EasyChair Preprint

№ 15689

Diversity Measure for Drift Detection in Data Streams

Osama A. Mahdi, Savitri Bevinakoppa and Sarabjot Singh

January 8, 2025

Diversity Measure for Concept Drift Detection in Data Streams

Osama Mahdi
School of IT and Engineering
Melbourne Institute of Technology
Melbourne, Australia
omahdi@mit.edu.au

Savitri Bevinakoppa
School of IT and Engineering
Melbourne Institute of Technology
Melbourne, Australia
sbevinakoppa@mit.edu.au

Sarabjot Singh
School of IT and Engineering
Melbourne Institute of Technology
Melbourne, Australia
sarabjotsingh@academic.mit.edu.au

***Abstract*— Concept drift is a notable challenge in machine learning, data mining, and applications involving big data and large-scale data processing. The employment of diversity measures has emerged as an effective strategy. We examine and investigate the role of diversity measures in detecting concept drift and provide a comparative analysis of four different approaches: DMDDM for drift detection in a fully supervised binary classification context, DMDDM-S in a semi-supervised context, DMODD for online drift detection in a fully supervised multi-classification context, and HBBE, a hybrid block-based ensemble designed for addressing different types of concept drifts. Our comparative analysis evaluates the efficacy of these methods in detecting concept drift and enhancing model performance. The results confirm the effectiveness of all four approaches within their respective settings. Moreover, this paper provides insights into potential advancements and research opportunities in the application of diversity measures for concept drift detection.**

Keywords*— *Concept drift, data stream, non-stationary environments, big data applications

I. INTRODUCTION

The constant transformation or evolution of data is a crucial concern in dynamic settings and applications, including aviation, autonomous vehicles, nuclear power plants, healthcare, defense, smart urban infrastructure, and the aerospace industry. Fundamentally, the critical characteristics of these environments are subject to change, potentially leading to negative consequences, such as endangering human lives, if not adequately addressed [1]. Consequently, learning methods must employ sophisticated algorithms to monitor these changes and adapt accordingly. Furthermore, the effectiveness of learning algorithms may vary due to the changing nature of incoming data, meaning that an algorithm that is effective today may become outdated following changes in the environment or data.

The literature on learning from data streams identifies the phenomenon of class distribution changes within data streams as concept drift [2]. In the context of machine learning, concept drift describes a situation where the statistical properties of the target variable, which the model is designed to forecast, shift over time [3]. This implies that the original input data's relevance to the model has altered significantly, yet the model remains oblivious to these modifications and, as a result, fails to make precise predications. Thus, it is imperative for learning algorithms to detect concept drift in dynamic data streams and accordingly adjust or renew their prediction models. To address this challenge, adaptive learning models are developed, employing drift detection methods to pinpoint the instances of drift in changing environments [2]. Concept drift is a phenomenon that impacts a broad range of applications, recognised and tackled across various domains including medicine, industry, education, and commerce. For instance, in the medical sector, the diminishing efficacy of antibiotics with prolonged use due to microbial resistance highlights concept drift; misuse of antibiotics can lead to resistance, compromising their effectiveness when critically needed. This reflects how alterations in medication usage can influence disease progression. In the financial sector, areas like bankruptcy forecasting or credit evaluation, traditionally viewed as stable, may experience concept drift due to underlying shifts in social trends and behaviors. In the realm of industrial monitoring, shifts in production processes, service monitoring, or consumer actions can lead to concept drift. Likewise, in the field of transportation, traffic control systems that utilize data mining to assess traffic conditions, such as vehicle density and accident rates, need to adapt to concept drift caused by seasonal or long-term changes in traffic flows.

Among the variety of concept drift techniques that have been proposed so far, the diversity measure has been used as a promising method for detecting concept drifts [4], [5], [6], [7], however, there is a necessity for a comparative analysis to ascertain their efficiency, identify the most effective approach, and delineate a trajectory for their sustained evolution in machine learning contexts. In this paper we examine the role of the diversity measure in detecting concept drift and compare four different ways of using them: DMDDM for drift detection in a fully supervised binary classification context [4], DMDDM-S in a semi-supervised framework [5], DMODD for online drift detection in a fully supervised multi classification context [7], and HBBE, a hybrid block-based ensemble designed for addressing various forms of concept drifts [6].

This study seeks to explore the following research questions: **(1)** *How do different methodologies, such as DMDDM in a fully supervised context, DMDDM-S in a semi-supervised framework, DMODD as online drift detection, and HBBE as a hybrid block-based ensemble, utilize diversity measures effectively for concept drift detection*? **(2)** *In a comparative analysis, which method among the four methods performs the best in terms of detecting concept drift and enhancing model performance*? **(3)** *What are the potential future directions and research opportunities in using diversity measures for concept drift detection*?

To answer these research questions, we present three analyses of the four methods, evaluating their effectiveness in detecting concept drift and their ability to improve model performance.

The paper is structured as follows: Section II provides an overview and notation on concept drift. Section III covers the literature review. Section IV discusses diversity measures and their role in concept drift detection. Section V presents the experimental results and analysis of DMDDM, DMDDM-S, DMODD, and HBEE. Finally, Section VI concludes the paper.

II. CONCEPT DRIFT

This section presents an overview of concept drift, including its definition, origins, varieties, and the methodologies for adapting to concept drift [2], [8].

A. The Definition of Concept Drift

Assuming P_{t0} denotes the joint probability distribution of the input variable x and the target variable y at time t_0, and P_{t1} denotes the joint probability distribution of x and y at time t_1, concept drift is said to occur if **Equation (1)** is satisfied when transitioning from t0 to t1.

$$\exists x : P_{t0}(x, y) \neq P_{t1}(x, y) \quad (1)$$

Currently, the distribution of the underlying data has shifted away from concept C_1 to a new concept C_2. This shift is attributed to the dynamics of joint probability $P_t(x, y) = P_t(x)$ $P_t(y|x)$, and if **Equation (2)** is fulfilled as time progresses from t_0 to t_1, a concept drift is observed. Variations in either $P_t(x)$ or $P_t(y|x)$ are capable of inducing concept drift.

$$\exists x : P_{t0}(x)P_{t0}(x|y) \neq P_{t1}(x)P_{t1}(x|y) \quad (2)$$

B. The Origins of Concept Drift

Based on the concept of concept drift and the properties of joint probability distributions, it is identified to have three primary origins:

Virtual Concept Drift: when there is a change in the probability of x, while the probability of y given x remains unchanged, i.e., $P_{t0}(x) \neq P_{t1}(x)$ and $P_{t0}(x|y) \neq P_{t1}(x|y)$. This case belongs to virtual concept drift, which does not affect its decision boundary and only changes the feature space. *Real Concept Drift*: when the probability of y given x undergoes a change, the probability of x remains the same, i.e., $P_{t0}(x|y) \neq P_{t1}(x|y)$ and $P_{t0}(x) = P_{t1}(x)$. This scenario significantly affects the prediction model, representing a genuine concept drift that alters both the feature space and its decision-making boundary. Also, in line with Bayesian decision theory [9], Equation (3) is derived:

$$P(y|x)\frac{P(y) * P(x|y)}{P(x)} \quad (3)$$

It is evident that $P_t(y)$ and $P_t(x|y)$ influence $P_t(y|x)$, thereby contributing to an indirect real concept drift. The various forms of concept drift resulting from distinct causes are depicted in Figure 1, where (X_1, X_2) symbolizes the two-dimensional feature space, and y denotes the category label.

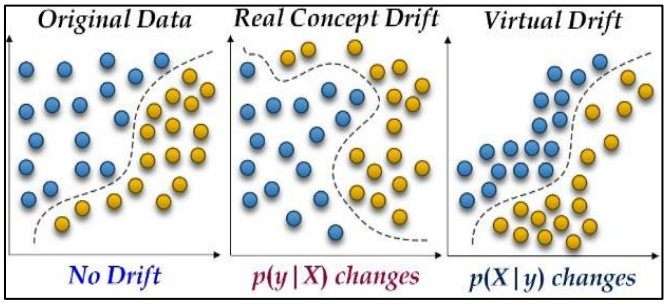

Figure 1. Two forms of drift are illustrated with instances shown as differently colored circles

C. The Types of Concept Drifts

Research identifies several patterns in concept changes, as depicted in Figure 2. An Sudden Drift occurs when the original distribution S_t is instantly replaced by a new distribution S_{t+1} at a specific time t, significantly impairing classifier performance. Gradual Drifts involve a slower change where examples from distributions S_t and S_{t+1} intermingle, with S_t examples decreasing over time and S_{t+1} examples increasing. Recurrent Concepts are concepts that were active in the past and may reappear after a period of absence.

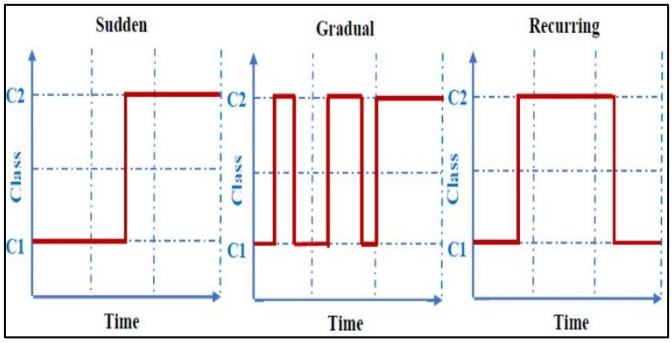

Figure 2. Patterns of Concept Drift.

III. BACKGROUND

In this section, we review well-known drift detectors from previous studies that are highly relevant to our research.

The Fast Hoeffding Drift Detection Method (FHDDM) [10] uses Hoeffding's inequality within a specified window size n to detect drifts. A drift is indicated if there's a significant change between current probabilities and the peak of accurate forecasts.

The Drift Detection Method (DDM) [11] is notable for using classifier error to detect drifts. An increase in error and training samples may indicate a shift. DDM sets a caution threshold, and when error rates reach it, incoming samples are placed in a window. If the error rate hits the drift threshold, the classifier is reconfigured using these samples.

The Adaptive Sliding Window (ADWIN) approach [12] shifts a window w across prediction outcomes, evaluating two sub-windows. If a significant difference in their averages is detected, ADWIN flags a concept drift and removes elements from the window's end until the variation disappears.

The HDDM_A and HDDM_W tests [13] use Hoeffding's bounds to identify drifts. HDDM_A compares moving averages, while HDDM_W examines the weight of these averages using the EMWA forgetting scheme [14]. The creators found that HDDM_A is better for immediate shifts, while HDDM_W excels at detecting gradual drifts.

The PH Test technique [15], used in signal processing for drift detection, calculates a cumulative discrepancy between observed values and their average up to the present moment T, captured by m_T. The minimum value of M_T, denoted as M_T, is updated over time. A significant discrepancy between m_T and M_T indicates a concept drift.

The SegDrift2 approach [16] uses two storage mechanisms: one combines new and old data, while the other stores only new entries. It evaluates the mean values in both repositories and identifies a concept drift when the discrepancy between these means exceeds a predefined threshold.

SEED [17] uses a window approach to compare two sub-windows. When a significant difference in their averages is detected, the earlier segment is discarded. SEED also applies

Hoeffding's Inequality with the Bonferroni correction to determine the test statistic and perform block compression, removing unnecessary cut points.

RDDM [18] was introduced to improve DDM's sensitivity. It discards older data points and periodically refreshes its statistics for drift detection. The creators noted that RDDM often outperforms DDM in accuracy and promptness but may result in more false positives and higher memory usage.

EDDM [19] modifies DDM by tracking the distance between consecutive errors instead of the error rate. Using the same warning system as DDM, EDDM signals drift when the gap between sequential errors widens, indicating unstable data concepts.

The Accuracy Updated Ensemble (AUE2) [20] uses an online classifier to update individual learning models directly, unlike the Adaptive Weighted Ensemble (AWE) which only adjusts weights. When no drift is detected, classifiers improve as if trained on a single, large dataset, allowing for a reduced block size without compromising accuracy.

Accuracy Weighted Ensemble (AWE) [21] trains a new classifier with each incoming data block using static algorithms like Naive Bayes, C4.5, or RIPPER. After training, the current classifiers are evaluated using mean square error on the latest data block. The top n classifiers are then selected to refresh the ensemble.

The Dynamic Weighted Majority (DWM) [22] weights incremental classifiers based on accuracy after each example. For every error, a classifier's weight is reduced by a set factor. Periodic evaluations of the ensemble may lead to adding new classifiers. However, training on many examples can create numerous components, indicating that future work could focus on pruning classifiers.

Learn++.NSE [23] employs a block-based ensemble strategy for incremental learning of concept drift. With each new data block, it trains a new classifier and uses a dynamically weighted majority voting system. The innovation lies in calculating voting weights based on the time-adjusted accuracy of each classifier across both current and past environments.

IV. Diversity Measure and its Role for Concept Drift Detection

In non-stationary environments with concept drift, methodologies are categorized into *Statistical-based*, *Windows-based*, and *Ensemble-based Methods* [2]. Statistical-based Methods monitor the learning process by observing changes in online error rates, where a significant decline in performance indicates concept drift. Windows-based Methods use a static reference window for historical data and a dynamic sliding window for recent data; a significant discrepancy between these windows signals concept drift. Ensemble-based Methods divide data streams into blocks, replacing the least effective ensemble member with a new classifier after evaluations, effectively identifying gradual drifts while maintaining accuracy. These techniques benchmark the efficacy of learning algorithms in managing concept drift.

In ensemble learning with static data, diversity among members is crucial. Evaluating diversity provides insights into methods fostering it, leading many studies to use diversity as a criterion for pruning ensemble components [25-27]. Limited efforts have aimed at enhancing diversity; for example, research in [27] examines diversity's impact on online ensemble learning and modifies the Poisson distribution in online bagging to better address concept drift, focusing on accuracy rather than diversity. To the authors' knowledge, using diversity measures to directly evaluate component classifiers and detect drifts is a novel approach, contrasting with previous methods that relied on classification accuracy. This innovation uses the disagreement measure and the PH test to create four distinct algorithms, each with a unique objective. The disagreement measure quantifies diversity as the proportion of discordant decisions out of total observations, reflecting performance variance between two classifiers on identical training sets. It indicates how classifiers respond differently to data stream changes and is one of the simplest diversity indicators [28]. These four algorithms, which incorporate diversity measures to adapt to drift in dynamic settings, represent a significant shift from existing methods, offering rapid drift response with minimal time and memory requirements. The following subsections (A, B, C, and D) will detail the primary contributions of each method.

A. *DMDDM as a Drift Detector in a Fully Supervised Binary Classification Context.*

To calculate the diversity between component classifiers on a pairwise basis, consider $X = x_1, \ldots, x_n$ as a labeled dataset and y' v = [y' v ($x1$), . . . , y' v (xn)] as an n-dimensional binary vector representing the outputs of classifier h_v. In this vector, y'_v (x_j) = 1 indicates a correct prediction of the class label by h_v, and 0 indicates an incorrect prediction. **Table 1** showcases (referred to as oracle outputs) all possible prediction outcomes for a pair of classifiers h_u and h_v, with the condition that $h_u = h_v$. Here, N_{ab} denotes the count of instances $x_j \in X$ where y'_u (x_j) = a and y'_v (x_j) = b. Consequently, the probability values for N_{ab} are outlined as follows:

- N_{10} represents the count of instances where classifier C_i forecasts class 1 while classifier C_j forecasts class 0.
- N_{01} represents the count of instances where classifier C_j forecasts class 1 while classifier Ci forecasts class 0.
- N_{11} represents the count of instances where both C_i and C_j forecast class 1.
- N_{00} represents the count of instances where both Ci and Ci forecast class 0.

Table 1: The association between two classifiers (2 × 2).

$h_u = h_v$	h_u corrects (1)	h_u incorrect (0)
h_v corrects (1)	N_{11}	N_{10}
h_v incorrect (0)	N_{01}	N_{00}

Calculating the diversity between two base classifiers (h_u and h_v) through the disagreement measure is quantified by Equation (4):

$$D_{u:v} = N^{10} + N^{01} \quad (4)$$

The PH test utilizes a variable m_T to track the cumulative differences in observed values e (error estimates). To calculate these values in a prequential (incremental with forgetting) manner, two primary methods are employed: *sliding windows and fading factors*. The fading factor method is applied across the four strategies. This method systematically removes outdated information by applying a factor to the previous summary, followed by the addition of a new value derived

from recent data. Conversely, alternative strategies employ sliding windows to maintain a collection of the d most recent examples at any given time, thereby constraining the sample size for analysis. Consequently, the *fading sum* $S_{x,\alpha}$ and the fading increment N_α at time *t* for a sequence of objects *x* are computed as follows:

$$S_{x:\alpha}(t) = x^t + \alpha + S_{x,\alpha}\,(t-1) \tag{5}$$

$$N_\alpha(t) = 1 + \alpha * N_\alpha\,(t-1) \tag{6}$$

Then, the *fading average* is computed at observation *i*:

$$M_\alpha(t) = \frac{S_\alpha(t)}{N_\alpha(t)} \tag{7}$$

Per [15], the fading factors method is more efficient in terms of time and memory usage than the sliding windows technique. Hence, across the four methodologies, the focus is on diversity measures observed through the fading factor, rather than on error rates. By incorporating the diversity value from Equation (4) into Equation (5), we derive the subsequent equations.

$$S_{u:v,\alpha}(t) = D_{u:v} + \alpha * S_{u:v,\alpha}\,(t-1) \tag{8}$$

$$M_\alpha(t) = \frac{S_{u:v,\alpha}(t)}{N_\alpha(t)} \tag{9}$$

Equation (9) is applicable alongside the PH test for tracking the diversity between two classifiers. Through this approach, the PH test triggers a drift alert when the predictions of the components (h_u and h_v) begin to diverge in an unexpected manner, effectively recognizing a notable rise in diversity. Equation (10) is then utilized to compute the cumulative difference m_T , representing the discrepancy between observed values and their average up to the present time *t*:

$$m_T = \sum_{t=1}^{T} (x_t - x_T^- - \delta) \tag{10}$$

Where $x_T^- = \frac{1}{T}\sum_{t=1}^{t} x_t$ and δ corresponds to the magnitude of changes that are allowed. Additionally, the minimum value of this variable is calculated as follows:

$$M_T = min(m_T, t = 1 \ldots T) \tag{11}$$

In the final step, the test observes the disparity between M_T and m_T: as follow:

$$PH_T = m_T - M_T. \tag{12}$$

Algorithm 1 outlines the DMDDM method from reference [6], starting with initial data stream sample processing to evaluate classifier predictions (lines 1-3). An oracle output table, as shown in Table 1, identifies instances (N_{10} and N_{01}) where classifier pairs produce differing outcomes on the same training data x^t. The algorithm counts occurrences where one classifier is correct and the other is incorrect (lines 4-5). The disagreement metric is calculated by aggregating these counts and normalizing them against the total number of classifiers (line 6). The fading factor technique is then applied, calculating the fading sum and increment (lines 7-8) and using the diversity score from line 6 as a substitute for error predictions from the PH test. The fading average is computed in line 9. To monitor classifier pair diversity, the PH test (lines 10-13) uses variable m_T to evaluate the cumulative deviation between the observed diversity measure and its average. This test compares the current m_T value against the lowest recorded M_T value to determine if the difference exceeds a threshold, indicating a drift.

Algorithm 1. Pseudocode of Drift Detection Method (DMDDM)

Required: *S*: Data Stream of Examples with Fully Labelled Data, Forgetting factor α: $0 < \alpha < 1$, Admissible change: δ= 0.1, Drift threshold: λ = 100, M_t: 1.0D

Output: Ensure: Drift {TRUE, FALSE}

1 For each example $x^t \in S$ DO
2 C_v_*Prediction* = get prediction using x_t
3 C_u_*Prediction* = get prediction using x_t
4 IF C_v _*Prediction* = 0.0 and C_u _*Prediction* = 1.0 THEN b++
5 IF C_u_*Prediction* = 0.0 and C_v_*Prediction* = 1 THEN c++
6 *The Disagreement*, $D_{u,v}$= $b + c$ / L
7 $S_{u:v,\,\alpha}\,(t) = D_{u:v} + \alpha \times S_{u:v,\alpha}\,(t-1)$
8 $N_\alpha(t) = 1 + \alpha \times N_\alpha\,(t-1)$
9 $M\alpha(t) = S_{u:v,\alpha(t)}$ / $N\alpha(t)$
10 SumDiversity = SumDiversity + $M\alpha(t)$
11 m_T = $(m_T + M_\alpha(t)\,(SumDiversity$ / instancesSeen) - $\delta)$
12 M_T = *GettheMin* (M_T, m_T)
13 $PH^{test} = m_T - M_T$
14 **IF** $PH^{test} > \lambda$ THEN
15 **Return** True and incrementally train classifier C_v and C_u with x^t
16 **Else Return** False
17 ENDFOR

B. DMDDM-S as a Drift Detector in a Semi-Supervised Binary Classification Context.

DMDDM-S [5] builds upon the concept of DMDDM by addressing situations where class labels for the incoming data stream are unavailable, particularly in the context of concept drift challenges. In a binary classification scenario involving a pair of classifiers, each component can predict an example as either 0 (negative class) or 1 (positive class). By analyzing the predictions from each classifier, we can determine the level of disagreement between their predictions. Thus, this method focuses on quantifying the discrepancy in predictions between pairs of classifiers without considering the actual class labels.

The DMDDM strategy initially detects concept drift in a semi-supervised setting (DMDDM-S), identifying disagreements in binary classification without actual labels. This new drift detection mechanism also identifies abrupt drifts in scenarios without class labels. To our knowledge, this is the first attempt to use such a method for concept drift detection. Additionally, we integrate k-prototype clustering to label unlabelled data and retrain the model with both new and previous labels to match the current concept. Our findings show that this drift detector, even with only 50% labelled data, identifies drifts more quickly and efficiently in terms of runtime and memory compared to traditional fully labelled methods.

Algorithm 2 introduces the DMDDM-S technique, which begins by processing each data stream example to get predictions from two classifiers (line 1). It identifies instances where these classifiers disagree and tallies cases where one is correct, and the other is incorrect (lines 1-3). Using a disagreement measure combined with the PH test, it detects concept drift (line 5). The disagreement measure is calculated by summing these observations and dividing by the total number of classifiers.

The fading factor method is then applied (lines 6-8), with the diversity value replacing error estimates from the original PH test. The modified PH test tracks classifier pair diversity (lines 9-13) using variable m_T to measure the disparity between the observed diversity and its average. A drift is indicated if the difference between m_T and the minimum M_T exceeds a threshold. Upon drift detection, the algorithm labels current unlabelled data for model retraining, maintaining accuracy. It merges windows of labelled (W_{ld}) and unlabelled data (W_{uld}) using K-prototype clustering. After drift detection, the drift detector is evaluated (lines 14-15), and the model is incrementally trained with newly labelled data (N_{ld}) (line 16). If the labelled data window (W_{ld}) reaches a predefined size, the oldest data is replaced with new data; otherwise, x^t is added to (W_{ld}).

Algorithm 2: Pseudocode of Diversity Measure as a Drift Detection Method in a Semi-Supervised Environment (DMDDM-S)

Required: S: data stream of examples (labeled), α: forgetting factor (0 < α < 1), δ: admissible change (e.g., δ = 0.01), λ: drift threshold, Classifiers (Hoeffding Tree, Perceptron): L = 2, Buffer sizes: |W_ld| = 100, |W_uld| = 100, Ensemble components: b,c = 0, M_T : 1.0D,

Output: Drift ∈ {TRUE, FALSE}

1 For each example $x^t \in S$ do
2. Obtain predictions C_v and C_u using x^t
3. If predictions differ ($C_v = 0$ *and* $C_u = 1$ or $C_v = 1$ *and* $C_u = 0$):
4. Increment disagreement counter b_c
5. Calculate disagreement ratio $D_{u,v} = b + c /l$
6. $S_{u:v,\alpha}(t) = D_{u,v} + \alpha + S_{u:v,\alpha}(t\text{-}1)$
7. $N_\alpha(t) = 1 + \alpha + N_\alpha(t\text{-}1)$
8. $M_\alpha(t) = S_{u:v,\alpha(t)} / N_\alpha(t)$
9. *Sumdiversity* = *SumDiversity* + $M_\alpha(t)$
10. Calculate current $m_T = m_T + M_\alpha + -$ (SumDiversity / instancesSeen) - δ)
11. $M_T = min(M_T, m_T)$
12. Compute $PH_{test} = m_T - M_T$
13. if $PH_{test} > \lambda$ (Return TRUE)
14. Manage buffer W_{ld} by updating with W_{uld}
15. $N_{ld} = K$ – *PrototypeClustering* (W_{ld})
16. Incrementally train classifiers C_v and C_u with x^t
17. If buffer W_{uld} is full, manage by removing oldest instances
18. End for

C. DMODD as an Online Drift Detector in a Fully Supervised K-Class Problem Context.

Initially, for binary classification (DMDDM, DMDDM-S), the disagreement measure $D_{v,u}$ (Eq. 1) was used. However, for multi-label classification, Table 1's method for distinguishing discrepancies between classifier pairs that misclassify the same instance with different labels is ineffective. This study refines the approach to track precise classifier predictions beyond binary correct/incorrect assessments. We introduce a contingency table $C_{i,j}$, which records instances x ∈ X where classifier $h_v(x) = i$ and classifier $h_u(x)$, as shown in Table 2 for multi-label issues. Aligned classifier pair decisions are cataloged along the diagonal of $C_{i,j}$. Eq. 13 calculates their similarity by summing these diagonal values and dividing by the total instance count n. To indicate potential drifts, we apply the PH test, as described in Eq 10 and 12 from our preliminary findings.

Table 2. Output of a Pair of Classifiers for the Multi-class Problem

	$h_u(x) = 0$	$h_u(x) = 1$	...	$h_u(x) = (k-1)$
$h_v(x) = 0$	C_{00}	C_{01}	...	$C_{0(k-1)}$
$h_v(x) = 1$	C_{10}	C_{11}	...	$C_{1(k-1)}$
...	...	...	...	...
$h_v(x) = (k-1)$	$C_{(k-1)0}$	$C_{(k-1)1}$	...	$C_{(k-1)(k-1)}$

A drift is suggested when this disparity exceeds a predetermined threshold (λ).

$$\theta = \frac{1}{n}\sum_{i=0}^{K}(C_{i,j}) \tag{13}$$

The DMODD, referenced in [7] and detailed in **Algorithm 3**, processes data stream examples and evaluates classifier predictions. It uses a contingency table to tally diverging classifier decisions (lines 1-3). This approach aggregates and normalizes these tallies by the total instance count (lines 4-5). It then calculates the fading sum, increment, and average (lines 6-8). The method tracks classifier diversity using the PH test to monitor cumulative variation (lines 9-14). A drift is identified and addressed when the disparity between current and minimum cumulative variations exceeds a threshold. The algorithm concludes with the incremental model update (line 15), keeping it responsive to new data and identified drifts.

Algorithm 3: DMODD: Diversity Measure as Online Drift Detector

Required: *S: A stream of labeled examples, Forgetting factor* α: 0 < α < -1, *Admissible change*: δ = 0.1, *Threshold*: λ = 100, *M_T*: 1.0D

Output: Drift can be either True or False.

1 **for** *each example* x^t in *S* **do**
2 *Cv* = obtain prediction using x^t;
3 *Cu* = obtain prediction using x^t;
4 *P* [*Cv prediction*] [*Cu prediction*] + +
5 *Sum*+ = *P*[i][i]
6 $Su{:}v,\alpha(t) = Sum + \alpha \times Sum\ (t-1)$;
7 $N\alpha(t) = 1 + \alpha \times N\alpha\ (t-1)$;
8 $M\alpha(t) = \frac{S_{u:v,\alpha}\ (t)}{N_u\ (t)}$;
9 *SumDiversity* = *SumDiversity* + $M_\alpha(t)$;
10 $m_T = (m_T + M_\alpha(t) - (SumDiversity/instancesSeen) - \delta)$;
11 $M_T = Min\ (M_T, m_T)$;
12 $PHtest = m_T - M_T$;
13 **if** *PHtest* > λ **then**
14 Return TRUE
15 incrementally train *Cv* and *Cu* with *x*;
16 **end**

D. HBBE is a Hybrid Block-Based Ensemble designed to address different types of concept drifts.

The examination of block-based ensembles and drift detection strategies enhances understanding of adaptive block-based ensembles and online drift detection, specifically their mechanisms for responding to concept drift. Consequently, a Hybrid Block-Based Ensemble (HBBE) framework has been developed and empirically proven [6]. This model outperforms other leading adaptive learning algorithms in predictive accuracy across various scenarios, including sudden, gradual, recurring, and multi-class challenges. HBBE merges an online drift detector (DMODD), tailored for K-class problems and capable of real-time processing, with block-based weighting to effectively address diverse forms of drifts. The operational mechanism of the block-based ensemble framework is characterized as follows: **(1)** An Online Drift Detector, tailored for K-class problems, processes data instance by instance to promptly enhance the ensemble's responsiveness to sudden drifts. **(2)** In a Block-Based manner, following every *d* examples, an evaluation is conducted alongside incremental updates to the ensemble's components, plus the integration of a new component. This process is aimed at bolstering the ensemble's ability to adapt to gradual drifts. **(3)** Upon drift detection by the online

detector, a new classifier (nominee) is constructed using the latest instances. This nominee is weighted and incorporated into the ensemble based on a specific criterion $\theta()$. Post-drift, existing components of the ensemble are re-adjusted in terms of their weight. **(4)** During periods of stability, where no drifts are identified, the framework operates akin to a conventional block-based ensemble. **(5)** The decision outputs from both the online drift detector and batch learners are amalgamated through a weighted majority vote, employing a measure of suitability to guide the weighting.

Algorithm 4 integrates a block-based ensemble with an online drift detector. It processes data stream examples one at a time (line 1) and accumulates them in a buffer with a defined capacity d (line 2). The online drift detector from Algorithm 1 is then applied (line 3). Upon detecting a drift or when the buffer is full, a new classifier is built using the latest buffer examples and assigned a weight (line 4). Weights are assigned to all ensemble components based on the buffer (line 5).

If the ensemble has fewer than k classifiers, the new classifier is added (lines 7-8); if it already has k classifiers, the least effective one is replaced (lines 9-10). This method swiftly adapts to sudden and gradual changes in the data stream.

Algorithm 4: Pseudocode of the Block-Based Ensemble and Drift detector (HBBE)

Required: S: data stream of examples (labelled), D: drift detector, k: number of ensemble members, B: example buffer of size d, (Q): classifier quality measure, t: example number
Mr: 1.0D
Output: E: ensemble of k weighted classifiers and 1 classifier with a drift detector

```
1   for each example x_t ∈ S do
2     B ← B ∪ {x_t}
3     if drift detected (Algorithm 1) OR |B| = d then
4       build and weight candidate classifier C' using B and Q (6);
5       weight all classifiers C_i in ensemble E using B and Q (8);
6     end
7     if |E| < k then
8     E ← E ∪ {C'}
9     else
10      if Q()' > Q() then
11      replace weakest ensemble member with C;
12      end
13    end
14  end
```

V. RESULTS AND ANALYSIS.

The performance of the four methods (DMDDM, DMDDM-S, DMODD, and HBBE) is evaluated through experiments using various concept drift detection methods like FHDDM, DDM, ADWIN, Wtest, PH Test, SeqDrift, Atest, STEPD, SEED, RDDM, and EDDM. Additionally, HBBE is compared with ensemble approaches such as AUE2, AWE, DWM, and Learn++.NSE. For full details of the literature and experiments settings of each method refer to the original works in [29].The following section details these experiments, evaluations, and findings. In addition, Table 3 shows the abbreviations of the measures that will be used during the analysis. To simplify the comparative analysis of each method, we will use the Weighted Sum Model (WSM), a multi-criteria decision-making method [30], [31]. WSM involves transforming and weighting each metric, then summing them to create a single score for each method. This score will represent the balance between all metrics and will be visualized using a bar chart.

The process includes:

- Normalizing each metric between 0 and 1.
- Assigning weights to each metric based on importance.
- Computing a single score for each method.

The method with the longest bar on the chart offers the best balance across metrics, with longer bars indicating better overall performance. This visualization helps easily identify which method optimally balances all the metrics.

Table 3: Abbreviations of Measures Used

Metric	Abbreviation	Definition
Average Delay Detection	ADD	Average time taken to detect a drift after it occurs
Average True Detection	ATD	Average rate of correctly identifying true drifts.
Average False Alarm	AFA	Average rate of incorrect warnings or alerts
Average False Negative	AFN	Average rate of missed detections or unflagged true drifts.
Average Detection Runtime (MS)	DRMS	Average time taken to complete a detection process, measured in milliseconds
Average Memory Usage (Byte)	MUB	Average amount of memory used during a process, measured in bytes.
Average Accuracy	ACC	Average rate at which a system correctly identifies or predicts both true and false events.

A. *RQ1*

To address RQ1, we must evaluate each method in relation to its counterparts. The next paragraphs show the results of the experiments and the analyses of each drift detector.

In the comparative visualization across the SEA and Sine1 datasets, detectors exhibit varied performance, as shown in Figure. 3. DMDDM stands out, particularly in Average True Detection (ATD) and Average False Negative (AFN) metrics, consistently scoring optimally. It ensures swift event detection, with low Average Delay Detection (ADD) values, especially in the SEA dataset. However, DMDDM shows a relatively higher Average False Alarm (AFA) in SEA, indicating a need to reduce false alerts. Other detectors like FHDDM and ADWIN show trade-offs between true detection and false alarms. For instance, ADWIN maintains a low ADD in Sine1 but has higher false alarms, impacting overall accuracy.

DMDDM achieves the highest Weighted Sum Model (WSM) scores, indicating balanced performance across all metrics, as shown in Figure 4. Its lower ADD and minimal false alarms/negatives (AFA and AFN) contribute significantly to this superior score. While DMDDM excels in rapid, accurate change detection, methods like FHDDM and Atest also perform well, especially in the Sine1 dataset, highlighting their potential in specific contexts. Conversely, ADWIN and PH Test generally reflect lower WSM scores, suggesting areas for improvement in detection speed, accuracy, or resource efficiency. Optimizing these models could enhance their applicability across varied data-streaming environments.

The visualization provides a comparative analysis of various drift detectors, including DMDDM-S, across the Sine1 and Mixed datasets. Figure 5 shows that DMDDM-S consistently performs well across most metrics, demonstrating robustness in diverse contexts. It has the lowest detection delay and minimal memory usage, making it suitable for real-

time applications with memory constraints. Although SEED and FHDDM also show low delay, DMDDM-S stands out for its balance between low delay and memory efficiency, optimizing computational resources. While SEED excels in Mean Accuracy, DMDDM-S has notable accuracy, especially in the Sine1 dataset. This balanced performance highlights DMDDM-S's viability as a drift detector, offering timely detection, memory efficiency, and good accuracy for dynamic environments. Further examination of specific use cases will enhance understanding of DMDDM-S's applicability. Figure 6 shows that DMDDM-S consistently achieves the highest WSM scores across both datasets, due to its low delay, efficient time usage, and commendable accuracy.

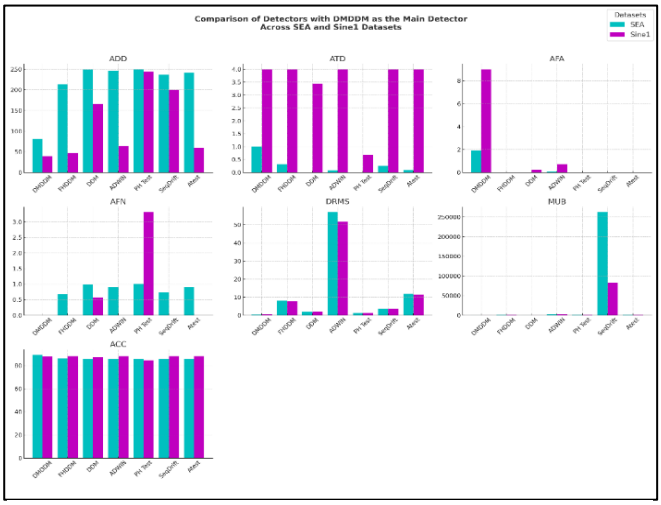

Figure 4. Comparison of detectors including DMDDM.

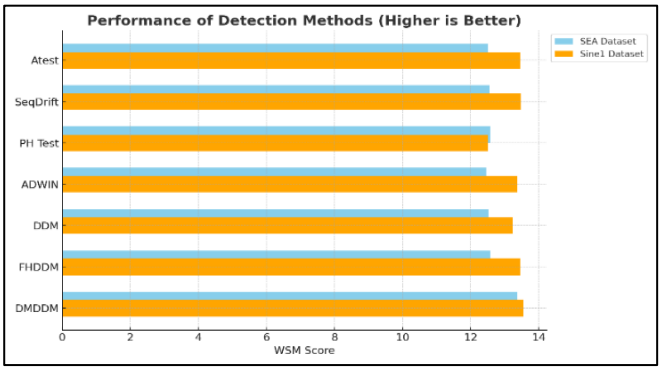

Figure 3. The balanced score for each method including DMDDM.

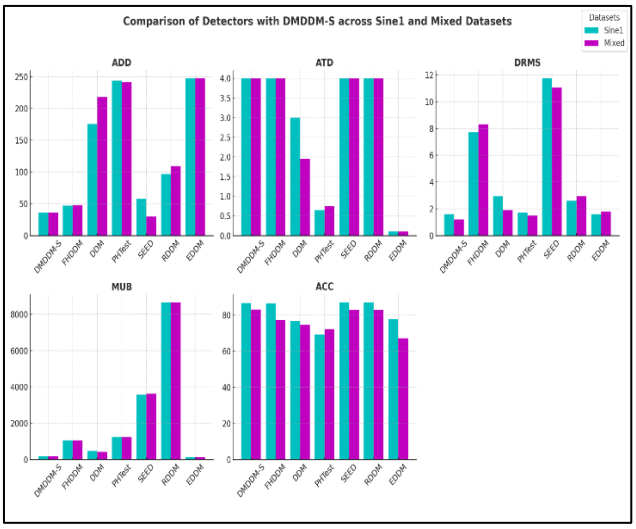

Figure 5. Comparison of detectors including DMDDM-S.

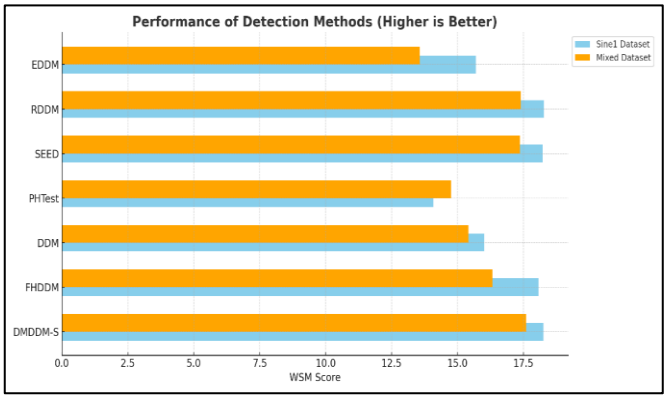

Figure 6. The balanced score for each method including DMDDM-S.

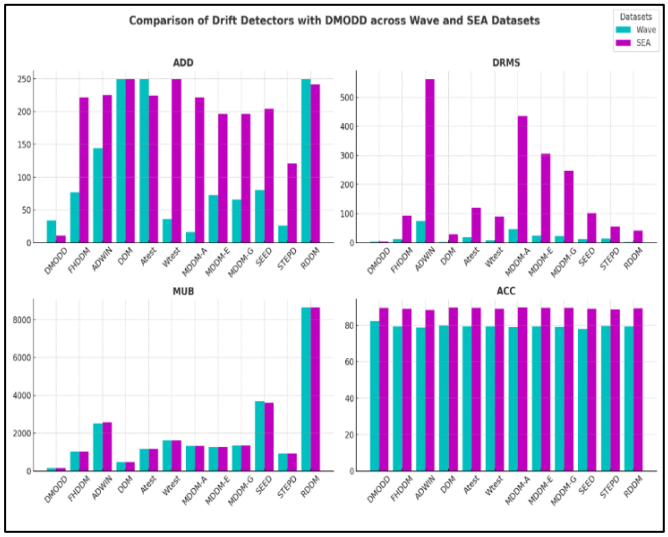

Figure 7. Comparison of detectors including DMODD.

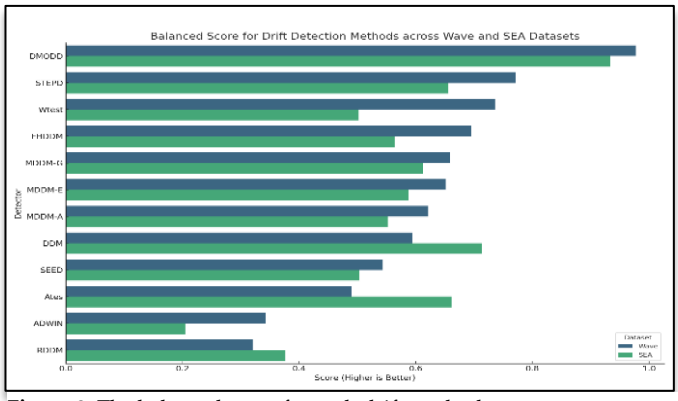

Figure 8. The balanced score for each drift method

DMDDM-S stands out for its efficient performance with low memory usage, balancing computational resources. While methods like SEED and RDDM perform well, particularly on the Sine1 dataset, they require more memory. PHTest and EDDM, on the other hand, show lower overall performance in terms of accuracy, delay, and time. Figure 7 compares drift detectors, with DMODD excelling in detection delay (ADD) across both the Wave and SEA datasets, showcasing its quick and efficient drift detection. datasets, indicating its efficiency in identifying concept drifts promptly.

Although DMODD does not lead in accuracy (ACC), its balanced performance across metrics, especially in minimizing DRMS and MUB, highlights its efficiency. While

detectors like MDDM-A also show low ADD in the Wave dataset, DMODD consistently performs well across both datasets, demonstrating its reliability.

Figure 8 shows that DMODD achieves a strong balance across all metrics in the SEA dataset, excelling in detection speed, runtime, memory efficiency, and accuracy. It remains competitive in the Wave dataset, showcasing its adaptability.

Figure 9 evaluates AUE2, DWM, HBBE, AWE, NSE, and DWM across multiple datasets. HBBE consistently shows strong accuracy, especially in the Wave and SEA datasets, indicating its reliable predictive capabilities.

However, it consumes more runtime and memory, particularly in the 'Airline' dataset. DWM consistently uses less memory and has a more favorable runtime, though sometimes at the cost of accuracy. NSE exhibits high runtime and memory usage in certain datasets like 'RBFGR', making it less suitable for scenarios with limited resources. Thus, HBBE is robust in accuracy but requires careful consideration due to its computational demands.

Figure 10 shows the Weighted Sum Model (WSM) scores for the five algorithms across multiple datasets. HBBE, with a WSM score of 0.550, demonstrates balanced performance in accuracy, runtime, and memory usage, making it suitable for varied contexts. DWM excels with the highest WSM score of 0.930, indicating an optimal balance. Conversely, NSE, with the lowest WSM score, may need optimization or could be used in specific scenarios where its strengths are advantageous. For specific metric-driven decisions, a deeper analysis of individual performances is essential.

Each methodology uses diversity measures uniquely for concept drift detection. DMDDM excels in fully supervised environments, offering high precision and efficiency, especially with significant drifts, and maintains high WSM scores in binary classification. DMDDM-S, ideal for semi-supervised settings, balances accuracy and resource use, performing well with limited labeled data. HBBE, tailored for multi-class challenges, adapts to various drift types and delivers exceptional accuracy but requires careful consideration of runtime and memory in resource-limited environments. Overall, each method shows distinct strengths and adaptability in detecting concept drift across different scenarios.

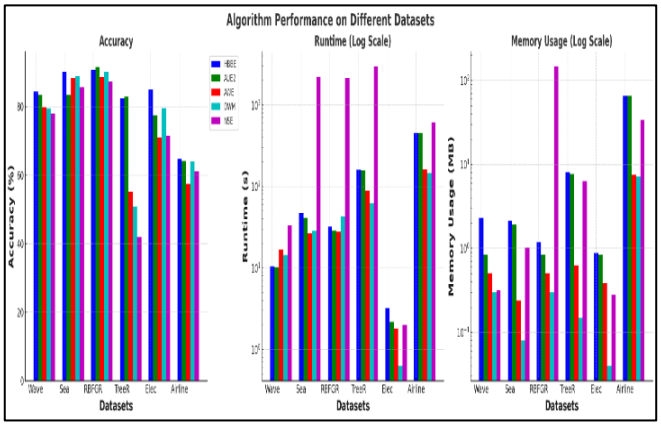

Figure 9. Comparison of detectors including HBBE.

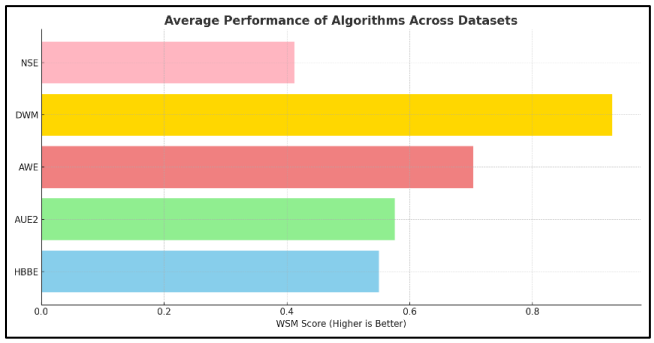

Figure 10. The balanced score for each drift detection method.

B. RQ2

To address RQ2, the comparative bar chart shows diverse WSM scores for DMDDM, DMDDM-S, DMODD, and HBBE across different datasets and scenarios. Figure 11 provides a succinct yet contextually varied overview of the algorithms' performances, requiring nuanced interpretation.

DMODD, with a WSM score of 20.72, appears to outperform the others, indicating a superior balance of accuracy, runtime, and memory usage. However, it's important to note that the scores originate from different analyses, potentially involving varied metrics and weights. Both DMDDM-S and HBBE have WSM scores of 0.55, reflecting similar balanced performances, while DMDDM has a score of 0.30, trailing slightly. This highlights the need to consider specific analytical contexts when interpreting these scores and deploying algorithms, ensuring alignment with the data's demands and characteristics.

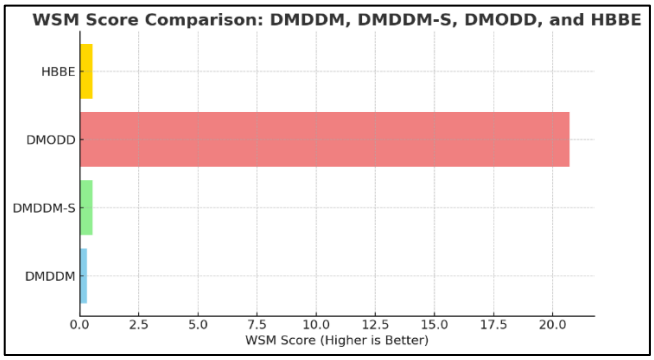

Figure 11. The WSM balanced score for each of the four diversity measures.

C. RQ3

To address RQ3, this subsection discusses the advantages, assumptions, and constraints of using diversity measures for concept drift handling.

The diversity measure is efficient in computational time and memory, relying on fewer variables compared to sliding window methods, which reduces memory use and update times. Most drift detectors operate with constant time complexity, but exceptions like ADWIN and SeqDrift have logarithmic complexity. However, the diversity measure has a higher false alarm rate due to parameter sensitivity. In DMDDM, increasing γ (100-200-300) reduces false alarms but may delay or miss changes, balancing between minimizing false alarms and avoiding delays. Despite higher detection delays, DMDDM has low time and memory requirements, thanks to the PH test. This could be improved by using dual fading factors or integrating fuzzy decision-

making for more nuanced detection. HBBE excels in accuracy, particularly in sudden drifts, though its runtime performance is affected by ensemble handling. Future research could focus on:

- Integrating drift detection in IDS for dynamic model adjustments in anomaly detection [32].
- Enhancing drift detection in Big Data to improve accuracy while reducing computational demands [33].
- Developing scalable drift detection methods for IoT, focusing on distributed algorithms for real-time processing [34].

These areas highlight the potential for further research and improvement in drift detection methodologies.

VI. CONCLUSION

Diversity measures provide adaptability to various drift types and robustness across datasets, making them effective and reliable for detecting concept drift. They offer a balanced combination of accuracy, adaptability, and resource efficiency, making them a preferred choice in scenarios requiring an understanding of evolving data patterns. The Weighted Sum Model (WSM) scores show that algorithms using diversity measures, like DMDDM, perform well in terms of accuracy, runtime, and memory usage, especially in binary classification.

References

[1] J. Gama, *Knowledge discovery from data streams*. CRC Press, 2010.

[2] O. A. Mahdi, N. Ali, E. Pardede, A. Alazab, T. Al-Quraishi, and B. Das, "Roadmap of Concept Drift Adaptation in Data Stream Mining, Years Later," *IEEE Access*, 2024.

[3] B. R. Prasad and S. Agarwal, "Stream data mining: platforms, algorithms, performance evaluators and research trends," *International journal of database theory and application*, vol. 9, no. 9, pp. 201–218, 2016.

[4] O. A. Mahdi, E. Pardede, N. Ali, and J. Cao, "Diversity measure as a new drift detection method in data streaming," *Knowl Based Syst*, vol. 191, p. 105227, 2020.

[5] O. A. Mahdi, E. Pardede, N. Ali, and J. Cao, "Fast reaction to sudden concept drift in the absence of class labels," *Applied Sciences*, vol. 10, no. 2, p. 606, 2020.

[6] O. A. Mahdi, E. Pardede, and N. Ali, "A hybrid block-based ensemble framework for the multi-class problem to react to different types of drifts," *Cluster Comput*, vol. 24, pp. 2327–2340, 2021.

[7] O. A. Mahdi, N. Ali, E. Pardede, and T. Al-Quraishi, "Online Concept Drift Detector: Optimally Balancing Delay Detection, Runtime, Memory, and Accuracy.," *Procedia Comput Sci*, vol. 237, pp. 559–567, 2024.

[8] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[9] P. E. Hart, D. G. Stork, and R. O. Duda, *Pattern classification*. Wiley Hoboken, 2000.

[10] A. Pesaranghader and H. L. Viktor, "Fast hoeffding drift detection method for evolving data streams," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16*, 2016, pp. 96–111.

[11] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*, 2004, pp. 286–295.

[12] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*, 2007, pp. 443–448.

[13] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-D\'\iaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on Hoeffding's bounds," *IEEE Trans Knowl Data Eng*, vol. 27, no. 3, pp. 810–823, 2014.

[14] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognit Lett*, vol. 33, no. 2, pp. 191–198, 2012.

[15] J. Gama, R. Sebastiao, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach Learn*, vol. 90, pp. 317–346, 2013.

[16] R. Pears, S. Sakthithasan, and Y. S. Koh, "Detecting concept change in dynamic data streams: A sequential approach based on reservoir sampling," *Mach Learn*, vol. 97, pp. 259–293, 2014.

[17] D. T. J. Huang, Y. S. Koh, G. Dobbie, and R. Pears, "Detecting volatility shift in data streams," in *2014 IEEE International Conference on Data Mining*, 2014, pp. 863–868.

[18] R. S. M. Barros, D. R. L. Cabral, P. M. Gonçalves Jr, and S. G. T. C. Santos, "RDDM: Reactive drift detection method," *Expert Syst Appl*, vol. 90, pp. 344–355, 2017.

[19] M. Baena-Garca, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth international workshop on knowledge discovery from data streams*, 2006, pp. 77–86.

[20] B. Krawczyk and M. Woźniak, "Reacting to different types of concept drift with adaptive and incremental one-class classifiers," in *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*, 2015, pp. 30–35.

[21] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226–235.

[22] J. Kolter and M. Maloof, "Dynamic weighted majority: A new ensemble method for tracking concept drift," in *Int'l Conf. Data Mining (ICDM)*, 2001.

[23] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans Neural Netw*, vol. 22, no. 10, pp. 1517–1531, 2011.

[24] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A new ensemble diversity measure applied to thinning ensembles," in *Multiple Classifier Systems: 4th International Workshop, MCS 2003 Guildford, UK, June 11–13, 2003 Proceedings 4*, 2003, pp. 306–316.

[25] G. Giacinto and F. Roli, "An approach to the automatic design of multiple classifier systems," *Pattern Recognit Lett*, vol. 22, no. 1, pp. 25–33, 2001.

[26] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, 1997, pp. 211–218.

[27] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans Knowl Data Eng*, vol. 22, no. 5, pp. 730–742, 2009.

[28] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[29] O. A. *Mahdi*, "Diversity Measures as New Concept Drift Detection Methods in Data Stream Mining," La Trobe University Melbourne, Australia 9, 2020.

[30] H. Taherdoost and M. Madanchian, "Multi-criteria decision making (MCDM) methods and concepts," *Encyclopedia*, vol. 3, no. 1, pp. 77–87, 2023.

[31] J. J. Thakkar, *Multi-criteria decision making*, vol. 336. Springer, 2021.

[32] O. A. Mahdi, A. Alazab, S. Bevinakoppa, N. Ali, and A. Khraisat, "Enhancing IoT Intrusion Detection System Performance with the Diversity Measure as a Novel Drift Detection Method," in *2023 9th International Conference on Information Technology Trends (ITT)*, 2023, pp. 50–54.

[33] R. Seraj and M. Ahmed, "Concept drift for big data," *Combating Security* Challenges *in the Age of Big Data: Powered by State-of-the-Art Artificial Intelligence Techniques*, pp. 29–43, 2020.

[34] F. E. Casado, D. Lema, M. F. Criado, R. Iglesias, C. V Regueiro, and S. Barro, "Concept drift detection and adaptation for federated and continual learning," *Multimed Tools Appl*, pp. 1–23, 2022.