République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et De la Recherche Scientifique
Université Mohamed El Bachir El Ibrahimi de Bordj-Bou-Arreridj Faculté des mathématiques et d'informatique
Département d'informatique



MEMOIRE
Présenté en vue de l'obtention du diplôme deMaster en informatique

Spécialité : RÉSEAUX ET MULTIMEDIA

THEME

Évaluation des algorithmes d'extraction des itemsets à haute utilité basés sur des représentations concises.

Présenté par : Mebarek Safa Maroua

Président : Ben Abid Sonia MCA à L'U. El Bachir El Ibrahimi-BBA
Examinateur : Saifi Abdelhamid MCA à L'U. El Bachir El Ibrahimi-BBA
Encadreur : Mr : Attia Abdelouahab Pr à L'U. El Bachir El Ibrahimi-BBA

Promotion : 2024/2025

Remerciement

En premier lieu, nous tenons à remercier notre DIEU "ALLAH", notre créateur de nous avoir donné pour accomplir ce travail.

Nous adressons nos vifs remerciements à nos encadreurs consultants Mr. Attia Abdelouahab et Mr.Bensaadi Mohamed, pour nous avoir guidés tout au long de ce travail, pour leur compréhension, leur patience, leur compétence, et leurs remarques qui nous ont été précieuses.

Nous présentons nos chaleureux remerciements aux enseignants du jury qui m'ont fait l'honneur de présider et d'examiner ce modeste travail.

Nous adressons nos sincères remerciements à tous les professeurs qui, par la qualité de leur enseignement et leur disponibilité, nous ont accompagnés tout au long de notre formation. Nous remercions également toutes les personnes ayant contribuées, de manière directe ou indirecte, à la réalisation de ce travail.

Dédicace

Je dédie ce mémoire à mes chers parents, qui ont toujours été présents à mes côtés, me soutenant sans relâche tout au long de ces longues années.

À mes frères, pour leur présence rassurante, leurs encouragements et leur soutien, même dans les moments les plus difficiles. Vous avez été pour moi des repères précieux, toujours là lorsque j'en avais besoin.

À mes deux neveux « Abed El Malek et Farah », source de joie et de motivation.

À toute ma famille qui m'a encouragé de près ou de loin,

Résumé

L’extraction des ensembles d’éléments à haute utilité (HUIM) constitue un enjeu majeur en exploration de données. Elle vise à identifier des combinaisons d’articles ayant un impact significatif sur des indicateurs clés tels que les ventes ou les profits. Face à l’explosion des volumes de données dans l’univers du Big Data, le développement d’algorithmes performants devient essentiel pour repérer ces ensembles à forte utilité de manière rapide et efficace.

Dans le cadre de cette étude comparative, nous avons évalué les performances de plusieurs algorithmes HUIM conçus pour produire une représentation concise des résultats. Des expérimentations ont été réalisées sur plusieurs jeux de données volumineux afin d’analyser les performances des algorithmes CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators et MinFHM en termes de temps d’exécution, de consommation mémoire et de nombre d’éléments utiles extraits.

Les résultats obtenus démontrent que ces algorithmes sont capables d’identifier efficacement des ensembles d’éléments à haute utilité, en affichant des performances notables, aussi bien en rapidité qu’en gestion optimale des ressources mémoire.

mots clés : *Big Data, Fouille de donnée , ensembles d'éléments à haute utilité, représentation concise*

ملخص

يُعدّ استخراج مجموعات العناصر ذات المنفعة العالية (HUIM) من المسائل الجوهرية في مجال تنقيب البيانات، إذ يهدف إلى تحديد مجموعات من العناصر التي تُحدث تأثيراً ملحوظاً على مؤشرات معينة مثل المبيعات أو الأرباح .ومع التزايد الهائل لحجم البيانات في عصر البيانات الضخمة، تبرز الحاجة الملحّة إلى تطوير خوارزميات فعّالة وقادرة على تنفيذ عمليات الاستخراج بسرعة وكفاءة.

في هذه الدراسة المقارنة، قمنا بتقييم أداء عدد من خوارزميات HUIM المصممة لتقديم تمثيل موجز ومضغوط للنتائج .وقد شملت التجارب استخدام مجموعات بيانات ضخمة لاختبار كفاءة الخوارزميات التالية MinFHM :، CHUI-MinerMax، CHUI-Miner ، و-HUCI Generators**، من حيث وقت التنفيذ، واستهلاك الذاكرة، وعدد العناصر عالية المنفعة المستخرجة.**

وقد أظهرت النتائج أن هذه الخوارزميات تتمتع بقدرة عالية على استخراج مجموعات العناصر ذات الفائدة الكبيرة، مع تحقيق أداء متميّز سواء من حيث سرعة التنفيذ أو الاستخدام الأمثل للموارد.

الكلمات المفتاحية : البيانات الضخمة، استخراج بيانات، التعدين ومجموعات العناصر عالية المنفعة ، التمثيل الموجز

Abstract

High Utility Itemset Mining (HUIM) is a critical task in data mining, focusing on identifying combinations of items that have a significant impact on specific metrics such as revenue or profit. With the exponential growth of data in the era of Big Data, the development of efficient algorithms for rapidly and accurately extracting high-utility itemsets has become increasingly essential.

In this comparative study, we evaluated the performance of various HUIM algorithms designed for concise representation. Extensive experiments were conducted on several large-scale datasets to assess the effectiveness of CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators and MinFHM in terms of execution time, memory consumption, and the number of high-utility itemsets discovered.

The results demonstrated that these algorithms are capable of efficiently mining high-utility patterns, delivering impressive performance both in speed and in optimized memory usage.

keywords : *Big Data , Data Mining , high-utility itemsets Mining , concise representation*

Sommaire

Sommaire

Sommaire.....-

Introduction Générale 1

Chapitre I

Généralité sur la Fouille de Donnée

1. Introduction: 5

2. L'extraction des connaissances à partir de donnée (ECD): 5

2.1 Définition : 5

1.2 Etapes du processus ECD : 5

3. Fouille de données (Data Mining) : 8

3.1 Définition : 9

3.2 L'évolution de la technologie système de BD : 9

3.3 Les tâches de la Fouille de Donnée : 10

3.4 Techniques du Fouille de Donnée : 12

3.5 Domaines d'application de la Fouille de Donnée : 14

4. Conclusion : 15

Chapitre II:

Extraction des ensembles d'items à haute utilité (HUIM)

1. Introduction : 18

2. Extraction d'ensembles d'items fréquents : .....**Erreur ! Signet non défini.**

2.1 Concepts Fondamentaux : 18

2.2. Défis de l'extraction des ensembles d'items fréquents : **Erreur ! Signet non défini.**

2.3. Algorithmes d'extraction des items fréquents : ..**Erreur ! Signet non défini.**

2.4. Recherche en largeur (Breadth-First Search) :....**Erreur ! Signet non défini.**

2.5. Recherche en profondeur (Depth-First Search) : **Erreur ! Signet non défini.**

2.6. Limitations de l'extraction des ensembles d'éléments fréquents. .....**Erreur ! Signet non défini.**

3. Extraction des ensembles d'items à haute utilité : .**Erreur ! Signet non défini.**

3.1. Concepts fondamentaux : .....**Erreur ! Signet non défini.**

3.2. Les principales propriétés du problème du (HUIM) :.... **Erreur ! Signet non défini.**

3.3. Algorithmes itemset à haute utilité : .....**Erreur ! Signet non défini.**

3.4 Analyse comparative des algorithmes HUIM :....**Erreur ! Signet non défini.**

4. Représentations concises des ensembles d'items à haute utilité :.....**Erreur ! Signet non défini.**

4.1. Concepts fondamentaux :.....**Erreur ! Signet non défini.**

4.2. Algorithmes de représentation concise : .....**Erreur ! Signet non défini.**

5. Conclusion :.....**Erreur ! Signet non défini.**

Chapitre III:

Réalisation & Interprétation

1. Introduction : 39

2. La présentation des outils de développement : 39

2.1. Eclipse : 39

2.2. Java :..... 39

2.3. Python :..... 39

2.4. Visual Studio :..... 40

2.5. Une bibliothèque de donnée extra source (SPMF) : 40

3. Les algorithmes proposes pour HUIM :..... 41

3.1. L'algorithme CHUI-Miner : .....**Erreur ! Signet non défini.**

3.2. L'algorithme CHUI-MinerMax : 43

3.3. l'algorithme HUCI-Miner-Generators : .....**Erreur ! Signet non défini.**

3.4. L'algorithme MinFHM : .....**Erreur ! Signet non défini.**

4. Conception et implémentation : 46

4.1. L'architecture générale : 46

4.2. le premier cas d'étude : 488

4.3. Le deuxième cas d'étude :..... 55

5. Conclusion :..... 59

Conclusion générale 60

Bibliographies 63

Liste des figures:

Figure	Description	Page
Figure I.1	Le processus ECD	6
Figure I.2	Classement des techniques du Data Mining	13
Figure II.3	Algorithme Apriori	23
Figure II.4	Espace de recherche pour I={a, b, c, d, e}	24
Figure II.5	Algorithme ECLAT	25
Figure III.6	Téléchargement du SPMF	42
Figure III.7	Algorithme CHUI-Miner	44
Figure III.8	Algorithme MinFHM	47
Figure III.9	Schéma des étapes de l'étude comparative proposée	49
Figure III.10	Base de donnée de produit cosmétique	50
Figure III.11	Eclipse IDE	51
Figure III.12	Un extrait d'un fichier format SPMF	52
Figure III.13	Créer un nouveau projet JAVA	52
Figure III.14	Ajouter SPMF	53
Figure III.15	Résultat de l'exécution de l'algorithme MinFHM	53
Figure III.16	Temps d'exécution des algorithmes (CHUI-Miner, CHUI-MinerMax, HUCI-Miner- Generators, MinFHM)	55
Figure III.17	La consommation du mémoire par les algorithmes (CHUI-Miner, CHUI-Miner, HUCI-Miner-Generators, MinFHM)	55
Figure III.18	Le Nombre des HUIM des algorithmes (CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators, MinFHM)	56
Figure III.19	La base FoodMart format SPMF	57
Figure III.20	Exécution de l'algorithme CHUI-Miner	58
Figure III.21	Résultats de l'exécution des algorithmes CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators, MinFHM	59

Liste des tableaux:

Tableau	Description	Page
Table I.1	Domaines d'application de la Fouille de Donnée	15
Table II.2	Base de donnée transactionnel	19
Table II.3	Résumé des algorithmes FIM	21
Table II.4	Base de donnée quantative des transactions	28
Table II.5	Valeurs d'utilité externe	28
Table II.6	Algorithmes populaires d'extraction d'ensembles d'items à haute utilité	34
Table II.7	Algorithmes pour extraire des représentations concises d'itemsets à haute utilité	36
Table III.8	Les statistiques de la base de donnée de produit cosmétique	50
Table III.9	Les statistiques de base de donnée (FoodMart)	57

Liste des formules :

Numéro de la formule	Définition de la formule	La formule	Page
1	1. Définition d'une base de données	D = (O, P, R)	18
2	Définition d'une transaction	$T = \{T_1, T_2, T_3, ..., T_n\}$	18
3	Itemset	$I = \{i_1, i_2, ..., i_n\}$	19
4	Support	$\mathbf{SUP(x)} = \frac{\lvert\{\mathbf{t} \in \mathbf{D}: \mathbf{X} \subseteq \mathbf{t}\}\rvert}{\lvert\mathbf{D}\rvert}$	19
5	Utilité d'un item	U(i,Tc)=p(i)×q(i,Tc)	28
6	Utilité d'un itemset dans une transaction	$U(X, Tc) = \sum i \in X$ (, Tc)	28
7	Utilité d'un itemset dans une base de données	$U(X,D) = \sum_{T \in D} \boldsymbol{U(X,T)}$	28
8	Utilité d'une transaction	$TU(Tc) = \sum_{i \in Tc} \boldsymbol{U(X,Tc)}$	29

Liste des abréviations :

Abreviations	Significations
KDD	knowledge discovery in databases.
ECD	Extraction des connaissances des données.
FIM	Frequent Itemset Mining.
HUIM	High Utility Itemset Mining.
HUI	High-Utility Itemset.
HUI-Miner	High-Utility Itemset- Miner.
TWU	The transaction-weighted utilization.
TU	The transaction utility.
UP-Growth	Utility Pattern Growth Algorithm.
CHUI	Closed High-Utility Itemset
MaxHUI	Maximal High-Utility Itemset.
GHUI	Generator High-Utility Itemset.
MinHUI	Minimal High-Utility Itemset.
SPMF	Sequential Pattern Mining Framework
EUCS	Estimated Utility Co-occurrence Structure

Introduction Générale

Introduction Générale

La fouille de données est un domaine apparu avec l'explosion des quantités d'informations stockées, grâce aux progrès importants des vitesses de traitement et des supports de stockage. La fouille de données vise à découvrir, parmi de vastes volumes de données, des informations précieuses permettant de mieux les comprendre ou de prédire le comportement des données futures. Le data mining utilise depuis son émergence divers outils statistiques et techniques d'intelligence artificielle pour atteindre ses objectifs.

L'exploration de données, ou extraction de connaissances à partir de données, constitue une solution efficace à cette problématique. Cette discipline interdisciplinaire s'appuie sur l'intégration de méthodes statistiques, de techniques d'apprentissage automatique et de principes de gestion de bases de données. Son objectif est de transformer de vastes ensembles de données en informations exploitables, en révélant des motifs, des relations et des tendances pertinentes. Ces connaissances extraites représentent un atout stratégique pour éclairer les décisions, anticiper les évolutions futures et approfondir la compréhension de phénomènes complexes.

Ce mémoire porte sur une étude approfondie des méthodes d'extraction de connaissances appliquées au domaine spécifique de la fouille d'ensembles d'items à haute utilité, également connu sous le nom de *High Utility Itemset Mining* (HUIM). Son objectif principal est d'analyser de manière détaillée les différents algorithmes utilisés pour identifier ces ensembles d'items, tout en garantissant une représentation concise et efficace. Face à la croissance exponentielle du volume de données dans le contexte du Big Data, il est devenu primordial de développer des algorithmes efficaces permettant d'extraire rapidement les ensembles d'items à haute utilité.

Le mémoire est divisé en trois chapitres principaux.Dans le premier chapitre, nous proposons une vue d'ensemble du data mining. Nous y introduisons les concepts fondamentaux ainsi que les principales techniques employées dans ce domaine. Nous retraçons ensuite l'évolution des systèmes de gestion de bases de données qui ont contribué à son développement. Enfin, nous présentons les différentes tâches liées à la fouille de données et mettons en évidence leurs multiples applications dans divers secteurs.

Le deuxième chapitre, intitulé « High Utility Itemset Mining (HUIM) », se concentre sur l'extraction d'ensembles d'items à haute utilité pour une représentation optimisée. Il explore les différents algorithmes développés pour

cette tâche, en présentant leurs concepts fondamentaux, leurs mécanismes de fonctionnement et leurs domaines d'application.

Le troisième chapitre, intitulé « Réalisation et interprétation », sera consacré à la mise en œuvre pratique des algorithmes CHUD (Closed High Utility Discovery), EFIM-Closed (Extended Frequent Itemset Mining), MinFHM et CHUI_MinerMax, ainsi qu'à l'interprétation des résultats obtenus. Il commencera par une explication des concepts fondamentaux de chaque algorithme, puis présentera leurs principales caractéristiques. Enfin, leur fonctionnement sera détaillé afin d'évaluer leur efficacité et leurs spécificités dans le contexte de l'extraction d'ensembles d'items à haute utilité.

Au cours de ce projet, nous avons utilisé plusieurs outils de développement, notamment Eclipse, Java, Python, Visual Studio, ainsi que la bibliothèque SPMF, dédiée à l'extraction d'ensembles d'items fréquents. Ces outils ont joué un rôle crucial dans la mise en œuvre et l'évaluation des algorithmes étudiés, nous permettant de vérifier leur efficacité et leurs performances.

Ce mémoire se conclut par une synthèse des principaux résultats obtenus, mettant en lumière les contributions de notre travail et proposant des pistes d'amélioration futures. Ces perspectives incluent l'optimisation des algorithmes existants, l'exploration de nouvelles approches pour l'extraction des ensembles d'items à haute utilité, ainsi que leur adaptation à d'autres domaines d'application afin d'élargir leur impact.

Chapitre 1
Généralité sur la Fouille de Donnée

1. Introduction:

Les techniques de fouille de données et d'intelligence artificielle sont largement utilisées dans de nombreux secteurs pour résoudre des problèmes complexes ou concevoir des solutions industrielles. En économie, elles permettent de prévoir des indicateurs clés, de révéler des informations cachées, de résoudre des problèmes complexes ou de développer des solutions industrielles. Dans le domaine de la santé, elles contribuent à détecter des maladies, personnaliser les traitements et analyser les tendances épidémiologiques. En finance, elles servent à identifier les fraudes, évaluer les risques et optimiser les investissements. En marketing, elles permettent de mieux comprendre les comportements des consommateurs et de personnaliser les offres. En gestion de la relation client, elles améliorent l'efficacité des interactions et facilitent l'acquisition de nouveaux clients. Combinées à l'intelligence artificielle, ces méthodes deviennent encore plus puissantes, capables d'apprendre, de s'adapter et de générer des solutions innovantes à partir de vastes ensembles de données. [1][2].

Dans ce chapitre, nous souhaitons identifier les diverses méthodes de data mining afin d'obtenir une vision globale de ces techniques, afin de déterminer les techniques adéquates pour les utiliser dans la résolution des problèmes identifiés.

2. L'extraction des connaissances à partir de donnée (ECD):

2.1 Définition :

Le KDD (Knowledge Discovery in Databases) est un processus structuré visant à extraire des informations utiles, inédites et potentiellement précieuses à partir de grands volumes de données (ECD : Extraction de Connaissances à partir de Données). Ce processus suit une approche itérative, où les différentes étapes allant de la sélection et du prétraitement des données jusqu'à l'extraction et à l'interprétation des connaissances sont répétées afin d'assurer des résultats fiables et pertinents.

2.2 Etapes du processus ECD :

L'Extraction de Connaissances à partir de Données (ECD), également connue sous le nom de *Knowledge Discovery in Databases (KDD)*, est un processus méthodique visant à transformer de grandes quantités de données brutes en informations utiles et exploitables. Dans un contexte où les organisations collectent et stockent des volumes croissants de données, l'ECD s'impose comme une solution indispensable pour révéler des modèles, des relations et des tendances cachés au sein de ces données.

L'ECD repose sur une série d'étapes successives, commençant par la collecte et la préparation des données, incluant leur nettoyage, leur transformation et la sélection des informations pertinentes. Une fois cette étape achevée, les techniques d'exploration de données sont appliquées, mobilisant des méthodes issues de la statistique, de l'apprentissage automatique et de la reconnaissance de formes. Les résultats obtenus sont ensuite interprétés et évalués afin d'identifier des connaissances actionnables.

Ce processus se distingue par son caractère itératif et interactif, permettant à l'utilisateur de réajuster les paramètres ou de tester différentes approches pour améliorer la qualité des connaissances découvertes. Que ce soit dans les domaines de la finance, du marketing, de la santé ou de l'industrie, l'ECD offre une capacité précieuse à exploiter les données pour éclairer les décisions, optimiser les stratégies et anticiper les tendances [3].

Proposé par Oussama Fayyad pour répondre aux besoins des entreprises, le processus d'ECD est itératif et interactif, exigeant l'intervention de l'utilisateur à chaque étape. Il peut nécessiter plusieurs itérations pour obtenir des résultats optimaux. (figure I .1).

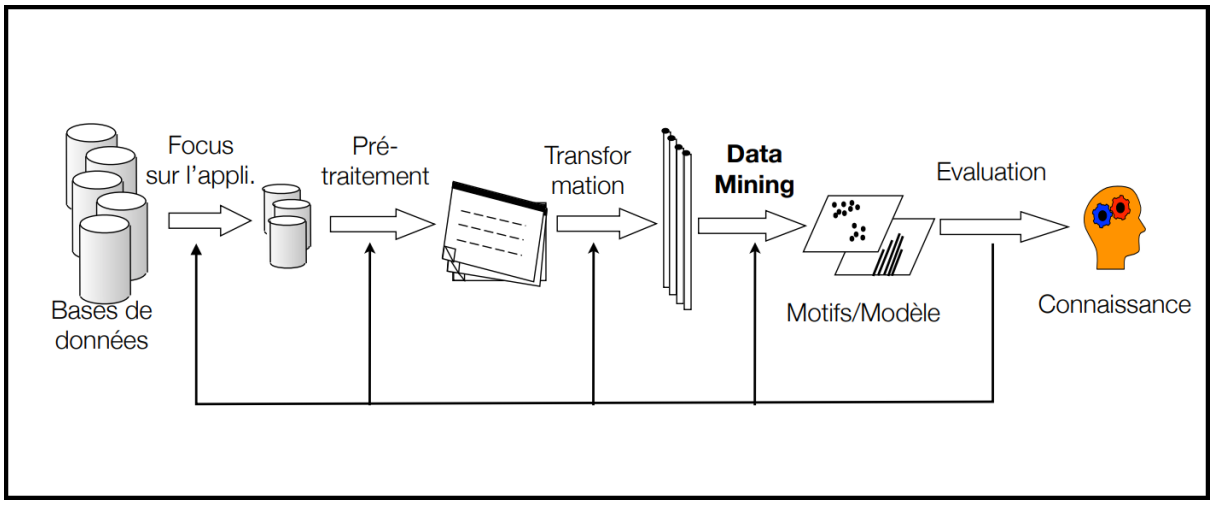


Figure I.1 : Le processus ECD

2.2.1 La consolidation (Définition et compréhension du problème) : Cette étape consiste à créer un jeu de données cible ou à se concentrer sur un sous-ensemble de variables ou d'échantillons de données pour effectuer la découverte.

Elle vise à regrouper les données relatives au système étudié dans une source unique : l'entrepôt de données. Ces données sont généralement hétérogènes. Leur lien est le plus souvent temporel, bien que certaines

applications se basent plutôt sur le nombre de cycles/tâches accomplis par le système cible.

2.2.2 Collecte et sélection des données : La sélection des données constitue la première étape du processus d'Extraction de Connaissances à partir de Données (ECD ou KDD). Elle consiste à identifier et extraire les données pertinentes qui seront utilisées pour l'analyse. Cette phase peut impliquer soit la sélection d'un jeu de données complet, soit la focalisation sur certaines variables, échantillons ou sous-ensembles, en fonction des objectifs de l'étude.

L'objectif principal est de ne conserver que les informations les plus utiles, afin d'améliorer à la fois l'efficacité du processus et la précision des résultats obtenus. Cette sélection repose sur une compréhension approfondie du domaine d'application, garantissant ainsi que les données choisies sont en adéquation avec les besoins de l'analyse.

Une sélection rigoureuse des données permet au processus KDD de produire des résultats fiables, pertinents et exploitables pour la prise de décision.

2.2.3 Prétraitement et nettoyage des données : Dans le processus KDD, le nettoyage des données est essentiel pour garantir que l'ensemble de données est précis et fiable, en corrigeant les erreurs, en gérant les valeurs manquantes, en supprimant les doublons et en traitant les données bruitées ou les valeurs aberrantes.

Traitement des valeurs manquantes : Les lacunes dans les données peuvent être comblées à l'aide de la moyenne, de la médiane ou d'une estimation statistique pour préserver la complétude du jeu de données.

Réduction du bruit : Le bruit est atténué par des techniques telles que le lissage, la régression ou le clustering, permettant de regrouper ou de corriger les données aberrantes.

Suppression des doublons : Les enregistrements dupliqués sont éliminés pour assurer la cohérence des données et éviter les biais dans l'analyse. Le nettoyage des données est une étape cruciale du KDD, car il améliore la qualité des données et optimise l'efficacité de l'exploration.

2.2.4 Fouille de donnée : Dans le processus KDD, le choix de la tâche de fouille de données constitue une étape cruciale. En fonction de l'objectif visé, cette tâche peut consister en une classification, une régression, un clustering (regroupement), ou une extraction de règles d'association.

Une fois la tâche définie, il est indispensable de sélectionner les algorithmes de fouille de données les plus appropriés. Ce choix repose sur la capacité des algorithmes à détecter efficacement et avec précision les motifs ou structures répondant aux objectifs de l'analyse.

Ainsi, l'adéquation entre la tâche, les méthodes utilisées et les résultats attendus est essentielle pour garantir la qualité et la pertinence des connaissances extraites.

2.2.5 L'interprétation et l'évaluation : L'évaluation dans le processus KDD consiste à analyser les motifs identifiés lors de la fouille de données afin d'en déterminer la pertinence et l'utilité.

Cette étape inclut le calcul d'un score d'intérêt pour chaque motif, permettant de distinguer les informations les plus significatives. Des techniques de visualisation et de synthèse sont ensuite utilisées pour faciliter la compréhension et l'accessibilité des résultats par l'utilisateur.

L'interprétation des résultats vise à présenter les connaissances extraites de manière claire et exploitable. Une communication efficace des résultats permet aux décideurs d'orienter leurs actions et d'élaborer des stratégies fondées sur les données, renforçant ainsi la prise de décision basée sur l'analyse.

3. Fouille de données (Data Mining) :

La fouille de données est un domaine qui est apparu avec l'explosion des quantités d'informations stockées, avec les progrès importants des vitesses de traitement et des supports de stockage. La fouille de données vise à découvrir, dans de grandes quantités de données, les informations précieuses qui peuvent aider à comprendre les données ou à prédire le comportement des données futures. Le data mining utilise depuis son apparition plusieurs outils statistiques et d'intelligence artificielle pour atteindre ses objectifs.

La fouille de données est souvent définie comme étant le processus de découverte de nouvelles connaissances en examinant de grandes quantités de données (stockées dans des entrepôts) en utilisant des technologies de reconnaissance de formes, ainsi que des techniques statistiques et mathématiques. Ces connaissances, que l'on ignore au début, peuvent être des corrélations, des modèles (ou patterns) ou des tendances générales au sein de ces données.

La science et l'ingénierie modernes sont basées sur l'idée d'analyser les problèmes pour comprendre leurs principes et développer les modèles mathématiques adéquats. Les données expérimentales sont utilisées par la suite

pour vérifier l'exactitude du système ou estimer quelques paramètres difficiles à modéliser mathématiquement. Cependant, dans la majorité des cas, les systèmes n'ont pas de principes compris ou qui sont trop complexes pour une modélisation mathématique.

Avec le développement des ordinateurs, on a pu rassembler une très grande quantité de données à propos de ces systèmes. La fouille de données est donc devenue une nécessité pour exploiter le plein potentiel des données disponibles et en extraire des connaissances exploitables. [1]

3.1 Définition :

La fouille de données, également appelée exploration de données ou *data mining* (en anglais), est une technique qui permet d'analyser de grandes quantités de données pour y découvrir des modèles, des relations ou des tendances. Cette méthode utilise des algorithmes et des techniques statistiques afin d'identifier des informations cachées et d'extraire des connaissances à partir de données brutes. Les résultats obtenus grâce à la fouille de données peuvent être utilisés pour la prise de décision, la prévision ou l'optimisation de processus[4].

Frawley et Piatetsky-Shapiro définissent la fouille de données comme « l'extraction d'informations originales, auparavant inconnues et potentiellement utiles à partir de données ».

3.2 L'évolution de la technologie système de BD :

L'expression « data mining » est apparue vers le début des années 1960 et avait, à cette époque, une connotation péjorative. En effet, les ordinateurs étaient de plus en plus utilisés pour effectuer toutes sortes de calculs qu'il n'était pas envisageable de réaliser manuellement auparavant. Certains chercheurs ont commencé à analyser sans a priori statistique des tableaux de données issus d'enquêtes ou d'expériences. Constatant que les résultats obtenus, loin d'être aberrants, étaient encourageants, ils furent incités à systématiser cette approche opportuniste. Les statisticiens officiels considéraient toutefois cette démarche comme peu scientifique et employèrent alors les termes « data mining » ou « data fishing » pour la critiquer.

Malgré ces réserves, le succès de cette méthode empirique s'est confirmé. L'analyse des données s'est développée, et son intérêt a grandi parallèlement à l'expansion des bases de données. Vers la fin des années 1980, des chercheurs en bases de données, tels que Rakesh Agrawal, ont commencé à travailler sur l'exploitation de données massives, comme celles générées par les tickets de

caisse des grandes surfaces, convaincus de la valeur potentielle de ces données dormantes. Ils utilisèrent d'abord l'expression « database mining », mais celle-ci étant déjà une marque déposée (*Database Mining Workstation*), c'est finalement «data mining » qui s'imposa.

En mars 1989, Gregory Piatetsky-Shapiro proposa le terme « knowledge discovery » (découverte de connaissances) lors d'un atelier dédié. Aujourd'hui, les expressions « data mining » et « Knowledge Discovery in Databases » (KDD, ou ECD en français) sont utilisées de manière interchangeable. Nous emploierons « data mining », plus courant dans la littérature.

La communauté du data mining a organisé sa première conférence en 1995, après une série d'ateliers sur le KDD entre 1989 et 1994. En 1998, sous l'égide de l'ACM (Association for Computing Machinery), un groupe spécialisé, le SIGKDD, a été créé pour rassembler les chercheurs internationaux du domaine. La première revue académique dédiée, « Data Mining and Knowledge Discovery» (éditée par Kluwer Academic Publishers), a été lancée en 1997.[5]

3.3 Les tâches de la fouille de donnée :

Contrairement à ce que l'on pourrait penser, le data mining n'est pas une solution miracle capable de résoudre tous les défis ou besoins d'une entreprise. Cependant, de nombreux problèmes d'ordre intellectuel, économique ou commercial peuvent être formulés sous la forme des tâches suivantes :

- La classification .
- L'estimation ;
- Le groupement par similitude (règles d'association) .
- L'analyse des clusters .
- La description .
- Le pattern minng.

Les trois premières tâches relèvent de l'apprentissage supervisé (supervised learning), où l'objectif est d'utiliser les données disponibles pour construire un modèle prédictif d'une variable cible.

En revanche, le groupement par similitude et l'analyse des clusters sont des tâches non supervisées (unsupervised learning), visant à découvrir des relations ou des structures intrinsèques dans les données.

La description, quant à elle, peut appartenir aux deux catégories : elle est parfois supervisée (si elle s'appuie sur des variables connues) et parfois non supervisée (si elle explore des motifs émergents) [6].

3.3.1 Classification :

Elle permet de prédire si une instance de données appartient à un groupe ou à une classe prédéfinie en fonction de ses caractéristiques. Plusieurs techniques peuvent être utilisées pour cela, notamment : les arbres de décision, les réseaux de neurones, etc. La régression, quant à elle, permet de prédire des valeurs numériques continues.

3.3.2 Estimation :

L'estimation présente des similitudes avec la classification, à ceci près que la variable de sortie est numérique plutôt que catégorique. Cette technique permet notamment de prédire des valeurs continues, par exemple pour compléter des champs manquants en s'appuyant sur les informations fournies par les autres attributs de l'enregistrement.

3.3.3 Prédiction :

La prédiction présente des similitudes avec la classification et l'estimation, mais s'inscrit dans une échelle temporelle différente. Comme pour ces autres méthodes, elle s'appuie sur des données passées et présentes, mais son résultat concerne un futur généralement spécifié. La seule façon d'évaluer la qualité d'une prédiction consiste à attendre que le moment prévu survienne.

3.3.4 Regroupement par similarité (ou clustering) :

Le regroupement par similarité a pour objectif d'assembler les éléments partageant des caractéristiques communes ou présentant une proximité dans leurs attributs. Parmi les méthodes les plus adaptées à cette approche figure l'extraction de règles d'association, qui permet d'identifier des relations fréquentes entre les éléments au sein des ensembles de données [6][7].

3.3.5 Analyse des clusters :

Le clustering consiste à regrouper des données, observations ou cas en classes d'objets similaires. Un cluster est conçu pour maximiser la similarité intra-cluster (entre objets d'un même groupe) et minimiser la similarité inter-clusters (entre objets de groupes différents). À la différence des méthodes supervisées, le clustering : ne s'appuie pas sur une variable cible prédéfinie, ne vise ni à classer,

ni à estimer, ni à prédire des valeurs. A pour objectif principal de segmenter les données en sous-groupes homogènes basés sur des mesures de distance.

Cette approche permet de diviser une population hétérogène en sous-populations homogènes, ces dernières n'étant pas prédéfinies (contrairement à la classification). Parmi les principales techniques de clustering, on trouve :

- Les réseaux de neurones (notamment les cartes auto-organisatrices).
- Les machines à vecteurs de support (SVM).
- Les méthodes de partitionnement (k-means).
- Les approches hiérarchiques [7]

3.3.6 Description :

Dans le processus d'analyse des données, les chercheurs et analystes cherchent fréquemment à identifier et caractériser des tendances sous-jacentes au sein des jeux de données. Cette description des motifs et tendances observés permet d'expliquer des phénomènes ou de valider des hypothèses.

La description constitue souvent l'une des premières phases du processus de data mining, où l'outil est sollicité pour synthétiser et interpréter les données issues de bases complexes. Cette étape descriptive conduit fréquemment à des analyses complémentaires visant à fournir des explications plus approfondies.

Parmi les techniques adaptées à cette tâche descriptive, l'extraction de règles d'association s'avère particulièrement pertinente [6].

3.3.7 Pattern mining :

L'extraction de motifs ou (pattern mining) constitue une tâche fondamentale du data mining visant à identifier des configurations récurrentes, des règles d'association ou des séquences significatives dans un jeu de données. Cette approche permet de mettre en évidence des schémas répétitifs et des tendances sous-jacentes qui, bien que non immédiatement perceptibles, peuvent révéler des insights stratégiques pour le processus décisionnel [6][7].

3.4 Techniques du Fouille de donnée :

Le data mining mobilise un ensemble de techniques pluridisciplinaires issues : des statistiques, de l'analyse de données, de l'informatique, pour l'exploration approfondie des jeux de données.

Les méthodes de fouille de données se répartissent en deux grandes catégories (Figure I.2) :

- Les techniques non supervisées (ou descriptives).
- Les techniques supervisées (ou prédictives).

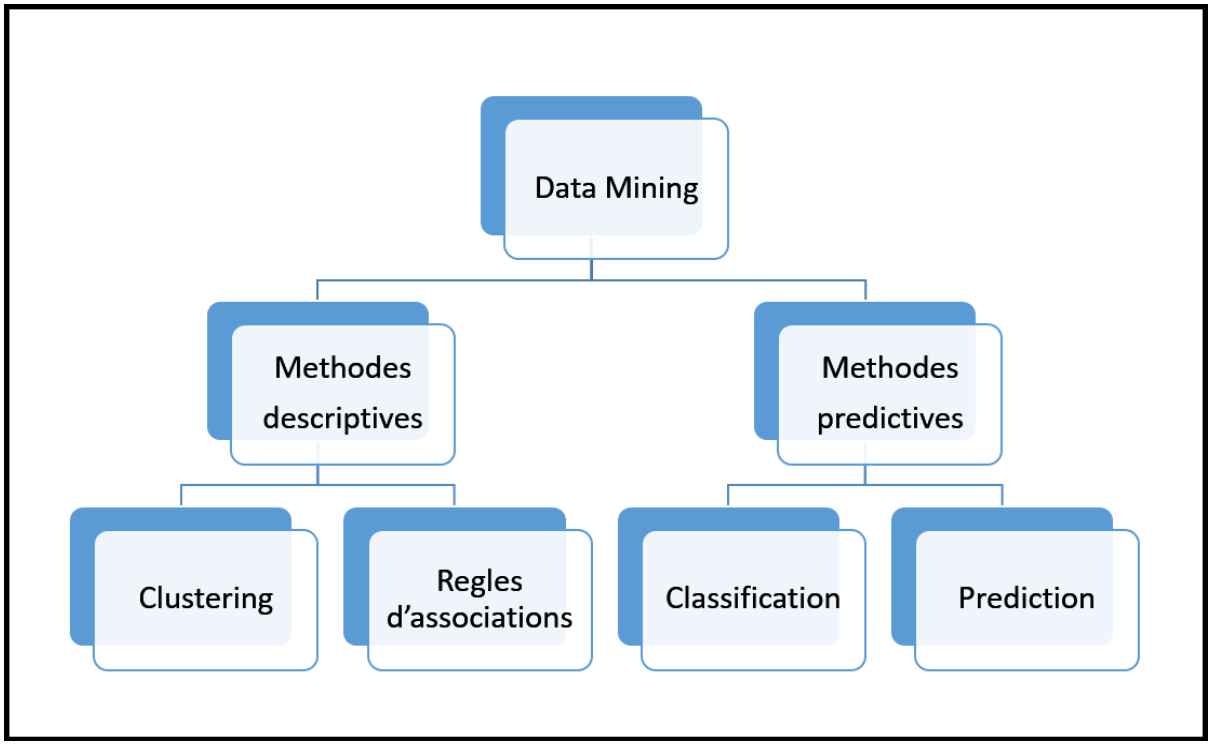


Figure I.2 : Classement des techniques du data mining

3.4.1 Les techniques non-supervisées ou descriptives :

Ces méthodes permettent de :

- Décrire les données
- Résumer, synthétiser ou réduire la dimensionalité des données
- Classifier les éléments selon leurs caractéristiques
- Révéler des informations masquées par le volume des données

Contrairement aux approches supervisées, elles ne s'appuient pas sur une variable cible à prédire. On les désigne sous le terme de techniques non supervisées. Principales techniques non supervisées :

a) Clustering (partitionnement de données) : Technique d'apprentissage automatique non supervisé visant à :

- Regrouper automatiquement des données similaires en clusters
- Découvrir des structures sous-jacentes.
- Identifier des regroupements naturels sans a priori.

Application typique : segmentation de clientèle sans critères préétablis.

b) Règles d'association : Méthode analytique permettant de :

- Détecter des relations significatives entre variables
- Identifier des motifs de cooccurrence fréquents
- Analyser des comportements transactionnels (ex : paniers d'achat)

Application typique : analyse des associations produits en grande distribution.

3.4.2 Les techniques supervisées ou prédictives:

Les techniques supervisées, également appelées techniques prédictives, correspondent à des méthodes d'apprentissage automatique permettant d'inférer une information cible à partir de données annotées. Ces approches s'appuient sur un jeu de données d'entraînement contenant des exemples dont les résultats sont connus, afin de construire un modèle capable de généraliser à de nouvelles observations. Nous présentons ici deux méthodes fondamentales :

a) Prédiction numérique : La prédiction constitue une technique supervisée centrale en analyse de données, caractérisée par :

- L'estimation de valeurs continues à partir de données historiques.
- L'utilisation exclusive de données étiquetées pour l'apprentissage.
- Une application typique aux problèmes de régression.

Contrairement à la classification discrète, cette méthode prédit des variables quantitatives, ce qui la rend particulièrement adaptée pour :

- La prévision de tendances temporelles.
- L'anticipation de comportements.
- L'estimation de valeurs futures.

b) Classification : La classification représente une méthode supervisée fondamentale permettant :

- L'affectation de nouvelles observations à des classes prédéfinies.
- L'apprentissage à partir d'exemples étiquetés.
- La prise de décision automatique selon des catégories discrètes.

Ses applications principales incluent :

- La détection d'anomalies et fraudes.
- Le diagnostic assisté en médecine.

- Le filtrage et catégorisation automatique de contenus.

3.5 Domaines d'application de la Fouille de Donnée :

Le tableau I.1 présente les principaux domaines d'application du data mining, accompagnés d'exemples concrets illustrant chaque secteur d'utilisation.

Domaines d'application	Exemples d'utilisation
Marketing et publicité	Recommandation de produits : utilisation de données d'achat pour recommander des produits complémentaires ou similaires à des clients existants ou potentiels.
Santé et médecine	Utilisation du data mining pour prédire le risque de diabète chez un patient à partir de données telles que l'âge, l'IMC, la pression artérielle, les antécédents familiaux, etc.
Finance et banque	Analyse des investissements : utilisation de données financières pour identifier les tendances du marché et prendre des décisions d'investissement éclairées.
Surveillance et sécurité	Détection de menaces : utilisation de données de sécurité pour détecter les menaces potentielles, comme les attaques de hackers ou les comportements suspects.
Education	Grâce aux données sur le comportement d'apprentissage, les plateformes éducatives peuvent recommander des contenus adaptés au rythme et aux préférences de chaque étudiant.

Table I.1 : Domaines d'application de la Fouille de Donnée

4. Conclusion :

Les techniques de data mining représentent des outils indispensables pour l'exploration et l'analyse des masses de données générées par les organisations modernes. Ces méthodes permettent aux analystes : d'identifier des schémas caches, détecter des tendances significatives et de révéler des corrélations non apparentes. Ces découvertes offrent aux décideurs des insights actionnables, conduisant à des prises de décision plus rationnelles et efficaces.

Dans ce premier chapitre, nous avons introduit le domaine de l'extraction de connaissances à partir des données, communément connu sous le nom anglais

de Data mining, ou son équivalent français de fouille de données. Nous avons essayé dans un premier temps présenter le processus complet d'ECD (Extraction des Connaissances à partir des Données), où nous avons révélé qu'il s'agit de tout un processus et nous avons exposé les différentes étapes qu'il comporte. Enfin, nous avons survolé les différentes tâches que peut réaliser le processus d'extraction de connaissances et nous avons présenté chacune d'elles.

Dans ce travail, nous avons exploré les différentes techniques de data mining et présenté les algorithmes les plus couramment utilisés pour accomplir des tâches fondamentales telles que le regroupement (clustering), la classification et la découverte de règles d'association.

Dans le prochain chapitre, nous procéderons à une analyse approfondie les algorithmes utilisés dans le High Utility Itemset Mining(HUIM). Nous nous pencherons en particulier sur les approches visant à réduire l'espace de recherche et les différentes méthodes et algorithmes de structuration des données appliquées dans le HUIM. Cela comprendra l'étude de techniques telles que la génération de motifs candidats, la sélection basée sur des seuils d'utilité, et l'utilisation de structures de données optimisées conçues pour accélérer le processus d'extraction des itemsets à forte utilité.

Chapitre II:
Extraction des ensembles d’items à haute utilité (HUIM)

1. Introduction :

Face à l'explosion récente des volumes de données dans tous les secteurs d'activité, les méthodes de traitement de l'information ont subi des transformations majeures. Cette révolution numérique a rendu crucial le développement de techniques performantes pour l'extraction de connaissances pertinentes.

Dans ce contexte, la recherche de motifs émergents s'est imposée comme :un domaine de recherche fondamental et un champ ayant généré de nombreuses avancées scientifiques.

L'extraction de motifs significatifs, classée parmi les méthodes descriptives, représente aujourd'hui l'une des approches les plus utilisées en fouille de données. Ce chapitre propose une étude approfondie des principales méthodes et algorithmes permettant d'identifier divers types de motifs au sein des jeux de données.

2.Extraction d'ensembles d'items fréquents :

L'extraction des motifs fréquents (Frequent Itemset Mining, FIM) est la tâche qui consiste à extraire tous les itemsets fréquents (ayant une fréquence d'apparition supérieure ou égale à un seuil donné) dans un ensemble de données.

Cette tâche a été proposée au début des années 1990 pour identifier des articles fréquemment achetés ensemble dans le cadre de l'analyse des paniers d'achat (Agrawal et al., 1993). Initialement, elle était appelée extraction des grands itemsets (large itemset mining).

Cette section présente certaines définitions formelles liées à cette tâche, ainsi que différents domaines d'application et plusieurs tâches connexes [8].

2.1 Concepts Fondamentaux :

Avant d'approfondir le FIM, il convient de clarifier certains termes importants :

2.1.1. Base de données : Une base de données est définie par un triplet

$D = (O, P, R)_1$, où : O est un ensemble fini d'objets, P un ensemble fini d'items et R une relation binaire entre O et P. Cet ensemble constitue notre cadre de travail[9].

2.1.2. Transaction : On définit $T = \{T_1, T_2, T_3, ..., T_n\}_2$, où chaque $T_i \subseteq A$, comme une transaction contenant des items issus de la base de données A. Chaque transaction T_i regroupe les éléments co-occurrents dans une opération donnée.

Exemple : Dans un système de caisse, chaque transaction représente les produits achetés ensemble par un client [9].

2.1.3. Base de données transactionnelle : Une base de données transactionnelle est un type particulier de base de données dans lequel les données sont structurées sous forme de transactions. Chaque transaction correspond à une collection d'items, généralement représentée comme un sous-ensemble d'un ensemble global d'items [9].

TID	Transaction
T1	{ a ,c ,d ,e }
T2	{ b ,c ,d }
T3	{ c ,e }
T4	{ a ,b ,c ,e }
T5	{ b ,c }

Table II.2 : Base de données transactionnelles

2.1.4. Item : Un item désigne un élément individuel ou un attribut d'une base de données transactionnelle. Par exemple, dans une base de données de ventes au détail, les items peuvent représenter des produits spécifiques [9].

2.1.5. Itemset

Un itemset est un sous-ensemble non vide d'éléments appartenant à l'ensemble global des items $I = \{i_1, i_2, ..., i_n\}_3$ d'une base de données transactionnelle. Chaque itemset représente une combinaison possible d'items pouvant apparaître simultanément dans une transaction (Table II.2).

Par exemple, les ensembles $\{X_1\}$ et $\{X_1, X_2\}$ sont des itemsets. Un itemset composé de k items est appelé *k-itemset,* où k = |X| représente la cardinalité de l'itemset [9].

2.1.6. Itemset fréquent : Soit un seuil *minsup* > 0, défini par l'utilisateur. Un itemset X est dit fréquent si son support, noté sup(X), vérifie l'inégalité sup(X) ≥ minsup. Dans le cas contraire, X est qualifié d'itemset non fréquent [10][11].

2.1.7. Le support : Le support d'un itemset est une mesure qui indique à quelle fréquence cet itemset apparaît dans une base de données transactionnelle. Formellement, pour un itemset X, le support est défini comme :

$$\mathbf{S(x)} = \frac{|\{\mathbf{t} \in \mathbf{D}: \mathbf{X} \subseteq \mathbf{t}\}|}{|\mathbf{D}|}_4$$

Autrement dit, le support représente le pourcentage de transactions dans lesquelles les items de X apparaissent ensemble [11].

La compréhension de ces définitions constitue une étape clé pour appréhender les principes fondamentaux de l'extraction des itemsets fréquents. Ces concepts établissent les bases nécessaires à l'identification des itemsets significatifs dans les transactions, en utilisant des mesures telles que le support pour évaluer leur fréquence. Une maîtrise approfondie de ces notions permet d'analyser les algorithmes et techniques spécifiques qui extraient efficacement les itemsets fréquents à partir de bases de données transactionnelles.

2.2. Défis de l'extraction des ensembles d'items fréquents :

L'extraction des itemsets fréquents (Frequent Itemset Mining, FIM) est un problème d'énumération. L'objectif est de lister l'ensemble des itemsets qui satisfont la contrainte de support minimal définie par l'utilisateur. Ainsi, une tâche de FIM a toujours une seule réponse correcte, c'est-à-dire l'ensemble complet de motifs fréquents respectant ce seuil. Cependant, il s'agit d'un problème complexe. L'approche naïve consisterait à générer tous les itemsets possibles, puis à filtrer ceux qui respectent la contrainte de support. Or, cette méthode est inefficace pour des raisons de complexité combinatoire. En effet, si une base transactionnelle contient m items distincts, le nombre total d'itemsets possibles est de $2m-1$.

Par exemple, avec 1 000 items distincts, on obtient $\mathbf{2^{1000}}$ *itemsets* possibles, un volume totalement ingérable avec une approche exhaustive. Il est important de souligner que le problème de FIM peut devenir difficile même sur de petites bases de données. Par exemple, une base contenant une seule transaction de 100 items génère déjà un espace de recherche de $\mathbf{2^{100}}$ *itemsets*, même si le support minimal (*minsup*) est fixé à 1. Cela montre que, dans le contexte de la FIM, le nombre d'itemsets dans l'espace de recherche est souvent plus critique que la taille des données elles-mêmes.

Plusieurs facteurs influencent la taille de cet espace de recherche tel que Le seuil de support minimal (minsup) et la similarité entre les transactions. Ces éléments rendent nécessaire l'utilisation d'algorithmes efficaces (comme Apriori, FP-Growth ou Eclat) pour réduire le nombre de combinaisons à explorer et optimiser les performances de l'extraction [12].

2.3. Algorithmes d'extraction des items fréquents

Pour identifier efficacement les ensembles d'éléments fréquents, il est essentiel de développer des algorithmes capables d'éviter une exploration exhaustive de l'espace de recherche (c'est-à-dire tous les sous-ensembles possibles), tout en traitant chaque ensemble de manière optimale. Plusieurs algorithmes performants ont été proposés (Tableau II.3) dans le cadre de la fouille d'ensembles fréquents (Frequent Itemset Mining, FIM). Parmi les plus connus figurent Apriori, FP-Growth, Eclat et LCM.

Bien que ces algorithmes partagent des entrées et des sorties similaires, ils se distinguent par leurs stratégies de recherche et les structures de données utilisées pour découvrir efficacement les ensembles fréquents [12]. Les méthodes d'extraction des items fréquents (FIM) diffèrent principalement sur les aspects suivants .

2.3.1. Stratégie de recherche : Selon la manière dont les algorithmes explorent l'espace des *itemsets*, certains utilisent une approche en profondeur (*depth-first search*, DFS), tandis que d'autres adoptent une approche en largeur (*breadth-first search*, BFS).

2.3.2. Représentation de la base de données : La base de données transactionnelle peut être représentée selon deux formats principaux :

- Représentation horizontale : chaque ligne correspond à une transaction et contient la liste des items qui la composent ;
- Représentation verticale : chaque item est associé à la liste des transactions dans lesquelles il apparaît.

Algorithme	Type de recherche	Représentation de la BD
Apriori	Recherche en largeur	Horizontal
FP_Growth	Recherche en profondeur	Horizontal
Eclat	Recherche en profondeur	Vertical
LCM	Recherche en profondeur	Horizontal

Table II.3 : Résumé des algorithmes FIM

2.3.3. Génération des ensembles d'items : Les algorithmes se distinguent par leur approche de génération et de sélection des ensembles d'items candidats à explorer dans l'espace de recherche.

2.3.4. Évaluation du support : Les méthodes de calcul du support des itemsets diffèrent selon les algorithmes, tout en garantissant systématiquement le respect du seuil de support minimum prédéfini.

2.4. Recherche en largeur (Breadth-First Search - BFS) :

Un algorithme basé sur la recherche en largeur (également appelé algorithme par niveaux), comme l'algorithme Apriori, part de l'hypothèse que la base de données contient m éléments distincts. Son principe consiste à explorer systématiquement l'espace des itemsets en suivant une approche ascendante : il examine d'abord tous les itemsets de taille 1 (éléments uniques), puis ceux de taille 2 (paires d'éléments), ensuite ceux de taille 3, et ainsi de suite jusqu'aux itemsets complets de taille m.

La Figure 3 illustre ce processus d'exploration pour un cas concret, en représentant l'ensemble des combinaisons possibles sous forme d'un diagramme de Hasse. Cette représentation met clairement en évidence la structure hiérarchique de l'espace de recherche.

Prenons l'exemple d'un ensemble d'items I = {a, b, c, d, e}. L'algorithme commence par examiner tous les itemsets de taille 1 ({a}, {b}, {c}, etc.), puis génère progressivement les combinaisons de taille supérieure : les paires ({a,b}, {a,c},...), les triplets ({a,b,c}, etc.), et ce jusqu'à l'itemset complet {a,b,c,d,e} contenant l'ensemble des éléments. Bien que cette méthode offre une exploration méthodique et exhaustive de l'espace de recherche, elle présente néanmoins un inconvénient majeur : sa complexité computationnelle (en temps et en mémoire) croît exponentiellement avec le nombre d'items, ce qui peut la rendre inefficace pour les grandes bases de données [12].

2.4.1. Apriori : un algorithme de recherche en largeur à parcours horizontal : Apriori est un algorithme pionnier développé par R. Agrawal et R. Srikant en 1994, spécialement conçu pour l'extraction d'itemsets fréquents dans le cadre de la génération de règles d'association. La Figure II.3 illustre le processus général de l'algorithme Apriori. Son fonctionnement repose sur cinq étapes claires:

- Détection des items fréquents élémentaires : L'algorithme commence par calculer le support de chaque item individuel. Seuls ceux atteignant le seuil minimum prédéfini sont retenus.

- Génération des candidats : À chaque itération k, les itemsets fréquents de taille k-1 sont combinés pour générer des candidats de taille k grâce à une opération de jointure spécifique.
- Élagage optimisé : La propriété d'Apriori (principe d'anti-monotonie) permet d'éliminer systématiquement tout candidat contenant un sous-ensemble non fréquent, réduisant ainsi drastiquement l'espace de recherche.
- Évaluation des supports : Les candidats restants sont validés par un scan complet de la base de données pour calculer leurs supports réels.
- Itération jusqu'à convergence : Le processus se répète jusqu'à ce qu'aucun nouvel itemset fréquent ne puisse être généré.

Le principal avantage d'Apriori réside dans sa capacité à éliminer précocement les combinaisons non pertinentes. Cependant, ses performances peuvent se dégrader significativement face à des bases de données volumineuses ou contenant un grand nombre d'items, en raison des scans répétés de la base [13].

Algorithm 1: The Apriori algorithm

input : D: a horizontal transaction database, *minsup*: a user-specified threshold

output: the set of frequent itemsets

1 Scan the database to calculate the support of all items in I;
2 $F_1 = \{i | i \in I \wedge sup(\{i\}) \geq minsup\}$; // F_1 : frequent 1-itemsets
3 $k = 2$;
4 **while** $F_k \neq \emptyset$ **do**
5 $C_k =$ CandidateGeneration (F_{k-1}) ; // C_k : candidate k-itemsets
6 Remove each candidate $X \in C_k$ that contains a $(k-1)$-itemset that is not in F_{k-1};
7 Scan the database to calculate the support of each candidate $X \in C_k$;
8 $F_k = \{X | X \in C_k \wedge sup(X) \geq minsup\}$; // F_k : frequent k-itemsets
9 $k = k + 1$;
10 **end**
11 return $\bigcup_{k=1...k} F_k$;

Figure II.3 : Algorithme Apriori

2.5. Recherche en profondeur (Depth-First Search - DFS) :

Contrairement à l'approche par niveaux (BFS), la recherche en profondeur explore exhaustivement chaque branche de l'espace des itemsets avant d'examiner les alternatives. Cette stratégie est principalement utilisée par des algorithmes comme FP-Growth, Eclat et LCM.

Le fonctionnement des algorithmes DFS s'articule autour des principes suivants : Ils débutent par les itemsets de taille unitaire, puis Ils étendent progressivement ces ensembles par ajout récursif d'items. Chaque branche est explorée exhaustivement avant de passer à la suivante.

L'illustration présentée dans la Figure II.4 montre clairement ce mécanisme d'exploration hiérarchique. Par exemple, le parcours s'effectuerait selon cette séquence :

{a}, {a, b}, {a, b, c}, {a, b, c, d}, {a, b, c, d, e}, {a, b,c, e}, {a, b, d}, {a, b, d, e}, {a, b, e}, {a, c}, {a, c, d}, {a, c, d, e}, {a, c, e}, {a, d}, {a,d, e},{a, e}, {b}, {b, c}, {b, c, d}, {b, c, d, e}, {b, c, e}, {b, d}, {b, d, e}, {b, e}, {c}, {c,d}, {c, d, e}, {c,e}, {d}, {d, e}, {e} .

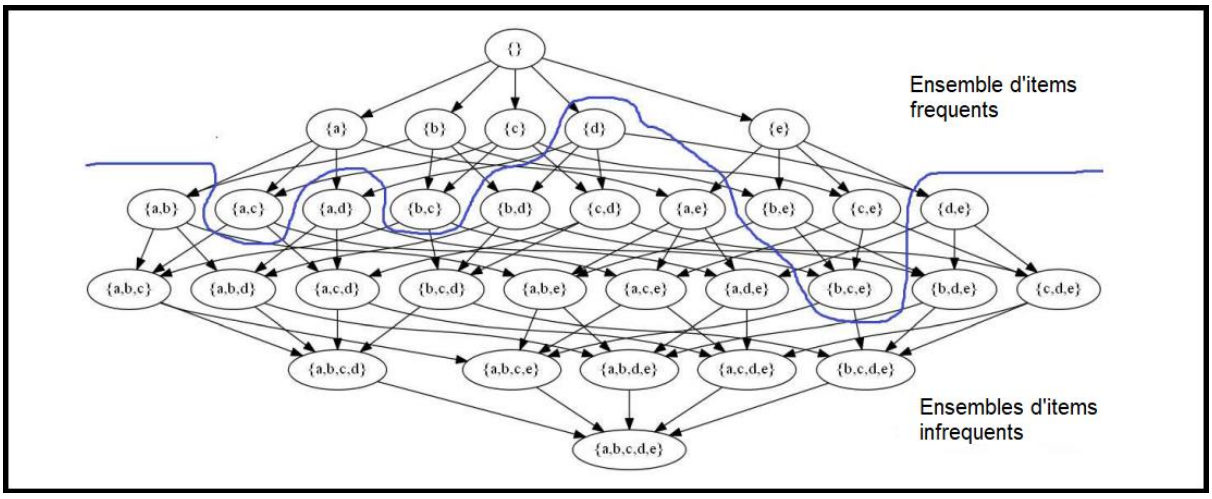


Figure II.4 : Espace de recherche pour I= { a, b, c, d, e }

2.5.1. Algorithme Eclat : L'algorithme Eclat (Figure II.5) est considéré comme un algorithme de recherche en profondeur (*depth-first search*), car il explore et génère tous les *itemsets* fréquents selon un parcours en profondeur. En

général, Eclat est beaucoup plus rapide qu'Apriori, notamment parce qu'il évite les lectures répétées de la base de données. Cependant, Eclat présente certaines limites. D'une part, comme il génère également des candidats sans avoir à accéder directement aux données sources, il peut parfois traiter des combinaisons d'items qui ne sont pas réellement présentes dans la base de données. D'autre part, bien que l'utilisation des *TID-lists* (listes d'identifiants de transactions) soit avantageuse, celles-ci peuvent consommer énormément de mémoire, surtout dans le cas de bases de données denses, où la majorité des items apparaissent dans presque toutes les transactions.

En réponse à cette problématique, plusieurs optimisations ont été proposées. Par exemple, une structure améliorée appelée *diffsets* permet de réduire la taille des *TID-lists*. D'autres méthodes consistent à encoder les *TID-lists* sous forme de vecteurs de bits, ce qui permet non seulement de réduire l'utilisation mémoire, mais aussi d'accélérer les opérations d'intersection, en particulier sur les jeux de données denses.

Il est également intéressant de noter que les *TID-lists* peuvent être utilisées dans des algorithmes à recherche en largeur. C'est le cas par exemple d'AprioriTID, une variante d'Apriori qui utilise les *TID-lists* pour calculer le support des *itemsets* [12].

Algorithm 2: The Eclat algorithm

input : R: a set of itemsets with their tidsets, $minsup$: a user-specified threshold

output: the set of frequent itemsets

```
1  foreach itemset X ∈ R such that |tid(X)| ≥ minsup do
2      Output X;                                   // X is a frequent itemset
3      E = ∅;                  // frequent itemsets that are extensions of X
4      foreach itemset Y ∈ R sharing all but the last item with X do
5          tid(X ∪ Y) = tid(X) ∩ tid(Y);          // calculate the tidset of X ∪ Y
6          if |tid(X ∪ Y)| ≥ minsup then                // if X ∪ Y is frequent
7              E = E ∪ {X ∪ Y};          // add X ∪ Y to frequent extensions of X
8          end
9          Eclat (E, minsup);                          // recursive call using E
10     end
11 end
```

Figure II.5 : Algorithme Eclat

2.6. Limitations de FIM :

Malgré son efficacité, la méthode d'extraction des ensembles d'éléments fréquents (FIM) présente plusieurs limites qui réduisent sa pertinence dans certaines applications. Une des principales critiques est qu'elle considère tous les items comme ayant une importance équivalente. Autrement dit, la valeur ou l'utilité réelle des items n'est pas prise en compte. De plus, FIM ne tient pas compte des quantités d'items présents dans les transactions. Par exemple, l'achat de cinq unités d'un produit est traité de la même manière que l'achat d'une seule unité, ce qui peut fausser l'analyse. Cela conduit à la découverte de nombreux motifs fréquents peu utiles, simplement parce qu'ils apparaissent souvent, et non parce qu'ils génèrent une valeur significative. Par exemple, l'itemset {pain, lait} peut être courant dans les transactions, mais il n'est pas nécessairement intéressant pour un commerçant, car il peut générer peu de profit. À l'inverse, des itemsets plus rares mais associés à une forte marge bénéficiaire peuvent être ignorés car ils ne dépassent pas le seuil de support.

Ces limitations ont conduit au développement de méthodes alternatives, notamment l'extraction d'itemsets à haute utilité, qui intègrent la notion de valeur économique ou stratégique des items.

3. Extraction des ensembles d'items à haute utilité :

L'extraction des ensembles d'items à haute utilité, ou *High Utility Itemset Mining* (HUIM), constitue une évolution de l'extraction des itemsets fréquents. Elle vise à identifier des motifs significatifs dans les bases de données en fonction de leur importance mesurée. Cette importance, appelée utilité, peut être définie selon divers critères objectifs, tels que le profit généré, la fréquence d'apparition ou encore le poids attribué aux items [11].

Contrairement à la fouille classique, HUIM permet de repérer des ensembles d'items qui, même peu fréquents, peuvent présenter une importance stratégique élevée. Par exemple, un produit vendu rarement mais avec une marge très élevée peut s'avérer plus intéressant qu'un produit fréquemment vendu à faible profit. Le principe fondamental de HUIM repose sur l'utilisation d'une fonction d'utilité qui évalue l'intérêt d'un itemset dans une transaction. Cette fonction peut varier selon le domaine d'application :

- Dans le commerce, il peut s'agir du revenu généré par la vente conjointe de plusieurs produits.
- Dans les services personnalisés, l'utilité peut refléter le degré de satisfaction ou d'engagement d'un utilisateur.
- Dans des domaines comme la logistique ou la santé, l'utilité peut être liée à la disponibilité, au coût ou à la priorité d'un item.

Pour appliquer HUIM, les données doivent être modélisées de manière enrichie, en associant à chaque item non seulement sa présence dans une transaction, mais aussi sa valeur d'utilité (par exemple : quantité achetée × gain unitaire). L'extraction des itemsets à haute utilité nécessite des algorithmes spécialisés, capables de gérer efficacement cette complexité supplémentaire. Ces algorithmes reposent généralement sur :

- La génération contrôlée de candidats potentiels.
- Des techniques d'élagage basées sur des seuils d'utilité minimum pour éliminer les combinaisons non prometteuses.
- L'utilisation de structures de données efficaces, comme les Utility Lists, les UP-Trees ou d'autres index compressés.

De nombreux algorithmes performants ont été proposés pour répondre à cette problématique, permettant de réduire le temps de calcul tout en maintenant une précision élevée. En somme, HUIM représente un changement de paradigme dans l'analyse de données transactionnelles, en plaçant l'accent non plus sur ce qui est fréquent, mais sur ce qui est valable ou rentable. Cette approche offre des résultats plus pertinents sur le plan économique ou stratégique, et permet de mieux accompagner la prise de décision dans des contextes réels [11].

Dans les sections suivantes, nous explorerons en détail les techniques clés et les algorithmes associés à HUIM, en analysant leurs principes, leurs avantages et leur capacité à révéler des motifs à forte valeur ajoutée.

3.1. Concepts fondamentaux :

3.1.1 Base de données quantitative des transactions : s

Une base de données de transactions quantitatives D est définie comme un ensemble de transactions $D=\{\mathbf{T_0},\mathbf{T_1},\ldots,\mathbf{T_n}\}$ où chaque transaction $\mathbf{T_q}$ est un

itemset , tel que $\mathbf{T_q} \subseteq I$, et est associée à un identifiant unique appelé TID (Transaction ID).

Chaque élément $i \in I$ (voir Table II.4) est caractérisé par: une utilité externe p(i), représentant, par exemple, le prix ou la valeur unitaire de l'article et une utilité interne q(i,Tc), représentant la quantité ou le nombre d'occurrences de l'article i dans la transaction Tc (voir Table II.5).

Item	Utilite externe
A	4
B	3
C	1
D	2
E	3
F	1
G	1

Table II.4 : Valeurs d'utilité externe

TID	Transaction
T1	(a,2),(b,5),(c,3),(d,3),(e,1),(f,5)
T2	(b,3),(c,3),(d,4),(e,1)
T3	(a,1),(c,1),(d,1)
T4	(a,3),(c,5),(e,2),(g,4)
T5	(b,2),(c,2),(e,1),(g,2)

Table II.5 : Base de données quantitative des transactions

3.1.2 Utilité d'un item :

L'utilité d'un item i dans une transaction Tc, notée U(i,Tc), est calculée comme le produit de son utilité externe p(i) et de sa quantité dans la transaction notée q(i,Tc) . Autrement dit : $U(i,Tc)=p(i)\times q(i,Tc)$[5]

Exemple : L'utilité de l'item b de la T1 est $U(b,T1) = 3\times 5 = 15$

a. Utilite d'un itemset dans une transaction :

L'utilité d'un itemset X dans une transaction Tc est définie comme suit :
$U(X, Tc) = \sum_{i \in X} U(X, Tc)$[6] ,s i $X \subset Tc$.
Exemple : L'utilité de l'itemset {a,b} dans la transaction T1 est :
$U(\{a,b\}, T1) = U(a, T1) + U(b, T1) = 23$

b. Utilité d'un itemset dans une base de données :

L'utilité d'un itemset X dans une base de données D, notée U(X,D), est définie comme la somme des utilités de X dans toutes les transactions $T \in D$ où X est présent. $U(X,D)=\sum_{T \in D} U(X,T)$[7]

Exemple : L'utilité de l'itemset {b,c} dans une base de donnée est :

U({b,c}) = U(b)+U(c) = U(b,T1)+U(b,T2)+U(b,T5)+U(c,T1)+U(c,T2)+U(c,T5) = 15 + 9 + 6 + 3 + 3 + 2 = 38

3.1.3. Itemset à haute utilité :

X est un itemset à haute utilité si son utilité u(X) n'est pas inférieur à un seuil d'utilité minutil prédéfini par l'utilisateur , tel que u(X) ≥ minutil. Sinon, X est un itemset de faible utilité [14].

Par exemple, on a Minutil= 30

U({b,c}) = 38 {b,c} est un itemset à utilité haute.

U({c,d}) = 23 {c,d}est un itemset a faible utilité.

3.1.4. L'utilité de la transaction :

L'utilité d'une transaction Tc, notée U(Tc), est la somme des utilités de tous les items présents dans cette transaction [14], l'utilité d'une transaction est défini comme suit :

$TU(Tc) = \sum_{i \in Tc} U(X, Tc)$[8]

Exemple : L'utilité de transaction T2 :

TU(T2) = U(b,T2) + U(c,T2) + U(d,T2) + U(e,T2)

= 9 + 3 + 8 + 3 = 23

3.2. Les principales propriétés du problème du (HUIM) :

Pour une base de données quantitative donnée et un seuil minimal d'utilité fixé, le problème de l'extraction des ensembles d'éléments à haute utilité admet une solution unique. Cette solution consiste à énumérer tous les motifs dont l'utilité est supérieure ou égale au seuil défini par l'utilisateur. Ce problème est considéré comme complexe, principalement pour deux raisons :

La première tient au nombre très élevé d'ensembles d'éléments à évaluer afin d'identifier ceux ayant une utilité élevée. En effet, si une base de données contient *m* éléments distincts, il existe $2^m - 1$ ensembles d'éléments possibles (en excluant l'ensemble vide).

Par exemple, pour un ensemble d'items I = {a, b, c}, les combinaisons possibles sont : {a}, {b}, {c}, {a, b}, {a, c}, {b, c} et {a, b, c}, soit 7 ensembles au total On peut donc générer $2^3 - 1 = 7$ ensembles d'éléments à partir de I = {a, b, c}.

Une approche naïve pour résoudre le problème de l'extraction des ensembles à haute utilité consiste à calculer l'utilité de chaque ensemble possible en parcourant la base de données, puis à ne conserver que ceux dont l'utilité dépasse le seuil défini. Même si cette méthode donne des résultats corrects, elle est extrêmement inefficace. En effet, le nombre d'ensembles à évaluer devient rapidement énorme. Par exemple, dans un magasin de détail disposant de 10 000 articles différents, il faudrait analyser $2^{10000} - 1$ ensembles, ce qui est totalement irréaliste.

Il est important de noter que ce problème peut être complexe même sur de petites bases de données. Une base ne contenant qu'une seule transaction de 100 articles peut déjà générer $2^{100}-1$ ensembles différents.

Cela montre que la taille de l'espace de recherche, c'est-à-dire le nombre d'ensembles à explorer, peut être énorme même si le nombre de transactions est faible. En réalité, cette taille dépend non seulement du nombre d'éléments dans la base, mais aussi de la similarité des transactions, de l'amplitude des valeurs d'utilité, ainsi que du seuil d'utilité minimal choisi par l'utilisateur.

Une deuxième difficulté majeure dans l'exploration des ensembles d'éléments à haute utilité réside dans le fait que ces ensembles sont souvent éparpillés dans l'espace de recherche. En conséquence, un algorithme doit examiner un grand nombre d'ensembles potentiels avant de repérer ceux qui satisfont le critère d'utilité. Pour mieux comprendre cela, on peut se référer à la figure II.4, qui propose une représentation graphique de l'espace de recherche sous forme de diagramme de Hasse. Dans ce type de diagramme, chaque ensemble d'éléments est représenté par un nœud, et une flèche est tracée d'un ensemble X vers un ensemble Y si, et seulement si, $X \subseteq Y$ et que la taille de Y est exactement celle de X plus un ($|X| + 1 = |Y|$).

Cela illustre le fait que la fonction d'utilité n'est ni monotone ni antimonotone, on ne peut ni garantir qu'un sur-ensemble aura une utilité plus élevée, ni l'inverse.

Propriété 1 : La mesure d'utilité n'est ni monotone ni antimonotone. Soit deux ensembles d'éléments X et Y tels que $X \subset Y$. La relation entre les utilités de X et Y est soit $u(X) < u(Y)$, $u(X) > u(Y)$, soit $u(X) = u(Y)$.

En raison de cette propriété, les ensembles d'éléments à haute utilité sont souvent éparpillés dans l'espace de recherche, comme le montre la figure II.4. Cette dispersion rend leur détection plus difficile. C'est l'une des principales

raisons pour lesquelles l'exploration d'ensembles d'éléments à haute utilité est plus complexe que l'exploration des ensembles d'éléments fréquents.

En effet, dans le cas des motifs fréquents, la mesure de support bénéficie de la propriété de monotonie, tel que le support d'un ensemble d'éléments est toujours supérieur ou égal à celui de chacun de ses super-ensembles. Cela permet d'éliminer efficacement des branches entières de l'espace de recherche. À l'inverse, l'absence de monotonie dans la mesure d'utilité empêche cette optimisation, rendant les méthodes classiques inadaptées et insuffisantes pour ce type de problème.

Propriété 2 : La mesure de support est monotone Soient deux ensembles d'éléments X et Y tels que $X \subset Y$. On en déduit que le support de X ne peut jamais être inférieur à celui de Y, $sup(X) \geq sup(Y)$

Autrement dit, plus un ensemble d'éléments contient d'items, moins il est susceptible d'apparaître fréquemment dans la base de données. Cette propriété est essentielle car elle permet d'élaguer efficacement l'espace de recherche dans les algorithmes d'extraction de motifs fréquents.

Propriété 3 : La TWU est une borne supérieure monotone de la mesure d'utilité Soit un itemset X. La *Transaction-Weighted Utilization* (TWU) de X constitue une borne supérieure monotone de son utilité. Autrement dit :

- $TWU(X) \geq u(X)$
- Et pour tout sur-ensemble $Y \supset X$, on a également :$TWU(X) \geq u(Y)$

Cela signifie que la TWU d'un itemset fournit non seulement une estimation supérieure de sa propre utilité, mais aussi une estimation supérieure de celle de tous ses sur-ensembles (super-set). Cette propriété est cruciale pour les techniques d'élagage dans l'exploration d'ensembles d'éléments à haute utilité, car elle permet d'éliminer de manière anticipée les itemsets dont le TWU est inférieur au seuil minimal d'utilité spécifié.

Propriété 4 : Élagage de l'espace de recherche à l'aide du TWU. Soit un itemset X. Si la *Transaction-Weighted Utilization* (TWU) de X est inférieure au seuil minimal d'utilité spécifié (TWU(X)<minutil, alors :

- X ne peut en aucun cas être un itemset à haute utilité ;
- De plus, aucun sur-ensemble de X ne peut satisfaire la contrainte d'utilité minimale.

Par conséquent, X et tous ses sur-ensembles peuvent être élagués du processus de recherche.

Les algorithmes d'extraction en deux phases exploitent cette propriété dérivée de la Propriété 2 comme stratégie principale pour réduire efficacement l'espace de recherche.

En raison de la taille potentiellement immense de l'espace de recherche dans l'exploration d'ensembles d'éléments à haute utilité, il est essentiel de concevoir des algorithmes performants capables d'éviter l'examen exhaustif de tous les ensembles d'éléments possibles. Ces algorithmes doivent non seulement minimiser les calculs superflus, mais aussi traiter chaque ensemble de manière efficace, tout en garantissant l'extraction complète de tous les motifs à haute utilité.

Par ailleurs, le fait que la mesure d'utilité ne soit ni monotone ni anti-monotone rend inapplicables les techniques classiques de réduction de l'espace de recherche, telles que celles utilisées dans l'exploration d'ensembles fréquents. En conséquence, de nouvelles approches spécifiques sont nécessaires pour relever les défis propres à l'extraction d'ensembles à haute utilité.

3.3. Algorithmes itemset à haute utilité :

De nombreux algorithmes ont été développés pour l'exploration d'ensembles d'éléments à haute utilité, notamment UMining, Two-Phase, IHUP, UP-Growth, HUP-Growth, MU-Growth, HUI-Miner, FHM, ULB-Miner, HUI-Miner* et EFIM. Bien que tous ces algorithmes aient la même entrée (une base de données transactionnelle et un seuil d'utilité minimale) et la même sortie (l'ensemble des itemsets à haute utilité), ils se distinguent par les structures de données utilisées et les stratégies d'exploration adoptées. Plus précisément, les principales différences résident dans :

1. La stratégie de parcours de l'espace de recherche (recherche en profondeur ou en largeur) .

2. Le mode de représentation des données, qu'il soit interne (en mémoire) ou externe (sur disque) .

3. La méthode de génération ou de sélection des itemsets à explorer .

4. Le calcul de l'utilité pour évaluer si un itemset satisfait la contrainte d'utilité minimale.

Ces choix de conception ont un impact direct sur les performances des algorithmes, notamment en termes de temps d'exécution, consommation mémoire et capacité à passer à l'échelle. Ils influencent également la complexité de mise en œuvre et la possibilité d'adaptation à d'autres tâches d'exploration de données.

En général, ces algorithmes s'inspirent des approches classiques d'exploration d'itemsets fréquents, tout en introduisant des mécanismes innovants pour surmonter les défis posés par le fait que la mesure d'utilité n'est ni monotone ni anti-monotone.

Les premiers algorithmes développés pour résoudre le problème de l'extraction d'ensembles d'éléments à haute utilité étaient incomplets, car ils reposaient sur des stratégies heuristiques visant à réduire l'espace de recherche, ce qui les empêchait de découvrir l'ensemble complet des itemsets à haute utilité. C'est notamment le cas des algorithmes UMining et UMining_H [11].

Il est également intéressant de souligner que le terme " extraction d'ensembles d'éléments à haute utilité " a été utilisé pour la première fois en 2003, bien que la formulation rigoureuse du problème, adoptée par la majorité des chercheurs aujourd'hui, ait été établie en 2005.

La table II.6 présente certains des algorithmes les plus populaires d'extraction d'ensembles d'items à haute utilité, accompagnés d'une comparaison de leurs principales caractéristiques. Cette comparaison porte notamment sur le type de stratégie de recherche adoptée (recherche en largeur ou en profondeur), le nombre de phases utilisées (une ou deux), ainsi que la représentation de la base de données (horizontale ou verticale) [11].

Algorithmes	Type de recherche	Représentation de la BD	Nbr de phases	Inspiré de l'algorithme
Two-Phase	Largeur	Horizontale	Deux	Apriori
UP-Growth	Profondeur	Horizontal(arbre de prefixes)	Deux	FP-Growth
HUI-Miner	Profondeur	Vertical (liste d'utilité)	Une	Eclat
FHM	Profondeur	Vertical (liste d'utilité)	Une	Eclat (amélioration HUI-Miner)
EFIM	Profondeur	Horizontal (avec fusionnement)	Une	LCM
IHUP	Largeur	Horizontal (arbre de préfixes)	Deux	Apriori / FP Growth

Table II.6 : Algorithmes populaires d'extraction d'ensembles d'items à haute utilité

3.4 Analyse comparative des algorithmes HUIM :

L'évolution des algorithmes HUIM reflète un effort constant pour résoudre les limitations du modèle de base introduit par *Two-Phase*. Bien qu'il soit le premier algorithme complet à introduire la notion de *Transaction-Weighted Utility* (TWU) pour élaguer l'espace de recherche, *Two-Phase* reste coûteux en temps et mémoire, notamment à cause de la génération excessive de candidats, y compris ceux qui n'existent pas dans la base.

Pour pallier cette faiblesse, des algorithmes tels que IHUP et UP-Growth ont introduit des structures arborescentes compressées comme IHUP-Tree et UP-Tree permettant de réduire le nombre d'accès à la base de données. Ces structures visent à conserver l'information utile tout en limitant les recalculs. UP-Growth, notamment, se distingue par l'introduction de stratégies d'élagage avancées (DLU, DGU) pour améliorer la performance.

D'un autre côté, des approches plus récentes comme HUI-Miner, FHM et EFIM ont adopté une stratégie basée sur une représentation verticale des données, combinée à une recherche en profondeur. Cette approche permet une exploration plus ciblée et évite la génération explicite de candidats, ce qui représente une avancée significative. Par exemple, FHM introduit le *Estimated Utility Co-occurrence Pruning* (EUCP), un mécanisme efficace pour réduire encore davantage le nombre d'extensions inutiles.

Enfin, EFIM (*Efficient High-Utility Itemset Mining*) est aujourd'hui considéré comme l'un des algorithmes les plus rapides et les plus efficaces. Il

combine plusieurs stratégies comme le fusionnement des transactions, l'optimisation mémoire et le filtrage dynamique, ce qui le rend particulièrement adapté aux grandes bases de données et aux contextes en temps réel.

4. Représentations concises des ensembles d'items à haute utilité :

4.1. Concepts fondamentaux :

Une limite importante des algorithmes d'extraction d'ensembles à haute utilité est qu'ils peuvent générer un grand nombre de motifs lorsque le seuil d'utilité est trop bas, rendant leur analyse difficile. Pour remédier à cela, des représentations concises ont été proposées afin de résumer efficacement les résultats. Ces représentations permettent de réduire considérablement le nombre de motifs tout en conservant l'information essentielle, ce qui facilite l'interprétation et accélère le processus d'extraction. On distingue principalement quatre types de représentations concises comme suit :

4.1.1 Closed HUI (CHUI) : Un ensemble d'items X est dit fermé à haute utilité (CHUI) s'il ne possède aucun sur-ensemble HUI Y ayant le même support, tel que X soit un sous-ensemble de Y ($X \subset Y$) et sup(X)=sup(Y). Cette propriété permet de réduire la redondance des motifs extraits tout en conservant les plus représentatifs. Les CHUI offrent une représentation compacte des résultats, souvent bien plus petite que l'ensemble complet des itemsets à haute utilité. En ajoutant certaines informations comme l'utilité, on obtient une représentation sans perte, permettant de reconstruire tous les motifs sans rebalayer la base de données [14][15].

4.1.2. Maximal HUI(MaxHUI) : Un itemset à haute utilité X est qualifié de maximal s'il n'existe aucun autre itemset à haute utilité Y tel que $X \subset Y$. Autrement dit, X ne doit être inclus dans aucun autre HUI (High-Utility Itemset) plus grand, ce qui en fait un motif final dans l'exploration des HUIs [11][14]

4.1.3. Générateur d'ensembles d'items à utilité haute (GHUI) : Un itemset X est qualifié de générateur à haute utilité (GHUI) s'il respecte deux conditions essentielles, premièrement X est un ensemble d'items dont l'utilité est supérieure ou égale à un seuil minimal prédéfini. Deuxièmes, Aucun sous-ensemble propre $Y \subset X$ n'est lui-même un itemset à haute utilité.

Autrement dit, un GHUI correspond à un motif minimal parmi les itemsets à haute utilité, ne contenant aucun sous-ensemble répondant également au critère d'utilité élevée. Il représente ainsi une forme compacte et non redondante de motifs utiles [11].

4.1.4. Ensembles à haute utilité minimale (MinHUI) : Un ensemble d'items X est qualifié d'ensemble à haute utilité minimale (MinHUI) s'il satisfait deux conditions :

1. Son utilité atteint ou dépasse le seuil minimal requis : $U(X) \geq minUtil$;
2. Aucun de ses sous-ensembles propres $Y \subset X$ ne vérifie ce critère d'utilité ($U(Y) \geq minUtil$) [11].

Contrairement aux ensembles à haute utilité maximale (MaxHUI), qui représentent les plus grands ensembles à forte valeur ajoutée, les MinHUI capturent les plus petits ensembles générateurs d'une utilité élevée. Cette approche permet d'identifier les motifs élémentaires à l'origine des profits significatifs.

4.2. Algorithmes de représentation concise :

Plusieurs algorithmes ont été développés pour extraire efficacement les représentations concises des ensembles d'items à haute utilité présentées précédemment. Le Tableau 7 résume ces algorithmes ainsi que leurs caractéristiques principales. De manière générale, l'extraction de représentations concises s'avère plus rapide que la découverte de l'ensemble complet des HUIs, grâce au nombre réduit de motifs à générer.

Algorithme	Motifs extraits	Nb de phases	Représentation de la BD	Basé sur
MinFHM	MinHUIs	Une	Verticale (liste -utilité)	FHM
GHUI-Miner	GHUIs	Une	Verticale (liste -utilité)	FHM
CHUD	CHUIs	Deux	Verticale (liste -utilité)	DCI_Closed
CHUI-Miner	CHUIs	Une	Verticale (liste -utilité)	DCI_Closed
CLS-Miner	CHUIs	Une	Verticale (liste -utilité)	FHM
EFIM-Closed	CHUIs	Une	Horizontale (avec fusionnement)	EFIM
GUIDE	MHUIs	Une	Flux de données (stream)	UPGrowth
CHUI-Mine	MHUIs	Une	Verticale (liste -utilité)	HUI-Miner

Table II.7 : Algorithmes pour extraire des représentations concises d'itemsets à haute utilité

5. Conclusion :

En résumé, cette étude propose une analyse approfondie du domaine de la fouille de données, avec un accent particulier sur l'extraction d'ensembles d'items à haute utilité (HUIM). Après avoir rappelé les fondements de la fouille des itemsets fréquents, nous avons démontré comment l'approche HUIM introduit une perspective plus réaliste et axée sur la valeur économique.

Nous avons également identifié les principales limitations du processus classique d'extraction des itemsets à haute utilité, en particulier la prolifération de motifs redondants lorsque le seuil d'utilité est paramétré trop bas. Pour pallier cette problématique, notre étude explore différentes représentations concises tels que les ensembles fermés, maximaux, générateurs et minimaux à haute utilité. Ces représentations offrent l’avantage de réduire significativement le volume de résultats tout en préservant l’essentiel de l’information utile à l’analyse.

Chapitre III:
Réalisation & Interprétation

1. Introduction :

Après avoir posé les bases théoriques dans les chapitres précédents, ce troisième chapitre est consacré à la phase d'implémentation et d'expérimentation des principaux algorithmes de fouille d'ensembles à haute utilité (HUIM). Cette étape revêt une importance cruciale, car elle permet de confronter les concepts théoriques aux contraintes réelles de performance, de scalabilité et d'interprétation des résultats dans un environnement applicatif.

Ce chapitre présente, dans un premier temps, les outils et environnements de développement utilisés, notamment Java, Python, Eclipse, et surtout la bibliothèque SPMF, qui offre une large collection d'algorithmes de fouille de motifs. L'intégration de ces outils a permis d'implémenter et d'adapter les algorithmes en fonction des contraintes spécifiques de chaque base de données.

Enfin, une évaluation expérimentale est menée afin d'analyser la performance des algorithmes (FHM, MinFHM, EFIM, HUIC-Miner, HUI-MMU) selon plusieurs critères tels que le temps d'exécution, la consommation mémoire, la qualité des motifs extraits et la compacité des résultats. Cette analyse comparative met en lumière les forces et les limites de chaque approche selon le type de données traité.

2. La présentation des outils de développement :

2.1. Eclipse :

Eclipse est un projet open source initié par IBM en 2001 pour remplacer VisualAge, et devenu un environnement multiplateforme extensible via des plugins. Il est géré par la Eclipse Foundation (fondée en 2004), qui supervise aujourd'hui plus de 420 projets, dont l'IDE Eclipse.

L'IDE Eclipse est un environnement de développement intégré (IDE) utilisé principalement pour Java, mais aussi pour C/C++, PHP, Python, JavaScript, etc. Enfin, Eclipse est un IDE puissant, extensible et open source, idéal pour les projets Java et multi-langages complexes. Son écosystème riche, son architecture de plugins robuste et ses récentes évolutions vers le cloud (Theia, Che) en font une plateforme toujours pertinente pour les développeurs professionnels [16].

2.2. Java :

Java est un langage de programmation orienté objet, développé par Sun Microsystems en 1995. Il est indépendant de la plateforme, ce qui signifie que l'on peut écrire un code une seule fois et l'exécuter partout grâce à la machine

virtuelle Java (JVM). Le langage Java est principalement utilisé pour le développement d'applications de bureau, d'applications web, d'applications Android et de systèmes d'entreprise. Java se distingue par sa simplicité, sa sécurité, sa gestion du multithreading et ses bonnes performances grâce au compilateur Just-In-Time (JIT) [17][18].

2.3. Python :

Python est un langage de programmation polyvalent, open source, puissant et facile à apprendre. Il a été créé à la fin des années 1980 par Guido van Rossum et publié pour la première fois en 1991. Conçu pour être simple à comprendre et agréable à utiliser, même pour les débutants, ce langage est aujourd'hui l'un des plus populaires au monde. Il est utilisé dans de nombreux domaines tels que le développement web, l'intelligence artificielle, la science des données, la robotique ou encore l'automatisation [19].

2.4. Visual Studio :

Visual Studio est un environnement de développement intégré (IDE) développé par Microsoft, conçu pour créer des applications Windows, web, mobiles, cloud et des jeux vidéo. Il prend en charge plusieurs langages tels que C#, C++, Python et JavaScript, et propose de puissants outils, notamment l'éditeur de code intelligent (IntelliSense), le débogueur intégré, la gestion de projets, l'intégration Git/GitHub ainsi que des concepteurs graphiques pour les interfaces utilisateur.

Disponible en versions Community (gratuite), Professional et Enterprise, Visual Studio s'adresse aussi bien aux débutants qu'aux professionnels. Il est largement utilisé dans le développement .NET et s'intègre parfaitement à l'écosystème Microsoft, notamment avec Azure [20].

2.5. Une bibliothèque de donnée extra source (SPMF) :

SPMF (Sequential Pattern Mining Framework) est une bibliothèque open source écrite en Java, spécialisée dans les algorithmes de fouille de motifs fréquents, séquentiels et d'association. Elle regroupe plus de 250 algorithmes pour l'extraction de motifs à partir de bases de données transactionnelles, séquentielles ou temporelles.

SPMF est largement utilisée dans les domaines de la recherche en science des données, de l'intelligence artificielle, de la bio-informatique et de l'analyse de comportements. Son principal avantage réside dans la mise à disposition

d'implémentations claires, compactes et facilement modifiables des algorithmes, ce qui en fait une ressource précieuse pour les étudiants, chercheurs et développeurs travaillant sur la fouille de données. Elle peut être utilisée via une interface graphique simple ou intégrée dans des projets Java [21].

Nous avons téléchargé la bibliotheque SPMF a partir du lien suivant :

- https://www.philippe-fournier viger .com/spmf/index.php?lin k=download. php

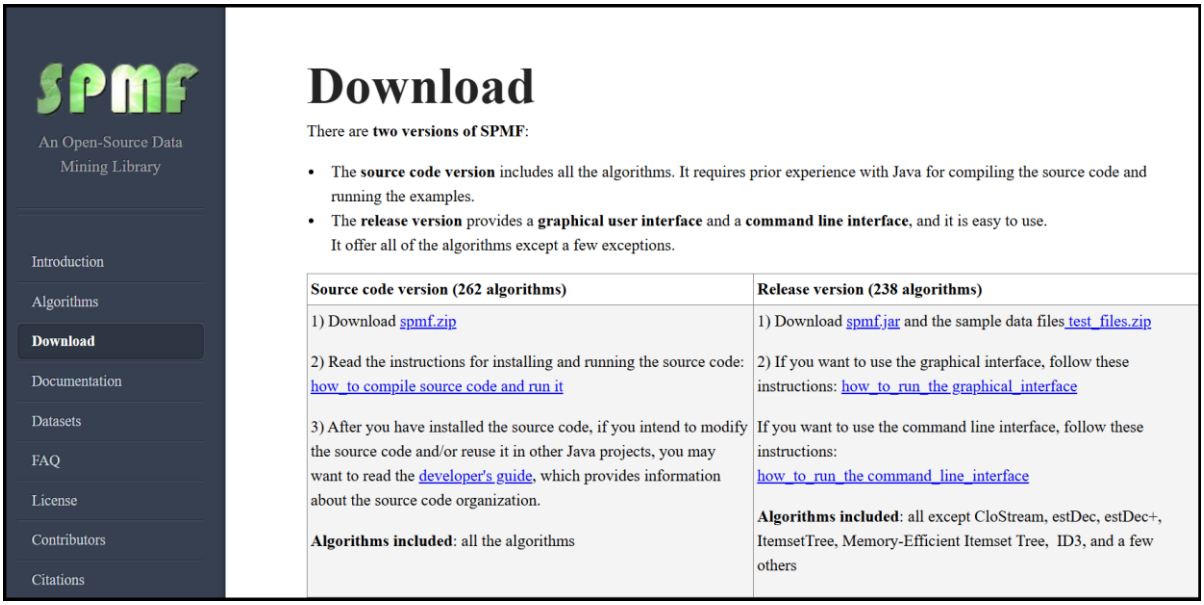


Figure III.6 :Telechargrmrnt du SPMF

3. Les algorithmes proposés pour HUIM :

Plusieurs algorithmes ont été proposés pour extraire les itemsets à haute utilité, parmi lesquels : FHM, MinFHM, EFIM, HUCI-Miner et HUI-MMU. Ces algorithmes utilisent la même entrée (une base de données transactionnelle quantitative associée à une valeur de minimum d'utilité, minutil) et produisent la même sortie (l'ensemble des itemsets à haute utilité).

Tous ces algorithmes modélisent l'espace de recherche de la fouille d'itemsets à haute utilité (HUIM) sous forme de graphe. Pour parcourir cet espace, ils adoptent soit une stratégie en profondeur (Depth-First Search, DFS), soit une stratégie en largeur (Breadth-First Search, BFS).

L'exploration n'est pas réalisée de manière exhaustive. Elle est optimisée grâce à des stratégies d'élagage (pruning strategies), qui permettent d'éliminer

efficacement les itemsets dont l'utilité est inférieure au seuil minimal, ainsi que leurs extensions. Ces techniques permettent aux algorithmes d'explorer l'espace de manière intelligente et de réduire considérablement le coût computationnel.

Les premiers algorithmes reposaient sur une approche en deux phases, consistant d'abord à générer les candidats, puis à évaluer leur utilité. Par la suite, des algorithmes plus performants ont été développés, adoptant une approche en une seule phase, ce qui améliore à la fois la vitesse d'exécution et l'efficacité mémoire.

3.1. L'algorithme CHUI-Miner :

Dans cette section, nous présentons un algorithme performant appelé CHUI-Miner (Closed High Utility Itemset Mining without Candidates), conçu pour extraire efficacement les itemsets fermés à haute utilité sans recourir à la génération explicite de candidats.

L'algorithme CHUI-Miner repose sur une structure de données innovante, appelée EU-List (Extended Utility-List). Cette structure permet de stocker, pour chaque itemset, les informations d'utilité associées aux transactions dans lesquelles il apparaît, facilitant ainsi le calcul précis de son utilité sans avoir à rescanner la base de données à chaque étape.

CHUI-Miner prend en entrée deux paramètres : une base de données transactionnelle D et un seuil d'utilité minimal (min_util) défini par l'utilisateur. En sortie, l'algorithme produit l'ensemble complet des itemsets fermés à haute utilité (CHUIs) présents dans la base de données.

Le fonctionnement de CHUI-Miner repose sur deux étapes principales. La première consiste à construire les EU-Lists des items dits prometteurs, c'est-à-dire ceux dont l'utilité dépasse le seuil spécifié. La seconde étape vise à générer les CHUIs à partir de ces EU-Lists, en combinant les informations stockées et en appliquant des techniques d'élagage ainsi qu'une vérification de fermeture, afin de garantir l'exhaustivité et la compacité des résultats. Le pseudo-code de l'algorithme est présenté à la figure III.7 [22].

Algorithm: CHUI-Miner

Input: *D*, *min_util*;
Output: The complete set of closed^{+} high utility itemsets;
01. ***Scan*** *D* once to find promising items by calculating TWU of items;
02. ***Scan*** *D* again to construct EU-List of promising items;
03. ***GEN-CHUI***(Ø, Ø, *P*, *min_util*);

Figure III.7 : Le pseudo-code de l'algorithme CHUI-Miner

3.1.1 Limitations de l'algorithme CHUI-Miner : Bien que CHUI-Miner soit un algorithme efficace pour extraire des itemsets fermés à haute utilité, il présente plusieurs limitations.

Tout d'abord, sa structure de données, l'EU-List, peut devenir très volumineuse, en particulier dans les bases denses ou lorsque le seuil d'utilité est faible. Cela entraîne une consommation mémoire importante, limitant ainsi son utilisation sur de grands volumes de données.

Par ailleurs, CHUI-Miner n'est pas conçu pour les bases dynamiques : toute modification de la base nécessite une réexécution complète de l'algorithme, ce qui le rend peu adapté aux environnements en temps réel ou évolutifs.

De plus, malgré l'absence de génération explicite de candidats, l'algorithme peut souffrir d'une explosion combinatoire dans les bases contenant un grand nombre d'items ou des transactions longues, ce qui impacte négativement ses performances.

Enfin, il ne produit que des itemsets fermés et non des itemsets maximaux, et ne dispose pas de version native pour le traitement parallèle ou distribué, ce qui restreint son utilisation dans des contextes de Big Data [22].

3.2. L'algorithme CHUI-MinerMax :

Dans la continuité des recherches sur les motifs à haute utilité de représentation concise, CHUI-MinerMax constitue une amélioration de l'algorithme CHUI-Miner, proposée par Fournier-Viger et al. en 2016. Alors que CHUI-Miner extrait les motifs fermés à haute utilité (Closed High Utility Itemsets), CHUI-MinerMax se concentre sur l'extraction des motifs générateurs à haute utilité maximaux (High Utility Closed Itemset Generators), ce qui permet de produire un ensemble encore plus réduit tout en préservant la capacité de reconstruire l'ensemble complet des motifs à haute utilité.

L'approche adoptée dans CHUI-MinerMax repose sur l'idée de ne conserver que les motifs informativement significatifs et non redondants, ce qui réduit considérablement la taille des résultats sans compromettre leur expressivité. L'algorithme utilise des structures de données similaires à celles de CHUI-Miner (comme les utility lists), mais intègre en plus des mécanismes permettant de filtrer les générateurs maximaux en fonction de leur fermeture utilitaire.

En pratique, CHUI-MinerMax s'avère efficace pour trouver un compromis entre concision et richesse des motifs : il extrait moins de motifs que CHUI-Miner tout en fournissant suffisamment d'informations pour permettre la reconstruction des motifs complets si nécessaire. Il est donc particulièrement adapté aux contextes où l'on cherche à réduire la surcharge cognitive liée à l'interprétation des résultats tout en conservant l'essentiel des connaissances [23].

3.2.1 Limitations de l'algorithme CHUI-MinerMax : L'algorithme CHUI-MinerMax, bien qu'efficace pour extraire des motifs concis à haute utilité, présente plusieurs limitations. Tout d'abord, en ne conservant que les générateurs maximaux, il peut omettre certains motifs utiles mais non maximaux, ce qui peut s'avérer problématique dans des analyses nécessitant une couverture exhaustive. Ensuite, la reconstruction des motifs d'origine à partir des générateurs impose un traitement supplémentaire, susceptible d'entraîner un surcoût en temps et en ressources.

Par ailleurs, comme d'autres algorithmes de fouille d'ensembles à haute utilité (HUIM), CHUI-MinerMax est sensible au choix du seuil d'utilité minimal (minUtil) : un seuil trop bas accroît fortement le nombre de calculs, tandis qu'un seuil trop élevé risque de faire passer à côté de motifs pertinents. Enfin, sa mise en œuvre algorithmique est plus complexe, notamment en raison des vérifications nécessaires pour garantir la minimalité des générateurs, ce qui peut en limiter l'accessibilité pour des utilisateurs non spécialisés [23].

3.3. L'algorithme HUCI-Miner-Generators :

L'algorithme HUCI-Miner-Generators a pour objectif d'extraire les itemsets fermés à haute utilité (HUCIs) ainsi que leurs générateurs minimaux (HUGens), afin de réduire la redondance des motifs extraits tout en préservant l'information essentielle.

Son fonctionnement repose sur deux étapes principales : tout d'abord, l'extraction des HUCIs à partir de la base de données ; ensuite, la recherche des générateurs pour chaque HUCI, c'est-à-dire des sous-ensembles minimaux qui

possèdent la même utilité et apparaissent dans les mêmes transactions. Ces générateurs permettent de produire des règles plus concises et informatives, particulièrement utiles pour l'analyse.

L'intérêt de cette méthode réside dans sa capacité à simplifier les résultats en ne conservant que les motifs les plus significatifs. Toutefois, le coût de calcul peut devenir élevé, car l'algorithme doit explorer de nombreux sous-ensembles, en particulier dans les bases de données denses.

En résumé, HUCI-Miner-Generators offre une représentation plus compacte et interprétable des motifs à haute utilité, tout en facilitant la génération de règles d'association pertinentes [24].

3.3.1 Limitations de l'algorithme HUCI-Miner-Generators Bien que HUCI-Miner-Generators permette d'extraire des motifs utiles de manière plus concise, il présente plusieurs limitations. L'algorithme est coûteux en calcul, car il doit examiner de nombreux sous-ensembles pour identifier les générateurs de chaque itemset fermé. Cela entraîne également une consommation mémoire importante, en particulier dans les bases de données volumineuses ou denses.

Il est également peu adapté aux bases dynamiques, puisqu'une simple modification des données impose de relancer l'ensemble du processus. De plus, son efficacité dépend fortement de la qualité de l'algorithme utilisé pour extraire les itemsets fermés. Enfin, dans des bases très denses, le nombre de motifs générés peut rester élevé, limitant ainsi les gains attendus en termes de compacité.

En somme, malgré sa capacité à réduire la redondance, HUCI-Miner-Generators reste limité par sa complexité algorithmique, sa dépendance à d'autres méthodes et son manque de flexibilité [24].

3.4. L'algorithme MinFHM :

L'algorithme MinFHM est une extension de l'algorithme FHM, conçu pour extraire uniquement les itemsets minimaux à haute utilité (MinHUIs) à partir d'une base de données transactionnelle contenant des valeurs d'utilité. Il commence par un premier balayage de la base afin de calculer la TWU (Transaction-Weighted Utility) de chaque item. Seuls les items dont la TWU est supérieure ou égale à un seuil minimal d'utilité (minutil) sont conservés, formant un ensemble I*. Ces items sont ensuite triés selon un ordre croissant de TWU.

Un second balayage permet de réorganiser les transactions selon cet ordre, de construire les listes d'utilité pour chaque item de I*, et de générer la structure

EUCS (Estimated Utility Co-Occurrence Structure), qui enregistre les valeurs de TWU pour toutes les cooccurrences valides d'items (sous forme de triplets (a,b,c)(a, b, c)(a,b,c), où c=TWU({a,b}). Cette structure, efficace en mémoire, accélère l'élagage de l'espace de recherche.

L'exploration des itemsets est ensuite réalisée à l'aide d'une recherche en profondeur (DFS) via une procédure récursive qui construit progressivement les itemsets candidats, tout en vérifiant leur utilité et en éliminant ceux qui ne satisfont pas les contraintes minimales. MinFHM permet ainsi une fouille plus ciblée, rapide et optimisée des motifs pertinents. La figure 8 présente un aperçu de la procédure de l'algorithme MinFHM [25].

Algorithm 1: The MinFHM algorithm

input : D: a transaction database, *minutil*: a user-specified threshold
output: the set of high-utility itemsets

1 Scan D to calculate the TWU of single items;
2 $I^* \leftarrow$ each item i such that TWU$(i) \geq$ *minutil*;
3 Let $\succ$ be the total order of TWU ascending values on I^*;
4 Scan D to build the utility-list of each item $i \in I^*$ and build the *EUCS*;
5 Output each item $i \in I^*$ such that SUM$(\{i\}.utilitylist.iutils) \geq$ *minutil*;
6 `Search` $(\emptyset, I^*,$ *minutil*, *EUCS*$)$;

Figure III.8 : Algorithme MinFHM

3.4.1 Limitations de l'algorithme MinFHM : L'algorithme MinFHM, bien qu'efficace pour extraire les itemsets minimaux à haute utilité, présente plusieurs limitations. Il nécessite deux balayages complets de la base de données, ce qui peut s'avérer coûteux pour les bases volumineuses. De plus, en se concentrant uniquement sur les itemsets minimaux, il ne découvre pas l'ensemble complet des itemsets à haute utilité, ce qui peut limiter son intérêt selon les objectifs d'analyse.

La construction et la gestion des utility-lists, notamment les opérations de jointure, peuvent engendrer une charge computationnelle importante, en particulier sur des bases denses ou contenant de longues transactions. Par ailleurs, bien que la structure EUCS (Estimated Utility Co-occurrence Structure), utilisée

pour optimiser l'élagage, soit conçue pour être efficace, sa taille peut croître rapidement avec le nombre d'items fréquents, entraînant une augmentation significative de l'utilisation mémoire.

Enfin, lorsque le seuil d'utilité minimale est très bas, le nombre d'itemsets candidats peut exploser, ce qui affecte fortement les performances globales de l'algorithme [25].

4. Conception et implémentation :

4.1. L'architecture Générale :

La conception globale d'un système, également appelée architecture générale, définit la structure du projet en précisant ses composants, leurs interactions et les principes de conception qui guident leur implémentation (voir la figure III.9).

Dans le cadre de la comparaison des algorithmes CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators et MinFHM pour l'exploration de données à haute utilité, l'architecture proposée se compose des étapes suivantes :

1. Sauvegarde de la base de données initiale, en veillant à la conservation et à l'intégrité des données d'origine.
2. Prétraitement des données, suivi de la phase de fouille de données en appliquant successivement les algorithmes CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators et MinFHM.
3. Interprétation des résultats obtenus et réalisation d'une comparaison approfondie afin d'évaluer leurs performances respectives et leur précision dans la détection des éléments à haute utilité.

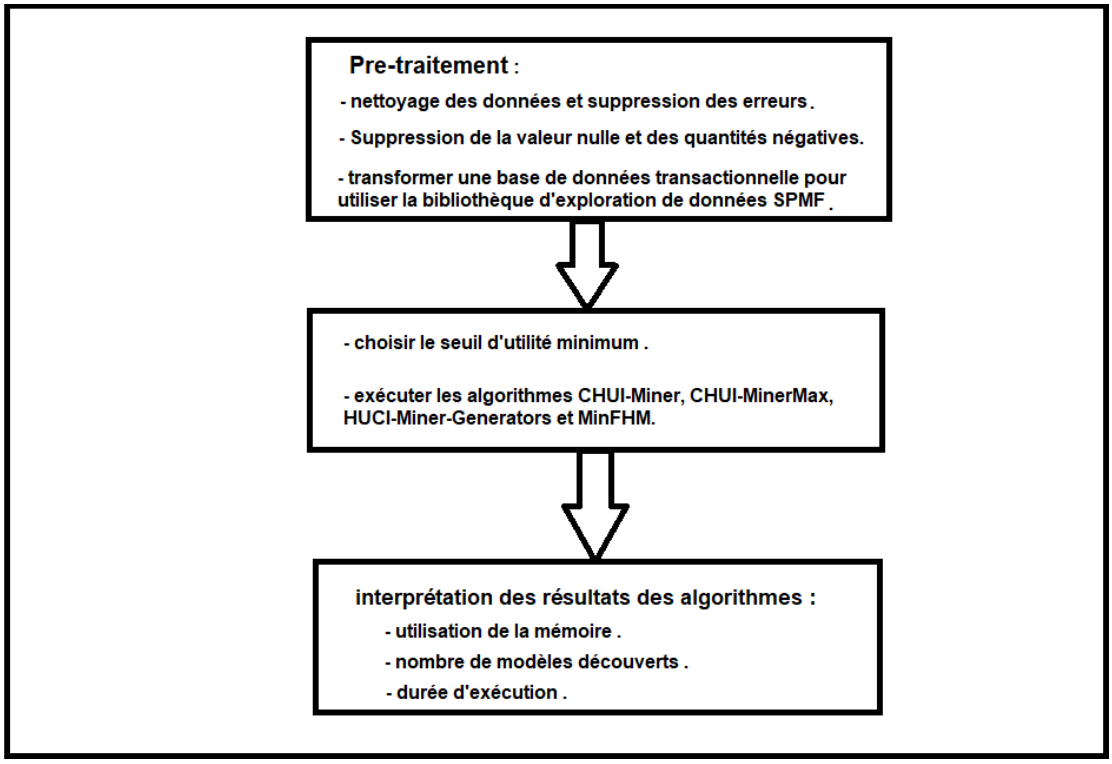


Figure III.9 : schéma des étapes de l'étude comparative proposée

4.2. le premier cas d'étude :

Les données traitées dans ce projet proviennent directement d'un magasin de cosmétiques (voir Figure III.10). La base de données principale enregistre des informations essentielles telles que les détails des produits, les ventes réalisées, les quantités vendues, ainsi que d'autres données clés liées aux transactions commerciales.

Ces données ont été extraites sous forme de fichiers Excel. Le fichier contient des enregistrements organisés ligne par ligne, où chaque ligne représente une opération de vente, c'est-à-dire l'achat d'un produit par un client. Chaque ligne est définie par les éléments suivants :

- La référence du produit : (IDProduit).
- Le nom du produit : (Designation).
- Le numéro de bon de commande (IDBon).
- La quantité vendu (Qte).
- Le profit généré par le produit (profit).

1	IDDetailBon	IDBon	IDProduit	Designation	Qte	Prix_Achat	PrixV_Detail	profit
2	1573	18	4407	KASSA HAODA NOIRE	1	100	150	50
3	4548	40	548	SAVON DOVE	1	155	170	15
4	4541	40	1377	BAD SENSODYNE DOUCEUR	1	360	400	40
5	4564	40	1483	CRE EMILY	1	160	200	40
6	4544	40	1663	PORTE SAVON FLOWER	1	43	70	27
7	4543	40	2621	GLD JOHNSONS BEBE 200ML	1	350	400	50
8	4547	40	2790	COTON TIEGE FAMILY	4	73	90	68
9	4539	40	3122	DNT COLGATE MAX WHITE ONE INTENSE	1	255	350	95
10	4563	40	3240	EYLINER KISS BEAUTY LOVELY	4	125	200	300
11	4552	40	4951	COUCHES DEROIT FRELAX PS	1	130	160	30

Figure III.10 : Base de données de produits cosmétiques

4.2.1 Les statistiques de la base de donné de produit cosmétique :

Le nombre des items	7723
Le nombre des transactions	170374
La taille de la base de donné	5,95 Mo
Total Utilité	45229212
Maximum Id of item	16295
La longueur moyenne des transactions	3,86
La longueur maximale d'une transaction	151

Table III.8 : Les statistiques de la base de donné de produit cosmétique

4.2.2 Prétraitement des données : Les fichiers fournis pour cette base de données ne sont pas directement exploitables par la bibliothèque SPMF (Sequential Pattern Mining Framework), car ils ne respectent pas le format requis. En effet, ces fichiers, souvent sous forme Excel ou CSV, présentent les données sous un format de tableau classique, où chaque ligne correspond à un produit acheté, accompagné d'informations telles que l'identifiant du produit, le numéro du bon de commande, la quantité et le profit.

Or, SPMF attend un format transactionnel spécifique, dans lequel chaque ligne représente une transaction complète, c'est-à-dire l'ensemble des produits achetés par un client lors d'un même acte d'achat. Ce format est structuré de la manière suivante : une liste des identifiants des produits achetés, suivie de l'utilité

totale de la transaction (Total Utility), puis des profits individuels de chaque produit.

Par conséquent, un prétraitement est nécessaire pour transformer les données initiales. Ce processus consiste à : regrouper les produits par numéro de bon (IDBon), extraire les identifiants des produits et leurs profits, calculer l'utilité totale, puis générer un fichier texte structuré conformément au format attendu par SPMF.

Ce prétraitement est une phase clé pour assurer la compatibilité des données avec les algorithmes de fouille à utilité élevée et garantir leur exécution sans erreur. La conversion a été réalisée à l'aide d'un script Python, où chaque transaction correspond à un ticket de caisse unique, regroupant l'ensemble des produits achetés lors de cette vente. La Figure III.11 présente un extrait de la base une fois convertie au format SPMF :

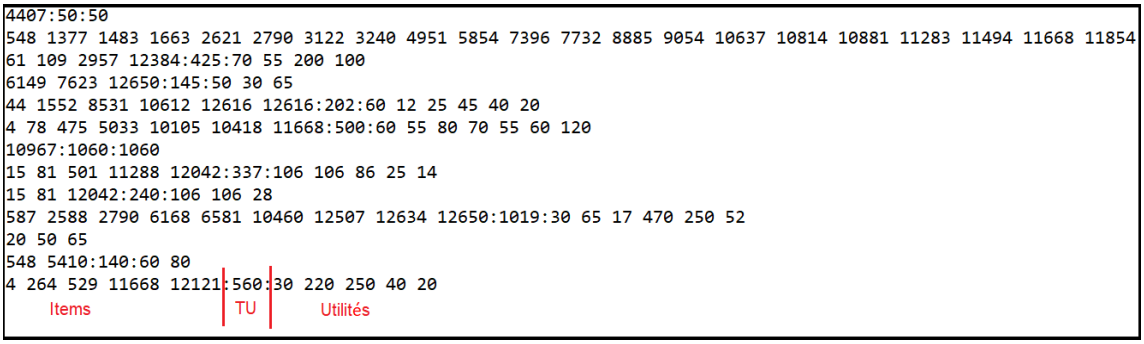


Figure III.11 : Un extrait d'un fichier format SPMF

Dans cette étape, nous appliquons des algorithmes d'extraction de motifs à utilité élevée pour obtenir une représentation concise des données, notamment CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators et MinFHM. Une analyse des résultats sera ensuite réalisée afin d'en tirer des conclusions pertinentes.

4.2.3 Etape de fouille de données : Eclipse est l'environnement de développement que nous avons choisi pour la mise en œuvre de ce projet (Figure III.12). Voici le lien de téléchargement :

https://www.eclipse.org/downloads/packages/release/2025-06/r/eclipse-ide-java-developers

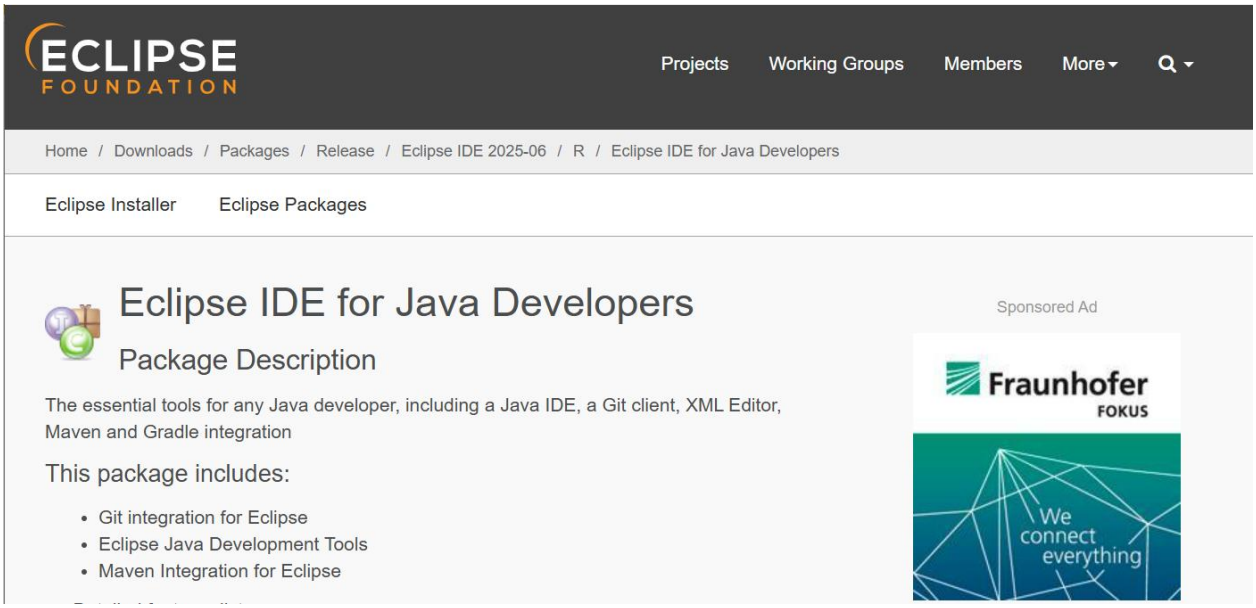


Figure III.12 : Eclipse IDE

Nous avons créé un nouveau projet Java et ajouté la bibliothèque SPMF au répertoire source (src) de ce projet (Figure III.13) .

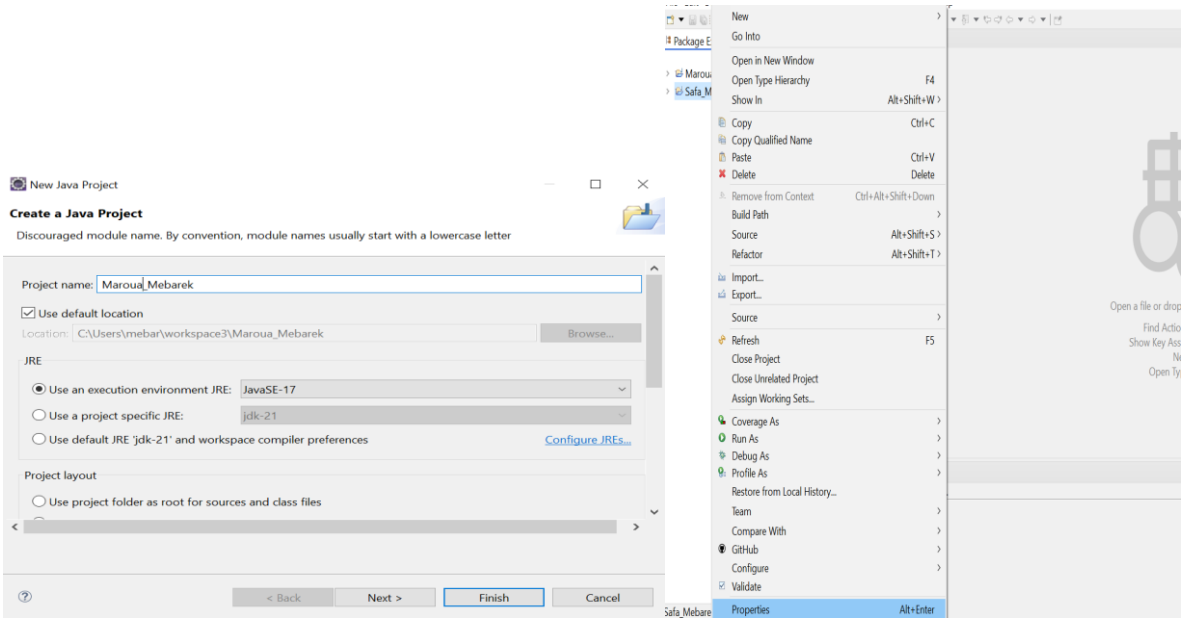


Figure III.13 : Créer un nouveau projet JAVA

Ajouter la bibliothèque SPMF (Figure III.14):

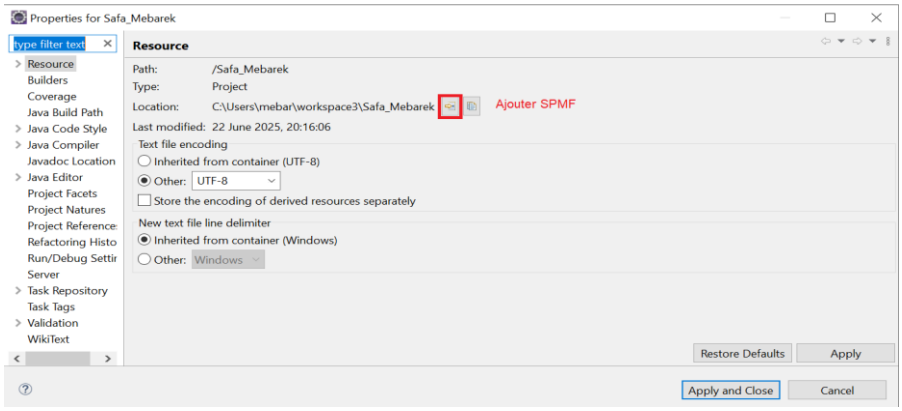


Figure III.14 : Ajouter SPMF

L'algorithme MinFHM a été lancé à l'aide de la bibliothèque SPMF, et les résultats ont été automatiquement enregistrés dans un fichier texte (EXPERIMENT_RESULT.txt). Un aperçu de ces résultats est présenté dans la Figure III.15 .

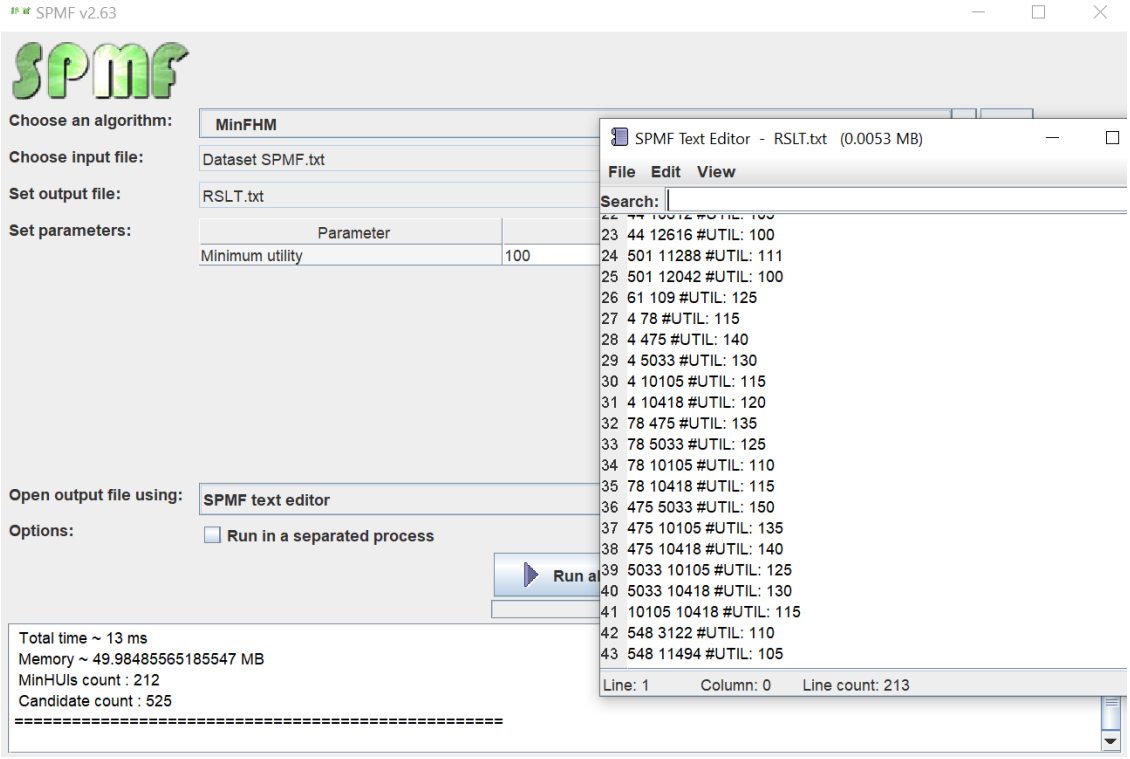


Figure III.15 : Résultat de l'exécution de l'algorithme MinFHM

4.2.4 Interprétation et comparaison les résultats : Nous avons réalisé des expériences afin d'évaluer les performances des algorithmes proposés selon trois critères : le temps d'exécution, l'utilisation de la mémoire et le nombre de motifs découverts lors de l'application des algorithmes CHUI-Miner, CHUI-

MinerMax, HUCI-Miner-Generators et MinFHM pour des représentations concises.

Les expérimentations ont été menées sur un ordinateur équipé d'un processeur Intel Core i3 de 11e génération (64 bits), fonctionnant sous Windows 10 et disposant de 8 Go de RAM. Les algorithmes ont été testés pour plusieurs valeurs du seuil minUtil, variant de 5 000 à 600 000. Les résultats obtenus sont synthétisés dans les figures Figure 16, Figure 17 et Figure 18.

4.2.5 Résultats :

Dans le cadre de notre expérimentation, nous avons évalué les performances de quatre algorithmes d'extraction d'ensembles d'items à haute utilité : CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators et MinFHM. Pour ce faire, nous les avons appliqués successivement à plusieurs ensembles de données.

Afin d'analyser leur comportement, nous avons fait varier progressivement le seuil d'utilité minimale (minUtil). Ce seuil détermine le niveau minimum de profit ou d'importance qu'un itemset doit atteindre pour être considéré comme utile. Nous l'avons diminué étape par étape afin d'observer la réaction des algorithmes dans différents contextes : augmentation du nombre d'itemsets générés, allongement du temps de traitement, surcharge mémoire, ou apparition de résultats significativement plus riches.

L'objectif de cette démarche était d'identifier les limites de chaque algorithme en termes de temps d'exécution, de consommation mémoire et de capacité à générer des ensembles d'items pertinents.

Pour chaque configuration testée, nous avons soigneusement mesuré et consigné : le temps d'exécution (en secondes), la mémoire utilisée (en mégaoctets) ainsi que le nombre total d'ensembles d'items à haute utilité extraits.

La relation entre le seuil minUtil et le temps nécessaire à l'exécution des algorithmes est présentée dans la figure suivante (Figure III.16) :

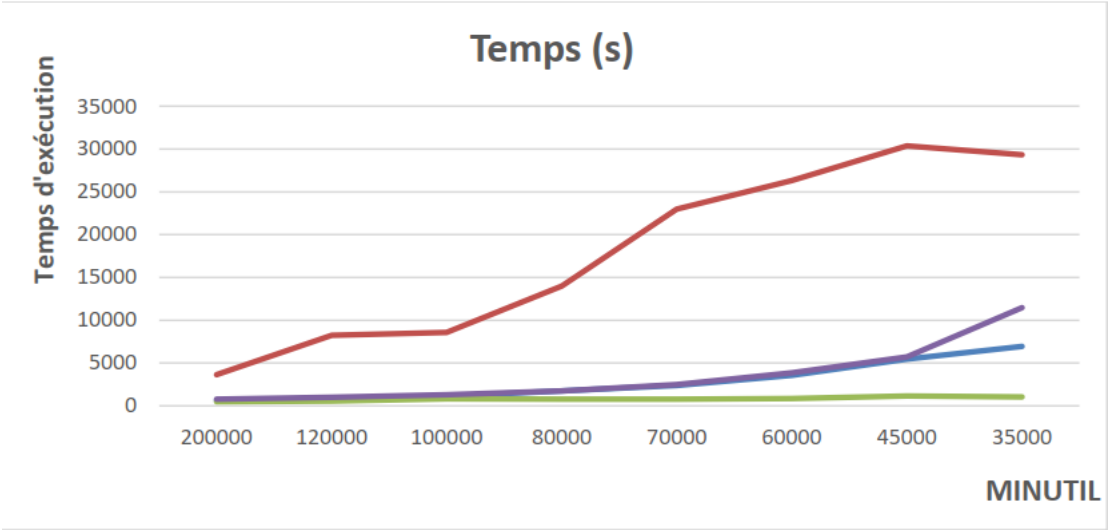


Figure III.16 : Temps d'exécution des algorithmes (CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators, MinFHM)

La Figure III.16 présente une comparaison des temps d'exécution. Pour chaque ensemble de données, MinFHM s'avère être l'algorithme le plus rapide, suivi de CHUI-Miner et CHUI-MinerMax. Enfin, HUCI-Miner-Generators affiche le temps d'exécution le plus long.

La relation entre le seuil minUtil et les ressources mémoire utilisées lors de l'exécution est illustrée dans la figure suivante (Figure III.17) :

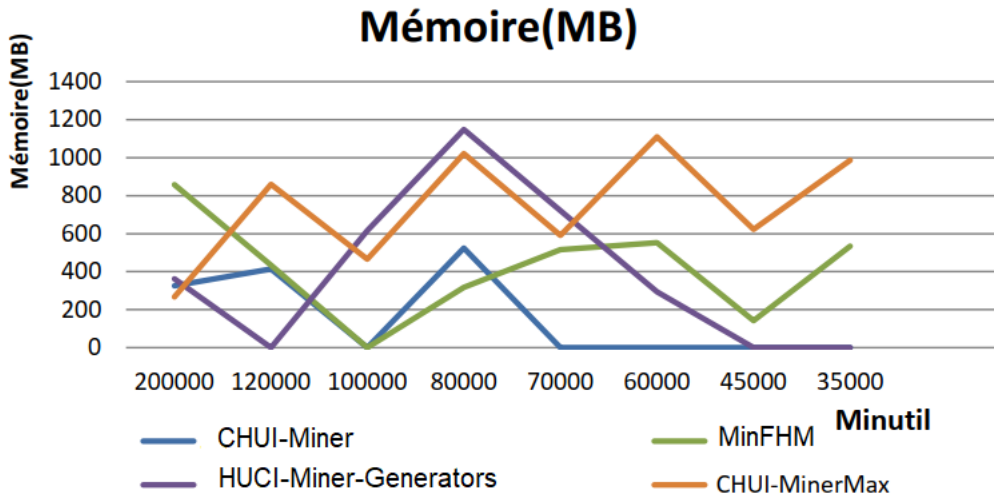


Figure III.17 : La consommation du mémoire par les algorithmes (CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators, MinFHM)

La Figure III.17 illustre la comparaison de la consommation mémoire des différents algorithmes. De manière générale, CHUI-Miner consomme moins de

mémoire que MinFHM et CHUI-MinerMax, tandis que HUCI-Miner-Generators utilise une quantité de mémoire significativement plus élevée.

La comparaison du nombre d'ensembles d'items à haute utilité extraits est présentée ci-dessous :

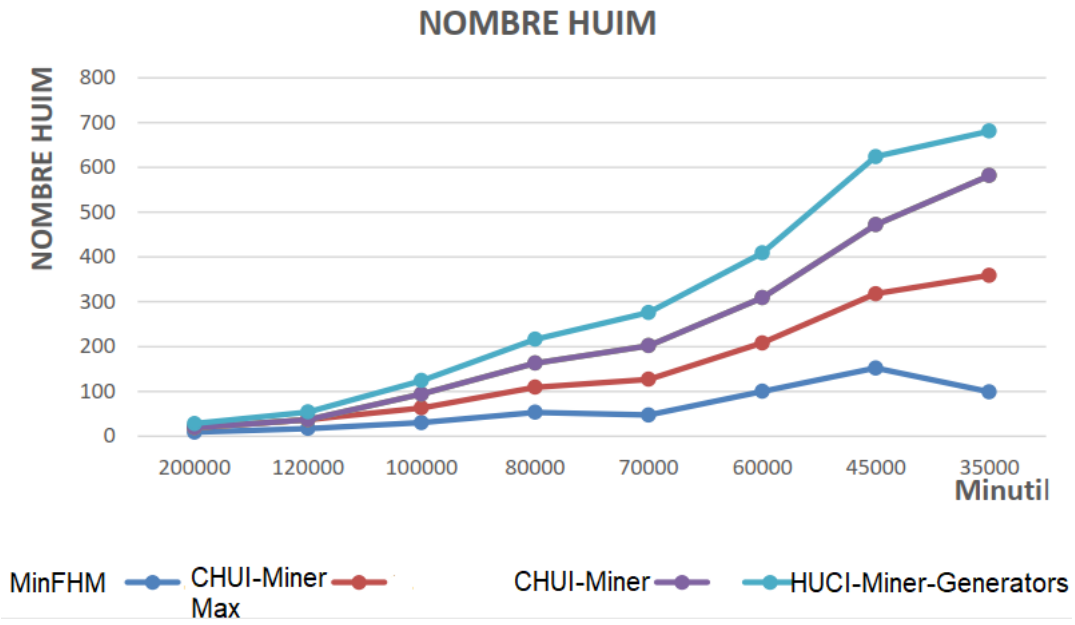


Figure III.18 :Nombre des HUIM des algorithmes (CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators, MinFHM)

La Figure III.18 présente une comparaison du nombre d'ensembles d'items à haute utilité extraits par chaque algorithme. MinFHM génère le plus petit nombre de motifs, suivi de CHUI_MinerMax, qui en extrait légèrement plus. CHUI-Miner en produit un nombre plus important, tandis que HUCI-Miner-Generators se distingue en générant la plus grande quantité d'ensembles à haute utilité.

4.3. Le deuxième cas d'étude :

Les données traitées dans ce projet proviennent directement d'un magasin. La base de données principale enregistre des informations essentielles telles que les détails des produits, les ventes réalisées, les quantités vendues, ainsi que d'autres données clés liées aux transactions commerciales. La Table III.9 présente les statistiques de la base de données FoodMart. Nous avons téléchargé cette base de données à partir du lien suivant :

https://www.philippe-fournier-viger.com/spmf/datasets/foodmart.txt

Le nombre d'items	1559
Le nombre des transactions	4141
La taille de la base de donnée	5.47Mo
Nombre moyen de produits par transaction	4.42

Table III.9 : Les statistiques de la base de données (FoodMart)

Pour cette base, la phase de prétraitement n'est pas nécessaire, car elle est déjà structurée et fournie au format SPMF. Nous pouvons donc passer directement à l'expérimentation avec les algorithmes de représentation concise. Voici un aperçu de la base de données FoodMart au format SPMF (Figure III.19) :

```
214 260 763:2417:846 552 1019
195 385 778 961:1744:354 312 816 262
12 328 871 934:2312:462 504 1016 330
282 359 715:1117:279 528 310
16 586 785 898 975 1368 1551:4992:1149 568 684 522 1128 710 231
151 557 1299:1406:582 272 552
128 224 540 667 1276 1336 1453:3810:106 346 153 798 429 1356 622
803 1346:1642:702 940
258 1147:942:516 426
308 475 1130 1350 1440:3239:872 783 465 276 843
497 701:898:748 150
248 354 760 774 1222 1430 1502:5723:344 1125 561 735 490 1344 1124
1234 1505:1324:224 1100
27 110 366 610 843 1065 1381 1419:5462:795 705 708 908 882 564 651 249
638 1191 1347:1895:944 420 531
16 216 326 409 925 1291 1314:5465:1532 1080 494 1173 304 690 192
```

Figure III.19 : La base FoodMart format SPMF

4.3.1 Exécution de l'algorithme CHUI-Miner :

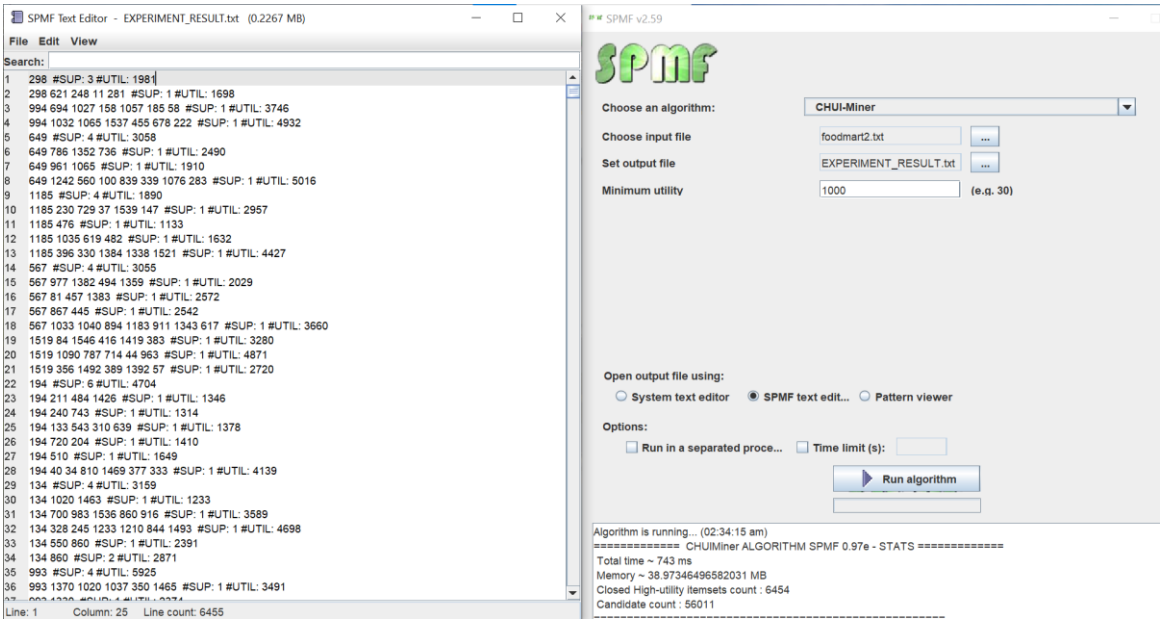


Figure III.20 : Exécution de l'algorithme CHUI-Miner

Afin d'évaluer les performances des algorithmes proposés, nous avons mené des expériences en nous appuyant sur trois critères principaux : le temps d'exécution, la consommation mémoire et le nombre de motifs (patterns) extraits. Ces tests ont été réalisés à l'aide des algorithmes CHUI-Miner, CHUI_MinerMax et HUCI-Miner-Generators, en utilisant des représentations concises.

La relation entre le seuil minutil, le temps nécessaire à l'exécution des algorithmes, la consommation mémoire et le nombre d'ensembles d'items à haute utilité extraits est présentée dans la figure suivante (Figure III.21) :

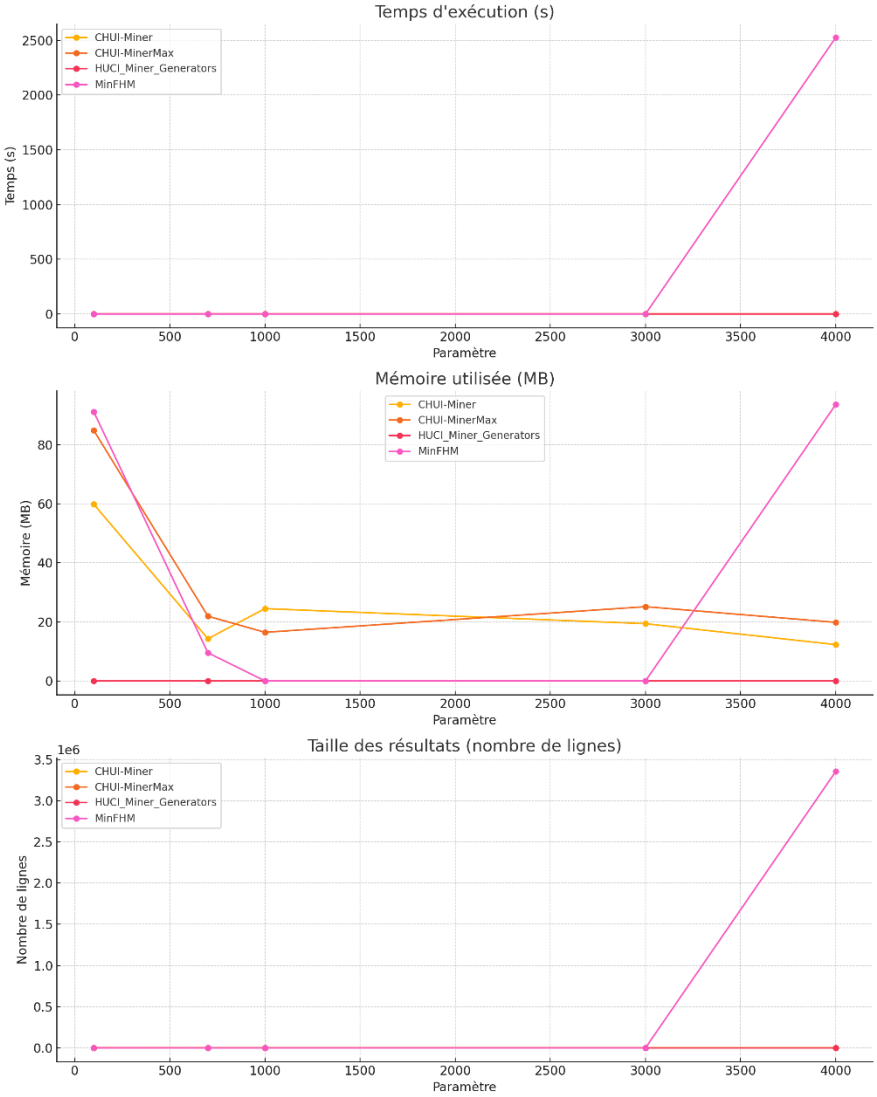


Figure III.21 : Résultats de l'exécution des algorithmes CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators, MinFHM

Comme illustré dans la Figure III.21, premièrement une comparaison des temps d'exécution montre que MinFHM est le plus rapide sur l'ensemble des jeux de données, suivi par CHUI-Miner, puis CHUI-Miner-Generators. Enfin, L'algorithme CHUI_MinerMax demande beaucoup de temps d'exécution.

Deuxièmement, une comparaison de consommation de mémoire montre que l'algorithme CHUI-Miner-Generators consomme moins d'espace que MinFHM et CHUI-Miner, tandis que HUCI-Miner-Geneators utilise une quantité de mémoire considérable.

Enfin, pour le troisièmes critères le nombre d'ensembles d'items à haute utilité extraits . On peut voir que l'algorithme HUCI-Miner-Generators et CHUI-

Miner retourne grand nombre d'ensemble d'éléments à utilité élevé que CHUI-MinerMax, puis MinFHM retourne un mois nombre .

5. Conclusion :

En résumé, cette étude comparative des algorithmes de fouille d'ensembles d'items à haute utilité (HUIM), dans le cadre d'une représentation concise appliquée aux bases de données transactionnelles, a permis d'évaluer les performances respectives de chaque méthode.

En nous appuyant sur plusieurs jeux de données au format SPMF, nous avons analysé les algorithmes selon trois critères essentiels : le temps d'exécution, la consommation mémoire et le nombre de motifs extraits.

Les résultats obtenus mettent en évidence les avantages spécifiques de chaque approche ainsi que leurs limites, ce qui peut orienter le choix d'un algorithme en fonction des besoins et des contraintes propres à un cas d'usage donné.

Conclusion générale

Conclusion générale:

Dans un contexte de croissance exponentielle des données, ce mémoire s'inscrit dans le domaine de la fouille de données, en particulier l'extraction des ensembles d'items à haute utilité (HUIM). Cette approche ne se limite pas à identifier des motifs fréquents, mais vise également à extraire des ensembles significatifs en termes de valeur métier. L'objectif principal de cette étude était d'évaluer l'impact des représentations concises sur les performances des algorithmes HUIM, en tenant compte de trois critères : le temps d'exécution, la mémoire consommée et la taille des résultats générés.

Pour cela, nous avons mené une analyse théorique approfondie, complétée par des expérimentations pratiques sur plusieurs jeux de données. Nous avons ainsi étudié et comparé les performances de quatre algorithmes : CHUI-Miner, CHUI-MinerMax, HUCI-Miner-Generators et MinFHM. Les résultats obtenus permettent de mieux comprendre le rôle des représentations concises dans l'optimisation des ressources et la réduction de la redondance, facilitant ainsi l'interprétation et l'exploitation des connaissances extraites.

Les résultats obtenus confirment l'efficacité des algorithmes étudiés pour extraire des ensembles d'items pertinents tout en optimisant l'utilisation des ressources. En particulier, des algorithmes comme MinFHM se sont distingués par leur capacité à produire des résultats concis et rapides, même sur des jeux de données complexes. Ces travaux mettent ainsi en évidence l'importance des représentations concises, qui permettent de réduire la redondance et de simplifier l'analyse des résultats, répondant aux besoins croissants d'efficacité dans les environnements Big Data.

Par ailleurs, notre étude souligne les limites des approches classiques telles qu'Apriori, qui ne prennent pas en compte l'utilité réelle des items, et met en avant l'intérêt des structures avancées comme les utility lists ou les représentations fermées et génératrices, bien mieux adaptées aux exigences actuelles. Elle révèle également les principaux défis associés au HUIM, notamment la croissance exponentielle de l'espace de recherche et la nécessité d'élagage intelligent pour améliorer les performances.

En conclusion, ce travail confirme le rôle central des représentations concises dans l'efficacité des méthodes HUIM et constitue une base solide pour le développement de solutions plus performantes. Il ouvre des perspectives intéressantes pour des recherches futures, telles que la conception de représentations plus adaptatives, l'amélioration de la qualité sémantique des

motifs extraits, ou l'adaptation de ces méthodes à des environnements temps réel et incertains. Ces avancées pourraient trouver des applications prometteuses dans des domaines variés comme le e-commerce, la santé ou les systèmes de recommandation.

Bibliographies

[1] A.Djeffal.cours " Fouille de données avancée".université Mohamed Khider, Biskra

[2] S.Bondyopadhyay and U.Maulik,L.B.Holder,D.J.Cook " Knowledge Discovery and Data Mining ", in Advanced Methods for Knowledge Discovery from Complex Data, Springer-Verlage, 2005

[3] J.Han,M.Kamber,J.Pei, "Data Mining : Concepts and Technique" 2006

[4] M.Hemam.cours, " Entrepot et Fouille de Donnee ". Université Abbes Lghrour. Faculté des Sciences et de la Technologie. Khenchela,pp40.derniere version 2022

[5] D. Sciences, "Parcours ecd - extraction des connaissances à partir des données,"pp. 1–2, 2002

[6] S.Tuffery Data mining et statistique décisionnelle université de Rennes 2005

[7] J.Han, M.Kamber, J.Pei ·Les technique de la fouille de donnée ,"Data Mining: Concepts and Techniques",2011

[8] Y.Abboud, Fouille de motifs : entre accessibilité et robustesse. PhDthesis, Université de Lorraine, 2018.

[9] https://www.philippe-fournier-viger.com/25yearsFIM.pdf

[10] J. M.Luma,P.Fournier-Viger,and S.Ventura, " Frequent Itemset Mining : a 25 Years Review ".

[11] P.Fournier- viger." Survey high utility itemset mining " ,2019 .

[12] P.Fournier- viger." A Survey of Itemset Mining " ,2017 .

[13] M.Mayo, " Frequent Pattern Mining and the Apriori Algorithm : a concise Technical overview ", 2016

[14] Y.Yang, J.Ding, H.Wang, H.Xing, E.Li, "A High Utility Itemset Mining Algorithm Based on Particle Filter ", 2023

[15] Dam, T.-L., Li, K., Fournier-Viger, P., Duong, H.: CLS-Miner: Efficient and Effective Closed High utility Itemset Mining. Frontiers of Computer Science. Springer 2018.

[16] Eclipse (projet), https://fr.wikipedia.org/wiki/Eclipse_projet, version Mars 14,2025

[17] Qu'est-ce que la technologie Java et pourquoi en ai-je besoin, https://www.java.com/fr/download/help/whatis_java.html , 2025

[18] “Introduction to java”, https://www.geeksforgeeks.org/java/introduction-to-java,version 14juin2025

[19] D.Amos,D.Bader,J.Jablonski, F.Heisler, “A pratical introduction to python”,2012_2020

[20] K.Chowdhury, “ Mastering Visual Studio 2019”,

[21] P.Fournier-Viger , Jerry Chun-Wei Lin, A.Gomariz, T.Gueniche, A.Soltani, Z.Deng, H.Thanh Lam, “SPMF: A Java Open-Source Pattern Mining Library”,2016

[22] C.Wei Wu , P.Fournier-Viger , J.Yuan Gu , V.S. Tseng, " Mining Closed High Utility Itemsets without Candidate Generation", 2012.

[23] P.Fournier-Viger, J.Chun-Wei Lin, C.Wei Wu , V.S.Tseng, “Mining Maximal Generators of High Utility Itemsets Using the CHUI-Miner Algorithm Framework”,2016

[24] P.Fournier-Viger, J.Chun-Wei Lin, C.Wei Wu , V.S.Tseng, “Mining Maximal Generators of High Utility Itemsets Using the CHUI-Miner Algorithm Framework.”, from Advanced Data Mining and Applications (ADMA). 2016

[25] P.Fournier-Viger, J.Chun-Wei Lin, C.Wei Wu , V.S.Tseng , U.Faghihi, “Mining Minimal High-Utility Itemsets”, 2016