Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.6, December 2025, pp. 3346~3355
e-ISSN: 2548-6861

Data Streaming Pipeline Model Using DBSTREAM-Based Online Machine Learning for E-Commerce User Segmentation

Muhammad Adin Musababa[1]*, Muhammad Fachrie[2]**
* Informatika, Universitas Teknologi Yogyakarta
** Sains Data, Universitas Teknologi Yogyakarta
muhammad.5220411328@student.uty.ac.id [1], muhammad.fachrie@staff.uty.ac.id [2],

Article Info

Article history:

Received 2025-10-19
Revised 2025-11-01
Accepted 2025-11-08

Keyword:

Data Streaming,
DBSTREAM,
Online Machine Learning,
Pipeline,
Segmentation.

ABSTRACT

The rapid development of information technology has driven major transformations in the digital business sector, particularly e-commerce. Consumers who shop at e-commerce sites generally have different characteristics, behaviors, and needs. Analyzing the behavior of each consumer is difficult to do manually, requiring an automation system that can help identify consumer behavior patterns adaptively. However, most customer segmentation approaches still rely on batch learning methods based on static data, making them unable to quickly adapt to changes in user behavior. This study aims to design a streaming data pipeline based on Online Machine Learning (OML) integrated with the Density-Based Clustering for Data Streams (DBSTREAM) algorithm to produce adaptive e-commerce user segmentation. The system was developed using Python with RabbitMQ as a real-time data stream simulator, MongoDB for storing results, and Streamlit as a visualization interface. The clustering process was performed incrementally using DBSTREAM, then stabilized through Hierarchical Agglomerative Clustering (HAC) to avoid over-segmentation. Evaluation using the Silhouette Coefficient and Davies-Bouldin Index (DBI) shows that the optimal model for the cluster threshold is in the range of 0.6 to 0.8 and for the fading factor is 0.0005 or even smaller, such as 0.0003. The evaluation results obtained a Silhouette value of -0.1125 and a DBI of 0.2796. These results prove that DBSTREAM-based OML integration is capable of forming consumer behavior segmentation efficiently and adaptively to continuous and real-time changes in streaming data.

I. INTRODUCTION

The rapid development of information technology has driven major transformations in the digital business sector, particularly e-commerce, which has grown to become one of the fastest-growing industries in the world [1]. Increased internet penetration and global digitalization have led to projections that the e-commerce industry will reach a market value of USD 5.4 trillion by 2025 [2]. These dynamics place consumers at the center of business strategy, where a deep understanding of user behavior is key to a company's competitive advantage [3], [4].

Consumers who shop at e-commerce sites generally have different characteristics, behaviors, and needs. Analyzing consumer behavior one by one is a problem for companies, so an approach is needed to simplify the identification of consumer behavior patterns [3]. Research on consumer segmentation in e-commerce has been conducted extensively using various algorithmic approaches. Hafidz Ardana [5]applied K-Means clustering to segment online sales customers and produced two main groups that describe differences in transaction levels, but there are still obstacles in this study. Hossain [6] compared centroid-based clustering with density-based clustering and concluded that DBSCAN is more effective in identifying unusual consumer behavior than K-Means, which is still used in this study.

However, the commonly used consumer behavior analysis approach still relies on static data and batch processing

methods. This method has significant limitations, namely, it is unable to respond to changes in user behavior in real-time, requires full retraining every time new data appears, and is inefficient in handling large amounts of data that continue to grow [7]. The research by [8] focused on streaming machine learning algorithms using online versions of Support Vector Machines (SVM) and K-Means on various big data systems, with the result that Twister2 had the lowest latency among the platforms tested. Meanwhile, [9] developed Clustering of Evolving Data Streams via a Density Grid-based Method (CEDGM), which improves the quality of data stream clustering by relying on core micro-clusters as a representation of dynamic data, though this approach still faces challenges in maintaining clustering quality over extended periods with varying data densities.

In fact, e-commerce user behavior data is massive, diverse, and dynamic, influenced by product preferences, shopping habits, and ever-changing market trends [10]. [11] presented a comprehensive survey on data stream clustering, emphasizing the major challenges in handling unlimited data, concept drift, and the need for fast processing with memory limitations. These conditions indicate a research gap in the provision of adaptive analytical models in streaming data environments.

To overcome these challenges, the Online Machine Learning (OML) approach is considered relevant because it can update models incrementally without requiring full retraining. In a more specific study, [12] introduced DBSTREAM, a density-based clustering algorithm designed for streaming data. This method overcomes the weakness of DBSCAN in maintaining density information between micro-clusters by utilizing a shared density graph, resulting in more accurate and efficient clustering. The study shows that the density-based approach in streaming data environments offers advantages over traditional methods, especially in detecting changes in data distribution. The integration of OML with the Density-Based Clustering Algorithm Designed for Use with Data Streams (DBSTREAM) provides opportunities to build adaptive consumer segmentation systems, including detecting non-linear patterns and identifying user groups with unique characteristics.

From a series of previous studies, it appears that although various methods have been developed, there is still a need for adaptive, efficient consumer segmentation models that are capable of operating in real time, particularly with the integration of OML, streaming data, and the DBSCAN algorithm. This study designs an OML-based e-commerce user segmentation system integrated with a real-time data streaming pipeline. The methods used involve incremental data flow pipeline simulation, DBSTREAM-based clustering processing, and evaluation using the Silhouette Coefficient and Davies-Bouldin Index (DBI) metrics. The main contribution of this research is to present an adaptive segmentation model that not only contributes to academic studies on the application of OML in streaming data but also offers practical solutions for the e-commerce industry to understand consumer behaviour in a more dynamic, efficient, and contextual manner.

II. METHODS

This research will be conducted gradually and sequentially in a systematic manner as illustrated in Figure 1 below. The research will be conducted experimentally using real data from one of the largest e-commerce platforms in China.

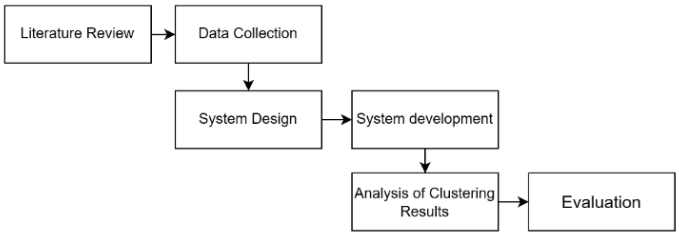

Figure 1. Research flow for the development of the OML system

In Figure 1, the initial stage of the research begins with conducting a series of literature studies from various relevant sources such as journal articles and books. After the e-commerce user data is obtained, the OML system design process will be carried out with streaming data. All system designs will be developed using the Python programming language and supporting libraries. The tools used in developing this system are RabitMQ, which simulates real-time streaming data, and MongoDB for the database.

The result of this system is a dashboard that contains all clustering results from the OML process, such as clustering result visualizations, processed data, and clustering data with features that have been processed for clustering result analysis. After the system creation process that has produced clusters, the clustering results will be evaluated to see how the OML system performs in processing data incrementally and in real time. The resulting clusters will then be analyzed further.

All testing will be carried out using the following hardware and operating system specifications. The system operates on Windows 11 and runs on an MSI Modern 14 C12MO laptop. It is powered by a 12th Gen Intel(R) Core(TM) i7-1255U processor with a base speed of 1.70 GHz, supported by 16 GB of RAM to ensure smooth computational performance. The device is equipped with a 512 GB SSD, providing fast data access throughout the testing process. Additionally, the laptop features an Intel(R) Iris(R) Xe Graphics GPU, which supports graphical processing needs during the execution of the tests.

A. Data Collection

The data used in this study was sourced from the Taobao User Behavior Data for Recommendation dataset, which is publicly available through the Alibaba Cloud Tianchi platform at https://tianchi.aliyun.com/dataset/649. This dataset is one of the largest and most comprehensive e-commerce datasets provided by the Alibaba Group for

research and recommendation system development purposes. This dataset contains 100 million Taobao e-commerce user data, with a random selection of approximately one million users who have various types of behaviors including clicks, purchases, and adding items to shopping carts. The attributes of the dataset are presented in detail in Table I.

TABEL I
EXPLANATION OF THE DATASET VARIABLES TO BE USED AND THE TYPE OF EACH VARIABLE

Variables	Explanation
User ID	Integer, serialized ID representing a user
Item ID	Integer, serialized ID representing an item
Category ID	Integer, serialized ID representing an item category
Behavior type	String, enumeration type from ('pv', 'buy', 'cart', 'fav')
Timestamp	Integer, indicating user behavior time in UNIX timestamp format

The dataset used in this study covers the period from November 25, 2017, to December 4, 2017, representing 10 days of intense user activity on the Taobao platform. This period was strategically chosen because it covers one of the peak periods of e-commerce activity in China, namely the period after Singles' Day (11.11), known as the world's largest online shopping festival.

B. Online Machine Learning

OML is a machine learning approach designed to process data incrementally, so that models can be updated whenever new data comes in without the need for full retraining. This process makes OML relevant for environments with large-scale, high-speed, and ever-growing data, such as consumer behavior data in e-commerce [7]. Mathematically, model parameter updates are performed through gradual optimization by adjusting weights based on the loss function and learning rate. This mechanism allows the model to adapt to data dynamics in real-time, while reducing the high computational costs typically associated with batch learning methods [13].

The advantage of OML lies not only in its efficiency, but also in its ability to accommodate concept drift, which is a change in data distribution over time that often occurs in real-world data streams [14]. By processing data gradually, OML can maintain the relevance of predictions and segmentation without losing important historical information. This approach has been widely proposed for various streaming data applications, including anomaly detection, classification, and clustering, and in the context of this study, it was used to build adaptive, responsive e-commerce user segmentation capable of capturing dynamically changing behavior patterns.

C. DBSTREAM Algorithm

The DBSTREAM algorithm is a density-based clustering algorithm designed to handle real-time streaming data. The main principle of this algorithm is to group data based on local density by utilizing two core concepts, namely micro-clusters and shared density graphs (SDG). Each time new data arrives, DBSTREAM incrementally updates the model representation by adding the point to the nearest micro-cluster with sufficient density, or forming a new micro-cluster if none meet the proximity radius threshold. Through this approach, DBSTREAM is able to maintain a cluster structure that is adaptive to changes in data patterns without the need to retrain the entire model [12].

In general, the DBSTREAM process is divided into two main stages, namely the online stage and the offline stage.

1) The Online stage serves to summarize the data flow that enters a set of micro-clusters. Each micro-cluster stores statistical information such as the number of points, centroids, and the last update time. When a new data point $x_t$enters, the system calculates the Euclidean distance between x_t and each micro-cluster centroid c_i. If the distance $d(x_t, c_i) < \epsilon$, then the point is added to micro-cluster i with a mathematical update of the parameters in equation (1):

$$c_i^{(t+1)} = c_i^{(t)} + \eta\left(x_t - c_i^{(t)}\right) \quad (1)$$

where in equation (1) the parameter η is the learning rate that determines the degree of influence of new data on the centroid position. In addition, the micro-cluster weights will be updated with a fading factor mechanism to reduce the influence of old data over time.

2) The Offline stage aims to form macro-clusters from a set of active micro-clusters by utilizing SDG. SDG is used to identify relationships between micro-clusters that share density in the data space. Two micro-clusters will be connected if the distance between their centroids is below the connectivity threshold and there is significant density overlap. The final clusters are then formed based on the connected components in the SDG, where each component represents a group of consistent behaviors or patterns.

The main advantage of DBSTREAM lies in its ability to handle changes in data distribution (concept drift) through a fading function mechanism that automatically removes old micro-clusters when their weight falls below a minimum value. This mechanism keeps the model memory-efficient and adaptive to new patterns without losing historical context. This approach makes DBSTREAM particularly well-suited for large-scale data streaming applications, such as real-time e-commerce user behavior segmentation.

D. System Design

The system design developed in this study was built using an integrated data streaming pipeline approach from the input process to the monitoring stage. The system architecture is shown in Figure 2.

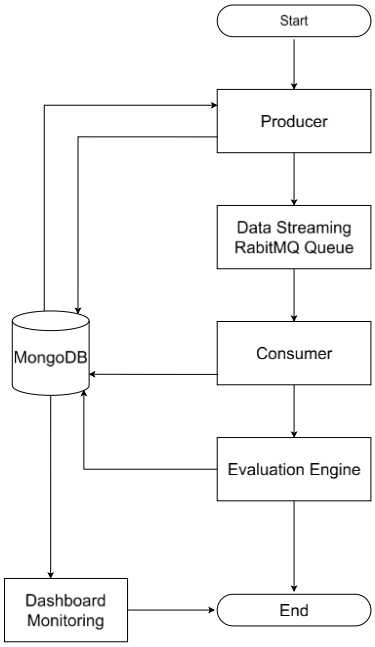

Figure 2. System design process flowchart

The Producer component acts as an intermediary in streaming data flow simulations that represent the continuous input of user behavior data from the database. Data is retrieved based on event IDs sorted in ascending order, so that each new piece of data is processed sequentially according to the time it arrived. The selection of triggers based on event IDs ensures that no data is missed and that all entries with unique identities can be sent in a timely manner. This mechanism mimics the actual conditions of an e-commerce system that receives user behavior data continuously and sequentially.

Before being fed into the system, data retrieved from the database first undergoes preprocessing. Behavior type variables are processed using encoding techniques, which are methods that convert categorical data into numerical representations so that they can be interpreted by machine learning algorithms [15]. This study uses an ordinal encoder, which assigns numerical values based on the hierarchical order between categories, such as view, cart, purchase, and favorite. This approach aims to represent user behavior proportionally in the form of integers that are easy for the model to process.

Next, the user ID, item ID, and category ID variables are processed using scaling techniques to standardize the range of values between variables. This process is important so that variables with large scales do not dominate other variables during model learning, while maintaining numerical stability in the computation process [16]. This method converts time values into the range [0,1], maintaining the relative proportions between times while ensuring that all features are on the same scale. The timestamp variable is normalized using MinMaxScaler, with a mathematical formula as shown in equation (2):

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (2)$$

Data that has undergone preprocessing will be sent in real time using a staged delivery mechanism. The delivery process is carried out in batches of 1,000 documents, which means that one delivery cycle consists of 1,000 pieces of data that are ready to be processed. Each data in the batch is sent individually at 0.01-second intervals to mimic the conditions of continuous data flow as in an actual production system. With this approach, the Producer not only functions as a data sender, but also as a realistic streaming flow simulator.

After the data is processed at the producer stage, the data stream is received by the consumer for incremental and real-time clustering using the DBSTREAM algorithm and will be implemented online using the River library. This algorithm processes each incoming data incrementally, forming new clusters or updating existing clusters based on the data density structure. This algorithm is designed to update the model incrementally in a partial-fit manner, rather than periodically in mini-batches. Each time new data is received, the model immediately updates the micro-cluster structure without waiting for a certain amount of data. The update is performed by directly adjusting the position of the density weight centroid of each micro-cluster, based on the Euclidean distance between the new data and the existing centroid. In addition to updating the centroids, the weights of each micro-cluster are also updated using a fading function to reduce the influence of old data on the model. This mechanism allows the model to continuously adapt to changes in data patterns (concept drift) while maintaining computational efficiency because it does not require full retraining as in batch learning methods. This application is fully incremental, fully online, and works based on the principle of partial-fit per instance date.

In this study, the clustering results obtained by DBSTREAM were further processed using Hierarchical Agglomerative Clustering (HAC) to combine similar and stable clusters at the centroid level. HAC is a hierarchical clustering method that works by repeatedly merging the smallest clusters until the desired number of clusters is reached or until the distance between clusters exceeds a certain threshold [17]. This merging process in this study was carried out using the Euclidean Distance approach, which calculates the distance between two cluster centroids using the formula in equation (3):

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \quad (3)$$

From equation (3), where a and b are the centroid vectors of the two clusters being compared. The use of Euclidean Distance allows clusters to be merged based on objective geometric proximity in feature space. This approach is applied to prevent over-clustering, a condition where the model produces too many clusters, resulting in biased or difficult-to-interpret behavior patterns. The HAC process will

be run periodically in the offline phase. In this application, HAC will be run with the same scheme as in the evaluation stage, where it will be run periodically every 200 data points processed. The final result of the HAC process is a stabilized macro-cluster that is then stored in a database for behavioral analysis, visualization, or other downstream analytics processes.

From the OML process with DBSTREAM that produces clusters and has been processed by cluster merging using HAC, to test the quality of clusters produced by the model. The evaluation process will be carried out every time 200 data have been entered and stored in the data buffer. The evaluation includes Silhouette Coefficient and DBI. The Silhouette Coefficient measures how similar an object is to its cluster compared to other clusters. Values range from -1 (incorrect cluster) to +1 (good cluster), with values close to 1 indicating good cluster separation [18]. The mathematical process of the Silhouette Coefficient is as shown in equation (4):

$$s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))} \quad (4)$$

The description in equation (4) states that the value of $a(i)$ is the average distance between point i and all points in the same cluster, and $a(i)$ is the average distance between point i and all points in the nearest cluster that is not cluster i.

DBI evaluates the ratio between intra-cluster dispersion (compactness) and inter-cluster distance. The smaller the DBI value, the better the clustering results. This matrix represents how dense and separate the clusters are [19]. The DBI evaluation calculation process is as shown in equation (5):

$$DBI = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{S_i+S_j}{M_{ij}}\right) \quad (5)$$

Equation 5, k is the number of clusters formed, S_i is the average distance from all points in cluster k to its centroid, and M_{ij} is the distance between the centroid of cluster i and cluster j.

After all processes from Consumer have been completed and the information has been stored in the database, the results will be further analyzed in the dashboard. This dashboard will display all information such as clustering result visualizations, processed data and clustering results, calculated evaluation results, and other related data information.

III. Result And Discussion

The system developed in this study has been successfully implemented and tested in accordance with the design described in the methodology section. The initial testing phase was conducted on the producer component, which functions to retrieve the latest data from the database, perform preprocessing, and send it in real time via RabbitMQ. Based on the test results shown in Figure 3, the system successfully displayed data that had undergone encoding and scaling processes and streamed it continuously to the OML process on the consumer side. This proves that the streaming data pipeline runs stably and according to design.

Figure 3. The producer process that has successfully collected data, preprocessed it, and sent it in real time using RabbitMQ.

In Figure 3 above, it can be seen that the data has been successfully processed or preprocessed and successfully sent in streaming and real-time using RabbitMQ, which can then be continued for core OML processing in the consumer. In the consumer process, a series of processes have been successfully carried out, such as receiving data sent from the producer, then performing real-time and incremental clustering. The HAC implementation scenario used to combine similar clusters has also run as intended. The results of the OML and HAC process testing are shown in Figure 4 below.

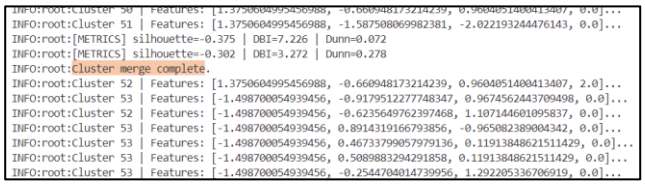

Figure 4. Logging of OML results, cluster margins, and evaluation

Based on the logging process shown in Figure 4, each new piece of data that comes in is immediately processed and integrated into the model. The HAC process used to combine similar micro-clusters also runs as it should. The evaluation results show that for the 400th data point, the Silhouette value was 0.375 and the DBI was 7.226, while for the 600th data point, the Silhouette value decreased to 0.302 and the DBI improved to 3.27. This pattern shows that the model is able to adapt to data dynamics gradually and consistently. For latency and throughput test results, the average time for the algorithm to process one data point is around 37 ms and the throughput is 22 data points per second. This shows that processing each

data point with a significant increase in quantity can be completed fairly quickly.

For further analysis, a dashboard was also created and tested by taking all stored information, such as cluster results with their variables. This data is very important for further business analysis. The processed and stored data is shown in Figure 5 below.

Figure 5. Data stored from the OML process for analysis in the dashboard

The OML process results are then stored in MongoDB for further analysis. Figure 5 shows the structure of the stored data, which contains evaluation values, cluster labels, and model variables. This data is then visualized through a Streamlit-based monitoring dashboard as shown in Figures 6 and 7.

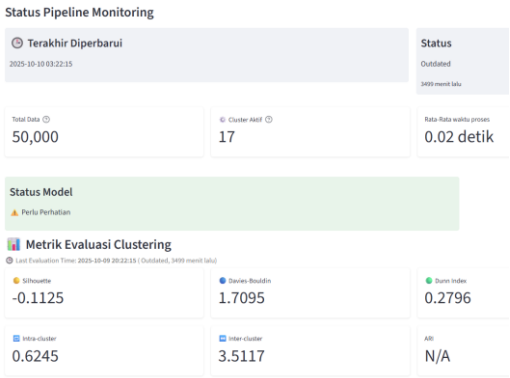

Figure 6. Main view of the monitoring dashboard

The dashboard serves as a visual interface that displays the current segmentation status, the amount of data that has been processed, the processing time per data, and model evaluation metrics. The two-dimensional visualization displayed on the dashboard provides an intuitive overview of the cluster distribution and allows administrators to monitor the segmentation quality in real-time.

Figure 6 above shows the main dashboard displaying information related to the segmentation status, including the last data update, the amount of data processed, and the processing time per data point. Next are the evaluation metrics that display the results of the Silhouette and DBI evaluation calculations. This dashboard clearly provides an overview of the performance of the OML model over time, which allows you to directly monitor the quality of the model being used for this clustering process. This can be used to adjust the model parameters in the future so that the model parameters are also adaptive, resulting in optimal clustering results.

Figure 7. Clustering plot visualization in the main dashboard

Figure 7 above shows the two-dimensional distribution of the clustering results using the DBSTREAM algorithm, after going through the cluster merging stage with HAC. Each point on the graph represents an e-commerce user mapped based on two main features resulting from dimension reduction using the Principal Component Analysis (PAC) method, namely Feature 1 and Feature 2, which describe the representation of user behavior after going through the data preprocessing and normalization stages. The color variation of the points indicates cluster membership, where the blue color gradient signifies the differences in cluster identity formed because of the model learning the patterns of similarity in user behavior.

From the visualization, the data distribution does not form an extremely separate pattern, but shows several dense clusters formed vertically around Feature 1 range between 0.1 and 0.3. This pattern indicates that most users have similar behavioral characteristics in that dimension, such as frequency of interaction or similar types of activities. Meanwhile, the data distribution with lighter colors and more scattered points depicts users with relatively unique or inconsistent behavior, which in the context of OML is considered an outlier. These results are consistent with a Silhouette value close to zero, indicating that the model is able to form distinct clusters without significant overlap, but with relatively close cluster boundaries. This shows that the segmentation process has successfully grouped users into several stable main behavioral groups, with sufficient density and separation to support further consumer behavior analysis.

To test the model's performance, a model parameter experiment will be conducted to see the evaluation results and clusters generated from each model parameter and HAC tried. Because streaming data is dynamic, cluster profiles can change over time (concept drift), therefore the testing and experiments conducted in this study are representative (representative of the test data period). This experiment will

only be attempted with 50,000 data points processed from the overall data taken based on the data ID. The parameters that produce a Silhouette close to one, the smallest DBI, and the minimum cluster results obtained will be used as the main model parameters for further analysis. The experiment format is as shown in Table II.

TABEL II
EXPERIMENT PARAMETER MODEL FOR EVALUATION AND CLUSTERING RESULTS

Parameters			Evaluation		Clustering Results
Cluster treshold	Fading factor	HAC Treshold	Silhoute	DBI	
0.2	0.0005	0.5	-0.3415	0.1759	14
0.2	0.005	0.5	-0.3809	2.4247	13
0.5	0.0005	0.5	-0.3305	0.2888	14
0.5	0.005	0.5	-0.3809	0.2918	13
0.8	**0.0005**	**0.5**	**-0.1125**	**0.2796**	**17**
0.8	0.005	0.5	-0.2167	0.2690	13

Based on the results of testing model parameters as shown in Table 2, the most optimal configuration was obtained at a cluster threshold of 0.8 and a fading factor of 0.0005, with a Silhouette evaluation value of -0.1125 and a DBI of 0.2796. A negative Silhouette value generally indicates overlapping clusters, where some data is located at the decision boundary between clusters. This phenomenon can be explained by the non-stationary nature of streaming data, where the distribution of data continues to change over time, as well as by the incremental mechanism of DBSTREAM, which updates the model based on each new data point (partial fit). In this context, Silhouette does not fully reflect poor cluster separation, but rather illustrates the model's continuous adaptation to the dynamics of the data distribution, which had not yet fully converged at the time of evaluation. The results of this experiment also show that a larger cluster threshold produces broader clusters and does not quickly separate the data into many small micro clusters. As for the fading factor variable, the larger the value of this parameter, the faster the model forgets old clusters, causing clusters to change quickly. Conversely, if the value is smaller, the model's computation will be slow and heavy because it will always remember the old clusters.

Meanwhile, the low DBI value (0.2796) indicates that the intra-cluster density is relatively good and the distance between clusters is sufficiently separated. The difference in results between these two metrics occurs because of their different measurement characteristics. Silhouette is more sensitive to the position of data at the cluster boundary, while DBI focuses more on the density and distance between cluster centers. Thus, the combination of a slightly negative Silhouette and a low DBI can be interpreted as the model still being in the process of dynamically adjusting to new data flows, but overall successfully forming a compact cluster structure that is not randomly distributed. This reinforces that the OML approach with DBSTREAM is able to maintain modeling stability in a constantly changing data environment, even though it does not always achieve perfect separability at every point in time.

From this experiment, the model's stability will be tested against the concept drift phenomenon, which is analyzed based on changes in Silhouette and DBI values over time. As shown in Figure 8 below:

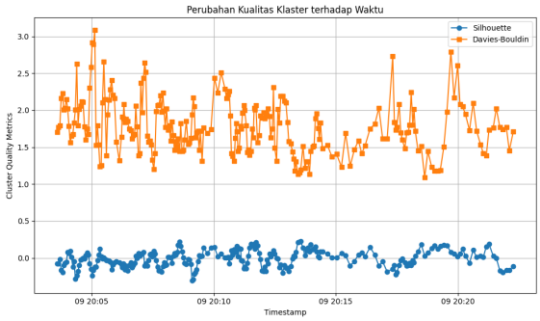

Figure 8 visualization of model performance in handling concept drift

From the visualization results in Figure 8, it can be seen that the Silhouette value tends to fluctuate around a value close to zero, while DBI varies between 1 and 3. This fluctuation pattern shows that the DBSTREAM model is able to adapt to changes in data distribution gradually, while maintaining a consistent cluster structure. Although there are local changes in evaluation values over time, no extreme decline in performance is found, indicating that the system successfully stabilizes the cluster formation process despite shifts in user behavior patterns or concept drift. These results indicate that the streaming data pipeline system approach with the DBSTREAM algorithm is capable of effectively handling non-stationary data flows. The model can incrementally update the cluster structure without losing historical context, while maintaining a balance between adaptivity and stability in the face of streaming data dynamics.

Model performance analysis will also be reviewed from the data addition process. This analysis will examine whether an increase in the amount of data processed will affect the number of clusters and model evaluation, and how the cluster merging process using HAC periodically aligns with the cluster formation process. The results of this analysis are shown in Figure 9.

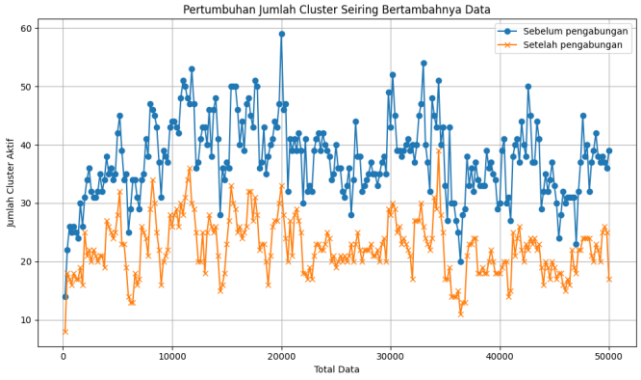

Figure 9. Growth in the number of clusters per data mining

Figure 9 shows the growth trend in the number of active clusters as the amount of data processed by the OML system increases. The blue line represents the number of clusters before merging using HAC, while the orange line shows the number of clusters after merging. In general, it can be observed that the number of clusters tends to increase in the early stages of the process when the model begins to form new behavior representations based on continuously incoming user data, with the highest cluster peak at 2000 data points. However, as the data volume exceeds 20,000, the rate of cluster growth begins to slow down and tends to stabilize at a certain range. This phenomenon indicates that the model begins to recognize recurring behavior patterns among e-commerce users, thereby reducing the need to form new clusters.

After implementing the cluster merging process using HAC, it was observed that the number of active clusters decreased significantly compared to before the merging. This was due to the HAC mechanism, which periodically merges clusters that have high proximity in feature space based on the Euclidean Distance metric. This merging process not only reduces cluster redundancy, but also stabilizes the model structure so that it does not experience over-clustering, which is a condition where the model forms too many small clusters that are not significant in behavior or form noise. The downward trend in the number of clusters after merging shows that the system successfully maintains representation efficiency without losing important information variation.

Figure 10 shows the results of the cluster distribution analysis formed from the DBSTREAM algorithm process. From the segmentation results of 50,000 test data, four main clusters with different levels of dominance were obtained. Cluster 0 is the largest group with a total of 36,325 data or about 72.73% of the total population, indicating that most e-commerce users have similar behavior patterns. Meanwhile, Cluster 2 and Cluster 3 occupy the next positions with percentages of 14.21% and 7.01%, respectively, and Cluster 5 accounts for 4.07% of the total data. The dominance of data in Cluster 0 indicates the existence of a group of users with relatively homogeneous activities, such as high interaction behavior towards certain products without significant variation in activities.

Cluster ID	Jumlah Data	Persentase (%)	User ID Dominan	Item ID Dominan	Category ID Dominan	Behavior Type Dominan
0	36325	72.73	0	0	0	pv
2	7096	14.21	0	0	0	pv
3	3502	7.01	0	0	-1	pv
5	2035	4.07	0	0	-1	pv

Figure 10. Analysis of the cluster results formed

From the analysis of the cluster results in Figure 10, when viewed from the Dominant Behavior Type variable, all clusters show a tendency for pv (page view) behavior as the dominant activity. This indicates that the majority of user interactions on e-commerce are still in the exploratory stage, namely browsing products without taking further actions such as adding to cart or purchasing. Meanwhile, in the Dominant Category ID variable, the value -1 in several clusters indicates the existence of categories that are not specifically identified, possibly due to limitations in the categorization attributes in the data. Thus, although the behavior analysis is still general in nature, these clustering results provide an initial overview of user clustering based on the intensity and type of activities carried out in real-time. In the future, the addition of behavioral variables such as transaction frequency or visit time is expected to enrich the interpretation of e-commerce user interaction patterns in a more in-depth and contextual manner.

System testing was also conducted by comparing the DBSTREAM algorithm with DenStream, both of which process based on data density. The comparison between the two algorithms is shown in Table III below:

TABEL III
COMPARISON BETWEEN THE DBSTREAM AND DENSTREAM ALGORITHMS

Method / Metrik	DBSTREAM	DenStream
Silhouette Score (periodik)	-0.1125	-0.5664
Davies–Bouldin Index (periodik)	0.2796	0.1561
Number of clusters	17	10
Throughput (data/s)	22	26
Latency per instance (ms)	37	44
Memory usage (MB)	81	127
Storage footprint (MB)	8	8

Testing results for both density-based streaming algorithms show slightly significant differences in terms of cluster quality and computational efficiency. Based on periodic evaluation results, the DBSTREAM algorithm obtained a Silhouette value of -0.1125 and a DBI of 0.2796, while DenStream produced a Silhouette of -0.5664 and a DBI of 0.1561. The DBSTREAM Silhouette value, which is closer to zero, indicates that the resulting cluster structure is relatively more separate and adaptive than DenStream, even though both face data distribution dynamics due to the non-stationary nature of streaming. On the other hand, the lower DBI value in DenStream indicates a higher level of intra-cluster density, but this also leads to the possibility of forming clusters that are too dense and homogeneous, thereby reducing the model's ability to capture a wider range of user behavior variations.

In terms of efficiency, DenStream shows higher throughput of 26 data/s but with the consequence of greater latency per instance of 44 ms and higher memory usage of 127 MB. Conversely, DBSTREAM is able to maintain lower latency of around 37 ms and more efficient memory consumption of 81 MB with a larger number of clusters, namely 17 clusters, indicating that this algorithm is more efficient in utilizing resources for incremental model updates. Additionally, both algorithms have the same storage footprint of 8 MB, indicating efficiency in storing data summaries or micro-clusters.

These results show that DBSTREAM excels in maintaining a balance between cluster stability, memory efficiency, and processing latency, while DenStream tends to produce more compact clusters that are less adaptive to

changes in streaming data distribution. These findings reinforce the selection of DBSTREAM as the primary algorithm in the Online Machine Learning pipeline designed for dynamic and continuous data environments.

From the overall results of the system testing discussed, the data streaming pipeline system with the DBSTREAM algorithm is capable of performing continuous clustering and can handle significant data changes in terms of concept drift. From testing with the batch learning method using the DBSCAN algorithm, 16 clusters were obtained from 50,000 data points, and the Silhouette evaluation result was 0.1411. The clusters produced from batch learning are quite stable because the entire clustering process is performed on a single dataset snapshot. This means that when new data enters or user behavior patterns change, DBSCAN must be retrained from scratch, which increases computational costs and processing time. In contrast, with the streaming data approach, the model is able to adapt to concept drift. This makes DBSTREAM more relevant for dynamic e-commerce environments, where user interaction patterns are constantly changing. Batch approaches such as DBSCAN have a time complexity of O(n^2) because they calculate the distance between all data pairs. On a large scale or with continuously growing data streams, this is inefficient and difficult to operate in real time. In contrast, DBSTREAM only updates active micro-clusters using a fading function, which makes its complexity close to linear with respect to the amount of active data.

Numerically, DBSCAN shows stable clustering results with a Silhouette value of 0.1411 and 16 clusters, while DBSTREAM produces a value of –0.1125 with a dynamically changing cluster structure. However, the main advantage of DBSTREAM lies in its ability to operate online, updating the model each time new data is received without retraining, while maintaining time and memory efficiency. Thus, although the static evaluation of DBSCAN appears to be better, the incremental approach of DBSTREAM is far superior conceptually and practically for constantly evolving data environments such as e-commerce.

IV. CONCLUSION

This research aims to create a streaming data pipeline for e-commerce user segmentation using an OML approach based on the DBSTREAM algorithm. The data used for streaming simulation comes from Taobao e-commerce. From the results of system design and development testing, the process of data collection, preprocessing, and streaming delivery has been successfully implemented in the producer process. Then, in the consumer process, cluster formation has been successfully carried out incrementally and has produced adaptive clusters with a periodic HAC merging mechanism, which minimizes over-segmentation as seen from the analysis results before and after the HAC process runs. The results of the model parameter experiment showed that the ideal parameters were a cluster threshold in the range of 0.6 to 0.8 and a fading factor of 0.0005 or even smaller, such as 0.0003. For the evaluation results, the model successfully ran OML incrementally with a silhouette score of -0.1125 and a DBI of 0.2796. This research successfully proves that the streaming data pipeline system that was built can operate stably in processing data to produce clusters that are adaptive to concept drift, which is continuously updated with a processing throughput of 22 data per second and an average latency of 37 ms. The DBSTREAM model that is run in the system consumes an average of 81 MB of memory resources and 8 MB of storage footprint.

However, limited information on several variables, such as Item ID and Type ID, which are numerical without semantic context, limits the ability to analyze user behavior in a more specific and qualitative manner. Therefore, further research is recommended to enrich user behavior attributes, such as transaction frequency, visit duration, or user interaction types, so that the resulting segmentation is not only structural but also provides a more in-depth and applicable interpretation of behavior in the context of e-commerce, as well as making model parameters more adaptive so that dynamic model parameters follow dynamic data developments, resulting in more optimal clusters.

REFERENCES

[1] E. Febrianty, L. Awalina, and W. I. Rahayu, "Optimalisasi Strategi Pemasaran dengan Segmentasi Pelanggan Menggunakan Penerapan K-Means Clustering pada Transaksi Online Retail Optimizing Marketing Strategies with Customer Segmentation Using K-Means Clustering on Online Retail Transactions," *Jurnal Teknologi dan Informasi (JATI)*, vol. 13, 2023, doi: 10.34010/jati.v13i2.

[2] F. Helmi, "Analisis Perilaku Pelanggan E-Commerce Menggunakan Model Klustering Dengan Algoritma K-Means," 2024. [Online]. Available: http://repository.unas.ac.id/10631/

[3] R. Siagian, P. S. Pahala Sirait, and A. Halima, "E-Commerce Customer Segmentation Using K-Means Algorithm and Length, Recency, Frequency, Monetary Model," *Journal Of Informatics And Telecommunication Engineering*, vol. 5, no. 1, pp. 21–30, Jul. 2021, doi: 10.31289/jite.v5i1.5182.

[4] J. Wu *et al.*, "An Empirical Study on Customer Segmentation by Purchase Behaviors Using a RFM Model and K -Means Algorithm," *Math Probl Eng*, vol. 2020, 2020, doi: 10.1155/2020/8884227.

[5] C. Hafidz Ardana *et al.*, "Segmentasi Pelanggan Penjualan Online Menggunakan Metode K-means Clustering," 2024.

[6] A. S. M. S. Hossain, "Customer segmentation using centroid based and density based clustering algorithms," in *2017 3rd International Conference on Electrical Information and Communication Technology (EICT)*, 2017, pp. 1–6. doi: 10.1109/EICT.2017.8275249.

[7] S. Shalev-Shwartz, "Online learning and online convex optimization," 2011. doi: 10.1561/2200000018.

[8] V. Abeykoon *et al.*, "Streaming Machine Learning Algorithms with Big Data Systems," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 5661–5666. doi: 10.1109/BigData47090.2019.9006337.

[9] M. Tareq, E. A. Sundararajan, M. Mohd, and N. S. Sani, "Online clustering of evolving data streams using a density grid-based method," *IEEE Access*, vol. 8, pp. 166472–166490, 2020, doi: 10.1109/ACCESS.2020.3021684.

[10] M. Fakhrun Nuha, Q. M. Baligh Ghoni, and A. Zarfan Shabirin, "Analisis Pola Perilaku Konsumen E-Commerce Menggunakan

Ensemble Learning: Studi pada Brazilian E-Commerce Public Dataset by Olist."

[11] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. De Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput Surv*, vol. 46, no. 1, Jan. 2013, doi: 10.1145/2522968.2522981.

[12] M. Hahsler and M. Bolaos, "Clustering Data Streams Based on Shared Density between Micro-Clusters," *IEEE Trans Knowl Data Eng*, vol. 28, no. 6, pp. 1449–1461, Jun. 2016, doi: 10.1109/TKDE.2016.2522412.

[13] S. Zhang, J. Liu, and X. Zuo, "Adaptive Online Incremental Learning for Evolving Data Streams," Jan. 2022, doi: 10.1016/j.asoc.2021.107255.

[14] E. Bartz and T. Bartz-Beielstein Editors, "Machine Learning: Foundations, Methodologies, and Applications Online Machine Learning A Practical Guide with Examples in Python."

[15] M. Guntara and F. D. Astuti, "Komparasi Kinerja Label-Encoding dengan One-Hot-Encoding pada Algoritma K-Nearest Neighbor menggunakan Himpunan Data Campuran," *JIKO (Jurnal Informatika dan Komputer)*, vol. 9, no. 2, p. 352, Jun. 2025, doi: 10.26798/jiko.v9i2.1605.

[16] P. Palinggik Allorerung, A. Erna, M. Bagussahrir, and S. Alam, "Analisis Performa Normalisasi Data untuk Klasifikasi K-Nearest Neighbor pada Dataset Penyakit," 2024.

[17] D. Destin *et al.*, "Mapping districts in West Java by under-five pneumonia indicators: an agglomerative hierarchical clustering study (Open Data Jabar 2023)," *Commun. Math. Biol. Neurosci.*, vol. 2025, no., Jan. 2025, doi: 10.28919/cmbn/9547.

[18] Y. Hasan, "Pengukuran Silhouette Score Dan Davies-Bouldin Index Pada Hasil Cluster K-Means Dan Dbscan," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 3S1, Oct. 2024, doi: 10.23960/jitet.v12i3S1.5001.

[19] M. Sholeh and K. Aeni, "Perbandingan Evaluasi Metode Davies Bouldin, Elbow dan Silhouette pada Model Clustering dengan Menggunakan Algoritma K-Means," *STRING (Satuan Tulisan Riset dan Inovasi Teknologi)*, vol. 8, no. 1, p. 56, Aug. 2023, doi: 10.30998/string.v8i1.16388.