Application of Improved density-based Algorithms to Data Stream and Performance Evaluation

by

AKINOSHO, Tajudeen Akanbi

(58300481)

A dissertation submitted in accordance with the requirements for the degree of

MASTER OF SCIENCE

In the subject

COMPUTING

at the

UNIVERSITY OF SOUTH AFRICA

Supervisors:

[+]Prof Wang Zenghui.

&

[*]Mr. Elias Tabane

OCTOBER 2023

[+]wangz@unisa.ac.za
[*]tabane@unisa.ac.za

Application of Improved Clustering Algorithms to Data Stream and Performance Evaluation

By

AKINOSHO, Tajudeen Akanbi

Abstract

Density-based algorithms are effective in the detection of clusters with arbitrary shapes and outliers even when information about the number of clusters is not available. Parameter specification in data stream clustering remains a challenge. Selecting a suitable parameter-tuning is germane in having a good clustering quality. The density-based algorithm *DenStream* is an example of data stream clustering algorithms that require several parameter specifications. In this dissertation, an improved *DenStream* with a modified distance measure was proposed and demonstrated with parameter-tuning in Massive Online Analysis (MOA) using synthetic and real-world datasets. The modified *DenStream* algorithm was compared against *CluStream, ClusTree* and *DenStream* in the presence of noise levels 0%, 10%, and 30% and manually selected epsilon parameters 0.02, 0.03, and 0.05 respectively. The epsilon parameter range [0.02 – 0.05] was not used due to some algorithm not working on real-world datasets. The effects on clustering qualities were evaluated and demonstrated using performance evaluation metrics CMM, Purity, Silhouette Coefficient, and Rand index on the synthetic and real-world datasets. Finally, the result shows that effectiveness of the algorithms depends on the parameter-tuning and no single algorithm is a one-size-fits-all for the performance metrics.

Keywords: data stream clustering, stream clustering, data stream, clustering, MOA, clusters, CluStream, DenStream, ClusTree, modified DenStream, arbitrary shape

Acknowledgments

All praise and adoration be to Allah (SWT) for HIS Grace over me and for granting me the opportunity to complete this research after my medical health challenges. I am grateful to Allah (SWT) that I was able to successfully attain my goal of finishing my master's degree.

I thank my supervisors, **Prof. Zenghui Wang**, and **Mr. Elias Tabane** for their wonderful support to secure a bursary during my period of need and for their encouragement to do and complete this research most especially during my medical challenges.

I appreciate my parents' late **Alhaji Musibaudeen Akinosho** and my mother **Alhaja Titilayo Akinosho (Nee Onitiri)** for their love, prayers, and support. Likewise, I appreciate my pillar of support, my love, and my treasure island**, Dr. Maryam Mojisola Bello-Akinosho** for her continuous support, love, and kindness before, during, and after my medical challenges. She pushed me to the limit to achieve this dream academic height. I also appreciate my children (**Abdus-Samad, Rahmatullah, and Naeemullah**) for their support and understanding at home to complete this academic task.

Everyone needs support and I found that embedded in my family. Over the years, I have benefitted from the support of my family. You all came together in my difficult times, mostly during and after my medical challenges. I appreciate my siblings, the **Akinoshos** from Ikija, Abeokuta in Ogun state, Nigeria, and the family of my wife the **Bellos** from Agbole Orunkan, Owu kingdom in Ogun state, Nigeria., for their financial and moral support and prayers.

I would also like to recognize the unflinching support of the following individuals and families: **Dr** and **Dr. (Mrs.) Adebesin**, **Dr.** and **Mrs. Raji, Prof. Rasheed** and **Mrs. Adeleke, Prof Bilamin Oboiren** and **Dr. Fatima Oboiren, Mr. Idris, Prof Saheed Oke** and **Mrs. Oke. Dr. Kashimawo Afolabi** and **Mrs. Afolabi.**

I appreciate the positive contributions of the National Institute for Theoretical Computational Sciences (NiTheCS) led by **Prof. Francesco Petruccione** and his team most especially **Binjamin Barsch** for providing an in-depth analysis of issues during his regular training sessions that improved and reshape my knowledge and understanding. I also appreciate **Mrs. Rene Kotze (NiTheCS)** for regularly sending me important messages about NiTheCS training, workshops, colloquium, and mini school.

Declaration of Authorship

Name: AKINOSHO, Tajudeen Akanbi

Student number: 58300481

Degree: Master of Science in Computing

"*Application of Improved Clustering Algorithms to Data Stream and Performance Evaluation*"

I declare that the above dissertation is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

I further declare that I submitted the dissertation to originality checking software and that it falls within the accepted requirements for originality.

I further declare that I have not previously submitted this work, or part of it, for examination at Unisa for another qualification or at any other higher education institution.

October 4, 2023

SIGNATURE DATE

To my family for all their love.

Table of Contents

List of Figure

List of Tables

List of abbreviations

ACM	Advance Cluster Maintenance
AIS	Artificial Immune System
ARFF	Attribute Relation File Format
ARI	Adjusted Rand Index
ATM	Automated Teller Machine
BOCEDS	Buffer-based Online Clustering for Evolving Data Stream
CEC	Clustering Evolving data streams based on the adaptive Chebychev distance.
CEDAS	Clustering Evolving Data streams into Arbitrary Shaped
CEDGM	Clustering of Evolving Data via a density Grid-based Method
CF	Cluster-Feature
CLARA	Clustering Large Applications
CLARANS	Clustering Large Applications Based Upon Randomized Search
CMC	Core Micro-Cluster
CMM	Clustering Mapping Measure
CODAS	Clustering Online Data-streams into Arbitrary Shapes
DBSCAN	Density Based Spatial Clustering of Applications with Noise
DENCLUE	DENsity-based CLUstEring
DenSOINN	Density Based Self Organizing Increment Neural Network
DFM	Dynamic Feature Mask
DFPS-C	Dynamic Fitness Proportionate Sharing Clustering
EDMStream	Evolution of Density Mountain stream clustering algorithm
ELKI	Environment for Developing KDD-Applications Supported by Index-Structures
EM	Expectation Maximization
HAC	Hierarchical Agglomerative Clustering
ICER	Irish Commission for Energy Regulation
ICFR	Incremental Clustering using F-value Regression analysis.
IoT	Internet of Things
MDSC	Multi-density Stream Clustering
MOA	Massive Online Analysis

NMI	Normalized Mutual Information
ODAC	Online Divisive Agglomerative Clustering
OpStream	Optimized Stream Clustering algorithm.
PAM	Partitioning Around Medoids
PyOD	Python Outlier Detection
RFID	Radio Frequency Identification
SOM	Self-organizing feature maps
SSQ	Sum of Squared Distance
STING	STatistical INformation Grid
SWEM	Sliding Window with Expectation Maximization
WOA	Whale Optimisation Algorithm

CHAPTER 1: Introduction

1.0 Introduction and background

The world is witnessing data revolution generated daily in large quantities. Lately, the volume, speed, and diversity of data from network devices and online resources have increased astronomically over time. This streaming data is unceasing, potentially unbounded, and continuously evolving (Abid et al., 2019). Extracting knowledge from such important data can lead to significant improvements in business, technology, finance, politics, economics, international relations, and other sectors of society. However, to maximize the benefits of continuous streams of data and their dynamic evolving nature in a dynamic environment, sophisticated data mining tools are required. A data stream is unlike the traditional static database as it is generated from sensor networks, weblogs, radio frequency identification (RFID), Twitter streams, health monitoring systems, connected Internet of Things (IoT) devices, mobile devices, and Automated Teller Machine (ATM) transactions (Agrawal & Adane, 2016). The continued availability of data has provided compelling reasons for developing tools to extract valuable knowledge. Data stream exploration has developed into a significant research interest in the data mining community due to the increasing generation of streaming information and importance of its applications (Agrawal & Adane, 2016; Amini et al., 2014; Tareq et al., 2022). Data stream clustering poses many challenges in the literature, such as detecting clusters of arbitrary shape, grouping data streams into several clusters with no prior knowledge, and preserving clusters dynamically. Several clustering algorithms are classified into six major groups: *Density-based, Hierarchical-based, Partitioning, Grid-based, Model-based,* and *Graph-based clustering.* The density-based algorithms is an important and reliable clustering algorithms for the detection of arbitrary-shaped clusters (Amini et al., 2016).

Researchers recently proposed numerous density-based clustering algorithms including Clustering Online Data-streams of Arbitrary Shapes (CODAS) (R. Hyde & Angelov, 2015); an improved version of CODAS (i-CODAS) (Islam et al., 2019b); Clustering of Evolving data streams based on Chebychev distance with false Merging (CEC-Merge) (Tareq & Sundararajan, 2020); Buffer-based Online Clustering for Evolving Data Stream (BOCEDS) (Islam et al., 2019a); and Clustering of Evolving Data-streams into Arbitrary Shapes (CEDAS) (R. Hyde et al., 2017). Although outlier detection in data streams poses many

challenges (Togbe et al., 2020; Tran et al., 2020; Yao et al., 2018), detecting clusters with multi-density data also remains a major challenge (Amini et al., 2016).

Outliers are categorized into three main groups: *global/point*, *collective*, and *contextual* outliers (S. Sadik & Gruenwald, 2014; Thakkar et al., 2016). In some instances, outliers/anomalies are classified into three main groupings: *unsupervised*, *semi-supervised,* and *supervised outlier detection* (Goldstein & Uchida, 2016; Han et al., 2012; Pawar et al., 2014) dependent on whether data are labeled or unlabeled. There are several algorithms proposed for multi-density data clustering in the literature, but these suffer some disadvantages: (1) some need more than a single pass; (2) some require the whole data; and (3) some have high computational time. The density-based clustering algorithm is dependable in arbitrary shape and noise detection. In this dissertation, the focus is to demonstrate the effects of adjusting parameters on data stream clustering algorithms.

The Massive Online Analysis (MOA) is an open-source and leading tool for the analysis and extracting of knowledge from streaming data and unsupervised outlier detection (Bifet et al., 2018). Arguably, MOA is the most popular framework for data stream mining with a wide range of algorithms and machine learning (ML) tools for classification, clustering, regression, outlier detection, multi-label, multi-target concept drift detection, feature analysis, and experimenter systems. It has a graphical user interface (GUI) and a workflow like the Waikato Environment for Knowledge Analysis (WEKA). There are several density-based algorithms for detecting arbitrary shape clusters and noise like CODAS and CEDAS. However, the focus of this study is to demonstrate the effects of adjusting parameters on data stream clustering algorithms CluStream, DenStream, and ClusTree found in the Massive Online Analysis (MOA) framework.

1.1 Problem Statement

In real-time data streaming, clustering analysis is vital to gain valuable knowledge from the streaming data. While several data stream clustering algorithms are available, there is no ‘one-size-fits-all algorithm for all’ types of problems and data sets. Several data stream clustering algorithms suffers some deficiencies (Ahmed et al, 2020; Kokate et al., 2018). There are several open challenges of data stream clustering. Zubaroğlu and Atalay (2020) identified the challenges as (1) Finding the number of clusters (2) Parameter specification (3) Lack of *de facto* evaluation criteria (4) Lack of high-quality benchmark data (5) No availability of one-size-fits-all algorithms system platforms (6) Lack of algorithms for both quantitative and categorical data. Parameters are often difficult to determine, notably for high-

dimensionality datasets (Han et al., 2012 p.446). The parameter settings of Density Based Spatial Clustering of Applications with Noise (DBSCAN) are problematic (Guan et al., 2019). Cao et al. (2006) tested the sensitivity of parameters on *DenStream* which is of interest in this research. Zubaroğlu and Atalay, (2020, p.1226) stated that parameters such as *k*, density threshold, decay rate, window length, and distance threshold are very susceptible to the input data and affect the clustering quality. Bahri et al. (2022) stated that in evolving data streams, algorithm choice and hyper-parameter tuning is tasking for non-experts because it requires the domain knowledge and human expertise in achieving optima results. This is of interest in this dissertation by manually looking at appropriate parameters best for the datasets.

1.2 Research Aim

Parameter fine-tuning is very challenging in data stream clustering. The aim of this dissertation is to implement a modified *DenStream* and investigate the sensitivity of the parameter settings against the original *DenStream* and other known bench algorithms (CluStream and ClusTree) in MOA. The clustering quality of the modified *DenStream* was demonstrated in MOA using performance evaluation metrics *Clustering Mapping Measure* (CMM), Purity, Silhouette Coefficient, and Rand index. The experimental results of the algorithms were merged and visualized using Python libraries (pandas and hvplot) for proper interpretation.

1.2.1 Research Objectives

The research objectives for this dissertation are thus:

(1) To identify the method of building a MOA repository from source to implement the modified algorithm.
(2) To identify the hyperparameters appropriate for parameter-tuning.
(3) To identify the performance metrics applicable for clustering quality.

1.3 Motivation

Mode of data streams are constantly evolving and arriving at a fast rate. The sources include smart devices, sensor networks, social media platforms, and financial data, among others. There is a need to gain useful insights into a large data stream. Many of the suitable algorithms for extremely high-frequency data stream clustering suffer some limitations (Ahmed et al., 2020; Haneen et al., 2018). The choice of parameter settings is a challenge

(Carnein et al., 2020c); while varying the density of data streams is relatively hard (Cao et al., 2006). Setting the appropriate parameters in data stream clustering requires domain knowledge and human expertise. Fortunately, an open-source data analytic tool, Massive Online Analysis (MOA), is available to perform the data stream clustering where the parameters are manually fixed, and evaluation performance metrics selected. This makes it appropriate for use in this study. It is envisaged that this research will achieve remarkable results and find usefulness in density-based clustering applications.

1.3.1 The Impact of this Research on Society

Data are unceasingly generated in society in diverse areas such as financial transactions, telephone calls, radio frequency identification (RFID), telecommunications, sensor monitoring, weblog clicks, weather monitoring, recommender systems, medical diagnoses, real-time surveillance, electricity usage prediction, and epidemics/disaster management. It is important to analyze these datasets as soon as they are generated and extract knowledge from them for a predictive purpose, as an example. One impact of our approach is the ability to predict possible future trends based on continuously generated instances. Once a clustering algorithm has learned how to categorize datasets, then such an algorithm can serve the purpose of a predictive tool when required. Such algorithms are very useful, for example, in predicting future load requirements based on online monitoring of electricity usage. Another possible application is detecting fraud in financial transactions or threats to cyber infrastructure. The application of unsupervised learning techniques, such as clustering, is important to any aspect of society where data is being generated continuously.

1.4 Overview of Methodological Approach

The methodology adopted in this dissertation to investigate the data stream clustering algorithms was by means of three methods using MOA framework in Figure 1-1. Firstly, the default parameters of the algorithms on MOA using the *RandomRBFGenerator* for synthetic datasets is implemented. Secondly, the effects of the default parameters with some noise levels on the streaming synthetic dataset are quantified. The final method consisted of an investigation into the behaviour of each algorithm and manually adjusted *DenStream* epsilon parameter (see Figure 1-4). The data sources were mainly from a synthetic data generator using the RBF data generator in MOA. This is the only known data generator for clustering in MOA framework (see Figure 1-2 and Figure 1-3); and real-world benchmarks and publicly available datasets in CSV (comma separator variable) file format; or ARFF (attribute relation

file format) files from the University of California Irvin (UCI) Machine Learning Repository (Dua & Graff, 2019); USP Data Stream Repository(Souza et al., 2020);OpenML platform (Vanschoren et al., 2014); Stream Clustering (Carnein, 2019); and Stream Data Mining Repository (X. Zhu, 2010).

im

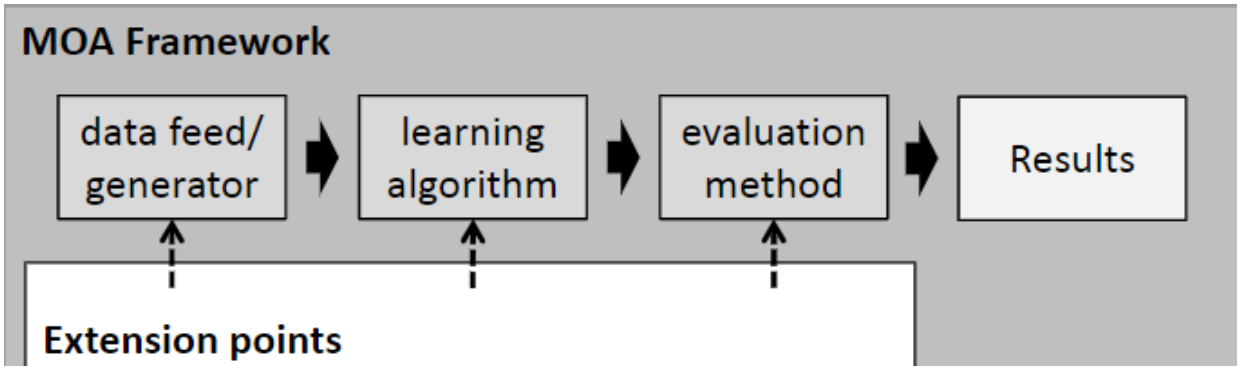


Figure 1-1: MOA framework for clustering adapted from (Kranen et al., 2012)

The Figure 1-1 shows the MOA workflow described as below:

- A data stream (feed, generator) from a file in ARFF format or CSV format using the class FileStream or SimpleCSVStream and configured by setting the parameters.
- An algorithm (that is a classifier) is selected and its parameters are set.
- The evaluation metrics are chosen.
- The results can be stored for visualisation after executing the task.

Figure 1-2 displays the settings for the RandomRBFGeneratorEvents stream with adjustable default parameters.

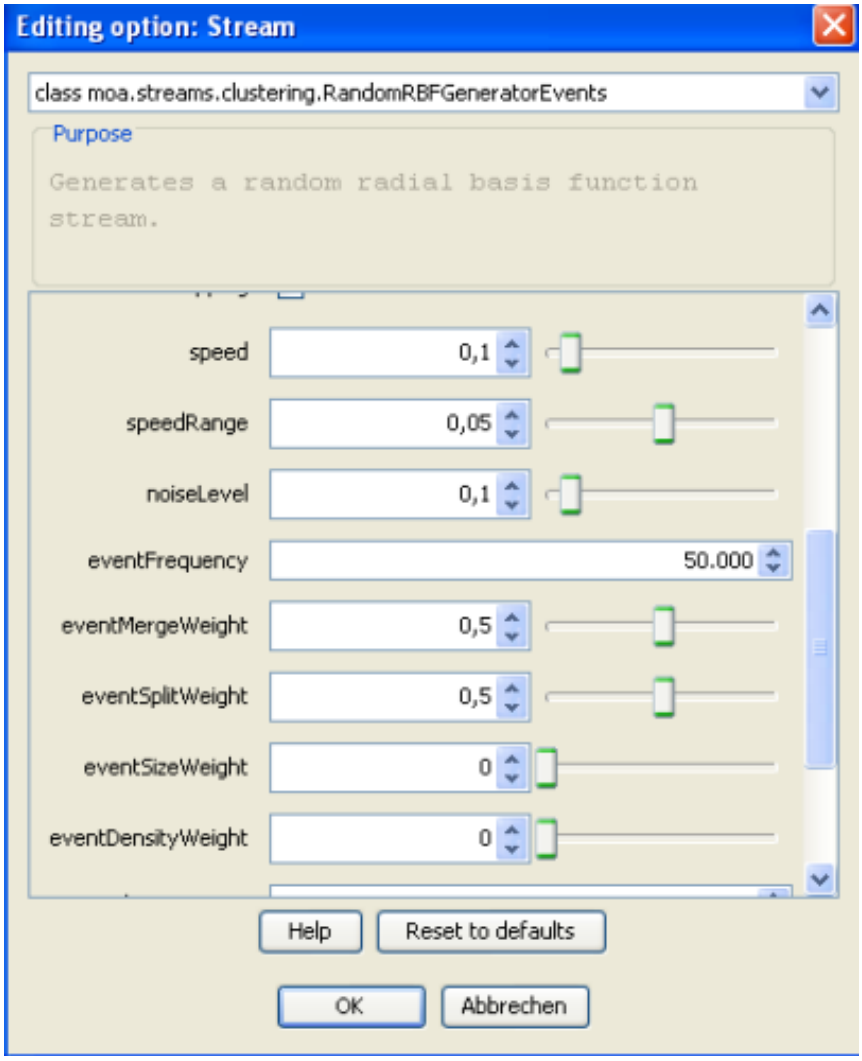


Figure 1-2: Option dialog for the RandomRBFGenerator stream data generator adapted from **(**(Kranen et al., 2010)

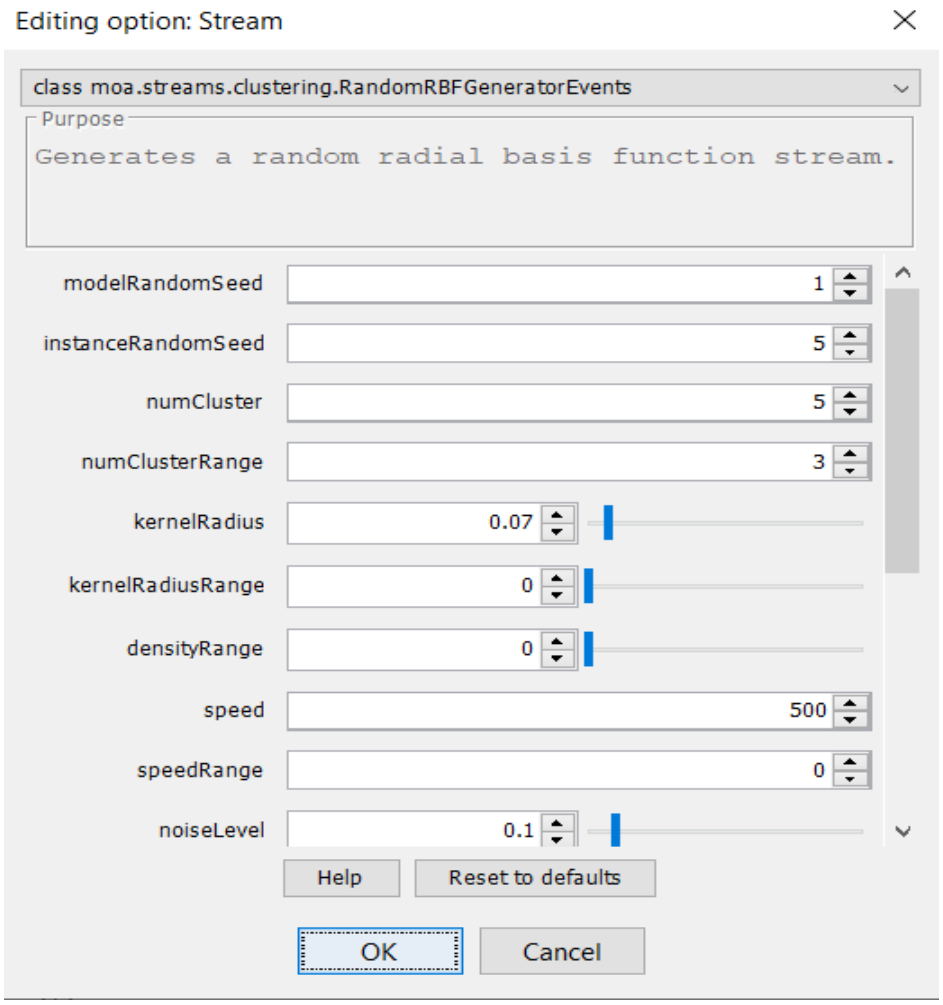


Figure 1-3: The new option dialog of RandomRBFGenerator data generator in MOA.

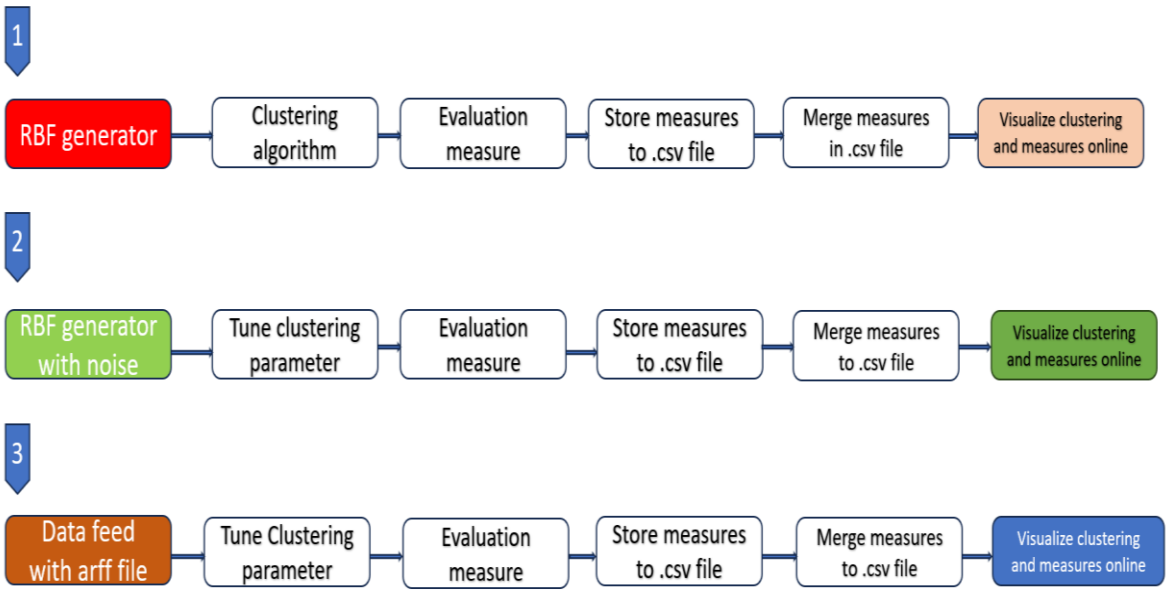


Figure 1-4: The methodology in this dissertation

1.5 Publications

The main goal of this research is to investigate and demonstrate the performance of improved *DenStream* algorithm against noise levels and sensitivity to parameter adjustment using streaming synthetic dataset and real-world datasets. In the following, we report the list of papers ready for publication.

- Akinosho, T. A., Tabane, E., & Wang, Z. (2023). A Comparative Analysis of Data Stream Clustering Algorithms. International Journal of Computing, 22(4), 439-446. https://doi.org/10.47839/ijc.22.4.3350
- Akinosho, T.A., Tabane, E. and Zenghui, W., 2023, October. Performance Evaluation of Data Stream Clustering Algorithm on Parameter Specification. In International Conference on Wireless Intelligent and Distributed Environment for Communication (pp. 173-189). Cham: Springer Nature Switzerland.

1.6 Organization of this Dissertation

The rest of this dissertation is structured in this order:

Chapter 2 presents recent research on density-based clustering algorithms, discusses other algorithms of interest, and provides a summary of related clustering techniques on performance metrics.

Chapter 3 describes the methodology of data stream clustering techniques. The MOA open-source software framework is presented.

Chapter 4 presents the experimental results, effects of the parameter tuning, demonstrated data visualization with the performance metrics.

Chapter 5 discusses the results of the experimental approach and summarizes the main contributions and future research directions.

CHAPTER 2: Literature Review

In this chapter, recent research on data stream clustering algorithms will be reviewed. Arbitrary shape cluster detection and outlier detection, offline-online phases, and parameter settings in several density-based techniques will be presented. Various clustering techniques associated with data stream clustering will be previewed and likewise several clustering techniques similarity distance measures will be presented. The performance evaluation metrics constantly used in data stream clustering will be described.

2.1 Data Stream Clustering

Clustering is the process of grouping sets of elements having common characteristics into homogeneous classes. Clustering is an unsupervised machine learning problem (Agrawal & Adane, 2016) useful for processing unlabeled data and appropriate in recognizing structures when information about data is available. In data stream clustering, groups of similar and dissimilar objects are clustered together separately (Ackermann et al., 2012; Carnein et al., 2020c). The clustering process in stream clustering algorithms is mostly divided into *online* and *offline* phases (Fahy et al., 2019; Ghesmoune et al., 2016b; Haneen et al., 2018; Khalilian et al., 2016; Xu et al., 2019). The online phase is dedicated to summarizing statistics of the data converted into micro-clusters used in the *offline* phase, while the summaries are re-clustered in the *offline* phase also called the clustering phase (Zubaroğlu & Atalay, 2019)to macro-cluster (Ahmed et al., 2020; Haneen et al., 2018; Roa et al., 2019). Most of the existing density-based algorithms are either *offline* or a hybrid of *offline* and *online* (Islam et al., 2019b). Some authors like Islam et al., (2019a), argued that *offline* algorithms are suitable for data stream clustering due to the inability to store data stream clustering and clusters in arbitrary shapes. Aljibawi et al. (2022) proposed an *offline-online* algorithm eMuDiS which is an enhanced version of MuDi-Stream algorithm by Amini et al. (2016). Figure 2-1 describes the *online-offline* clustering paradigm. Table 2-1 presents some of the stream clustering algorithms with *offline-online* phases.

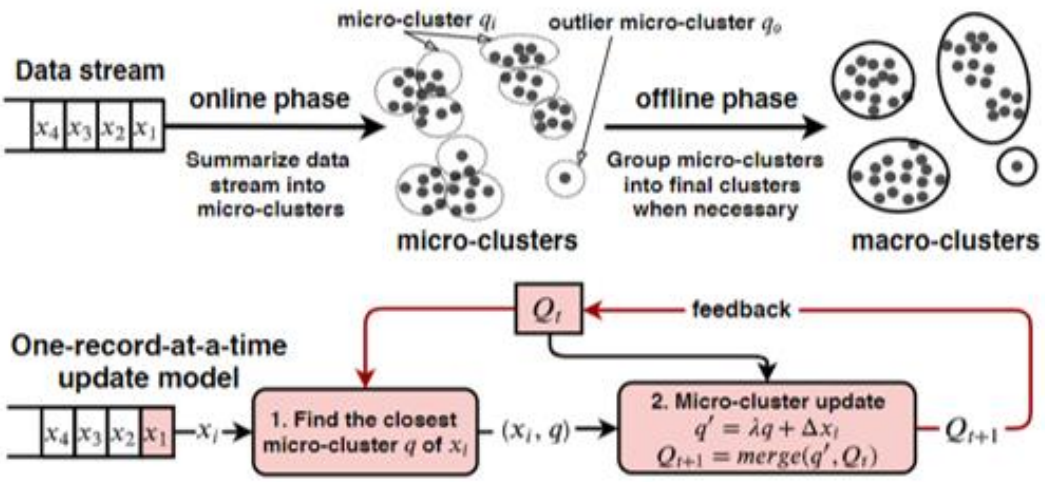


Figure 2-1: Online – offline clustering paradigm (Source: Zubaroğlu and Atalay, 2019).

Table 2-1: Showing algorithms with Online or hybrid clustering paradigm**.**

Article	**Algorithm**	**Online / Offline**	**clustering**
Cao et al., (2006)	DenStream	offline-online	DBSCAN
Zubaroğlu and Atalay, (2019)	UMAP		*k*-Means
Hyde and Angelov, (2015)	CODAS	Online	Grid
Hyde et al., (2017)	CEDAS	Online	Density
Aggarwal et al., (2003)	CluStream	offline-online	*k*-Means
Islam et al., (2019a)	i-CODAS	Online	Density
Chen and Tu, (2007)	D-Stream	offline-online	Density
Forestiero et al., (2013)	FlockStream	Online	Density
Fahy and Yang, (2019a)	MDSC	Online	Density
Amini et al., (2016)	MuDi-Stream	offline-online	Density & Grid
Li et al., (2022)	ESA-Stream	offline-online	Density & Grid
Tareq et al., (2020a)	CEC	Online	Density & Grid
Bezdek and Keller, (2021)	Fuzzy C-Means	Offline	Fuzzy
Carvalho et at., (2016)	SOM	Offline	Grid
Aljibawi et al., (2022)	eMuDiS	Offline-online	Density
Xu et al., (2019)	DenSOINN	offline-online	Model

2.2 Arbitrary Shape Clusters

Density-based algorithms are resourceful in the detection of arbitrary-shaped clusters and outliers' detection. Hyde and Angelov (2015) proposed an *Online* density-based algorithm known as Clustering Online Data-streams into Arbitrary Shapes (CODAS). The algorithm uses a simple local density for micro-cluster initialization which is later merged into clusters. Rather than using a fixed radius, CODAS employ a global micro-cluster radius that is constant (Islam et al., 2019b). The micro-cluster in CODAS used for storage has a 'core' and 'non-core' region (R. Hyde & Angelov, 2015). CODAS uses the Euclidean distance for distance measurement calculation. CODAS was compared for purity and accuracy with *DenStream*, Chameleon (Karypis et al., 1998); DBSCAN (Estert et al., 1996); ELM (O'Callaghan et al., 2002); and DEC (Oussous et al., 2018) and it achieved comparable results. Although CODAS was developed for online data stream clustering, clusters are not allowed to evolve and update discarded micro-clusters (R. Hyde & Angelov, 2015; R. W. Hyde et al., 2017; Saddam et al., 2020). Islam et al. (2019a) argued that it is erroneous to set the optimal value of micro-cluster radius, therefore they proposed an improved Clustering Online Data-streams into Arbitrary Shapes (i-CODAS) to maintain local micro-cluster radius. According to Islam et al., (2019a), i-CODAS is less dependent on users to set the optimal value parameter. The formation and separation of clusters in i-CODAS are confirmed by the minimum or maximum radius values. Both CODAS and i-CODAS can detect arbitrary-shaped clusters and noise.

Hyde et al. (2017) proposed "Clustering Evolving Data streams into Arbitrary Shaped" (CEDAS). CEDAS is a fully online two-stage technique that: (i) produces micro-clusters; (ii) merges the micro-clusters into macro-clusters. The technique uses the Euclidean distance measure in a fully online method. The authors compared CEDAS against *CluStream*, *DenStream*, and MR-Stream when evaluated using processing speed, detection of intrusion, dimensional effects, adaptation to evolving data, purity, and Big Data, using both the real-world London Air Quality and the KDDCup99 datasets. CEDAS can detect arbitrary-shaped clusters and noise, but its drawback is a lower processing time. CEDAS witnessed some improvements such as the buffer-based online clustering for evolving data stream (BOCEDS) proposed by Islam et al., (2019b). BOCEDS is an entirely online density-based algorithm that reduces dependency on users by recursively updating the micro-cluster radius to its local optimal level. BOCEDS momentarily separates irrelevant clusters from fully irrelevant clusters by using a buffer to store the irrelevant micro-clusters. BOCEDS performed well

against *CluStream, DenStream* CEDAS, and CODAS based on purity, accuracy, noise sensitivity, speed, memory efficiency, and scalability.

Carnein et al. (2017) carried out an extensive comparison of ten different data stream clustering algorithms using a standardized testing environment. According to the authors, this is a novel comparable study of these algorithms. The comparative study was carried out using numerous synthetic and real-world datasets. The authors proved that: (i) grid-based algorithms require sufficient micro-clusters and (ii) arbitrary-shaped clusters are difficult to identify. To reduce computational, memory, and still find arbitrary-shaped clusters, Attaoui et al., (2022) proposed IMOC-Stream. IMOC-Stream uses the Ant-tree algorithm to determine a cluster's neighborhood and is free from user-defined numbers of clusters. The experimental study was carried out using high-dimensional datasets and the performance evaluation measures indicate that IMOC-Stream outperforms other algorithms on NMI and ARI.

Tareq et al. (2020a) also proposed an online clustering algorithm known as the clustering evolving data streams based on the adaptive Chebychev distance (CEC). In CEC, the summary of evolving data streams is stored as a core micro-cluster (CMCs). CEC is used for calculating the distance between an inbound data point and the CMC center. CEC was evaluated against CEDAS based on cluster purity, accuracy, and percentage of data points assigned to clusters. CEC can handle high-dimensional datasets. Recently, Tareq et al. (2020b) proposed the "Clustering of Evolving Data via a density Grid-based Method" (CEDGM). CEDGM is a novel technique that uses grid granularity for the data reduction process.

Mansalis et al. (2018) presented an analysis of benchmark stream clustering algorithms. The applications of *CluStream, DenStream,* and *ClusTree* were appraised. The authors evaluated the performance based on metrics such as the *Clustering Mapping Measure* (CMM), *Sum of Squared Distance* (SSQ), and *Purity* for different parameter settings. The authors, however, stated that SSQ is not appropriate for arbitrary shaped clusters. They vary two user-specified parameters of *DenStream* (outlier threshold β and the decay factor λ) and reported that *DenStream* outperformed *CluStream* and *ClusTree* on clustering quality based on window size. The results also show that both *ClusTree* and *CluStream* outshined *DenStream* performance metric CMM. However, only the real-world datasets *Adult-Census, Electricity, Covertype*, and *Poker-Hand* were used. Roa et al. (2019) proposed a two-stage strategy clustering algorithm: *slower scale density-based algorithm* and *fast scale distance-based algorithm* to speed up enormous data arriving at a fast rate. The authors evaluated their

algorithm against *CluStream* and *DenStream* using performance metrics *multi-density test, robust path-based test,* and *concept drift experiment*. Their algorithm outperformed both *DenStream* and *CluStream*.

Amini et al. (2016) investigated the challenges in clustering algorithms like detecting clusters in multi-density data. They argue that several of the implemented multi-density clustering algorithms are inappropriate for data stream clustering and proposed a *MuDi-Stream* to address the drawback. The authors point out that the proposed method is an improvement over the *DenStream* algorithm. Fahy and Yang (2019a), however, proposed a *Multi-density Stream Clustering* (MDSC) algorithm for the gap *multi-density* and tracing deviations in a *dynamic stream*. The method discovers and tracks *multi-density* clusters continuously. The MDSC is a *cluster-feature* (CF) in the form (*N, LS, SS, t*). The *N* is the number data of points in the cluster $\{ \overline{Xi} \}\, i = \{1, \cdots, N\}$; *LS* is the linear sum of points (i.e, $\sum_{i=1}^{N} \overline{Xi}$); *SS* is the square sum of points (i.e., $\sum_{i=1}^{N} \overline{Xi}^2$); and *t* is the time stamp. The performance of MDSC was evaluated using four real and three synthetic datasets on three external evaluation metrics (*Purity, F-measure,* and *Rand index*) and compared against four known *density clustering algorithms.* The resultant value shows that MDSC outperforms well against the peer algorithms. Fahy and Yang (2019a) argue that MDSC can track changes in seasonal and cyclic behavior and is robust to noise. However, other potentials need to be explored. Aljibawi et al., (2022) recently proposed an enhanced version of *MuDi-Stream,* code named *eMuDiS.* The algorithm addresses the issue of streaming speed and stream dimension. Aljibawi et al. (2022) demonstrated that *eMuDiS* outperforms *MuDi-Stream* on both real and synthetic datasets.

Fahy and Yang (2019b) noted three types of changes in the data stream which are *concept evolution*, *concept drift*, and *feature level.* The authors examine two ways in which changes occur at the feature level (*feature drift* and *feature evolution*). Fahy and Yang (2019b) highlighted two problems of high-dimensional data: (i) distance measurement; and (ii) the concept of density. To mitigate the problem, Fahy and Yang (2019b) proposed a dynamic feature mask (DFM) clustering technique. This method addresses the two documented challenges of data streams (feature *drift* and *clustering high-dimensional streams*). The DFM technique can detect and track *feature drift* and *feature evolution.* Fahy and Yang (2019b) evaluate the DFM against the density-based algorithms (*CEDAS, MDSC, ACSC, and DenStream*). The results showed that DFM improves performance and reduces execution time

when used alongside any density-based stream clustering algorithms and increases accuracy and lower execution time.

Abid et al. (2019) highlighted some challenges of data stream such as *concept drift, infinite length, feature evolution, novelty detection,* and ways of addressing them. They posited that the developing nature is the most critical aspect of the data stream process. The authors proposed a novel data stream clustering technique, AIS-Clus. This technique uses the Artificial Immune System (AIS) meta-heuristic, which is described as a "bio-inspired algorithm" (Abid et al., 2019). The AIS-Clus is then compared against *CluStream* and *DenStream* on MOA. Yeoh et al. (2019) argue that near-perfect data stream clustering algorithms should address "*concept drift*" and "*concept evolution*". They proposed a novel *OpStream*, an optimized stream clustering algorithm that fused meta-heuristic optimization with data stream clustering. The authors categorized the novel algorithm into the *initialization* and *online phases.* They investigated the three variations of the novel algorithm which are *Whale Optimisation Algorithm* (WOA-OpStrem), BAT-OpStream, and *Differential Evolution* (DE-OpStream) against *CluStream* and *DenStream* on four synthetic datasets and a real-world dataset. The results indicated that the three variant algorithms performed better on synthetic datasets than *DenStream* and *CluStream.* However, *DenStream* showed a more robust performance than the three algorithms on the *KDDC-99* dataset.

Carnein and Trautmann, (2018) proposed *evoStream* a novel stream clustering algorithm that utilizes a heuristic optimization algorithm to improve the macro-clusters solution using idle time and computational resources. The authors applied the *DBSTREAM* concepts to build the algorithm for its speed and flexibility. They utilized four real-world datasets *Powersupply, Sensor, KDDCup99,* and *Covertype* to evaluate the algorithm against many state-of-the-art algorithms. The technique displayed a robust performance against the benchmark algorithms. However, the authors only utilize the online phase of *DBSTREAM* for the stream and the *Sum of Squares* (SSQ) performance evaluation measure. (Carnein and Trautmann 2019a) implemented a new stream clustering algorithm *userStream* in customer segmentation to identify and track customer segments. The algorithm employs time-faded *Clustering Feature* (CF) theory and a two-phase clustering approach: *online and offline phase.* The authors appraised the performance of the algorithm using real-world datasets from home furniture and textile sectors and *Silhouette* performance measures. The resultant output showed that the algorithm is valuable in tracking and identifying customer segments. However, the authors did not test the algorithm against any of the benchmark algorithms.

Recently, Carnein et al. (2020b) proposed *confStream,* an innovative ensemble-based approach, to implement an automated algorithm configuration for *DenStream*. Carnein et al., (2020b) used the *Silhouette with* evaluation measures to appraise the cluster quality of *confStream* against *DenStream.* They use both synthetic dataset *Random Radial Basic Function* (Random RBF) and real-world datasets (*Covertype*, and *Sensor*). The *confStream* has a robust performance over *DenStream* in improving configuration. This ensemble approach is, however, more time consuming than individual algorithms (Carnein et al., 2020a). Ahmed et al. (2020) proposed an online-offline density-based algorithm, *DGStream* with a discrete-time step model. This algorithm uses the DBSCAN algorithm at the online and offline phases and feature vector. Ahmed et al. (2020) argues that *DGStream* is suitable for recent information like stock markets. The *DGStream* algorithm was evaluated with different parameter settings against *DStream, ClusTree,* and *DenStream* on both streaming synthetic and real-world datasets. Moreover, the *DGStream* outperforms these algorithms using Chameleon synthetic dataset performance metrics: F1-score, recall, purity, precision, and time. On real-world datasets KDDCup'99, Forest Covertype, Adult-Census, and the National Stocks Exchange of India (NSE Stocks, 2017). Ahmed et al. (2020) state that *DGStream* is appropriate for handling outliers and noise with the minimum time complexity. However, the research only uses numerical variables datasets.

Lee et al. (2019), investigated the challenges of density-based clustering and developed a hybrid data streams clustering algorithm that fuses density-based and model-based algorithms. The algorithm tested on both real-world and synthetic datasets performed excellently in detecting data streams with noise. The authors proved the algorithm could detect clusters faster and find optimal parameters proficiently. However, the paper compared the algorithm to only *DenStream* algorithm.

Gajowniczek et al., (2020) proposed an algorithm for clustering multiple data streams in time series. The algorithm is evaluated on a smart metering sensor dataset from the Irish Commission for Energy Regulation (ICER). The authors only utilized 1000 households' electricity consumption from a total of 4182 households due to missing values. They observed seasonal cycles for annual, weekly, and daily electricity consumption. The results of the study indicated that the algorithm is appropriate for clustering the flow of data and suitable for segmenting electricity consumers, based on their usage and socio-economic behaviours. Although the study focused on electricity consumption, the authors, however,

make a case for its applications in areas such as the stock market, and banking sectors, among others.

Xu et al. (2019) proposed a novel data streams clustering algorithm, the *Density Based Self Organizing Increment Neural Network (DenSOINN).* The algorithm utilizes both a self-adaptive distance metric and a novel density-based method to solve problems of data normalization and finding clusters in a neural network, respectively. The evaluation shows that *DenSOINN* has a strong performance on both synthetic and real-world datasets over other algorithms.

Gong et al. (2018) proposed a novel *Evolution of Density Mountain* stream clustering algorithm (*EDMStream*). The *EDMStream* has the following abilities: (i) return updated clustering results faster; (ii) adjust and adapt itself to changes in data distribution; and (iii) dynamically adjust to the user's preference. The authors compare the algorithm with well-known benchmark algorithms: *DenStream*, *D-Stream*, *DBSTREAM*, and *MR-Stream*, and reported that *EDMStream* has a robust performance and exhibits 7-15 high speed over the other algorithms.

Yan et al., (2019), proposed a two-phased dynamic stream clustering algorithm Dynamic Fitness Proportionate Sharing Clustering (*DFPS-clustering*) algorithm. The authors compare the *DFPS-clustering* algorithm against other known two-phase algorithms (*CluStream*, *STREAM*, *DBStream*, *D-Stream*, and *HDDStream*). The *DFPS-clustering* was evaluated using three synthetic and four real-world datasets. The resultant output suggested that *DFPS-clustering* has a robust performance against other algorithms with a limitation of high computational cost.

Wang and Wang, (2018) proposed the *DCluStream* algorithm to address the challenges of judging outliers and eliminating outdated data in time. The algorithm improves the *CluStream* algorithm by: (i) adding at the online micro-clustering phase the decay time window mechanism; and (ii) A buffer processing mechanism for memory storage. The study focuses solely on improving online micro-clustering. This is divided into two parts: (i) handling new data in real-time; and (ii) adjusting global micro clusters. The algorithm was compared with *CluStream* using the *KDDCup99* dataset. The authors reported that *DCluStream* exhibited an improved clustering quality and reduces the processing time.

Li et al. (2022), proposed an Efficient Self Adaptive Stream (ESA-Stream) a fully online data stream algorithm for learning parameter settings dynamically. The algorithm can detect

arbitrary-shaped clusters and speedup dimensionality reduction using the density grid-base clustering technique. The authors evaluate the performance of ESA-Stream using both synthetic and real-world datasets. The ESA-Stream outperforms state-of-the-art baselines in both efficiency and effectiveness.

In the field of medicine, Al-Shammari et al. (2019) proposed a density-based clustering algorithm that combined *Piece-wise Aggregate Approximation* and *density-based with noise* (*PAA+DBSCAN*). The algorithm is suitable for the initial clustering of patients with similar symptoms and *Advance Cluster Maintenance* (ACM) which is an incremental maintenance approach in medical clusters. This approach is important in identifying and helping patients with risks and underlining health challenges such as high blood pressure. The authors argue that the new algorithm can group new patients into clusters of similar symptoms and track those whose status is unstable while keeping close contact with those who are stable.

2.3 Time Window Techniques

There are several time-window techniques for data streams. The *time-window* of data objects is given as $\boldsymbol{W[i,j] = (x_i, x_{i+1}, \ldots, x_j)}$, where i < j. The most popular time-window techniques are *damped/fading window*, *landmark window*, *sliding window*, and *tilted window model* (Carnein and Trautmann, 2019b; De Andrade Silva and Hruschka, 2016; Ghesmoune et al., 2016b; Gomes et al., 2017; Laha and Putatunda, 2018; Yarlagadda et al., 2018; Yeoh et al., 2019; Youn et al., 2018) see Figure 2.2.

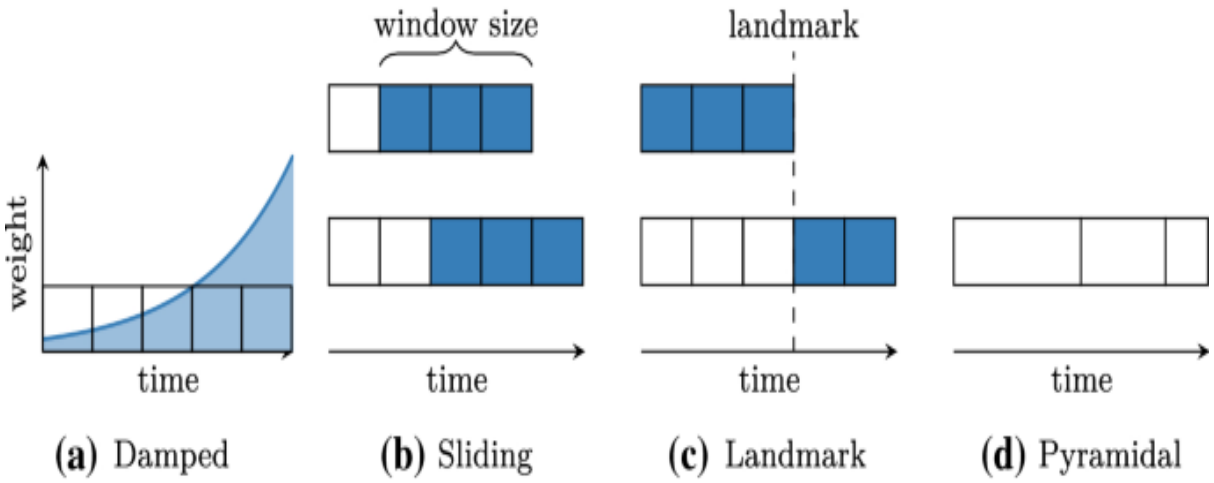


Figure 2-2: Time window models for data stream clustering techniques. Source: Carnein and Trautmann (2019b).

Agrawal and Adane (2016) summarized the efforts of researchers in data stream mining. They presented a study of several data stream algorithms. The three data stream models presented were the *landmark*, *damped*, and *sliding windows*. The authors identified terminologies and approaches in data stream mining and outlined future research issues which would assist in further research in the field.

1. ***Landmark window***

   In the *landmark window* model, discarding older data points is not required. The clustering is applied from the initial starting point or timestamps t_1 to the current timestamps t_c, W $[t_1, t_c]$. Examples of landmark window models include *CluStream* (C. C. Aggarwal et al., 2003); *BIRCH* (T. Zhang et al., 1996).

2. ***Sliding windows***

   There are two basic types of sliding windows, count-based and time-based windows (Kontaki et al., 2016). In a *sliding window*, old data expire as new data arrives for analysis using the principle of first-In-First-Out (FIFO) (J. Shao et al., 2019). The sliding window model has been proposed in much research (Kontaki et al., 2016; G. Li et al., 2018; Lin & Su, 2019; Youn et al., 2018).

3. ***Damped window.***

   In the *damped/fading window* model, a data object is assigned varying weights based on the arrival time where new entries received higher weights than older ones. The data point weight decreases exponentially with time t through a fading function $f(t) = 2^{-\lambda t}$ with $\lambda > 0$. An example of a damped window model is the *DenStream* (Cao et al., 2006).

4. ***Tilted time window***.

   In the *tilted time window* model, different granularity levels are used based on recent data points. The most current data is the finest granularity which becomes coarse as data points get old. Examples of tilted-time window models are CluStream (C. C. Aggarwal et al., 2003); HPStream (C. Aggarwal et al., 2004); and StreamKM++ (Ackermann et al., 2012).

2.4 Clustering Techniques

Several stream clustering algorithms in the literature have been implemented in MOA. Those currently implemented are *CluStream* (C. C. Aggarwal et al., 2003), *DenStream* (Cao et al., 2006); *ClusTree* (Kranen et al., 2011); *D-Stream* (Y. Chen & Tu, 2007; Tu & Chen, 2008);

StreamKM++(Ackermann et al., 2012); *CobWeb* (Fisher, 1996); *confStream* (Carnein et al., 2020b), among others. Several of these algorithms are classified into *Partitioning*, *Density-based*, *Model-based*, *Grid-based*, *Hierarchical-based*, and *Graph-based* (Kokate et al., 2018; Mansalis et al., 2018); details of which are provided in the next sections.

2.4.1 Partitioning Clustering

The partitioning clustering is a sphere-shaped cluster. It is partitioned into both soft (fuzzy/probabilistic) clustering and hard (crisp) clustering (Bezdek & Keller, 2021; Chenaghlou, 2019; Moshtaghi et al., 2019; Rathore, 2018). In crisp clustering, the data point belongs to a cluster or not whereas, in *fuzzy clustering*, the data point could be assigned to one or more clusters, (Kuwil et al., 2020). The hard (crisp) clustering is susceptible to local minimum than fuzzy clustering (Aggarwal & Reddy, 2014). There are several partitioning-based techniques in the literature such as k-means (Ordonez, 2003); k-medoids or Partitioning Around Medoids (PAM) (Kaufman & Rousseeuw, 1990); k-medians, k-mode, k-center, Clustering LARge Applications (CLARA) (Kaufman & Rousseeuw, 1990); *CluStream* (Aggarwal et al., 2003); StreamKM++ (Ackermann et al., 2012); Clustering Large Applications Based Upon Randomized Search (CLARANS) Ng and Han (2002); (C. C. Aggarwal & Reddy, 2014; Andreopoulos et al., 2009; Mittal et al., 2019). The most established are the *k-medoids* and *k-means.* The *k-means* is not suitable for sphere-shaped clusters although it can be used when the number of clusters is known.

2.4.2 Hierarchical-based Clustering

Hierarchical-based clustering is partitioned into: (i) *divisive clustering*; (ii) *agglomerative algorithms* (Al-shammari, 2019; Hassani, 2015; Lee et al., 2019; Loureiro et al., 2005; Rathore, 2018). The *disivise* is a top-down/hierarchical approach while the *agglomerative clustering* utilizes the bottom-up/sequential approach. In *agglomerative clustering*, each object starts as a single cluster and is merged into large clusters using a similarity measure until the final cluster condition is met. *Divisive clustering,* on the other hand, works the reverse way, it started with all objects in one large cluster and repeatedly splits into smaller clusters based on the dissimilarity measure (Hassani, 2015; Jiri Skala, 2012). Examples of *divisive* and *agglomerative* approaches are *Online Divisive Agglomerative Clustering* (ODAC) and *Hierarchical Agglomerative Clustering* (HAC) (Rodrigues et al., 2006). In hierarchical clustering, the distance between two clusters is determined using linkage such as *Single*, *Complete*, or *Average* linkage (Andreopoulos et al., 2009). The hierarchical-based

clustering in the literature includes Chameleon (Karypis et al., 1998); *BIRCH* (Zhang et al., 1996); *CURE* (Guha et al., 1998); and *ROCK* (Guha et al., 2000).

2.4.3 Grid-based Clustering

This method uses equal grid cells partitioning to accelerate the clustering process. Compared to other clustering algorithms, grid-based clustering has an agile processing time and can competently handle datasets grapple with noise. Most grid-based algorithms can detect arbitrary-shaped clusters. Grid-based can be combined with other clustering like density-based to form a hybrid clustering approach. There are several types of grid-based clustering algorithm which include the fast and grid-based clustering for hybrid data stream (FGCH) (J. Chen et al., 2019); density-and-grid-based (DGB) clustering (B. Wu & Wilamowski, 2017); Clustering of Evolving Data streams via a density Grid-based Method (CEDGM) (Tareq, Sundararajan, Mohd, et al., 2020); Grid-K-means (E. Zhu et al., 2019); and STING (STatistical INformation Grid approach) (W. Wang et al., 1997).

2.4.4 Density-based Clustering

Density-based algorithms are effectual in the arbitrary-shaped clusters, noise, and outliers detection. Density-based clustering is non-parametric method due to the non-assumptions about the number of clusters (Aggarwal & Reddy, 2014). *DenStream* (Cao et al., 2006); DBSCAN (Estert et al., 1996); *MuDi-Stream* (Amini et al., (2016), DENCLUE (DENsity-based CLUstEring) (Hinneburg & Keim, 2003); CODAS (Hyde and Angelov, 2015), CEDAS (Hyde et al., 2017); and BOCEDS (Islam et al., 2019b) are examples of density-based algorithms for clustering evolving data streams.

2.4.5 Model-based Clustering

The model-based clustering method is based on a statistical model and permits objects to be in multiple groups. The model-driven clustering relies on a specific model for each cluster to identify the most suitable one. There are several model-based clustering techniques documented: *CobWeb* (Fisher, 1996); Expectation Maximization (EM) Moon (1996); CluDistream (Zhou et al., 2006); Self-organizing feature maps (SOMs) (Carvalho et at., 2016); SWEM (Sliding Window with Expectation Maximization) (Dang et al., 2009); and ICFR (Incremental Clustering using F-value Regression analysis). For further information about model-based clustering see Carnein and Trautmann (2019b); H. Shao et al. (2019); Sharma et al. (2018); and Singh (2015).

2.4.6 Fuzzy Clustering

In fuzzy *C*-Means clustering (FCM), objects are connected in the cluster range [0, 1] (Bezdek & Keller, 2021). The FCM algorithm is sensitive to outliers and each data object could be grouped into more than one cluster (Rasyid & Andayani, 2018). Many examples of FCM include FUZZ-CARE by Song et al. (2020); FuzzyStream (de Abreu Lopes & de Arruda Camargo, 2017); and d-FuzzyStream (Schick et al., 2018) among others.

The summary of some of the data stream clustering algorithms is presented in Table 2-2 below. The table shows the various techniques, the algorithms that have used the techniques, the Time window model assigned to each algorithm, the cluster shape of each algorithm, their data type, and how they can handle noise.

Table 2-2: Summary of the Data Stream Clustering Algorithms

Technique	**Algorithm**	**Time Window Model**	**Cluster shape**	**Data type**	**Handle noise**
Ordonez, 2003	*K*-means		Arbitrary	Numerical	No
Ackermann et al., 2012.	StreamKM++	Pyramidal	Spherical	Numerical	
Kaufman and Rousseeuw, 1990	PAM		Arbitrary	Numerical	No
Kaufman and Rousseeuw, 1990	CLARA		Arbitrary	Numerical	No
Ng and Han, 2002	CLARANS		Arbitrary	Numerical	No
Aggarwal et al., 2003	CluStream	Pyramidal	Arbitrary	Numerical	No
Zhang et al., 1996	BIRCH	Landmark	Arbitrary	Numerical	No
Guha et al., 1998	CURE		Arbitrary	Numerical	Yes
Guha et al., 2000	ROCK		Tree	Categorical	No
Karypis et al., 1998	Chameleon		Arbitrary	Numerical & Categorical	No
Rodrigues et al., 2006	ODAC		Hyper-ellipsis	Categorical	
Tareq et al., 2020b	CEDGM				
Agrawal et al., 1998	CLIQUE			Numerical	Yes

Wang et al., 1997	STING		Arbitrary	Spatial	Yes
Cao et al., 2006	DenStream	Damped	Arbitrary	Numerical	Yes
Estert et al., 1996	DBSCAN		Arbitrary	Spatial	No
Chen and Tu, 2007	D-Stream	Damped	Arbitrary		Yes
Kranen et al., 2011	ClusTree	Damped	Arbitrary		Yes
Hyde and Angelov (2015)	CODAS		Arbitrary		Yes
Hyde et al. 2017	CEDAS	Damped	Arbitrary		Yes
Islam et al., 2019b	BOCEDS	Damped	Arbitrary	Spatial	Yes
Bezdek and Keller, 2021	FCM			Numerical	Yes
de Abreu Lopes and de Arruda Camargo, 2017	FuzzyStream				Yes
Schick et al., 2018	d-FuzzyStream				Yes
Song et al., 2020	FUZZY-CARE				
Moon, 1996	EM			Spatial	
Fisher, 1996	CobWeb		Tree	Numerical	
Carvalho et at., 2016	SOM			Numerical	No
Zhou et al., 2006	CluDistream				

Figure 2-3 presents the flow chart of the data stream clustering algorithms categorization.

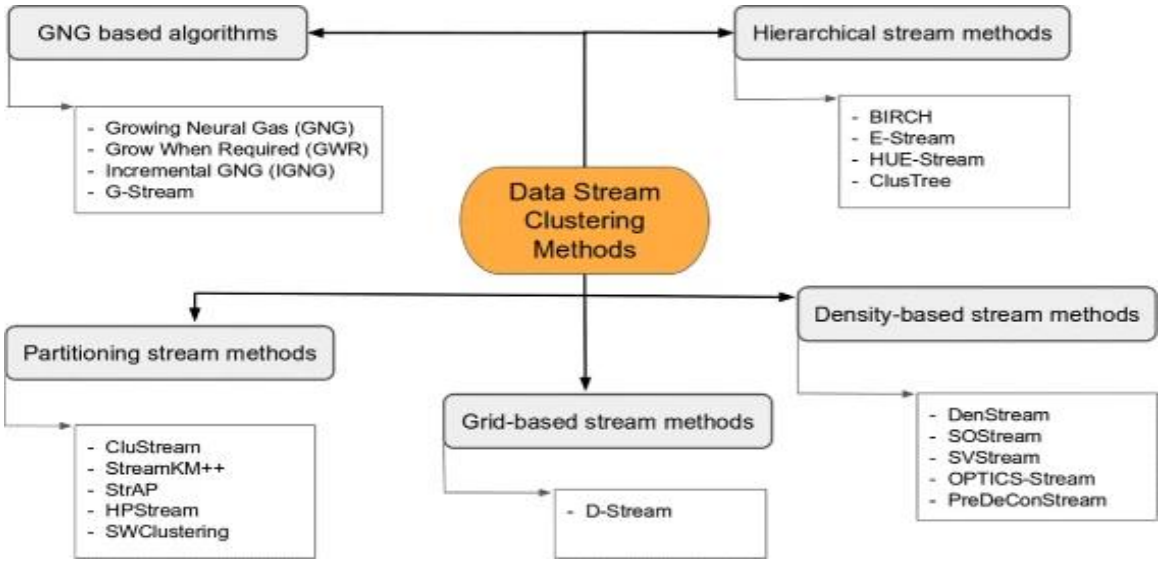


Figure 2-3: Data stream clustering methods (adapted from: Ghesmoune et al., 2016a**).**

2.5 Similarity and Distances

There are many distance measures around that can be found in studies such as those of Zhang et al. (2023). In this section, the discussion will focus on distance measures for modeling the similarity of data in the literature. The distance measure is described as a *metric* in Norm vector space (that is, a vector space with a norm defined) if the conditions described in Franke and Geyer-Schulz, (2007; and Rastin (2018) are satisfied:

- *Non-negativity*: $d(x, \quad y) \geq 0$ for all x and y
- *Identity*: $d(x, y) = 0$ if and only if x = y
- *Symmetry*: $d(x, y) = d(y, x)$ for all x and y
- *Triangular inequality*: $d(x, y) \leq d(x, z) + d(z, y)$ for all x, y and z

Tareq et al. (2020a) state some of the important distance measurement features: (i) the distance from one point to itself is always zero; (ii) distance is always positive; (iii) distance in x - y is the same as the distance in y - x; and (iv) the distance from x - y is equal to the sum of the distance from x - z, and z - y.

The most generalized distance metric is the *Minkowski distance* defined as*:*

$$d(x, y) = \sqrt[k]{\sum_{i=1}^{n} |xi - yi|^k} \tag{2.1}$$

$$= \left(\sum_{i=1}^{n} |xi - yi|^k \right)^{1/k} \tag{2.2}$$

where:

n = number of dimensions

$k \neq 0$ is order parameters or any real number

x_i, y_i = data points

The *Minkowski distance* is the generalized L_p*-norm* represented as $\|x - y\|_p$

The value of k in the above formula can be manipulated to derive other distance measures as:

- For k = 1, the *Minkowski distance* gives the *Manhattan distance* between *x* and *y* as:

$$d(x, y) = \sum_{i=1}^{n} |xi - yi| \tag{2.3}$$

where:

n = number of dimensions

x_i, y_i = data points

The *Manhattan distance* is the L_1*-norm* represented as $\|x - y\|_1$

- For k = 2, the *Minkowski distance* gives the *Euclidean distance* between *x* and *y* as:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2} \tag{2.4}$$

where:

n = number of dimensions

xi, *yi* = data points

The *Euclidean distance* is the most widely used of the distance measures. The *Euclidean distance* is the L_2*-norm* represented as $\|x - y\|_2$. The square root of *Euclidean distance* when removed gives another metric known as the *squared Euclidean distance* (*SED*) in equation (2.5).

$$d^2(x,y) = \sum_{i=1}^{n}(xi - yi)^2 \tag{2.5}$$

- For k = ∞, leads to *Chebyshev distance* also known as the *Chessboard distance.*

$$d(x,y) = \lim_{k\to\infty}\left(\sum_{i=1}^{n}|xi - yi|^k\right)^{\frac{1}{k}} \tag{2.6}$$

$$= \max_{i=1}|xi - yi| \tag{2.7}$$

where:

n = number of dimensions

k = order parameters

xi, *yi* = data points

The *Chebyshev distance* is the L_∞*-norm*, represented as $\|x - y\|_\infty$. Several studies, such as Tareq et al. (2020b, 2020a); and Tareq and Sundararajan (2021, 2020) have used the *Chebyshev distance*.

Other notable distance measures include:

- *Cosine similarity*: Cosine similarity is the measure of the angular distance between two vector points. The cosine similarity has usage in text mining to calculate the similarity between tweets (Ghaemi & Farnaghi, 2019); outlier detection may experience difficulties when there is uncertainty in the measurement of *similarity*/*dissimilarity* between two data points (S. Sadik & Gruenwald, 2014; S. M. Sadik, 2013). The cosine similarity formula is given using the dot product:

$$\cos(\theta) = \frac{\vec{a}.\vec{b}}{\|\vec{a}\|\|\vec{b}\|} \tag{2.8}$$

$$= \frac{\sum_{i=1}^{n} ab}{\sqrt{\sum_{i=1}^{n} a^2}\sqrt{\sum_{i=1}^{n} b^2}} \tag{2.9}$$

  The cosine distance is the difference between 1 and the cosine similarity, i.e., 1 - $\cos(\theta)$. As the cosine distance increases, the cosine similarity decreases, and vice versa. Ghaemi and Farnaghi (2019) used cosine similarity in their research.
- *Jaccard index*: The *Jaccard index* or *Jaccard coefficient* measures the similarity between two sets. Given two sets A and B, the *Jaccard coefficient* is the ratio of their intersection and union.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.10}$$

$$J(A,B) = \frac{\sum_{i=1}^{n} \min(A,B)}{\sum_{i=1}^{n} \max(A,B)} \tag{2.11}$$

  The similarity measure can be transformed to the distance metric as *Jaccard distance*, which only takes values between 0 and 1*:*

$$d(A,B) = 1 - J(A,B) \tag{2.12}$$

If the *Jaccard coefficient* is higher, the similarity will be higher and likewise, if lower, the similarity will be lower (Li et al., 2018). The Jaccard coefficient can also be used in terms of *True Positives* (TP), *False Positives* (FP), and *False Negatives* (FN). This formula is the ratio of TP and the summation of TP, FP, and FN (S. M. Sadik, 2013).

$$JC = \frac{TP}{TP + FP + FN} \tag{2.13}$$

Several studies reportedly used the *Jaccard index* (see Amini et al., 2016; Rodriguez et al., 2019; Sadik, 2013).

2.6 Clustering Performance Metrics

The clustering performance metric is divided into *intrinsic* and *extrinsic* methods (Ahmed et al., 2020). When the ground truths are available it is known as the *extrinsic/external* method which is a supervised method, otherwise it is an unsupervised method. The *extrinsic* method includes the *Clustering Mapping Measure* (CMM), *Recall*, *Precision, Rand index*, and *Purity* (Kremer et al., 2011); while an *intrinsic* method is the *Silhouette coefficient.* The formulas are:

- *Rand index***:** The Rand index (RI) is the fraction of the sum of TP and TN over the sum of TP, FP, TN, and FN. The RI measures the accuracy of two clustering using value range between 0 and 1.

$$RI = \frac{TP+TN}{TP+FP+TN+FN} \tag{2.14}$$

Where *TP* is true positive, *TN* is true negative, *FP* is false positive, *and FN* is false negative.

- *Adjusted Rand Index*: The adjusted Rand Index (ARI) is an updated form of the Rand index (RI). The ARI can be used for similarity measures between two data clustering.

$$ARI = \frac{Index - Expected\ Index}{Maximum\ Index - Expected\ Index} \tag{2.15}$$

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)} \tag{2.16}$$

- *Precision:* Precision is the ratio of true positive instances over the sum of TP and FP instances.

$$P = \frac{TP}{TP + FP} \tag{2.17}$$

- *Recall:* Recall or sensitivity is the ratio of TP instances over the sum TP and FN.

$$R = \frac{TP}{TP + FN} \tag{2.18}$$

- *Purity:* Purity is the rate of suitably classified instances given as:

$$Purity = \frac{1}{N}\sum_{k=1}^{c} \max\left(\omega \cap \varphi\right) \tag{2.19}$$

Where ω refers to the total number of clusters, while φ refers to the total number of classes.

- *Silhouette* (Rousseeuw, 1987): The *Silhouette coefficient* contrast the mean distance in cluster *a* against the least mean distance in cluster *b b*etween its object *i* to every other point in the same cluster. The *Silhouette coefficient* fuse together the Separation and Cohesion measures. Its values range between 0 and 1.

$$J(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.20}$$

where:

$a(i)$ = mean dissimilarity of *i* to all objects of *A* given in the equation below

$$a(i) = \frac{1}{|A| - 1}\sum_{j \in A, i \neq j} dist(i, j) \tag{2.21}$$

$$d(i, C) = \frac{1}{|C|}\sum_{j \in C} dist(i, j) \tag{2.22}$$

For all cluster $(C \neq A)$ obtained from $d(i, C)$ then

$b(i)$ = mean dissimilarity of *i* to all objects of *A* given in the equation below

$$b(i) = \min\big(d(i, C)\big) \tag{2.23}$$

The equation of Silhouette can be simplified to

$$SI(i) = 1 - \frac{a(i)}{b(i)} \tag{2.24}$$

Table 2-3 presents some of the available internal and external clustering validation measures in the literature and adapted from Kranen et al. (2010); and Kremer et al. (2011).

Table 2-3: Internal and external clustering validation measures

Internal Measures	External Measures
Dunn's indices (Dunn, 1973)	Completeness (Rosenberg & Hirschberg, 2007)
Tau A (L. J. Hubert & Levin, 1976)	Purity (Zhao & Karypis, 2004)
Tau C (L. J. Hubert & Levin, 1976)	Homogeneity (Rosenberg & Hirschberg, 2007)
Tau (Rohlf, 2003)	Precision (Van Rijsbergen, 1979)
Silhouette coefficient (Rousseeuw, 1987)	Recall (Van Rijsbergen, 1979)
Sum of square distance (SSQ) (Aggarwal et al., 2003)	Rand index (J. Wu et al., 2009)
Davies-Bouldin index (Davies & Bouldin, 1979)	F-measure (Van Rijsbergen, 1979)
Gamma (Baker & Hubert, 1975)	V-measure (Rosenberg & Hirschberg, 2007)
Log Likelihood (Hartigan, 1975)	Hubert Γ statistics (L. Hubert & Arabie, 1985)
Adj. Ratio of Clustering (L. J. Hubert & Levin, 1976)	Minkowski score
Calinski-Harabasz index (Caliński & Harabasz, 1974)	Cluster-based entropy (Zhao & Karypis, 2004)
Fagan's Index (L. J. Hubert & Levin, 1976)	Adjusted Rand Index (L. Hubert & Arabie, 1985)

2.7 Summary

In summary, this section discussed data stream clustering and its recent adoption. Several windowing techniques (*landmark window*, *damped window*, *sliding window* and *tilted window*) were presented. Data stream clustering techniques were also explained (partitioning-based, hierarchical-based, density-based, grid-based model-based, and graph-based). Several distance measures like *Minkowski, Manhattan, Euclidean, squared Euclidean distance, Chebyshev, Cosine similarity,* and *Jaccard index* were described. Lastly, the performance evaluation metrics for evaluation like *Recall, Rand index, Precision, Adjusted Rand Index, Purity, Clustering Mapping Measure* (CMM), and *Silhouette Coefficient* were described.

CHAPTER 3: Research Methodology

In this chapter, the research objective to "the method of building a MOA repository from source to implement the modified algorithm" was described. The recent version of MOA framework for data stream clustering is presented. The state-of-the-art data stream clustering algorithms (*DenStream, ClusTree* and *CluStream*) in MOA and the proposed modified *DenStream* is described. The repository for stream datasets will also be shown. The performance metrics used for the experimental evaluation will be outlined. Finally, the process of obtaining an ethics clearance certificate (see Appendix A) for this study will be described.

3.1 Massive Online Analysis Graphical User Interface (GUI)

The recent version of MOA released in April 2023 is MOA-2023.04.0-bin. This can be downloaded along with Java JDK 8 or later. Alternatively, it can be clone from Git repository using commands:

- git clone https://github.com/Waikato/moa.git or
- git clone https://github.com/MatthiasCarnein/moa.git

Download and 'import' the project in Intellij IDEA. MOA can be run using the graphical user interface (GUI) (see Figure 3-1 and Figure 3-2). There are two important files to run MOA, the moa.jar and sizeofag-1.0.4.jar. The files are located at the lib directory of the MOA framework. To run the command, change directory to the lib in MOA framework and execute the commands:

```
java -cp moa.jar -javaagent:sizeofag-1.0.4.jar moa.gui.GUI
```

The -Xmx4G can be included to increase the maximum heap size to 4GB when the default setting of 16 to 64MB appears too small (Akinosho et al., 2023). MOA can also be built from the source code using the guidelines presented in Gomes et al. (2020). The process involves downloading and installing IntelliJ IDEA latest version and importing the moa.git from GitHub. The step-by-step method is described in McDonald (2015). Another way of building MOA code from source is given by Gonmes et. al. (2020) using the IntelliJ IDEA Community version. In this dissertation, the IntelliJ IDEA Community Edition 2022.2 was used.

Figure 3-1: MOA Graphical User Interface (GUI).

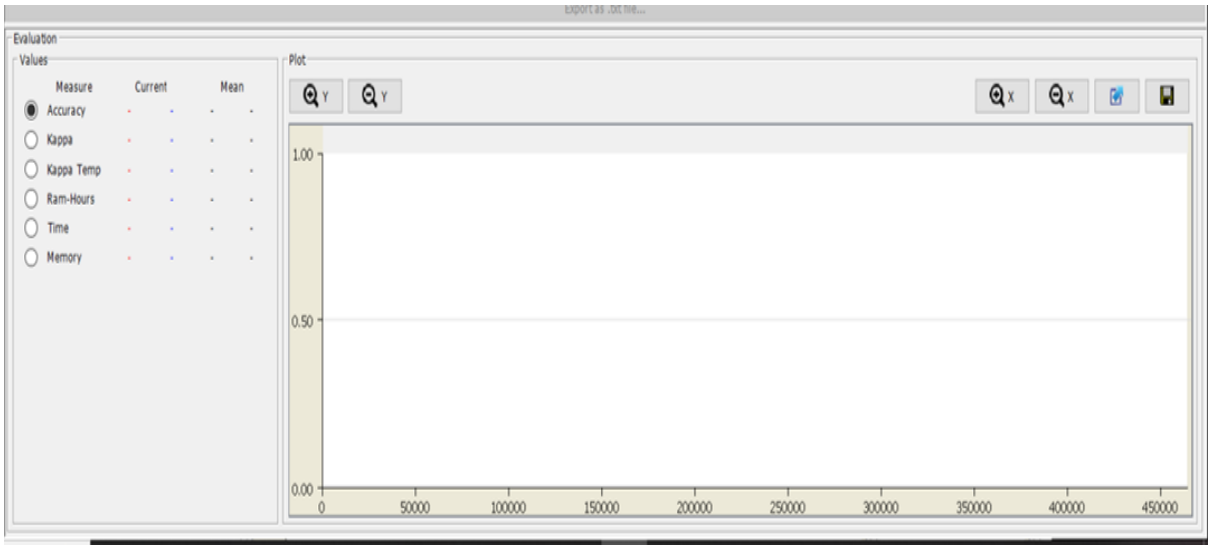


Figure 3-2: MOA graph output interface

3.2 State-of-the-art Clustering

This dissertation used the three state-of-the-art data stream clustering algorithms in MOA:

- *CluStream* (Aggarwal et al., 2003): The *Clutream* algorithm is an online-offline clustering. An online which is a micro clustering model and an offline which is a macro clustering model. The *CluStream* is a partitioning-based algorithm with spherical-shaped cluster. However, *CluStream* is sensitive to outliers and unable to detect arbitrary-shaped clusters.

- *DenStream* (Cao et al., 2006): The *DenStream* algorithm is a density-based algorithm with the ability to discover arbitrary-shaped clusters in an evolving data stream. The DenStream algorithm can handle outliers, but it is risky when there is noise. The *DenStream* algorithm has three micro-cluster features which are: *core micro-cluster* for summarizing clusters with arbitrary-shapes; *potential core micro-cluster* to identify potential clusters; and *outlier micro-cluster* for outliers and not dependent on many user-defined parameters.
- *ClusTree* (Kranen et al., 2011): The *ClusTree* is a hierarchical-based algorithm that can adapt to the speed of the stream due to its parameter less. ClusTree can detect outliers, novelty in the stream, and concept drift.

The parameter settings for *CluStream*, *ClusTree*, and *DenStream* in MOA are preset and when selected appear as shown in Figure 3-3, Figure 3-4, and Figure 3-5 respectively. MOA's default parameter setting for *DenStream* is different from Cao et al. (2006).

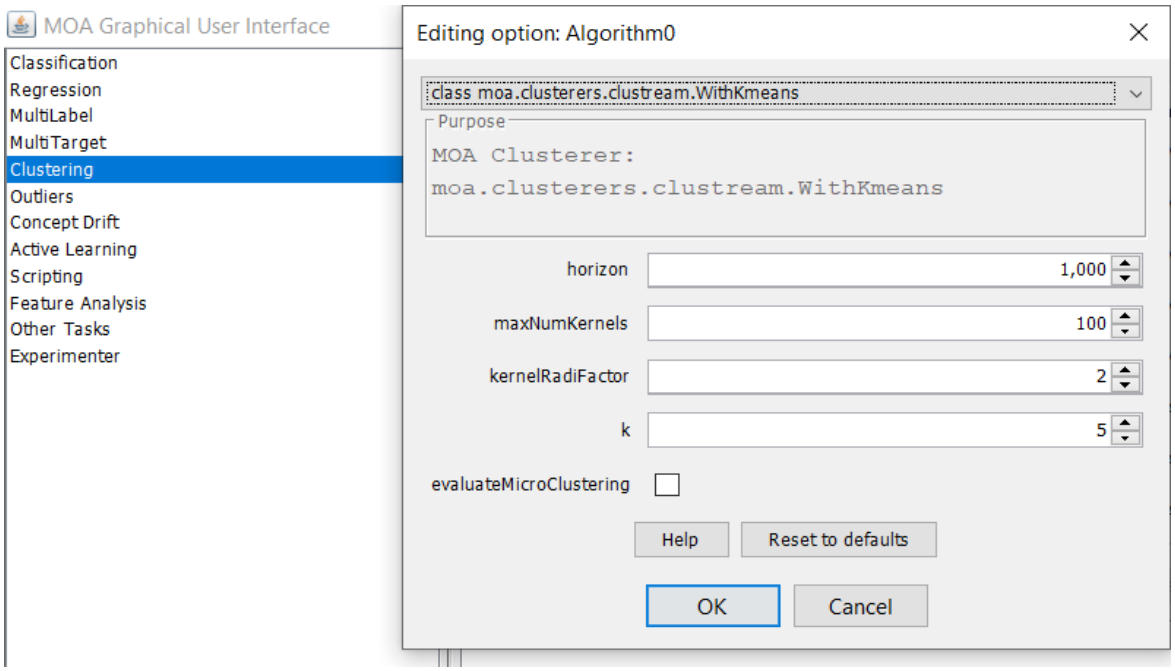


Figure 3-3: MOA Clusterer: CluStream parameter setup

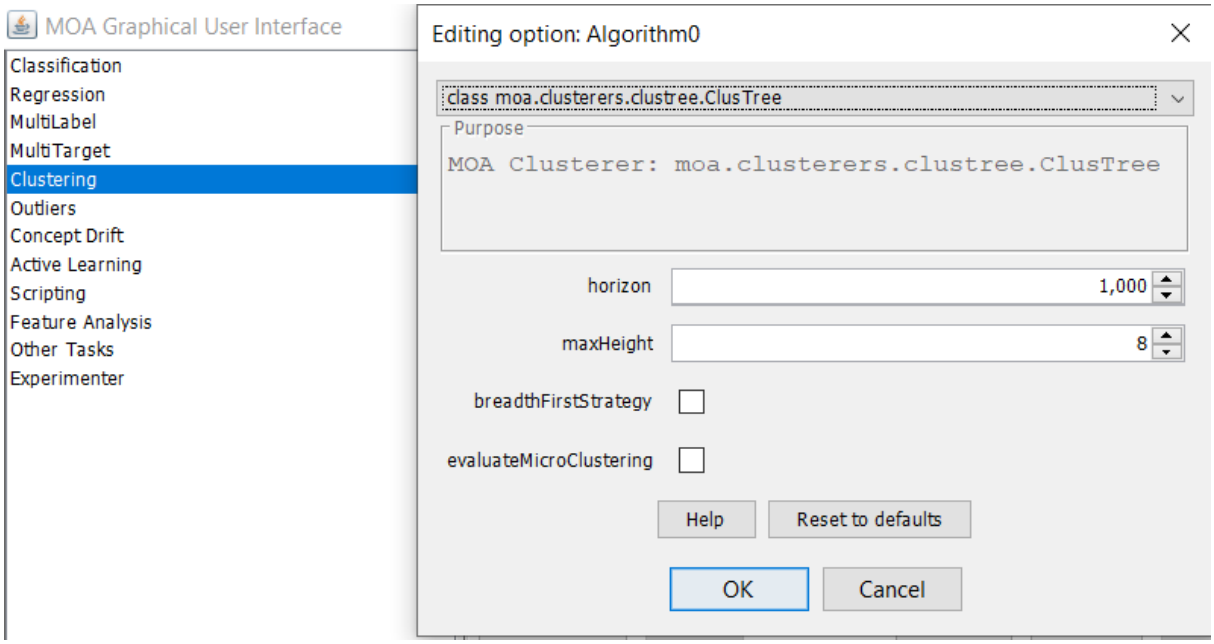


Figure 3-4: MOA Clusterer: ClusTree parameter setup

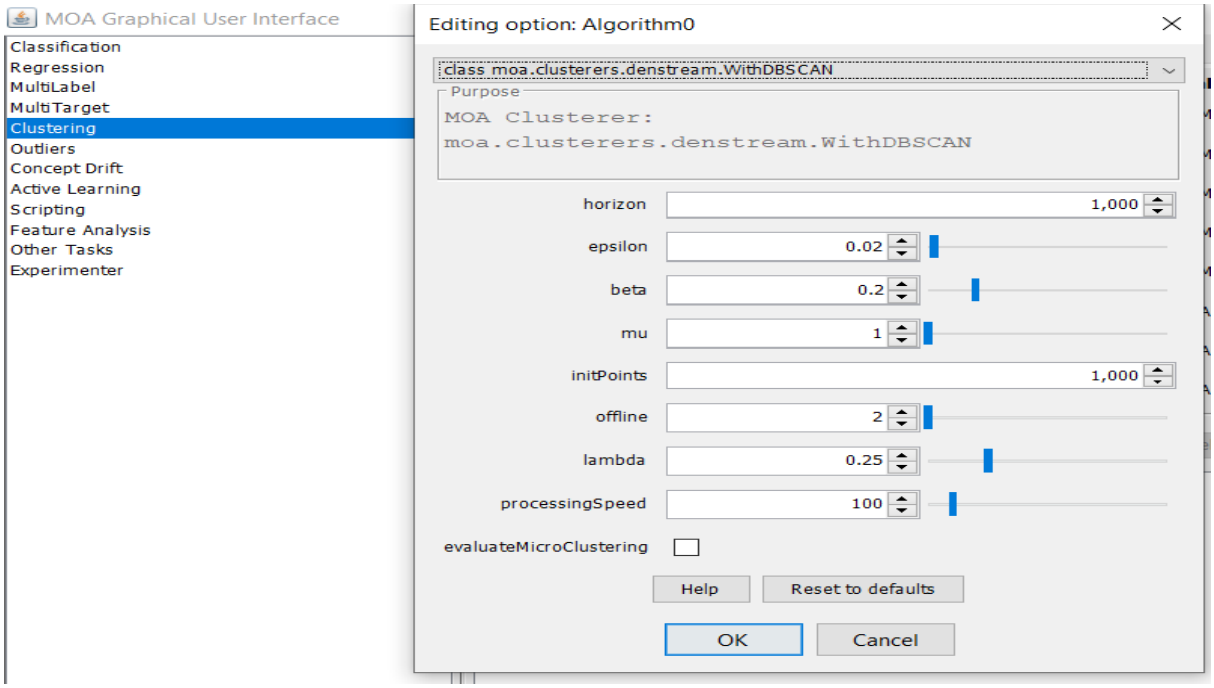


Figure 3-5: MOA Clusterer: DenStream parameter setup

The summary of the algorithms and their parameters is given in Table 3-1, while the description of the parameters of *DenStream* clustering is shown in Figure 3-6.

Table 3-1: Summary of algorithms and parameters in MOA (adapted from Carnein et al., 2020b).

Algorithm	Configuration	Type	Range	Default
DenStream	*E*	Numeric	[0,1]	0.02
	B	Numeric	[0,1]	0.2
	M	Integer	{0…10000}	1
	O	Integer	{2…20}	2
	L	Numeric	[0,1]	0.25
CluStream	*K*	Integer	{2…20}	5
	M	Integer	{1…10000}	100
	T	Integer	{1…10}	2
ClusTree	*H*	Integer	{1…20}	8
	B	Boolean	{1,0}	1

Phase	Parameter	Description	Value range
Initializing	*initial_points*	Number of points for initializing potential- and outlier-cluster list	1~maximum integer
	min_points	Minimum points for constructing a cluster when initializing with DBSCAN	1~maximum integer
Online	*epsilon*	Maximum radius for a core-micro-cluster	Float value larger than 0
	lambda	Decay factor for weight	0–1
	beta	Weight threshold for outlier-clusters	0–1
	mu	Weight threshold for core-micro-clusters	0~maximum float value
Pruning	*tp*	Time interval for pruning	Any rational time interval
Offline	*offline*	Multiplier of *epsilon* for meaningful clusters in DBSCAN	2~maximum spatial range of the input data stream

Figure 3-6: Parameters in DenStream adapted from Li et al. (2020)

There are several options for analyzing evaluation outputs in MOA given in Kranen et al. (2010) see Figure 3-7:

- The running stream can be stopped with the result passed to a WEKA explorer for further analysis (see Figure 3-8).
- The evaluation measures can be stored at every time intervals in .csv file format and analyzed offline using any programming languages (Python, R, gnuplot etc). This is the approach adopted for this dissertation.

- We can visualize the clustering results using the performance metrics online. Two algorithms or one algorithm with two different parameter settings can be visualized at the same time.

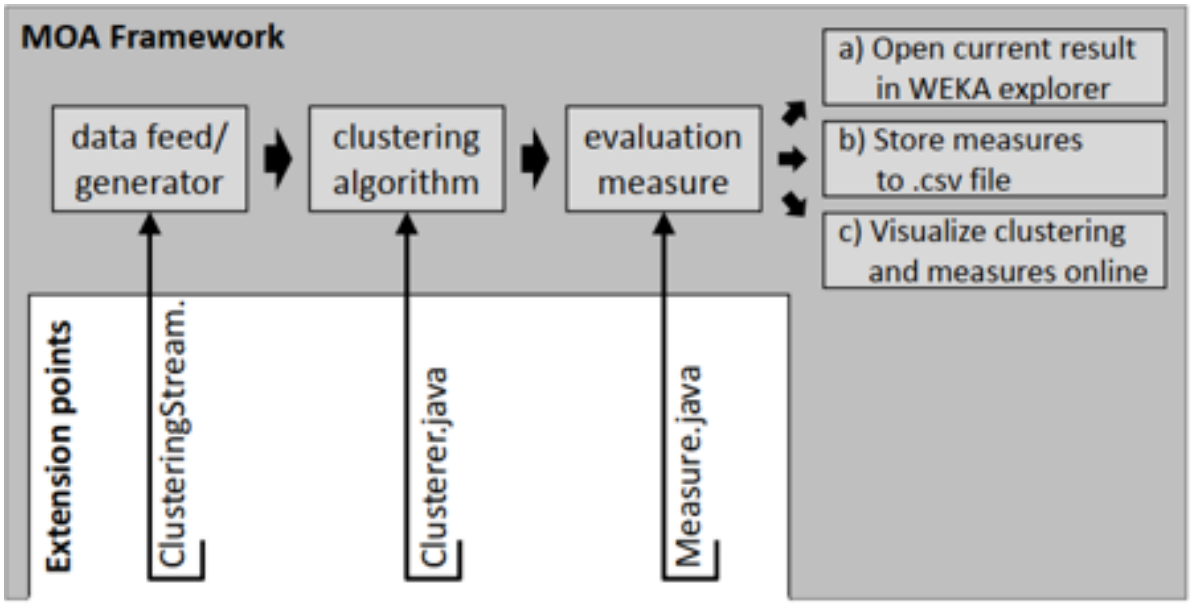


Figure 3-7: MOA framework adapted from Kranen et al. (2010).

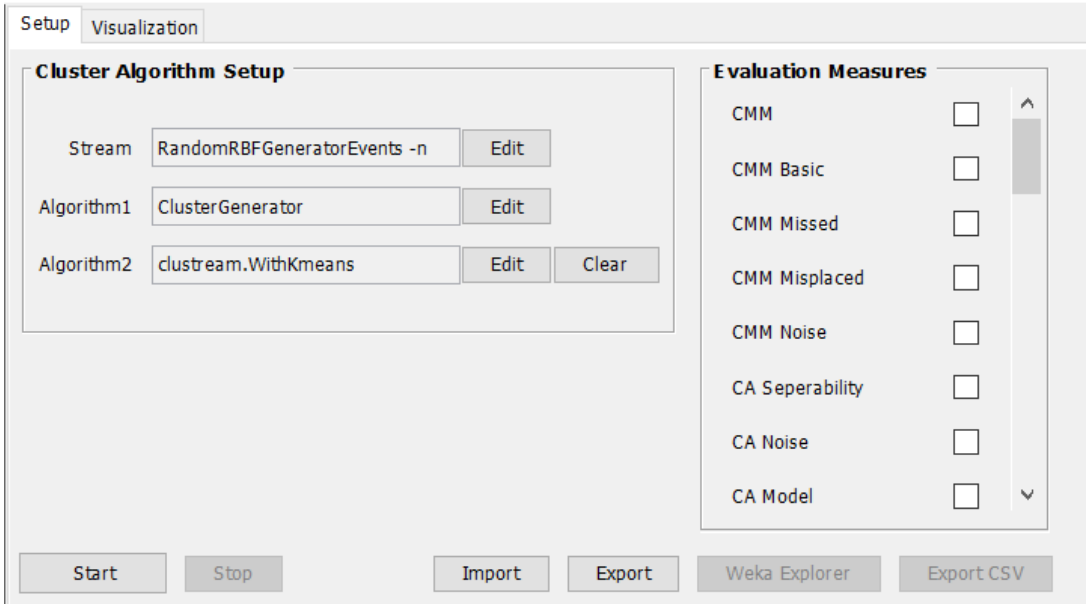


Figure 3-8: Clustering algorithm setup and result output options**.**

3.3 The modified DenStream

We implemented a modified distance/similarity measure of *DenStream* algorithm in MOA to improve it. The *DenStream* algorithm is divided into two phases (online and offline). The online is described in Algorithm 1 and the offline phase in Algorithm 2 adapted from Cao et al. (2006). In the offline phase, *DenStream* algorithm applies DBSCAN algorithm and has been proving to be computationally expensive. The proposed method is aimed at reducing the computational overheads through modifying the distance/similarity measure, by taking the absolute value of the squared differences rather than taking the square root as implemented in Euclidean distance.

Algorithm 1 Merging (p)

```
1: Try to merge p into its nearest p-micro-cluster c_p;
2: if r_p (the new radius of c_p) ≤ ε then
3:    Merge p into c_p;
4: else
5:    Try to merge p into its nearest o-micro-cluster c_o;
6:    if r_o (the new radius of c_o) ≤ ε then
7:       Merge p into c_o;
8:       if w (the new weight of c_o) > βμ then
9:          Remove c_o from outlier-buffer and create a
            new p-micro-cluster by c_o;
10:      end if
11:   else
12:      Create a new o-micro-cluster by p and insert it
         into the outlier-buffer;
13:   end if
14: end if
```

Algorithm 2 DenStream $(DS, \epsilon, \beta, \mu, \lambda)$

$T_p = \lceil \frac{1}{\lambda} \log(\frac{\beta\mu}{\beta\mu-1}) \rceil$;
Get the next point p at current time t from data stream DS;
Merging(p);
if (t mod T_p)=0 **then**
  for each p-micro-cluster c_p **do**
    if w_p(the weight of c_p)< $\beta\mu$ **then**
      Delete c_p;
    end if
  end for
  for each o-micro-cluster c_o **do**
    $\xi = \frac{2^{-\lambda(t-to+T_p)}-1}{2^{-\lambda T_p}-1}$;
    if w_o(the weight of c_o)< ξ **then**
      Delete c_o;
    end if
  end for
end if
if a clustering request arrives **then**
  Generating clusters;
end if

The modified *DenStream* was implemented in Java, compiled, and run on IntelliJ IDEA Community Edition 2022.2 as described in Section 3-1.

3.4 Data Collection

Real-world benchmarks and publicly available datasets are from the University of California Irvin (UCI) Machine Learning Repository (Dua & Graff, 2019); USP Data Stream Repository (*USP DS Repository*, n.d.); Stream Clustering (Carnein, 2019); Stream Data Mining Repository (X. Zhu, 2010); and Outlier Detection Datasets (ODDS) repository (Rayana, 2016). However, there is a shortage of suitable datasets for data stream mining which often results in researchers using synthetic datasets. A synthetic dataset has the advantage of being reproduced and cost-effective in terms of storage and transmission. In this research, the synthetic dataset generator available in MOA was used as well as the publicly available datasets suitable for data streaming tasks.

Papers/articles were sourced from Google Scholar; the Association of Computing Machinery (ACM) website; the University of South Africa library, Mendeley Reference Manager Software; and wizdom.ai website (https://www.wizdom.ai); to study the limitations of data stream clustering. Keywords with synonyms were used to search current papers/articles and imported into Mendeley Reference Manager. Those papers/articles that have limited information were populated and set to the required citation reference style.

3.5 Datasets

In this dissertation, a streaming synthetic dataset and real-world datasets were used to evaluate the performance of the modified *DenStream* algorithm against other algorithms (*CluStream, ClusTree,* and *DenStream).* The streaming synthetic dataset was generated in MOA with the *RandomRBFGenerator while* the two real-world datasets *Electricity* and *Forest Covertype* are publicly available.

3.5.1 Synthetic dataset

- **RandomRBFGenerator:** The RandomRBFGenerator stream incessantly changes and varies the true cluster location. It is available in MOA with some parameters such as *numClusterRange, kernelRadius, modelRandomSeed, instantRandomSeed,* and *numCluster* among others.

3.5.2 Real World datasets

- **Forest Covertype dataset:** The Forest Covertype contains 581,012 instances, 54 attributes where 10 are continuous attributes and the rest are binary attributes. The The Forest Covertype is classified into seven types. The dataset is readily available at the UCI machine learning site, and it is from the US Forest Service (USFS). The Forest Covertype has been widely applied in several data stream studies. In this dissertation, the normalized version is used.
- **Electricity dataset:** The Electricity data is publicly available from the Australian New South Wales Electricity Market. The Electricity data has 45,312 instances. For performance optimization, the normalized version of the Electricity data is used in this dissertation. The initial stream used is 5000 instances and then 40000 instances. At the end, 45000 instances were used for the analysis.

3.6 Evaluation Platform Parameter Setup

The experiment was executed using the current MOA release-2023.04.0 on HP ProBook 450 G7, Processor: Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz 2.11 GHz; RAM: 16.00 GB. System type: 64-bit, x64, Operating System: Windows 10 Pro. This is tabulated in Table 3-2.

Table 3-2: System setup

Component	Description
System	HP ProBook 450 G7
Processor	Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz 2.11 GHz
RAM size	16.00 GB
System type	64-bit x64
Operating System	Windows 10 Pro

3.7 Performance Evaluation

Performance evaluation metrics are divided into two types: *extrinsic* and *intrinsic* methods (Ahmed et al., 2020). The ground-truths when available is known as an *extrinsic/external* method and it is a supervised method, otherwise it is the *intrinsic/internal* method, and it is an unsupervised method. Performance evaluation measures are important in the determination of the quality of clustering results (Kremer et al., 2011). The external measures employ the ground truth for comparing clustering but are lacking in most streaming applications (Ghesmoune et al., 2016). The internal evaluation measures the compactness and separation (structure and properties) of clusters (Hassani and Seldi, 2017; Kremer et al., 2011).

There are numerous evaluation measures available in MOA. Examples are *Clustering Mapping Measure* (CMM), *Recall, Rand index, Precision, Purity, Sum of Square distance (SSQ), Completeness, Homogeneity,* and *Silhouette.* It should be noted that not all evaluation measures are appropriate for all clustering types. They have the disadvantages of (1) not being able to handle overlying due to merging, drifting, and noise; and (2) achieving suboptimal results even with the ground truth test. However, CMM addressed these shortcomings (Kremer et al., 2011). The evaluation measures (completeness and homogeneity) are only appropriate for offline and static clustering with ground truth (Puschmann et al., 2017). In this research, four of the evaluation measures for clustering quality consisting of both external and internal measures were used. They are *CMM, Purity, Silhouette Coefficient, and Rand index.* Their descriptions and formulas have given in Section 2.6, equations 2-14, 2-19, and 2.20 of Chapter Two.

3.8 Ethical Clearance

A request for the research ethics clearance was made to the *Unisa College of Science, Engineering and Technology's (CSET) Ethics Review Committee* on 7 December 2020 in compliance with *the Unisa Policy on Research Ethics and the Standard Operating Procedure on research Ethics Risk Assessment*. Ethics approval was granted for three years until December 2023. The clearance certificate is attached in Appendix A.

3.9 Summary

In summary, this chapter presented the MOA framework and described the data stream clustering algorithms used in this dissertation. The data collection source and the datasets for the study were also presented. The performance evaluation metrics which comprise both internal and external measures were described with emphasis on the performance metrics used in this dissertation.

CHAPTER 4: Experimental Results and Analysis

This chapter presents the research objective to "identify the hyperparameters appropriate for parameter-tuning" by carrying out parameter setup for the synthetic generator in MOA using *RandomRBFGenerator* and the state-of-the-art algorithms (*CluStream, ClusTree, DenStream*) using their default parameter setup. The parameter-tuning for the research was demonstrated. The experimental results were merged and visualized using Python libraries (Pandas, Plotly, and hvplot). The research objective to "identify the performance metrics applicable for clustering quality" was described. The experimental results will be presented to demonstrate the clustering quality of performance evaluation metrics (CMM, Purity, Silhouette coefficient, and Rand index).

4.1 Experimental Parameter Setup

Clustering algorithm parameter settings are important in achieving proper micro-clusters (Mansalis et al., 2018). The experimental setup for the synthetic generator in MOA using *RandomRBFGenerator* and the state-of-the-art data stream clustering algorithms (*CluStream, ClusTree, DenStream*) are given in Figure 3-3, Figure 3-4, Figure 3-5. The parameter setting in MOA is not automated but done manually and are presented as follows:

The *RandomRBFGeneratorEvents* default parameter setup in MOA.

- KernelRadius = 0.025 (The average radii of the centroids in the model)
- Noise = 0.1 (i.e, every one-tenth data item is randomly generated)
- Speed = 500 (Kernels move a predefined distance of 0.01 every X point)
- SpeedRange = 10 Speed/Velocity point offset)
- noiseLevel (default: 0.1) (Noise level)
- EventFrequency = 50000 (Event frequency. Enable at least one of the events below and set numClusterRan)

The DenStream (with DBSCAN) default parameter setup in MOA.

- ✓ horizon = 1000 (Range of the window)
- ✓ epsilon = 0.02 (Defines the epsilon neighbourhood)
- ✓ beta = 0.2,
- ✓ mu = 1,
- ✓ initPoints = 1000 (Number of points to use for initialization)

- ✓ offline = 2 (offline multiplier for epsilion)
- ✓ lambda = 0.25,
- ✓ processingSpeed = 100 (Number of incoming points per time unit)

The CluStream (WithKmeans) default parameter setup in MOA.

- ✓ horizon = 1000 (Range of the window)
- ✓ maxNumKernels = 100 (Maximum number of micro kernels to use)
- ✓ kernelRadiFactor = 2 (Multiplier for the kernel radius)
- ✓ k=5 (k of macro k-means (number of clusters))

The ClusTree default parameter setup in MOA.

- ✓ horizon = 1000 (Range of the window)
- ✓ maxHeight = 8 (The maximal height of the tree)
- ✓ breadthFirstStrategy (Use breadth first strategy)

Two vital user-defined parameters of *DenStream* algorithm are (1) the outlier threshold β or beta; and (2) the decay factor λ or lambda as stated in Mansalis et al. (2018)which controls the importance of historical objects.

4.2 Experimental with default settings

This section addressed the research objective 2 and research objective 3. It demonstrates the effects of parameter tuning and compares the algorithms *CluStream, ClusTree,* and *DenStream* on synthetic data with a manually fixed default 10% noise level, 0% noise level, and 30% noise level on *RandomRBFGenerator* for 205000 instances. We have chosen the *RandomRBFGenerator,* which continuously evolves to alter the location of the true cluster as the input stream generator. The *RandomRBFGenerator* is the lone available streaming generator in MOA framework. The algorithms were also tested on real-world datasets (Electricity and Forest Covertype) using 205000 instances and 45000 instances respectively. The settings started with 5000 instances, then with 50000 instances until 205000 instances in reached. The output is presented in Figure 4-1 for *CluStream* and *DenStream.* The individual output is presented in Figure 4-2, Figure 4-3, and Figure 4-4 for *CluStream, ClusTree,* and *DenStream* respectively.

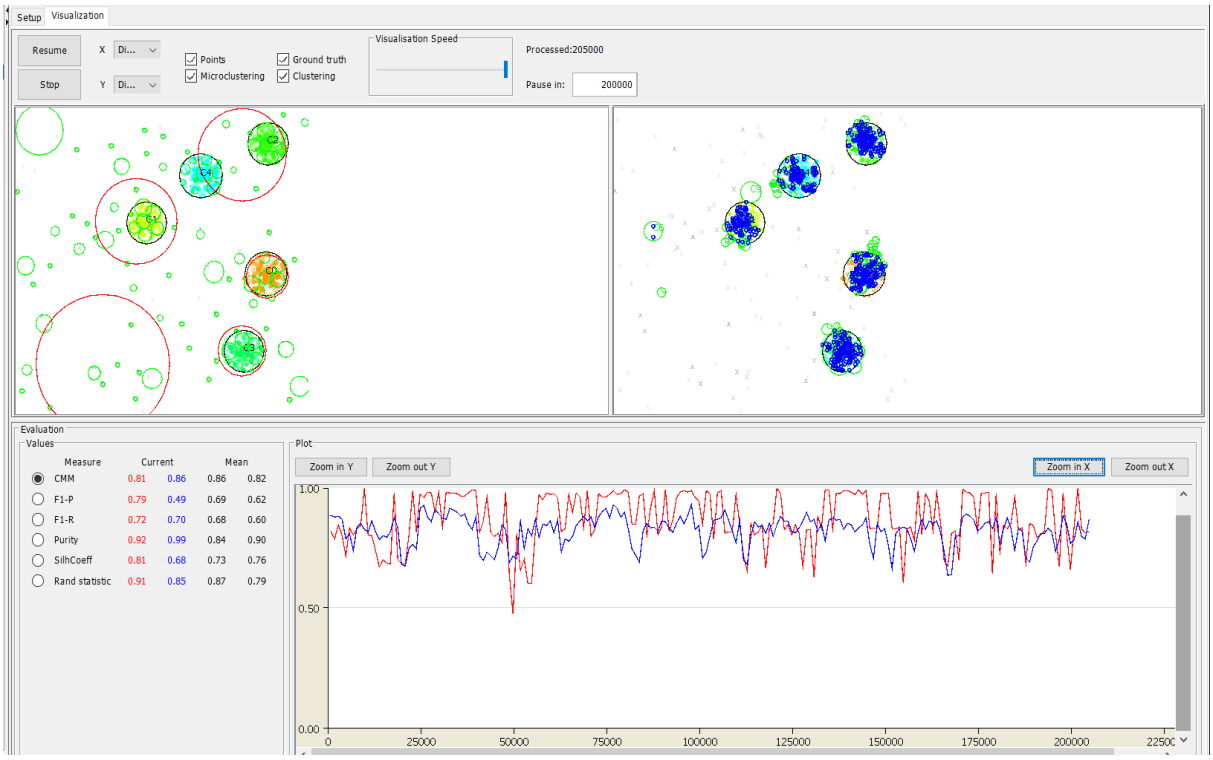


Figure 4-1: MOA screenshot of *RandomRBF*. *CluStream* is on the right while *DenStream* is on the left. The line graph is the output after 205000 instances.

Figure 4-2 shows the output of CluStream in (red contour) for RandomRBFGenerator with 10% noise after 205000 instances. The stream points display in deep colours, and the *black*/*gray-coloured* circles is the ground truth cluster boundaries. The former state is the *gray* circles, showing that the clusters are moving. The noise points represented by the *black* (faded out to gray) points. Micro-clustering in green contour, and clustering in red rings because CluStream was selected as Algorithm1.

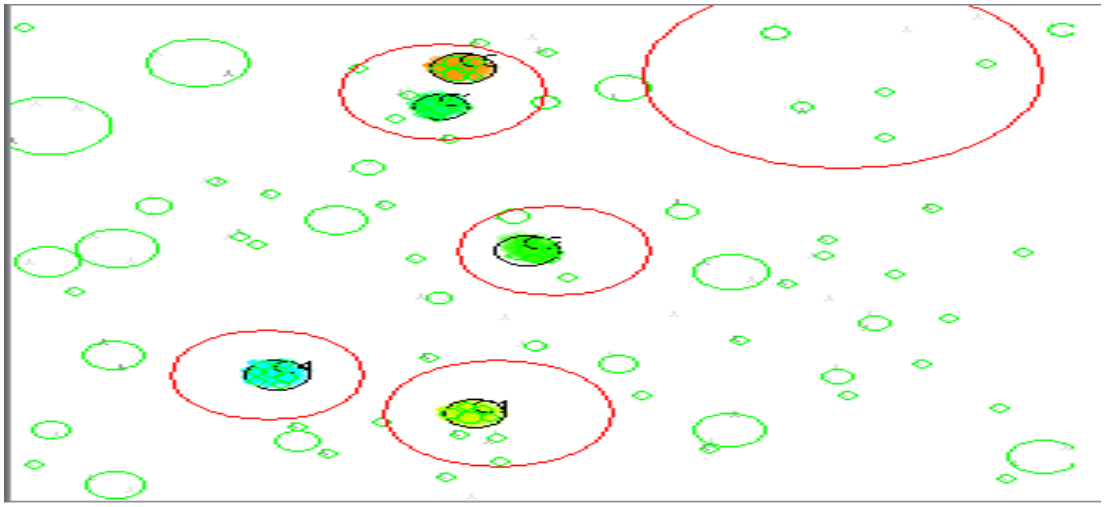


Figure 4- 2: CluStream (red contour) for RandomRBFGenerator with 10% noise after 205000 instances.

Figure 4-3 shows *ClusTree* in (blue contour) for *RandomRBFGenerator* with 10% noise after 205000 instances. The stream points are in deep colours, and ground truth display either *black*/*gray*-coloured circles, micro-clustering in green contour, and clustering in blue rings since ClusTree was selected as Algorithm 2.

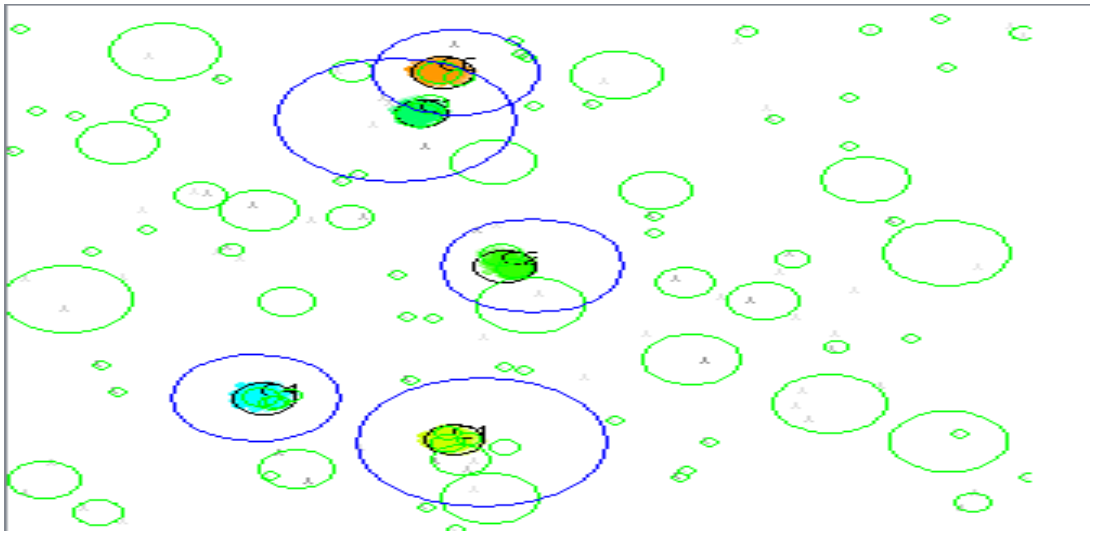


Figure 4-3: ClusTree (blue contour) for RandomRBF with 10% noise after 205000 instances.

Figure 4-4 displays the *DenStream* for *RandomRBFGenerator* with 10% noise after 205000 instances. The stream points are deep colours, the *black*/*gray-coloured* circles represent the ground truth cluster, micro-clustering in green contour, and clustering in blue colours because DenStream was selected as Algorithm 2.

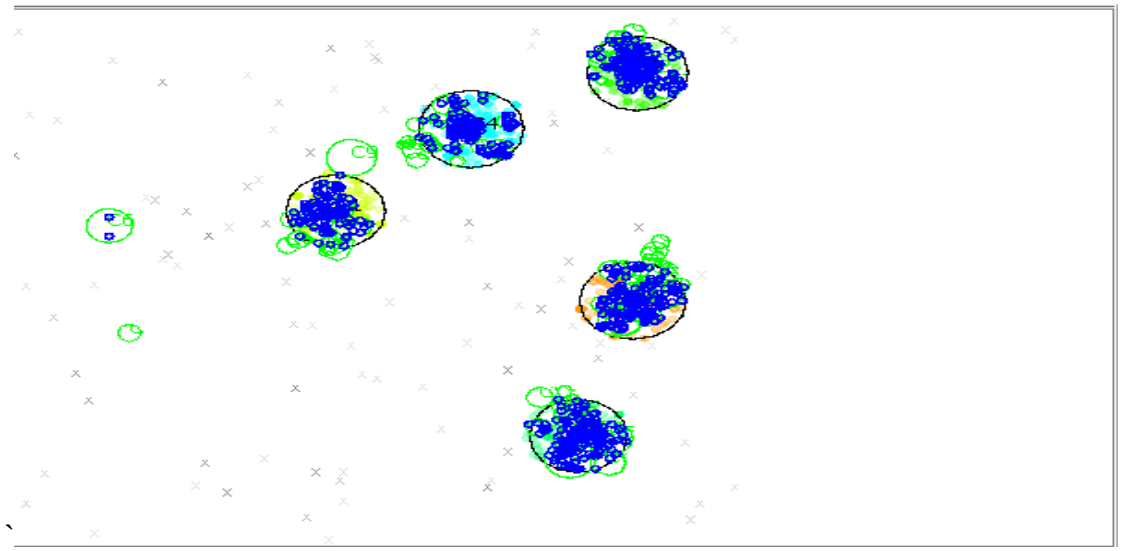


Figure 4-4: DenStream for RandomRBF with 10% noise after 205000 instances.

Figure 4-5 is the output of the modified *DenStream* for *RandomRBFGenerator* with 10% noise after 205000 instances. The stream points are in deep colours, the *black*/*gray-coloured* circles represent the ground truth cluster, micro-clustering in green contour, and clustering blue colour because the modified DenStream was selected as Algorithm 2.

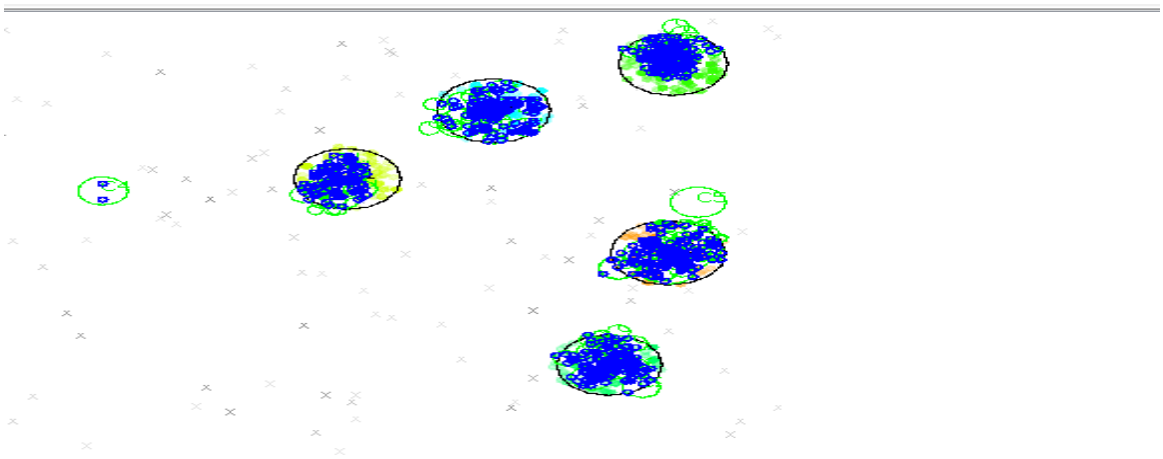


Figure 4-5: The modified DenStream for RandomRBF with 10% noise after 205000 instances.

The visualization of the performance metric CMM for *CluStream* against *DenStream* is presented in Figure 4-6. The figure shows *CluStream* in red colour and *DenStream* in blue colour, the x-axis is the scaling of the instances, and the y-axis is the metrics values. Other performance metrics (Purity, Silhouette Coefficient, and Rand index) running at background can be visualized (see Appendix B).

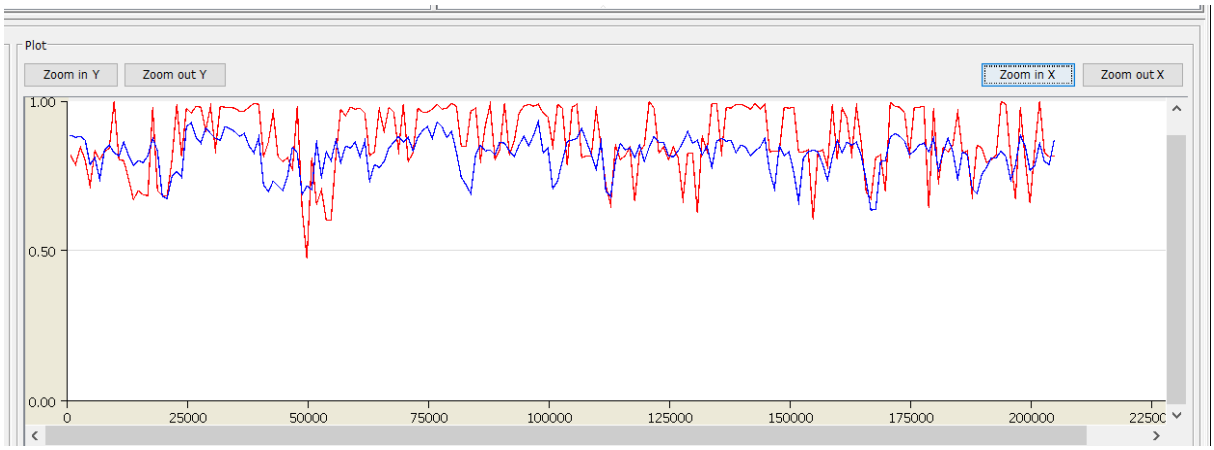


Figure 4-6: The line graph of *RandomRBF* with 10% noise level after 205000 instances using CMM metric for *CluStream* and *DenStream.*

The visualization of the performance metric CMM for *CluStream* against the modified *DenStream* is presented in Figure 4-7. The figure shows *CluStream* in red, modified *DenStream* in blue, x-axis represents the instances, and the y-axis is the metrics values. Other performance metrics (Purity, Silhouette Coefficient, and Rand index).

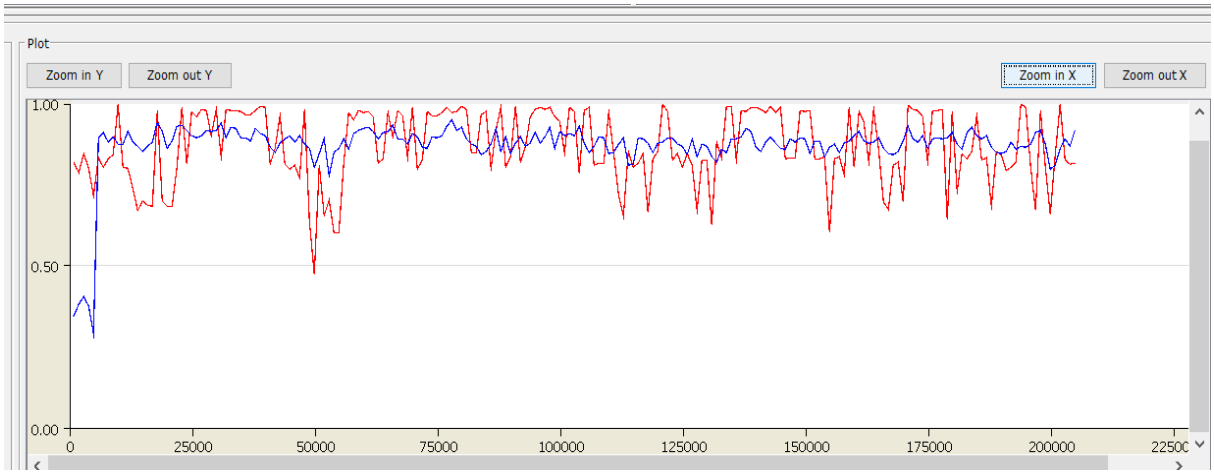


Figure 4-7: The line **g**raph of *RandomRBF* with 10% noise level after 205000 instances using CMM metric for *CluStream* and modified *DenStream.*

In Figure 4-8, the algorithms *CluStream* and *ClusTree* are run against each other. The stream points are in deep colours, ground truth in *black*/gray-*coloured* circle, micro-clustering in green contour, and *CluStream* in red-coloured circles because it was selected as Algorithm1 and *ClusTree* in blue-coloured circles because it was selected as Algorithm2.

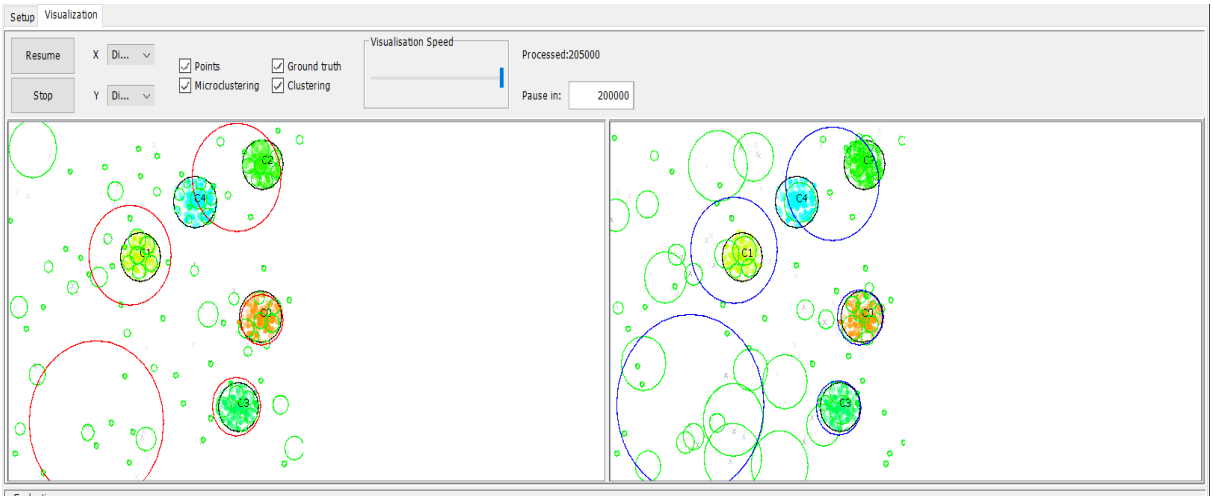


Figure 4-8: *RandomRBF* for *CluStream* on the left and *ClusTree* on the right after 205000 instances.

Figure 4-9 shows the visualization of the performance metric CMM on both *CluStream* and *ClusTree.* The figure shows *CluStream* in red and *ClusTree* in blue, The visualization of other performance metrics can be displayed if clicked (see Appendix B).

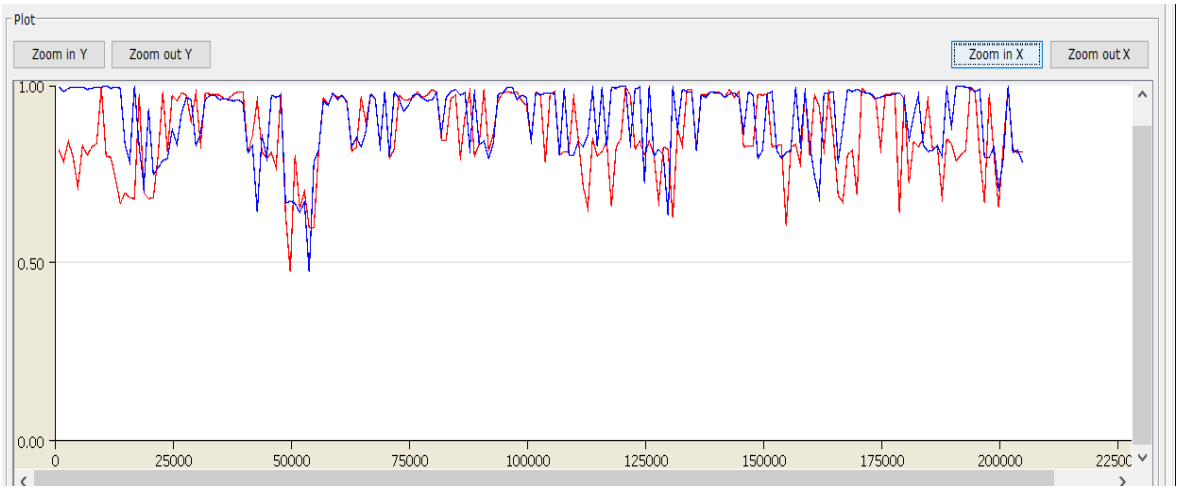


Figure 4-9: The line graph of *RandomRBF* with 10% noise level after 205000 instances using CMM metric for *CluStream* and *ClusTree.*

4.2.1 RandomRBFGenerator with default Noise Level

The 10% default noise level on *RandomRBFGenerator* for the different algorithms was demonstrated. The average value of the performance evaluation metrics (CMM, Purity, Silhouette Coefficient, and Rand index) was taken and is presented in Table 4-1.

The Massive Online Analysis (MOA) has no implementation for more than two clustering algorithms, the output for the paired algorithms was exported as a CSV file, merged in Microsoft Excel, and read using Python libraries (pandas and hvplot). The Jupyter Notebook was used to carry out the data visualization. The data visualization for *RandomRBFGenerator* with a 10% noise level for the algorithms using the performance metrics CMM is presented in Figure 4-10. The figure shows modified *DenStream* has its lowest drop at instance 5000 and moved up from instance 5100 to maintain an average value of 0.864978. *ClusTree* has its lowest drop at instance 53000 but has an overall average value of 0.902690. *CluStream* lowest point at instance 49000 and average value of 0.862702 for all instances. *DenStream* has its highest point at instance 97000 with a value of 0.925710 and has an overall average of 0.816985. Note the scaling on the x-axis is calibrated per 1000 instances on the hvplot for this dissertation.

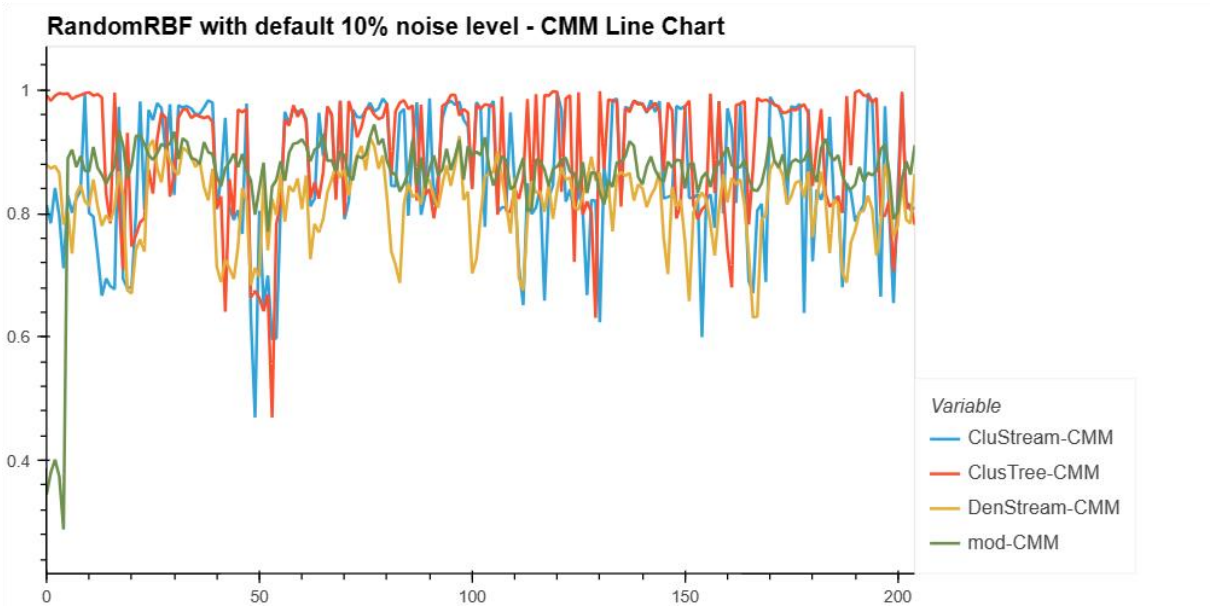


Figure 4-10: The line graph of RandomRBF using CMM on ClusTree, CluStream, DenStream, and modified DenStream.

In Figure 4-11, the performance metric Purity for *RandomRBFGenerator* with a noise level of 10% is presented. The line graph shows that *DenStream* has its lowest point at instance 23000. Its average value is 0.899071. *ClusTree* lowest point is at instance 20000 with a value of 0.664741 and its overall average value is 0.864506. *CluStream* dropped at these instances (106000, 130000, 135000, 154000, and 180000) and has an overall average value of 0.844106. The modified *DenStream* average value is 0.951441.

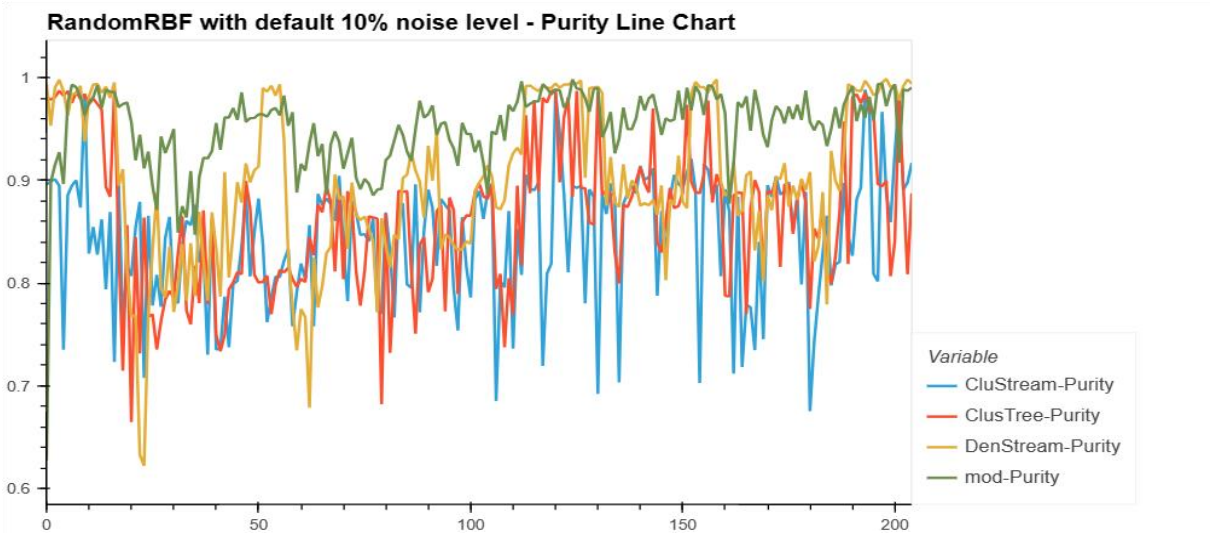


Figure 4-11: The line graph of RandomRBF with 10% noise level using Purity on ClusTree, CluStream, DenStream and modified DenStream.

The performance metric Silhouette Coefficient for *RandomRBFGenerator* with 10% noise level is presented in Figure 4-12. The figure shows that *CluStream* experienced its lowest point at instance 20000. Its overall average value is 0.729607. *DenStream* highest point occur at instance 193000 with a value of 0.973106 and lowest point at instance 39000 with a value of 0.430803. *ClusTree* lowest point is at instance 23000 and lowest point at instance 25000 with a value of 0.469584.

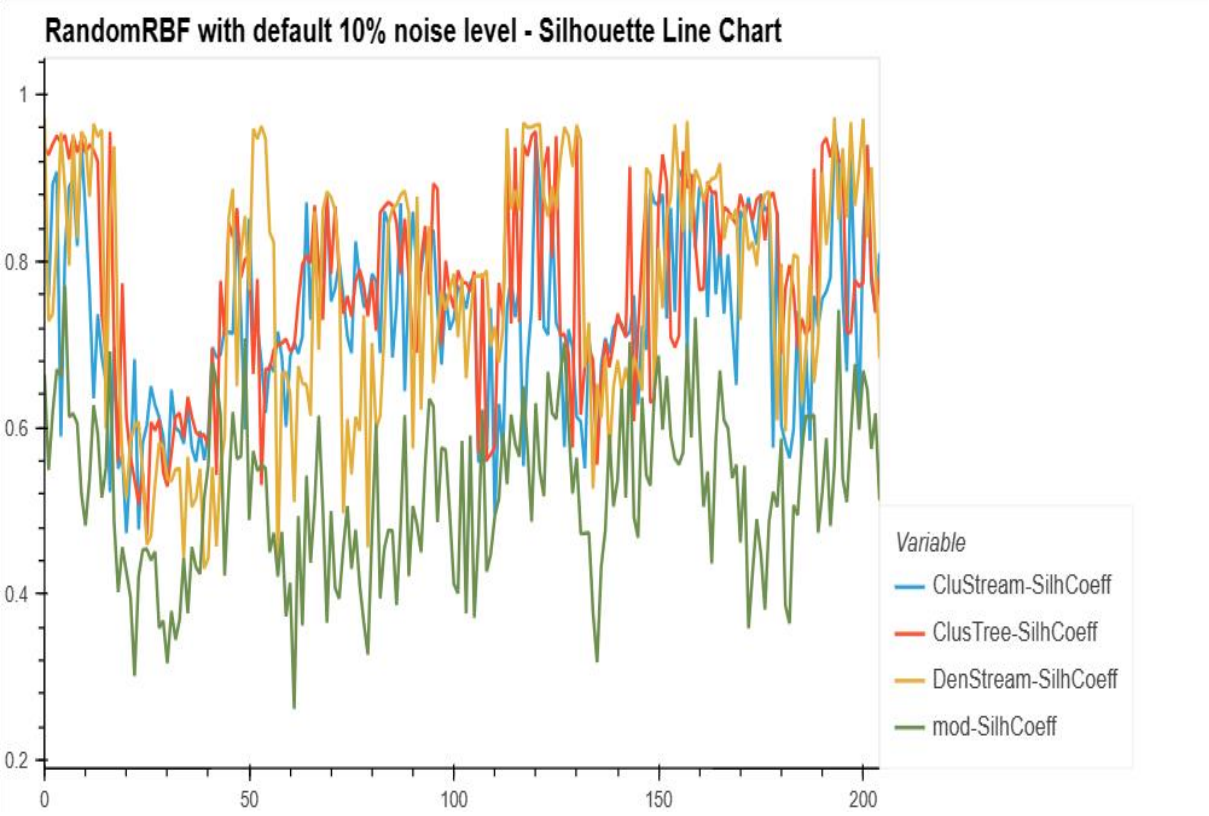


Figure 4-12: The line graph of RandomRBF with 10% noise level using Silhouette Coefficient on ClusTree, CluStream, DenStream and modified DenStream.

In Figure 4-13 for performance metric Rand index on *RandomRBFGenerator* with default 10% noise level. The figure shows that *DenStream* lowest point occurred at instance 23000 with 0.606088 and highest point at instance 128000 with a value of 0.884188. *ClusTree* has the highest point at instance 16000 with a value of 0.994054 and the lowest point at instance 23000 with a value of 0.687736. *CluStream* has its lowest point at instance 23000 with a value of 0.774931 and the highest point at instance 119000 with a value of 0.993235.

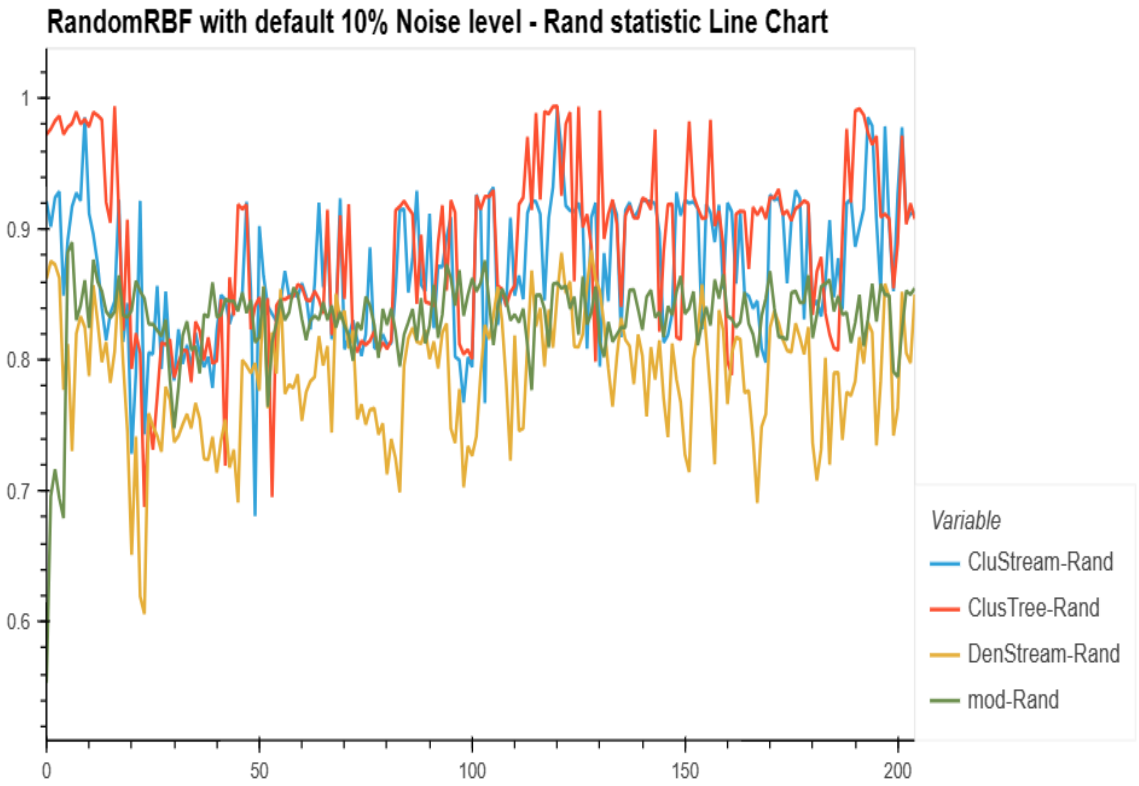


Figure 4-13: The line graph of RandomRBF with 10% noise level using Rand index on ClusTree, CluStream, DenStream and modified DenStream.

The results in Table 4-1 show that *ClusTree* outperforms other algorithms on performance metrics (CMM, Silhouette Coefficient and Rand index) with an average value of 0.902690, 0.771385, and 0.885936 respectively. The modified *DenStream* outperforms other algorithms on performance metric Purity with an average value of 0.951441. The modified *DenStream* also shows a better performance against *DenStream* on metrics CMM and Rand index.

Table 4-1: RandomRBF with default settings on the algorithms

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.862702	**0.902690**	0.816985	0.864978
Purity	0.844106	0.864506	0.899071	**0.951441**
Silhouette	0.729607	**0.771385**	0.762202	0.521549
Rand index	0.870000	**0.885936**	0.788309	0.831517

The performance metrics are combined using a bar chart and shown in Figure 4-14. This clearly explains the output in Table 4-1 showing the performance of the modified *DenStream* against other algorithms especially *DenStream* which is of interest in this dissertation.

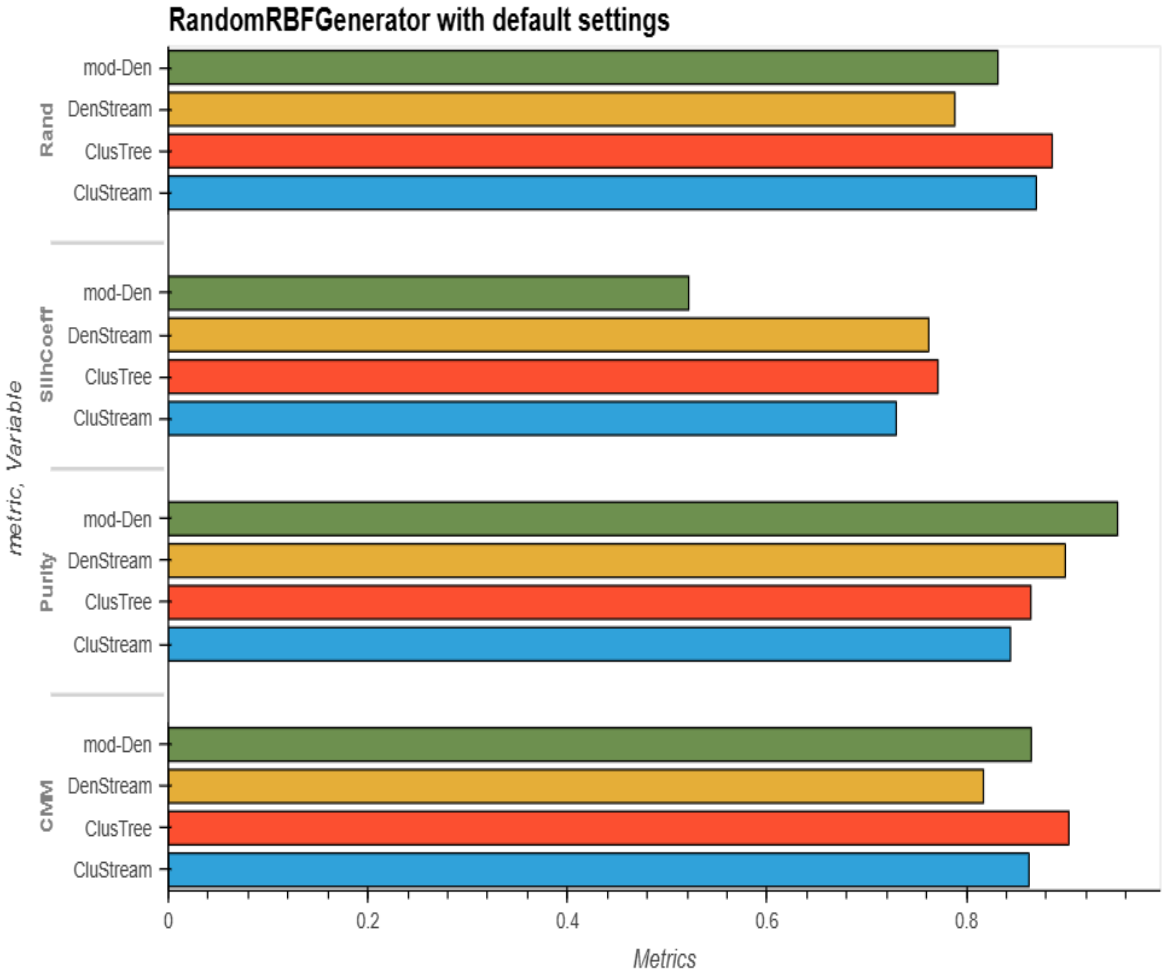


Figure 4-14: Performance metrics barplots of CluStream ClusTree, DenStream and modified DenStream on RandomRBF with default setting.

4.2.2 Forest Covertype with default settings.

The performance of the algorithms was demonstrated on Forest Covertype dataset using 205000 instances. The clustering qualities were evaluated using the metrics CMM, Purity, Silhouette Coefficient, and Rand index and the results of the algorithms *CluStream*, *ClusTree*, *DenStream* and modified *DenStream* are given in Figure 4-15, Figure 4-16, Figure 4-17, and Figure 4-18 respectively.

In Figure 4-15, the graph illustrates the *CluStream* algorithm on Forest Covertype dataset. Several features of the graph show clustering in red rings, the green rings represent the micro-clustering, and the black ring, C0, C1, and C4 represent the ground-truth.

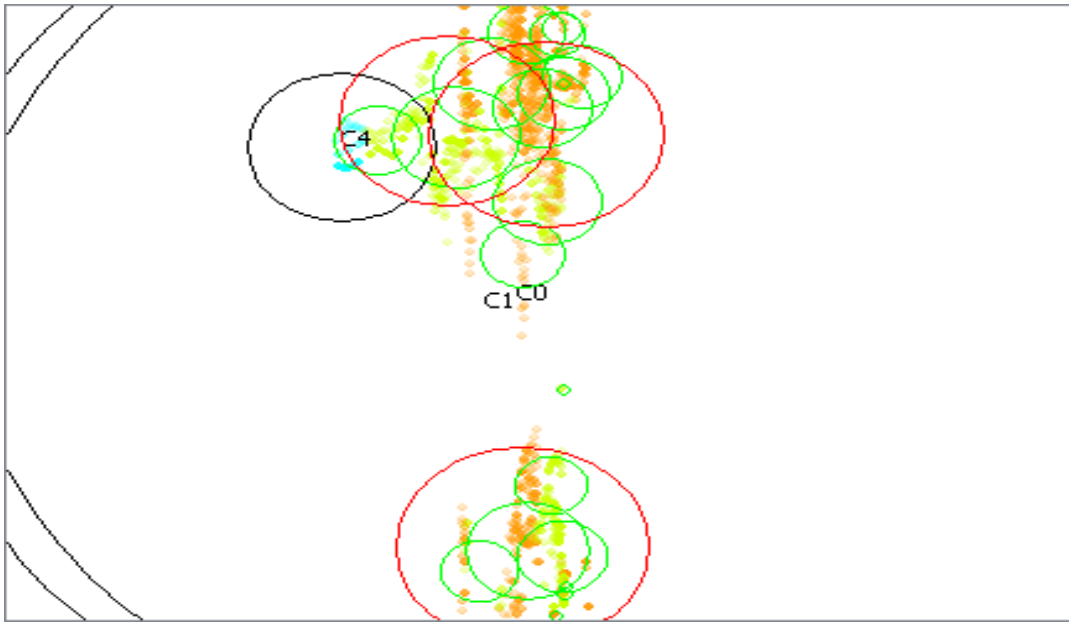


Figure 4-15: The Forest Covertype dataset on CluStream. The red rings show the clustering, the green rings are micro-clustering, and the black ring is ground-truth.

Figure 4-16 visualizes *ClusTree* algorithm on Forest Covertype dataset. The graph displays clustering in blue rings since it was selected as Algorithm2, the green rings represent the micro-clustering, and the black ring, C0, C1, and C4 represent the ground-truth.

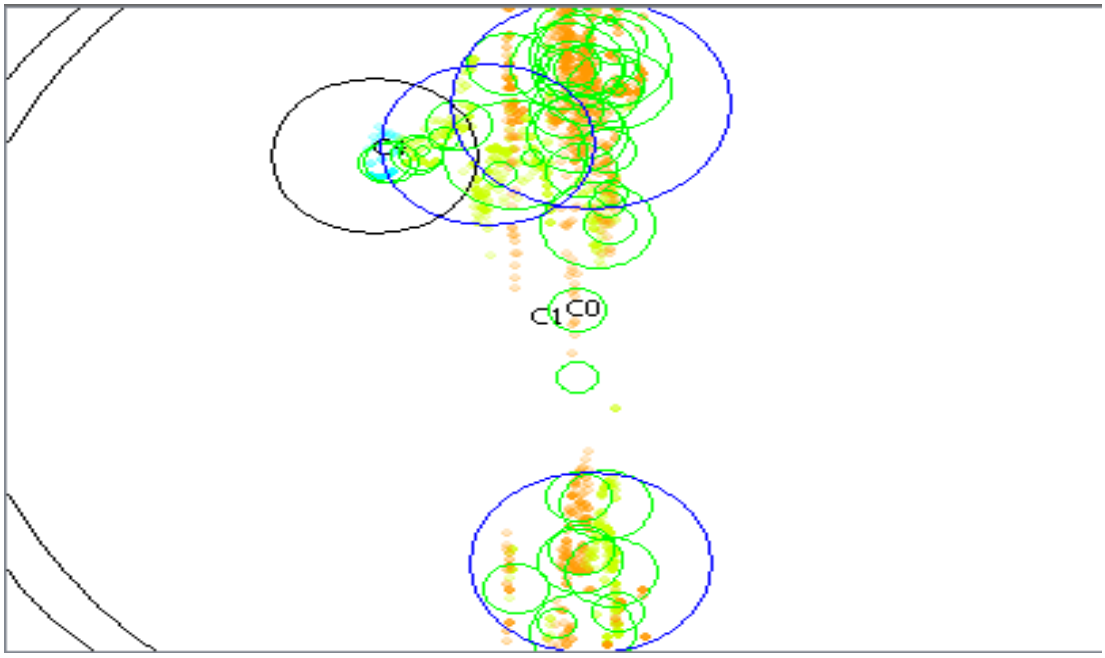


Figure 4-16: The Forest Covertype dataset on ClusTree. The blue rings show the clustering, the green rings are the micro-clustering, and the black ring is the ground-truth.

Figure 4-17 shows the output of *DenStream* on Forest Covertype dataset. In the figure, the clustering is indicated with tiny blue rings since it was selected as Algorithm2, the green rings represent the micro-clustering, and the black ring, C0, C1, and C4 represent the ground-truth.

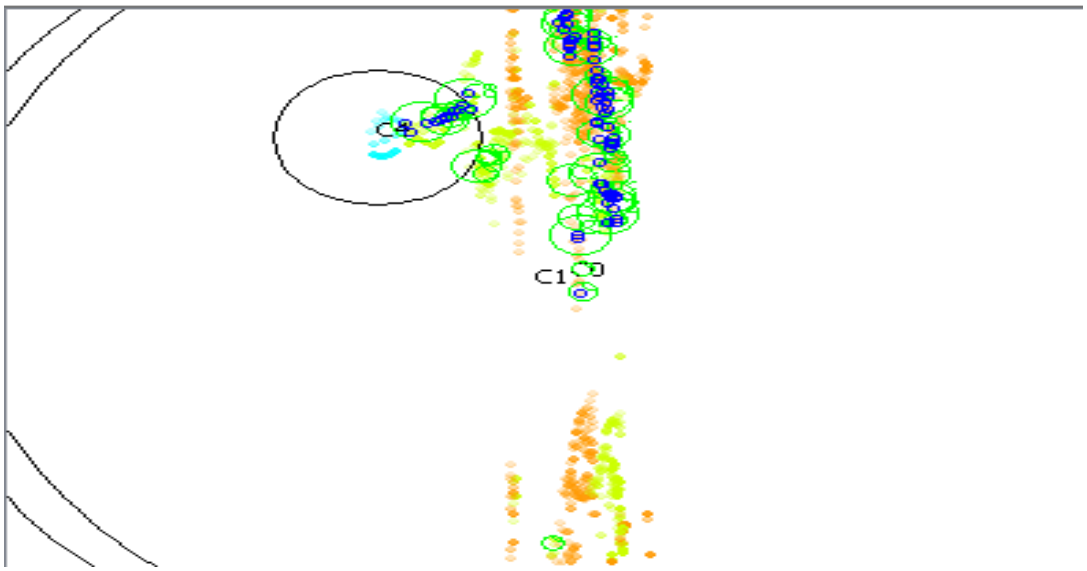


Figure 4-17: The Forest Covertype dataset on DenStream. The tiny blue rings are the clustering, the green rings, are the micro-clustering, and the black ring is the ground-truth.

Figure 4-18 visualizes the output of the modified *DenStream* on Forest Covertype dataset. As explained in the other algorithms, the tiny blue rings indicate the clustering, the green rings represent the micro-clustering, and the black ring, C0, C1, and C4 represent the ground-truth.

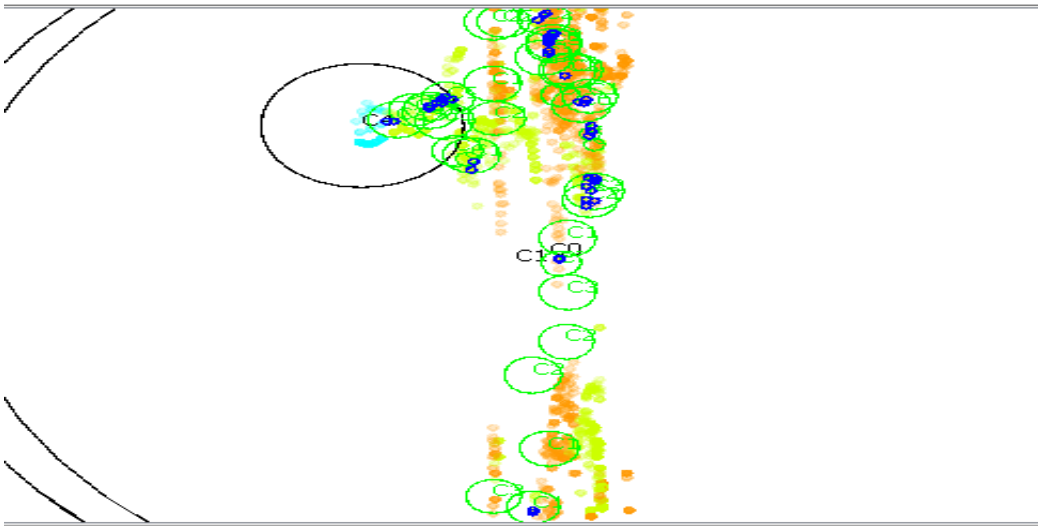


Figure 4-18: The modified DenStream for Forest Covertype dataset. The tiny blue rings are the clustering, the green rings are the micro-clustering, and the black ring is the ground-truth.

The graph plots showing the performance of the algorithms on each metric can be seen in Figure 4-19, Figure 4-20, Figure 4-21, and Figure 4-22. The performance metric CMM results on Forest Covertype dataset with default settings is presented in Figure 4-19. In the figure, *DenStream* average value is less than 0.5 with its highest point at instance 192000 with a value of 0.426978 and lowest point at instance 10000 with a value of 0.367879. *ClusTree* has its lowest point at instance 185000 with a value of 0.582144 and the highest point at instance 31000 with a value of 0.812673. *CluStream* average value is 0.749105. The modified *DenStream* average value is 0.387108.

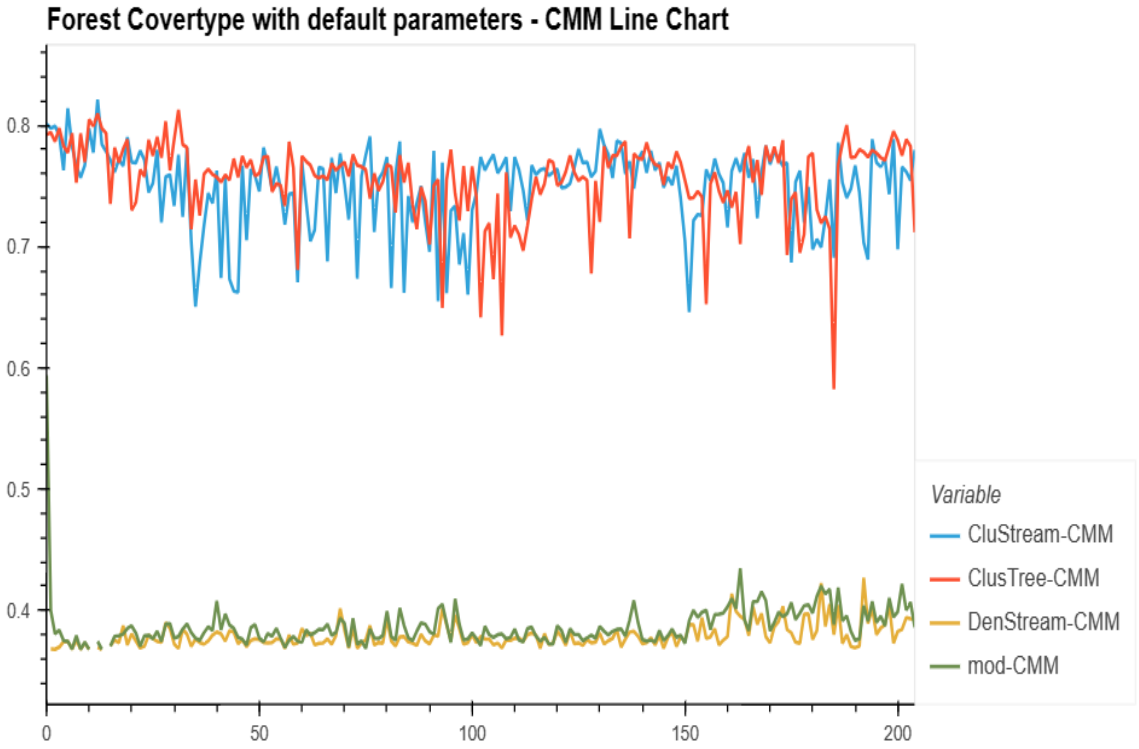


Figure 4-19: The line graph of Forest Covertype using CMM on ClusTree, CluStream, DenStream, and modified DenStream.

In Figure 4-20, the performance metric Purity for the Forest Covertype dataset with the default parameters of the algorithms is presented. *DenStream* has the highest mean average with 0.972418 over *CluStream*, *ClusTree*, modified *DenStream* (see Table 4-2). *DenStream* and modified *DenStream* experienced breaks at instances 11000, 12000, 14000, and 15000. All the algorithms have an average value over and above 0.910000.

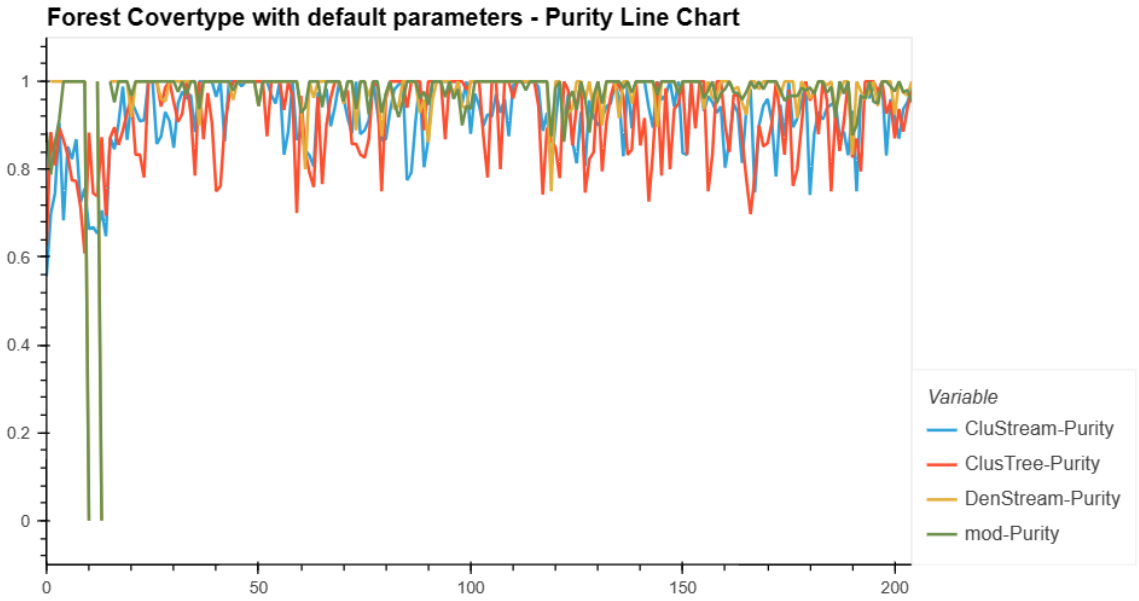


Figure 4-20: The line graph of Forest Covertype using Purity on ClusTree, CluStream, DenStream, and modified DenStream**.**

The performance metric Silhouette Coefficient for Forest Covertype on default parameters for the three algorithms is presented in Figure 4-21. *DenStream* has its highest points of 1.0000 at instances (1000 and 6000) and an average value of 0.754788. The highest point of *ClusTree* with a value occurs at instances 59000, 87000, 90000, 93000, 97000, 102000, 105000, 128000, 155000, and 163000. *ClusTree* average value is 0.829813. *CluStream* average value is 0.801819. The modified *DenStream* sharply dropped in instances 122000 and 125000 and has an average value 0.747071.

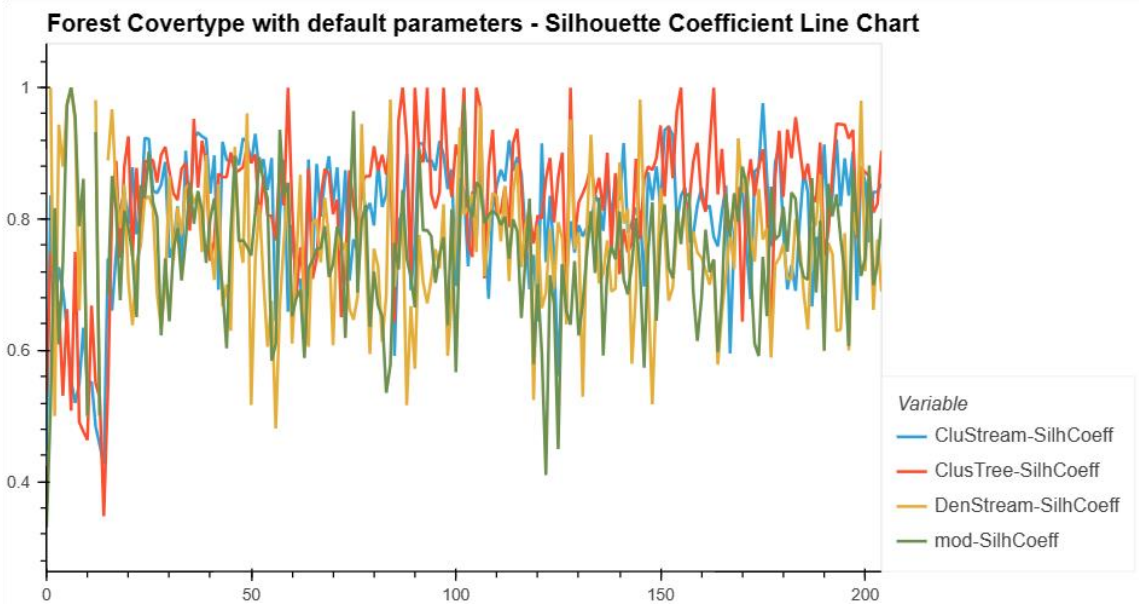


Figure 4-21: The line graph of Forest Covertype using Silhouette Coefficient on ClusTree, CluStream, DenStream, and modified DenStream**.**

In Figure 4-22, the performance metric Rand index for the Forest Covertype dataset with default parameters for algorithms is presented. *ClusTree* highest point is at instance 94000 with a value of 0.783357 and the lowest point is at instance 13000 with a value of 0.202679. *CluStream* has its highest point at instance 94000 with a value of 0.784346 and lowest point at instance 13000 with a value of 0.203417. The highest point of *DenStream* is at instance 92000 with a value of 0.811588 and the lowest point is at instance 13000 with a value of 0.161694. The modified *DenStream* experienced its highest point at instance 94000 and lowest points at instances 6000 and 13000. The modified *DenStream* average value is 0.579084.

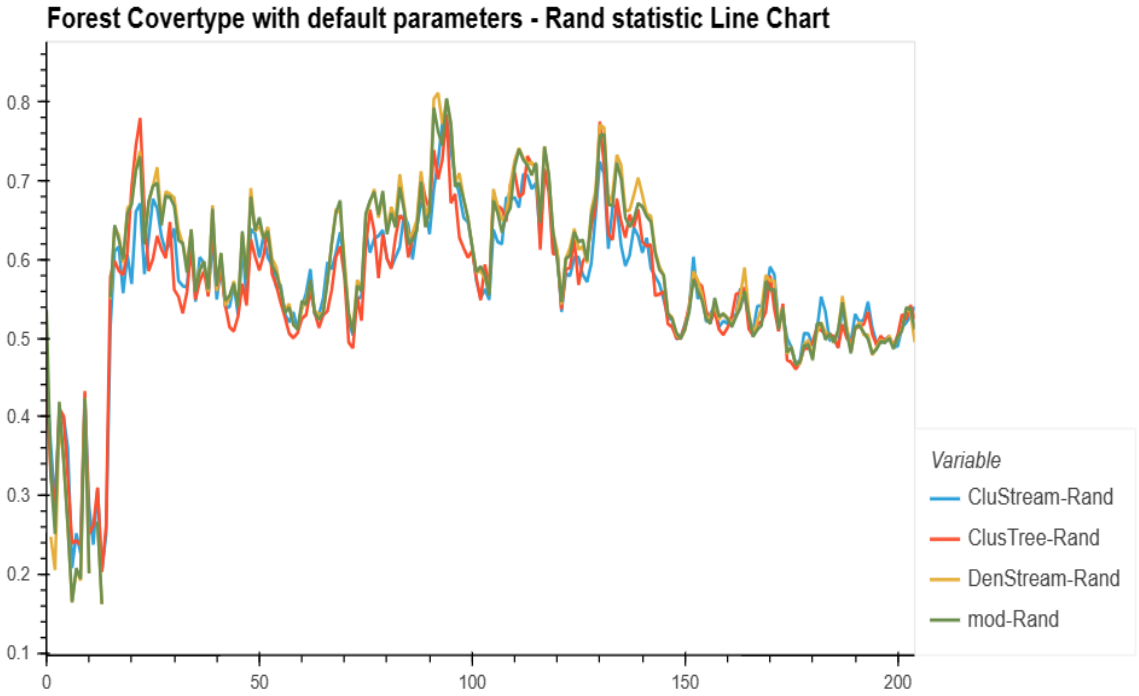


Figure 4-22: The line graph of Forest Covertype using Rand index on ClusTree, CluStream, DenStream, and modified DenStream.

Table 4-2 illustrates evaluation measure using the Forest Covertype with default setting. The results show that *CluStream* outperforms all other algorithms on metrics CMM with a mean value 0.749105. *ClusTree* outperforms other algorithms on metric Silhouette coefficient with a mean value 0.829813. *DenStream* outperforms other algorithms on performance metrics (Purity and Rand index) with 0.972418 and 0.582461. However, the modified *DenStream* outperforms *DenStream* on performance metric CMM with 0.387108.

Table 4-2: Forest Covertype dataset with default settings on the algorithms.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	**0.749105**	0.564100	0.379326	0.387108
Purity	0.918534	0.911704	**0.972418**	0.971124
Silhouette	0.801819	**0.829813**	0.754788	0.747071
Rand index	0.564100	0.562055	**0.582461**	0.579084

Figure 4-23 illustrates the visualization of the bar chart for the algorithms on Forest Covertype dataset with the performance metrics CMM, Purity, Silhouette Coefficient, and Rand index with both *DenStream* and modified *DenStream* outperforming *CluStream* and *ClusTree* on metrics Purity and Rand index.

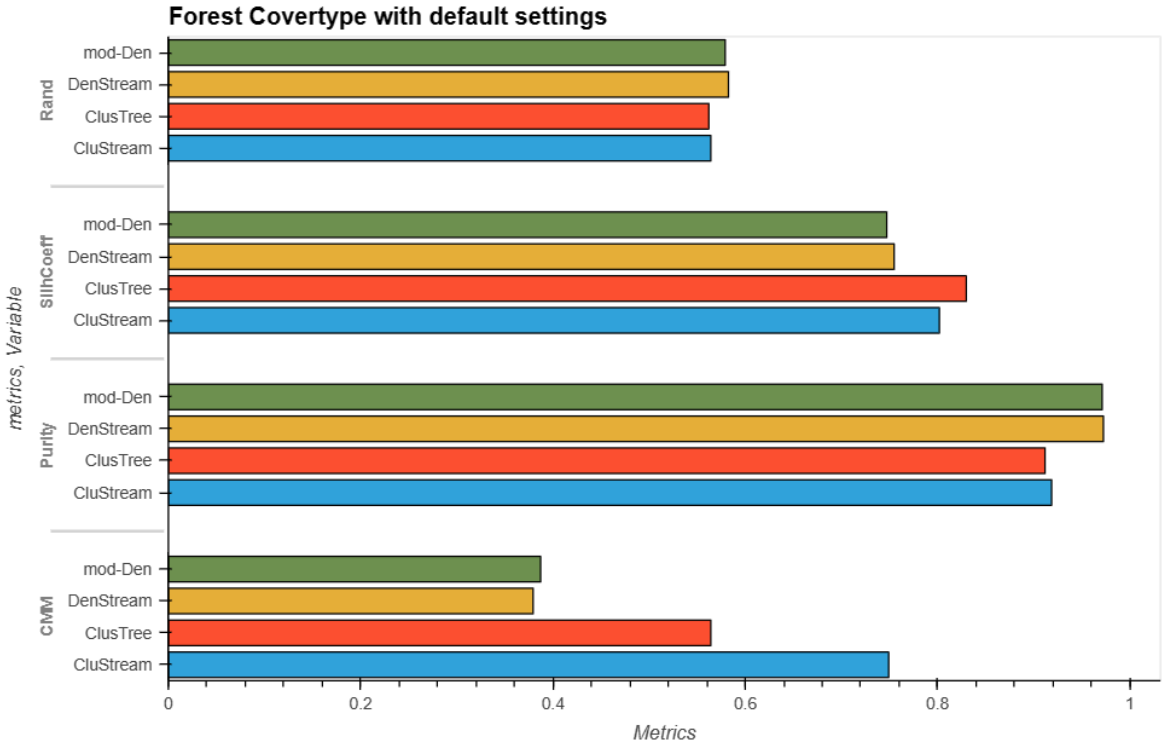


Figure 4-23: Barchart showing ClusTree, CluStream, DenStream, and modified DenStream on Forest Covertype dataset.

4.2.3 Electricity with default settings

The stream settings for the Electricity dataset start with 5000 instances initially and then run to 45000 instances for the analysis. The stream outputs for the *CluStream*, *ClusTree*, DenStream and the modified *DenStream* are indicated in Figure 4-24, Figure 4-25, Figure 4-26, and Figure 4-27 respectively.

Figure 4-24 displays the output of *CluStream* on Electricity dataset after 45000 instances with the stream points displayed in deep colours, ground-truth comes as a black/gray-coloured circles, micro-clustering in green contours, and clustering in red rings because it was selected as Algorithm1 in MOA.

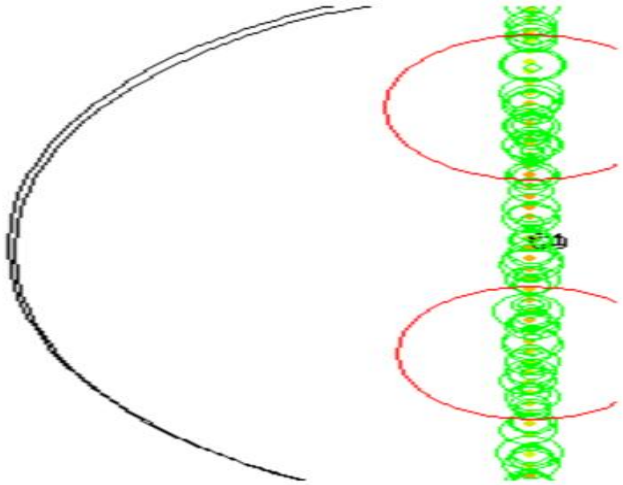


Figure 4-24: CluStream for Electricity dataset after 45000 instances.

Figure 4-25 shows the output of *ClusTree* on Electricity dataset after 45000 instances with the stream points displayed in deep colours, ground-truth comes as a black/gray-coloured circles, micro-clustering in green contours, and clustering in blue rings because it was selected as Algorithm2 in MOA.

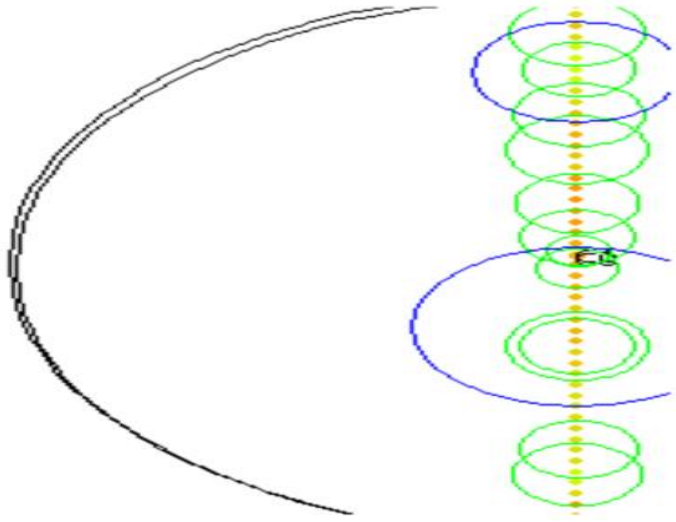


Figure 4-25: ClusTree for Electricity dataset after 45000 instances

Figure 4-26 displays the output of *DenStream* on Electricity dataset after 45000 instances with the stream points displayed in deep colours, ground-truth comes as a black/gray-coloured circles, micro-clustering in green contours, and clustering tiny blue rings because it was selected as Algorithm2 in MOA.

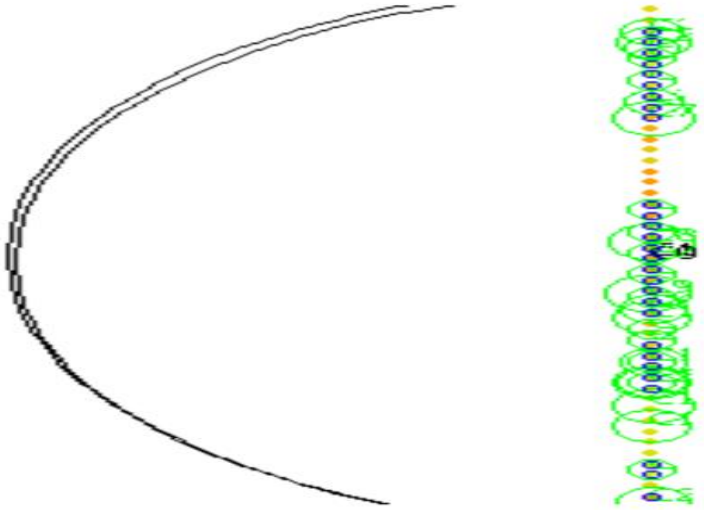


Figure 4-26: DenStream for Electricity dataset after 45000 instances.

Figure 4-27 shows the output of the modified *DenStream* on Electricity dataset after 45000 instances with the stream points displayed in deep colours, ground-truth comes as a black/gray-coloured circles, micro-clustering in green contours, and clustering tiny blue rings because it was selected as Algorithm2 in MOA.

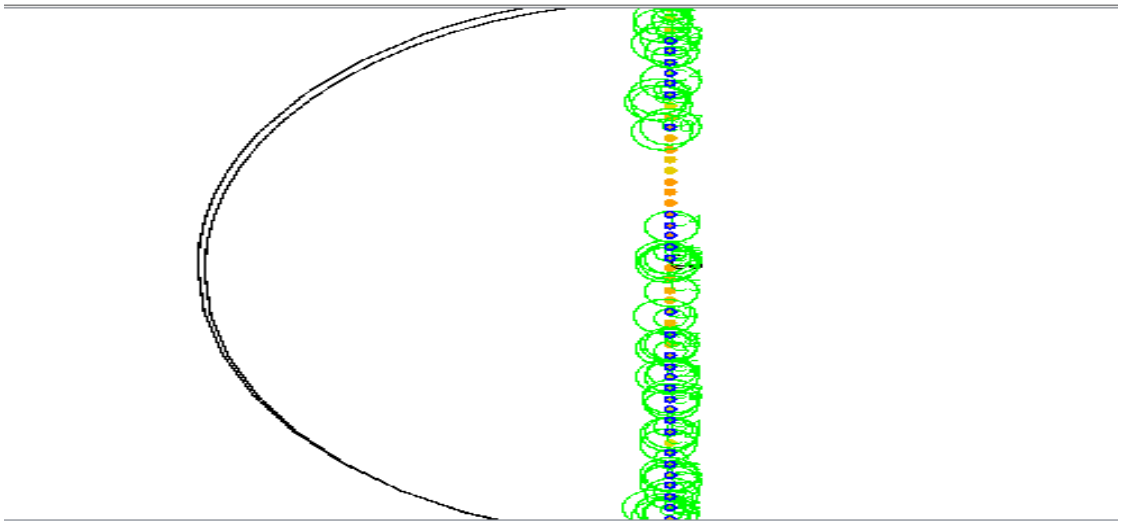


Figure 4-27: The modified DenStream for Electricity dataset after 45000 instances

The line graph charts for the performance metrics CMM, Purity, Silhouette Coefficient, and Rand index for *CluStream*, *ClusTree*, *DenStream* and modified *DenStream* are presented in Figure 4-28, Figure 4-29, Figure 4-30, and Figure 4-31 respectively. In Figure 4-28, the modified *DenStream* was on an upward trajectory on metric CMM until it drops at instance 4000 and continues in this way until the end. The highest point experienced by the modified *DenStream* is between instance zero and 1000 and has an average value of 0.544408. *CluStream* and *ClusTree* maintained values between 0.65 and 0.85, Their highest points are at instance 29000 with values 0.830561 and 0.840458 respectively.

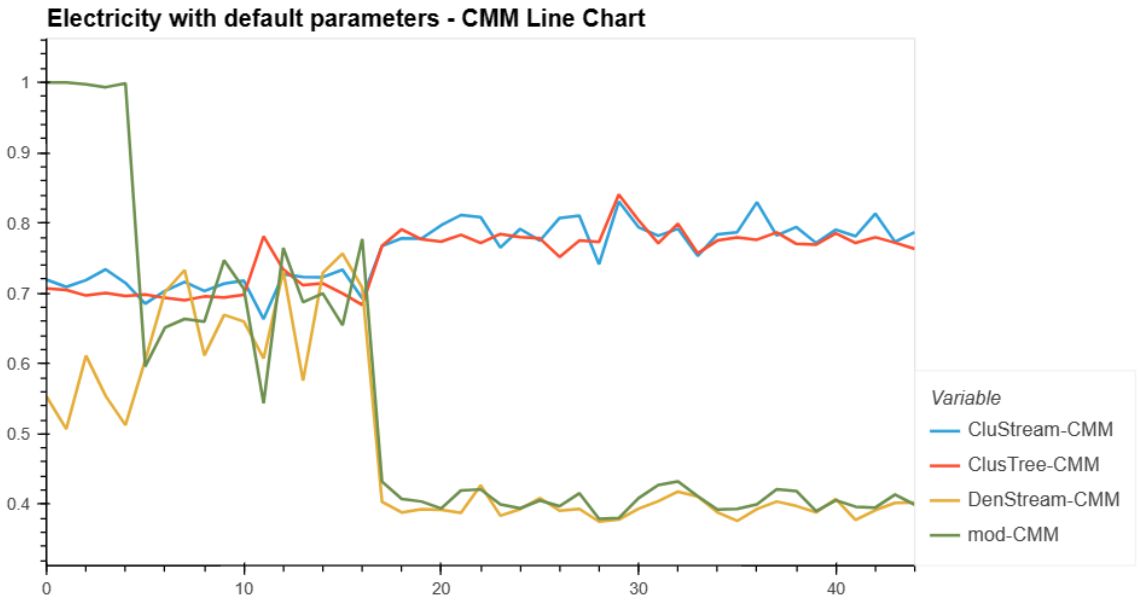


Figure 4-28: The line graph of Electricity using performance metric CMM on ClusTree, CluStream, DenStream, and modified DenStream.

The performance metric Purity for the Electricity dataset using the default settings of the algorithms is presented in Figure 4-29. *CluStream* attains its peak point at instance 13000 and has an average value of 0.776815. *ClusTree's* highest point is at instance 21000 and has an average value of 0.703810. The highest point of *DenStream* occurs at instance 19000 and ends with an average value of 0.897288. The modified *DenStream's* highest point is at instance 25000 and has an average value of 0.874840.

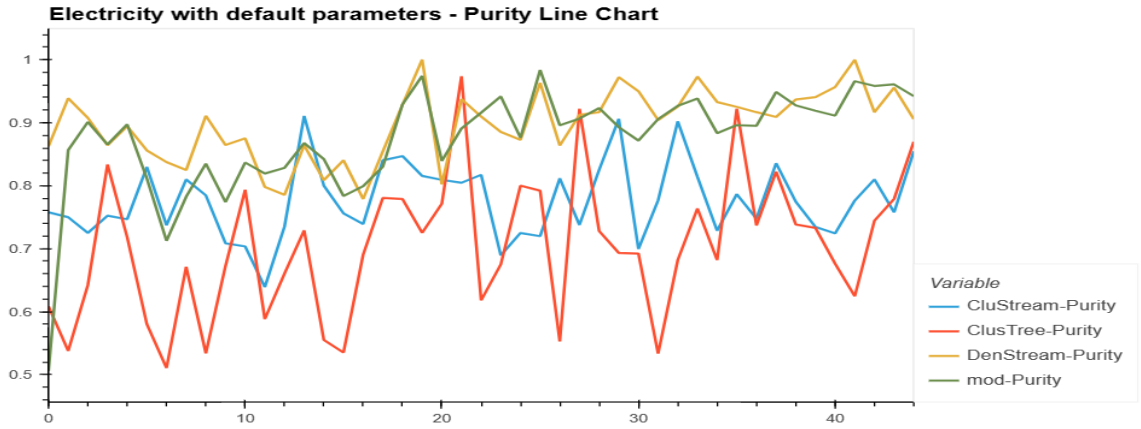


Figure 4-29: The line graph of Electricity using performance metric Purity on ClusTree, CluStream, DenStream, and modified DenStream.

Figure 4-30 presents the performance metric Silhouette Coefficient on Electricity dataset. The line graph shows that *DenStream* raises from instance 15000 to its highest point at instance 19000. The modified *DenStream* dropped from its highest point instance zero to its lowest point at instance 1000. *ClusTree's* highest peak is at instances 26000, 31000, and 36000 respectively. *CluStream* experienced its highest point at instance 5000.

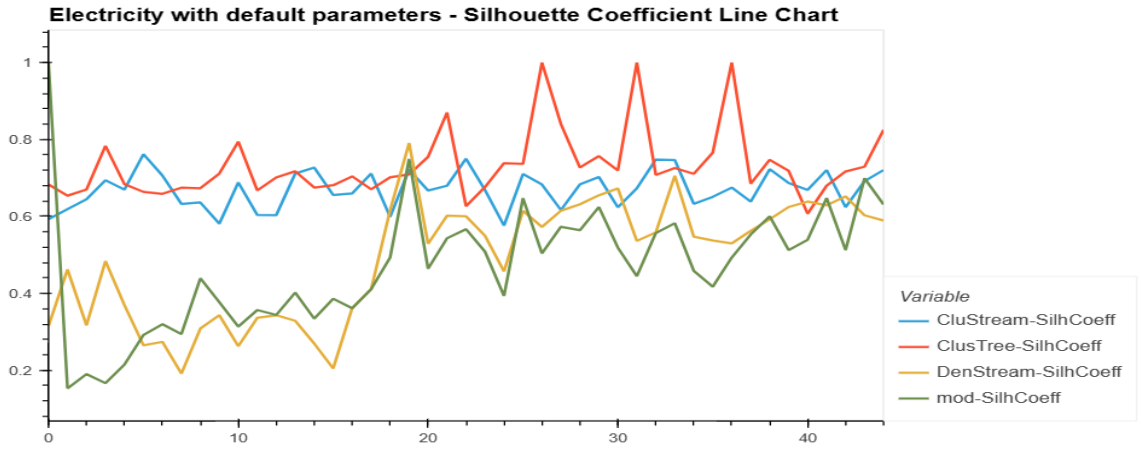


Figure 4-30: The line graph of Electricity using performance metric Silhouette Coefficient on ClusTree, CluStream, DenStream, and modified DenStream.

Figure 4-31 illustrates the performance metric Rand index for the Electricity dataset using the default parameter settings. The highest point of *CluStream* is obtained at instance 1000 and its lowest point at instance 22000. *ClusTree* attains its highest peak point at instance 1000 and least point at instance 44000. *DenStream's* highest point is at instance 35000 and lowest point at instance 4000. The modified *DenStream's* highest point occurs at instance 21000 and lowest point at instance 16000.

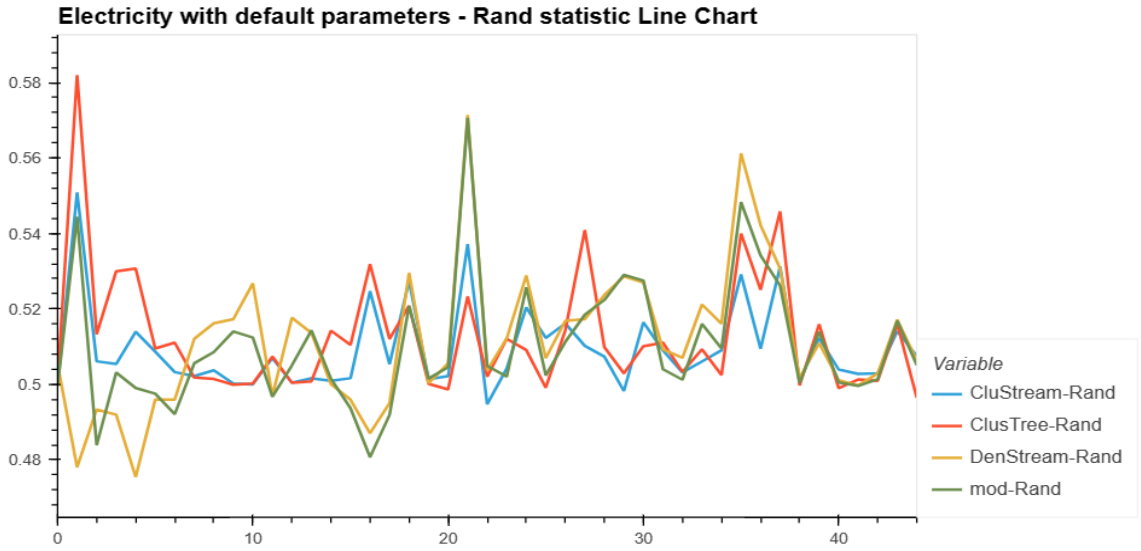


Figure 4-31: The line graph of Electricity using performance metric Rand index on ClusTree, CluStream, DenStream, and modified DenStream.

The average evaluation metrics for the default parameter settings of the algorithms on Electricity dataset is presented in Table 4-3. In summary, the average points show that *CluStream* outperforms other algorithms on performance metric CMM with a mean value of 0.759476. *ClusTree* outperforms on metrics Silhouette Coefficient and Rand index with average values 0.732103, and 0.512624 respectively. *DenStream* also outperforms other algorithms on performance metric Purity with 0.897288. However, the modified *DenStream* outperforms *DenStream* on performance metric CMM with a mean value 0.544408.

Table 4-3: Electricity dataset with default settings on the algorithms.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	**0.759476**	0.751279	0.486608	0.544408
Purity	0.776815	0.703810	**0.897288**	0.874840
Silhouette	0.670710	**0.732103**	0.490165	0.469995
Rand index	0.509466	**0.512624**	0.511428	0.510279

Figure 4-32 shows the visualization of the bar chart for the average values of the algorithms on Electricity dataset with the performance metrics CMM, Purity, Silhouette Coefficient, and Rand index.

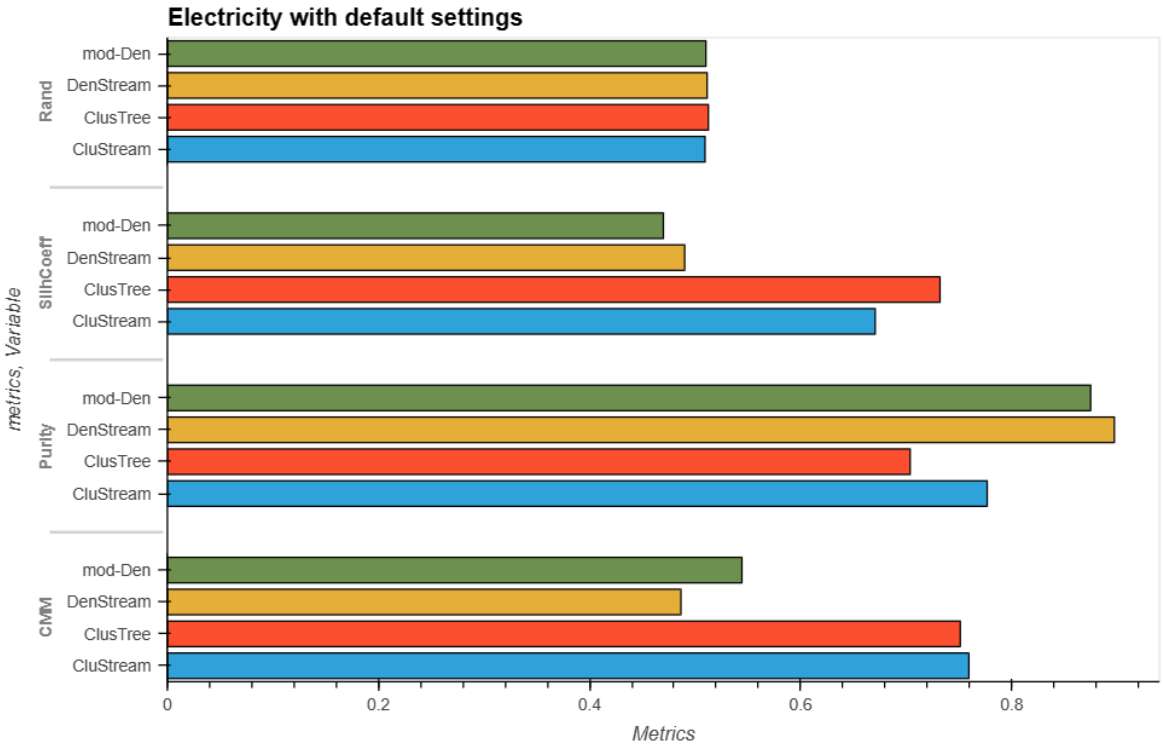


Figure 4-32: Bar plots of ClusTree, CluStream, DenStream, and modified DenStream on Electricity dataset

4.2.4 Effects of Epsilon parameter tuning on Synthetic dataset.

In this section, the effects of *DenStream* and the modified *DenStream* epsilon 0.03 and 0.05 are demonstrated against *CluStream* and *ClusTree algorithms* on *RandomRBFGenerator* with 0% and 30% noise levels. The line graphs of their performance metrics are presented in Figure 4-33, Figure 4-34, Figure 4-35, and Figure 4-36. The experimental results are presented in Table 4-4.

Figure 4-33 presents the line graph of performance metric CMM on *RandomRBFGenerator* 0% noise level and epsilon parameter set at 0.03 for *DenStream* and modified *DenStream. ClusTree* demonstrates a better performance after the initial value at instance zero and maintains values between the intervals [0.98, 1.00]. *CluStream* preserves a value range between the intervals [0.96, 0.99]. *DenStream's* lowest point is at instance 16000 with a value of 0.583. The modified *DenStream* has its lowest point at instance 170000.

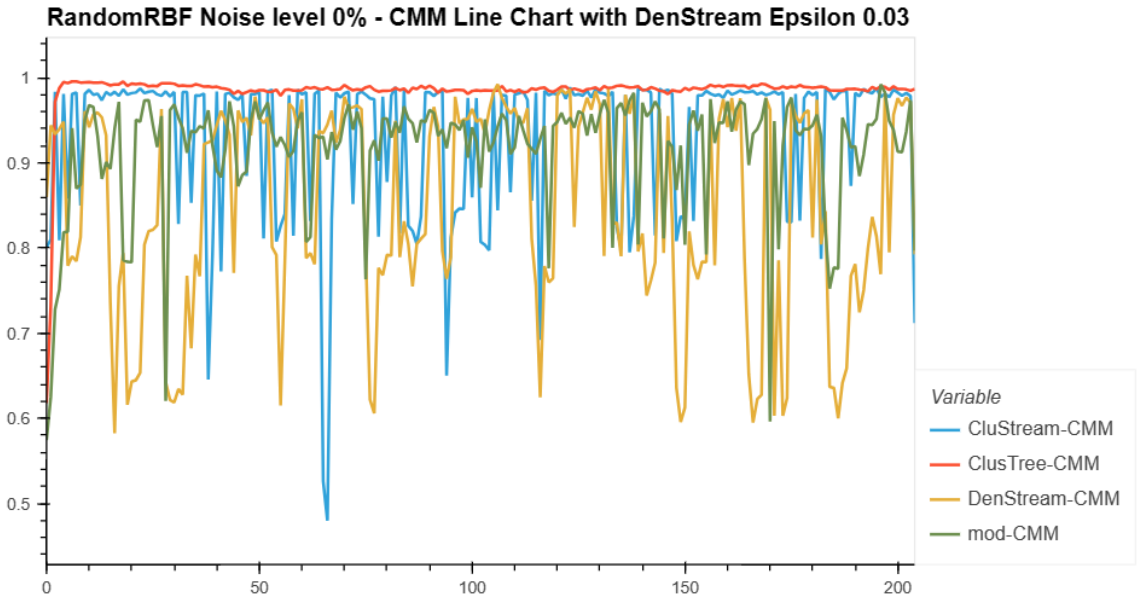


Figure 4-33: The line graph of RandomRBF with 0% noise level using CMM on modified DenStream and DenStream epsilon set at 0.03.

Figure 4-34 is the line graph of performance metric Purity on *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.03. *CluStream* outperforms other algorithms with an average value of 0.984838. *ClusTree* has its lowest point at instance 136000 with a value of 0.868. *DenStream's* lowest point is at instance 133000 with a value of 0.432. *DenStream* also dropped at instances 19000, 31000, 131000, 134000, 135000, 185000, and 188000. The modified *DenStream's* lowest point is at instance 170000.

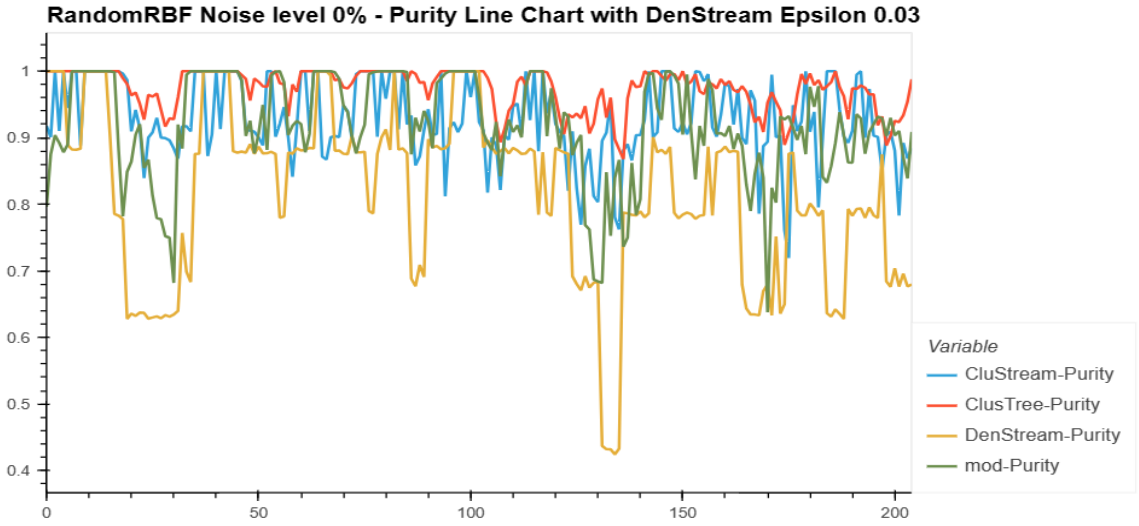


Figure 4-34: The line graph of RandomRBF with 0% noise level using Purity on modified DenStream and DenStream epsilon set at 0.03.

Figure 4-35 is the output of performance metric Silhouette Coefficient on *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.03. *CluStream* outperforms with an average value of 0.821436. *ClusTree* has the lowest average value of 0.565896. The modified *DenStream's* lowest point is at instance 167000.

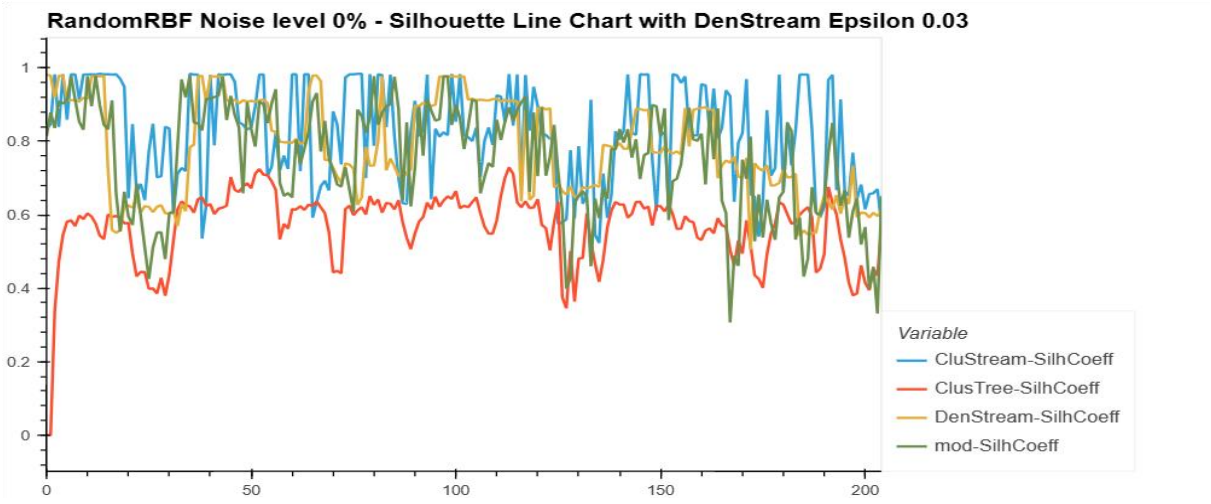


Figure 4-35: The line graph of RandomRBF with noise level 0% using Silhouette Coefficient on modified DenStream and DenStream epsilon set at 0.03.

Figure 4-36 is the visualized line graph of performance metric Rand index on *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.03. The modified *DenStream* has its lowest point at instances 28000 and 170000. The modified *DenStream* outperforms other algorithms with an average value of 0.889164. *DenStream* has the lowest average value of 0.823520.

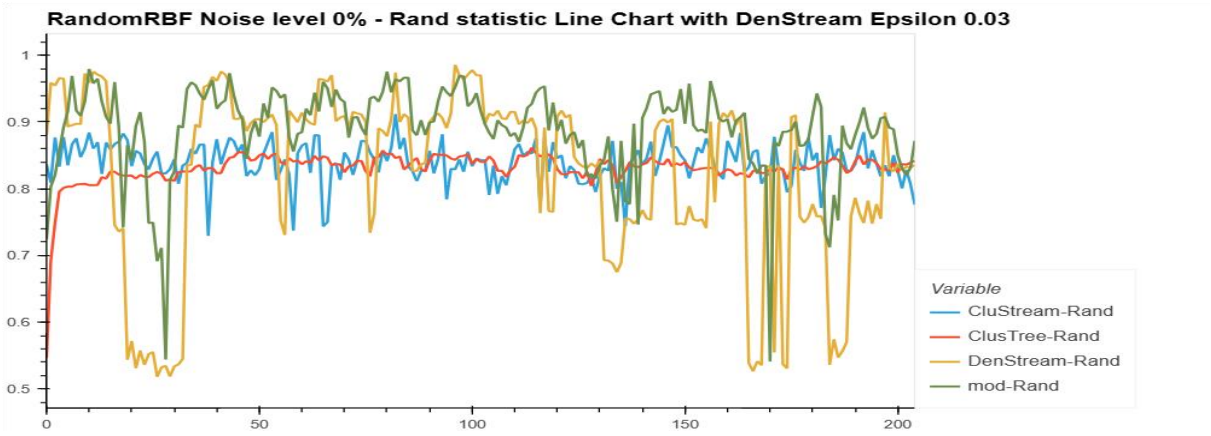


Figure 4-36: The line graph of RandomRBF with noise level 0% using Rand index on modified DenStream and DenStream epsilon set at 0.03.

In Table 4-4, *ClusTree* outperforms on performance metrics CMM and Purity with 0.984734 and 0.974619 respectively. *CluStream* outperforms in terms of performance metric Silhouette Coefficient with a value of 0.821436. The modified *DenStream* outperforms on metric Rand index with an average value 0.889164.

Table 4-4: RandomRBF 0% noise level with epsilon parameter set at 0.03.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.933418	**0.984734**	0.848488	0.917020
Purity	0.927379	**0.974619**	0.815577	0.918202
Silhouette	**0.821436**	0.565896	0.782959	0.751973
Rand index	0.839968	0.830699	0.823520	**0.889164**

Figure 4-37 presents the visualization of the bar chart for the average values of the algorithms on *RandomRBFGenerator* with 0% noise level on performance metrics CMM, Purity, Silhouette Coefficient, and Rand index. The modified *DenStream* outperformed *DenStream* on metrics CMM, Purity and Rand index.

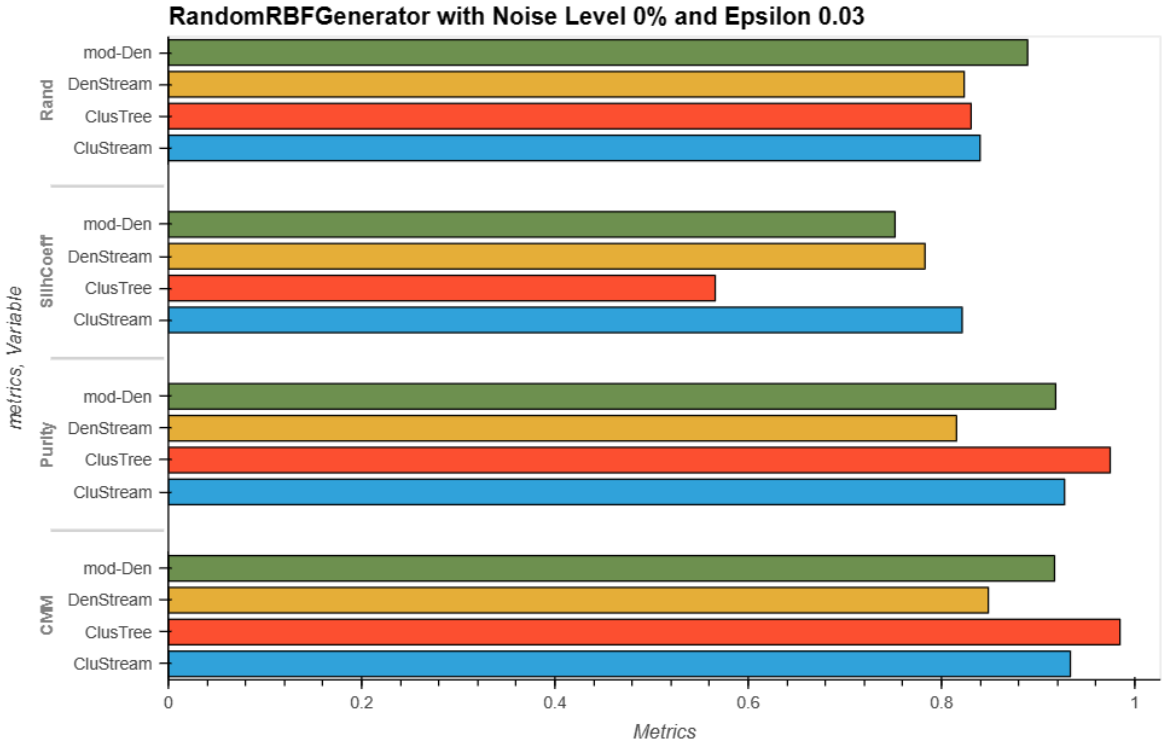


Figure 4-37: Bar plots of RandomRBF with noise level 0% on ClusTree, CluStream, DenStream, and modified DenStream.

Figure 4-38 is the line graph of performance metric CMM on *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *DenStream* outperforms both *CluStream* and *ClusTree*. *DenStream's* highest and lowest points are at instances 46000 and 174000 with values 0.590 and 0.918 respectively. On the average, *DenStream* outperforms both *ClusTree* and *CluStream* with a value of 0.825596. *ClusTree's* lowest point is at instance zero with a value of 0.603 and its highest point at instance 46000 with a value of 0.809. *CluStream's* values are in between the intervals [0.7, 0.8].

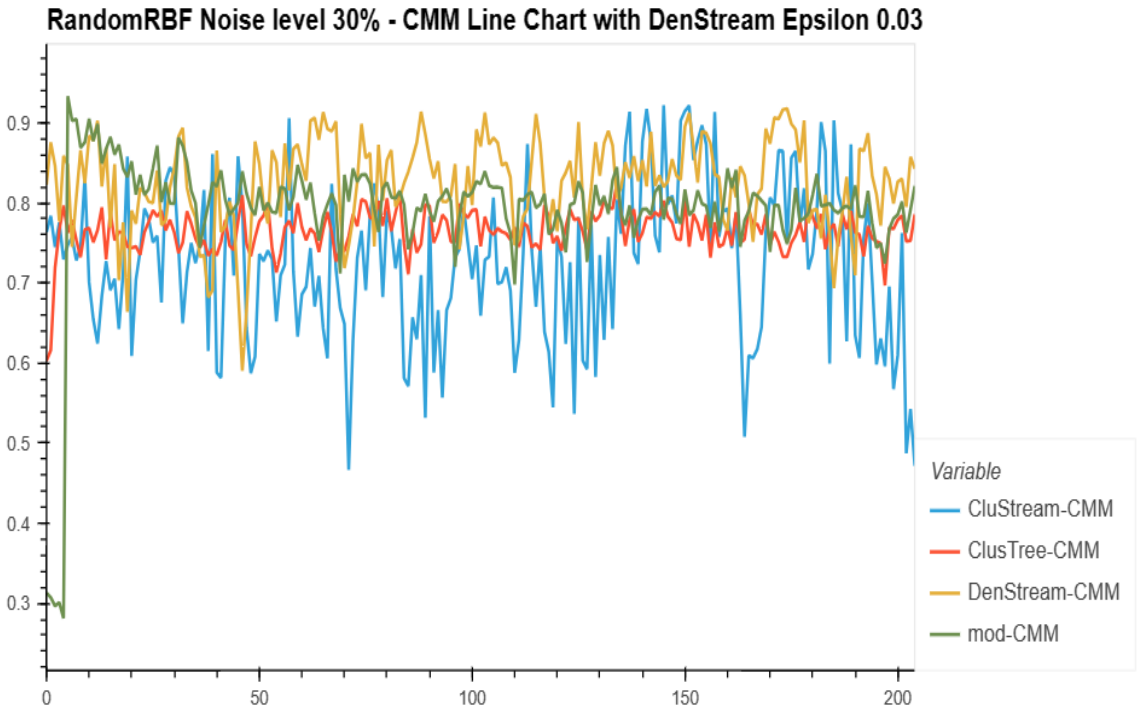


Figure 4-38: The line graph of RandomRBF with noise level 30% using CMM on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-39 is the visualized line graph of performance metric Purity on *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *CluStream* outperforms both *DenStream* and *ClusTree* with an average value of 0.980379. The lowest point of *DenStream* occurs at instance 43000 with a value of 0.704. The lowest point of *ClusTree* occurs at instance 33000 with a value of 0.926.

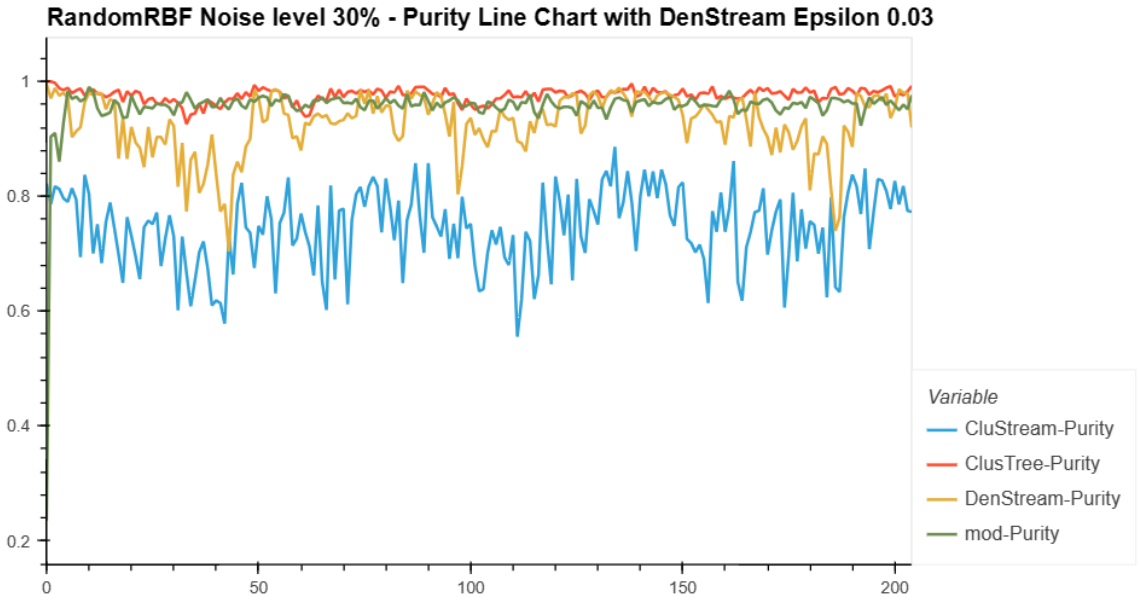


Figure 4-39: The line graph of RandomRBF with noise level 30% using Purity on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-40 is the visualized line graph of performance metric Silhouette Coefficient on *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *DenStream* performs better than both *CluStream* and *ClusTree* with an average value of 0.747127. *CluStream's* highest point occurs at instance 2000 with a value of 0.650 and lowest point at instance 175000 with value of 0.240. *ClusTree's* highest point occurs at instance 134000 with value of 0.653.

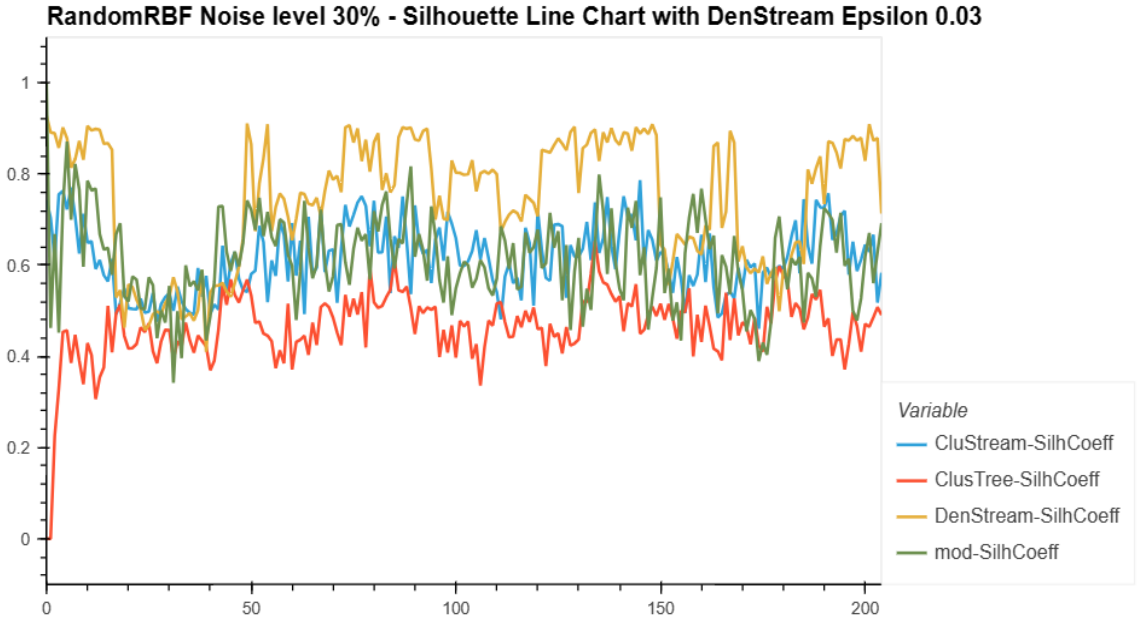


Figure 4-40: The line graph of RandomRBF with noise level 30% using Silhouette Coefficient on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-41 is the visualized line graph of performance metric Rand index on *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.03. *ClusTree* outperforms both *CluStream* and *DenStream* on average with a value of 0.841067. *DenStream's* lowest point occurs at instance 38000 with a value of 0.597 and highest point at instance 88000 with a value of 0.878. The lowest point of *CluStream* occurs at instance 153000 with a value of 0.750.

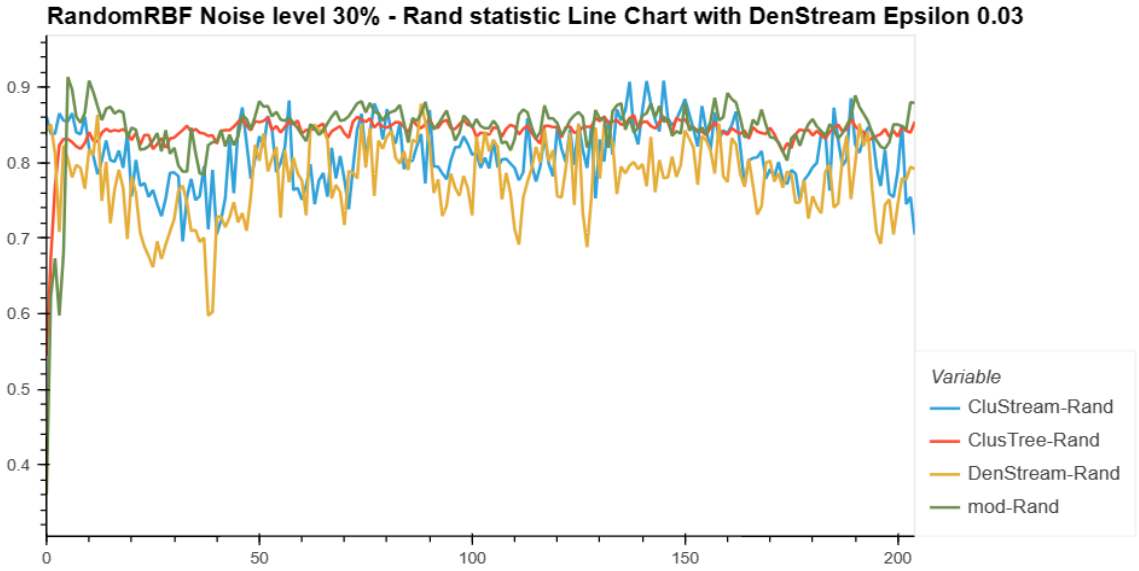


Figure 4-41: The line graph of RandomRBF with noise level 30% using Rand index on DenStream and modified DenStream epsilon set at 0.03.

Table 4-5 summarizes the evaluation performance of the algorithms. The table shows that *DenStream* performs better than other algorithms on metrics CMM and Silhouette Coefficient with 0.825596 and 0.747127 respectively. The modified *DenStream* outperforms on metric Rand index with 0.844152. *ClusTree* outshine on metric Purity with an average value of 0.976311. The modified *DenStream* also shows a better performance against *DenStream* on metric Purity.

Table 4-5: RandomRBF 30% noise level with epsilon parameter set at 0.03.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.725730	0.764806	**0.825596**	0.792180
Purity	0.744893	**0.976311**	0.927619	0.956226
Silhouette	0.615896	0.464856	**0.747127**	0.612858
Rand index	0.811282	0.841067	0.778500	**0.844152**

Figure 4-42 presents the visualization of the bar chart for the average values of the algorithms *ClusTree, CluStream,* and *DenStream* on *RandomRBFGenerator* with 30% noise level and epsilon parameter set at 0.03 on performance metrics CMM, Purity, Silhouette Coefficient, and Rand index. The modified *DenStream* outperformed *DenStream* on performance metrics Purity and Rand index and at least one algorithm on the other metrics CMM and Silhouette Coefficient.

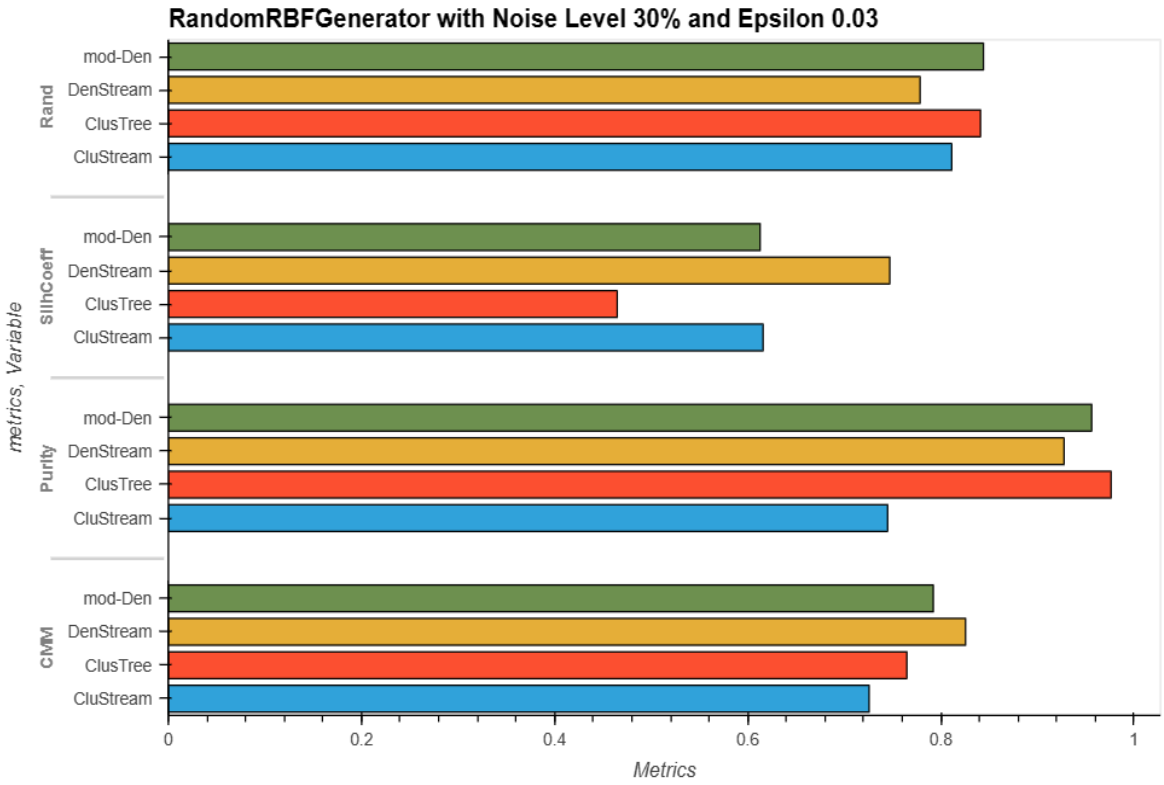


Figure 4-42: Bar plots of CluStream ClusTree, DenStream, and modified DenStream epsilon set at 0.03 on RandomRBF with noise level 30%.

Figure 4-43 is the visualized line graph of performance metric CMM on *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.05. *DenStream's* lowest point occurs at instance 173000 with a value of 0.410 and highest point at instance 46000 with a value of 1.00. *CluStream* maintains uniform value within the intervals [0.97, 0.99]. However, *ClusTree* outperforms both *CluStream* and *DenStream* after the initial poor start with an average value of 0.984734.

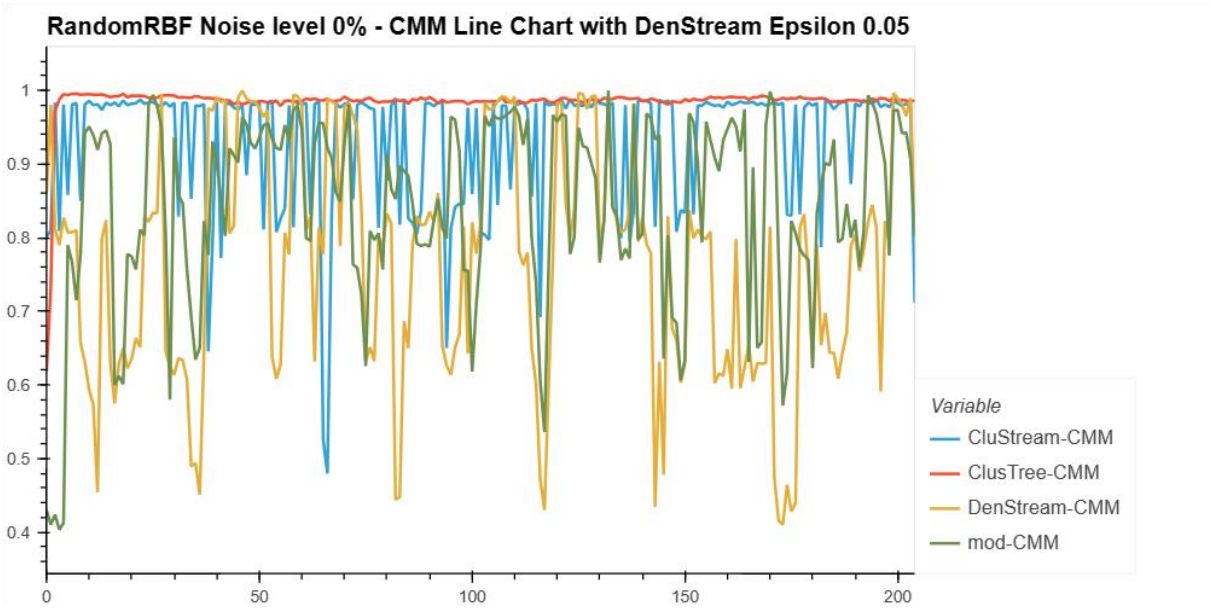


Figure 4-43: The line graph of RandomRBF with noise level 0% using CMM on modified DenStream epsilon set at 0.05.

Figure 4-44 is the line graph of performance metric Purity on *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.05. *DenStream's* highest point value is 1.00 which occurs at instances 38000 to 42000 and lowest point value 0.209 at instance 175000. *CluStream* outperforms both *ClusTree* and *DenStream* with an average value of 0.984838. *ClusTree's* lowest point occurs at instance 136000 with a value of 0.868 and highest point value is 1.00.

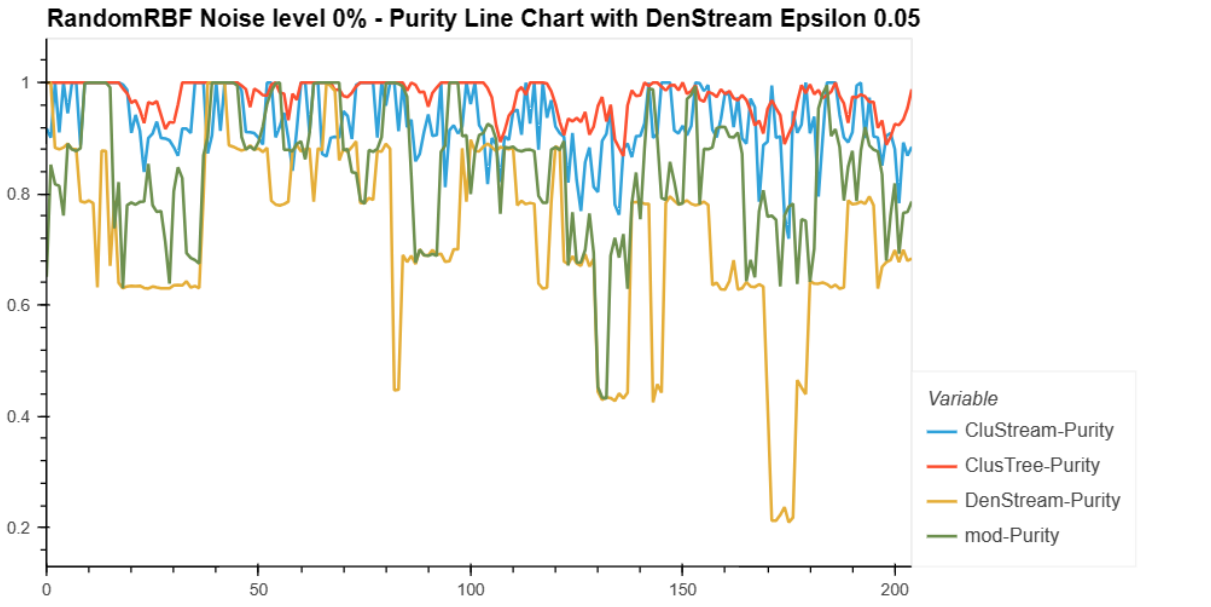


Figure 4-44: The line graph of RandomRBF with noise level 0% using Purity on modified DenStream epsilon set at 0.05.

Figure 4-45 illustrates the output of performance metric Silhouette Coefficient on *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.05. *DenStream* outperforms both *ClusTree* and *CluStream* with 0.742449. *DenStream's* highest point value is 1.00 at instance intervals [171000 – 176000]. *CluStream's* lowest point value is 0.436 at instance 174000. *ClusTree's* highest point value is 0.729 at instance 113000 and lowest point value occurs at instance zero.

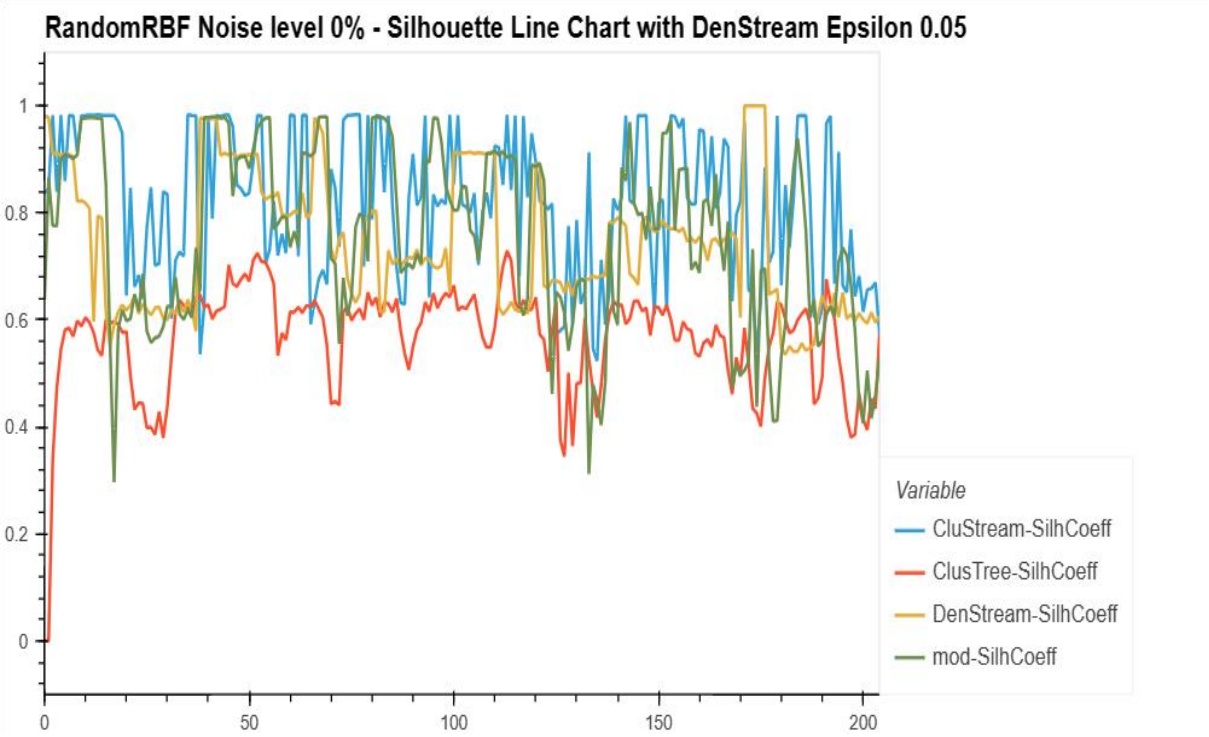


Figure 4-45: The line graph of RandomRBF with noise level 0% using Silhouette Coefficient on modified DenStream epsilon set at 0.05.

Figure 4-46 is the line graph of performance metric Rand index on *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.05. *DenStream's* lowest point is at instance 175000 with a value of 0.207. *ClusTree* outperforms both *DenStream* and *ClusTree* with an average value of 0.830699. *CluStream* value ranges between the intervals 0.7 and 0.8.

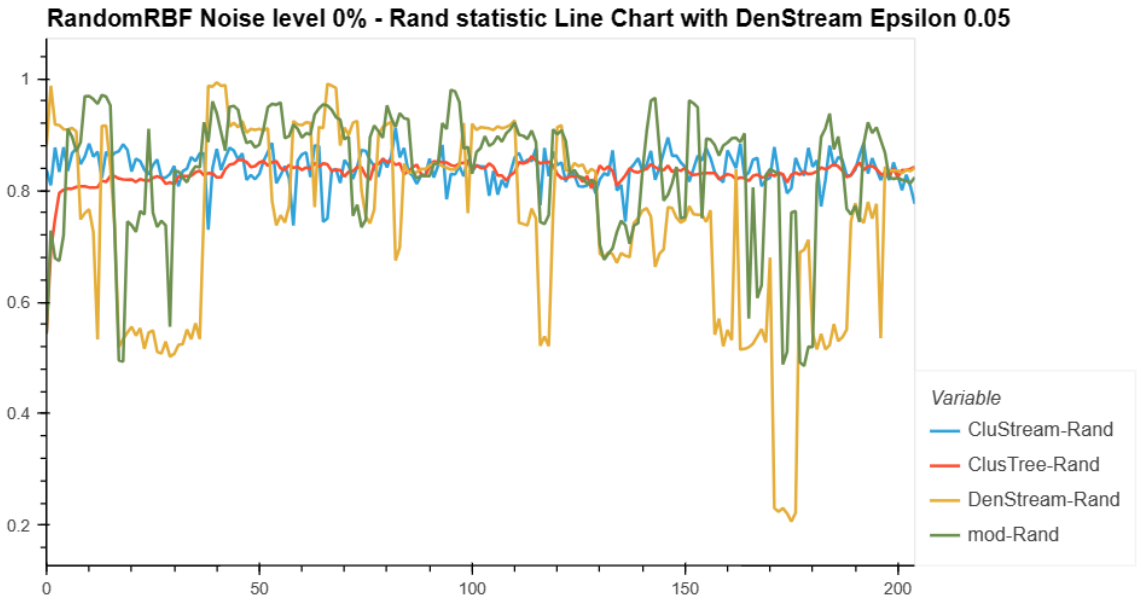


Figure 4-46: The line graph of RandomRBF with noise level 0% using Rand index on modified DenStream epsilon set at 0.05.

Table 4-6 indicates *ClusTree* outperforms on performance metrics CMM and Purity with 0.984734 and 0.974619 respectively. *CluStream* outperforms on performance metrics Silhouette Coefficient and Rand index with 0.821436 and 0.839968 respectively. The modified *DenStream* however, outperforms *DenStream* on all the metrics.

Table 4-6: RandomRBF 0% noise level with epsilon parameter set at 0.05.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.933418	**0.984734**	0.767276	0.842120
Purity	0.927379	**0.974619**	0.723268	0.842179
Silhouette	**0.821436**	0.565896	0.742449	0.750881
Rand index	**0.839968**	0.830699	0.745373	0.837461

Figure 4-47 is the visualized bar chart for the average values of the algorithms *DenStream*, *ClusTree*, and *CluStream* on *RandomRBFGenerator* with 0% noise level and epsilon parameter set at 0.05 using performance metrics CMM, Purity, Silhouette Coefficient, and Rand index. The modified *DenStream* outperformed *DenStream* on all the performance metrics.

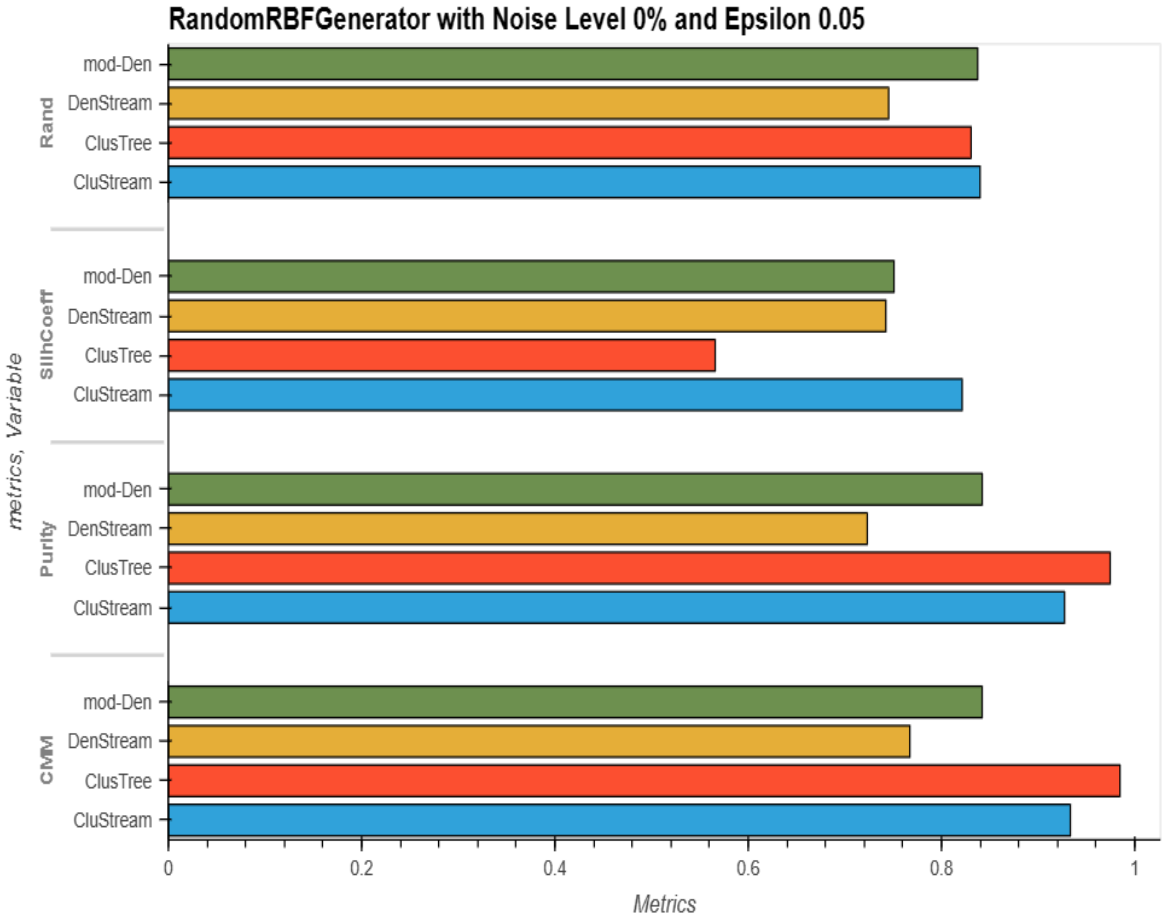


Figure 4-47: Bar plots of ClusTree, CluStream, DenStream, and modified DenStream epsilon set at 0.05 on RandomRB with noise level 0%.

Figure 4-48 is the visualized line graph of performance metric CMM on *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05. *DenStream's* highest point is at instance 173000 and lowest point at instance 74000. *ClusTree* outperforms other algorithms with an average value of 0.764806. *ClusTree's* highest point occurs at instance 46000. The modified *DenStream's* lowest point occurs at instance 4000 and outperforms both *DenStream* and *CluStream* with an average value of 0.731058 and its highest point occurs at instance 145000.

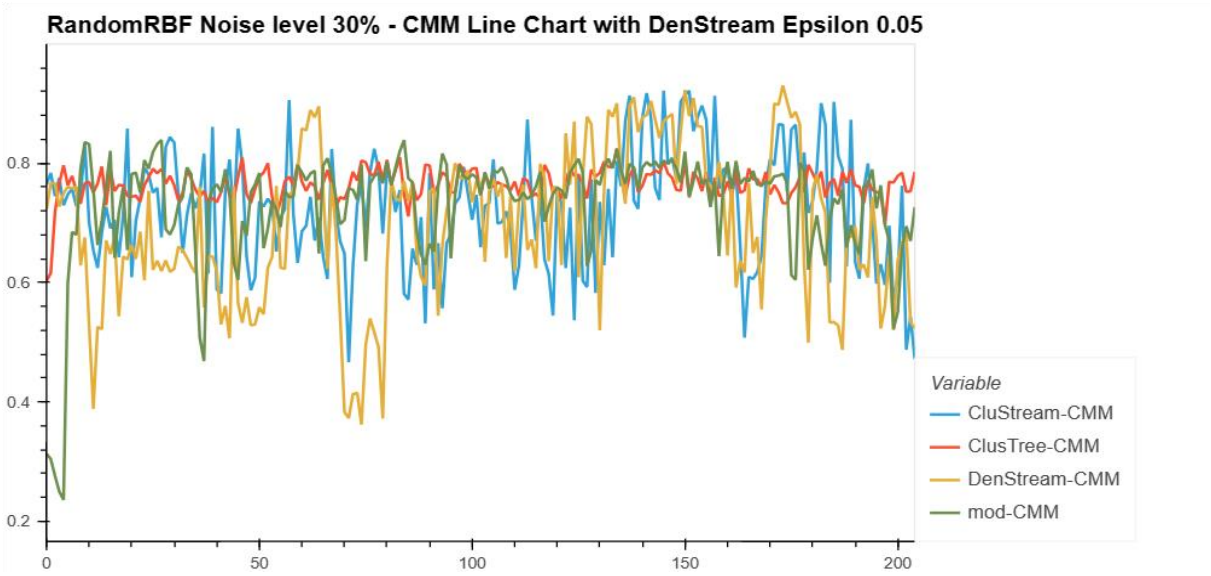


Figure 4-48: The line graph of RandomRBF with noise level 30% using CMM on modified DenStream epsilon set at 0.05.

Figure 4-49 illustrates the line graph of performance metric Purity on *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05. *ClusTree* outperforms other algorithms with an average value of 0.976311. *ClusTree's* lowest point occurs at instance 33000. *CluStream's* highest point occurs at instances 134000 and lowest point at instance 111000. The modified *DenStream* however, outperformed both *CluStream* and *DenStream* with an average value of 0.943678.

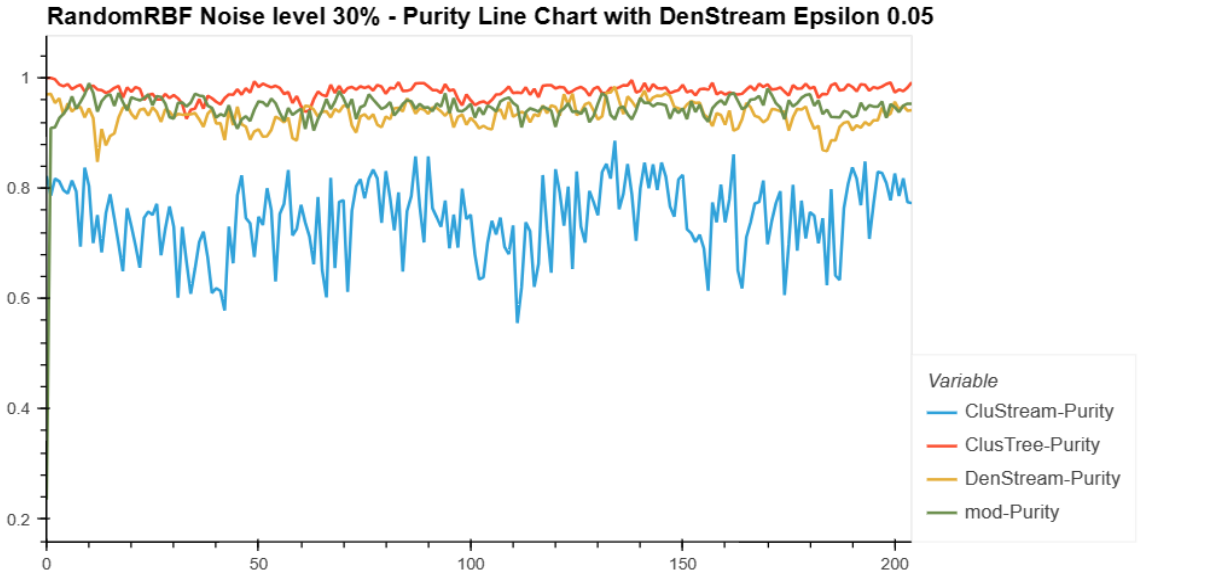


Figure 4-49: The line graph of RandomRBF with noise level 30% using Purity on modified DenStream epsilon set at 0.05.

Figure 4-50 is the visualized line graph of performance metric Silhouette Coefficient on *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set to 0.05. The modified *DenStream* outperforms other algorithms with an average value of 0.644882. *DenStream's* highest point occurs at instance 134000. *CluStream's* lowest point occurs at instance 175000 and has an average value of 0.615896. *ClusTree's* highest point occurs at instance 134000 and has the least average value of 0.464858.

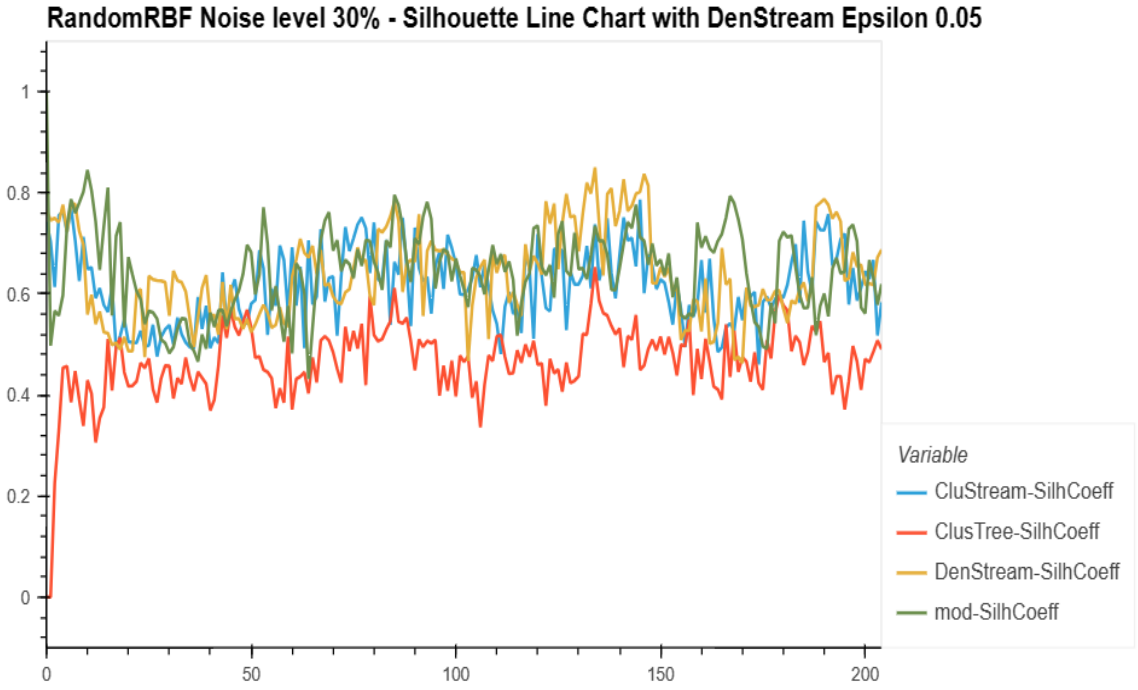


Figure 4-50: The line graph of RandomRBF with noise level 30% using Silhouette Coefficient on modified DenStream epsilon set st 0.05.

Figure 4-51 is the visualized line graph of performance metric Rand index on *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05. *ClusTree* outperforms other algorithms with an average value of 0.841067. *DenStream's* lowest point occurs at instance 27000 and highest point at instance 137000. *CluStream's* lowest point occurs at instance 71000 and highest point at instances 141000 and 145000. The modified *DenStream's* highest point occurs at instance 10000 and lowest point at instance 37000. The modified *DenStream* has an average of 0.830016.

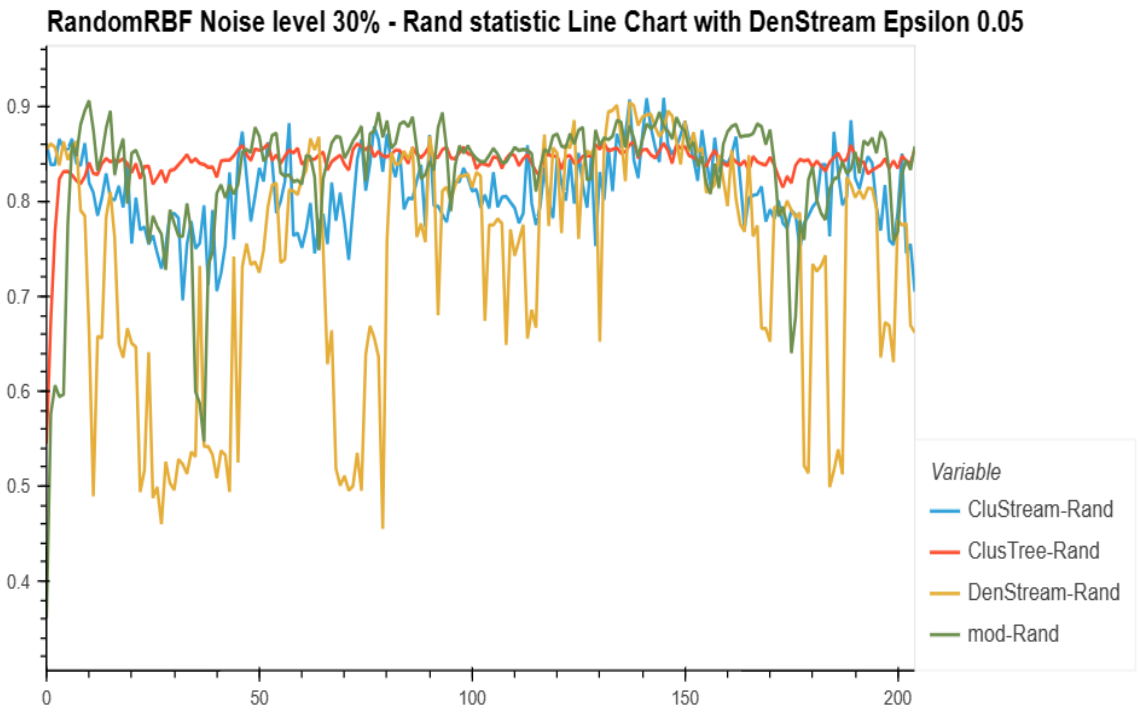


Figure 4-51: The line graph of RandomRBF with noise level 30% using Rand index on modified DenStream epsilon set at 0.05.

In Table 4-7, *ClusTree* outperforms on performance metrics CMM, Purity, and Rand index with 0.764806, 0.976311, and 0.841067 respectively. The modified *DenStream* outperforms on metric Silhouette Coefficient with a value of 0.644882. The modified *DenStream* likewise outperforms against *DenStream* on all metrics.

Table 4-7: RandomRBF 30% noise level with epsilon parameter set at 0.05.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.725730	**0.764806**	0.694887	0.731058
Purity	0.744893	**0.976311**	0.932780	0.943678
Silhouette	0.615896	0.464856	0.639189	**0.644882**
Rand index	0.811282	**0.841067**	0.738834	0.830016

Figure 4-52 presents the visualization of the bar chart for the average values of the algorithms on *RandomRBFGenerator* with 30% noise level and epsilon parameter set to 0.05 on performance metrics CMM, Purity, Silhouette Coefficient, and Rand index. The modified *DenStream* outperformed *DenStream* on all the performance metrics.

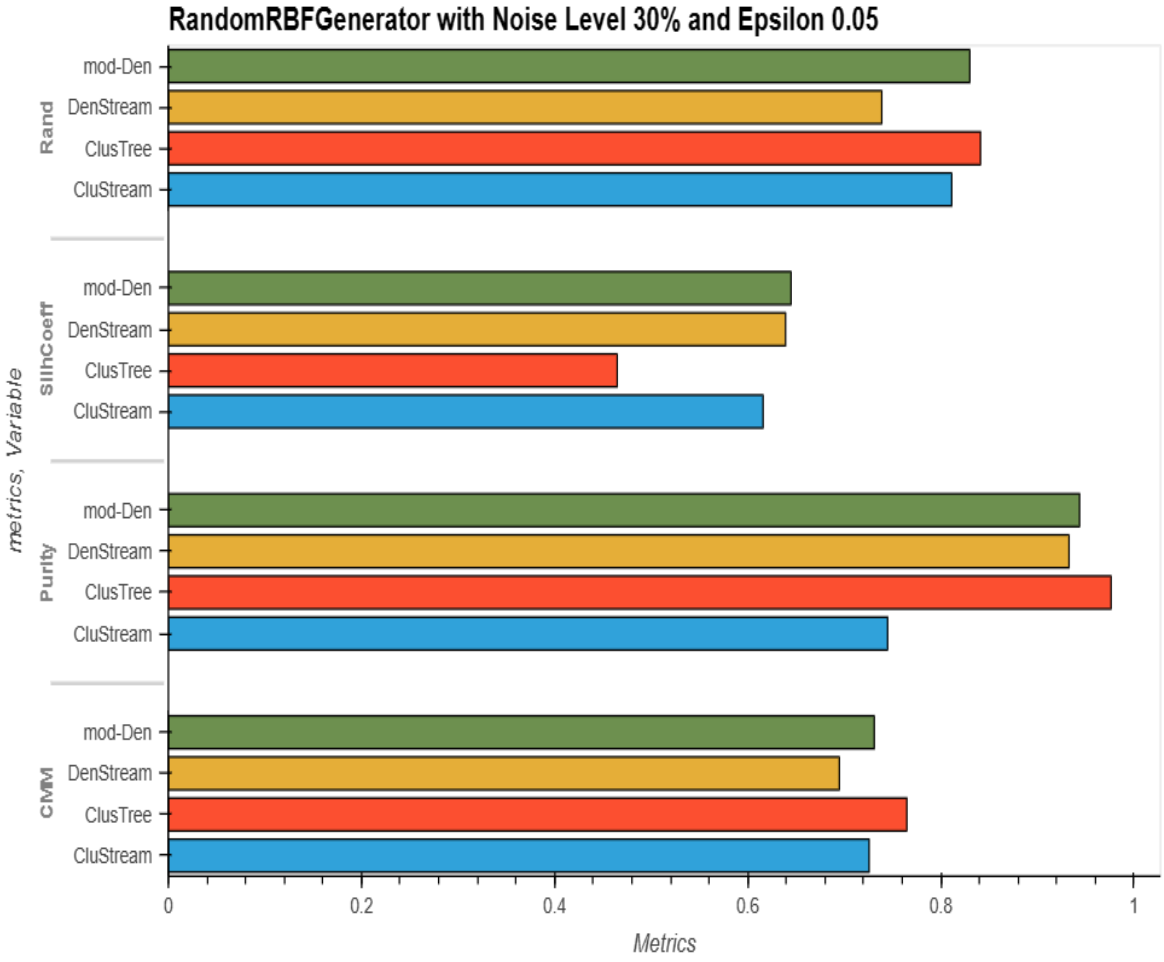


Figure 4-52: Bar plots of ClusTree, CluStream, DenStream, and modified DenStream epsilon set at 0.05 on RandomRBF with noise level 30%.

4.2.6 Effects of Epsilon parameter tuning on Forest Covertype.

We experimented with the effects of *DenStream* and modified *DenStream* epsilon parameter set at 0.03 on real-world dataset Forest Covertype and presented the experimental evaluation. Figure 4-53 illustrates the line graph of performance metric CMM on Forest Covertype, and epsilon parameter of *DenStream* and modified *DenStream* set to 0.03. *ClusTree* outperforms other algorithms with an average value of 0.764913. *ClusTree's* highest point occurs at instance 15000 and lowest point at instance 142000. *CluStream's* highest point happens at instance 12000 and lowest point at instance 35000. *DenStream's* lowest point occurs at instance 191000 and highest point at instance 160000. The modified *DenStream's* highest point occurs at instance 1000 and has an average value of 0.427090.

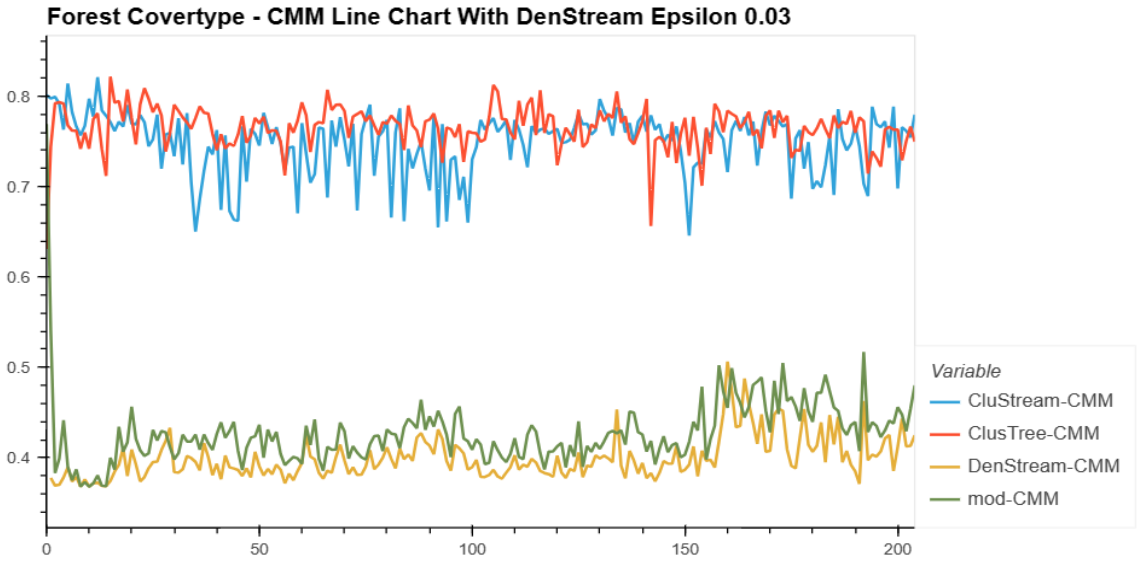


Figure 4-53: The line graph of Forest Covertype using CMM on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-54 is the visualized line graph of performance metric Purity on Forest Covertype, and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *DenStream* lowest point occurs at instance 89000. The modified *DenStream's* lowest point occurs at instance 13000 and has an average value of 0.973233. *ClusTree's* lowest point appears at instance 16000 and outperforms other algorithm with an average value of 0.983300. *CluStream* has the least average value 0.915834.

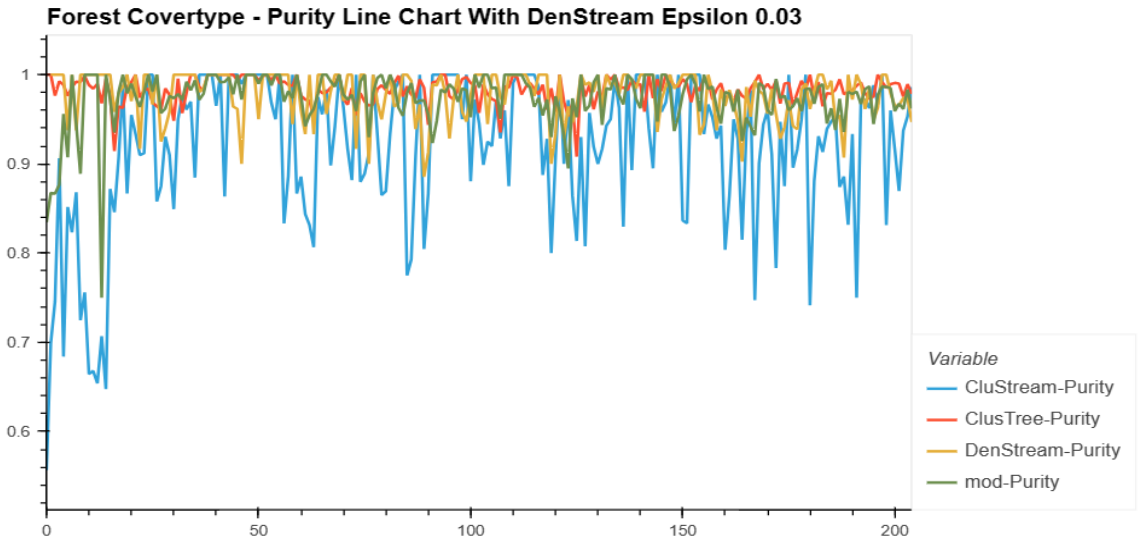


Figure 4-54: The line graph of Forest Covertype using Purity on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-55 is the visualized line graph of performance metric Silhouette Coefficient on Forest Covertype and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *CluStream* outperforms other algorithms with an average value of 0.801819. *DenStream's* highest point occurs at instance 13000 and lowest point at instance 89000. *ClusTree's* highest point occurs at instance 21000 and lowest point at instance zero. The modified *DenStream's* highest point occurs at instance 10000 and has an average value of 0.715490.

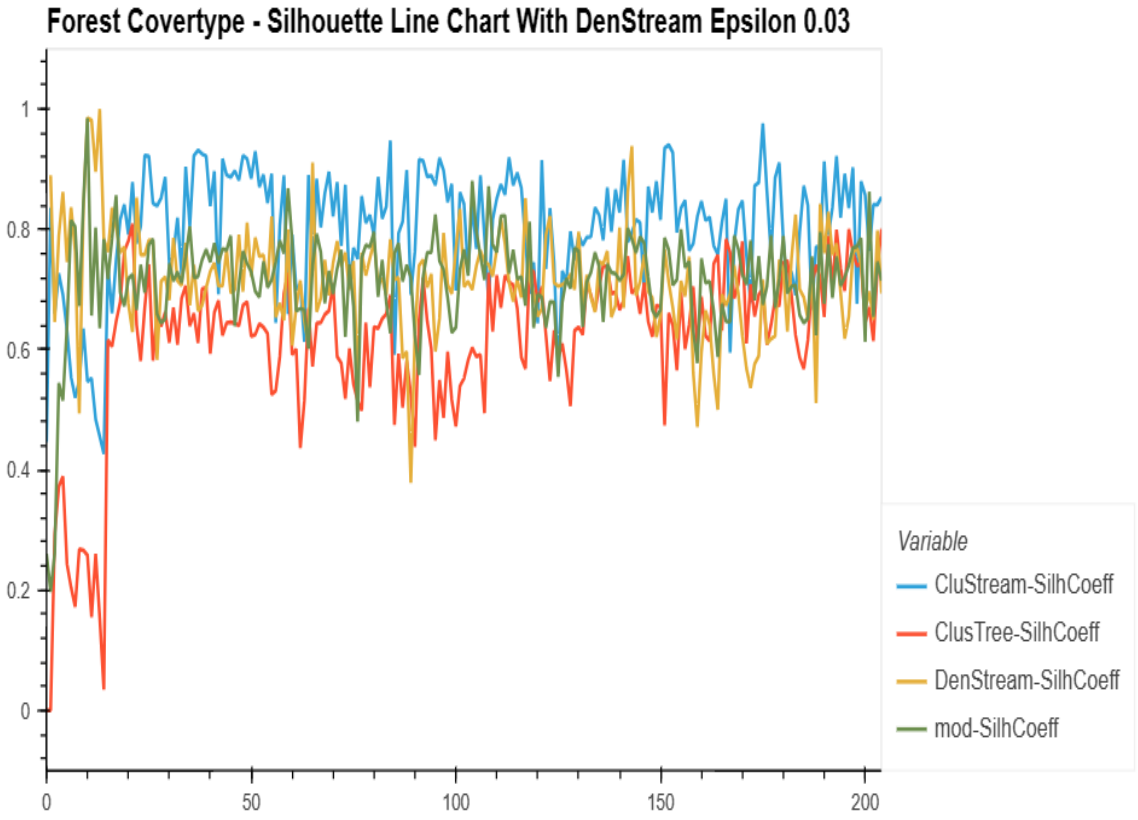


Figure 4-55: The line graph of Forest Covertype using Silhouette Coefficient on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-56 is the visualized line graph of performance metric Rand index on Forest Covertype and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *CluStream* outperforms other algorithms with an average value of 0.564000 and highest point at instance 92000. *DenStream's* highest point is at instance 92000 and lowest point at instance 13000. *ClusTree's* highest point occurs at instance 93000. The modified *DenStream's* lowest point occurs at instance 13000 and has an average value of 0.552186.

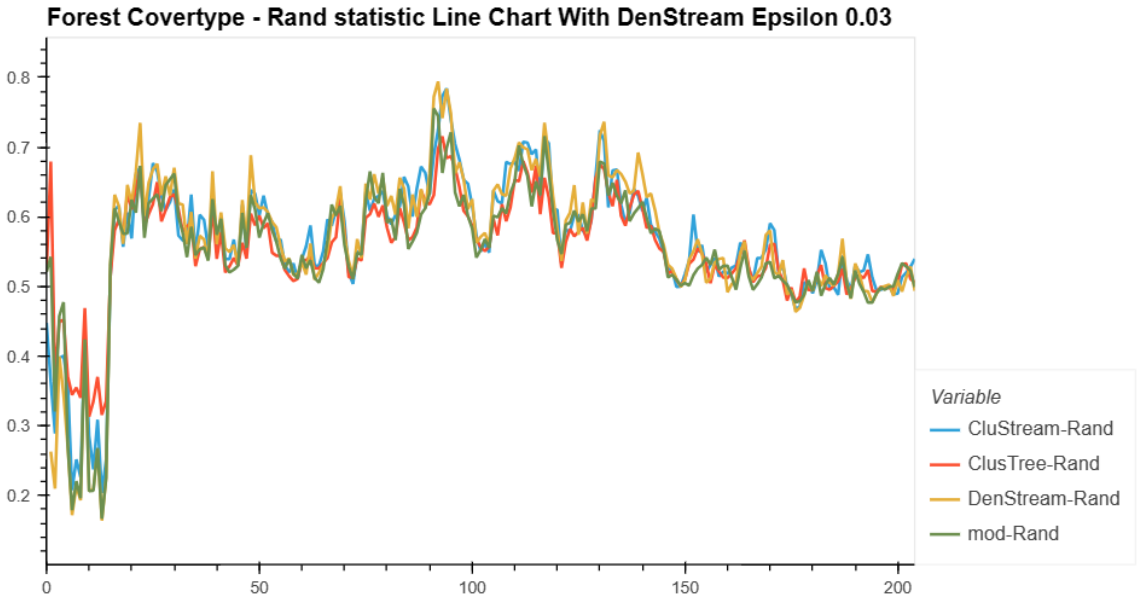


Figure 4-56: The line graph of Forest Covertype using Rand index on DenStream and modified DenStream epsilon set at 0.03.

Table 4-8 presents the performance of the three algorithms on Forest Covertype dataset, *DenStream* and modified *DenStream* epsilon parameter set at 0.03. *ClusTree* outperforms other algorithms using performance metrics CMM and Purity with 0.764913 and 0.983300 respectively. *CluStream* outperforms on performance metrics Silhouette Coefficient and Rand index with 0.801819 and 0.564100 respectively. The modified *DenStream* however outperforms *DenStream* on metric CMM.

Table 4-8: Forest Covertype dataset on epsilon parameter set at 0.03.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.749105	**0.764913**	0.399430	0.427090
Purity	0.918534	**0.983300**	0.978341	0.973233
Silhouette	**0.801819**	0.613921	0.716577	0.715490
Rand index	**0.564100**	0.555170	0.564352	0.552186

Figure 4-57 presents the visualization of the bar chart for the average values of the algorithms on Forest Covertype dataset with epsilon parameter set at 0.03 on performance metrics CMM, Purity, Silhouette Coefficient, and Rand index. The modified *DenStream* shows a better performance against some algorithms at some metrics.

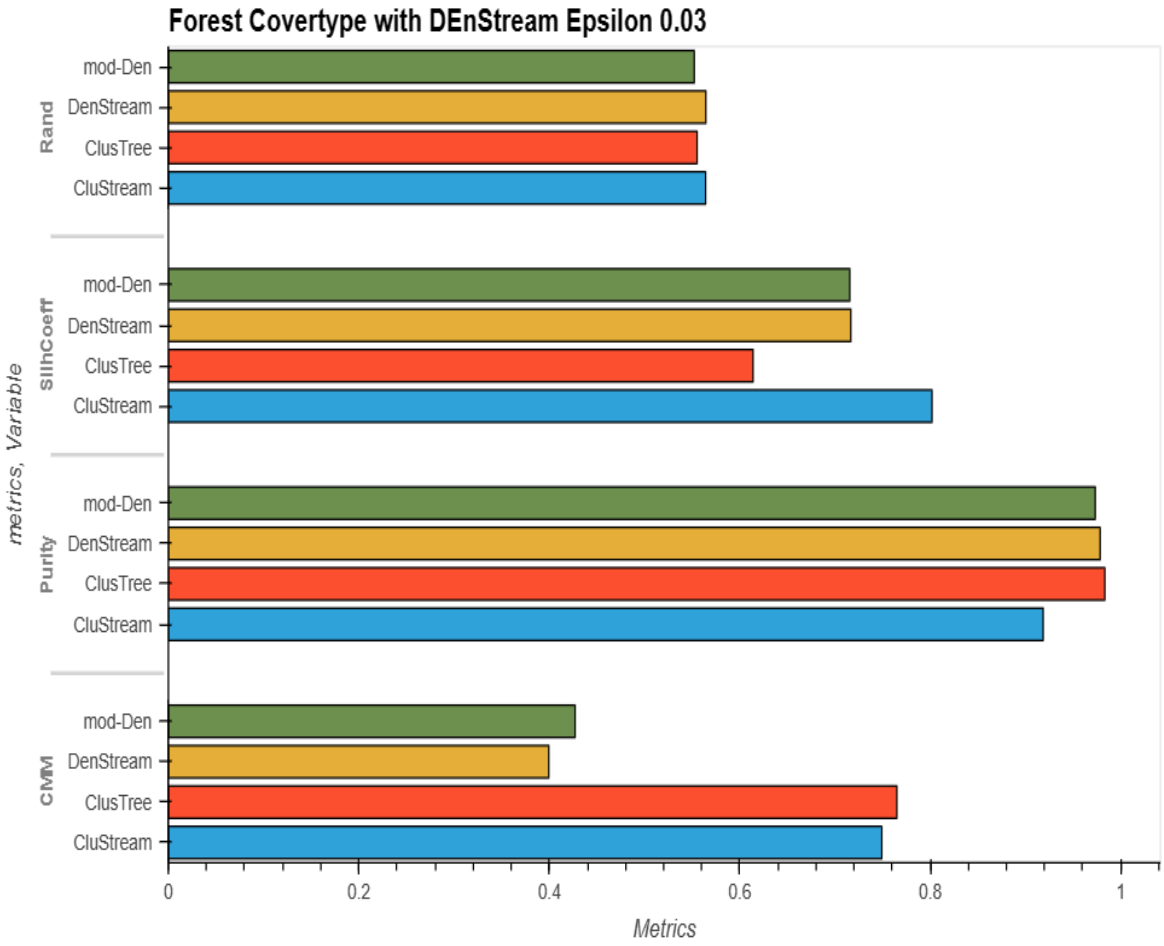


Figure 4-57: Bar plots of ClusTree, CluStream, DenStream, and modified DenStream epsilon set at 0.03 on Forest Covertype dataset.

Figure 4-58 presents the output of performance metric CMM on Forest Covertype, *DenStream* and modified *DenStream* epsilon parameter set at 0.05. *ClusTree* outperforms other algorithms with an average value of 0.764913. *ClusTree's* highest point occurs at instance 15000 and lowest point at instance 142000. *DenStream's* highest point happens at instance 161000 and lowest point at instance 14000. *CluStream's* highest point value is at instance 12000 and lowest point at instance 151000. The modified *DenStream* has its highest point at instance 204000 and lowest point at instance 14000. The modified *DenStream's* average value is 0.518633 and outperforms that of *DenStream* with an average value of 0.484317.

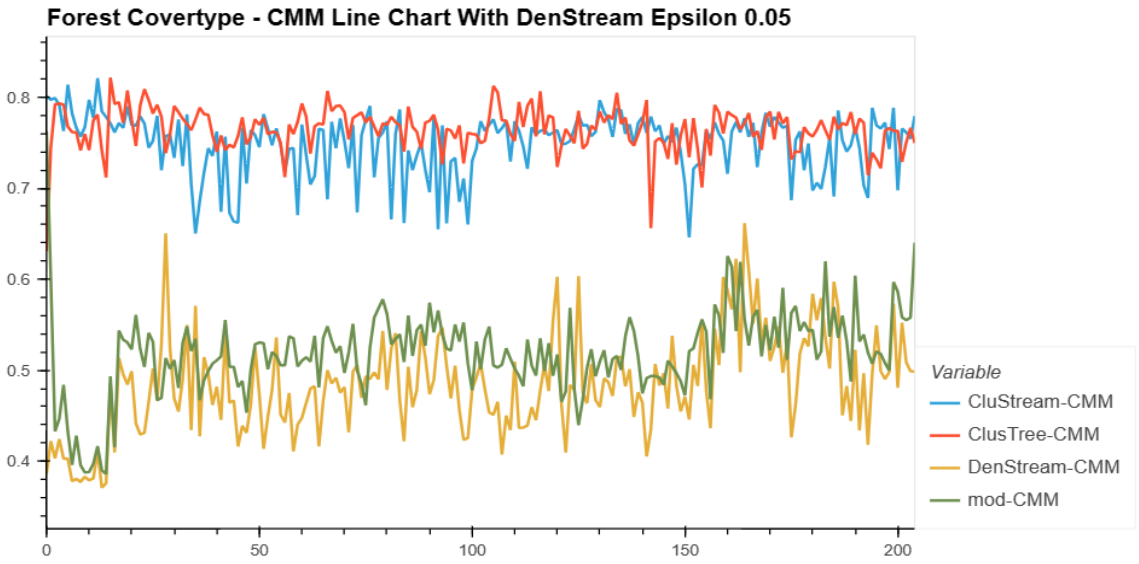


Figure 4-58: The line graph of Forest Covertype dataset using CMM on DenStream and modified DenStream epsilon set at 0.05.

Figure 4-59 is the visualized line graph of performance metric Purity on Forest Covertype dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05. *DenStream's* lowest point occurs at instance 61000 and has an average value of 0.955072. *ClusTree's* lowest point occurs at instance 125000 and has better performance with an average value of 0.983300. *CluStream's* lowest point occurs at instance 14000 and underperformed with an average value of 0.918534. The modified *DenStream* outdoes *DenStream* with an average value of 0.960213.

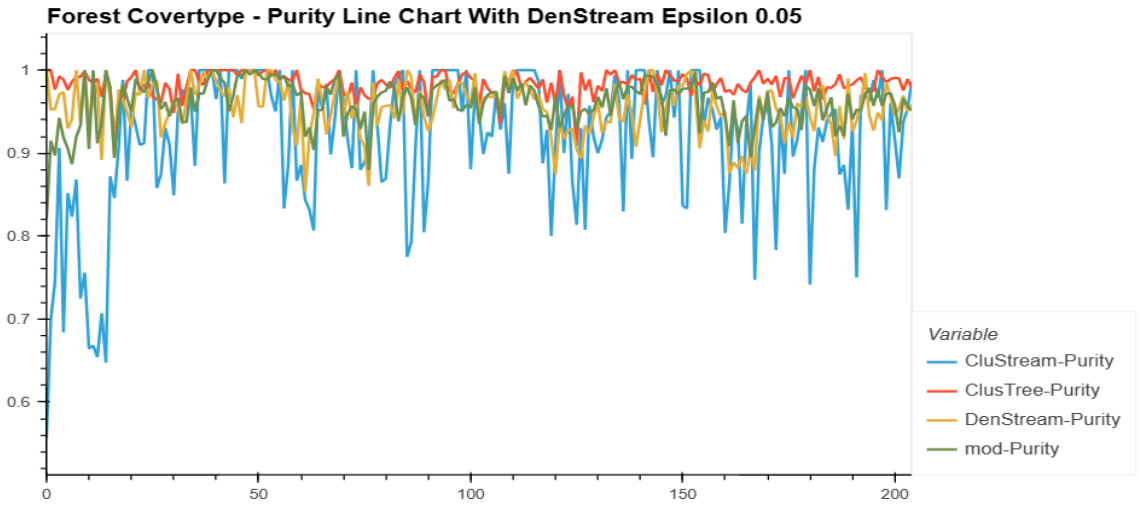


Figure 4-59: The line graph of Forest Covertype dataset using Purity DenStream and modified DenStream epsilon set at 0.05.

Figure 4-60 is the visualized line graph of performance metric Silhouette Coefficient on Forest Covertype dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05. *DenStream's* highest point value is at instance 14000 and lowest point value at instance 166000. *CluStream* outruns other algorithms with an average value of 0.801819. *ClusTree* underperformed with an average value of 0.613921. *ClusTree's* highest point value is at instance 21000 and lowest point at instance zero. The modified *DenStream* outdoes *DenStream* and *CluStream* with an average value of 0.664602 as against 0.629341 and 0.613921 achieved by *DenStream* and *CluStream* respectively.

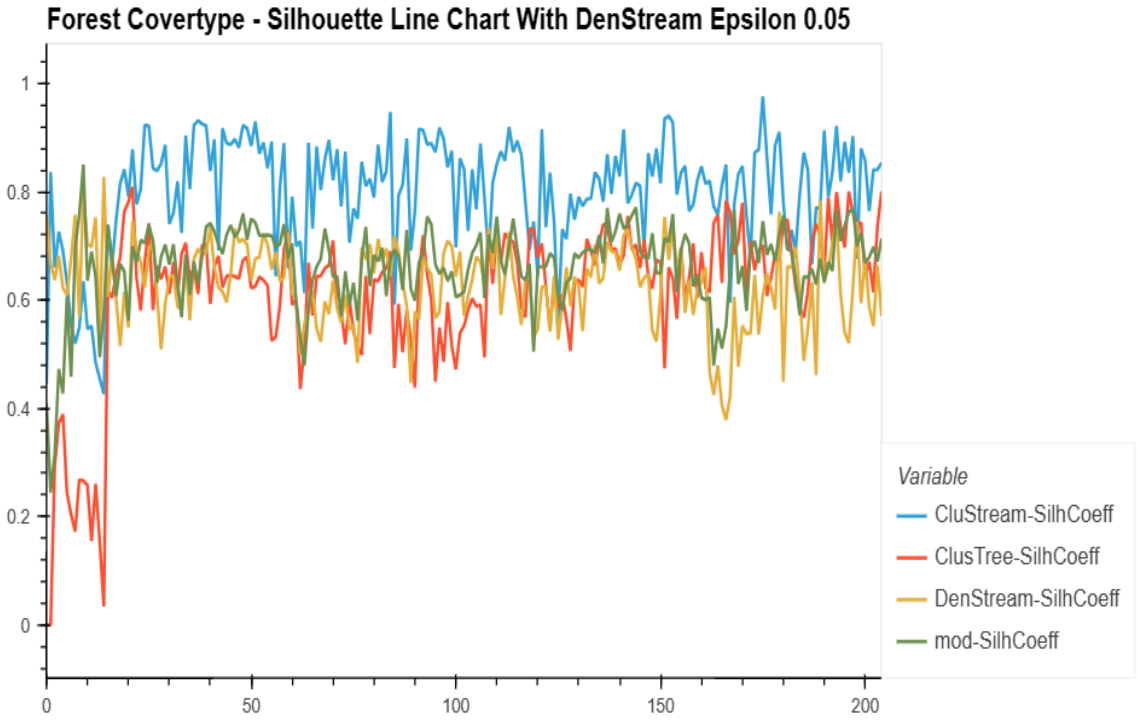


Figure 4-60: The line graph of Forest Covertype dataset using Silhouette Coefficient on DenStream and modified DenStream epsilon set at 0.05.

Figure 4-61 is the visualized line graph of performance metric Rand index on Forest Covertype dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05. *CluStream* outperforms other algorithms with 0.564100. *CluStream's* highest point occurs at instance 93000. *DenStream's* lowest point value is at instance 13000 and highest point is at instance 48000. The modified *DenStream* underperformed with an average value of 0.496804.

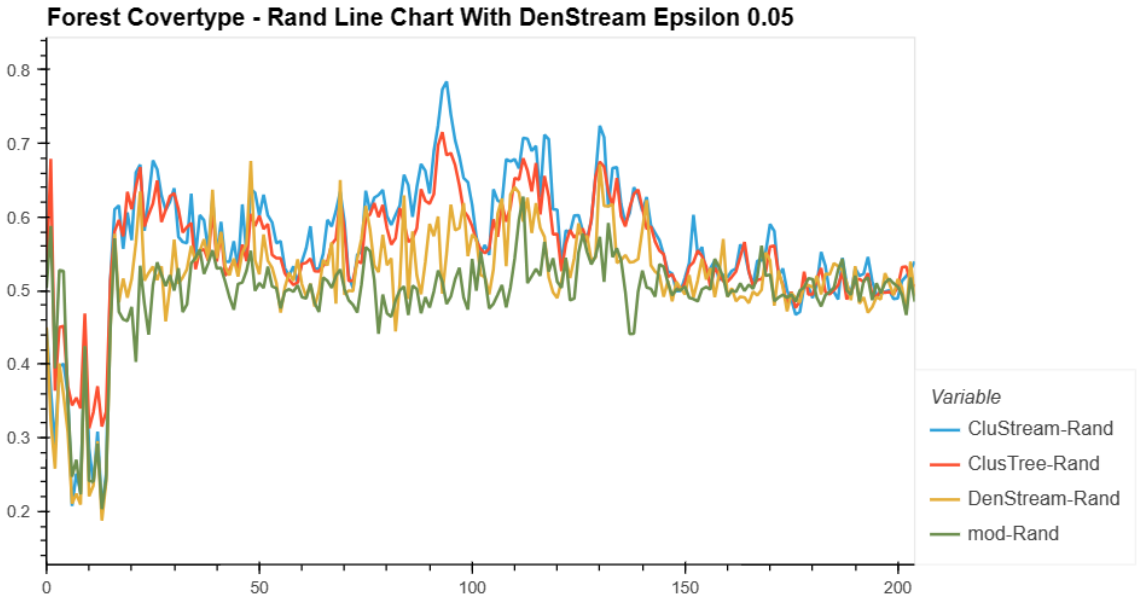


Figure 4-61: The line graph of Forest Covertype dataset using Rand index on DenStream and modified DenStream epsilon set at 0.05.

In Table 4-9, the summary of performance of the three algorithms on Forest Covertype, *DenStream* and modified *DenStream* epsilon 0.05 show that *ClusTree* performed best on performance metrics CMM and Purity with 0.764913 and 0.983300 respectively. *CluStream* outperforms on performance metrics Silhouette Coefficient and Rand index with 0.801819 and 0.564100 respectively. The modified *DenStream* outperforms *DenStream* on metrics CMM, Silhouette Coefficient, and Purity.

Table 4-9: Forest Covertype dataset on epsilon parameter set at 0.05.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.749105	**0.764913**	0.484317	0.518633
Purity	0.918534	**0.983300**	0.955072	0.960213
Silhouette	**0.801819**	0.613921	0.629341	0.664602
Rand index	**0.564100**	0.555170	0.517388	0.496804

Figure 4-62 is the visualized bar chart for the average values of the algorithms on Forest Covertype dataset with epsilon parameter set at 0.05 on performance metrics CMM, Purity, Silhouette Coefficient, and Rand index. The modified *DenStream* outruns *DenStream* on metrics CMM, Purity, and Silhouette Coefficient.

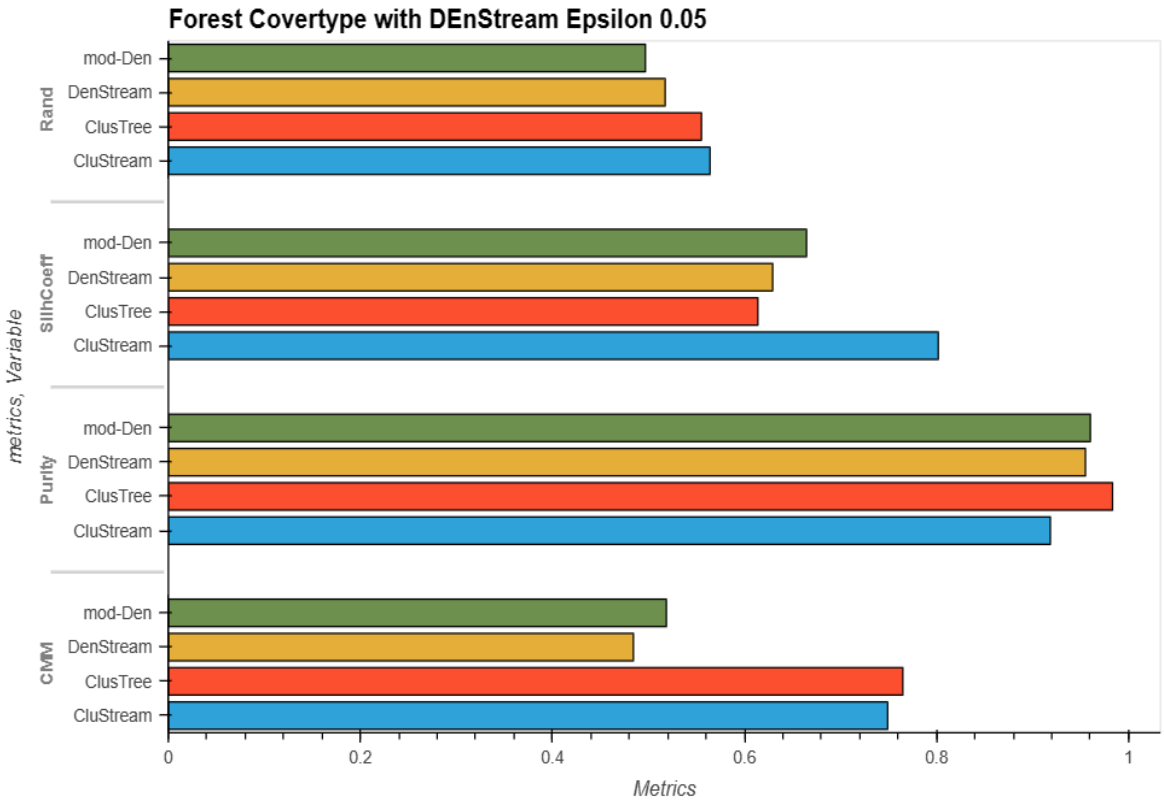


Figure 4-62: Bar plots of CluStream, ClusTree, DenStream, and modified DenStream epsilon set at 0.05 on Forest Covertype dataset.

4.2.8 Effects of Epsilon parameter tuning on Electricity dataset.

This section demonstrates the performance of the three algorithms on Electricity dataset. The evaluation shows the effects of epsilon parameter adjustment on modified *DenStream* and *DenStream* against *CluStream* and *ClusTree*. The line chart for *DenStream* and modified *DenStream* with epsilon 0.03 against *ClusTree* and *CluStream* algorithms are presented in Figure 4-63, Figure 4-64, Figure 4-65, and Figure 4-66 using performance metrics CMM, Purity, Silhouette Coefficient and Rand index.

Figure 4-63 is the visualized line graph of performance metric CMM on Electricity dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *ClusTree* outperforms other algorithms with an average value of 0.795963. *ClusTree's* highest point occurs at instance 40000 and lowest point at instance zero. *DenStream* highest point value is at instance 8000 and lowest point is at instance 23000. *DenStream* has the least average value of 0.600013. *CluStream's* highest point value is at instance 19000 and lowest point is at instance zero. *CluStream's* average value is 0.759476. The modified *DenStream* outclasses *DenStream* with an average value of 0.643626. The modified *DenStream's* highest point occurs at instances zero to 4000.

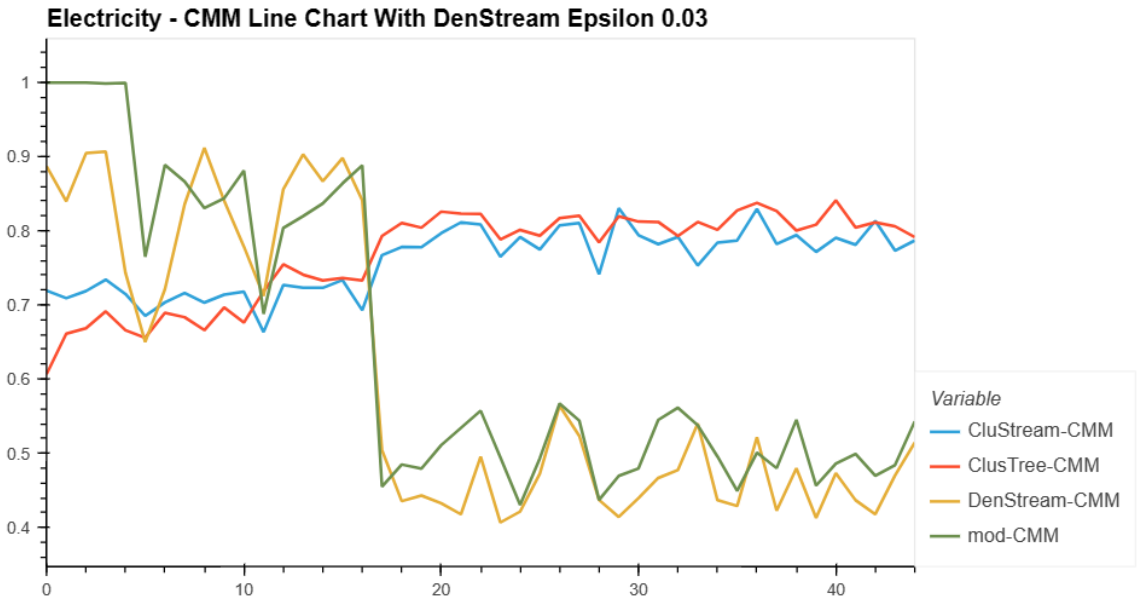


Figure 4-63: The line graph of Electricity dataset using CMM on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-64 is the visualized line graph of performance metric Purity on Electricity dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *ClusTree* outperforms other algorithms with an average value of 0.869373. *DenStream's* lowest point occurs at instance 13000. *ClusTree's* lowest point is at instance 12000 and highest point is at instance zero. The modified *DenStream's* highest point is at 24000 and has an average value of 0.843952 which outruns both *DenStream* and *ClusTree.*

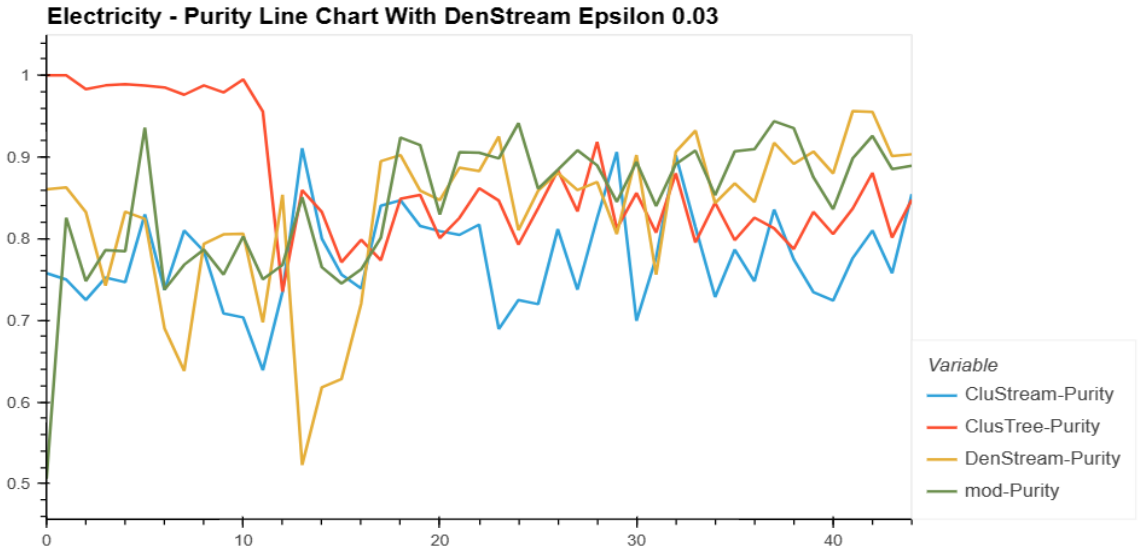


Figure 4-64: The line graph of Electricity dataset using Purity on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-65 is the visualized line graph of performance metric Silhouette Coefficient on Electricity dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *CluStream* outperforms other algorithms with an average value of 0.670170. *CluStream's* lowest point occurs at instance zero and the highest point is at instance 13000. *DenStream's* highest point value is 1.000 at the instances 13000 - 15000 and lowest point is at instance 17000. *ClusTree's* highest point value is at instance 41000. The modified *DenStream's* highest point is at instance zero and its lowest point value is at instance 5000.

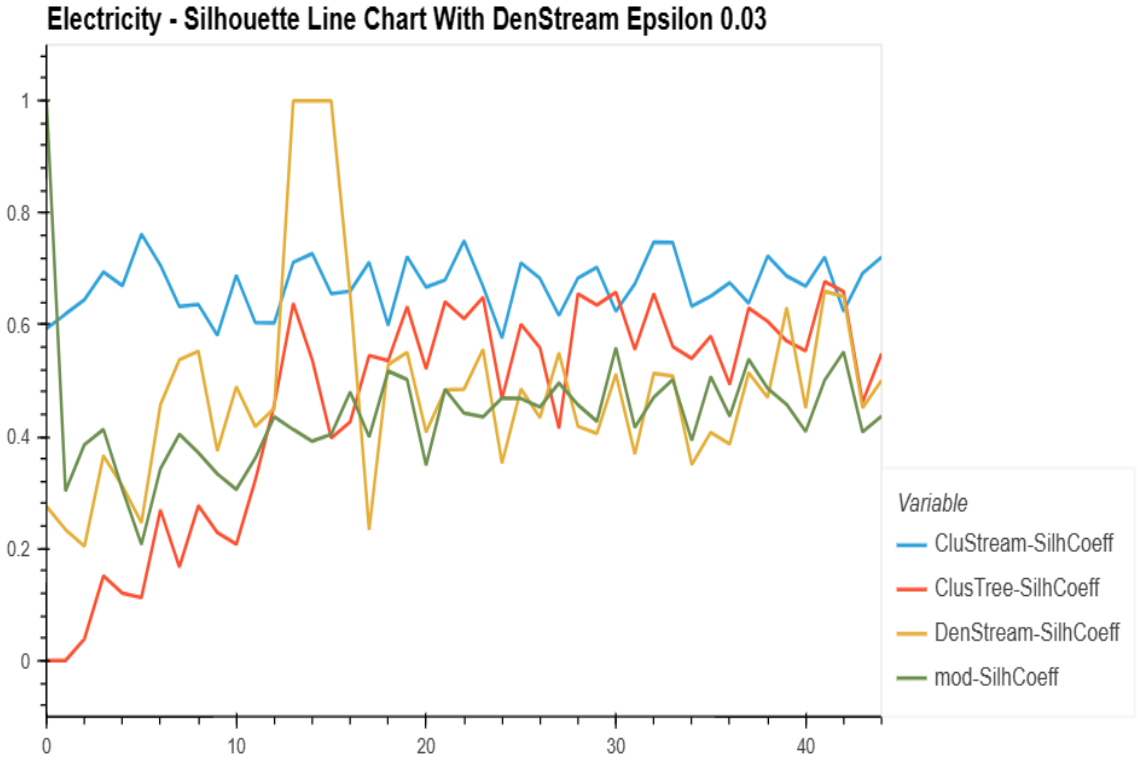


Figure 4-65: The line graph of Electricity dataset using Silhouette Coefficient on DenStream and modified DenStream epsilon set at 0.03.

Figure 4-66 illustrates the line graph of performance metric Rand index on Electricity dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03. *DenStream* outperforms other algorithms with an average value of with values 0.511141. *DenStream's* highest point is at instance 8000 with a value of 0.577 and the lowest point is at instances 1000 and 33000 with a value of 0.490. *ClusTree's* highest point is at instance 21000 with a value of 0.538 and lowest point at instance 1000 with a value of 0.460. *CluStream's* highest point value is 0.571 at instance 21000 and lowest point is at instances zero and 19000 with a value of 0.500. The modified *DenStream* was outclassed by other algorithms.

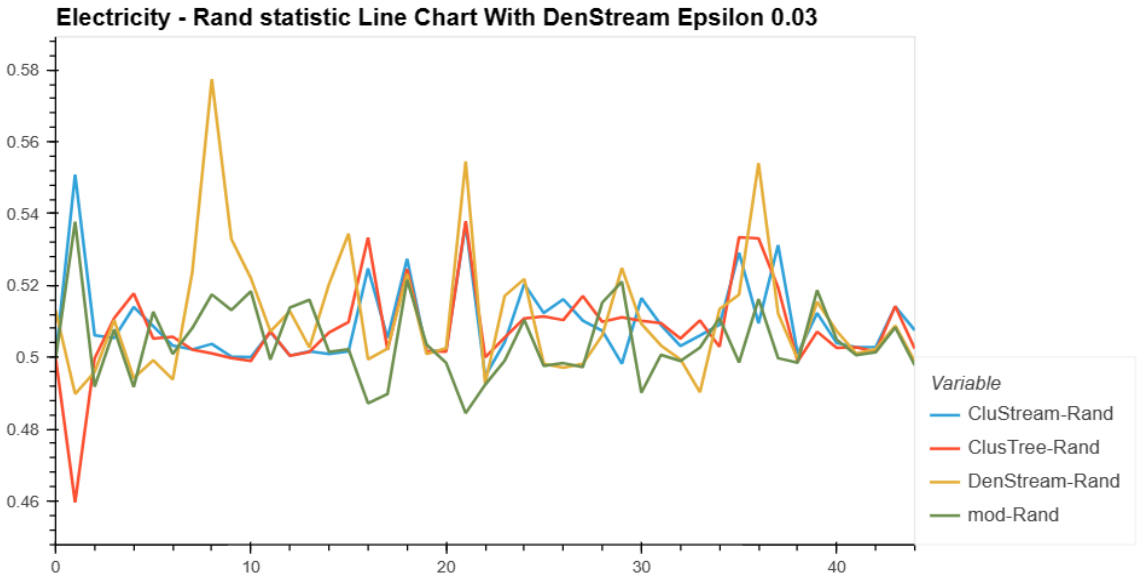


Figure 4-66: The line graph of Electricity dataset using Rand index on DenStream and modified DenStream epsilon set at 0.03.

Table 4-10 presents the tabulated performance of the algorithms on Forest Covertype and *DenStream* and modified *DenStream* epsilon parameter set at 0.03 shows that *ClusTree* outperforms on performance metrics CMM and Purity with 0.765963 and 0.869373 respectively. *CluStream* outperforms on performance metrics Silhouette Coefficient with 0.670710. *DenStream* outperforms on metric Rand index with a value of 0.511141. However, the modified *DenStream* outperforms against *DenStream* on metrics CMM and Purity.

Table 4-10: Electricity dataset on epsilon parameter set at 0.03.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.759476	**0.765963**	0.600013	0.643626
Purity	0.776815	**0.869373**	0.830573	0.843952
Silhouette	**0.670170**	0.457088	0.485307	0.440715
Rand index	0.509466	0.507946	**0.511141**	0.504372

Figure 4-67 is the visualized bar chart for the average values of the algorithms on Electricity dataset with epsilon parameter set at 0.03 on performance metrics CMM, Purity, Silhouette Coefficient, and Rand index. The modified *DenStream* outperformed *DenStream* on metrics CMM and Purity.

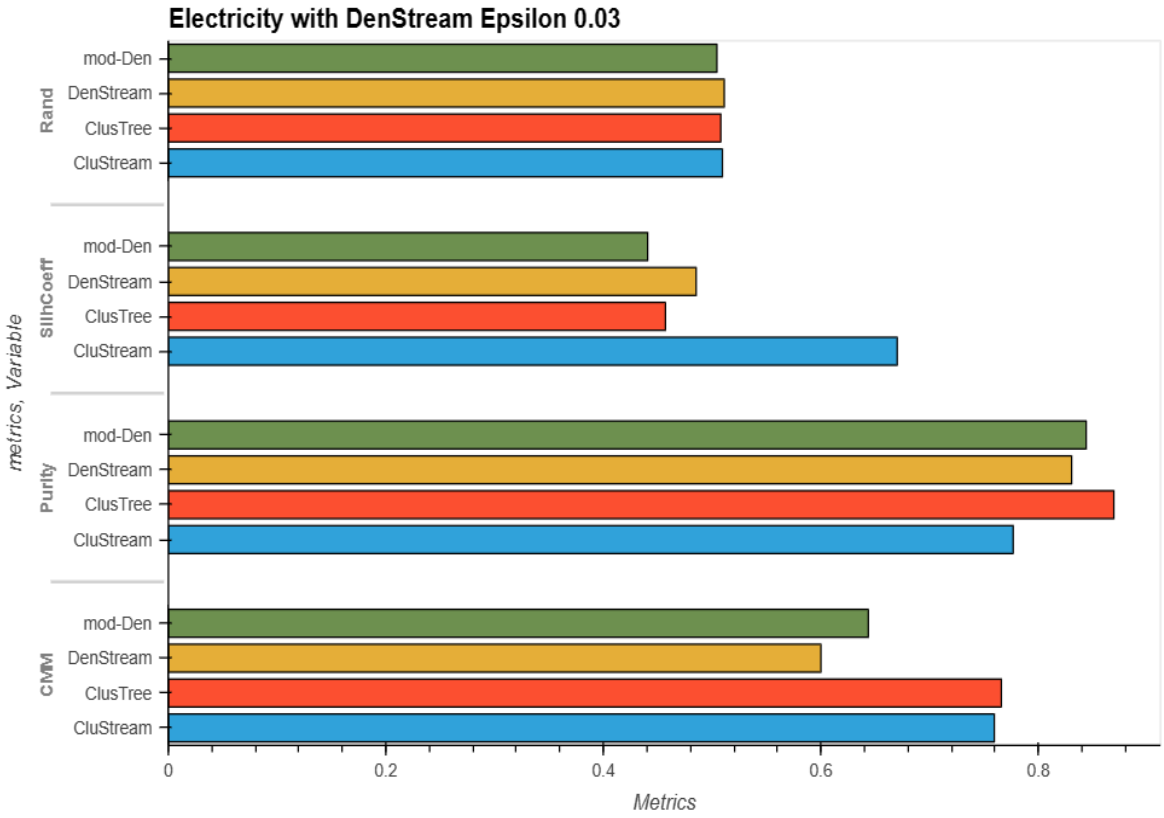


Figure 4-67: Bar plots of CluStream, ClusTree, DenStream, and modified DenStream epsilon set at 0.03 on Electricity dataset.

Again, the effect of modified *DenStream* with adjusted epsilon 0.05 was demonstrated against *CluStream, ClusTree,* and *DenStream* algorithms on the Electricity dataset. The performance metrics CMM, Purity, Silhouette Coefficient, and Rand index were used. The line graph of the performance metrics is presented in Figure 4-68, Figure 4-69, Figure 4-70, and Figure 4-71.

Figure 4-68 presents the line graph of performance metric CMM on Electricity dataset, *DenStream* and modified *DenStream* epsilon parameter set at 0.05. *DenStream* shows a better performance over *ClusTree* and *CluStream* with an average value of 0.783626. The modified *DenStream* likewise outperforms both *ClusTree* and *CluStream* with an average value of 0.776266. *DenStream's* highest point value is at instance instances zero - 4000 and lowest point at instance 35000. The highest point of the modified *DenStream* occurs at instances zero – 4000 and lowest point at instance 18000. *CluStream's* lowest point is at instance 11000 and highest point at instance 29000. *ClusTree's* highest point occurs at instance 40000 and lowest point at instance zero.

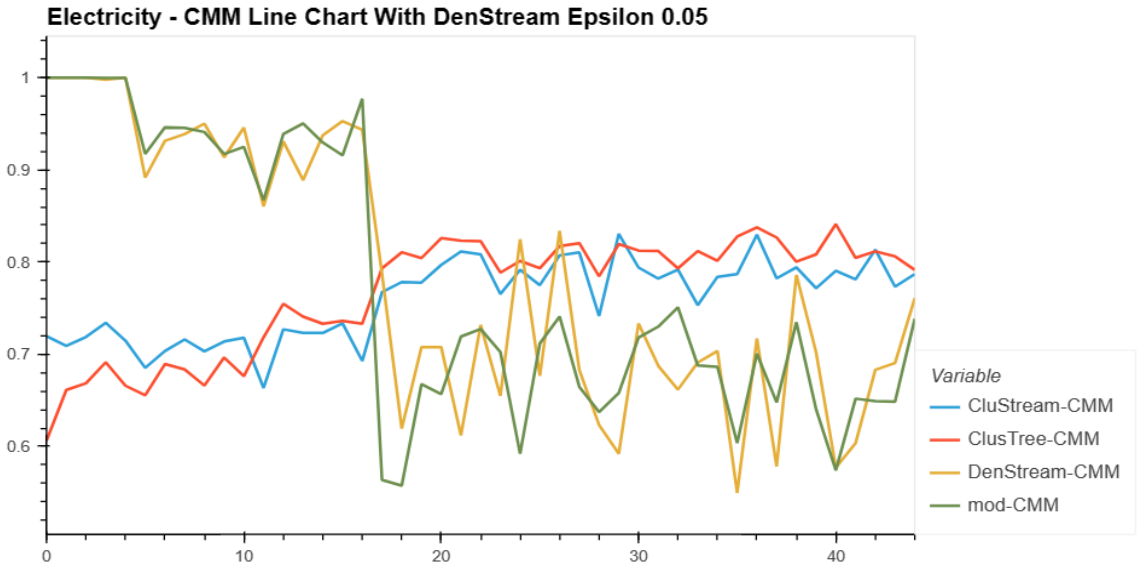


Figure 4-68: The line graph of Electricity dataset using CMM on DenStream and modified DenStream epsilon set at 0.05.

Figure 4-69 is the visualized line graph of performance metric Purity on Electricity dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05. *ClusTree* outperforms other algorithms with an average value of 0.869373. The modified *DenStream* outperforms both *CluStream* and *DenStream* with an average value of 0.784972. *DenStream's* lowest point occurs at instance 5000 and highest point at instance 39000. *ClusTree's* lowest point occurs at instance 12000. *CluStream's* highest point occurs at instance 13000 and lowest point at instance 11000.

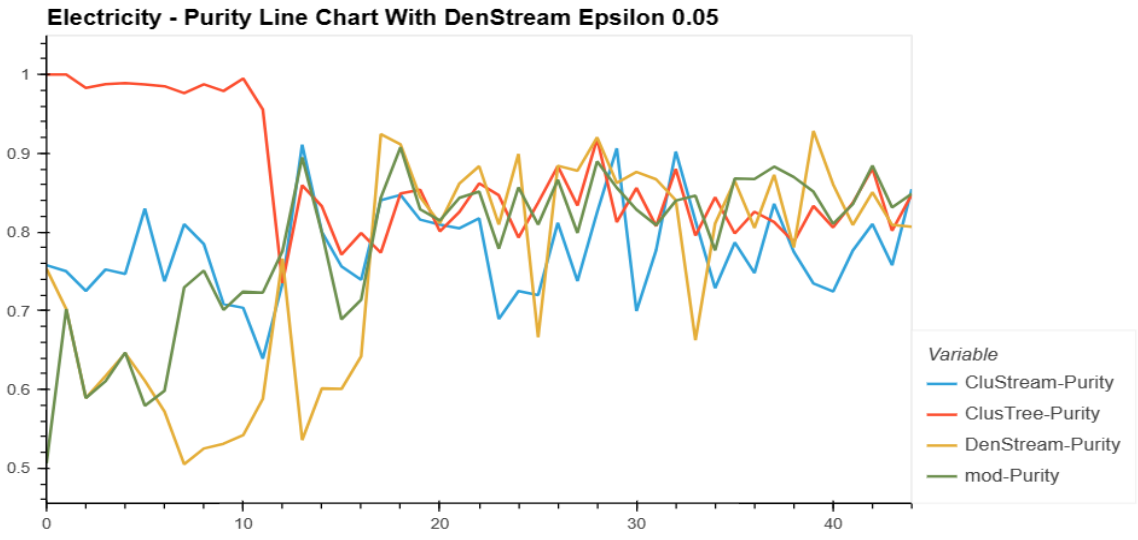


Figure 4-69: The line graph of Electricity dataset using Purity on DenStream and modified DenStream epsilon set at 0.05.

Figure 4-70 is the visualized line graph of performance metric Silhouette Coefficient on Electricity dataset and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05. *DenStream* demonstrates a better performance than *ClusTree* and *CluStream* with an average value of 0.569716. *DenStream's* highest point of 1.00 occurs along many instances and lowest point is at instance 42000. *ClusTree's* highest point is at instance 41000 and lowest point is at instance zero. *CluStream's* highest point occurs at instance 5000 and lowest point at instance 24000.

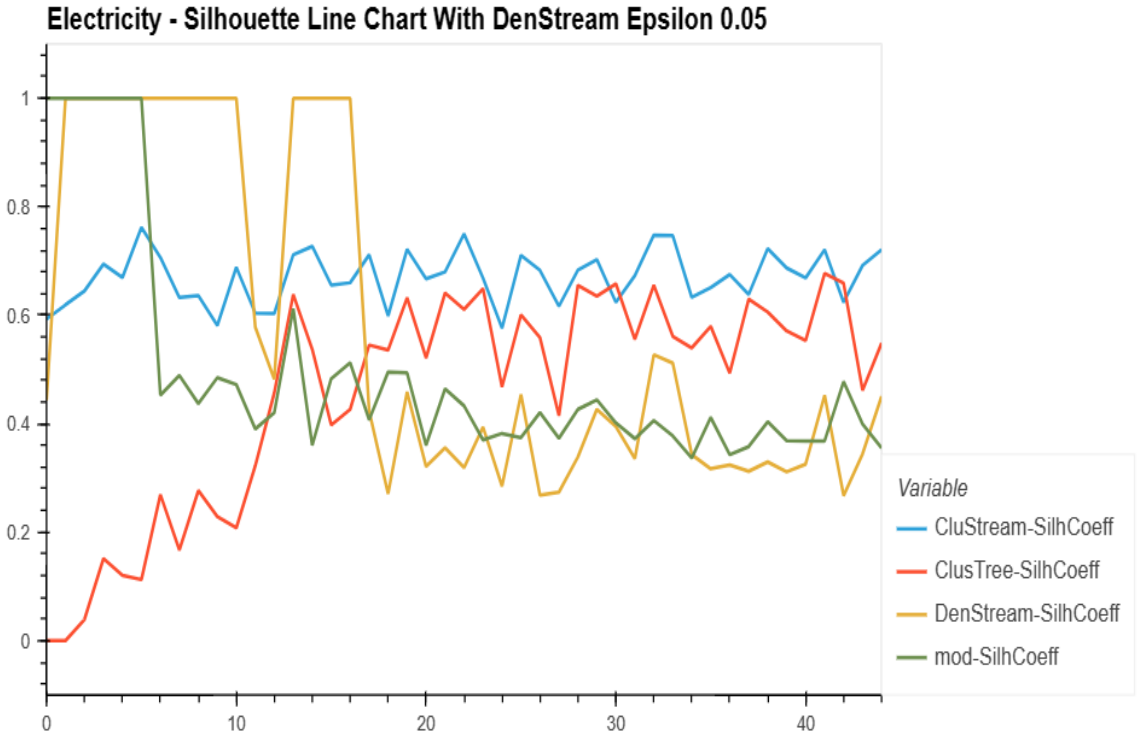


Figure 4-70: The line graph of Electricity dataset using Silhouette Coefficient on DenStream and modified DenStream epsilon set at 0.05.

Figure 4-71 is the visualized line graph of performance metric Rand index on Electricity dataset and epsilon parametrer of *DenStream* and modified *DenStream* set at 0.05. *CluStream* outperforms other algorithms with an average value of 0.509466. *CluStream's* highest point occurs at instance 1000 and lowest point at instances 19000 and 41000. *ClusTree's* lowest point is at instance 1000 and highest point at instance 21000. *DenStream's* highest point is at instance 1000 and lowest point at instance 35000. The modified *DenStream* however, underperforms other algorithms with an average of 0.502474.

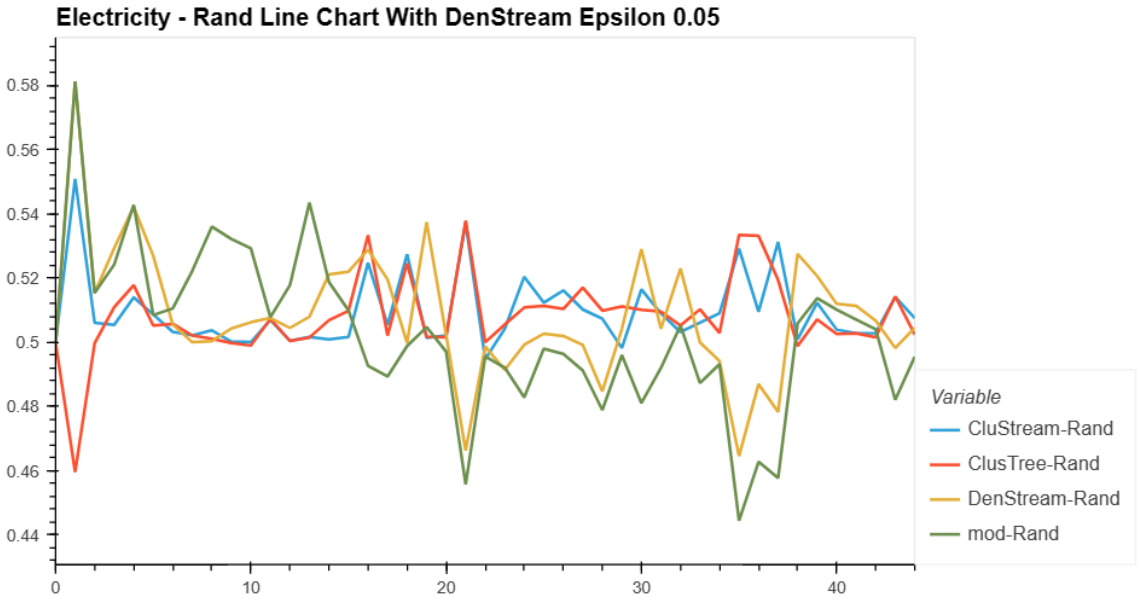


Figure 4-71: The line graph of Electricity dataset on DenStream and modified DenStream epsilon set at 0.05.

Table 4-11 indicates that *DenStream* outperforms on performance metric CMM with 0.783626. *CluStream* outperforms other algorithms on performance metrics Silhouette Coefficient and Rand index with 0.670170 and 0.509466 respectively. *ClusTree* outperforms on performance metric Purity with 0.869373. The modified *DenStream* outperforms *DenStream* on metric Purity.

Table 4-11: Electricity dataset on epsilon parameter set at 0.05.

Metrics	CluStream	ClusTree	DenStream	mod-DenStream
CMM	0.759476	0.765963	**0.783626**	0.776266
Purity	0.776815	**0.869373**	0.753844	0.784972
Silhouette	**0.670170**	0.457088	0.569716	0.495679
Rand index	**0.509466**	0.507946	0.508257	0.502474

Figure 4-72 presents the visualization of the bar chart for the algorithms on Electricity dataset with epsilon parameter set at 0.05 on performance metrics CMM, Purity, Silhouette Coefficient, and Rand index.

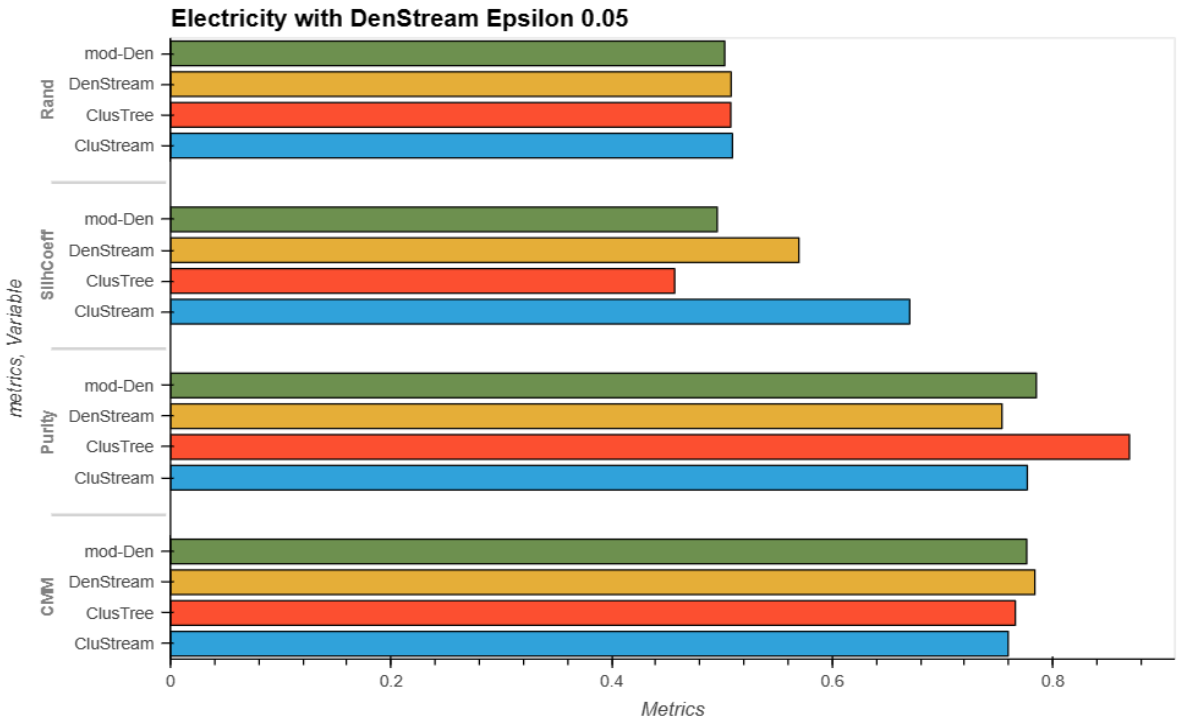


Figure 4-72: Bar plots of CluStream, ClusTree, DenStream, and modified DenStream epsilon set at 0.05 on Electricity dataset.

4.3 Discussion

The choice of a suitable parameter settings in data stream clustering requires expert knowledge. The first research objective to "identify the method of building a MOA repository from source to implement the modified algorithm" was addressed in chapter 3 with implementation of the modified *DenStream.* The second research objective to "identify the hyperparameters appropriate for parameter-tuning" was used to demonstrate the effects of noise levels and epsilon parameter-tuning on *DenStream* and modified *DenStream.* using synthetic data stream and real-world datasets. The values 0.02, 0.03 and 0.05 were set as the epsilon parameter tuning and the noise level set between 0%, 10%, and 30%. Lastly, the third research objective identifies the performance metrics CMM, Silhouette coefficient, and Rand index.

The experimental results using the default parameter settings on the *RandomRBFGenerator* in Table 4-1. The results indicate that *ClusTree* outperforms other algorithms on performance metrics CMM, Silhouette Coefficient and Rand index with an average value of 0.902690, 0.771385, and 0.885936 respectively. The modified *DenStream* outperforms other algorithms on metric Purity with an average value of 0.951441 and likewise outperforms *DenStream o*n metrics CMM and Rand index.

On the performance of Forest Covertype using the default parameter settings in Table 4-2, the results show that *CluStream* outperform all other algorithms on metrics CMM with an average value 0.749105. *ClusTree* outperforms other algorithms on metric Silhouette coefficient with a mean value 0.829813. *DenStream* outperforms other algorithms on metrics Purity and Rand index with an average value of 0.972418 and 0.582461 respectively. The average value between *DenStream* and modified *DenStream* on metric Purity looks very similar. However, the modified *DenStream* outperforms *DenStream* on metric CMM with 0.387108.

On the performance using the Electricity dataset with default settings in Table 4-3, *CluStream* outperforms other algorithms using metric CMM with 0.759476. *ClusTree* outperforms other algorithms using metrics Silhouette Coefficient, and Rand index with 0.732103 and 0.512624 respectively. *DenStream* also outperforms other algorithms on performance metric Purity with 0.897288. The modified *DenStream* however, outperforms *DenStream* using metric CMM with 0.544408.

On *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03 in Table 4-4, *CluStream* outperformed all other algorithms on Silhouette Coefficient with a value of 0.821436. *ClusTree* outperforms other algorithms using metrics CMM and Purity with 0.984734 and 0.974619 respectively. Lastly, the modified *DenStream* outperforms on Rand index with a value of 0.889164.

On *RandomRBFGenerator* with 0% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05 in Table 4-6, *ClusTree* outperforms other algorithms using metrics CMM and Purity with 0.984734 and 0.974619 respectively. *CluStream* outperforms on performance metrics Silhouette Coefficient and Rand index with 0.821436 and 0.839968 respectively. The modified *DenStream* however, outperforms *DenStream* on all the metrics.

On *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.03 in Table 4-5, *DenStream* outperforms other algorithms using metrics CMM and Silhouette Coefficient with an 0.825596 and 0.747127 respectively. *ClusTree* outperforms other algorithms using metric Purity with 0.976311. The modified *DenStream* outperforms other algorithms using metric Rand index with 0.844152. The modified *DenStream* also shows a better performance against *DenStream* on metric Purity.

On *RandomRBFGenerator* with 30% noise level and epsilon parameter of *DenStream* and modified *DenStream* set at 0.05 in Table 4-7, *ClusTree* outperforms other algorithms on

performance metrics CMM, Purity, and Rand index with 0.764806, 0.976311, and 0.841067 respectively. The modified *DenStream* outperforms on metric Silhouette Coefficient with a value of 0.644882. The modified *DenStream* likewise outperforms against *DenStream* on all metrics.

On the Forest Covertype dataset with epsilon parameter of *DenStream* and modified *DenStream* set at 0.03 in Table 4-8, *CluStream* outperforms other algorithms using metrics Silhouette Coefficient and Rand index with 0.801819 and 0.564100 respectively. *ClusTree* outperforms other algorithms using metrics CMM and Purity with 0.764913 and 0.983300 respectively. The modified *DenStream* however outperforms *DenStream* on metric CMM.

On the Forest Covertype dataset with epsilon parameter of *DenStream* and modified *DenStream* set at 0.05 in Table 4-9, *ClusTree* has a better performance on metrics CMM and Purity with 0.764913 and 0.983300 respectively. *CluStream* outperforms other algorithms using metrics Silhouette Coefficient and Rand index with 0.801819 and 0.564100 respectively. The modified *DenStream* outperforms *DenStream* on metrics CMM, Purity, and Silhouette Coefficient.

On the Electricity datas with epsilon parameter of *DenStream* and modified *DenStream* set at 0.03 in Table 4-10, *ClusTree* outperforms other algorithms using metrics CMM and Purity with 0.765963 and 0.869373 respectively. *CluStream* outperforms other algorithms using metrics Silhouette Coefficient with 0.670710. *DenStream* outperforms other algorithms using metric Rand index with 0.511141. However, the modified *DenStream* outperforms against *DenStream* on metrics CMM and Purity.

On the Electricity dataset with epsilon parameter of *DenStream* and modified *DenStream* set at 0.05 in Table 4-11, *DenStream* outperforms other algorithms using metric CMM with 0.783626. *CluStream* outperforms other algorithms using metrics Silhouette Coefficient and Rand index with 0.670170 and 0.509466 respectively. *ClusTree* outperforms other algorithms using metric Purity with 0.869373. The modified *DenStream* however, outperforms *DenStream* on performance metric Purity.

CHAPTER 5: Conclusions and Future Work

5.0 Conclusions

In conclusion, this dissertation demonstrates the performance of a modified *DenStream* algorithm against state-of-the-art algorithms *CluStream, ClusTree,* and *DenStream* on the Massive Online Analysis (MOA) tool. The analysis involves the streaming synthetic dataset generated in MOA using the *RandomRBFGenerator* and real-world datasets (Electricity and Forest Covertype). The research objective to "identify the method of building a MOA repository from source to implement the modified algorithm" was done using the IntelliJ IDEA Community Edition 2022.2 and we implemented the modified *DenStream* in Java on it. We demonstrated the modified *DenStream* against other algorithms using the default parameter settings for synthetic dataset generated using *RandomRBFGenerator* with 205000 instances. On the real-world dataset Forest Covertype, it involves using 205000 instances, and on the Electricity dataset, it involves using 45000 instances.

To answer the research objective "identify the hyperparameters appropriate for parameter-tuning", we identified two hyperparameters (epsilon and minPts) in *DenStream* suitable for parameter specification adjustment. We demonstrated the effects *RandomRBFGenerator* noise level adjustment between 0%, the default 10%, and 30% on the modified *DenStream* against algorithms and implemented epsilon parameter-tuning using the default 0.02, 0.03 and 0.05 respectively. We also compared the effects against other algorithms using *RandomRBFGenerator* noise level between 0%, the default 10%, and 30% on real-world datasets (Electricity and Forest Covertype).

The experimentation involves identifying appropriate performance metrics for clustering quality which addressed the research objective to "identify the performance metrics applicable for clustering quality". We identified the performance metrics CMM, Silhouette Coefficient, and Rand index and demonstrated the evaluation of the algorithms using these metrics. The results based on *RandomRBFGenerator* with default settings show that *CluStream* performed better on performance metrics (CMM, Silhouette Coefficient, and Rand index) compared to other algorithms. The modified *DenStream* outperforms other algorithms on metric Purity and shows a better performance against *DenStream* on metrics CMM and Rand index.

On Forest Covertype with default settings, *CluStream* outperforms all other algorithms on performance metrics CMM, *DenStream* outperforms other algorithms using metrics Purity

and Rand index, *ClusTree* outperforms other algorithms using metric Silhouette coefficient. However, the modified *DenStream* outperforms *DenStream* on performance metric CMM. The Electricity dataset with default settings indicates that *CluStream* outperforms other algorithms on metric CMM; *ClusTree* outperforms other algorithms using metrics Silhouette Coefficient and Rand index; *DenStream* also outperforms on performance metric Purity. However, the modified *DenStream* outperforms *DenStream* using metric CMM.

On parameter-tuning and noise levels, the modified *DenStream* outperformed *DenStream* on performance metrics CMM, Purity and Rand index on *RandomRBFGenerator* with 0% noise level and epsilon parameter set at 0.03 and 0.05. On *RandomRBFGenerator* with 30% noise level and epsilon parameter set at 0.03 and 0.05, the modified *DenStream* outperformed *DenStream* on performance metrics Purity and Rand index and other algorithms (*CluStream* and *ClusTree*) in at least one metric (CMM and Silhouette Coefficient) using epsilon parameter set at 0.03. The modified *DenStream* also outperforms other algorithms on performance metric Silhouette Coefficient with epsilon parameter set at 0.05. The modified *DenStream* outruns *DenStream* on all performance metrics using epsilon parameter at 0.05.

Lastly, using real-world datasets (Electricity and Forest Covertype) shows that on Forest Covertype with epsilon parameter set at 0.03, the modified *DenStream* outruns some algorithms at some metrics. The experimental results using Forest Covertype dataset with epsilon parameter set at 0.05 show that modified *DenStream* outclasses *DenStream* on metrics CMM, Purity, and Silhouette Coefficient. *ClusTree* outperforms other algorithms metric Silhouette Coefficient and *DenStream* outperforms on metric CMM, *CluStream* outperforms other algorithms on metrics Purity and Rand index using Electricity dataset with epsilon parameter set at 0.03. The Electricity dataset with epsilon parameter set at 0.05, shows that *DenStream* outperforms other algorithms using metric CMM. *ClusTree* outperforms on performance metric Silhouette Coefficient; and *CluStream* outperforms other algorithms using metrics Rand index and Purity. However, the modified *DenStream* outshine *DenStream* on performance metric Purity.

We were unable to demonstrate the performance of the modified *DenStream* on memory usage and time because the MOA framework could not display the chart for memory and time for comparison on the algorithms.

5.1 Future Work

The future work of this research will try to improve on the modified algorithm and experiment on the effects of hyper-parameters tuning like the decay factor and outlier threshold. The implementation of an improved algorithm on the most critical challenges of parameter settings in data stream clustering which can detect clusters of arbitrary shape, group data streams into clusters, preserve clusters dynamically, and the visualization of the memory usage and time are other future directions.

Hybrid algorithm is another direction for future research. Proposing a hybrid algorithm robust arbitrary shapes detection and resistance to noise will be of interest to both researchers and academia.

References

Abid, A., Jamoussi, S., & Hamadou, A. Ben. (2019). AIS-Clus: A Bio-Inspired Method for Textual Data Stream Clustering. *Vietnam Journal of Computer Science*, *06*(02), 223–256. https://doi.org/10.1142/s2196888819500143

Ackermann, M. R., Lammersen, C., Sohler, C., Swierkot, K., & Raupach, C. (2012). StreamKM++: A Clustering Algorithm for Data Stream. *Journal of Experimental Algorithmics*, *17*(1), 173–187. https://doi.org/https://doi.org/10.1145/ 2133803.2184450

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2003). A Framework for Clustering Evolving Data Streams. *{VLDB} 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, 81–92. http://www.vldb.org/conf/2003/papers/S04P02.pdf

Aggarwal, C. C., & Reddy, C. K. (2014). DATA Clustering: Algorithms and Applications. In *CRC Press*. CRC Press Taylor & Francis Group.

Aggarwal, C., Han, J., Wang, J., & Yu, P. (2004). A Framework for Projected Clustering of High Dimensional Data Streams. *Proceedings 2004 VLDB Conference*, 852–863. https://doi.org/10.1016/b978-012088469-8/50075-9

Agrawal, L. S., & Adane, D. S. (2016). Models and Issues in Data Stream Mining. *International Journal of Computer Science and Applications*, *9*(1), 6–10.

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 94–105.

Ahmed, R., Dalkılıç, G., & Erten, Y. (2020). DGStream: High quality and efficiency stream clustering algorithm. *Expert Systems with Applications*, *141*, 112947–112959. https://doi.org/10.1016/j.eswa.2019.112947

Akinosho, T. A., Tabane, E., & Zenghui, W. (2023). Performance Evaluation of Data Stream Clustering Algorithm on Parameter Specification. *International Conference on Wireless Intelligent and Distributed Environment for Communication*, 173–189.

Aljibawi, M., Zakree, M., Nazri, A., Nor, A., & Sani, S. (2022). AN ENHANCED MUDI-STREAM ALGORITHM FOR CLUSTERING DATA STREAM. *Article in Journal of Theoretical and Applied Information Technology*, *15*(9). https://www.researchgate.net/publication/360748957

Al-shammari, A. (2019). *Towards Improving Data Summarisation and their Dynamic Maintenance* (Issue September). Swinburne University of Technology.

Al-Shammari, A., Zhou, R., Naseriparsaa, M., & Liu, C. (2019). An effective density-based clustering and dynamic maintenance framework for evolving medical data streams. *International Journal of Medical Informatics*, *126*(February), 176–186. https://doi.org/10.1016/j.ijmedinf.2019.03.016

Amini, A., Saboohi, H., Herawan, T., & Wah, T. Y. (2016). MuDi-Stream: A multi density clustering algorithm for evolving data stream. *Journal of Network and Computer Applications*, *59*, 370–385. https://doi.org/10.1016/j.jnca.2014.11.007

Amini, A., Wah, Y., & Saboohi, H. (2014). Amini A, Wah TY, Saboohi H. On density-based data streams clustering algorithms: A survey On Density-Based Data Streams Clustering Algorithms: A Survey. *JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY*, *29*(1), 116–141. https://doi.org/10.1007/s11390-013-1416-3

Andreopoulos, B., An, A., Wang, X., & Schroeder, M. (2009). A roadmap of clustering algorithms: Finding a match for a biomedical application. In *Briefings in Bioinformatics* (Vol. 10, Issue 3, pp. 297–314). https://doi.org/10.1093/bib/bbn058

Attaoui, M. O., Azzag, H., Lebbah, M., & Keskes, N. (2022). *Improved Multi-objective Data Stream Clustering with Time and Memory Optimization*. http://arxiv.org/abs/2201.05079

Bahri, M., Salutari, F., Putina, A., & Sozio, M. (2022). AutoML: state of the art with a focus on anomaly detection, challenges, and research directions. In *International Journal of Data Science and Analytics* (Vol. 14, Issue 2, pp. 113–126). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/s41060-022-00309-0

Baker, F. B., & Hubert, L. J. (1975). Measuring the power of hierarchical cluster analysis. *Journal of the American Statistical Association*, *70*(349), 31–38.

Bezdek, J. C., & Keller, J. M. (2021). Streaming Data Analysis: Clustering or Classification? *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *51*(1), 91–102. https://doi.org/10.1109/TSMC.2020.3035957

Bifet, A., Read, J., Holmes, G., & Pfahringer, B. (2018). Streaming Data Mining with Massive Online Analytics (MOA). *Data Mining in Time Series and Streaming Databases*, 1–25.

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-Theory and Methods*, *3*(1), 1–27.

Cao, F., Ester, M., Qian, W., & Zhou, A. (2006). Density-Based Clustering over an Evolving Data Stream with Noise. *SIAM Conference on Data Mining*, 328–339.

Cao, F., Estert, M., Qian, W., & Zhou, A. (2006). Density-Based Clustering over an Evolving Data Stream with Noise. *In Proceedings of the 2006 SIAM International Conference on Data Mining.*, 328–339. https://doi.org/10.1137/1.9781611972764.29

Carnein, M. (2019). *Stream Clustering*. https://www.matthias-carnein.de/streamclustering

Carnein, M., Assenmacher, D., & Trautmann, H. (2017). An empirical comparison of stream clustering algorithms. *ACM International Conference on Computing Frontiers 2017, CF 2017*, 361–366. https://doi.org/10.1145/3075564.3078887

Carnein, M., & Trautmann, H. (2018). evoStream – Evolutionary Stream Clustering Utilizing Idle Times. *Big Data Research*, *14*, 101–111. https://doi.org/10.1016/j.bdr.2018.05.005

Carnein, M., & Trautmann, H. (2019a). Customer segmentation based on transactional data using stream clustering. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11439 LNAI*, 280–292. https://doi.org/10.1007/978-3-030-16148-4_22

Carnein, M., & Trautmann, H. (2019b). Optimizing Data Stream Representation: An Extensive Survey on Stream Clustering Algorithms. *Business & Information Systems Engineering (BISE), 61*(3), 277–297.

Carnein, M., Trautmann, H., Bifet, A., & Pfahringer, B. (2020a). confstream: Automated algorithm selection and configuration of stream clustering algorithms. *Learning and Intelligent Optimization: 14th International Conference, LION 14, Athens, Greece, May 24–28, 2020, Revised Selected Papers 14*, 80–95.

Carnein, M., Trautmann, H., Bifet, A., & Pfahringer, B. (2020b). confstream: automated algorithm selection and configuration of stream clustering algorithms. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12096 LNCS*, 80–95. https://doi.org/10.1007/978-3-030-53552-0_10

Carnein, M., Trautmann, H., Bifet, A., & Pfahringer, B. (2020c). Towards automated configuration of stream clustering algorithms. *Communications in Computer and Information Science, 1167 CCIS*, 137–143. https://doi.org/10.1007/978-3-030-43823-4_12

Chen, J., Lin, X., Xuan, Q., & Xiang, Y. (2019). FGCH: a fast and grid-based clustering algorithm for hybrid data stream. *Applied Intelligence, 49*(4), 1228–1244. https://doi.org/10.1007/s10489-018-1324-x

Chen, Y., & Tu, L. (2007). Density-Based Clustering for Real-Time Stream Data. *In Proceedings 13th ACM SIGKDD International Conference on Knowledege Discovery and Data Mining*, 133–142. https://doi.org/10.4135/9781452229669.n66

Chenaghlou, M. (2019). *Data Stream Clustering and Anomaly Detection* (Issue October). University of Melbourne.

Dang, X. H., Lee, V. C. S., Ng, W. K., & Ong, K. L. (2009). Incremental and adaptive clustering stream data over sliding window. *Database and Expert Systems Applications: 20th International Conference, DEXA 2009, Linz, Austria, August 31–September 4, 2009. Proceedings 20*, 660–674.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2*, 224–227.

de Abreu Lopes, P., & de Arruda Camargo, H. (2017). Fuzzstream: Fuzzy data stream clustering based on the online-offline framework. *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1–6.

de Andrade Silva, J., & Hruschka, E. R. (2016). A support system for clustering data streams with a variable number of clusters. *ACM Transactions on Autonomous and Adaptive Systems*, *11*(2), 1–26. https://doi.org/10.1145/2932704

Dua, D., & Graff, C. (2019). *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml

Dunn, J. C. (1973). *A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters*.

Estert, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). Density-Based Clustering Methods. *In KDD*, *96*(34), 226–231. https://doi.org/10.1016/B978-044452701-1.00067-3

Fahy, C., & Yang, S. (2019a). Dynamic Feature Selection for Clustering High Dimensional Data Streams. *IEEE Access*, *7*, 127128–127140. https://doi.org/10.1109/ACCESS.2019.2932308

Fahy, C., & Yang, S. (2019b). Finding and Tracking Multi-Density Clusters in Online Dynamic Data Streams. *IEEE Transactions on Big Data*, *20*(20), 1–15.

Fahy, C., Yang, S., & Gongora, M. (2019). Ant Colony Stream Clustering: A Fast Density Clustering Algorithm for Dynamic Data Streams. *IEEE Transactions on Cybernetics*, *49*(6), 2215–2228. https://doi.org/10.1109/TCYB.2018.2822552

Fisher, D. (1996). Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, *4*, 147–178.

Forestiero, A., Pizzuti, C., & Spezzano, G. (2013). A single pass algorithm for clustering evolving data streams based on swarm intelligence. *Data Mining and Knowledge Discovery*, *26*(1), 1–26. https://doi.org/10.1007/s10618-011-0242-x

Franke, M., & Geyer-Schulz, A. (2009). An update algorithm for restricted random walk clustering for dynamic data sets. *Advances in Data Analysis and Classification*, *3*(1), 63–92. https://doi.org/10.1007/s11634-009-0039-6

Gajowniczek, K., Bator, M., Zabkowski, T., Orlowski, A., & Loo, C. K. (2020). Simulation study on the electricity data streams time series clustering. *Energies*, *13*(924), 1–25. https://doi.org/10.3390/en13040924

Ghaemi, Z., & Farnaghi, M. (2019). A Varied Density-based Clustering Approach for Event Detection from Heterogeneous Twitter Data. *ISPRS International Journal of Geo-Information*, *8*(2). https://doi.org/10.3390/ijgi8020082

Ghesmoune, M., Lebbah, M., & Azzag, H. (2016a). A new Growing Neural Gas for clustering data streams. *Neural Networks*, *78*, 36–50. https://doi.org/10.1016/j.neunet.2016.02.003

Ghesmoune, M., Lebbah, M., & Azzag, H. (2016b). State-of-the-art on clustering data streams. *Big Data Analytics*, *1*(13), 1–27. https://doi.org/10.1186/s41044-016-0011-3

Goldstein, M., & Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLoS ONE*, *11*(4). https://doi.org/10.1371/journal.pone.0152173

Gomes, H. M., Bahri, M., & Bifet, A. (2020). *Tutorial 6: Building MOA from the source*. https://moa.cms.waikato.ac.nz

Gomes, H. M., Barddal, J. P., Enembreck, A. F., & Bifet, A. (2017). A survey on ensemble learning for data stream classification. In *ACM Computing Surveys* (Vol. 50, Issue 2, pp. 23–41). Association for Computing Machinery. https://doi.org/10.1145/3054925

Gong, S., Zhang, Y., & Yu, G. (2018). Clustering stream data by exploring the evolution of density mountain. *Proceedings of the VLDB Endowment*, *11*(4), 393–405. https://doi.org/10.1145/3164135.3164136

Guha, S., Rastogi, R., & K. Shim, R. (2000). A robust clustering algorithm for categorical attributes. *Information Systems*, *25*(5), 345–366.

Guha, S., Rastogi, R., & Shim, K. (1998). CURE: An Efficient Clustering Algorithm for Large Databases. *ACM Sigmod Record*, *27*(2), 73–84.

Han, J., Kamber, M., & Pei, J. (2012). Data mining concepts and techniques. In *The Morgan Kaufmann Series in Data Management Systems* (3rd Editio). The Morgan Kaufmann Series in Data Management Systems. https://doi.org/10.1109/ICMIRA.2013.45

Haneen, A. A., Noraziah, A., & Abd Wahab, M. H. (2018). A Review on Data Stream Classification. *Journal of Physics: Conference Series*, *1018*(1), 1–7. https://doi.org/10.1088/1742-6596/1018/1/012019

Hartigan, J. (1975). Quick Clustering Algorithms. In *Applied Statistics* (Vol. 25). https://doi.org/10.2307/2346526

Hassani, M. (2015). *Efficient Clustering of Big Data Streams*.

Hinneburg, A., & Keim, D. A. (2003). A General Approach to Clustering in Large Databases with Noise. *Knowledge and Information Systems*, *5*(4), 387–415. https://doi.org/10.1007/s10115-003-0086-9

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*, 193–218.

Hubert, L. J., & Levin, J. R. (1976). A general statistical framework for assessing categorical clustering in free recall. *Psychological Bulletin*, *83*(6), 1072.

Hyde, R., & Angelov, P. (2015). A new online clustering approach for data in arbitrary shaped clusters. *Proceedings - 2015 IEEE 2nd International Conference on Cybernetics, CYBCONF 2015*, 228–233. https://doi.org/10.1109/CYBConf.2015.7175937

Hyde, R., Angelov, P., & MacKenzie, A. R. (2017). Fully online clustering of evolving data streams into arbitrarily shaped clusters. *Information Sciences*, *382–383*, 96–114. https://doi.org/10.1016/j.ins.2016.12.004

Hyde, R. W., Angelov, P., Mackenzie, ; A R, & Nie, F. (2017). Fully Online Clustering of Evolving Data Streams into Arbitrarily Shaped Clusters. *Information Sciences*, *382*, 96–114.

Islam, M. K., Ahmed, M. M., & Zamli, K. Z. (2019a). A buffer-based online clustering for evolving data stream. *Information Sciences*, *489*, 113–135. https://doi.org/10.1016/j.ins.2019.03.022

Islam, M. K., Ahmed, M. M., & Zamli, K. Z. (2019b). I-CODAS: An improved online data stream clustering in arbitrary shaped clusters. *Engineering Letters*, *27*(4), 752–762.

Jiri Skala, I. (2012). *Algorithms for manipulating large geometric data*. University of West Bohemia.

Karypis, G., Han, E. S., & Kumar, V. (1998). CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *IEEE Computer*, *32*(8), 68–75.

Kaufman, L., & Rousseeuw, P. J. (1990). Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. In *Finding groups in data: an introduction to cluster analysis*. Wiley New York.

Khalilian, M., Mustapha, N., & Sulaiman, N. (2016). Data stream clustering by divide and conquer approach based on vector model. *Journal of Big Data*, *3*(1), 1–21. https://doi.org/10.1186/s40537-015-0036-x

Kokate, U., Deshpande, A., Mahalle, P., & Patil, P. (2018). Data Stream Clustering Techniques, Applications, and Models: Comparative Analysis and Discussion. *Big Data and Cognitive Computing*, *2*(4), 32. https://doi.org/10.3390/bdcc2040032

Kontaki, M., Gounaris, A., Papadopoulos, A. N., Tsichlas, K., & Manolopoulos, Y. (2016). Efficient and flexible algorithms for monitoring distance-based outliers over data streams. *Information Systems*, *55*, 37–53. https://doi.org/10.1016/j.is.2015.07.006

Kranen, P., Assent, I., Baldauf, C., & Seidl, T. (2011). The ClusTree: Indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, *29*(2), 249–272. https://doi.org/10.1007/s10115-010-0342-8

Kranen, P., Kremer, H., Jansen, T., Seidl, T., Bifet, A., Holmes, G., & Pfahringer, B. (2010). Clustering Performance on Evolving Data Streams: Assessing Algorithms and Evaluation Measures within MOA. *IEEE International Conference on Data Mining Workshops* , 1400–1403.

Kranen, P., Kremer, H., Jansen, T., Seidl, T., Bifet, A., Holmes, G., Pfahringer, B., & Read, J. (2012). Stream Data Mining Using the MOA Framework. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7239 LNCS*(PART 2). https://doi.org/10.1007/978-3-642-29035-0

Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., & Pfahringer, B. (2011). An effective evaluation measure for clustering on evolving data streams. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 868–876. https://doi.org/10.1145/2020408.2020555

Kuwil, F. H., Atila, Ü., Abu-Issa, R., & Murtagh, F. (2020). A novel data clustering algorithm based on gravity center methodology. *Expert Systems with Applications*, *156*. https://doi.org/10.1016/j.eswa.2020.113435

Laha, A. K., & Putatunda, S. (2018). Real time location prediction with taxi-GPS data streams. *Transportation Research Part C: Emerging Technologies*, *92*, 298–322. https://doi.org/10.1016/j.trc.2018.05.005

Lee, J., Lee, T. H., & Jun, C. H. (2019). Hybrid data stream clustering by controlling decision error. *Intelligent Data Analysis*, *23*(3), 717–732. https://doi.org/10.3233/IDA-183869

Li, G., Wang, J., Liang, J., & Yue, C. (2018). The application of a double CUSUM algorithm in industrial data stream anomaly detection. *Symmetry*, *10*(7), 1–14. https://doi.org/10.3390/sym10070264

Li, M., Croitoru, A., & Yue, S. (2020). GeoDenStream: An improved DenStream clustering method for managing entity data within geographical data streams. *Computers & Geosciences*, *144*, 104563.

Li, Y., Li, H., Wang, Z., Liu, B., Cui, J., & Fei, H. (2022). ESA-Stream: Efficient Self-Adaptive Online Data Stream Clustering. *IEEE Transactions on Knowledge and Data Engineering*, *34*(2). https://doi.org/10.1109/TKDE.2020.2990196

Lin, L., & Su, J. (2019). Anomaly detection method for sensor network data streams based on sliding window sampling and optimized clustering. *Safety Science*, *118*(February), 70–75. https://doi.org/10.1016/j.ssci.2019.04.047

Loureiro, A., Torgo, L., & Soares, C. (2005). Outlier Detection Using Clustering Methods: a Data Cleaning Application. *Proceedings of the Data Mining for Business Workshop*, 57–62.

Mansalis, S., Ntoutsi, E., Pelekis, N., & Theodoridis, Y. (2018a). An evaluation of data stream clustering algorithms. *Statistical Analysis and Data Mining*, *11*(4), 167–187. https://doi.org/10.1002/sam.11380

Mansalis, S., Ntoutsi, E., Pelekis, N., & Theodoridis, Y. (2018b). An evaluation of data stream clustering algorithms. *Statistical Analysis and Data Mining*, *11*(4), 167–187. https://doi.org/10.1002/sam.11380

McDonald, M. (2015, March 4). *Using TortoiseHg with Git*. https://mcmblog.azurewebsites.net/using-tortoisehg-with-git/

Mittal, M., Goyal, L. M., Hemanth, D. J., & Sethi, J. K. (2019). Clustering approaches for high-dimensional databases: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(3), 1–14. https://doi.org/10.1002/widm.1300

Moon, T. K. (1996). EM_tutorial. *IEEE Signal Processing Magazine*, *13*(6), 47–60.

Moshtaghi, M., Bezdek, J. C., Erfani, S. M., Leckie, C., & Bailey, J. (2019). Online Cluster Validity Indices for Performance Monitoring of Streaming Data Clustering. *International Journal of Intelligent Systems*, *34*(4), 541–563.

Ng, R. T., & Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, *14*(5), 1003–1016. https://doi.org/10.1109/TKDE.2002.1033770

O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., & Motwani, R. (2002). Streaming-Data Algorithms For High-Quality Clustering. *Proceedings - 18th Int. Conf. Data Eng*, 685–694.

Ordonez, C. (2003). Clustering binary data streams with k-means. *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 12–19.

Oussous, A., Benjelloun, F. Z., Ait Lahcen, A., & Belfkih, S. (2018). Big Data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*, *30*(4), 431–448. https://doi.org/10.1016/j.jksuci.2017.06.001

Pawar, Ms. A. D., Kalavadekar, Prof. P. N., & Tambe, Ms. S. N. (2014). A Survey on Outlier Detection Techniques for Credit Card Fraud Detection. *IOSR Journal of Computer Engineering*, *16*(2), 44–48. https://doi.org/10.9790/0661-16264448

Rastin, P. (2018). *Automatic and Adaptive Learning for Relational Data Stream Clustering*.

Rasyid, L. A., & Andayani, S. (2018). Review on Clustering Algorithms Based on Data Type: Towards the Method for Data Combined of Numeric-Fuzzy Linguistics Linguistics. *Journal of Physics: Conference Series*, *1097*(012082), 1–10.

Rathore, P. (2018). *Big Data Cluster Analysis and its Applications* (Vol. 45). https://doi.org/10.1201/9780429465185-12

Rayana, S. (2016). *Outlier Detection DataSets*. ODDS Library. http://odds.cs.stonybrook.edu

Roa, N. B., Travé-massuyès, L., & Grisales, V. (2019). A novel algorithm for dynamic clustering: properties and performance. *In 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 565–570.

Rodrigues, P. P., Gama, J., & Pedroso, P. J. (2006). ODAC: Hierarchical Clustering of Time Series Data Streams ∗. *In Proceedings of the Sixth SIAM International Conference on Data Mining*, 499–503.

Rodriguez, M. Z., Comin, C. H., Casanova, D., Bruno, O. M., Amancio, D. R., Costa, L. da F., & Rodrigues, F. A. (2019). Clustering algorithms: A comparative approach. *PLoS ONE*, *14*(1), 1–34. https://doi.org/10.1371/journal.pone.0210236

Rohlf, F. (2003). Methods of Comparing Classifications. *Annual Review of Ecology and Systematics*, *5*, 101–113. https://doi.org/10.1146/annurev.es.05.110174.000533

Rosenberg, A., & Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 410–420.

Saddam, A., Nasser, S. M., & Sundararajan, E. A. (2020). *Online Clstering of Evolving Data Streams into Arbitrary Shaped Clusters (CEDAS) using Parallel Programming*.

Sadik, S., & Gruenwald, L. (2014). Research issues in outlier detection for data streams. *ACM SIGKDD Explorations Newsletter*, *15*(1), 33–40. https://doi.org/10.1145/2594473.2594479

Sadik, S. M. (2013). *Online Detection of Outliers for Data Streams* [University of Oklahoma]. /citations?view_op=view_citation&continue=/scholar%3Fhl%3Dpt-BR%26as_sdt%3D0,5%26scilib%3D1&citilm=1&citation_for_view=wS0xi2wAAAAJ:2osOgNQ5qMEC&hl=pt-BR&oi=p

Schick, L., de Abreu Lopes, P., & de Arruda Camargo, H. (2018). D-Fuzzstream: A dispersion-based fuzzy data stream clustering. *IEEE International Conference on Fuzzy Systems*. https://doi.org/10.1109/Fuzz-Ieee.2018.8491534

Shao, H., Zhang, P., Chen, X., Li, F., & Du, G. (2019). A Hybrid and Parameter-Free Clustering Algorithm for Large Data Sets. *IEEE Access*, *7*, 24806–24818. https://doi.org/10.1109/ACCESS.2019.2900260

Shao, J., Tan, Y., Gao, L., Yang, Q., Plant, C., & Assent, I. (2019). Synchronization-based clustering on evolving data stream. *Information Sciences*, *501*, 573–587. https://doi.org/10.1016/j.ins.2018.09.035

Sharma, N., Masih, S., & Makhija, P. (2018). A Survey on Clustering Algorithms for Data Streams. *International Journal of Computer Applications*. https://doi.org/10.5120/ijca2018918014

Singh, S. (2015). *Master Thesis Spatial Temporal Analysis of Social Media Data Submitted by*. Technische Universitat Munchen.

Song, Y., Lu, J., Lu, H., & Zhang, G. (2020). Fuzzy Clustering-Based Adaptive Regression for Drifting Data Streams. *IEEE Transactions on Fuzzy Systems*, *28*(3), 544–557. https://doi.org/10.1109/TFUZZ.2019.2910714

Souza, V. M. A., dos Reis, D. M., Maletzke, A. G., & Batista, G. E. A. P. A. (2020). Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, *34*(6), 1805–1858. https://doi.org/10.1007/s10618-020-00698-5

Tareq, M., & Sundararajan, E. A. (2020). A New Density-Based Method for Clustering Data Stream Using Genetic Algorithm. *Technology Reports of Kansai University*, *62*(11), 6557–6572.

Tareq, M., & Sundararajan, E. A. (2021). An Evolving Approach to Data Streams Clustering Based on Chebychev with False Merging. *Journal of Theoretical and Applied Information Technology*, *99*(9), 1955–1965.

Tareq, M., Sundararajan, E. A., Harwood, A., & Bakar, A. A. (2022). A Systematic Review of Density Grid-Based Clustering for Data Streams. In *IEEE Access* (Vol. 10). https://doi.org/10.1109/ACCESS.2021.3134704

Tareq, M., Sundararajan, E. A., & Mohd, M. (2020). Online Clustering of Evolving Data Stream Based on adaptive Chebychev Distance. *In Proc. 281st Int. Conf. IIER*, 41–46.

Tareq, M., Sundararajan, E. A., Mohd, M., & Sani, N. S. (2020). Online Clustering of Evolving Data Streams Using a Density Grid-Based Method. *IEEE Access*, *8*, 166472–166490. https://doi.org/10.1109/access.2020.3021684

Thakkar, P., Vala, J., & Prajapati, V. (2016). Survey on Outlier Detection in Data Stream. *International Journal of Computer Applications*, *136*(2), 13–16. https://doi.org/10.5120/ijca2016908257

Togbe, M. U., Barry, M., Boly, A., Chabchoub, Y., Chiky, R., Montiel, J., & Tran, V. T. (2020). Anomaly Detection for Data Streams Based on Isolation Forest Using Scikit-Multiflow. *The 20th International Conference on Computational Science and Its Applications (ICCSA 2020), Jul 2020, Caligari, Italy. Hal-02874869v2 HAL*, 1–16. https://doi.org/10.1007/978-3-030-58811-3_2

Tran, L., Mun, M. Y., & Shahabi, C. (2020). Real-time distance-based outlier detection in data streams. *Proceedings of the VLDB Endowment*, *14*(2), 141–153. https://doi.org/10.14778/3425879.3425885

Tu, L., & Chen, Y. (2008). Stream Data Clustering Based on Grid Density and Attraction. *ACM Transactions on Computational Logic*, *1*(1), 1–26.

USP DS Repository. (n.d.).

Van Rijsbergen, C. J. (1979). Information retrieval 2nd edition butterworths. *London Available on Internet*.

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2014). OpenML: networked science in machine learning. *SIGKDD Explorations*, *15*(2), 49–60. https://doi.org/10.1145/2641190.2641198

Wang, W., Yang, J., & Muntz, R. (1997). STING: A Statistical Information Grid Approach to Spatial Data Mining. *Proceedings of the 23rd VLDB Conference, Anthens, Greece*, 186–195.

Wang, X., & Wang, L. (2018). Research on data stream clustering algorithm based on decay time window. *ACM International Conference Proceeding Series*, 1–7. https://doi.org/10.1145/3207677.3277972

Wu, B., & Wilamowski, B. M. (2017). A fast density and grid-based clustering method for data with arbitrary shapes and noise. *IEEE Transactions on Industrial Informatics*, *13*(4), 1620–1628. https://doi.org/10.1109/TII.2016.2628747

Wu, J., Xiong, H., & Chen, J. (2009). Adapting the right measures for k-means clustering. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 877–886.

Xu, B., Shen, F., & Zhao, J. (2019). A density-based competitive data stream clustering network with self-adaptive distance metric. *Neural Networks*, *110*, 141–158. https://doi.org/10.1016/j.neunet.2018.11.008

Yan, X., Razeghi-Jahromi, M., Homaifar, A., Erol, B. A., Girma, A., & Tunstel, E. (2019). A novel streaming data clustering algorithm based on fitness proportionate sharing. *IEEE Access*, *7*, 184985–185000. https://doi.org/10.1109/ACCESS.2019.2922162

Yao, H., Fu, X., Yang, Y., & Postolache, O. (2018). An incremental local outlier detection method in the data stream. *Applied Sciences (Switzerland)*, *8*(8). https://doi.org/10.3390/app8081248

Yarlagadda, A., Jonnalagedda, M., & Munaga, K. (2018). Clustering Based on Correlation Fractal Dimension Over an Evolving Data Stream. In *The International Arab Journal of Information Technology* (Vol. 15, Issue 1).

Yeoh, J. M., Caraffini, F., Homapour, E., Santucci, V., & Milani, A. (2019). A Clustering System for Dynamic Data Streams Based on Metaheuristic Optimisation. *Mathematics*, *7*(12), 1229–1252. https://doi.org/10.3390/math7121229

Youn, J., Shim, J., & Lee, S. G. (2018). Efficient Data Stream Clustering with Sliding Windows Based on Locality-Sensitive Hashing. *IEEE Access*, *6*, 63757–63776. https://doi.org/10.1109/ACCESS.2018.2877138

Zhang, C., Huang, W., Niu, T., Liu, Z., Li, G., & Cao, D. (2023). Review of Clustering Technology and Its Application in Coordinating Vehicle Subsystems. *Automotive Innovation*, *6*(1), 89–115. https://doi.org/10.1007/s42154-022-00205-0

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, *25*(2), 103–114. https://doi.org/10.1145/235968.233324

Zhao, Y., & Karypis, G. (2004). Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, *55*, 311–331.

Zhou, A., Cao, F., Yan, Y., Sha, C., & He, X. (2006). Distributed data stream clustering: A fast em-based approach. *2007 IEEE 23rd International Conference on Data Engineering*, 736–745.

Zhu, E., Zhang, Y., Wen, P., & Liu, F. (2019). Neurocomputing Fast and stable clustering analysis based on Grid-mapping K-means algorithm and new clustering validity index. *Neurocomputing*, *363*, 149–170. https://doi.org/10.1016/j.neucom.2019.07.048

Zhu, X. (2010). *Stream data mining repository*. http://www.cse.fau.edu/~xqzhu/stream.html

Zubaroğlu, A., & Atalay, V. (2019). Online embedding and clustering of data streams. *ACM International Conference Proceeding Series*, 142–146. https://doi.org/10.1145/3372454.3372481

Zubaroğlu, A., & Atalay, V. (2020). Data stream clustering: a review. *Artificial Intelligence Review*, *1*, 1–38. https://doi.org/10.1007/s10462-020-09874-x

Appendix A: Ethics Approval form

UNISA university of south africa

UNISA COLLEGE OF SCIENCE, ENGINEERING AND TECHNOLOGY'S (CSET) ETHICS REVIEW COMMITTEE

7 December 2020

ERC Reference #: 2020/CSET/SOC/020
Name: Tajudeen Akanbi Akinosho
Student #: 58300481

Dear Mr Akinosho

Decision: Ethics Approval from 7 December 2020 to 6 December 2023 (No humans involved)

Researcher: Mr Tajudeen Akanbi Akinosho
58300481@mylife.unisa.ac.za, tajuakins1@gmail.com, +2348100309779

Supervisors: Prof. Zenghui Wang
Department of Electrical and Mining Engineering, wangz@unisa.ac.za, 011 471 3513

Mr Elias Tabane
Department of Information Systems, tabane@unisa.ac.za, 011 470-2620

Working title of research:

Application of Improved Clustering Algorithms to Data Stream and Performance Evaluation

Qualification: MSc in Computing

Thank you for the application for research ethics clearance by the Unisa College of Science, Engineering and Technology's (CSET) Ethics Review Committee for the above mentioned research. Ethics approval is granted for 3 years.

*The **negligible risk application** was expedited by the College of Science, Engineering and Technology's (CSET) Ethics Review Committee on 7 December 2020 in compliance with the Unisa Policy on Research Ethics and the Standard Operating Procedure on Research Ethics Risk Assessment. The decision will be tabled at the next Committee meeting for ratification.*

The proposed research may now commence with the provisions that:

1. The researcher will ensure that the research project adheres to the relevant guidelines set out in the Unisa COVID-19 position statement on research ethics

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

attached.

2. The researcher(s) will ensure that the research project adheres to the values and principles expressed in the UNISA Policy on Research Ethics.
3. Any adverse circumstance arising in the undertaking of the research project that is relevant to the ethicality of the study should be communicated in writing to the College of Science, Engineering and Technology's (CSET) Ethics Review Committee.
4. The researcher(s) will conduct the study according to the methods and procedures set out in the approved application.
5. Any changes that can affect the study-related risks for the research participants, particularly in terms of assurances made with regards to the protection of participants' privacy and the confidentiality of the data, should be reported to the Committee in writing, accompanied by a progress report.
6. The researcher will ensure that the research project adheres to any applicable national legislation, professional codes of conduct, institutional guidelines and scientific standards relevant to the specific field of study. Adherence to the following South African legislation is important, if applicable: Protection of Personal Information Act, no 4 of 2013; Children's act no 38 of 2005 and the National Health Act, no 61 of 2003.
7. Only de-identified research data may be used for secondary research purposes in future on condition that the research objectives are similar to those of the original research. Secondary use of identifiable human research data require additional ethics clearance.
8. No field work activities may continue after the expiry date 6 December 2023. Submission of a completed research ethics progress report will constitute an application for renewal of Ethics Research Committee approval.

Note

The reference number 2020/CSET/SOC/020 should be clearly indicated on all forms of communication with the intended research participants, as well as with the Committee.

Yours sincerely,

Dr C Pilkington
Chair of School of Computing Ethics Review Subcommittee
College of Science, Engineering and Technology (CSET)
E-mail: pilkicl@unisa.ac.za
Tel: (011) 471-2130

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

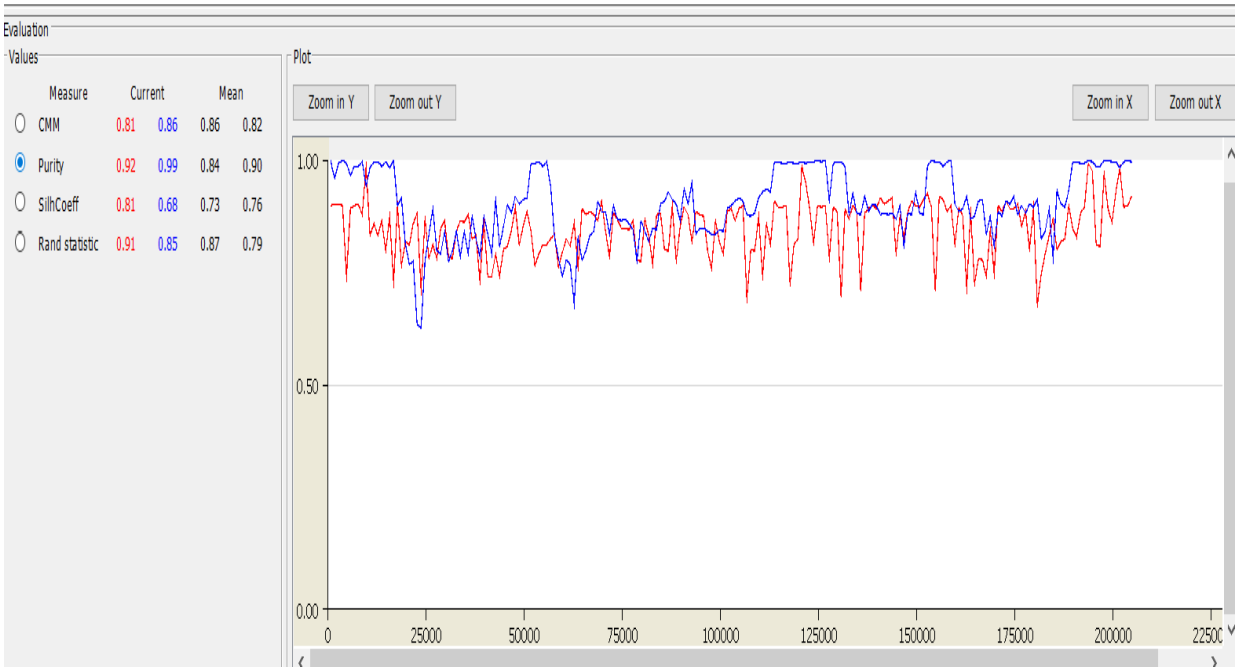
Prof. E Mnkandla
Director: School of Computing
College of Science Engineering and Technology (CSET)
E-mail: mnkane@unisa.ac.za
Tel: (011) 670 9104

Prof. B Mamba
Executive Dean
College of Science Engineering and Technology (CSET)
E-mail: mambabb@unisa.ac.za
Tel: (011) 670 9230

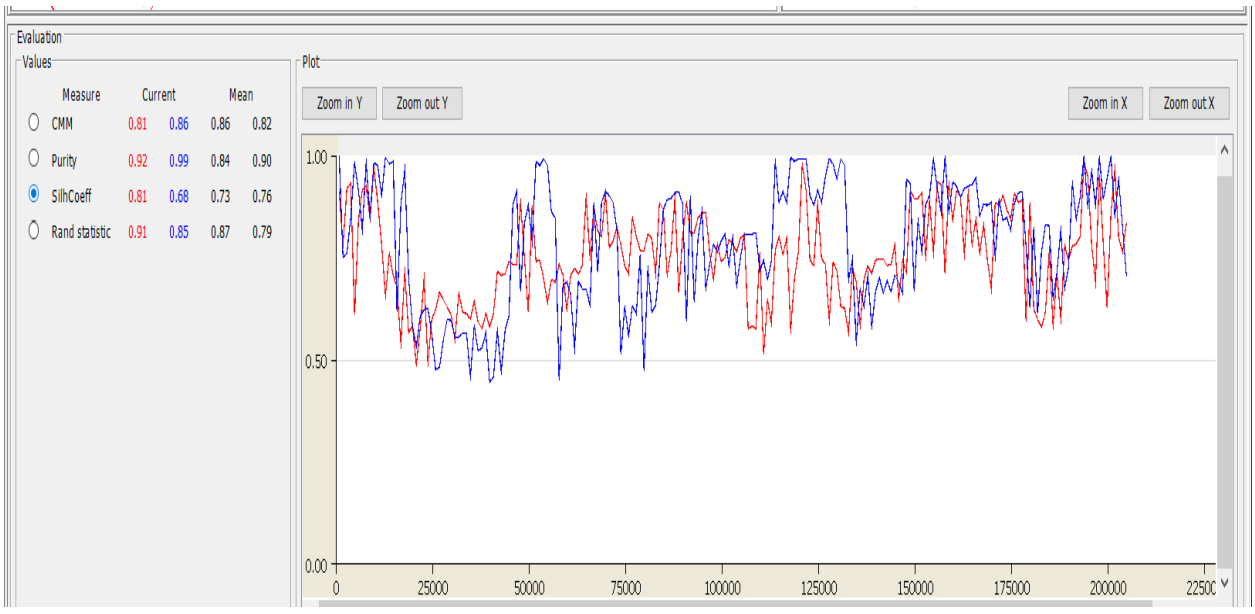
University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

Appendix B: Visualized Metrics (Purity, Silhouette Coefficient, and Rand index)

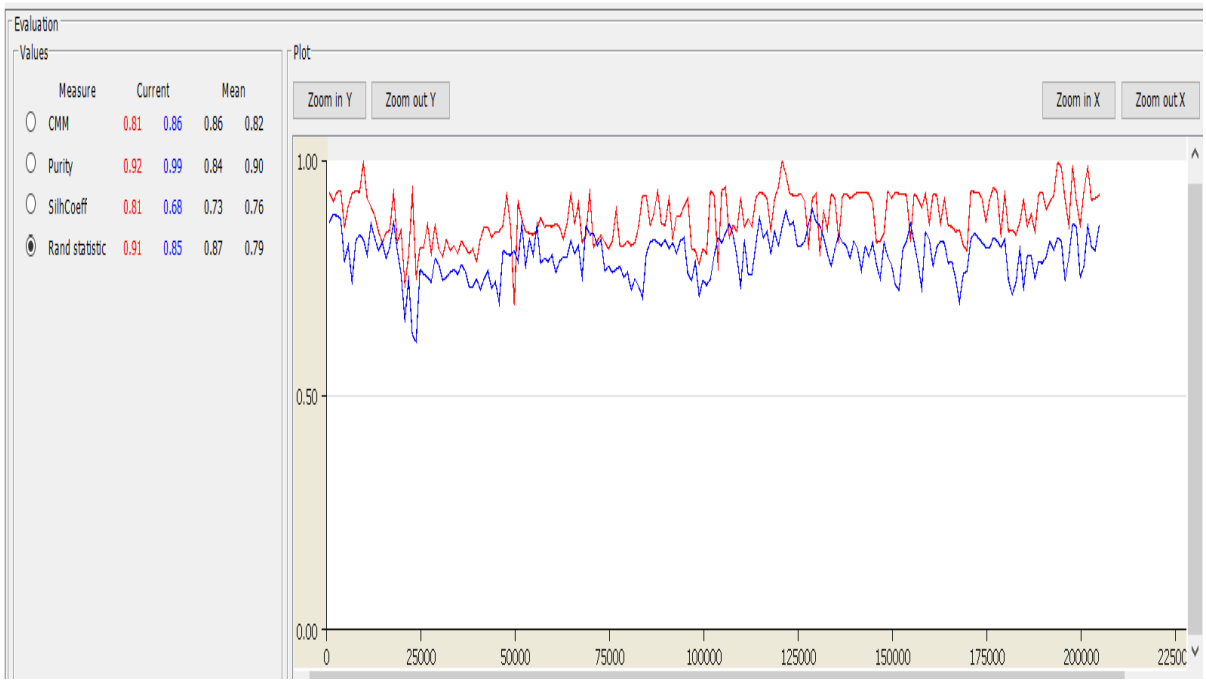
The visualization of the metric Purity on *CluStream* against *DenStream* running in the background when other selected metrics is running is presented below.



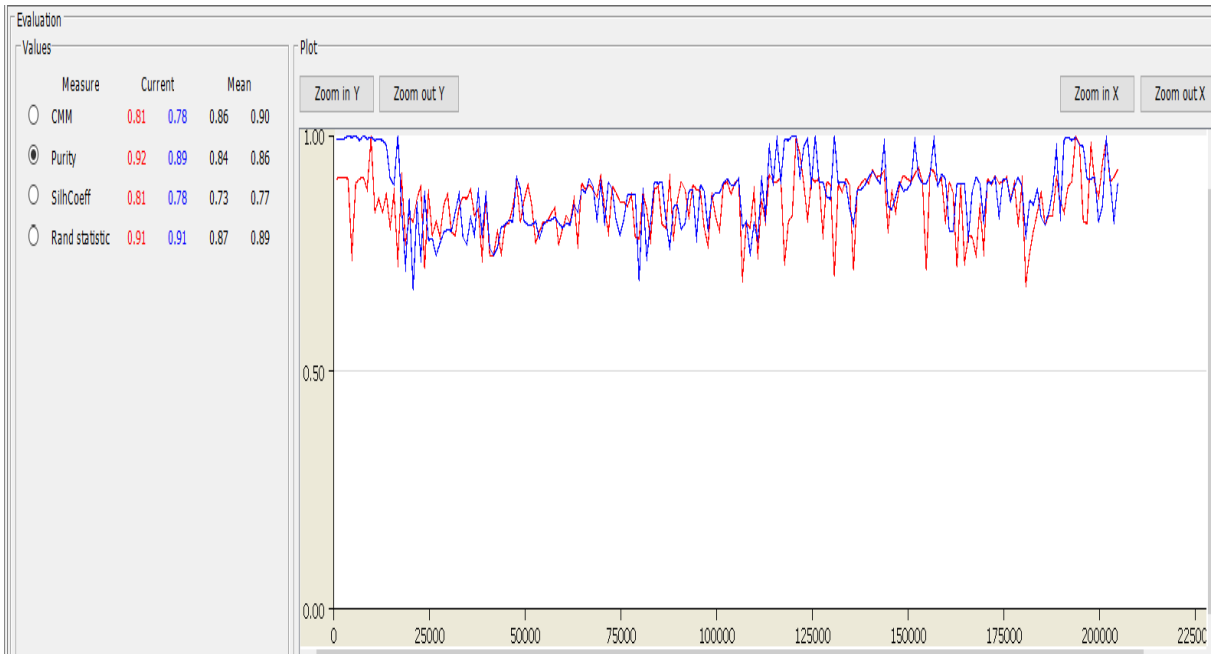
The visualization of the metric Silhouette coefficient on *CluStream* against *DenStream* running in the background when other selected metrics is running is presented below.



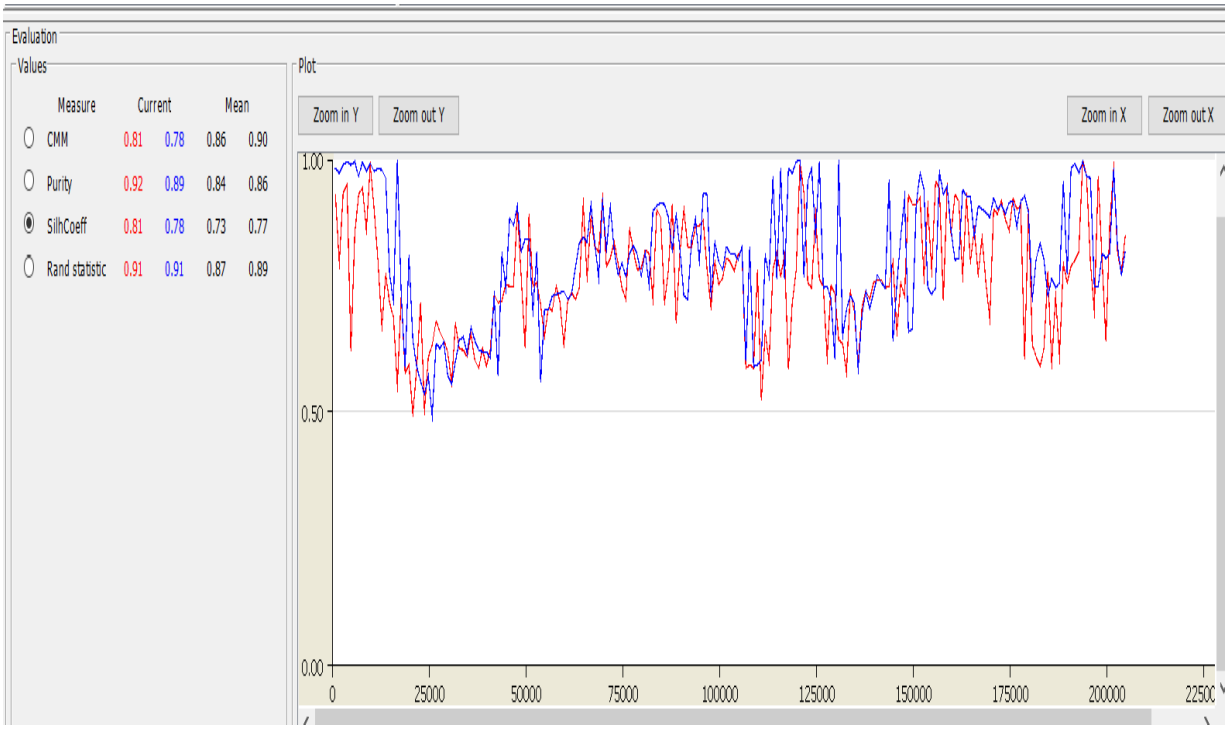
The visualization of the metric Rand index on *CluStream* against *DenStream* running in the background when other selected metrics is running is presented below.



The visualization of the metric Purity on *CluStream* against *ClusTree* running in the background when other selected metrics is running is presented below.



The visualization of the metric Silhouette coefficient on *CluStream* against *ClusTree* running in the background when other selected metrics is running is presented below.



The visualization of the metric Rand index on *CluStream* against *ClusTree* running in the background when other selected metrics is running is presented below.