Y) de la forme $X \Rightarrow Y$, où $X \cap Y = \emptyset$. X est appelé l'*antécédent* et Y le *conséquent*.

Le support d'un itemset i est la portion des objets qui contiennent parmi leurs attributs tous les éléments de i .

La règle ($X \Rightarrow Y$) a un *support* de s dans l'ensemble des objets D si la règle est vérifiée dans $s\%$ de la base de transactions. Le support de la règle est défini comme étant $support(X \cup Y)$.

La règle ($X \Rightarrow Y$) a pour *confiance* c si $c\%$ des objets de D qui contiennent les attributs X contiennent aussi les attributs Y .

La confiance d'une règle ($X \Rightarrow Y$) est définie comme étant : $support(X \cup Y) / support(X)$.

Une règle ($X \Rightarrow Y$) qui a un support égal à " $s\%$ " et une confiance égale à " $c\%$ " est souvent représentée dans la littérature par la forme suivante [24] :

$X \Rightarrow Y[s\%, c\%]$; qui se traduit par : "Si X alors Y dans $c\%$ des cas, dans $s\%$ de la base".

Le nombre de règles trouvées pouvant être très important, un autre objectif de l'extraction est de ne générer que les règles dont le support et la confiance sont supérieurs respectivement aux seuils minimum de support (*MinSup*) et de confiance (*MinConf*). Ces deux entités sont définies de façon empirique par l'utilisateur.

Un itemset dont le support est supérieur à *MinSup* est appelé itemset fréquent [32].

La règle ($X \Rightarrow Y$) est une règle "intéressante" si $X \cup Y$ est un itemset fréquent et la confiance de la règle est supérieure ou égale à *MinConf* [32].

La découverte de règles d'associations se fait en deux étapes :

1. La recherche des itemsets fréquents ;
2. La recherche, à partir de ces itemsets, de règles "intéressantes".

La performance de l'extraction est surtout déterminée par la première étape qui est la plus coûteuse. Pour cela, les algorithmes Apriori et FP-Growth sont parmi les plus utilisés [33].

4.4. Les algorithmes de recherche d'associations

4.4.1. L'algorithme Apriori

Développé par Agrawal R. and Srikant R. [32], 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.

Parmi les avantages qui ont fait de Apriori un algorithme très populaire, on cite [24] :

- ✓ Il fournit des résultats clairs: règles faciles à interpréter ;
- ✓ Il est relativement simple ;
- ✓ Il n'exige aucune hypothèse préalable (apprentissage non supervisé) ;
- ✓ Il est facilement adaptable aux séries temporelles (ex. un client ayant acheté le produit A est susceptible d'acheter le produit B dans deux ans).

Cependant, l'algorithme Apriori n'est pas exempté de faiblesses [24] :

- Il est très coûteux en temps de calcul ;
- Il produit un nombre important de règles triviales ou inutiles (qui seront détruites par la suite) ;
- Algorithme non efficace pour les règles "rares" ;

4.4.2. L'algorithme FP-Growth

Mis en œuvre par Han, Pei, Yin et Mao, FP-Growth (*Frequent-Pattern Growth*) est un algorithme complètement innovant par rapport à l'algorithme Apriori [33].

FP-Growth utilise une structure de données compacte appelée FP-Tree (*Frequent Pattern Tree*) qui permet d'échapper à la nécessité de devoir scanner de façon répétée la base de données. De plus, les éléments étant triés dans la FP-Tree, la recherche d'associations est accélérée.

Un FP-Tree est composée d'une racine nulle et d'un ensemble de nœuds étiquetés par l'élément y présent. Un nœud contient aussi le nombre d'occurrences des itemsets où figurent la portion du chemin jusqu'à ce nœud et un lien inter-nœud vers les autres occurrences du même élément figurant dans d'autres séquences d'itemsets. Une table d'entête pointe sur la première occurrence de chaque élément.

L'avantage de FP-Tree est qu'il suffit de suivre les liens inter-nœuds pour connaître toutes les associations fréquentes où figure l'élément fréquent (recherche par élément) [33].

4.5. Conclusion

Nous avons présenté, dans cette section, les concepts généraux et les principaux algorithmes de recherche d'associations, Apriori et FP-Growth.

Apriori est un algorithme qui permet de découvrir les ensembles d'items fréquents, de manière incrémentale, en partant de ceux dont la longueur est 1. Les règles d'association sont ensuite générées à partir de ces itemsets fréquents.

FP-Growth est un algorithme qui permet, grâce à la structure *FP-tree* de filtrer les règles selon les éléments présents.

Les résultats fournis par les règles d'association peuvent être exploités pour en élargir le champ d'application. De même, certaines règles peuvent jouer un rôle de classification. En effet, si l'on dispose d'un attribut d'étiquetage des individus appartenant à un groupe, on peut espérer trouver des règles d'association dont la conclusion porte sur cet attribut du groupe. La règle prend alors la forme suivante :

Si (*attribut_1* ET *attribut_2* ... ET *attribut_n*) Alors (*individu_x* ∈ *groupe_y*)

La règle est alors de facto un classificateur. Il faudra probablement plusieurs règles de ce type pour former un classificateur complet, il peut être un réseau, une table ou une arborescence de règles.

Enfin, nous ajoutons qu'une analyse des règles peut être précédée d'une sélection d'attributs, ce qui aura l'avantage de réduire le coût de calcul et rendre l'analyse beaucoup plus abordable.

5. Conclusion

Nous avons présenté, dans cet état de l'art, les algorithmes de data mining couramment utilisés dans le but d'effectuer l'une des principales tâches de fouille de données (le clustering, la classification, la recherche d'associations).

Nous avons constaté que la littérature sur le sujet abonde d'algorithmes dont l'origine, le principe, le but et la performance diffèrent de l'un à l'autre. Cette richesse est due au fait que le data mining touche à plusieurs disciplines : les statistiques, l'analyse de données, l'intelligence artificielle et les bases de données.

Choisir le bon algorithme pour son projet de data mining est une tâche qui n'est pas toujours aisée. En effet, le choix d'un algorithme de data mining naît, généralement, d'un compromis entre plusieurs facteurs :

- Tâche à effectuer (classification, recherche d'association...) ;
- Type de données (l'algorithme doit être capable de traiter le type de données disponibles pour l'analyse) ;
- Quantité des données (l'algorithme choisi doit être adapté pour la quantité de données, surtout quand celle-ci est importante) ;
- Dimension des données (nombre d'attributs) ;
- Temps de traitement (certains algorithmes présentent un temps de traitement moins élevé que d'autres, ce qui les rend plus intéressants) ;
- Optimalité des résultats (certains algorithmes ne donnent pas un résultat optimal comme le cas des heuristiques) ;
- Interprétabilité des résultats (un algorithme qui fournit des résultats les plus en accord avec le métier est privilégié) ;
- Complexité de la mise en œuvre ;
- ...

Enfin, un mauvais choix de l'algorithme d'application engendre de lourdes conséquences sur le plan de la performance et des résultats. Dans certains cas, il peut faire échouer tout le projet.

Chapitre III : Conception et mise en Ê uvre

1. Introduction

Après avoir fait un tour d'horizon des techniques de data mining qui existent dans la littérature, ce troisième chapitre est consacré à la partie conceptuelle de notre application. Rappelons que l'objectif principal de notre travail est de développer une plateforme qui permet d'exécuter et de comparer différents algorithmes de data mining notamment pour effectuer les trois tâches principales (le clustering, la classification et l'extraction des règles d'association). Notons que notre étude est restreinte aux algorithmes qui traitent des données numériques.

Commençons, d'abord, par une présentation de l'architecture générale de notre application avant de développer les différents algorithmes.

2. Architecture générale de l'application

L'application que nous nous proposons de développer peut se voir comme étant l'interconnexion de plusieurs modules complémentaires indépendants l'un de l'autre. La Figure 13 illustre les différents modules et les interactions qui représentent des échanges de l'information.

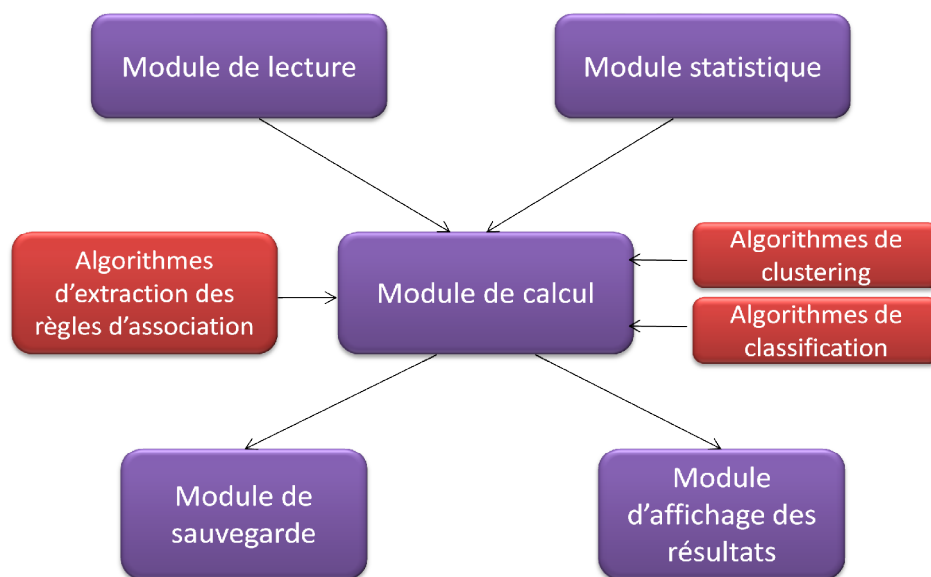


Figure 13 : Architecture générale de l'application

2.1. Module de lecture

On entend par module de lecture, ou chargement de données, toutes les opérations nécessaires pour accéder aux données brutes stockées dans des fichiers plats et les charger en mémoire en passant par deux étapes : accès aux données et nettoyage des données.

2.1.1. Accès aux données

Les données sont disponibles sous forme de fichier texte (benchmark) qui possède la structure suivante :

- La première ligne indique le nom de la population et le nombre d'individus ;
- La deuxième ligne indique le nombre d'attributs ;
- La troisième ligne représente les noms des différents attributs séparés par des espaces ;
- La quatrième ligne représente les valeurs de coupures des attributs (valeurs de discrétisation) ;
- Les individus sont ensuite représentés par un ensemble de valeurs numériques réelles signées, séparées par des espaces en commençant par un identifiant qui est censé designer chaque individu de manière unique. L'ensemble des individus est organisé à raison d'un individu par ligne.

2.1.2. Nettoyage des données

Les valeurs des attributs, n'étant éventuellement pas contrôlés, portent des valeurs parfois absentes, erronées ou aberrantes. Il est donc, nécessaire de prévoir un traitement adéquat pour chacun de ces problèmes.

Les individus comportant des champs non renseignés, des valeurs erronées, mal typés ou aberrantes sont ignorés.

2.2. Module de calcul

C'est le module qui constitue le noyau de l'application. En interaction avec les autres modules, il permet d'appliquer un algorithme de data mining à un jeu de données précédemment chargé.

Pour résoudre le problème de clustering, nous avons opté pour les algorithmes suivants :

- Un algorithme de partitionnement qui adopte la représentation par centroïde : k-Means, car il est le plus simple et plus utilisé [7] ;
- Deux algorithmes de partitionnement adoptant la représentation par médoïde : PAM et CLARA, car ils sont parmi les plus connus dans leur catégorie [17] ;
- Un algorithme hiérarchique ascendant ;
- L'algorithme génétique car il permet d'atteindre la solution optimale.

Pour résoudre le problème de classification, nous avons opté pour les deux techniques suivantes qui sont parmi les plus populaires [24] :

- L'algorithme k-NN ;
- L'arbre de décision.

Pour extraire les règles d'association, nous avons choisi l'algorithme Apriori car il constitue la base de la majorité des autres algorithmes de recherche d'associations [33].

2.3. Module statistique

C'est le module qui assure les différentes fonctions statistiques suivantes :

- Distance de Minkowski ;
- Distance entre deux classes ;
- Inertie ;
- Erreur de classification.

2.3.1. Distance de Minkowski

C'est la mesure de similarité entre deux individus. Cette fonction est largement utilisée dans les algorithmes de data mining. En effet, la plupart des algorithmes de clustering et de classification y font appel [19].

La procédure de calcul de la distance de Minkowski prend en paramètre deux individus o_1 , o_2 , avec n comme degré de pondération. Elle se résume comme suit :

Entrée : Deux individus o_1 et o_2 , Degré de pondération n

Somme = 0 ;

Pour chaque attribut i

 Somme = Somme + $(o_1.\text{attributs}[i] - o_2.\text{attributs}[i])^n$

Retourner : La racine $n^{\text{ième}}$ de Somme.

Sortie : Distance de Minkowski entre o_1 et o_2 avec degré de pondération n

Algorithme 1 : Distance de Minkowski

2.3.2. Distance entre deux classes

Pour mesurer la distance entre deux classes, trois techniques différentes sont mises en œuvre, il s'agit du :

- Lien simple ;
- Lien moyen ;
- Lien complet.

Nous avons combiné les trois techniques dans un seul algorithme. Il prend, en paramètre, les deux classes c_1 , c_2 et le type de distance utilisé. Il se déroule comme suit :

Entrée : Deux classes c_1 et c_2 , Type de distance \in {lien simple, lien moyen, lien complet}

```
Si (type = lien simple)
    Distance = Minimum(Distance Minkowski( $o_i$ ,  $o_j$ , 2)) où  $o_i \in c_1$ ,  $o_j \in c_2$  ;
Sinon
    Si (type = lien complet)
        Distance = Maximum(Distance Minkowski( $o_i$ ,  $o_j$ , 2)) où  $o_i \in c_1$ ,  $o_j \in c_2$  ;
    Sinon
        Distance = Moyenne(Distance Minkowski( $o_i$ ,  $o_j$ , 2)) où  $o_i \in c_1$ ,  $o_j \in c_2$  ;
Retourner Distance ;
```

Sortie : Distance entre c_1 et c_2

Algorithme 2 : Distance entre deux classes

2.3.3. Inertie

Inertie d'une population :

L'inertie d'une population mesure l'homogénéité de ses individus. Elle est calculée, en utilisant la distance de Minkowski avec degré de pondération égal à 2 (distance euclidienne). L'algorithme est le suivant :

Entrée : Ensemble d'individus o_i , centre de gravité g

Inertie = Somme (Distance de Minkowski ($g, o_i, 2$)) ; // 2 est le degré de pondération

Retourner Inertie ;

Sortie : Inertie de la population d'individus $\{o_i\}$

Algorithme 3 : Inertie d'une population

Inertie intra-classe d'une partition :

L'inertie intra-classe est définie pour une partition. Elle mesure le rapprochement des individus qui appartiennent à la même classe. L'algorithme de calcul est le suivant :

Entrée : Ensemble d'individus de la partition o_i , centres de gravité des classes g_k

Inertie = \sum_k Somme (Inertie de la population (o_j, g_k)) ; // tq o_j Classe c_k

Retourner Inertie ;

Sortie : Inertie intra-classe de la partition p

Algorithme 4 : Inertie intra-classe

Inertie inter-classe d'une partition :

L'inertie inter-classe est définie pour une partition. Elle permet de mesurer l'éloignement entre les classes. L'algorithme de calcul est le suivant :

Entrée : Centres de gravité des classes g_k

Soit la population t composée des centres g_k ;

Calculer g le centre de gravité de t ;

Retourner Inertie de la population (g, t) ;

Sortie : Inertie inter-classe de la partition p

Algorithme 5 : Inertie inter-classe

2.3.4. Erreur de classification

Cette entité mesure le taux d'erreur d'un algorithme de classification (sa fiabilité). Elle est calculée sur la base d'une population de test qui comporte des individus étiquetés. Les individus sont alors re-classifiés par l'algorithme puis l'erreur est calculée comme étant la portion des individus incorrectement classifiés.

$$\frac{\sum_{i=1}^n \text{Erreur}_i}{\sum_{i=1}^n \text{Individus}_i} \quad [34]$$

2.4. Module d'affichage

Le module d'affichage permet de visualiser les données et les résultats. Les techniques de visualisation diffèrent selon les données que l'on veut représenter. En effet, des travaux en intelligence artificielle [33] ont montré qu'il n'existe que trois formes de représentation des connaissances, en l'occurrence, le tableau, le graphe et l'arbre. Le data mining, n'échappant pas à la règle, nous nous servons des trois moyens pour la visualisation des données et des résultats.

2.4.1. Visualisation des données

La population d'individus est visualisée dans un tableau bidimensionnel qui comporte les individus en lignes et les attributs en colonnes (Figure 14). Une telle représentation a l'avantage d'être claire et facilement lisible. Pour mieux connaître les données sur lesquelles on travaille, nous avons enrichi cette représentation avec les fonctionnalités suivantes :

- Possibilité de trier les individus selon la valeur d'un attribut. Cela permet de repérer facilement les valeurs aberrantes (bruits) et, éventuellement, les éliminer ;
- Possibilité de rajouter, supprimer ou modifier un individu.

Renommer :

Imprimer

+ -

	ID	CLASSE	RESIDANT	SERIEBAC	ANALD	O
0	00/0008	-1	0	4	8,42	9,
1	00/0011	-1	0	4	11,58	11
2	00/0016	-1	0	4	8,17	13
3	00/0020	-1	0	4	9,33	11
4	00/0040	-1	0	5	8,17	11
5	00/0052	-1	0	5	7,6	10
6	00/0055	-1	0	5	10	9
7	00/0091	-1	0	7	8,5	9,
8	00/0093	-1	0	4	9,71	11
9	00/0099	-1	0	4	8,42	11
10	00/0100	-1	0	4	9,83	12

Figure 14 : Visualisation des données dans un tableau

2.4.2. Visualisation des partitions

Les résultats de clustering sont visualisés par projection sur le plan. La projection se fait soit par l'ACP, soit sur deux attributs (à la fois). Les individus de la même classe sont ensuite colorés par la même couleur. La figure ci-dessous est un exemple de représentation graphique d'une partition par la technique de projection ACP.

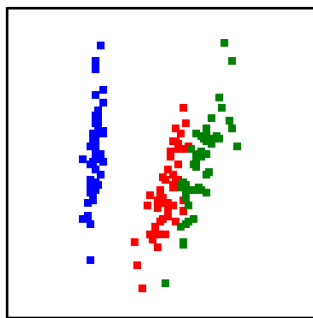


Figure 15 : Visualisation graphique d'une partition par la technique de projection ACP

2.4.3. Visualisation de l'arbre de décision

L'arbre de décision est représenté sous forme graphique (Figure 16) par un arbre où :

- Les nœuds internes (correspondants à des tests) sont étiquetés par l'attribut à tester ;
- De chaque nœud, partent deux branches, la première correspond à un test positif (valeur de l'attribut supérieur à la valeur de coupure), la deuxième correspond à un test négatif ;
- Les feuilles sont étiquetées, chacune, par les numéros des classes correspondantes et les degrés d'appartenances à chaque classe.

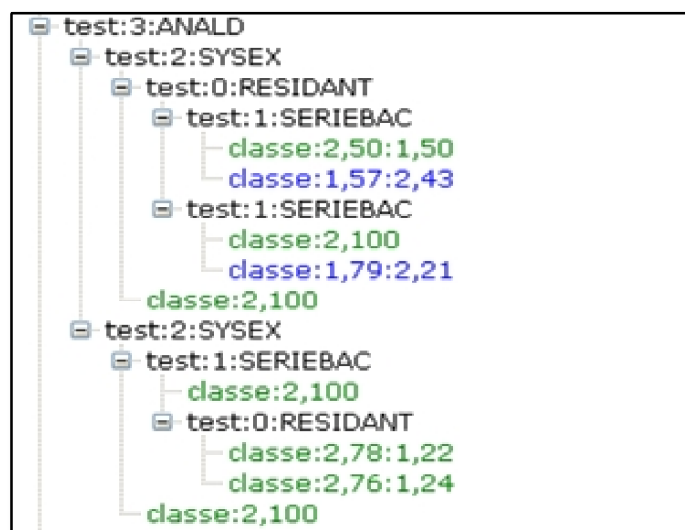


Figure 16 : Représentation graphique d'un arbre de décision

2.5. Module de sauvegarde

Ce module permet d'enregistrer les résultats et les rapports d'exécution des algorithmes, notamment les résultats de clustering, dans des fichiers pour consultation ultérieure.

3. Mise en Œuvre des algorithmes

Dans ce qui suit, nous présentons la mise en œuvre des algorithmes de data mining, ceux-ci étant répartis dans trois tâches : le clustering, la classification et les règles d'association.

3.1. Le clustering

Rappelons que l'objectif de cette étape est de synthétiser les données de départ dans des groupes de données homogènes (classes). Parmi les algorithmes de clustering énoncés dans le Chapitre II, nous avons opté pour des algorithmes de partitionnement (k-Means, PAM et CLARA), un algorithme hiérarchique ascendant et l'algorithme génétique.

3.1.1. L'algorithme k-Means

L'algorithme k-Means est considéré comme un standard dans le domaine de clustering. Sa simplicité de mise en œuvre et sa capacité de traiter des populations de tailles importantes font de k-Means un algorithme qui répond parfaitement à un grand nombre de problèmes.

L'algorithme prend en paramètre k , le nombre de classes voulu, et se résume comme suit [24] :

Entrée : Population de m individus, k

Choisir aléatoirement k centres initiaux c_1, \dots, c_k

Répéter

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

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

Sortie : Partition.

Algorithme 6 : L'algorithme k-Means [24]

Pour calculer la distance entre les individus, nous avons utilisé la distance de Minkowski avec un degré de pondération égal à 2 (distance euclidienne).

3.1.2. L'Algorithme PAM

PAM est un algorithme qui recherche les médoides des classes ; chaque classe étant représentée par son médoide.

L'algorithme prend en paramètre k , le nombre de classes voulu, et se résume comme suit [25] :

Entrée : Population d'individus $\{O_{ij}\}$, k

Choisir arbitrairement k médoides.

Répéter

 Affecter chaque individu restant au médoide le plus proche.

 Choisir aléatoirement un non-médoide O_r

 Pour chaque médoide O_j

 Calculer le coût TC (valeur de la fonction objectif) du remplacement de O_j par O_r

 Si (TC diminue)

 Remplacer O_j par O_r

 Calculer la nouvelle partition.

 Finsi

 FinPour

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

Sortie : Partition.

Algorithme 7 : L'algorithme PAM [25]

La fonction objectif mesure la qualité de la partition obtenue. Sa formule est donnée dans le Chapitre II (voir 2.4.1.2).

3.1.3. L'Algorithme CLARA

CLARA est un algorithme qui applique PAM sur plusieurs échantillons aléatoires. Il prend en paramètre la taille de l'échantillon *tailleEchantillon*, le nombre d'essais *NbEssais* et le nombre de classes k et se déroule comme suit :

Entrée : Population d'individus $\{O_i\}$, *tailleEchantillon*, *NbEssais*, *k*

CoutMinimum = $+\infty$;

Pour *i* allant de 1 à *NbEssais*

 Choisir un échantillon aléatoire *e* de taille égale à *tailleEchantillon*

p = Partition résultant de l'application de PAM sur *e* ;

 Calculer le cout *c* de la partition résultante *p*

 Si (*c* < CoutMinimum)

 Meilleure partition = *p* ;

Retourner Meilleure partition ;

Sortie : Partition.

Algorithme 8 : L'Algorithme CLARA [17]

3.1.4. L'Algorithme hiérarchique ascendant

Cet algorithme commence par considérer chaque individu dans une classe et procède à la construction de nouvelles classes par fusion des deux classes les plus proches (en terme de distance). Il prend, en paramètre, le type de distance inter-classe et ressemble à ceci :

Entrée : Population d'individus *p*, Mesure de distance entre classes *d* \in {lien simple, lien moyen, lien complet}

Initialiser les classes (chaque individu dans une classe).

Répéter

 Fusionner les deux classes les plus proches (selon la distance *d*) en une seule.

Jusqu'à atteindre un critère d'arrêt.

Sortie : Partition.

Algorithme 9 : L'Algorithme hiérarchique ascendant

Les critères d'arrêt que nous avons envisagés sont :

- Atteindre un certain nombre de classes ;
- Tous les individus appartiennent à la même classe (aller jusqu'au bout).

3.1.5. L'algorithme génétique

La famille des algorithmes génétique est une modélisation de l'évolution naturelle pour résoudre un problème de recherche, dans notre cas un problème de clustering. Les solutions potentielles sont codées sous forme de séquences, le plus souvent des chaînes de bits, appelées chromosomes ou génotypes. À partir d'une population initiale générée aléatoirement, de nouvelles populations sont générées itérativement par application d'opérateurs génétiques de sélection, croisement et mutation jusqu'à satisfaction d'un critère d'arrêt.

Les algorithmes génétiques suivent tous le schéma de fonctionnement illustré par la Figure 17.

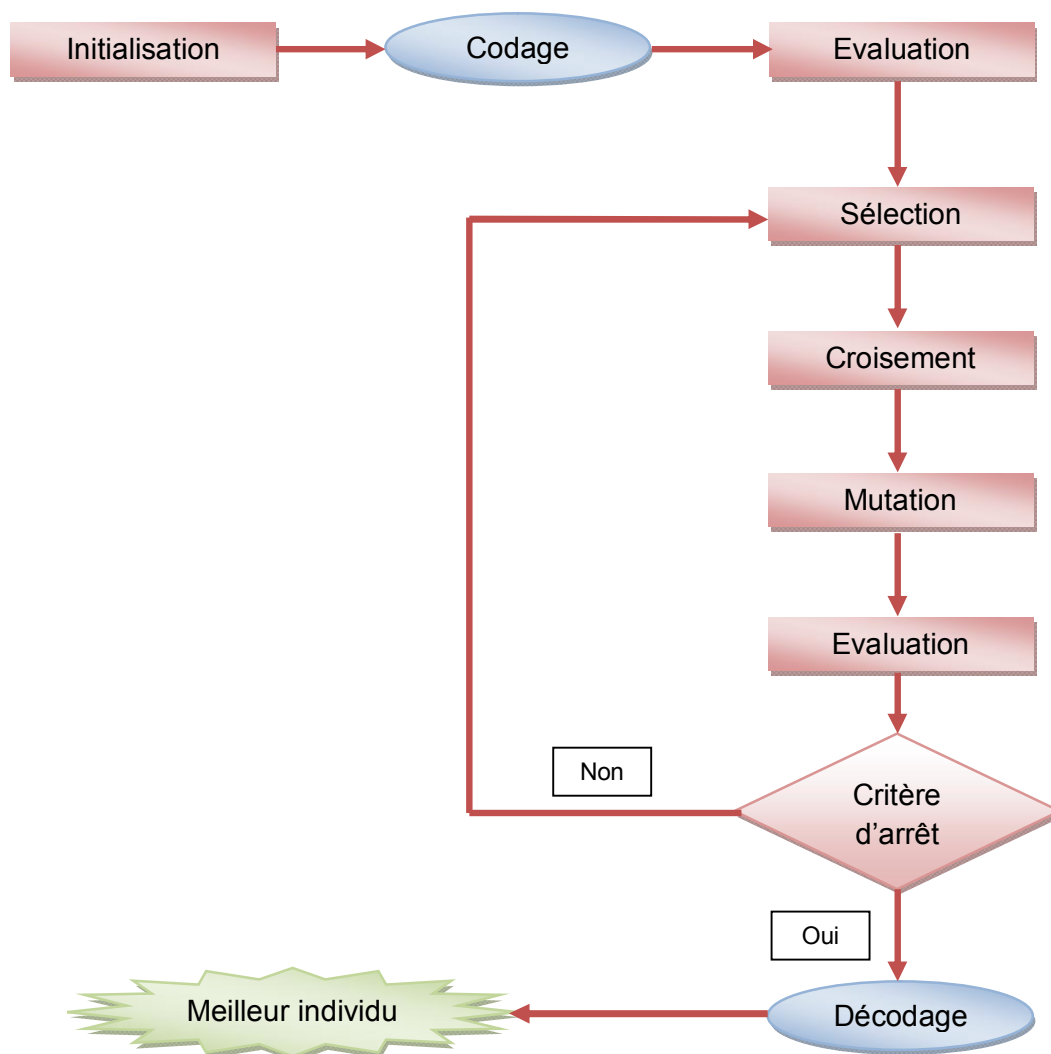


Figure 17 : Déroulement de l'algorithme génétique [35]

3.1.5.1. L'initialisation

Tout d'abord, on crée la première population de solutions potentielles qui sera amenée à évoluer. On définit ainsi à cette étape la taille N de cette population qui doit être constante tout au long de l'exécution de l'algorithme.

Dans le cas du clustering, l'initialisation se fait soit aléatoirement (étiquetage aléatoire des individus), soit comme étant les N résultats d'exécution d'un algorithme simple de clustering (k-Means) que l'on souhaite améliorer par la suite.

Chacune des solutions potentielles qui constituent la population initiale doit être représentée sous forme codée (chromosome). La difficulté de l'algorithme réside en fait dans le choix d'une fonction du codage des partitions. En effet, il existe plusieurs techniques de codage des partitions parmi lesquelles nous avons choisis les deux techniques suivantes car elles sont les plus utilisées [36] :

- Codage par classe ;
- Codage par vecteur.

Codage par classe :

Dans cette approche, chaque solution est un tableau où chaque case contient un individu et la classe y correspondante (Figure 18). Cette représentation a l'avantage de simplifier les opérateurs de croisement et de mutation. Elle est néanmoins inadaptée aux grandes populations.

o_1	o_2	o_3	o_n
Classe 3	Classe 5	Classe 3			Classe 2

Figure 18 : Codage de partitions par classe

Codage par vecteur :

Cette représentation considère le chromosome comme étant les vecteurs centroides des classes. L'affectation des objets se fait par le principe de k-Means (chaque objet est affecté au centre le plus proche). Cette représentation, légère, est adaptée aux grandes populations.

3.1.5.2. L'évaluation

L'évaluation d'une solution se fait par le biais d'une fonction d'évaluation, également appelée fonction objectif ou encore fitness, qui mesure sa qualité. La construction de cette fonction doit être particulièrement optimisée car elle sera exécutée un grand nombre de fois, jusqu'à N fois à toutes les générations, ce qui fait que la rapidité de l'algorithme dépend essentiellement d'elle [35].

La fonction d'évaluation doit aussi pouvoir tenir compte des solutions invalides si le problème doit satisfaire des contraintes que les opérateurs de mutation et croisement ne respectent pas.

Dans un problème de clustering, il existe plusieurs fonctions d'évaluation parmi lesquelles, nous optons pour les deux suivantes car elles sont parmi les plus populaires [36] :

- Inertie intra-classe (à minimiser) ;
- Inertie intra-classe / Inertie inter-classe (à minimiser) ;

3.1.5.3. La sélection

L'objectif de cette étape est de ne garder que les individus les plus performants. Parmi les techniques de sélection les plus utilisées, on trouve [35] :

- La sélection déterministe ;
- La sélection par roulette.

La sélection déterministe :

On sélectionne toujours les meilleurs individus et on écarte totalement les plus mauvais. Cela suppose un tri de l'ensemble des solutions selon leurs fitness.

La sélection par roulette (*Roulette Wheel Selection*) :

Chaque individu a une chance d'être sélectionné proportionnelle à sa performance (fitness). Pour utiliser l'image de la roulette, chaque individu possède une portion de la roulette dont l'angle dépend de son adaptation. On fait tourner la roulette et on sélectionne l'individu qui possède la case où elle est tombée (Figure 19).

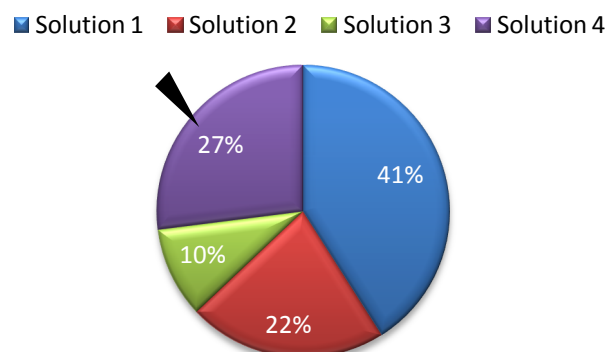


Figure 19 : Roulette de sélection

Cette technique est mise en œuvre en associant un intervalle de valeurs entre 0 et x à chaque solution, x étant la probabilité que la solution soit sélectionnée. La largeur de l'intervalle est donc proportionnelle à la performance de la solution (Figure 20).

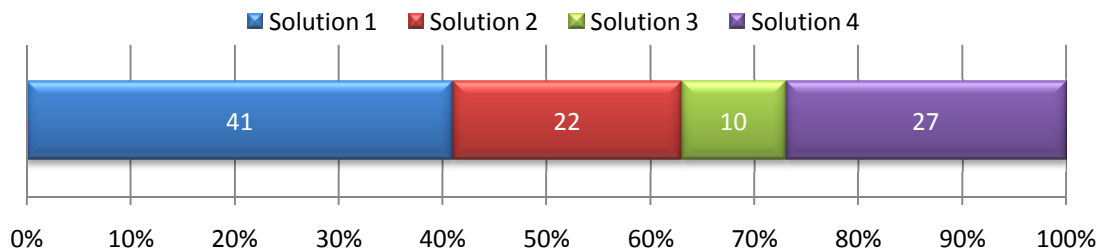


Figure 20 : Intervalles de sélection

La mise en œuvre de cette technique se fait par la réalisation d'une variable aléatoire x de loi uniforme entre 0 et 100. La valeur de x détermine l'individu qui sera sélectionné selon l'algorithme suivant :

Entrée : L'ensemble des N individus (solutions) S , Leurs fitness respectives F

Calculer les fitness proportionnelles (en %) à partir de F dans Fq

Trier le tableau Fq

Effectuer une nouvelle réalisation de x

Chercher i tel que $Fq[i-1] < x < Fq[i]$

Retourner L'individu correspondant à $Fq[i]$;

Sortie : L'individu sélectionné.

Algorithme 10 : L'algorithme de sélection par la roulette

L'algorithme ci-dessus est utilisé pour sélectionner un seul individu, il est donc exécuté autant de fois que d'individus à sélectionner.

3.1.5.4. Le croisement

Une fois certains individus sélectionnés, on les fait se croiser entre eux. Pour cela on utilise l'opérateur de croisement. C'est l'opérateur essentiel de recherche d'un algorithme génétique [35]. Il combine les chromosomes de quelques individus pour en obtenir de nouveaux. Cette opération est effectuée avec une probabilité (probabilité de croisement). Les techniques de croisement diffèrent selon le codage utilisé :

- Codage par classe : Le croisement se fait en un ou plusieurs points de coupure ;
- Codage par vecteur : Le croisement se fait par la moyenne des vecteurs.

Croisement en un point de coupure :

On choisit au hasard un point de coupure identique sur les deux chromosomes parents (site de croisement) et on échange les fragments situés après le point de coupure pour donner les deux nouveaux chromosomes enfants (Figure 21).

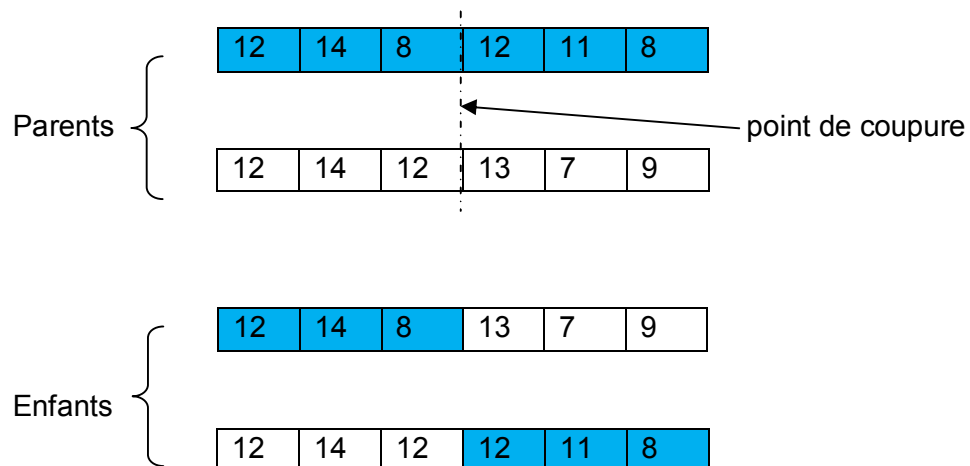


Figure 21 : Croisement en un point de coupure

Croisement en k points de coupure :

Généralisation à k points de coupure de la méthode précédente (Figure 22).

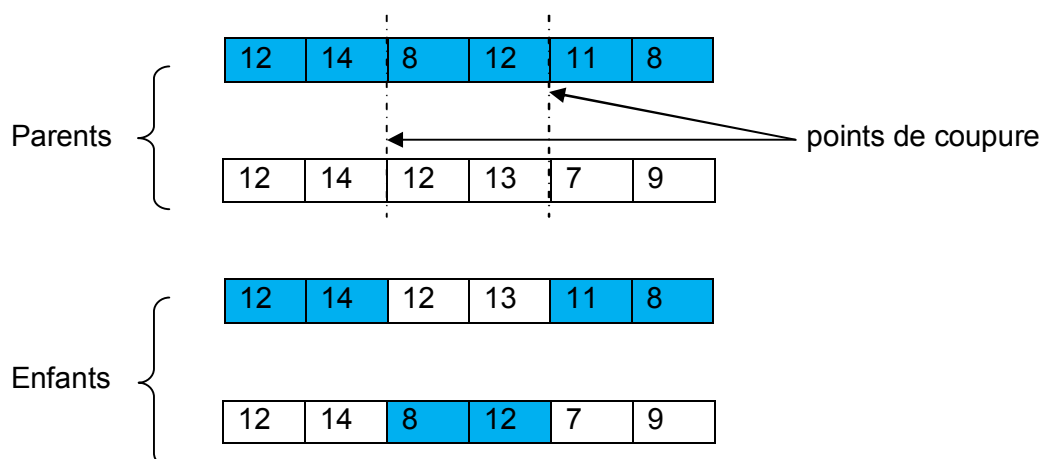


Figure 22 : Croisement en deux points de coupure

Croisement par la moyenne des vecteurs :

Dans le cas du codage par vecteur, un autre type de croisement est envisagé. Il s'agit de calculer la moyenne des vecteurs parents. Cette technique ne génère qu'un seul chromosome fils.

Exemple : Soient les deux vecteurs parents à croiser : (12.5, 9.45, 21) et (11.33, 19.66, 11.46). Le vecteur fils résultant est : (11.91, 14.55, 16.23).

3.1.5.5. La mutation

L'opérateur de mutation modifie aléatoirement la valeur de certains bits d'un chromosome avec une faible probabilité [35]. L'intérêt de cet opérateur est d'éviter une dérive génétique ; certains gènes favorisés par le hasard peuvent se répandre au détriment des autres et sont ainsi présents au même endroit sur tous les chromosomes. La mutation donne alors la chance aux solutions potentielles de s'approcher de l'optimum global d'autant que le permet la précision du codage.

On a alors principalement le croisement pour explorer globalement et entièrement l'espace de recherche et la mutation pour la recherche locale et l'optimisation de solutions déjà utilisables.

Pour le clustering, il existe deux méthodes pour réaliser la mutation selon le type du codage utilisé :

- Codage par classe : La mutation consiste à modifier aléatoirement la classe d'un individu choisi de façon aléatoire (Figure 23).

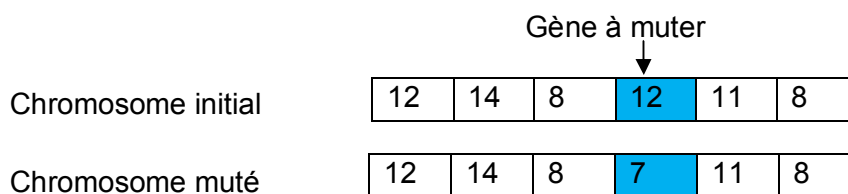


Figure 23 : Exemple de mutation

- Codage par vecteur : La mutation consiste à rajouter à chaque composante du vecteur une réalisation d'une loi normale centrée et d'écart-type fixé par l'utilisateur (mutation gaussienne).

3.1.5.6. Le critère d'arrêt

L'algorithme s'arrête quand le meilleur individu d'une génération dépasse un certain stade de performance ou alors quand un certain nombre d'individus sont identiques à la solution locale optimale (convergence partielle) [35].

Pour résoudre le problème de clustering, les deux critères d'arrêt suivants sont envisagés :

- Atteindre un certain seuil de qualité ;
- Atteindre un certain nombre d'itérations.

3.2. La classification

Rappelons que l'objectif de cette étape est d'associer une classe à un individu nouvellement présenté en s'appuyant sur un ensemble d'apprentissage (population d'individus étiquetés). Parmi les algorithmes de classification énoncés dans le chapitre II, nous avons opté pour les deux algorithmes suivants :

- L'algorithme des k plus proches voisins k -NN ;
- L'arbre de décision.

3.2.1. L'algorithme k -NN

Cet algorithme utilise la recherche de cas similaires déjà résolus pour prendre une décision. Il prend en paramètre k , le nombre de voisins, et un ensemble d'apprentissage X de m individus étiquetés et se déroule comme suit :

Entrée : L'ensemble X , un individu y à classer, nombre de voisins k

- Déterminer les k plus proches individus de y , dans X , en terme de distance.
- c est la classe la plus représentée dans les voisins.

Sortie : La classe de y est c .

Algorithme 11 : L'algorithme k -NN [5]

3.2.2. L'arbre de décision

Parmi les algorithmes de construction des arbres de décision évoqués dans le Chapitre II, ID3 est un algorithme basique qui procède à la construction de l'arbre de manière récursive en choisissant, à chaque fois, le meilleur attribut qui sépare mieux les individus. La procédure récursive ID3 prend en entrée l'ensemble d'apprentissage S et se déroule comme suit [37] :

Entrée : Population d'individus étiquetés S

Si (Tous les individus de S appartiennent à la même classe)
 | Créer une feuille portant le nom de cette classe ;
 Sinon
 | Choisir le meilleur attribut pour créer un nœud ;
 | Le test associé à ce nœud sépare S en deux parties : S_p et S_n
 | ID3 (S_p).
 | ID3 (S_n).
 Finsi

Sortie : Arbre de décision.

Algorithme 12 : L'algorithme ID3 [37]

Cet algorithme est plus adapté à des données catégoriques. Cependant, il peut fonctionner avec des données numériques en procédant à la discrétisation des valeurs. Pour cela, une valeur de coupure est fixée pour chaque attribut. Le test est effectué pour chaque attribut par rapport à cette valeur. Le test est positif si l'attribut se situe au dessus de la valeur de coupure. Autrement, il est négatif.

La sélection du meilleur attribut dans ID3 revient à maximiser une entité dite "gain d'information" [37] qui repose sur le concept de l'entropie. Pour un ensemble S d'objets, le gain d'information relatif à l'attribut A de l'ensemble d'objets S est défini par la formule suivante :

$$G(S, A) = H(S) - \sum_{i=1}^n \frac{P_i}{|S|} H(S_i) \quad [37]$$

Où :

$$H(S) = - \sum_{i=1}^n \frac{P_i}{|S|} \log_2 \frac{P_i}{|S|}$$

Tel que P_i = nombre d'éléments dans la classe C_i .

Et :

$$H(S_v) = - \sum_{s \in S_v} \frac{1}{|S_v|} \log_2 \frac{1}{|S_v|}$$

Tel que $S_v = \{s \in S \mid A(s) = v\}$; sous-ensemble de S où l'attribut A prend la valeur v.

L'attribut qui sépare mieux les objets est celui qui maximise le gain d'information, ce qui revient à privilégier les attributs ayant un grand nombre de valeurs.

La classification floue avec les arbres de décision :

La classification floue consiste à affecter l'objet à classer à plusieurs classes avec différents degrés d'appartenance au lieu de l'affecter à une seule classe de manière définitive.

Les capacités de classification des arbres de décision peuvent être étendues pour effectuer une classification floue, et ce, en apportant la modification suivante à l'algorithme ID3 :

Si tous les attributs sont testés sans pour autant que les objets ne soient tous de même classe, on crée une feuille contenant toutes les classes présentes dans l'ensemble d'objets X avec des degrés d'appartenance proportionnels à leurs effectifs dans X .

Exemple :

Soit l'ensemble X des couples (*objet*, *classe*) suivant :

$X = \{(o_1, c_0), (o_2, c_0), (o_3, c_1)\}$;

Les degrés d'appartenance sont de 67% pour c_0 et de 33% pour c_1 .

3.3. Les règles d'association

Parmi les algorithmes de recherche d'associations évoqués dans le chapitre II, L'algorithme Apriori est considéré comme un algorithme de référence. En effet, la plupart des algorithmes de recherche des règles d'association sont basés sur lui [33].

L'algorithme Apriori prend en paramètre la population d'individus D et le seuil minimum de support $MinSup$ et se résume comme suit :

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

$k=0$, $L_k = \emptyset$.

C_1 = l'ensemble des 1-itemsets dans D .

L_1 = l'ensemble des 1-itemsets fréquents de C_1 .

Tant que (L_{k+1}) est non vide

C_{k+1} = $(k+1)$ -itemsets candidats générés à partir de (L_k)

L_{k+1} = ensemble des $(k+1)$ -itemsets fréquents de C_{k+1} (support > SupMin)

$k = k+1$;

Retourner : $\cup L_k$

Sortie : L'ensemble de tous les itemsets fréquents.

Algorithme 13 : L'algorithme Apriori [32]

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

Entrée : Ensemble des k -itemsets L_k

Extraire tous les éléments contenus dans tous les k -itemsets de L_k dans un tableau t

Pour chaque k -itemset i dans L_k

 Pour chaque élément e dans t

 Si (i ne contient pas e parmi ses éléments)

 Rajouter e à i ; (i devient maintenant un $(k+1)$ -itemset)

Sortie : Un ensemble de $(k+1)$ -itemsets

Algorithme 14 : L'algorithme de génération des itemsets candidats

Exemple :

Soit les 3-itemsets $i_1 = \{1, 2, 3\}$, $i_2 = \{1, 2, 4\}$.

Les éléments extraits à partir de i_1 et i_2 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 , le nombre des sous-items est égal à $\sum_{j=1}^{k-1} \binom{k}{j}$;
2. Générer toutes les implications possibles entre les éléments de t (règles potentielles); Pour chaque itemset fréquent de t de taille n , le nombre des implications générées est égal à $2^{\binom{n}{2}} - n$;
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 : $\{1\} \rightarrow \{2\}$ (confiance $> \text{ConfMin}$), $\{2\} \rightarrow \{1\}$ (confiance $< \text{ConfMin}$).

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

4. Environnement de mise en Œuvre

Nous avons développé notre application sous la plateforme "Microsoft © Visual Studio 2008" Version 9.0.21022.8 RTM. Comme langage de programmation, nous avons opté pour C#, et ce pour les raisons suivantes :

- C'est un langage orienté objet, ce qui facilite énormément notre travail ;
- C'est un langage simple et efficace ;
- Sa richesse en bibliothèques offertes par la plateforme .NET rend les tâches de manipulation des fichiers, création des interfaces graphiques et gestion des structures de données... beaucoup plus aisées. Ceci représente pour nous un atout intéressant.

4.1. Les classes d'objets

La classe "Individu" :

C'est la classe qui représente un individu. Elle est composée des champs suivants :

- Un identificateur "id" de type chaîne de caractères qui désigne chaque individu de manière unique ;
- Un tableau dynamique "attributs" qui contient les valeurs numériques réelles des différents attributs de l'individu, entre autres sa classe ;

Cette classe regroupe les méthodes suivantes :

- Méthodes d'accès et de modification ;
- Méthode de calcul de distance entre deux individus (distance de Minkowski).

La classe "Population" :

Cette classe représente une population d'individus. Elle contient les champs suivants :

- Un tableau dynamique "individus" qui représente l'ensemble des individus de la population ;

Cette classe regroupe les méthodes suivantes :

- Méthode de calcul de l'inertie d'une population ;
- Méthode "DiviserPopulation" pour diviser la population en deux sous-populations : positive et négative selon la valeur d'un attribut (utilisée dans l'algorithme ID3) ;
- Méthodes de sauvegarde et de chargement à partir des fichiers.

La classe "Cluster" :

Cette classe représente un cluster (une classe). Elle comporte les champs suivants :

- Un identificateur "numéro" de type entier qui indique le numéro de la classe ;
- Un champ "nombreIndividus" qui renseigne le nombre d'individus dans la classe ;
- Un tableau dynamique "individus" qui représente l'ensemble des individus de la classe ;
- Un champ "centre" de type Individu qui est le représentant de la classe (centroïde ou médioïde) ;
- Une chaîne de caractère "label" qui indique le nom de la classe.

Cette classe regroupe les méthodes suivantes :

- Méthodes de calcul de distances entre les classes (lien simple, lien moyen, lien complet) ;
- Méthode de calcul du centre de gravité du cluster ;
- Méthode de fusion de deux classes en un seul (utilisée dans l'algorithme hiérarchique ascendant).

La classe "Partition" :

Cette classe représente une partition. Elle comporte les champs suivants :

- Un tableau dynamique "clusters" qui regroupe l'ensemble des clusters ;
- Un entier "effectif" qui représente le nombre d'individus dans la partition ;
- Une chaîne de caractère "label" qui indique le nom de la partition.

Cette classe regroupe les méthodes suivantes :

- Méthodes de calcul de l'inertie (inter-classe et intra-classe) de la partition ;
- Méthodes de sauvegarde et de chargement à partir des fichiers ;

La classe "KMeans" :

Cette classe permet de faire appel à l'algorithme k-Means. Elle contient parmi ses membres les paramètres de l'algorithme.

La classe "PAM" :

Cette classe permet d'appliquer l'algorithme PAM. Elle contient parmi ses membres les paramètres de l'algorithme.

La classe "CLARA" :

Cette classe permet d'appliquer l'algorithme CLARA à un ensemble d'individus. Elle contient parmi ses membres les paramètres de l'algorithme. Elle contient, également, une méthode d'échantillonnage de la population de façon aléatoire.

La classe "HierarchiqA" :

Cette classe permet d'appliquer l'algorithme hiérarchique ascendant à une population d'individus. Elle contient les champs suivants :

- Un entier "nbClusterMin" qui représente le critère d'arrêt (nombre minimum de clusters) ;
- Une variable "métrique" qui indique la distance inter-classe utilisée (lien simple, lien moyen, lien complet) ;

Cette classe est dotée d'une méthode qui permet d'initialiser la configuration (chaque individu dans une classe).

La classe 'Chromosome' :

Cette classe permet de représenter une solution potentielle au problème génétique (partition) sous forme codée (chromosome). Ce chromosome peut être de deux types définis chacun par une classe qui hérite de la classe mère "Chromosome" :

- "ChromoClasse" qui représente une partition codée par classe.
- "ChromoCentre" qui représente une partition codée par vecteur centroïde ;

La classe "AG" :

Cette classe permet d'appliquer l'algorithme génétique pour partitionner un ensemble d'individus. La classe comporte les champs suivants :

- Un entier "n" qui représente la taille de la population de solutions ;
- Un tableau dynamique de taille n qui contient les solutions potentielle aux problèmes codés sous forme de chromosomes ;
- Un tableau dynamique de taille n qui contient les fitness respectifs des solutions.

La classe AG est dotée des méthodes suivantes :

- Méthodes de codage et de décodage des solutions (conversion des chromosomes en partitions et vice-versa) ;
- Méthodes d'initialisation de la population initiale de solutions (aléatoire, par k-Means) ;
- Méthodes d'évaluation des solutions (inertie intra-classe, inertie intra-classe/inertie inter-classe) ;
- Méthodes de sélection (déterministe, par roue de loterie) ;
- Méthodes de croisement (en un ou plusieurs points, par le vecteur moyen) ;
- Méthodes de mutation (mutation aléatoire, mutation gaussienne).

La classe "KNN" :

Cette classe permet d'appliquer l'algorithme k-NN. Elle contient parmi ses membres un entier "k" qui renseigne le nombre de voisins.

Cette classe est dotée d'une fonction qui permet de calculer les voisins d'un individu dans une population. Cela se fait comme suit :

- Trier les individus de l'ensemble d'apprentissage selon leurs distances de l'individu à classer ;
- Prendre les "K" premiers individus.

La classe “ArbreD” :

Cette classe regroupe tous les algorithmes applicables pour la construction de l'arbre de décision et la classification des individus. Elle est dotée des méthodes suivantes :

- Méthodes de calcul de l'entropie et du gain d'information (utilisées dans l'algorithme ID3) ;
- Méthode récursive “ID3” de construction de l'arbre.
- Méthode qui permet de calculer les degrés d'appartenance pour effectuer une classification floue. Cette méthode est également utilisée pour calculer la classe majoritairement représentative dans une population d'individus.
- Méthode récursive “Classifier” qui permet de classer un individu en utilisant l'arbre déjà construit.

La classe “APriori” :

Cette classe permet d'appliquer l'algorithme Apriori sur une population d'individus. Les paramètres “SupMin”, “ConfMin” sont des membres de la classe. Parmi les méthodes de cette classe :

- Méthode de génération des 1-itemSets ; elle est utilisée pour initialiser l'espace de recherche ;
- La méthode “EliminerItemSetsSupportFaible” dont le rôle est d'éliminer les itemsets qui ne respectent pas la contrainte sur le support ;

La classe “ItemSet” :

Cette classe représente un itemset qui correspond à un ensemble d'attributs. Elle est dotée de méthodes qui assurent les fonctions suivantes :

- Calculer le support de l'itemset ;
- Tester si l'itemset est fréquent ou pas ;
- Générer les $(k+1)$ -itemsets à partir d'un k -itemset ;
- Générer tous les sous-itemsets d'un itemset ;

La classe “Regle” :

Cette classe représente une règle d'association. Elle contient parmi ses membres :

- Deux variables de type ItemSet qui définissent, respectivement, l'antécédent et le conséquent de la règle ;
- Deux valeurs qui indiquent le support et la confiance de la règle.

4.2. L'interface utilisateur

La fenêtre principale de l'application est représentée par la Figure 24. Elle permet l'accès à d'autres fenêtres pour simplifier et optimiser les tâches.

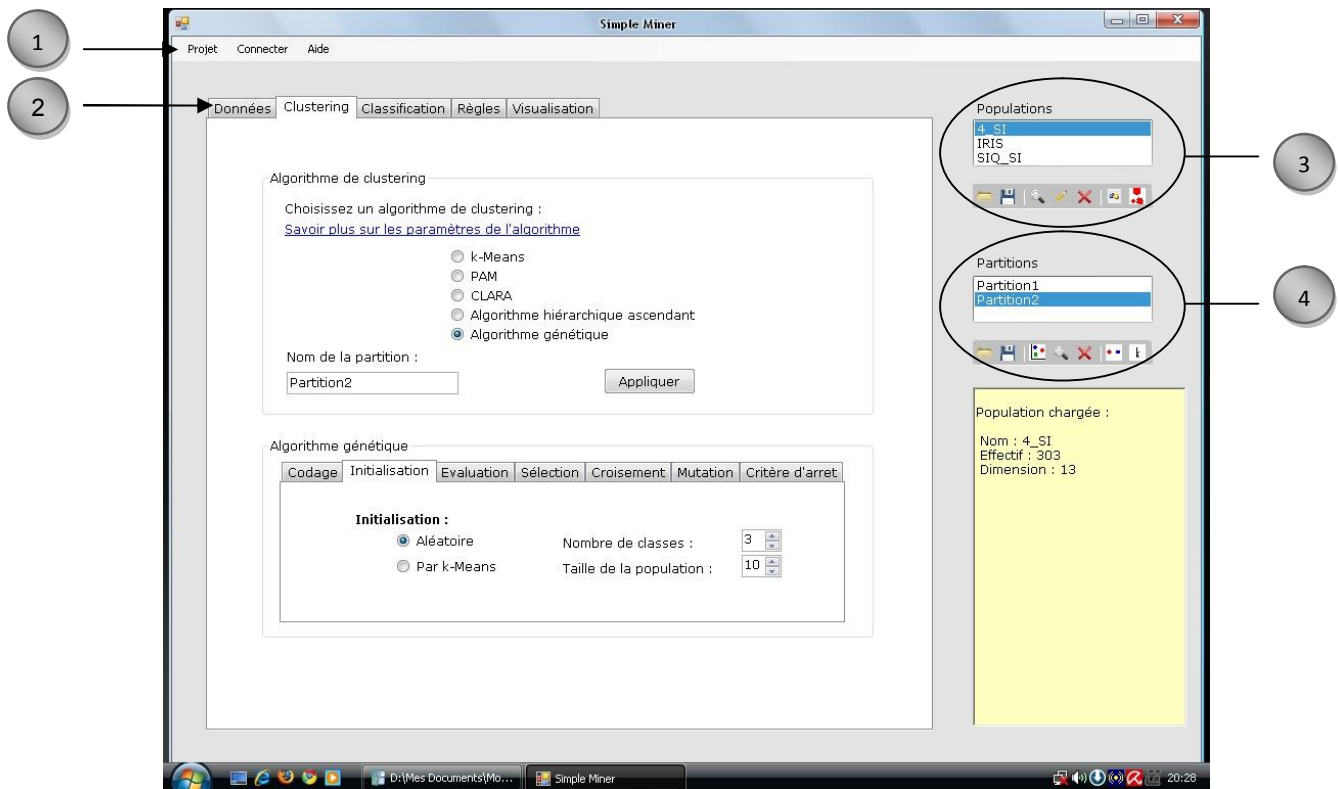


Figure 24 : Fenêtre principale de l'interface utilisateur

Figure 24.1 : Menu général.

Le menu général permet de créer, ouvrir et enregistrer un projet. Il permet également de personnaliser l'affichage, ce qui simplifie et optimise les tâches ;

Figure 24.2 : Onglets de navigation.

Les cinq onglets de navigation permettent de basculer entre les différentes étapes de data mining : Chargement de données, clustering, classification, règles d'association et visualisation ;

Figure 24.3 : Liste des populations.

Toutes les populations chargées sont conservées dans la liste des populations. On peut, donc, facilement basculer entre les populations en utilisant cette liste.

Figure 24.4 : Liste des partitions.

La liste des partitions contient toutes les partitions chargées ou obtenus en appliquant un algorithme de clustering sur une population.

5. Conclusion

Nous avons présenté, dans ce chapitre, l'architecture générale de notre application avant de détailler la structure et le fonctionnement des différents modules. Nous avons, ensuite, présenté la mise en œuvre de nos algorithmes de data mining.

Parmi ces algorithmes, l'algorithme génétique s'avère jouir d'une bonne réputation dans la résolution des problèmes d'optimisation. Bien que cet algorithme ait montré son efficacité face au problème de clustering, il présente quelques petites difficultés de mise en œuvre comme l'implémentation des opérations génétiques, notamment le croisement.

La mise en œuvre des algorithmes a été énormément facilitée par l'adoption de la programmation orientée objet. Nous avons utilisé une dizaine de classes différentes, chacune regroupe des membres et des méthodes. Pour chaque algorithme, nous avons consacré une classe qui permet de paramétrer l'algorithme et y faire appel. Nous avons également prévu une classe pour la population et une classe pour la partition, ce qui permet de mieux gérer les individus et les classes d'individus.

L'utilisation des bibliothèques .NET a également facilité notre travail. En effet, l'interface utilisateur, la manipulation des fichiers, la connexion des bases de données et d'autres tâches sont devenues très simples à réaliser, ce qui contribue à augmenter la rentabilité de notre application.

Chapitre IV : Tests et résultats

1. Introduction

Afin de tester et comparer l'efficacité et la performance des algorithmes, nous allons effectuer une série d'essais sur tous les algorithmes programmés avec différents paramètres que nous résumons dans ce chapitre. Pour chaque tâche, deux types de tests seront effectués : des tests sur les algorithmes et des tests sur les benchmarks.

Nous allons tester, en premier lieu, les algorithmes de clustering. Une série de tests individuels (sur chacun des algorithmes) seront effectués dans le but de déterminer les meilleurs paramètres pour chaque algorithme. Les performances (temps d'exécution, qualité des résultats...) seront ensuite mesurées pour tous les algorithmes dans le but d'établir un comparatif.

Nous allons ensuite aborder les tests sur les algorithmes de classification supervisée en utilisant différents benchmarks (dans ce cas, les benchmarks utilisés doivent définir des populations étiquetées pour pouvoir mesurer les erreurs de classification). La performance de chaque algorithme (taux d'erreur) seront ensuite évalués afin qu'un comparatif puisse être établi.

Nous allons, enfin, tester l'algorithme Apriori sur différents benchmarks et mesurer ses performances en fonction des paramètres dans l'objectif de déterminer la meilleure configuration de paramètres qui donne les meilleurs résultats.

Tous les tests sont effectués sur une machine HP dotée d'un processeur Intel Core 2 Duo, avec une RAM de 3 GO tournant sous le système Microsoft © Windows XP Professionnel, version 2002 SP2. Plusieurs tests (une dizaine) sont effectués pour chaque opération, nous allons retenir la moyenne des résultats.

2. Présentation des benchmarks

Nous allons présenter, dans ce qui suit, les quatre benchmarks qui servent de jeux d'essai à nos algorithmes. Il s'agit des benchmarks ESI, le benchmark IRIS, les benchmarks IMAGE et le benchmark SUPERETTE. Des échantillons, un pour chaque benchmark, sont donnés en annexe (*voir Annexe*).

2.1. Benchmarks ESI

La famille des benchmarks ESI représente une population d'étudiants de l'Ecole nationale Supérieure d'Informatique (ESI) de différentes promotions. Ces benchmarks ont été obtenus auprès du service de scolarité de l'ESI sous forme d'une base de données ACCESS que nous avons convertie dans un fichier plat compatible avec notre plateforme.

Les benchmarks comportent dans la totalité 3676 individus partagés entre six populations selon l'année et l'option d'étude de la façon suivante :

Population		Taille
1TRC	1 ^{ère} année, tronc commun	1284
2TRC	2 ^{ème} année, tronc commun	956
3SI	3 ^{ème} année, systèmes d'information	379
3SIQ	3 ^{ème} année, systèmes informatiques	413
4SI	4 ^{ème} année, systèmes d'information	303
4SIQ	4 ^{ème} année, systèmes informatiques	341

Tableau 1 : La famille des benchmarks ESI

Chaque population est décrite par différents attributs. Les attributs communs à tous les benchmarks ESI sont énumérés dans tableau suivant :

Attribut	Signification
RESIDANT	1 si l'élève bénéficie d'une chambre à la cité universitaire ; 0 sinon.
MOYENNE	Moyenne obtenue en fin d'année (mois de juin)
MENTION	Mention obtenue en fin d'année ; ce champ prend les valeurs suivantes : 1 : Très bien 2 : Bien 3 : Assez bien 4 : Passable 6 : Rattrapage 7 : Abandon 8 : Redouble
RATTRAP	Nombre de modules au rattrapage
SERIEBAC	Série du bac ; cet attribut prend les valeurs suivantes : 4 : Sciences exactes 5 : Sciences naturelles et de la vie 7 : Génie électrique 8 : Génie mécanique
DECISION	Décision du conseil en fin d'année ; ce champ prend les valeurs suivantes : 1 : Admis 2 : Admis avec rachat 3 : Redouble 4 : Exclu.

Tableau 2 : Attributs communs aux benchmarks ESI

Les autres attributs représentent les notes obtenues dans les différents modules durant l'année scolaire (la note de chaque module est calculée comme étant la moyenne de trois épreuves trimestrielles).

Le tableau ci-dessous décrit les attributs du benchmark ESI-1TRC :

Attribut	Signification
IECO	Economie
ANG	Anglais
STRM	Structure de la machine
ALGO	Algorithmique
MATHS	Mathématiques
ELECT	Electronique

Tableau 3 : Attributs du benchmark ESI-1TRC

Le tableau suivant décrit les attributs du benchmark ESI-2TRC :

Attribut	Signification
SYSEX	Systèmes d'exploitation
MATHS	Mathématiques
SYSIN	Systèmes d'information
ALGO	Algorithmique
STRM	Structure de la machine
ANG	Anglais
STATS	Statistiques
LOGM	Logique mathématique

Tableau 4 : Attributs du benchmark ESI-2TRC

Le tableau suivant décrit les attributs du benchmark ESI-3SI :

Attribut	Signification
GEST	Gestion
RO	Recherche opérationnelle
SYSEX	Systèmes d'exploitation
FAT	Files d'attente
ANUM	Analyse numérique
ANG	Anglais
MCSI	Méthodes de conception des systèmes d'information
BDD	Bases de données

Tableau 5 : Attributs du benchmark ESI-3SI

Le tableau ci-dessous décrit les attributs du benchmark ESI-3SIQ :

Attribut	Signification
SYSEX	Systèmes d'exploitation
RO	Recherche opérationnelle
ANUM	Analyse numérique
THL	Théorie de langages
STRM	Structure de la machine
TELET	Télétraitement
ANG	Anglais
ELECT	Electronique

Tableau 6 : Attributs du benchmark ESI-3SIQ

Le tableau suivant décrit les attributs du benchmark ESI-4SI :

Attribut	Signification
GEST	Gestion
TELET	Télétraitement
ANALD	Analyse de données
ORGA	Organisation
MCP	Méthodes de conception des programmes
RO	Recherche opérationnelle
MCSI	Méthodes de conception des systèmes d'information

Tableau 7 : Attributs du benchmark ESI-4SI

Le tableau suivant décrit les attributs du benchmark ESI-4SIQ :

Attribut	Signification
FAT	Files d'attente
MCP	Méthodes de conception des programmes
BDD	Bases de données
SYSEX	Systèmes d'exploitation
AUTO	Automatique
ANALD	Analyse de données
COMP	Compilation
ARCHO	Architecture des ordinateurs

Tableau 8 : Attributs du benchmark ESI-4SIQ

Les populations des différents benchmarks ESI ne sont pas étiquetées, i.e. les classes d'étudiants ne sont pas connues. Il est donc intéressant de procéder à un clustering pour découvrir les partitions d'étudiants similaires.

Sur la base de ces six benchmarks, nous avons construit un septième benchmark ESI-SIQ-SI de 396 étudiants de la manière suivante :

- Chaque individu du benchmark ESI-SIQ-SI décrit un étudiant de la troisième année avec ses attributs de la première et la deuxième année ;
- La classe de l'individu est déterminée selon l'option d'études (classe 0 pour SIQ, classe 1 pour SI).

ESI-SIQ-SI est un benchmark qui présente des intérêts forts intéressants ; il peut servir, entre autres, à classer les étudiants de la deuxième année selon leurs notes obtenus en première et deuxième année et d'autres attributs, comme la décision, le rattrapage, la mention... Les étudiants sont alors répartis sur les deux spécialités en se basant sur la partition décrite par le benchmark ESI-SIQ-SI.

2.2. Benchmark IRIS [38]

Le benchmark IRIS décrit 150 observations correspondant à trois variétés de la plante iris ("*setosa*", "*versicolor*" et "*virginica*") à partir de leurs caractéristiques morphologiques. Chaque individu est défini par ses quatre attributs numériques qui sont :

- La longueur des sépales (LNG_SEP) en cm ;
- La largeur des sépales (LRG_SEP) en cm ;
- La longueur des pétales (LNG_PET) en cm ;
- La largeur des pétales (LRG_PET) en cm.

La partition IRIS possède les caractéristiques suivantes :

- Nombre de classes : 3 ;
- Nombre d'individus dans chaque classe : 50 ;
- Inertie intra-classe : 89,33 ;
- Inertie inter-classe : 11,89.

La population de départ étant déjà partitionnée en classes, il s'avère donc intéressant d'utiliser le benchmark IRIS pour évaluer la performance des algorithmes de classification.

2.3. Benchmarks IMAGE

L'image numérique est un ensemble de points juxtaposés et colorés que l'on appelle "pixels". Chaque pixel possède deux coordonnées (x , y), qui déterminent son emplacement dans l'image, et une couleur composée, d'un mélange des trois couleurs principales (rouge, bleu, vert) avec différentes proportions comprises entre 0 et 255.

La segmentation d'image repose sur la similarité entre les pixels (calculée sur la base de ses attributs, les trois couleurs) pour segmenter l'image numériques en plusieurs régions (classes de pixels) différentes. Pour cela, une population, où chaque individu représente un pixel, doit être construite avant d'être soumise à un algorithme de clustering.

Exemple :

La Figure 25 représente une image numérique carrée de 96 pixels², ce qui donne une population de 9216 individus.


	Pixels	X	Y	Rouge	Bleu	Vert
	0	0	0	255	0	197
	10	1	10	71	87	220
	...					

Figure 25 : Exemple d'un benchmark IMAGE

Les benchmarks IMAGE sont souvent de taille importante (dizaines voire centaines de milliers d'individus) ; il est, donc, intéressant de les utiliser pour tester les algorithmes de clustering face à de grandes populations.

2.4. Benchmark SUPERETTE

Le benchmark SUPERETTE est une population de 129 tickets de supérette obtenus gracieusement auprès de la supérette "UNIVERS SHOP – Beau lieu". Chaque individu de la population est décrit par 11 attributs. Les attributs englobent, chacun, une famille d'articles représentés par le tableau suivant :

Attribut	Signification	Exemples d'articles
BOISSON	Boissons	Boissons gazeuses, jus
EAU	Eau minérale	
SUCRERIE	Sucreries	Bonbons, chocolat, chewing-gum,
PATE	Pates alimentaires	Spaghetti, macaronis, couscous, riz
PAIN	Produits de boulangerie	Pain, pain amélioré, farine, semoule
LAIT	Laits et produits laitiers	Lait, petit lait, beurre, margarine
CAFE_THE	Café ou thé	Café, thé
POISSON	Poissons	Thon, poisson conserve
DESSERT	Desserts	Yaourt, glaces, crèmes
FROMAGE	Fromages	Fromage fondu, gruyère, fromage rouge
CONSERVE	Produits conserves	Concentré de tomate, confiture

Tableau 9 : Attributs du benchmark SUPERETTE

La valeur de chaque attribut définit son absence sur le ticket si elle est égale à 0. Sinon, elle représente la quantité achetée.

Exemple :

La figure ci-dessous représente un ticket de supérette et l'individu y correspondant.

B	E	S	P	P	L	C	P	D	F	C
O	A	U	A	A	A	A	O	E	R	O
I	U	C	T	I	I	F	I	S	M	N
S		R				-	S	S	A	S
S		E				T	O	E	E	E
O		R				H	N	R	R	R
N		I				E		G	V	E
4	0	4	0	0	0	0	0	6	0	0

15/04/10				
UNIVERS SHOP BEAULIEU				
Ticket: 1122150410023				
Article	PV	RM	Qte	Total
HICOPS MAHBOUL	15		1	15,00
RIMY 100g	25		1	25,00
RIMY 100g	25		1	25,00
TCUDJA ORANGE	80		1	80,00
1,25L				
TCUDJA ORANGE	80		1	80,00
1,25L				
LU MAJOR AU	50		1	50,00
CHOCOLAT 190GR				
PENOTT PM	225		1	225,00
SCUMMAM	13		6	78,00
AFOMATISE				
BCNBON	5		1	5,00
Nbr Produits	14			
Total à Payer:				563,00
Paiement :	ESPECE		1000	4:7
2	2	AMINA	17:10:19	15/04/2013

NB: RECLAMATION DANS 24H MAX AVEC
PRESENTATION DU TICKET
**MERCI DE VOTRE VISITE **




Figure 26 : Exemple d'un individu du benchmark SUPERETTE

Le benchmark SUPERETTE est utilisé pour extraire les règles d'association qui peuvent exister entre les articles (découvrir les articles qui figurent ensemble sur le même ticket de supérette).

3. Tests de clustering

La qualité des résultats des algorithmes de clustering ainsi que leurs performances dépendent, éventuellement, de leurs paramètres. Nous allons, dans ce qui suit, tester chacun de nos algorithmes avec différentes valeurs des paramètres. Ces tests, individuels, nous permettent de déterminer, pour chaque algorithme, la meilleure configuration qui permet de bien exploiter la puissance de l'algorithme en vue d'avoir les meilleurs résultats. Les paramètres communs à tous les algorithmes, comme le nombre des classes, sont utilisés pour établir le comparatif.

Le Tableau 10 montre les résultats d'exécution des algorithmes de clustering sur le benchmark ESI-3SIQ. Ceci permet de donner un premier aperçu de l'exécution des algorithmes.

Algorithme	Paramètres	Nombre de classes	Temps d'exécution [ms]	Inertie intra-classe	Inertie inter-classe	Inertie Intra/inter
k-Means		3	220	17034	350	48,66
PAM		3	20252	26169	326	80,27
CLARA	Taille de l'échantillon = 60	3	777	25110	250	100,24
Hiérarchique Ascendant	Métrique = lien simple	3	208578	26642	947	28,13
AG - codage par classe	Nombre d'itérations = 400	3	52503	16289	593	27,46
AG - codage par vecteur	Nombre d'itérations = 400	3	29154	15555	545	29,54

Tableau 10 : Premiers tests sur les algorithmes de clustering

Ces premiers tests nous ont permis de dégager les résultats préliminaires suivants :

- Le temps d'exécution varie amplement d'un algorithme à l'autre ;
- La qualité des résultats varie aussi en fonction de l'algorithme utilisé ;
- L'exécution de certains algorithmes plusieurs fois sur le même benchmark peut mener à des résultats différents, notamment k-Means, PAM, CLARA et l'AG. Ceci est dû au fait que ces algorithmes utilisent des fonctions aléatoires non déterministes ; ce qui nous a conduits à penser à effectuer plusieurs tests pour chaque opération et en prendre la moyenne.

3.1. Tests sur les algorithmes

3.1.1. L'algorithme CLARA

Dans les deux tests qui suivent, nous allons essayer de cerner l'effet des deux paramètres de l'algorithme CLARA, en l'occurrence, la taille de l'échantillon et le nombre d'essais. Les résultats sont obtenus en appliquant l'algorithme CLARA sur le benchmark ESI-4SIQ avec un nombre de classes égal à 3 (pour les deux tests).

Le graphique illustré par la Figure 27 montre l'évolution de la qualité du résultat (mesurée par le rapport inertie intra-classe / inertie inter-classe) en fonction de la taille de l'échantillon (le nombre d'essais étant fixé à 4).

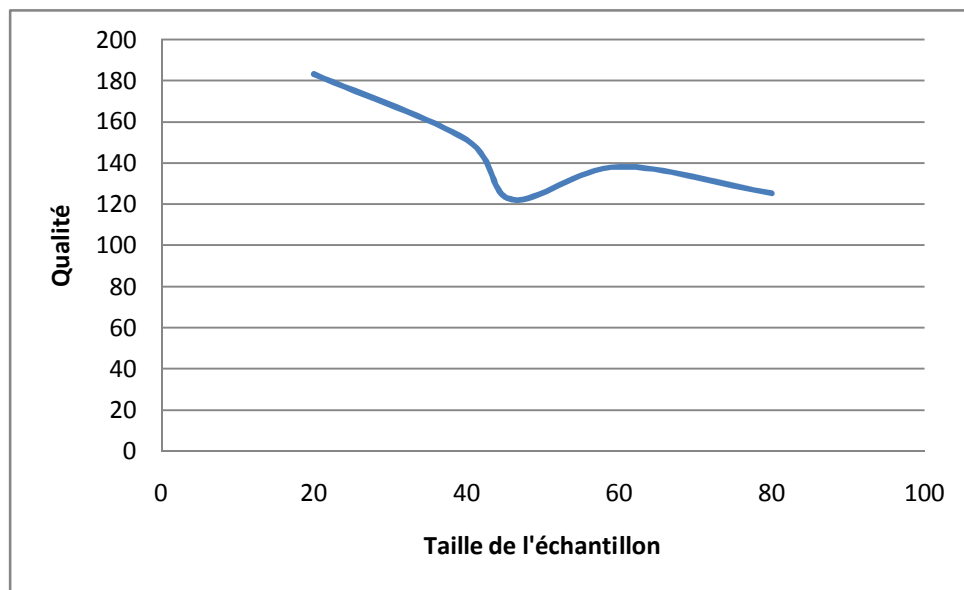


Figure 27 : Qualité de clustering de CLARA en fonction de la taille de l'échantillon

On remarque, sur le graphique précédent, que la meilleure qualité de CLARA est atteinte pour une taille d'échantillon égale ou proche de $46 = 40 + 2 * 3$, ce qui confirme les expérimentations de leurs auteurs (voir Chapitre II, 2.4.1.4).

La Figure 28 représente un graphique qui montre l'évolution de la qualité du résultat (mesurée par le rapport inertie intra-classe / inertie inter-classe) en fonction du nombre d'essais (la taille de l'échantillon étant fixée à 46).

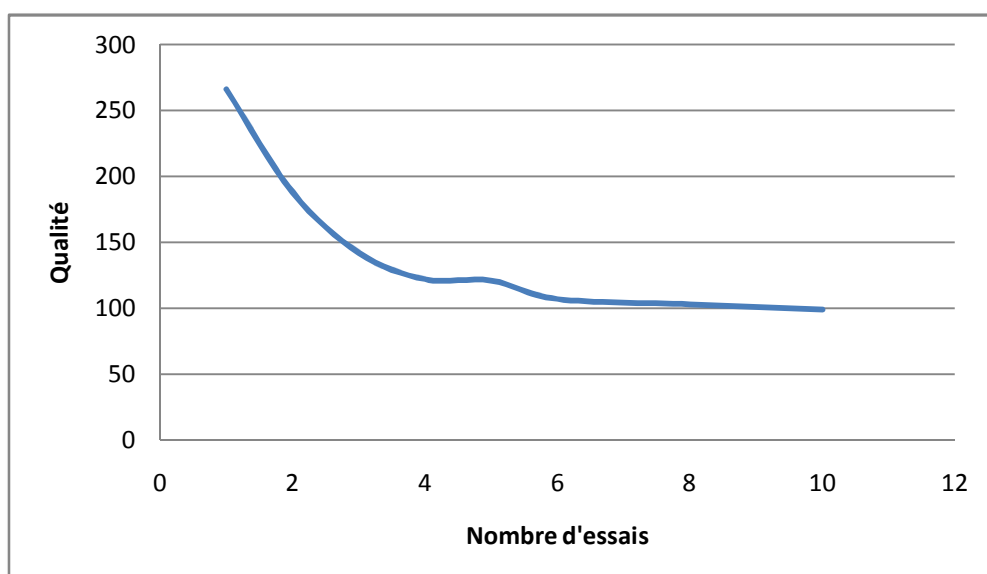


Figure 28 : Qualité de clustering de CLARA en fonction du nombre d'essais

Le Tableau 11 montre les temps d'exécution correspondants aux différentes valeurs du nombre d'essais précédentes.

Nombre d'essais	Temps d'exécution [ms]
1	60
2	145
3	215
4	249
5	281
6	1001
8	1625
10	1901

Tableau 11 : Temps d'exécution de CLARA en fonction du nombre d'essais

On constate, sur le graphique illustré par la Figure 28, qu'il y a une relation proportionnelle entre la qualité du résultat et le nombre d'essais. Ceci s'explique par le fait que si le nombre d'essais augmente, les meilleurs médoides ont plus de chance d'être sélectionnés, ce qui donne un meilleur résultat.

On remarque, également sur le Tableau 11, qu'au-delà d'une certaine valeur du nombre d'essais (entre 4 et 6), la qualité continue à s'améliorer légèrement mais au profit du temps d'exécution qui monte brusquement, ce qui rend ces solutions moins intéressantes. Ceci s'explique par le fait que 4 à 6 échantillons sont, souvent, suffisants pour explorer la totalité des individus et donner la chance à la majorité d'entre eux pour être sélectionnés parmi les médoides.

Conclusion :

La configuration d'échantillons de taille égale à $40 + 2 * k$, k étant le nombre de classes, et un nombre d'essais égal à 4 permet, souvent, d'atteindre le meilleur résultat de l'algorithme CLARA, et de réaliser un bon compromis : qualité de clustering / temps d'exécution.

3.1.2. L'algorithme hiérarchique ascendant

Les résultats fournis par l'algorithme hiérarchique ascendant dépendent, éventuellement, de la métrique inter-classe utilisée. Nous allons essayer par le test suivant de déterminer l'effet de ce paramètre sur les performances de l'algorithme.

Les résultats du Tableau 12 sont obtenus en appliquant l'algorithme hiérarchique ascendant sur le benchmark ESI-3SIQ avec un nombre de classes variant de 3 à 5.

Métrique inter-classe	Nombre de classes	Temps d'exécution [ms]	Inertie intra-classe	Inertie inter-classe	Inertie intra-classe / inertie inter-classe
Lien simple	k=3	218578	26642	947	28,13
	k=4	204250	26477	1334	18,84
	k=5	203872	26265	1464	17,94
Lien moyen	k=3	273022	26309	1072	24,54
	k=4	270722	26161	1540	16,98
	k=5	270563	25996	2205	11,78
Lien complet	k=3	252882	18667	792	23,56
	k=4	211198	15616	833	18,74
	k=5	200697	14886	1156	12,88

Tableau 12 : Résultats de l'algorithme hiérarchique ascendant en fonction de la métrique inter-classe

On remarque, sur le tableau ci-dessus, que la qualité du résultat diffère, légèrement, d'une métrique à une autre. On remarque aussi que le temps de calcul est plus important en utilisant le lien complet, ceci est dû au calcul de la moyenne des objets.

Conclusion :

Il est préférable de ne pas utiliser le lien moyen car il est le plus coûteux en temps de calcul.

3.1.3. L'algorithme génétique

La série des tests qui suivent sont effectués pour déterminer la meilleure configuration de l'algorithme génétique pour les deux codages : par classe et par vecteur centroïde. Il est également opportun d'effectuer une petite comparaison entre les deux codages. La qualité de clustering est mesurée par l'inertie intra-classe, Nous utilisons le benchmark ESI-4SIQ. Les paramètres concernés sont étudiés dans l'ordre suivant :

- Mode d'initialisation (aléatoire, par k-Means) ;
- Nombre d'itérations ;
- Taille de la population de solutions ;

Le graphique illustré par la Figure 29 montre l'effet du mode d'initialisation sur la qualité de clustering. L'algorithme est configuré comme suit :

- Nombre de classes : 3 ;
- Taille de la population de solutions : 10 ;
- Fonction fitness : inertie intra-classe ;
- Nombre d'itérations : 10 ;
- Probabilité de croisement : 0.9 ;
- Probabilité de mutation : 0,01 ;
- Croisement en 1 point ;
- Politique de sélection : déterministe.

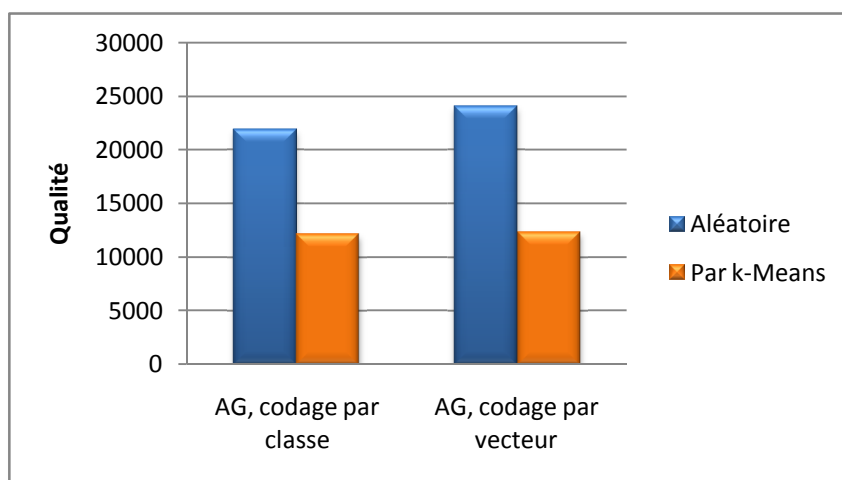


Figure 29 : Qualité de clustering de l'AG en fonction du mode d'initialisation

On remarque, sur le graphique ci-dessus, que l'initialisation par k-Means donne toujours le meilleur résultat que l'initialisation aléatoire, et ce quelque soit le codage utilisé. Ceci s'explique par le fait que démarrer avec une partition obtenue par k-Means accélère la convergence et résume plusieurs générations de l'AG.

La Figure 30, respectivement la Figure 31, montrent l'évolution de la qualité de clustering, respectivement, le temps d'exécution (ms), en fonction du nombre d'itérations en fixant les autres paramètres comme suit :

- Nombre de classes : 3 ;
- Taille de la population de solutions : 10 ;
- Fonction fitness : inertie intra-classe ;
- Probabilité de croisement : 0.9 ;
- Probabilité de mutation : 0,01 ;
- Politique de sélection : déterministe ;
- Croisement en 1 point ;
- Initialisation : aléatoire.

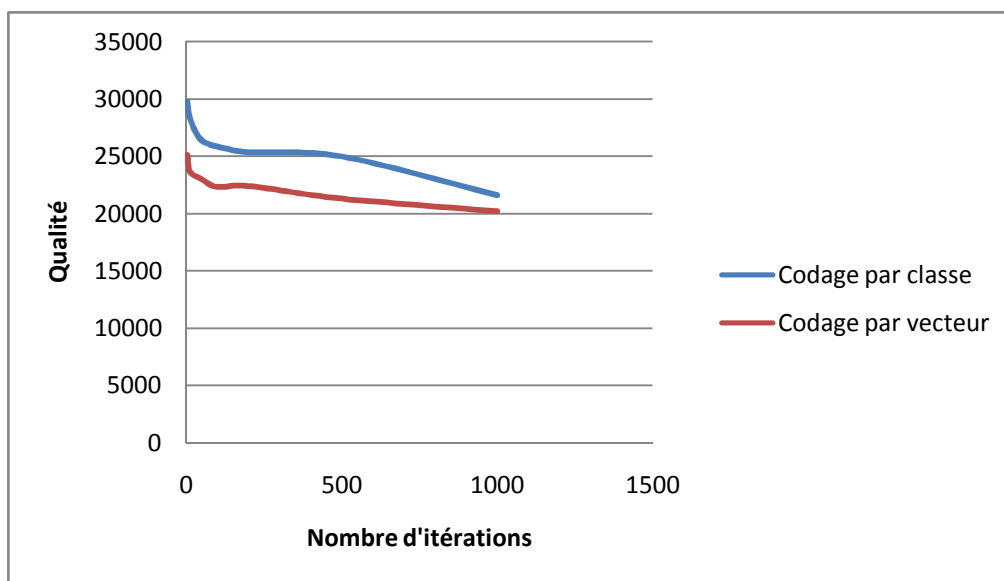


Figure 30 : Qualité de clustering de l'AG en fonction du nombre d'itérations

On remarque, sur le graphe ci-dessus, que la qualité du résultat varie proportionnellement au nombre d'itérations, et ce pour les deux codages utilisés. On remarque, aussi, que la convergence est plus rapide dans les 100 premières itérations.

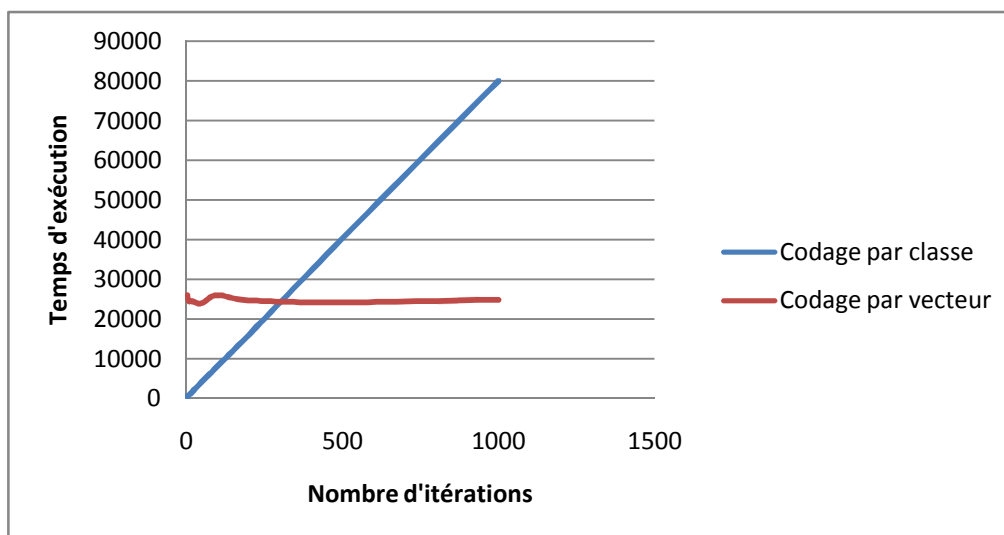


Figure 31 : Temps d'exécution de l'AG en fonction du nombre d'itérations

On remarque, sur le graphique ci-dessus, que le temps de calcul de l'AG dépend fortement du codage utilisé. En effet, en codage par classe, le temps de calcul augmente de façon exponentielle conformément au nombre d'itérations et devient, au bout de quelques centaines d'itérations, inacceptable. Par contre, en codage par vecteur, le temps de calcul semble être indépendant du nombre d'itérations, du moins dans les 1000 premières itérations.

Conclusion :

Un nombre d'itérations inférieur à 500 constitue un bon compromis qualité de clustering / temps d'exécution pour l'algorithme génétique, codage par classe. Par contre, le codage par vecteur offre une plus grande tolérance en temps de calcul et le nombre d'itérations peut être élevé à quelques milliers.

Les graphiques illustrés par la Figure 32, respectivement la Figure 33, montrent la variation de la qualité du résultat, respectivement, le temps d'exécution (ms) de l'AG, en fonction de la taille de la population de solutions potentielles. Les paramètres de l'algorithme ont été fixés comme suit :

- Nombre de classes : 3 ;
- Nombre d'itérations : 500 ;
- Fonction fitness : inertie intra-classe ;
- Probabilité de croisement ;
- Probabilité de croisement : 0.9 ;
- Probabilité de mutation : 0,01 ;
- Politique de sélection : déterministe ;
- Croisement en 1 point ;
- Initialisation : aléatoire.

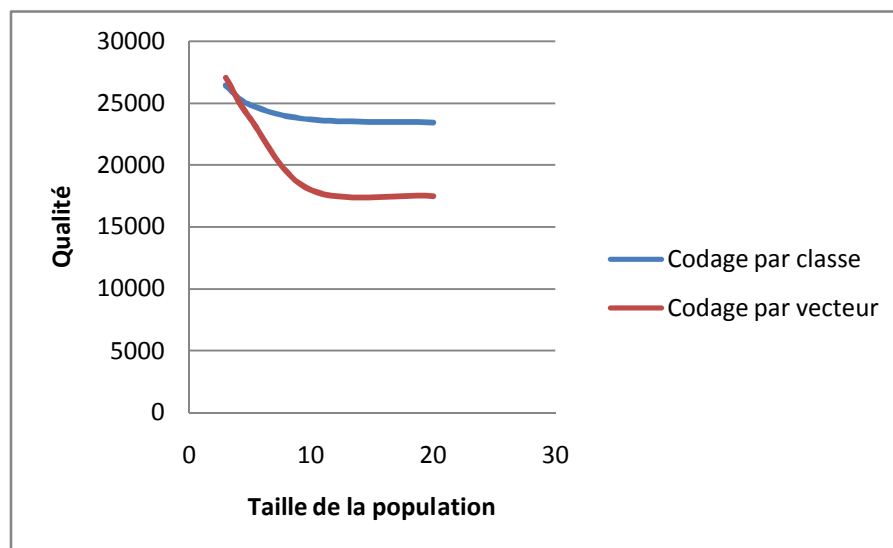


Figure 32 : Qualité de clustering de l'AG en fonction de la taille de la population de solutions

On remarque, sur le graphique ci-dessus, que la qualité du résultat de l'algorithme génétique augmente proportionnellement à la taille de la population (nombre de solutions potentielles) quand cette dernière est inférieure à 10. Dépassé ce nombre, la qualité semble être stagnée et l'augmentation du nombre de solutions n'apporte aucune amélioration significative.

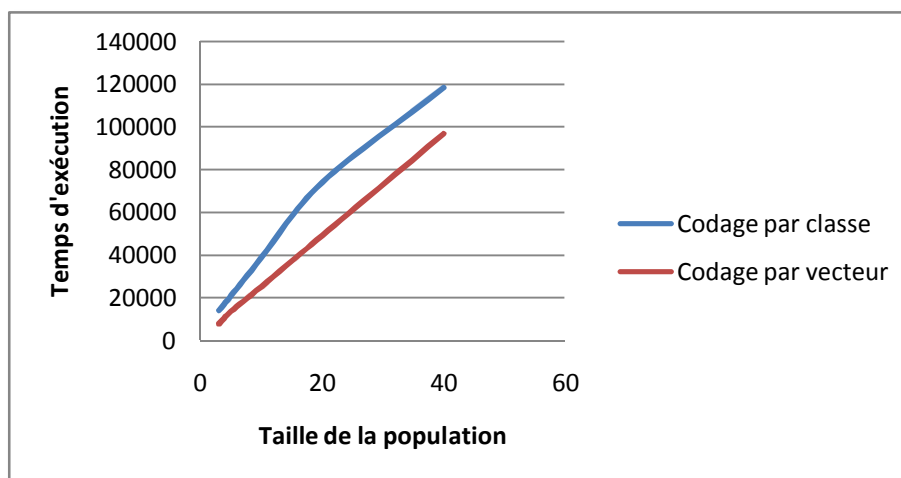


Figure 33 : Temps d'exécution de l'AG en fonction de la taille de la population de solutions

On remarque, sur le graphique ci-dessus, que le temps de calcul monte rapidement avec l'augmentation du nombre de solutions potentielles ; il devient inacceptable si cette dernière dépasse une vingtaine.

Conclusion :

La taille de la population de solutions est déterminante pour les performances de l'algorithme génétique. Elle ne doit pas être trop élevée pour que le temps de calcul soit raisonnable, ni trop faible pour que la solution trouvée soit intéressante.

3.1.4. Comparatif des algorithmes de clustering

Dans cette section, nous allons tenter d'établir un comparatif des algorithmes de clustering. Chaque algorithme est configuré avec ses meilleurs paramètres (Tableau 13) obtenus par les tests précédents. La comparaison se fait sur la base d'un seul critère, à la fois, parmi les deux critères suivants :

- Temps d'exécution (en fonction de la taille et la dimension de la population) ;
- Qualité des résultats.

Algorithme	Paramètres
CLARA	Taille de l'échantillon : $40+2*k$ Nombre d'essais : 5
Algorithme hiérarchique ascendant	Métrique inter-classe : Lien simple
AG	Initialisation : par k-Means Taille de la population : 10 Nombre d'itérations : 100

Tableau 13 : Meilleurs paramètres des algorithmes de clustering

La Figure 34 représente l'évolution du temps d'exécution (ms) des différents algorithmes de clustering en fonction de la taille de la population. Le test est effectué sur des échantillons du benchmark ESI-1TRC de tailles différentes. Le nombre de classes est fixé à 3 pour tous les algorithmes.

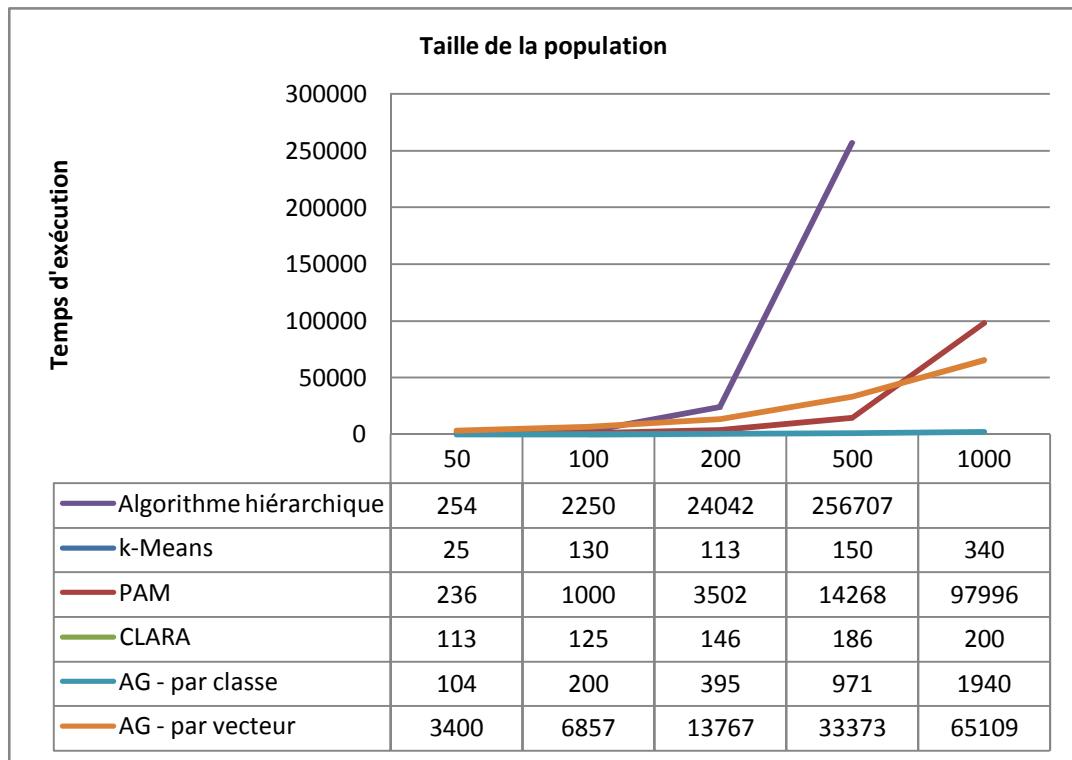


Figure 34 : Temps d'exécution des algorithmes de clustering en fonction de la taille de la population

Sur le graphique ci-dessus, on remarque que :

- Le temps d'exécution varie conformément à la taille de la population, et ce pour tous les algorithmes ;
- L'algorithme CLARA présente la plus faible variation en temps de calcul, suivi de k-Means, AG codage par classe, AG codage par vecteur et puis PAM.

Le graphique ci-dessous (Figure 35) représente les variations du temps d'exécution (ms) en fonction de la dimension (nombre d'attributs). Le benchmark utilisé est ESI-SIQ-SI (396 étudiants) avec différentes configurations d'attributs.

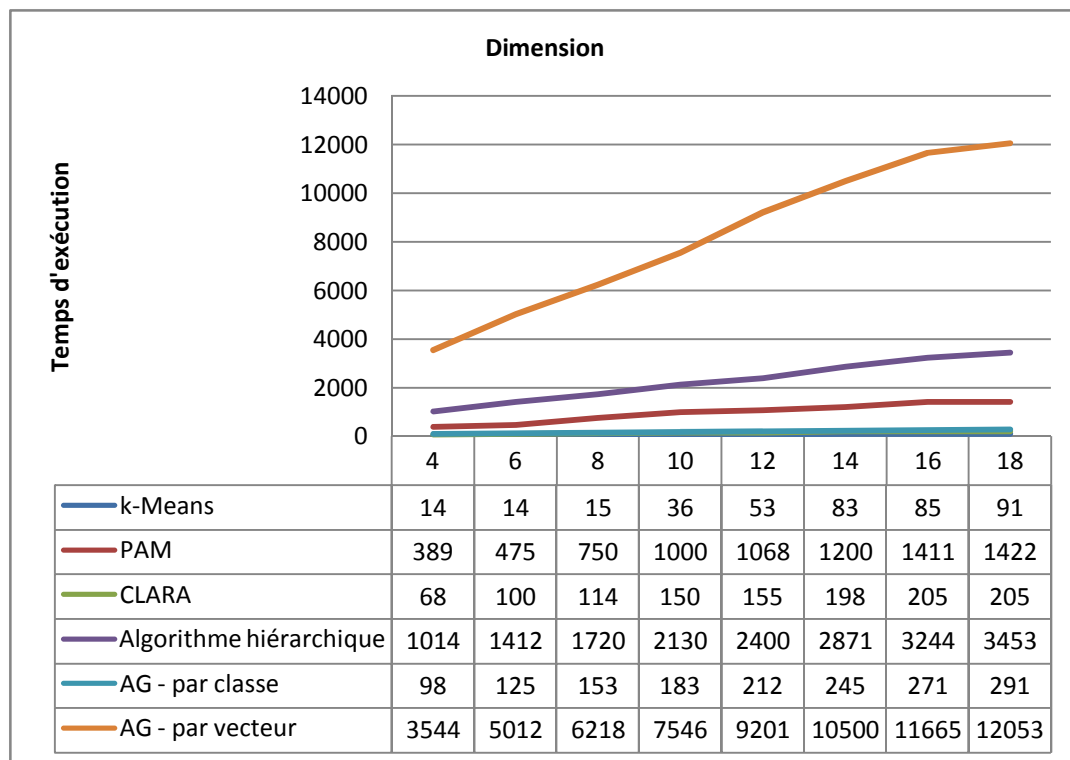


Figure 35 : Temps d'exécution des algorithmes de clustering en fonction de la dimension de la population

On remarque, sur le graphique ci-dessus, que :

- Le temps d'exécution augmente à mesure que la dimension augmente, et ce pour tous les algorithmes ;
- L'algorithme k-Means présente la plus faible variation en temps d'exécution, suivi de l'AG codage par classe, CLARA, PAM et puis l'algorithme hiérarchique ascendant.

Conclusion :

- CLARA est l'algorithme qui présente le meilleur temps de calcul face à de grandes populations, ceci est dû à sa politique de réduction de la population par échantillonnage.
- K-Means est l'algorithme le moins coûteux en temps de calcul face à des populations de grande dimension ;
- Les algorithmes de clustering sont classés, selon leurs temps d'exécution, dans le tableau suivant :

Populations de grande taille		Populations de grande dimension	
1	CLARA	1	k-Means
2	k-Means	2	AG, codage par classe
3	AG, codage par classe	3	CLARA
4	AG, codage par vecteur	4	PAM
5	PAM	5	Algorithme hiérarchique ascendant
6	Algorithme hiérarchique ascendant	6	AG, codage par vecteur

Tableau 14 : Classement des algorithmes de clustering selon le temps d'exécution

Le graphique ci-dessous montre l'évolution de la qualité des résultats des différents algorithmes de clustering, celle-ci étant mesurée par le rapport inertie intra-classe / inertie inter-classe. Le benchmark utilisé est ESI-4SIQ avec un nombre de classes fixé à 3. Chaque algorithme est paramétré avec les meilleures valeurs trouvées précédemment (Tableau 13). L'algorithme génétique démarre avec une partition obtenue par k-Means et un nombre d'itérations égal à 500.

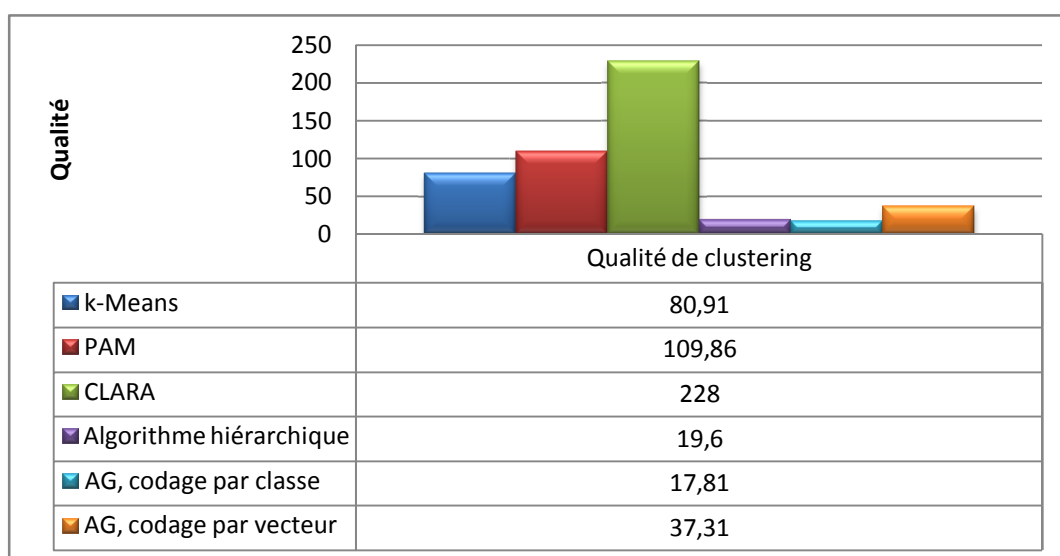


Figure 36 : Qualité des résultats des algorithmes de clustering

On remarque, sur le graphique ci-dessus, que la qualité des résultats des algorithmes de clustering varie amplement d'un algorithme à l'autre ; l'algorithme génétique, par exemple, atteint une qualité dix fois meilleure que celle atteinte par l'algorithme CLARA.

Conclusion :

- L'algorithme génétique, codage par classe, fournit toujours la meilleure qualité de partition (inertie intra-classe/inertie inter-classe) par rapport aux autres algorithmes de clustering ;
- Les algorithmes de clustering sont classés, selon la qualité des résultats, dans le tableau suivant :

1	AG, codage par classe
2	Algorithme hiérarchique ascendant
3	AG, codage par vecteur
4	k-Means
5	PAM
6	CLARA

Tableau 15 : Classement des algorithmes de clustering selon la qualité des résultats

La Figure 37 montre l'évolution de la qualité des résultats des algorithmes clustering (mesurée par le rapport inertie intra-classe / inertie inter-classe) en fonction du nombre de classes k .

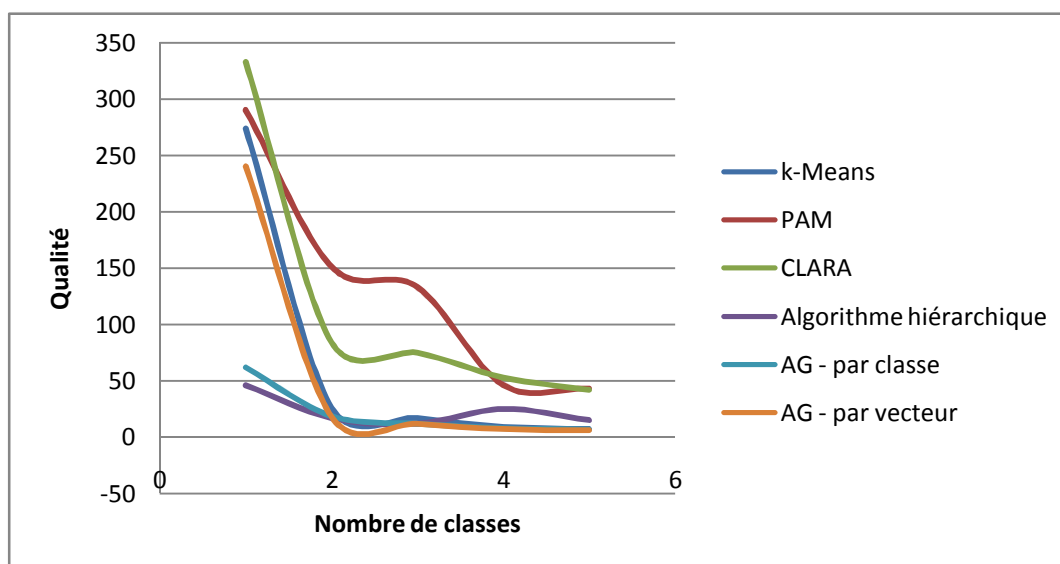


Figure 37 : Qualité des résultats des algorithmes de clustering en fonction du nombre de classes

On remarque, sur le graphique ci-dessus, que la qualité des résultats de clustering augmente à mesure que le nombre de classes augmente, et ce pour tous les algorithmes. En effet, plus le nombre de classes est important, plus le nombre d'individus dans chaque classe diminue, ce qui diminue l'inertie intra-classe. L'idéal est d'avoir un seul individu dans chaque classe, ce qui est inacceptable.

Conclusion :

Le nombre de classes est fixé de façon empirique en fonction des besoins de l'utilisateur.

3.2. Tests sur les benchmarks

Dans cette deuxième phase, nous nous intéressons aux tests de clustering sur un benchmark de la famille ESI, qui est ESI-4SIQ et un benchmark IMAGE.

3.2.1. Benchmark ESI-4SIQ

L'objectif de ce test est de donner une signification aux classes d'étudiants obtenues par le clustering de la population ESI-4SIQ. La Figure 38 représente une projection sur les attributs (AUTO, MCP) du résultat d'application de l'algorithme génétique sur la population d'étudiants. Les paramètres de l'algorithme ont été fixés comme suit :

- Codage : par classe ;
- Nombre de classes : 3 ;
- Initialisation : par k-Means ;
- Fonction fitness : inertie intra-classe / inertie inter-classe ;
- Nombre d'itérations : 500.

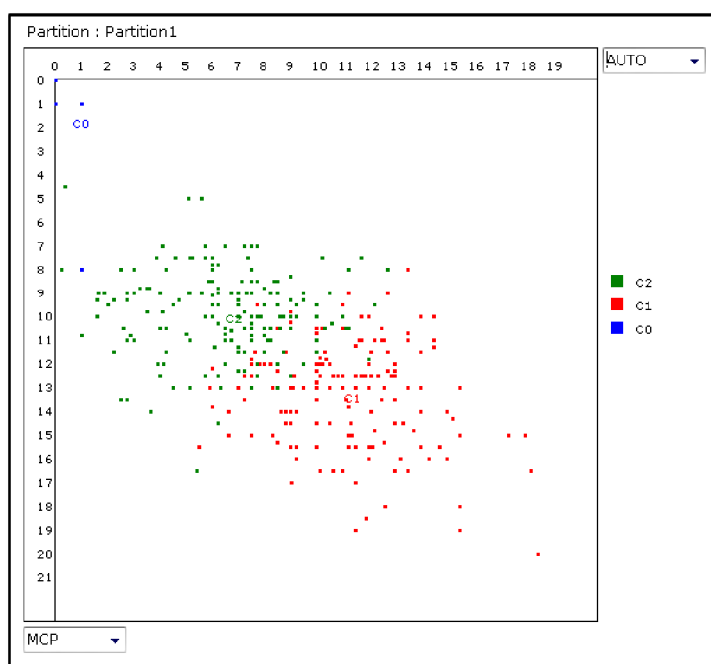


Figure 38 : Résultat de clustering sur le benchmark ESI-4SIQ (projection sur 2 attributs)

La partition résultante possède les caractéristiques suivantes :

- Inertie intra-classe = 12900,57 ;
- Inertie inter-classe = 724,34.

Les trois classes résultantes sont représentées dans le tableau ci-dessous.

Classe	Couleur	Effectif	Centre	Signification
C0	bleue	8	RESIDANT = 0,38 SERIEBAC = 3,62 FAT = 0,91 MCP = 1,62 BDD = 0,73 SYSEX = 1,45 AUTO = 0,62 ANALD = 1,04 COMP = 0,94 ARCHO = 0,88 MOYENNE = 0,33 DECISION = 3,25 MENTION = 7,5 RATTRAP = 0	<p>Cette classe englobe les étudiants qui ont des notes très médiocres (MOYENNE=0,33). Ces étudiants redoublent (MENTION=7) ou abandonnent (MENTION=8) s'ils ne sont pas exclus (DECISION=3).</p>
C1	rouge	166	RESIDANT = 0,31 SERIEBAC = 4,70 FAT = 11,19 MCP = 13,20 BDD = 13,88 SYSEX = 13,86 AUTO = 11,04 ANALD = 11,84 COMP = 13,63 ARCHO = 14,68 MOYENNE = 13,12 DECISION = 1,01 MENTION = 2,93 RATTRAP = 0	<p>Cette classe est constituée de la portion d'étudiants excellents ou en-dessus de la moyenne (MOYENNE=13,12). Leurs mentions varient entre "Bien" (MENTION=3) et "Très bien" (MENTION=2). Ils sont très souvent admis sans rachat (DECISION=1).</p>
C2	verte	167	RESIDANT = 0,42 SERIEBAC = 4,77 FAT = 7,26 MCP = 9,86 BDD = 10,81 SYSEX = 11,70 AUTO = 6,48 ANALD = 9,41 COMP = 10,31 ARCHO = 11,69 MOYENNE = 9,99 DECISION = 1,38 MENTION = 4,86 RATTRAP = 2,11	<p>Cette classe contient la portion d'étudiants moyens (MOYENNE=9,99). Ils ont des mentions le plus souvent "Passable". La décision varie entre "Admis" (DECISION=1) et "Admis avec rachat" (DECISION=2).</p>

Tableau 16 : Résultats de clustering appliqué sur le benchmark ESI-4SIQ

3.2.2. Benchmark IMAGE

La figure ci-dessous représente les résultats d'application de différents algorithmes de clustering sur une image de 96×96 pixels². Le benchmark correspondant comporte une population de pixels de taille égale à 9216 individus.


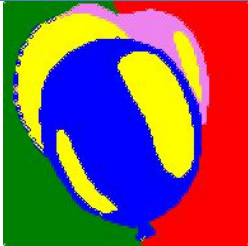

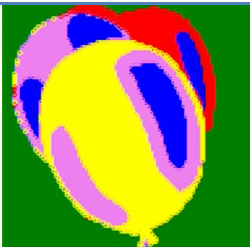

Algorithme	Paramètres	Résultat	Inertie intra-classe / inertie inter-classe
Image d'origine			
k-Means	$k = 5$.		329
CLARA	$k = 5$; Taille de l'échantillon = 46 ; Nombre d'essais = 5.		343
AG, codage par classe	$k = 5$, Initialisation par k-Means ; Fitness = inertie intra-classe / inertie inter-classe ; Nombre d'itérations = 100.		281
AG, codage par vecteur	$k = 5$, Initialisation par k-Means ; Fitness = inertie intra-classe / inertie inter-classe ; Nombre d'itérations = 100.		281

Tableau 17 : Résultats de clustering appliqué sur le benchmark IMAGE

N.B.

L'algorithme PAM et l'algorithme hiérarchique n'ont pas été exécutés sur le benchmark IMAGE car ils présentent un temps de calcul trop élevé (plus de 10 mn).

On constate, sur le tableau ci-dessus, que l'algorithme génétique donne la meilleure qualité de clustering, d'ailleurs il est le seul qui réussit à identifier un fond uniforme. On constate aussi que la solution trouvée par l'AG est exactement la même pour les deux codages utilisés. Ceci peut être expliqué par le fait que l'AG parvient à trouver la solution optimale (pour 5 classes).

4. Tests de classification

Cette deuxième section est consacrée aux tests effectués sur les algorithmes de classification. Nous allons, dans un premier temps, chercher la meilleure configuration (valeurs de paramètres) pour chacun des algorithmes de classification. Ensuite, les algorithmes sont testés face à différentes situations pour établir un comparatif.

Les tests individuels sur chaque algorithme sont effectués sur un ensemble d'apprentissage et un ensemble de test obtenus par l'échantillonnage du benchmark IRIS. L'ensemble d'apprentissage sert à construire le modèle et l'ensemble de test sert à évaluer la performance du modèle construit en calculant le taux d'erreur.

4.1. Tests sur les algorithmes

4.1.1. L'algorithme k-NN

Le graphique ci-dessous représente l'évolution de l'erreur de classification de l'algorithme k-NN (en pourcentage) en fonction de la taille de l'ensemble d'apprentissage ; k étant fixé à 50.

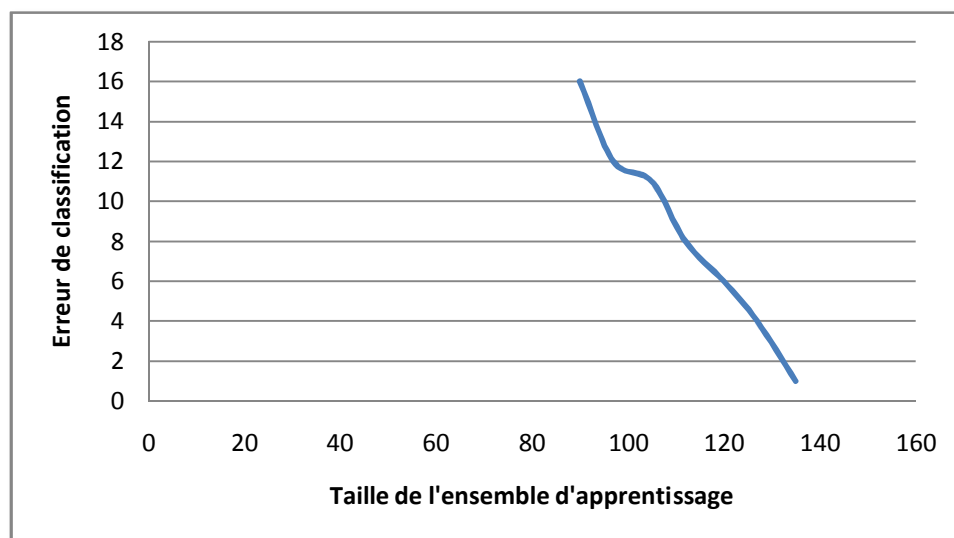


Figure 39 : Erreur de classification de k-NN en fonction de la taille de l'ensemble d'apprentissage

On remarque, sur le graphique ci-dessus, que l'erreur de classification de l'algorithme k-NN baisse au fur et à mesure que la taille de l'ensemble d'apprentissage augmente. Ceci s'explique par le fait que l'algorithme k-NN adopte le raisonnement à partir de cas ; plus le nombre de cas déjà résolus augmente, plus la chance de trouver des cas similaires à l'individu à classer augmente, ce qui réduit le taux d'erreur de la classification.

Conclusion :

Un ensemble d'apprentissage de 80% donne un bon résultat de classification (taux d'erreur inférieur à 10%), ce qui confirme les expérimentations (*voir Chapitre I, 2.2.5*).

Maintenant, on fixe la taille de l'ensemble d'apprentissage à 80% et on fait varier le nombre de voisins k . Pour chaque valeur de k , l'erreur de classification est calculée (Figure 40).

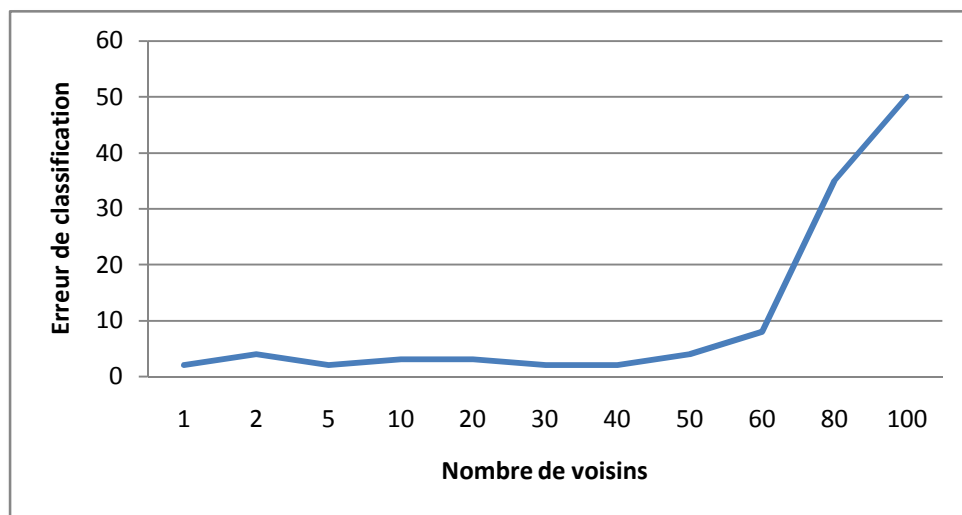


Figure 40 : Erreur de classification de k-NN en fonction du nombre de voisins k

On remarque que l'erreur de classification est très acceptable pour un nombre de voisins qui varie entre 1 et 50. Au-delà de cette valeur, les performances de l'algorithme se dégradent et le taux d'erreur s'élève brusquement. Ceci s'explique par le fait que considérer un plus grand nombre de voisins élargit le champ de voisinage ; les voisins peuvent, ainsi, se trouver loin de l'individu à classer, ce qui mène à des erreurs de classification.

Conclusion :

- La valeur du paramètre k est déterminante pour les performances de l'algorithme k-NN ;
- 1-NN donne souvent de bons résultats. Il est donc préférable d'utiliser 1-NN quand on ne sait pas la meilleure valeur de k .

4.1.2. L'arbre de décision

Tous les tests sur les arbres de décision sont effectués en fixant la valeur de discrétisation de chaque attribut à sa moyenne calculée sur la population d'apprentissage.

Le graphique suivant (Figure 41) montre la variation du taux d'erreur (en pourcentage) commise par l'arbre de décision en fonction de la taille de l'ensemble d'apprentissage.

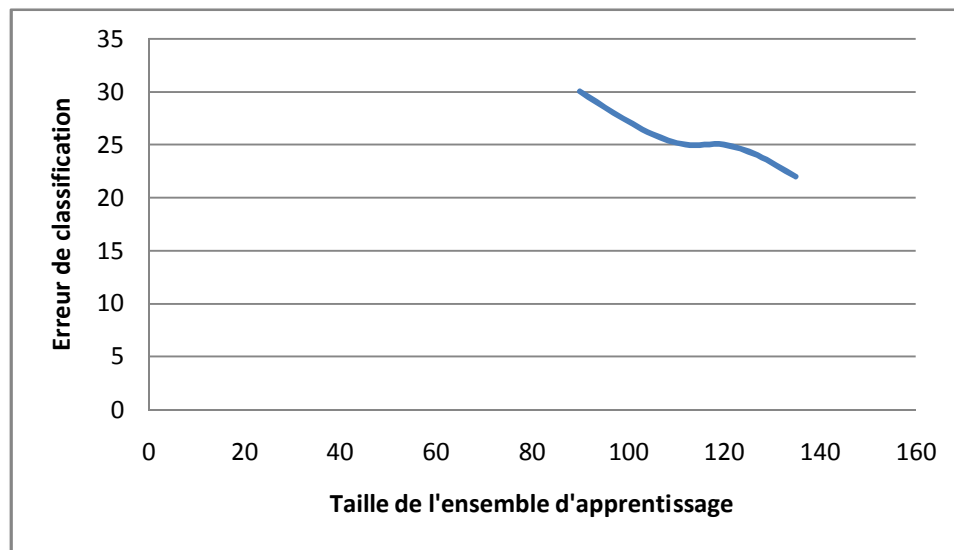


Figure 41 : Erreur de classification de l'arbre de décision en fonction de la taille de l'ensemble d'apprentissage

On remarque, sur le graphique ci-dessus, que le taux d'erreur de l'arbre de décision diminue au fur et à mesure que la taille de l'ensemble d'apprentissage augmente. On remarque, également, que le meilleur taux d'erreur est toujours supérieur à 20%, ce qui est relativement élevé. Ceci peut être dû au faible nombre de points de coupures que nous avons choisi pour la discrétisation des attributs (égal à 1).

Conclusion :

L'algorithme ID3 est inefficace quand le nombre de points de coupure est petit ; il faut donc augmenter ce nombre si l'on veut améliorer la performance de l'algorithme.

4.1.3. Comparatif des algorithmes de classification

Les tests sont effectués sur les deux benchmarks IRIS et ESI-SIQ-SI avec un ensemble d'apprentissage de 80% et un ensemble de test de 20%.

Le graphique suivant montre une comparaison entre l'algorithme k-NN ($k=50$) et l'arbre de décision en terme de taux d'erreur commise (%).

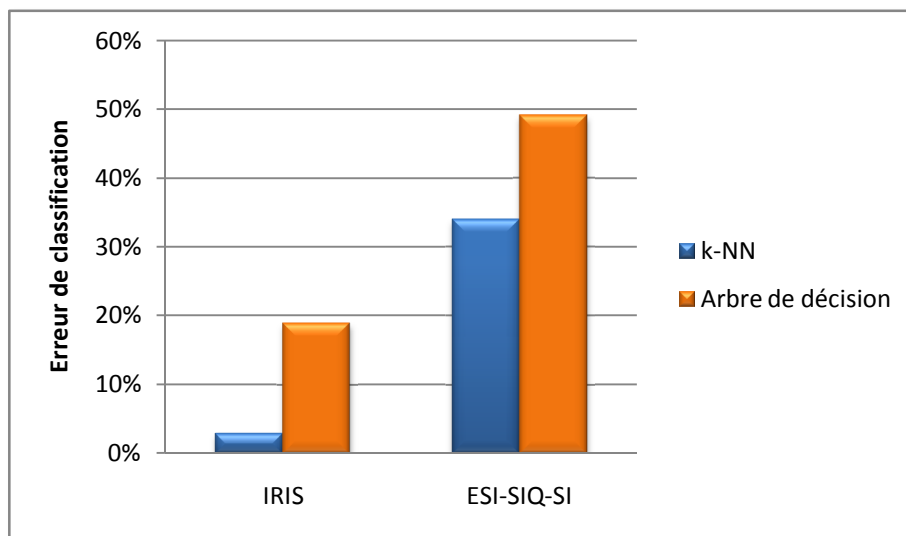


Figure 42 : Comparaison du taux d'erreur des algorithmes de classification

On remarque, sur le graphique ci-dessus, que l'algorithme k-NN est toujours meilleur que l'arbre de décision, et ce dans les deux tests effectués. On remarque également que les deux algorithmes présentent un taux d'erreur relativement élevé dans le test effectué sur le benchmark ESI-SIQ-SI (taux d'erreur > 30%). Ceci s'explique par le fait que la partition ESI-SIQ-SI contient des classes qui sont très entrelacées, ce qui complique la tâche de classification.

Conclusion :

L'algorithme k-NN donne toujours de meilleurs résultats que l'arbre de décision.

4.2. Tests sur les benchmarks

Le tableau ci-dessous montre les résultats des algorithmes de classification sur un ensemble de test constitué de six individus. L'ensemble d'apprentissage est considéré comme étant l'ensemble des individus du benchmark ESI-SIQ-SI. L'ensemble du test est formé de données réelles obtenues auprès des étudiants de deuxième année (2009/2010). L'algorithme k-NN est configuré avec $k=50$. Pour l'arbre de décision, nous avons fixé les valeurs de coupure de tous les modules à 10.

Etudiant n°	RESIDANT	SERIEBAC	IECO1	ANG1	STRM1	ALGO1	MATHS1	ELECT1	DECISION1	MONTION1	SYSEX2	MATHS2	SYSIN2	LOGM2	ALGO2	STRM2	ANG2	STAT2	Classe k-NN	Classe AD
1	0	5	18	17	15	14	15	17	1	2	10	15	12,5	13	5	12	15	15	SI	SI
2	0	5	16	17	11	10	11	11	1	3	9	10	8	9	9	9	17	11	SI	SIQ
3	0	4	13	13	12	10	11	9	1	3	11	9	12	8	7	11	12	10	SIQ	SIQ
4	1	4	15	14,5	8	9,5	8	11,5	1	3	4,5	9	8	11,5	2	7,5	10,7	4	SIQ	SI
5	1	4	11	12	10	8,5	10	15,5	1	3	17	12	5	12	4	14	10	14	SI	SI
6	0	5	18	17,6	15,1	14,7	11,6	14	1	2	16	12	10	13,5	11	13	16	13,5	SI	SI

Tableau 18 : Tests de classification sur le benchmark ESI-SIQ-SI

5. Tests des règles d'association

Les tests qui suivent sont effectués sur l'algorithme Apriori. L'objectif est de déterminer les meilleurs paramètres de l'algorithme qui permettent de réaliser un bon compromis entre le temps d'exécution et le nombre de règles intéressantes générées.

5.1. Tests sur l'algorithme Apriori

Le graphe ci-dessous montre la variation du nombre de règles générées en fonction du support minimum (SupMin). Les résultats sont obtenus en appliquant l'algorithme Apriori sur le benchmark ESI-1TRC avec une confiance minimum fixée à 0,01.

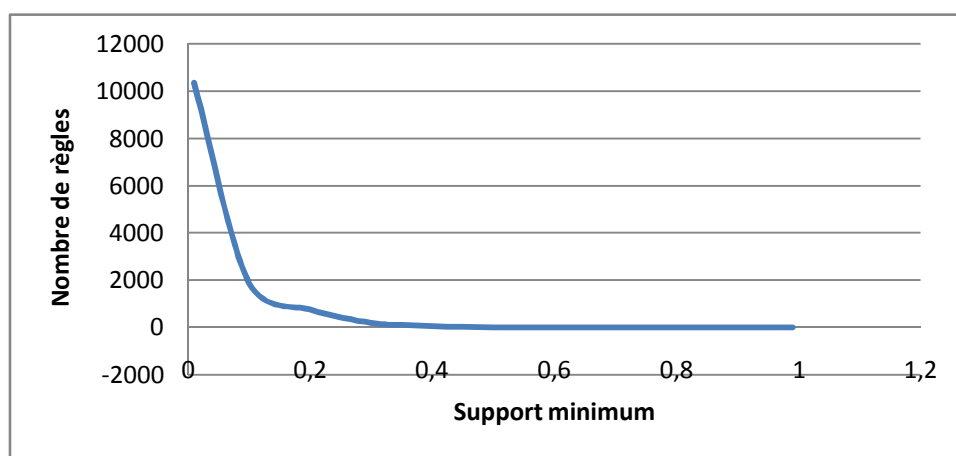


Figure 43 : Nombre de règles de l'algorithme Apriori en fonction du support minimum

On remarque, sur le graphique ci-dessus, que le nombre de règles générées baisse quand le support minimum augmente. Ce nombre s'annule au bout d'un support minimum qui dépasse légèrement 0.5 et devient très grand quand ce dernier descend en dessous de 0.2.

Conclusion :

Un support minimum supérieur à 0.2 permet de limiter considérablement le nombre de règles générées.

Un support minimum inférieur à 0.2 permet de générer plus de règles qui sont "rares" (se produisent chez moins de 20% de la population).

Le graphique ci-dessous montre la variation du nombre de règles en fonction de la confiance minimum (ConfMin). Les résultats sont obtenus en appliquant l'algorithme Apriori sur le benchmark ESI-1TRC avec un support minimum fixé à 0,3.

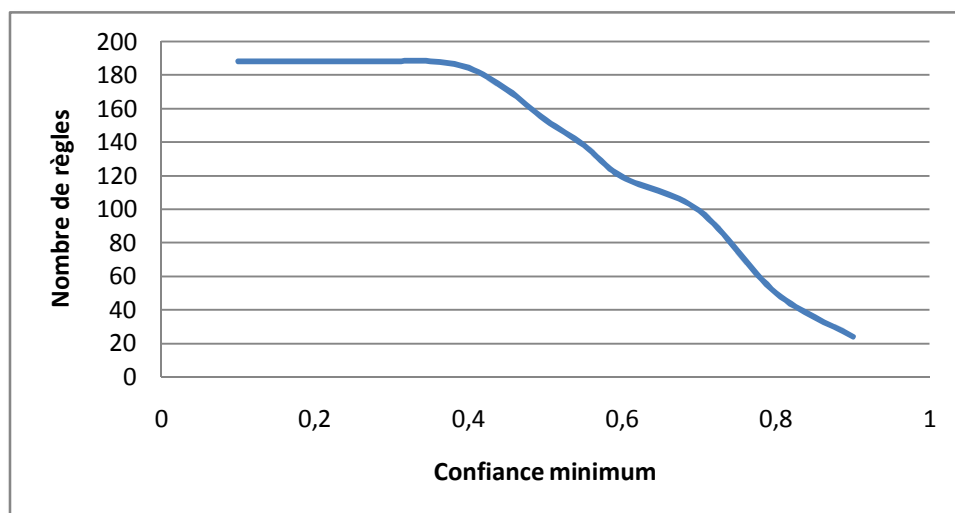


Figure 44 : Nombre de règles de l'algorithme Apriori en fonction du support minimum

On remarque, sur le graphique ci-dessus, que les mêmes règles sont générées pour une confiance minimum variant entre 0,1 et 0,4 ; i.e. que les règles intéressantes se trouvent en dehors de cet intervalle. On constate également qu'une confiance minimum dépassant légèrement 0,4 permet de filtrer les règles pour n'en garder que les plus intéressantes.

Conclusion :

Une confiance minimum supérieure à 0,4 permet d'éliminer les règles non intéressantes.

Le graphe suivant montre la variation du temps d'exécution (ms) de l'algorithme Apriori en fonction de la dimension des individus (nombre d'attributs). Les résultats sont obtenus en appliquant l'algorithme Apriori sur le benchmark ESI-1TRC avec une confiance minimum égale à 0,7 et un support minimum égal à 0,3. La réduction de dimension (suppression d'attributs) se fait de manière arbitraire.

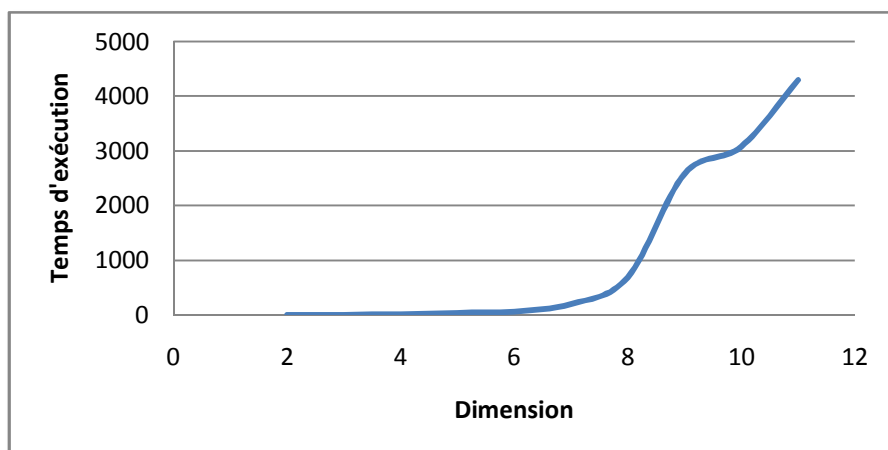


Figure 45 : Temps d'exécution de l'algorithme Apriori en fonction de la dimension

On remarque, sur le graphique ci-dessus, que le temps d'exécution varie proportionnellement à la dimension de la population ; il monte en flèche quand cette dernière dépasse 7 et devient prohibitif quand elle dépasse une dizaine.

Conclusion :

Apriori est un algorithme très sensible au nombre d'attributs. Il est donc fortement recommandé de penser à une réduction de la dimension avant d'appliquer l'algorithme.

1.1. Tests sur les benchmarks

1.1.1. Benchmark SUPERETTE

L'application de l'algorithme Apriori sur le benchmark SUPERETTE avec les paramètres (MinSup = 0,3 et MinConf = 0,7) donne 13465 règles d'association. Les valeurs de coupure de tous les attributs étant fixées à 0,5, l'article sera présent sur le ticket si sa valeur est supérieure à 0,5. Quelques règles, parmi les plus intéressantes, sont représentées dans le tableau suivant :

Antécédent	Conséquent	Support	Confiance
CONSERVE	PATE	0,83	0,97
LAIT	CAFE_THE	0,78	0,94
PAIN & FROMAGE	POISSON	0,56	0,95
FROMAGE	DESSERT	0,62	0,84
PATE	CONSERVE	0,83	0,92
PATE	POISSON	0,80	0,89
LAIT & CAFE_THE	SUCRERIE	0,56	0,72
À			

Tableau 19 : Résultats d'application de l'algorithme Apriori sur le benchmark SUPERETTE

1.1.2. Benchmark ESI-3SIQ

L'application de l'algorithme Apriori sur le benchmark ESI-3SIQ avec les paramètres (MinSup = 0,3 et MinConf = 0,7) donne 925 règles d'association. Les valeurs de coupure de tous les modules ont été fixées à 10. Quelques règles, parmi les plus intéressantes, sont représentées dans le tableau suivant :

Antécédent	Conséquent	Support	Confiance
ANUM<10	ANG<10	0,62	0,91
RO>10	SYSEX>10	0,41	0,73
THL>10	SYSEX>10	0,34	0,85
ELECT<10	TELET<10	0,51	0,80
SYSEX<10 & THL<10	ANUM<10	0,33	0,87
ELECT<10	STRM<10 & ANG<10	0,51	0,80
À			

Tableau 20 : Résultats d'application de l'algorithme Apriori sur le benchmark ESI-3SIQ

6. Conclusion

Nous avons réussi, à travers les séries de tests effectués, à suivre le comportement des algorithmes de data mining, que nous avons programmés, face à différents jeux de données réelles. Ces tests nous ont également permis de déterminer les meilleurs paramètres de chaque algorithme.

D'abord, les tests effectués sur l'algorithme génétique ont montré que le codage par vecteur s'adapte mieux quand le nombre d'itérations est important, contrairement au codage par classe où le nombre d'itérations agit directement sur le temps d'exécution. Nous avons également conclu que la taille de la population de solutions ne doit pas être trop grande ni trop faible pour réaliser un bon compromis entre le temps d'exécution et la qualité du résultat.

Nous avons montré, à travers les tests comparatifs, que l'algorithme CLARA présente le meilleur temps d'exécution face à des populations de grande taille alors qu'il cède la place à k-Means face à des populations de grande dimension. Nous avons aussi montré que l'AG, codage par classe, donne la meilleure qualité de clustering. Enfin, nous avons déduit que le nombre de classes doit être fixé par l'utilisateur en fonction de ses besoins.

Ensuite, les tests effectués sur les algorithmes de classification ont montré qu'un ensemble d'apprentissage de 80% donne un taux d'erreur raisonnable. Nous avons montré, à travers ces tests, que la performance de l'algorithme k-NN est très dépendante du paramètre k ; ce dernier ne peut être fixé qu'après plusieurs essais avec différentes valeurs. Néanmoins, 1-NN s'est révélé être très efficace quand on ne sait pas la meilleure valeur de k .

Les tests effectués sur les arbres de décision ont montré leur grande sensibilité et la comparaison avec l'algorithme k-NN a donné l'avantage toujours à l'algorithme k-NN.

Enfin, les tests effectués sur l'algorithme Apriori ont révélé que le temps d'exécution de cet algorithme est très fortement lié au nombre d'attributs. Il est donc indispensable de réduire la dimension quand celle-ci est très grande avant d'appliquer l'algorithme.

Conclusions et perspectives

Au terme de ce mémoire, nous procédons dans les lignes qui suivent à un récapitulatif du travail effectué. Rappelons que notre travail consistait à développer une application qui permet d'appliquer des algorithmes de data mining pour la classification automatique des données et l'extraction des règles d'association.

Nous nous sommes attelés, dans un premier temps, à définir les différents concepts liés au data mining et au processus ECD. Nous avons montré que le data mining n'est qu'une étape de traitement au sein du processus ECD.

Dans un second temps, nous avons procédé à un état de l'art des différentes techniques et algorithmes de data mining. Nous avons vu qu'il en existe une grande variété. Chaque algorithme est adapté à un contexte particulier ; il peut donc réussir dans un contexte et échouer dans un autre. Nous avons conclu que le choix du bon algorithme se fait en fonction des données et des besoins.

La troisième partie de ce mémoire était consacrée à l'implémentation des algorithmes, entre autres, l'algorithme génétique appliqué pour résoudre le problème de clustering. A ce niveau, nous avons aussi présenté l'architecture générale de notre application avant de détailler la structure et le fonctionnement des différents modules.

La dernière partie de ce mémoire portait sur les tests et les résultats de tous les algorithmes implémentés. Les tests sur les algorithmes de clustering nous ont permis de ressentir la grande difficulté de choisir le bon algorithme. Il est donc nécessaire d'effectuer plusieurs essais avec différents paramètres avant de choisir l'un ou l'autre des algorithmes. Néanmoins, nous avons constaté que l'algorithme k-Means et l'AG ont réussi dans la plupart des tests effectués et ont donné des résultats plus ou moins bons.

Les tests sur les algorithmes de classification ont montré la grande sensibilité des arbres de décision et ont donné l'avantage toujours à l'algorithme k-NN. Ce dernier s'est avéré être très efficace pourvu qu'il soit bien paramétré.

Les tests que nous avons effectués sur l'algorithme Apriori ont montré sa faiblesse, en terme du temps d'exécution, devant des populations de grande dimension. Il est donc indispensable de procéder à une réduction de dimension avant d'exécuter l'algorithme.

Ce travail n'a pas été exempt d'obstacles. En effet, nous avons été confrontés à différentes difficultés au niveau de l'implémentation de l'algorithme génétique, notamment les opérations génétiques.

Comme perspectives par rapport à ce travail, il serait intéressant de rajouter un algorithme de clustering à base de densité (DBSCAN par exemple) et un algorithme hiérarchique descendant (comme DIANA). Il serait aussi intéressant de renforcer avec un autre algorithme de recherche d'associations (comme FP-Growth) pour pouvoir comparer les résultats avec l'algorithme Apriori.

Une autre perspective consiste à étendre le champ d'application pour intégrer d'autres domaines comme la gestion du risque dans les prêts bancaires, le diagnostic médical, la fidélisation des clients dans les sociétés commerciales...etc.

Bibliographie

- [1]. Djamel Abdelkader ZIGHED, Ricco RAKOTOMALALA. "*Extraction de connaissances à partir de données (ECD)*". Article en ligne. 2002. (Consulté le 03.01.2010). Disponible sur : <http://www.techniques-ingenieur.fr/book/h3744/extraction-de-connaissances-a-partir-de-donnees--ecd-.html>
- [2]. K. J. Cios, W. Pedryecz, R. W. Swinniarsky, L. A. Kurgan. "*Data Mining : A Knowledge Discovery Approach*". Ouvrage. Editions Springer Science. 2007.
- [3]. Stéphane Tufféry. "*Data Mining et statistique décisionnelle*". Cours en ligne Master 2 Data Mining. Université de Rennes. (Consulté le 12.03.2010). Disponible sur : <http://data.mining.free.fr>
- [4]. Riadh Ben Messaoud. "*Data mining*". Rapport de Licence C.E.STAT. Institut Universitaire de Technologie Lumière, université Lumière - Lyon 2. Avril 2007.
- [5]. Marc Tommasi, Rémi Gilleron. "*Découverte de connaissances à partir de données*". Article en ligne. GRAppA (Groupe de Recherche sur l'apprentissage Automatique), Université Charles de Gaulles, Lille 3. 2000. (Consulté le 20.01.2010). Disponible sur : <http://www.grappa.univ-lille3.fr/polys>
- [6]. René Lefébure, Gilles Venturi. *Le Data Mining* Ouvrage. Edition Eyrolles. 1998.
- [7]. Hammouda Mohamed. "*Utilisation des techniques de data mining pour La modélisation du parcours scolaire et la prédiction du succès et du risque d'écchec*". Mémoire de Magister. Université de Saad Dahleb Blida, département d'Informatique. 2009.
- [8]. Stéphane Tufféry. *Data mining et statistiques décisionnelles, l'intelligence dans les bases de données* Ouvrage. Edition TECHNIP. 2005.
- [9]. Larbi Hariche, Mohamed Oubenami, Yekhlief Ziani. "*Méthodologie data mining*". Article en ligne. (Consulté le 29.04.2010). Disponible sur : <http://www.eisti.fr/~lassi/PFE/ID/DataMining/SITE>
- [10]. Wikipédia, l'encyclopédie libre. "*Exploration de données*". Article en ligne. (Consulté le 03.12.2009). Disponible sur : http://fr.wikipedia.org/wiki/Exploration_de_données
- [11]. Michel J. A. Berry, Gordon Linoff. *Data mining, techniques appliquées au marketing, à la vente et aux services clients* Ouvrage. Editions InterEditions. 1997.
- [12]. Jiawei Han, Micheline Kamber. *Data Mining, concepts and techniques* Ouvrage. Edition Morgan Kaufmann Publishers. 2000.
- [13]. XindongWu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, Dan Steinberg. "*The Top Ten Algorithms in Data Mining*". Ouvrage. Editions Springer-Verlag. 2007.

- [14] Michel VOLLE. "*Intranet et Datamining*". Article en ligne. 2001. (Consulté le 21.03.2010). Disponible sur : <http://www.volle.com/lectures/ACM1.htm>
- [15] Guy Carrel Ngongang Badjio. "*Etude des outils de data mining dans les systèmes d'information décisionnelles*". Mémoire de fin d'études d'ingénieur de conception en informatique, ENSP. 2006.
- [16] Nicola Beck. "*Application de méthodes de clustering traditionnelles et extension au cadre multicritère*". Mémoire d'ingénieur. Université Libre de Bruxelles, faculté des sciences appliquées. 2006.
- [17] Le Anh Tuan, Ifi Hanoi. "*Réduction de base de données par la classification automatique*". Rapport de stage. Institut de la francophonie pour l'informatique (IFI). 2004.
- [18] Chareles Christophe. "*Une méthode rapide à la navigation fondée sur k-means, algorithme de classification non supervisée*". 2004.
- [19] Jiawei Han, Micheline Kamber. "*DM Book*". Ouvrage. Editions Elsevier Inc. 2006.
- [20] Clark F. Olson. "*Parallel algorithms for hierarchical clustering. Parallel Computing*". Ouvrage. Edition Elsevier Science Publishers. 1995.
- [21] Nicolas Pasquier, "*Data Mining, Clustering*". Cours en ligne, Université de Nice - Sophia-Antipolis. (Consulté le 21.04.2010). Disponible sur : <http://www.i3s.unice.fr/~pasquier/web/?download=m2ntdp.dm.cours.clustering.pdf>
- [22] Pierre Emmanuel Jouve. "*Apprentissage Non Supervisé et Extraction de Connaissance à partir de Données*". Thèse de Doctorat en Informatique. Université Lumière Lyon 2. 2003.
- [23] Pavel Berkhin. "*Survey of clustering data mining techniques*". Article en ligne. Accrue Software Inc. 2002. (Consulté le 03.01.2010). Disponible sur : http://www.ee.ucr.edu/~barth/EE242/clustering_survey.pdf
- [24] E-G Talbi. "*Fouille de données (Data Mining) - Un tour d'horizon*". Présentation en ligne. Laboratoire d'informatique fondamentale de Lille (LIFL). (Consulté le 02.11.2009). Disponible sur : <http://www2.lifl.fr/~talbi/Cours-Data-Mining.pdf>
- [25] Martine Collard, Florent Masseglia, Nicolas Pasquier. "*Fouille de données, Biologie et Sécurité*". Cours en ligne. 2008. (Consulté le 21.02.2010). Disponible sur : <http://www.i3s.unice.fr/~mcollard/m2ifi>
- [26] R.T. Ng, J. Han. "*Efficient and effective clustering methods for spatial data mining*". In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, pages 144-155, Santiago de Chile, Chile. 1994.

- [27]. Sofian Maabout. "*Regroupement (Clustering)*". Présentation en ligne. Laboratoire Bordelais de Recherche en Informatique (LaBRI). (Consulté le 02.02.2010). Disponible sur : http://www.labri.fr/perso/maabout/dess_isc/clustering.ppt
- [28]. Damien Masse. "*Arbres de décision*". Cours en ligne, Master 2 ISC. Laboratoire d'Informatique des Systèmes Complexes, Université de Brest. (Consulté le 13.05.2010). Disponible sur : http://www.lisyc.univ-brest.fr/pages_perso/dmasse/cours/cours_decision.pdf
- [29]. Fabien Moutarde. "*Algorithmes Evolutionnistes (et autres heuristiques d'inspirations biologiques)*". Cours en ligne. Centre de robotique (CAOR), Ecole des mines de Paris. 2008. (Consulté le 03.03.2010). Disponible sur : http://www.ensmp.fr/~moutarde/slides_algosEvolutionnistes.pdf
- [30]. Fabien Moutarde. "*Brève introduction aux arbres de décision*". Cours en ligne. Centre de Robotique (CAOR), Ecole des Mines de Paris. 2008. (Consulté le 13.01.2010). Disponible sur : www.ensmp.fr/~moutarde/slides_AD.pdf
- [31]. Maria Malek. "*Fouille de données : Approche de règles d'association*". Cours de deuxième année, EISTI. (Consulté le 01.12.2009). Disponible sur : <http://www.eisti.fr/~mma/HTML-IA/Cours/Cours4/ia03.pdf>
- [32]. Cheikh Talibouya Diop, Moussa Lo, Fatou Kamara Sangaré. "*Intégration de règles d'association pour améliorer la recherche d'informations XML*". Article. Laboratoire d'Analyse Numérique et d'Informatique UFR de Sciences Appliquées et de Technologie, Université Gaston Berger, Sénégal. 2007.
- [33]. Guillaume Calas. "*Etude des principaux algorithmes de data mining*". Article en ligne. Ecole d'ingénieurs en informatique (EPITA). 2009. (Consulté le 04.04.2010). Disponible sur : <http://guillaume.calas.free.fr/data/Publications/DM-Algos.pdf>
- [34]. Fabrice Muhlenbach. "*Evaluation de la qualité de la représentation en fouille de données*". Thèse de doctorat, Université Lumière Lyon II, 2002.
- [35]. J. Dréo, A. Petrowski, P. Siarry, E. Taillard. "*Métaheuristiques pour l'optimisation difficile*". Ouvrage. Editions Eyrolles. 2005.
- [36]. Fatima-Zohra Kettaf, Jean-Pierre Asselin de Beauville. "*Des algorithmes évolutionnaires pour la classification automatique*". Article en ligne. (Consulté le 15.02.2010). Disponible sur : <http://www-rocq.inria.fr/axis/modulad/archives/numero-25/Kettaf-25/Desalgorithmesevolutionnaires.pdf>
- [37]. Krzysztof Cios, Witold Pedrycz, Roman Swiniarski, "Data Mining Methods for Knowledge Discovery". Ouvrage. Edition Kluwer Academic Publishers. 2000.
- [38]. Michael Marshall. "*IRIS Data Set*". Fichier de données en ligne. (Téléchargé le 02.05.2010). Disponible sur : <http://archive.ics.uci.edu/ml/datasets/Iris>

Annexe : Benchmarks

Echantillon du benchmark IMAGE

```
IMAGE_BALLONS 9216
NbAttributs 5
Labels Bleu Vert Rouge X Y
Coupures 1 1 1 1 1
0 -1 58 65 58 0 0
1 -1 57 64 57 0 1
2 -1 57 64 57 0 2
3 -1 57 64 57 0 3
4 -1 58 65 58 0 4
5 -1 57 64 57 0 5
6 -1 56 63 56 0 6
7 -1 54 61 54 0 7
8 -1 57 61 55 0 8
9 -1 56 60 54 0 9
...
```

Echantillon du benchmark ESI-1TRC

```
ESI-1TRC 1284
NbAttributs 12
Labels RESIDANT SERIEBAC MATHS IECO ELECT ANG ALGO STRM MOYENNE DECISION MENTION RATTRAP
Coupures 0,5 6 10 10 10 10 10 10 10 4 4 3
05/0200 -1 1 5 6,92 1,5 12,33 7,17 12,5 11 9,44 1 6 3
05/0076 -1 1 5 9,08 11,33 9,41 18,67 12,5 11,67 11,46 1 4 0
05/0081 -1 1 6 6,33 5,58 8,33 12 10 10,92 8,81 1 6 3
05/0194 -1 0 5 5 9,08 7,16 8,83 10 10,92 8,32 3 6 4
04/0240 -1 1 4 4,42 1 6,08 4,25 8,5 5,33 5,49 4 8 0
05/0263 -1 1 5 10,5 13 14,5 10 12 13 12,2 1 3 0
05/0275 -1 0 5 7,58 8,25 10,41 10,58 10,5 10 9,53 2 6 2
04/0120 -1 1 5 4,33 12 3,66 12,08 10 10 7,93 4 6 2
04/0381 -1 1 7 8,75 11 11,58 11,58 12,5 12,75 11,31 1 4 0
05/0247 -1 1 4 9,83 5,83 17,16 8,58 8 8,83 10,09 1 4 0
...
```

Echantillon du benchmark ESI-2TRC

```
ESI-2TRC 956
NbAttributs 14
Labels RESIDANT SERIEBAC STATS ANG MATHS SYSEX SYSIN LOGM STRM ALGO MOYENNE DECISION MENTION
RATTRAP
Coupures 0,5 6 10 10 10 10 10 10 10 10 10 4 4 3
04/0106 -1 1 4 8,83 11,42 10,17 14,14 8 12 12,42 12,6 11,28 1 4 0
04/0343 -1 1 4 11,75 15,33 15,33 13,6 8 12,17 18,88 11,26 13,1 1 3 0
04/0397 -1 1 6 5,67 14,67 10 8,57 8,67 9,33 12,08 9,26 9,55 1 6 5
04/0358 -1 1 5 11 12,25 10,75 13,43 10,17 13,83 15,17 11,6 12,23 1 3 0
04/0339 -1 0 5 8 13,83 6 13,29 7,83 10,33 11,25 11,93 10,14 1 4 0
03/0097 -1 0 5 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0 5 8 0
03/0079 -1 0 5 7,67 16,17 8,92 9,71 8,83 9,5 12,63 6,67 9,48 1 6 6
04/0129 -1 1 4 9 8,67 10,58 7,6 4,17 7,5 11,33 5,93 7,99 3 6 6
...
```

Echantillon du benchmark ESI-3SI

```
ESI-3SI 379
NbAttributs 14
Labels RESIDANT SERIEBAC ANG ANUM BDD FAT RO SYSEX MCSI GEST MOYENNE DECISION MENTION
RATTRAP
Coupures 0,5 6 10 10 10 10 10 10 10 10 10 4 4 3
03/0037 -1 0 5 13,83 11,75 14,14 16 12,5 16,17 12,3 16,83 13,95 1 3 0
03/0297 -1 0 5 14,92 11,5 15,57 16,67 14 13 10,6 16,87 13,88 1 3 0
03/0032 -1 0 4 13,25 13 15,71 14,67 10 15,67 11,5 16,04 13,52 1 3 0
03/0155 -1 0 5 14,75 13,25 14,71 14,5 10,5 14,17 9,6 15,46 13,05 1 3 0
03/0206 -1 0 4 12,67 9,75 14,14 15,17 14,5 14,33 10,4 15,04 13,01 1 3 0
02/0194 -1 0 5 12,25 9,75 14,57 13,5 11,5 14,83 10,8 14,75 12,55 1 3 0
03/0006 -1 0 5 11,75 12,75 13,29 16,5 8 10,83 10,7 15 12,19 1 3 0
03/0182 -1 0 5 12,5 8,5 14,14 10,33 12 14,5 10,2 16,25 12,19 1 3 0
...
```

Echantillon du benchmark ESI-3SIQ

```
ESI-3SIQ 413
NbAttributs 14
Labels RESIDANT SERIEBAC SYSEX RO ANUM THL STRM ANG TELET ELECT MOYENNE DECISION MENTION
RATTRAP
Coupures 0,5 6 10 10 10 10 10 10 10 10 10 4 4 3
0001 -1 0 4 10,5 10,93 14,75 12,33 11,64 8,42 10,17 12,25 11,41 1 4 0
0002 -1 1 7 8,75 7,29 11,33 9,67 10,13 8,5 8,5 10,81 9,35 2 6 5
0003 -1 0 4 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0 5 8 0
0004 -1 0 5 9,75 10,21 9 9,67 8,14 14,08 10,92 10,44 10,13 1 4 0
0006 -1 0 5 10,5 8,86 10,25 12,17 11,97 10,83 10,5 11,5 10,8 1 4 0
0007 -1 0 5 7 10,71 12 11,17 11,91 11,5 10 8,06 10,1 1 4 0
0008 -1 0 4 14,38 13,43 16,42 15,17 15,57 13,58 10,5 13,63 13,97 1 3 0
0009 -1 1 5 11,75 11,86 16,25 14,83 14,36 11,83 10,33 14,06 13,04 1 3 0
0010 -1 0 8 10,88 8,21 10,25 4,83 10,83 12,17 10,17 10 9,64 1 6 2
...
```

Echantillon du benchmark ESI-4SI

```
ESI-4SI 303
NbAttributs 13
Labels RESIDANT SERIEBAC ANALD ORGA RO MCP TELET MCSI GEST MOYENNE DECISION MENTION RATTRAP
Coupures 0,5 6 10 10 10 10 10 10 10 10 4 4 3
02/0202 -1 0 4 15 15,25 14,83 15,5 15,46 12,3 16 14,64 1 2 0
02/0130 -1 0 5 15,25 15,5 14,08 15,3 16,2 12 16 14,62 1 2 0
02/0211 -1 0 5 14,04 15,5 14,08 14,3 15,6 13,2 16 14,56 1 2 0
02/0025 -1 0 5 13,92 14,25 13,67 16,9 16,44 12,7 16 14,55 1 2 0
02/0172 -1 0 5 14,42 13 15 15,9 16,34 11,6 15 14,14 1 2 0
02/0122 -1 0 5 12,33 12,25 16,42 14,6 16,28 13,7 12 13,89 1 3 0
01/0050 -1 0 4 13,08 14,5 13,83 12,5 16,22 12,9 13 13,7 1 3 0
02/0248 -1 0 5 12,17 12,75 16,58 13,5 16,46 11,1 14 13,56 1 3 0
01/0085 -1 0 7 11,33 15,5 16,75 12,5 15,98 11,7 12 13,55 1 3 0
...
```

Echantillon du benchmark ESI-4SIQ

```
ESI-4SIQ 341
NbAttributs 14
Labels RESIDANT SERIEBAC FAT MCP BDD SYSEX AUTO ANALD COMP ARCHO MOYENNE DECISION MENTION
RATTRAP
Coupures 0,5 6 10 10 10 10 10 10 10 10 10 4 4 3
02/0005 -1 0 5 11,83 15,5 17,64 14,08 12,17 14,17 15,88 14,5 14,72 1 2 0
02/0008 -1 0 5 8,83 10 11,86 11,33 7,83 10,17 13,25 18,25 11,62 1 4 0
02/0019 -1 0 5 9 12 13,07 13,67 12 10,83 14,19 18,25 13,09 1 3 0
02/0020 -1 0 5 9,83 14 14,36 12,67 9 11,5 14 15,63 12,84 1 3 0
01/0258 -1 0 8 8,33 7,5 8,57 9,17 5,17 9,17 7 11,38 8,26 2 6 7
02/0043 -1 0 4 11,5 16,5 15,57 12,25 10,17 10,67 13,5 15,38 13,25 1 3 0
02/0058 -1 0 4 8,33 13 11,14 14,33 9,5 11,17 13,13 15,88 12,33 1 3 0
02/0059 -1 0 4 15,5 15 16,57 13,58 7,5 11,17 14,63 18 14,14 1 2 0
00/0161 -1 0 7 5 13 7,71 11,75 4,5 7,5 10,88 10,75 9,29 4 6 4
...
```

Echantillon du benchmark ESI-SIQ-SI

```
ESI-SIQ-SI 396
NbAttributs 19
Labels RESIDANT SERIEBAC MATHS1 IEC01 ELECT1 ANG1 ALGO1 STRM1 STATS2 ANG2 MATHS2 SYSEX2
SYSIN2 LOGM2 STRM2 ALGO2 DECISION MENTION RATTRAP
Coupures 0,5 6 10 10 10 10 10 10 10 10 10 10 10 10 4 4 3
05/0178 0 0 5 6,33 5,17 13,5 14,33 10 9,33 10 12,75 7 10,86 11,25 10,17 10,67 11 1 4
05/0263 0 1 5 10,5 13 14,5 10 12 13 10,25 10,92 9,33 12,71 6,42 9,5 13 13,25 3 7
05/0289 0 1 5 13,42 11,83 16,67 12,5 7,5 11,92 6,5 14,67 13,5 8,14 8,75 12,5 9,25 8 3 6
05/0088 0 1 5 8 13,33 13,5 14,66 12,5 12,08 8 14,5 10,17 10,57 8,33 13 11,92 9,25 1 4
05/0080 0 1 7 9,25 12,33 15,66 12,37 12,5 12,33 10,83 15,58 11,67 13 12,75 10,5 12,5 14 1 3
05/0248 0 1 5 11,58 6,75 11,33 11 11,5 10,17 8,5 8,75 7,83 14,71 9 7,67 13,17 14 1 6
05/0068 0 0 5 12 11,83 16,33 15,67 10,5 13,58 10 13,92 11 13,64 8,92 11 10,58 8,75 1 4
05/0247 0 1 4 9,83 5,83 17,16 8,58 8 8,83 12,33 11,83 16 12,57 7,5 6,83 12,25 11 3 6
...
```

Echantillon du benchmark SUPERETTE

```
SUPERETTE 129
NbAttributs 11
Labels BOISSON EAU SUCRERIE PATE PAIN LAIT CAFE_THE POISSON FROMAGE DESSERT CONSERVE
Coupures 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0,5
1 -1 0 0 1 0 0 0 0 0 18 4 0
2 -1 0 0 0 0 0 0 1 0 0 0 0
3 -1 0 0 0 0 0 0 0 0 0 3 0
4 -1 1 0 0 0 3 0 0 2 0 0 0
5 -1 2 0 0 0 0 0 0 0 0 0 0
6 -1 1 6 1 0 0 1 2 0 1 0 0
7 -1 1 0 4 0 0 0 0 0 2 0 0
8 -1 1 0 0 1 0 0 0 3 2 0 1
9 -1 0 6 0 0 4 0 0 0 0 0 0
10 -1 0 0 0 0 3 0 0 0 0 0 0
...
```

Echantillon du benchmark IRIS

```
IRIS 150
NbAttributs 4
Labels LNG_SEP LRG_SEP LNG_PET LRG_PET
Coupures 0 0 0 0
0 0 5,1 3,5 1,4 0,2
1 0 4,9 3,0 1,4 0,2
2 0 4,7 3,2 1,3 0,2
81 1 5,5 2,4 3,7 1,0
86 1 6,7 3,1 4,7 1,5
87 1 6,3 2,3 4,4 1,3
82 1 5,8 2,7 3,9 1,25
125 2 7,2 3,2 6,0 1,8
126 2 6,2 2,8 4,8 1,8
127 2 6,1 3,0 4,9 1,8
...
```