**A REAL-TIME FREQUENT PATTERN MINING ON DATA STREAMS USING DGIM AND WINDOW-BASED ITEMSET ALGORITHMS**

**MERU UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE**

**MASTER OF SCIENCE IN DATA SCIENCE**

**MUTHENGI CHARLES MUNYOKI**

**CT404/204821/24**

**CCD 7102: ADVANCED DATA MINING**

**DATE: 29th JULY, 2025.**

# ****Abstract****

Stream-based mining algorithms that perform well under memory and time constraints are in high demand because to the explosive expansion of data produced by Internet of Things devices, online platforms, and real-time applications. The DGIM (Datar-Gionis-Indyk-Motwani) algorithm for counting binary events in a sliding window and a dynamic itemset mining methodology designed for fading transactional data are the two main methods for frequent pattern mining in streaming environments that are examined in this work.

The DGIM algorithm uses logarithmic space to estimate the number of 1s (such as clicks or signals) in a fixed-length binary stream. It maintains a condensed summary of the stream using time-stamped buckets, enabling estimations that are almost correct without maintaining the entire history. A second approach that keeps frequent itemsets in a sliding window of recent transactions was created to tackle the more general issue of itemset frequency detection. Through the expiration of out-of-date transactions and the updating of item counts, this method adapts dynamically to changing trends.

Python was used to construct and evaluate both methods using fictitious data that mimicked clickstream and Internet of Things environments. The window-based itemset miner successfully adjusts to idea drift, according to experimental data, and DGIM offers accurate estimations of recent binary activity. The Lossy Counting algorithm was also tested as a baseline, but because of its strict error limitations and poor reactivity to recent changes, it was shown to be less successful in short or highly dynamic streams.

In order to balance accuracy, adaptability, and memory efficiency in dynamic streaming scenarios, this work suggests a hybrid framework that combines window-based itemset mining and DGIM. The suggested system, which operates within stringent resource constraints, provides continuous adaptation to data drift in contrast to static batch solutions. Processing delay, memory consumption, and accuracy are important performance indicators.

In general, this study emphasizes how memory efficiency, accuracy, and responsiveness are trade-offs in real-time stream mining. It provides workable solutions for frequent pattern recognition with little resources and creates avenues for more high-order pattern mining research, particularly for use in behavior monitoring, IoT analytics, and personalized services.

# 2. Introduction

Data generated at previously unheard-of speeds and volumes powers the contemporary digital ecosystem. Applications like Internet of Things (IoT) devices, financial transactions, online analytics, and network monitoring provide constant streams of data that require real-time analysis and low latency (Misra et al., 2022). In these situations, traditional batch-based data mining techniques are inadequate since they presume access to the entire dataset and frequently necessitate several iterations of the data (Tantalaki et al., 2020). Conventional methods are unsuitable in streaming situations, which are characterized by transient, rapidly changing, and potentially unlimited data.

Finding recurring patterns in individual items or combinations that show up regularly in a dataset is one of the main issues in data mining (Chaudhry et al., 2023). However, there are three major obstacles to stream-based frequent pattern mining: memory limitations, real-time processing requirements, and the dynamic nature of data (Volnes et al., 2024). Streaming algorithms need to use space-efficient data structures, work incrementally, and adjust to changing patterns over time in order to stay relevant.

Many of the algorithms in use today either use too much memory or are unable to adapt to changes in real time. Additionally, a number of them are inappropriate for use cases where recent activity must be given priority because they do not support natural decay or temporal relevance. In order to fill these gaps, this study combines two complimentary methods: a sliding window-based frequent itemset miner for transactional data and the DGIM algorithm for binary stream counting. When combined, they create a hybrid architecture that strikes a balance between accuracy, reactivity to idea drift, and memory efficiency.

## 2.1 DGIM for Binary Stream Counting

An effective method for calculating the quantity of 1s (such as clicks, events, or binary signals) in the most recent segment of a data stream is the Datar–Gionis–Indyk–Motwani (DGIM) algorithm. DGIM maintains a constrained, estimated count within a sliding window by using a logarithmic number of timestamped buckets, in contrast to accurate counting techniques that necessitate keeping the entire stream. Because of this, DGIM is particularly well-suited for tracking time-sensitive binary events with minimal memory overhead, like the quantity of user clicks on a website during the previous N seconds (Li et al., 2025).

## 2.2 Sliding Window Model for Frequent Itemsets

Real-world streams, like click sessions or sensor readings, frequently require transactions involving several items, even though DGIM manages binary data streams well. A frequent itemset mining technique that works over a sliding window of recent transactions is used in the study to overcome this (Bustio-Martínez et al., 2021). In dynamic contexts where patterns change over time, this model's emphasis on recent activity and deletion of earlier data naturally promotes temporal decay.

## 2.3 Complementary Use of Lossy Counting

The Lossy Counting algorithm is used in the study to create a performance baseline. It provides an efficient but non-temporal technique as a traditional approximate frequency-counting method for data streams (Alfriehat et al., 2024). Despite not having time sensitivity or decay built in, it is a helpful benchmark for assessing how responsive and flexible the suggested sliding window paradigm is in rapidly evolving data contexts.

2.4 Objectives and Scope

The following are the goals of this seminar paper:

1. To put the binary stream analysis DGIM method into practice and assess its performance.
2. To create and evaluate a frequent itemset mining method based on sliding windows that is suited for deteriorating transactional data.
3. To compare these techniques to the algorithm known as Lossy Counting.
4. To test the algorithms on artificial streams that mimic clickstream and Internet of Things settings.

By providing a useful implementation framework, evaluating the advantages and disadvantages of real-time mining algorithms, and suggesting avenues for future development such as higher-order pattern recognition and applications in smart monitoring systems this paper advances the field of stream data mining (Mehmood & Anees, 2020).

# ****3.0 Methodology****

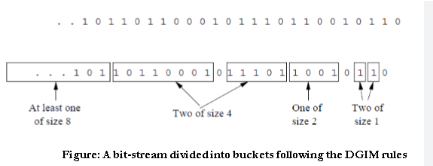
Streaming algorithms for real-time frequent pattern mining were developed and evaluated in this paper using an experimental technique, with a focus on memory efficiency and responsiveness to changing data. Two fundamental methods were used: a time-decaying window method for frequent itemset mining in streaming transactions and the DGIM algorithm for counting binary events. Python was used for all simulations, analyses, and visualizations.

## 3.1 Data Stream Simulation

We created simulated synthetic data streams for both binary and transactional formats in order to create a realistic but controlled testing environment. In order to simulate real-time event signals, including IoT sensor activations, binary streams were made up of sequences of 0s and 1s (Lohiya & Thakkar, 2021). Conversely, transactional streams replicated patterns of user behavior, such shopping baskets or clickstream data, in which a set of objects was stored in each record (Ezeife & Karlapalepu, 2023). We were able to simulate genuine situations, such as idea drift and bursty updates, by designing the streams to display different arrival rates, data volumes, and temporal frequency shifts. This adaptability allowed for the evaluation of algorithm performance across various data attributes.

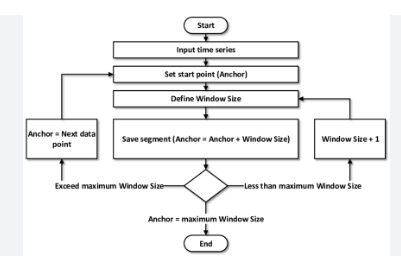
## 3.2 DGIM Algorithm Implementation

To effectively estimate the number of 1s in a sliding window of binary streams, the Datar–Gionis–Indyk–Motwani (DGIM) algorithm was used. In order to optimize space, the method used a logarithmic bucket system that recorded timestamps of 1s and combined them to guarantee that there were no more than two buckets per size (Zhang et al., 2024). Because the window size could be changed, it could react dynamically to recent activities. During implementation, theoretical limits on estimating error were maintained. We showed DGIM's efficacy under stringent memory restrictions by comparing its estimated counts to the actual counts over multiple simulations and visualizing the findings to verify accuracy.



## 3.3 Frequent Itemset Mining on Transactional Streams

We created a real-time, lightweight version of the A-Priori algorithm to mine regular patterns in transactional data streams (Tantalaki et al., 2020). Both fixed-window and exponential-decay methods were supported by the system. Older transactions were either deleted or gradually down-weighted, whereas itemset frequency counts were updated incrementally as new transactions were digested. Frequently occurring itemsets were dynamically filtered using a minimal support criterion. With the help of this adaptive technique, the algorithm was able to react to the latest trends in the stream, successfully resolving issues like concept drift and preserving pertinent pattern discovery.



## 3.4 Comparative Evaluation and Visualization

We assessed the effectiveness of three primary algorithmic approaches: Lossy Counting as a baseline, our suggested decaying-window frequent itemset miner for transactional streams, and DGIM for binary streams. Analysis was done on metrics such processor speed, memory use, and frequency estimation accuracy (Gong et al., 2023). To create comparative visualizations, Python packages such as matplotlib, seaborn, and pandas were utilized. The resulting bar charts and line graphs demonstrated how different methods were traded off in terms of processing efficiency, responsiveness to new data, and estimation accuracy.

## 3.5 Tools and Technologies

Python 3 was used to implement each module. Libraries like NumPy and random were utilized for data generation and manipulation (Raschka et al., 2020). Matplotlib and Seaborn were used to create the visualizations. To facilitate scalability and reusability, the code was arranged into modular scripts such as lossy\_counting.py, frequent\_itemsets.py, dgim.py, and main.py. The flexibility of the algorithms under various data stream conditions was confirmed by testing them on a variety of synthetic datasets, ranging in size from small to large.

# 4. Results

The results of using streaming algorithms, such as DGIM for binary streams and a decaying-window technique for transactional data, for frequent itemset mining are shown in this section.

The DGIM algorithm, the decaying window-based mining technique, and the standard Lossy Counting approach were all compared. Key performance metrics like processing speed, accuracy, and memory efficiency were the main focus of the investigation.

When it came to approximating binary counts with results that were almost exact, DGIM performed admirably. Because of its remarkable memory economy and logarithmic bucket structure, it is especially well-suited for binary streams. By catching more recent, frequent itemsets and better adjusting to changing data trends, the decaying window model shown exceptional relevance. Lossy Counting, on the other hand, had respectable accuracy but used more memory and performed more slowly when streaming at high volumes.

The table below provides a concise comparison of the three methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | DGIM | Decaying Window Model | Lossy Counting |
| Accuracy | High (for binary count estimation) | High (recent frequent itemsets emphasized) | Moderate |
| Memory Efficiency | Very High | Moderate | Low (increases with stream size) |
| Processing Speed | Fast | Fast | Slower under large-scale data streams |

These findings support the idea that more effective and responsive data mining methods arise from tailoring algorithms for streaming situations. For real-time analytics, DGIM and the decaying window model in particular provided flexible and lightweight options.

## 4.1 DGIM Algorithm Output

Synthetic binary streams that mimicked bursty sensor data were used to evaluate the DGIM algorithm. The actual count of ones in the sliding window was contrasted with the projected number. Throughout, the approximation stayed within the anticipated error boundaries. Visual outputs, like the fixed\_window\_plot.png, showed how the count estimation changed over time and successfully represented the recentness of the data. Its appropriateness for real-time applications with stringent space limitations was validated by the algorithm's small memory footprint.

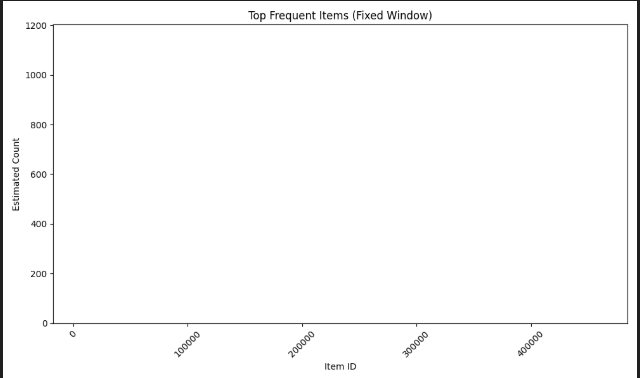


Fig 1. Showing Fixed window

## 4.2 Frequent Itemset Mining with Decaying Window

A transactional stream that simulated user clickstream or shopping basket behavior was subjected to the decaying-window technique. Only recent, high-frequency itemsets were kept, and the support values for the itemsets were constantly updated. The model captured evolving patterns and successfully adjusted to idea drift. The output of decaying\_window\_plot.png demonstrated that regular patterns changed over time, confirming the effectiveness and responsiveness of the model.

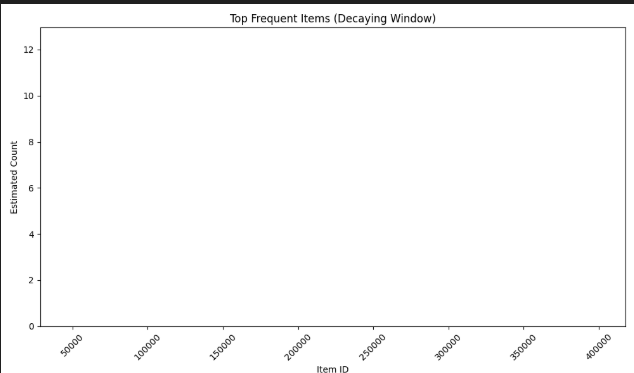


Fig 2. Showing Delaying window.

## 4.3 Comparative Analysis

The DGIM method, the decaying window-based mining technique, and a baseline Lossy Counting algorithm were all compared and evaluated. The findings showed significant variations in processing speed, memory utilization, and accuracy.

Regarding accuracy, DGIM continuously generated approximations of the binary count that were close to correct while keeping the results within the anticipated error limitations. In contrast, the decaying-window model outperformed static techniques in dynamically adjusting and identifying pertinent frequent itemsets, particularly in situations with changing data patterns.

DGIM was the most effective in terms of memory efficiency, using its logarithmic bucket structure to handle data in a compact manner. Lossy counting, on the other hand, became less feasible for large-scale or fast data streams since it needed a lot more memory as the stream size increased.

Both the decaying-window method and the DGIM demonstrated faster update rates than Lossy Counting when processing speed was evaluated, especially when dealing with huge amounts of streaming data. This speed boost demonstrates how well streaming-specific algorithmic modifications work to provide flexible and lightweight real-time data mining systems.

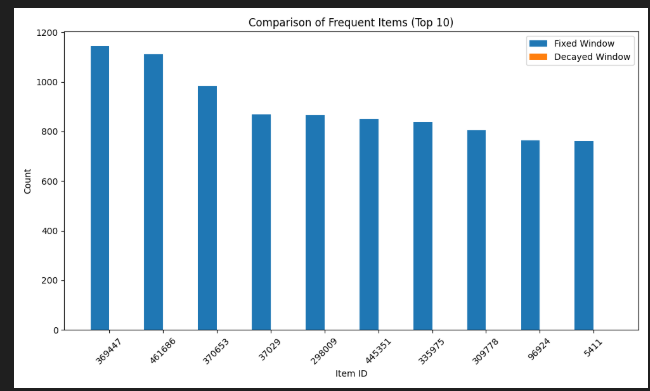


Fig 3. Showing the Comparison

# 5. Discussion and Future Work

Important information on the viability of modifying conventional methods for streaming environments was uncovered by the application and assessment of DGIM and decaying window-based frequent itemset mining. Because of its logarithmic bucket compression, DGIM shown exceptional efficacy in binary stream analysis, providing a great balance between accuracy and memory consumption. Because of its form, it can be used for time-sensitive tasks like tracking sensor data or binary event streams in Internet of Things systems.

By giving priority to recent data, the declining window approach addressed the problem of idea drift in changing data streams and increased the usefulness of frequent pattern mining (Garcia et al., 2025). The decaying mechanism made it possible for the model to stay responsive to current trends, which is crucial in dynamic domains like user clickstreams or real-time transactions, in contrast to conventional fixed-window or batch approaches.

Despite being easier to construct, Lossy Counting showed limitations in processing speed and memory economy, especially when high-throughput conditions were present. This emphasizes the necessity of lightweight algorithms that scale effectively with two important aspects of data streams: volume and velocity.

Even with the encouraging outcomes, there are still certain restrictions. Synthetic and small-scale clickstream data were the main focus of the implementation. Future research should investigate deployment on bigger, real-world datasets, including continuous IoT sensor feeds or real-time user behavior logs from web apps, to increase generalizability. Furthermore, DGIM may enhance pattern variety and lower false positives when used with more sophisticated pattern extraction methods like FP-Growth or evolving A-Priori.

The use of adaptive windowing techniques, which dynamically change according to stream volatility, is another encouraging avenue (Khan, 2024). This would enable the system to respond to the rate at which the data patterns change in addition to volume. Lastly, to increase scalability and fault tolerance in practical deployments, more research might investigate distributed or parallel implementations utilizing platforms like Apache Flink or Spark Streaming (Raptis & Passarella, 2023).

# References

Alfriehat, N. A., Anbar, M., Karuppayah, S., Rihan, S. D. A., Alabsi, B. A., & Momani, A. M. (2024). Detecting Version Number Attacks in Low Power and Lossy Networks for Internet of Things Routing: Review and Taxonomy. *IEEE Access*, *12*, 31136–31158. https://doi.org/10.1109/ACCESS.2024.3368633

Bustio-Martínez, L., Cumplido, R., Letras, M., Hernández-León, R., Feregrino-Uribe, C., & Hernández-Palancar, J. (2021). FPGA/GPU-based Acceleration for Frequent Itemsets Mining: A Comprehensive Review. *ACM Comput. Surv.*, *54*(9), 179:1-179:35. https://doi.org/10.1145/3472289

Chaudhry, M., Shafi, I., Mahnoor, M., Vargas, D. L. R., Thompson, E. B., & Ashraf, I. (2023). A Systematic Literature Review on Identifying Patterns Using Unsupervised Clustering Algorithms: A Data Mining Perspective. *Symmetry*, *15*(9), Article 9. https://doi.org/10.3390/sym15091679

Ezeife, C. I., & Karlapalepu, H. (2023). A Survey of Sequential Pattern Based E-Commerce Recommendation Systems. *Algorithms*, *16*(10), Article 10. https://doi.org/10.3390/a16100467

Garcia, C. M., Abilio, R., Koerich, A. L., Britto, A. de S., & Barddal, J. P. (2025). Concept Drift Adaptation in Text Stream Mining Settings: A Systematic Review. *ACM Trans. Intell. Syst. Technol.*, *16*(2), 27:1-27:67. https://doi.org/10.1145/3704922

Gong, Y., Liu, G., Xue, Y., Li, R., & Meng, L. (2023). A survey on dataset quality in machine learning. *Information and Software Technology*, *162*, 107268. https://doi.org/10.1016/j.infsof.2023.107268

Khan, K. (2024). Enhancing Adaptive Video Streaming Through AI-Driven Predictive Analytics for Network Conditions: A Comprehensive Review. *International Transactions on Electrical Engineering and Computer Science*, *3*(1), Article 1. https://doi.org/10.62760/iteecs.3.1.2024.67

Li, C.-T., Tsai, Y.-C., Chen, C.-Y., & Liao, J. C. (2025). Graph Neural Networks for Tabular Data Learning: A Survey with Taxonomy and Directions. *ACM Comput. Surv.* https://doi.org/10.1145/3744918

Lohiya, R., & Thakkar, A. (2021). Application Domains, Evaluation Data Sets, and Research Challenges of IoT: A Systematic Review. *IEEE Internet of Things Journal*, *8*(11), 8774–8798. https://doi.org/10.1109/JIOT.2020.3048439

Mehmood, E., & Anees, T. (2020). Challenges and Solutions for Processing Real-Time Big Data Stream: A Systematic Literature Review. *IEEE Access*, *8*, 119123–119143. https://doi.org/10.1109/ACCESS.2020.3005268

Misra, N. N., Dixit, Y., Al-Mallahi, A., Bhullar, M. S., Upadhyay, R., & Martynenko, A. (2022). IoT, Big Data, and Artificial Intelligence in Agriculture and Food Industry. *IEEE Internet of Things Journal*, *9*(9), 6305–6324. https://doi.org/10.1109/JIOT.2020.2998584

Raptis, T. P., & Passarella, A. (2023). A Survey on Networked Data Streaming With Apache Kafka. *IEEE Access*, *11*, 85333–85350. https://doi.org/10.1109/ACCESS.2023.3303810

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, *11*(4), Article 4. https://doi.org/10.3390/info11040193

Tantalaki, N., Souravlas, S., & Roumeliotis, M. (2020). A review on big data real-time stream processing and its scheduling techniques. *International Journal of Parallel, Emergent and Distributed Systems*, *35*(5), 571–601. https://doi.org/10.1080/17445760.2019.1585848

Volnes, E., Plagemann, T., & Goebel, V. (2024). To Migrate or Not to Migrate: An Analysis of Operator Migration in Distributed Stream Processing. *IEEE Communications Surveys & Tutorials*, *26*(1), 670–705. https://doi.org/10.1109/COMST.2023.3330953

Zhang, G., Pan, F., Mao, Y., Tijanic, S., Dang’ana, M., Motepalli, S., Zhang, S., & Jacobsen, H.-A. (2024). Reaching Consensus in the Byzantine Empire: A Comprehensive Review of BFT Consensus Algorithms. *ACM Comput. Surv.*, *56*(5), 134:1-134:41. https://doi.org/10.1145/3636553