République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université de 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

Master 2 en informatique

Spécialité : Réseau & Multimédia

THEME

Analyse Comparative des Algorithmes de Haute utilité pour représentation concise dans les bases de données transactionnelles

Présenté par :

- BENDJEFFAL LYNDA
- HARRAR IBTISSEM

Soutenu publiquement le :

Devant le jury composé de:

Président : ZAOUACHE Djaaffar

Examinateur : BENMESSAHAL Ilyes

Encadreur : ATTIA Abdelouahab

2023/2024

Remerciement

Tout d'abord, nous remercions Dieu Tout-Puissant qui nous a donné la force et la patience pour accomplir cet humble travail.

La première personne que nous tenons à remercier est notre ***encadrant Mr Abdelouahab ATTIA*** *pour ses conseils, sa confiance, sa patience et ses bonnes explications qui ont constitué une grande contribution et sans lesquelles il n'aurait pas été possible de mener à bien ce travail.*

Nous adressons également nos sincères remerciements aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant de l'examiner.

Notre travail s'enrichit de leurs suggestions.

Nous tenons à remercier ***Mr. Mohamed Bensaadi,*** *pour le temps qu'il a consacré à la correction de ce mémoire*

Nous tenons à exprimer nos sincères remerciements à tous Des professeurs qui nous ont enseigné et qui par leurs compétences

Ils nous ont soutenus dans la poursuite de nos études.

Enfin, nous tenons également à remercier tout le monde Il a participé directement ou indirectement à la réalisation de ces travaux.

Dédicaces

Je remercie Dieu qui m'a donné naissance, santé et plénitude de mes sens.

En ce jour solennel, qui vient Couronner mes efforts, je profite De l'occasion pour exprimer toutes mes Gratitudes en vers ma Famille.

Pour les deux êtres, qui m'ont donné la vie, qui m'ont vu grandir, qui m'ont transmis tout Le savoir et qui étaient pour moi un cœur veillant pendant toute ma vie, les deux que je ne pourrais jamais assez remercier , À ma mère et mon père.

À mes chers frères et chères sœurs ,A tout les enfants de la famille ,

À ma collègue de travail Linda .

Mes intentions vont aux professeurs Mr Abdelouahab ATTIA et Mr. Mohamed Bensaadi, mes directeurs de recherche pour leur aide précieuse, leur patience et l'intérêt qu'ils portaient à mon travail.

A tous mes amis Kouki , Lamia ... ,une tendresse particulière pour vous, amour et respect.

Ibtissem

Dédicace

Je remercie Dieu qui m'a donné naissance, santé et plénitude de mes sens.

En ce jour solennel, qui vient Couronner mes efforts, je profite De l'occasion pour exprimer toutes mes Gratitudes en vers ma Famille.

Je dédie ce mémoire à mes chers parents qui ont été toujours à mes côtés et m'ont toujours soutenu tout au long de ces longues années d'études. En signe de reconnaissance, qu'ils trouvent ici, l'expression de ma profonde gratitude pour tout ce qu'ils ont consenti d'efforts et de moyens pour me voir réussir dans mes études.

A mes frère et leur femme

A ma petit fille Tasnim

A ma binôme ibtissem

A toute ma famille Et A toutes mes amies, . A tous les gens qui me connaissent.

A Mr. Mohamed Bensaadi, pour ses conseils avisés, orientations et qui m'ont permis de mener à bien cette étude

Lynda

Résumé

L'extraction d'ensembles d'éléments à haute utilité (HUIM) est un problème important dans l'exploration de données, consistant en des combinaisons d'éléments qui ont un impact significatif sur une mesure spécifique telle que les ventes ou les bénéfices. Face à la croissance exponentielle des données dans le monde du Big Data, il devient impératif de concevoir des algorithmes efficaces pour extraire rapidement et efficacement ces ensembles d'articles à forte utilité. Dans cette étude comparative, nous avons examiné les performances des algorithmes HUIM pour une représentation concise. Plusieurs grands ensembles de données ont été utilisés pour mener des expériences visant à évaluer les performances de MinFHM, CHUI-MinerMax, EFIM_Closed et CHUD concernant le temps d'exécution, la consommation de mémoire et le nombre d'éléments hautement utilitaires extraits. Selon les résultats, ces algorithmes se sont avérés capables d'extraire efficacement des ensembles d'éléments de grande utilité, avec des performances remarquables en termes de vitesse et d'utilisation optimale de la mémoire.

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

ملخص

يعد استخراج مجموعة العناصر ذات المنفعة العالية (HUIM) مشكلة مهمة في استخراج البيانات، حيث يتكون من مجموعات من العناصر التي تؤثر بشكل كبير على مقياس معين مثل المبيعات أو الأرباح. في مواجهة النمو الهائل للبيانات في عالم البيانات الضخمة، يصبح من الضروري تصميم خوارزميات فعالة لاستخراج هذه المجموعات من المقالات عالية الفائدة بسرعة وكفاءة. في هذه الدراسة المقارنة، قمنا بفحص أداء خوارزميات HUIM للتمثيل الموجز. تم استخدام العديد من مجموعات البيانات الكبيرة لإجراء تجارب لتقييم أداء MinFHM و CHUI-MinerMax و EFIM_Closed و CHUD فيما يتعلق بوقت التنفيذ واستهلاك الذاكرة وعدد العناصر ذات الفائدة العالية المستخرجة. ووفقاً للنتائج، فقد ثبت أن هذه الخوارزميات قادرة على استخراج مجموعات عناصر ذات فائدة عالية بكفاءة، مع أداء رائع من حيث السرعة والاستخدام الأمثل للذاكرة.

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

Abstract

High utility item set extraction (HUIM) is an important problem in data mining, consisting of combinations of items that significantly impact a specific metric such as sales or profits. Faced with the exponential growth of data in the world of Big Data, it becomes imperative to design efficient algorithms to extract these sets of high-utility articles quickly and efficiently. In this comparative study, we examined the performance of HUIM algorithms for concise representation. Several large datasets were used to conduct experiments to evaluate the performance of MinFHM, CHUI-MinerMax, EFIM_Closed and CHUD concerning execution time, memory consumption, and number of high utility items mined. According to the results, these algorithms were proven to be able to efficiently extract high utility item sets, with remarkable performance in terms of speed and optimal memory usage.

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

Table de matières

Liste des figures

Liste des Tableaux

List des Abréviations

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.

EFIM_Closed: Efficient high-utility Itemset Mining-closed.

FHM: Faster High-Utility Itemset Mining Algorithm.

SPMF: Sequential Pattern Mining Framework.

Introduction Générale

Le Data Mining, ou exploration de données, est un domaine multidisciplinaire qui vise à découvrir des modèles, des structures et des tendances significatives dans de vastes ensembles de données. Dans un monde où la quantité de données générées chaque jour augmente de façon exponentielle, le Data Mining joue un rôle essentiel dans l'extraction de connaissances exploitables à partir de ces données.

La discipline de l'exploration de données se présente comme une réponse à cette problématique. Aussi appelée découverte de connaissances à partir de données, elle se distingue comme un domaine interdisciplinaire qui associe des techniques de statistiques, d'apprentissage automatique et de gestion de bases de données. Son objectif est d'extraire des informations pertinentes à partir de vastes ensembles de données en identifiant des modèles, des relations et des tendances significatifs. Ces connaissances ainsi obtenues sont cruciales pour guider les prises de décision éclairées, anticiper les résultats futurs et approfondir la compréhension des phénomènes complexes.

Ce mémoire se concentre sur l'exploration des méthodes d'extraction de connaissances dans le domaine spécifique de la fouille des ensembles d'items à haute utilité (High Utility Itemset Mining ou HUIM). L'objectif principal est d'analyser en détail les algorithmes utilisés pour identifier ces ensembles d'items à haute utilité pour la représentation concise. Ces ensembles font référence à des combinaisons d'articles qui exercent un impact notable sur une mesure particulière, comme les ventes, les bénéfices, ou la satisfaction des clients.

Le mémoire se divise en trois chapitres essentiels :

Le Chapitre I, baptisé "Aperçu général sur la fouille de données", offre une perspective globale sur ce domaine. Il met en avant les techniques, l'évolution au sein des systèmes de bases de données, ainsi que les diverses tâches et applications associées à la fouille de données

Le Chapitre II, intitulé "High Utility Itemset Mining (HUIM)", se focalise sur l'extraction des ensembles d'items de haute utilité pour la représentation concise. Nous présenterons les algorithmes utilisés pour cette extraction.

Le Chapitre III, intitulé "Réalisation et interprétation", se concentrera sur la mise en œuvre et l'interprétation des résultats obtenus grâce aux algorithmes CHUD (Closed High Utility Discovery) et EFIM-Closed (Extended Frequent Itemset Mining) et les algorithme MinFHM et CHUI_MinerMax. Nous expliquons leurs concepts de base, leurs caractéristiques de base et leur fonctionnement.

Enfin, nous présenterons les outils de développement utilisés dans ce projet, tels que Java, Python, Visual Studio, ainsi que la bibliothèque SPMF pour l'extraction des ensembles d'items fréquents.

Chapitre I : Aperçu Général sur le Data Mining

Chapitre I : Aperçu Général sur le Data Mining

1 Introduction

Les méthodes de data mining sont couramment employées dans le domaine économique. Comme prédire certains indicateurs économiques, détecter des informations dissimulées, résoudre des problèmes ou trouver des solutions dans le domaine industriel, ainsi que dans les relations avec les clients en analysant leurs données et leurs comportements pour améliorer le rapport coût-efficacité de la relation avec les clients ou attirer de nouveaux clients.

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

2.1 Définition

KDD (Knowledge Discovery in Databases) est un processus qui implique l'extraction d'informations utiles, jusqu'alors inconnues et potentiellement précieuses à partir de grands ensembles de données. Le processus KDD est un processus itératif et nécessite plusieurs itérations des étapes ci-dessus pour extraire des connaissances précises à partir des données [1] .

2.2 Etapes du processus ECD

L'ECD est une méthode interactive et itérant qui cherche à extraire des informations à partir de données. Il comprend différentes étapes comme la préparation des données, l'exploration des données et l'analyse des résultats (Faux! Source de référence non trouvée.). L'utilisateur à la possibilité de Vous pouvez vous connecter au processus à tout moment, et il est adapté en fonction des résultats obtenus. Pour trouver des informations utiles, l'ECD utilise des méthodes statistiques, d'apprentissage automatique, de reconnaissance de motifs et de visualisation. Selon (Fayyad et al ,1996) [1].

2.2.1 La consolidation (Définition et compréhension du problème)

Les différentes données sur le domaine que l'on souhaite étudier sont regroupées et unifiées dans un cadre conceptuel commun lors de cette première étape. Certains traitements sont également nécessaires pour nettoyer les données lors de cette étape. En

général, une fois cette étape terminée, on reçoit un entrepôt de données qui servira de base de données pour la suite du processus [2].

2.2.2 Collecte et sélection des données

Au cours de cette étape, l'accent est mis sur la façon dont les données sont produites et recueillies. Les textes, les bases de données, les pages web, etc. peuvent être disponibles. Il arrive parfois qu'il soit nécessaire de prendre une copie d'un système d'information en cours d'exécution, puis de collecter les données provenant de sources éventuellement différentes. Par exemple, on prend une partie significative des données à partir de laquelle on élabore un modèle qui anticipe les données à venir [2].

2.2.3 Prétraitement

Il est essentiel de « préparer » les données collectées. Tout d'abord, il est nécessaire de les nettoyer car elles peuvent présenter différentes anomalies : des données peuvent être omises en raison d'erreurs de frappe ou d'erreurs du système lui-même. Dans ce cas, il est nécessaire de remplacer ces données ou de supprimer complètement leurs enregistrements. Il arrive parfois qu'il soit nécessaire de modifier les données afin d'uniformiser leur poids. La normalisation des données est un exemple de ces transformations, qui implique la projection des données dans un intervalle spécifique, par exemple. La réduction des données est également incluse dans le prétraitement, ce qui permet de diminuer le nombre d'attributs afin de faciliter les calculs et de représenter les données dans un format optimal pour l'exploration [2].

2.2.4 Construction du modèle (Data Mining)

C'est l'étape qui constitue véritablement le cœur du processus d'Extraction de Connaissances dans les données. En effet, l'espace de recherche est en général très vaste, et c'est ici qu'il faudra utiliser, lorsque cela est possible, les contraintes spécifiées par l'utilisateur lors de l'extraction. Si le jeu de données est volumineux, faire plusieurs passes sur les données peut être coûteux en temps. Des techniques telles que les réseaux de neurones, les arbres de décision, les réseaux bayésiens et clustering sont utilisées [2].

2.2.5 L'interprétation et l'évaluation

À cette étape, il est nécessaire d'examiner si les modèles et motifs générés par la phase de fouille de données correspondent à une connaissance nouvelle et pertinente pour le domaine considéré. Dans le cas de l'extraction d'ensembles d'items ou de motifs séquentiels, il est fréquent d'obtenir plusieurs milliers de motifs parmi lesquels seuls quelques-uns s'avèrent véritablement intéressants, ce qui demande une analyse approfondie et souvent longue. En revanche, pour les clusters ou les arbres de décision, les modèles générés sont généralement de taille plus réduite, ce qui peut simplifier leur analyse, mais leur interprétation et leur validation demeurent très complexes [2].

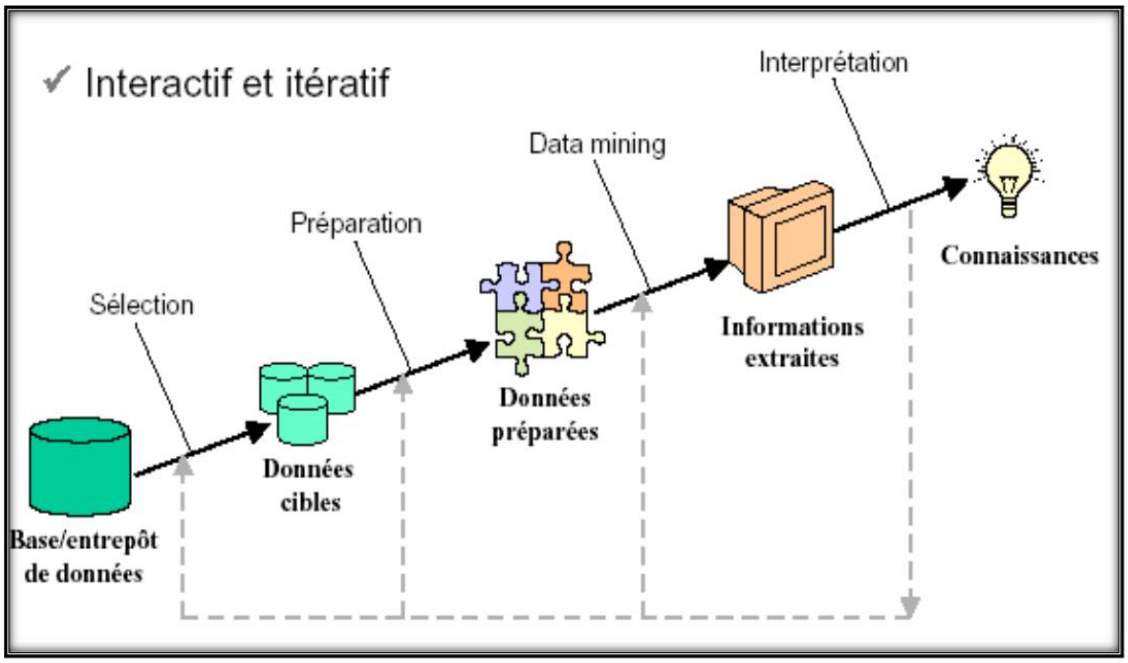

Figure 1: ETAPES DU PROCESSUS ECD [2]

3 Fouille de données (Data Mining)

Le Data-Mining ou « *fouille de données* » regroupe un ensemble des techniques offre la possibilité d'extraire, à partir d'un volume considérable de données brutes des connaissances originales jusqu'alors inconnues [3]. Au cours de ce premier chapitre, nous abordons la question de « La collecte de données ». Dans un premier temps, nous abordons la présentation du processus KDD et du data-mining, ainsi que ses diverses tâches et techniques. En réalité, on peut classer les techniques de data-mining en deux catégories : les techniques prédictives et les techniques descriptives. La fouille de motifs

(Pattern Mining) est une technique descriptive. Puis, on évoque les difficultés et les objectifs de la collecte de données. En terminant notre chapitre, nous exposons les objectifs de la fouille de données.

3.1 Définition

La fouille de données est un domaine qui est apparu avec l'explosion des quantités d'informations stockées, avec le progrès important des vitesses de traitement et des supports de stockage. La fouille de données vise à découvrir, dans les grandes quantités de données, les informations précieuses qui peuvent aider à comprendre les données ou à prédire le comportement des données futures. Le datamining utilise depuis sont apparition plusieurs outils de statistiques et d'intelligence artificielle pour atteindre ses objectifs. La fouille de données s'intègre dans le processus d'extraction des connaissances à partir des données ECD ou (KDD : Knowledge Discovery from Data en anglais). Ce domaine en pleine expansion est souvent appelé le data mining [4].

3.2 L'évolution de la technologie système de BD

L'exploration de données a attiré beaucoup d'attention dans l'industrie de l'information et dans la société dans son ensemble ces dernières années, en raison de la grande disponibilité d'énormes quantités de données et de la nécessité imminente de transformer ces données en informations et connaissances utiles.

Les informations et les connaissances acquises peuvent être utilisées pour des applications allant de l'analyse de marché, à la détection des fraudes et à la fidélisation des clients, en passant par le contrôle de la production et l'exploration scientifique.

L'exploration de données peut être considérée comme le résultat de l'évolution naturelle des technologies de l'information. L'industrie des systèmes de bases de données a connu une évolution dans le développement des fonctionnalités suivantes (Figure 2) : collecte de données et création de bases de données, gestion des données (y compris le stockage et la récupération des données, et gestion des bases de données).

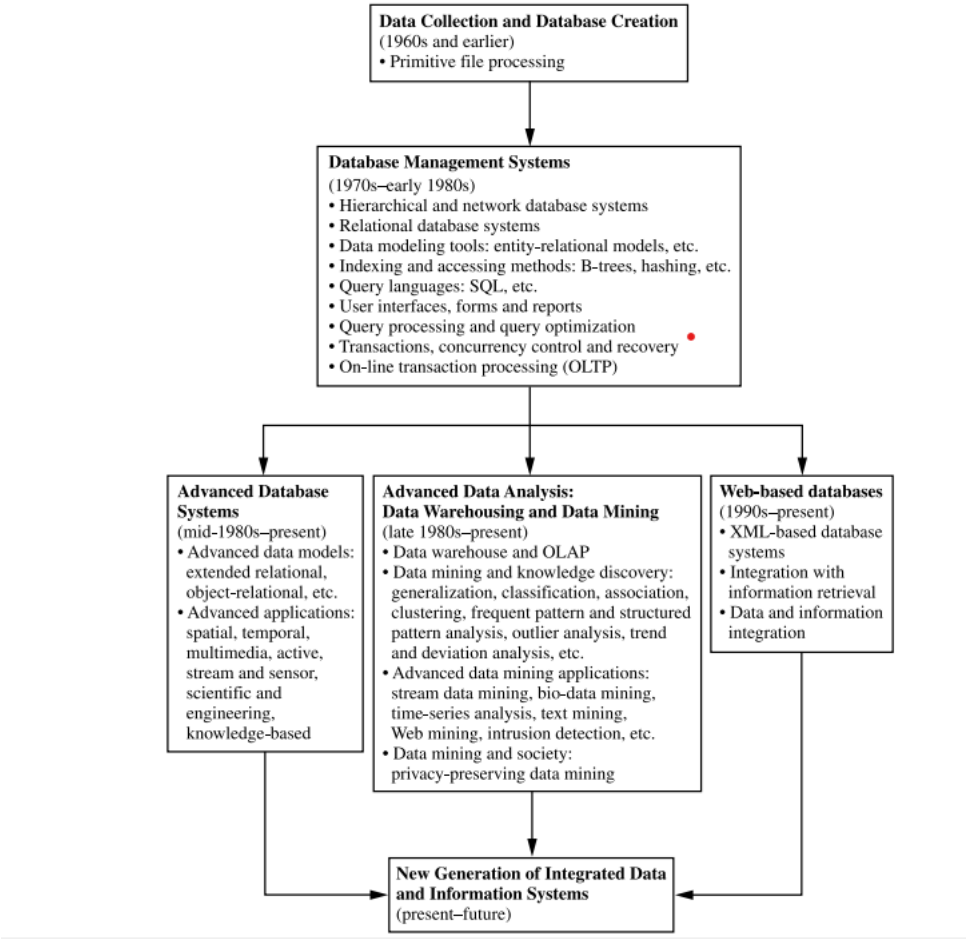

Figure 2:L'EVOLUTION DE LA TECHNOLOGIE SYSTEME DE BD [5]

L'expression "data mining" a vu le jour au début des années 1960 et avait alors une connotation négative. À cette époque, avec l'avènement des ordinateurs, de plus en plus de calculs étaient automatisés, ce qui rendait possible le traitement de données à grande échelle. Certains chercheurs ont commencé à explorer les données sans préjugés statistiques, en analysant des tableaux de données provenant d'enquêtes ou d'expériences. Malgré les critiques initiales des statisticiens officiels, ces chercheurs ont constaté que les résultats obtenus étaient prometteurs, ce qui les a encouragés à poursuivre cette approche empirique. L'analyse des données a donc gagné en popularité, malgré les réticences initiales.

En France, cette approche empirique a été associée à la diffusion de l'analyse de données auprès du grand public, notamment grâce à des promoteurs tels que Jean-Paul Benzecri, qui ont également dû faire face aux critiques des statisticiens traditionnels.

Dans les années 1980, des chercheurs en bases de données, comme Rakesh Agrawal, ont commencé à explorer l'exploitation des vastes ensembles de données, comme les tickets de caisse des grandes surfaces, convaincus de leur potentiel inexploité. Bien que le terme "database mining" ait été initialement utilisé, il a été remplacé par "data mining" lorsque le premier s'est avéré être une marque déposée. En mars 1989, Shapiro Piatetski a proposé le terme "knowledge discovery" lors d'un atelier sur la découverte de connaissances dans les bases de données. Actuellement, les termes "data mining" et "knowledge discovery in Databases" (KDD ou ECD en français) sont largement utilisés de manière interchangeable, avec "data mining" étant le plus couramment employé dans la littérature.

La communauté du "data mining" a organisé sa première conférence en 1995, après plusieurs ateliers sur le KDD entre 1989 et 1994. En 1998, sous l'égide de l'ACM, le chapitre spécial ACMSIGKDD a été créé pour réunir la communauté internationale du KDD. La première revue spécialisée dans ce domaine, "Data Mining and Knowledge Discovery Journal", a été lancée en 1997 par "Kluwers" [5].

3.3 Les tâches de Data Mining

Les tâches de fouille de données comme la classification, le clustering et les règles d’association peuvent être appliqués dans plusieurs différents domaines. Ces tâches utilisent d’une manière générale deux stratégies, à savoir l'apprentissage supervisé et l'apprentissage non supervisé. Dans l'apprentissage supervisé, un ensemble de données d’apprentissage est utilisé pour déterminer les paramètres du modèle, alors que dans l'apprentissage non supervisé, on n’utilise pas un ensemble de données d’apprentissage, et le but du processus est de construire un modèle qui décrit des régularités intéressantes dans les données [6].

3.3.1 Classification

L'objectif est de rassembler des éléments dans des catégories préétablies en fonction de leurs propriétés similaires.

Supposons qu'un décideur veuille classer ses employés par tranches de revenu, ou n'importe quelle autre caractéristique associée à cette personne, comme l'âge, le sexe et la profession. Cette tâche est une tâche de classification [2].

3.3.2 Estimation

L'estimation est similaire à la classification, sauf que la variable cible est numérique plutôt que catégorique. Les modèles sont construits en utilisant des données, qui fournissent la valeur de la variable cible, ainsi que les « prédicteurs » [2].

3.3.3 Prédiction

La prédiction partage des similarités avec la classification et l'estimation, mais se distingue par le fait que les résultats sont orientés vers l'avenir Par exemple, dans le domaine du marketing, une tâche de prédiction pourrait consister à anticiper le prix d'un stock dans trois mois [2].

3.3.4 Regroupement par similarité (ou clustering)

Le Clustering désigne le regroupement des données, des observations ou des cas dans des classes d'objets similaires. Un cluster maximise la similarité des objets de du même cluster et minimise la similarité des objets de cluster différents. En effet, il n'y a pas de variable cible pour le clustering. La tâche de clustering ne cherche pas à classer, estimer, ou prédire la valeur d'une variable cible. Mais plutôt à segmenter l'ensemble des données en sous-groupes relativement homogènes à l'aide de mesures de distances [2].

3.3.5 Analyse des clusters

Une fois que les éléments ont été regroupés, cette tâche consiste à analyser les caractéristiques communes des éléments au sein de chaque groupe. Par exemple, analyser les caractéristiques démographiques des différents segments de clients [1].

3.3.6 Description

En data-mining, la méthode de description est une méthode analytique employée afin d'extraire des informations pertinentes à partir de vastes ensembles de données ou de bases de données complexes. Son objectif est de mettre en évidence des modèles et des structures dissimulés dans les données, ce qui permet de produire des descriptions compréhensibles et pratiques des caractéristiques déterminées [2].

3.4 Techniques du Data Mining

Les principaux algorithmes de la fouille de données sont classés en deux ensembles de techniques :

3.4.1 Les Techniques Descriptives

a. **La visualisation des données :** Ces techniques descriptives permettent de représenter graphiquement les données de manière compréhensible et informative, facilitant ainsi la détection de schémas, de tendances ou d'anomalies [3].

b. **Le clustering :** Cette technique regroupe les données similaires en fonction de leurs similarités ou de leurs distances. Elle permet d'identifier des groupes ou des segments homogènes au sein des données, ce qui facilite la compréhension des relations entre les instances de ces données [3].

c. **Les règles d'association :** les règles d'associations consistent à découvrir des relations intéressantes et fréquentes entre les éléments d'un ensemble de données. Elles permettent d'identifier des combinaisons d'items qui se produisent souvent ensemble [3].

d. **Pattern Mining :** le *pattern mining* vise à révéler des modèles intéressants, utiles ou inattendus dans les bases de données. De nombreux algorithmes d'exploration de motifs ont été conçus pour trouver divers types de motifs tels que les motifs fréquents, les règles d'association et les motifs séquentiels fréquents et les motifs profitables [3].

3.4.2 Pattern Mining (La Fouille de Motifs ou Recherche des Patterns)

Pattern mining est un sous-domaine clé de data-mining, son but et de développer des algorithmes pour découvrir des modèles intéressants dans les bases de données. Les modèles découverts peuvent être utilisés pour aider à comprendre les données et également pour effectuer d'autres tâches comme la classification et la prédiction. La recherche des patterns peut être utilisée pour des fins commerciales pour analyser le panier de la ménagère puis développer des stratégies marketing sur les achats alimentaires des consommateurs. A partir de l'analyse de nombreuses transactions [3].

3.4.3 Les Techniques Prédictives

Ces techniques permettant d'extrapoler et de prédire de nouvelles informations et connaissances à partir des données [3]. Il existe plusieurs techniques prédictives tel que :

a. **La Classification :** L'objectif de la classification supervisée est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets [3].

b. **La Régression :** le but de la régression est d'estimer une valeur (numérique) de sortie à partir des valeurs d'un ensemble de caractéristiques en entrée. Par exemple, estimer le prix d'une maison en se basant sur sa surface, nombre des étages, son emplacement, etc. [3].

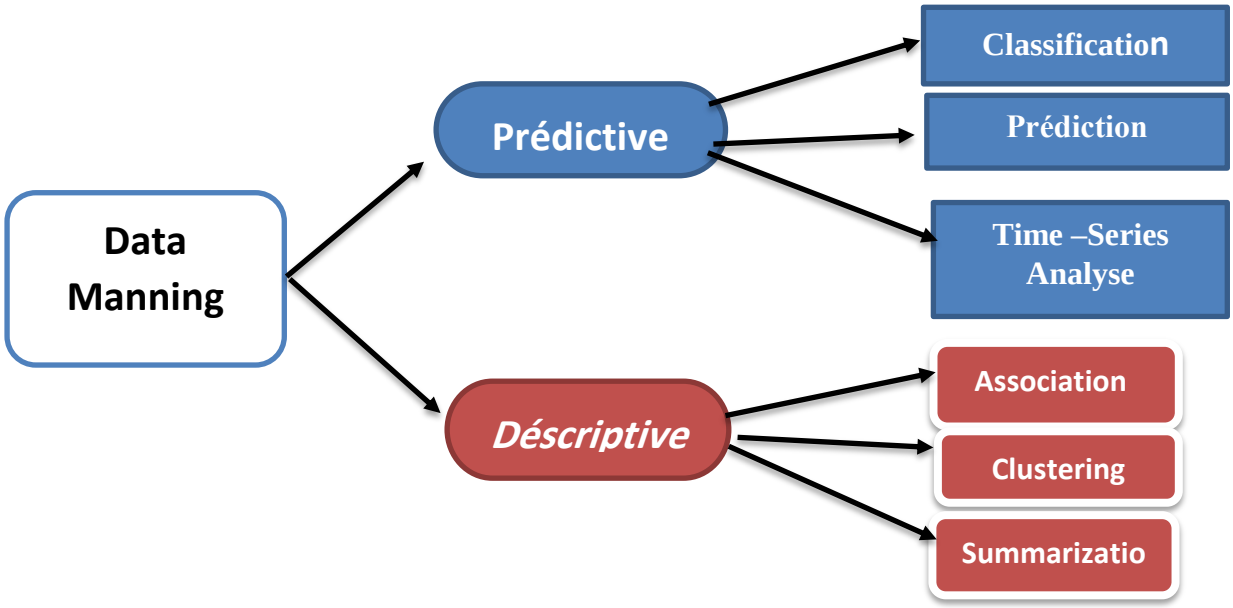

Figure 3: Les Technique De Data-mining [3]

En pratique, il n'existe pas de méthode de fouille de données universellement supérieure. Les compromis doivent être faits en fonction des besoins spécifiques et des fonctionnalités connues des outils disponibles. Ainsi, le choix de la méthode appropriée dépend de divers facteurs, tels que :

• La nature et la disponibilité des données.
• Les savoirs à disposition.
• L'environnement de l'entreprise.
• Les différentes tâches.
• La finalité du modèle construit.

• La difficulté de l'élaboration du modèle.

• La difficulté d'utilisation du modèle et ses résultats.

3.5 Domaines d'application du Data Mining

Tableau 1: Domaines d'application du data mining

Secteurs d'activité	Exemples d'applications
Bancaire	• Évaluer comment les clients réagissent aux variations des taux d'intérêt. • Repérer la clientèle la plus fidèle.
La détection de fraude	• Recherche de fraudes avec des cartes de crédit. • Repérage de demandes erronées de remboursement médical.
Assurances	• Éliminer la fraude avec les cartes de crédit. • Repérage de demandes de remboursement médical erroné
La télécommunication	• Prédiction des modèles d'appels téléphoniques. • Gestion des ressources et du trafic réseau.

4 Conclusion

Le Data Mining sont des technologies indispensables dans le nouveau monde axé sur la connaissance, où nous créons des modèles à partir de données et de bases de données pour appréhender et explorer notre monde. En collectant des informations, il est possible d'améliorer nos activités, notre gouvernement et notre bien-être. Les outils appropriés permettent à tout le monde de commencer à explorer cette nouvelle technologie, dans le but de devenir un expert en extraction de données.

Chapitre II: High Utility Itemset Mining HUIM

Chapitre II: High Utility Itemset Mining HUIM

1 Introduction

La recherche des algorithmes d'extraction d'itemsets a commencé dans les années 1990, avec des algorithmes pour découvrir des modèles fréquents dans les bases de données, pour le but de faciliter la compréhension des humains et soutenir la prise de décision dans de nombreux domaines surtout le marketing.

Dans ce chapitre, on va parler premièrement sur l'extraction d'ensembles d'items fréquents(FIM), L'une des tâches fondamentales de l'exploration de données est l'extraction des itemsets fréquents, qui consiste à identifier des itemsets qui Co-occurrent fréquemment dans une base de données transactionnelle. De plus, nous discuterons de l'algorithme Apriori, qui est essentielle dans ce domaine. Cependant, malgré son efficacité, l'extraction d'itemsets fréquents présente des limitations inhérentes qui ont suscité l'exploration d'approches alternatives.

L'extraction de l'ensemble des éléments à haute utilité est l'une des méthodes alternatives pour extraire l'ensemble des éléments fréquents. Où nous discuterons du problème de l'ensemble des éléments à haute utilité et de leur limitations .en dernier nous avons faire une présentation sur des Algorithmes de Haute utilité pour représentation concise dans les bases de données transactionnelles.

2 Frequent Itemset Mining(FIM)

L'extraction d'ensembles d'items fréquents (FIM) implique d'extraire tous les ensembles d'items fréquents présents dans un jeu de données, avec une fréquence d'occurrence supérieure à un seuil spécifié. Cette tâche a été proposée au début des années 1990 dans le cadre de l'analyse des paniers d'achat pour découvrir les items qui co-occurrent fréquemment. Elle a été initialement nommée "minage d'ensembles d'articles larges" [1].

2.1 Concept fondamentaux

Afin de mieux comprendre le FIM, nous allons définir quelques termes importants :

2.1.1 Transaction

Soit T= {T1, T2, T3, …, T_n}, $T_i \subset D$. On appelle T_i un ensemble de lignes contenant les occurrences de la base de données D. Tous les T_i sont appelés des transactions. Dans l'exemple familier du panier de la ménagère, les transactions sont les tickets de caisse

(c.à.d. les achats réalisés par les clients). [7]

2.1.2 Item

Un item est un élément ou attribut dans un base de données transactionnelle D. par exemple dans la base de données de vente au détail, les items peut représenter des produit spécifiques [1].

2.1.3 Itemset

Un itemset est une collection d'un ou plusieurs items. Cela peut être un seul item (itemset singleton) ou d'une combinaison de plusieurs items (itemset multi-items). Par exemple, {A},{B}et{A,B}sont des itemsets [1] .

2.1.4 Frequent itemset

Un Itemset est fréquent si et seulement si son support est supérieur ou égal à un support minimum défini par l'utilisateur [1].

2.1.5 Mesure de support

Le support (fréquence) d'un ensemble d'éléments X dans une base de données de transactions D est noté sup(X) et défini comme $sup(X) = |\{T|X \subset T \wedge T \in D\}|$, c'est-à-dire le nombre de transactions contenant X .

Par exemple, la prise en charge de l'ensemble d'itemset {a,c} dans la base de données du tableau 3 est 3, puisque cet ensemble d'éléments apparaît dans trois transactions (T0, T2 et T3). Cette définition de la mesure de soutien est appelée soutien relatif. Une autre définition équivalente consiste à exprimer le support en pourcentage du nombre total de transactions (appelé support absolu) [8].

2.1.6 Bases de données transactionnelles

Les bases de données transactionnelles se composent d'une série de transactions D, chacune représentant un ensemble d'item. les transaction peuvent être comme des enregistrements ou des instances dans la base de données [3].

Exemple **:** le tableau 2 représente un exemple d'une base de données transactionnelle D, cette ase contient 5 transaction et des itemsets tel que :

a : Amoxiciline / b : Bedilixe / c : Clamoxyl / d : Doliprane/ e : Effralgan

Tableau 2: Base de données transactionnelle.[1]

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

La compréhension de ces définitions est essentielle pour comprendre les concepts fondamentaux de l'extraction des ensembles d'items fréquents. Les bases requises sont fournies pour identifier et déterminer les itemsets fréquents dans les transactions, en utilisant des mesures de support pour déterminer leur fréquence. En sachant ce que signifient ces définitions, il est possible d'approfondir la compréhension des algorithmes et des techniques spécifiques employés pour extraire de manière efficace les ensembles d'éléments fréquents à partir des bases de données de transaction.

2.2 Problème de FIM (Frequent itemset mining)

L'extraction des itemsets fréquents, aussi appelée "mining itemset fréquent", représente un défi majeur dans le domaine de la fouille de données. Une difficulté majeure lorsqu'on extrait des itemsets fréquents à partir de données est de gérer le grand nombre d'itemsets générés, surtout lorsque le seuil de support minimum est fixé à un niveau bas. Effectivement, un itemset fréquent comprend un nombre croissant de sous-ensembles fréquents plus courts, ce qui peut entraîner une multiplication des motifs et compliquer la tâche de recherche et d'analyse [9][10][11].

Afin de trouver une solution à ce problème, des algorithmes comme l'algorithme Apriori ont été développés. L'algorithme Apriori utilise une approche itérative pour trouver les itemsets fréquents en examinant les itemsets de k pour trouver les itemsets de (k + 1). Le principe d'anti-monotonie repose sur le fait que tout sous-ensemble non vide d'un itemset fréquent est également fréquent. Cela aide à diminuer la recherche en éliminant plus

rapidement les itemsets non fréquents, ce qui favorise une extraction plus efficace des itemsets fréquents [10][11].

En bref, le problème de l'extraction des itemsets fréquents dans le cadre du "mining de itemsets fréquents" réside dans la gestion de la multiplication des motifs en raison de la nature combinatoire des itemsets fréquents. On a développé des algorithmes tels que Apriori afin de relever ce défi en utilisant des stratégies efficaces pour repérer les itemsets fréquents tout en réduisant la redondance et en améliorant l'efficacité de l'extraction des motifs fréquents.

2.3 Limitations de FIM (Frequent itemset mining)

L'un des principaux problèmes du FIM est qu'il traite tous les éléments (items) de la même importance(utilité), qu'ils soient très utiles ou non [8]. Le fait que les quantités d'éléments soient ignorées dans FIM est un autre problème. Par exemple, il est considéré comme identique si un acheteur achète cinq ou dix pains. Ainsi, l'exploration de modèles fréquents est capable de découvrir un grand nombre de modèles fréquents sans intérêt [8].

En prenant un exemple concret, l'article {pain, lait} est un article courant (largement acheté par les clients), mais il n'est pas intéressant pour un commerçant car il ne génère pas beaucoup de profit (utilité). Les algorithmes d'exploration de modèles fréquents peuvent manquer les modèles rares qui génèrent un profit élevé [8].

2.4 Algorithmes d'extraction des items fréquents

Pour trouver de manière efficace les ensembles d'items fréquents, il est important de développer des algorithmes qui évitent d'explorer tous les ensembles d'items possibles dans l'ensemble de recherche et qui traitent efficacement chaque ensemble d'items Proposer différents algorithmes efficaces pour extraire des ensembles d'éléments fréquents, tels que de renommés algorithmes tels que : Apriori, FP_Growth, Eclat. Malgré leur entrée et leur sortie identiques, ces algorithmes se distinguent par les stratégies et les structures de données qu'ils utilisent pour détecter efficacement les ensembles d'éléments fréquents [1].

En particulier, les algorithmes d'extraction des ensembles d'items fréquents présentent des différences dans les domaines suivants.

2.4.1 Stratégie de recherche

il est possible d'utiliser soit une approche de recherche approfondie (depth-first search) ou une approche de recherche en largeur (Breadth-first search) [1].

2.4.2 Représentation de la base de données

utiliser soit une représentation horizontale, ou une représentation verticale de la base de données [1].

2.4.3 Génération des ensembles d'items

Ils diffèrent dans la façon dont ils génèrent ou définir les prochains ensembles d'items à explorer dans l'espace de recherche [1].

2.4.4 Evaluation du support

Ils recourent à diverses approches pour calculer le support des ensembles d'items afin de déterminer s'ils satisfont à la contrainte de support minimale.

Tableau 3 : Résume des algorithmes FIM

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

2.5 Recherche en largeur (Breadth-First Search)

En utilisant un algorithme de recherche en largeur (également appelé algorithme de niveau), comme l'algorithme Apriori, on suppose qu'il existe m éléments dans une base de données. Cet algorithme examine l'espace de recherche des ensembles d'items en commençant par les 1-ensembles d'items, puis les 2-ensembles, les 3-ensembles, et ainsi de suite jusqu'aux m-ensembles. Par exemple la (Figure 4) illustre l'espace de recherche de tous les ensembles d'items possibles pour l'exemple donné. Cette figure présente l'espace de recherche sous la forme d'un diagramme de Hasse.

Dans un algorithme de recherche en largeur, les 1-ensembles d'éléments {a}, {b}, {c}, {d} et {e} seront d'abord. Il produira ensuite les deux ensembles d'items (a, b}, a, c}, a, d), puis les trois ensembles d'items, et ainsi de suite, jusqu'à ce qu'il produise l'ensemble {a, b, c, d, e} qui regroupe tous les items de la base de données [1].

2.6 Recherche en profondeur (Depth-First Search)

Les algorithmes de recherche approfondie tels que FPGrowth, Eclat débutent par chaque ensemble d'items de taille 1, puis essaient de manière récursive d'ajouter des items à l'ensemble d'items actuel afin de créer des ensembles d'items plus grands. Par exemple, dans l'exemple en cours, un algorithme de recherche en profondeur typique explorerait les ensembles d'items dans l'ordre suivant : {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} .

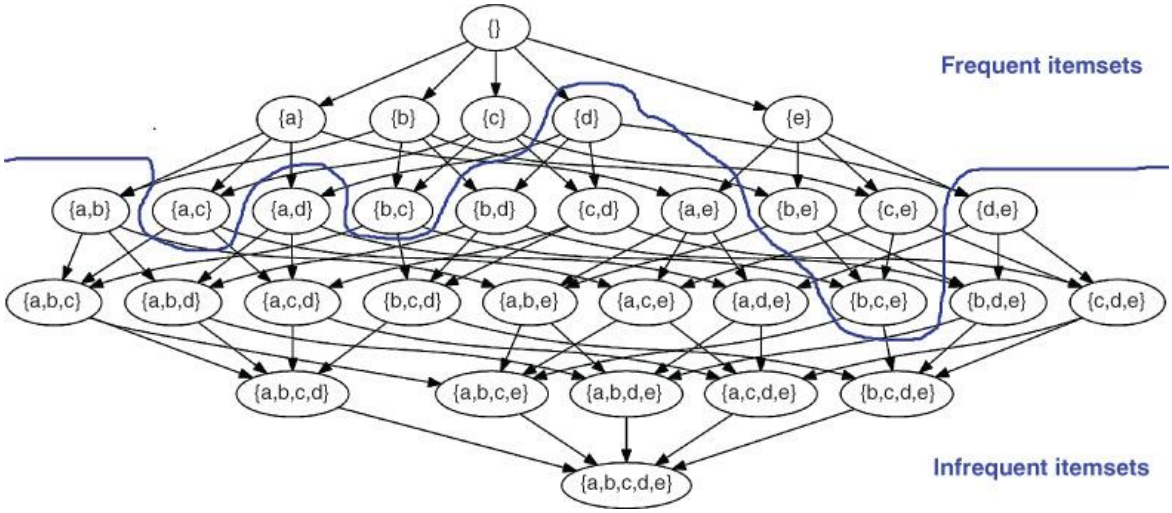

Figure 4: L'Espace de recherche pour I= {a, b, c, d, e}.

2.7 Aperçu de l'algorithme Apriori

L'algorithme Apriori, conçu en 1994 par Agrawal et Srikant, est un algorithme d'exploration de données permettant de découvrir des propriétés qui reviennent fréquemment dans un ensemble de données et d'en déduire une catégorisation sous forme de règles d'association

L'algorithme apriori est basé sur le principe que si un ensemble d'éléments est fréquent. alors tous ses sous-ensembles les sont également. Voici un aperçu de la procédure de l'algorithme Apriori :

Algorithm 1 L'algorithme Apriori

Calculer L_1
$k \leftarrow 2$
TANTQUE $L_{k-1} <> \phi$ **FAIRE**
  $C_k \leftarrow apriori - gen(L_{k-1})$
  TANTQUE $t \in D$ **FAIRE**
    $C_t = sousensemble(C_k, t)$
    TANTQUE $c \in C_t$ **FAIRE**
      c.count++
    FIN TANTQUE
  FIN TANTQUE
  $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$
  $k \leftarrow k + 1$
FIN TANTQUE
RETOURNER $\bigcup_k L_k$

Algorithme 1 : Algorithme Apriori

Plusieurs étapes sont incluses dans l'algorithme Apriori. Tout d'abord, il analyse la base de données de transactions pour calculer le support de chaque item individuel et identifie les items fréquents de taille 1. Ensuite, il génère de manière itérative les items de candidats en joignant les items fréquents de taille (k-1) à partir des itérations précédentes. Cependant, avant de calculer le support de ces candidats, l'algorithme utilise la propriété d'Apriori pour éliminer les candidats non fréquents. Cela signifie que si un sous-ensemble d'un candidat n'est pas fréquent, alors ce candidat lui-même ne peut pas être fréquent. Cela réduit l'espace de recherche en éliminant les candidats inutiles. Ensuite, l'algorithme calcule le support de chaque candidat restant en analysant à nouveau la base de données de transactions. Les candidats avec un support supérieur ou égal au seuil minimum sont considérés comme des itemsets fréquents. L'algorithme continue ce processus itératif jusqu'à ce qu'il n'y ait plus itemsets fréquents trouvés. Finalement, il retourne l'union de tous les itemsets fréquents découverts [1].

3 High Utility Itemset Mining HUIM

L'extraction des ensembles d'items à utilité élevée élargit le concept de l'extraction des ensembles d'items fréquents en surmontant la restriction de se baser uniquement sur la fréquence des items. Cette méthode vise à découvrir des motifs ayant une grande valeur pour les utilisateurs en prenant en considération des critères comme le profit, l'utilité ou d'autres mesures spécifiques.

Le but de l'extraction des ensembles d'items à utilité haut est de trouver des ensembles d'items qui maximisent une fonction d'utilité prédéfinie. Cette fonction d'utilité peut être définie en fonction de différents facteurs, tels que le coût, le profit, la satisfaction du client, le temps passé, ou d'autres considérations spécifiques au domaine d'application.

Il est particulièrement pertinent de trouver des ensembles d'items à haut utilité dans des domaines tels que le marketing, où l'objectif est de repérer les ensembles d'items qui peuvent générer un profit élevé, ou dans la personnalisation de services, où l'objectif est de recommander des ensembles d'items qui répondent aux besoins spécifiques des utilisateurs.

Il est essentiel de réaliser une modélisation adéquate des données, où chaque item est lié à une mesure d'utilité ou de profit. Ensuite, les algorithmes d'extraction des ensembles d'items à haut utilité utilisent ces informations pour repérer les ensembles d'éléments les plus pertinents et les plus rentables.

La recherche en fouille de données s'est intéressée de plus en plus à l'extraction des ensembles d'items à utilité haut, et de nombreux algorithmes efficaces ont été développés pour résoudre ce problème. Pour accélérer le processus d'extraction, ces algorithmes utilisent des méthodes comme la création de candidats, la prune basée sur des seuils d'utilité et des structures de données spécialisées.

Cette approche offre de nouvelles perspectives pour découvrir des motifs intéressants et importants dans les données, allant au-delà de la simple fréquence des items. Elle permet aux utilisateurs de découvrir des ensembles d'items qui ont un impact significatif sur leurs objectifs et de prendre des décisions plus éclairées et plus rentables.

Dans la suite de ce chapitre, nous examinerons en détail les aspects de l'extraction des ensembles d'éléments à grande utilité, incluant les diverses techniques et algorithmes employés, tels que la création de candidats, la prune basée sur des seuils d'utilité et l'utilisation de structures de données spécialisées pour accélérer le processus d'utilisation.

Chapitre II: High Utility Itemset Mining HUIM

3.1 Concepts fondamentaux

3.1.1 Base de données quantitative des transactions

Une base de données de transaction quantitatives D L'expression D est définie comme suit : D = {T0, T1,..., Tn}, où chaque transaction Tq est un Itemset (Tq ⊆ I) avec un identifiant unique q (TID). Chaque élément i de l'ensemble I possède une utilité externe p(i)(Table 5) et une utilité interne q(i, Tc)(Table 4) dans la transaction Tc [3].

Tableau 4: Base de données quantitative transactionnelle (interne).

Tid	Transaction
T1	(a, 1), (b, 5), (c, 1), (d, 3), (e, 1)
T2	(b, 4), (c, 3), (d, 3), (e, 1)
T3	(a, 1), (c, 1), (d, 1)
T4	(a, 2), (c, 6), (e, 2)
T5	(b, 2), (c, 2), (e, 1)

Tableau 5: Valeure d'étulité (externe) .

Item	a	b	c	d	e
profit	5	2	1	2	3

3.1.2 Utility d'un item

L'utilité d'un item i dans une transaction Tc est notée U (i, Tc) et définie comme

p(i) × q (i, Tc). Ou p(i) est l'utilité externe de l'item i, et q (i,T_c) est la quantité de i dans la transaction T_c.

Exemple : L'utilité de l'item **a** dans **T4** est U (a, T4)=10.

a. Utilité d'un itemset dans une transaction

L'utilité d'un itemset X dans une transaction T_c est définie comme suit :

$$U(X, T_c) = \sum_{i \in X} U(X, T_c) \text{ ,si } X \subset T_c.$$

Exemple : L'utilité de l'itemset **{a,b}** dans la transaction T1 est :

U({a,b}, T1) = U(a, T1)+ U(b, T1)= 15.

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

L'utilité d'un itemset X dans une base de données D est noté U (X , D) et définie comme suit : $U(X, D) = \sum_{T \in D} U(X, T)$,

Exemple : L'utilité de l'itemset **{a,c}** dans une la base de données est :

u({a, c}) = u(a)+u(c) = u(a,T0)+ u(a,T2) + u(a,T3) + u(c,T0) + u(c,T2) + u(c,T3) =28.

3.1.3 Itemset à utilité haut

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 ; (c'est-à-dire u(X) ≥ minutil). Sinon, X est un itemset de faible utilité.

Exemple : Minutil=25.

- U ({a, c}) = 28 {a, c} est un itemset à utilité haut.
- U ({a, b}) = 15 {a, b} est un itemset à faible utilité.

3.2 Problème de HUIM

Le problème de l'extraction d'ensembles d'éléments à haute utilité est assez intéressant pour les raisons suivantes :

- Premièrement, il peut être plus intéressant d'un point de vue pratique de découvrir des ensembles d'articles qui génèrent un profit élevé dans les transactions clients que ceux qui sont achetés fréquemment.

- Deuxièmement, du point de vue de la recherche, le problème de l'extraction d'ensembles d'éléments à haute utilité est plus difficile. Dans l'exploration fréquente d'ensembles d'éléments, il existe une propriété bien connue de la fréquence (support) des ensembles d'éléments qui indique que, étant donné un ensemble d'éléments, tous ses sur-ensembles doivent avoir un support inférieur ou égal. Ceci est souvent appelé la « propriété Apriori » ou propriété « anti-monotonicité » et est très puissant pour élaguer l'espace de recherche car si un ensemble d'éléments est peu fréquent alors nous savons que tous ses sur-ensembles sont également peu fréquents et peuvent être élagués. Dans

l'exploitation minière d'éléments à haute utilité, une telle propriété n'existe pas. Ainsi, étant donné un ensemble d'éléments, l'utilité de ses sur-ensembles peut être supérieure, inférieure ou identique.

Il existe plusieurs algorithmes qui ont été proposés pour découvrir des itemsets à haute utilités tels que : HUI-Miner, FHM et Two-phase.

3.3 Difficulté HUIM Algorithmes

Propriétés clés du problème de l'extraction d'ensembles d'itemset à haute utilité Pour une base de données quantitative et un seuil d'utilité minimum donnés, le problème de l'extraction d'ensembles d'itemset à haute utilité a toujours une solution unique. Il s'agit d'énumérer tous les modèles qui ont une utilité supérieure ou égale au seuil d'utilité minimum spécifié par l'utilisateur. Le problème de l'extraction d'ensembles d'itemset à haute utilité est difficile pour deux raisons principales :

a) La première raison est que le nombre d'items à considérer peut être très important pour trouver ceux qui ont une grande utilité. Généralement, si une base de données contient m éléments distincts, il y a 2^m-1 ensembles d'itemsets possibles qui peuvent être générés.

Par exemple : si I = {a,b,c}, les éléments possibles sont {a} ; {b}; {c}; {a,b}; {a,c}; {b,c}; et {a,b,c}. Ainsi, il y a 2^3-1 =7 ensembles d'itemsets, qui peuvent être formés avec I = {a, b, c}.

Une approche naïve pour résoudre le problème de l'exploration des ensembles d'itemsets à haute utilité consiste à compter les utilités de tous les ensembles d'itemsets possibles en analysant la base de données, pour ensuite conserver les ensembles d'éléments à haute utilité. Bien que cette approche produise le résultat correct, elle est inefficace. La raison est que le nombre d'ensembles d'itemsets possibles peut être très grand. Par exemple, si un magasin de détail a 10 000 articles sur ses étagères (m = 10 000), les utilités de $2^{10,000} - 1$ itemsets d'articles possibles doivent être calculées, ce qui est ingérable avec l'approche naïve. Il convient de noter que le problème de l'exploration d'ensembles d'itemsets à haute utilité peut être très difficile, même pour de petites bases de données.

b) Une deuxième raison pour laquelle le problème de l'exploration des ensembles d'itemsets à haute utilité est difficile est que les itemsets à haute utilité sont souvent dispersés dans l'espace de recherche. Ainsi, de nombreux itemsets doit être pris en compte par un algorithme avant de pouvoir trouver les itemsets à haute utilité réels. Cette difficulté

est présente par ce que contrairement au problème de FIM ou le support d'itemset est toujours supérieur ou égal au support de l'un de ses sur-ensembles (super-sets), dans le cas de HUIM, l'utilité d'un itemset peut être supérieure, inférieure ou égale à l'utilité de n'importe lequel de ses sur-ensembles/sous-ensembles. En d'autres termes, l'utilité a une propriété est qu'elle n'est ni monotone ni anti-monotone.

- **Propriété 1 (La mesure d'utilité n'est ni monotone ni anti-monotone)**

Soient deux ensembles itemset X et Y tels que $X \subset Y$. La relation entre les utilités de X et Y peut être $u(X) < u(Y)$, $u(X) > u(Y)$ ou $u(X) = u(Y)$ [3].

- **Propriété 2 (Le TWU est une borne supérieure monotone sur la mesure d'utilité)**

Soit un itemset X. Le TWU de X est supérieur ou égal à son utilité ($TWU(X) \geq u(X)$). De plus, le TWU de X est supérieur ou égal à l'utilité de ses sur-ensembles ou d'autre façon ($TWU(X) \geq u(Y)$ pour tout $Y \supset X$) [1].

- **Propriété 3 (Élagage de l'espace de recherche en utilisant le TWU)**

Pour tout itemset X, si $TWU(X) < minutil$, alors X est un itemset de faible utilité ainsi que tous ses sur-ensembles. Les algorithmes basés sur deux phases utilisent la propriété 2 comme propriété principale pour élaguer l'espace de recherche [3].

3.4 Algorithmes itemset à haute utilité

Certains algorithmes populaires d'extraction d'ensembles d'items à utilité haut présente dans le tableau 6 avec une comparaison de leurs caractéristiques en termes de type de recherche (recherche en largeur ou en profondeur), nombre de phases (une ou deux), représentation de la base de données (horizontale ou verticale) [1].

Tableau 6 : Les algorithmes nécessaires d'extraction d'itemsets à haute utilité.

Algorithme	**Description**
Two-Phase	Adopte une représentation Horizontal et utilise une recherche en largeur, nombre de phases est une phase , c'est un extension de l'algorithme Apriori .
HUI-Miner	Adopte une représentation verticale et utilise une recherche en profondeur pour découvrir les motifs et calculer leur utilité sans analyses coûteuses de la base de données, nombre de phases est une phase, algorithme d'extraction d'ensembles fréquents le plus similaire est Eclat.
FHM	Adopte une représentation verticale et utilise une recherche en profondeur, nombre de phases est une phase, algorithme d'extraction d'ensembles fréquents le plus similaire est Eclat.
EFIM	Adopte une représentation Horizontal et utilise une recherche en profondeur, nombre de phases est une phase, c'est une extension de l'algorithme Eclat.

4 Limitation des algorithmes HUIM

Le problème traditionnel rencontré dans l'extraction de l'ensemble des HUIM est la taille importante de l'ensemble résultant Ce problème a conduit les chercheurs à étudier l'extraction de représentations concises ou compactes pour l'ensemble des HUIM.

Les représentations compactes sont des ensembles de petite taille par rapport à l'ensemble des HUIM, mais elles conservent l'ensemble des informations.

5 Représentations concises

Nous examinerons dans cette partie l'idée de la représentation concise qui permet de réduire et de comprimer la taille des ensembles d'éléments à grande utilité tout en préservant leur utilité et leur signification. La représentations concises présente de nombreux avantages , tels que la réduction de la redondance, l'amélioration de l'efficacité des algorithmes d'extraction et une interprétation plus précise des motifs découverts.

Chapitre II: High Utility Itemset Mining HUIM

5.1 Concepts fondamentaux

5.1.1 Closed HUI (CHUI)

Un itemset X est un CHUI si et seulement s'il existe aucun superset propre de X qui soit un high-utility itemset(HUI) et qui ait même support (fréquence) [13] [14].

En d'autres termes, il ne devrait pas exister de HUI Y plus grand avec la même valeur d'utilité que X.

5.1.2 Maximal HUI (MaxHUI)

Un HUI X est considéré comme un HUI maximal s'il n'existe aucun autre HUI Y tel que X soit un sous-ensemble propre de Y ($X \subset Y$). En d'autres termes, il ne devrait pas exister de HUI Y plus grand qui contient X.

5.1.3 Générateur d'ensembles d'items à utilité haute (GHUI)

Un itemset X est un GHUI s'il satisfait deux condition [1] :

1. X est un ensemble d'items à utilité haute, c'est-à-dire que son utilité est supérieure ou égale à un seuil minimum donné.
2. Il n'existe pas de sous-ensemble propre Y de X tel que Y est aussi un HUIM.

En d'autres termes, un GHUI est un HUIM minimal qui n'a pas de sous-ensemble propre satisfaisant également le critère d'utilité élevée.

5.1.4 Minimal HUI (MinHUI)

Un ensemble d'items X est considéré comme un ensemble d'items à utilité élevée minimal (MinHUI) si et seulement si $u(X) \geq$ minutil et qu'il n'existe pas d'ensemble d'items $Y \subset X$ tel que $u(Y) \geq$ minutil [1].

Cette représentation proposée est l'opposée des ensembles d'items à utilité élevée maximaux (MaxHUI), c'est-à-dire qu'elle se compose des ensembles d'items ***les plus petits***qui génèrent un bénéfice élevé ***plutôt*** que des plus grands.

5.2 Algorithmes de représentation concise

Plusieurs algorithmes ont été développés pour trouver de manière efficace les formes compactes des ensembles d'items à haute utilité mentionnés précédemment. Le tableau 7 présente un aperçu de ces algorithmes et de leurs caractéristiques. Dans le nombreux cas,

l'extraction des formes compactes soit beaucoup plus rapide que découvrir l'ensemble des ensembles d'items à haute utilité, car cela permet de trouver moins d'ensembles d'items.

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

Algorithme	NB de phases	Représentation BD	Extension	Année de développement
EFIM-Closed	Une	Horizontal	EFIM	2016
Min_FHM	une	Vertical	FHM	2016
CLS_Miner	Une	Vertical	FHM	2019
CHUD	Deux	Vertical	DCI_Closed	2011
CHUI_Miner	Une	Vertical	DCI_Closed	2019
GUIDE	Une	Vertical	UPGrowth	2019
CHUI-Miner	Une	Vertical	HUI-Miner	2015

6 Conclusion

Dans ce chapitre nous avons donné une étude détaillée sur le domaine de la fouille de données. Nous avons présenté premièrement le problème de base de ce domaine qui est l'extraction des itemsets fréquents(FIM). Ensuite, nous avons expliqué en détails l'extraction des itemsets à haute utilité (HUIM).

Les algorithmes HUIM explorent les bases de données transactionnelles pour identifier des itemset à haute utilité, Ces algorithmes utilisent des techniques telles que la recherche en largeur ou la recherche en profondeur , et nous avons présenté une description de quelque algorithmes d'extraction d'ensemble d'items a haute utilité dans une tableau . Finalement, nous avons parlé sur la représentation concise principales d'ensembles d'itemsets à haute utilité .

Chapitre III: Réalisation & Interprétation

Chapitre III: Réalisation & Interprétation

1 Introduction

Dans ce chapitre, nous explorons en détail l'implémentation des algorithmes de fouille d'ensembles d'itemsets à haute utilité (HUIM) pour la représentation concise et l'interprétation des résultats obtenus. Les algorithmes CHUD ,EFIM-Closed , MinFHM et CHUI_MinerMax se démarquent comme des solutions efficaces pour accomplir cette tâche. Donc , nous avons mis en œuvre ces des algorithmes sur deux bases de données transactionnelles différentes. .

La première base de donnée est réel qui appelée "base de cosmétique " concerne les ventes des produits dans une boutique de cosmétique , et la deuxième base de données concerne les champignons "mushrooms" est un jeu de données utilisé pour classer les champignons comme étant définitivement comestible, définitivement toxique ou de comestibilité inconnue et non recommandée Nous accordons également une attention particulière aux outils de développement employés dans notre projet.

On présente les différentes technologies, langages de programmation et bibliothèques utilisées pour extraire les ensembles d'items fréquents, telles que Eclipse, Java, python et la bibliothèque SPMF. Dans notre environnement de développement, nous exposons comment nous avons mis en place ces outils et intégré les algorithmes de recherche d'ensembles d'éléments à grande utilité.

Nous fournirons également une description détaillée de l'architecture générale de notre implémentation . Nous exposons la composition du code, les diverses classes et modules employés afin de mettre en place les algorithmes de recherche d'ensembles d'éléments à grande valeur.

Nous examinons aussi les aspects de performance tels que l'efficacité de l'utilisation de la mémoire et l'amélioration du temps d'exécution, qui ont été pris en considération lors de la mise en œuvre des algorithmes.

2 L'algorithme Closed HUI

L'algorithme CHUD (Closed High Utility Discovery) est un algorithme fait référence à un type spécifique d'algorithme utilisé dans la fouille de données pour extraire les ensembles d'items fréquents à haute utilité (High Utility Itemsets) , Il s'agit d'une variante de l'algorithme d'ensembles à haute utilité qui vise à identifier les ensembles d'itemsets qui apparaissent fréquemment ensemble dans une base de données et qui ont une utilité élevée, comme définie par une fonction d'utilité. L’algorithme CHUD est conçu pour être efficace dans la découverte

d'ensembles à haute utilité fermés, c'est-à-dire des ensembles d'éléments qui ne peuvent pas être étendus en ajoutant d'autres éléments sans réduire leur utilité. Cette propriété de fermeture est importante car elle permet d'identifier des ensembles itemsets qui sont réellement pertinents et non simplement des sous-ensembles d'autres ensembles plus grands. L'algorithme CHUD utilise une approche basée sur l'exploration de l'espace des ensembles d'itemsets pour identifier les ensembles à haute utilité fermés. Il commence par identifier les éléments individuels à haute utilité, puis explore les ensembles d'éléments de taille croissante pour identifier les ensembles à haute utilité fermés. L'algorithme CHUD a été conçu pour être efficace dans des bases de données de transactions où les éléments peuvent avoir des utilités différentes et où les ensembles d'itemsets peuvent avoir des tailles variables. Il a été testé sur des données réelles et a montré des résultats prometteurs dans la découverte d'ensembles à haute utilité fermés.

L'objectif principal : L'algorithme CHUD se distingue par sa capacité à réduire le nombre de candidats à examiner en exploitant la propriété de fermeture, ce qui permet une exploration plus efficace de l'espace de recherche et une extraction plus rapide des itemsets à haute utilité pertinents. En éliminant les redondances et en se concentrant sur les itemsets fermés, CHUD améliore la performance et la pertinence des résultats dans le processus de découverte d'itemsets à haute utilité.

3 L'algorithme EFIM-Closed

L'algorithme EFIM-Closed utilise un espace de recherche qui est représenté comme un arbre d'ensemble d'items (figure 7). Cet arbre est construit en utilisant un ordre total sur les éléments, généralement l'ordre lexicographique. L'algorithme explore cet espace de recherche en utilisant une recherche en profondeur, commençant par le nœud racine (l'ensemble vide). Pour chaque ensemble d'items, EFIM-Closed ajoute un item à la fois en fonction de l'ordre défini, générant ainsi des ensembles d'items plus grands.

3.1 L'espace de recherche

La procédure de recherche est une recherche en profondeur qui explore l'espace de recherche de tous les ensembles d'éléments à l'aide d'un arbre d'énumération d'ensembles. Il part de la racine (l'ensemble vide) et ajoute récursivement un élément à la fois à l'ensemble d'éléments actuel selon un ordre prédéfini (généralement l'ordre lexicographique). Ce processus se poursuit jusqu'à ce que tous les ensembles d'éléments

possibles soient générés [14].

La procédure de recherche est conçue pour découvrir efficacement des ensembles d'éléments très utiles en réduisant le nombre d'analyses de base de données et en utilisant de nouvelles limites supérieures pour élaguer l'espace de recherche.

Voici les étapes clés de la procédure de recherche :

1. **Initialisation :** L'ensemble d'éléments actuel α est initialisé comme l'ensemble vide.
2. **Projection de la base de données** : La base de données projetée α est générée en filtrant les transactions qui contiennent des éléments dans α.
3. **Objets primaires et secondaires** : Les éléments primaires sont ceux qui peuvent étendre α selon l'ordre prédéfini. Les éléments secondaires sont ceux qui doivent être pris en compte dans les extensions de α, c'est-à-dire les éléments qui ont une utilité locale supérieure au seuil d'utilité minimum.
4. **Recherche en profondeur d'abord :** l'algorithme ajoute récursivement un élément à la fois à α selon l'ordre prédéfini, générant ainsi des ensembles d'éléments plus grands.
5. **Comptage d'utilités :** L'utilité de chaque ensemble d'éléments est calculée à l'aide de la technique Fast Utility Counting (FAC), qui consiste à additionner les utilités internes et externes des éléments de l'ensemble d'éléments.
6. **Élagage :** l'espace de recherche est élagué à l'aide de l'utilitaire de sous-arborescence et des limites supérieures de l'utilitaire local pour éliminer les ensembles d'éléments de faible utilité et leurs extensions.

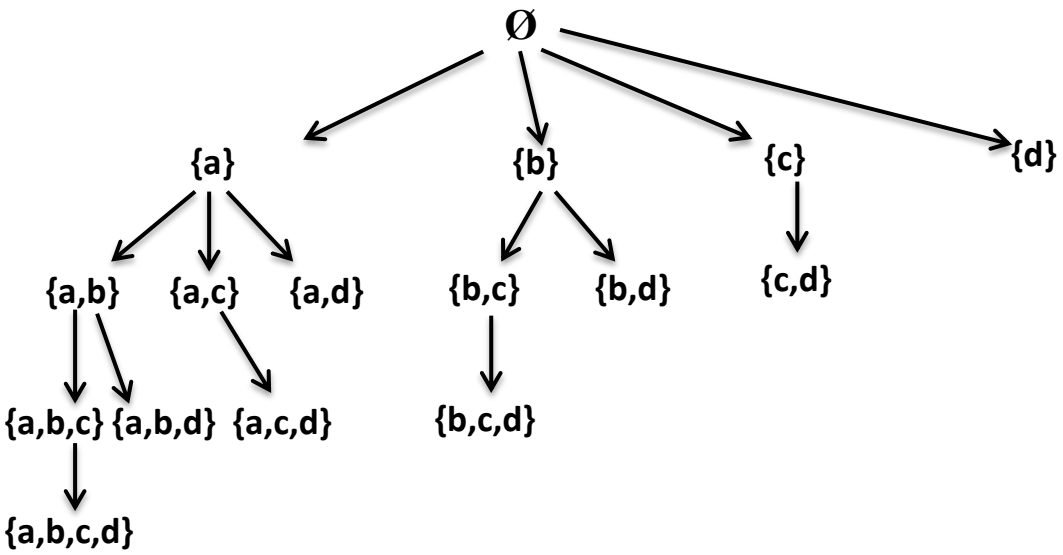

Figure 5:Arbre d'énumération pour I = {a, b, c, d}. [4]

3.2 Code d'algorithme EFIM-Closed

Dans cette sous-section, nous présentons l'algorithme EFIM-Closed proposé, qui combine toutes les idées présentées dans les sous-sections précédentes. La procédure principale (Algorithme 2) prend en entrée une base de données de transactions et le seuil minutil. L'algorithme considère initialement que l'ensemble d'éléments courant α est l'ensemble vide. L'algorithme analyse ensuite la base de données une fois pour calculer l'utilité locale de chaque élément par rapport à son utilisation. α, en utilisant un tableau utilitaire-bin. Ensuite, l'utilité locale de chaque élément est comparée à minutil pour obtenir les éléments secondaires par rapport à α, c'est-à-dire les éléments qui devraient être pris en compte dans les extensions de α. Ensuite, ces éléments sont triés par ordre croissant de TWU. La base de données est ensuite analysée une fois pour supprimer tous les éléments qui ne sont pas des éléments secondaires par rapport à α depuis. Si une transaction devient vide, elle est supprimée de la base de données. Ensuite, la base de données est analysée à nouveau pour trier les transactions selon l'ordre T afin de permettre la fusion des transactions, par la suite. Ensuite, l'algorithme analyse à nouveau la base de données pour calculer l'utilité du sous-arbre de chaque élément secondaire par rapport à chaque élément secondaire. α, en utilisant un tableau utilitaire-bin. Par la suite, l'algorithme appelle la procédure récursive Search pour effectuer la première recherche en profondeur à partir de α.

Algorithm 1: The EFIM-Closed algorithm

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

1 $\alpha = \emptyset$
2 Calculate $lu(\alpha, i)$ for all items $i \in I$ by scanning *D*, using a utility-bin array;
3 $Secondary(\alpha) = \{i | i \in I \wedge lu(\alpha, i) \geq minutil\}$;
4 Let $>$ be the total order of TWU ascending values on $Secondary(\alpha)$;
5 Scan *D* to remove each item $i \not\in Secondary(\alpha)$ from the transactions, and delete empty transactions;
6 Sort transactions in *D* according to $>_T$;
7 Calculate the sub-tree utility $su(\alpha, i)$ of each item $i \in Secondary(\alpha)$ by scanning *D*, using a utility-bin array;
8 $Primary(\alpha) = \{i | i \in Secondary(\alpha) \wedge su(\alpha, i) \geq minutil\}$;
9 Search $(\alpha, D, Primary(\alpha), Secondary(\alpha), minutil)$;

Chapitre III: Réalisation & Interprétation

4 L'algorithme MinFHM

L'algorithme MinFHM est un algorithme de fouille de données utilisé dans le domaine de l'exploration de motifs fréquents. HUI signifie "High Utility Itemset", ce qui se réfère à un ensemble d'articles dans une base de données transactionnelle qui a une valeur ou une utilité élevée selon certains critères définis. L'objectif principal de MinHUI est trouver les itemsets de haute utilité minimales dans un ensemble de transactions.

4.1 Code d'algorithme MinFHM

Algorithm 2: 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 ***MinHUI*** ← *each item* ***i*** *such that* ***utility(i) ≥ minutil*** *;*

3 I^* ← *each item* ***i*** *such that* ***TWU(i) ≥ minutil*** *and* ***utility(i) < minutil****;*

4 *Let > be the total order of* ***TWU*** *ascending values on* I^****;***

5 *Scan* ***D*** *to build the* ***utility-list*** *of each item* $i \in I^*$ *and build the* ***EUCS****;*

6 ***Output*** *each item* $i \in I^*$ *such that* ***SUM({i}.utilitylist.iutils) ≥ minutil****;*

7 ***Search (∅,I*, minutil, EUCS);***

8 ***end***

4.2 Principe de fonctionnement

1. **Scan de la base de données :** Le premier scan de la base de données permet de calculer le TWU (Total Weighted Utility) de chaque item.

2. **Identification des items de haute utilité :** Les items dont le TWU est supérieur ou égal au seuil de minutil sont identifiés et stockés dans le set I*.

3. **Construction de la structure EUCS** : Un deuxième scan de la base de données est effectué pour construire la structure EUCS (Estimated Utility Co-Occurrence Structure). Cette structure est utilisée pour stocker les triplets de formes (a, b, c) où $a, b \in I^*$ et c représente le TWU de l'ensemble $\{a, b\}$.

4. **Recherche des itemsets de haute utilité minimales :** Une exploration en profondeur des itemsets est initiée en appelant la procédure récursive "Search" avec l'itemset vide $\emptyset$, le set I^*, le seuil de minutil et la structure EUCS.

4.3 Optimisations

- **Ordre total :** Les items sont ordonnés selon leur TWU pour faciliter les comparaisons.
- **Pruning :** L'algorithme utilise des propriétés pour éliminer des itemsets qui ne peuvent pas être des MinHUIs, ce qui réduit l'espace de recherche.
- **Pruning de l'extension transitive :** L'algorithme ne explore pas les extensions transitives des itemsets MinHUIs, ce qui permet de réduire encore plus l'espace de recherche.

5 Présentation des outils de développement

5.1 NetBeans

L'environnement de développement intégré NetBeans est largement utilisé pour le langage de programmation Java. NetBeans est une plateforme open source et gratuite, ce qui en fait un choix apprécié parmi les développeurs Java. Il supporte aussi différents frameworks et technologies tels que JavaFX, Spring, Hibernate et Maven [3].

5.2 Java

java est bien plus qu'un simple langage de programmation. C'est à la fois un langage de programmation et une plate-forme informatique. C'est l'une des raisons pour lesquelles Java est si populaire et largement adopté, car il offre une portabilité et une compatibilité étendues. Le fait que Java soit présent sur plus de 850 millions d'ordinateurs de bureau et sur plus d'un milliard de périphériques dans le monde, y compris des smartphones et des téléviseurs connectés, souligne son importance et son ubiquité dans le paysage technologique actuel [1].

5.3 Python

Python est un langage de programmation polyvalent et populaire connu pour sa simplicité et sa lisibilité. Il est largement utilisé dans le développement Web, l'analyse de données, l'intelligence artificielle, le calcul scientifique, etc. La syntaxe de Python est conçue pour être intuitive et facile à apprendre, ce qui en fait un excellent langage pour les programmeurs débutants et expérimentés. De plus, sa vaste bibliothèque standard et ses innombrables packages tiers le rendent encore plus puissant et flexible. Que vous créiez un script simple ou une application complexe, Python dispose d'outils et de ressources pour effectuer le travail efficacement.

5.4 Une bibliothèque de donnée extra source (SPMF)

5.4.1 Un Aperçu sur SPMF

SPMF est une bibliothèque open-source réputée, écrite en Java, destinée à l'exploration de données, plus spécifiquement dans le domaine de l'extraction de modèles miniers. Cette richesse fonctionnelle en fait un outil polyvalent pour diverses tâches d'exploration de données.

L'un des points forts de SPMF est sa facilité d'intégration dans d'autres projets Java. SPMF peut être utilisé de différentes manières, que ce soit via son interface utilisateur intuitive, en ligne de commande ou intégré dans d'autres applications Java. Cette polyvalence contribue à sa popularité et à son adoption dans la communauté des développeurs [15].

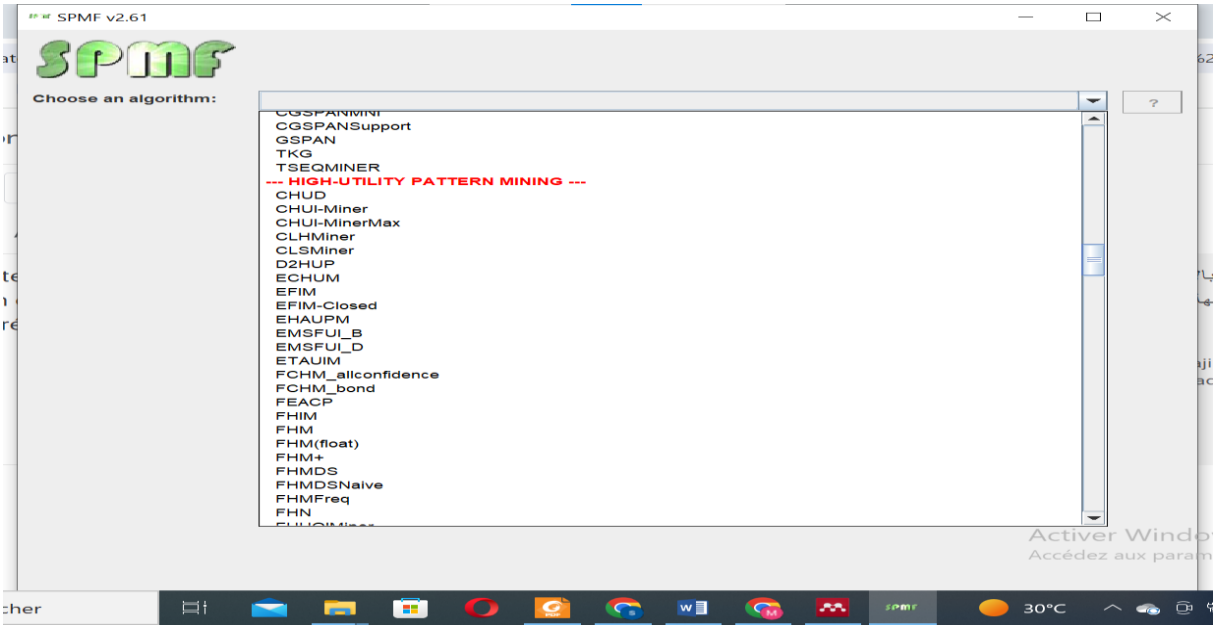

Figure 6: L'interface de SPMF

5.4.2 Variant de SPMF

SPMF est proposé dans deux formats différents :

a) **La version du code source** couvre l'ensemble des algorithmes. Pour compiler le code source et exécuter les exemples, il est indispensable d'avoir une expérience préalable avec Java.

b) **La version graphique** offre une interface utilisateur visuelle ainsi qu'une interface de commande. Cette version offre une utilisation plus simple. L'illustration ci-dessous présente l'interface de la version graphique de SPMF [3].

- Nous avons téléchargé la bibliothèque SPMF à partir du lien suivant: https://www.philippe-fournier viger .com/spmf/index.php?lin k=download. php

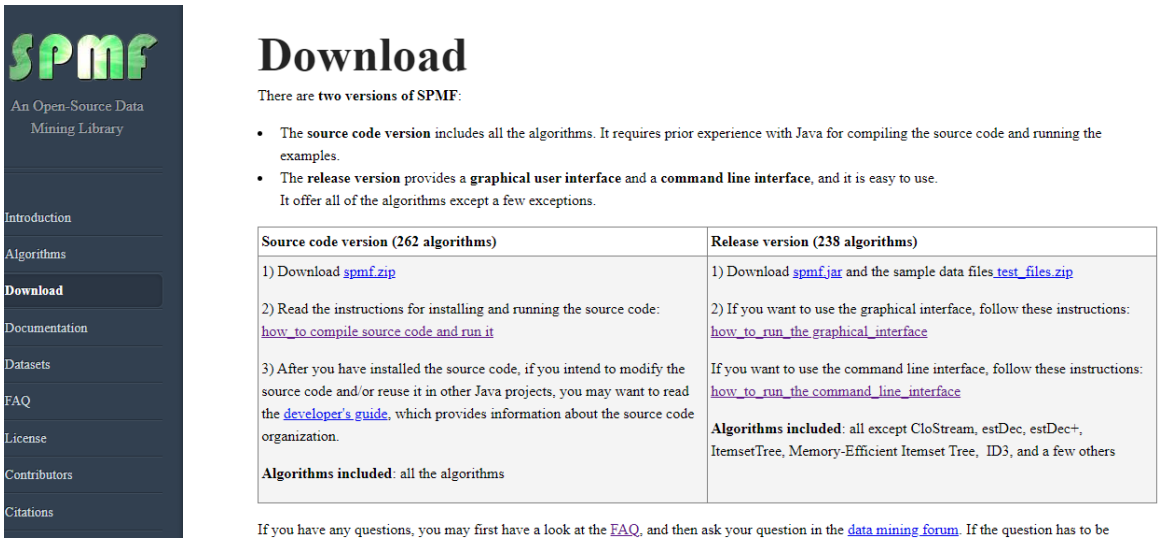

Figure 7:ETAPE DE TELECHARGEMENT

6 Conception et implémentation

6.1 Le Premier Cas d'Etude (base de données de cosmétique)

Nous avons pris les données que nous allons traiter à partir du magasin de cosmétiques. cette base de données principale est enregistré les détails des produits, des ventes , la quantité et d'autres informations clés. Les données extraites viennent en fichier Excel. Un fichier contient les détails sous forme des lignes : Chaque ligne représente les détails d'un produit qui a été acheté par un client. Chaque ligne est caractérisée par : La référence des

produit(IDProduit), le nom de produit(Designation), le numéro de bon(IDBon), la quantité vendue de produit(Qte), et le profit de chaque produit.

	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 8 : Base de données de Cosmétique

6.1.1 L'étape de prétraitement des données

Les fichiers fournis pour cette base de données ne sont pas compatibles avec la bibliothèque SPMF. Par conséquent, un processus de prétraitement est nécessaire pour transformer ces fichiers en une base de données transactionnelle au format SPMF, lequel est pris en charge par l'outil SPMF. Le format SPMF est un fichier texte composé de plusieurs lignes, chacune représentant les transactions effectuées par un seul client.

Chaque ligne est de la forme :

{item1 item2 item3 …} : TU : {Profit1 Profit2 Profit3 …}

Tels que :

{item1 item2 item3 …} sont les produits achetés par un client

{Profit1 Profit2 Profit3 …} représente les prix de ces produit

TU : est utilité totale de cette transaction

Les fichiers de base ont été convertis en format SPMF en utilisant un script Python. Les transactions sont déterminées par le numéro de bon (IDBon). En d'autres termes, une transaction contient tous les produits achetés et écrits dans un ticket.

Voici un aperçu de la base après la transformation en format SPMF :

Chapitre III: Réalisation & Interprétation

```
4407:50:50
548 1377 1483 1663 2621 2790 3122 3240 4951 5854 7396 7732 8885 9054
10637 10814 10881 11283 11494 11668 11854 11871 11895 12246 12306 12327
12384 12507 12542 12599:4511:15 40 40 27 50 68 95 300 30 168 9 72 65 600
160 520 720 150 90 200 140 50 20 380 160 60 50 20 112 100
61 109 2957 12384:425:70 55 200 100
6149 7623 12650:145:50 30 65
44 1552 8531 10612 12616 12616:202:60 12 25 45 40 20
4 78 475 5033 10105 10418 11668:500:60 55 80 70 55 60 120
10967:1060:1060
15 81 501 11288 12042:337:106 106 86 25 14
15 81 12042:240:106 106 28
587 2588 2790 6168 6581 10460 12507 12634 12650:1019:30 65 17 470 250 52
20 50 65
548 5410:140:60 80
4 264 529 11668 12121:560:30 220 250 40 20
514:40:40
11 79 104 126 320 398 435 616 1287 5189 6875 6947 8131 11234
12231:1975:50 50 30 50 110 30 50 115 70 400 55 50 20 45 850
2077 9678:135:35 100
9 17 75 284 2076 10856 11388:444:35 45 55 70 17 102 120
619 2494 5864 7460:605:50 55 150 350
408 409 534 2395:88:10 50 10 18
265 492 501 2121 3670 10644 12135 12471:1011:30 180 43 430 50 26 92 160
76 2021 7623:133:75 28 30
2444 11870 12055:258:48 10 200
356 1365 8255:132:27 70 35
```

Figure 9: Un extrait du ficher SPMF de base cosmétique

dans cette étape , nous allons appliquer les algorithme de haute utilité pour représentation concise (EFIM_Closed , CHUI_MinerMax , MinFHM et CHUD) sur la base de données et on va effectuer une analyse des résultats pour tirer des conclusions utiles.

6.1.2 L'étape de fouille de données

Nous avons téléchargé Eclipse IDE à partir du lien suivant (Figure 10):

https://www.eclipse.org/downloads/packages/release/oxygen/3a/eclipse-ide-java-developers.

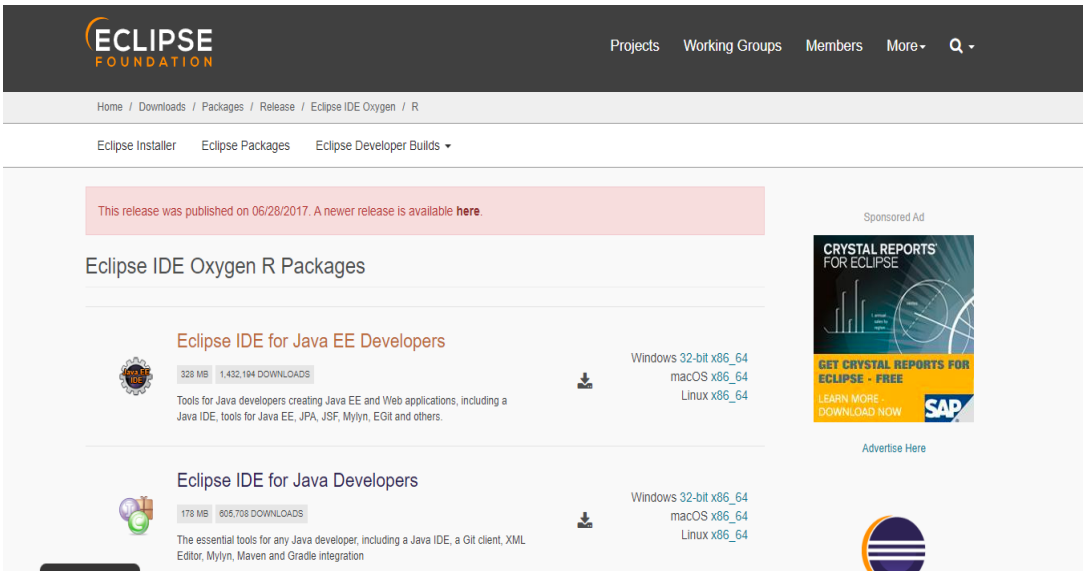

Figure 10 : Eclipse IDE

Chapitre III: Réalisation & Interprétation

Nous avons créé un nouveau projet Java et ajouté la bibliothèque SPMF au répertoire source (src) de ce projet. (Figure 11)

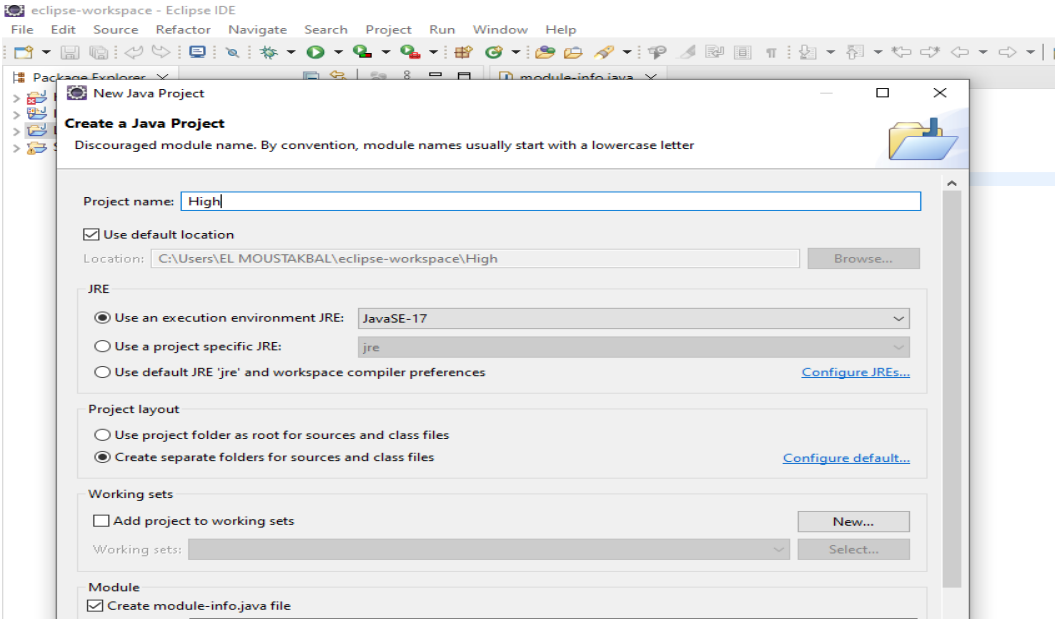

Figure 11 : Création d'un nouveau projet JAVA

Ajouter la bibliothèque SPMF : (Figure 12)

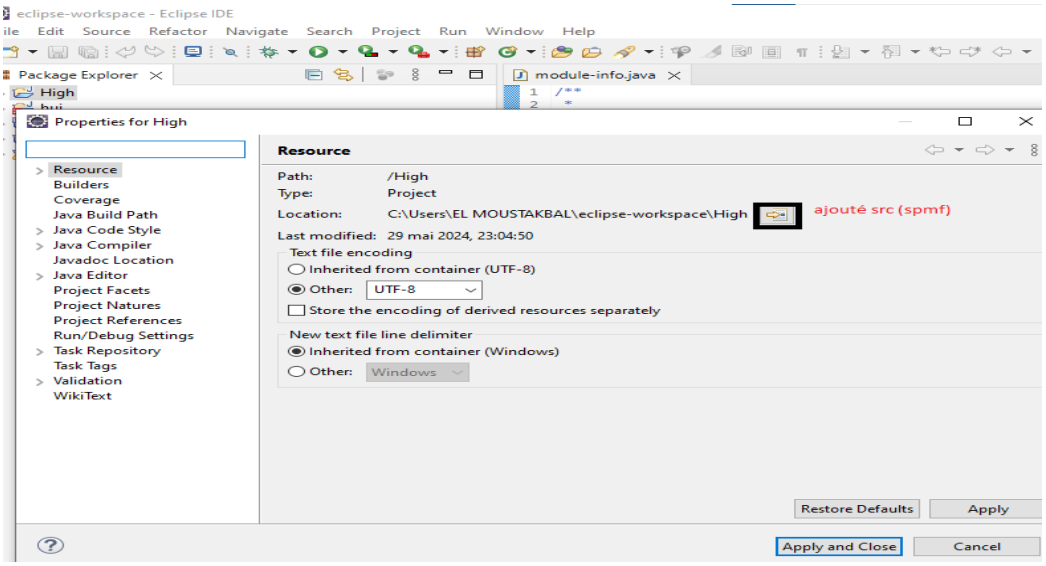

Figure 12 : Ajoute la bibliothèque SPMF

Chapitre III: Réalisation & Interprétation

Nous avons fait des tests dans (Figure 13)

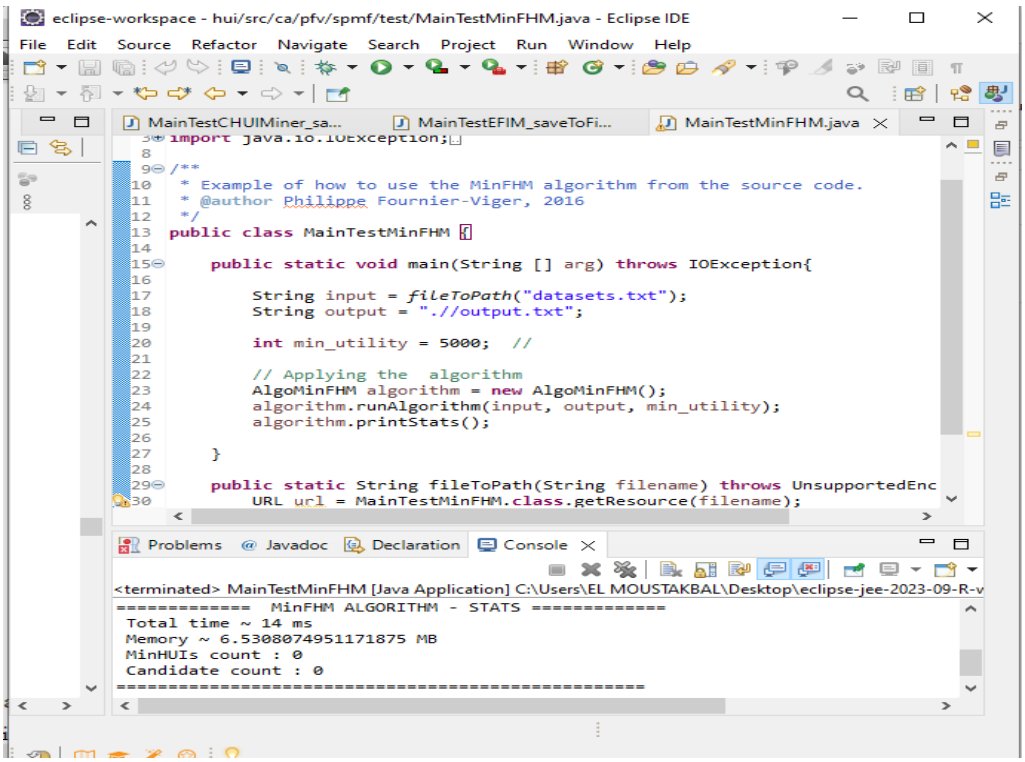

Figure 13 : Exécution d'algorithme MinFHM avec la base cosmétique

Ou on d'autre manière on exécuter l''algorithme MinFHM par SPMF et donne le résultat suivant et enregistré dans un fichier text , Un extrait de ficher des résultats est présenté par (Figure 14)et (Figure 15) .

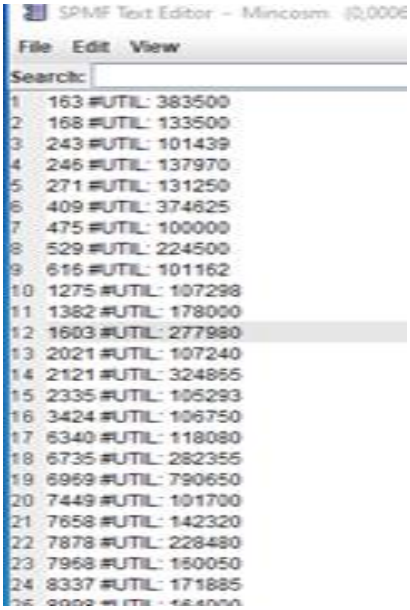

Figure 14: Extrait de fichier des résultats

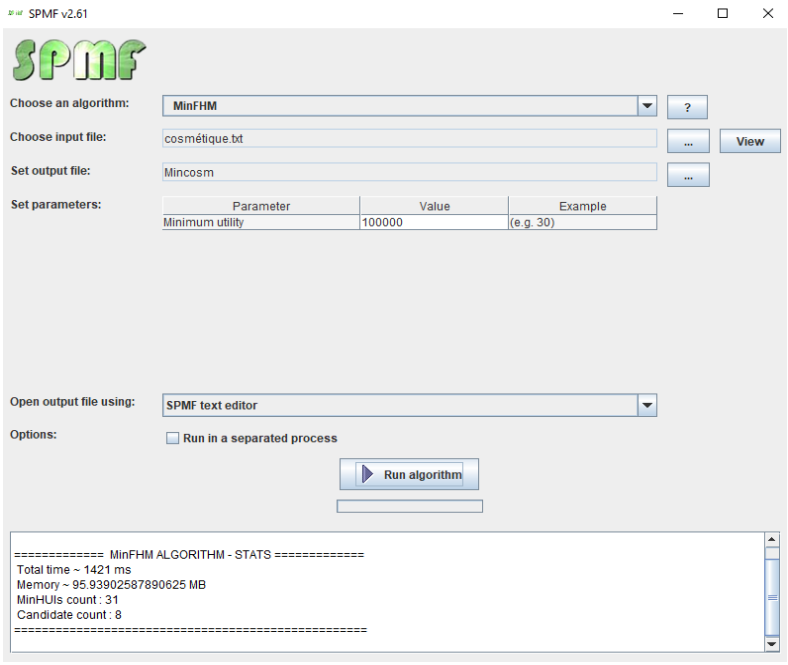

Figure 15: Exécuter l'algorithme MinFHM par SPMF

6.1.3 Interprétation et comparaison les résultats

Nous avons réalisé des expériences pour évaluer les performances des algorithmes proposé en fonction de trois critères : le temps d'exécution, l'utilisation du mémoire et le nombre de patterns découverts lors de l'application des algorithmes MinFHM, EFIM_Closed , CHUD et CHUI_MinerMax en représentations concises.

les algorithme ont été effectuées sur un ordinateur équipé d'un processeur *Core i5 de 6^{e} génération 64 bits*, fonctionnant sous *Windows 10* et disposant de *16,0 Go de RAM libre* , ces algorithme est tester selon plusieurs valeurs de seuil minutil. La valeur de seuil est changée de 5000 à 600000. Le résultat des algorithmes est présenté par le (Table 8) et résumée par les Figures **(**Figure16) (Figure 17) (Figure18).

Chapitre III: Réalisation & Interprétation

Tableau 8: Résultats de l'extraction des algorithmes EFIM Closed ,CHUD ET MinFHM , CHUI_MinerMax sur la base de cosmétique

Algorithme		Minutil							
		200000	120000	100000	80000	70000	60000	45000	35000
EFIM Closed	Temps d'exécution	10,5	14,9	18,7	28,8	39,4	59,5	91,1	116
	Mémoire (MB)	325,9	413,5	473 ,2	524,4	551 ,9	557 ,4	527 ,4	516 ,9
	Nombre de HUIM	9	17	30	53	47	100	152	99
CHUD	Temps d'exécution	70	137,6	143	233,4	383,3	438,85	506,4	488,7
	Mémoire (MB)	857,6	433,6	592 ,1	317,5	515,4	552,4	142,4	533,8
	Nombre de HUIM	10	20	33	56	80	108	166	260
Min_FHM	Temps d'exécution	7,8	9,6	13,5	12,7	13	14,05	19,2	17,4
	Mémoire (MB)	361,4	976,1	616,1	1148,6	720,6	293,6	803,6	375,6
	Nombre de HUIM	0	0	31	54	75	101	154	223
CHUI_MinerMax	Temps d'exécution	12,9	16,9	21,8	29,4	41,2	64,2	95,2	191,2
	Mémoire (MB)	266,3	859,1	466,1	1021,4	590,4	1109,4	622,4	986,1
	Nombre de HUIM	9	17	30	53	74	100	152	99

6.1.4 Résultats

Nous avons d'abord exécuté les algorithmes CHUD et EFIM-Closed et les algorithme MinFHM et CHUI_MinerMax sur chaque ensemble de données, Nous avons graduellement diminué le seuil de minutil jusqu'à ce que les algorithmes présentent des variations dans leur temps d'exécution, dépassent la mémoire disponible, ou qu'une solution nettement meilleure se manifeste. Pour chaque jeu de données, nous avons consigné les temps d'exécution (S), l'utilisation de la mémoire (MB) et le nombre d'ensembles d'éléments à haute utilité générés.

Influence du seuil minutil sur le temps d'exécution:

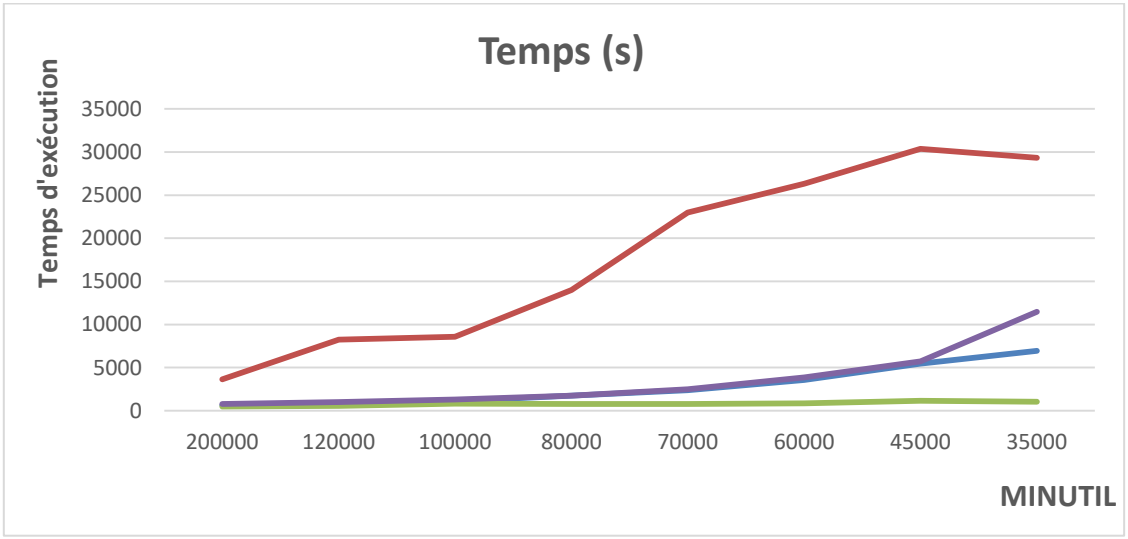

Figure 16 : Temps d'exécution des algorithmes EFIM_Closed ,CHUD, MinFHM, CHUD_Miner_Max

On peut observer une comparaison des temps d'exécution dans la Figure 17. Le premier algorithme sur chaque ensemble de données est MinFHM , suivi de EFIM-Closed et CHUI_Miner_Max , enfin CHUD qui demande beaucoup de temps.

Chapitre III: Réalisation & Interprétation

Influence du seuil minutil sur l'utilisation de la mémoire :

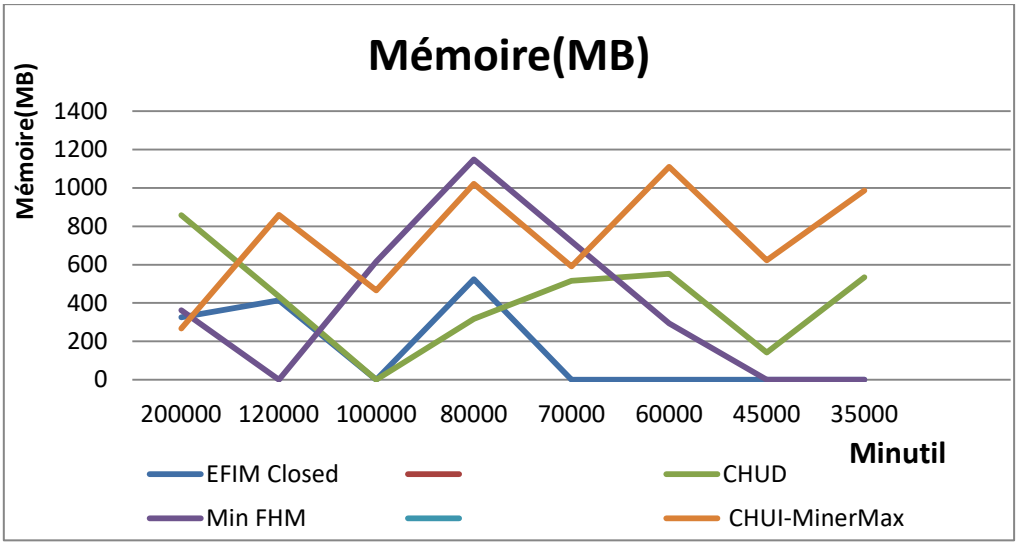

Figure 17 : L'espace mémoire occupe de l'extraction des algorithmes EFIM_Closed ,CHUD, MinFHM, CHUD_Miner_Max

La comparaison de l'utilisation de la mémoire est illustrée dans la Figure 18. En comparaison avec les ensembles de données, EFIM-Closed consomme moins d'espace que CHUD et CHUI_MinerMax, tandis que MinFHM utilise une quantité de mémoire considérable.

Comparaison du nombre de HUIM :

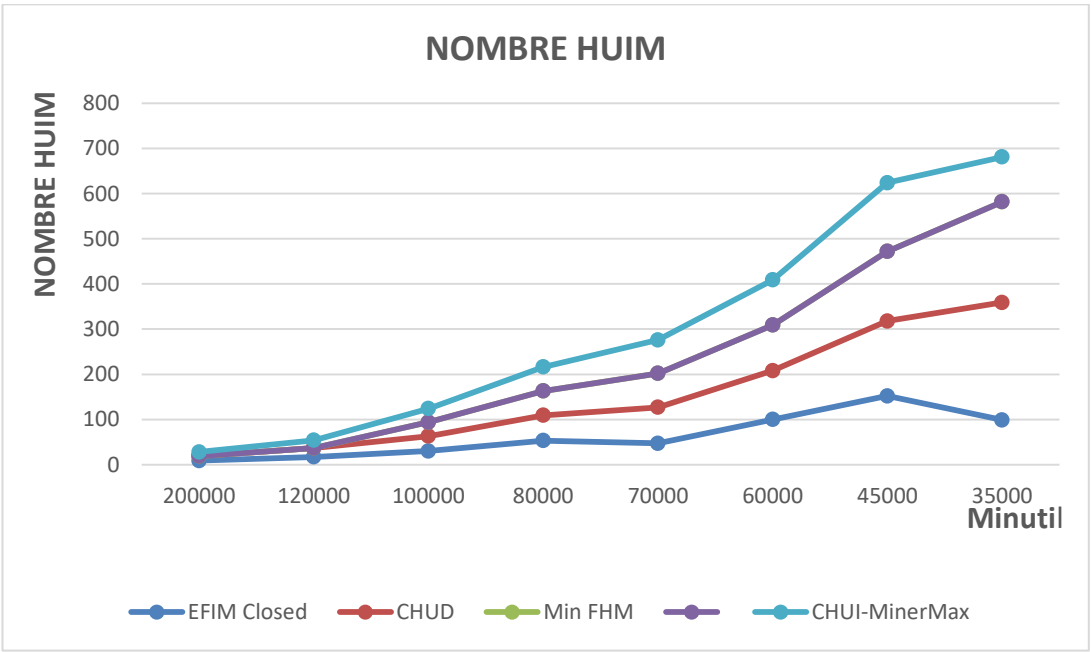

Figure 18 : Nombre HUIM dans les algorithmes EFIM_Closed ,CHUD, MinFHM, CHUD_MinerMax .

La Figure 19 présente une comparaison de nombre d'ensembles d'itemsets à grande utilité qui sorti . EFIM-Closed retourne moins d'ensembles d'itemsets à haute utilité que CHUD, CHUD en retourne moins que MinFHM, puis CHUI_MinerMax en retourne un grand nombre.

6.2 Le deuxièmes Cas d'Etude (base de champignons)

Nous avons pris les données que nous allons traiter à partir d'un échantillon de champignons (Mushrooms) à branchies de la famille des champignons Agaricus et Lepiota tirées du Guide de terrain de la Audubon Society sur les champignons nord-américains (1981). la base de données de champignons extraites viennent en fichier Excel. qui contient les détails de champignons.

La base de données "mushrooms" est un jeu de données utilisé pour classer les champignons comme étant définitivement comestible, définitivement toxique ou de comestibilité inconnue et non recommandée. Cette dernière classe était combinée avec la classe toxique.

Nous avons téléchargé cette base de données à partir du lien suivant **:**

https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php#c1 .

Pour cette base, on n'a pas besoin d'une phase de pré-traitement car la base est déjà préparée et présentée sous format SPMF. donc, nous allons directement tester cette base avec les algorithme de représentation concise

Voici un aperçu de la base de données de champignons en format SPMF :

Figure 19: Base de champignons en format SPMF

6.2.1 Exécution de algorithme MinFHM

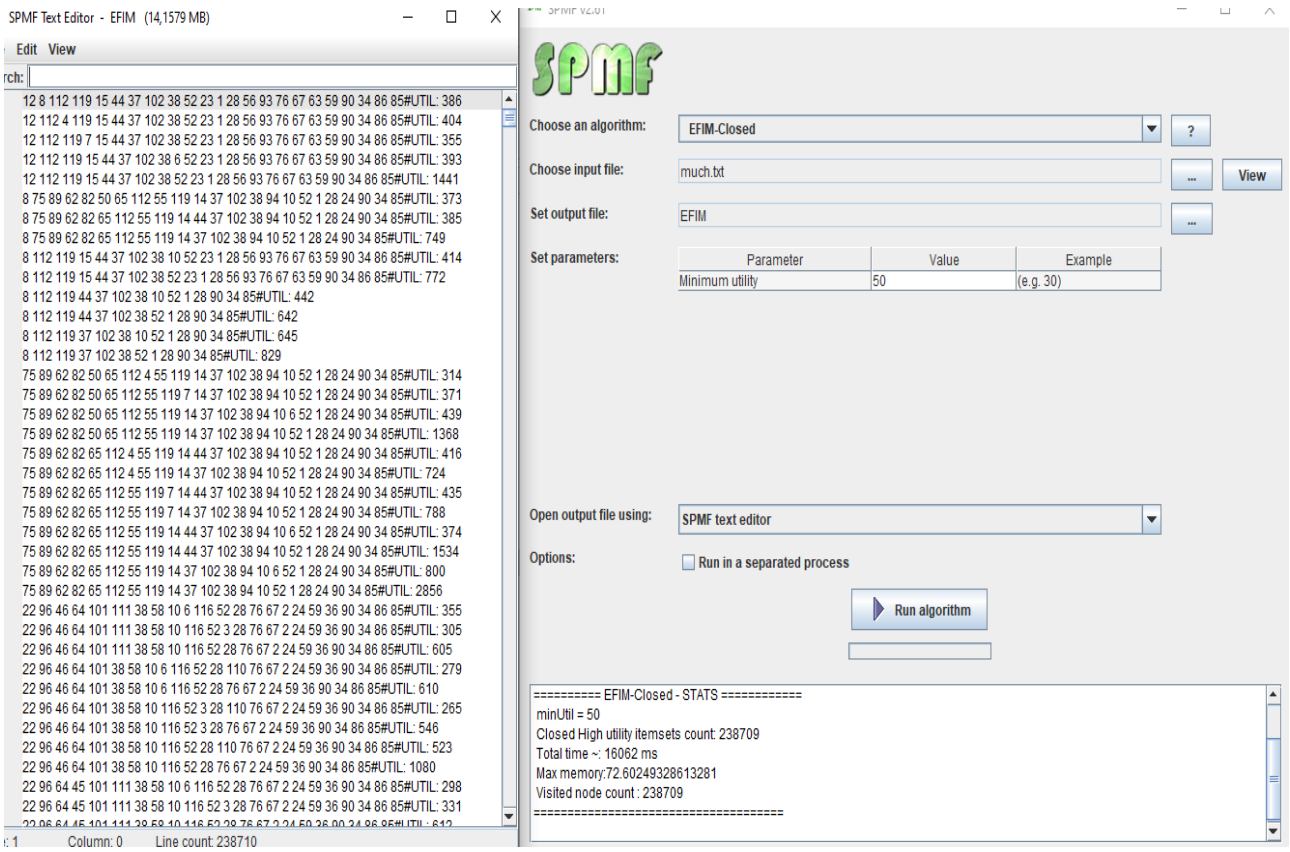

Figure 20: Exécution d'algorithme MinFHM

6.2.2 Interprétation et comparaison les résultats

Pour évaluer les performances des algorithmes proposés, nous avons effectué des expériences en se basant sur trois critères : le temps d'exécution, l'utilisation du mémoire et le nombre de patterns découverts lors de l'application des algorithmes MinFHM, EFIM_Closed, CHUD et CHUI_MinerMax en représentations concis .(tableau 9)

Tableau 9: Résultat de l'exécution des algorithme EFIM Closed , MinFHM et CHUI_MinerMax et CHUD.

Algorithme		Minutil							
		70000	60000	50000	40000	35000	30000	25000	20000
EFIM_Closed	Temps d'exécution	73,7	69,5	79,4	89,8	94	103,7	108,1	118,5
	Mémoire (MB)	128	169,9	263,9	272,9	275	289,3	350,1	361,1
	Nombre de HUIM	9917	12615	16617	22034	25992	30736	36950	45891
CHUI_MinerMa	Temps d'exécution	132	132,8	190,1	272,1	368,1	501,6	859,9	1856
	Mémoire (MB)	38,7	90,1	36,1	50,8	74,2	44,7	82,3	80,5
	Nombre de HUIM	1576	2013	3515	4285	4853	5524	6994	9931
MinFHM	Temps d'exécution	12,8	7,8	3,9	2,85	1,8	1,8	2,1	1,6
	Mémoire (MB)	357,6	251,4	362,6	190,8	270,6	351,1	174,5	251,1
	Nombre de HUIM	47	47	43	31	36	40	45	48
CHUD	Temps d'exécution	202,6	185,4	193,2	201	199,7	205,5	213	225,5
	Mémoire (MB)	110,2	124,5	133,8	160,7	202,7	222,2	225,4	303,4
	Nombre de HUIM	9917	12615	16617	22034	25992	30736	36950	45891

6.2.3 Résultats

Dans un premier temps, nous avons exécuté les algorithmes CHUD et EFIM-Closed et les algorithme MinFHM et CHUI_MinerMax sur chaque ensemble de données. Nous avons progressivement réduit le seuil minutil jusqu'à ce que les algorithmes prennent diffèrent de temps pour s'exécuter, dépassent la mémoire disponible ou qu'un gagnant clair soit observé. Pour chaque ensemble de données, nous avons enregistré les temps d'exécution(S), la mémoire utilisée (MB) et le nombre d'ensembles d'éléments à haute utilité exécuté.

Influence du seuil minutil sur le temps d'exécution:

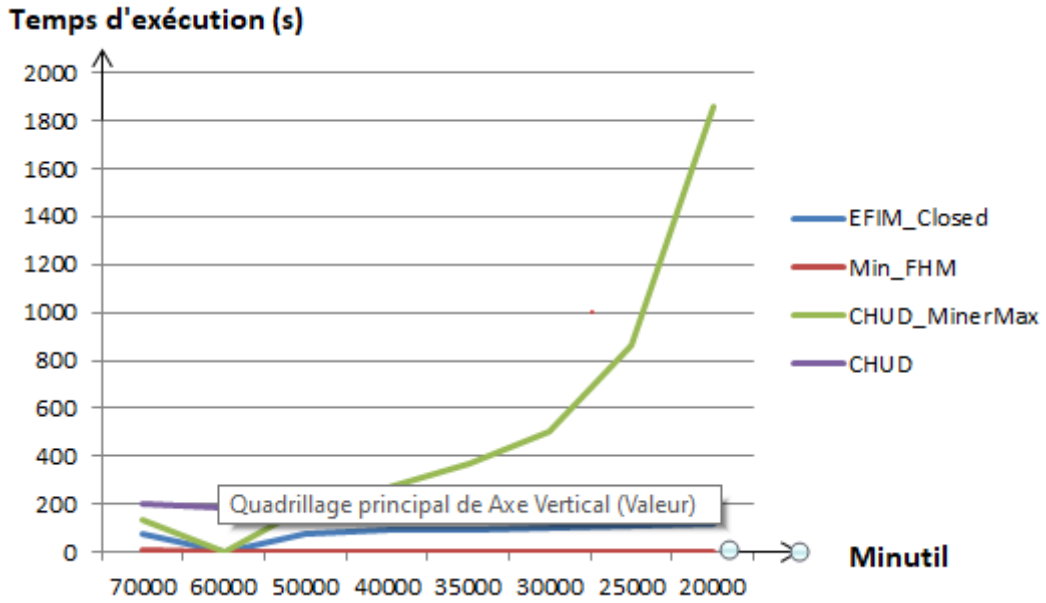

Figure 21 : Temps d'exécutiondes algorithmes EFIM_Closed , MinFHM et CHUI_MinerMax et CHUD .

On peut observer une comparaison des temps d'exécution dans la Figure 21. Le premier algorithme sur chaque ensemble de données est MinFHM, suivi de EFIM_Closed , puis de CHUD , enfin CHUI_MinerMax qui demande beaucoup de temps d'exécution .

Chapitre III: Réalisation & Interprétation

Influence du seuil minutil sur l'utilisation de la mémoire :

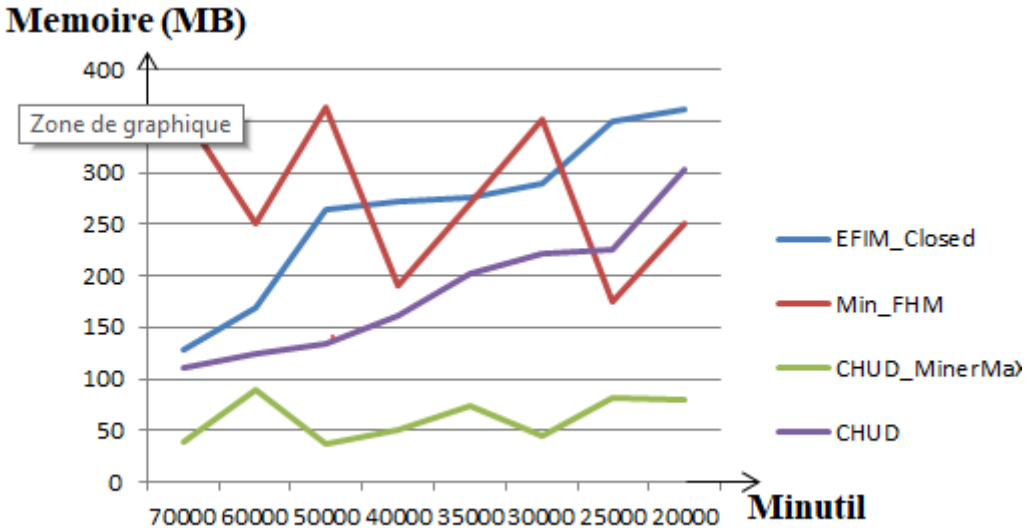

Figure 22:Espace mémoire occupe de l'exécution des algorithmes EFIM_Closed , MinFHM et CHUI_MinerMax et CHUD .

La comparaison de l'utilisation de la mémoire est illustrée dans la Figure 22. En comparaison avec les ensembles de données, CHUI_MinerMax consomme moins d'espace que CHUD et EFIM-Closed, tandis que MinFHM utilise une quantité de mémoire considérable.

Comparaison du nombre de HIM :

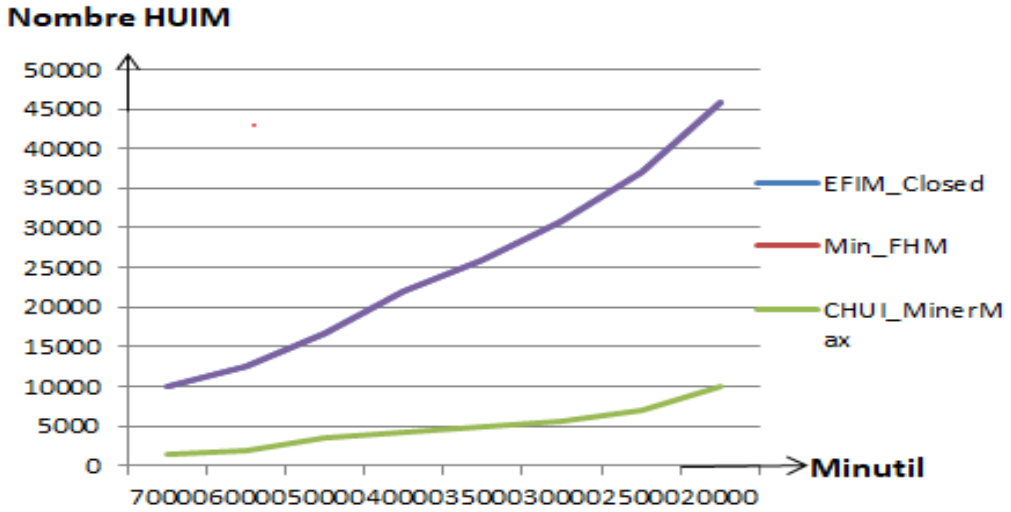

Figure 23:le nombre de HUIM exécuté dans les algorithmes EFIM_Closed , MinFHM et CHUI_MinerMax et CHUD .

La Figure 23 présente une comparaison de nombre de HUIM exécuté. EFIM Closed et CHUD retourne grands de nombre de HUIM que CHUI_MinerMax, puis MinFHM en retourne un moin nombre.

Conclusion

En conclusion , cette étude comparative des algorithmes de fouille d'ensembles d'items à haute utilité (HUIM) pour une représentation concise dans les bases de données transactionnelles a permis de mettre en lumière les forces et les faiblesses de chaque approche. En utilisant des bases de données et des fichiers SPMF dans divers cas d'étude, nous avons évalué les performances de ces algorithmes en termes de temps d'exécution, d'utilisation de la mémoire, et du nombre d'ensembles HUIM découverts.

Conclusion générale

Dans ce travail, nous avons examiné attentivement le domaine de la recherche de données, en mettant l'accent sur la recherche d'ensembles d'éléments à haute utilité pour une représentation concise de l'extraction des ensembles d'éléments à haute utilité]. Les méthodes d'extraction de connaissances utilisées dans ce domaine ont été examinées et les algorithmes clés ont été analysés pour extraire des ensembles d'items à haute utilité.

Tout d'abord, nous avons compris l'importance croissante de la fouille de données dans notre monde moderne, où la quantité de données générées chaque jour est immense. La fouille de données nous offre une opportunité précieuse de découvrir des connaissances cachées, des modèles et des tendances à partir de ces données massives.

Ensuite, nous avons examiné les différentes tâches et applications de la fouille de données, en comprenant les étapes du processus de fouille de données, de la préparation des données à l'interprétation des résultats. Nous avons également exploré l'évolution de la fouille de données dans le contexte des systèmes de bases de données, en comprenant comment elle s'est développée pour répondre aux défis croissants liés à l'analyse des données.

Nous avons étudié les algorithmes d'extraction des ensembles d'items fréquents, tels que la recherche en largeur et la recherche en profondeur, ainsi que l'algorithme Apriori. Nous avons souligné les limitations de la fouille des ensembles d'items fréquents et avons identifié la nécessité d'extraire des ensembles d'items ayant un impact significatif sur une mesure spécifique.

Pour répondre à cette nécessité, nous avons présenté les algorithmes de HUIM pour représentation concise pour l'extraction des ensembles d'items à haute utilité. De plus, nous avons exploré les représentations concises utilisées dans la fouille de données, pour réduire l'espace de recherche, ainsi que l'utilité list pour accélérer le processus d'extraction des ensembles d'items à haute utilité.

La fouille de données et en particulier la fouille d'ensembles d'items à haute utilité offrent des opportunités passionnantes pour découvrir des connaissances précieuses à partir de données massives.

Bibliographie :

[1] M. Bakour, (2023) " Étude comparative entre les algorithmes de haute utilité et les algorithmes de représentation concise ", Université de Mohamed El Bachir El Ibrahimi de Bordj Bou Arreridj.

[2] K .Kamilia , (2019) " Chapitre 1 : extraction de connaissance "Université Mouloud Mammeri de Tizi Ouzou .

[3] M. Sena and C. Aya, "Découverte des Motifs Profitables dans les Base de Données Transactionnelles ," 2022/2023.

[4] E. Sup, R. S. Universit, M. Khider, B. Facult, S. Exactes, and A. Djeffal, “Cours Fouille de données avancée Plan du cours,” 2013.

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

[6] H. Benamar, “Etude explorative d’outils pour le data mining,” 2007.

[7] C. I. Nombré et al., “ALOA2i : Optimisation d'extraction des K-itemsets Frequents (pour K≤ 2) To cite this version : HAL Id : hal-01386948,” 2016.

[8] P. Fournier-Viger, J. Chun-Wei Lin, T. Truong-Chi, and R. Nkambou, “A Survey of High Utility Itemset Mining,” Stud. Big Data, vol. 51, pp. 1–45, 2019, doi: 10.1007/978-3-030-04921-8_1.

[9] W. Rungsarityotin, “Algorithm to identify protein complexes from high-throughput data.” 2007.

[10] K. Discovery, “Chapter 3 : Frequent Itemset Mining,” pp. 1–63, 2016.

[11] M. F. Messaouda, “Optimisation par Colonies de Fourmis pour l ’ Extraction des Itemsets à partir de données incertaines,” pp. 2021–2022, 2022.

[12] J. D. Laborda, P. Torrijos, J. M. Puerta, and J. A. Gámez, “Parallel structural learning of Bayesian networks: Iterative divide and conquer algorithm based on structural fusion,” Knowledge-Based Syst., vol. 296, no. April, p. 111840, 2024, doi: 10.1016/j.knosys.2024.111840.

[13] T. Wei, B. Wang, Y. Zhang, K. Hu, Y. Yao, and H. Liu, “FCHUIM: Efficient Frequent and Closed High-Utility Itemsets Mining,” IEEE Access, vol. 8, pp. 109928–109939, 2020, doi: 10.1109/ACCESS.2020.3001975.

[14] S. Zida, P. Fournier-Viger, J. C. W. Lin, C. W. Wu, and V. S. Tseng, “EFIM: A highly efficient algorithm for high-utility itemset mining,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9413, pp. 530–546, 2015, doi: 10.1007/978-3-319-27060-9_44.

[15] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. W. Wu, and V. S. Tseng, “SPMF: A java open-source pattern mining library,” J. Mach. Learn. Res., vol. 15, pp. 3389–3393, 2015.