# Online Semi-Supervised Learning in Non-Stationary Environments

By:
**Mobin M Idrees**

Supervisor:
**Dr Frederic Stahl**

THESIS

Submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy in the Department of Computer Science at the School of Mathematical, Physical and Computational Sciences
(SMPCS)

September 2023
Reading, UK

# ABSTRACT

Existing Data Stream Mining (DSM) algorithms assume the availability of labelled and balanced data, immediately or after some delay, to extract worthwhile knowledge from the continuous and rapid data streams. However, in many real-world applications such as Robotics, Weather Monitoring, Fraud Detection Systems, Cyber Security, and Computer Network Traffic Flow, an enormous amount of high-speed data is generated by Internet of Things sensors and real-time data on the Internet. Manual labelling of these data streams is not practical due to time consumption and the need for domain expertise. Another challenge is learning under Non-Stationary Environments (NSEs), which occurs due to changes in the data distributions in a set of input variables and/or class labels. The problem of Extreme Verification Latency (EVL) under NSEs is referred to as Initially Labelled Non-Stationary Environment (ILNSE). This is a challenging task because the learning algorithms have no access to the true class labels directly when the concept evolves. Several approaches exist that deal with NSE and EVL in isolation. However, few algorithms address both issues simultaneously. This research directly responds to ILNSE's challenge in proposing two novel algorithms "Predictor for Streaming Data with Scarce Labels" (PSDSL) and Heterogeneous Dynamic Weighted Majority (HDWM) classifier. PSDSL is an Online Semi-Supervised Learning (OSSL) method for real-time DSM and is closely related to label scarcity issues in online machine learning.

The key capabilities of PSDSL include learning from a small amount of labelled data in an incremental or online manner and being available to predict at any time. To achieve this, PSDSL utilises both labelled and unlabelled data to train the prediction models, meaning it continuously learns from incoming data and updates the model as new labelled or unlabelled data becomes available over time. Furthermore, it can predict under NSE conditions under the scarcity of class labels. PSDSL is built on top of the HDWM classifier, which preserves the diversity of the classifiers. PSDSL and HDWM can intelligently switch and adapt to the conditions. The PSDSL adapts to learning states between self-learning, micro-clustering and CGC, whichever approach is beneficial, based on the characteristics of the data stream. HDWM makes use of "seed" learners of different types in an ensemble to maintain its diversity. The ensembles are simply the combination of predictive models grouped to improve the predictive performance of a single classifier.

PSDSL is empirically evaluated against COMPOSE, $LEVEL_{IW}$, SCARGC and MClassification on benchmarks, NSE datasets as well as Massive Online Analysis (MOA) data streams and

real-world datasets. The results showed that PSDSL performed significantly better than existing approaches on most real-time data streams including randomised data instances. PSDSL performed significantly better than 'Static' i.e. the classifier is not updated after it is trained with the first examples in the data streams. When applied to MOA-generated data streams, PSDSL ranked highest (1.5) and thus performed significantly better than SCARGC, while SCARGC performed the same as the Static. PSDSL achieved better average prediction accuracies in a short time than SCARGC.

The HDWM algorithm is evaluated on artificial and real-world data streams against existing well-known approaches such as the heterogeneous WMA and the homogeneous Dynamic DWM algorithm. The results showed that HDWM performed significantly better than WMA and DWM. Also, when recurring concept drifts were present, the predictive performance of HDWM showed an improvement over DWM. In both drift and real-world streams, significance tests and post hoc comparisons found significant differences between algorithms, HDWM performed significantly better than DWM and WMA when applied to MOA data streams and 4 real-world datasets Electric, Spam, Sensor and Forest cover. The seeding mechanism and dynamic inclusion of new base learners in the HDWM algorithms benefit from the use of both forgetting and retaining the models. The algorithm also provides the independence of selecting the optimal base classifier in its ensemble depending on the problem.

A new approach, Envelope-Clustering is introduced to resolve the cluster overlap conflicts during the cluster labelling process. In this process, PSDSL transforms the centroids' information of micro-clusters into micro-instances and generates new clusters called Envelopes. The nearest envelope clusters assist the conflicted micro-clusters and successfully guide the cluster labelling process after the concept drifts in the absence of true class labels. PSDSL has been evaluated on real-world problem 'keystroke dynamics', and the results show that PSDSL achieved higher prediction accuracy (85.3%) and SCARGC (81.6%), while the Static (49.0%) significantly degrades the performance due to changes in the users typing pattern. Furthermore, the predictive accuracies of SCARGC are found highly fluctuated between (41.1% to 81.6%) based on different values of parameter 'k' (number of clusters), while PSDSL automatically determine the best values for this parameter.

# ACKNOWLEDGEMENTS

I dedicate this work to my friend Dr Hisham Al-Namar who suddenly passed away last year, whose enthusiasm for my research and encouragement was invaluable. Although he did not see the work to completion, his spirit of generosity continues to inspire me.

Firstly, I would like to express my humble gratitude to my supervisors, Dr Frederic Stahl, PhD and Professor Atta Badii, whose guidance and advice have been outstanding. I would also like to thank Frederic for his generosity, motivation and professionalism. He has shown to be a great person not only for his knowledge but also for his personality.

I wish to thank my wife, Sabeen, who has stood by me through all my travails and supported the family during much of my research. Along with her, I want to acknowledge my two sons, Orhan and Eyhab who have never known their dad as anything but a student.

I would like to thank not only my assessors Dr. Huizhi Liang and Dr. Lily Sun for their guidance and useful feedback during my confirmation of registration, but also Kristine Aldridge who has always been kind and made me feel welcomed. Also, I appreciate the co-authors of my publications especially Dr. Leandro Minku and Prof. Atta Badii during my research for their reviews and for using extracts from the papers. Special thanks to Prof. Albert Bifet, Prof. Geoff Holmes and Prof. Bernhard Pfahringer for making their Massive Online Analysis framework open source.

I also thank the Student Financial Support for giving me an award towards my costs during the COVID-19 lockdown, especially Matt Daley and Sam Young for their effort in this regard. I appreciate the highly professional team, all the technical and administrative staff of the School of Computer Science Mr. Alessandro Leidi, and Dr. Karin Whiteside for Reading's Researcher Development Programme that addressed my needs in knowledge and intellectual abilities as well as personal effectiveness.

Finally, I would like to extend my heartfelt gratitude to all my friends, especially Mr. Gulraiz Khan who always helped me whenever I was needed. Thank my parents for their countless blessings, spiritual support and sacrifice and my entire family for the good times in my life.

# ORIGINAL AUTHORSHIP

***Declaration**: I, Mobin M. Idrees, declare that this thesis titled, "Online Semi-Supervised Learning in Non-Stationary Environments" and the work presented in it is done wholly and mainly while in candidature for a research degree at the University of Reading, UK. I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.'*

Mobin M. Idrees

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ALGORITHMS

# LIST OF ABBREVIATIONS

| Abbreviation | Meaning |
|---|---|
| ACE | Adaptive Classifiers-Ensemble |
| AddExp | Addictive Base learner Ensembles |
| AWE | Accuracy Weighted Ensemble |
| BkM | Bisecting K-Means |
| BLAST | Short for best last |
| CD | Critical Difference |
| CGC | Cluster Guided Classification |
| COMPOSE | COMPacted Object Sample Extraction |
| CP | Compact Percentage |
| CSE | Core Support Extraction |
| CUSUM | Cumulative sum |
| DDD | Diversity for Dealing with Drifts |
| DDM | Drift Detection Method |
| df | Degree of Freedom |
| DSM | Data Stream Mining |
| DWM | Dynamic Weighted Majority |
| EDDM | Early Drift Detection Method |
| EVL | Extreme Verification Latency |
| FEAC | Fast Evolutionary Algorithm for Clustering |
| FN | False Negative |
| FP | False positive |
| GMM | Gaussian Mixture Models |
| GT | Ground Truth |
| HEFT | Heterogeneous Ensemble with Feature drifT |
| ILNSE | Initially Labelled Non-Stationary Environment |
| IoT | Internet of Things |
| IWLSPC | Importance weighted least squares probabilistic classifier |
| $LEVEL_{IW}$ | Learning extreme verification latency with importance weighting |

# LIST OF ABBREVIATIONS (Cont.)

| Abbreviation | Meaning |
|---|---|
| MOA | Massive Online Analysis |
| M3 | Modal Mixture Model |
| MC | Micro-Clusters |
| MC-NN | Micro-Cluster Nearest Neighbour |
| NSE | Non-Stationary Environments |
| OAUE | Online Accuracy Updated Ensemble |
| ODAC | Online Divisive-Agglomerative Clustering |
| OMRk | Ordered Multiple Runs of K-Means |
| pc' | Expected accuracy of classifier |
| PL | Pre-Labelling |
| SCARGC | Stream Classification Algorithm Guided by Clustering |
| SEA | Streaming Ensemble Algorithm |
| SS | Squared Sum |
| SSL | Semi-Supervised Learning |
| STEPD | Statistical Test of Equal Proportion to Detect concept drift |
| TN | True Negative |
| Todi | Two Online Classifiers for Learning and Detecting Concept Drift |
| TP | True positive |
| VL | Verification Latency |
| WMA | Weighted Majority Algorithm |
| 1CDT | One Class Diagonal Translation |
| 1CHT | One Class Horizontal Translation |
| 1CSurr | One Class Surrounding another Class |
| 4CE1CF | Four class expanding and one class fixed |
| 4CR | Four class rotating |
| 4CRE | Four Classes Rotating with Expansion |
| FG_2C_2D | Two Bidimensional Classes as Four Gaussians |
| UG_2C_2D | Two Bidimensional Unimodal Gaussian Classes |
| RRBF | Random Radial Basis Function |
| RTree | Random Tree |

# GLOSSARY OF VARIABLES

| Symbol | Description |
|---|---|
| $\tau = \{x_i ; y_i\}$ | Initial set of labelled instances $x \in X$; $y \in Y = \{1, ..., c\}$ |
| $\tau'$ | Predicted instances |
| $\beta$ | Factor to decrease weights, $0 \leq \beta < 1$ |
| $\rho$ | Period between base learner removal, creation and weight update |
| $\sigma \in \mathbb{R}^c$ | Sum of weighted prediction for each class |
| $\lambda$ | Local predictions from base learners |
| $\delta$ | Active Drift detection Method |
| $\mu$ | Micro-clustering algorithm such as CluStream |
| $\Phi$ | Clustering algorithm such as K-Means |
| $C^t$ | Clusters at time 't' |
| {1, ..., c} | Set of Class Labels |
| $\varepsilon$ | Set of heterogeneous base classifiers |
| $\varepsilon_{GT}$ | Set of classifiers trained on true class labels |
| $\varepsilon_{PL}$ | Classifiers train on pseudo-labels |
| A | Confidence drift threshold |
| $\Lambda$ | Class votes |
| K | Number of centroids |
| $K_{max}$ | Max limit of centroids |
| P | Purity threshold |
| $\theta$ | Pool or batch size |
| KNN | K-Nearest Neighbours clusters |
| Q | Centroids of clusters C {1...k} |

# Chapter 1 Introduction

## 1.1 Research Context and Motivation

Rapid innovations and elevated human dependency on technology are the key factors in the enormous expansion of digital data. The data over the Internet is transmitted in the form of an ordered sequence of instances called data streams. Internet of Things (IoT) devices generate data streams from monitoring sensors, smart electrical appliances, vehicles, robotics, etc., other sources include video streaming data, banking transactions and geo-location of people and the objective is to extract the worthwhile knowledge from the data streams.

Data Stream Mining (DSM) [1] has emerged in response to the generation of high-speed data streams for extracting worthwhile knowledge from the data streams. The application of DSM includes monitoring biodiversity and ecosystems [2], Internet of Robotic Things for environment exploration [3], Sentiment Analysis [4], analysing big traffic data of large cellular networks [5], Fraud-Detection Systems [6], Cyber Security [7], Human Activity Recognition [8] etc.

Unlike traditional offline learning where complete datasets are available for training. DSM applies online machine learning algorithms; ideally single-pass approaches due to the 'velocity' factor of Big Data. DSM represents the velocity of large datasets, which is one of the four aspects of "Big Data", the other three being volume, variety, and veracity [9].

For predictive analytics, supervised learning [10] [11] is beneficial when the ground truth class labels are available, however, this is not the case for real-world data streams. Unsupervised learning [12] [13] [14] does not essentially require the class labels, still the clustering algorithms identify similar data and partition it into different clusters.

Semi-Supervised Learning (SSL) is a better choice because it makes use of both labelled and unlabelled streaming data. Unlabelled data is cheaper to acquire than labelled data and can improve the learning rate [15]. For example, it has been shown to be helpful in active learning, proper learning, and co-training, see [16] [17].

Non-Stationary Environments (NSEs) is a special case of data stream mining in which the data streams evolve over time. The probability distributions of data can be seen as different concepts at different times. When new data arrives, the distribution of data may change over time, therefore the learning models need to adapt and best represent the current concept. Learning in NSEs requires the updating of predictive models to deal with changes in the underlying probability distribution [18]. A survey by [19] and [20] provide comprehensive information on NSEs adaptation methods. A comprehensive study of SSL algorithms dealing with NSEs is available in Chapter 2.

In machine learning literature, the scarcity or delay of labelled data is referred to as Verification Latency (VL) [21][23]. In another scenario, only limited labelled data is followed by completely unlabelled instances; this is referred to as Extreme Verification Latency (EVL) [22] [24] [25] [26]. DSM algorithms assume the availability of labelled data, immediately or after some delay, to update the accuracy of the classifier and update the prediction model.

Initially Labelled Non-Stationary Environment (ILNSE) [27] addresses both EVL and NSEs issues simultaneously. For example, one of the real-world problems, keystroke dynamics [135] recognizes users by their typing patterns instead of the straightforward password and login verification, the patterns are utilised as a second security layer for user authentication. Concept drifts are present in the dataset because the typing patterns of users change over time as they learn to type more quickly and accurately. In another scenario, autonomous robots [28] are initially trained inside a specific environment on labelled data. Later, they are sent to explore an unknown environment without the supervision of humans. These robots also need to adapt themselves to changing environments under the condition of verification latency i.e. scarcity of true class labels.

This research comes at the intersection of EVL and NSE, while the data stream clustering and stream classification are mutually intersected with both EVL and NSE. Figure 1.1 shows the standing of the presented work in the literature.

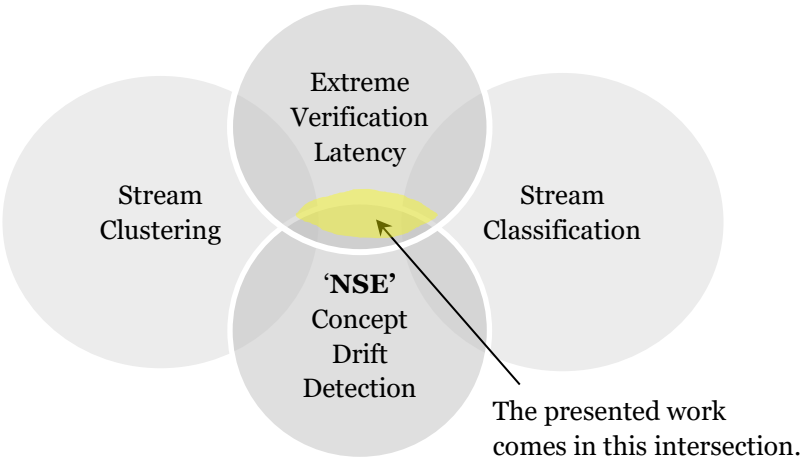


Figure 1.1 Standing of the presented work in the literature.

Stream Classification [29][30] algorithm learns from labelled data and at any time predicts/assigns class labels for unseen examples from one of the predefined classes. Classification is an important method of DSM used for decision-making on unseen examples. 'Stream Clustering' [12] [13] [14] identifies similarities between objects without the need for labels. SSL is closely related to EVL, as it relies on clustering algorithms that make use of unlabelled data to predict the pseudo-labels which are fed to the classifiers to incrementally update the prediction models, and this continues in a loop. This process is

also referred to as Cluster Guided Classification (CGC) [31]. A general approach to deal with ILNSE is to apply both clustering and classification algorithms and apply drift detection methods.

To deal with ILNSE, existing approaches rely on cluster-guided classification, which further applies centroid-based clustering or micro-clustering. Centroid-based clustering such as K-Means [33] organises the data into non-hierarchical clusters. Micro-clustering is used to store summary statistics of data points. The authors of [31] proposed a Stream Classification Algorithm Guided by Clustering (SCARGC) centroid-based clustering, COMPacted Object Sample Extraction (COMPOSE) [22] applies a geometry-based framework to learn from non-stationary streaming data, micro-cluster for Classification (MClassification) [34] applies a micro-clustering approach to classify evolving data streams with infinitely delayed labels and Learning extreme verification latency with importance weighting ($LEVEL_{IW}$) [74].

Diversity measures of predictive models have not received much research interest for evolving data streams [35]. Diversity is one of the key characteristics to consider in the formation of multiple learning algorithms called ensembles. Ensembles [10][35][36] are collections of learning models that are grouped to produce predictive performance higher than a single classifier. Ensembles are broadly classified into Homogeneous and Heterogeneous ensembles. The terms homogeneous and heterogeneous refer to the data mining algorithms used in the process, where homogeneous refers to the use of only one data mining algorithm, and heterogeneous refers to the use of different data mining algorithms [37]. Chapter 2 outlines a key issue when the diversity of ensembles is compromised due to the exclusion of base learners.

## 1.2 Research Motivation

One of the recent technological needs in DSM is finding an efficient and faster way of extracting worthwhile knowledge from the data streams. Streaming data brings new challenges to the existing state-of-the-art classification algorithms such as Naïve Bayes (NB) [38] Hoeffding Trees (HT) [39] or K-Nearest Neighbour (KNN) [40] because the ground truth class labels are not available. Hence, the algorithms need to learn under class label scarcity in a single pass through the data.

Moreover, the NSEs make the online classifiers worse over time. A streaming classifier should be able to timely detect and adapt concept drifts to best reflect the current concept at any time. Concept drift is a characteristic of data streams, it occurs due to changes in the distributions in a set of input variables and/or class labels. However, due to the small amount of labelled data, it is difficult to know which type of machine learning algorithm would be best to use as a base model beforehand. Therefore, the motivation is to

1) Develop an online SSL classifier capable of learning from both labelled and unlabelled data in NSEs.

2) Analyse the diversity of learning models and their effect on predictive performances.

3) Reduce its dependency on human-predefined parameters.

The next Section 1.3 describes the formulation of the problem, Section 1.4 research questions, Section 1.5 describes research aims and objectives, Section 1.6 describes the methodology used in the research, Section 1.7 presents contributions of the presented research, Section 1.8 provides organisation of the thesis and Section 1.9 summarises the chapter.

## 1.3 Problem Formulation

Several approaches exist that address the problems associated with NSE and EVL in isolation. However, few algorithms address both issues simultaneously. ILNSE is a challenging task because the learning algorithms have no access to the true class labels directly after the drift occurs. The existing approaches require more than one technique such as cluster-guided classification [31], self-learning [41][42] micro-clustering [43]. However, from the literature, it is not clear on what conditions one approach is better than the other and what causes other approaches to fail.

In data stream mining, the prediction models are trained under specific environments in the presence of labelled data 'x' for which class labels 'y' and probability distributions $P_t$ (x, y) and $P_t$ (x) are known. Figure 1.2 depicts this problem, the accuracy plot in the figure is virtual because the estimate of accuracy is not possible due to the scarcity of true class label after the time $t_{UE}$.

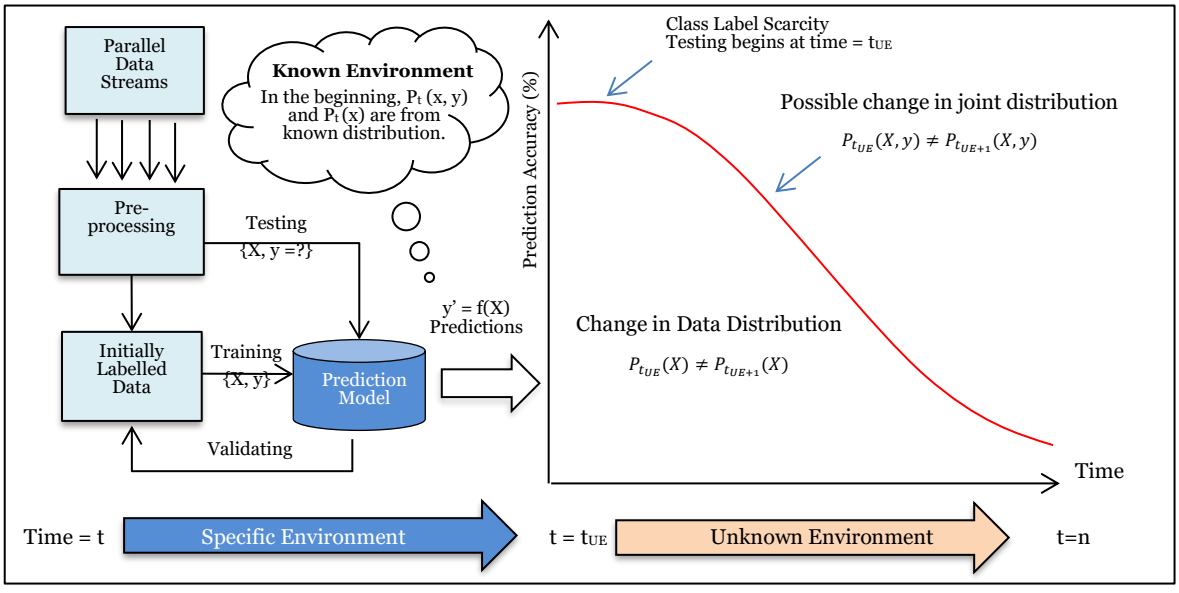


Figure 1.2 Graphical representation of the problem, labelled data trains the prediction models in a known environment and failed due to concept drifts in an unknown environment.

However, when the prediction models are deployed in a future time ($t_{UE}$) to an Unknown Environment, changes may occur in a set of input variables 'x', i.e. $P_t(x) \neq P_{t+1}(x)$. The prediction models are unable to restore learning due to the scarcity of true class labels and fail to represent the current concept. Most of the real-world data streams are continuous and infinite. Unlike offline datasets, in data streams, there is no prior information about the number of classes and this value may change in the future. Under specific conditions, the CGC algorithms could be more effective than self-learning if the data is favouring clustering, i.e. high-purity clusters. These issues make it difficult to choose the right EVL approach for different problems. Two factors influencing the predictive results of ILNSE covered in this research are hyperparameters and randomisation.

A good combination of different types of models can also sometimes lead to better predictive performance than the use of a single type of model [35]. Therefore, it would be desirable for online learning algorithms applied to NSEs not only to detect which is the best type of model maintaining the highest classification accuracies but also to use a combination of different types of models if that is found to be beneficial.

Despite using the weighting mechanism, some approaches do not exploit one of the key aspects - the use of different types of base models. Dynamic ensembles are one of the most popular ensemble approaches to deal with concept drift. However, their diversity is compromised when a poor performing base learner is removed from the ensemble, which may be deemed beneficial in future concepts. Figure 1.3 illustrates the problem of diversity in NSEs. In Concept A, three different base learners or experts are stored in a dynamic ensemble that has a feature to conditionally include or exclude a base learner. In the case of concept drift concept B occurs, if a base learner 'B' is removed from the ensemble, its diversity is compromised.

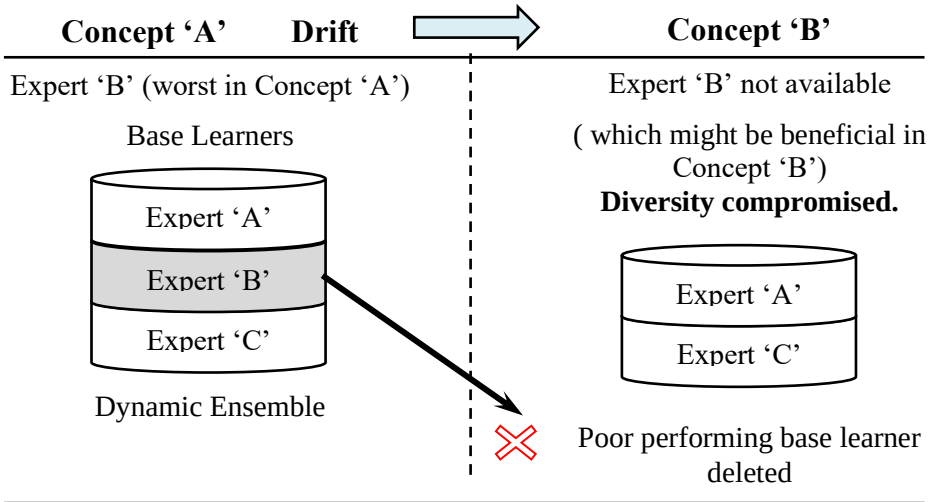


Figure 1.3 Diversity compromised in Non-Stationary Environment due to the exclusion of base learner from the ensemble.

Maintaining diversity in a dynamic online ensemble, especially when excluding one of its base classifiers, is crucial for the ensemble's adaptability to changing data distributions. Diversity in this context refers to using multiple prediction models called ensembles, that are different from each other in some way. The terminologies used here are explained in Section 2.1. Having diverse models can increase the likelihood that at least one of them will be able to adapt effectively to the changing data. The diversity of online prediction models can have a significant impact on predictive performance in NSEs. Many existing diverse ensemble techniques leverage meta-learning [36] [88], guiding the determination of which learning techniques are effective for specific types of data. Another study [89] constructed a diverse ensemble using three distinct tree-based ensembles (Random Forest [90], Rotation Forest [91], and Extremely Randomised Trees [92]). Kolter's Weighted Majority Algorithm (WMA) [11] achieves heterogeneity among its base learners, enhancing ensemble diversity. However, it can not dynamically add new base learners and lacks explicit methods to detect and handle concept drift, making it less effective in NSEs.

Hyperparameters are the parameters that must be initialised before learning begins. One of the pitfalls of the clustering algorithms is their dependency on parameters such as the number of clusters that must be specified before the learning. This parameter has a great influence on clustering results. In offline machine learning, this parameter is iteratively tuned on finite datasets. Two well-known algorithms for estimating the number of centroids 'k' in data streams are namely: Ordered Multiple Runs of K-Means (OMRk) [44] and Bisecting K-Means (BkM) [45].

Randomisation is an effect in which the training examples are shuffled, it is different to noise, as it is not a random displacement of examples. The predictive accuracy of learning models depends on the instances seen so far [46]. The predictive performance of classifiers depends on the order of instances in the dataset [47]. Apart from that in many real-world data streams the instances arrive in sequential order, which are directly fed to the online learning models. Both randomisation and hyperparameters have influences on the clustering results.

## 1.4 Research Questions

Following are the questions addressed in the presented research, which are followed by answers to the research questions and contributions to knowledge.

**Research Question 1.** How does the scarcity of true class labels impact prediction accuracy in non-stationary environments, especially when learning algorithms lack direct access to true class labels immediately following concept drifts?

To address the scarcity of class labels, existing approaches are broadly classified into three categories: cluster-guided classification, self-learning using classifiers, and micro-clustering**.** EVL in general deals with unlabelled data using clusters. However, when clusters overlap, micro-clustering is a better option, but it requires extra processing time, which is

not ideal for DSM. However, in both cluster-guided classification and micro-clustering, the new centroids receive labels from the 'nearest' centroids. It is more likely that micro-clusters receive wrong class labels due to micro-cluster overlaps. On the other hand, self-learning predicts the class labels for unlabelled data (also called pseudo-labels) and re-train the same learning models on these pseudo-labels. Wrong predictions of pseudo-labels further degrade predictive performance over time.

**Research Question 2.** What strategy should be implemented to maintain the diversity of a dynamic online ensemble when excluding one of its base classifiers?

Due to the small amount of initial labelled data, determining the most suitable machine learning algorithm is challenging. However, existing literature lacks evidence explaining why certain prediction models are beneficial or which models to be avoided immediately after the concept drift. Most of the work in online NSEs focuses on updating prediction models to quickly recover from concept drift, while little attention has been dedicated to investigating the most suitable type of predictive model at any given time.

**Research Question 3.** How can hyperparameters be autonomously optimized in Incremental Learning, considering the evolving nature of data streams and minimizing manual parameter tuning for improved classification?

Existing ILNSE approaches based on CGC rely on manual selection of the number of centroids 'k. Most clustering approaches necessitate prior knowledge of the number of classes to generate corresponding centroids. In real-world streams, which are often unstructured and noisy, the prior knowledge of the number of classes is unknown. While offline machine learning iteratively tunes this parameter on finite datasets, data streams are evolving and infinite, making the selection of an optimal value of 'k' is challenging. Current approaches manually tune this parameter by applying arbitrary values, potentially leading to poor classification results. This issue should be comprehensively addressed to reduce dependency on human feedback.

**Research Question 4.** Are existing approaches always successful when applied to different problems? What strategy should be adopted if one of the EVL approaches fails?

One approach to deal with EVL challenges involves implementing an active switching mechanism, i.e. applying all existing EVL approaches to initial labelled data and determining the most effective one for future use. Additionally, when cluster-guided or micro-clustering approaches detect concept drift due to cluster overlaps, the learning algorithm should integrate an efficient mechanism to prevent the incorrect propagation of class labels to the clusters.

## 1.5 Research Aim and Objectives

The primary aim of the research is to scrutinize existing ILNSE approaches that address non-stationary data streams, identifying gaps and limitations. The objective is to propose a new algorithm to fill these gaps while reducing human dependency on existing DSM algorithms. To achieve this aim, a few investigations, experimental studies, and empirical evaluations have been performed and involved discussing the following proposition:

"Develop an online SSL method that can make use of unlabelled data from a streaming data source in real-time with the ability to adapt to concept drifts. Analyse the impact of diversity and heterogeneity in high volume and high velocity streaming data".

Listed below are the key objectives of this research.

1. Conduct a comprehensive literature review to understand the challenging issues in the existing data stream mining algorithms.
2. Critically assess the strengths and limitations of current algorithms, particularly those addressing the scarcity of class labels.
3. Identify gaps in the literature that indicate a need for the development of a new algorithm to address these limitations effectively.
4. Develop and implement the proposed algorithm.
5. Ensure the algorithm accommodates real-time processing constraints inherent in data stream environments, addresses the need for diversity in ensembles and adaptability to non-stationary characteristics.
6. Conduct through testing and evaluation to validate the algorithm's effectiveness in comparison to existing approaches.
7. Disseminate findings through conference presentations and peer-reviewed publications.

## 1.6 Research Methodology

This section describes the nature of this research and the required methods to be applied in this research appropriately. The applied methods and procedures for data collection, validation, and evaluation techniques are also described. It further describes the automation of experiments.

The review covers the literature published in the past decade. A total of 403 articles were obtained using the keywords "Extreme Verification Latency" and/or "non-stationary". A total of 403 ACM (15), ScienceDirect (217), IEEE (70) and Scopus (185) articles were retrieved of which 108 duplicated articles were eliminated and 295 articles were shortlisted after analysing the keywords and abstracts. i.e. the abstracts which did not specify the research in the core of 'Semi-Supervised Learning' were excluded.

This research focuses on developing a new Machine Learning algorithm, in this process, several variants of the algorithms may evolve and by applying different parameters the predictive accuracies of these algorithms may change on different data streams and real-world datasets. To determine the statistically significant differences between algorithms, non-parametric tests were carried out using Demsar's methodology [49]. For the statistical test, the Friedman test was applied with $\alpha = 0.05$. The null hypothesis states that the performances of all the algorithms are similar therefore their ranks should be equal. In the experiments, the null hypothesis is, "no statistical difference between the algorithms". If the null hypothesis was rejected, the Nemenyi post hoc test [50] was applied to identify which pairs of algorithms differ from each other and were represented in boxplots.

Non-stationary datasets used in the experiments are provided by the authors of SCARGC in [31] which are available for the Machine Learning community. The artificial data streams used in the experiments are generated through the Massive Online Analysis (MOA) workbench [46]. The characteristics and configuration of these data streams are summarised in Table 4.1. The MOA commands to generate these streams are available in APPENDIX I.

A framework for fully automated evaluation of experiments has been developed in the MOA. The source code of the automation program and the scripts of the experiments are publicly available to the research community [54]. In this automation, the Friedman test and post hoc test for multiple classifiers and datasets are performed automatically. The experiments' scripts contain a list of MOA tasks which are performed in a sequential order, the classification accuracies are automatically summarised in tables, and the rankings and the box plots are automatically generated.

The following are some assumptions that were used in this thesis:

- The algorithm has no control over the sequence of the examples arriving online.
- The algorithm should be able to predict at any given time after seeing any number of examples.
- Data streams have a fixed number of features, ideally less than 500 and there are a limited number of class labels, typically less than 10.

## 1.7 Contributions of the Presented Research

As a part of this research, an 'Online Semi-Supervised Learning (OSSL) framework for DSM has been developed for MOA. The framework provides options to simultaneously run online clustering and classification algorithms. A baseline method has been developed to evaluate OSSL algorithms. Support of two training strategies namely incremental and batch incremental has been provided. A semi-supervised evaluation method has been proposed, which incrementally evaluates the pseudo-labelling accuracies. Figure 1.4 demonstrates the key features of Predictor for Streaming Data with Scarce Labels (PSDSL) and

Heterogeneous Dynamic Weighted Majority (HDWM) which has been published in the paper.

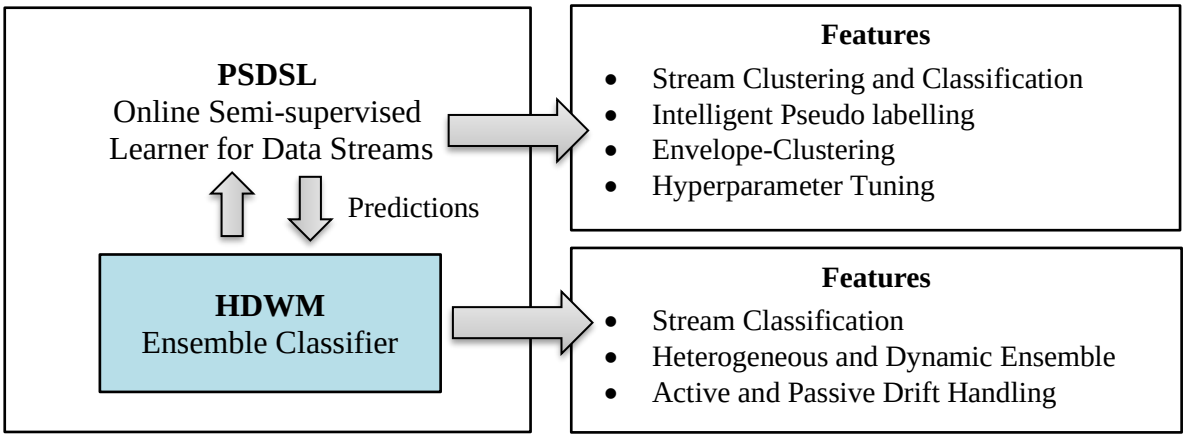


Figure 1.4 Development of algorithms and corresponding key features.

This research provides the following novel contributions:

1. An online SSL algorithm PSDSL has been proposed, PSDSL is made capable of intelligently switching between self-learning, CGC and micro-clustering strategies, based on the problem it is applied to, i.e. the different characteristics of the data streams.

2. Implemented SCARGC [31] algorithm and self-learning approach in MOA to compare it with PSDSL. The average prediction accuracy of SCARGC was found higher than COMPOSE [22], LEVEL$_{IW}$ [74], and MClassification [34].

3. Empirically evaluated PSDSL against standalone approaches namely COMPOSE, LEVEL$_{IW}$, SCARGC and MClassification on benchmarks NSE datasets [31] MOA data streams and real-world datasets.

4. To analyse the influence of diversity on predictive performance, 'Static vs. Dynamic' and 'Heterogeneous vs. Homogeneous' classifiers were studied. As a result, an HDWM classifier has been implemented which preserves the diversity of classifiers.

5. The HDWM algorithm has been evaluated on artificial and real-world data streams against existing well-known approaches such as a Weighted Majority Algorithm (WMA) [11] and a Dynamic Weighted Majority (DWM) [10]. The results show that HDWM maintained the diversity and performed significantly better than WMA in NSEs. Also, when recurring concept drifts were present.

6. Introduced auto parameter tuning mechanism to eliminate the human dependency and to determine the best value of number of centroids 'k' from initial labelled data.

7. A new approach called Envelope-Clustering has been introduced to resolve the conflict during cluster labelling and suggested a confidence measure approach to ensure the quality and correctness of labels assigned to the clusters.

8. Developed online SSL framework in MOA which enables researchers to implement online SSL for DSM, mainly in the areas of EVL and NSE.

Some of the material presented in this thesis were published in the following papers:

1. **Idrees, M. M., Stahl, F., & Badii, A. (2022)**. "Adaptive Learning with Extreme Verification Latency in Non-Stationary Environments," in IEEE Access, vol. 10, pp. 127345-127364, 2022, DOI: 10.1109/ACCESS.2022.3225225.

   This paper builds the foundation for solving the problem of extreme verification latency in non-stationary environments, an adaptive SSL learning algorithm (PSDSL) has been developed, which has been explained in Chapter 5.

2. **Lukats D., Berghöfer E., Stahl, F., Schneider J., Pieck D., Idrees, M. (2021)** et al., "Towards Concept Change Detection in Marine Ecosystems," OCEANS 2021: San Diego – Porto, 2021, pp. 1-10, DIO: 10.23919/OCEANS44145.2021.9706015.

   This paper presented the implementation of DDM and EDDM which are the drift detection methods used in the HDWM algorithm and described in Chapter 4. These methods have been applied for change detection in marine ecosystems.

3. **Idrees, M. M., Minku, L. L., Stahl, F., & Badii, A. (2020)**. A heterogeneous online learning ensemble for Non-Stationary environments. Knowledge-Based Systems, 188, 104983. DOI: 10.1016/j.knosys.2019.104983.

   The heterogeneity of online ensembles in non-stationary environments has been investigated and presented in Chapter 3. The heterogeneous algorithm (HDWM) has been developed and published in the paper above and described in Chapter 4.

## 1.8 The Organisation of the Thesis

This thesis is organised into six chapters which aim to support the foundation behind the contributions to knowledge. Figure 1.5 provides an overview of how the research questions correspond to the investigations conducted in each chapter.

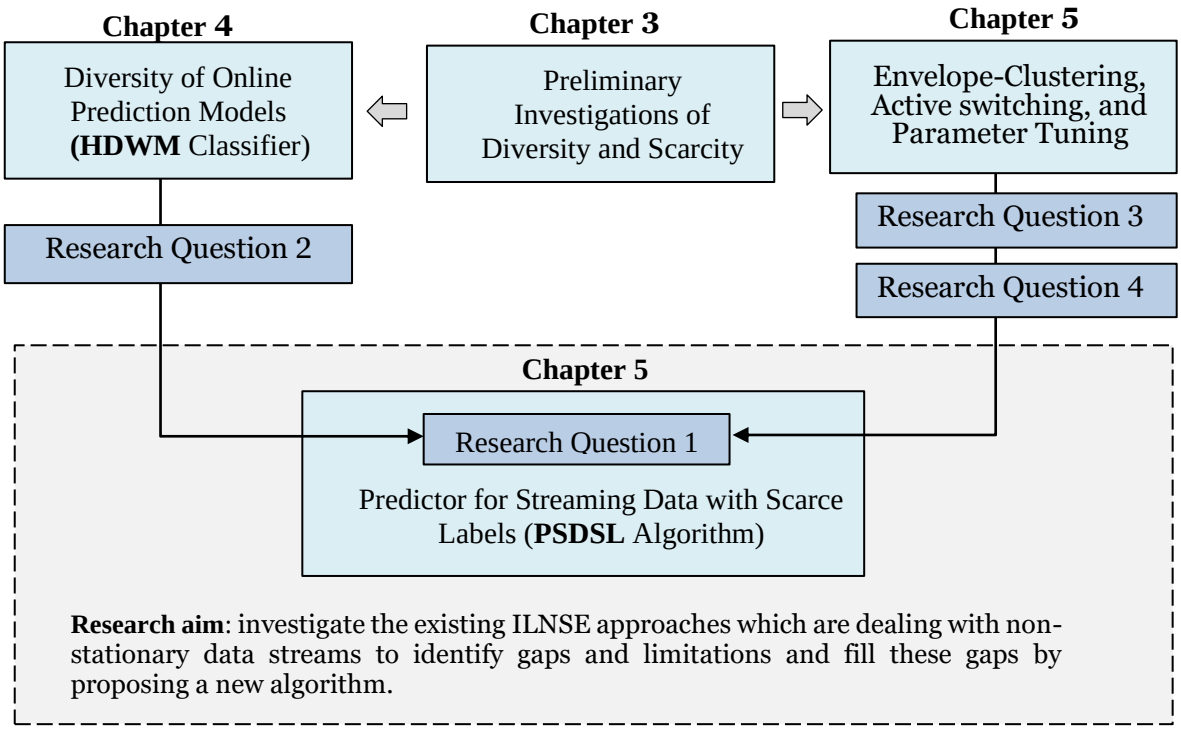


Figure 1.5 Overview of how research questions are mapped with the aim of research.

**Chapter 2** introduces the essential background of EVL and NSEs in data mining and the characteristics of each type of approach in a streaming environment. Limitations of existing verification latency algorithms for data streams were evaluated and compared with others. In this chapter, there is also a comparison between traditional evaluation methods with incremental and batch learning and evaluations in online settings.

**Chapter 3** describes the preliminary investigations with the help of real scenarios to help establish concrete evidence of the problem. To analyse the diversity of ensemble classifiers, two types of online ensemble classifiers i.e. 1) Homogeneous ensembles and 2) Heterogeneous ensembles have been investigated on artificially generated SEA [64] and STAGGER [10] datasets. The problem of class label scarcity and its effect on prediction accuracies of learning models on a real-world problem, i.e. the Keystroke dataset [135] has also been analysed in this chapter.

**Chapter 4** develops a new technique to overcome problems of existing dynamic ensembles that may undergo loss of diversity due to the exclusion of base learners. This chapter also investigates the HDWM classifier and performs experiments for evaluating it with DWM due to its dynamicity and WMA for its diversity. The HDWM has been evaluated on MOA data streams and real-world datasets. The results showed that the proposed HDWM approach maintained the ensemble diversity of classifiers.

**Chapter 5** proposes a method that deals with the scarcity of class labels under NSEs. The PSDSL algorithm was developed which is capable of intelligent data-driven adaptation for

convergence to the best pseudo-labelling strategy based on the given problem domain. This chapter also describes the new concepts of envelope clustering to resolve the conflict in transferring the labels to the clusters due to overlaps. It covers active switching and Parameter Tuning aiming to reduce the dependency of machine learning on human input have been discussed in this chapter. Finally, the chapter compares the predictive performances of PSDSL under NSEs and class label scarcity. PSDSL has been evaluated against SCARGC, $LEVEL_{IW}$, COMPOSE, and MClassification. To verify statistically significant differences between algorithms, the chapter presents experiments for the PSDSL classifier on MOA data streams and real-world datasets.

## 1.9 Summary

Existing DSM algorithms assume the availability of labelled and stationary data. However, in many real-world applications such as Robotics, Internet of Things sensors, real-time data on the Internet, Surveillance Cameras etc., high-speed big data streams are unlabelled and non-stationary. These are the key challenges faced by learning models in extracting worthwhile knowledge from the data streams. In the literature, this problem has been identified as ILNSE i.e. EVL under NSEs.

From the above requirements, it was clear that 'online SSL' is a better choice which can help in EVL scenarios as it makes use of both labelled and unlabelled data and incrementally updates the prediction models. More specifically, CGC predicts the pseudo-labels for the unlabelled data and the labels are fed to the classifiers to incrementally update the prediction models. Secondly, the prediction models need to be learned under NSEs because the data evolves and the underlying probability distributions may change over time, resulting in concept drifts. ILNSE is challenging because the learning algorithms have no access to the true class labels after the concept drifts.

The literature reveals several approaches for EVL and NSE in isolation, however, few approaches address both issues simultaneously. Major gaps were identified, i.e. it is not clear from the literature that when and under what conditions one ILNSE approach is better than the other, and what causes one approach to fail. Diversity is intuitively an important feature of meta classifiers, as there must be some variations between the predictions of the base learners. Little work has been dedicated to investigating what type of predictive model is most suitable at any given time.

The role of diversity has not been visible in the literature in the presence of concept drift. The hyperparameters have a great influence on learning algorithm results; the existing EVL approaches are rely on manual parameter selection. To accomplish the objectives, this research investigates the class label scarcity and diversity problem under NSEs. Targets to develop an 'online SSL' method for data streams with the ability to adapt to concept drifts. Validate the system in a controlled environment, apply the new approach to real-world problems and compare the results with existing approaches.

# Chapter 2 Literature Review

In this chapter, relevant studies, findings, and methodologies that contribute to the understanding of label scarcity, ensemble diversity and optimization of hyperparameters, will be examined. The literature suggests several approaches dealing Non-Stationary Environment (NSE) and Extreme Verification Latency (EVL) in isolation. However, few algorithms address both issues simultaneously. The (RQ1), which is related to the scarcity of true class labels in non-stationary conditions, the literature recognizes this issue as Initially Labelled Non-Stationary Environment (ILNSE), where the data distribution may change over time. A review of the literature shows that ILNSE approaches require more than one technique such as Cluster-guided Classification (CGC) [31], self-learning [41][42] micro-clustering [43]. However, from the literature, it is not clear on what conditions one approach is better than the other and what causes other approaches to fail.

In ILNSE scenarios, human dependency can be significant due to the challenges posed by concept drift, label scarcity, and non-stationary. As outlined in (RQ3), human analysts often need to tune these parameters to optimize the model's performance for the specific problem and data. Commonly tuned parameters might include learning rates, regularization terms, and the number of clusters. The literature suggests combining two well-known algorithms for estimating the number of clusters, OMRk [44] and BkM [45]. The concept of hyperparameters is further explained in detail in Section 2.7.

Finally, the literature delves into the question of whether existing approaches for learning in non-stationary conditions with class label scarcity are consistently successful when applied to various problems (RQ4). The literature suggests that the CGC [31] approach can help adapt to these changes by identifying clusters of similar data points and retraining or updating classifiers for each cluster when necessary.

## 2.1 Definitions and Terminology

In this research online SSL approach has been considered which focuses on techniques that leverage both the labelled and unlabelled data to build a predictive model [59]. For labelled data $\{x_L\}$ the labels are drawn from a discrete set of multi-classes $\{1 \ldots c\}$ or $y \in \{-1, 1\}$ for binary classification. The unlabelled instances $\{x_U\}$ for which class labels are not available. The labelled instance $\{x_L, y_L\}$ learns a model $y = f(x)$, so that for any example $\{x \in X\}$ it can predict correct labels 'y'.

**Definition 2.1.** A data stream is a real-time, continuous, ordered sequence of data. A data stream 'S' is represented as $S = \{x_1, x_2, x_3 \ldots x_N\}$ where $x_i$ is $i^{th}$ instance. The instance is a d-dimensional feature vector that goes to infinity.

The data is generated by different sources such as sensor networks, the internet of things, credit card transactions, network traffic data, telecommunication, stock market, satellite, weather forecasting etc.

**Definition 2.2.** EVL is a scenario in which limited labelled data is available for training and true class labels are unavailable in the future or available after some delay.

In most real-world data streams true class labels are not available and the number of classes is not known in advance. Therefore, clustering, being unsupervised, is one of the most suitable data mining and data analysis methods for data streams.

EVL relies on the SSL approach in which the data streams contain labelled instances $X_L = (x_1, \ldots, x_L)$ in the beginning for which class labels $Y_L = (y_1, \ldots, y_L) \subset Y$ are available and for unlabelled instances $X_U = (x_{L+1}, \ldots, x_U)$ the class labels are unavailable. A model trains on $X_L$ which predicts pseudo-labels for $X_U$ and updates the training Model. EVL is perhaps the most challenging case of all machine learning problems: labels for training data are never available – except perhaps those provided initially, yet the classification algorithm is asked to learn and track a drifting distribution with no access to labelled data.

**Definition 2.3.** Self-learning [35] or Active Learning [55][56] trains the classifiers on labelled data, predicts the pseudo-labels for the unlabelled data and finally re-trains the classifiers on the newly labelled data, this continues in a loop. While Active Learning decides when and which instances should be used for labelling.

**Definition 2.5.** CGC [31] is an EVL approach in which a clustering step is followed by a classification, these steps repeatedly apply in a closed-loop fashion. The clustering algorithms make use of unlabelled data to predict the pseudo-labels which are fed to the classifiers to update the prediction models.

**Definition 2.6.** Ensembles are collections of learning models that are grouped to produce predictive performance. These predictive models $\{C_i \ldots C_N\}$ learn some function 'f' to solve a particular problem. To get a prediction on example 'x' each member in the ensemble predicts $f(C_i(x) \ldots C_N(x))$, where $C_i(x)$ denotes the prediction of member $C_i$ for 'x'. In weighted voting, each member $C_i$ has a corresponding weight $w_i$. For a binary class problem, with classes in {0, 1}, weighted voting is calculated using Equation 2.1.

$$\text{Prediction}(x) = \begin{cases} 1 & \text{if } \sum_i w_i \cdot C_i(x) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Where $\theta$ is a user-provided value for splitting the classes.

Literature shows that in terms of diversity, the ensembles are broadly classified into homogeneous and heterogeneous, taking into consideration the drift handling capabilities

the ensembles are further categorised into active and passive ensembles. A summarised information about these categories is available in Section 2.4.2.

**Definition 2.7.** Concept drift is a characteristic of the data stream, which describes that the underlying distribution of data may change and evolve over time either due to the change in the target concepts (classes of examples) and/or change in attribute distributions [60].
Surveys by [19] and [20] provide comprehensive information on NSEs and concept drift adaptation methods. Changes in distributions occur in a set of input variables 'x' and/or class labels 'y', i.e. $P_t(x, y) \neq P_{t+1}(x, y)$ at the time 't'. Two different types of concept drifts exist,

1. "Virtual drift" in which only the distribution of input data 'x' changes, i.e. $P_t(x) \neq P_{t+1}(x)$ and does not affect the class labels, i.e. $P_t(y|x) = P_{t+1}(y|x)$.
2. "Real drift" [20] refers to changes in class labels due to changes in the distribution $P(y|x)$.

Several approaches available in the literature for concept drift detection and adaptation exist, e.g. [20] [60] [61] [62] [63]. However, existing drift detection approaches rely on true class labels, taking into consideration that true labels are not available in EVL scenarios, the drift handling in EVL is extremely challenging. The next section summarises SSL approaches dealing with EVL and NSEs.

## 2.2 Learning Strategies in Data Streams

Existing work on learning in NSEs can also be divided into online and block-based approaches [48] also called chunk-based or batches. This research focuses on ensemble approaches, which can be further divided into three categories: online ensembles, which learn incrementally one example at a time, block-based ensembles, which process blocks of data and a combination of online and block-based ensembles that combine both these approaches. Online and block-based are described in Section 2.2.1 and 2.2.2.

### 2.2.1 Online and Incremental Learning

In online stream learning tasks, the training examples with corresponding class labels are presented to the learning algorithms to train the hypothesis $H: X \rightarrow Y$ and predict a class label for a given input at any time-step 't'. Online approaches process each new training instance separately and then discard it.

### 2.2.2 Block-Based Learning

The data stream 'S' is partitioned into evenly sized blocks $\{B_1 \ldots B_n\}$ where each block contains equal examples. Block-based approaches wait for a whole new chunk or batch of data to arrive; and then use this new chunk for training before discarding it. The first block-based ensemble was the Streaming Ensemble Algorithm (SEA) [64], which applies a

heuristic replacement strategy based on accuracy and diversity in which the worst classifiers are replaced with a new classifier trained on the most recent examples. Wang et al. proposed Accuracy Weighted Ensemble (AWE) [65] that processes a stream in chunks to build a new classifier for each new chunk. Block-based approaches do not react to sudden changes sufficiently quickly, while ensemble approaches that process streams incrementally, do not take advantage of periodical adaptation mechanisms [48].

### 2.2.3 Hybrid Learning

A hybrid online and block-based approach was proposed by Nishida with the Adaptive Classifier Ensemble (ACE) [66] their approach aims at tracking the error rate of a single classifier with each incoming example.

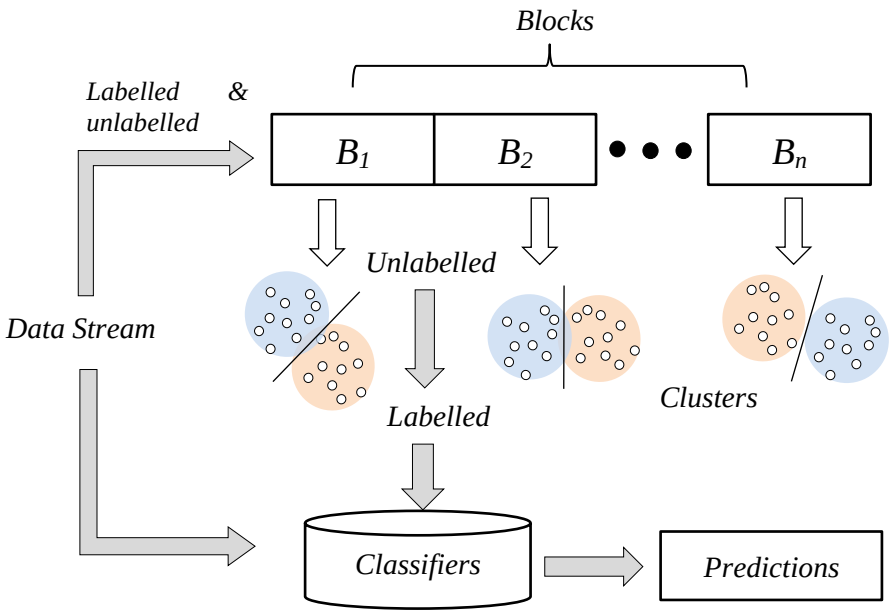


Figure 2.1 Online block-based learning in data streams, formation of clusters in blocks to retrieve class labels for unlabelled data and updating the classifier.

As shown in Figure 2.1 the incoming data stream is divided into the same sized blocks $\{B_1,....B_n\}$, the instances in each block form clusters, the centroids are continuously updated and the unlabelled data receives the labels of the nearest centroids. In parallel, using prequential evaluation each instance is used for predictions before training the classifiers.

### 2.2.4 Windowing Approaches

There are three commonly used time window models in data streams [67]: (a) Sliding windows; (b) Damped windows; and (c) Landmark windows, these are described below.

#### a) Sliding window

In this approach, only the most recent examples are stored in a fixed or variable size window 'w' at time 't' such that the points within the window have a weight of '1', for the rest the weight is '0'. It applies the first in, first out rule to store and discard the examples

from the window. ADWIN [68] change detector uses sliding windows whose size, instead of being fixed a priori, is recomputed online according to the rate of change observed from the data in the window itself. SWClustering [69] was compared with CluStream [70]. In their investigations, the CluStream failed to capture the precise distribution of recent records because old records had a great influence on the formation of the micro-cluster.

#### b) Damped window

The most recent examples are prioritised by assigning weights to objects from the data stream [71]. More recent examples receive a higher weight than older examples, and the weights of the examples decrease over time. It uses decay functions such as $f(t) = 2^{-\lambda t}$ where 't' is the time elapsed and λ is the rate of decay.

#### c) Landmark window

Some definitions state that the landmark window considers the data in the data stream from the beginning until now and all the instances have an equal weight '1'. The landmark window model splits data streams into fixed-size, non-overlapping chunks and maintains the tuples that arrive after the landmark [72]. This approach is usually applied when periodic results are needed (e.g. yearly, monthly, quarterly). When a new window is set, then the previous window including its data instances is deleted [73].

### 2.2.5 Discussion

The hybrid learning approach is more beneficial in online SSL as the instances arrive at a high rate and the prediction models need to predict the class labels in an online manner. The pool or window in which the examples are temporarily stored is used for clustering. This technique is useful when the learning models have tight time restrictions, and the models have no time to read previous examples. On the other hand, the window models approach directly impacts the prediction accuracies of the model as they are aligned with the evaluation of the learning models, these approaches are also used for detecting the concept drifts. While the block-based and online learning strategies focus on structuring the examples. The performance of block-based algorithms heavily depends on the size of the data blocks. Larger blocks can produce more accurate classifiers and report many concepts drift in each block. On the other hand, smaller blocks can miss a concept drift, and these usually produce less accurate classifiers.

## 2.3 SSL Approaches for NSEs

This section presents a comprehensive study of online SSL algorithms that can learn from data streams under NSEs and EVL. As shown in Figure 2.2, EVL is a sub-problem of SSL which is handled using Ensemble-based, Active-learning and Graph-based approaches.

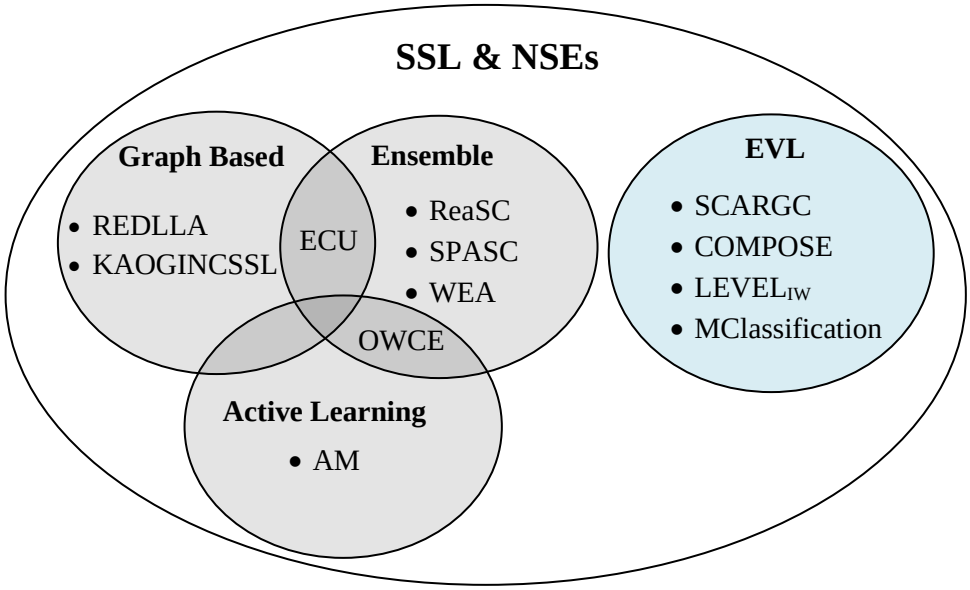


Figure 2.2 SSL Approaches dealing with NSEs, showing graph-based, ensemble and active learning approaches and the algorithms dealing with EVL.

### 2.3.1 EVL Approaches for NSEs

A general approach to deal with ILNSE is to apply both clustering and classification algorithms and apply active or passive drift detection methods. CGC is one of the approaches in which clusters and micro-clustering predict the pseudo-labels, which further update the classifiers. Table 2.1 shows approaches dealing with ILNSE.

Table 2.1 Algorithms dealing with ILNSE.

| Algorithm | Description | Drift Handling |
|---|---|---|
| SCARGC [31] | CGC, K-Means,<br>k-Nearest Neighbours (k-NN) | Gradual |
| LEVEL$_{IW}$ [74] | Semi-supervised<br>Pool and Accuracy based | Recurrent |
| COMPOSE [22] | CGC, K-Means and<br>Gaussian Mixture Models (GMM) | Gradual |
| MClassification [34] | Micro-clustering | Gradual |

#### a) SCARGC Algorithm

SCARGC [31] applies K-Nearest Neighbour to build the classification models. The algorithm stores instances in batches or a pool. The initial classification is trained using labelled instances 'T' and predicts the pseudo-labels for the unlabelled instances and stores them in the pool. When the pool size reaches 'θ' which is a user-provided value, and the clusters are formed, new centroids receive their labels from previous centroids. The new

centroids are used for the prediction of new class labels for the pool data. As shown in Algorithm 2.1 it follows a closed loop by switching between clustering and classification. The algorithm starts building an initial classifier using the available labelled data with 'c' classes. The initial labelled data is divided into k ≥ c clusters. If k = c, it uses the 'c' classes as initial clusters. If k > c, it runs a clustering algorithm, and associates each cluster to one class. The algorithm uses a simple centroid similarity calculation between the current and previous centroids using Euclidean distance and the labels are obtained by the simple nearest neighbour algorithm.

### b) MClassification

MClassification [34] uses the concept of micro-clusters. The centre and radius are calculated using linear (LS) and squared sum (SS) of N data points. As shown in Algorithm 2.2, micro-clusters are generated for initially labelled examples, and new unlabelled examples get labels from the nearest clusters using the Euclidean distance. The new data points absorb and increase in the radius and centroids. If the radius is increased from the threshold set by the user, it creates a new micro-cluster, and this process repeats in a loop for each newly received unlabelled example. The centroid and radius are computed using Equation 2.2 and 2.3.

$$\text{Centroid} = \frac{\overrightarrow{LS}}{N} \tag{2.2}$$

$$\text{Radius} = \sqrt{\frac{\overrightarrow{SS}}{N} + (\frac{\overrightarrow{LS}}{N})^2} \tag{2.3}$$

Where:

- $\overrightarrow{LS} = \sum_1^N \vec{x}_i$ is the linear sum in N data points.
- $\overrightarrow{SS} = \sum_1^N (\vec{x}_i)^2$ is the squared sum in N data points.
- N = number of data points.
- y = class label for a set of data points.

New data points are absorbed in the existing micro-clusters, and this results in an increase in the radius and centroids. If the radius is increased from the threshold set by the user, it creates a new micro-cluster, and this process repeats in a loop for each newly received unlabelled data point. For example, a new data point $\vec{x}$ can be absorbed in $MC_A = (N_A, \overrightarrow{LS_A}, \overrightarrow{SS_A})$ updating the summary statics in the following way:

$$\overrightarrow{LS_A} \leftarrow \overrightarrow{LS_A} + \vec{x}$$

$$\overrightarrow{SS_A} \leftarrow \overrightarrow{SS_A} + (\vec{x})^2$$

$$N_A \leftarrow N_A + 1$$

Similarly, when merging two disjoint micro-clusters $MC_A$ and $MC_B$ the union of these two clusters is equal to the sum of its parts and the sufficient statistics is calculated as:

$$\overrightarrow{LS}_C \leftarrow \overrightarrow{LS}_A + \overrightarrow{LS}_B$$
$$\overrightarrow{SS}_C \leftarrow \overrightarrow{SS}_A + \overrightarrow{SS}_B$$
$$N_C \leftarrow N_A + N_B$$

Micro-clusters are generated for initially labelled examples, and new unlabelled data instances are accorded their respective labels from the nearest clusters based on the Euclidean distance. In this way, new data points are absorbed, and this results in an increase in the radius and centroids. If the radius is increased from the threshold set by the user, it creates a new micro-cluster, and this process repeats in a loop for each newly received unlabelled example.

### c) COMPOSE

COMPOSE [22] also addresses the EVL problem outlined in Algorithm 2.3. Initially, the labelled instances build a base classifier, either the Gaussian mixture model or KNN to obtain a hypothesis and predict class labels. The GMM can determine the probability which tells us the degree of association among data points and closest clusters. It then constructs the α-shape (density estimation) using Compaction Percentage (CP) and assigns the labels that typically lie in the centre of the feature space for each class. The Core Support Extraction (CSE) extracts those newly labelled data drawn from the centre region of the current distribution.

### d) LEVEL$_{IW}$

This approach suggested by [74] relies on the importance-weighted least-squares probabilistic classifier IWLSPC [75], which predicts the labels for the unlabelled test data. The pseudo-code and details of this approach are described below and summarised in Algorithm 2.4. To predict the labels for the unlabelled test data the algorithm takes four parameters. 1) The training data at the current time step, 2) the corresponding label 3) the unlabelled test data at the current time step, and 4) the kernel width value σ which decides the feature space to which training data will be mapped. The algorithm then follows a closed loop.

### e) APT

Arbitrary Sub-Population Tracker algorithm (APT) [76] applies maximization (EM) algorithm for one-to-one assignment between labelled instances in time-step 't' and unlabelled instances in time-step 't + 1'. As shown in Algorithm 2.5, the EM algorithm predicts which examples are most likely to correspond to a given sub-population, and the algorithm determines which drift parameters maximise the expectation. Finally, the classifier is updated to reflect the population parameters of the newly received data.

**Algorithm 2.1 SCARGC Algorithm (T, K, θ)** [31]

**Input**: Initial training data T; Data Stream DS;
θ pool size; Number of clusters k

**Output**: Updated classifier Φ; Label y for each x ∈ DS

```
1  Φ ← buildClassifier(T )
2  if k = c then
3     C ← actualclasses(T )
4  else
5     C ← kMean(T',k )
6  end if
7  labels ← ∅
8  while DS has events do
9     x ← next event(DS)
10    y ← Φ(x)
11
12    pool ← pool ∪ {x}
13    labels ← labels ∪ {y}
      if |pool| = θ then
14       if countExamplesPerClass(labels, c) ≥ 10 then
15          C ← clustering(pool, k)
16          M ← matching(C, labels)
17          if concordance(M) ≠ 1 then
18             T ′ ← label_data(pool, C)
19             Φ ← buildClassifier(T ′ )
20             pool ← ∅
21             labels ← ∅
```

**Algorithm 2.2 MClassification Algorithm** [34]

**Input**: Maximum micro-cluster radius 'r'

1 Receive initial labelled data $D_{init} = \{x_i ; y_i\}$ ; i = 1, ..., T ; x ∈ X; y∈Y ={1, ..., c}

2 Build T micro-clusters as $MC_i = (N_i , LS_i , SS_i , y_i)$; i= 1...,T where N = number of data points ; $LS = \Sigma_{j=1}^{N}$ ; $SS = \Sigma_{j=1}^{N} (x_j)^2$

3 Calculate sufficient statistics of each micro-cluster using Eq. 2.2 and 2.3

4 Receive unlabelled data $U^t = \{ x_u^t \in X , u = 1...,N\}$

5 Measure distance between $x_t$ and each micro-cluster centroids $centroid_i$ ; i = {1... T} i.e. Dist ($centroid_i$, $x_t$) to find the closest micro-cluster, say $MC_N$, where Dist represents the Euclidean distance.

6 Assign label of $MC_N$ i.e. $y_t$ to classify example $x_t$

7 Add example $x_t$ to $MC_N$ and compute its sufficient statistics $radius_N$ ; and $centroid_N$

8 **if** $radius_N > r$ **then**

9 Create a new micro-cluster for example $x_t$ say $MC'_N = (N'_N , LS'_N , SS'_N , y_t)$

10 **else**

11 Add example $x_t$ to $MC_N$ and update its statistics as $(LS_N) \leftarrow (LS_N) + x_t$; $(SS_N) \leftarrow (SS_N) + (x_t)^2$ $N_N \leftarrow N_N + 1$

12 **end if**

13 Go to step 4 and repeat

**Algorithm 2.3 COMPOSE Algorithm** [22]

**Input**: SSL algorithm – SSL with relevant free parameters.
CSE algorithm – CSE; α-shape detail level-α Compaction percentage – CP

Receive initial labelled data $D_{init} = \{x_i ; y_i\}$ ; i = 1, ..., M ; x ∈ X; y ∈ Y = {1, ..., c}
Set $L^0 = \{ x_i^t \}$ // initial instances
Set $Y^0 = \{ y_i^t \}$ // corresponding labels of initial instances
**for** t = 0, 1,.... **do**
    Receive unlabelled data $U^t = \{ x_u^t \in X , u = 1...,N\}$
    Run **SSL** with $L^t$, $Y^t$, and $U^t$ to obtain hypothesis, $h^t : X \rightarrow Y$
    Let $D^t = \{( x_l^t , y_l^t ) \cup (x_u^t , h^t(x_u^t))\}$
    Set $L^{t+1} = \emptyset$, $Y^{t+1} = \emptyset$
    **for** each class c = 1, 2, ...., C **do**
        Run **CSE** with CP, α and $D_c^t$ to extract core supports, $CS_c$
        Add core supports to labelled data $L^{t+1} = L^{t+1} \cup CS_c$
        $Y^{t+1} = Y^{t+1} \cup \{y_u : u \in [|CS_c|], y = c\}$
    **end for**
**end for**

**Algorithm 2.4 LEVEL$_{IW}$ Algorithm** [74]

**Inputs:** Importance weighted least squares probabilistic classifier – IWLSPC; Kernel width value σ

At t = 0, receive initial data x ∈ X and the corresponding labels y ∈ Y = 1, . . . , C
Set $x_{te}^{t=0} = x$
Set $y_{te}^{t=0} = y$

**for** t = 1, ...., do
    Receive new unlabelled test data $x_{te}^t \in X$
    Set $x_{tr}^t = x_{te}^{t-1}$
    Set $y_{tr}^t = y_{te}^{t-1}$
    Call IWLSPC $x_{tr}^t, x_{te}^t, y_{tr}^t$ and σ to estimate $y_{te}^t$
**end for**

**Algorithm 2.5 APT Algorithm** [76]

**Inputs**: Initial labelled data $D_{init}$; A clustering algorithm with its own free parameters; a suitable bandwidth matrices calculation algorithm; a suitable Expectation-Maximization (EM) algorithm with its free parameters.

Receive M training examples form $D_{init} = \{x_i ; y_i\}$; i = 1, ..., M ; x ∈ X; y ∈ {1, ..., c}
Run clustering algorithm to partition the data into 'K' disjoint subsets and associate each cluster to one class among 'c' classes.
Estimate the conditional feature distribution of the data.

4 Receive new unlabelled instances $U_t = \{x_u^t \in X, u = 1, ..., N\}$ and assume N = M to associate each new instance to one previous example.

5 Compute instance-to-exemplar correspondence by maximizing the likelihood using EM algorithm.

6 Pass the cluster assignment from the example to their assigned instances to achieve instance-to-cluster assignment.

7 Pass the class of an example $x_i$ i.e. $y_i$ to the class of its assigned instance.

8 Go to step 2 and Repeat.

### 2.3.2 Other SSL Approaches for NSEs

Realistic Stream Classifier (ReaSC) [30], Semi-supervised Pool and Accuracy Stream Classification (SPASC) [77] and Weight Estimation Algorithm (WEA) [78] handle concept drifts by updating the ensemble. Active Mining (AM) [55] is an Active Learning approach that applies a decision tree for partially labelled data to handle gradual drifts. Table 2.2 summarises SSL approaches dealing with NSEs.

Table 2.2 SSL approaches dealing with NSEs.

| Algorithm | Description | Category | Drift Handling |
|---|---|---|---|
| ReaSC [30] | K-Means Micro-Clustering and K Nearest Neighbour | Ensemble | Gradual |
| SPASC [77] | Semi-supervised Pool and Accuracy based | Ensemble | Recurrent |
| WEA [78] | K-Means and GMM | Ensemble | Gradual |
| AM [55] | Demand-driven Active Mining of data streams, Decision Tree | Active Learning | Gradual |
| OWCE [56] | Minimum-variance, Optimal Weight Classifier | Active Learning, Ensemble | Gradual |
| REDLLA [79] | K-Means to spread labels at the leaves of a tree. | Graph-based | Recurrent |
| ECU [57] | Graph Based, Weighted Ensemble Classification and Clustering | Graph-based Ensemble | Abrupt |
| KAOGINCSSL [80] | Graph based SSL | Graph-Based | Abrupt |

Optimal Weight Classifier Ensemble (OWCE) [56] applies both Ensemble-based and active learning. REDLLA (Recurring Concept Drifts and Limited Labelled data) [79] applies a

decision tree of clusters for tracking the recurring concept drifts. ECU (Ensemble Classification and Clustering) [57] is a graph-based ensemble in which prediction for new instances is achieved using voting from both classifiers and clusters. KAOGINCSSL [80] is a graph-based classifier that learns from partially labelled instances.

### 2.3.3 Comparative Analysis of Existing EVL approaches

In a comparative analysis of existing EVL learning algorithms, which was presented in [81], the authors compared the classification accuracies for all three versions of COMPOSE with SCARGC, MClassification and $LEVEL_{IW}$ on 15 datasets. As shown in Table 2.3 none of these algorithms showed significant differences among them. The ranking for these algorithms is shown in the parenthesis, whereas the lower ranks represent better algorithms.

Table 2.3 Average classification accuracy of COMPOSE, SCARGC, MClassification and $LEVEL_{IW}$ presented in [81].

| **DATASETS** | **COMPOSE (α-shape)** | **COMPOSE (GMM)** | **FAST COMPOSE** | **SCARGC (1-NN)** | **SCARGC (SVM)** | **MClassification** | **$LEVEL_{IW}$** |
|---|---|---|---|---|---|---|---|
| 1CDT | 99.9(2) | 99.8(5) | 99.9(1) | 99.6(7) | 99.7(6) | 99.8(4) | 99.9(3) |
| 1CHT | 99.6(2) | 99.3(6) | 99.5(3) | 99.6(1) | 99.2(7) | 99.3(5) | 99.5(4) |
| 1Csurr | 90.9(5) | 89.7(6) | 95.6(1) | 94.5(3) | 94.9(2) | 85.1(7) | 91.3(4) |
| 2CDT | 96.5(1) | 95.9(2) | 95.1(4) | 87.7(6) | 87.8(5) | 95.2(3) | 58.3(7) |
| 2CHT | 90.3(1) | 89.6(2) | 89.4(3) | 83.6(5) | 83.4(6) | 87.9(4) | 52.1(7) |
| 4CE1CF | 93.9(5) | 93.9(6) | 93.9(4) | 94.0(3) | 92.8(7) | 94.4(2) | 97.7(1) |
| 4CR | 99.9(2.5) | 99.9(2.5) | 99.9(2.5) | 99.9(6) | 98.9(7) | 99.9(5) | 99.9(2.5) |
| 4CRE-V2 | 92.5(1) | 92.3(3) | 92.4(2) | 91.3(6) | 91.4(5) | 91.6(4) | 24.1(7) |
| FG_2C_2D | 87.9(6) | 95.5(5) | 95.5(3) | 95.5(4) | 95.6(2) | 62.5(7) | 95.7(1) |
| GEARS_2C_2D | 90.9(7) | 95.8(3) | 91.2(6) | 95.9(2) | 95.8(4) | 94.7(5) | 97.7(1) |
| MG_2C_2D | 93.1(2) | 93.2(1) | 93.0(3) | 92.9(5) | 92.9(4) | 80.6(7) | 85.4(6) |
| UG_2C_2D | 95.6(3) | 95.7(1) | 95.6(5) | 95.6(2) | 95.6(4) | 95.3(6) | 74.3(7) |
| UG_2C_3D | 94.9(3) | 95.2(1) | 95.1(2) | 94.8(5) | 94.9(4) | 94.7(6) | 64.7(7) |
| UG_2C_5D | 92.0(2) | 92.1(1) | 91.9(3) | 91.3(4) | 90.9(6) | 91.2(5) | 80.1(7) |
| keystroke | 84.3(7) | 87.2(5) | 85.9(6) | 88.0(3.5) | 88.0(3.5) | 90.6(1) | 90.5(2) |
| **AVG Rank** | **3.3** | **3.4** | **3.1** | **4.1** | **4.8** | **4.5** | **4.6** |

### 2.3.4 Discussion

The most influential parameter for $LEVEL_{IW}$ is the value of the kernel width 'σ' as used in the Gaussian kernel. The algorithm relies on Core Support Extraction (CSE), which is computationally very expensive, especially for high-dimensional data. Secondly, COMPOSE

(α-shape) strongly depends on the parameter Compaction Percentage which defines the percentage of currently labelled instances to use as core supports, and choosing the best value is problematic. The next section discusses online ensembles which are commonly used in NSEs. The APT algorithm was not included in the analyses, as its steep computational complexity was prohibitive to running some of the larger datasets [80]. This behaviour of APT was also previously reported even on a simple bi-dimensional problem [22].

## 2.4 Data Stream Classification using Ensembles

Data stream classification is a process to extract effective knowledge and thereby unlock valuable insights arising from large amounts of real-time data. Ensemble methods are one of the most promising research directions [82]. Ensembles combine multiple learning algorithms to obtain better predictive performance than single classifiers. Various studies about ensemble approaches for NSE have been published over the years, such as in [46] [66] [83]. Online Ensembles can be categorised as follows.

- Active or passive drift handling ensembles.
- Static or dynamic ensembles.
- Homogeneous or heterogeneous ensembles.

Diversity is one of the key characteristics to consider in the formation of ensembles. However, diversity measures have not received much research interest for evolving data streams [35]. Static ensembles do not add a new member in the ensemble while dynamic ensembles include and exclude the learning models. Figure 2.3 illustrates the categorization of algorithms into Homogeneous and Heterogeneous ensembles.

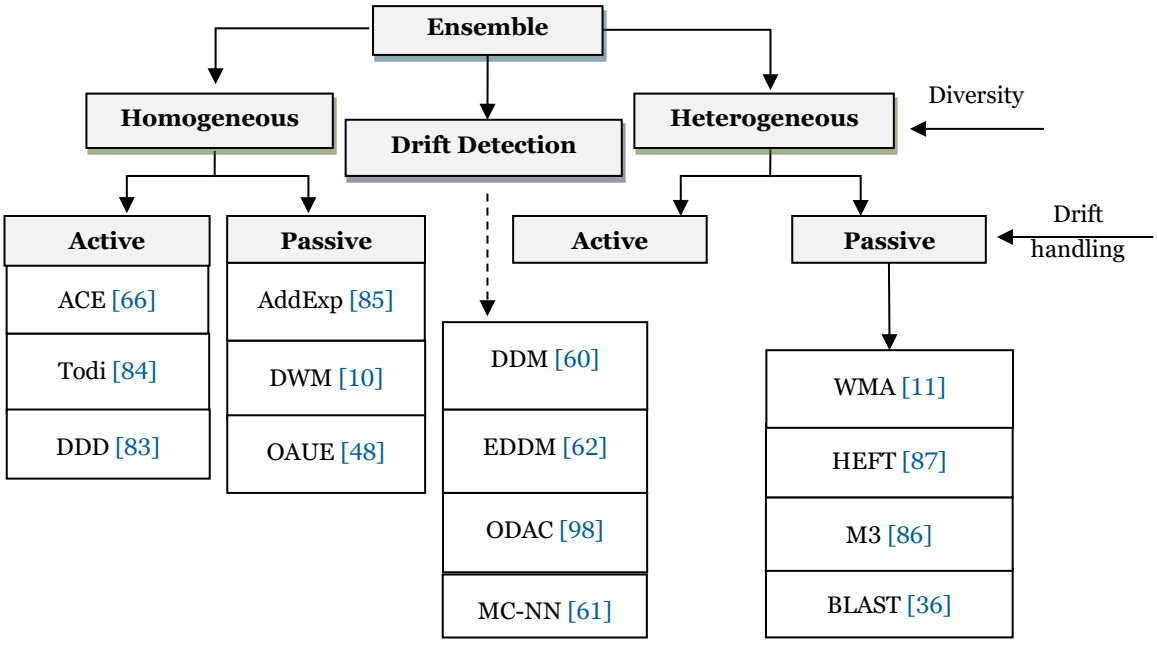


Figure 2.3 Diversity and drift handling approaches of ensemble classifiers, based on drift handling capabilities and selection of base classifiers.

Examples of 'homogeneous active' approaches are Diversity for Dealing with Drifts (DDD) [83], and Adaptive Classifiers-Ensemble (ACE) [66]. Two Online Classifiers for Learning and Detecting Concept Drift (Todi) [84]. Whereas 'Homogeneous passive' approaches are Addictive Base Learner Ensembles (AddExp) [85] , DWM [10] and Online Accuracy Updated Ensembles (OAUE) [48]. A few of the 'heterogeneous passive' ensembles are WMA [11], Modal Mixture Model (M3) [86], Heterogeneous Ensemble with Feature drifT (HEFT) [87] and BLAST (short for best last) [36].

### 2.4.1 Heterogeneous Passive Ensemble

Most existing heterogeneous ensemble techniques rely on meta-learning [36] [88] this helps in deciding which learning techniques work well on what data. The authors of [87] proposed a general framework to integrate feature selection and heterogeneous ensemble learning for data stream classification. The authors in [89] built a heterogeneous ensemble using three different tree-based ensembles (Random Forest [90], Rotation Forest [91], and Extremely Randomised Trees [92]). It was shown that running heterogeneous/different, or homogeneous/similar data stream classification techniques over vertically partitioned data (data partitioned according to the feature space) resulted in comparable performance to batch and centralised learning techniques [37].

**WMA** [11] uses fixed numbers of base learners $C = (C_1, C_2 \ldots C_L)$ with an initial weight '$w_i$ ' equal to '1'. The weight is penalised by a factor of ratio $\beta$ on each wrong prediction i.e. ($w_i \leftarrow \beta w_i$), where the value of $\beta$ is a user provided value between 0 and 1. The final prediction is made based on the weighted majority vote among the base learners $C_i$. The diversity of base learners has a significant effect in improving the performance on different streams. WMA base learners are heterogeneous, potentially helping to produce more diverse ensembles. However, it lacks the option to dynamically add new base learners. The algorithm has no explicit method to detect and handle concept drift thus being less effective in NSEs.

**M3** [86] is a heterogeneous chunk-based ensemble for NSEs. New classifier members are added to the ensemble at each data chunk and the weights are computed based on past performances. A weighting mechanism is used to deal with NSEs. The algorithm continuously updates the models regardless of whether real drift occurs or not.

**HEFT** [87] is an online classifier that incorporates feature selection by applying the Fast Correlation Filter algorithm [93] that dynamically updates the relevant feature subsets for data streams. This is beneficial because NSEs may present feature drift [87] [94]. In high-dimensional datasets, not all features are significant for training a classifier and the relevance of a feature may grow or shrink over time. Given a set of p different classifier types, $M = \{M_1, M_2 \ldots M_p\}$, the ensemble is initialised with 'k' classifiers of each model in M. It determines the most discriminative feature subset on a chunk using a sliding window.

If the subset is different from the previous one, there is a feature drift. The approach then looks for the most accurate classifier having the smallest aggregated error and builds a new classifier. Finally, it removes the classifier with the least accuracy from the ensemble and adds the best classifier to the ensemble. However, after the initialization stage, the algorithm never utilises the 'M' models to create new classifiers. Therefore, there are chances that the ensemble may become homogeneous again in the future.

**BLAST** [36] introduced an Online Performance Estimation framework to weight the votes of (heterogeneous) ensemble members. Based on the zero/one loss function, i.e. returns '1' on correct predictions and '0' otherwise, the weights are increased accordingly. Based on the performances on w (window s) it nominates one of its members to be an active classifier and sets its weight to '1' and the weights of the remaining classifiers to '0'. The weights are updated on a predefined interval. The HEFT and OAUE apply a similar approach in which worst performing models are replaced with new learners, unlike the BLAST that temporarily reduces the weights of a poorly performing member. However, it utilises a static ensemble size similar to WMA.

### 2.4.2 Active and Passive Homogeneous approaches

This section presents related work on passive and active online learning approaches for NSEs which are based on Homogeneous ensembles.

#### a) Active approaches

Active approaches for dealing with NSEs are typically based on single learners. They use concept drift detection methods to determine whether a concept drift has occurred. When concept drift detection occurs, methods for dealing with concept drift are triggered. A common strategy is to reset the single learner to learn the new concept from scratch [29] [46]. A few ensemble-based active approaches are also available in the literature.

**ACE** [66] is an active online ensemble that consists of one online learner, a set of offline classifiers trained on old data, and a method that uses offline classifiers to detect concept drift. Ensemble predictions are based on a weighted majority vote across all classifiers. The classifier weights are based on their accuracy on the most recent training examples. ACE claims to be able to handle sudden, gradual, and recurring concepts better than other systems. However, its integral drift mechanism restricts the algorithm to integrate with other drift detection methods.

**Todi** [84] is based on two online classifiers 'H0' and 'H1' for learning and detecting concept drift. Drift detections are performed based on a statistical test of equal proportions to compare 'H0's performance on recent and old training examples. When a concept drift is detected, 'H0 is reset. 'H1" is never reinitialised upon drift detection but can be replaced by 'H0' when a concept drift is confirmed. Keeping the two classifiers can help to deal with false positive drift detections, as 'H1" can be selected for prediction in the case that the reset 'H0'

classifier is inaccurate after the drift detection. The Todi predictions are the predictions given by the classifier with the best accuracy with the most recent training examples.

**DDD** [83] is an online active ensemble learning approach that creates different ensembles with different levels of diversity to achieve robustness for different types of concept drift. A drift detection method is used to activate very high diversity ensembles which are not helpful during stable concepts, but that can help to deal with slow drifts, or drifts that do not cause too many changes with respect to the current concept. Even though these approaches are based on single learners rather than heterogeneous ensembles, their use of drift detection methods can inspire the proposal of novel heterogeneous ensemble approaches.

### b) **Passive approaches**

Most passive learning approaches (those that do not rely on drift detection methods) deal with concept drift by maintaining an ensemble of base models and use weights to emphasise the models believed to best represent the current concept [35].

**AddExp** [85] adds a new base model (a.k.a. base learner) for every wrong classification given by the ensemble. The weight assigned to the new base model is equal to the total weight of the ensemble multiplied by the parameter $\gamma \in (0, 1)$. The mechanism to update the weight of each base model is analogous to WMA. The weight of each base model is updated by being multiplied by a pre-defined parameter $\beta$, which is a user provided value between 0 and 1, when it gives a wrong prediction. A pruning method eliminates the oldest base models for reducing the ensemble size. Alternatively, the base models whose weight is below a certain threshold can be deleted. The prediction given by the ensemble is the weighted majority vote of the predictions given by the base models.

**OAUE** [48] combines chunk-based and online ensemble methods. The weights of the base learners are calculated by estimating the prediction error on the last d examples. The window size is utilised to create a new base learner for a set of examples and periodically removes the weaker base learners from the ensemble. The output is predicted by aggregating the predictions of base learners using a weighted voting rule. However, the algorithm is highly dependent on the window size. It is likely therefore that a small window size may lose the sudden concept drift, while a larger window may result in false concept detection.

**DWM** [10] is one of the most popular ensemble approaches to deal with concept drift. Each base learner is associated with a weight. Weights start with value one and are multiplied by a predefined parameter ($\beta$, $0 \leq \beta < 1$), when their associated learner gives a wrong prediction in a time step multiple of period $\rho$. This weighting mechanism of DWM is inspired by the WMA. The predictions are based on the weighted majority vote derived from the base learners. DWM enables removal and addition of base learners at every $\rho$ time step. A new

base learner is added whenever the ensemble prediction is wrong in a time step multiple of parameter 'ρ', where the value of ρ must be provided by the user. Removal of learners is controlled by a predefined weight threshold parameter θ. A base learner is removed if its corresponding weight is lower than θ in a time step multiple of 'ρ'. In this way, new learners are created to learn new concepts and poorly performing learners, which possibly had learnt old concepts, are removed. The algorithm normalises the weights by uniformly scaling them such that the highest weight will be equal to one. This is done to prevent newly added base learners from dominating the decision-making of existing ones. However, despite using the WMA weighting mechanism, DWM does not exploit one of the key aspects of WMA — the use of different types of base models.

### 2.4.3 Concept drift detection approaches

Several approaches for drift detection are available in the literature [95][96][97][20]. Although most of these approaches rely on true class labels, some data stream clustering algorithms adapt to concept drift implicitly as part of the learning process. More specifically in CGC, when new instances arrive, the clusters are updated to reflect new concepts. Gama et al. [20] categorised concept drifts detection into four groups, Sequential Analysis, Statistical Process Control, Window based and Contextual approaches.

The number of clustering algorithms explicitly addressing concept drift is very limited and to address the non-stationary nature of data, most available algorithms apply window models [14]. Exceptions to Online Divisive-Agglomerative Clustering (ODAC) [98] and Fast Evolutionary Algorithm for Clustering data stream (FEAC-Stream) [99] that use explicit concept drifts adaptation.

The ODAC [98] algorithm partitions the streams in different time windows. It constructs an incremental tree-like hierarchy of clusters and continuously monitors the diameters of clusters. The split and merge operators are based on these diameters and the confidence levels which are given by the Hoeffding bounds. The authors observed that in stationary datasets the diameter of a cluster reduces every time a split occurs. The system tests for aggregation, so in case of concept drift it will not start to grow unnecessarily.

FEAC-Stream [99] uses the Page–Hinkley Test [100] to detect concept drifts. It is a Sequential Analysis test which was applied to detect whether the assignment of an object to the closest cluster increases the intra-cluster distances significantly.

Cumulative sum approach (CUSUM) [101] is also a Sequential Analysis technique that triggers the alarm when the mean of the input data is significantly different from zero. CUSUM applied in [102] for identifying virtual drifts in data stream clustering problems. The authors also mentioned that the data stream clustering algorithms treat concept drift implicitly as part of the learning process. The need for explicit drift detection and adaptation is often neglected.

Drift Detection Method (DDM) [60] and Early Drift Detection Method (EDDM) [62] are well known representatives of Statistical Process Control and rely on the estimate of classifier error rate. Micro-Cluster Nearest Neighbour (MC-NN) [61], algorithm aims to keep a recent and accurate summary of the data stream and these micro-clusters are used for feature selection and detecting concept drifts. In EVL this estimate is not possible due to unavailability of true class labels, therefore DDM and EDDM are not beneficial in such conditions.

The Statistical Test of Equal Proportion to Detect concept drift (STEPD) [84] monitors the two predictive accuracies of a single online classifier, i.e. accuracy among the most recent examples and overall accuracy from the beginning of the learning. It detects significant decreases in these predictive accuracies by using a statistical test of equal proportions. If the accuracies are statistically similar, then it is assumed that there is no concept drift. If the accuracies are significantly different, then a concept drift is detected. STEPD uses significance levels for drifts and warnings. Like DDM and EDDM, it stores examples in a short-term memory during the warning period and re-builds the classifier on drift detection based on the stored examples.

### 2.4.4 Discussion

Existing passive ensembles can be seen as performing dynamic model selection approaches when they assign different weights to their base learners and when they decide to remove base learners from the ensemble. However, these approaches have not exploited the use of different types of base learners, i.e. they have not exploited the potential benefit of heterogeneous ensembles. Even though the weighting mechanism of DWM was inspired by WMA, which is a heterogeneous ensemble, all the base learners in DWM are homogeneous, e.g. either all of them are Naïve Bayes or all of them are Hoeffding Trees. It is worth investigating how to combine heterogeneity and dynamicity features and develop an algorithm which is dynamic and diverse at the same time.

## 2.5 Data Stream Clustering

In stream clustering, semantically similar objects are moved closer to each other, and the algorithms try to group similar objects. However, stream clustering differs from offline clustering, the data streams arrive at high speed, therefore in stream clustering only single pass on data is possible and it is not feasible to store the data. Clustering is the most appropriate method for real-time data stream processing because it does not require labelled instances and can adapt to concept drifts [32]. Its main purpose is to group similar objects closest to each other. Several surveys and reviews on stream clustering algorithms are available [12][13][14]. Stream clustering differs from offline clustering, the data streams arrive at high speed, therefore in stream clustering only single pass on data is possible and it is not feasible to store the data. The stream clustering algorithms can be categorised into three main classes: hierarchical, partitioning based, and density based.

### 2.5.1 Partitioning Approaches

It groups the instances into a predefined number of clusters based on similarity. The examples are K-Means [33] for static clustering. For Stream clustering, incremental K-Means [103], CluStream [70], StreamKM++[104], Stream LSearch [105], SWClustering [69]. K-Means [33] is a clustering algorithm widely used in data mining. It groups the instances into a predefined number of clusters by using the Euclidean distance between each instance and the centroid using Equation 2.4.

$$f = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left| x_i^k - \omega_k \right|^2 \tag{2.4}$$

Where, 'K' is the number of clusters, $\omega_k$ is the centroid of the $k^{th}$ cluster, $n_k$ is the number of instances assigned to the $k^{th}$ cluster, $w_i^k$ is the $i^{th}$ instance belonging to the $k^{th}$ cluster.

$$\omega_k = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i^k \tag{2.5}$$

CluStream [70] uses two stages, online micro-clustering, and offline online macro-clustering. In the online phase the instances are assigned to the micro-clusters or create new micro-clusters, the nearest clusters can be merged to create space for new clusters. The offline stages apply a weighted K-Means algorithm on the micro-clusters, to obtain the final clusters.

The micro-clusters operate as an intermediate statistical representation for large volume of data. On the other hand, the macro-clusters store a compact summary statistic of the micro-clusters to produce clusters for analysis on demand.

### 2.5.2 Hierarchical Clustering

Hierarchical clustering uses binary trees and are divided into agglomerative and divisive. Agglomerative algorithms (bottom-up) assign every instance as a cluster itself, and gradually merge the similar clusters to reduce the cluster count. The divisive algorithm (top-down) is opposite, it starts with a single cluster containing all instances, then gradually breaks the clusters into smaller clusters. Examples are Hierarchical K-Means [106] BIRCH [107], CHAMELEON [108], ODAC [98], E-Stream [109] and HUE-Stream [110].

### 2.5.3 Density Based Clustering

Density based algorithms are intended to group arbitrary shaped clusters and detect the number of clusters. Examples are DenStream [111] , LDBSCAN [112], ACSC [113], D-Stream [114] and MR-Stream [115], OPTICS [116], which combines agglomerative and density approaches. Density-based hierarchical methods using sliding windows are also available streaming data clustering [117].

### 2.5.4 Micro-clustering

BIRCH [107] introduced the Cluster Feature (CF) or micro-cluster which is a triple: CF = (N, LS, and SS), where N is the number of instances, (LS) is a vector with the linear sum and (SS) is a vector with the square sum of the N points. The centroid and radius are computed using Equation 2.2 and 2.3.

### 2.5.5 Discussion

The problem of online SSL is relying on the choice of clustering algorithms. When data arrives in streams, several problems arise here, such as computation and data storage. Existing state-of-the-art online SSL algorithms use micro-clusters for acquiring the pseudo-labels for large unlabelled data, which further train the classifiers to be used for predictions. Density based algorithms are ideal for streams because it does not rely on the number of clusters, it can group clusters of arbitrary shapes and can handle outliers and noise. These algorithms often require all the raw instances to form clusters, which seems unrealistic in data stream applications due to the time limitation. The agglomerative algorithms are too slow for large datasets due to their complexity $O(n^3)$. While the Partitioning based streaming algorithms, e.g. CluStream, produce spherical clusters, therefore are less accurate than the density based. However, several parameters must be chosen manually.

## 2.6 Data Stream Evaluation Methods

For evaluating learning models in the data stream, three alternatives are prequential, holdout [51] evaluations and Kappa statistics [52]. The details of these evaluation methods are discussed as below.

### 2.6.1 Prequential

It is a general methodology to evaluate learning algorithms in data streams learning. Prequential evaluation [118] also called test-then-train in which each new example is used to test the model and then use the same example to train the model. According to [46] the prequential error is computed based on an accumulated sum of a loss function 'L' between the prediction $y_t$ and observed values $\hat{y}_t$ using Equation 2.6

$$p_0 = \sum_{t=1}^{n} L(\hat{y}_t, y_t) \tag{2.6}$$

Prequential evaluation can be applied with sliding windows and decaying factors to improve classification results in evolving data streams [119]. However, this method can report initial poor performance due to fewer training examples seen by the model.

### 2.6.2 Holdout

It uses predefined partitions of train and test instances, as it holds a subset of examples to be used as a training set at regular intervals. In a holdout test set with M examples, the 0-1 loss is computed using Equation 2.7.

$$H_e(i) = \frac{1}{M}\sum_{k=1}^{M} L(y_k, \hat{y}_k) = \frac{1}{M}\sum_{k=1}^{M} e_k \tag{2.7}$$

### 2.6.3 Kappa Statistics

Kappa statistics was introduced by [52] as a more sensitive measure for quantifying the predictive performance of streaming classifiers. The Kappa statistic 'k' is calculated using Equation 2.8.

$$k = \frac{p_0 - p_c}{1 - p_c} \tag{2.8}$$

Where '$p_0$' is the relative observed accuracy of the classifier calculated using Equation 2.9

$$p_0 = \frac{TP - TN}{n} \tag{2.9}$$

Where, 'TP' = true positive' and 'TN' = true negative and 'n' is the total number of observations. '$p_c$' is the expected accuracy of the classifier agreed with the ground truth label. It is calculated using Equation 2.10

$$p_c = \frac{(TP + FN)(TP + FP)}{n^2} + \frac{(FP + TN)(FN + TN)}{n^2} \tag{2.10}$$

### 2.6.4 Discussion

Kappa statistics is a more sensitive measure for quantifying the predictive performance of streaming classifiers since it is not sure if the classes are balanced. Class imbalance is a condition when there are much more examples of a given class than the others and this class may be emphasised in detriment of the other classes. For evolving data streams, performing the error estimation is the key difference from traditional data mining evaluation because the cross-validation may be too expensive. The holdout evaluation requires the user to specify two parameters, i.e. the size of the window and the number of examples to test in each window. However, choosing the right values for these parameters are problematic.

## 2.7 Hyperparameter Tuning

Hyperparameters are parameters that must be initialised before learning begins. Several data stream clustering algorithms apply K-Means due to its simplicity, scalability, and empirical success in many real-word applications. However, one of the pitfalls of the K-Means is its dependency on the number of clusters 'k' that must be specified prior to the learning. In an effort to extend K-Means-Based Algorithms for evolving data streams with variable number of clusters the authors in [120] illustrated the potential of the proposed

framework by using three state-of-the-art algorithms for clustering data streams – Stream Lsearch [105], CluStream [70] and StreamKM++[104], - combined with two well-known algorithms for estimating the number of clusters, OMRk [44] and BkM [45].

## 2.8 Randomisation in Data Streams

In data streams, continuous data arrives at high speed and there is practically no control over the sequence of training data presented to the learning algorithms. Randomisation is thus different to noise, as it is not a random displacement of examples. The output of an adaptive classifier at every time step depends on the instances seen so far. Hence, performance depends on the order of instances in the dataset [47], the authors suggested executing multiple tests with randomised copies of a data stream. Devices can generate noisy data due to sensor inaccuracies or malfunctions. Cleaning the data helps in removing noise, ensuring that the model is not influenced by irrelevant or erroneous information. Cleaning data from devices before dealing with drifts is a critical step to ensure the reliability and effectiveness of models, especially in dynamic and evolving environments.

## 2.9 Visualising Data Stream

Mining data streams has attracted a big deal of attention over the last decade [121]. However little work has been done for visualisation of data stream mining. Traditional data Analysis systems like JMP [122] and WEKA [123] provide a wide range of interactive visualisation techniques and exploratory data Analysis tools. However, these tools are intended to work on offline datasets. MOA [46] is another comprehensive data stream mining tool that provides real-time visualisation of clustering and as well as outlier detections as shown in Figure 2.4.

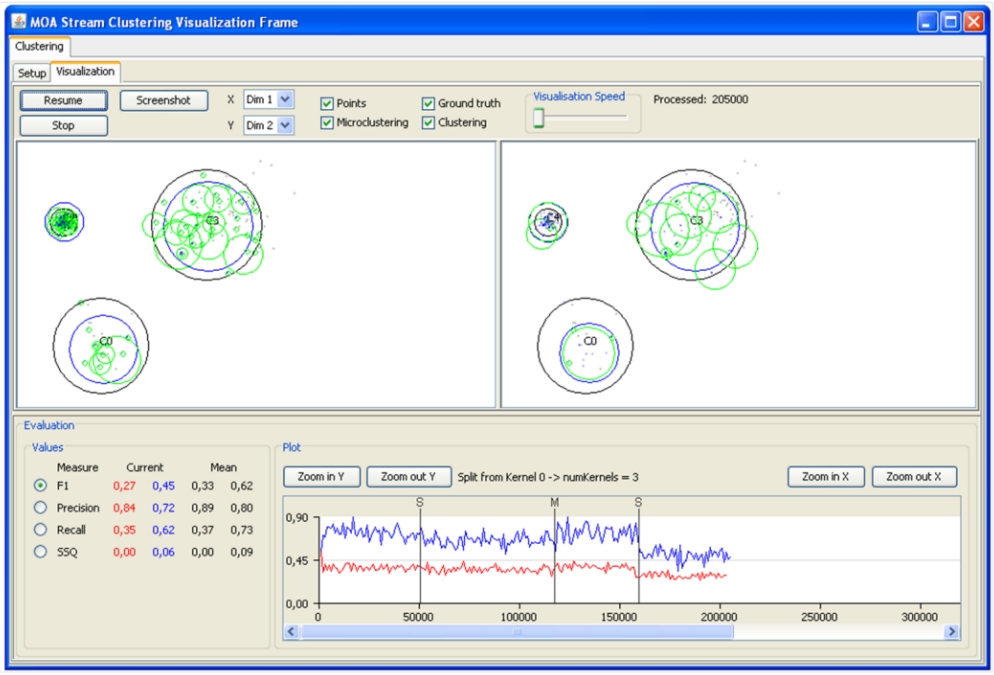


Figure 2.4 Visualisation of online Clustering in MOA showing ground truth and micro-clustering along with performance measures.

It provides a variety of visualisation support for supervised and unsupervised however the visualisation of SSL is not available. In the research a visualisation tool for SSL has been developed for MOA.

Gephi [132] is an interactive and hierarchical graph-based tool for real-time visualization and exploration tool. It includes layout algorithms such as force-based and multi-level algorithms. Figure 2.5 shows worldwide flight routes network as connected graph of airport dataset [133]. The visualisation in Gephi has been created and can be found here [54]. The directed graph is using 5,623 nodes (airports) and 37,596 edges (routes) presented as a mixture overlay between network graph and geographic data using the Geo Layout plugin.

Figure 2.5 Visualizing Airline Routes Network using Gephi showing airports and routes using mixed overlay between network graph and geographic data.

## 2.10 Limitations of the Approaches described in the Literature

To deal with EVL, the available approaches are not compared with each other and in extent to our knowledge no comprehensive studies are available that justify the one approach over the other. Furthermore, the reasons why one approach performs better than the other when applied to different characteristics of data streams and why the other approach fails, is still not clear in the literature. This section highlights a few gaps of the literature which are identified in the literature review process.

**Gap 1:** To deal with the diversity of learning models, existing approaches are static or dynamic and homogeneous or heterogeneous. NSEs approaches are either active or passive therefore restricting to adaptation of either gradual or abrupt drifts. It has been a key challenge in data stream mining, as some algorithms heavily rely on forgetting mechanisms while others retain previous learning.

**Gap 2:** EVL deals with unlabelled data more effectively using clusters, however under NSEs when the clusters overlap, existing approaches relying on micro-clustering. Apart from that, this approach is computationally expensive for mining high speed data streams.

Furthermore, existing EVL approaches are human dependent on redefining the best type of predictive models for a particular problem or pseudo-labelling strategy. i.e. CGC, or self-learning.

The predictive performance of SCARGC is highly dependent on clustering, and it also requires some prior knowledge such as the number of centroids 'k' and pool size 'θ' which may significantly affect the predictive performance when such information is not available. To choose the best value of 'k' which is suitable for a particular data stream, the algorithm needs to run several times with different values of 'k' and pick the 'k' that gives the best predictive accuracy.

While analysing the results published in the respective papers of $LEVEL_{IW}$ and COMPOSE, it is difficult to determine which performs better, it seems to be strongly dependent on the application. COMPOSE showed better results than $LEVEL_{IW}$ when there was a significant class overlap. COMPOSE uses the parameter 'k', the number of centroids for, and $LEVEL_{IW}$ uses parameter σ which is the value of the kernel bandwidth. However, in case of complete class overlap and a condition when no ground truth data is available, it is very challenging for the algorithm to recover learning from it.

## 2.11 Summary

In this literature review, a comprehensive examination of existing studies, findings, and methodologies have been conducted to enhance the understanding of key challenges in data stream mining. The focus areas included label scarcity, maintaining ensemble diversity, autonomous hyperparameter optimization, and the success and adaptability of existing approaches. Addressing the scarcity of true class labels in non-stationary conditions (RQ1), the Initially ILNSE approach has been identified and various techniques such as CGC, self-learning, and micro-clustering have been explored. The literature addresses NSEs and EVL separately, with limited algorithms handling both. Furthermore, a lack of clarity was noted regarding the comparative effectiveness of these approaches. While analysing the results published in the respective papers of $LEVEL_{IW}$ and COMPOSE, it was difficult to determine which algorithm performs better, both these algorithms were highly dependent on the characteristics of datasets. COMPOSE showed better results than $LEVEL_{IW}$ in the presence of significant class overlap. However, the drawback of COMPOSE is parameter 'k', which is used to define the number of centroids, the algorithm applies arbitrary values for 'k' to achieve the best predictive performances. $LEVEL_{IW}$ is based on parameter σ which is the value of the kernel bandwidth. In case of complete class overlap and a condition when no ground truth data is available, it is very challenging for the existing algorithm to recover learning from it.

Concerning the diversity of the prediction models (RQ2), in the case of WMA its base learners are heterogeneous, potentially helping to produce more diverse ensembles. However, it lacks the option to dynamically add new base learners. The algorithm has no

explicit method to detect and handle concept drift thus being less effective in NSEs. Despite using the WMA weighting mechanism, DWM does not exploit one of the key aspects from WMA - the use of different types of base models. The HEFT and Online OAUE apply a similar approach in which worst performing models are replaced with new learners, unlike the BLAST that temporarily reduces the weights of a poorly performing member. However, it utilises a static ensemble size like WMA.

ILNSE scenarios often involve significant human dependency, especially for parameter tuning (RQ3). The literature review provided an in-depth analysis of various hyperparameter tuning techniques employed in data stream mining. The literature showed that, existing CGC approaches rely on prior knowledge concerning the number of classes for generating the corresponding centroids. The evaluation of SCARGC involved the use of various arbitrary values for 'k' and pool size θ, leading to adverse effects on the prediction results.

The (RQ4) addresses the consistency of success for existing approaches across diverse problems. The literature suggests that the CGC approach is effective in adapting to changes by identifying and updating clusters. This way, the model can focus on adapting to the evolving characteristics of different data clusters. self-learning approaches can help by iteratively updating the model with new data points. When the environment becomes non-stationary, the model can adapt by incorporating the most recent information from the unlabelled data into its classification decisions. Micro-clustering can be especially useful in ILNSE because it can capture fine-grained changes in the data distribution. By identifying micro-clusters within the data, the model can monitor and adapt to subtle shifts in the data's characteristics over time, which is essential in a non-stationary environment.

The common challenge in all these approaches is the potential for misclassification, especially when dealing with overlapping clusters or incorrect pseudo-labels. To address these issues, researchers might be exploring methods to improve the accuracy of cluster assignments, reduce the processing time of micro-clustering, or develop more robust self-learning techniques that can adapt to evolving data distributions while mitigating the impact of incorrect pseudo-labels. The choice of approach may depend on the specific requirements and constraints of the application, as well as the nature of the data and the available computational resources. Researchers in this field likely work on refining and combining these methods to achieve better results in non-stationary environments.

Learning task in data streams are divided into Online and block-based approaches, in online learning the training examples are presented to the learning algorithms at any time-step, which learns and predict a class label. On the other hand, in block-Based learning the data stream is partitioned into different sized blocks containing equal examples. Block-based approaches wait for a whole new chunk or batch of data to arrive; and then use this new chunk for training before discarding it. The block/batches are useful in creating clusters which can be used for pseudo-labelling in ILNSE. A hybrid approach applies both

incremental and block-based learning approach. Therefore, applying the hybrid approach is more beneficial in ILNSE. One issue arises here is choosing the right value of batch size, it is likely that a small batch size may lose the sudden concept drift, while a larger window may result in false concept detection.

The literature review highlighted the need for more clarity on the comparative effectiveness of ILNSE approaches and the challenges associated with diverse ensemble techniques. It also emphasized the significant role of human analysts in parameter tuning and underlined the potential of advanced algorithms for improved cluster estimation. The review provided a foundation for understanding the current landscape of data stream mining challenges and solutions, paving the way for further research in this dynamic field. analysts may need to develop strategies for identifying and mitigating label noise, which can adversely affect model training and adaptation. This might involve techniques like outlier detection, consensus labelling, or active learning to improve the predictive performances of data stream mining algorithms. The following chapter undertakes a preliminary investigation into Extreme Verification Latency and its impact on prediction accuracies, aiming to pinpoint gaps within the existing literature.

# Chapter 3 Preliminary Investigations on Extreme Verification Latency

This chapter focuses on investigating the research questions by experimenting the problem discussed in 0. i.e. the problem of EVL and its effect on prediction accuracies. The research aims at creating a new algorithm for detecting changing environments. The literature reveals that there are several algorithms exists that has ability to detect drift on streaming data. Online ensemble classifiers such as DWM [10] has been successfully used to improve the accuracy of single classifier in online and incremental learning and applies passive drift detection approach which is suitable to detect gradual drifts. The dynamic ensemble method of DWM combines the decisions of all the learners to predict the classes. The method uses mechanism to dynamically add or remove the base learner based on the global algorithm's performance.

The DDM is an active drift detection method initially proposed by Gama et al. [60] monitors the number of errors produced by the learning model during the prediction and suitable for detecting sudden drifts. DWM is lacking active drift detection approach to handle sudden drift and applied a passive approach i.e. represents weights to handle gradual drift. A recent study revealed that different diversity levels in an ensemble of learning machines are required to maintain high generalization on both old and new concepts. Therefore, a comprehensive investigation has been carried out to incorporate DDM in DWM.

In the data stream mining, there is limitations to store all the data, for building a decision tree there is the need to reuse the training examples to compute the best splitting attributes. Domingos and Hulten [136] proposed the Hoeffding Tree (HT) which is a very fast decision tree algorithm for streaming data. The most interesting feature of the HT is that it builds a tree that provably converges to the tree built by a batch learner with sufficiently large data [46]. HT applies Hoeffding bound [136] 'ε' used to split decision in the tree is calculated using Equation 3.1.

$$\epsilon = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}} \qquad 3.1$$

Where, 'n' is the count of independent observations of the variable 'r' and R is the range of 'r'. δ is the desired probability and its value is provided by the user.

The Naïve Bayes (NB) [38] classifier is based on Bayes' theorem in which the probability of an event occurring is calculated. A data stream $X = \{X_1, X_2 \ldots X_N\}$ having discrete set of 'y' multi-classes. The authors in [38] determines the estimate of probability that an example is belonging to class 'y' given the features X is calculated in using Equation 3.2 and for all possible values of the class 'y' determine the maximum probability using Equation 3.3.

$$P(y|\hat{x}) = \frac{P(y)\prod_{i=1}^{N} P(\hat{x}_i|y)}{P(\hat{x})} \quad 3.2$$

$$y = \text{argmax}_y\, P(y)\ \prod_{i=1}^{N} P(\hat{x}_i|y) \quad 3.3$$

Where, P(y) is called class probability and P ($\hat{x}_i$ | y) is the conditional probability.

This chapter is organised as follows: Section 3.1 presents the analysis of EVL, Section 3.2 investigates the heterogeneous and homogeneous classifiers.

## 3.1 Analysis of Verification Latency

This section investigates the problem of class label scarcity also called EVL and its effect on prediction accuracies on learning models. Under ILNSE the prediction accuracies of learning models are greatly influenced by the scarcity of true class labels which are never available to update the prediction models. Therefore, it is beneficial to analyses the concept drifts and its consequences on prediction accuracies under the class label scarcity.

### 3.1.1 Experimental Design

All the experiments are evaluated in terms of predictive performance. As shown in Figure 3.1, the Keystroke datasets has been used with EVL option in which the size of labelled data is set to 150 examples for a total of 55 thousand examples.

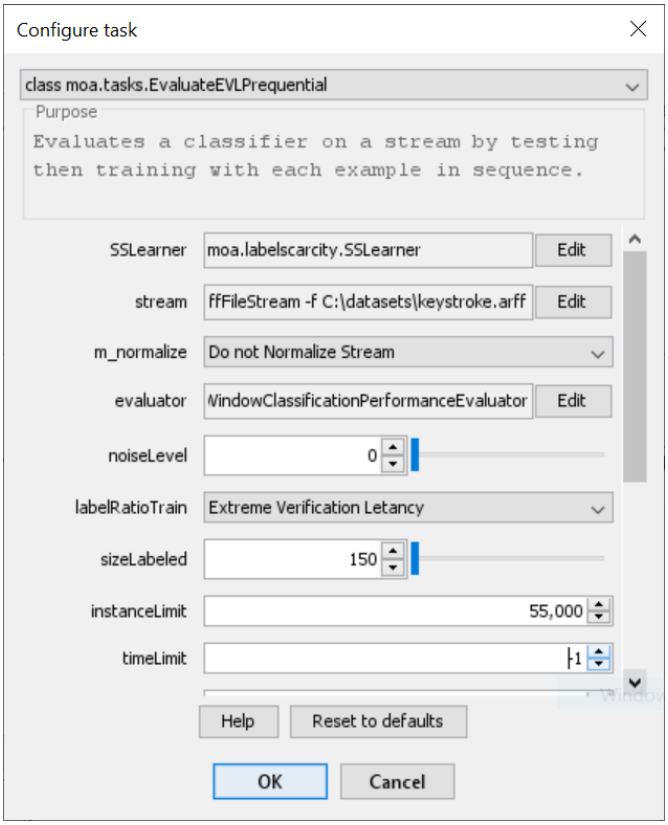


Figure 3.1 New Task 'EvaluateEVLPrequential' created in MOA.

A new Task 'EvaluateEVLPrequential' has been created in MOA [46], the EVL learning task is provided with an option to define the size of labelled data and purposely removes the true class labels from the remaining part of the data stream. An option has been added to compare the evaluation results of 1) Applying 100 training data with 2) EVL i.e. partially initial labelled data streams.

### 3.1.2 Data Streams and datasets.

SEA_Drifts, HyperPlane (Incremental), RandomTree (No Drift) [46] data streams has been applied in this experiment. The description of Random Tree and HyperPlane is available in Section 4.4.1. SEA_Drifts has been explained in Section 3.2.2. The HyperPlane Incremental drift is induced by changing the values of the HyperPlane weights as the time advances. It is possible to modify the orientation and position of HyperPlane by gradually changing the values of weights.

The Keystroke dataset [135] task is to predict the typing rhythms of genuine users and impostors, based on their typing patterns. The dataset consists of 10 attributes and 4 class labels containing records obtained from the users in 8 different sessions who typed a fixed password. The configuration uses a batch size of 150 examples and a total of 150 training examples i.e. only the first batch out of 55k has been used to train the classifier.

### 3.1.3 Significant Findings

The results show that under EVL conditions all the datasets have underperformed in terms of prediction accuracies, as shown in Table 3.1, the EVL highly effected the keystroke dataset and Incremental drifts and a slight reduction in the prediction accuracy in the data stream with no drifts.

Table 3.1 Comparison of Prediction accuracies (%) of EVL and No EVL on drift streams, no drift and real-world dataset.

| **Streams (with Drifts)** | **EVL (%)** | **100% labelled. training set** | **Difference (%)** |
|---|---|---|---|
| SEA_Drifts | ↓ 69.9 | ↑ 77.1 | 7.2 |
| HyperPlane (Incremental) | ↓ 53.8 | ↑ 82.8 | 29 |
| RandomTree (No Drift) | ↓ 60.3 | ↑ 64.1 | 3.8 |
| Keystroke | ↓ 49.0 | ↑ 89.0 | 40 |
| **Average (%)** | **58.2** | **78.2** | **20** |

The SEA dataset consists of sudden concept drifts at epochs 25k and 75k as shown in the dashed vertical lines. Under EVL the classifier trains on initial 50 labelled examples and predicts fine until the first abrupt concept drift appears at epoch 25k. As shown in Figure 3.2 the predictions get worst between 25k and 75k, it is possible to timely recover from drift

and train on the new concept but under EVL, no true class labels are available. Likewise in the HyperPlane data stream in which incremental drift exists and not true class labels are available.

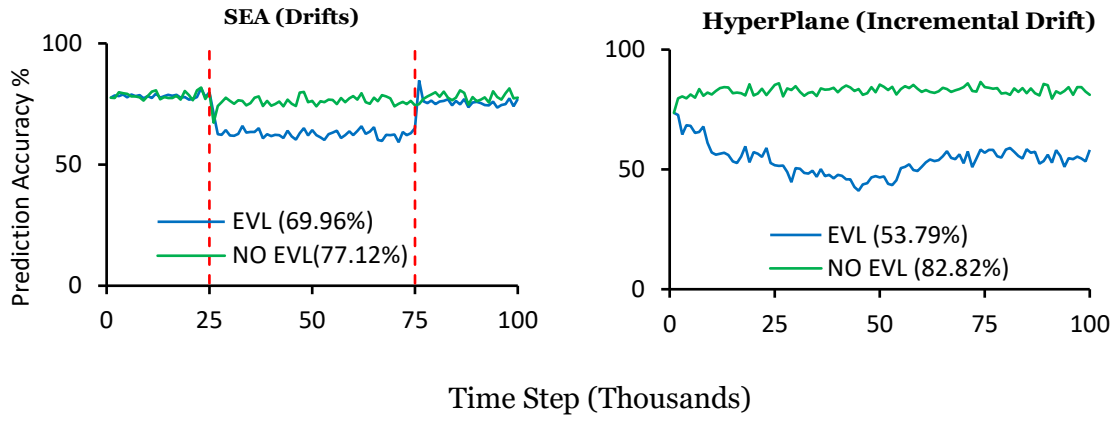


Figure 3.2 Comparison of EVL and No EVL on Prediction accuracy (%) on SEA Drift and HyperPlane incremental drift streams, the vertical dotted lines representing sudden concept drifts.

In the Real-world dataset 'Keystroke' the true location of concept drift is not available. Figure 3.3 show the prediction accuracies of Keystroke and RandomTree under no EVL and with EVL. It is evident from the plots that when no drifts are present the EVL does not significantly affects the prediction capabilities, while the concept drifts significantly reduce the predictive performance as in case of Keystroke, HyperPlane (Incremental Drift) and SEA (Drifts) data streams.

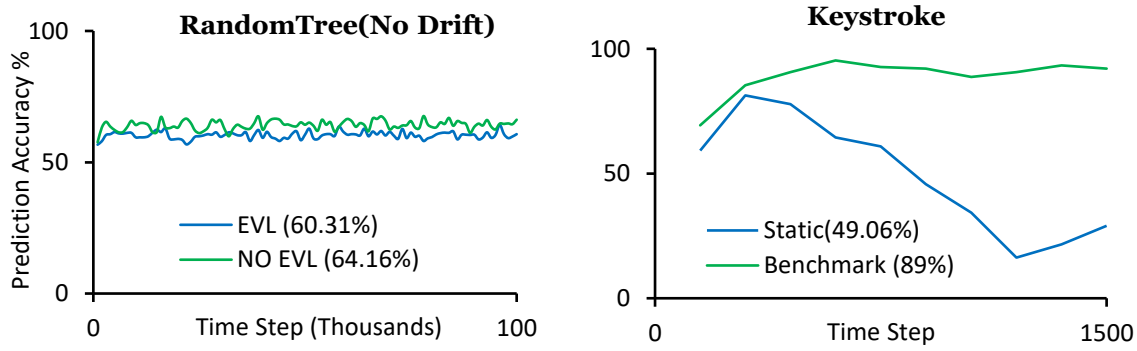


Figure 3.3 Comparison of EVL and No EVL and its effect on Prediction accuracy (%) on Real-world dataset and RandomTree having no drifts.

## 3.2 Heterogeneous VS Homogeneous Classifiers

As discussed in the literature review Section 2.4, the two categories of ensemble methods based on configurations are: 1) Homogeneous ensembles: It uses same types of learners and 2) Heterogeneous ensembles: It uses different types of base learning algorithms. The base learners can be NB or HT etc. The WMA [11] consists of heterogeneous base learners in its ensemble but the size of learners in the ensemble is fixed. By integrating features of both these classifiers and by incorporation the existing drift handling methods in the classifier

would theoretically increase the performance and accuracy of the classifier. DWM is homogeneous ensemble, but it has ability to deal with sudden drifts by adding and removing the base classifiers. The following experiment will analyse the effect of homogeneous and heterogeneous ensemble in the context of drift handing capabilities.

### 3.2.1 Experimental Design

All the experiments are evaluated in terms predictive performance and performed on machines with Core i7 @ 3.4 GHz, 4 GB of RAM and executed within the MOA [46] framework. The defaults parameters of DWM and WMA has been applied in the experiments. The artificial datasets used in the experiments are generated in the MOA, a detailed description of Random Tree and HyperPlane is available in Section 4.4.1. Prequential evaluation [46] has been applied using frequency 1000 and instance size 100k.

### 3.2.2 Data Streams

The RandomTrees generates a stream based on a randomly generated tree. Hyperplane is a flat n-dimensional space useful for simulating gradually drifting concepts. The orientation and position can be modified by slightly changing its relative size of the weights. The STAGGER [10] dataset consists of 120 training instances, each instance presented with 100 test instances. Generated the dataset consists of 12,000 training and testing examples. The training instances contain the following concepts changing at epoch 40 and 80.

- Concept 1 = size ==small && colour == red
- Concept 2 = colour ==green || shape == circle
- Concept 3 = size ==medium ||size == large

The SEA [64] concepts consist of four concepts and three attributes, where Attribute (i) $\in \mathbb{R}$ such that $0.0 \leq xi \leq 10.0$. The target concept is Attribute (1) + Attribute (2) $\leq$ b, where b $\in$ {9, 7, 8, 9.5}.

- Concept 1 = Attribute (1) + Attribute (2) $\leq 8$
- Concept 2 = Attribute (1) + Attribute (2) $\leq 9$
- Concept 3 = Attribute (1) + Attribute (2) $\leq 7$
- Concept 4 = Attribute (1) + Attribute (2) $\leq 9.5$

The SEA_Drift datasets used in Section 3.2.4 for the experiments on concept drifts, consist of 100,000 training, each instance presented with 2500 test instances therefore PeriodicHeldout evaluation has been applied for these datasets. The training instances contain sudden concepts drifts at epoch 2k and 75k. Figure 3.4 showing the learning strategies and time changing concepts of SEA Drift data stream. The MOA commands to generate this stream is available in APPENDIX I.

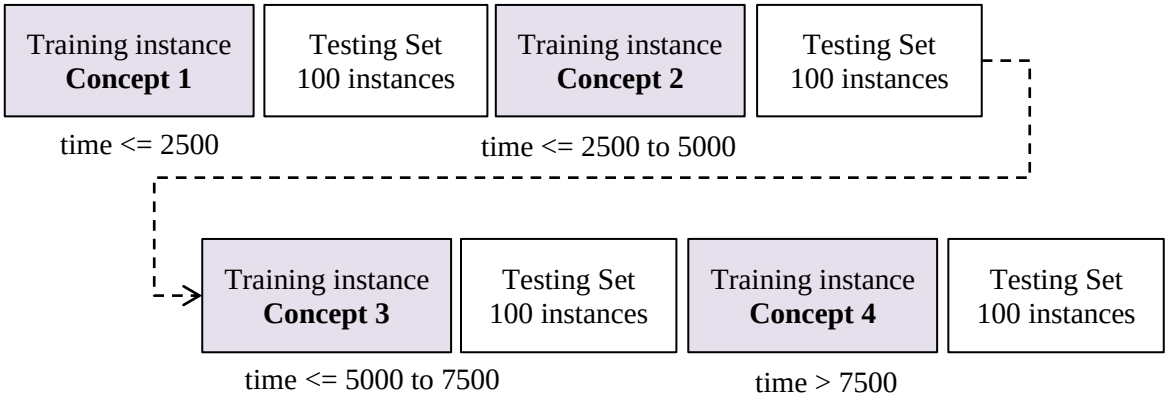


Figure 3.4 Changing concepts and learning strategies in SEA data streams.

### 3.2.3 Significant Findings in Heterogeneity

The prequential evaluation results in Figure 3.5 show that on RandomTree data stream the prediction accuracy of HT classifier is (92.1%), which is higher than the NB classifier (73.5%). Contrary to that, on Hyperplane data stream the prediction accuracy of NB is higher (93.9%) than HT which is (89.4%). Therefore, it is extremely difficult to choose the best performing classifiers for a specific problem.

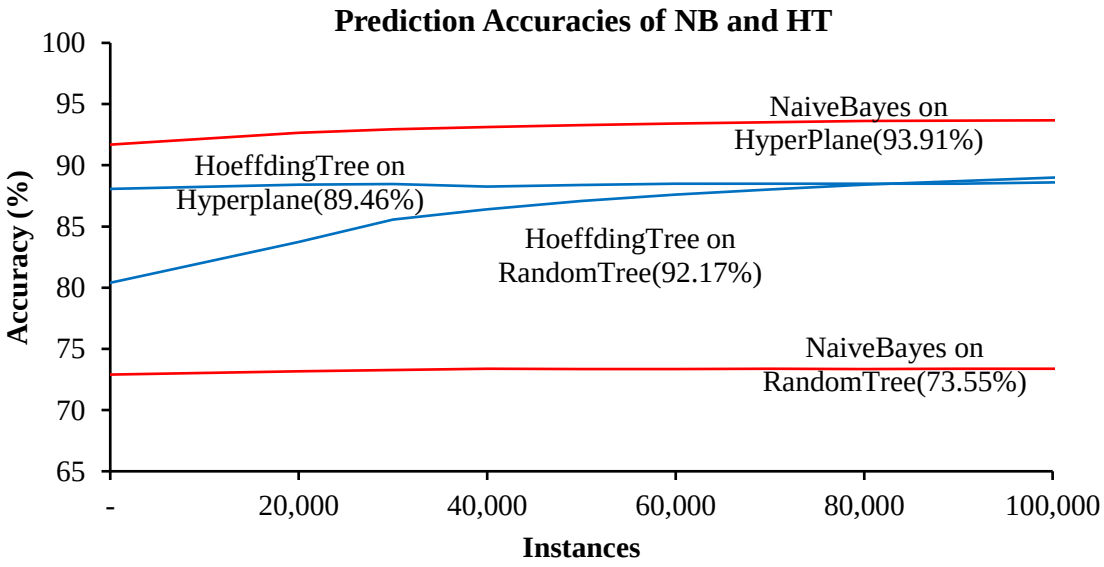


Figure 3.5 Comparison of Prediction accuracy (%) of NB and HT classifiers on HP and RandomTree data streams.

The DWM-NB (DWM using NB as base classifiers) when applied to Random Tree, the prediction accuracy is low (72.35%) but improved in DWM-HT (85.45%) (DWM using HT as base classifiers). Likewise, when DWM-NB base classifier is applied to HyperPlane data stream, the prediction accuracy is higher (93.13%) than the DWM-HT base classifier (84.64). This shows that the DWM which is a dynamic ensemble of classifiers also failed to fill this prediction accuracy gap.

Table 3.2 Prediction accuracies (%) of DWM-NB and DWM-HT.

| Stream | DWM-NB | DWM-HT | Difference |
|---|---|---|---|
| RandomTree | 72.35 | 85.45 | ↑ 13.1 |
| HyperPlane | 93.15 | 84.64 | ↓ 8.51 |
| **Average (%)** | **82.75** | **85.04** | **-2.29** |

Apart from that, the WMA which is heterogeneous ensemble when applied on the similar problem, the results in Table 3.3 shows that WMA overcomes this problem and maintains the prediction accuracy (average 89.1%) on RandomTree and HyperPlane (94.0%). This improvement is because the WMA uses both HT and NB base classifiers in its ensemble.

Table 3.3 Prediction accuracies (%) of WMA Heterogeneous ensemble classifiers.

| Streams (No Drifts) | Heterogeneous (WMA) |
|---|---|
| RandomTree | 89.1 |
| HyperPlane | 94.06 |
| **Average (%)** | **91.58** |

### 3.2.4 Significant Findings in Concept Drifts

In another experiment, the WMA fails on STAGGER and SEA datasets which consists of two sudden drifts at 25k and 75k. The results in Table 3.4 shows that when WMA is applied to STAGGER_Drift and SEA_Drift, the WMA did not adapt to sudden drifts as compared to DWM which deals the sudden drifts by added a new base classifier. Despite that the WMA is heterogeneous, it is not capable to adapt to sudden concept drifts.

Table 3.4 Prediction accuracies (%) Homogeneous DWM and Heterogeneous WMA ensemble classifiers.

| Streams (with Drifts) | Homogeneous DWM-NB | Heterogeneous WMA |
|---|---|---|
| SEA_Drift | ↑ 87.98 | ↓ 85.79 |
| STAGGER | ↑ 86.20 | ↓ 55.08 |
| **Average (%)** | **87.1** | **70.4** |

Figure 3.7 shows results on STAGGET datasets, WMA which is a heterogeneous classifier and a better choice when no drift, as in case of Hyperplane and RandomTree, However the WMA is unable to restore learning after the sudden drift at location ‘40’ and ‘80’. DWM on the other hand is reactive to drifts but lacking heterogeneity. This requires investigating the

cause of this varying results and requires an approach that adapts to the given problem, timing and conditions and maintains the accuracy on both these data streams.

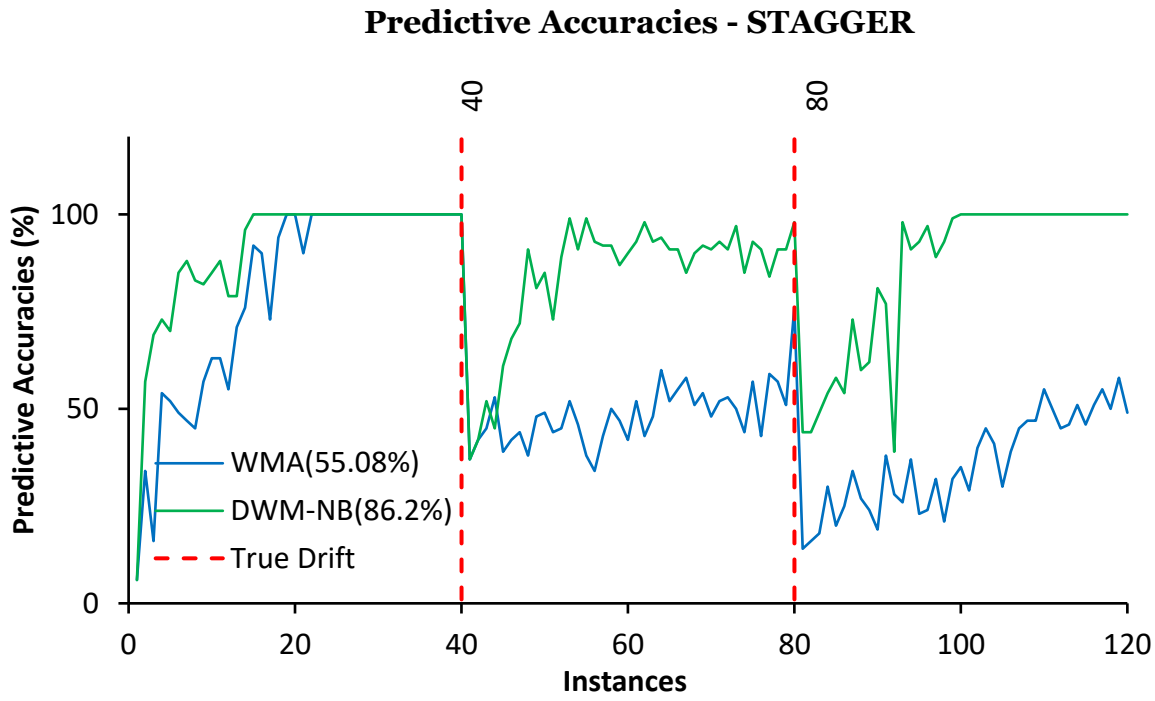


Figure 3.6 shows results on SEA datasets, the WMA is not able to recover from the sudden drift at 25k and 75k, however DWM recovered from the sudden drifts.

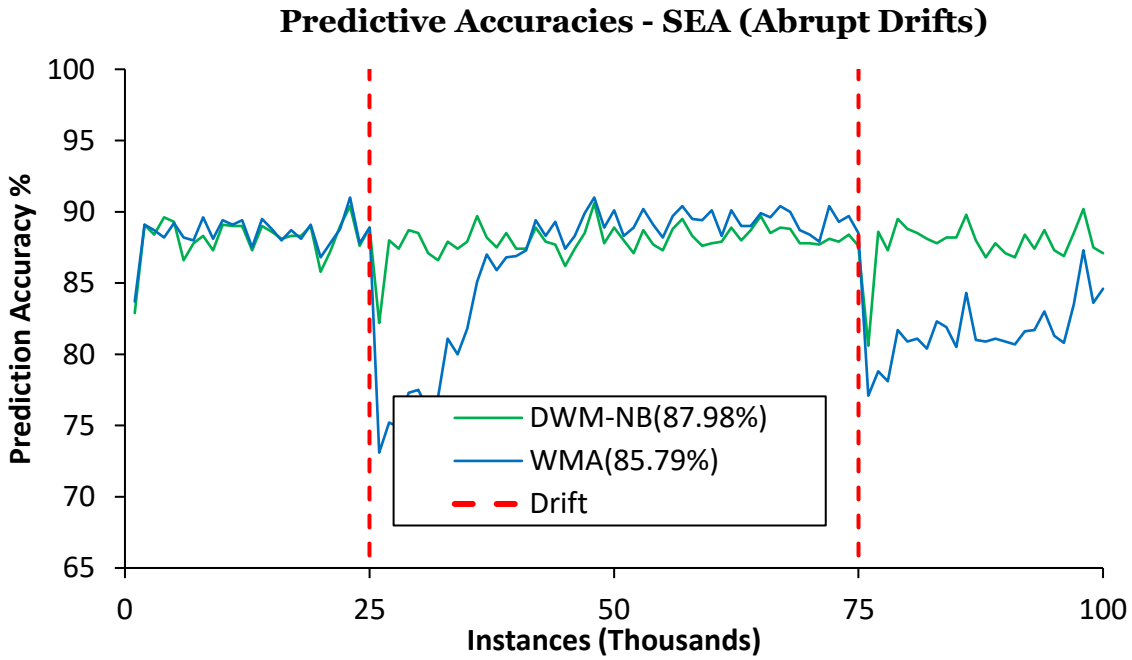


Figure 3.7 Comparison of Prediction accuracy (%) of WMA and DWM on SEA dataset, the vertical dotted lines representing sudden concept drifts.

## 3.3 Implementation of SCARGC in MOA

The implementation of SCARGC algorithm in JULIA language is available here [134]. The SCARGC algorithm has been implemented in MOA to enable experiments on MOA data streams and to standardise its comparison with our PSDSL approach. Figure 3.8 shows our implementation of SCARGC in MOA, to assure the correctness of our implementation the figure below showing the prediction accuracies achieved in 2CDT datasets using the pool size 300 and only 50 instances out of 16,000 were labelled.

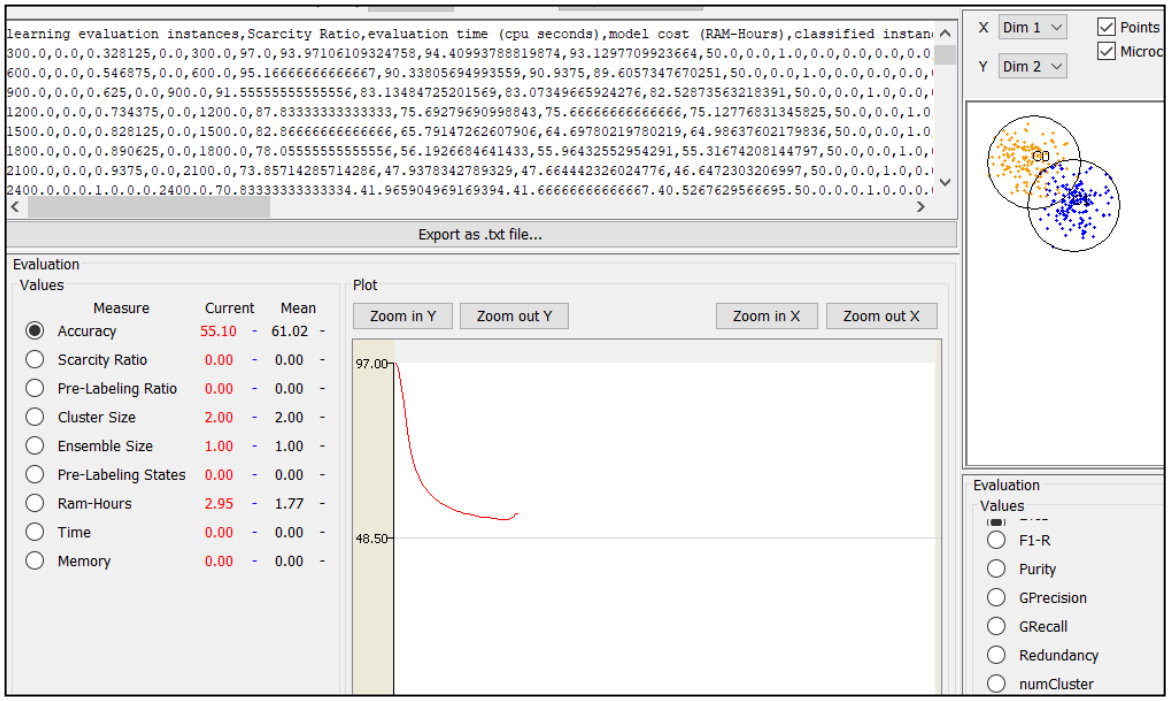


Figure 3.8 The implementation of SCARGC in MOA applied on 2CDT dataset.

A new option to edit the choice of base classifier has been added to experiment SCARGC to apply different base classifiers, different clustering algorithms and pool size/period of the data streams could be configured in the SCARGC algorithm. Figure 3.9 showing SCARGC algorithms applying Naïve Bayes classifier and CluStream Clusterer.

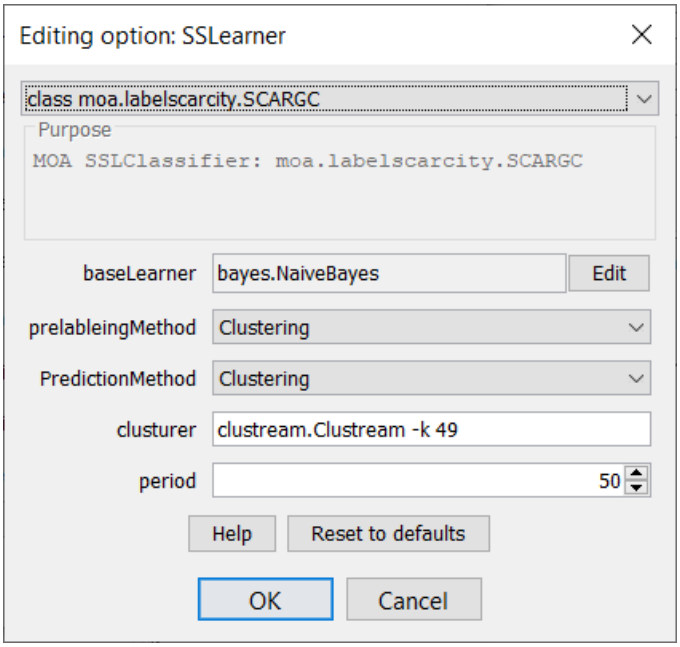


Figure 3.9 Editing option for configuring the SCARGC algorithm in MOA.

## 3.4 Discussion

The presented chapter delves into the investigation of research questions outlined in Chapter 1, particularly focusing on the challenge of evolving environments (EVL) and its impact on prediction accuracies. The literature review established the existence of various algorithms designed for detecting drift in streaming data, with online ensemble classifiers like DWM showing success in enhancing accuracy in online and incremental learning. Considering the findings, the chapter suggests the development of an 'Online SSL' method tailored for data streams with the ability to adapt to concept drifts under EVL conditions. The implementation of the SCARGC algorithm in MOA served as a benchmark for comparisons in a controlled environment. The investigations lay the groundwork for understanding the shortcomings of existing EVL approaches, particularly in preserving the diversity of online classifiers.

## 3.5 Summary

This chapter focused on addressing RQ1 and RQ2 in close relation to the influence of scarcity of true class labels and diversity of online classifiers under NSEs and its effect on predictive performances with the help of real scenarios. To analyse the effect of EVL under concept drifts, synthetic data streams and real-world datasets were applied. Diversity was found to be one of the key characteristics of online ensembles, i.e. multiple learning algorithms staked together. To analyse the diversity of ensemble classifiers, Homogeneous vs. Heterogeneous and static vs. dynamic ensembles were investigated on the synthetic data streams.

The experimental design consists of developing a new evaluation task for EVL in which each new example is used to test the prediction model and the same example is then used to train

the model. The new mechanism also allows the users to define the size of initially labelled data and purposely removes the true class labels from the remaining part of the data streams. Synthetic data streams SEA and HyperPlane were generated by inducing sudden and incremental drifts respectively. To analyse the effect on real-world datasets 'Keystroke' was evaluated for which the information about the true drift locations and their type is not available. Furthermore, to analyse the effect of EVL when no concept drifts existed, a RandomTree data stream was generated. This design helped in determining the influence of concept drifts on EVL conditions. The results showed that under EVL all the datasets underperformed in terms of prediction accuracies when concept drifts were present, furthermore, highly affected the keystroke dataset and incremental drifts in the data streams. The predictions get worse after the concept drifts, as no true class labels are available to learn the new concepts. It was also analysed that when no drifts were present in the data streams, EVL did not significantly affect the prediction capabilities.

To investigate the diversity of online classifier ensembles, DWM and WMA were investigated for the 'heterogeneity' factors. The results showed that due to a small amount of initial labelled data, it is difficult to choose the correct type of base learning algorithm i.e. NB and HT in the ensemble. Perhaps on one data stream, the prediction accuracy of HT was found higher than the NB classifier. Apart from that, on a different data stream, the prediction accuracy of NB was found higher than HT. Therefore, it is extremely difficult to choose the best-performing classifiers for specific problem.

In the perspectives of the above findings, it is worth developing an 'Online SSL' method for data streams with the ability to adapt to concept drifts under EVL. For benchmark comparison, the SCARGC algorithm was implemented in MOA to enable the experiments on data streams and to standardise the comparisons in a controlled environment. The investigations in this chapter also built a concrete foundation to investigate the root cause of the failure of existing EVL approaches due to the diversity of online ensembles. This chapter supported achieving the goal of overcoming the problems of losing the diversity of online classifiers due to the exclusion of base learners from the ensembles, which will be addressed in Chapter 4.

# Chapter 4 Heterogeneous Online Learning Ensemble for NSEs

This chapter aims at addressing the Gap 1 which was raised in Chapter 1, i.e. dealing with the diversity of online learning models. More specifically, answering the (RQ2), i.e. overcoming the problems of existing dynamic ensembles that may undergo loss of diversity due to the exclusion of base learners. Even though it is well known that various types of predictive models (e.g. Naïve Bayes, Hoeffding Trees, Multilayer Perceptron, etc.) can provide a very different predictive performance depending on the problem being tackled, little work has been dedicated to the investigation of what type of predictive model is most adequate over time in NSEs.

Several approaches to handling concept drift can be found in the literature. Most studies in this area are concerned with how to quickly detect and/or adapt to concept drift. In particular, "Active approaches use methods to explicitly detect concept drifts. If a drift is detected, new predictive models are typically created to learn the new concept, thus helping the system to recover from the concept. Passive approaches do not use concept drift detection methods. Instead, they usually maintain an ensemble of predictive models called "base models and use weights to emphasise the models believed to best represent the current concept. These approaches also typically create new base models and enable the deletion of old base models to help in dealing with concept drifts.

For instance, when delivering online learning, it is difficult to know which type of machine learning algorithm would be best to use as a base model for an ensemble learning algorithm beforehand, due to the initially small amount of data available for evaluating base models. However, as more data is received, it is desirable that online ensemble learning algorithms automatically identify which types of base learners work best for the application domain. In addition, if the best type of base learner changes due to concept drift, online ensemble learning algorithms should also be able to automatically identify which types of models are best suited to the situation encountered after concept drift.

## 4.1 The HDWM Algorithm

In this research, an online heterogeneous ensemble learning algorithm has been proposed for NSEs known as the Heterogeneous Dynamic Weighted Majority (HDWM). The purpose of introducing this new algorithm is to determine the best types of base models to be used over time in NSEs. This will enable to keep different types of base models and use them to improve predictive performance to manage concept drift.

An overview of the proposed HDWM approach is shown in Figure 4.1. HDWM maintains a dynamic list of learners. In Stage 1, the seed learners $\mathcal{E}_1$ to $\mathcal{E}_a$ are initialised. In Stage 2, the

learners in the dynamic learners' collection are prequentially tested on each instance in the data stream. In Stage 3, the same instance is used for training the dynamic list. In Stage 4, the predictions from each of the base learners are combined and based on the weights given to the base classifiers, the global predictions are generated. On globally wrong predictions a best performing base learner is cloned from the list of seed learners and added to the dynamic list. The max size of the dynamic list is controlled using parameter $B_{max}$. The learners of the ensemble ($\mathcal{E}_m$) make their predictions use their corresponding weights $w_m$.

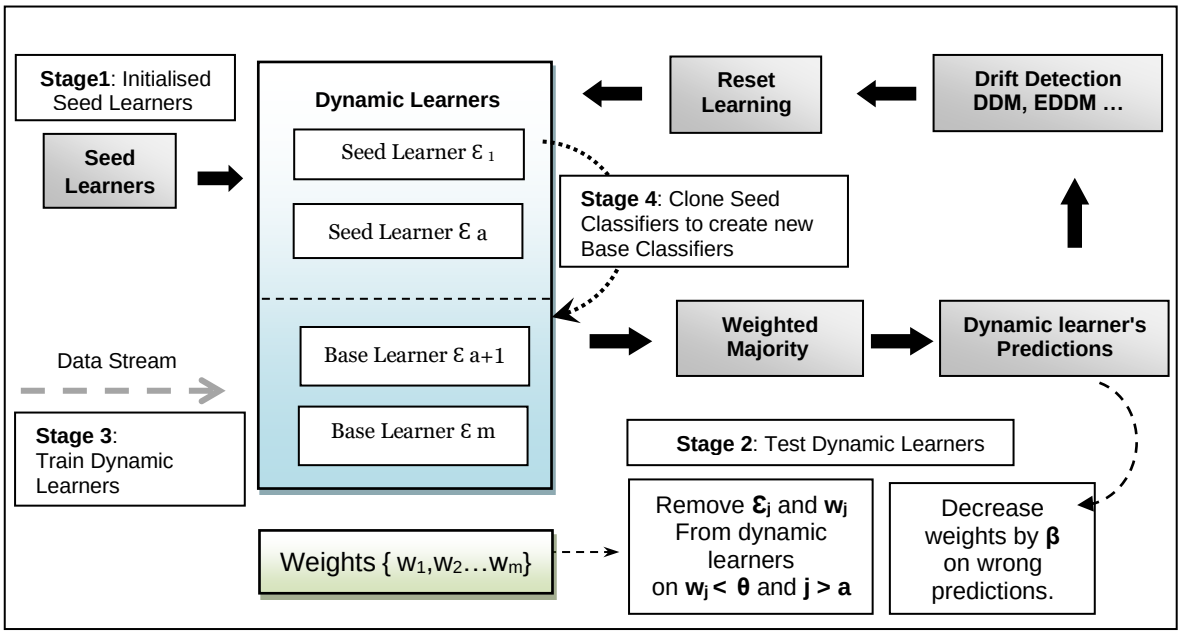


Figure 4.1 Overview of HDWM.

The global predictions on instances $x_i$ for class label $y'_i$ from a set of classes 'C' is based on the prediction made by 'm' base learners in dynamic list, $\mathcal{E}^j(x_i) \in C$. The ground truth for each example consists of pairs ($x_i$, $y_i$), and the aim was to combine the weighted predictions of each learner using their corresponding weight $w_j$ using majority voting as shown in Equation 4.1.

$$\acute{y}_i = \arg\max_{c \in C} \sum_{j \,|\mathcal{E}^j(x_i)=c}^{n} \omega_i^j \qquad (4.1)$$

Each learner in Ɛ is associated to a weight {$w_1$, w2... $w_m$}. The method to update the weights is similar as defined in Dynamic Weighted Majority (DWM) [10], i.e. by being multiplied by a factor β ($0 \leq \beta < 1$) upon misclassifications at time-steps multiple of Period 'ρ', where ρ >= 1 is a predefined parameter set by user. HDWM implements both an active and passive approach for handling concept drifts, so that it can efficiently deal with different types of drift (gradual and abrupt). To implement a passive approach, HDWM removes weaker

learners and their associated weights from the dynamic list once their weights fall below the value predefined in parameter θ. After every 'ρ' time-steps, it performs the following tasks.

1) When the global prediction of the ensemble is wrong, a new learner is cloned from the "best" seed. The best seed corresponding to the base learner in Ɛ with the best weight.
2) Once the ensemble size exceeds a user predefined threshold $B_{max}$, remove the base learner among $Ɛ_j$, $a+1 \leq j \leq m$, which has the lowest weight.

These two approaches restrict the ensemble size to reduce the computational costs while enabling the ensemble to remain heterogeneous. To implement an active approach, HDWM uses parameter δ to select a concept drift detection method, e.g. DDM [60] or EDDM [62] and link it to each base learner in the ensemble. The predictions taken from the base learners are injected into their corresponding drift detection methods to detect concept drifts and warnings. To handle concept drifts, HDWM has two options 1), reset the learning of the seeds and their corresponding weights and re-train them. 2) delete the weakest learners and create new learners of the same type as the best performing learner by cloning its seed.

The HDWM is outlined in Algorithm 4.1, initially, the seed learners '$Ɛ_1$ to $Ɛ_a$ are initialised based on their base learning algorithm (line 2). Each learner in the dynamic list is assigned equal weight 1.0 (line 3). Each base learner $Ɛ_j$ in the dynamic list is asked for predictions on '$x_i$' instances (Line 8), where 'i' is the time-step and 'x' is the vector representing attributes in the data-stream. Similar to the DWM rule [10] the weights of the learners are decreased on incorrect predictions (Line 10-11). Over time when the ensemble grows, the base learners whose weights fall below θ are deleted while keeping intact the seeds in Ɛ for future use (Line 13-15) and set the flag d = 1 which indicates that the base learner has been deleted. By ensuring that at least one base learner of each type is maintained in Ɛ, it is certain that a given type of base learner can repopulate the ensemble whenever it becomes beneficial, even if this follows a period when this type of base learner was not beneficial.

**Algorithm 4.1 HDWM ($\{x,y\}_n^1$, β, θ, ρ)**

**Input**: $\{x, y\}_n^1$ : Stream of examples and class label
$\{\text{LearningAlgorithm}\}_a^1$: Set of Heterogeneous Seed Base Learning Algorithms
β: factor to decrease weights, $0 \leq \beta < 1$
θ: threshold to delete base learner
ρ: period between base learner removal, creation, and weight update
$\{\varepsilon,w,\delta\}_m^1$ : Set of Seeds, Dynamic learners, and Drift Detection Method
d {0,1}: base learner delete flag
$B_{max}$: Max size of ensemble
$c \in \mathbb{N}^*$: Number of classes, $c \geq 2$
$\wedge, \lambda \in \{1,...,c\}$: global and local predictions
$\sigma \in \mathbb{R}^c$: sum of weighted prediction for each class

```
1   for seed = 1 to a                                   // Loop over seeds
2       εseed ← Initialised_Seeds                       // Clone seeds to Dynamic List
        (LearningAlgorithmseed)
3       wseed ← 1.0
4   end for
5   for i = 1 to n                                      // Loop over examples
6       for j = 1 to m                                  // Loop over ensemble of learners
7           d ← 0                                       // Learner's delete flag
8           λ = Classify (εj, xi )                      // Classify using learners
9           if (i mod ρ = 0) then
10              if (λ≠ yi ) then
11                  wj ← β wj                           // Update weight using DWM rule
12              end if
13              if (wj < θ and j > a) then              // j>a prevents deletion of seeds
14                  {εj, wj } ← remove ({εj, wj }, θ)   // Delete learners, weights < θ
15                  d ← 1;                              // Set deleted flag to True
16              end if
17              if (d ≠ 1) then                         // If no learners are deleted
18                  σλ ← σλ + wj
19                  wmin ← min(w), wmax ← max(w)
20              end if
21          end if
22          Call Active Drift Handler (λ , ε, xi ) (Algorithm 4.2)
23      end for
24      ∧ ← argmaxj σj
25      if (i mod ρ = 0) then
26          w ← normalise-weights ();                   // Using DWM rule
27          if (∧ ≠ yi) then                            // Global prediction is wrong
28              Call Passive Drift Handler (Algorithm 4.3)
29          end if
30          if size(ε) = Bmax then
31              { ε,w} ← remove({ε,w}, wmin)
32          end if
33          for i = 1 to m
34              Train (εi, xj )
35          end for
36      end if
37  end for
```

If no learner is deleted (line 17), the base learner's prediction is used to compute the weighted sum for each class (line 18). The maximum and minimum weights are stored in appropriate variables (line 19). The class with the most weight is then set as the global prediction (line 24). Weights are normalised (Line 26) and the parameter ρ is used to control the period for adding or removing the new dynamic learners.

An active drift detection method such as DDM or EDDM is invoked (line 22) and in the case of drift detection by any of the base learners, the Active Handle Drift (Algorithm 4.2) is invoked. The integration method for active drift detection is explained in Section 4.2. On global wrong predictions (Line 27) the Passive Drift Handler (Algorithm 4.3) is invoked on (line 28). To control the ensemble size (line 30-32) parameter $B_{max}$ is a user defined value to remove weaker learners from the dynamic learners list.

### 4.1.1 Active Drift Handling

Algorithm 4.2 outlines Active drift handling in HDWM. The seeds are reset upon the occurrence of drifts. The weight of the seeds is set to 0.5 instead of 1.0 (Lines 3-6) to prevent the domination of seeds over the new base learners. Finally, the seed learners are trained when the warning state is detected.

**Algorithm 4.2 HDWM Active drift Handling ($\lambda$, $\varepsilon$, $\delta$, w, $x_j$)**

Input: $\varepsilon$: Set of Seeds and Dynamic learners
$\lambda$: local predictions from base learners
w: ensemble weights
$\delta$: Drift detection Method

**Output**: none

```
1   δ_local ← DriftDetectionMethod(λ)
2   if (δ_local drift = true ) then          // drift is detected
3     | for seed = 1 to a
4     |   | ε_seed ← reset
5     |   | w_seed ← 0.5
6     | end for
7   end if
8   if (δ_local warning = true) then     // warning is detected
9     | for j = 1 to a                  // Loop over seed learners
10    |   Train (ε_i, x_j )
11    | end for
12  end if
```

### 4.1.2 Passive Drift Handler

Algorithm 4.3 implements the Passive drift handling mechanism in HDWM. In the case of globally wrong predictions the index position and the type of best seed learner is determined (line 1), a new classifier of a similar type is created (line 2) and added to the list of dynamic learners from the seed learner (line 3). New learners are given weights 0.5 (line 5) to prevent new learners dominating over the existing ones.

**Algorithm 4.3 PassiveHandleDrift (ε, w)**

Input: ε: Set of Seeds and Dynamic learners
w: ensemble weights
$w_{max}$ : maximum weight
m: size of the dynamic learners
$\{LearningAlgorithm\}_a^1$: Set of Seed Base Learners
**Output**: none

1 Seed ← bestLearner { ε, $w_{max}$ }
2 $N_{seed}$ ← Initialised_Seeds {$LearningAlgorithm_{seed}$}
3 ε ← ε U $N_{seed}$ // append classifier to dynamic list
4 m ← m+1
5 $w_m$ ← 0.5

## 4.2 Integration with DDM

The purpose of integrating DWM with DDM is to cope with drift when it is detected and take appropriate measures. Possible options include removing the poor performing base classifiers from the ensemble or resetting the weights when a drift is detected. The newly developed DWM-DDM (Dynamic Weighted Majority with Drift Detection) algorithm uses an online ensemble and an explicit drift detection method for detecting changes in environment.

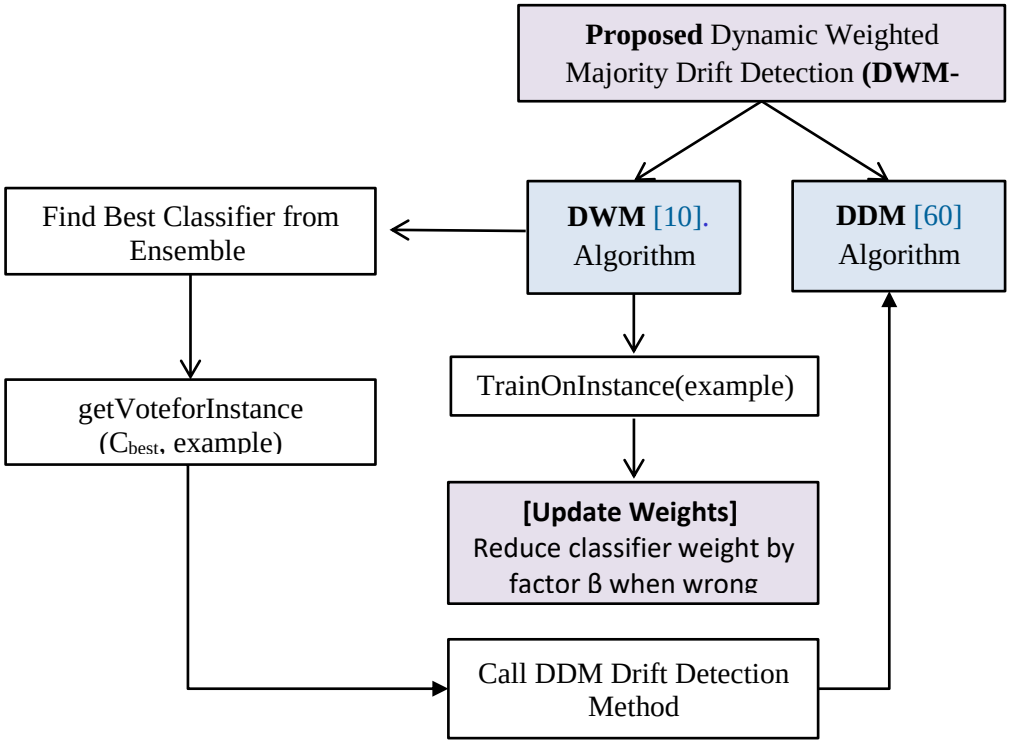


Figure 4.2 Integration of DWM and DDM using online ensemble and an explicit drift detection method for detecting changes in environment.

The integration mechanism is based on a set of ensemble classifiers accepting input "x" and predicting class label y', at the same time, the drift is detected based on the true class label "y" as shown in Figure 4.2. DWM uses fixed numbers of base learners C = ($C_1$, $C_2$ . . . $C_L$) with an initial weight '$w_i$' equal to '1'. The weight is penalised by a factor of ratio β on each wrong prediction i.e. ($w_i \leftarrow \beta w_i$). The best classifier $C_{best}$ from the ensemble is chosen for the prediction; the process is based on the highest weights of classifiers. The prediction results are sent to the DDM algorithm for detecting the drift. The DWM-DD penalises the weights of the classifiers who predicts the wrong class labels and reduces the weights of the classifiers by a factor of β which is a user provided value between 0 and 1. The warning level in DDM is reached if $p_i + s_i \geq p_{min} + 2 \times s_{min}$ and the drift level is reached if $p_i + s_i \geq p_{min} + 3 \times s_{min}$. Where $p_i$ and $s_i$ are the error rate and standard deviation at instant 'i'. $p_{min}$ and $s_{min}$ are the minimum recorded error rate and standard deviation.

**Algorithm 4.4 HDWM-DDM ($\{x,y\}_n^1$, β, δ, ρ)**

Input: $\{x, y\}_n^1$ : Stream of examples and class label
$\{\varepsilon,w\}_m^1$: Classifier with associated weights
δ : Drift detection algorithm such as DDM or EDDM
β: factor to decrease weights, $0 \leq \beta < 1$
ρ: period between weight updates

```
1   w ← 1
2   for i = 1 to n                          // Loop over examples
3     for j = 1 to m                        // Loop over ensemble of learners
4       λ ← Classify (εj, xi )
5       if (i mod ρ = 0) then
6         if (λ≠ yi ) then
7           wj ← β wj                       // Update weight using DWM rule
8         Cbest ← max(C, wi)
9       end if
10    end for
11    y' ← Classify (Cbest , xi )
12    drift_status ← δ (yi , y' )           // drift detection using DDM or EDDM
13    if (i mod ρ = 0) then
14      w ← normalise-weights ();
15      for i = 1 to m
16        Train (εi, xj )
17      if drift_status = True then
18        ResetLearning (ε, w)
19    end if
20  end for
```

The pseudo-code for DWM-DD is outlined in Algorithm 4.4; the algorithms iterate through the training examples. Each classifier $\varepsilon_j$ in the ensemble is asked for predictions on '$x_i$' instances (Line 4), where 'i' is the time-step and 'x' is the vector representing attributes in the data-stream. At an interval of ρ (line 5) the weights of the learners are decreased on incorrect predictions (line 6-7) similar to the DWM rule [10]. Next, the classifier having a max weight is stored in $C_{best}$ (line 8). The $C_{best}$ is asked to predict the y' class labels (line 11).

To implement an active approach, DWM-DD uses parameter δ to select a concept drift detection method, e.g. DDM [60] or EDDM [62] (line 12). The prediction 'y' is injected into drift detection methods to detect concept drifts and warnings. To handle concept drifts, DWM-DD resets the learning, and their corresponding weights (line 18) are re-trained.

## 4.3 Analysis of HDWM

This Section investigates the HDWM algorithms and compares their accuracy and drift handling capabilities with WMA (due to its heterogeneity) and DWM (due to its ability to dynamically include and exclude base learners from the ensemble). Different variations of HDWM are compared to evaluate its sensitivity to parameters (e.g. drift and warning threshold, ensemble size) and variations of the algorithm that deactivate some of its characteristics (e.g. drift detection, warning detection, weighted vote). The second set of experiments concern the evaluation of computational resources usage (CPU time and RAM-Hours). Finally, experiments were presented comparing HDWM and other state-of-the-art ensemble classifiers.

For evaluating learning models in the data stream, three methods used are prequential [118], holdout [51] and Kappa [52] evaluations. Prequential evaluation, also known as interleaved test-then-train, is applied due to the streaming and non-stationary nature of data. This method is applied because it provides a consistent and fair evaluation metric over time. Traditional offline evaluation metrics such as precision, recall or F1 score may not be suitable for data streams due to the evolving nature of the data. Holdout evaluation provides a mechanism for assessing the performance of a model in a controlled manner while dealing with the challenges posed by streaming data. This method was applied to help validate the effectiveness of HDWM by assessing its performance on separate subsets of data that the model has not seen during training. Since prediction accuracy can be misleading on datasets with class imbalance or temporal dependencies, Kappa M and Kappa Temporal were also used. Kappa M has advantages over Kappa statistic as it has a zero value for a majority class classifier [52]. Kappa Temporal is applied since it replaces the majority class classifier with the NoChange classifier [127]. This enables better estimations for datasets with temporal dependencies.

The prequential accuracy is calculated based on the MOA Windows Classification Performance Evaluator [46] with a window size of 1000. This evaluator facilitates the evaluation of classification models on streaming data in a sliding window. The window moves through the stream, and at each step, the model is evaluated on the most recent instances within the window. In HDWM, the evaluator helps detecting concept drift by monitoring changes in model performance within the sliding window. The Holdout method uses predefined partitions of train and test instances. However, it requires labelled test datasets which are difficult to obtain readily for real-world applications. The Holdout method is applied in STAGGER (Drift) as predefined partitions of training and testing instances were used; the details are explained in Section 4.4.

## 4.4 Data Streams

The artificial data streams used in the experiments are generated through the MOA workbench [46] are described in Section 4.4.1 and the real-world datasets are described in Section 4.4.2. The artificial datasets are generated so that the true position of concept drift is known, whereas in real-world datasets the true location of the drifts is not available. The description of the data streams and parameters are shown in Table 4.1.

Table 4.1 Description of the data streams and parameters.

| Streams | # Instances | # Features | Classes | # Drifts | Period | Freq. | Evaluation |
|---|---|---|---|---|---|---|---|
| SEA (S) | 2500 K | 3 | 2 | 2 | 50 | 1K | P |
| STAGGER (S) | 12 K | 3 | 2 | 2 | 1 | 1 | H |
| RTree R | 100K | 10 | 2 | 2 | 50 | 1K | P |
| LED (S) | | 7 | 10 | 1 | | | |
| Wave (S) | | 40 | 3 | 1 | | | |
| Hyperplane (G) | | 10 | 2 | 3 | | | |
| SEA (G and S) | | 3 | 2 | 2 | | | |
| RRBF(G) | | 2 | 5 | 2 | | | |
| Electricity | 45,312 | 8 | 2 | N/A | 50 | 100 | P |
| Spam | 9,324 | 500 | 2 | | | 500 | |
| Sensor | 100K | 5 | 58 | | | 1K | |
| Forest Cover | 100K | 54 | 7 | | | 1K | |

[P] = Prequential Evaluation, [H] Periodic Holdout Evaluation,
[R]= Recurrent Drift, [S] = Sudden Drift, [G] =Gradual Drift

### 4.4.1 MOA Data Streams

The details of the streams are given below, and the MOA commands to generate these streams are available in APPENDIX I.

- SEA data stream contains three attributes, function $x_i \in R$ and the value of $x_i$ is between 1.0 and 10.0. The target concept is determined using the equation $y = [x_0 + x_1 + x_2 \leq \theta]$, such that $\theta \in \{7, 8, 9, 9.5\}$.
- RandomTrees generates a stream based on a randomly generated tree.
- LED generates a stream defined by a 7-segment LED display and the task is to predict the digit (0-9).

- Hyperplane is a flat n-dimensional space useful for simulating gradually drifting concepts. The orientation and position can be modified by slightly changing its relative size of the weights.
- Random Radial Basis Function (RRBF) consists of a fixed number of randomly positioned centroids with a single standard deviation, class label and weight.

### 4.4.2 Real-World Datasets

Sensor dataset [128] deployed in the Intel Berkeley Research Lab, the sensor ID feature is used to label the class. The dataset consists of 220k instances; the input attributes include time-stamped topology information, along with humidity, temperature, light and voltage. The true drift locations are not known, but gradual drifts exist as the light during working hours is generally stronger than at night, and the temperature readings of specific sensors may rise if there are meetings in the room.

**Spam email dataset** [129] contains input attributes that represent a gradual concept drift by the SpamAssassin collection. The dataset consists of 9,324 instances, 500 attributes and two target classes i.e. spam and legitimate. The attributes are representing the presence of a given word in the email.

**Electricity dataset** [130] contains data consisting of 45,312 instances for a period of two years collected from the Australian New South Wales Electricity Market. Input attributes include day of the week, the NSW electricity demand, the Victoria electricity demand, and the scheduled electricity transfer between states. The binary prediction task is to identify the change (up or down) of the price relative to a moving average. The concept drift appears due to changes in consumption habits due to unexpected events and seasonality.

**Forest Cover type** [131] dataset consists of the observation (30 x 30 metre cell) determined from the US Forest Service (USFS) Region 2 Resource Information System (RIS) data. The task is to predict the type of forest cover from cartographic variables such as Elevation, Slope, soil type etc.

## 4.5 Test Configuration

All the experiments are evaluated in terms of time and predictive performance. Processing time is measured in seconds and is based on the CPU time used for training and testing. All the experiments were performed on machines with Core i7 @ 3.4 GHz, 4 GB of RAM and experiments are presented in terms of CPU time. All experiments were executed within the MOA [46] framework.

The cross-validation techniques for measuring model performance are not suitable as the data streams originate from NSEs. Therefore, the prequential method [46] was used, which is a commonly accepted estimation procedure in NSEs. In this method each example is first used to test the model before it is used for training. The advantage of this method is that all the

instances are used in training and testing, and therefore no specific holdout set is needed. To determine the statistically significant differences between algorithms, non-parametric tests were carried out using the Demsar's methodology from [49] For the statistical test the Friedman test was applied with $\alpha = 0.05$ and the null hypothesis, "no statistical difference between the algorithms". If the null hypothesis was rejected, the Nemenyi [50] post hoc test was used to identify which pairs of algorithms differ from each other.

Table 4.2 Parameters used in the experiments.

| Code | Description |
|---|---|
| β | Penalise learner's weight on wrong prediction |
| θ | Threshold of weights to remove base learners |
| Period | The interval to create or remove base learners or to manipulate their weights |
| Freq. | The number of training examples between samples of learning performance |

The base learners used in DWM are NB (Naïve Bayes) and HT (Hoeffding Tree). HDWM and WMA are using four base learners, i.e. HT-MC (Majority Class at leaves), HT-NB (Naïve Bayes at leaves), HT-NBAdaptive and NB. The values $\beta = 0.05$ and $\theta = 0.01$ are used as per the default values used in DWM. Table 4.2 gives a description of the parameters used in the experiments. For the large data streams (size > 100K) and real-world datasets, the period is '50'. For small datasets, the period is '1'. 'Freq' is the MOA sample frequency parameter corresponding to the number of training examples between samples of learning performance. Freq=1k is used for instances more than 100k and for smaller streams a lower value is applied. To investigate the heterogeneity and its influence on active and passive drift handling approaches, a variant of HDWM, HDWM-P was developed which is heterogeneous although not utilising the Active Drift handling option. This variant is used in the experiments in Section 4.8. The details of variants used in the experiments are described in Table 4.3.

Table 4.3 Variants used in the experiments.

| Algorithms | Description of Algorithm |
|---|---|
| HDWM | HDWM uses Naïve Bayes and Hoeffding Tree; its Heterogeneous ensemble uses both Active and Passive Drift Handling. |
| HDWM – P | HDWM uses Passive Drift Handling, as used in Heterogeneity Analysis. |
| DWM-NB | DWM algorithm using Naïve Bayes as base classifier |
| DWM-HT | DWM algorithm using Hoeffding Tree as base classifier |
| WMA | WMA using HT-MC (Majority Class at leaves), HT-NB (Naïve Bayes at leaves), HT-NBAdaptive and NB |

## 4.6 Analysis on MOA Streams

This section analysis the HDWM algorithm on MOA data stream and real-world datasets. The predictive accuracies and kappa statistics have been compared against with DWM-NB, DWM-HT and WMA.

### 4.6.1 Predictive Performance

The predictive capabilities of our new approach were tested on artificial data-streams and real-world datasets, corresponding ranks are determined such that higher averages are representing lower ranks. Significance tests and post hoc comparisons on ranks are performed to determine significance level and CD. The predictive accuracies of HDWM, DWM and WMA are shown in Table 4.4.

Table 4.4 Predictive Accuracies (%) of DWM-NB, DWM-HT, WMA and HDWM.

| **Streams** | **HDWM** | **DWM-NB** | **DWM-HT** | **WMA** |
|---|---|---|---|---|
| SEA (S) | 88.12 (1) | 87.98 (2) | 87.71 (3) | 85.79 (4) |
| STAGGER (S) | 82.8 (1) | 81.82 (2) | 81.26 (3) | 55.08 (4) |
| RTree R | 84.42 (1) | 74.05 (4) | 75.32 (3) | 79.78 (2) |
| LED (S) | 73.37 (3) | 73.41 (1.5) | 73.41 (1.5) | 65.01 (4) |
| Wave (S) | 82.16 (1) | 80.31 (4) | 80.34 (3) | 80.65 (2) |
| Hyperplane (G) | 88.12 (2) | 88.08 (3) | 88.19 (1) | 80.54 (4) |
| SEA (G and S) | 87.64 (1) | 87.58 (2) | 87.21 (3) | 85.71 (4) |
| RRBF(G) | 92.59 (3) | 92.65 (2) | 93.09 (1) | 77.93 (4) |
| Electricity | 88.4 (1) | 79.73 (4) | 84.06 (2) | 80.92 (3) |
| Spam | 90.54 (1) | 87.83 (4) | 88.39 (2) | 88.04 (3) |
| Sensor | 92.68 (1) | 90.79 (3) | 90.96 (2) | 72.86 (4) |
| Forest Cover | 89.8 (1) | 82.92 (2) | 79.33 (4) | 80.65 (3) |
| **Avg. Ranks** | **1.42** | 2.79 | 2.38 | 3.42 |

Table 4.5 Kappa Temporal DWM-NB, DWM-HT WMA and HDWM.

| **Streams** | **HDWM** | **DWM-NB** | **DWM-HT** | **WMA** |
|---|---|---|---|---|
| SEA (S) | 73.81 (1) | 73.47 (2) | 72.87 (3) | 68.84 (4) |
| STAGGER (S) | 49.2 (1) | 40.14 (2) | 39.44 (3) | -19.43 (4) |
| RTree R | 68.69 (1) | 47.73 (4) | 50.34 (3) | 59.35 (2) |
| LED (S) | 70.54 (1) | 70.47 (2) | 70.46 (3) | 61.14 (4) |
| Wave (S) | 73.36 (1) | 70.41 (4) | 70.46 (3) | 70.94 (2) |
| Hyperplane (G) | 75.05 (3) | 76.14 (2) | 76.37 (1) | 61.07 (4) |
| SEA (G and S) | 71.68 (3) | 73.03 (1) | 72.22 (2) | 66.69 (4) |
| RRBF(G) | 91.14 (2) | 91.13 (3) | 91.66 (1) | 73.39 (4) |
| Electricity | 16.91 (1) | -44.88 (4) | -14.83 (2) | -36.85 (3) |
| Sensor | 92.5 (1) | 90.78 (3) | 90.95 (2) | 72.84 (4) |
| Forest Cover | -153.1 (1) | -361.2 (3) | -163.1 (2) | -388.9 (4) |
| **Avg. Ranks** | **1.45** | 2.73 | 2.27 | 3.55 |

Table 4.5 provides the Kappa measures for the experiments. The larger the Kappa value, the more generalised the classifier, negative Kappa values indicate low predictive accuracy. Kappa values for Spam and Forest Cover datasets were negative in HDWM, DWM and WMA due to the large numbers of attributes in these datasets.

In both drift and real-world data streams, the $\chi^2_r$ statistic is 15.25 (df =3, N = 12) and the p-value 0.0016 shows significant differences at the level of significance of 0.05. The method to calculate chi-squared and p-value is described in the paper [49]. The Nemenyi test [50] was applied for pairwise comparison. The CD is 1.35. It is evident from the box plot in Figure 4.3 that HDWM performed significantly better than DWM-NB i.e. (2.79 – 1.42 = 1.38 > 1.35) and WMA (3.42 – 1.42 = 2.0 > 1.35).

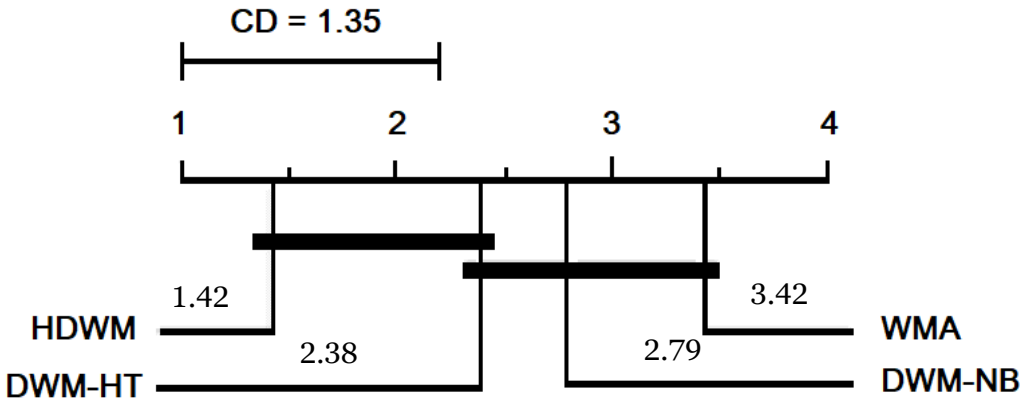


Figure 4.3 Bar chart for pairwise comparisons of ranks on predictive accuracies (%) between HDWM, DWM-HT, DWM-NB and WMA showing HDWM performed significantly better than other approaches.

The statistical tests applied on Kappa Temporal on drift and real-world streams, with the $\chi^2_r$ statistic of 15.76 (df =3, N = 11) and the p-value of 0.0012 showed significant differences at the level of significance of 0.05. In statistical tests for Kappa M on both drift and real-world streams, the $\chi^2_r$ statistic is 15.10 (df =3, N = 11) and the p-value 0.0017 also shows significant differences at the level of significance of 0.05. The Nemenyi test was applied for Kappa Temporal and Kappa M for pairwise comparison. The CD is 1.41. HDWM performed significantly better than WMA.

### 4.6.2 Analysis of Results

In this section, an in-depth Analysis of the results achieved in the previous experiment are presented using the artificial data streams with concept drift. The predictive performances are analysed, and the capabilities of each algorithm are graphically presented to investigate how these algorithms react to different type of drifts. The ensemble size was also analysed. The Ensemble Size in a dynamic base classifier is an important factor for balancing performance because a larger ensemble requires more processing time but may improve predictive accuracy.

### 4.6.3 Significant Findings

In HDWM the seeds are never deleted and retain the previously learnt concepts, this helps HDWM in appropriately dealing with recurring concept drifts. Figure 4.4 (left), represents RandomTree recurring concept drifts. HDWM (85.27%) and WMA (79.78%) handled the drift on a recurring concept at 75,000 instances. DWM-NB (74.05%) and DWM-HT (75.32) were unable to cope after the first sudden drift at 25,000. The base learners in DWM forgot the previous learnt concepts due to inclusion and removal of their base learners; unlike the WMA whose base learners are never deleted.

In RRBF Figure 4.4 (right), which represents gradual drifts, HDWM (92.59%) and DWM can deal with concept drifts appropriately due to periodically including new base learners while WMA does not; this being due to its static ensemble size. HDWM not only maintained the predictive accuracy of DWM, but slightly improved it. SEA Figure 4.5 (left), represents abrupt drifts at 25,000 instances and 75,000 instances. HDWM and DWM handled these drifts appropriately, however, WMA failed to adapt to the new concept. SEA (Mixed) Figure 4.5 (right), represents gradual and sudden drifts. Gradual drift is centred around instance 25,000 with a window of 10,000 instances and is represented using a dotted line while the sudden drift occurs at 75,000 instances. DWM and HDWM both handled these drifts appropriately, but WMA reacted late on mixed concept drifts.

## 4.7 Analysis on Real-World Datasets

Several challenges emerge when dealing with real-world classification problems. The primary issues are the identification and location of the concept drifts. Accordingly, the HDWM was also evaluated on real-world data streams; namely: Sensor [128], Spam email dataset [129], Electricity [130] and Forest Cover type [131]. As there are only 4 datasets and thus 4 observations, no significance test was performed. However, the obtained results show improvements. As shown in Figure 4.6, HDWM achieved the highest predictive accuracies on Spam email (90.54%), Electricity (89.4%), Forest Cover type (91.03%) and Sensor (92.04%).

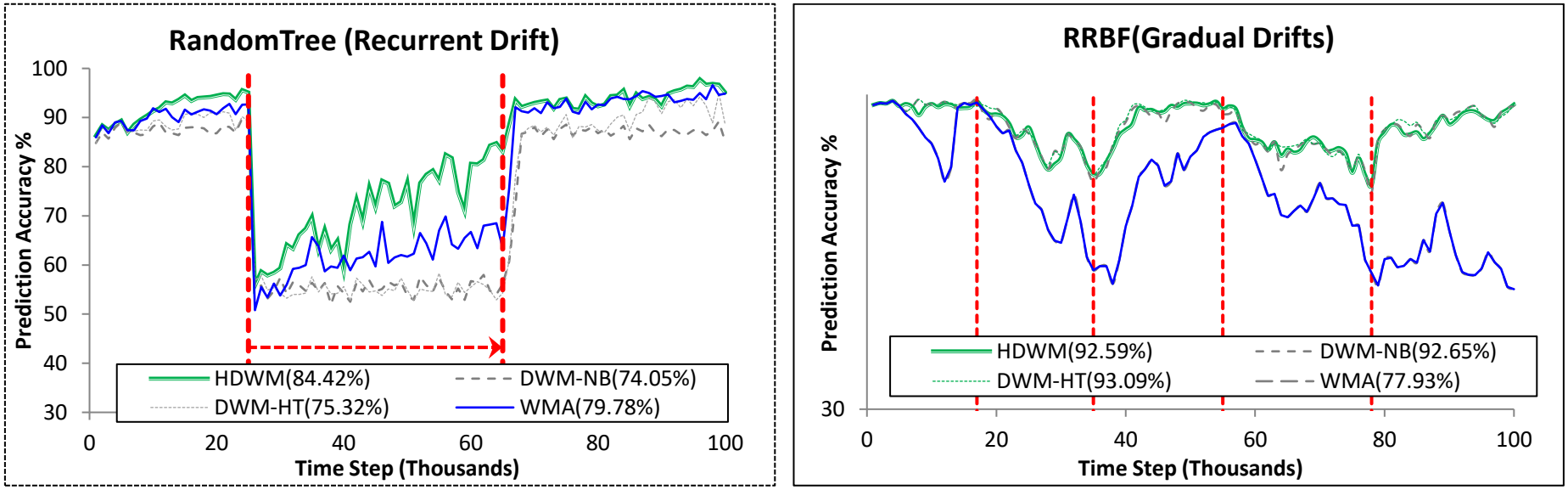


**Figure 4.4** Predictive Accuracies RandomTree (left) and RRBF (right) on Artificial Data Streams. Solid and dashed vertical black lines indicate the centre point of the drifts, and start/end of the drifts, respectively. The time steps between the start and end of the start and end of the drift (inclusive) compose the drift window.

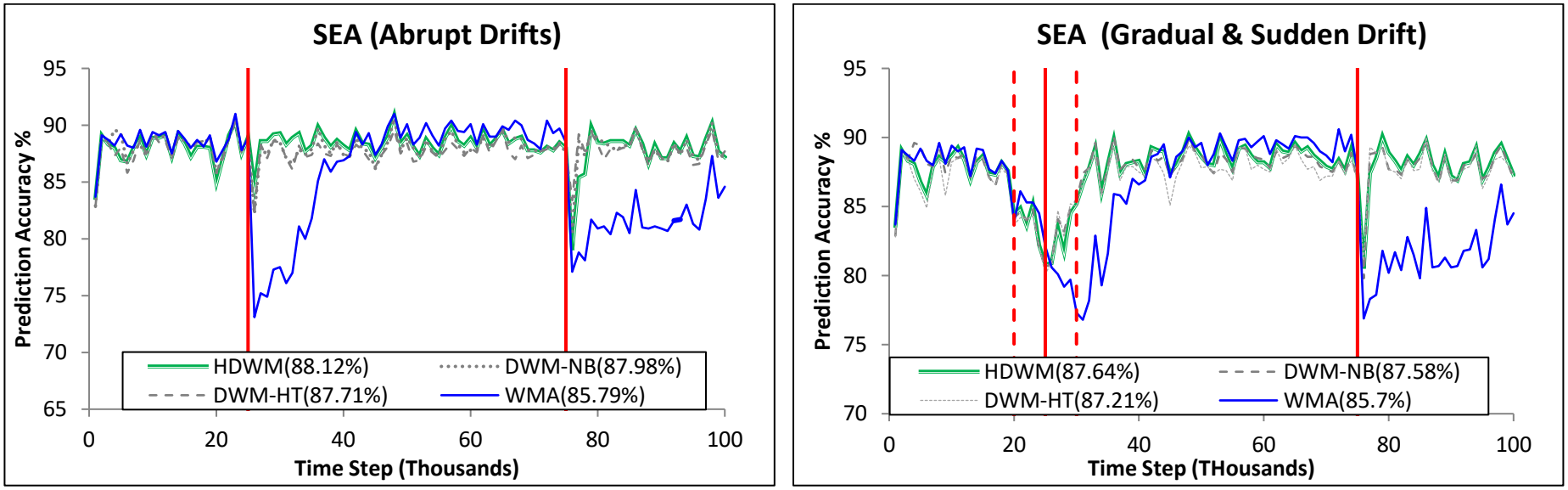


**Figure 4.5** Predictive Accuracies SEA Abrupt (left) and SEA Mixed (right) on Artificial Data Streams.

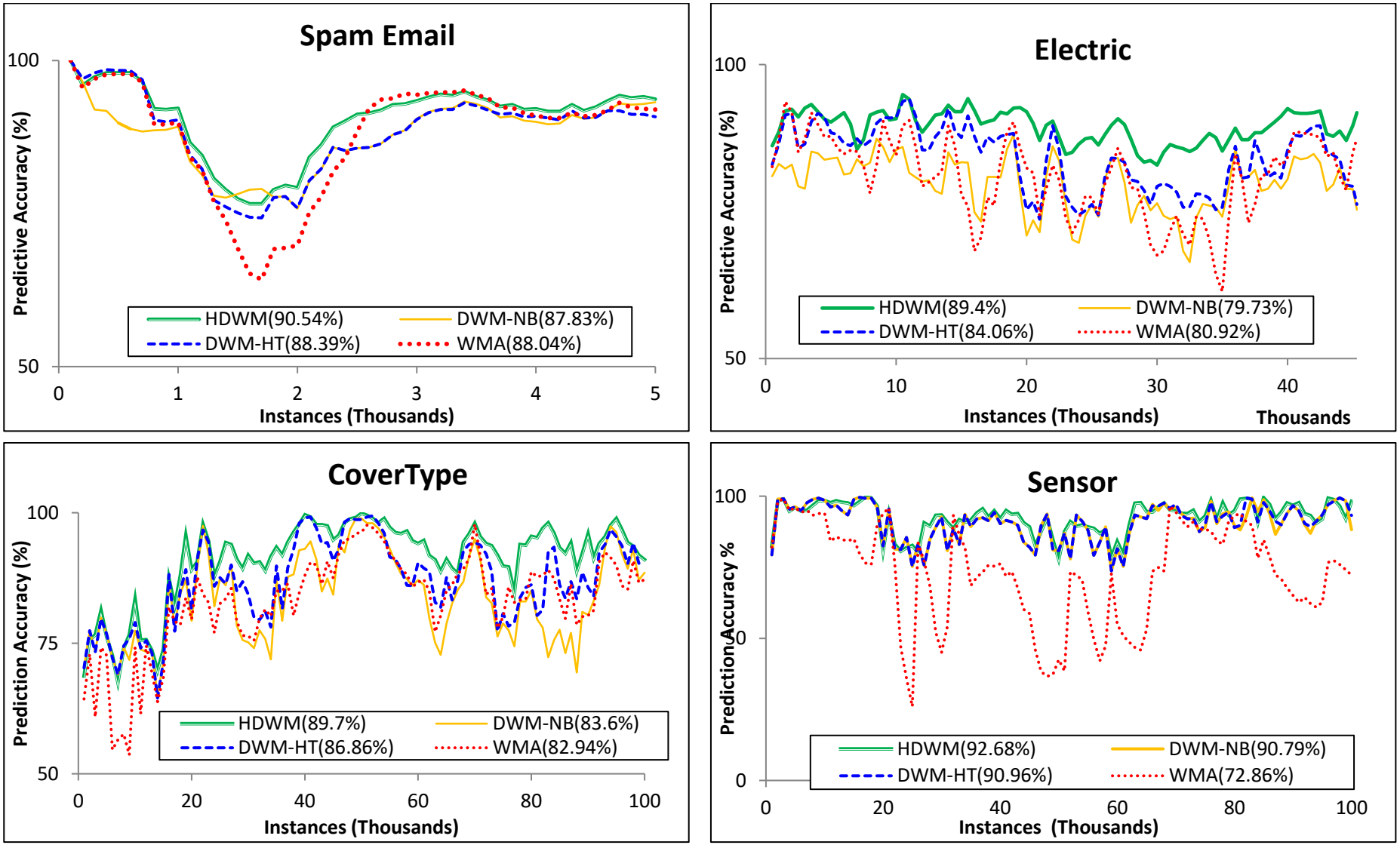


**Figure 4.6** Average Predictive Accuracies Real-world datasets.

## 4.8 Analysis of Heterogeneity

The objective of this Analysis is to investigate how the heterogeneity of an ensemble affects its predictive performance, and whether the higher accuracy achieved in HDWM is due to its heterogeneity or due to the active drift handling capabilities. The results of these experiments are shown in Table 4.6.

Table 4.6 Heterogeneity Test, Predictive Accuracies (%)

| **Streams** | **HDWM -P** | **DWM (NB)** | **DWM (HT)** |
|---|---|---|---|
| SEA (S) | 87.73 (2) | 87.98 (1) | 87.71 (3) |
| STAGGER (S) | 82.31 (1) | 81.82 (2) | 81.26 (3) |
| RTree R | 75.51 (1) | 74.05 (3) | 75.32 (2) |
| LED (S) | 73.44 (1) | 73.42 (2) | 73.41 (3) |
| Wave (S) | 80.35 (1) | 80.31 (3) | 80.34 (2) |
| Hyperplane (G) | 88.21 (1) | 88.08 (3) | 88.19 (2) |
| SEA (G and S) | 87.26 (2) | 87.58 (1) | 87.21 (3) |
| RRBF(G) | 93.04 (2) | 92.65 (3) | 93.09 (1) |
| Electricity | 84.09 (1) | 79.73 (3) | 84.06 (2) |
| Spam | 88.72 (1) | 87.83 (3) | 88.39 (2) |
| Sensor | 90.98 (1) | 90.79 (3) | 90.96 (2) |
| Forest Cover | 86.92 (1) | 82.92 (2) | 79.33 (3) |
| **Avg. Ranks** | **1.25** | **2.42** | **2.33** |

For this experiment the DWM performance was compared with the Naïve Base and Hoeffding Tree as base learners in its ensemble and compared it with HDWM-P (a variant of HDWM without active drift handling) which is reliant on a passive approach similar to the DWM. The Friedman statistics in a heterogeneity test, the $\chi^2_r$ statistic is 10.16 (df=2, N = 12) and the p-value 0.0062 indicates significant differences at the level of significance of 0.05. Post-hoc test using the Nemenyi test was applied for pairwise comparison.

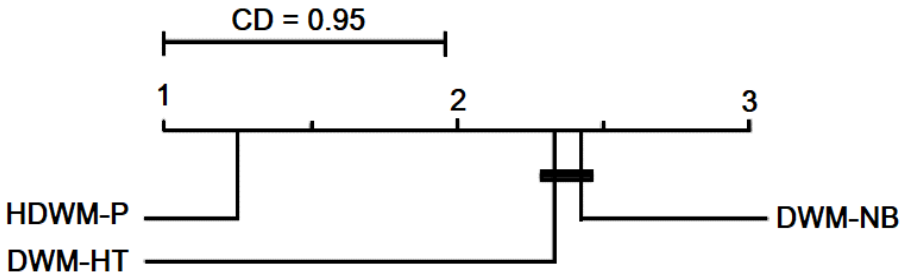


Figure 4.7 Boxplot for Heterogeneity Test.

The CD is 0.95. Boxplot in Figure 4.7 shows that HDWM-P performed significantly better than DWM-NB i.e. (2.42 – 1.25 = 1.08 > 0.95). Given that the main difference between HDWM-P and DWM is the heterogeneity, these results indicate that heterogeneity plays a key

role in improving the HDWM accuracy over DWM. In particular, the model switching mechanism maintained the accuracy and made it independent of manually selecting base learners.

## 4.9 Sensitivity Analysis of Hyperparameters

In terms of how to set the parameters in real-world problems, the difficulty is that the best values may change over time. Potentially, one could run multiple versions of the approach with different parameter settings. The parameters 'β', 'θ' and 'Period' were analysed and their effect on prediction accuracy, ensemble size and drift detections. The values for β ' and ' θ ' are randomly chosen between 0 and 1.

Table 4.7 Effect of 'Period' on Predictive Accuracies % & Drift Detection, β = 0.5 and theta = 0.01 (Fixed).

| **Streams** | **Period =1** | | **Period =25** | | **Period =50** | |
|---|---|---|---|---|---|---|
| | **ACC%** | **# Drifts** | **ACC%** | **# Drifts** | **ACC%** | **# Drifts** |
| SEA (S) | 84.0 (3) | 0 | 87.9 (2) | 2 | 88.1 (1) | 2 |
| STAGGER (S) | 85.2 (1) | 0 | 61.3 (2) | 0 | 60.8 (3) | 0 |
| RTree R | 76.5 (3) | 6 | 82.5 (2) | 1 | 84.4 (1) | 1 |
| LED (S) | 54.8 (3) | 4 | 72.2 (2) | 1 | 73.4 (1) | 1 |
| Wave (S) | 78.2 (3) | 5 | 82.1 (2) | 0 | 82.2 (1) | 0 |
| Hyperplane | 77.5 (3) | 4 | 85.5 (2) | 0 | 87.5 (1) | 0 |
| SEA (G and S) | 82.9 (3) | 0 | 87.7 (1) | 1 | 87.1 (2) | 4 |
| RRBF(G) | 90.8 (3) | 8 | 93.0 (1) | 4 | 92.6 (2) | 8 |
| Electricity | 89.4 (1) | 8 | 89.3 (2) | 2 | 88.4 (3) | 2 |
| Spam | 93.9 (1) | 2 | 89.9 (2) | 0 | 89.7 (3) | 0 |
| Sensor | 83.2 (3) | 26 | 93.6 (1) | 1 | 92.5 (2) | 3 |
| Forest Cover | 90.7 (1) | 10 | 90.4 (2) | 0 | 89.7 (3) | 0 |
| **Avg. (Ranks)** | **82.3(2.2)** | **6.08** | **84.6(1.7)** | **1.0** | **84.7(2.0)** | **1.75** |

While the period was also analysed on randomly values 1, 25 and 50. The period = 1 represents inclusion of all the instances in the data stream and then gradually increased by skipping 25 instances. The results on the 'effect of 'Period' on Predictive Accuracy and Drift Detection' is shown in Table 4.7. As evident from the table, the average prediction accuracy is gradually increasing while the number of drift detections is decreasing by applying a larger value of 'period'. The effect of 'β' on Predictive Accuracy and Ensemble Size is analysed by keeping a static value of 'Period = 50'. This value was chosen for subsequent experiments, as it achieved the highest accuracies in the experiments outlined in Table 4.7. In Table 4.8, the average ensemble size and accuracy is increasing by choosing a larger value of 'β'.

Table 4.8 Effect of 'β' on Accuracies % & Ensemble Size, Period = 50 and θ = 0.01(Fixed).

| Streams | β = 0.1 | | β = 0.5 | | β = 0.75 | |
|---|---|---|---|---|---|---|
| | ACC% | Ensemble Size | ACC% | Ensemble Size | ACC% | Ensemble Size |
| SEA (S) | 87.6 (3) | 13.5 | 88.1 (1) | 23.6 | 87.9 (2) | 23.7 |
| STAGGER (S) | 60.8 (1) | 4.0 | 60.8 (2) | 4.0 | 60.7 (3) | 4.0 |
| RTree R | 75.2 (3) | 8.5 | 84.4 (2) | 13.7 | 88.8 (1) | 20.5 |
| LED (S) | 72.5 (3) | 9.4 | 73.4 (1) | 23.8 | 73.4 (2) | 24.5 |
| Wave (S) | 80.3 (3) | 13.3 | 82.2 (2) | 17.4 | 83.5 (1) | 24.5 |
| Hyperplane | 88.2 (1) | 9.4 | 87.5 (3) | 19.8 | 87.6 (2) | 24.4 |
| SEA (G and S) | 87.4 (2) | 12.3 | 87.1 (3) | 17.2 | 88.3 (1) | 23.2 |
| RRBF(G) | 92.6 (2) | 8.6 | 92.6 (1) | 14.8 | 92.5 (3) | 20.3 |
| Electricity | 85.8 (3) | 7.07 | 88.4 (2) | 10.7 | 89.7 (1) | 15.8 |
| Spam | 89.2 (3) | 6.36 | 89.7 (2) | 8.6 | 90.1 (1) | 9.5 |
| Sensor | 93.0 (1) | 6.74 | 92.5 (2) | 10.3 | 91.7 (3) | 13.8 |
| Forest | 85.7 (3) | 7.17 | 89.7 (2) | 11.5 | 91.3 (1) | 16.5 |
| **Avg. (Ranks)** | **83.2(2.2)** | **8.8** | **84.7(2.0)** | **14.6** | **85.5(1.7)** | **18.3** |

Table 4.9 Effect of 'theta' on Accuracies % & Ensemble Size, Period = 50, β = 0.5 (Fixed).

| Streams | θ = 0.01 | | θ = 0.05 | | θ = 0.1 | |
|---|---|---|---|---|---|---|
| | ACC% | CPU time | ACC% | CPU time | ACC% | CPU time |
| SEA (S) | 88.1 (2) | 102.5 | 88.1 (1) | 103.6 | 88.0 (3) | 95.1 |
| STAGGER (S) | 60.8 (2) | 0.04 | 60.8 (2) | 1.0 | 60.8 (2) | 1.0 |
| RTree R | 84.4 (1) | 238.5 | 81.0 (2) | 195.9 | 79.9 (3) | 155.0 |
| LED (S) | 73.4 (1) | 664.5 | 73.3 (2) | 690.0 | 73.3 (3) | 589.3 |
| Wave (S) | 82.2 (1) | 1195.6 | 81.9 (2) | 766.1 | 81.5 (3) | 730.1 |
| Hyperplane | 87.5 (3) | 508.6 | 87.8 (2) | 429.3 | 88.2 (1) | 343.1 |
| SEA (G and S) | 87.1 (3) | 127.1 | 87.7 (1) | 106.4 | 87.6 (2) | 99.7 |
| RRBF(G) | 92.6 (2) | 203.4 | 92.6 (1) | 127.2 | 92.5 (3) | 121.5 |
| Electricity | 88.4 (1) | 148.4 | 88.3 (2) | 153.5 | 87.9 (3) | 127.6 |
| Spam | 89.7 (3) | 148.9 | 90.0 (1) | 155.6 | 89.9 (2) | 128.1 |
| Sensor | 92.5 (2) | 106.5 | 92.5 (3) | 965.2 | 92.9 (1) | 788.1 |
| Forest | 89.7 (1) | 668.2 | 89.3 (2) | 607.0 | 88.4 (3) | 492.6 |
| **Avg. (Ranks)** | **84.7(1.8)** | **442.5** | **84.4(1.8)** | **358.4** | **85.5(2.3)** | **305.9** |

Parameter 'θ' was analysed on predictive accuracies and CPU-time. Beta = 0.5 was fixed due to the moderate average ensemble size in the experiment. The results in Table 4.9 show that the CPU-time slightly decreased by increasing the value of θ. By increasing 'θ' the average ranks increased from 1.8 to 2.3. The lower ranks show a higher predictive performance.

## 4.10 Analysis of the Effects of different Ensemble Sizes

Due to the seed learners that always remain in the dynamic list, HDWM maintained a larger ensemble size (Average 27.6) and 11.29 in real-world datasets. Table 4.10 and Table 4.11 represents average ensemble sizes achieved in HDWM and DWM. The plots for the ensemble size are shown in Figure 4.8 and Figure 4.9.

Table 4.10 Average ensemble size in Artificial MOA streams.

| **Streams** | **HDWM** | **DWM-NB** | **DWM-HT** |
|---|---|---|---|
| SEA (S) | 61.39 (3) | 35.72 (2) | 25.38 (1) |
| STAGGER (S) | 12.18 (3) | 7.73 (2) | 7.07 (1) |
| RTree R | 13.19 (1) | 28.37 (3) | 16.69 (2) |
| LED (S) | 33.94 (1) | 37.1 (2.5) | 37.1 (2.5) |
| Wave (S) | 18.02 (1) | 37.83 (3) | 29.09 (2) |
| Hyperplane (G) | 22.91 (3) | 14.28 (2) | 13.52 (1) |
| SEA (G and S) | 43.56 (3) | 37.89 (2) | 25.6 (1) |
| RBF(G) | 16.26 (1) | 8.76 (2) | 10.48 (1) |
| **Average** | **27.6** | **25.9** | **20.6** |

Table 4.11 Average ensemble size (%) real-world datasets.

| **Datasets** | **HDWM** | **DWM-NB** | **DWM-HT** |
|---|---|---|---|
| Electricity | 12.26 (3) | 11.33 (1) | 11.88 (2) |
| Spam | 11.45 (3) | 7.79 (1) | 8.12 (2) |
| Sensor | 8.04 (1) | 8.58 (2) | 9.06 (3) |
| Forest Cover | 13.41 (2) | 15.26 (3) | 10.04 (1) |
| **Average** | **11.29** | **10.74** | **9.78** |

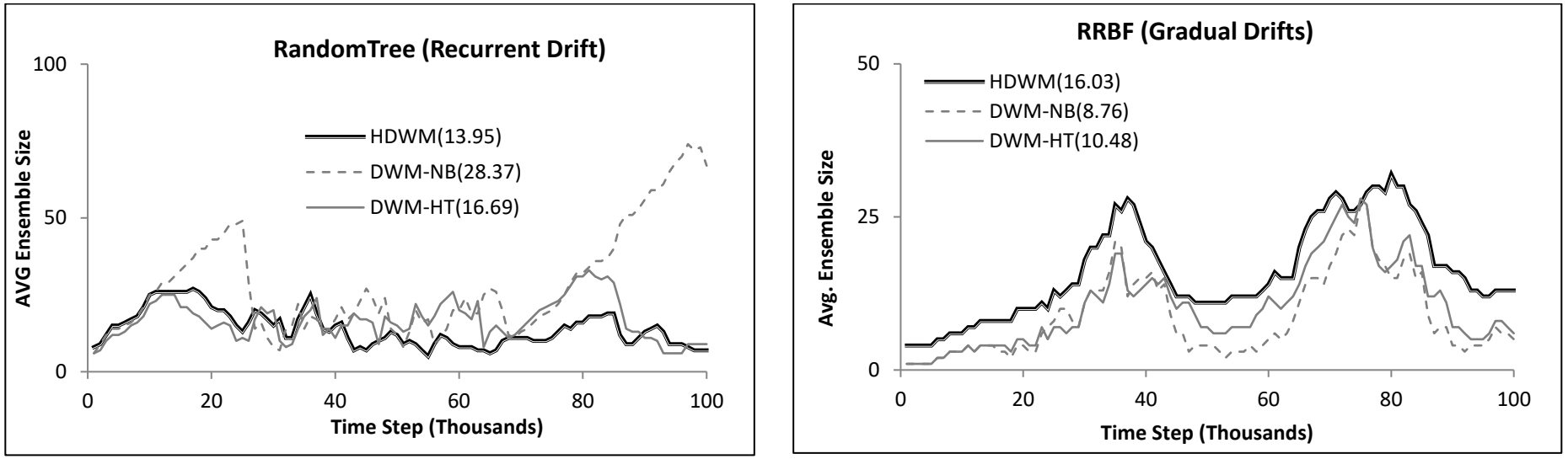


**Figure 4.8** Average Ensemble Size RandomTree (left) and RRBF (right) in Artificial Data Streams.

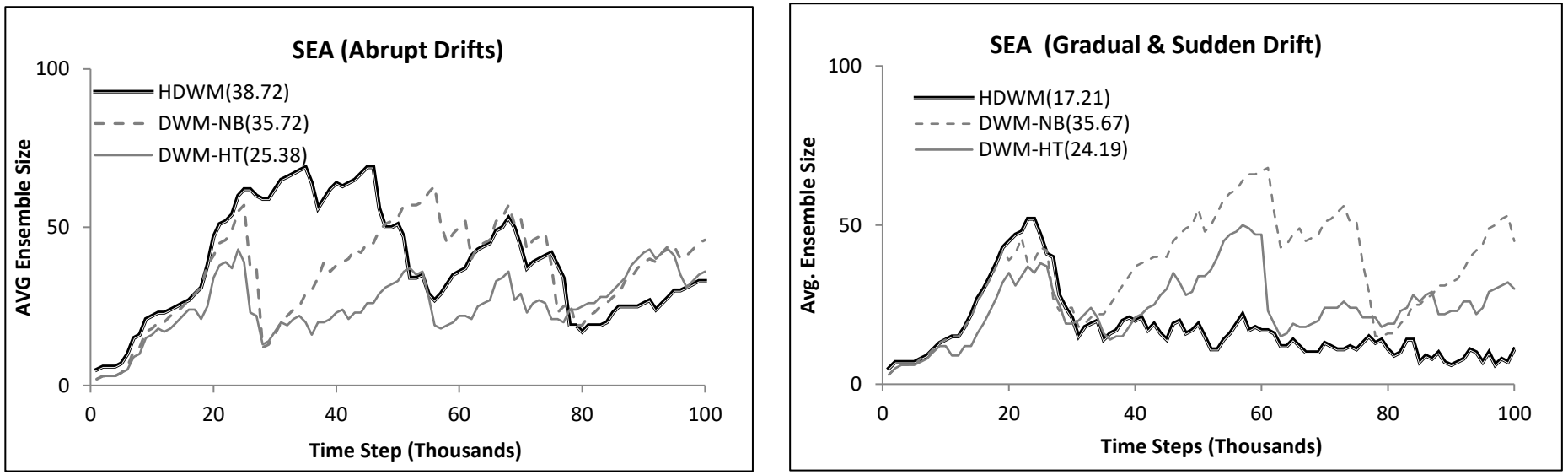


**Figure 4.9** Average Ensemble Size SEA Abrupt (left) and SEA Mixed (right) in Artificial Data Streams.

## 4.11 Comparison of Resource Consumption

To analyse the benefits in terms of resource usage, the algorithms HDWM, DWM and WMA have been compared. Runtime evaluations are measured in CPU seconds by setting max size of ensemble ($B_{max}$) to 25, 50,100 for all the datasets. It is expected that HDWM requires more processing time compared with WMA and DWM due to the seed learners that always reside in the ensemble. As shown in Figure 4.10, the total CPU time is increasing by setting a larger value of $B_{max}$, however, the average predictive accuracies are not significantly affected.

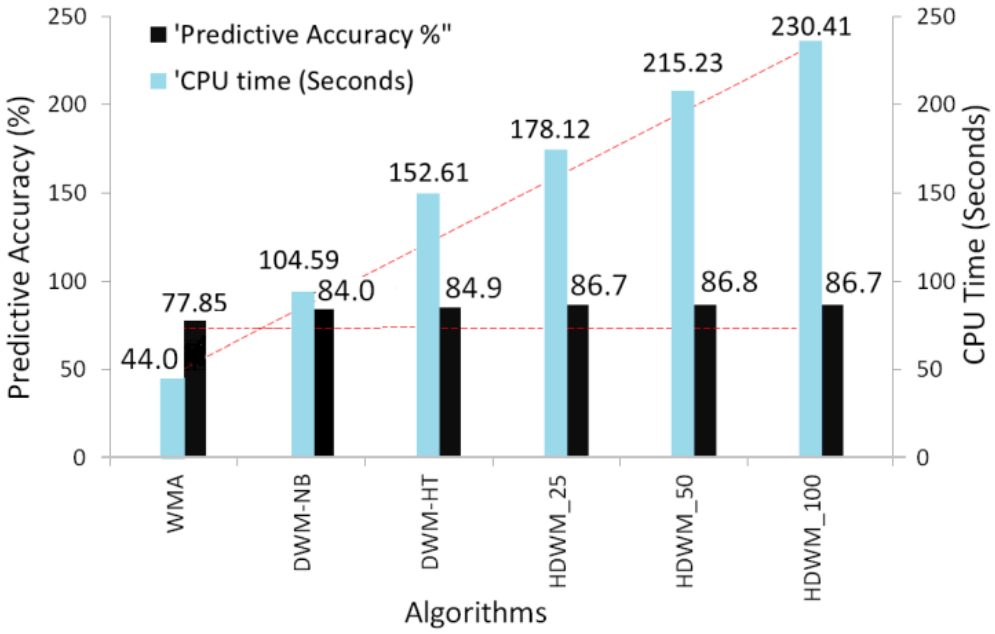


Figure 4.10 CPU time (Seconds) and Predictive Accuracies of HDWM, DWM and WMA.

## 4.12 Complexity Analysis

The time complexity of online ensemble classifiers heavily depends on the choice of base classifiers. HDWM applies NB [38], HT [39] and KNN [40] base classifiers. Based on the worst time complexity of these base classifiers, the total time complexity of HDWM for 'τ' number of labelled training examples is O (τ · d) + O (d · v · c). The space complexity for storing the likelihood of each feature with respect to classes is O (l · d · v · c). Where 'd' is dimensionality of the attributes, 'v' values per attribute, 'c' is number of classes and 'l' is the current number of leaves.

## 4.13 Effect of Prediction Method

The objective of this experiment id to investigate the predictive performance of HDWM algorithm due to change of prediction methods i.e. 1) prediction taken from classifiers, and 2) Prediction taken from Clusters. The data stream RandomRBFGeneratorEvents [46] that is the stream designed for clustering in MOA is shown in Figure 4.11, which is based on the

random Radial Basis Function that adds drift to samples in a stream. A total 5 clusters are generated at 5% noise level. The SEADriftStream and RandomRBFGeneratorEvents are evaluated using HDWM classifier and MOA cluster. Max number of clusters 5 is applied, which is the default settings for MOA clustering.

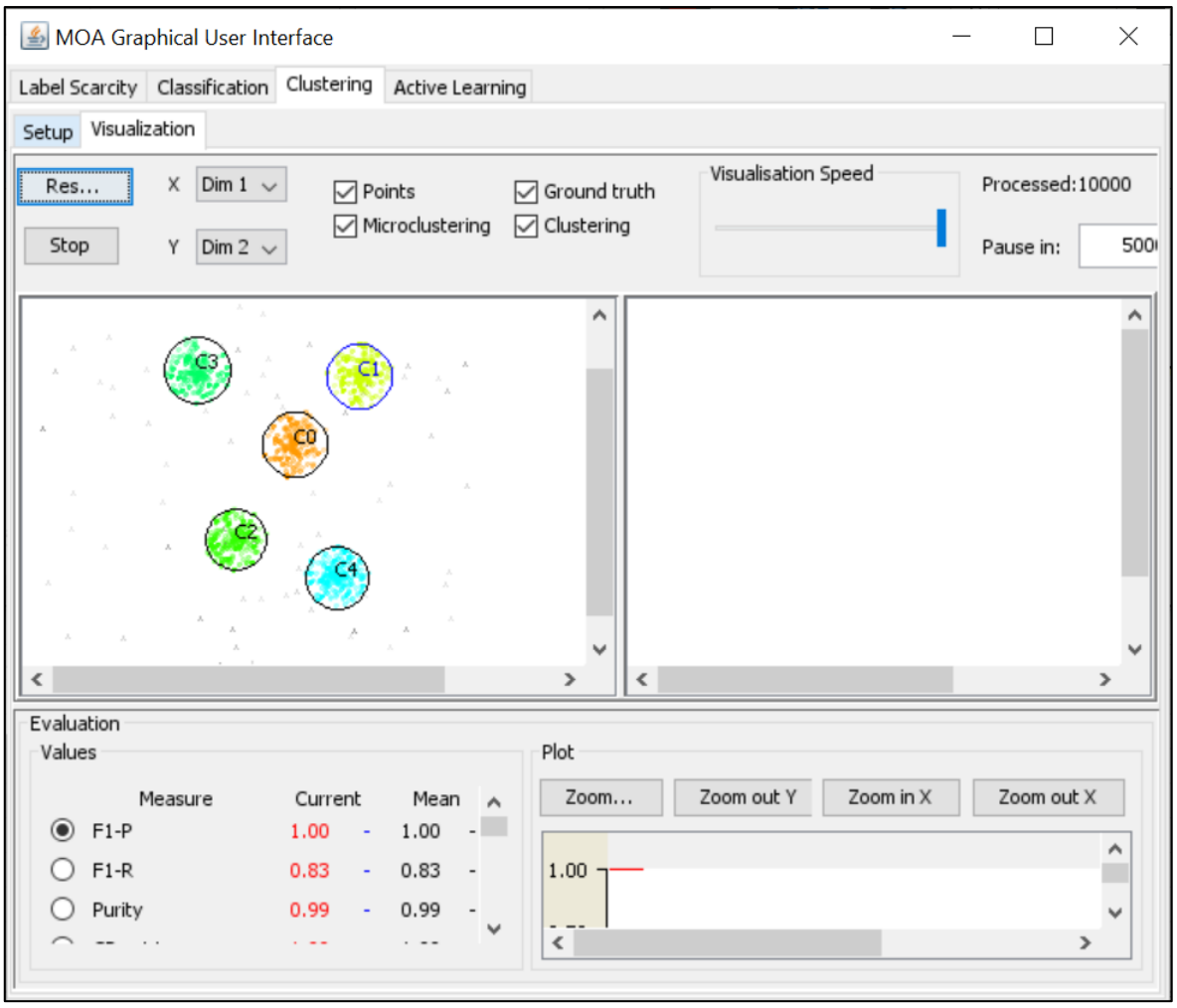


Figure 4.11 RandomRBFGeneratorEvent Stream in MOA.

### 4.13.1 Evaluation Results

The evaluation results are showing in Figure 4.12 and Figure 4.13. The results show that the prediction accuracy is higher (89.5%) in RandomRBFGeneratorEvent when a clustering algorithm is applied. However, prediction accuracy is low (56.7%) when classifier (HDWM) is applied. The evaluation results of SEADriftStream in Figure 4.14 and Figure 4.15 shows that the prediction accuracy is lower (68.75 %) when a clustering algorithm is applied. However, prediction accuracy is higher (81.89%) when classifier (HDWM) is applied.

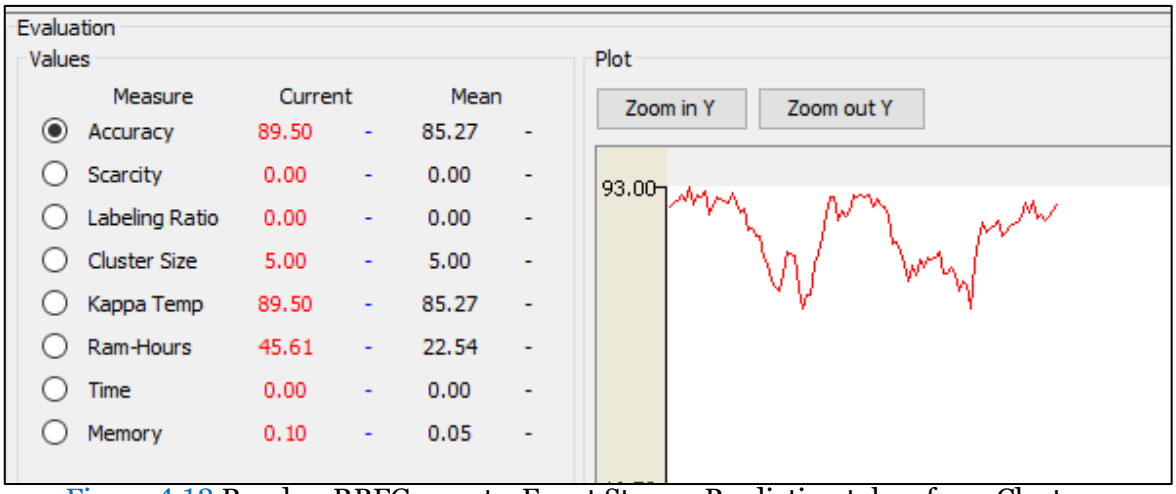


Figure 4.12 RandomRBFGeneratorEvent Stream Prediction taken from Clusters.

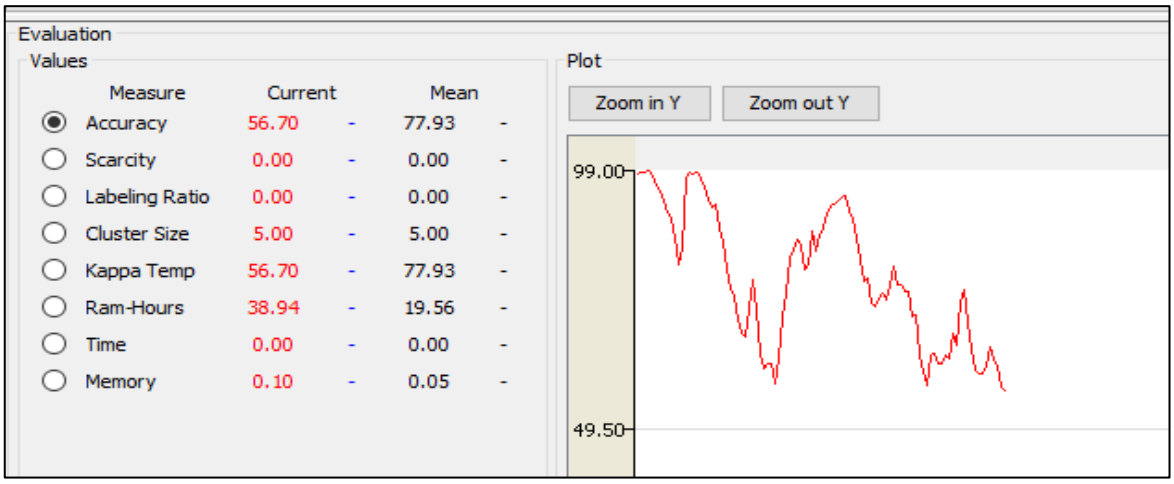


Figure 4.13 RandomRBFGeneratorEvent Stream Prediction taken from Classifier.

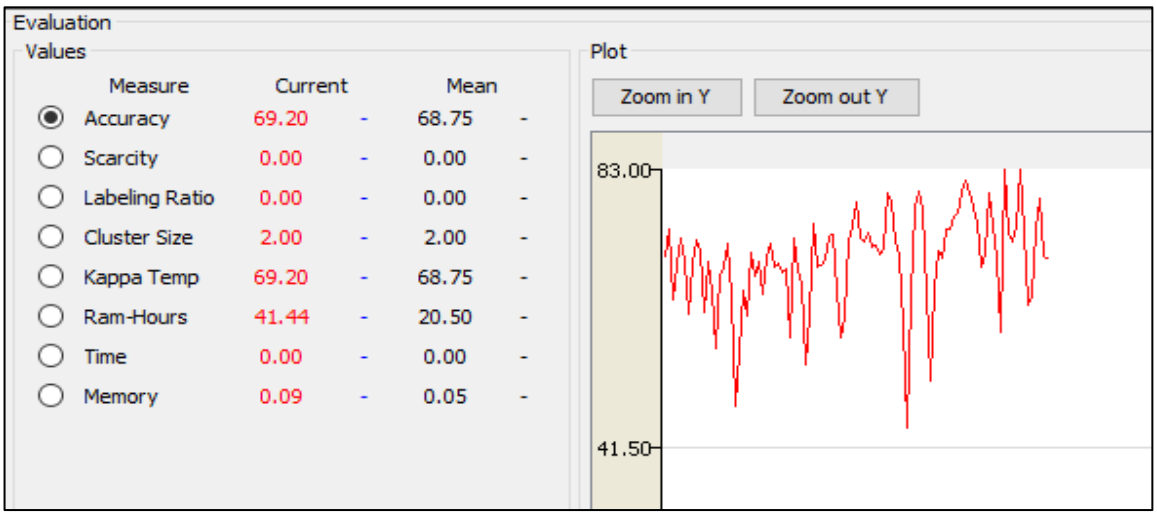


Figure 4.14 SEADriftStream Prediction taken from Clusters.

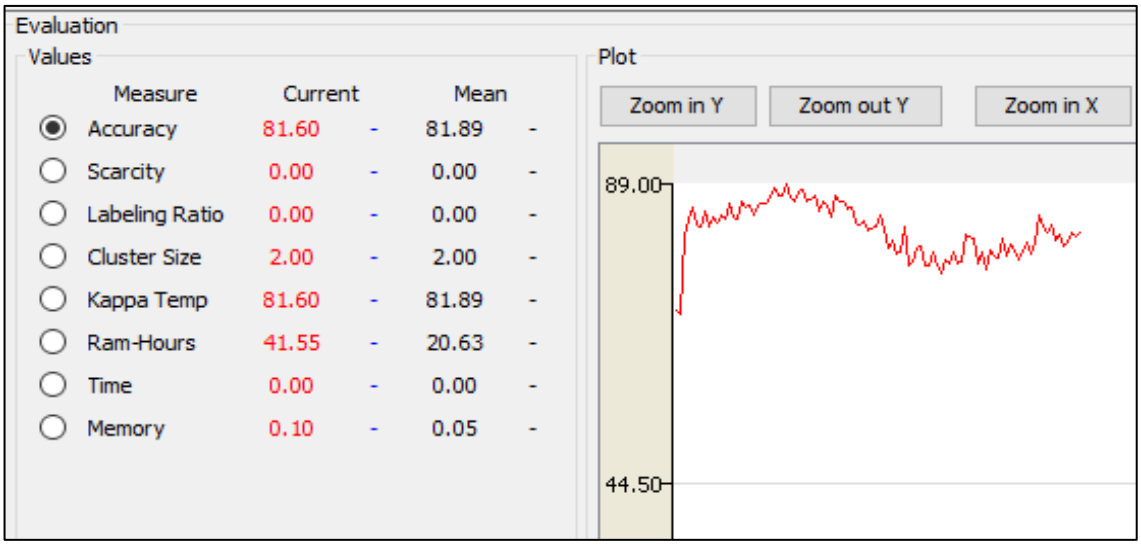


Figure 4.15 SEADriftStream Prediction taken from Classifiers.

### 4.13.2 Significant Findings

It is evident from the experiment that a specific algorithm method works well on particular stream. It is extremely difficult for a learning algorithm to decide on which prediction method should be used. A prediction Method is required that maintains prediction accuracy on both these streams.

## 4.14 Discussion

The predictive performances of various algorithms are scrutinized, and their reactions to different types of drifts are graphically illustrated. Notably, the ensemble size, a critical factor in balancing performance in dynamic base classifiers, is examined, considering the trade-off between processing time and predictive accuracy. The impact of ensemble heterogeneity on predictive performance, specifically investigated whether HDWM's higher accuracy is attributed to its heterogeneity or active drift handling. These results highlighted the crucial role of heterogeneity, emphasizing the model switching mechanism's role in maintaining accuracy independently of manual base learner selection. The integration of DWM with DDM enabled the explicit drift detection to identify changes in the environment, allowing for adaptive measures like removing poor-performing classifiers or resetting weights when a drift is detected. The sensitivity analysis of hyperparameters explored the challenge of setting parameters in real-world problems, acknowledging that optimal values may vary over time. The study assessed the impact different parameters on prediction accuracy, ensemble size, and drift detections. It was analysed that larger 'period' boosted the prediction accuracy and reduces drift detections.

The experiments used artificial data streams, where the position of concept drift is known. Real-world datasets are also employed, lacking known drift locations, adding unpredictability to the analysis. It has been observed that due to non-stationary data streams from NSEs,

traditional cross-validation techniques are unsuitable. Instead, the prequential method was used, ensuring each example is tested before being trained on. For statistical analysis of algorithm differences, non-parametric tests and post hoc tests identified specific algorithm pairs with significant differences. This approach guarantees a rigorous evaluation of algorithms in terms of both time efficiency and predictive performance.

## 4.15 Summary

This chapter addressed the gaps that were identified in Chapter 3, i.e. varying predictive performances of online classifier ensembles due to the diversity of base classifiers. It was found extremely difficult to choose the best-performing classifiers for a specific problem. More specifically, answered the (RQ2), i.e. obstacles in determining the type of machine learning algorithm that would be best to use in EVL conditions and overcoming the problems of existing dynamic ensembles that may undergo loss of diversity due to the exclusion of base learners.

As a response to this question, a novel heterogeneous ensemble approach, HDWM was proposed, which is capable of intelligently switching between different types of base models in an ensemble. The new approach revealed the ability to reduce human dependency on redefining the best type of predictive models for a particular problem. The proposed 'Seed' mechanism makes use of different types of base classifiers in its ensemble to maintain its diversity. These seeds are never deleted and retain the previously learned concepts, which helps to deal with the recurring concept drifts. It also implements both an active and passive approach for handling concept drifts, the results showed that the new approach efficiently handled both gradual and sudden concept drifts.

The results showed that WMA maintained the diversity but was unable to deal with the concept drifts due to its inability to create or delete base learners. DWM on the other hand is a dynamic ensemble but lacks diversity as it does not benefit from multiple types of base learners. HDWM overcomes these problems and deals with concept drift through the addition and removal of base learners. It achieves that by ensuring that seed learners of any type can repopulate the ensemble whenever they become beneficial.

The HDWM was evaluated against WMA (due to its heterogeneity) and DWM (due to its dynamicity) as well as sensitivity to its parameters. Eight synthetic data streams were generated with concept drifts, and four real-world datasets 'Sensors', 'Spam Email', 'Electricity' and 'Forest Cover' were used in the experiments. Apart from prequential evaluation, the periodic holdout approach and kappa statistics were applied as the evaluation matrices. Non-parametric tests were carried out to determine the statistically significant differences between the algorithms and if the null hypothesis was rejected, post-hoc tests were used to identify which pairs of algorithms differ from each other.

The results showed that HDWM performed significantly better than WMA and DWM, also, when recurring concept drifts were present. The predictive performance of HDWM showed an improvement over DWM in both drift and real-world streams. It can be concluded that HDWM is independent of deciding which type of base classifier should be used. In another experiment, HDWM was compared with CluStream clustering on SEADriftStream and RandomRBFGeneratorEvents. The interesting results showed that the prediction accuracy of HDWM was found lower than the CluStream when applied to RandomRBF GeneratorEvents. On the other hand, HDWM outperformed CluStream on SEADriftStream.

It was evident from the experiment that a specific algorithm works well on data streams. Therefore, it is extremely difficult for a learning algorithm to decide on which prediction method should be applied. Furthermore, HDWM has been designed as a supervised learning algorithm, which means it assumes the availability of labelled data. It can be concluded that under the EVL conditions, it is more challenging to determine which type of machine learning algorithm would be best to use due to the small amount of initial labelled data. The next Chapter 5 focuses on developing an online SSL for ILNSE which is capable of learning from both labelled and unlabelled data.

# Chapter 5 Predictor for Streaming Data with Scarce Labels

## 5.1 Introduction

The aim of this chapter is to investigate and answer the (RQ1) which was raised in Chapter 1, i.e. the scarcity of true class labels and its impact on prediction accuracy in non-stationary environments, especially when learning algorithms lack direct access to true class labels immediately following concept drifts. The new developed HDWM is a supervised learning classifier which is highly dependent on true class labels. The results showed that the HDWM automatically identifies the types of predictive models best suited to the situation encountered after different types of drifts, such as gradual, sudden and recurring.

Extreme Verification Latency (EVL) and Non-Stationary Environments (NSEs) have recently gained significant attention in the data stream mining community due to an enormous growth streaming data which is evolving and unlabelled. Therefore, extracting worthwhile knowledge is challenging from real-time data streams. The term, Initially Labelled Non-Stationary Environment (ILNSE) is used in the literature that simultaneously deals with EVL and NSE. Learning under Initially Labelled Non-Stationary Environment (ILNSE) is challenging task because the learning algorithms have no access to the true class labels directly after the concept drift and manual labelling of these data streams is not practical due to time consumption and need for domain expertise. The existing approaches require more than one technique such as CGC [31], self-learning [41] [42] micro-clustering [43]. However, from the literature it is not clear on what conditions one approach is better than the other and what causes other approaches to fail.

However, learning becomes more challenging when a small set of initially labelled data is followed by data which consists of only unlabelled data. To deal with label scarcity problem usually more than one technique is applied, for instance the authors of some approaches [22] applied clustering and ensemble learning to deal with label scarcity and drift handling. A series of online surveys incorporate the latest developments in the field of Online Semi-Supervised Learning (OSSL) methods which is closely related to label scarcity issue in online machine learning. However, the existing approaches focus on offline learning for static data and make two basic assumptions, 1) the availability of large training dataset; and (2) training and test data is stationary.

A data stream environment has different requirements from the traditional batch learning setting [46]. Further requirements for the OSSL scenario have been identified and can be derived as follows.

- **Requirement 1** Process a labelled example at a time and inspect it only once (at most).

- **Requirement 2** Use a limited amount of time and memory.
- **Requirement 3** Process unlabelled example in small batches and predict pseudo-labels.
- **Requirement 4** Available to predict at any time.

Figure 5.1 shows the typical use of an online SSL data stream classification algorithm, and how the requirements fit in a repeating cycle.

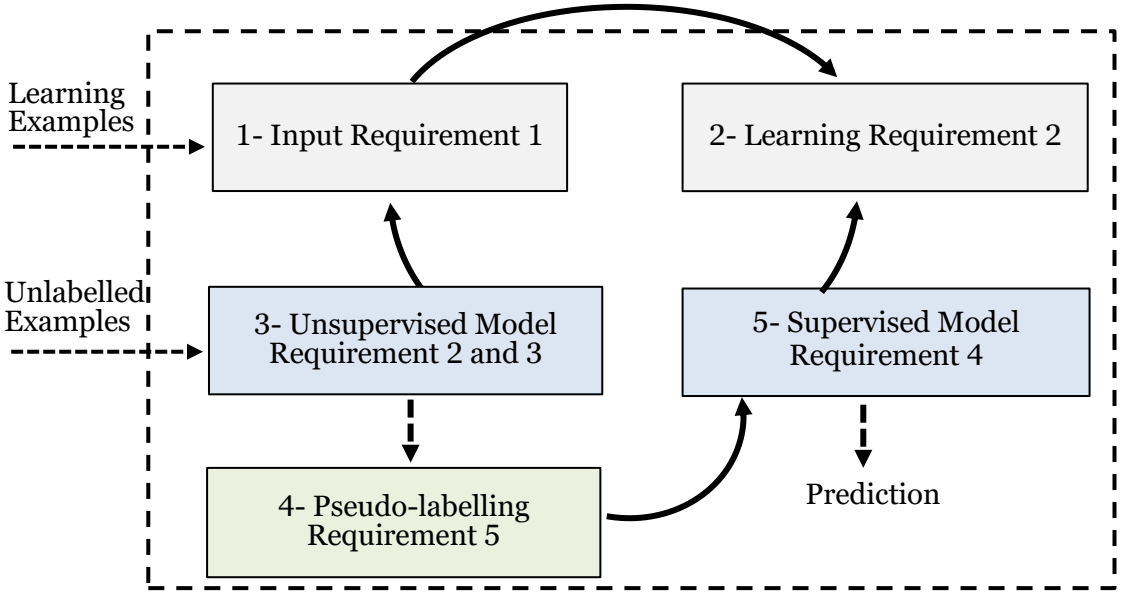


Figure 5.1 The Data Stream Online Semi-Supervised Learning Cycle.

1. The supervised learning algorithm is passed the next available labelled example from the stream (Requirement 1).
2. The algorithm processes the labelled example, updates its data structures. It does so without exceeding the memory and time bounds set on it (Req. 2).
3. The unsupervised learning algorithm is passed the next available unseen example from the stream (Req. 2 and Req. 3).
4. The unsupervised learning algorithm is ready to accept the next example. On request, it can predict the pseudo-labels (class of unseen examples) (Req. 4).
5. The supervised learning algorithm is ready to accept the pseudo-labelled example and update the model. On request, it can predict the class of unseen examples (Req. 4)

## 5.2 Overview of PSDSL

This research directly responds to ILNSE challenge in proposing a novel algorithm "Predictor for Streaming Data with Scarce Labels" (PSDSL), which is capable of intelligently selects the best pseudo-labelling strategy based on the given problem domain. PSDSL is implemented in MOA [46] which is an open-source framework for data stream mining. The PSDSL performs the following tasks on the initial labelled data.

1. Decide on the best classifier from a pool of Heterogeneous classifiers.

2. Decide on the pseudo-labelling strategy, i.e. Cluster guided or self-learning using classifiers.
3. Build offline micro-clusters and apply them online on-demand only in the case of drift detection.
4. Perform hyperparameter tuning to determine the best value of 'k'.

Figure 5.2 depicts the visual abstract of PSDSL algorithm. HDWM classifier is trained on a small amount of labelled data from the data streams, at this stage the best learning parameters are identified during the hyperparameter tuning phase. When unlabelled data arrives, clusters are generated to predicts the pseudo labels which retrain the HDWM classifier and finally predicts the class labels in the unseen environment.

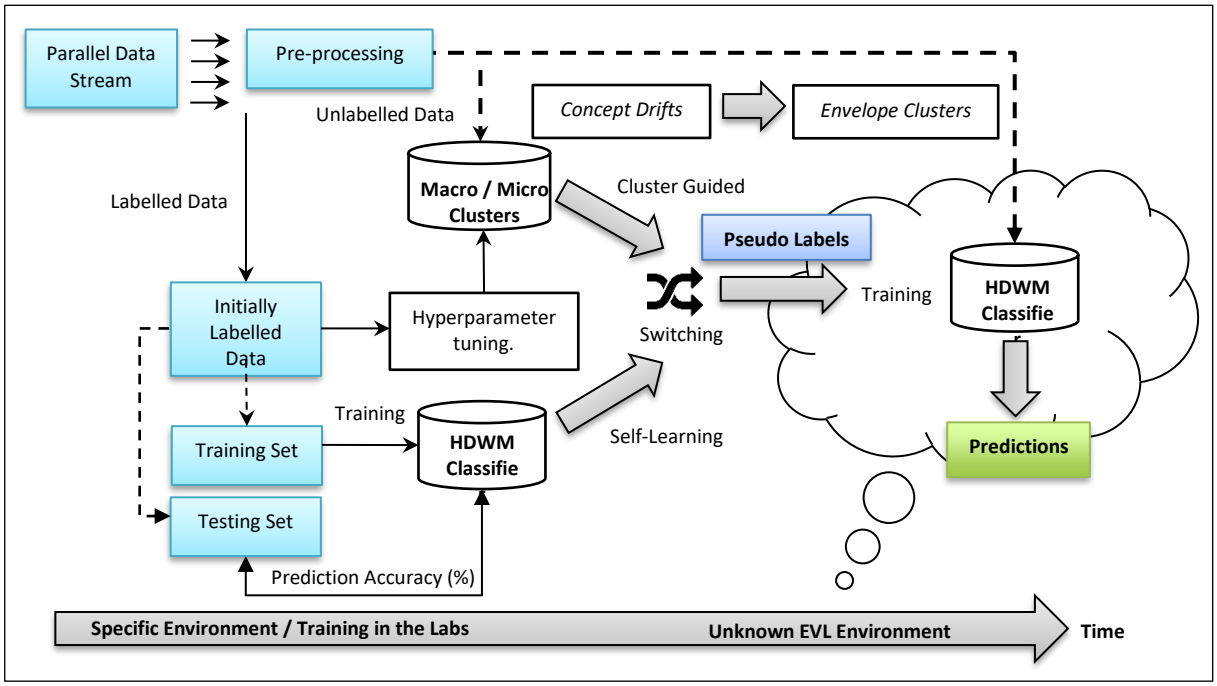


Figure 5.2 Visual abstract for the predictor for streaming data with scarce labels (PSDSL), HDWM is trained on small amount of labelled data, performs hyperparameter tuning and pseudo labels are then predicted to re-train HDWM for final predictions.

The data streams are continuous; therefore, the data has been divided into several batches B= $\{B_1, B_2 \ldots B_n\}$. The first batch is completely labelled, followed by labelled instances $X_L = (x_1 \ldots x_L)$ for which class labels $Y_L = (y_1 \ldots y_L) \subset Y$ are available and for unlabelled instances $X_U = (x_{L+1} \ldots x_U)$ the class labels are unavailable. The unlabelled data stream generates and updates the clustering on real-time data streams. To handle the Virtual drifts that occur due to changes in the distribution of input data i.e. $P_t(x) \neq P_{t+1}(x)$, PSDSL establishes a mapping between current and previous clusters ($C_t \longrightarrow C_{t+1}$) by assigning the current centroid the label which is the same label of the 'k' nearest past centroid.

## 5.3 Pseudo-labelling process of PSDSL algorithm

As shown in Figure 5.3, In step 1, a set of heterogeneous classifiers are trained on a small number of labelled data examples, and ground truth clusters are formed. This information of ground truth clusters is passed to the switching of pseudo-labelling states (step 3) and hyperparameter tuning (step 2) which are explained in Section 5.4 and 5.5 respectively. In step 4 overlapping of the clusters is determined, if confidence levels of cluster labels fall below a user-provided threshold, envelope-clusters are formed to resolve the conflict in labelling. The envelope-clusters are explained in Section 5.6. Finally, in step 5, the pseudo-labels are fed back to update the classifiers for predictions.

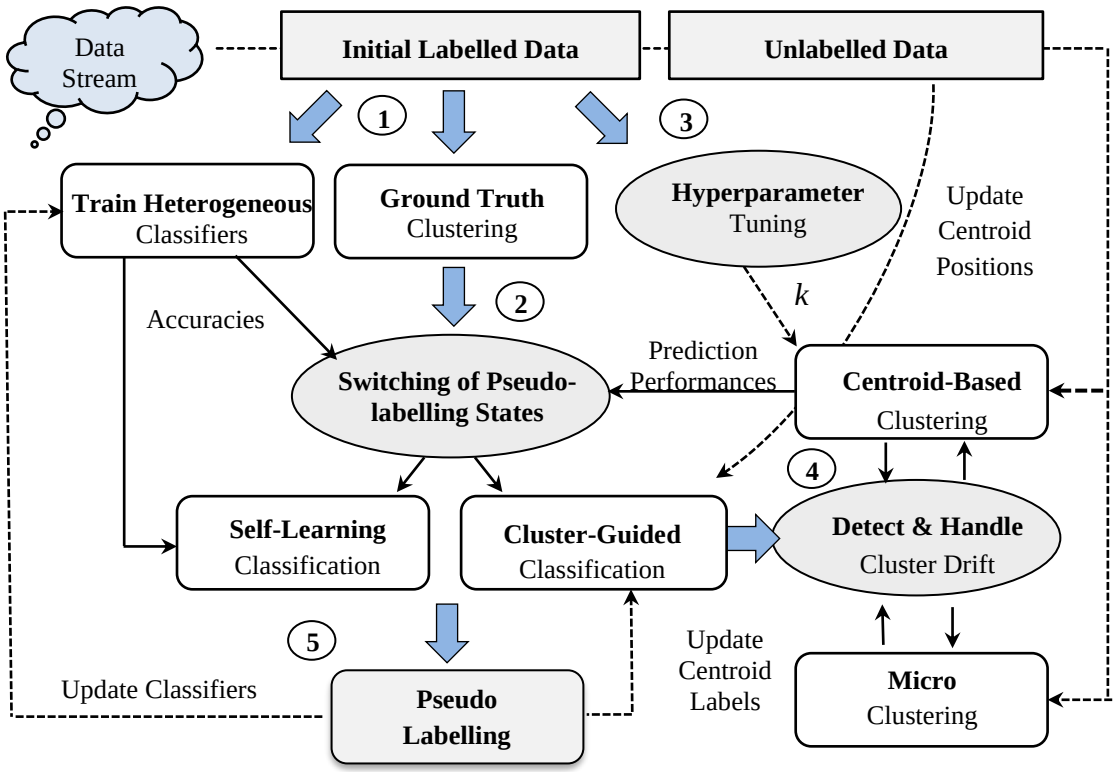


Figure 5.3 Illustrative Pseudo-labelling process of PSDSL algorithm, key steps include training of heterogeneous classifiers and generating clusters by using limited amount of labelled data.

## 5.4 Switching of Pseudo-Labelling States

PSDSL can switch between the three learning states for pseudo-labelling, 1) Cluster guided 2) Self-learning and 3) micro-clustering. The switching mechanism of PSDSL is illustrated in Figure 5.4. In a situation where pseudo-labelling is not improving the predictive performance on initially labelled data, PSDSL switches off the pseudo-labelling state. For this, Ensemble 'GT (Ground Truth)' is trained on the complete set of initial labelled data, while Ensemble 'PL (Pseudo-Labelling)' is trained only on 80 % of the training data. Ensemble 'PL' predicts the pseudo-labels for the remaining 20% and trains itself on these pseudo-labels. If the prediction

accuracy of Ensemble 'PL' improves over 'GT', the self-learning state is enabled, otherwise it is suspended. The cluster guided state is enabled when the mean values of F1-P and F1-R [124] is higher than a user provided threshold 'ρ'.

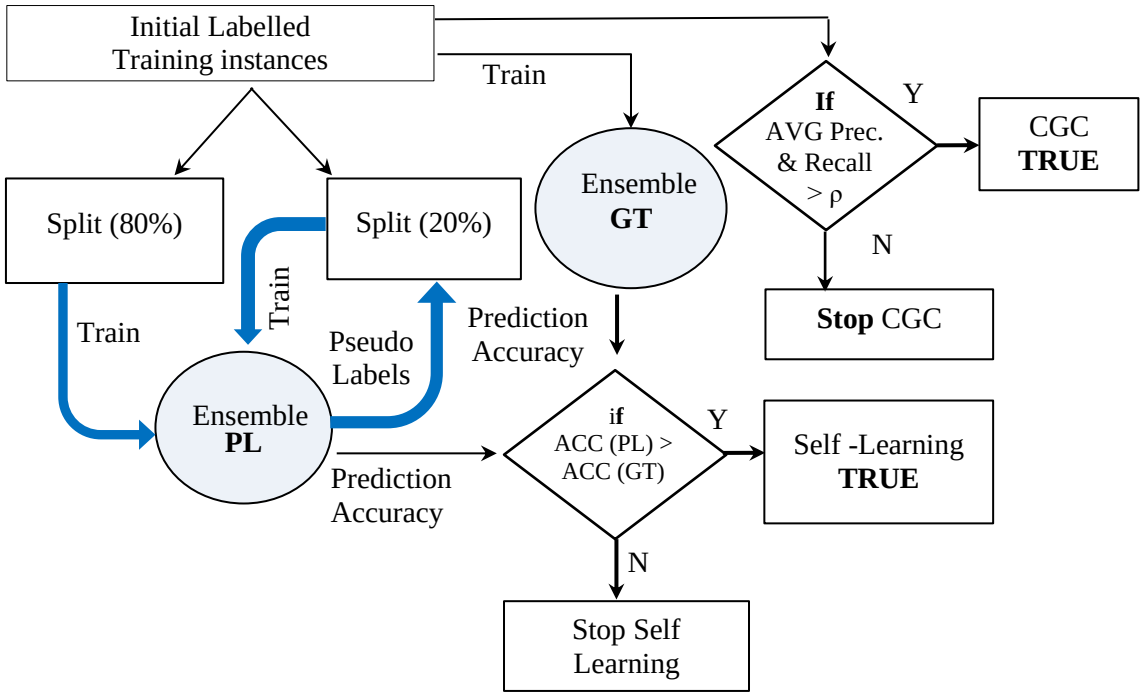


Figure 5.4 Switching of Pseudo-Labelling States between self-learning and CGC based on the prediction accuracies from ground truth and pseudo-labelling ensemble classifiers and comparing it with the average precision and recall of the clusters.

The names F1-P and F1-R of evaluation metrics are given as the same names mentioned in MOA [46]. The F1-measure is the harmonic mean of precision and recall. F1-P calculates the total F1-score for each found cluster instead of for all ground truth clusters. While the F1-R is calculated by maximising F1 for each ground truth class. Figure 5.5 showing a GUI for Semi-Supervised Learner (SSLearner) developed in MOA to experiment with switching strategies.

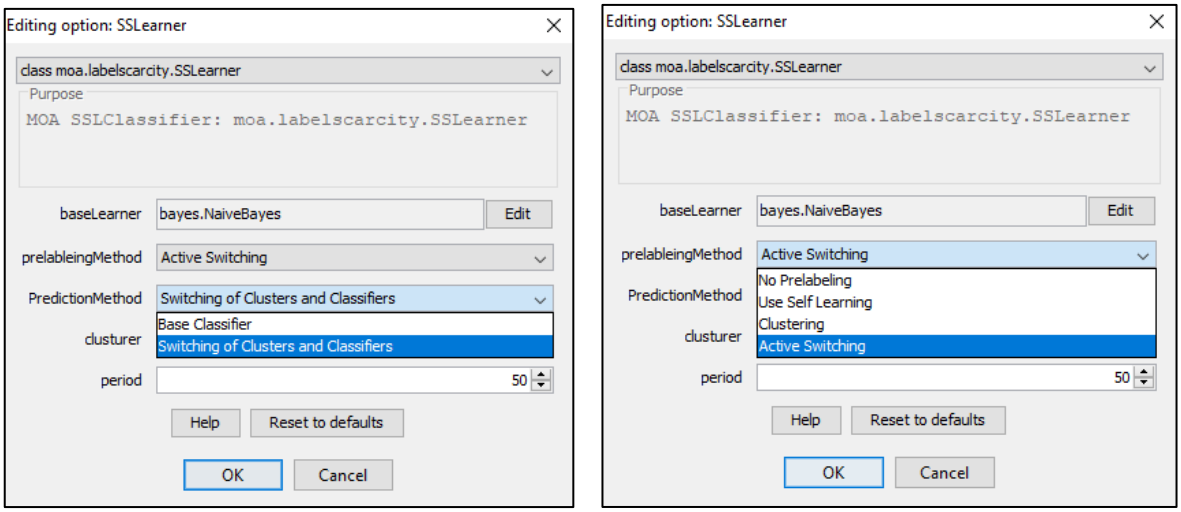


Figure 5.5 (left) Prediction methods (right) pseudo-labelling methods.

The supported pseudo-labelling method are "No Pre-labelling", "Self-Learning", "Clustering", and “Active Switching". The available prediction methods are “Base Classifier” and "Switching of Clusters and Classifiers". Figure 5.6 shows the graphical interface which enables the visualisation of different states taken by PSDSL for pseudo-labelling.

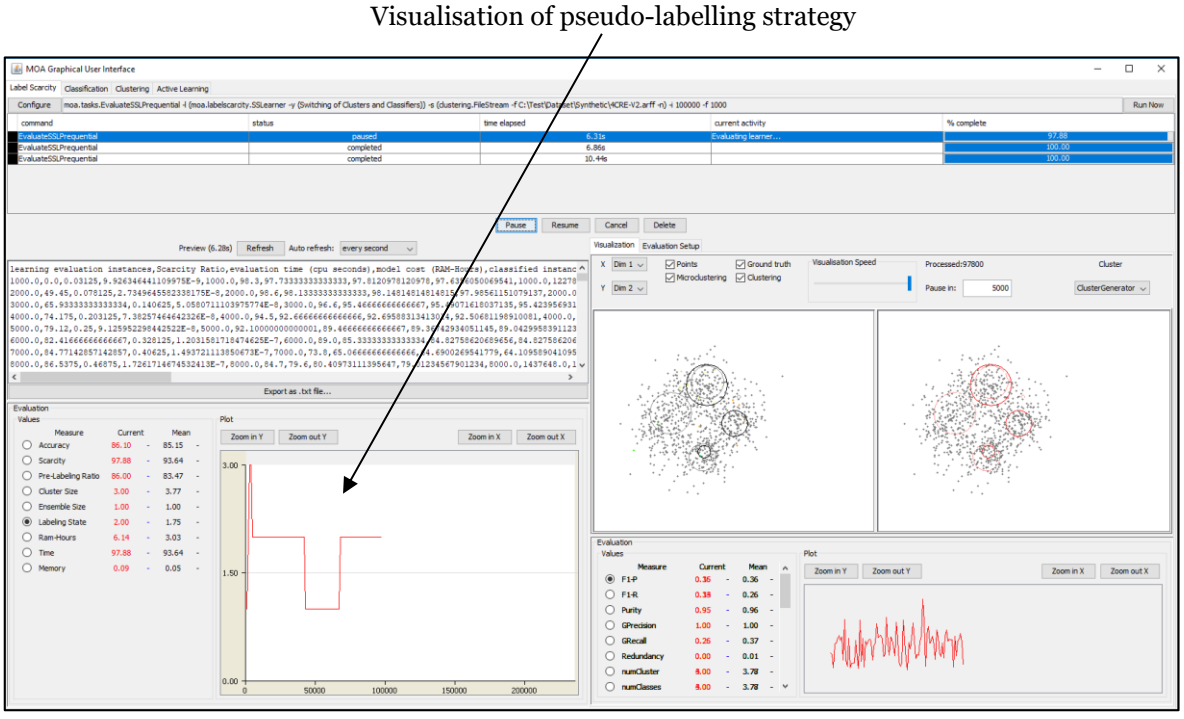


Figure 5.6 Switching of pseudo-labelling strategy, 0=No pseudo-labelling, 1=self-learning, 2 = CGC.

## 5.5 Cluster Guided Classification in PSDSL

To utilise the information associated with unlabelled instances, CGC make use of clusters for predicting the pseudo-labels, which further update the classification models. In case of gradual movement of these clusters, the concept drifts are detected. For this purpose, PSDSL relies on CGC for tracking the changing concepts. Figure 5.7 shows concept $v_1$ (t) and $v_2$ (t+1) as a function of time 't' and 'x' are the input attributes, and 'c' are the classes. The steps used in this approach are given below.

1) In concept $v_1$, at time (t), the labelled instances generate $\{C_1 \ldots C_n\}$ clusters representing $\{c_1 \ldots c_n\}$ classes in the initial labelled data.
2) When unlabelled data arrives at time (t+1) and data distribution changes, the concept $v_1$ changes to $v_2$ and the clusters receive labels from the nearest clusters.
3) More new data arrives at time (t+2) for which class labels are missing (shown in white circles) and pseudo-labels are required.
4) At time (t+3), the unlabelled instances 'x' receive pseudo-labels from the nearest clusters 'C' using the Euclidean distances between 'C' and 'x'.

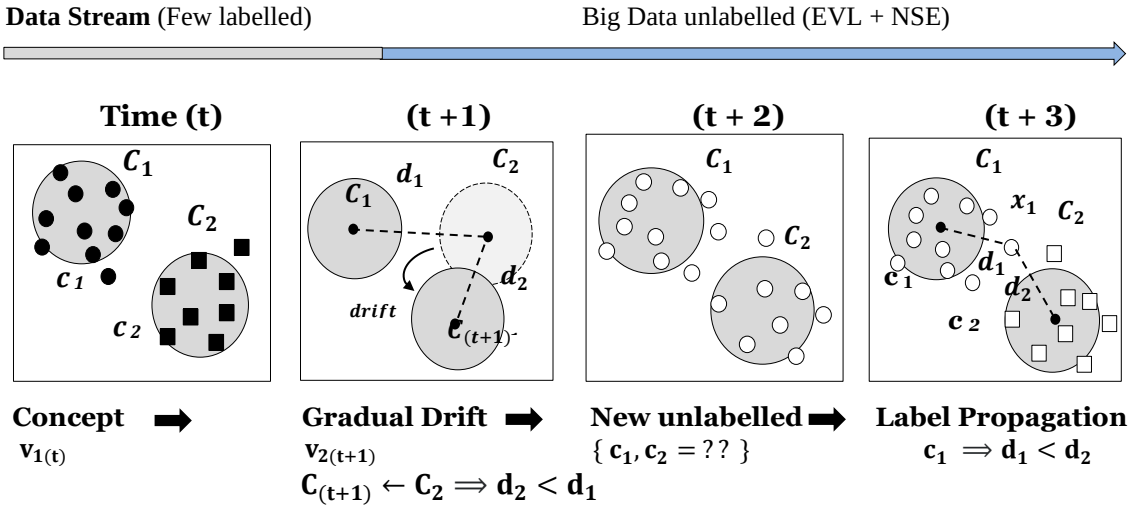


Figure 5.7 Cluster Guided Classification in PSDSL showing representation of different concepts at time 't' due to gradual drifts and the process of label propagation from nearest clusters.

## 5.6 Envelope-Clustering New Approach

Micro-clustering state applies on-demand when overlaps between clusters are detected. When clusters overlap, the nearest labelling approach undergoes common issues such as losing the correct labels. Envelope-clusters detects and resolves the labels assigned to the clusters. Current micro-clusters receive their labels from the previously labelled clusters and vote for the class labels from 'k' nearest neighbours.

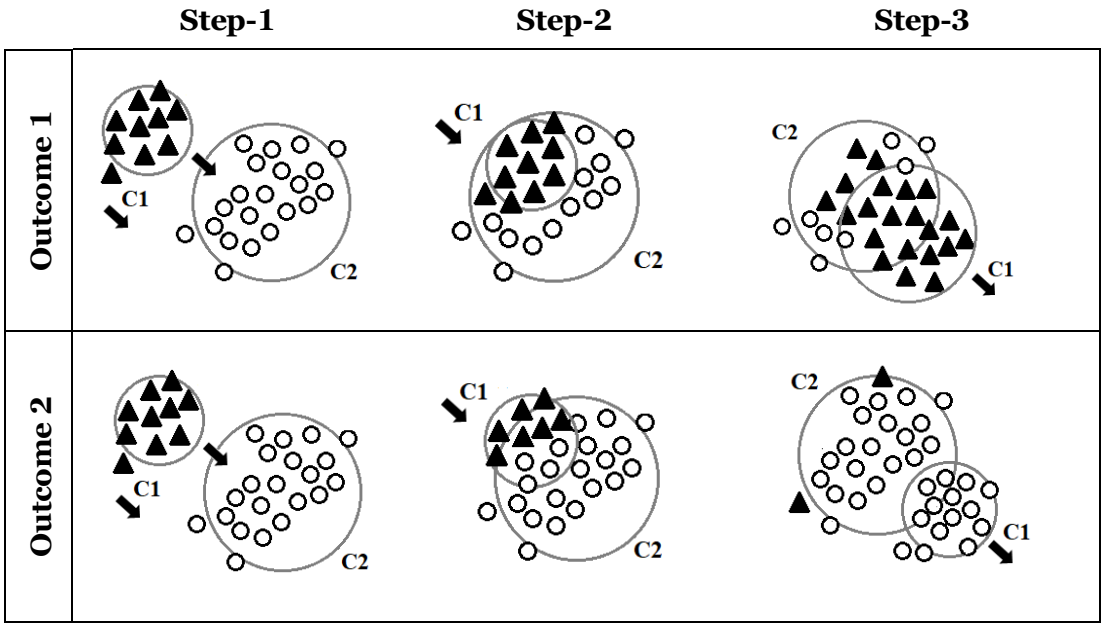


Figure 5.8 Cluster overlapping in 1Csurr dataset [26] showing one class surrounding the other and resulting in two outcomes. 1) C1 transfers its label to 'C2' or 2) C1' gets re-labelled upon intersection with C2.

As shown in Figure 5.8, one group of clusters is stationary i.e. C2, and C1 is crossing it. There are two possible outcomes 1) Triangle cluster 'C1' transfers its label to 'C2' upon intersection with C2 as the circle cluster and converts the Circle cluster to Triangle; or, outcome 2) whereby the Triangle cluster 'C1' gets re-labelled upon intersection with C2, thus turning the Triangle cluster circle. The conflicted clusters receive labels from the corresponding envelope-clusters. Section 5.6.1 and Section 5.6.2 describes conflict detection and resolution steps in detail respectively.

### 5.6.1 Proposed Conflict Detection Method

The confidence level for the cluster labelling on the votes from k-nearest neighbours is calculated as in Equation 5.1. When the confidence level reaches below a user-provided threshold 'α' it reports the drift; otherwise, it transfers the labels to the corresponding clusters.

$$\text{Confidence Level} = \frac{\text{Votes } (\text{Max}(\lambda) - \text{Min}(\lambda))}{\sum N} \tag{5.1}$$

Where, λ are the class votes, Min and Max are the minimum and maximum number of votes per class and 'N' are total votes. Figure 5.9 shows a plot for the 1Csurr dataset [125] as an example; circle and triangle clusters are successfully labelled from previous clusters (unfilled circle and triangles) with high confidence levels.

The figure shows 6 conflicts (diamond) at threshold α = 0.5 and 3-nearest neighbours. For λ= [1,2] i.e. '1' vote for 'class 0' and '2' votes for class '1', the confidence level is = (2-1)/3 =

0.3 < 0.5 threshold. When there were no conflicts, λ= [3, 0] the confidence ratio = (3-0)/3 = 1.0 > 0.5 resulted in successful label transfer shown in filled circle and triangle clusters.

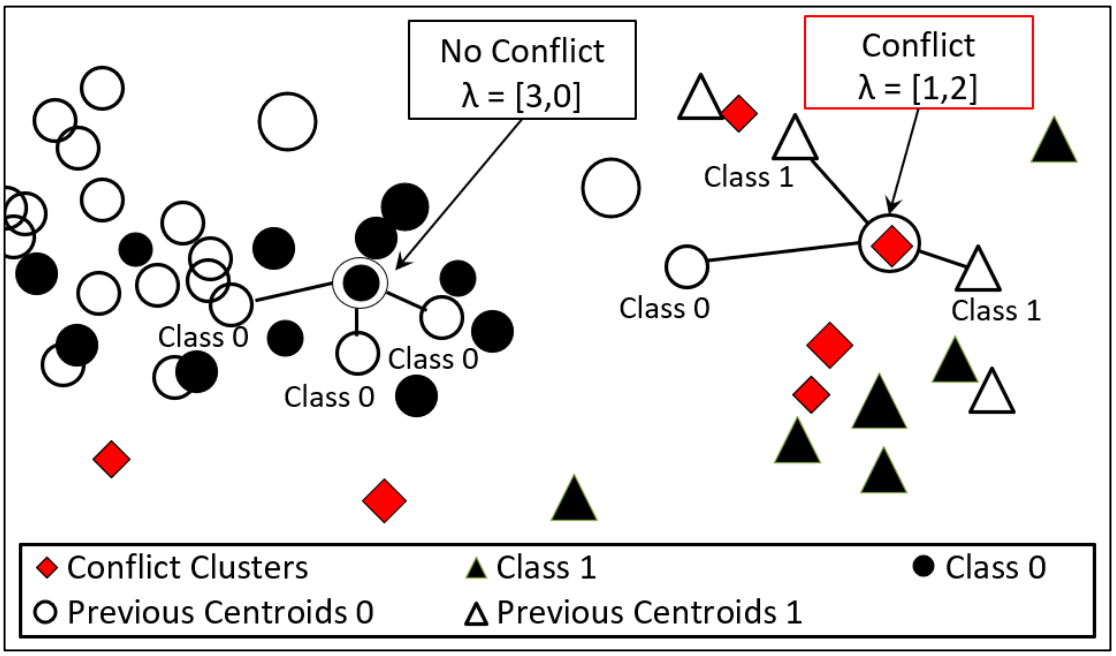


Figure 5.9 Conflict detection in micro-cluster using class votes from 3-nearest neighbours using threshold α = 0.5. The diamond shape representing conflicts in cluster labelling due to low confidence value.

### 5.6.2 Conflict Resolution

PSDSL generates envelope-clusters by transforming the micro-clusters into micro instances. Envelope-clusters are generated using centroid-based clustering, such as K-Means. When no cluster overlaps are detected, the concept of Envelope-Clustering applies online micro-clustering to calculate and store the summary statistics of the data stream; thus, applying it offline to generate macro-clusters when overlaps are detected, increases the processing speed of micro-clustering. Figure 5.10 shows that the labels to the 'red' diamonds, which are conflicted micro-instances are assigned using the nearest envelope clusters. These nearest clusters are determined by calculating Euclidean distances among each conflicted micro-instances and the envelope clusters.

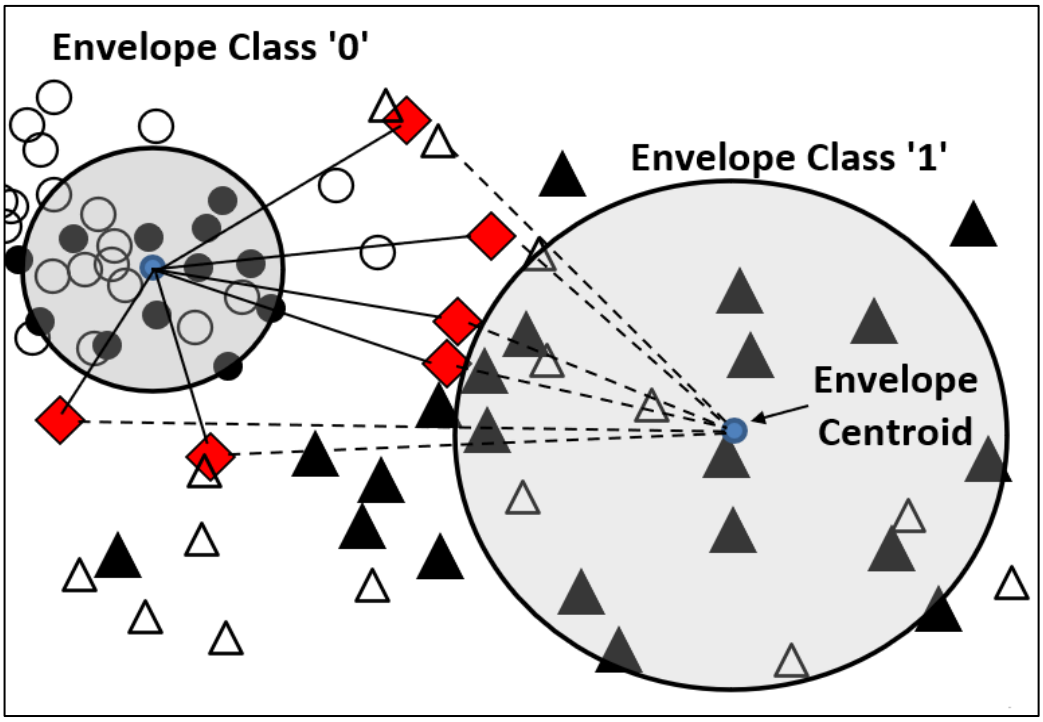


Figure 5.10 Envelope-Clustering for conflict resolution. The filled circle and triangle represent recent micro-instances, and the opaque circle and triangles are previous micro-instances.

## 5.7 Hyperparameter Tuning

This step is an essential automated hyperparameter tuning approach used in PSDSL that determines the number of centroids 'k' to be used in clustering using the few initial labelled instances. The cluster evaluation uses extrinsic methods to assign a score to the clustering when the ground truth is available. It applies the mean values of (F1-P), (F1-R) and purity (P) [126] to determine the optimum value of 'k'. The Purity is a measure of the quality of clusters and determines the extent to which clusters contain a single class.

## 5.8 PSDSL Pseudo code

The pseudocode for PSDSL is depicted in Algorithm 5.1. In EVL, initially available labelled examples are of significant use in hyperparameter tuning to determine the optimal values for 'k' (number of centroids). This parameter tuning approach is described in Algorithm 5.2. These labelled examples also play an important role in automatically deciding the best pseudo-labelling approaches, such as self-learning or CGC.

**Algorithm 5.1 PSDSL ($\tau$, S, $\varepsilon$, $\Phi$, $\mu$, $\theta$, $\rho$, $K_{max}$)**

**Input** S: $\{x_i; y_i\}$ ; i = 1, ..., n: Stream of examples
$\tau$: Initial set of labelled examples
$\varepsilon$: Set of Heterogeneous Base Classifiers {1...m}
$\mu$: Micro-clustering algorithm such as CluStream
$\Phi$: Clustering algorithm like K-Means
$\theta$: Pool Size
P: purity threshold, default is 0.95
$K_{max}$ : Maximum number of centroids

```
1  pool ← ∅
2  K_best, Φ_Purity, μ_Purity ← TuneParameter (τ, Φ, μ, K_max)        // (Algorithm 5.2)
3  SwitchingLearningStates (Φ_Purity, μ_Purity)                      // (Algorithm 5.3)
4  for i = 1 to n                                                    // loop over instances
5      C_micro ← Clustering (μ, x_i)
6      if (x_i is labelled) then                                     // receive initial labelled data
7          C ← GTClustering (Φ, τ)                                   // ground truth cluster
8          ε ← buildClassifier (τ)                                   // build initial classifier
9      else                                                          // unlabelled instances arrive
10         if (self-Learning = true) then
11             τ' ← getVoteforinstance (ε_best, x_i)                 // predict labels
12             ε ← TrainClassifier (τ')
13         else
14             pool ← pool ∪ {x_i}
15             if (i mod θ = 0) then                                 // periodic execution
16                 drift ← false;
17                 μpoints ← DetectClusterDrift (C, C_micro)         // (Algorithm 5.4)
18                 if (drift = true) then
19                     C_Envelope ← Clustering (Φ, C, μpoints,
                       k_best)
20                     LabelCentroids (C, C_Envelope)
21                     C_Envelope ← {CEnvelope}∪ { C }
22                     C^{t+1} ← LabelCentroids (C_Envelope , C_micro )
23                 else
24                     C^{t+1} ← Clustering (Φ, C, pool, k_best)
25                     LabelCentroids (C, C^{t+1})
26                     C^{t+1} ← {C^{t+1}} ∪ {C}                     // merge centroids
27                 end if
28                 τ' ← Labeldata (pool, C^{t+1})
29                 ε ← TrainClassifier (τ')
30                 pool ← ∅
31             end if
32         end if
33     end if
34 end for
```

The PSDSL algorithm maintains a set of 'm' base classifiers and clusters. Inputs to the algorithm are 'n' training examples in which $\tau$ instances are labelled, followed by complete unlabelled examples. A list of all the parameters used in the algorithms is available in Table

5.2. As shown in Algorithm 5.1, both labelled and unlabelled instances incrementally create the micro-clusters (line 5). When labelled instances arrive (line 6) a clustering algorithm is executed to generate $C^t$ and divide the data into clusters and associates each cluster with one of the classes (line 7) and trains the initial classifier $\varepsilon$.

Upon arrival of unlabelled instances (line 9) it determines the learning state described in Algorithm 5.3, if the self-learning state is active, it applies prequential evaluation to predict the pseudo-labels by using the best classifier in the ensemble and re-training the ensemble on these predicted pseudo-labels (line 10-12). In a condition when self-learning state is inactive, it performs CGC (line 14-32). The unlabelled examples are stored in a pool or batch of size $\theta$ (line 14) the value of which is set by the user and periodically performs the tasks listed in lines (16 - 30). The pool data is periodically analysed for potential drifts due to cluster overlaps in micro-clusters (line 17). This process returns labelled micro-cluster instances and reports the state of drifts as described in Algorithm 5.4.

If drift is detected, envelope-clusters are formed using micro-cluster instances such that each cluster represents a class in the data (line 19). Envelope-clusters then transfer their labels to the nearest conflicted micro-clusters (lines 20-22). If no drift is detected, the clustering algorithm $\Phi$ obtains $C_{t+1}$ on the pool data (line 24) by applying the best values of 'k' obtained in Algorithm 5.2. Each new centroid receives its label from the nearest centroids using the Euclidean distances between $C_t$ and $C_{t+1}$ (line 25). Finally, a set of heterogeneous base classifiers is trained using the pseudo-labelled instances (line 29). The source code for the algorithms is available in APPENDIX III.

### 5.8.1 Algorithm: Hyperparameter Tuning

As outlined in Algorithm 5.2, there are three input parameters, a set of labelled instances, a clustering algorithm, and $K_{max}$ which is the maximum number of centroids (k) provided by the user. Initially, the ground truth centroids are generated using the labelled instances (line 2) such that {c = k} where 'c' is the number of classes. Lines 3 and 4 generate and evaluate purities µPurit for micro-clusters. Line 5 begins the loop to determine the best value of 'k' by iterating in the range from 'k=2' to $K_{max}$. In line 6, new clusters are generated after eliminating the ground truth labels from the labelled data. A user-provided clustering algorithm is applied while passing the incremented values of 'k'. In line 7, the ground truth clustering and current clustering are evaluated, and the corresponding F1-P, F1-R, and P are stored in sets of 'Fprp' and 'Purities' lines 8 and 9 respectively.

### 5.8.2 Algorithm: Switching Learning States

The switching algorithm takes $\mu_{Purity}$ and $\Phi_{Purity}$ inputs, and the parameter $\rho$ is the switching threshold set by the user. Algorithm 5.3 outlines the switching algorithm; The ensemble '$\varepsilon_{GT}$' (Ground Truth) is trained on initial labelled data (line 5), this training set splits in the ratio of 80% and 20% (line 6). Another ensemble $\varepsilon_{PL}$ (Pseudo Label) 'trains on 80% of this training examples, then $\varepsilon_{PL}$ predicts the pseudo-labels for the remaining 20% and retrains itself on the

predicted pseudo-labels (line 7,8). As the ground truth labels of initial training set are known, the predictive performance of both $\varepsilon_{GT}$ and $\varepsilon_{PL}$ is compared, if the overall prediction accuracy of $\varepsilon_{PL}$ becomes higher than the $\varepsilon_{GT}$, the self-learning state becomes active, otherwise the pseudo-labelling is suspended.

### 5.8.3 Algorithm: Detect Cluster Drift

The algorithm to detect cluster drift is available in Algorithm 5.4 the current micro-clusters $C_{t+1}$ are associated with previous clusters $C_t$ by measuring similarity between 'k' nearest centroids q t i; i = {1, ..., k} using Euclidean distance, i.e. Dist ($q^t$, $q^{t+1}$) (line 7). The 'k' nearest clusters votes for the class labels to the current clusters (line 12). To calculate the conflict ratio, min-max values of the votes are applied to the formula (line 15). If the ratio reaches above the user-provided drift threshold, current micro-clusters are assigned the label of the majority class vote.

**Algorithm 5.2 PSDSL Auto tuning Parameter 'k' (τ, Φ, μ, $K_{max}$)**

**Input**: τ : initial set of labelled examples
Φ: Clustering Algorithm
μ: Micro-clustering algorithm
$K_{max}$ : Maximum number of Centroids

**Output**: K, $\Phi_{Purity}$ , $\mu_{Purity}$

```
1  i ← 0
2  C  ← GTClustering (Φ, τ)
3  C μ ← Clustering (μ, τ)
4  μ Purity ← evaluateClustering (C, C μ)
5  for k = 2 to Kmax
6  |  C t + 1 ← Clustering (Φ, τ, k)
7  |  {F1P, F1R, P} ← evaluateClustering (C, C t + 1)
8  |  Fprp[i] ← (F1P+F1R + P)/3
9  |  Purities[i] = P;
10 |  i++;
11 end for
12 k ← argmax (Fprp) + 2
13 Φ Purity ← max (Purities)
```

**Algorithm 5.3** SwitchingLearningStates ($\Phi_{Purity}$, $\mu_{Purity}$, ρ)

**Input**: τ : initial set of labelled examples
$\{\varepsilon_{GT}\}_m^1$: Set of Classifiers train on true class labels
$\{\varepsilon_{PL}\}_m^1$: Classifiers train on pseudo-labels
μ : Micro-clustering algorithm, ρ: purity threshold

**Output**: none

```
1  Self-learning ←false
2  CGC ←false
3  split_pos ← trainSize * 0.8
4  for i = 1 to size| τ |
5  |  εGT ← train (τ i)
```

```
    if (i < split_pos) then ε_PL ← train (τ_i) else
        τ'_i ← predict (ε_PL , τ_i)
        ε_PS ← train (τ'_i)
    end if
end for
C_micro ← Clustering (μ , τ)
μ ACC% ← evaluate (C_micro , C_GT )
if (μ ACC% > ρ then CGC ←true else self-learning ←true  end if
if ACC (ε_PS ) < Acc(ε_GT ) then self-learning ←false end if
```

Algorithm 5.4 DetectClusterDrift (knn, C, $C_{micro}$)

**Input**: C: past clusters, $C_{micro}$ : Current micro-clusters
α : Drift Threshold, knn : num of nearest neighbours,
c : No. of classes
**Output**: Drift state, labelled micro-clusters

```
Labelledcluster ← ∅
conflict ← false
λ [c]
[q, q^{t+1} ]← getcentroids[C, C_micro ]
for (i = 0 to |C|)
    for (j = 0 to | C_micro |)
        distances[ i ][ j ] ← dist (q_i ,q^{t+1}_j)
    dist ← sort(distances[ i ])
    for ( j = 0 to | dist |)
        if (binarySearch (dist_j , knn) then
            classID ← C_j          // 'k' nearest distances
            λ [ classID++] ← 1     // add vote to class
    end for
    maxpair[ ] = minMax(λ);
    ratio ← (maxpair[0] - maxpair[1])/sum(maxpair)
    if (ratio > α) then
        C_micro ← label(C_micro , argmax(λ)) // Set id of cluster to max class vote.
        C_Labelled ← { C_Labelled } U { C_micro }   // Add to labelled clusters
    else
        conflict ← true
    end if
end for
```

## 5.9 Complexity of PSDSL

PSDSL is a single pass algorithm which splits the data stream into batches of predefined size such that each batch contains n examples. These batches are sequentially processed, requiring less computational time and space because only the information regarding the centroids and data points of the current batch is stored in the memory. The complexity of PSDSL depends on the choice of learners. PSDSL intelligently switches learning strategies and applies an HDWM classifier for self-learning or K-Means and CluStream for CGC and micro-clustering respectively. Under EVL, when labelled data arrives, PSDSL executes hyperparameter tuning and switch learning strategy only once.

Hyperparameter tuning begins with formation of ground truth clusters which is dominated by the complexity of sorting, which takes O ($n \cdot \log n$) time. Next, it iterates on different values of $K_{max}$ to generate clusters, this phase takes O ($n \cdot k \cdot i \cdot K_{max}$) time, where i is the number of iterations. It takes O ($n \cdot k$) space because only the information of distances and centroids are stored in the memory.

State switching (Algorithm 5.3) trains and evaluates HDWM ensemble classifier 'Ɛ', online micro-clusters 'μ' and offline clusters Φ. The overall time complexity for classifiers for 'τ' number of labelled training examples is O ($\tau \cdot Ɛ$) and for clustering is O ($\tau_{\mu} + \tau_{\Phi}$). The time complexity of online ensemble classifiers heavily depends on the choice of base classifiers. HDWM applies NB [38], HT [39] and K-Nearest KNN [40] base classifiers. Based on the worst time complexity of these base classifiers, the total time complexity of HDWM is O ($\tau \cdot d$) + O ($d \cdot v \cdot c$). The space complexity for storing the likelihood of each feature with respect to classes is O ($l \cdot d \cdot v \cdot c$). Where 'd' is dimensionality of the attributes, 'v' values per attribute, 'c' is number of classes and 'l' is the current number of leaves.

When unlabelled examples arrive, additional time and space is required for predicting the pseudo-labels for unseen examples. PSDSL selects a best performing base classifier '$Ɛ_{best}$' and assigns it for pseudo-labelling in small batches, this phase takes O(n) time for predictions. For clustering the nth batch using i number of iterations it takes O ($n \cdot k \cdot i$) time and O ($n \cdot k$) space to store the centroid and data points. Micro-clustering takes O ($q \cdot n \cdot i \cdot N_{Init}$) for the online phase, where q is the number of micro-clusters and $N_{Init}$ is the initial number of examples. For merging two micro-clusters it takes O ($q \cdot n$) and offline phase takes O ($q \cdot n \cdot k \cdot i$) time.

The time complexity of drift detection (Algorithm 5.4) depends on the time required to compute the distances from previous centroids to the current centroids. The drifts are detected at a regular interval 'θ'. In the worst case, when all the batches contain drift, the time complexity to compute distances is O (log n) using binary search. Once the drifts are detected, transforming the micro-clusters into micro-instances and generation of envelope-clusters requires O (1) and O ($n \cdot k \cdot i$) time respectively.

Therefore, the total time complexity of PSDSL in the worst scenario is O ($n \cdot k \cdot i$) + O ($q \cdot n \cdot N_{Init}$) + O ($q \cdot n$) + O ($q \cdot n \cdot k \cdot i$) which approximates to O ($q \cdot n \cdot k \cdot i$ ) ≈ O (N) as n is much larger than 'q', 'k' and 'i'. The value of parameter k is constant, which has already been tuned using Algorithm 5.2. PSDSL requires fewer iterations for convergence because the initial centroids are trained on initial labelled data from the streams and the new centroids obtain their labels from the nearest centroids. COMPOSE on the other hand is of order O ($n^{(d+1)/2}$) i.e. exponential in dimensionality [79]. SCARGC has the worst time complexity O ($n \cdot k \cdot i$).

## 5.10 GUI for online Semi-Supervise Learner

The purpose of developing the GUI for SSL is to visualize the results of both classification and clustering and analyse the results and compare results for different configurations. Furthermore, for qualitative analysis of the clustering and classification models.

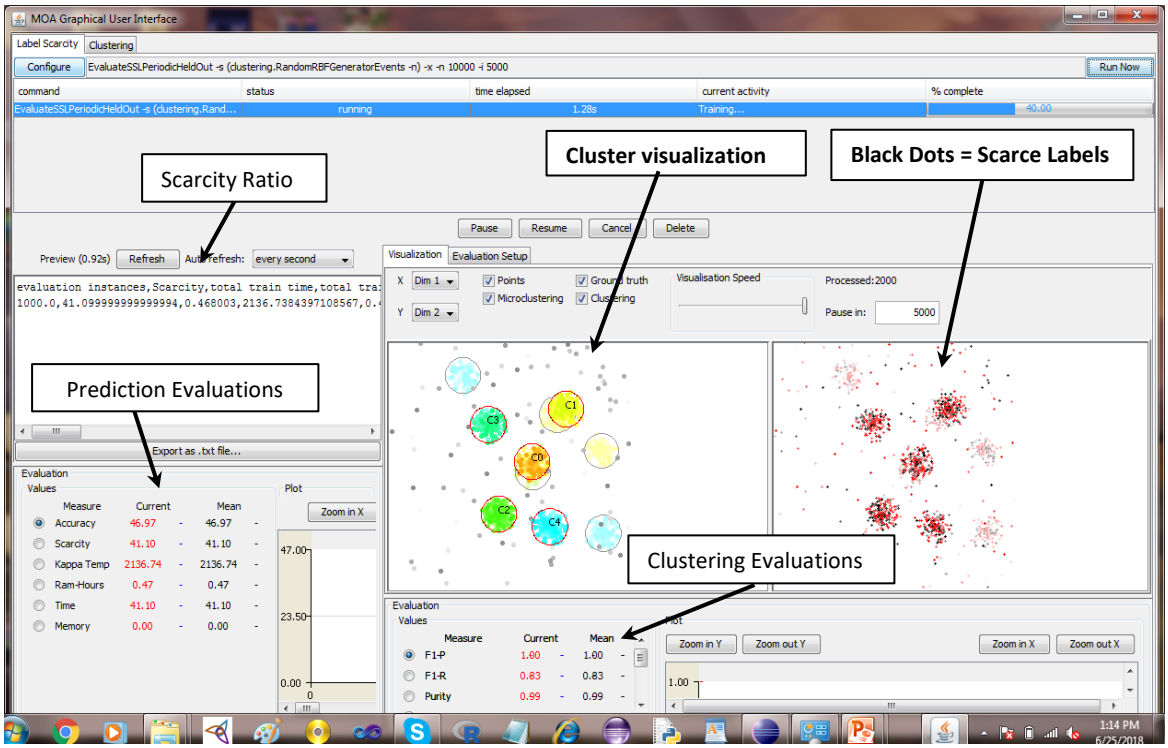


Figure 5.11 GUI for online SSL for data streams in MOA.

The clustering results as shown in the right part of the GUI as shown in Figure 5.11 includes online evaluation of clusters such as F1, Precision, Recall, purity etc. The GUI also provides option to pause and resume the stream at any time, adjust the speed of visualization, choose the attributes to be displayed and visualize both the labelled and unlabelled data points in respective windows.

The development process includes the extension and implementation of new classes in MOA. The classes SSLMainTask is responsible to execute and maintain the SSL learning task. Similarly, the SSLearner class controls the features and functionality of classifiers and clusters and controlling the scarcity ratio of the training examples. The SSLVisualizer is responsible for presenting and timely updating examples, the clustering and micro clustering and evaluation results on the screen.

The pre-labelling accuracy, micro and macro clustering and ground truth clusters along with switching of learning strategies are visually displayed. The classification results are shown in the left part of the GUI which includes online evaluation of classifiers such as the current and mean prediction accuracies, class label scarcity, time and memory details.

## 5.11 SSL Prequential and Periodic Holdout Tasks

Two online SSL learning tasks has been proposed, i.e. SSL Prequential and SSLHoldout. The Holdout evaluation provides more accurate measurements on more recent data, as it has a forgetting mechanism. There is an option to include this mechanism in Prequential evaluation by using sliding window or fading factors that weigh the instances using a decay factor. Algorithm 5.5 outlines overall stages of SSL periodic holdout task.

**Algorithm 5.5 SSLHoldout Evaluation ($\{x, y\}_n^1$, $n_{test}$, $m_{bound}$, α, )**

Input: $\{x, y\}_n^1$ : $n_{train}$ : Stream of examples and class label
$n_{test}$ : Holdout examples as a test set
$m_{bound}$: the maximum memory allocation
α: Percentage of initial labelled examples

**Output**: none

```
1   While evaluation is required do
2       for i = 1 to n_train do                          // iterate on labelled examples
3           prob ← random%100
4           if (prob >= α) then
5               x_i ← {x_i, y_i}                         // remove the class label from the example
6               Train and update the unsupervised model on {x_i}, ensuring m_bound is valid
7           end if
8       λ = Classify (x_i ) // predict pseudo-label using supervised model
9       if y_i is available then
10          Train and update the supervised model on {x_i, y_i}
11      else
12          Train and update the supervised model on pseudo-labels {x_i, λ}
13      end if
14      Evaluate pseudo-labelling accuracy (y_i, λ)
15      for i = 1 to n_test do
16          Get the next example from the stream
17          Test the model and update model accuracy
18      end for
19  end while
```

In the proposed SSLHoldout evaluation, the data stream is divided into two mutually exclusive subsets. Initially, the class labels of training examples are randomly eliminated (line 3-5), and an unsupervised model is trained (line 6). Next, the pseudo-labels 'λ' are predicted using a supervised model (line 8). In case, if the arriving example is labelled, the supervised model is trained on true class labels, otherwise pseudo-labels are used for training and updating. A set of predefined size of examples are used in training phase before an evaluation is performed on the test set. Similarly, in the proposed SSLPrequential the error of the model is computed from the sequence of instances. For each instance in the stream, the predictive model makes a prediction, and then incrementally update the model. A specified ratio of the true class labels of examples are removed and the evaluation of both clusters and classifiers are performed along with the visualisation of examples and clusters. The source code for SSL periodic held out task is available in APPENDIX IV.

## 5.12 Evaluation of PSDSL

This section empirically evaluates PSDSL algorithm against existing standalone EVL approaches namely SCARGC [31], $LEVEL_{IW}$ [74], COMPOSE [22], and MClassification [34] on benchmarks NSE datasets [31] as well as MOA data streams and real-world datasets. These are the most relevant approaches dealing with EVL and NSEs. The APT algorithm [76] was not included in the analyses, as its steep computational complexity was prohibitive on running some of the larger datasets. A comparative analysis among these algorithms has been presented in Section 2.3 .

To verify statistically significant differences between algorithms, the Friedman test was applied, which is a suitable non-parametric test for multiple algorithms on multiple datasets [49]. The Friedman test was applied with α=0.05 to test the null hypothesis that there is "no statistical difference between the algorithms". The Nemenyi post-hoc test [50] has been applied to identify which pairs of algorithms differ from each other. In EVL few initial ground truth labels are available; therefore, internal evaluations were applied to the clusters i.e. Purity, Precision and Recall [124].

### 5.12.1 Experimental Setup

The evaluation procedure used are Kappa statistics and prequential testing. Prequential testing is a facility of the MOA [46] in which each instance is used to test the model before it is used for training, and the accuracy is updated incrementally. Prequential accuracy estimate is appropriate when all classes are approximately balanced [53]. Kappa statistics is a more sensitive measure for quantifying the predictive performance of streaming classifiers since it cannot be ascertained whether the classes were balanced.

The PSDSL was compared with existing EVL approaches, static and benchmark setting. To determine how PSDSL performs with and without pseudo-labelling the 'Static' approach was used in which PSDSL does not apply pseudo-labelling. Further, to analyse the consequences of unlabelled examples in the data stream and their impact on predictive performance, 95% of the class labels were removed from each dataset and PSDSL was compared in the 'benchmark' setting in which all the training examples are labelled. The MOA commands to execute these experiments are available in APPENDIX II.

All the experiments are evaluated in terms of time consumption and predictive performance. Processing time is measured in seconds and is based on the CPU time used for training and testing. All the experiments were performed on machines with Core i7 @ 3.4 GHz, 4 GB of RAM. The experiments performed on non-stationary datasets [125] using MOA-generated streams [46] and real-world datasets. The details of parameters used in the experiments for these existing EVL approaches are provided in Table 5.1.

Table 5.1 Algorithms and parameters used in the experiments.

| **Algorithm** | **Description** |
|---|---|
| Static | The PSDSL classifier is not updated after it is trained with the first examples in the data streams. i.e. no pseudo-labelling is applied. |
| SCARGC | Applied KNN base classifier and applying the parameters suggested by Souza et al. [31] |
| Benchmark | PSDSL classifier is applied on 100% labelled examples. |
| COMPOSE | Gaussian mixture models (GMM) as core supports extraction, applied the parameters suggested by Dyer et al [22] |
| MClassification | r = 0.1 and \|T\| = 150 and where \|T \| represents the size of the initial labelled set and r is the maximum radius of MC [34] |
| $LEVEL_{IW}$ | importance weighted least squares probabilistic base classifier using the default parameters suggested by Umer et al [74] |

### 5.12.2 PSDSL Learning Parameters

The parameter used in Evaluate EVL Prequential and PSDSL are shown in Table 5.2 and Table 5.3.

Table 5.2 Learning parameters used in Evaluate EVL Prequential.

| Param | Command | Description | Default |
|---|---|---|---|
| -l | SSLearner | Semi-Supervised learner to train | SSLearner |
| -s | stream | Stream to learn from | RandomTreeGenerator |
| -k | normalise | Normalize data stream | Do not Normalize Stream |
| -e | evaluatorOption | Classification performance evaluation method | Window Classification Performance Evaluator |
| -n | noiselevel | Set Noise Level to data stream | false |
| -r | trainsize | Size of labelled data | 50 |
| -i | instanceLimit | Maximum number of instances to test/train | 1000 |
| -f | sampleFrequency | How many instances between Samples of the learning performance | 100 |

Table 5.3 Learning parameters used in PSDSL.

| Parm. | Command | Description | Default |
|---|---|---|---|
| -l | learner | Set the classifiers (HDWM) | HoeffdingTree -l MC kNN HoeffdingTree -l NBAdaptive NaiveBayes |
| -q | clusturer | Clustering algorithm | Clustream |
| -h | horizon | timeWindowOption | 1000 |
| -k | maxNumKernels | Maximum number of micro kernels to use | 100 |
| -x | prelableing | Switch “No Pre-labelling" or "Use Pre-labelling" | Use Pre-labelling |
| -p | maxcluster | Number of Max Clusters | 15 |
| -k | knn | Number of nearest neighbours | 3 |
| - α | threshold | When the confidence level reaches below ‘α’ it reports the drift and generates envelope clusters | 0.1 |
| -u | PurityThreshold | Threshold for switching pre-labelling strategies i.e. self-learning, micro or CGC. | 0.95 |

### 5.12.3 Non-Stationary datasets

Non-stationary datasets used in the experiments were provided by the authors of SCARGC [31] and are available to the machine learning community [125]. These datasets have been randomised and made available for further research [54]. They provide datasets with incremental changes over time. Here, Unimodal Gaussian datasets represent two bi-dimensional Gaussian clusters rotating around a common axis. The distance between the Gaussian components changes over time. Class overlap exists in these datasets. The datasets UG-2C-2D, UG-2C-3D, and MG-2C2D were originally proposed by Dyer et al [22].

### 5.12.4 MOA Data Streams

The artificial data streams used in the experiments are generated through the MOA workbench [46]; the number of instances is 100,000 and the batch size is 1000 in all the streams. The MOA commands to generate these streams are available in APPENDIX II.

### 5.12.5 Real-World Dataset

Keystroke dataset [135] task is to predict one of four users based on their typing patterns. The dataset contains keystroke records obtained from the users in 8 different sessions who typed a fixed password. The description of the datasets used in the experiments is provided in Table 5.4 and Table 5.5.

Table 5.4 Description of Benchmark datasets.

| Datasets | # of Instances | # of Feature | # of Classes | Drift Interval | Class Overlap |
|---|---|---|---|---|---|
| 1CDT | 16,000 | 2 | 2 | 400 | No |
| 1CHT | 16,000 | 2 | 2 | 400 | No |
| 1CSurr | 55,283 | 2 | 2 | 600 | yes |
| 2CDT | 16,000 | 2 | 2 | 400 | yes |
| 2CHT | 16,000 | 2 | 2 | 400 | yes |
| 4CE1CF | 173,000 | 2 | 4 | 750 | No |
| 4CR | 144,000 | 2 | 4 | 400 | No |
| 4CRE-V1 | 125,000 | 2 | 4 | 1,000 | yes |
| 4CRE-V2 | 183,000 | 2 | 4 | 1,000 | yes |
| 5CVT | 24,000 | 2 | 5 | 1,000 | yes |
| FG_2C_2D | 200,000 | 2 | 2 | 2,000 | No |
| GEARS_2C_2D | 200,000 | 2 | 2 | 2,000 | No |
| MG_2C_2D | 200,000 | 2 | 2 | 2,000 | yes |
| UG_2C_2D | 100,000 | 2 | 2 | 1,000 | yes |
| UG_2C_3D | 200,000 | 3 | 2 | 2,000 | yes |
| UG_2C_5D | 200,000 | 5 | 2 | 2,000 | yes |

C=Class, D=Diagonal, V=Vertical, H=Horizontal, T=Transaction, R=Rotating, E=Expanding U= Unimodal, G= Multimodal, G= Gaussian

The batch size for the MOA Stream is 300 and for keystroke is 150. The information about drifts and class overlap is not available for the real-world datasets. In Section 5.13 and 5.14, the predictive capabilities of PSDSL were tested on MOA data streams, benchmark non-stationary datasets and real-world datasets.

Table 5.5 Description of MOA streams.

| MOA Stream | Instances | Feature | Classes | Drifts |
|---|---|---|---|---|
| SEA | | 3 | 2 | 2 |
| RandomTree | | 10 | 2 | 2 |
| LED | 100K | 24 | 10 | 1 |
| Wave | | 21 | 3 | 1 |
| Hyperplane | | 10 | 2 | 3 |
| RRBF | | 2 | 5 | 2 |
| Keystroke [135] | 1600 | 10 | 4 | NA |

## 5.13 Comparative Analysis of PSDSL on Benchmark Datasets

Predictive accuracies of PSDSL, COMPOSE, $LEVEL_{IW}$, SCARGC and MClassification (MC) were evaluated on benchmark non-stationary datasets [125] that have also been used in the original SCARGC publication. Table 5.6 shows the Friedman statistic $X^2_r$ is 18.93 (df =5, n = 15). The p-value = .0019 shows significant difference in the algorithms at ($p < .05$). The number in the brackets represents the ranks, the lower the rank and the higher the predictive performance.

Table 5.6 Average accuracies on benchmark datasets.

| Datasets | Static | COMPOSE (GMM) | $LEVEL_{IW}$ | SCARGC (1-NN) | MC | PSDSL |
|---|---|---|---|---|---|---|
| 1CDT | 99.0(6) | 99.8(3) | **99.9(1)** | 99.7(5) | 99.8(2) | 99.6(5) |
| 1CHT | 94.5(6) | 99.3(3) | **99.5(1)** | 99.2(4.5) | 99.3(2) | 99.0(5) |
| 1CSurr | 65.6(6) | 89.7(4) | 91.3(3) | 94.3(2) | 85.1(5) | **94.5(1)** |
| 2CDT | 55.0(6) | **95.9(1)** | 58.3(5) | 90.9(3) | 95.2(2) | 90.7(4) |
| 2CHT | 54.5(5) | **89.6(1)** | 52.1(6) | 85.0(4) | 87.8(2) | 85.8(3) |
| 4CE1CF | 94.7(3) | 93.9(6) | **97.7(1)** | 94.0(5) | 94.3(4) | 94.7(2) |
| 4CR | 25.2(6) | 99.9(2.5) | **99.9(1)** | 99.9(5) | 99.9(2.5) | 99.9(4) |
| 4CRE-V2 | 26.2(5) | **92.3(1)** | 24.1(6) | 91.9(2.5) | 91.5(4) | 91.8(3) |
| 5CVT | 48.1(4) | 45.1(5) | 33.1(6) | **90.1(1)** | 88.4(2) | 84.9(3) |
| FG_2C_2D | 81.3(4) | 95.5(2) | **95.7(1)** | 95.1(3) | 62.4(6) | 64.9(5) |
| GEARS | 94.9(5) | 95.8(4) | **97.7(1)** | 95.9(3) | 94.7(6) | 95.9(2) |
| MG_2C_2D | 51.6(6) | **93.2(1)** | 85.4(3) | 92.7(2) | 80.5(4) | 64.5(5) |
| UG_2C_2D | 45.8(6) | **95.7(1)** | 74.3(5) | 95.5(2.5) | 95.2(4) | 95.5(2) |
| UG_2C_3D | 64.1(6) | **95.2(1)** | 64.6(5) | 94.7(3) | 94.7(4) | 94.8(2) |
| UG_2C_5D | 69.2(6) | **92.1(1)** | 80.1(5) | 90.9(4) | 91.2(2) | 91.0(3) |
| Keystroke | 68.7(6) | 87.2(4) | 90.5(2) | 87.7(3) | **90.6(1)** | 85.3(5) |
| **Average** | 64.93(5.3) | 91.2(2.5) | 77.8(3.2) | **93.6(3.1)** | 90.7(3.3) | 89.6(3.3) |

Figure 5.12 shows a critical difference diagram on ranked accuracies for non-stationary datasets. For 6 algorithms and 16 datasets, the Critical Difference (CD) for the Nemenyi [50] at (α=0.05) is (CD=1.82). The solid bar shows no significant differences between COMPOSE, $LEVEL_{IW}$, SCARGC, MClassification and PSDSL, however these performed significantly better than 'Static'.

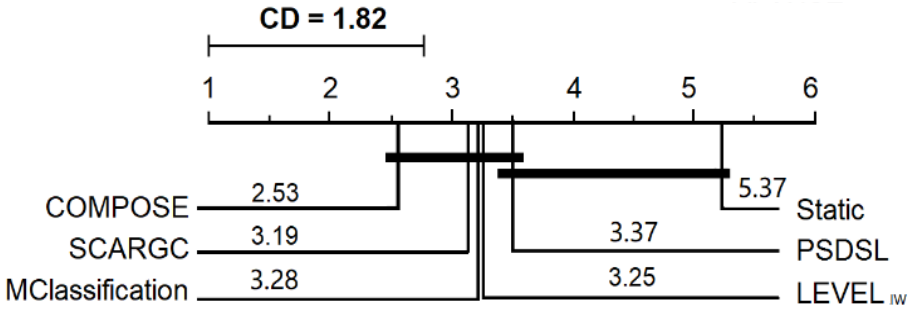


Figure 5.12 Critical Difference diagram for Non-Stationary Dataset's accuracies

Table 5.7 presents the evaluation time in seconds; the results show that PSDSL achieved similar accuracies in less average computation time (8.01 seconds) on non-stationary datasets.

Table 5.7 Evaluation time in seconds (Non-Stationary Datasets).

| **Datasets** | **COMPOSE (GMM)** | **$LEVEL_{IW}$** | **SCARGC (1-NN)** | **MC** | **PSDSL** |
|---|---|---|---|---|---|
| 1CDT | 4.2 | 15.0 | 1.5 | 64.5 | **1.2** |
| 1CHT | 4.0 | 15.3 | 1.2 | 62.6 | **0.4** |
| 1CSurr | 7.3 | 43.8 | 4.2 | 220.9 | **3.8** |
| 2CDT | 2.9 | 15.7 | 1.0 | 62.8 | **0.7** |
| 2CHT | 3.5 | 15.8 | 1.6 | 60.7 | **0.9** |
| 4CE1CF | 44.1 | 137.8 | 15.1 | 775.9 | **10.7** |
| 4CR | 55.9 | 148.3 | 12.7 | 608.0 | **7.0** |
| 4CRE-V2 | 34.8 | 147.8 | 15.2 | 641.6 | **10.9** |
| FG_2C_2D | 16.0 | 185.7 | 8.2 | 870.2 | **6.2** |
| GEARS_2C_2D | 14.4 | 186.4 | 17.5 | 497.7 | **10.0** |
| MG_2C_2D | 15.4 | 190.8 | 18.0 | 740.5 | **10.6** |
| UG_2C_2D | 16.9 | 72.7 | 9.1 | 362.8 | **6.0** |
| UG_2C_3D | 15.6 | 176.5 | 19.4 | 881.7 | **16.0** |
| UG_2C_5D | 15.9 | 176.8 | **21.2** | 977.3 | 27.2 |
| **Average** | 17.9 | 109.2 | 10.4 | 487.5 | **8.0** |

Thus, $LEVEL_{IW}$ is found to be the second-lowest performing algorithm in terms of computational complexity after MClassification and performs significantly worse than all other algorithms except SCARGC and PSDSL. In another experiment, Naïve Bayes [38] classifier is applied as a base classifier, Table 5.8 shows an average 7.18 % improvements in the prediction accuracies when pre-labelling was applied.

Table 5.8 Predictive Accuracies (%) PSDSL applied using Naïve Bayes Classifier.

| Datasets | Without Pre-labelling | Pre-labelling | Gain |
|---|---|---|---|
| UG_2C_2D | 52.516 | 68.357 | +15.841 |
| UG_2C_3D | 62.821 | 68.286 | +5.465 |
| 1CSurr | 63.943 | 73.329 | +9.386 |
| 4CR | 19.771 | 34.076 | +14.305 |
| 4CRE-V2 | 24.080 | 32.892 | +8.812 |
| Stagger | 52.808 | 52.925 | +0.117 |
| RandomTree | 62.783 | 62.873 | +0.090 |
| LED | 44.387 | 48.655 | +4.268 |
| Hyperplane | 61.813 | 70.045 | +8.232 |
| SEA_Mixed | 82.408 | 84.972 | +2.564 |
| RandomRBF | 25.666 | 39.25 | +13.584 |
| CovType | 63.19 | 70.50 | +7.30 |
| Sensor | 14.878 | 18.263 | +3.385 |
| **Average** | **48.54** | **55.72** | **7.18** |

## 5.14 Analysis of MOA Data Streams

Previous sets of experiments are performed on offline datasets, a comprehensive analysis was made on MOA data streams. As can be seen from Table V the average prediction accuracy of SCARGC is highest (93.64%) in all the benchmark datasets, therefore it has been implemented it in the MOA to compare it with our approach. A recent comparative analysis in the literature [81] reports no statistical significance at $\alpha = 0.05$ for classification accuracy among COMPOSE, $LEVEL_{IW}$, MClassification and SCARGC). $LEVEL_{IW}$ performs rather poorly on benchmark datasets with significant between-class overlap. MClassification and $LEVEL_{IW}$ are found to be computationally inefficient.

To analyse SCARGC and PSDSL, Prequential Accuracies, Kappa Statistics and Evaluation time were used and the ranks for each algorithm were calculated. It is noted that SCARGC and PSDSL were compared with the 'Static' and benchmarked approaches, which are described in Table 5.2. The first batch i.e. 300 instances were kept labelled and the class labels of the remaining data stream were removed.

### 5.14.1 Prequential Accuracies

In EVL these accuracies could not be evaluated due to the scarcity of true class labels; as true labels become available, the accuracy is calculated and presented for comparison.

Table 5.9 presents the average accuracy (in %) achieved by the methods over the 12 MOA streams. The best results were highlighted in a comparison between the proposed method PSDSL and SCARGC, benchmark, and Static.

Table 5.9 Predictive Accuracies (%) PSDSL (MOA Streams).

| MOA Stream | Static | Benchmark | SCARGC | PSDSL |
|---|---|---|---|---|
| SEA (Sudden Drift) | 69.9(3) | 77.1(2) | 67.9(4) | **78.0(1)** |
| LED (Sudden Drift) | 27.8(3) | **63.6(1)** | 22.3(4) | 41.7(2) |
| Wave (Sudden) | 75.6(2) | **89.2(1)** | 60.4(4) | 75.3(3) |
| RRBF (Gradual Drift) | 26.8(4) | **98.3(1)** | 97.9(3) | 98.0(2) |
| HP (Incremental) | 53.7(3) | **82.8(1)** | 50.5(4) | 54.5(2) |
| RandomTree (Recurring) | 63.1(3) | 63.7(2) | 56.0(4) | **71.3(1)** |
| SEA (No Drift) | 77.1(2) | 76.9(3) | 68.6(4) | **86.5(1)** |
| LED (No Drift) | 45.1(3) | 64.1(2) | 39.2(4) | **74.0(1)** |
| Hyperplane (No Drift) | 83.1(3) | 83.3(2) | 61.2(4) | **89.8(1)** |
| RandomTree (No Drift) | 60.3(3) | 64.1(2) | 54.8(4) | **65.6(1)** |
| RRBF (No Drift) | 13.7(4) | **94.5(1)** | 50.5(3) | 54.5(2) |
| Wave (No Drift) | 76.4(2) | 75.3(3) | 54.3(4) | **83.1(1)** |
| **Average (Rank)** | 56.0(2.9) | 77.7(1.7) | 57.0(3.8) | **72.7(1.5)** |

The overall results show that PSDSL performed better than all other approaches. The Friedman statistic $X^2_r$ is 24.05 (df =3, n = 11), the p-value = .00002 shows a significant difference in the algorithms at ($p < .05$). The number in brackets represents the rank. To determine which algorithm(s) performed differently, Figure 5.13 is the critical difference diagram on ranked accuracies for MOA streams. The connected solid lines represent groups of algorithms that are like each other, and any two algorithms are significantly different if the difference between their average ranks is at least CD [49]. For 4 algorithms and 12 streams, the CD for the Nemenyi [50] at $\alpha$=0.05 is 1.41. The results show two groups of algorithms, i.e. PSDSL - Benchmark and SCARGC-Statics. Significant differences are found between PSDSL and SCARGC, while the performance of PSDSL is closer to the benchmark, while no significant difference was found between SCARGC and Static.

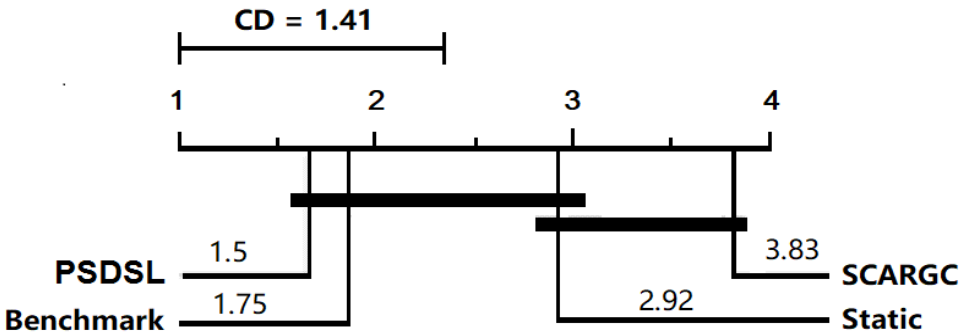


Figure 5.13 Critical Difference diagram for MOA Streams Accuracies, comparison of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at p = 0.05) are connected.

### 5.14.2 Kappa Statistics

The Kappa evaluation measure is widely used in data stream mining, as it can handle both multi-class and imbalanced class problems. The larger the Kappa value, the more generalised and better the classifier. The kappa statistics show similar results compared with average accuracy, in which PSDSL performs significantly better than other algorithms. Table 5.10 provides the Kappa statistics for the experiments.

Table 5.10 Average kappa statistics on MOA streams

| MOA Stream | Static | Benchmark | SCARGC | PSDSL |
|---|---|---|---|---|
| SEA (Sudden Drift) | 42.3(3) | 52.3(2) | 35.6(4) | **55.6(1)** |
| LED (Sudden Drift) | 19.8(3) | **59.5(1)** | 13.4(4) | 34.6(2) |
| Wave (Sudden) | 33.3(2) | **75.4(1)** | 9.3(4) | 32.5(3) |
| RRBF (Gradual Drift) | 8.3(4) | **97.9(1)** | 97.4(3) | 97.5(2) |
| HP(Incremental) | 7.6(3) | **65.6(1)** | 0.6(4) | 9.0(2) |
| RandomTree (Recurring) | 25.2(3) | 25.5(2) | 9.8(4) | **41.4(1)** |
| SEA (No Drift) | 52.9(2) | 52.8(3) | 37.0(4) | **70.0(1)** |
| LED (No Drift) | 39.0(3) | 60.1(2) | 32.5(4) | **71.1(1)** |
| Hyperplane (No Drift) | 66.3(3) | 66.7(2) | 22.3(4) | **79.6(1)** |
| RandomTree (No Drift) | 22.7(3) | **60.1(1)** | 9.7(4) | 29.1(2) |
| RRBF (No Drift) | 1.5(3) | 88.9(1) | 0.6(4) | 9.0(2) |
| Wave (No Drift) | 64.6(2) | 63.0(3) | 31.6(4) | **74.7(1)** |
| **Average (Rank)** | 32.0(2.8) | 64.0(1.6) | 25.0(3.9) | **50.4(1.5)** |

### 5.14.3 Evaluation Time

Table 5.11 presents the Evaluation time in Seconds for Static, Benchmark, SCARGC and PSDSL on MOA Streams.

Table 5.11 Evaluation time in seconds (MOA streams).

| MOA Stream | Static | Benchmark | SCARGC | PSDSL |
|---|---|---|---|---|
| SEA (Sudden Drift) | 7.5 | 9.8 | 120.9 | 35.89 |
| LED (Sudden Drift) | 182.3 | 55.37 | 164.9 | 70.04 |
| Wave (Sudden) | 43.81 | 46.56 | 109 | 116.8 |
| RRBF (Gradual Drift) | 11.93 | 10.5 | 146.2 | 27.9 |
| HP(Incremental) | 29.1 | 27.18 | 122.1 | 50.62 |
| RandomTree (Recurring) | 9.56 | 24.48 | 136.8 | 49.45 |
| SEA (No Drift) | 6.9 | 7.3 | 95.14 | 36.71 |
| LED (No Drift) | 156.75 | 66.2 | 164.7 | 69.20 |
| Hyperplane (No Drift) | 20.14 | 20.54 | 93.34 | 49.93 |
| RRBF (No Drift) | 4.26 | 11.87 | 72.09 | 3.63 |
| RandomTree (No Drift) | 18.89 | 55.07 | 100.98 | 72.17 |
| Wave (No Drift) | 37.4 | 40.76 | 115.2 | 118.2 |
| **Average** | **44.05** | **31.30** | **120.11** | **58.38** |

The results show that PSDSL achieved better average accuracies (72.7%) in less average computation time (58.38 seconds) than SCARGC Accuracy = 57.0% in 120.11 seconds, but not as far as Benchmark and Static because these do not apply pseudo-labelling.

## 5.15 Analysis on Real-world Problem

A keystroke dataset [134][135] applied in the research is a collection of data that records information about keystrokes, such as the time and duration of each keystroke, the keys that were pressed, and the order in which they were pressed. Keystroke datasets can be used for a variety of purposes, including:

1. Biometric identification: Keystroke dynamics, or the way a person types on a keyboard, can be used as a biometric identifier, similar to a fingerprint or a retinal scan. Keystroke datasets can be used to train machine learning models to recognize the unique typing patterns of individuals.

2. Behavioural analysis: Keystroke datasets can also be used to analyse user behaviour, such as how quickly or accurately they type, which keys they tend to use most frequently, and whether they make frequent mistakes.

3. Cyber security: Keystroke datasets can be used to detect fraudulent activity, such as when someone attempts to impersonate another person by typing in their login credentials.

The task of this dataset is to predict one of four users based on their typing patterns. The dataset contains keystroke records obtained from 51 subjects who typed a 10-character password (.tie5Roanl). Each subject typed 400 repetitions of the password over 8 sessions of 50 repetitions each (spread over different days). Each record in the dataset contains information about the time and duration of each keystroke, the key that was pressed, and the order in which the keys were pressed.

### 5.15.1 Features Exploratory Analysis

Feature exploratory analysis is an important step in predictive analysis which helps understanding the characteristics of the different features or variables in a dataset. The keystroke dataset contains data from four users, identified as "S1," "S2," "S3," and "S4," and includes a total of 1600 records, with 400 records for each user. The actual keystroke dataset [135] includes a total of 31 features, such as key down, key, up, hold time, and latency, which can be used to train machine learning models to recognize the unique typing patterns of each user. A total 10 features (fight time) are applied (i.e. one feature for each character in the password) to compare the predictive performance with the benchmarks algorithms. This reduced dataset is available by the authors of the ATISLabs [134]. The Flight time is the time difference between the key press and when it is released.

A correlation heat map [137] is a graphical representation of the correlation matrix, which shows the pairwise correlation coefficients between different variables in a dataset. Figure 5.14 showing correlation heat map of keystroke dataset, the features { . t R a n i } are highly correlated (dark blue on the heat map) with respect to the target class, while the features { a , n } and { l , n } are having medium positive correlation within the features.


Figure 5.14 Correlation heat map of keystroke dataset.

### 5.15.2 Clustering Analysis

In clustering the movement of centroids refers to the movement of the central point or average of a set of data points. It is often used in clustering algorithms to track how the cluster centres move as new data points are added or removed from the dataset. In case of keystroke dataset, the location of centroids can be useful for understanding how the distribution of data is changing over time. This can be due to changes in the environment or other factors that affect the data. Figure 5.15 showing movement of centroids in keystroke dataset at different timestamps.

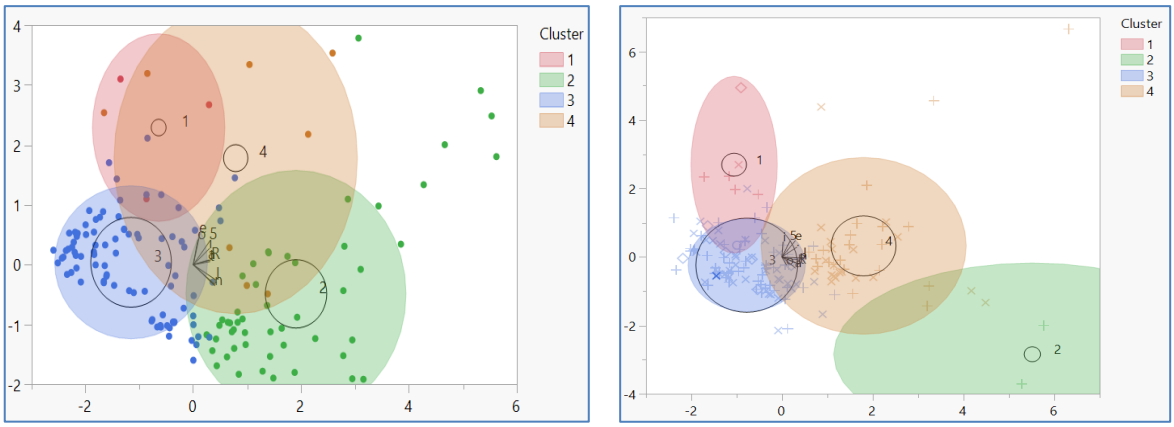


Figure 5.15 Movement of clusters in keystroke dataset, time step 300 (left) and time step 600 (right).

It is evident from the figures that the centroids are non-stationary and moving, this is because the position of the centroids depends on the data points that are assigned to the cluster, and as new data points are added or removed, the position of the centroids may shift. In keystroke dynamics, this change gradual change in the position of centroids is due to gradual change in

the typing speed of the users, as they learn to type passwords faster and more accurate over time. The next section demonstrates the experiments and significant findings of PSDSL applied on the keystroke datasets.

### 5.15.3 Experimental Setup

Considering the task of predicting 1 user out of 4, the PSDSL and SCARGC algorithms are evaluated under Static (no pseudo-labelling) and Benchmark scenarios (complete labelled). As each user types the password 50 time per session, the classifiers are trained on first 150 examples and incrementally evaluated on data from remaining 7 sessions.

All the experiments are evaluated in terms of time consumption and predictive performance. Processing time is measured in seconds and is based on the CPU time used for training and testing. All the experiments were performed on machines with Core i7 @ 3.4 GHz, 4 GB of RAM. The details of parameters used in the experiments for these existing EVL approaches are provided in Table 5.12.

Table 5.12 Algorithms and parameters used in the experiments.

| **Algorithm** | **Description** |
|---|---|
| Static | The classifier is not updated after it is trained with the first examples in the data streams. i.e. no pseudo-labelling is applied. |
| SCARGC | Applied KNN base classifier and applying the parameters suggested by Souza et al. [31] |
| Benchmark | Both PSDSL and SCARGC classifier are evaluated on 100% labelled examples. |
| PSDSL | HDWM Classifier, envelop clustering, nearest neighbours =3, purity threshold = 0.95, Pool Size = 150 |

## 5.16 Significant Findings

As the PSDSL does not apply a CGC approach on MOA streams and switches to a self-learning state, this improvement is due to the switching mechanism of heterogeneous base classifiers. As shown in Figure 5.16, the PSDSL achieved the highest predictive performance as compared to the SCARGC (81.67%) and Static (49%) is the accuracy when no EVL is handled by using pseudo labels. The prediction time as shown in Figure 5.17 are lower than the Benchmark and Static but slightly higher than the SCARGC.

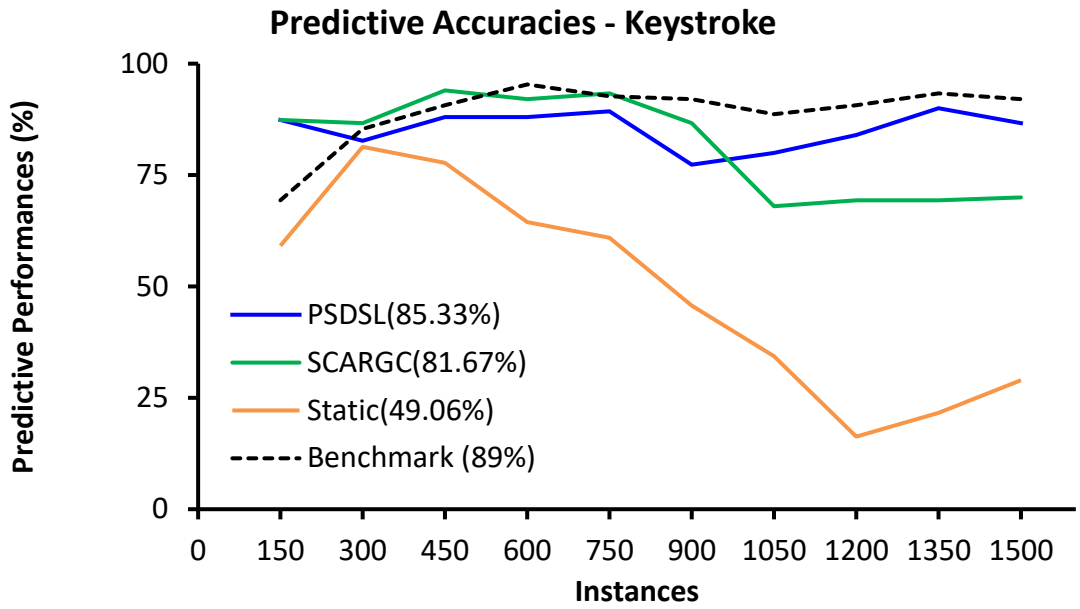


Figure 5.16 Prediction accuracies keystroke dataset, comparing PSDSL, SCARGC, Static and Benchmark.

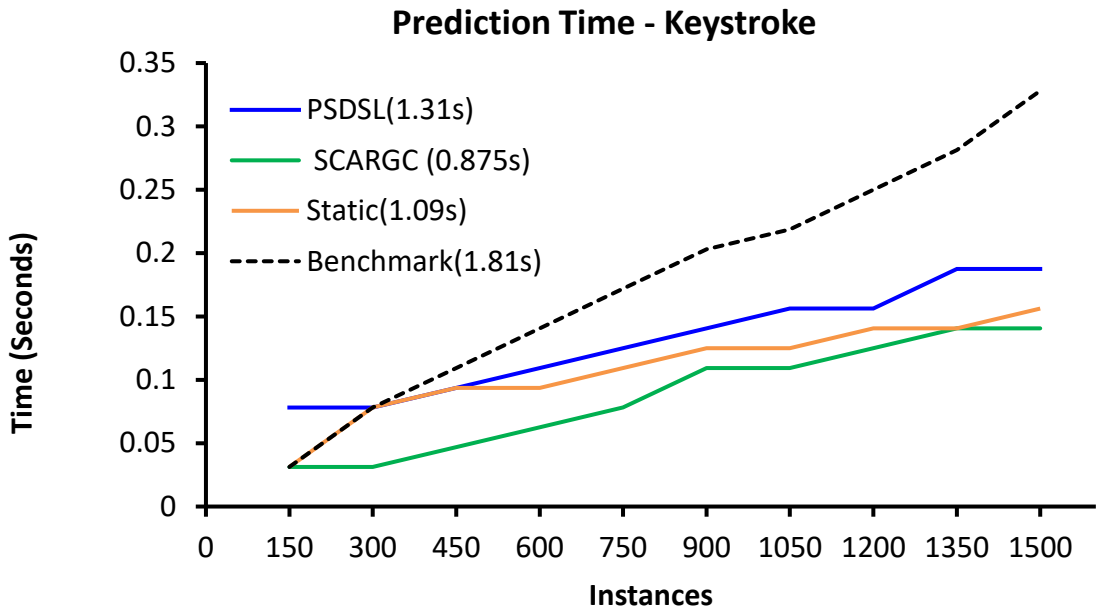


Figure 5.17 Comparison of prediction time PSDSL, SCARGC, Static and Benchmark.

In SCARGC algorithm the value of 'k' which is the initial number of clusters and it value is manually chosen, Figure 5.18 is showing the predictive accuracies of SCARGC on keystroke dataset by applying different values of this parameter 'k'. In this experiment 'k=4, 6, 8 and 11 have been applied, the results shows that the predictive accuracies have changed significantly (41.1% to 81.6%).

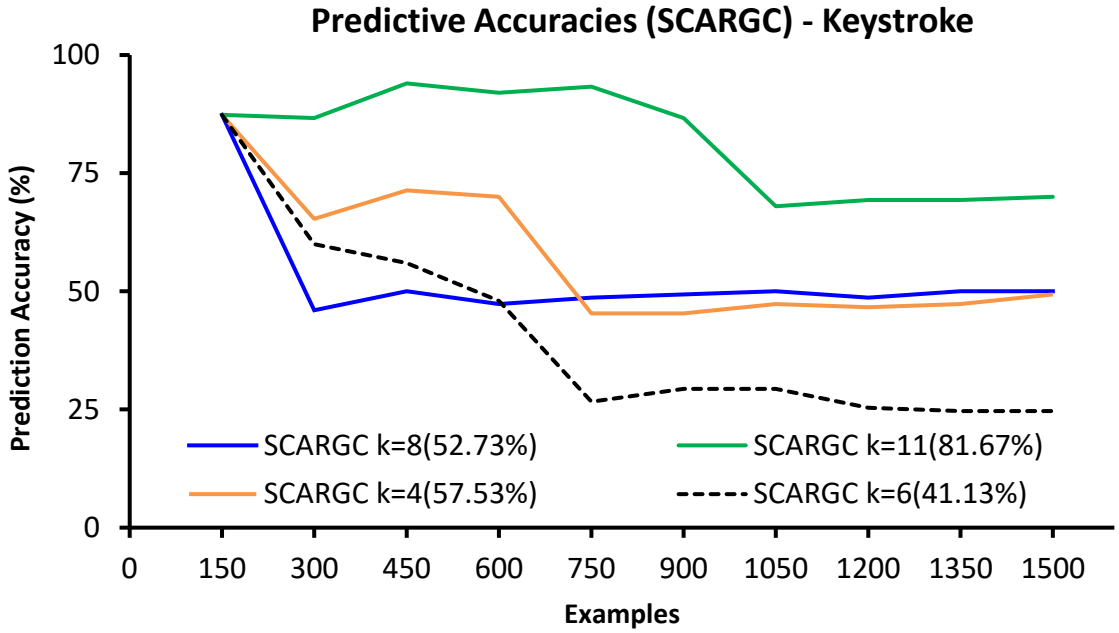


Figure 5.18 Predictive performance of SCARGC by applying different values of 'k'.

Figure 5.19 shows the results of experiment performed on keystroke dataset [137]. The PSDSL detects and resolves the cluster labelling issue arises in the micro-clusters in a situation when there is a dispute in micro-cluster labelling.

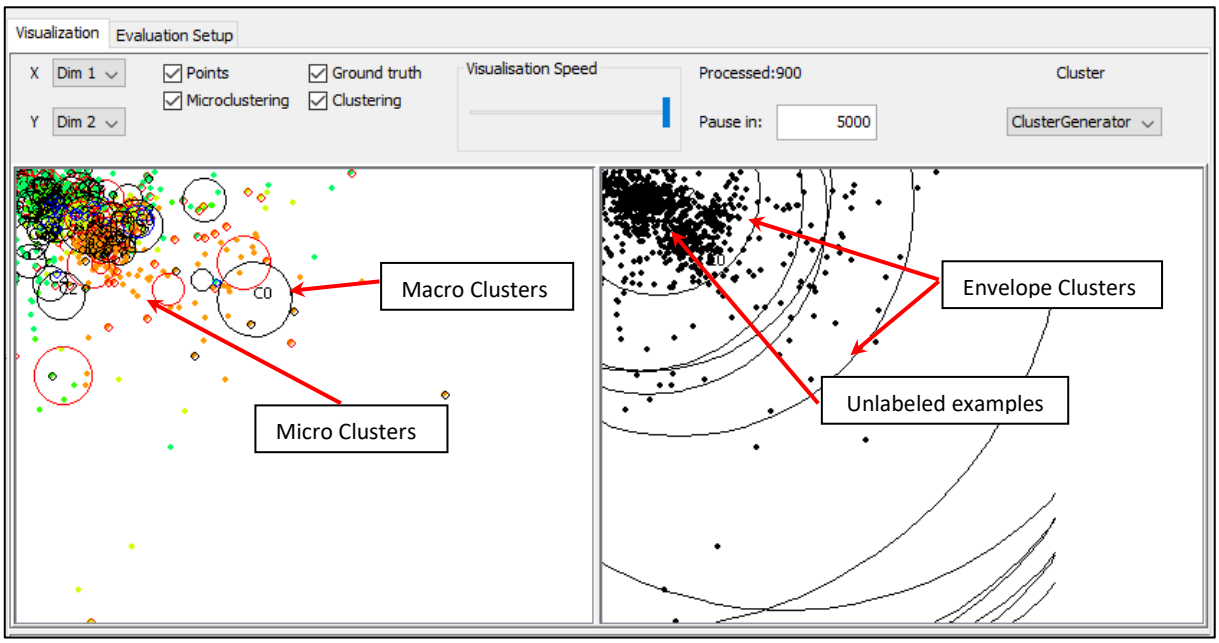


Figure 5.19 Visualisation of envelope, macro and micro-clusters in Keystroke dataset [137].

To overcome these issues, PSDSL transforms the centroids' information of micro-clusters into micro-instances, these instances generate clusters using a centroid based clustering

algorithm such as K-Means to generate cluster called envelope clusters. Finally, the nearest envelope clusters assist the conflicted micro-clusters in assigning their labels. The visualisation shows envelope clusters in the right pan, and the micro and macro clusters and shown in the right pan. A total 55 thousand examples were trained out of which only 150 initial examples were labelled followed by complete unlabelled examples which are shown in the right pan. The envelope clusters increased the prediction accuracy.

Figure 5.20 shows the predictive accuracy plots for MOA Streams in which no drift is induced. The results show that PSDSL performed significantly better than SCARGC on all the MOA Streams when there are no concept drifts.

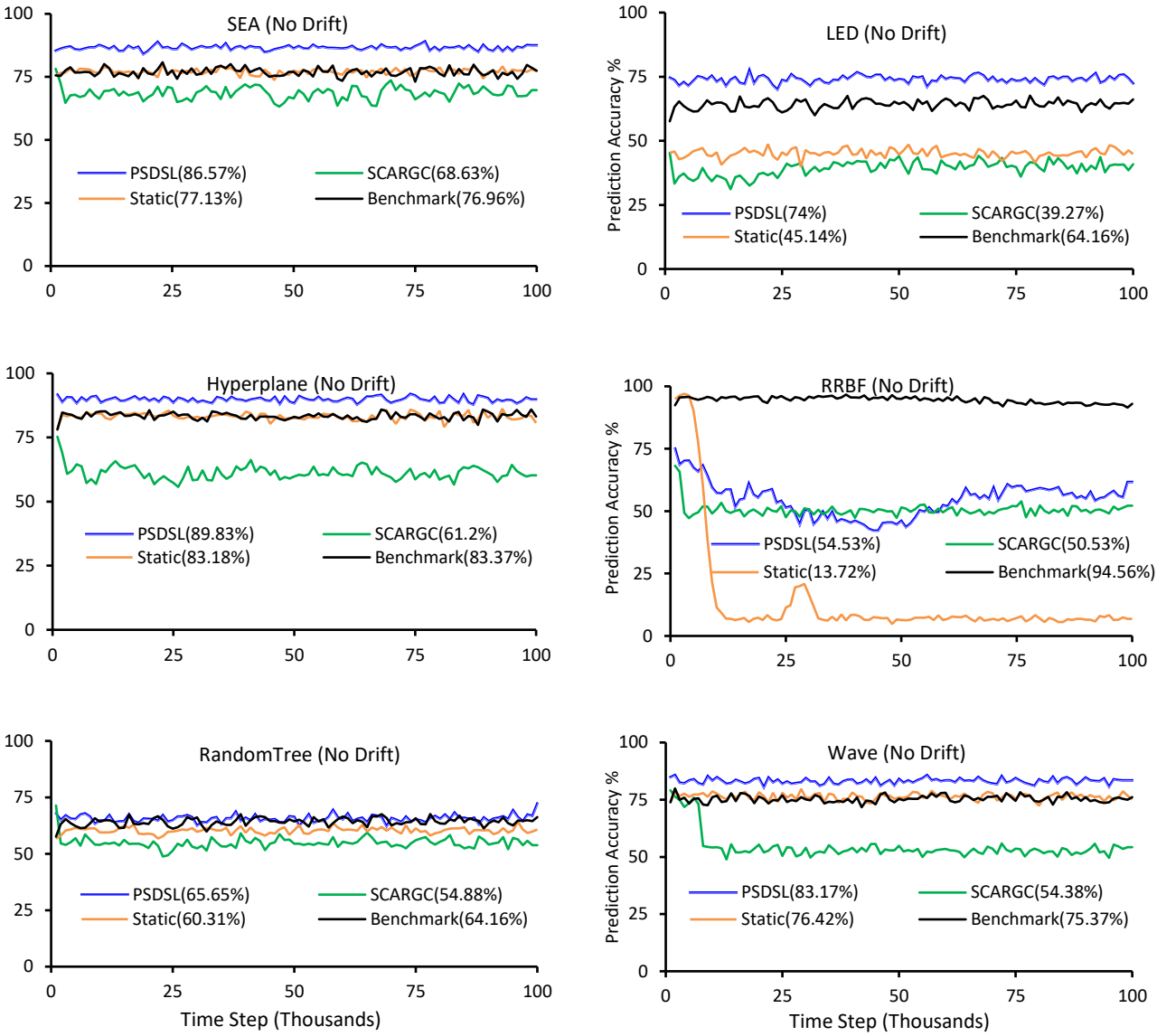


Figure 5.20 Predictive accuracy plots for MOA Streams (No drift).

Figure 5.21 shows the predictive accuracy plots for MOA Streams in which artificial drift is induced. The results show that in EVL, when the CGC fails, restoring from the concept drift is challenging due to unavailability of true class labels. In SEA (Abrupt) and RandomTree

(Recurring Drift) streams, all the algorithms restored learning after the sudden drifts. However, the graphs show that, before the first and after the last drift the PSDSL predictive performance is higher than the competing algorithms. This demonstrates that under EVL conditions, PSDSL adapted to the abrupt as well as recurring drifts better than other algorithms. On LED which is a multi-class problem, and Hyperplane which contains incremental drifts, none of the approaches adapted to the drifts in these two streams. Overall, PSDSL performed better than other approaches on drift induced MOA streams.

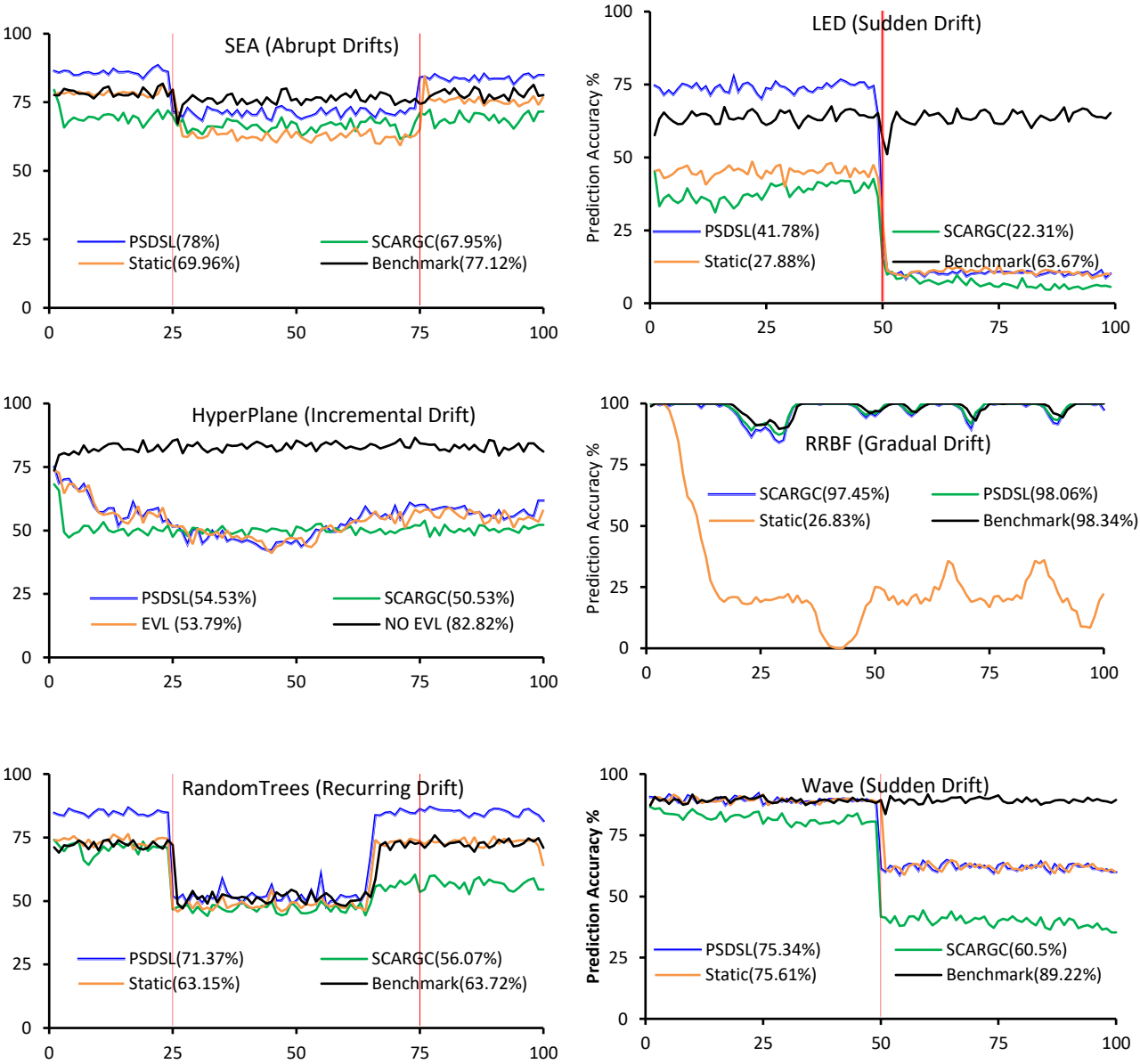


Figure 5.21 Predictive Accuracy Plots for MOA Streams (Artificial drift induced), red vertical lines representing the actual location of abrupt drifts.

SCARGC performed best in non-stationary datasets, however its predictive performance did not improve when applied to MOA data streams. To further investigate the cause(s) of this failure, a randomisation analysis was made and is presented in Section 5.16.1.

### 5.16.1 Analysis of Randomisation

This experiment analyses the sequence of training data and its influence on prediction accuracies for CGC algorithms. In data streams, continuous data arrives at high speed and there is practically no control over the sequence of training data presented to the learning algorithms. Randomisation is thus different to noise, as it is not a random displacement of examples, but a random order in which data instances are presented to the learning algorithm. In case of noise, the data needed to be cleaned before PSDSL deals with drifts. In this section, (RQ4) is addressed, which is "are existing ILNSE approaches always successful when applied to different problems and why does this approach sometimes fail?". The benchmark non-stationary datasets [31] [125] are randomised by shuffling the order of examples in the datasets.

Figure 5.22 shows a plot of Four Class Rotating (4CR) [125] dataset. The plot '4CR original dataset' on the left shows initial 1000 examples, and on the right '4CR randomised' are initial 1000 instances after shuffling 144k instances in the dataset. The centroids in the dataset are gradually rotating, therefore the examples which are located above the 1000 appeared in the first batch and resulted in a noise effect. The change in the order of examples resulted in the loss of cluster boundaries. This is the scenario in real-time data streams, i.e. no control over the order of examples.

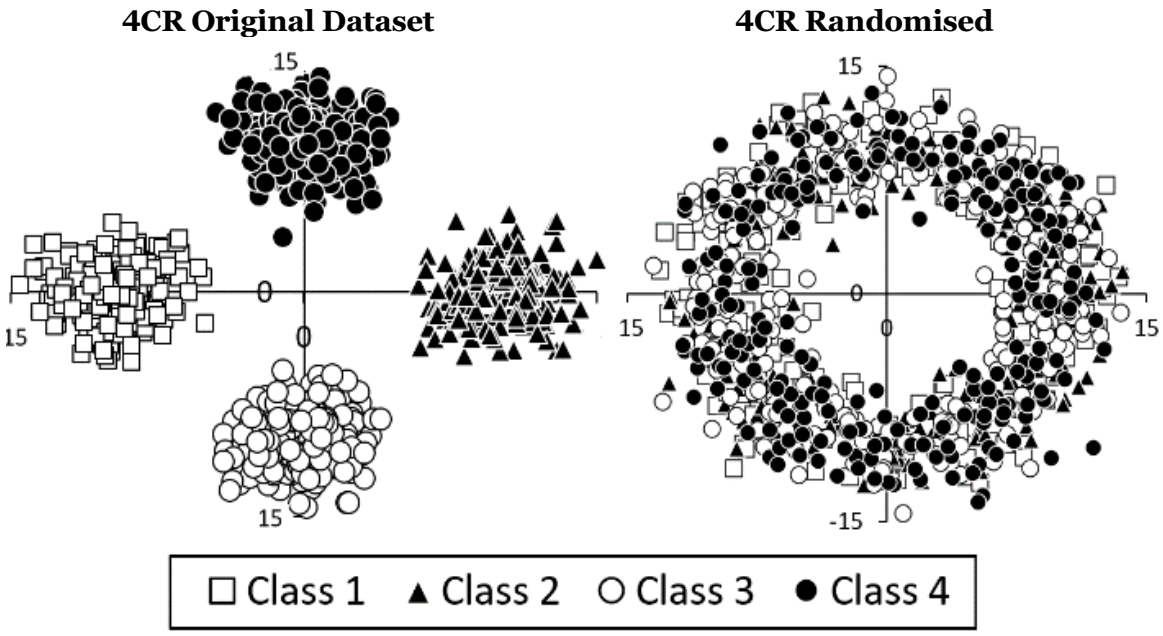


Figure 5.22 Plot for initial 1000 instances of 4CR dataset versus randomised 4CR dataset.

The results in Table 5.13 show the prediction accuracies achieved by the SCARGC and PSDSL algorithm on original and randomised datasets. The results show that SCARGC had a significant drop in average prediction accuracy by 35.3% on randomised datasets, whereas PSDSL only dropped by 20.9%.

Table 5.13 Predictive accuracy (in %) on original and randomised benchmark datasets.

| Datasets | SCARGC Original | SCARGC Randomised | PSDSL Original | PSDSL Randomised |
|---|---|---|---|---|
| 1CDT | 99.8 | 88.3 | 99.6 | 99.2 |
| 1CHT | 99.3 | 88.4 | 99.1 | 98.2 |
| 1CSurr | 94.4 | 59.4 | 94.5 | 66.7 |
| 2CDT | 90.9 | 59.9 | 90.7 | 60.3 |
| 2CHT | 85.0 | 59.7 | 85.8 | 60.1 |
| 4CE1CF | 94.1 | 92.9 | 94.7 | 95.31 |
| 4CR | 99.9 | 24.9 | 99.9 | 25.1 |
| 4CRE-V2 | 91.9 | 24.3 | 91.8 | 25.0 |
| 5CVT | 90.1 | 38.9 | 84.9 | 66.9 |
| FG_2C_2D | 95.2 | 43.2 | 64.9 | 74.3 |
| GEARS | 95.9 | 95.4 | 95.9 | 95.6 |
| MG_2C_2D | 92.7 | 50.1 | 64.5 | 59.6 |
| UG_2C_2D | 95.6 | 53.4 | 95.6 | 58.2 |
| UG_2C_3D | 94.8 | 51.0 | 94.8 | 70.5 |
| UG_2C_5D | 91.0 | 51.1 | 91.0 | 79.8 |
| **Average** | **94.0** | **58.7** | **89.8** | **68.9** |

### 5.16.2 Analysis of Switching Mechanism in PSDSL

To address (RQ4), what strategy should be adopted if the CGC or self-learning approaches fail? PSDSL is made capable of intelligently switching learning states 'CGC with K-Means, micro-clusters, or self-learning. For this set of experiments, the first batch of the data stream is 100% labelled, followed by 5% labelled instances in each batch of size 1000.

As shown in Table 5.16 the switching mode in PSDSL is dependent on {F1-P, F1-R and purity} of K-Means and micro-clusters. Whichever is higher, it adapts the learning mode accordingly. For values lower than threshold 'ρ' such as in randomised datasets or MOA Streams, it switches to self-learning. Further, it monitors the performance of pseudo-labelling. In the case that pseudo-labelling does not improve the predictive performance on initial labelled data, PSDSL suspends the pseudo-labelling.

Table 5.14 shows that overall, 7.2% prediction accuracy of classifier (NB) has been improved by applying self-learning. Table 5.15 shows improvement (23.9%) when active switching of classifier (NB) and clusters (K-Means) was applied.

As shown in Table 5.16 the switching mode in PSDSL is dependent on {F1-P, F1-R and purity} of K-Means and micro-clusters. Whichever is higher, it adapts the learning mode accordingly. For values lower than threshold 'ρ' such as in randomised datasets or MOA Streams, it switches to self-learning. Further, it monitors the performance of pseudo-labelling. In the case that pseudo-labelling does not improve the predictive performance on initial labelled data, PSDSL suspends the pseudo-labelling.

Table 5.14 Prediction accuracies (%) by applying self-learning.

| Datasets | No Pre-labelling | Pre-labelling | Gain/Loss |
|---|---|---|---|
| UG_2C_2D | 52.51 | 68.35 | +15.84 |
| UG_2C_3D | 62.82 | 68.28 | +5.46 |
| 1CSurr | 63.94 | 73.32 | +9.38 |
| 4CR | 19.77 | 34.07 | +14.30 |
| 4CRE-V2 | 24.08 | 32.89 | +8.81 |
| STAGGER | 52.80 | 52.92 | +0.11 |
| RandomTree | 62.78 | 62.87 | +0.09 |
| LED | 44.38 | 48.65 | +4.26 |
| Hyperplane | 61.81 | 70.04 | +8.23 |
| SEA_Mixed | 82.40 | 84.97 | +2.56 |
| RandomRBF | 25.66 | 39.25 | +13.58 |
| Sensor | 14.87 | 18.26 | +3.38 |
| Covtype | 63.19 | 70.50 | +7.30 |
| **Average (%)** | **48.5** | **55.7** | **7.2** |

Table 5.15 Prediction accuracies (%) self by applying active switching of classifiers and clusters as prediction method.

| Datasets | No Pre-labelling | Pre-labelling | Gain/Loss |
|---|---|---|---|
| UG_2C_2D | 52.51 | 94.20 | +41.69 |
| UG_2C_3D | 62.82 | 92.66 | +29.84 |
| 1CSurr | 63.94 | 85.97 | +22.02 |
| 4CR | 19.77 | 95.76 | +75.99 |
| 4CRE-V2 | 24.08 | 84.76 | +60.68 |
| STAGGER | 52.80 | 55.20 | +2.39 |
| RandomTree | 62.78 | 66.10 | +3.32 |
| LED | 44.38 | 48.67 | +4.29 |
| Hyperplane | 61.81 | 68.94 | +7.12 |
| SEA_Mixed | 82.40 | 82.59 | +0.18 |
| RandomRBF | 25.66 | 74.764 | +49.09 |
| Sensor | 14.87 | 22.12 | +7.24 |
| Covtype | 63.19 | 69.47 | +6.28 |
| **Average (%)** | **45.5** | **72.4** | **23.9** |

Table 5.16 PSDSL purity and switching mode.

| Non-Stationary Datasets | $\mu_{ACC\%}$ | $\Phi$ Purity | $\mu$ Purity | LM |
|---|---|---|---|---|
| 1CDT | 100 | **0.97** ↑ | 0.91 | C |
| 1CHT | 100 | 0.94 | **0.95**↑ | M |
| 1CSurr | 100 | 0.97 | **0.98**↑ | M |
| 2CDT | 100 | **1.00** ↑ | 0.98 | C |
| 2CHT | 100 | **0.97** ↑ | 0.92 | C |
| 4CE1CF | 100 | **0.89** ↑ | 0.86 | C |
| 4CR | 100 | 1.00 | 1.00 | C |
| 4CRE-V2 | 100 | **1.00** ↑ | 0.97 | C |
| 5CVT | 100 | 0.90 | **0.96**↑ | M |
| FG_2C_2D | 100 | **0.92** ↑ | 0.90 | C |
| GEARS_2C_2D | 100 | **0.95** ↑ | 0.92 | C |

| Non-Stationary Datasets | $\mu_{ACC\%}$ | $\Phi$ Purity | $\mu$ Purity | LM |
|---|---|---|---|---|
| MG_2C_2D | 100 | 1.00 | 1.00 | C |
| UG_2C_2D | 100 | 1.00 | 1.00 | C |
| UG_2C_3D | 100 | 1.00 | 1.00 | C |
| UG_2C_5D | 100 | 1.00 | 1.00 | C |
| MG_2C_2D | 100 | 1.00 | 1.00 | C |
| SEA (Sudden Drift) | 86.6 | 0.75 | 0.76 | S |
| LED (Sudden Drift) | 47.1 | 0.00 | 0.35 | S |
| Wave (Sudden) | 85.3 | 0.00 | 0.64 | S |
| RRBF (Gradual Drift) | 100 | 1.00 | 1.00 | C |
| Hyperplane (Incremental) | 76.2 | 0.67 | 0.56 | S |
| RandomTree (Recurring) | 79.1 | **1.00** ↑ | 0.62 | S |
| SEA (No Drift) | 86.3 | 0.70 | 0.80 | S |
| LED (No Drift) | 47.1 | 0.00 | 0.35 | S |
| Hyperplane ((No Drift) | 79.2 | **1.00** ↑ | 0.58 | S |
| RandomTree (No Drift) | 92.4 | **0.99** ↑ | 0.54 | S |
| Wave (No Drift) | 76.7 | 0.00 | 0.60 | S |
| KEYSTROKE | 99.3 | 0.34 | **0.93**↑ | M |

$\mu_{Purity}$= CluStream micro-clustering purities, $\Phi_{Purity}$= K-Means purities, $\mu$ACC%= pseudo-labelling Accuracy for $\mu$, LM= Learning modes {C = CGC using K-Means, M = micro-clustering, S= Self-learning}

As shown in Figure 5.23, a GUI has been developed for visually outputs the current state of the switching mode in PSDSL.

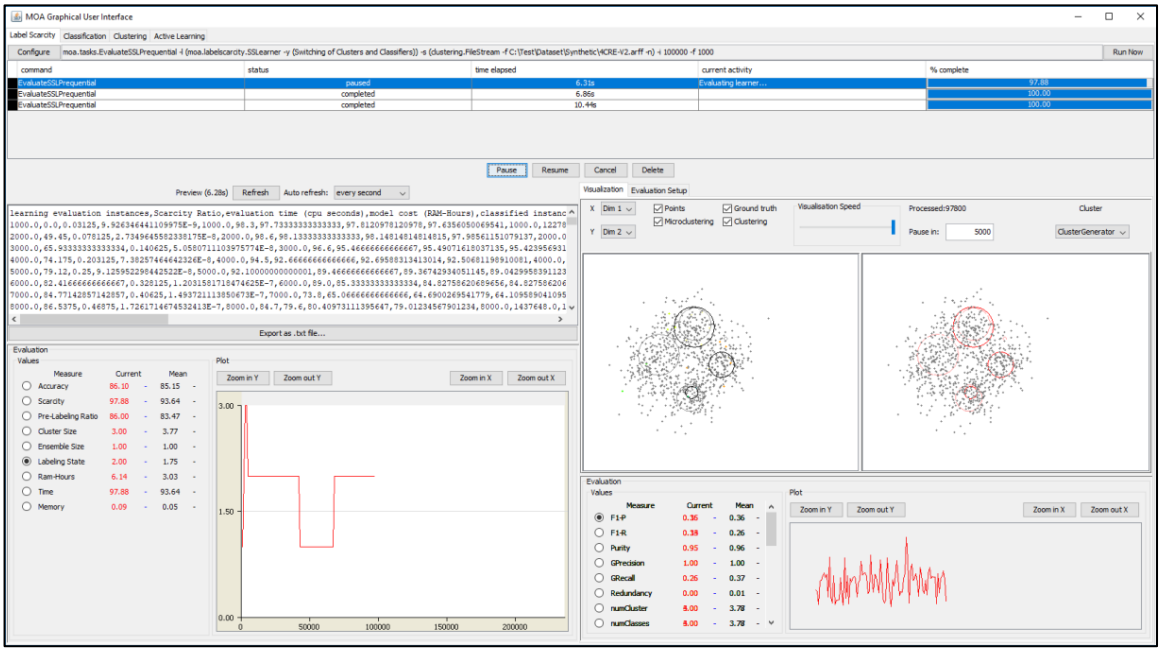


Figure 5.23 GUI for switching learning states.

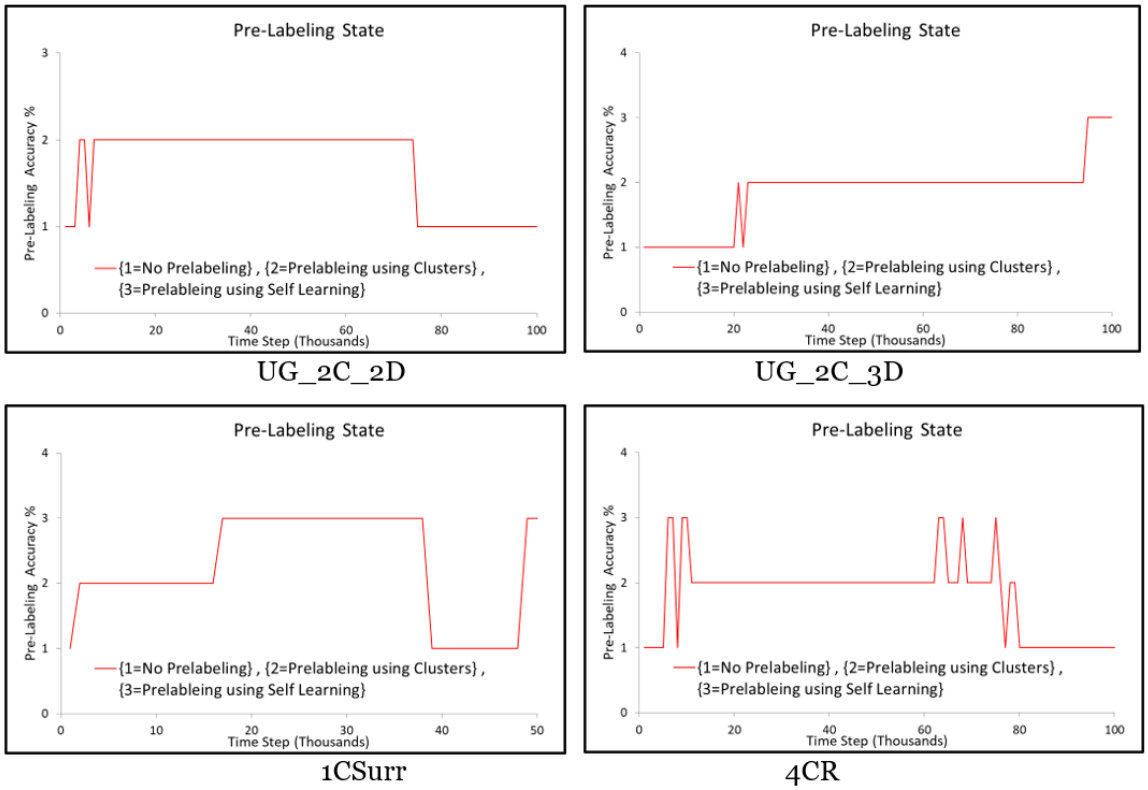


Figure 5.24 Switching states of Non-Stationary Datasets, 1=No pre-labelling, 2= pre-labelling using clusters and 3= pre-labelling using classifiers as self-learning.

Figure 5.24 showing switching states for non-stationary datasets. It is evident from the plots that the PDSL, switches the learning strategies based on conditions and on different times steps the pseudo-labelling was terminated as resumed afterwards because it was not beneficial at that specific period of time.

### 5.16.3 Analysis of Pseudo-labelling without Switching

This experiment determines the sensitivity analysis of switching mechanism. The effect of pseudo-labelling and without pseudo-labelling has been analysed on different characteristics of datasets, which includes numerical, categorical, and mixed attributes and normalized data has been shown in Table 5.17.

It is evident from the results that in some data streams the pseudo-labelling reduced the overall prediction accuracies, therefore a mechanism has been added to intelligently determine and stop the pre-labelling strategy if it is not beneficial.

Table 5.17 PSDSL Analysis of Pseudo-labelling without Switching.

| Datasets | Types of attributes | No pseudo-labelling (1) | Pseudo-labelling (2) | Gain/Loss (2) – (1) |
|---|---|---|---|---|
| UG_2C_2D | N, Norm | 93.985 | 95.653 | 1.668 |
| UG_2C_3D | | 92.957 | 93.678 | 0.721 |
| 1CSurr | | 87.518 | 92.093 | 4.575 |
| 4CR | | 97.015 | 99.870 | 2.855 |
| 4CRE-V2 | | 86.038 | 89.857 | 3.819 |
| Stagger | C | 76.708 | 72.450 | **-4.258** |
| RandomTree | C and N | 70.5 | 67.858 | **-2.642** |
| LED | C | 44.979 | 36.445 | **-8.534** |
| Hyperplane | N | 64.286 | 71.007 | 6.721 |
| SEA_Mixed | C | 81.835 | 79.956 | **-1.879** |
| RandomRBF | N | 21.23 | 80.167 | 58.937 |
| Sensor | N | 15.095 | 22.788 | 7.693 |
| Covtype | C and N, Norm | 69.206 | 71.930 | 2.724 |

N = Numerical, C= Categorical, Norm = Normalised

## 5.17 Hyperparameter Tuning

This section presents the analysis carried out to address (RQ3): Does this approach depend on parameters that require manual tuning by the users before inducing the training models? As shown in Table 5.18,

Table 5.18 PSDSL auto-tuned 'k' and learning mode.

| Non-Stationary Datasets | No. of Classes in datasets | SCARGC Manual 'k' | PSDSL Auto-tune 'k' | SCARGC ACC% | PSDSL ACC% |
|---|---|---|---|---|---|
| 1CDT | 2 | = 2 | 2 | **99.75** | 99.67 |
| 1CHT | 2 | = 2 | 2 | **99.25** | 99.09 |
| 1CSurr | 2 | **≠ 4** | 4 | 94.35 | **94.51** |
| 2CDT | 2 | = 2 | 2 | **90.92** | 90.74 |
| 2CHT | 2 | = 2 | 2 | 85.02 | **85.82** |
| 4CE1CF | 4 | ≠ 5 | 5 | 94.08 | **94.09** |
| 4CR | 4 | = 4 | 4 | 99.95 | **99.97** |
| 4CRE-V2 | 4 | = 4 | 4 | **91.9** | 91.88 |
| 5CVT | 4 | ≠ 5 | 5 | **90.15** | 84.99 |
| FG_2C_2D | 5 | **≠ 4** | ≠ 2 | **95.16** | 64.94 |
| GEARS_2C_2D | 2 | = 2 | 2 | 95.89 | **95.93** |
| MG_2C_2D | 2 | **≠ 4** | **≠ 2** | **92.71** | 64.52 |
| UG_2C_2D | 2 | = 2 | 2 | 95.56 | **95.57** |
| UG_2C_3D | 2 | = 2 | 2 | 94.77 | **94.80** |
| UG_2C_5D | 2 | = 2 | 2 | 90.98 | 91.05 |
| KEYSTROKE | 4 | ≠ 12 | ***4** | **87.72** | 85.33 |

SCARGC applies k=4 for (1CSurr) which is a binary class problem; similarly, SCARGC applies k=4 for (FG_2C_2D) (MG_2C_2D) which contains 5 and 2 classes in the datasets respectively. Furthermore, the real-world dataset 'keystroke' contains 4-classes, but SCARGC applies k=12 (number of centroids). In SCARGC these values need to be manually chosen by the user to achieve the best results. Contrary to this, PSDSL automatically tuned the best values for the 'k'. As evident in the Table, in most of the datasets, PSDSL predicted similar values for 'k' as SCARGC. However, the difference is that the parameter 'k' was sent manually in SCARGC, while PSDSL automatically adapts and parameter 'k' to optimise the classification results over time.

## 5.18 Parameter Sensitivity Analysis

The influence of the PSDSL parameters pool size (θ) and number of labelled examples |T| is analysed against the prediction accuracy. Figure 5.25 shows the prediction accuracy in % on different values of θ from 300 to 1500 and |T | from 50 to 1000. As it is clear from the plot, increasing the pool size increases the prediction accuracy; however, |T| has no significant effect on the accuracy.

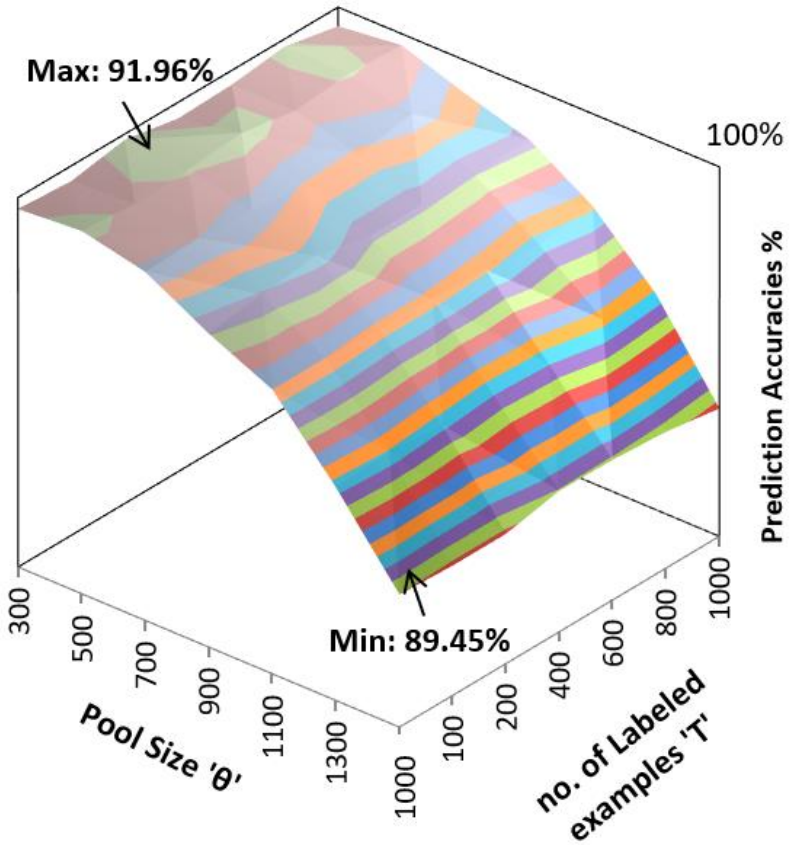


Figure 5.25 Prediction accuracy for 4CRE-V2 dataset changing values of pool size θ and size of initial labelled data 'T'.

## 5.19 Development of online SSL framework for data streams

An online SSL framework is developed in MOA which will enable the DSM researchers to implement online SSL for DSM, mainly in the areas of EVL and NSE. The option was to visually monitor the real-time prediction results and evaluations for both clustering and classification algorithms. The clustering results appears in the right part of the GUI and the classification results appears in the left part of the GUI. The clustering evaluations includes an online evaluation of clusters such as F1, Precision, Recall, purity etc. Figure 5.26 shows SSLHoldout evaluation in which two mutually exclusive subsets are created for training and testing. Figure 5.27 shows the SSL Learner developed in MOA. The option is added to apply an ensemble of classifiers and clusters, with an option to define the drift handling method.

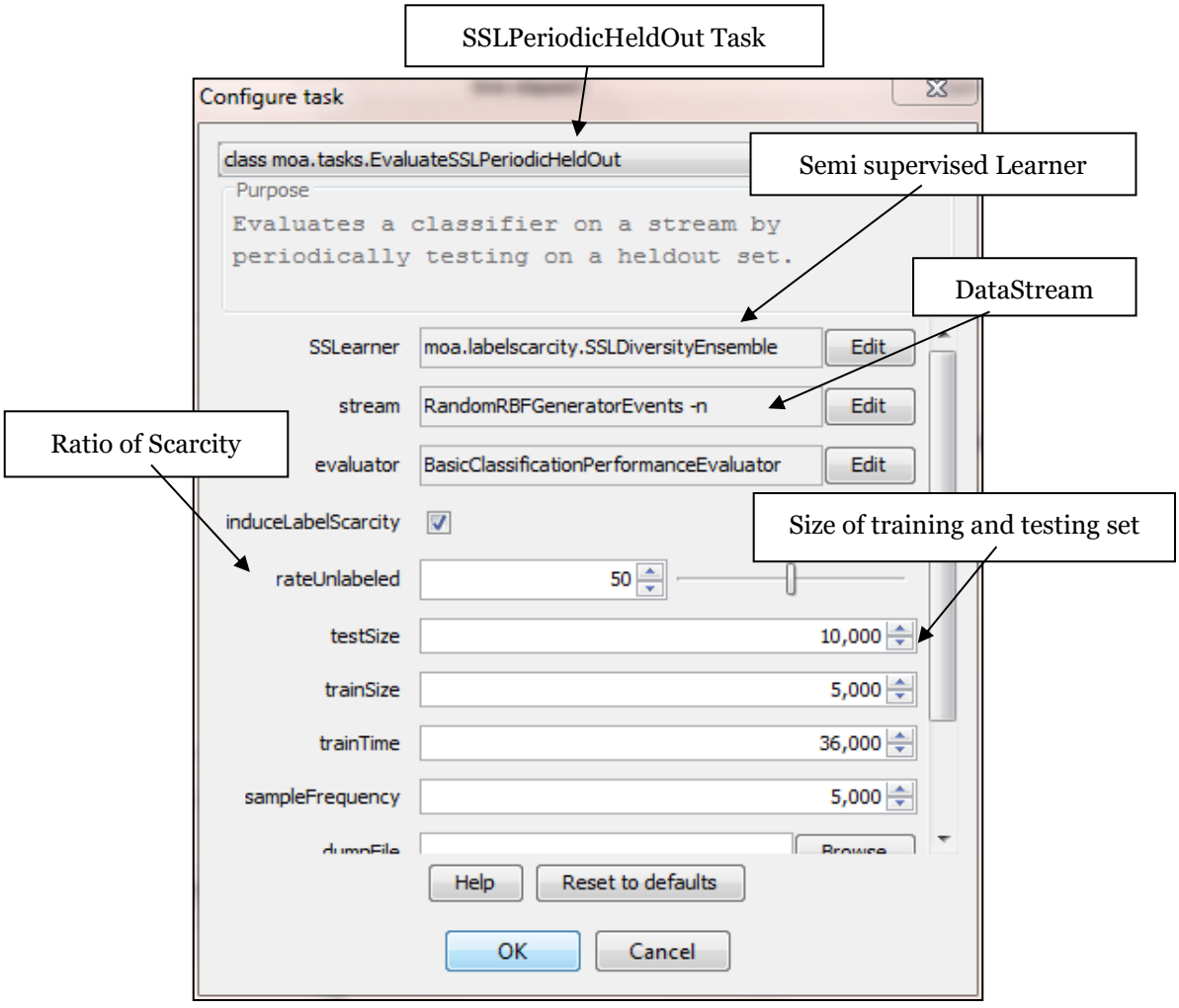


Figure 5.26 GUI for SSL Periodic Holdout Evaluation Task developed in MOA.

The evaluation of classifiers includes average prediction accuracies, class label scarcity, time and memory consumption etc. In the SSL framework, the pseudo-labelling accuracy, micro-macro clustering and the ground truth clusters along with switching of learning strategies are visually displayed. PSDSL was evaluated on non-stationary datasets, synthetic data-

streams, and real-world datasets. The approach has shown promising results on randomised datasets as well as on synthetic data-streams, as compared with state-of-the-art approaches. This is the first large-scale study on an adaptive extreme verification approach that supports automatic parameter tuning and intelligent switching of pseudo-labelling strategy, thus reducing the dependency of machine learning on human input. To measure and analyse the performance of the clustering and classification models two SSL learning tasks, i.e. SSLPrequential and Holdout have been developed.

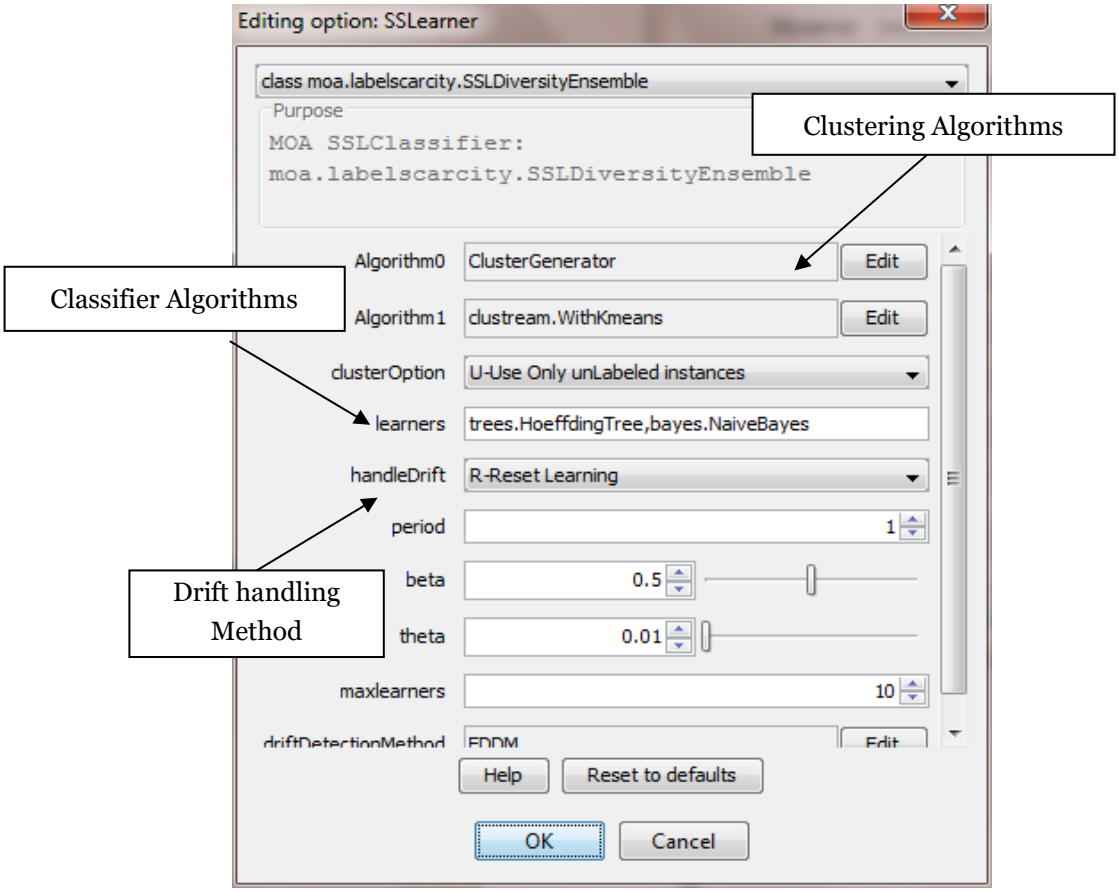


Figure 5.27 A developed Semi-Supervised Diversity Ensemble (SSLearner).

Initially, both the supervised and unsupervised models are trained on labelled examples after randomly eliminating the true class labels from a certain ratio from the training set. An option is created to predict the pseudo-labels using supervised models for unlabelled examples, and the supervised models are re-trained on these new labels. The accuracies of pseudo-labelling, ratio of scarcity and the evaluation of both supervised and unsupervised models are updated incrementally. Similarly, the error of the model is computed for the sequence of each example in the SSL Prequential task.

## 5.20 Critical Evaluation of Research and Scientific Contributions

This section provides a critical evaluation of the entire research journey and outcomes, which involves an assessment of the research process, and evaluating the answers to the research questions about the robustness and alignment with the intended contributions. Evaluates the methodology, data analysis, and interpretation to provide a comprehensive understanding of how the research outcomes contribute to advancing knowledge in the data stream mining domain.

### 5.20.1 Evaluation of Research Questions

The research addresses several key questions, providing insights and contributions to knowledge. In (RQ1), the focus is on methods enabling data stream mining algorithms to learn efficiently from limited labelled data under non-stationary conditions. Various approaches, such as cluster-guided classification, self-learning using classifiers, and micro-clustering, are explored, each presenting its challenges and trade-offs. (RQ2) delves into the impact of ensemble diversity on predictive performance in non-stationary environments. The lack of evidence in the literature regarding the suitability of prediction models post-concept drift prompts a critical examination of the types of models applicable at different times. (RQ3) revolves around methodologies for automatic hyperparameter discovery and adjustment, specifically addressing the dependency on parameters tuned by human analysts in existing ILNSE approaches. Finally, (RQ4) considers the success of existing approaches across diverse problems and proposes an active switching mechanism when an EVL approach fails. Additionally, strategies for preventing incorrect propagation of class labels during concept drifts are suggested.

### 5.20.2 Evaluation of Scientific Contributions

This research makes noteworthy contributions to the field. Through empirical evaluation, PSDSL exhibits higher prediction accuracy compared to existing approaches like SCARGC, COMPOSE, $\text{LEVEL}_{\text{IW}}$, and MClassification. The published work established the groundwork for addressing extreme verification latency in non-stationary environments. Additionally, the implementation of DDM and EDDM, the drift detection methods utilized in the HDWM algorithm has been published. These methods have been applied for change detection in the marine ecosystem. The heterogeneity of online ensembles in non-stationary environments, culminating in the development of the heterogeneous algorithm (HDWM) has also been published.

## 5.21 Discussion

As outlined in the methodology, PSDSL underwent a comparative analysis with established EVL approaches, evaluated without pseudo-labelling and benchmark settings, where 95% of class labels are removed. The evaluation used Kappa statistics and prequential testing, with the latter assessing accuracy incrementally, suitable for balanced classes. Kappa statistics provided a sensitive measure in streaming classifiers, regardless of class balance

certainty. The nearest labelling approach faced challenges when clusters overlap. Envelope-clusters addressed this issue by detecting and resolving labels assigned to clusters.

The GUI for OSSL aimed to visualize and qualitatively analysing both clustering and classification models for a comprehensive evaluation. Requirements for OSSL were identified as, include processing one labelled example at a time, using limited time and memory, handling unlabelled examples in small batches with pseudo-label predictions, and being available for prediction at any time.

PSDSL's improved performance on MOA streams, particularly in self-learning states without the CGC approach and outperformed SCARGC which showed varying predictive accuracies on the 'keystroke' dataset with different values of the parameter 'k'. PSDSL addressed this issue by applying automated hypermeter tuning and resolved cluster labelling issues that were raised in micro-clusters when conflicting labels occurred.

## 5.22 Summary

This chapter suggested a novel approach that deals with the scarcity of class labels under NSEs. i.e. ILNSE. To address RQ1, a new algorithm PSDSL was made capable of intelligently selecting the best pseudo-labelling strategy based on the given problem domain. it could switch on the pseudo-labelling strategy, i.e. cluster-guided, self-learning or micro-clustering, and select whichever approach is beneficial, based on the characteristics of the data stream. To address RQ2, PSDSL was made capable of automatically choosing the best classifier from a pool of heterogeneous classifiers by applying the strategy from the HDWM classifier hence preserving the diversity of the ensemble classifier.

The PSDSL algorithm also introduced automated parameter tuning that aimed to address RQ3 for reducing the dependency of machine learning on human' input. PSDSL addressed the RQ4 i.e. what strategy should be adopted if one of the EVL approaches fails? by proposing a new concept of Envelope-Clustering which aims at resolving the conflict in assigning the class labels to the clusters in case of cluster overlaps.

Predictive performances of PSDSL were compared against existing EVL approaches namely SCARGC, $LEVEL_{IW}$, COMPOSE, and MClassification. Finally, to determine significant differences between algorithms the chapter presented experiments on non-stationary benchmark datasets, MOA data streams and real-world datasets. The results showed that PSDSL performed significantly better than SCARGC on most real-time data streams, including randomised data instances. Thus, the prediction performance of pseudo-labelling has been evaluated by automatically switching between self-labelling and cluster labelling based on the characteristics of the training instances. It was also discovered that SCARGC or COMPOSE performed well for certain datasets in which centroids are moving with a constant velocity. However, when SCARGC was evaluated after shuffling the training instances of the same datasets by changing the training orders, its predictive performance was significantly reduced.

Finally, it was found that, for SCARGC to achieve the best results in different datasets, the values of 'k' needed to be manually chosen, whereas, in contrast, PSDSL achieved similar predictive accuracies without the need for manual selection of the value of the parameter 'k'. This novel approach proposed in this research further paves the way for reducing the dependency of machine learning on human input which essentially liberates the process from this hard constraint, as a critical bottleneck, to enable mass-scale deployment of dynamically adaptive labelling of data instances in various emerging data streams.

# Chapter 6 Conclusions and Future Work

This research mainly focused on improving the DSM algorithms which assume the availability of labelled data, immediately or after some delay (verification latency), to update the accuracy of the classifier and at the same time predict under the condition of NSEs. In many real-world applications of online data stream mining, the data originates from different sources such as sensor devices, social media, business/financial transactions, etc. The data evolves over time, and therefore extracting worthwhile knowledge is hard to achieve in such NSEs. The underlying probability distributions of the data stream change over time, resulting in concept drifts.

Due to the streaming nature of the data and scarcity of class label, the problem was identified as online SSL. The literature is referring to this problem as ILNSE. The ILNSE addresses both EVL and NSEs simultaneously, for example, autonomous robots are initially trained on labelled data, and later they are asked to predict the class labels on unseen data. They are also sent to explore an unknown environment without the supervision of humans. The robots need to learn under NSEs and do so under the scarcity of true class labels (EVL). Another application in banking is credit card fraud detection. The actual class labels (fraud or non-fraud) of the transactions are not available to update the online prediction models until the user receives and reviews the monthly statement. Furthermore, the transactions of the credit cards contain concept drifts due to the customers' patterns of spending, which change seasonally and/or during holidays.

## 6.1 Summary of Findings

It was analysed that in data streams the instances arrive in a sequential order which is directly fed into the online learning models, thus storing, and referring to the previous data is not practical due to time limitations. The output of an adaptive classifier at every time step depends on instances the classifier has been trained on to-date. Hence, performance depends on the order of instances in the dataset. Existing benchmarks for non-stationary datasets are designed to evaluate CGC on EVL, by inducing gradual shifting to the clusters. CGC showed promising results due to the high purity of clusters; however, when the order of these datasets is randomised the CGC performance drops considerably. This supported the fact that the existing CGC approaches succeed only under certain conditions.

The results for all the non-stationary datasets provided by the authors of SCARGC were verified with our implementation. Furthermore, the visualisation of clusters and real-time evaluation of prediction results at predefined regular intervals were displayed using a graphical interface. Finally, PSDSL was empirically evaluated against standalone approaches namely COMPOSE, $\text{LEVEL}_{\text{IW}}$, SCARGC and MClassification on benchmarks

NSE datasets [31] MOA data streams and real-world datasets. It was concluded that the existing approaches such as SCARGC or COMPOSE perform well for certain datasets in which centroids are moving with a constant velocity.

However, when SCARGC was evaluated after shuffling the training instances of the same datasets by changing the training orders, its predictive performance was significantly reduced. The results showed that PSDSL performed significantly better than SCARGC on most real-time data streams, including randomised data instances. Thus, the prediction performance of pseudo-labelling has been evaluated by automatically switching between self-labelling and clusters labelling based on the characteristics of the training instances.

The PSDSL algorithm performed better than SCARGC for some non-stationary datasets when these were randomised. PSDSL was evaluated on artificially induced MOA streams and real-world data streams and the results showed significantly enhanced performance over SCARGC for most of the MOA streams. The sequence of training data has been analysed the sequence of training data and its influence on prediction accuracies on 15 benchmark non-stationary datasets. The results showed that SCARGC had a significant drop in average prediction accuracy by 35.3% on randomised datasets, whereas PSDSL performed significantly better than existing approaches.

Most work in DSM is concerned with updating the learning system so that it can quickly recover from concept drift, while little work has been dedicated to investigating what type of predictive model is most suitable at any given time. It was aimed to investigate the benefits of online model selection for predictive modelling in NSEs. To analyse the influence of diversity on predictive performance (RQ2), 'Static vs. Dynamic' and 'Heterogeneous vs. Homogeneous' classifiers were comprehensively studied.

### 6.1.1 Reasons for the Failure of Existing EVL Approaches

The existing EVL approaches such as CGC rely on the assumption that the data follows a normal or Gaussian distribution. This supports the clustering process by helping to generate distinct clusters. This assumption also makes CGC a more effective choice in class labels imputation for missing class labels. However, the normal (Gaussian) EVL approaches cannot hold for randomised datasets or for real-world data streams, as most such streams are unstructured and contain noise.

The choice of pseudo-labels in cluster labelling could be problematic because the pseudo-labels are predicted using the same learning model on which it was trained, and the same models are used to predictions. Furthermore, due to NSEs these labels could make the models less reliable over time due to concept drifts. In CGC, the labels from the nearest clusters are transferred, however, the algorithm does not implement a confidence measure approach to assure the quality and correctness of labels assigned to the clusters.

It was also analysed that EVL deals with unlabelled data more effectively when clustering is applied, however under NSEs when the clusters overlap, the results showed that existing micro-clustering is more beneficial. Micro-clustering approach is a computationally expensive task for data streams mining.

The key feature i.e. diversity of learning models has recently gained attention by the DSM research community. Even though ensembles have been developed to handle NSEs, the literature did not contain any deep study of why and how the diversity of ensembles can be helpful in EVL conditions. • it is difficult to determine which type of machine learning model would be best to use. There is no evidence in the literature that suggests why, and which type of prediction models are beneficial or which models should not be used right after the concept drift.

Existing ILNSE approaches were found to be heavily dependent on the parameters 'k' which needs to be tuned before building the training models (RQ3). Most of the existing CGC approaches require prior knowledge of the number of classes to generate the corresponding centroids. However, in most of the real-world streams, the prior knowledge of the number of classes is not available. The SCARGC and COMPOSE were evaluated with different arbitrary values of 'k' and pool size θ, which negatively changed the prediction results.

### 6.1.2 Key Findings from Comparative Analysis

SCARGC, COMPOSE, MClassification and LEVEL$_{IW}$ addressing the ILNSE, but these are highly dependent on the parameters defined by the users as well as the problems these are applied to i.e. characteristic of data streams. In the new approach, PSDSL automatically decides the use of the best classifier from a pool of heterogeneous classifiers, it can switch on the pseudo-labelling strategy, i.e. cluster guided, self-learning or micro-clustering, and selects whichever approach is beneficial, based on the characteristics of the data stream.

To investigate the diversity of online classifier (RQ2) under ILNSE conditions, DWM[10] and WMA [11] algorithms were investigated for the 'dynamicity' and 'heterogeneity' factors of the online ensembles, which includes determining the reasons and mechanism for exclusion and inclusion the base learners from an ensemble. It was concluded that under the EVL conditions it is difficult to determine which type of machine learning algorithm would be best to use due to small amount of initial labelled data. There is no evidence found in the literature that suggests why and which type of prediction models are beneficial right after the concept drift. Most work under NSEs is concerned with updating the prediction models for adapting the concept drift, while little work has been dedicated to investigating the diversity of the online ensembles.

PSDSL has been evaluated on 'keystroke dynamics' dataset which is a collection of data that records the timing and pattern of keystrokes made by an individual while typing on a keyboard. The results shows that PSDSL successfully guided the cluster labelling process

after the concept drift in the absence of true class labels. PSDSL and SCARGC when evaluated on 10% of the labelled examples, PSDSL achieved higher prediction accuracy (85.3%) then SCARGC (81.6%). Without handling ILNSE approach, the predictive accuracy on this dataset is 49.0%. Furthermore, the predictive accuracy of SCARGC was found highly fluctuated in the range of (41.1% to 81.6%) based on the parameter 'k' (number of clusters), apart from that, the PSDSL automatically fine-tuned the best values of 'k.' for generating the centroids to guide the pre-labelling process.

### 6.1.3 How PSDSL and HDWM resolve the identified issues?

PSDSL applies HDWM classifier for self-learning, it is an online ensemble classifier that implements both an active and passive approach to simultaneously deal with gradual and abrupt concept drifts. HDWM is made heterogeneous to maintain different types to base classifiers and preserving the diversity. PSDSL and HDWM filled the gaps in the literature in the following ways.

- This research introduced a novel approach called Envelope-Clustering which is a centroid-based clustering approach for micro-clustering applied to resolve the conflict during the cluster labelling. Issues were identified in existing approaches in the cluster labelling phase when nearest neighbour algorithm is applied. It was identified that when one group of clusters crossing other groups (gradual drifts), the clusters may receive wrong labels. Either the moving clusters receive the labels of the stationary clusters, or the moving cluster transfers its label to the stationary clusters.
- The proposed switching mechanism of PSDSL automatically switches the pseudo-labelling strategy and the algorithm adapts to the learning mode accordingly. In the case that pseudo-labelling does not improve the predictive performance on initial labelled data, PSDSL suspends the pseudo-labelling.
- To address (RQ3), PSDSL was made capable to automatically tune the best values for the parameter, (number of clusters 'k)'. It was found that, for SCARGC to achieve the best results in different datasets, the values of 'k' needed to be manually chosen, whereas, in contrast, PSDSL achieved similar predictive accuracies without the need for manual selection of the value of the parameter 'k'. The novel approach proposed in this research further paves the way for reducing the dependency of machine learning on human input.
- HDWM automatically identifies which types of predictive models best suited to the situation encountered after concept drifts. HDWM's seeding mechanism and dynamic inclusion of new base learners benefiting the use of both forgetting and retaining the models and therefore adaptive to both sudden and gradual drifts.
- HDWM was designed in such a way that it made use of "seed" learners of different types to maintain ensemble diversity. This overcomes the problems of existing dynamic

ensembles that may undergo loss of diversity due to the exclusion of base learners. The algorithm was evaluated on artificial and real-world data streams against existing well-known approaches such as a heterogeneous WMA and a homogeneous DWM. The results showed that HDWM performed significantly better than WMA in under NSEs. Also, when recurring concept drifts were present, the predictive performance of HDWM showed an improvement over DWM.

- The seeding mechanism and dynamic inclusion of new base learners in the HDWM algorithms benefiting the use of both forgetting and retaining the models. HDWM achieved similar prediction accuracies as compared to the WMA and DWM but using a smaller size of ensemble and reduced CPU time. The algorithm also provided the independence of selecting the optimal base classifier in its ensemble depending on the problem.

- The development of HDWM algorithms revealed the ability to reduce human dependency on redefining the best type of predictive models for a particular problem. The algorithm exhibited responsive adaptation; dealing appropriately with changing environments in a shorter period to increase the reliability and predictive accuracy of the model. It was also found that heterogeneity was a key enabler for the improved accuracy achieved by HDWM. HDWM improved the predictive accuracy in the presence of different types of drifts, such as Gradual, Sudden and Recurring. It has been a key challenge in data stream mining, as some algorithms heavily rely on forgetting mechanisms while others retain previous learning.

## 6.2 Recommendations for Future Work

Future work in the realm of ILNSE involves the exploration of the integration of active learning techniques tailored for non-stationary environments. Design algorithms that intelligently select which instances to label, considering the evolving nature of the data distribution. The developed pseudo-labelling approach in PSDSL selects all the unlabelled examples in the pool and assigns predicted labels to them. It is also worth investigating transfer learning techniques that facilitate knowledge transfer between different stages of non-stationary data. Develop models capable of leveraging information gained from initial labels to accelerate learning on emerging patterns.

It is also recommended to consider applying transfer learning for ensemble diversity, which involves leveraging knowledge gained from pre-trained models to enhance the diversity of individual models within an ensemble. By incorporating insights from different sources or stages of learning, transfer learning aims to create a more varied set of base models within the ensemble. This diversity contributes to the ensemble's ability to capture and leverage a broader range of patterns and information, enhancing its overall effectiveness in handling complex tasks and adapting to different data distributions.

It is beneficial to investigate the HDWM performance on more diverse problems and in the presence of large number of attributes. Furthermore, investigate to reduce its dependency on human predefined parameters such as β, which is the weight to penalise the leaner models on each wrong prediction and the parameter ρ which is the period between base learner removal, creation and updating the weight.

The predictive performance of PSDSL is highly dependent on pool size 'θ' which may significantly affect the predictive performance. A larger value of θ' may result in a higher processing time required in the formation of clusters. On the other hand, a lower value of θ may result in losing the important clustering information. Sliding window is a widely accepted model for DSM because it has ability the to emphasise on more recent data. One of the approaches for determining the pool/window size is to obtain it from the user.

However, the user must have prior knowledge about the time location of concept drifts within the data streams, which is practically not possible due to unpredictable evolving nature of data streams. By applying a fixed window size, the performance of the predictive model is degraded due to concept drifts. Based on these conditions, it is useful to investigate on variable size sliding window for observing recent concept changes and the window size can be determined dynamically based on the amounts of concept drifts that occur within the data streams.

The conflict detection mechanism and assigning the class labels to the cluster in a case of overlaps of the centroids, particularly when one clusters is passing through another cluster, such as in 1Csurr dataset. It is beneficial to track the direction and velocity of the clusters and store it in the summary statistics. This information could be useful in assigning more accurate class labels to the clusters after the overlaps.

Another research direction is to apply PSDSL in robotics is in the context of autonomous driving. In this case, the robot or vehicle makes decisions based on a stream of sensor data, such as camera images and radar data. However, verifying the correctness of these decisions can be difficult and time-consuming, particularly when it comes to situations that are rare or unexpected. For example, consider the case where a self-driving car encounters a situation where the road is blocked by a fallen tree. The vehicle must decide whether to stop or attempt to navigate around the obstacle. However, verifying the correctness of this decision may require human review, as it can be difficult to determine the best course of action based solely on sensor data. PSDSL's ability to adapt learn from past data and improve the accuracy of the robot's decision-making process. For example, a model could be trained to recognize different types of obstacles and make decisions based on this information.

Feature engineering is an important aspect of DSM that involves identifying and extracting relevant features from the data to build accurate predictive models. This aspect must be explored in the future because the feature engineering helps to reduce the dimensionality of the data, identify the most important features, improve model accuracy, reduce computational complexity, improve interpretability, and increase the robustness of the

predictive models to noise and outliers. Overall, feature engineering is essential in data stream mining to build more accurate and robust predictive models. It is also worth to investigate learning on imbalanced data, which refers to the situation where the classes of interest are not evenly represented in the data, leading to biased models that may not accurately represent the minority class. Techniques such as oversampling, under sampling, and cost-sensitive learning can be used to address imbalanced data.

Finally, the research identified potential future directions for research in hyperparameter tuning for data stream mining. This includes exploring adaptive and online hyperparameter tuning strategies, developing benchmark datasets that capture realistic streaming scenarios, and investigating the integration of domain knowledge into the tuning process.

## 6.3 References


[1] Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining data streams: a review. ACM Sigmod Record, 34(2), 18-26, DOI: 10.1145/1083784.1083789

[2] Lukats, D., Berghöfer, E., Stahl, F., Schneider, J., Pieck, D., Idrees, M. M. & Zielinski, O. (2021, September). Towards Concept Change Detection in Marine Ecosystems. In OCEANS 2021: San Diego–Porto (pp. 1-10). IEEE. DOI: 10.23919/OCEANS44145.2021.9706015

[3] Vermesan, O., Bahr, R., Ottella, M., Serrano, M., Karlsen, T., Wahlstrøm, T., & Gamba, M. T. (2020). Internet of robotic things intelligent connectivity and platforms. Frontiers in Robotics and AI, 7, 104. DOI: 10.3389/frobt.2020.00104

[4] Cambria, E., Li, Y., Xing, F. Z., Poria, S., & Kwok, K. (2020, October). SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis. In Proceedings of the 29th ACM international conference on information & knowledge management (pp. 105-114). DOI: 10.1145/3340531.3412003

[5] Liu, J., Liu, F., & Ansari, N. (2014). Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop. IEEE network, 28(4), 32-39. DOI: 10.1109/MNET.2014.6863129

[6] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015, July). Credit card fraud detection and concept-drift adaptation with delayed supervised information. In 2015 international joint conference on Neural networks (IJCNN) (pp. 1-8). IEEE. DOI: 10.1109/IJCNN.2015.7280527

[7] Watkins, L., Beck, S., Zook, J., Buczak, A., Chavis, J., Robinson, W. H. & Mishra, S. (2017, January). Using semi-supervised machine learning to address the big data problem in DNS networks. In 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 1-6). IEEE. DOI: 10.1109/CCWC.2017.7868376

[8] Chernbumroong, S., Atkins, A. S., & Yu, H. (2011, September). Activity classification using a single wrist-worn accelerometer. In 2011 5th International Conference on Software, Knowledge Information, Industrial Management and Applications (SKIMA) Proceedings (pp. 1-6). IEEE. DOI: 10.1109/SKIMA.2011.6089975

[9] Le, T., Stahl, F., Gaber, M. M., Gomes, J. B., & Di Fatta, G. (2017). On expressiveness and uncertainty awareness in rule-based classification for data streams. Neurocomputing, 265, 127-141. DOI: 10.1016/j.neucom.2017.05.081

[10] Kolter, J. Z., & Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. The Journal of Machine Learning Research, 8, 2755-2790. DOI: 10.1109/ICDM.2003.1250911

[11] Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm.

Information and computation, 108(2), 212-261. DOI: 10.1006/inco.1994.1009

[12] Zubaroğlu, A., & Atalay, V. (2020). Data Stream Clustering: A Review. arXiv preprint arXiv:2007.10781.

[13] Nguyen, H. L., Woon, Y. K., & Ng, W. K. (2015). A survey on data stream clustering and classification. Knowledge and information systems, 45(3), 535-569. DOI: 10.1007/s10115-014-0808-1

[14] Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. D., & Gama, J. (2013). Data stream clustering: A survey. ACM Computing Surveys (CSUR), 46(1), 1-31. DOI: 10.1145/2522968.2522981

[15] Göpfert, C., Ben-David, S., Bousquet, O., Gelly, S., Tolstikhin, I., & Urner, R. (2019, June). When can unlabelled data improve the learning rate? In Conference on Learning Theory (pp. 1500-1518). PMLR. DOI: 10.48550/arXiv.1905.11866

[16] Urner, R., Shalev-Shwartz, S., & Ben-David, S. (2011, January). Access to unlabelled data can speed up prediction time. In ICML.

[17] Urner, R., & Ben-David, S. (2013, December). Probabilistic lipschitzness a niceness assumption for deterministic labels. In Learning Faster from Easy Data-Workshop@ NIPS (Vol. 2, p. 1). https://arxiv.org/pdf/2205.09817.pdf

[18] Idrees, M. M., Minku, L. L., Stahl, F., & Badii, A. (2020a). A heterogeneous online learning ensemble for NSEs. Knowledge-Based Systems, 188, 104983. DOI: 10.1016/j.knosys.2019.104983

[19] Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. IEEE Computational Intelligence Magazine, 10(4), 12-25. DOI: 10.1109/MCI.2015.2471196

[20] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM computing surveys (CSUR), 46(4), 1-37. DOI: 10.1145/2523813

[21] Cabral, G. G., Minku, L. L., Shihab, E., & Mujahid, S. (2019, May). Class imbalance evolution and verification latency in just-in-time software defect prediction. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 666-676). IEEE. DOI: 10.1109/ICSE.2019.00076

[22] Dyer, K. B., Capo, R., & Polikar, R. (2013). Compose: A semisupervised learning framework for initially labelled nonstationary streaming data. IEEE transactions on neural networks and learning systems, 25(1), 12-26. DOI: 10.1109/TNNLS.2013.2277712

[23] Marrs, G. R., Hickey, R. J., & Black, M. M. (2010, September). The impact of latency on online classification learning with concept drift. In International Conference on

Knowledge Science, Engineering and Management (pp. 459-469). Springer, Berlin, Heidelberg.DOI: 10.1007/978-3-642-15280-1_42

[24] Razavi-Far, R., Hallaji, E., Saif, M., & Ditzler, G. (2018). A novelty detector and extreme verification latency model for nonstationary environments. IEEE Transactions on Industrial Electronics, 66(1), 561-570. IEEE. DOI: 10.1109/TIE.2018.2826477.

[25] Bilbao, M. N., & Del Ser, J. (2018). Concept Tracking and Adaptation for Drifting Data Streams under Extreme Verification Latency. Intelligent Distributed Computing XII, 798, 11. DOI: 10.1007/978-3-319-99626-4_2

[26] Frederickson, C., & Polikar, R. (2018, July). Resampling Techniques for Learning Under Extreme Verification Latency with Class Imbalance. In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

[27] Dyer, K. B., & Polikar, R. (2012, June). Semi-supervised learning in initially labelled Non-Stationary environments with gradual drift. In The 2012 International Joint Conference on Neural Networks (IJCNN) (pp. 1-9). IEEE.

[28] Suzuki, E., Deguchi, Y., Takayama, D., Takano, S., Scuturici, V. M., & Petit, J. M. (2013, September). Towards Facilitating the Development of Monitoring Systems with Low-Cost Autonomous Mobile Robots. In International Workshop on Information Search, Integration, and Personalization (pp. 57-70). Springer, Cham. DOI: 10.1007/978-3-319-08732-0_5

[29] Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017a). A survey on ensemble learning for data stream classification. ACM Computing Surveys (CSUR), 50(2), 1-36.

[30] Masud, M. M., Woolam, C., Gao, J., Khan, L., Han, J., Hamlen, K. W., & Oza, N. C. (2012). Facing the reality of data stream classification: coping with scarcity of labelled data. Knowledge and information systems, 33(1), 213-244.

[31] Souza, V. M., Silva, D. F., Gama, J., & Batista, G. E. (2015a, June). Data stream classification guided by clustering on nonstationary environments and extreme verification latency (SCARGC). In Proceedings of the 2015 SIAM International Conference on Data Mining (pp. 873-881). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9781611974010.98.

[32] Spinosa, E. J., de Leon F. de Carvalho, A. P., & Gama, J. (2007, March). Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. In Proceedings of the 2007 ACM symposium on Applied computing (pp. 448-452). DOI: 10.1145/1244002.1244107

[33] MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

[34] Souza, V. M., Silva, D. F., Batista, G. E., & Gama, J. (2015b, December). Classification of evolving data streams with infinitely delayed labels. In 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA) (pp. 214-219). IEEE. DOI: 10.1109/ICMLA.2015.174

[35] Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. Information Fusion, 37, 132-156.

[36] Van Rijn, J. N., Holmes, G., Pfahringer, B., & Vanschoren, J. (2015, November). Having a blast: Meta-learning and heterogeneous ensembles for data streams. In 2015 IEEE international conference on data mining (pp. 1003-1008). IEEE. DOI: 10.1109/ICDM.2015.55

[37] Stahl, F., Gaber, M. M., Aldridge, P., May, D., Liu, H., Bramer, M., & Philip, S. Y. (2012). Homogeneous and heterogeneous distributed classification for pocket data mining. In Transactions on Large-Scale Data-and Knowledge-Centreed Systems V (pp. 183-205). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-28148-8_8

[38] Rish, I. (2001, August). An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).

[39] Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. In The Collected Works of Wassily Hoeffding (pp. 409-426). Springer, New York, NY.

[40] Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. Nonparametric discrimination: Consistency properties. International Statistical Review/Revue Internationale de Statistique, 57(3), 238-247.

[41] Hu, Z., Bodyanskiy, Y. V., Tyshchenko, O. K., & Boiko, O. O. (2018). A neuro-fuzzy Kohonen network for data stream possibilistic clustering and its online self-learning procedure. Applied soft computing, 68, 710-718. DOI: 10.1016/j.asoc.2017.09.042

[42] Korycki, Ł., & Krawczyk, B. (2017, May). Combining active learning and self-labelling for data stream mining. In International Conference on Computer Recognition Systems (pp. 481-490). Springer, Cham, DOI: 10.1007/978-3-319-59162-9_50

[43] Din, S. U., & Shao, J. (2020). Exploiting evolving micro-clusters for data stream classification with emerging class detection. Information Sciences, 507, 404-420. DOI: 10.1016/j.ins.2019.08.050

[44] Naldi, M. C., Ricardo J. C., Eduardo R. H., and Carvalho F., "Efficiency issues of evolutionary k-means." Applied Soft Computing 11, no. 2 (2011): 1938-1952. DOI:10.1016/j.asoc.2010.06.010

[45] Yin, Y., Wei, C., Zhang, G., & Li, C. (2012, November). Implementation of Space Optimized Bisecting K-Means (BKM) Based on Hadoop. In 2012 Ninth Web

Information Systems and Applications Conference (pp. 170-175). IEEE. DOI 10.1109/WISA.2012.47

[46] Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., & Seidl, T. (2010, September). Moa: Massive online analysis, a framework for stream classification and clustering. In Proceedings of the First Workshop on Applications of Pattern Analysis (pp. 44-50). PMLR. DOI:10.1007/s10994-014-5441-4

[47] Žliobaitė, I. (2011, October). Controlled permutations for testing adaptive classifiers. In International Conference on Discovery Science (pp. 365-379). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-24477-3_29

[48] Brzezinski, D., & Stefanowski, J. (2014). Combining block-based and online methods in learning ensembles from concept drifting data streams. Information Sciences, 265, 50-67. DOI: 10.1016/j.ins.2013.12.011

[49] Demšar, J., (2006). Statistical comparisons of classifiers over multiple datasets. The Journal of Machine Learning Research, 7, pp.1-30.

[50] Nemenyi P. B. 1963, Distribution-free multiple comparisons. PhD thesis, Princeton University

[51] Gama, J., Sebastiao, R., & Rodrigues, P. P. (2009, June). Issues in evaluation of stream learning algorithms. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 329-338). DOI: 10.1145/1557019.1557060

[52] Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1), 37-46. DOI: 10.1177/001316446002000104

[53] Japkowicz, N., & Shah, M. (2011). Evaluating learning algorithms: a classification perspective. Cambridge University Press. DOI: 10.1017/CBO9780511921803

[54] Idrees M., Non-Stationary datasets [Online]. Available: https://github.com/mimm1/Non-Stationary-environments [Accessed May 2022]

[55] Fan, W., Huang, Y. A., Wang, H., & Yu, P. S. (2004, April). Active mining of data streams. In Proceedings of the 2004 SIAM International Conference on Data Mining (pp. 457-461). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9781611972740.46

[56] Zhu, X., Zhang, P., Lin, X., & Shi, Y. (2010). Active learning from stream data using optimal weight classifier ensemble. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 40(6), 1607-1621. DOI: 10.1109/TSMCB.2010.2042445

[57] Zhang, P., Zhu, X., Tan, J., & Guo, L. (2010, December). Classifier and cluster ensembles for mining concept drifting data streams. In 2010 IEEE International Conference on Data Mining (pp. 1175-1180). IEEE. DOI: 10.1109/ICDM.2010.125

[58] Wu, X., Li, P., & Hu, X. (2012). Learning from concept drifting data streams with unlabelled data. Neurocomputing, 92, 145-155. DOI: 10.1016/j.neucom.2011.08.041

[59] Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. IEEE Transactions on Neural Networks, 20(3), 542-542.

[60] Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004, September). Learning with drift detection. In Brazilian symposium on artificial intelligence (pp. 286-295). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-28645-5_29

[61] Hammoodi, M. S., Stahl, F., & Badii, A. (2018). Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining. Knowledge-Based Systems, 161, 205-239. DOI: 10.1016/j.knosys.2018.08.007

[62] Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., & Morales-Bueno, R. (2006, September). Early drift detection method. In Fourth international workshop on knowledge discovery from data streams (Vol. 6, pp. 77-86).

[63] Sun, Y., Tang, K., Zhu, Z., & Yao, X. (2018). Concept drift adaptation by exploiting historical knowledge. IEEE transactions on neural networks and learning. DOI: 10.1109/TNNLS.2017.2775225.

[64] Street, W. N., & Kim, Y. (2001, August). A streaming ensemble algorithm (SEA) for large-scale classification. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 377-382). DOI: 10.1145/502512.502568

[65] Wang, H., Fan, W., Yu, P. S., & Han, J. (2003, August). Mining concept-drifting data streams using ensemble classifiers. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 226-235).

[66] Nishida, K., Yamauchi, K., & Omori, T. (2005, June). ACE: Adaptive classifiers-ensemble system for concept-drifting environments. In International Workshop on Multiple Classifier Systems (pp. 176-185). Springer, Berlin, Heidelberg. DOI: 10.1109/ICMLC.2007.4370772

[67] Zhu, Y., & Shasha, D. (2002, January). Statstream: Statistical monitoring of thousands of data streams in real time. In VLDB'02: Proceedings of the 28th International Conference on Very Large Databases (pp. 358-369). Morgan Kaufmann.

[68] Bifet, A., & Gavalda, R. (2007, April). Learning from time-changing data with adaptive windowing. In Proceedings of the 2007 SIAM international conference on data mining (pp. 443-448). Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9781611972771.42

[69] Zhou, A., Cao, F., Qian, W., & Jin, C. (2008). Tracking clusters in evolving data streams over sliding windows. Knowledge and Information Systems, 15(2), 181-214. DOI: 10.1007/s10115-007-0070-x

[70] Aggarwal, C. C., Philip, S. Y., Han, J., & Wang, J. (2003, January). A framework for clustering evolving data streams. In Proceedings 2003 VLDB conference (pp. 81-92). Morgan Kaufmann. DOI: 10.1016/B978-012722442-8/50016-1.

[71] Jiang, N., & Gruenwald, L. (2006). Research issues in data stream association rule mining. ACM Sigmod Record, 35(1), 14-19. DOI: 10.1145/1121995.1121998

[72] 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. DOI: 10.1109/ACCESS.2018.2877138

[73] Stahl, F., Le, T., Badii, A., & Gaber, M. M. (2021). A Frequent Pattern Conjunction Heuristic for Rule Generation in Data Streams. Information, 12(1), 24. ISSN 2078-2489

[74] Umer, M., Polikar, R., & Frederickson, C. (2017, May). Level iw: Learning extreme verification latency with importance weighting. In 2017 International Joint Conference on Neural Networks (IJCNN) (pp. 1740-1747). IEEE. DOI: 10.1109/IJCNN.2017.7966061.

[75] Hachiya, H., Sugiyama, M., & Ueda, N. (2012). Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. Neurocomputing, 80, 93-101. DOI: 10.1016/j.neucom.2011.09.016

[76] Krempl, G.. "The algorithm APT to classify in concurrence of latency and drift." Intl. Symposium on Intelligent Data Analysis. Springer, Berlin, Heidelberg, 2011. DOI: 10.1007/978-3-642-24800-9_22

[77] Hosseini, M. J., Gholipour, A., & Beigy, H. (2016). An ensemble of cluster-based classifiers for semi-supervised classification of Non-Stationary data streams. Knowledge and information systems, 46(3), 567-597.

[78] Ditzler, G., & Polikar, R. (2011, July). Semi-supervised learning in nonstationary environments. In The 2011 International Joint Conference on Neural Networks (pp. 2741-2748). IEEE. DOI: 10.1109/IJCNN.2011.6033578

[79] Li, P., Wu, X., & Hu, X. (2010, October). Mining recurring concept drifts with limited labelled streaming data. In Proceedings of 2nd Asian conference on machine learning (pp. 241-252). JMLR Workshop and Conference Proceedings. DOI: 10.1145/2089094.2089105

[80] Bertini, J. R., Lopes, A. D. A., & Zhao, L. (2012). Partially labelled data stream classification with the semi-supervised K-associated graph. Journal of the Brazilian Computer Society, 18(4), 299-310. DOI: 10.1007/s13173-012-0072-8

[81] Umer, M., & Polikar, R. (2020). Comparative analysis of extreme verification latency learning algorithms. arXiv preprint arXiv:2011.14917.

[82] Woźniak, M., Grana, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. Information Fusion, 16, 3-17. DOI: 10.1016/j.inffus.2013.04.006

[83] Minku, L. L., & Yao, X. (2011). DDD: A new ensemble approach for dealing with concept drift. IEEE transactions on knowledge and data engineering, 24(4), 619-633. DOI: 10.1109/TKDE.2011.58

[84] Nishida, K. (2008). Learning and Detecting Concept Drift, PhD thesis, Hokkaido University, Japan

[85] Kolter, J. Z., & Maloof, M. A. (2005, August). Using additive expert ensembles to cope with concept drift. In Proceedings of the 22nd international conference on Machine learning (pp. 449-456). DOI: 10.1145/1102351.1102408

[86] Parker, B. S., Khan, L., & Bifet, A. (2014, December). Incremental ensemble classifier addressing Non-Stationary fast data streams. In 2014 IEEE International Conference on Data Mining Workshop (pp. 716-723). IEEE. DOI: 10.1109/ICDMW.2014.116

[87] Nguyen, H. L., Woon, Y. K., Ng, W. K., & Wan, L. (2012, May). Heterogeneous ensemble for feature drifts in data streams. In Pacific-Asia conference on knowledge discovery and data mining (pp. 1-12). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-30220-6_1

[88] Rossi, A. L. D., de Leon Ferreira, A. C. P., Soares, C., & De Souza, B. F. (2014). MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data. Neurocomputing, 127, 52-64. DOI: 10.1016/j.neucom.2013.05.048

[89] Cheng, W. X., Katuwal, R., Suganthan, P. N., & Qiu, X. (2017). A heterogeneous ensemble of trees. In 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1-6). IEEE. DOI: 10.1109/SSCI.2017.8285445.

[90] Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). IEEE. DOI: 10.1109/ICDAR.1995.598994

[91] Rodriguez, J. J., Kuncheva, L. I., & Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. IEEE transactions on pattern analysis and machine intelligence, 28(10), 1619-1630. DOI: 10.1109/TPAMI.2006.211

[92] Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomised trees. Machine learning, 63(1), 3-42. DOI: 10.1007/s10994-006-6226-1

[93] Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In Proceedings of the 20th international conference

on machine learning (ICML-03) (pp. 856-863).

[94] Barddal, J. P., Gomes, H. M., & Enembreck, F. (2015, November). A survey on feature drift adaptation. In 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 1053-1060). IEEE. DOI: 10.1016/j.jss.2016.07.005

[95] Yang, L., & Shami, A. (2021). A Lightweight Concept Drift Detection and Adaptation Framework for IoT Data Streams. DOI: 10.1109/IOTM.0001.2100012

[96] Bayram, B., Köroğlu, B., & Gönen, M. (2020, December). Improving Fraud Detection and Concept Drift Adaptation in Credit Card Transactions Using Incremental Gradient Boosting Trees. In 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 545-550). IEEE. DOI: 10.1109/ICMLA51294.2020.00091

[97] Sun, Y., Tang, K., Zhu, Z., & Yao, X. (2018). Concept drift adaptation by exploiting historical knowledge. IEEE transactions on neural networks and learning systems, 29(10), 4822-4832. DOI: 10.1109/TNNLS.2017.2775225

[98] Rodrigues, P. P., Gama, J., & Pedroso, J. P. (2006, April). ODAC: Hierarchical clustering of time series data streams. In Proceedings of the 2006 SIAM international conference on data mining (pp. 499-503). Society for Industrial and Applied Mathematics. DOI: 10.1016/j.patrec.2011.11.022

[99] Silva, A. J., Hruschka, E. R., & Gama, J. (2017). An evolutionary algorithm for clustering data streams with a variable number of clusters. Expert Systems with Applications, 67, 228-238. DOI:10.1016/j.eswa.2016.09.020

[100] Mouss, H., Mouss, D., Mouss, N., & Sefouhi, L. (2004, July). Test of page-hinckley, an approach for fault detection in an agro-alimentary production system. In 2004 5th Asian Control Conference (IEEE Cat. No. 04EX904) (Vol. 2, pp. 815-818). IEEE. ISBN:0-7803-8873-9

[101] Williams, S. M., Parry, B. R., & Schlup, M. M. (1992). Quality control: an application of the cusum. BMJ: British medical journal, 304(6838), 1359. DOI: 10.1136/bmj.304.6838.1359

[102] Namitha K. and G. Santhosh Kumar. 2020. CUSUM Based Concept Drift Detector for Data Stream Clustering. In Proceedings of the 2020 the 4th International Conference on Big Data and Internet of Things (BDIOT 2020). Association for Computing Machinery, New York, NY, USA, 90–95. DOI: 10.1145/3421537.3421548.

[103] Ordonez, C. (2003, June). Clustering binary data streams with k-means. In Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (pp. 12-19).

[104] Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., & Sohler, C. (2012). Streamkm++ a clustering algorithm for data streams. Journal of Experimental Algorithmics (JEA), 17, 2-1. DOI: 10.1145/2133803.2184450

[105] O'callaghan, L., Mishra, N., Meyerson, A., Guha, S., & Motwani, R. (2002, February). Streaming-data algorithms for high-quality clustering. In Proceedings 18th International Conference on Data Engineering (pp. 685-694). IEEE. DOI:10.1109/ICDE.2002.994785

[106] Xu, T. S., Chiang, H. D., Liu, G. Y., & Tan, C. W. (2015). Hierarchical K-means method for clustering large-scale advanced metering infrastructure data. IEEE Transactions on Power Delivery, 32(2), 609-616. DOI: 10.1109/TPWRD.2015.2479941

[107] Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: an efficient data clustering method for very large databases. ACM sigmod record, 25(2), 103-114. DOI: 10.1145/235968.233324

[108] Karypis, G., Han, E. H., & Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. Computer, 32(8), 68-75. DOI: 10.1109/2.781637

[109] Udommanetanakit, K., Rakthanmanon, T., & Waiyamai, K. (2007, August). E-stream: Evolution-based technique for stream clustering. In International conference on advanced data mining and applications (pp. 605-615). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-73871-8_58

[110] Meesuksabai, W., Kangkachit, T., & Waiyamai, K. (2011, December). Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty. In International Conference on Advanced Data Mining and Applications (pp. 27-40). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-25856-5_3

[111] Cao, F., Estert, M., Qian, W., & Zhou, A. (2006, April). Density-based clustering over an evolving data stream with noise. In Proceedings of the 2006 SIAM international conference on data mining (pp. 328-339). Society for industrial and applied mathematics. DOI: 10.1137/1.9781611972764.29

[112] Duan L, Xiong D, Lee J, Guo F (2006) A local density based spatial clustering algorithm with noise. vol 32, pp 4061–4066, DOI 10.1109/ICSMC.2006.384769

[113] Fahy, C., Yang, S., & Gongora, M. (2018). Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams. IEEE transactions on cybernetics, 49(6), 2215-2228. DOI: 10.1109/TCYB.2018.2822552

[114] Chen, Y., & Tu, L. (2007, August). Density-based clustering for real-time stream data. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 133-142). DOI:10.1145/1281192.1281210

[115] Wan, L., Ng, W. K., Dang, X. H., Yu, P. S., & Zhang, K. (2009). Density-based clustering of data streams at multiple resolutions. ACM Transactions on Knowledge discovery from Data (TKDD), 3(3), 1-28. DOI: 10.1145/1552303.1552307

[116] Ankerst, M., Breunig, M. M., Kriegel, H. P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. ACM Sigmod record, 28(2), 49-60. DOI: 10.1145/304181.304187

[117] Tu, Q., Lu, J. F., Yuan, B., Tang, J. B., & Yang, J. Y. (2012). Density-based hierarchical clustering for streaming data. Pattern Recognition Letters, 33(5), 641-645. DOI: 10.1016/j.patrec.2011.11.022

[118] Dawid, A. P. (1984), The prequential approach. Journal of the Royal Statistical Society-A, 147:278–292.

[119] Gama J, Sebastião R, Rodrigues PP (2013) On evaluating stream learning algorithms. Mach Learn 90(3):317–346. DOI: 10.1007/s10994-012-5320-9

[120] Andrade Silva, J., & Hruschka, E. R. (2011, December). Extending k-means-based algorithms for evolving data streams with variable number of clusters. In 2011 10th International Conference on Machine Learning and Applications and Workshops (Vol. 2, pp. 14-19). IEEE. DOI: 10.1109/ICMLA.2011.67.

[121] Stahl, F., Gabrys, B., Gaber, M. M, and Berendsen, M. (2013) An overview of interactive visual data mining techniques for knowledge discovery. WIREs: Data Mining and Knowledge Discovery, Wiley, 3 (4). pp. 239-256. ISSN 1942-4795 DOI: 10.1002/widm.1093

[122] SAS Institute. 2018. Using JMP 14. SAS Institute Inc., USA. ISBN:978-1-63526-541-5

[123] Holmes, G., Donkin, A., & Witten, I. H. (1994, November). Weka: A machine learning workbench. In Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference (pp. 357-361). IEEE.

[124] Schütze, H., Manning, C. D., & Raghavan, P. (2008). Introduction to information retrieval (Vol. 39, pp. 234-65). Cambridge: Cambridge University Press.

[125] Souza V. M. et al., Non-stationary datasets archive, [Online]. Available: https://sites.google.com/site/nonstationaryarchive/datasets [Accessed June 13, 2022].

[126] Reddy, H. V., Agrawal, P., & Raju, S. V. (2013, August). Data labelling method based on cluster purity using relative rough entropy for categorical data clustering. In 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 500-506). IEEE. DOI: 10.1109/ICACCI.2013.6637222

[127] Žliobaitė, I., Bifet, A., Read, J., Pfahringer, B., & Holmes, G. (2015). Evaluation methods and decision theory for classification of streaming data with temporal dependence. Machine Learning, 98(3), 455-482. DOI: 10.1007/s10994-014-5441-4

[128] Bodik, P., Hong, W., Guestrin, C., Madden, S., Paskin, M., & Thibaux, R. (2004). Intel lab data. Online dataset, 90. http://db.csail.mit.edu/labdata/labdata.html [Accessed May 2018]

[129] The Apache SpamAssasin Project - http://spamassassin.apache.org/ [Accessed May 2018]

[130] M. Harries. “Splice-2 comparative evaluation: Electricity pricing". Technical report, The University of South Wales, 1999.

[131] Massive Online Analysis, datasets https://moa.cms.waikato.ac.nz/datasets/ [Assessed Jan 2019]

[132] Bastian M., Heymann S., Jacomy M. (2009). Gephi: an open source Software for exploring and manipulating networks. International AAAI Conference on Weblogs and Social Media

[133] Airport, airline and route data [Online]. Available https://openflights.org/data.html Retrieved 2018-01-01. [Accessed May 2022]

[134] GitHub - ATISLabs/SCARGC.jl: A Julia implementation of Stream Classification Algorithm Guided by Clustering – SCARGC [Online]. Available: https://github.com/ATISLabs/SCARGC.jl

[135] Killourhy, K., & Maxion, R. (2010, September). Why did my detector do that?!. In International workshop on recent advances in intrusion detection (pp. 256-276). Springer, Berlin, Heidelberg.

[136] Domingos, P., & Hulten, G. (2003). A general framework for mining massive data streams. Journal of Computational and Graphical Statistics, 12(4), 945-949.

[137] Haarman, B. C. et al., (2015). Feature-expression heat maps–A new visual method to explore complex associations between two variable sets. Journal of biomedical informatics, 53, 156-161.

## APPENDIX I. DataStream Generation Commands

Massive Online Analysis (MOA) commands to generate drift streams and running prequential evaluation task on data stream and real-world datasets.

---

**System.out.println("SEA Drift");**

DoTask.main("EvaluatePrequential -l (drift.HDWM) -s (ConceptDriftStream -s (generators.SEAGenerator -f 4) -d (ConceptDriftStream -s (generators.SEAGenerator -f 3) -d (generators.SEAGenerator -f 2) -p 50000 -w 1) -p 25000 -w 1) -i 100000 -f 1000".split(" "));

**System.out.println("STAGGER Drift");**

DoTask.main("EvaluatePeriodicHeldOutTest -l (drift.HDWM -p 1) -s (ArffFileStream -f C:\\test\\Dataset\\Drifts\\stagger.arff) -n 100 -i 120 -f 1".split(" "));

**System.out.println("RTREE Recurring Drift");**

DoTask.main("EvaluatePrequential -l (drift.HDWM) -s (RecurrentConceptDriftStream -x 10000 -s (generators.RandomTreeGenerator -o 0) -d (generators.RandomTreeGenerator -u 0) -p 25000 -w 1) -i 100000 -f 1000".split(" "));

**System.out.println("LED");**

DoTask.main("EvaluatePrequential -l drift.HDWM -s (ConceptDriftStream -s generators.LEDGenerator -d (generators.LEDGeneratorDrift -d 7) -p 50000) -i 100000 -f 1000".split(" "));

**System.out.println("Wave Drift");**

DoTask.main("EvaluatePrequential -l drift.HDWM -s (ConceptDriftStream -s generators.WaveformGenerator -d (generators.WaveformGeneratorDrift -d 20) -p 50000 -w 1) -i 100000 -f 1000".split(" "));

**System.out.println("Hyperplane Incremental Drift");**

DoTask.main("EvaluatePrequential -l (drift.HDWM) -s (generators.HyperplaneGenerator -k 10 -t 0.01) -i 100000 -f 1000".split(" "));

**System.out.println("SEA Mixed Drift");**

DoTask.main("EvaluatePrequential -l (drift.HDWM) -s (ConceptDriftStream -s (generators.SEAGenerator -f 2) -d (ConceptDriftStream -s (generators.SEAGenerator -f 3) -d (generators.SEAGenerator -f 4) -p 50000 -w 1) -p 25000 -w 10000) -i 100000 -f 1000".split(" "));

**System.out.println("RandomRBF");**

DoTask.main("EvaluatePrequential -l (drift.HDWM) -s (clustering.RandomRBFGeneratorEvents -n) -i 100000 -f 1000".split(" "));

**Real-world Experiments**

**System.out.println("Electric");**

```
DoTask.main("EvaluatePrequential -l (drift.HDWM) -s (ArffFileStream -f
C:\\test\\Dataset\\Drifts\\elec.arff) -f 500".split(" "));
```

**System.out.println("Spam email");**

```
DoTask.main("EvaluatePrequential -l (drift.HDWM) -s (ArffFileStream -f
C:\\test\\Dataset\\Drifts\\spam_data.arff) -i 5000 -f 100".split(" "));
```

**System.out.println("Sensor");**

```
DoTask.main("EvaluatePrequential -l (drift.HDWM) -s (ArffFileStream -f
C:\\test\\Dataset\\Drifts\\sensor.arff) -i 100000 -f 1000".split(" "));
```

**System.out.println("covtypeNorm.arff");**

```
DoTask.main("EvaluatePrequential -l (drift.HDWM -p 5) -s (ArffFileStream -f
C:\\test\\Dataset\\Drifts\\covtypeNorm.arff) -i 100000 -f 1000".split(" "));
```

## APPENDIX II. Non-Stationary Dataset

MOA commands to Execute EVL prequential evaluation task on non-stationary datasets, real-time data streams and real-world datasets.

---

**1CSurr:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\1CSurr.arff) -r 50 -i 55000 -f 300

**1CDT:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 200 -k 49)) -s (ArffFileStream -f C:\\1CDT.arff) -i 16000 -f 300.

**1CHT:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 300 -k 49)) -s (ArffFileStream -f C:\\1CHT.arff) -i 16000 -f 300.

**2CDT:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\2CDT.arff) -i 16000 -f 300.

**2CHT:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\2CHT.arff) -i 16000 -f 300.

**4CE1CF:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\4CE1CF.arff) -i 173000 -f 300.

**4CR:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\4CR.arff) -i 144400 -f 300.

**4CRE-V1:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100)) -s (ArffFileStream -f C:\\4CRE-V1.arff) -i 125000 -f 300.

**4CRE-V2:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\4CRE-V2.arff) -i 183000 -f 300.

**5CVT:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\5CVT.arff) -i 24000 -f 300.

**FG_2C_2D:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\FG_2C_2D.arff) -i 100000 -f 300.

**GEARS_2C_2D:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\GEARS_2C_2D.arff) -i 200000 -f 300.

**MG_2C_2D:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\MG_2C_2D.arff) -i 200000 -f 300.

**UG_2C_2D:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\UG_2C_2D.arff) -i 200000 -f 300.

**UG_2C_3D:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\UG_2C_3D.arff) -i 200000 -f 300.

**UG_2C_5D:** EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 -k 49)) -s (ArffFileStream -f C:\\UG_2C_5D.arff) -i 200000 -f 300.

### MOA Data Streams

**SEA_Sudden:** EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\SEA_Sudden.arff) -r 1000 -i 2000 -f 1000

**LED_sudden:** EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\LED_sudden.arff) -r 1000 -i 2000 -f 1000

**RBF_Gradual:** EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\RBF_Gradual.arff) -r 1000 -i 2000 -f 1000

**HyperIncremental:** EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\HyperIncremental.arff) -r 1000 -i 2000 -f 1000

**Random Trees Recurring Drift**: EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f (C:\\Random Trees Recurring Drift.arff)) -r 1000 -i 2000 -f 1000

**SEA_NoDrift**: EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\SEA_NoDrift.arff) -r 1000 -i 2000 -f 1000

**LED_NoDrift:** EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\LED_NoDrift.arff) -r 1000 -i 2000 -f 1000

**Hyperplane_NoDrift:** EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\Hyper_NoDrift.arff) -r 1000 -i 2000 -f 1000

**RBF_NoDrift:** EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\RBF_NoDrift.arff) -r 1000 -i 2000 -f 1000

**Wave_NoDrift:** EvaluateEVLPrequential -l moa.labelscarcity.PSDSL -s (ArffFileStream -f C:\\Wave_NoDrift.arff) -r 1000 -i 2000 -f 1000

### Real-world Dataset

Keystroke: EvaluateEVLPrequential -l (moa.labelscarcity.PSDSL -q (clustream.Clustream -h 100 - k 149)) -s (ArffFileStream -f C:\\keystroke.arff) -e (WindowClassificationPerformanceEvaluator -w 150) -r 150 -i 55000 -f 150

## APPENDIX III. Source code for Train Algorithms

**Algorithm 1:** Source code to train classifier and clusters on input labelled examples, predict the pseudo-codes for unlabelled examples and re-train the classifier. Output is trained HDWM classifier.

```
/**
Author: Mobin M. Idrees
*/
public void trainOnInstanceImpl(Instance inst) {
   this.epochs++;
   double[] Pr_cluster = new double[num_classes];
   DataPoint point0 = new DataPoint(epochs);
   while(poolData.size() >= batch_size)
      poolData.removeFirst();
   while(labeledData.size() >= batch_size)
      labeledData.removeFirst();
   if(epochs <= trainSize) // Labeled examples arrive
      {
      labeledData.add(point0);
      Train(HDWM,point0)
      }
   else { // unlabeled examples arrive
     if(prelabelingstate == 0) {
      Pr_cluster = knnClassification(labeledData,inst);
      Instance labelinst= InstancePreLabeling(Pr_cluster,inst);
      DataPoint labelpoint = new DataPoint(labelinst, epochs);
      poolData.add(labelpoint);
      }
  else {
     DataPoint labelpoint = new DataPoint(epochs);
     poolData.add(labelpoint);
   }
  }
    if (epochs == trainSize)    // Build centroid using labeled data
       Centroids = findCentroids(labeledData); //Algorithm 5
    if (testepochs % batch_size  == 0 && epochs > trainSize)  {
     Clustering tempCentroids = Kmean(poolData);
     Clustering intermed= findLabelForCentroids(Centroids, tempCentroids);
//Algorithm 6
    Centroids = (Clustering) intermed.copy();
    FindLabels(poolData,intermed,Centroids); //Algorithm 3
  }
  Train(HDWM,poolData);
}
```

**Algorithm 2:** Predictor for Streaming Data with Scarce Labels (PSDSL).

```
/**
Author: Mobin M. Idrees
*/
public void trainOnInstanceImpl(Instance inst) {
	this.epochs++;
	Initialize(inst);
	if(this.epochs <= trainSize)
		DetermineSelfLearningState(inst);
	if (epochs == batch_size)
		this.bestseed = bestseed();
	if (epochs == trainSize)  {
		FindBestK(inst);
		double acc_micro_count = 0;
		double acc_micro = 0;
		MicroLabeling(labeledData,Centroids);
		Clustering kmean = findCentroids(labeledData);
		kmean = MobinKmean(kmean,labeledData,this.K);
		MicroLabeling(labeledData,kmean);
		for(int i = 0; i < labeledData.size(); i++ ) {
			double[] Pr_micro =
getClusterWeightedVote(Centroids,labeledData.get(i));
			if(Utils.maxIndex(Pr_micro) == labeledData.get(i).classValue() )
				acc_micro_count++;
		acc_micro = ((double)acc_micro_count/trainSize);
		}
		if(acc_micro > PurityThreshold.getValue()){
			if((this.microPurity+this.microFP) >
(this.kMEAN_Purity+this.kMEAN_FP)){
				MicroLabeling(labeledData,Centroids);
				System.out.printf(" micro ");
				microLearning = true;
			}
			else
			{
				System.out.printf(" CGC ");
				CGC = true;
				if(num_classes == this.K) {
					Centroids = findCentroids(labeledData);
				}
				else
				{
					Clustering tempClustering =
findCentroids(labeledData);
					Centroids =
MobinKmean(tempClustering,labeledData,this.K);
					MicroLabeling(labeledData,Centroids);
				}
			}
			prelableing.setChosenIndex(1);
			foundClustering.clear();
			foundClustering.add(Centroids);
		}
		else
		{
			System.out.printf(" SELF ");
			double preAccCLA =
this.prelabelearner[this.bestseed].evaluator.getFractionCorrectlyClassified()*100;
			double seedAccCLA =
this.seedlearner[this.bestseed].evaluator.getFractionCorrectlyClassified()*100;
```

```
        if(preAccCLA > seedAccCLA) {
                        prelableing.setChosenIndex(1);
                        selfLearning = true;
                }
                else {
                        prelableing.setChosenIndex(0);  // do not apply prelabeling
                        selfLearning = false;
                }
        }
        }
        if (testepochs % batch_size  == 0 && epochs > trainSize)  {
                if( microLearning == true || CGC == true)  {
                        microlearning(inst);
                        labeledData.clear();
                        labeledData.addAll(poolData);
                        correctlabelAssigned = 0;
                        if(prelableing.getChosenIndex() == 1)
                                for(int i = 0; i < labeledData.size(); i++ ) {

        labeledData.get(i).setClassValue(newLabeledData[i]);
                                        // train classifiers
        this.seedlearner[this.bestseed].trainOnInstance(labeledData.get(i));
                                        if(labeledData.get(i).classValue() ==
labeledData.get(i).getTrueLabel() )
                                                correctlabelAssigned++;
                                        correctlabelAssignedRatio =
((double)correctlabelAssigned/batch_size)*100;
                                }
                        foundClustering.clear();
                        foundClustering.add(Centroids);
                }
                poolData.clear();
        }
}
```

Algorithm 3:  Heterogeneous Dynamic Weighted Majority (HDWM) classifier.

```
public void trainOnInstanceImpl(Instance inst) {
   this.epochs++;
      double[] Pr = new double[inst.numClasses()];
      bestLearnerIndex = 0;
      double maxWeight = 0.0;
      double weakestExpertWeight = 1.0;
      int weakestExpertIndex = -1;
      boolean prediction = false;
      boolean driftstate = false ;
      // Loop over seeds
     for (int i = 0; i < this.experts.size(); i++) {
        boolean deleted = false;
           double[] pr = this.experts.get(i).getVotesForInstance(inst);
           int yHat = Utils.maxIndex(pr);
           if(this.epochs % this.periodOption.getValue() == 0)
           {
              if ((yHat != (int) inst.classValue()))
                 this.weights.set(i, this.weights.get(i) * this.betaOption.getValue());
           }
                        // delete learner's that has weight below theta and that does not belong
to seed experts
                                if (weights.get(i) < this.thetaOption.getValue() && i > bagSize
```

```
                        experts.remove(i);
                    weights.remove(i);
                        ddm.remove(i);
                    deleted = true;
                }

            // do not take prediction and do not add weight if the learner is deleted
                    if(!deleted){
                      Pr[yHat] += this.weights.get(i);
                      maxWeight = Math.max(maxWeight, this.weights.get(i));
                      if (this.weights.get(i) < weakestExpertWeight && i > bagSize) {
                        weakestExpertIndex = i;
                        weakestExpertWeight = weights.get(i);
                      }
                      if ((yHat != (int) inst.classValue()))
                        prediction = false;
                      else
                      prediction = true;
                          this.ddm.get(i).input(prediction ? 0.0 : 1.0);
                      if (this.ddm.get(i).getChange() ) {
                      // numberOfDrifts++;
                       driftstate = true;
                       }
                      if (this.ddm.get(i).getWarningZone()  )
                        for (int j = 0; j < experts.size(); j++)
                                this.experts.get(j).trainOnInstance(inst);
                } // deleted
        }
        //Active Drift
             if (driftstate ==true && this.epochs % this.periodOption.getValue() == 0)
              {  //
        removeWeakestLearner(weakestExpertIndex);
             int index = getIndexOfMin(weights);
        this.weights.set(index, 0.5);
             }
        if (this.epochs % this.periodOption.getValue() == 0) {
           //Global Prediction
           int yHat = Utils.maxIndex(Pr);
                if (yHat != (int) inst.classValue())
                   prediction = false;
                else
                        prediction = true;
                driftDetectionGlobal.input(prediction ? 0.0 : 1.0);
                if (driftDetectionGlobal.getChange() )
                         numberOfDrifts++;
                if (driftDetectionGlobal.getWarningZone() )
                        numberOfWarning++;
                scaleWeights(maxWeight);
                if (yHat != (int) inst.classValue()) {
               if (experts.size() >= this.maxexpertsOption.getValue() && experts.size() >
bagSize) {
                  removeWeakestLearner(weakestExpertIndex);
                }
                // add new learner when Global wrong prediction is detected
                addLearner();
           }
         }
        for (Classifier expert : this.experts) {
           expert.trainOnInstance(inst);
        }
```

**Algorithm 4:** Source code to predict class label for a test example. Inputs is test example data Point, Output is predictions for each class.

```
/**
 Author: Mobin M. Idrees
*/

public double[] getVotesForInstance(Instance inst) {
       double[] Pr_cluster = new double[num_classes];
         if (this.trainingWeightSeenByModel > 0.0)
            Pr_cluster = knnClassification(labeledData,inst);
         testepochs++;
         return Pr_cluster;
 }
```

**Algorithm 5:** Source code to determine class labels for unlabelled examples stored in a pool. Inputs is unlabelled data Point, previous and current centroids. Output is labelled data points.

```
/**
 Author: Mobin M. Idrees
*/

private void FindLabels(points, Clustering intermed, Clustering centroids) {
  double[] Pr_cluster;
  labeledData.clear();
  for (int i = 0; i < points.size(); i++) {
       Pr_cluster = getknnClassification(points.get(i));
       Instance inst= InstancePreLabeling(Pr_cluster,points.get(i));
       labeledData.add((DataPoint) inst);
   }
}
```

**Algorithm 6:** Source code to determine nearest cluster by using K-Nearest Neighbour algorithm. Inputs is unlabeled dataPoint. Output is nearest centroids for the data point.

```
/**
Author: Mobin M. Idrees
*/
private double[] getknnClassification(DataPoint dataPoint) {
 double[] Pr_cluster = new double[num_classes];
 SphereCluster Kernel = null;
 SphereCluster closestKernel = null;
 for ( int i = 0; i < foundClustering.size(); i++ ) {
    Clustering c = foundClustering.get(i)
    double minDistance = Double.MAX_VALUE;
    for ( int j = 0; j < c.size(); j++ ) {
      Kernel = (SphereCluster) c.get(j);
```

```
        double distance =  Kernel.getCenterDistance(dataPoint);
        if (distance < minDistance  ) {
          minDistance = distance;
          closestKernel=Kernel;
        }
      }
  }
 if(closestKernel != null) {
    double yHat_cluster = closestKernel.getId();
    if(yHat_cluster != -1)
      Pr_cluster[(int) yHat_cluster] += 1;
 }
 return Pr_cluster;
}
```

**Algorithm 7:** Source code to generate ground truth clusters by using initial labelled data. Inputs are labelled data. Output is ground truth centroids.

```
/**
 Author: Mobin M. Idrees

public Clustering findCentroids(points){
  HashMap<Integer, Integer> labelMap = classValues(points);
  num_classes = labelMap.size();
  num_Attributes = points.get(0).dataset().numAttributes()-1;
  Attribute classLabel = points.get(0).dataset().classAttribute();
  num_classes = labelMap.size();
  sorted_points = new ArrayList[num_classes];
  oldcenters = new ArrayList[num_classes];
  for (int i = 0; i < num_classes; i++) {
      sorted_points[i] = new ArrayList<Instance>();
      oldcenters[i] = new ArrayList<Double>();
  }
  for (Instance point : points) {
     int clusterid = (int)point.classValue();
     sorted_points[labelMap.get(clusterid)].add((Instance)point);
  }
  clusters = new AutoExpandVector<Cluster>();
  for (int i = 0; i < num_classes; i++) {
     if(!sorted_points[i].isEmpty()) {
        oldcenters[i].addAll(getCentroids(sorted_points[i]))  ;
        SphereCluster s = new SphereCluster();
        double[] cent  = new double[oldcenters[i].size()];
        for (int j = 0; j < oldcenters[i].size(); j++)
           cent[j] = oldcenters[i].get(j).doubleValue();
        s.setCenter(cent);
        s.setId(sorted_points[i].get(0).classValue());
        s.setGroundTruth(sorted_points[i].get(0).classValue());
        clusters.add(s);
     }
  }
return new Clustering(this.clusters);
```

**Algorithm 8:** Source code to determine label for current centroid. Inputs are previous and current centroids. Output labelled centroids.

```
/**
Author: Mobin M. Idrees
*/

private Clustering findLabelForCentroids(Clustering gtCentroids, Clustering
tempCurrentCentroids) {
   AutoExpandVector<Cluster> interm = new AutoExpandVector<Cluster>() ;
   double[] cent  = new double[this.num_Attributes];
   double[] gtLabels  = new double[tempCurrentCentroids.size()];
   for(int i =0; i < tempCurrentCentroids.size(); i++ ) {
      int closestCluster = 0;
       double minDistance = Double.MAX_VALUE;
       double distances;
       int bestPoint = 0;
      for (int j=0; j< tempCurrentCentroids.size(); j++){
          distances = distance(gtCentroids.get(i).getCenter(),
tempCurrentCentroids.get(j).getCenter());
         if (distances  < minDistance) {
              minDistance = distances ;
              bestPoint = j;
          }
       }
       gtLabels[i] = bestPoint;
       double[] c = gtCentroids.get(i).getCenter();
       double[] t = tempCurrentCentroids.get(bestPoint).getCenter();
       for (int j=0; j< cent.length; j++){
         ArrayList<Double> temp_points = (ArrayList<Double>) new ArrayList();
         temp_points.add(c[j]);
         temp_points.add(t[j]);
         cent[j] = median(temp_points);
       }
         interm.add((Cluster) new SphereCluster(cent,0).copy());
         interm.get(i).setGroundTruth(bestPoint);
         interm.get(i).setId(bestPoint);
         tempCurrentCentroids.get(i).setGroundTruth(bestPoint);
         tempCurrentCentroids.get(i).setId(bestPoint);
    }  // end iterate macro
   return new Clustering (interm);
}
```

## APPENDIX IV. Source code for SSL Periodic Holdout Test

**Algorithm 1:** Source code for evaluating label scarcity classifier on a stream by periodically testing on a holdout set.

```
/**
 Author: Mobin M. Idrees
*/
public class EvaluateSSLPeriodicHeldOut extends SSLMainTask {
 // Training Begin
   while (instancesProcessed < this.trainSizeOption.getValue()) {
            //batch or chunk loop
      while (instancesProcessed < instancesTarget && stream.hasMoreInstances() == true)
{
         InduceLabelScarcity(scarsetrainInst,instancesProcessed);
         SSLearner.trainOnInstance((Example) scarsetrainInst);
         if (monitor.resultVisualRequested())
             visualizer.drawpoints(scarsetrainInst,instancesProcessed);
          // End of batch or chunk loop
         pointarray0=SSLearner.getCluster();
         gtClustering0 = SSLearner.gtClustering0(pointarray0);
         macro0 = SSLearner.gtmacro0(gtClustering0);
         evalClustering0 = macro0;
         evaluateClustering(evalClustering0, gtClustering0, pointarray0);
         visualizer.drawClusterings(SSLearner);
      }
 // Testing Begin
     double[] prediction = SSLearner.getVotesForInstance(testInst);
     evaluator.addResult(testInst, prediction);
    }
```

**Algorithm 2:** Source code for inducing label scarcity on a stream, input is labelled example from data stream and time step. Output is unlabelled example

```
protected void InduceLabelScarcity(Instance inst, long epochs) {
Instance inst = inst.getData();
prob = Math.random()%100;
if ( prob >= (double) this.m_dUlRate.getValue()/100){
      inst.setScarceLabel(false);
else {
      inst.setScarceLabel(true);
      unLabeledCount++;
}
```