ISSN 2723-6471

Development of Skyline Query Algorithm for Individual Preference Recommendation in Streaming Data

Ruhul Amin[1,*], Taufik Djatna[2], Annisa[3], Sukaesih Sitanggang[4]

[1,3,4]*Department of Computer Science, Faculty of Mathematics and Natural Science, IPB University, Bogor 16680, Indonesia*

[1]*Department of Computer Science, Faculty of Technology Information, Universitas Nusa Mandiri, East Jakarta 13620, Indonesia*

[2]*Department of Agroindustrial Technology, Faculty of Agriculture Engineering and Technology, IPB University, Bogor 16680, Indonesia*

(Received: December 9, 2024; Revised: December 19, 2024; Accepted: January 19, 2025; Available online: March 4, 2025)

Abstract

The ability of a recommendation system to deliver relevant outcomes is significantly influenced by its adaptability to the dynamic nature of individual user preferences. Data-streaming-based recommendation systems face substantial challenges in aligning recommendations with rapid shifts in user preferences. Previous research on the development of skyline query algorithms has predominantly focused on processing efficiency and parallel performance optimization yet has not addressed the dynamic nature of individual user preferences—an essential factor for generating relevant and responsive recommendations in streaming data environments. This study aims to develop a skyline query algorithm called Distributed Data Skyline (DDSky) to provide recommendations based on dynamic individual user preferences within data-streaming contexts. DDSky leverages the Recency, Frequency, Monetary, and Rating (RFMRT) model to capture real-time changes in user preferences. This model is integrated with parallel skyline computation and structured to enhance the data processing efficiency on a large scale. The parallel processing approach divides tasks into smaller subtasks executed simultaneously across multiple threads. This strategy enables the simultaneous processing of attributes such as price, distance, and individual user preferences, thereby delivering relevant and responsive recommendations to real-time changes in user preferences. The DDSky algorithm was evaluated using a local dataset from the JALITA application and compared with the Eager algorithm. The results demonstrated that DDSky outperformed Eager, achieving an average recall value of 0.45 and an F1-measure of 0.55, compared to Eager's recall value of 0.33 and F1-measure of 0.47. Furthermore, DDSky achieved an average precision of 0.73, which closely approached Eager's precision of 0.82. Additionally, DDSky exhibited optimal throughput performance for datasets containing up to 10,000 items with high flexibility across various data types. With its unique technical approach, DDSky delivers more responsive and relevant recommendations to dynamic user preferences, establishing its superiority in data-streaming-based recommendation systems.

Keywords: DDSky, Dynamic Individual Preferences, RFMRT Model, Streaming Data, System Recommendation

1. Introduction

In the era of big data, the support of streaming data processing is essential [1] because it enables the continual and real-time processing of large data volumes [2], [3], particularly in applications such as recommendation systems. Streaming-based recommendation systems are influenced by their ability to process dynamic transaction histories, which reflect individual preferences that change over time [4]. Individual preferences refer to a person's desires or tendencies toward various products or services [5]. For example, in location-based recommendation systems, these changing preferences include user location shifts during transactions. The ability of recommendation systems to adapt to dynamic user preferences is key to delivering accurate recommendations [6].

Users of recommendation systems require query operators to process data and identify the most suitable outcomes based on their preferences [7]. Relying solely on exact matches between preferences and database records through query operators often fails to yield appropriate recommendations, as no recommendation might simultaneously satisfy criteria such as low cost, good taste, high ratings, and proximity. Skyline query is a method that identifies a set of data objects that align with user preferences, ensuring that no object in the set is dominated by another [8], [9]. An object is

***Corresponding author: Ruhul Amin (ruhulamin@apps.ipb.ac.id)**
DOI: https://doi.org/10.47738/jads.v6i2.599

ISSN 2723-6471

considered dominant if it has values no worse than those of another object in all dimensions and is strictly better in at least one dimension [10]. In this context, data objects refer to specific products or services. Skyline queries provide recommendations that involve multiple attributes [11]. However, this method has a limitation: the generated recommendations fail to adapt to dynamic individual preferences, which can change based on the location and evolving user priorities over time [12]. Beyond adapting to dynamic preferences, recommendation systems face significant difficulties processing streaming data. Streaming data's continuous, high-speed, and large-volume nature necessitates real-time updates to preference models. Efficient computational techniques are therefore required to ensure that recommendations remain relevant, responsive, and timely [4].

Previous studies [13] proposed using Local Split Decision (LSD) trees to accelerate skyline query computations on dynamic data. LSD Trees leverage geometric structures to store and prune irrelevant data, expediting repeated skyline query processing. However, this approach lacks flexibility in adapting to dynamically changing individual preferences, particularly in real-time environments that require swift responses to evolving user preferences. In addition, its performance deteriorates with high-dimensional data. Other studies [14] have successfully implemented a multicore-based parallel model for continuous skyline queries on high-dimensional data. However, this study did not consider dynamically changing individual user preferences. Without preference adaptation, the recommendations lacked personalization and responsiveness to individual user preference shifts. Another study developed a Distributed Parallel Model (DPM) for skyline queries on uncertain data streams in cloud environments, achieving high scalability, load balancing, and significant reductions in processing time [15]. Although the DPM model effectively processes large-scale skyline queries, it does not account for dynamic individual user preferences. The parallel DPM model focuses on optimizing parallel performance without mechanisms to update or adapt individual preferences in streaming data, thereby reducing the relevance of recommendations for users.

Based on prior studies, no comprehensive approach has been proposed to address dynamic individual preference calculations in streaming data, which can change in real-time according to user location or current preferences. This study aims to develop a skyline query algorithm for individual user preferences that generates recommendations based on dynamically changing user preferences. The algorithm that was developed is called Distributed Data Skyline (DDSky). DDSky is designed to provide recommendations based on dynamic individual preferences and process streaming data in real-time. This process enables DDSky to capture user preference changes promptly and adjust recommendations accordingly. DDSky leverages the Recency, Frequency, Monetary, and Rating (RFMRT) model, specifically designed to capture and analyze real-time changes in individual user preferences, ensuring that recommendations remain accurate and responsive to evolving user needs.

In contrast to collaborative filtering [16] approaches that rely on analyzing user preference similarities, which are often ineffective in capturing dynamic individual preferences owing to their focus on collective historical patterns, DDSky emphasizes real-time individual preference changes. Similarly, content-based filtering approaches, which rely on item attributes to provide recommendations, are limited by the explicit information available in item attributes and often struggle to account for implicit contextual changes in user preferences. However, the RFMRT model is designed to capture individual preference changes directly through recent transactional data, making it more responsive to real-time preference shifts. By integrating this model into DDSky, the system can provide recommendations tailored to change individual user preferences dynamically.

Furthermore, DDSky utilizes streaming data processing and parallel computing technologies to enhance its efficiency in managing large-scale and dynamic data. By integrating rating indicators and historical transaction data, DDSky can generate more accurate and relevant recommendations, outperforming traditional methods that struggle to adapt to dynamic individual preferences. This study introduces significant contributions to the field of recommendation systems. The development of the DDSky algorithm enables recommendations that dynamically adjust to individual user preferences in streaming data environments, ensuring real-time adaptability to changing user behaviors. Additionally, implementing parallel computing optimizes the efficiency of processing vast and continuously flowing data, allowing for seamless scalability and responsiveness. Moreover, the dynamic individual preference model incorporated within DDSky effectively captures and analyzes fluctuations in user preferences over time, enabling the system to refine its recommendations by evolving user interactions. These advancements collectively position DDSky as a robust and adaptive solution for real-time recommendation generation in dynamic data-driven applications.

ISSN 2723-6471

The remainder of this paper is organized as follows: Section 2 reviews relevant literature on the Recency, Frequency, Monetary (RFM) model, and skyline queries in streaming data. Section 3 discusses the methodology. Section 4 presents and discusses the results. Section 5 concludes the study with recommendations for potential extensions and future research directions.

2. Related Work

2.1. Analysis of the RFM Model (Recency, Frequency, Monetary)

The Recency, Frequency, Monetary model was first proposed by Hughes (1996), particularly in data-driven marketing strategies [17]. Since its introduction, the RFM model has been widely utilized to assess customer value and predict purchasing behavior. By evaluating three key attributes—recency, frequency, and monetary value—this model enables companies to identify customer segments with high potential for repeat purchases and significant contributions to overall revenue [18]. Recency measures the time interval between a customer's last and most recent purchase, with shorter intervals indicating higher engagement and a greater likelihood of future interactions with the company's products or services. On the other hand, frequency reflects the number of transactions a customer completes within a given period, where a higher frequency signifies stronger customer loyalty and a deeper relationship with the brand. Meanwhile, monetary value represents the total amount a customer spends over a specific timeframe, helping businesses recognize high-value customers who generate substantial revenue. By leveraging these three dimensions, the RFM model provides a structured framework for organizations to segment their customers, optimize marketing efforts, and develop targeted retention strategies. Its ability to predict future purchasing patterns based on past behaviors makes it a valuable tool for enhancing customer relationship management and improving business performance [19].

Previous studies [20] validated the effectiveness of the RFM model, concluding that high recency (R) and frequency (F) values are positively correlated with a customer's likelihood of repeat transactions. Moreover, a high monetary value (M) is associated with an increased probability of purchasing products or services from the company. These findings underscore the relevance of the RFM model in identifying high-value customers and developing targeted and effective marketing strategies. In the context of streaming data, the RFM model holds significant potential, particularly for capturing real-time changes in individual user preferences. Given the dynamic nature of data processing, the attributes in the RFM model can serve as a foundation for developing adaptive and responsive individual preference models. This approach provides a robust basis for integrating the RFM model into streaming data-based recommendation algorithms, aimed at enhancing the relevance and personalization of recommendations, which is the core focus of this research.

2.2. Skyline Query in Streaming Data

Skyline query research has advanced rapidly, with various innovative approaches introduced to enhance computational efficiency, particularly for dynamic and large-scale data. The study in [21] proposed a framework that efficiently manages skyline queries in streaming data by periodically updating the index structure while considering the validity period of records. By incorporating time-aware indexing mechanisms, the proposed approach ensures that outdated records are efficiently removed, thereby maintaining the accuracy and relevance of skyline results in dynamic streaming environments. Other studies [22] introduced Parallel Real-time Skyline Segmentation (PRSS), a novel and more efficient approach for real-time applications. PRSS offers a faster and more scalable solution to address the challenges of processing streaming data using a sliding window approach on multicore

A notable study [13] proposed local split decision (LSD) trees to accelerate skyline query computations on dynamic data. This structure enables efficient storage and pruning of irrelevant data, expediting repeated query processing. However, the LSD tree approach has limitations in adapting to dynamically changing individual preferences, particularly in real-time environments that require rapid responses to user preference changes. Furthermore, LSD Trees exhibit performance degradation when applied to high-dimensional data. Another study successfully implemented a multicore-based parallel model for continuous skyline queries on high-dimensional data [14]. Although effective in managing data complexity, this model does not account for dynamically changing individual user preferences, resulting in less personalized recommendations and less responsiveness to user preference shifts.

 ISSN 2723-6471

A study [15] proposed the Distributed Parallel Model (DPM) for skyline queries on uncertain data streams in cloud environments, emphasizing high scalability. The model significantly improved the performance by optimizing the load balancing and reducing the processing time. However, this approach does not incorporate mechanisms to update or adapt individual user preferences in streaming data, potentially diminishing the relevance of recommendations in personalization.

Although these studies have contributed significantly to improving the computational efficiency of skyline queries, none have specifically addressed dynamically changing individual user preferences. The lack of adaptation to evolving user preferences limits the effectiveness of recommendations, particularly in streaming data environments that require real-time adjustments. This study aims to bridge this gap by developing an algorithm that dynamically adapts to individual user preferences, enhancing recommendations' relevance and quality.

3. Methodology

This study develops an algorithm called Distributed Data Skyline (DDSky) to generate recommendations based on dynamically changing individual user preferences. The algorithm was designed to address several limitations of previous research, particularly in handling dynamic individual user preferences. By leveraging streaming data processing, this new algorithm can produce more accurate recommendations that adapt to rapid preference changes, irrespective of time and user location. This framework is a significant advancement in the evolution of recommendation systems, with applications such as providing personalized recommendations for local Indonesian culinary options tailored to dynamic individual preferences. The workflow of the DDSky algorithm is illustrated in figure 1.

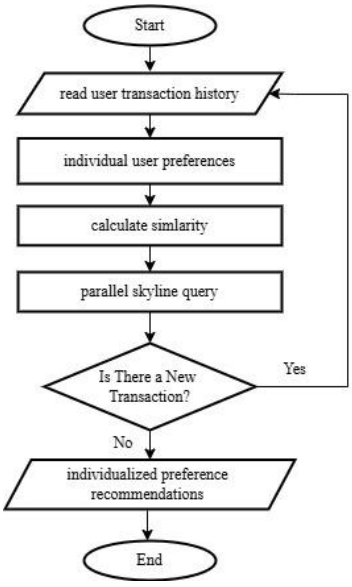

Figure 1. DDSky Workflow.

The DDSky algorithm processes the input from a user's transaction history. These historical transaction data are analyzed to identify individual user preferences using Algorithm 1. The identified individual preferences are then compared to the items available around the user's location to calculate the similarity. The similarity results are used as the preference attribute, which is subsequently processed along with the price and distance attributes through a skyline query. The three resulting attributes—price, preference, and distance—are processed parallel to identify skyline objects using Algorithm 2 on streaming data. Skyline processing is continuously performed upon receiving new data inputs, ensuring that recommendations remain responsive to dynamic individual user preferences. This parallel processing approach enhances the algorithm's efficiency in generating real-time personalized recommendations.

 ISSN 2723-6471

3.1. Development of the Individual Preference Recommendation Model

The individual preference recommendation model was designed to capture the dynamic individual preferences of users. This model utilizes the recency, frequency, monetary value, and rating attributes from user transaction histories, collectively called RFMRT. The developed RFMRT model can process data in real time (streaming), accommodating changes in individual user preferences over time. The notations used in this recommendation model are presented in table 1.

Table 1. Notations of the Individual Preference Recommendation Model (Algorithm 1)

No	Notation	Definition
1	u	User of the recommendation system, where u = 1, 2, 3,…,n; n n is the total number of user.
2	I	Product (items) recommended to the user, where I= 1, 2, 3, …, m; m is the total number of products (items).
3	T	A series containing dates in the recommendation system, from the start of the transaction (t_1) to the most recent transaction (t_{new}). In this case, $T = t_1, t_2, t_3, \dots, t, \dots, t_{new}$.
4	$TB_{u,I,t}$	A time series corresponding to the time (HH:MM:SS) of item I purchased by user u at times $TB = tb_1, tb_2, tb_3, \dots, tb, \dots, tb_{new}$
5	$f_{u,I,t,tb}$	Frequency of user u puchasing any item I on date t at time tb
6	$F_{u,I}$	The cumulative current frequency of user u purchasing item I. This represents the sum of frequencies from the start of transactions up to one transaction before the most recent transaction. 'Current' refers to the event on the current date (t_{new}) and at the current time tb_{new}.
7	$R_{u,i}$	The difference in days between the current date in the system t_{new} and the last transaction date made by user u for item I; $R_{u,i}$ is a non-negative integer. In practice, a user u may make more than one transaction at t_{new} (transactions occurring on the same day but at different times) for the same item I. In such cases, the value of $R_{u,i}$ is set to 0 ($R_{u,i} = 0$).
8	$M_{u,I}$	The current condition for the accumulation of money spent by user u on item I.
9	$RT_{u,I}$	The rating given by user u to item I, recorded in the recommendation system at the time of the last transaction. The rating value can range from 1 to 5, where a higher number indicates a better condition.
10	$P_{u,I}$	The current preference value of user u for item I.

Dynamic individual user preferences can be formulated using Equation 1 :

$$P_{u,I,new} = (F_{u,I,new} + M_{u,I,new} + RT_{u,I}) - R_{u,i} \quad (1)$$

Algorithm for Individual User Preferences

Algorithm 1 was developed to calculate the current preference value of user u for item I.

Input : $R_{u,i}$, $F_{u,I}$, $M_{u,I}$, $RT_{u,I}$
Output : $P_{u,I}$, max ($P_{u,I}$)
Proses :

```
1  Begin
2  For u = 1 to n
3     For i = 1 to m do
4        P_{u,I} = (F_{u,I} + M_{u,I} + RT_{u,I}) − R_{u,i}
5     End for
6     Max (P_{u,I})
7  End For
8  Write ('P_{u,I}'), Max ('P_{u,I}')
9  End
```

Algorithm 1: Individual User Preference (user)

 ISSN 2723-6471

The current preference value of the user u for item I can be calculated using Algorithm 1. This algorithm takes inputs from the attributes $R_{u,i}$, $F_{u,I}, M_{u,I}, RT_{u,I}$. The output of this algorithm is the current preference value of the user u for item I ($P_{u,I}$). The process begins by setting the value of u from 1 to n, where n is the number of users for whom the individual preferences are calculated. Next, the preference calculation for each user is performed for each item, considering the various factors that influence the preferences. Then, the preference for each item is calculated by summing the values of the attributes $F_{u,I,new}, M_{u,I,new}, RT_{u,I}$. The result of this sum is subtracted from the value of the attribute $R_{u,i.}$ Finally, the algorithm searches for the maximum value $P_{u,I,new}$ among all items to determine which item has the highest value.

3.2. Parallel Skyline Query

Skyline query processing is performed parallel to streaming data to efficiently generate recommendations tailored to dynamically changing user preferences. Algorithm 2 outlines the approach for executing skyline queries in parallel based on dynamic individual user preferences, leveraging the Distributed Parallel Model proposed in [15]. The notations used in Algorithm 2 are listed in table 2.

Table 2. Notations of the Individual Preference Recommendation Model (Algorithm 1)

Notation	Definition
D_s	Data streams
W_i	Local sliding window, where $i = 1,2,3, \dots, n;$ n is the total number of local sliding windows.
e_{new}	The newly arrived streaming object in W
e_{old}	The expired streaming object in W
S_i	Local skyline, where S_i is the result of the skyline computation from the local sliding window W_i
S_{global}	Global skyline, where S_{global} is the result of the global skyline obtained from the combination of all local skylines S_i

Algorithm 2 operates by distributing data streams (D_s) into local sliding windows (W_i), where i ranges from 1 to n ($1 \leq i \leq n$). Each local sliding window (W_i) has a maximum capacity equivalent to the total number of sliding windows (n). When the amount of data in a sliding window (W_i) exceeds its maximum capacity, the oldest data item (e_{old}) is removed to accommodate the new data item (e_{new}). This removal adheres to the First-In, First-Out (FIFO) principle, ensuring that only the most recent data are retained for skyline computation.

Using the First-In-First-Out technique to manage data within local sliding windows in real-time scenarios provides several advantages. The algorithm effectively reduces the computational load by systematically removing older data, as fewer items need to be processed in each iteration. This optimization significantly enhances the overall computation speed, ensuring the system operates more efficiently. Moreover, sliding windows keep the data constantly updated, directly improving the relevance of the information being processed. As a result, the recommendation outcomes are closely aligned with current user preferences, maintaining high accuracy and relevance. Additionally, by minimizing the volume of data the algorithm processes, response times are greatly improved. This mechanism enables the system to quickly adapt to user behavior and preferences changes, providing timely, precise, and real-time recommendations.

 ISSN 2723-6471

Input : $D_s = \{e_0, e_1, e_2, e_3, e_4, e_5, \ldots\}$
Output : S_{global}
Proses :

```
Begin
For Each sliding window W_i(1 ≤ i ≤ n)initialize empty;
   t=0;
   While there comes a new item e_new to W_t Do
      t=(t+1)%n
      W_t receives e_new;
      if size of W_t equals (n+1) Then
         //when W_t is full, remove the e_old
         remove e_old from W_t;
      Endif
      add e_new to W_t;
      True;
      for i=1 to n Do
         if size of W_i is less than (n-1) then
            windowFull = False;
            break;
         Endif
      Endfor
      if windowFull then
         for Each sliding window W_i(1 ≤ i ≤ n)
            skyline_query(Pc, Dt, Pr)
         Endfor
      Endif
   Endwhile
   prelimanary_global_skyline_set = union of all skyline in W_i;
   global_skyline_set=[];
   for Each item e in prelimanary_global_skyline_set Do
      if not any item O in prelimanary_global_skyline_set dominates e then
         add e global_skyline_set;
   Endfor
Endfor
End
```

Algorithm 2: DDSky with Parallel Processing

This process enables efficient parallel computation in each partition (P_i), where skyline computation is performed for the local sliding window (W_i). The results from all sliding windows ($W_1, W_2, \ldots, W_n$) are merged to form the global skyline, representing the final output. This merging ensures the algorithm efficiently generates recommendations based on user preferences in dynamic streaming data environments. By integrating the FIFO principle, the DDSky algorithm enhances computational efficiency. It ensures adaptability to real-time data processing requirements, making it an ideal solution for applications based on streaming data.

3.3. Scenario of DDSky Algorithm Workflow

Figure 2 depicts the operation of the DDSky algorithm. An individual user U_1 performs purchase transactions involving five local culinary items I_1, I_2, I_3, I_4 dan I_5, while located at Location 1. Each transaction made by the user is stored in a system database. Based on the stored transaction history, the user's preference values were calculated using the RFMRT model, as outlined in Algorithm 1. This process generates the user's current preference value $P_{u,I}$, where the items with the highest preference values reflect the user's primary preferences.

 ISSN 2723-6471

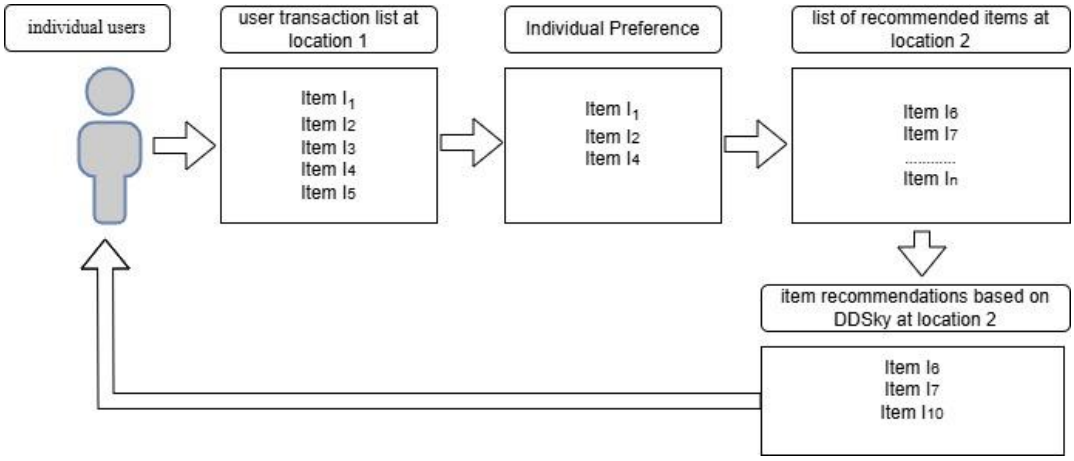

Figure 2. Scenario of DDSky Algorithm Workflow

In this scenario, the primary preferences of U_1 are I_1, I_2 and I_4. User U_1 then dynamically moves from Location 1 to Location 2. Upon arrival at Location 2, the system collects data on the available items, including attributes such as geographic coordinates (longitude and latitude), price, and distance from the user. Each item at Location 2 was further analyzed to measure its similarity with the user's individual preferences I_1, I_2 and I_4. The results of this similarity analysis were used to compute the preference attributes for each item in location 2.

Once the preference attributes are established, the system performs a skyline query process in parallel on streaming data using three main attributes: distance, price, and preference. The skyline query aims to filter optimal items where any other item across all attributes does not dominate each recommended item. The outcome of this process is a recommendation list for user U_1 at Location 2, such as I_6, I_7 and I_{10}, which satisfies the criteria of being close in distance, affordable in price, and aligned with the user's transaction history preferences. This scenario demonstrates how DDSky effectively handles dynamic user location changes and delivers relevant and responsive recommendations to individual user preferences.

3.4. Parallelization Design

This study proposes a DDSky algorithm to generate recommendations based on dynamic individual user preferences. The skyline query process is executed parallel to handle rapidly changing and large-scale data, leveraging multithreading parallel computing technology to enhance the speed and efficiency of streaming data processing. The DDSky algorithm uses a distributed parallel model and employs Apache Kafka technology for real-time data processing. Specifically, the framework for parallel processing is illustrated in figure 3, which consists of three types of nodes for parallel skyline query processing on streaming data: Monitor Node (M), Partition Node (P), and Query Node (Q).

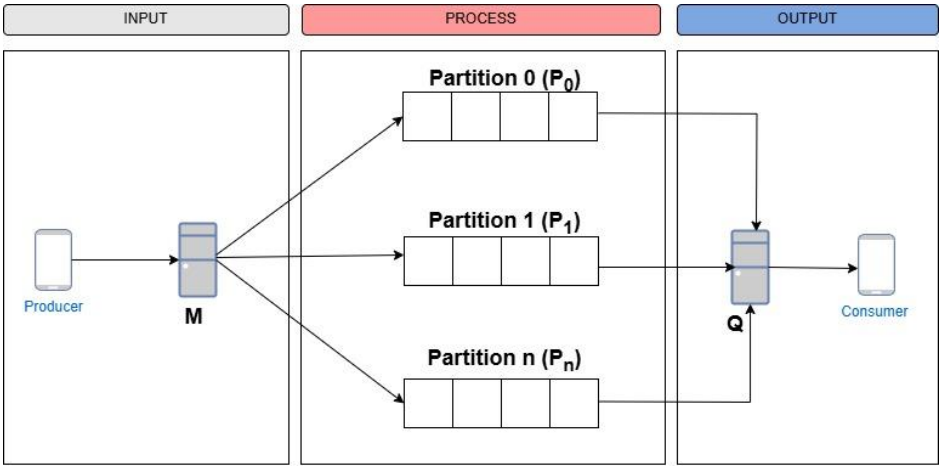

Figure 3. Parallel Skyline Query Processing Framework

 ISSN 2723-6471

The monitor node (M) is responsible for transmitting streaming data (DS= $\dots, e_6, e_5, e_4, e_3, e_2, e_1, e_0$) produced by the data producer before it enters the parallel computation nodes P_i. For each incoming data point $e_0, e_1, e_2, \dots, e_n$, the monitor node performs the following tasks: Assigns a timestamp (e_{arr}) to each data point corresponding to the current system time, and Distributes each data point $e_0, e_1, e_2, \dots, e_n$ to a specific partition node (P_i) for individual preference computation and skyline query processing, respectively. This process continues until the internal time of the data point reaches its expiration time $e_{exp} = e_{arr} + N$, where N denotes the size of the sliding window. Once the data is allocated, the Partition Nodes (P_i) take over the computational workload. These nodes are responsible for processing user preferences and executing parallel skyline queries based on the assigned data. Each partition node maintains its own sliding window, ensuring that only relevant and recent data is considered in real-time processing. This localized data management strategy enhances efficiency by minimizing the computational overhead while maintaining high accuracy in preference calculations and skyline query results. The final stage of the process is managed by the Query Node (Q), which continuously gathers skyline results from all partition nodes and consolidates them into a final output for the end user. By aggregating results from multiple sources, the query node ensures that recommendations are comprehensive and reflect the most relevant data. This distributed, and parallel processing framework enables the DDSky algorithm to efficiently adapt to the demands of real-time dynamic data environments, delivering personalized recommendations with high responsiveness and precision. The system maintains its ability to process large-scale streaming data through this systematic approach while ensuring optimal performance and user satisfaction.

4. Experimental Evaluation

In this study, the evaluation was divided into two scenarios. The first evaluation aimed to measure the accuracy of the DDSky algorithm using precision, recall, and F1 measure metrics. The second evaluation focused on the computation time required by the DDSky algorithm to generate skyline objects. The results of the accuracy and computation time produced by the DDSky algorithm were compared with those of other algorithms.

4.1. Dataset

This study uses a dataset of local culinary profiles, local culinary vendor profiles, and user transaction histories sourced from the JALITA (Jajanan Asli Nusantara Pintar) application. JALITA is a mobile-based application for a local culinary recommendation system based on individual user preferences. Another dataset consists of individual user preferences from a questionnaire distributed to users. The JALITA application is a mobile-based recommendation system for Indonesian local cuisines based on user preferences.

4.2. Accuracy Evaluation of the DDSky Algorithm

Accuracy evaluation uses recall, precision, and F1 metrics, which have been applied previously in the research [23]. The definitions and equations used for each of the metrics are as follows:

Precision was defined as the percentage of items recommended by the user. Precision measures how well a system recommends relevant and liked items. The precision calculation method for precision is given in Equation (2).

$$\text{precision} = \frac{|\text{prefered} \cap \text{recommended}|}{\text{recommended}} \quad (2)$$

Recall is defined as the percentage of liked items that are recommended. Recall measures how well a system recommends items that the user truly likes. The calculation method for recall is given by Equation (3).

$$\boldsymbol{recall} = \frac{|prefered \cap recommended|}{prefered} \quad (3)$$

F1-measure is a balanced combination of precision and recall. The F1-measure combines the precision and recall metrics into a single value that provides an overall view of the algorithm's performance of the algorithm. The calculation method for the F1-measure is given by Equation (4).

$$\text{F1} = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}} \quad (4)$$

Where preferred are items liked by the user, and recommended are the set of skyline objects produced by the algorithm.

In the accuracy evaluation, this study compared the DDSky algorithm with the Eager parallel processing algorithm [14]. Both algorithms share fundamental similarities, as they are designed to process skyline queries in streaming data environments, requiring time efficiency and adaptability to real-time data changes. DDSky and Eager implement parallel processing techniques to handle large-scale streaming data, enabling both to achieve high performance in dynamic data scenarios. Additionally, both algorithms utilize a sliding window model to limit the data processed within specific intervals, ensuring that only relevant data is considered. The output of both algorithms is a skyline set comprising optimal objects based on user preferences.

The testing was performed by comparing the recommendation outputs of both algorithms against test data reflecting actual user preferences in dynamic streaming data scenarios. Identical datasets ensured evaluation parity, with normalized attributes to standardize value scales. The accuracy results for the DDSky algorithm are presented in table 3, demonstrating its superiority in capturing dynamic user preferences. Conversely, table 4 provides the evaluation results for the Eager algorithm, which excels in processing efficiency but is less responsive to changes in user preferences. This comparison offers a comprehensive overview of the strengths and weaknesses of each algorithm in a streaming data environment.

Table 3. Accuracy Evaluation Results for the DDSky Algorithm

UserID	intersection	prefered	recommended	precision	recall	F1
4	5	53	6	0.83	0.09	0.17
5	5	40	6	0.83	0.13	0.22
6	5	30	7	0.71	0.17	0.27
.....	.....	.....	.....	.....	.....	.....
46	4	31	9	0.44	0.13	0.20
47	4	47	7	0.57	0.09	0.15
48	1	21	4	0.25	0.05	0.08
average				0.73	0.11	0.19

Table 4. Evaluation of the accuracy results of the Eager algorithm

UserID	intersection	prefered	recommended	precision	recall	F1
4	4	53	4	1.00	0.08	0.14
5	4	40	4	1.00	0.10	0.18
6	3	30	4	0.75	0.10	0.18
.....	.....	.....	.....	.....	.....	.....
46	3	31	4	0.75	0.10	0.17
47	2	47	4	0.50	0.04	0.08
48	1	21	4	0.25	0.05	0.08
average				0.82	0.08	0.15

The results shown in figure 4 indicate that DDSky achieves a precision of 0.73, a recall of 0.45, and an F1-measure of 0.55. In contrast, the Eager algorithm, which emphasizes distributed and parallel processing without considering individual user preferences [14], demonstrates a precision of 0.82, a recall of 0.33, and an F1-measure of 0.47. Based on the accuracy evaluation of the two algorithms, while DDSky exhibits a slightly lower precision value, its primary focus lies in the ability to capture dynamic individual user preferences in real-time. This decrease in precision occurs because DDSky is designed to broaden the scope of recommendations by adapting to changes in dynamic individual user preferences. However, this may result in some less relevant recommendations. An increase in recall often

 ISSN 2723-6471

negatively affects precision. This phenomenon arises because adding more items to the recommendation list increases the likelihood of capturing relevant items (enhancing recall) and raises the risk of including irrelevant items, thereby reducing precision [24]. Nevertheless, the improved recall value indicates that the algorithm is more responsive to encompassing items that align with user preferences, making it more adaptive to changes caused by temporal and locational factors. This trade-off is acceptable in systems based on dynamic preferences, where responsiveness and adaptability are prioritized to enhance the relevance and personalization of recommendations.

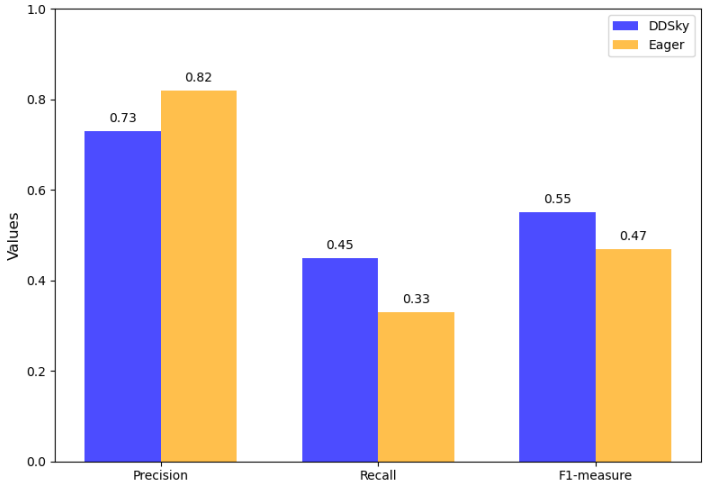

Figure 4. Comparative analysis of performance metrics between the DDSky and Eager algorithms.

4.3. Throughput Evaluation

The second scenario in this evaluation was the throughput testing of the DDSky algorithm. We used a synthetic dataset of various types to test throughput, including independent, correlated, and anticorrelated data. These different dataset types were used to assess the performance of the DDSky algorithm under various data conditions. Throughput is measured to determine how efficiently DDSky processes large volumes of data with diverse types while ensuring that the improvement in recommendation quality does not come at the cost of the algorithm's performance in terms of processing speed. The throughput evaluation results for the DDSky algorithm are shown in figure 5.

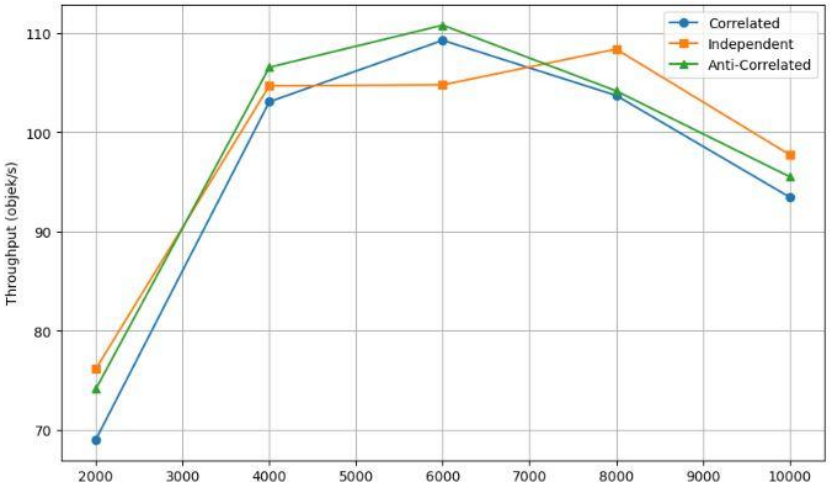

Figure 5. Throughput Evaluation Results for DDSky

The evaluation results indicate that DDSky's performance varies owing to the differing distribution characteristics of each data type, which impacts the computational load of the algorithm. Correlated data exhibit mutually supportive patterns among attributes, leading to quicker dominance in skyline processing and achieving optimal throughput at 6,000 data points. However, beyond this optimal point, the reduced number of skyline candidates causes a decline in throughput. In contrast, anticorrelated data, where attributes conflict with one another, increase the number of skyline candidates needing processing. This results in a higher throughput than correlated data, although the optimal pattern

remains similar. For independent data, where attribute distributions lack specific patterns, DDSky requires more time to identify the skyline, gradually increasing throughput at 8,000 data points.

This throughput variation indicates that DDSky's performance is influenced by the complexity of the dominance relationships among data objects, which heavily depends on the distribution of the processed data types. These findings underscore DDSky's ability to handle various data distribution patterns efficiently, ensuring consistent performance for large-scale datasets of up to 10,000 objects without significant throughput degradation. These results confirm DDSky's capability to maintain efficient skyline computation in diverse and dynamic streaming data environments.

5. Conclusion

This study has developed the DDSky algorithm, which is designed to provide recommendations based on the dynamic individual preferences of users of streaming data. The results show that DDSky outperforms the Eager algorithm, with an average recall of 0.11 and an F1 measure of 0.19, surpassing Eager, which has a recall of 0.08 and an F1 measure of 0.15. This result indicates that DDSky generates accurate and relevant recommendations more effectively. Additionally, the study successfully developed the RFMRT model, which can identify individual user preferences using transaction history and user ratings data.

The main contribution of this study is the development of a skyline query algorithm that can adapt to dynamic user preferences in the context of streaming data. However, this study had several limitations. Its focus is confined to local Indonesian culinary recommendation systems, leaving the application of DDSky in other domains, such as e-commerce, social media, and retail business, unexplored in depth. These domains hold significant potential for leveraging DDSky, for instance, by recommending relevant retail products based on customers' purchase histories or curating personalized media content aligned with users' dynamically evolving preferences.

Therefore, future research is recommended to test the application of the DDSky algorithm in various other domains to evaluate its flexibility and to integrate the individual preference model with machine learning to improve the prediction capabilities and adaptability to changes in user preferences. Evaluations in more complex real-time scenarios are also suggested to ensure the algorithm's effectiveness under real-world conditions. Further studies should also focus on improving the scalability and performance of the algorithm in handling huge data volumes by developing additional optimization techniques and more advanced data processing strategies. Thus, the recommendations generated will be more relevant and responsive to real-time changes in user preferences.

6. Declarations

6.1. Author Contributions

Conceptualization: R.A., T.D., A., and S.S.; Methodology: R.A., T.D.; Software: R.A.; Validation: R.A., A., S.S., and T.D.; Formal Analysis: R.A., S.S., and T.D.; Investigation: R.A.; Resources: S.S.; Data Curation: S.S.; Writing—Original Draft Preparation: R.A., S.S., and T.D.; Writing—Review and Editing: S.S., R.A., and T.D.; Visualization: R.A. All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. 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.

References

[1] A. B. Amale, K. K. Bajaj, M. D. Shamout, L. C. Campos Ramírez, M. L. Medina Vásquez, and J. R. Yataco Torrealva, "Real-Time Analytics with Big Data and Streaming Computation," *2023 Int. Conf. Power Energy, Environ. Intell. Control. PEEIC 2023*, pp. 1668–1673, 2023, doi: 10.1109/PEEIC59336.2023.10452008.

[2] H. Kumar, P. J. Soh, and M. A. Ismail, "Big Data Streaming Platforms: A Review," *Iraqi J. Comput. Sci. Math.*, vol. 3, no. 2, pp. 95–100, Apr. 2022, doi: 10.52866/IJCSM.2022.02.01.010.

[3] A. Lemzin, "Streaming Data Processing," *Asian J. Res. Comput. Sci.*, vol. 15, no. 1, pp. 11–21, Jan. 2023, doi: 10.9734/AJRCOS/2023/V15I1311.

[4] Y. Wu and Y. Yusof, "Emerging Trends in Real-time Recommendation Systems: A Deep Dive into Multi-behavior Streaming Processing and Recommendation for E-commerce Platforms," *J. Internet Serv. Inf. Secur.*, vol. 14, no. 4, pp. 45–66, Oct. 2024, doi: 10.58346/JISIS.2024.I4.003.

[5] M. J. Osborne and A. Rubinstein, "Consumer preferences," *in Models in Microeconomic Theory, Open Book Publishers*, 2023, vol. 2023, no. 6, pp. 45–56. doi: 10.11647/OBP.0361.04.

[6] R. M. Roy, "An E-Commerce Recommendation System Based on Dynamic Analysis of Customer Behavior," *Int. J. Sci. Technol. Eng.*, vol. 12, no. 11, pp. 37–48, Nov. 2024, doi: 10.22214/IJRASET.2024.64867.

[7] Y. Zhu et al., "Query-based Interactive Recommendation by Meta-Path and Adapted Attention-GRU," *Int. Conf. Inf. Knowl. Manag. Proc.*, vol. 2019, no. 6, pp. 2585–2593, Jun. 2019, doi: 10.1145/3357384.3357805.

[8] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," *Proc. - Int. Conf. Data Eng.*, vol. 2001, no. 4, pp. 421–430, 2001, doi: 10.1109/icde.2001.914855.

[9] R. Amin, T. Djatna, A. Annisa, and I. S. Sitanggang, "Skyline Query Based on User Preferences in Cellular Environments," *JITK (Jurnal Ilmu Pengetah. dan Teknol. Komputer)*, vol. 9, no. 1, pp. 143–153, 2023, doi: 10.33480/jitk.v9i1.4192.

[10] B. J. Santoso, R. M. Ijtihadie, and I. N. Y. Mahottama, "Answering Durable Skyline Queries on Multidimensional Time Series Data Using Grid-Based Approach," 2023 14th International Conference on Information & Communication Technology and System (ICTS). IEEE, vol. 2023, no. 10, pp. 331-337, 2023. doi: 10.1109/icts58770.2023.10330858.

[11] Y. Shu, J. Zhang, W. E. Zhang, D. Zuo, and Q. Z. Sheng, "IQSrec: An Efficient and Diversified Skyline Services Recommendation on Incomplete QoS," IEEE Trans. Serv. Comput., vol. 16, no. 3, pp. 1934–1948, May 2023, doi: 10.1109/TSC.2022.3189503.

[12] M. Luo, X. Zhang, J. Li, P. Duan, and S. Lu, "User Dynamic Preference Construction Method Based on Behavior Sequence," Sci. Program., vol. 2022, no. 2022, pp. 1-15, 2022, doi: 10.1155/2022/6101045.

[13] D. Köppl, "Dynamic Skyline Computation with LSD Trees," *Analytics*, vol. 2, no. 1, pp. 146–162, 2023, doi: 10.3390/analytics2010009.

[14] T. De Matteis, S. Di Girolamo, and G. Mencagli, "A multicore parallelization of continuous skyline queries on data streams," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9233, no. 1, pp. 402–413, 2015, doi: 10.1007/978-3-662-48096-0_31.

[15] X. Li, Y. Wang, X. Li, and Y. Wang, "Parallel skyline queries over uncertain data streams in cloud computing environments," *Int. J. Web Grid Serv.*, vol. 10, no. 1, pp. 24–53, 2014, doi: 10.1504/IJWGS.2014.058759.

[16] K. Bhareti, S. Perera, S. Jamal, M. H. Pallege, V. Akash, and S. Wiieweera, "A Literature Review of Recommendation Systems," *2020 IEEE Int. Conf. Innov. Technol. INOCON 2020*, vol. 2020, no. 1, pp. 1–7, 2020, doi: 10.1109/INOCON50539.2020.9298450.

[17] A. M. Hughes, "Strategic Database Marketing," *Database Mark. Inst.*, 1994, p. 400.

[18] A. Handojo, N. Pujawan, B. Santosa, and M. L. Singgih, "A multi layer recency frequency monetary method for customer priority segmentation in online transaction," *Cogent Eng.*, vol. 10, no. 1, pp. 0–19, 2023, doi: 10.1080/23311916.2022.2162679.

 ISSN 2723-6471

[19] Chinazor Prisca Amajuoyi, Luther Kington Nwobodo, and Ayodeji Enoch Adegbola, "Utilizing predictive analytics to boost customer loyalty and drive business expansion," *GSC Adv. Res. Rev.*, vol. 19, no. 3, pp. 191–202, Jun. 2024, doi: 10.30574/GSCARR.2024.19.3.0210.

[20] A. J. Christy, A. Umamakeswari, L. Priyatharsini, and A. Neyaa, "RFM ranking – An effective approach to customer segmentation," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 33, no. 10, pp. 1251–1257, 2021, doi: 10.1016/j.jksuci.2018.09.004.

[21] K. Alami and S. Maabout, "A framework for multidimensional skyline queries over streaming data," *Data Knowl. Eng*., vol. no. 1, pp. 101792, 2020, doi: 10.1016/j.datak.2020.101792.

[22] W. Khames, A. Hadjali, and M. Lagha, "Parallel continuous skyline query over high-dimensional data stream windows," *Distrib. Parallel Databases*, Dec. vol. 2024, no. 7, pp. 469-524, 2024, doi: 10.1007/S10619-024-07443-7.

[23] F. Rhimi, S. B. Yahia and S. B. Ahmed, "Enhancing Skyline Computation with Collaborative Filtering Techniques for QoS-Based Web Services Selection," *2015 IEEE 14th International Symposium on Network Computing and Applications, Cambridge, MA, USA, 2015*, vol. 2015, no. 9, pp. 247-250, doi: 10.1109/NCA.2015.18.

[24] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *J. Mach. Learn. Res*., vol. 10, no. 9, pp. 2935–2962, 2009. doi: 10.5555/1577069.1755883.