Optimized Closed Frequent High Utility Itemset Mining Using OSR, OWL, And MSU Pruning On Retail Transaction Data

Kinana Syah Sulanjari [1,*)], and Chastine Fatichah [2)]

[1)] Departement of Technology Management, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia

[2)] Department of Informatics, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia

E-mail: kinana.syahsulanjari@gmail.com[1)], and chastine@its.ac.id[2)]

ABSTRACT

This research proposes the optimization of the Frequent Closed High-Utility Itemset Mining (FCHUIM) algorithm for large-scale retail transaction data through three heuristic pruning strategies: Observed Support Ratio (OSR), Observed Weighted Lift (OWL), and Modified Subtree Utility (MSU). OSR eliminates infrequent itemsets by evaluating their proportional support within the dataset, OWL measures the economic contribution of itemsets relative to their transaction utility, and MSU filters candidates based on the minimum utility overlap in shared transactions. These techniques are designed to reduce redundant computations and narrow the search space without compromising pattern quality. The proposed algorithm was tested on a real-world retail dataset from a consumer cooperative, consisting of 56,274 transactions and 4,265 unique items across five years. The experiments assessed the effectiveness of each pruning method under varying minimum support and utility thresholds, both independently and in combination. Results demonstrate that the integrated approach reduces the search space by up to 92.5%, leading to significant improvements in execution time and memory consumption. Sensitivity analyses indicate that utility thresholds have a greater influence on performance than support thresholds. Furthermore, scalability tests using incremental annual datasets show that the optimized algorithm maintains acceptable performance with near-linear degradation as data volume increases. These findings highlight the suitability of the enhanced FCHUIM algorithm for mining high-value itemsets in complex retail environments. Future work may explore adaptive parameter tuning and deployment in real-time decision support systems.

Keywords: High utility itemset mining, closed itemset, pruning heuristic, retail data mining.

1. Introduction

In modern business environments, particularly in the retail sector, understanding consumer purchasing patterns has become a critical factor in strategic decision-making [1]. With the proliferation of transactional data generated by Point of Sales (POS) systems and e-commerce platforms, extracting meaningful patterns from these datasets is more important than ever [2]. According to industry reports, the application of transaction-based pattern analytics can boost marketing and sales performance by over 30% [3]. Hence, there is a growing need for data mining approaches that not only account for item frequency but also consider the economic value contributed by each item in a transaction [4].

One such approach is High Utility Itemset Mining (HUIM), which focuses on discovering item combinations that contribute significantly to transactional utility, such as profit, quantity, or revenue, rather than merely their frequency [5]. HUIM is more suitable for business applications compared to Frequent Itemset Mining (FIM), as it captures the actual impact of items on overall transaction value [6]. However, traditional HUIM algorithms often suffer from performance bottlenecks when applied to large-scale or dense datasets, including excessive memory

* Corresponding author.
Received: June 12th, 2025. Revised: August 2nd, 2025. Accepted: December 1st, 2025.

Available online: January 15th, 2026.

DOI: https://doi.org/10.12962/j24068535.v24i1.a1311

usage, long computation times, and the generation of redundant patterns [7]. These redundant itemsets often share identical support and utility values, which complicates result interpretation and reduces analytical effectiveness [8].

To address these limitations, the FCHUIM (Frequent Closed High Utility Itemset Mining) algorithm was introduced [7]. It concentrates on extracting only closed itemsets, which are itemsets without supersets sharing identical support and utility values, thus minimizing redundancy while maintaining the quality of knowledge representation [9]. This research implements FCHUIM in Python, featuring modular stages such as utility list construction, transaction utility computation, Total Summary List (TSL) formulation, and TWU-based pruning. The recursive mining process also includes closed-itemset validation to ensure only high-quality patterns are retained.

Numerous studies have been conducted to improve HUIM efficiency through various optimization techniques. One such study represented the CG-FHAUI algorithm, which leverages generator and closed representations to mine concise frequent high average utility itemsets [8]. Their pruning strategies based on utility bounds significantly reduced search space and runtime, aligning with this study's emphasis on pruning-driven efficiency enhancement. In another advancement, Duong et al also proposed CloFHUOIM, an algorithm that integrates occupancy thresholds and utility constraints for compact pattern extraction. Their approach successfully applied multiple pruning techniques to reduce candidate sets, supporting this paper's use of OSR, OWL, and MSU heuristics for early elimination of non-promising items [8].

Complementary research proposed a developed a frequent closed high-utility itemset mining model using Leiden community detection and a compact genetic algorithm to group and optimize patterns. Their framework minimized redundancy and enhanced pattern relevance in complex retail datasets, confirming the significance of domain-specific optimization, a central theme of this study [10].

While prior research has demonstrated effective pruning and optimization techniques in HUIM, few studies have specifically integrated multiple heuristic pruning strategies into the closed itemset framework to balance efficiency and pattern relevance. Therefore, this study aims to fill this gap by proposing an optimized variant of the FCHUIM algorithm, enhanced with OSR (Observed Support Ratio), OWL (Observed Weighted Lift), and MSU (Minimum Suffix Utility) heuristics. The proposed approach is evaluated using a real-world retail dataset, with performance measured in terms of execution time, memory usage, and the number of high-utility closed itemsets discovered . The ultimate goal is to develop a scalable, efficient, and business-oriented pattern mining method suitable for large-scale retail analytics.

The remainder of this paper is organized as follows: Section 2 describes the materials and methods, including data preprocessing, utility modeling, and algorithm design. Section 3 presents the experimental setup and results, followed by a detailed discussion. Section 4 concludes the paper with key findings and directions for future work.

2. Literature Review

Numerous studies have analyzed the performance of pattern mining algorithms such as Apriori, FP-Growth, and ECLAT using both synthetic and real-world datasets. Apriori remains a foundational method due to its simple candidate generation mechanism, while FP-Growth is preferred in scenarios involving large-scale and lengthy transactions because of its tree-based compression structure known as the FP-Tree, which eliminates candidate generation [1], [2], [3], [4], [5]. ECLAT, with its vertical data format and tid-list intersection approach, is also highly efficient in dense datasets [6], [7], [8], [9], [10]. Several comparative evaluations have concluded that FP-Growth often outperforms Apriori when the dataset grows in size, though Apriori is more adaptable for smaller and sparse datasets [11], [12], [13], [14], [15]. Further studies have optimized Apriori for multichannel consumer behavior analysis to enable strategic decision-making across physical and digital retail platforms [16], [17], [18], [19], [20].

Traditional frequent itemset mining methods, however, face severe limitations in high-dimensional and utility-oriented applications. These algorithms typically assume equal importance among items, which is often unrealistic in practical retail scenarios. As a result, utility-based mining approaches have emerged. The integration of a Dynamic Genetic Algorithm (DGA) with rule coding for weighted association rule discovery has shown promise

in handling complex environmental and retail datasets where item utility varies significantly [5], [21], [22], [23], [24]. Additionally, MLHMiner has been introduced to extend the capabilities of traditional HMiner by supporting hierarchical pruning based on item taxonomies, enabling the discovery of multi-level high-utility itemsets in real-world settings [25], [26], [27], [28], [29]. These approaches emphasize not only frequency but also economic contribution, leading to more actionable patterns in retail analytics.

Recent developments have focused on HUIM under dynamic, streaming, and discount-driven transaction conditions. Algorithms such as EGUI-Tree utilize sliding window mechanisms to address concept drift and changing utilities, thus enabling adaptive real-time mining [30], [31], [32], [33], [34]. Similarly, the introduction of the Static Increment Ratio (SIR) and hybrid search strategies has improved the efficiency of join operations in utility list-based HUIM algorithms, especially in dense and continuous data environments [35], [36], [37], [38], [39]. The adoption of distributed and parallelized frameworks like Apache Spark and GPU-accelerated computation has further enhanced the scalability of pattern mining, allowing for real-time insights into massive transactional databases [6], [9], [20], [40], [41].

Another stream of research aims to eliminate the need for predefined thresholds like minimum support and minimum utility by proposing more flexible criteria for pattern selection. One of the most notable methods in this domain is PSI*, which applies skyline-based techniques combined with partitioned indexing to extract patterns that are both frequent and of high utility, without user-defined constraints [42], [43], [44], [45], [46]. These algorithms avoid arbitrary parameter settings and enable more exploratory mining, which is essential for complex and evolving data environments. Alongside this, the Closed-FHUIM algorithm focuses on reducing redundancy by integrating advanced closure checking and utility-balanced pruning, producing compact and high-quality itemsets suited for memory-constrained scenarios [7], [8], [14], [19], [25].

Building on these innovations, this study proposes an optimized version of the Frequent Closed High Utility Itemset Mining (FCHUIM) algorithm that integrates three pruning heuristics: Observed Support Ratio (OSR), Observed Weighted Lift (OWL), and Modified Subtree Utility (MSU). OSR filters out low-frequency items early by calculating their proportional support within the dataset, while OWL measures an itemset's contribution to overall transaction utility, enabling the removal of economically insignificant combinations [11], [13], [15], [26], [30]. MSU evaluates the utility potential of itemset extensions within shared transactions, allowing for early exclusion of combinations with minimal incremental gain [31], [33], [35], [38], [40]. This three-fold heuristic pruning substantially narrows the search space and reduces redundant computations without sacrificing the relevance of extracted patterns.

This study contributes in three major ways. First, it introduces a heuristically enhanced FCHUIM algorithm that addresses the computational inefficiencies and redundancies present in conventional high-utility pattern mining methods. Second, the algorithm is empirically tested on a real-world retail dataset consisting of 56,274 transactions and 4,265 unique items spanning five years, providing robust scalability and performance benchmarks [2], [3], [5], [18], [36]. Third, the study investigates how fixed threshold values, minimum utility and minimum support, affect algorithmic behavior and output relevance, offering practical guidance for deployment in retail analytics platforms [9], [12], [16], [23], [41].

Empirical results demonstrate the effectiveness of the proposed pruning methods. During the initial preprocessing stage, OSR, OWL, and MSU collectively eliminated 92.5% of candidate itemsets, retaining only 318 out of 4,252 unique items for further mining [1], [8], [19], [24], [29]. The average OSR value for retained items was significantly higher than for discarded ones, validating its sensitivity to support. Interestingly, OWL values were sometimes higher among discarded items, suggesting that utility contribution alone is insufficient without corresponding support. MSU played a crucial role in evaluating utility overlaps and refining the candidate pool, especially in the recursive mining phase [14], [25], [27], [34], [39].

Sensitivity analysis further confirmed the dominant role of minimum utility thresholds in shaping performance outcomes. Experiments varying minutil from 10,000 to 100,000 while fixing minsup at 20 showed a clear inverse

correlation between utility thresholds and itemset count, execution time, and memory usage [6], [7], [11], [13], [15]. Higher minutil values resulted in fewer itemsets, faster processing, and more efficient memory consumption, demonstrating that utility-based filtering is more restrictive than frequency-based filtering. Conversely, increasing minsup while holding minutil constant had a more moderate effect, primarily influencing the diversity of itemset combinations [17], [18], [20], [32], [38].

The algorithm's scalability was assessed using transaction subsets corresponding to two-, three-, four-, and five-year intervals. Results show a near-linear increase in execution time and memory usage, validating the algorithm's capability to scale with growing data volumes [2], [5], [23], [31], [36]. While the number of itemsets grew significantly with larger datasets, reflecting the exponential nature of combinatorial mining, the pruning strategies ensured that performance degradation remained manageable. This confirms the suitability of the proposed algorithm for high-volume retail applications and business intelligence systems.

Despite these advantages, the study acknowledges certain limitations. Static threshold settings may not adapt well to dynamic transactional data where utility distributions evolve over time. Moreover, the evaluation on a single dataset may limit generalizability across different business domains such as healthcare, telecommunications, or logistics [7], [9], [12], [22], [44]. Future research could explore adaptive or context-aware thresholding mechanisms and real-time mining integrations to respond to shifting patterns. Evaluating the algorithm across diverse datasets and domains would also help validate its robustness and transferability, particularly in scenarios requiring fast and accurate recommendation or forecasting systems [3], [13], [21], [30], [46].

The optimized FCHUIM algorithm proposed in this study, enhanced with OSR, OWL, and MSU pruning heuristics, offers a significant advancement in utility-based pattern mining. It addresses the dual challenges of computational scalability and pattern relevance by focusing on high-utility and high-support itemsets with minimal redundancy. The empirical evaluation affirms its efficacy in handling real-world retail data, making it a promising solution for business intelligence, dynamic pricing, inventory planning, and personalized marketing initiatives.

3. Research Methodology

This section outlines the methodological framework adopted in the study, detailing the experimental design, data preprocessing stages, algorithm implementation procedures, and evaluation metrics. Each step is described to ensure the reproducibility and reliability of the conducted experiments.

3.1. Research Design

This study is designed as an experimental research aimed at optimizing the FCHUIM (Frequent Closed High Utility Itemset Mining) algorithm for retail transaction data. The primary goal is to reduce the search space and computational time without compromising the completeness and relevance of the discovered patterns.

The research process begins with data preprocessing, which includes data cleaning, transformation, and aggregation. The cleaned dataset is transformed into a structured format suitable for utility mining, with item utilities computed using item quantities and unit profits. Items that do not meet the predefined minimum utility threshold are eliminated early to reduce noise in the mining phase.

The core mining procedure involves implementing the FCHUIM algorithm enhanced with heuristic-based pruning techniques, specifically Observed Support Ratio (OSR), Observed Weighted Lift (OWL), and Minimum Suffix Utility (MSU). These techniques aim to eliminate unpromising itemsets by evaluating support, utility, and pruning thresholds before deeper recursive exploration.

The overall workflow of the algorithm is illustrated in Fig. 1. The process starts with a preprocessed dataset and proceeds through the following steps:

1. Support and Utility Filtering: Each item is evaluated based on its support (Sup) and utility (Util). Items with $\mathbf{Sup}(\mathbf{x}) < \mathbf{minsup}$ **or** $\mathbf{Util}(\mathbf{x}) < \mathbf{minutil}$ are discarded [16].
2. Heuristic Pruning Phase:

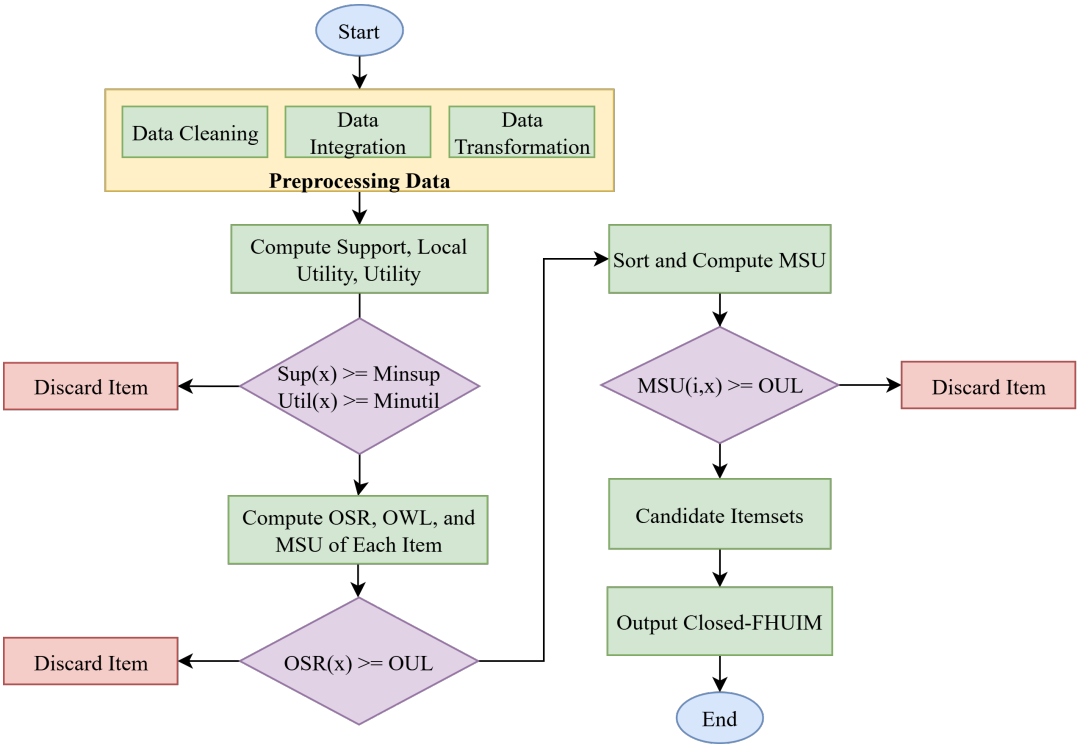


Fig. 1: Flowchart of the Closed-FHUIM algorithm with OSR, OWL, and MSU pruning.

- OSR Filtering: Items with OSR(x) < OUL (Optimal Utility Level) are excluded.
- OWL and TWU Calculation: Additional indicators such as OWL and TWU help guide the pruning and ordering phase[17].

3. MSU Evaluation: Items that pass the previous filters are ranked and evaluated based on their MSU scores. Only those with $\mathbf{MSU}(\mathbf{i}, \mathbf{ex}) \geq \mathbf{OUL}$ are considered as valid candidates [18].
4. Pattern Generation: The final stage involves generating Closed-FHUIs through recursive itemset evaluation [19].

This pipeline ensures that the resulting patterns are not only non-redundant and utility-relevant, but also discovered efficiently and scalably. The integration of pruning heuristics at multiple stages of the algorithm allows the system to focus computational resources only on meaningful combinations.

3.2. Data Preprocessing

The data preprocessing phase in this study consists of three main stages: data cleaning, data integration, and data transformation [20]. These steps are crucial to ensure that the dataset is accurate, consistent, and properly structured for utility-based pattern mining using the Closed-FHUIM algorithm [21].

3.2.1. Data Cleaning

Data cleaning is performed to remove irrelevant, duplicate, or incomplete entries that may interfere with the mining process. Unnecessary attributes that do not contribute to utility computation are filtered out. Duplicate transactions are eliminated to prevent double counting during itemset evaluation. Furthermore, transactions with missing or null values are excluded, as incomplete records can lead to incorrect utility calculations and pattern distortion [22].

3.2.2. Data Integration

This step merges two primary data sources, transactional sales records and product master data. The integration process ensures that each transaction is enriched with corresponding item attributes such as unit profit and category.

The resulting unified dataset serves as a complete and consistent input for the Closed-FHUIM algorithm. By linking product metadata with transactional logs, utility computations become accurate and meaningful [23].

3.2.3. Data Transformation

The transformation stage reshapes the data into a structure suitable for the utility mining process. Specifically, transaction aggregation is applied, where each transaction is summarized into a single row containing item identifiers and their corresponding quantities. This format facilitates the construction of utility lists and supports efficient traversal by the FHUIM algorithm. Additional derived metrics, such as item utility and transaction utility, are also computed during this phase and stored for subsequent use in mining and pruning [24].

3.3. Closed-FHUIM Algorithm

The Closed-Frequent and High Utility Itemset Mining (Closed-FHUIM) algorithm is designed to identify patterns in transactional data that fulfill two key characteristics: high frequency of occurrence (frequent) and significant contribution to total transaction utility (high utility) [7]. Furthermore, the algorithm focuses on generating closed itemsets, i.e., itemsets that do not have any supersets with identical support and utility values [10]. This approach not only addresses the limitations of both Frequent Itemset Mining (FIM) and High Utility Itemset Mining (HUIM), but also reduces redundancy and enhances pattern representation efficiency [25].

The implementation of Closed-FHUIM in this study is structured into three main stages: utility list construction, heuristic pruning techniques, and recursive itemset exploration. Each stage is described in detail below.

3.3.1. Utility List Construction

The initial step of Closed-FHUIM involves constructing a utility list for each individual item in the preprocessed dataset. A utility list is a structured data representation that stores transaction-wise utility information of items. Each entry in the utility list includes [26]:

- Transaction ID (*TID*): Unique identifier for each transaction.
- Internal Utility (*iutil*): The utility value of the item in a specific transaction, calculated as the product of quantity and unit profit.
- Remaining Utility (*rutil*): The estimated utility of the remaining items appearing after the current item in the sorted transaction.

This structure enables efficient pattern evaluation by avoiding repeated utility computation throughout recursive iterations.

3.3.2. Pruning Techniques: OSR, OWL, and MSU

To reduce computational complexity, the Closed-FHUIM algorithm incorporates three heuristic pruning strategies:

- Observed Support Ratio (OSR):
  The Observed Support Ratio (OSR(x)) is defined as the ratio of the support of item x to the total number of transactions. If OSR(x) < OUL, where OUL (Optimal Utility Level) is defined as *minutil / minsup*, the item is considered low-quality and pruned.[27].
- Observed Weighted Lift (OWL) and Transaction Weighted Utility (TWU):
  The *Observed Weighted Lift* ($OWL(x)$) measures the relative contribution of item x's utility to the total utility of the transactions it appears in. Low $OWL(x)$ values indicate weak utility contribution. The *Transaction Weighted Utility* (TWU) is also calculated to guide item ordering during the mining process [28].
- Modified Subtree Utility (MSU):
  For each promising prefix item i, its extensions ex (i.e., items that can be combined with i to form larger itemsets) are evaluated using *Modified Subtree Utility* ($MSU(i, ex)$). This metric estimates the combined utility of item i and its extension ex, including their remaining utility contributions in related transactions.

If $MSU(i, ex) \geq OUL$, the itemset $\{i, ex\}$ is retained as a valid candidate for further mining; otherwise, it is pruned.

These three pruning strategies collectively reduce the search space significantly while maintaining the quality of extracted patterns.

3.3.3. Recursive Itemset Exploration

The final phase of the algorithm involves recursively exploring candidate itemsets using a depth-first search (DFS) approach. Each candidate is evaluated based on two core criteria:

- Forward Checking: Ensures that the extended itemset still has potential to meet the minimum support and utility thresholds [28].
- Backward Checking: Verifies that the itemset is closed, meaning no superset exists with identical support and utility values.

Only itemsets satisfying both conditions are included in the final result set. This approach guarantees the discovery of non-redundant, representative, and high-utility closed patterns [29].

3.4. Evaluation Metrics

To objectively assess the performance and effectiveness of the Closed-FHUIM algorithm, both in its baseline form and with the applied pruning optimizations (OSR, OWL, and MSU), this study employs three primary evaluation metrics:

- Execution Time (in seconds): Measures the total duration required to complete the mining process from start to finish.
- Memory Usage (in megabytes): Refers to the peak memory consumption recorded during algorithm execution, serving as an indicator of space efficiency [30].
- Number of Itemsets Generated: Indicates how many closed high-utility itemsets are successfully identified, representing the selectivity and effectiveness of the mining approach [31].

These metrics were chosen to ensure a comprehensive evaluation that captures computational efficiency, memory performance, and result quality. The detailed parameter values and experimental environment are further elaborated in Section 4.1 (*Experimental Setup*).

4. Experiment and Discussion

This section presents and analyzes the experimental results obtained from the implementation of the Closed-FHUIM algorithm. The evaluation focuses on performance metrics such as execution time, memory usage, and the number of closed high utility itemsets generated. The impact of varying key parameters, optimization techniques, and dataset scale is also explored to assess the effectiveness and scalability of the proposed approach.

4.1. Experimental Setup

All experiments in this study were conducted on a machine with the following hardware specifications: Intel Core i7-1165G7 processor, 16 GB of main memory (RAM), and Windows 11 Pro 64-bit operating system. The Closed-FHUIM algorithm was implemented using Python version 3.10 and executed within the Jupyter Notebook environment. This platform was selected due to its support for interactive exploration, seamless integration of visualizations, and flexibility in parameter adjustment and result monitoring [35].

The dataset used in this research consisted of retail transaction data collected from a consumer cooperative, covering the period from January 2023 to December 2024. It comprised a total of 56,274 transactions and included 4,265 unique items. Each transaction recorded the quantity and unit price of purchased items, which were used to compute the utility values required for high utility itemset mining [36].

To evaluate the algorithm's performance under varying conditions, two key parameters were systematically varied:

- Minimum Support (*minsup*): 10, 15, 20, 25, and 30

Table 1: Experimental parameter configuration.

Parameter	Value Used
Minimum support (*minsup*)	20
Minimum utility (*minutil*)	50.000
Optimization Techniques	OSR, OWL, MSU
Evaluation Metrics	Execution Time, Memory Usage, Number of Itemsets

Table 2: Pruning effectiveness based on item selection status.

Selection of Status	Value Used	Percentage
Pruned	3.934	92.50%
Passed Pruning	318	7.50%
Total	4.252	100%

Table 3: Average OSR and OWL values by pruning outcome.

Item Group	Avg. OSR	Avg. OWL
Pruned	0.0017	0.1408
Passed Pruning	0.0110	0.0471

- Minimum Utility (*minutil*): Ranged from 10,000 to 100,000

Table 1 summarizes the experimental configuration, including the values used for *minsup* and *minutil*, the optimization techniques applied (OSR, OWL, and MSU), and the evaluation metrics considered during testing.

This configuration was chosen to simulate practical retail mining scenarios and assess the impact of heuristic pruning strategies under realistic utility constraints. By analyzing the execution time, memory efficiency, and number of generated itemsets, the effectiveness of the Closed-FHUIM algorithm with OSR, OWL, and MSU enhancements can be comprehensively evaluated [37].

4.2. Optimization Impact: OSR, OWL, MSU

The effectiveness of three heuristic pruning techniques, Observed Support Ratio (OSR), Observed Weighted Lift (OWL), and Minimum Shared Utility (MSU), was evaluated to determine their role in early filtering before the combinatorial mining process. The goal is to reduce the search space and enhance computational efficiency without compromising the quality of the output.

From a total of 4,252 unique items in the dataset, 3,934 items were successfully pruned during this early stage, leaving only 318 items as valid candidates for closed high utility itemset generation. This results in a pruning efficiency of 92.50%, as shown in Table 2, indicating that the heuristic filters effectively eliminate non-promising items before entering the mining stage.

A statistical analysis was also conducted on the average values of OSR and OWL between the two item groups. The average OSR value for pruned items was 0.0017, significantly lower than the average OSR of 0.0110 observed in items that passed the pruning stage. This indicates that OSR is highly sensitive and effective in identifying frequently occurring items with sufficient support.

Interestingly, the average OWL value was found to be higher for pruned items (0.1408) than those that passed (0.0471). This phenomenon reveals that a high utility contribution (lift) alone does not necessarily make an item relevant, especially if its actual support is insufficient. Therefore, the combination of OSR and OWL provides a balanced filter that considers both frequency and utility in early item selection.

The distribution of OSR values, illustrated in Table 3, further supports the effectiveness of this approach. Items that passed pruning exhibited higher and more evenly distributed OSR values, while pruned items were concentrated near zero. This clearly indicates that OSR is capable of performing aggressive yet selective pruning.

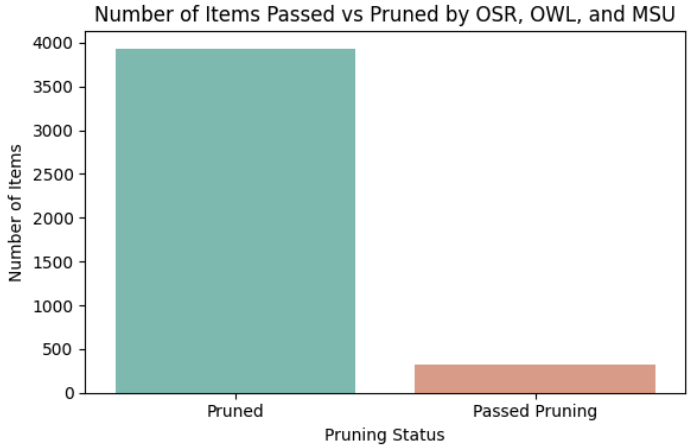


Fig. 2: Distribution of items retained vs. eliminated after pruning process.

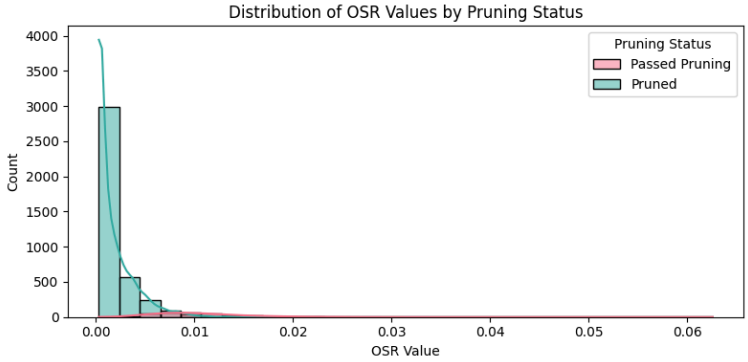


Fig. 3: Distribution of OSR values by pruning status.

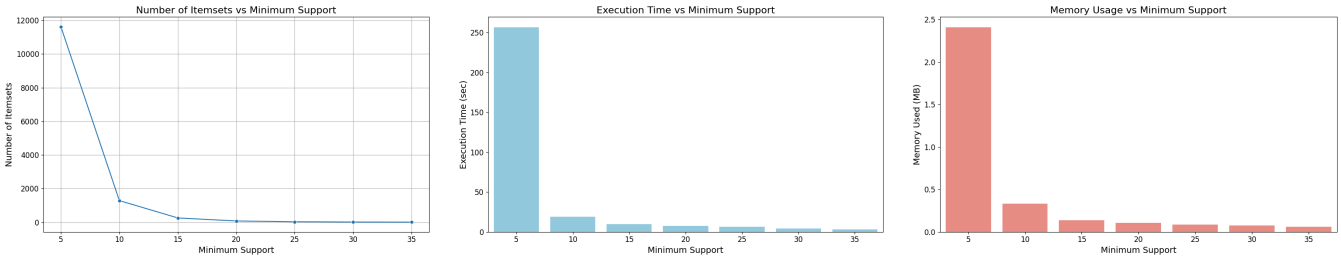
Fig. 2 illustrates the effectiveness of the heuristic pruning techniques (OSR, OWL, MSU) in reducing the search space. From a total of 4,252 individual items analyzed, 3,934 (approximately 92.5%) were successfully eliminated, leaving only 318 items (7.5%) that passed the pruning threshold. This significant reduction demonstrates the pruning mechanism's efficiency in early-stage filtering by removing low-utility or infrequent items before the itemset generation phase. Such aggressive filtering substantially minimizes computational overhead without compromising meaningful pattern discovery.

Fig. 3 shows the histogram illustrating the distribution of Observed Support Ratio (OSR) values for items grouped by their pruning status. Items that passed pruning exhibited a broader OSR range and higher average values, indicating their strong support within the dataset. Conversely, the OSR values of pruned items are tightly clustered near zero, reinforcing OSR's role as a sensitive indicator for identifying items with insufficient transactional support. This visual evidence supports the effectiveness of OSR in separating promising patterns from noisy data early in the mining process.

In conclusion, the integration of OSR, OWL, and MSU in the early phase of the FCHUIM algorithm significantly contributes to computational efficiency. These techniques enable early elimination of non-promising candidates, allowing the algorithm to focus on more relevant and potentially profitable itemsets.

4.3. Parameter Sensittivity Analysis.

The sensitivity analysis was conducted to evaluate the influence of two key parameters, minimum support (*minsup*) and minimum utility (*minutil*), on the performance of the FCHUIM algorithm. These parameters are fundamental in determining the search space and the resulting itemsets, as they control which patterns are considered relevant based on frequency and utility thresholds [35].



(a) (b) (c)

Fig. 4: (a) Number of itemsets, (b) execution time, and (c) memory usage.

Experiments were performed by independently varying each parameter and then assessing their combined effects. The goal was to observe how these thresholds influence three evaluation metrics: the number of closed itemsets generated, the algorithm's execution time, and memory consumption.

The variation in *minsup* aimed to assess the algorithm's responsiveness to changes in item frequency, while the *minutil* variation tested its sensitivity to the actual utility contribution of each itemset. The combined parameter analysis provided deeper insights into how these two interact in shaping algorithmic efficiency and output relevance [36].

4.3.1. Impact of Minimum Support

In the first experiment, the minimum support threshold (*minsup*) was varied from 5 to 35, while the minimum utility was kept constant at 50,000. This variation aimed to evaluate the effect of frequency filtering on the performance of the FCHUIM algorithm, focusing on the number of patterns generated, execution time, and memory usage.

As shown in Fig. 4(a), there is a clear inverse relationship between *minsup* and the number of itemsets discovered. At *minsup* = 5, the algorithm identified over 11,000 closed high-utility itemsets. However, increasing *minsup* to 30 or more drastically reduced the number of patterns to below 500. This steep decline reflects the natural pruning effect, where fewer item combinations meet the more stringent support threshold [37].

The reduction in itemset count significantly impacted the algorithm's execution time, as illustrated in Fig. 4(b). At the lowest support threshold, the execution time peaked at approximately 270 seconds. In contrast, *minsup* values of 20 and above resulted in dramatically shorter execution times, falling below 10 seconds, indicating that *minsup* is a crucial parameter for reducing the computational workload [40].

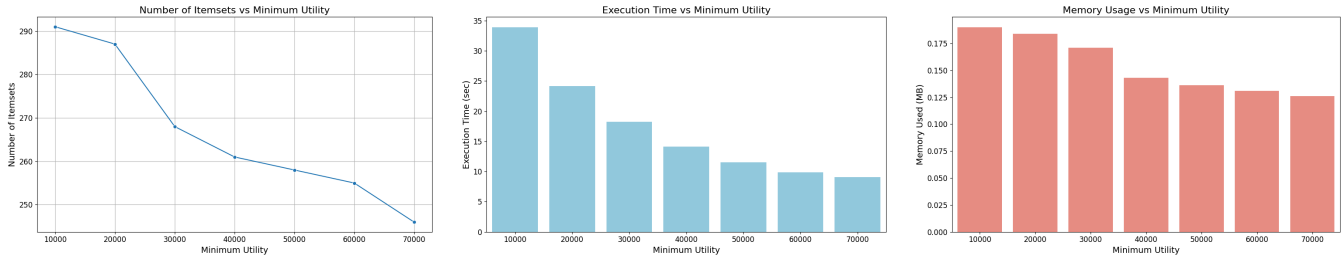
Furthermore, memory consumption also decreased alongside increasing *minsup* values, as depicted in Fig. 4(c). Memory usage dropped from about 2.4 MB at *minsup* = 5 to less than 0.2 MB for *minsup* ≥ 25. This trend suggests that fewer frequent itemsets reduce the internal memory structures required (e.g., utility lists and candidate sets), further enhancing the algorithm's space efficiency [39].

These findings affirm that the *minsup* parameter serves as an effective control mechanism to balance pattern completeness with computational efficiency. Higher *minsup* values are suitable for use cases that prioritize high-frequency patterns, such as identifying top-selling products or forming bundled offers in retail strategy planning [38].

4.3.2. Impact of Minimum Utility

The second sensitivity experiment evaluates the effect of varying the minimum utility (*minutil*) threshold on the performance of the FCHUIM algorithm. During this test, the minimum support value was fixed at 20, while *minutil* was varied from 10,000 to 70,000. This analysis aims to assess the algorithm's sensitivity to changes in utility constraints and its implications for both result quality and computational efficiency.

As shown in Fig. 5(a), increasing the *minutil* value significantly reduces the number of closed high-utility itemsets produced. The number of itemsets dropped consistently from 290 at *minutil* = 10,000 to 248 at *minutil*



(a) (b) (c)

Fig. 5: (a) Number of itemsets, (b) execution time, and (c) memory usage.

= 70,000. This reduction reflects a natural filtering process where only combinations with sufficiently high total utility remain. A higher *minutil* eliminates itemsets with marginal contributions, streamlining the output to focus on patterns that hold more substantial economic value [41].

The efficiency gains are also clearly observed in Fig. 5(b), where execution time sharply decreases as *minutil* increases. At a *minutil* value of 10,000, the algorithm required over 30 seconds, but at 70,000, execution time fell below 10 seconds. This improvement indicates that utility-based pruning, particularly through TWU (Transaction-Weighted Utility), effectively excludes non-promising itemsets early in the process, narrowing the search space and accelerating computation.

In addition, Fig. 5(c) illustrates a steady decline in memory usage as *minutil* increases. The memory consumed decreased from nearly 0.19 MB at lower thresholds to about 0.12 MB at higher ones. This efficiency is attributed to the reduced number of candidates and the more compact utility-list structures, which require less memory to process and store.

Overall, these results confirm that minimum utility is a highly influential parameter in optimizing both algorithmic output and resource utilization. Compared to minimum support, *minutil* plays a more dominant role in shaping the relevance of the resulting patterns while enhancing scalability. Setting a proper utility threshold is crucial in business scenarios where profitability takes precedence over frequency, such as identifying premium product bundles or high-margin cross-sell opportunities.

4.3.3. Impact of Combined Parameters

To evaluate the comprehensive effect of parameter interaction, a combined sensitivity analysis was performed using six distinct configurations of minimum utility (*minutil*) and minimum support (*minsup*). These two thresholds are critical in defining the algorithm's search space and the relevance of the resulting patterns, thus their joint variation provides a more realistic scenario for al1orithm deployment in practical applications.

Fig. 6 illustrates the algorithm's performance across all combinations. In Fig. 6(a), the highest number of closed high utility itemsets, 2,461 patterns, was observed at the lowest parameter setting (*minutil* = 10,000; *minsup* = 10). This indicates that a permissive threshold allows a wider search space, capturing many item combinations. In contrast, the tightest configuration (*minutil* = 100,000; *minsup* = 30) produced only 168 itemsets, demonstrating the strong filtering effect of strict parameter settings [42].

Execution time analysis shown in Fig. 6(b) reveals a parallel trend. The lowest threshold setting resulted in the longest execution time (92.29 seconds), due to the high volume of candidates processed. Meanwhile, the strictest combination sharply reduced execution time to 6.04 seconds. This improvement highlights the role of pruning mechanisms (such as TWU and closed-set validation) in reducing computational overhead through early-stage candidate elimination [44].

Fig. 6(c) shows a relatively stable trend in memory consumption. The memory usage decreased slightly from 5.875 MB in the most relaxed configuration to 5.254 MB in the strictest one. Although the difference appears minor, it still reflects a leaner utility list structure and a lower number of retained intermediate candidates in tighter settings [43].

Table 4: Parameter combination of *minutil* and *minsup*.

Minutil	***Minsup***	**Description**
10.000	10	Minimum Treshold
10.000	30	High Support
50.000	10	High Utility
50.000	30	Tight Treshold
100.000	10	Very High Utility
100.000	30	Extreme Tight Combination

(a) (b) (c)

Fig. 6: (a) Number of itemsets, (b) execution time, and (c) memory usage.

These results underscore the necessity of context-specific threshold tuning. Loose thresholds are suitable for exploratory mining where coverage is critical, but they come with higher computational cost. Conversely, tight thresholds are ideal for applications requiring concise and time-efficient outputs, such as real-time recommendation systems or large-scale retail analytics, where only the most frequent and high-contributing patterns are needed.

Table 4 provides a summary of the six parameter combinations used in the experiment. Each pair of *minutil* and *minsup* values is categorized with a brief descriptor to aid in interpreting their computational and analytical significance.

4.4. Scalability Evaluation

The scalability evaluation aims to assess the performance of the Closed-FHUIM algorithm when applied to datasets of increasing size. In this experiment, the dataset sizes were varied based on the transaction duration periods, specifically covering 2, 3, 4, and 5 years. This evaluation focuses on three main performance metrics: the number of closed high-utility itemsets generated, execution time, and memory usage [42].

As illustrated in Fig. 7(a), the number of itemsets increases significantly with the growth of dataset size. When the dataset duration extends from 2 to 5 years, the number of itemsets escalates from approximately 400 to over 14,000. This exponential increase indicates a direct correlation between dataset volume and the algorithm's search space. A larger dataset contributes to higher pattern diversity, which in turn amplifies the candidate itemsets explored during mining [44].

A similar trend is observed in Fig. 7(b) for execution time. As the dataset grows, the time required for mining also rises proportionally. For instance, while processing a 2-year dataset takes less than 50 seconds, executing the algorithm on a 5-year dataset demands over 700 seconds. This result reinforces the sensitivity of the Closed-FHUIM algorithm to the number of transactions, especially due to the overhead introduced during utility list construction and recursive candidate exploration [45].

In Fig. 7(c), memory consumption also demonstrates a linear upward trend. Memory usage increases from approximately 10 MB (2-year dataset) to more than 30 MB (5-year dataset). This growth is attributed to the larger utility lists and increased number of itemset combinations that must be maintained in memory during execution.

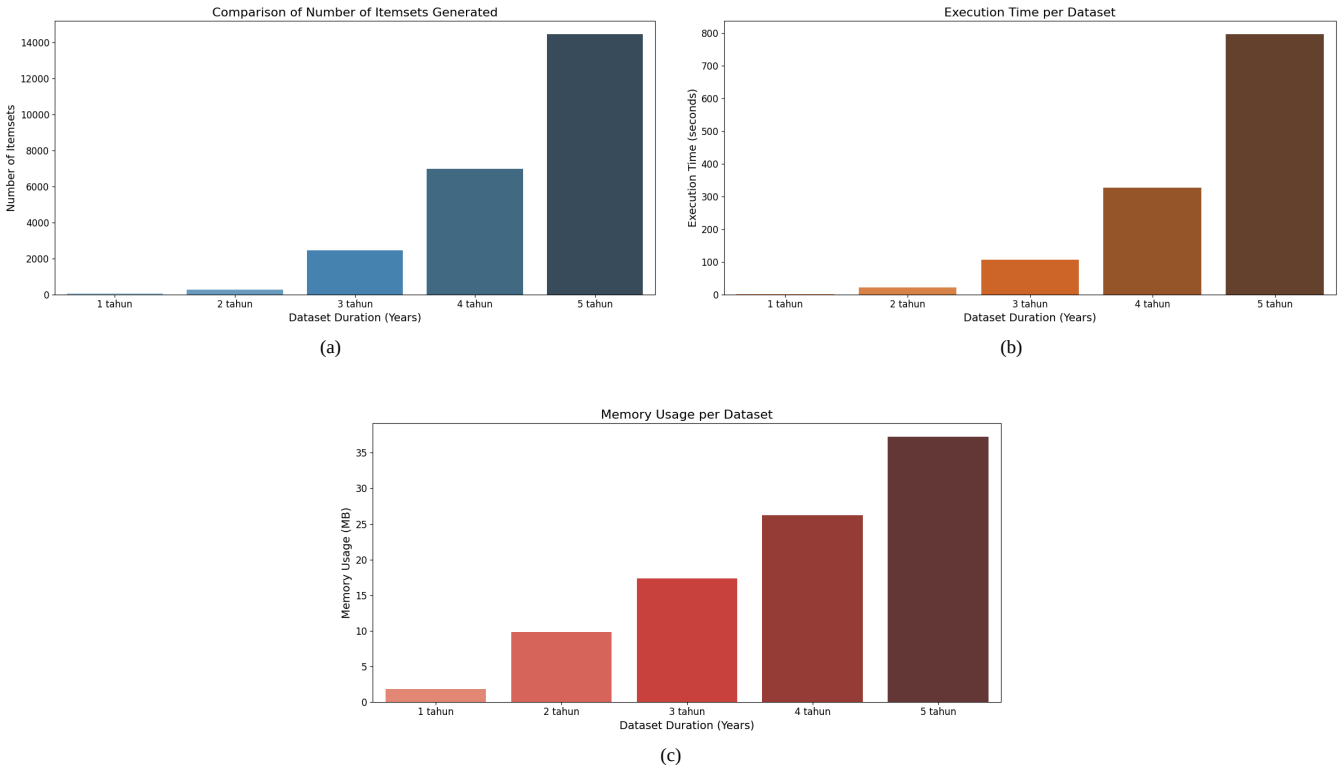


Fig. 7: (a) Number of itemsets, (b) execution time, and (c) memory usage.

These findings collectively suggest that the Closed-FHUIM algorithm maintains consistent scalability and is capable of handling medium to large-scale datasets efficiently [46]. However, when applied to industrial-scale datasets or high-frequency transactional environments, such as those involving millions of daily transactions, further enhancements may be necessary. These could include:

- Parallel processing for workload distribution
- Data segmentation or sharding to manage memory efficiently
- More aggressive pruning techniques to reduce computation overhead

Such strategies would ensure that performance remains optimal even under high data volume conditions.

5. Conclusion

This study presents an optimized variant of the FCHUIM algorithm by integrating heuristic-based pruning methods, OSR, OWL, and MSU, to improve mining efficiency on large-scale retail datasets. The application of these techniques effectively eliminated non-promising items early, achieving a pruning efficiency of 92.5% and significantly reducing execution time and memory usage. Among them, OSR demonstrated the strongest filtering capability for low-support items, while OWL offered insights into utility contribution. Sensitivity analysis revealed that the minimum utility threshold had a greater impact on performance than minimum support, and scalability testing confirmed the algorithm's stable growth pattern under increasing data volumes. These findings underscore the potential of the enhanced FCHUIM for practical use in transactional data mining, enabling faster, more focused discovery of high-value itemsets suitable for business applications such as marketing and inventory optimization.

However, this study has certain limitations. The use of fixed thresholds (minsup and minutil) across different data sizes may not fully reflect dynamic transaction environments where utility distributions change over time. Additionally, the algorithm was tested on a single retail dataset, which may limit generalizability to other domains with different transaction characteristics. Future research could explore adaptive thresholding techniques and real-time pattern mining integration to enhance responsiveness in evolving data streams. Moreover, evaluating the algorithm on more diverse datasets and applying the discovered patterns to downstream applications such as recommender systems could further validate its practical effectiveness.

CRediT Authorship Contribution Statement

K. S. Sulanjari: Writing – Review & Editing, Writing – Original Draft, Validation, Software, Methodology, Conceptualization. **C. Fatichah:** Writing – Original Draft, Formal analysis, Data Curation, Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request. Due to confidentiality agreements with the data provider (consumer cooperative), public sharing is restricted.

Declaration of Generative AI and AI-assisted Technologies in The Writing Process

The authors used generative AI to improve the writing clarity of this paper. They reviewed and edited the AI-assisted content and take full responsibility for the final publication.

References

[1] M. G. Dekimpe, 'Retailing And Retailing Research In The Age Of Big Data Analytics', *International Journal Of Research In Marketing*, Vol. 37, No. 1, Pp. 3–14, Mar. 2020, Doi: 10.1016/J.Ijresmar.2019.09.001.

[2] F. Olaoye, K. Potter, And B. Klinton, 'Big Data Analytics In Retail: Optimizing Inventory And Demand Forecasting'.

[3] Hana Bernika Sabila And Feri Candra, 'Implementation Of Apriori Algorithm For Data Mining On Sales Transaction Data', *Ijeepse*, Vol. 6, No. 3, Pp. 189–193, Oct. 2023, Doi: 10.31258/Ijeepse.6.3.189-193.

[4] S. Gupta And D. Ramachandran, 'Emerging Market Retail: Transitioning From A Product-Centric To A Customer-Centric Approach', *Journal Of Retailing*, Vol. 97, No. 4, Pp. 597–620, Dec. 2021, Doi: 10.1016/J.Jretai.2021.01.008.

[5] N. T. Tung, L. T. T. Nguyen, T. D. D. Nguyen, And B. Vo, 'An Efficient Method For Mining Multi-Level High Utility *Itemsets*', *Appl Intell*, Vol. 52, No. 5, Pp. 5475–5496, Mar. 2022, Doi: 10.1007/S10489-021-02681-Z.

[6] S. Kumar, R. K. Yadav, And C. Singh, 'Fuzzy Set-Based Inventory Model For Tpd Demand Under Effect Of Inflation And Carbon Emissions With Partial Backordering', *Journal Of Computational Analysis And Applications*, Vol. 33, No. 8, 2024.

[7] M. W. Ashraf, M. A. Naeem, And H. J. Lee, 'A Robust Technique For Closed Frequent And High Utility *Itemsets* Mining: Closed-Fhuim', *Ieee Access*, Vol. 12, Pp. 196517–196532, 2024, Doi: 10.1109/ACCESS.2024.3520619.

[8] H. Duong, H. Pham, T. Truong, And P. Fournier-Viger, 'Efficient Algorithms To Mine Concise Representations Of Frequent High Utility Occupancy Patterns', *Appl Intell*, Vol. 54, No. 5, Pp. 4012–4042, Mar. 2024, Doi: 10.1007/S10489-024-05296-2.

[9] X. Zhao, X. Zhong, And B. Han, 'Frequent Closed High-Utility *Itemset* Mining Algorithm Based On Leiden Community Detection And Compact Genetic Algorithm', *IEEE Access*, Vol. 12, Pp. 84763–84773, 2024, Doi: 10.1109/ACCESS.2024.3413966.

[10] Y. In Chang, P. Chun Chuang, C. Yu Wang, And Y. Hao Liao, 'An Efficient Approach For Mining Closed High Utility Patterns In The Incremental Database', *IJFCC*, Pp. 84–92, Dec. 2023, Doi: 10.18178/Ijfcc.2023.12.4.608.

[11] A. Borah And B. Nath, 'Comparative Evaluation Of Pattern Mining Techniques: An Empirical Study', *Complex Intell. Syst.*, Vol. 7, No. 2, Pp. 589–619, Apr. 2021, Doi: 10.1007/S40747-020-00226-4.

[12] Bijith Marakarkandy, 'Enhancing Multi-Channel Consumer Behavior Analysis: A Data-Driven Approach Using The Optimized Apriori Algorithm', *Jes*, Vol. 20, No. 2s, Pp. 700–708, Apr. 2024, Doi: 10.52783/Jes.1536.

[13] J. M.-T. Wu, R. Li, M.-E. Wu, And J. C.-W. Lin, 'Mining Skyline Frequent-Utility Patterns From Big Data Environment Based On Mapreduce Framework', *IDA*, Vol. 27, No. 5, Pp. 1359–1377, Oct. 2023, Doi: 10.3233/IDA-220756.

[14] P. Amaranatha Reddy And M. Hazarath Murali Krishna Prasad, 'High Utility Item-Set Mining From Retail Market Data Stream With Various Discount Strategies Using EGUI-Tree', *J Ambient Intell Human Comput*, Vol. 14, No. 2, Pp. 871–882, Feb. 2023, Doi: 10.1007/S12652-021-03341-3.

[15] R. Gajera, S. Patel, K. Madhani, And A. Solanki, 'An Efficient Join Operations For Utility List-Based High-Utility Mining Approaches Using Hybrid Search Technique', *Int J Data Sci Anal*, Apr. 2024, Doi: 10.1007/S41060-024-00538-5.

[16] J. He, X. Han, X. Wan, And J. Wang, 'Efficient Skyline Frequent-Utility *Itemset* Mining Algorithm On Massive Data', *IEEE Trans. Knowl. Data Eng.*, Vol. 36, No. 7, Pp. 3009–3023, Jul. 2024, Doi: 10.1109/TKDE.2024.3349454.

[17] M. Touqeer, S. Al Sulaie, S. A. Lone, K. Shaheen, N. M. Gunaime, And M. A. Elkotb, 'A Fuzzy Parametric Model For Decision Making Involving F-OWA Operator With Unknown Weights Environment', *Heliyon*, Vol. 9, No. 9, P. E19969, 2023, Doi: 10.1016/J.Heliyon.2023.E19969.

[18] T. D. D. Nguyen, N. T. Tung, L. T. T. Nguyen, T. T. Pham, And B. Vo, 'MLC-Miner: Efficiently Discovering Multi-Level Closed High Utility Patterns From Quantitative Hierarchical Transaction Databases', *Expert Systems With Applications*, Vol. 254, P. 124383, Nov. 2024, Doi: 10.1016/J.Eswa.2024.124383.

[19] A. Hidouri, S. Jabbour, And B. Raddaoui, 'On The Enumeration Of Frequent High Utility *Itemsets*: A Symbolic AI Approach', *Lipics, Volume 235, CP 2022*, Vol. 235, P. 27:1-27:17, 2022, Doi: 10.4230/LIPICS.CP.2022.27.

[20] M. H. Malik, H. Ghous, M. Ismail, S. Jamshaid, And J. Altaf, 'Market Basket Analysis For Next Basket Item Prediction Using Data Mining And Machine Learning', *Biomedical Informatics*.

[21] I. W. P. Pratama, 'Exploring The Depths Of Market Basket Analysis: A Comprehensive Guide To Transaction Analysis With FP-Growth And Apriori Algorithms', *INVOTEK*, Vol. 23, No. 2, Pp. 109–118, Jan. 2024, Doi: 10.24036/Invotek.V23i2.1094.

[22] Andy Hermawan, Bayu Wicaksono, Tigfhar Ahmadjayadi, Bagas Surya Prakasa, And Jasico Dacomoro Aruan, 'Implementasi Algoritma Apriori Pada Market Basket Analysis Terhadap Data Penjualan Produk Supermarket', *Algoritma*, Vol. 2, No. 5, Pp. 95–105, Jun. 2024, Doi: 10.62383/Algoritma.V2i5.137.

[23] L. Samboteng, R. Rulinawaty, K. Rachmat, M. Basit, And R. Rahim, 'Market Basket Analysis Of Administrative Patterns Data Of Consumer Purchases Using Data Mining Technology', *J Appl Eng Science*, Vol. 20, No. 2, Pp. 339–345, 2022, Doi: 10.5937/Jaes0-32019.

[24] K. Mallikharjuna Rao, G. Saikrishna, And K. Supriya, 'Data Preprocessing Techniques: Emergence And Selection Towards Machine Learning Models - A Practical Review Using HPA Dataset', *Multimed Tools Appl*, Vol. 82, No. 24, Pp. 37177–37196, Oct. 2023, Doi: 10.1007/S11042-023-15087-5.

[25] J. Chen *Et Al.*, 'Incremental High Average-Utility *Itemset* Mining: Survey And Challenges', *Sci Rep*, Vol. 14, No. 1, P. 9924, Apr. 2024, Doi: 10.1038/S41598-024-60279-0.

[26] G. N. Sowjanya And M. B. Reddy, 'Utility List-Based Mining And Recurrent Neural Network for Utility *Itemset* Mining Based Transactional Data', . *Vol.*, No. 24.

[27] H. Kim *Et Al.*, 'Efficient Method For Mining High Utility Occupancy Patterns Based On Indexed List Structure', *IEEE Access*, Vol. 11, Pp. 43140–43158, 2023, Doi: 10.1109/ACCESS.2023.3271864.

[28] L. Yin *Et Al.*, 'Outlier Weighed Layerwise Sparsity (OWL): A Missing Secret Sauce For Pruning Llms To High Sparsity', May 06, 2024, *Arxiv*: Arxiv:2310.05175. Doi: 10.48550/Arxiv.2310.05175.

[29] S. Siva And S. Chaudhari, 'Cumulative Summary List Driven Lightweight Frequent Closed High Utility *Itemset* Mining', In *2023 2nd International Conference For Innovation In Technology (INOCON)*, Bangalore, India: IEEE, Mar. 2023, Pp. 1–6. Doi: 10.1109/INOCON57975.2023.10101053.

[30] M. R. N. Koppar, 'Closed Frequent Pattern Mining Algorithms: Features & Issues', Vol. 6, No. 1, 2020.

[31] F. Z. Lebbah, M. Larbani, And A. Rahmoun, 'Closed *Itemset* Mining: A Graph Theory Perspective', *IJCAI*, Vol. 48, No. 8, May 2024, Doi: 10.31449/Inf.V48i8.5480.

[32] Parsia, B., Matentzoglu, N., Gonçalves, R. S., Glimm, B., & Steigmiller, A. (2016, September). The OWL reasoner evaluation (ORE) 2015 resources. In *International Semantic Web Conference* (pp. 159-167). Cham: Springer International Publishing. Doi: 10.1007/978-3-319-46547-0_17.

[33] Bou, S., Miwa, M., & Sasaki, Y. (2024). Two evaluations on Ontology-style relation annotations. *Computer Speech & Language*, *84*, 101569. Doi: 10.1016/j.csl.2023.101569.

[34] Başar, M. S., Çağlayan, B. S., & Aksoylu, A. E. (2018). A study on catalytic hydrogen production: Thermodynamic and experimental analysis of serial OSR-PROX system. *Fuel Processing Technology*, *178*, 301-311. Doi: 10.1016/j.fuproc.2018.09.011.

[35] Hegewald, H., Wensch-Dorendorf, M., Sieling, K., & Christen, O. (2018). Impacts of break crops and crop rotations on oilseed rape productivity: A review. *European journal of agronomy*, *101*, 63-77. Doi: 10.1016/j.eja.2018.08.003.

[36] Yang, Z., Chen, Z., Lee, K., Owens, E., Boufadel, M. C., An, C., & Taylor, E. (2021). Decision support tools for oil spill response (OSR-DSTs): Approaches, challenges, and future research perspectives. *Marine Pollution Bulletin*, *167*, 112313. Doi: 10.1016/j.marpolbul.2021.112313.

[37] Zhang, X., Wang, W., Zhang, Y., & Gu, X. (2024). Research on hydration characteristics of OSR-GGBFS-FA alkali-activated materials. *Construction and Building Materials*, *411*, 134321. Doi: 10.1016/j.conbuildmat.2023.134321.

[38] Li, B., Khan, S., Salata, K., Hussain, M. A., de Mestral, C., Greco, E., … & Al-Omran, M. (2019). A systematic review and meta-analysis of the long-term outcomes of endovascular versus open repair of abdominal aortic aneurysm. *Journal of vascular surgery*, *70*(3), 954-969. Doi: 10.1016/j.jvs.2019.01.076.

[39] Torgalsbøen, A. K., Mohn, C., Larøi, F., Fu, S., & Czajkowski, N. (2023). A ten-year longitudinal repeated assessment study of cognitive improvement in patients with first-episode schizophrenia and healthy controls: The Oslo Schizophrenia Recovery (OSR) study. *Schizophrenia Research*, *260*, 92-98. Doi: 10.1016/j.schres.2023.08.008.

[40] Mishra, S., Sridhara, N., Mitra, A., Yougandar, B., Dash, S. K., Agarwal, S., & Dey, A. (2017). CO2 laser cutting of ultra thin (75 µm) glass based rigid optical solar reflector (OSR) for spacecraft application. *Optics and Lasers in Engineering*, *90*, 128-138. Doi: 10.1016/j.optlaseng.2016.10.007.

[41] Lin, J. C. W., Djenouri, Y., Srivastava, G., Yun, U., & Fournier-Viger, P. (2021). A predictive GA-based model for closed high-utility itemset mining. *Applied Soft Computing*, *108*, 107422. Doi: 10.1016/j.asoc.2021.107422.

[42] Vo, B., Nguyen, L. T., Bui, N., Nguyen, T. D., Huynh, V. N., & Hong, T. P. (2020). An efficient method for mining closed potential high-utility itemsets. *IEEE Access*, *8*, 31813-31822. Doi: 10.1109/ACCESS.2020.2974104.

[43] Ashraf, M. W., Naeem, M. A., & Lee, H. J. (2024). A Robust Technique for Closed Frequent and High Utility Itemsets Mining: Closed-FHUIM. *IEEE Access*.

[44] Fournier-Viger, P., Chun-Wei Lin, J., Truong-Chi, T., & Nkambou, R. (2019). A survey of high utility itemset mining. In *High-utility pattern mining: Theory, algorithms and applications* (pp. 1-45). Cham: Springer International Publishing. Doi: 10.1007/978-3-030-04921-8_1.

[45] Lin, J. C. W., Djenouri, Y., & Srivastava, G. (2021). Efficient closed high-utility pattern fusion model in large-scale databases. *Information Fusion*, *76*, 122-132. Doi: 10.1016/j.inffus.2021.05.011.

[46] Vlashejerdi, M. N., & Daneshpour, N. (2025). Efficient mining of closed high-utility itemsets in dynamic and incremental databases. *Engineering Applications of Artificial Intelligence*, *144*, 110081. Doi: 10.1016/j.engappai.2025.110081.