



# A Novel Hybrid Concept Drift Detector Ensemble for Handling Pure and Hybrid Drift in Non-Stationary Data Streams

Muralikrishnan Ramane<sup>1\*</sup> and Gnanasekar J M<sup>2</sup>

## Abstract

Cognitive analytics deals with complex problems that arises due to dynamically evolving environments. Online machine learning methodologies learn continuously from nonstationary and evolving data streams and can handle dynamism in near real time. Evolving data can lead to change or drift in learned concepts and data distributions due to population variations. The literature presents several drift detection techniques which identify drift as real or virtual and are not sophisticated to detect pure and hybrid drift types. Past methods monitor drift in a disconnected fashion and this disconnection leads to imprecise and redundant model update requests. The main objective of this work is to detect concept drift and differentiate its types precisely, thus improving the quality and robustness of the drift handling process. This work proposes a novel hybrid concept drift detector ensemble, that utilizes a connected drift detection method for detecting pure and hybrid drifts. Further, a parallel framework is introduced to monitor input data and learner output concurrently. Experiments were conducted separately to evaluate individual and connected techniques using synthetic datasets of one million samples containing abrupt and gradual drift. This work utilizes three leaner error-ratebased methods along with the 2-Sample Kolmogorov-Smirnov test for finding concept dissimilarities. The proposed hybrid technique utilizes parallel processing to reduce computation time and performs better in aspects of drift detection, drift type differentiation, and precise model update requests.

6170

**Keywords:** Concept Drift Detection, Ensemble Technique, Machine Learning, Cognitive Systems

**DOI Number:** 10.14704/nq.2022.20.8.NQ44641

**NeuroQuantology** 2022;20(8):6170-6186

## Introduction

Cognitive systems are the class of machines that self-learn and adapt to dynamic environments using machine intelligence. These systems can identify patterns and drifts in the environment and incrementally adapt to situations in near real time. Modern big data architectures

generate huge volumes of streaming data in high velocity and are categorized into static and evolving data streams. Evolving data (Khamassiet *al*, 2018;

Kremplet *al*, 2014) e.g., stock market data, sensor data, etc., gets constantly updated



because the underlying process that generates it can change over time. Detecting and quantifying such change are fundamental challenges in data stream settings. To analyse

such data, highly sophisticated techniques are required to handle dynamism (Ditzler *et al*, 2015; Tran *et al*, 2014) and uncertainty.

---

\***Corresponding author:** Muralikrishnan Ramane, Email: murali.itpro@gmail.com

<sup>1</sup>Teaching Fellow, Department of Information Technology, University College of Engineering Villupuram,

Kakuppam, 605 103, India. Email: murali.itpro@gmail.com

<sup>2</sup>Professor, Department of Computer Science & Engineering, Sri Venkateswara College of Engineering, Sriperumbudur, Tamilnadu, India – 602 117, Email: jmg\_sekar@yahoo.com

---

Drift analysis (Tsymbal, 2004; Widmer *et al*, 1996; Zliobaite *et al*, 2016) is a long-standing process in machine learning where change is rampant. Several real-world scenarios like the current global pandemic can cause errors in machine learner's estimates, thus leading to more unreliable forecasts. In several real-world applications, systems take intelligence-driven decisions and so are dependent on machine learner's reliability. It is a fact that "the reliability of a machine learner is inversely proportional to input data uncertainty i.e., misleading data."

Concept drift is defined as (Moreno-Torres *et al*, 2012; Webb *et al*, 2016) the non-stationarity of data and its altering relationship with the target class over time. The two major types of concept drift are class/real concept drift and covariate/virtual concept drift. It is categorized as abrupt and gradual drift based on drifting duration or drift width. Drift detection is the most important phase in the drift handling process. It is a change detection technique (Kifer *et al*, 2004; Patil, 2019; Ramirez-Gallego *et al*, 2017; Sebastiao *et al*, 2009) that looks for deviations in learner performance or input data distribution.

The drift can occur in three forms, as described in the paper (Lu *et al*, 2018) and can occur from three different types of sources. Source I, II, and III generates virtual drift, real drift, and hybrid drift respectively. The drift is categorized into the above types based on how the posterior probability and the probability of independent variables change between the test and training data. The drift detection process in streaming

data consists of four stages of operation, such as I. Data Retrieval, where data is received one at a time and organized into data chunks using windowing techniques. II. Data Modelling, here the data is abstracted, and key features containing sensitive information are extracted. III. Statistical Calculation, where test statistics calculation, i.e., the dissimilarity measurement or distance estimation between adjoining time intervals, is performed and at last IV. Hypothesis Testing, which confirms drift, if the change is statistically significant. Most drift detection methods operate in this fashion and follow a generalized framework as discussed in the paper (Lu *et al*, 2018; Agrahari *et al*, 2021) to handle drift.

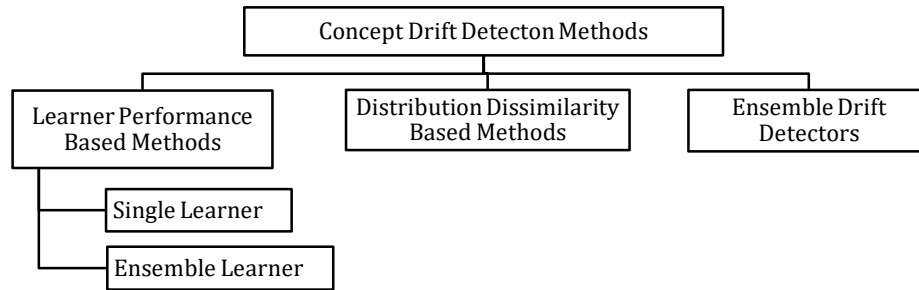
This paper provides three major contributions to support robust drift detection and handling. This paper proposes a novel hybrid concept drift detector ensemble which enables the detection of pure and hybrid concept drifts. Further, this paper suggests a parallel drift analysis framework which utilizes a novel connected drift detection method to monitor learning model and data distribution concurrently. Finally, this work introduces a data store where, the drift occurrence data is preserved as event logs for future analysis. The remainder of this paper is structured as follows. Section 2 presents a literature review on past drift detection techniques, discusses current issues and defines the objectives. Section 3 presents the proposed hybrid ensemble and the parallel connected drift handling framework. Section 4 discusses experiments conducted and their results along with graphs, and Section 5

concludes the paper.

### Related Works

Drift detection methods proposed in the past follow diverse techniques which can be grouped based on Dissimilarity Analysis, Sequential Analysis, Window Monitoring, Statistical Analysis, Significance Analysis, Data Distribution, Decision Boundary, and Model Dependency.

These categories are generalized based on how the drift is detected, what metric is utilized to confirm drift and what is monitored. The two most popular systems of measurements used for monitoring drift are, Learner Performance and Input Data Distribution Dissimilarity. As shown in Fig. 1, the survey is split into three sections.



**Figure 1.** A Review of Concept Drift Detection Methods

This work focuses on ensemble drift detectors that utilize multiple drift detection methods and combine them using various strategies to achieve superiority in drift detection. This survey includes detection methods that operate on streaming data settings and has neglected some disparate works

### Learner Performance Based Methods

Learner performance-based drift detection methods (LPBM) are capable of detecting only real or class drift. This section of the survey discusses single learner-based methods, that are popular and included in past surveys, authored by (Barros *et al*, 2018; Dehghanet *al*, 2016; Goncalves *et al*, 2014). The foremost and most popular technique is Drift Detection Method, DDM which uses a base classifier to monitor the overall online error rate, where a new learner is built when the warning level is reached and replaces the old learner when drift is confirmed. A reactive version of DDM, RDDM overcomes the performance loss problem due to decreased sensitivity by discarding older instances. Early Drift Detection Method (EDDM) in comparison to DDM has increased efficiency and confirms drift, when the distance between two classification errors

increases. The fuzzy DDM (FW-DDM) improves on DDM using fuzzy time windows. The EWMA for Concept Drift Detection (ECCD) employs an EWMA chart to track changes and compares the estimated online error rate and theoretical error to confirm drift. An online dynamic extreme learning machine (DELME) uses a double-hidden single-layered neural network to detect concept drift. When the accuracy of the ELM gets below the threshold, the current classifier is updated.

Statistical test-based algorithms such as STEPDP compare the overall accuracy of the base learner with the accuracy of recent instances. Fisher's Exact Test-based method (FTDD) improves STEPDP's efficiency, when either data samples or the number of errors is small. WSTD is similar to STEPDP but uses two windows of data to detect drift and uses Wilcoxon Rank Sum statistical test. Hoeffding's Inequality based methods HDDM and FHDDM monitor the performance of the base learner where the former uses some probability inequalities on distributional changes and the latter compares the recent probability of correct predictions with maximum probability.

### Ensemble Learner Performance Based Methods



Ensemble Learner's Performance-based methods (ELPBM) detect drift efficiently by improving classifier performance using multiple learners combined into a single ensemble. These types of methods are also only capable of detecting real drift since they do not monitor data distribution changes. This section of the paper presents past online ensemble methods included in review papers (Barros et al, 2019; Krawczyk et al, 2017). Online ensembles are divided into passive or active categories, based on whether the drift detectors are implicit or explicitly available.

Dynamic Weighted Majority (DWM), assigns weight to classifiers and is reduced by a multiplicative constant for a wrong prediction. When the ensemble misclassifies a new classifier is added. Addictive Expert Ensembles (AddExp), is similar to DWM which adds a new classifier after updating the weight of the old classifier and when the ensemble reaches the pre-defined size, the oldest classifier is pruned. Horse Racing Ensemble (HRE), utilizes a one-component classifier that represents the final prediction and is selected based on the probability distribution. Concept Drift Committee (CDC) uses weights to represent the classifier's accuracy and adds a new classifier when new samples arrive. Online Accuracy Updated Ensemble (OAUE) incrementally updates component classifiers instead of learning new classifiers. Anticipative Dynamic Adaptation to Concept Changes (ADACC) recognizes concepts from incoming samples and optimizes the control over online classifiers. The WWH algorithm utilizes an adaptive ensemble and selects the best learner by aggregating component predictions, similar to the weighted majority algorithm.

Active approaches do explicit drift detection and quickly react to concept drifts. These are unable to react to drifts when detectors fail. Adaptive Classifiers-Ensemble (ACE) uses both online and batch classifiers to learn from new and old examples respectively. Drift is confirmed if the difference in accuracy is outside the confidence interval between the

last and preceding  $W$  examples. An online classifier-based method (TODI), uses two classifiers to detect concept drift. When drift is confirmed any one of the classifiers is rebuilt and replaces the current model. It detects drift by utilizing the statistical test of equal proportions on the accuracy of examples. A diversity-based drift detection technique (DDD) utilizes a low diversity ensemble for learning and post drift detection uses a high diversity ensemble for learning and predictions.

### *Data Distribution Dissimilarity Based Methods*

6173

Data distribution dissimilarity-based drift detection methods (DDBM) are capable of detecting virtual drift with the capabilities to easily find where and how severe the input data has drifted. This section of the review includes papers from existing review papers (Goldenberg et al, 2019; Sobolewski et al, 2013; Wang et al, 2020). These methods are relatively costlier since it calculates the distance between distributions of all relevant independent variables. The methods are categorized based on statistical distance measurements, total variation, statistical change, competence models, and density estimations.

The foremost research work Relativized Discrepancy is based on a distance function called total variation which calculates the distance between distributions to confirm the change. Hellinger Distance is a feature-based drift detection method, that compares current and reference data distribution and calculates divergence between them to confirm drift. Kullback-Leibler Divergence is an information theory-based measure of disparity. It compares the log difference between the probability of data in the original distribution with the approximating distribution. Kolmogorov-Smirnov Statistic finds the significant upper bound of difference between two adjacent empirical distributions and confirms drift when the  $p$ -value reaches the threshold. Wilcoxon Rank Sum Test, a non-parametric alternative to the two-sample  $t$ -test, compares the ranks of



pairs of distributions to the significance table to confirm drift.

Density-based drift detection methods detect drift by estimating the density of distributions and comparing them to confirm drift. kdqTree is a density-based drift detection algorithm that uses Kullback-Leibler divergence, to quantify the difference. Equal Density Estimation is a non-parametric method based on the estimation of equal density regions. Statistical Change Detection technique (SCD) uses a kernel density estimator to compare observed data with the baseline dataset for dissimilarity detection. Principal Component Analysis-based method (PCA) uses density estimators with dynamic divergence metrics to compare the data distributions. Competence Model-based Drift

Detection technique uses competence measurement to find dissimilarity between distributions. Least Squares Density Difference LSDD-CDT and LSDD-INC (incremental), both are probability density function-free method that monitors the difference between two non-overlapping data windows to estimate the difference between the two pdfs.

### Ensemble Drift Detectors

This section of the survey discusses some closely related research works that propose similar techniques which combine multiple drift detectors to achieve better drift detection. The existing ensemble drift detectors are listed in Tab. 1.

**Table 1.** List of Ensemble Drift Detectors

S.No	Ensemble Method	Detectors Combined	Drift Type
1.	OGMMF-VRD	Online Gaussian Mixture Model with Noise Filter	Real and Virtual
2.	STED	Statistical Tests (Brown-Forsythe, O'Brien, and ANOVA) combined with a Two-Voting Strategy	Virtual
3.	EDFS	Incremental Kolmogorov-Smirnov Test, LDCNet, EHCDD	Virtual
4.	CCDD	DDM, HDDM <sub>A</sub> , HDDM <sub>w</sub> , CUMSUM, and Page-Hinkley with combination rules (ALO, ALHD, AD)	Real
5.	e-Detector	DDM, EDDM, ADWIN, STEP, and EnDDM with Early-Find Early-Report Rule	Real
6.	DDE	HDDM <sub>A</sub> , HDDM <sub>w</sub> , DDM, ECDD, and ADWIN combined in 3 different strategies	Real

OGMMF-VRD (Oliveira et al, 2021) is an ensemble of drift detectors that utilize an Online Gaussian Mixture Model with a Noise Filter. It analyses the impact of both virtual and real drifts on classifiers and proposes an unsupervised/supervised methodology to train the GMM and achieve better robustness to noise. Experiments using both synthetic and real-world datasets achieved the best results in terms of runtime, G-Mean, and average accuracy.

STED (Perez et al, 2020) is an ensemble of concept drift detectors that combines statistical tests such as Brown-Forsythe, O'Brien, and ANOVA to improve drift detection efficiency. A

voting strategy is utilized to compare results and signal warnings with a majority voting technique to detect concept drifts. This technique used Hoeffding Tree as a base learner and proved its efficiency by comparing 24 artificial and 7 real-world datasets.

EDFS(Korycki et al, 2019) is a feature subspace-based ensemble for drift detection. It is an unsupervised technique that recognizes local changes and uses incremental Kolmogorov-Smirnov tests for drift detection in feature subspaces. It combines univariate detectors using a two-level combination strategy, that works based on the majority voting technique. An experimental study showed significant



benefits as compared with the univariate approach.

Combined Concept Drift Detectors (CCDD) (Wozniak et al, 2016) presents three combination rules (ALO-at least one, ALHD-at least half, and AD-all detectors) with heterogeneous drift detectors (DDM, HDDM<sub>A</sub>, HDDM<sub>W</sub>, CUMSUM, and Page-Hinkley) to achieve better detection efficiency. To ensure diversity in detectors, the ensemble is implemented based on five detectors with different learners. Experiments were conducted to estimate detector sensitivity, the number of false alarms, and the computational complexity of the model.

e-Detector(Du et al, 2015) proposes a selective ensemble paradigm and a fusion rule to consolidate results from multiple detectors, because equivalent change indicators lead to a performance decrease in e-Detector. The benchmark methods such as DDM, EDDM, ADWIN, STEP, and EnDDM were used as candidates for experiments and the results showed that e-Detector performs consistently better than those benchmarks.

Drift Detection Ensemble(DDE) (Maciel et al, 2015) aggregates the warnings and drift detections of three detectors in parallel. Different strategies and configurations are chosen based on the sensitivity of the ensemble. The technique utilizes one instance of the base detector during stable concepts and one more if the warning level is reached. HDDM<sub>A</sub>, HDDM<sub>W</sub>, DDM, ECDD, and ADWIN are unified in three different combinations. Experiments with six drift detectors were compared using their default configurations over the 20 datasets and the overall performance of the DDE method was statistically superior to other several detectors.

### *Problem Description & Objectives*

Reviewing drift detection methods from the past two decades, it is found that most detection techniques operate similarly and have common drawbacks. The past drift detection methods update the model based on the drift

type that occurred. In case of real drift, the model is rebuilt from the scratch by changing the entire training pipeline and trained with new datasets. On the contrary, when a virtual drift occurs the model is just retrained i.e., refitting the same model on the latest datasets. Rebuilding a learning model is way much costlier than just retraining it and the correct decision on what to do depends on differentiating drift type precisely. The existing drift detection methods are impure and so do not detect drift in its pure form i.e., the methods detecting virtual drift do not concern learner performance and similarly, the methods detecting real drift do not monitor distribution dissimilarities. There is no hybrid drift detector ensemble in the past that combines heterogeneous methods which can monitor both learner performance and data distribution dissimilarities. Further, the methods are not connected and thus do not communicate change status with one another. Past methods do not differentiate drift precisely into Pure Real, Pure Virtual, and Hybrid drift. Further, the drift occurrence information is also not preserved for future analysis. Motivated by the above issues this work aims to achieve the following objectives:

- I. To Design a Hybrid Concept Drift Detector Ensemble, which is capable of unifying heterogeneous drift detection methods.
- II. To Design a Connected Drift detection Framework, that is capable of monitoring learner performance changes and distribution dissimilarities concurrently.
- III. To Preserve Drift Occurrence Data, so that to make the drift understanding module more effective in predicting future drift occurrences.

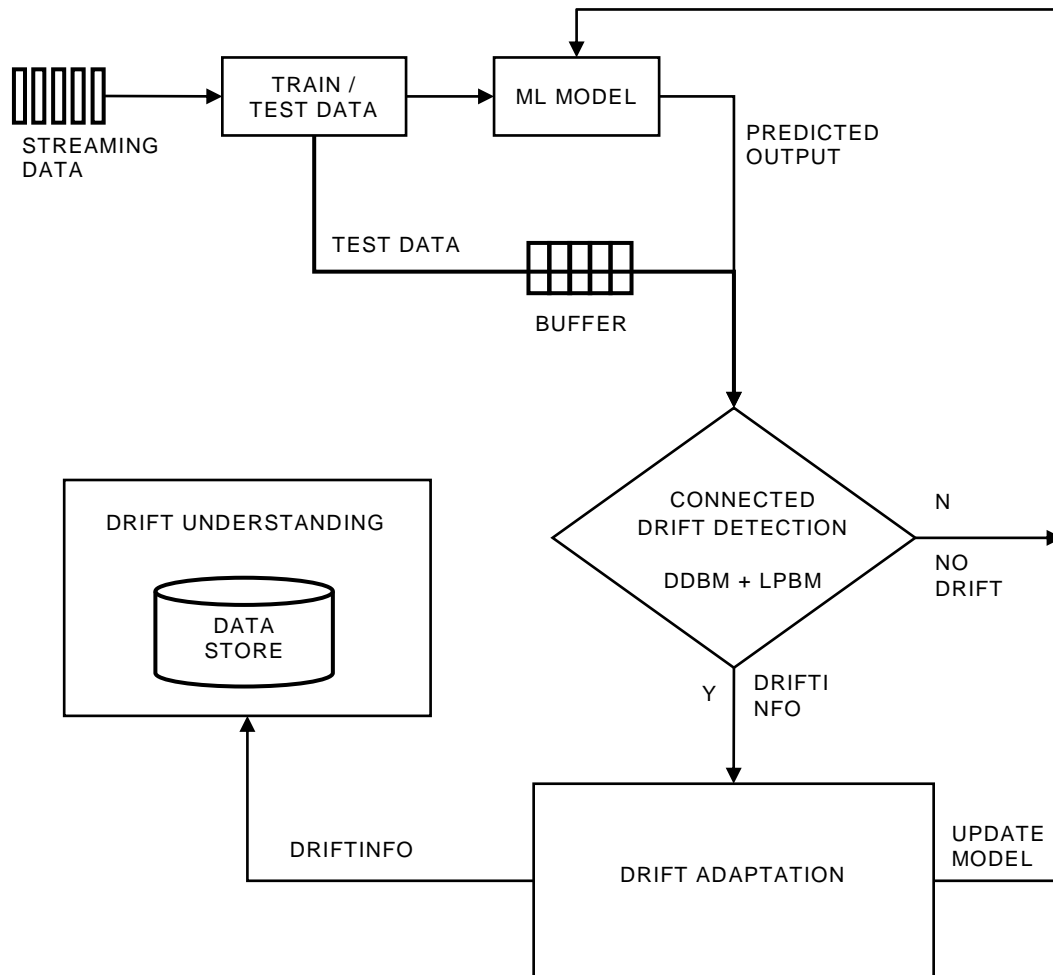
### **Hybrid Concept Drift Detector Ensemble**

This research work proposes a heterogeneous ensemble that unifies drift detection methods to achieve better drift handling performance. The ensemble consists of two mixed categories of methods that monitor both input data for



distribution dissimilarities and learner output for performance degradation concurrently. The parallel framework, as shown in Fig. 2 is designed to handle real and virtual drift concurrently. The streaming input data is split in a ratio into train and test data and forwarded to the ML model for prediction. Concurrently, the data is sent to the drift detection module for

distribution dissimilarity analysis. A data buffer is utilized to synchronize input data with ML Model output since it is necessary to analyze both simultaneously. The drift handling framework consists of three major components such as I. Drift Detection, II. Drift Understanding and III. Drift Adaptation.



**Figure 2.** A Parallel Connected Drift Detection Framework

**Connected Drift Detection Method**

This paper proposes a Connected Drift Detection Method (CDDM) that assists in unifying two heterogeneous drift detection methods. The CDDM can monitor both learner output and input data simultaneously in synchronization and performs two sub-tasks such as I. Learner Performance Analysis and II.

**Distribution Dissimilarity Analysis.**

**Learner Performance Analysis**

The LPBM detection methods evaluate the prediction output of the learner continuously at adjacent time intervals. The learner performance-based drift detection technique estimates the dissimilarity in performance of the learner using various measures and the



most common is the online error rate. This work uses the three most popular learner error rate-based drift detection algorithms that calculate the dissimilarity of the base classifier and if found significant the model is updated. The three methods are utilized separately and combined separately with a distribution dissimilarity monitoring method.

DDM Drift Detection Method [30] uses a base classifier to predict the incoming streaming data and monitors the overall online error rate. It calculates the number of errors in a sample of  $n$  inputs i.e., no of wrong predictions. For each input  $i$  in the sequence, the error rate is the probability of observing False,  $p_i$ , with standard deviation given by  $s_i = \sqrt{p_i(1-p_i)/i}$ . At reaching warning level denoted by  $p_i + s_i \geq p_{min} + 2 \cdot s_{min}$  the input examples are stored for possible drift, and at reaching drift level denoted by  $p_i + s_i \geq p_{min} + 3 \cdot s_{min}$  the drift is confirmed. The old model is discarded and a new model is built using a new set of inputs.

EDDM Early Drift Detection Method [31] works on the idea of finding the distance between two classification errors. The average distance between two errors is represented as  $p'_i$  and its standard deviation  $s'_i$ . The  $\alpha$  is the maximum threshold for the warning level and  $\beta$  for the drift level. The  $p'_{max} + 2 \cdot s'_{max}$  represents the maximum distance between errors. The warning level and drift level is represented as,

$$(p'_i + 2 \cdot s'_i) / (p'_{max} + 2 \cdot s'_{max}) < \alpha \text{ and}$$

$$(p'_i + 2 \cdot s'_i) / (p'_{max} + 2 \cdot s'_{max}) < \beta$$

respectively. If the threshold is crossed, the values for  $p'_{max}$  and  $s'_{max}$  are reset. The current model is discarded and a new model is rebuilt using the samples taken after the warning level trigger. DDM is capable of detecting abrupt drift occurrences and performs poorly for gradual drift whereas EDDM provides good detection performance for both.

HDDM<sub>A</sub> Hoeffding's Inequality Drift Detection Method [32] monitors the performance of the

base learner using probability inequalities to obtain theoretical guarantees on distributional changes. HDDM<sub>A</sub> is different from both the above methods and uses moving averages among classification errors to detect drift. This technique computes the statistical distance ( $\rho$ ) between the last observed values that is statistically far from the expected one ( $\mu$ ). The warning level is set at 95 percent denoted by  $Pr\{\mu - 2\sigma \leq \rho \leq \mu + 2\sigma\} \approx 0.95$  and the drift level at 99 percent denoted by  $Pr\{\mu - 3\sigma \leq \rho \leq \mu + 3\sigma\} \approx 0.99$ . The  $\alpha_w$  is the desired significance level for warning and  $\alpha_D$  for drift, where  $\alpha$  is the standard deviation. A hypothetical alternative classifier replaces the old one when a drift signal is triggered.

### Distribution Dissimilarity Analysis

DDBM detection methods evaluate input data distribution for finding change and confirm drift if the change in data distribution is significant. Distribution differences are usually measured using statistical distance functions that find the difference between current and past cumulative probability distribution samples. This work uses the Two-Sample Kolmogorov Smirnov Test which is used for evaluating distributions as proposed by [33] and recently was proved to be efficient for virtual drift detection by [23]. It finds the upper bound of difference between two data distributions. The two-sample KS-test is non-parametric and does not assume the distribution of data. The statistic is computed as  $D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)|$  where  $F_{1,n}$  and  $F_{2,m}$  are the empirical distribution function of samples and  $x_1, \dots, x_n$  are independent and identically distributed (i.i.d.) random variables lying in the domain of real numbers with a common cumulative distribution function. The statistic performs the KS-test to reject the null hypothesis at level  $\alpha$  by computing  $\sqrt{\frac{nm}{n_m}} D_{n,m} > K_\alpha$ . So, to describe the K-S test compares the distributions of two samples by measuring the distance between the empirical distribution functions, considering both their location and shape.



### Pure and Hybrid Drift Detection

Pure and Hybrid drift detection brings numerous quality and performance improvements to the existing drift handling process. Concept drift is generally characterised by input data and its relationship with the target class. The pure and hybrid drift can be clearly defined from past representations [8, 9].

The input data is formed by a vector of attributes represented by 'X' and the class label by 'y'. Each object or instance 'i' is formed by (X, y), and the set of instances  $P(X, Y)$  is referred to as the dataset or concept.

Concept =  $P(X, Y)$  or  $P(X)$ (1)

As discussed earlier, drift can be categorized as Real and Virtual. A real concept is represented as the joint distribution of covariates X and class variables Y denoted as  $P(X, Y)$  and a virtual concept is represented as the distribution of dependent variables X denoted as  $P(X)$ .

Real Concept =  $P(X, Y)$ (2)

Virtual Concept =  $P(X)$ (3)

This work aims to find concept drift in  $X \rightarrow Y$  classification problems where the class label y is causally determined by the values of the covariates X. The posterior distribution is represented as  $P(y|X)$  and the distribution of dependent variables is represented by  $P(X)$ . This work monitors change in posterior distribution and dependent variables concurrently, so that it can detect the pure and hybrid drift types.

A real concept drift occurs when posterior probability distribution varies between test and training samples whereas a virtual concept drift occurs when the probability distribution of dependent variables varies in test and training samples.

Class / Real Concept Drift:

$$P_t(y|X) \neq P_{t-1}(y|X)(4)$$

Covariate / Virtual Concept Drift:

$$P_t(X) \neq P_{t-1}(X)(5)$$

From the above definitions, the pure and hybrid drift can be represented. The Pure Real Concept drift occurs when the change occurs in posterior probability distribution whereas the probability distribution of dependent variables remains the same in test and train samples. Similarly, the Pure Virtual Concept drift occurs when the probability distribution of dependent variables changes whereas the posterior probability distribution remains the same between the test and training samples

The Pure Real Drift:

$$P_t(y|X) \neq P_{t-1}(y|X)$$

$$\text{where } P_t(X) = P_{t-1}(X)(6)$$

The Pure Virtual Drift:

$$P_t(X) \neq P_{t-1}(X)$$

$$\text{where } P_t(y|X) = P_{t-1}(y|X)(7)$$

The third form of drift, the Hybrid Drift is detected when the change occurs in both posterior probability distribution and probability distribution of dependent variables between the test and training samples at the same time instant or within a short time duration. The hybrid drift is represented as,

The Hybrid Drift:

$$P_t(y|X) \neq P_{t-1}(y|X) \wedge P_t(X) \neq P_{t-1}(X)(8)$$

The connected drift detection method receives input streaming data synchronized by a buffer and the predicted output from the learning model concurrently. The CDDM utilizes parallel processing to improve computation time and shares a vector memory DriftFlag to find drift type. The flag is shared with two subtasks such as LPBM and DDBM. The DDBM task finds distribution dissimilarities in the streaming data for a given sample window and the LPBM task



monitors learner error performance. If distribution drift is confirmed it assigns the flag a corresponding value and simultaneously for the same sample the learner performance drift is monitored. The type of drift is decided only after monitoring both the measures.

A Data Store is a novel addition to the drift handling framework and is introduced as part of the drift understanding module. The data store is a file storage and is utilized to store drift occurrence-related data in the form of event logs. A sample of drift occurrence data is shown in Tab. 2 of section 4.3. The drift understanding module collects drift occurrence data from the detection phase and preserves it in the data store for future analysis. The detailed information and techniques for storage and analysis of drift occurrence data will be discussed in future extensions of this research work. papers. Similarly, the drift adaptation phase will also be discussed in the future paper.

### Experimental Analysis

This work was built using Scikit-multiflow framework that uses a python library for processing streaming data. The window size (100) and sample size (30) and the significance level  $\alpha$  were set to default values. Three groups of experiments in which the Learner Performance and Distribution Dissimilarity-based methods were tested independently and in unification. Two base learners such as Naive Bayes (NB) and Hoeffding Tree (HT) classifiers were used for prediction. Three learner error rate-based algorithms Drift Detection Method (DDM), Early Drift Detection Method (EDDM), and Hoeffding's Inequality Based Drift Detection Method (HDDM) were utilized to monitor learner performance and the Kolmogorov Statistic (KS) Test was used to monitor the input data distribution dissimilarities. The unified method combined both the above heterogeneous techniques in three combinations. The connected technique was implemented as a sequential and parallel

process to compare computation time.

### Dataset

This work utilized a SEA data generator for generating synthetic datasets with 1 million samples and was split into a 70:30 ratio for training and testing purposes. The SEA generator has 4 different classification functions with different conditions. The functions are:

$$\text{func0: } \text{Sum}(\text{att1}, \text{att2}) \leq 8$$

$$\text{func1: } \text{Sum}(\text{att1}, \text{att2}) \leq 9$$

$$\text{func2: } \text{Sum}(\text{att1}, \text{att2}) \leq 7$$

$$\text{func3: } \text{Sum}(\text{att1}, \text{att2}) \leq 9.5$$

To generate drift the classification functions can be changed abruptly or gradually to obtain respective drifts in the dataset. The function generating the attribute values has a threshold and the sum of the two relevant attributes are unique for each classification function. The abrupt drift was generated for every 5000 samples and the gradual drift was generated for every 2500 samples. The dataset inherently had distribution drift and so it was utilized to find pure and hybrid occurrences of real and virtual drift. It generates numeric and continuous attributes with 3 independent features and one target class variable. The values of the attributes vary from 0 to 10 and only 2 attributes of them are relevant to the classification task.

### Pseudocode

This is abstract pseudocode of the connected drift detection method. The CDDM algorithm executes in parallel and receives both input data and predicted output simultaneously. The input data is forwarded to the DDBM subtask to find distribution dissimilarities. Concurrently, the input data is forwarded to the machine learner to make a prediction, and its output is given to LPBM subtask for calculating learner performance dissimilarity.



---

**Algorithm: Connected Drift Detection Method - Parallel (CDDM-P)**

---

```
1: Drift_Info = CDDM(Input_Data, Predicted_Output)
2:   define Drift_Info[D_Type, D_Time]
3:   define Exp_Class, Pred_Class
4:   define DriftFlag[ ]
5:   define ID, PO
6:   define method LPBM(PO)
7:   define method DDBM(ID)
8:   DDBM(Input_Data)
9:   while sample < max_samples
10:     for(i in attrib_count)
11:       Data = ID[0][i]
12:       KS-Test(Data)
13:       if Drift_Detected()
14:         DriftFlag[sample_ID] = 1
15:       end if
16:     end for
17:   end while
18:   LPBM(Predicted_Output)
19:   while l_sample < max_samples
20:     if Pred_Class != Exp_Class
21:       EDDM(error)
22:     else
23:       EDDM(no error)
24:     if Warning_Detected()
25:       log.write(Warning)
26:     if Drift_detected()
27:       if DriftFlag[sample_ID] == 1
28:         log.write(D_Type, D_Time) // Hybrid Drift
29:       else if DriftFlag[sample_ID] == 0
30:         log.write(D_Type, D_Time) // Pure Real Drift
31:       else
32:         log.write(D_Type, D_Time) // Pure Virtual Drift
33:     end while
34:   //Multiprocessing for parallel execution
35:   Process1(DDBM)
36:   Process1.start()
37:   Process2(LPBM)
38:   Process2.start()
39: return Drift_Info
```

6180



40://end

**Results and Inference**

The results were collected from individual executions of LPBM and DDBM methods and are compared with the CDDM method. The graphs were plotted for comparing the detection methods based on their capability to detect the type of drift, average no. of model retrain or rebuild requests, and execution time. The CDDM technique is more of a quality improvement to the existing drift detection techniques and achieves robust drift handling capability. So, this technique cannot be

compared for performance with other concept drift detector ensembles. A sample of drift occurrence data is shown in Tab. 2, and Fig. 3 shows the complete drift occurrence data with detection time. The detection time represents the  $i^{th}$  instance from the incoming test sample stream and the type represents Pure Real, Pure Virtual, and Hybrid drift. In the process of detecting hybrid drift, the default sample size was increased from 100 to 200 so that the algorithm can detect some distant occurrences of real and virtual drift.

**Table 2.** Drift Occurrence Data Sample

INDEX	Detection Time	Drift Type	INDEX	Detection Time	Drift Type	INDEX	Detection Time	Drift Type	INDEX	Detection Time	Drift Type
1	1110	1	6	38434	3	31	98429	3	46	137477	3
2	1287	2	7	39029	2	32	99036	2	47	139640	3
3	4505	3	8	40120	2	33	102555	3	48	141294	2
4	8429	2	9	42605	3	34	103206	3	49	141485	3
5	10362	1	0	49585	2	35	110806	3	50	146626	3
6	10440	3	1	53200	3	36	112690	2	51	147481	2
7	15920	2	2	56549	3	37	112861	3	52	148780	2
8	16116	2	3	65669	3	38	117683	3	53	149040	3
9	19982	3	4	67471	2	39	118778	2	54	150810	3
10	25476	2	5	67659	3	40	119604	3	55	155177	3
11	25665	3	6	70808	3	41	126629	3	56	159606	3



1 2	29031	2	2 7	88439	3	4 2	128774	3	5 7	162503	2
1 3	31108	1	2 8	92599	3	4 3	129044	2	5 8	162695	3
1 4	31154	3	2 9	94902	2	4 4	134912	3	5 9	167606	2
1 5	37468	3	3 0	96704	1	4 5	136708	1	6 0	168774	3
1 – Pure Real Drift			2 – Pure Virtual Drift			3 – Hybrid Drift					

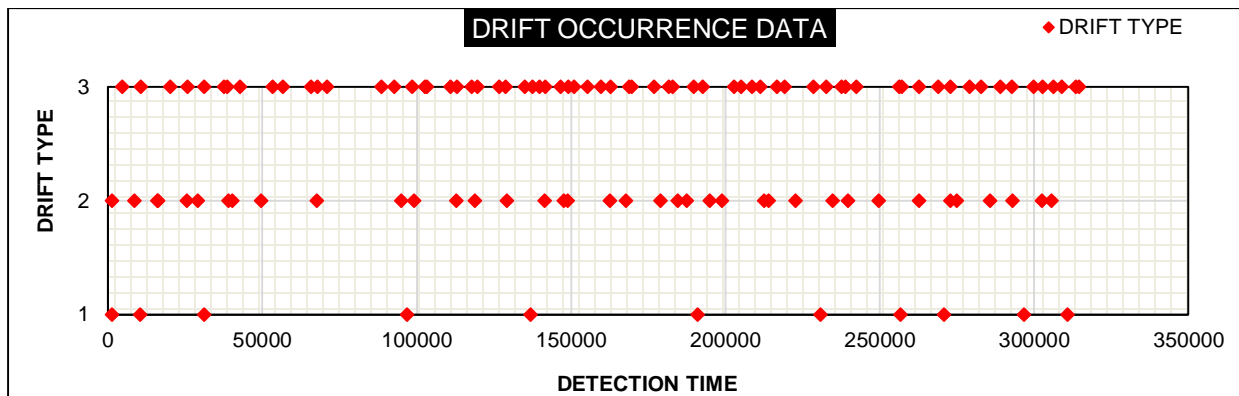


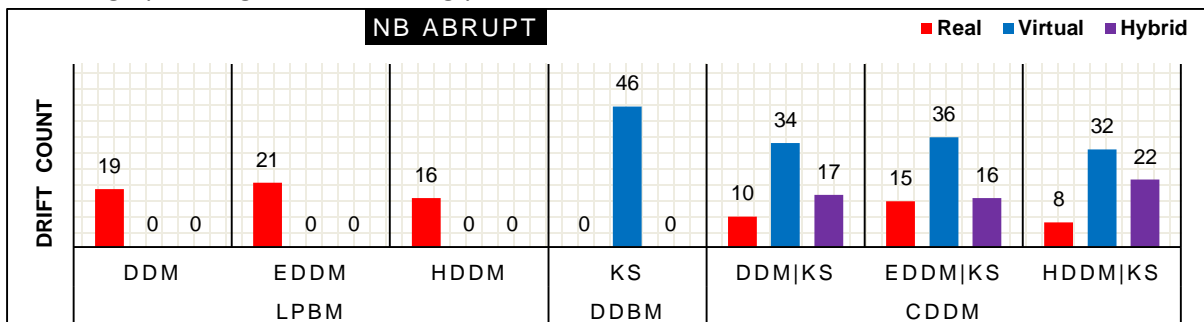
Figure 3. Drift Occurrence Data

6182

### Drift Type Differentiation

The proposed method is capable of detecting pure drift and differentiating the type of drift. As shown below in Fig. 4a and Fig. 4b, the LPBM, DDBM, and CDDM techniques are compared on their ability to detect real, virtual, and hybrid drift. The graph is organised accordingly where

each figure represents the combinations of base learner NB and HT with abrupt and gradual datasets. From the graph, it is inferred that the LPBM and DDBM methods are capable of detecting only real and virtual drift respectively. Further, the CDDM technique is capable of differentiating drift as pure real, pure virtual, and hybrid drift precisely.



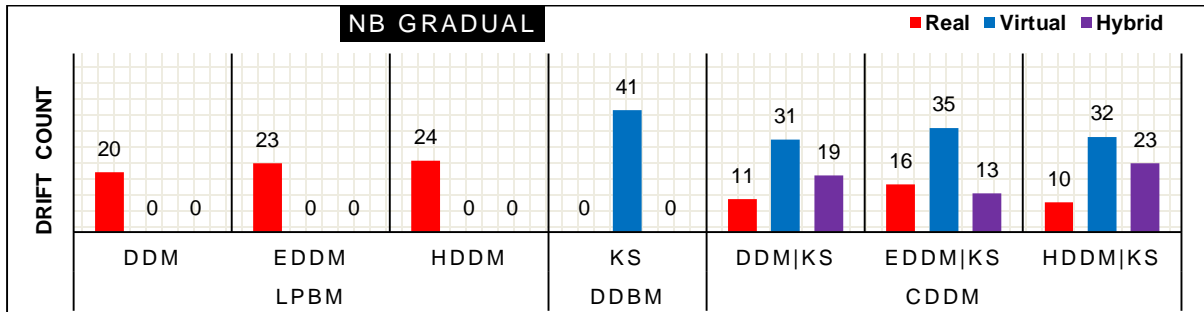


Figure 4a. Drift Type Differentiation: LPBM vs DDBM vs CDDM with NB as Base Learner

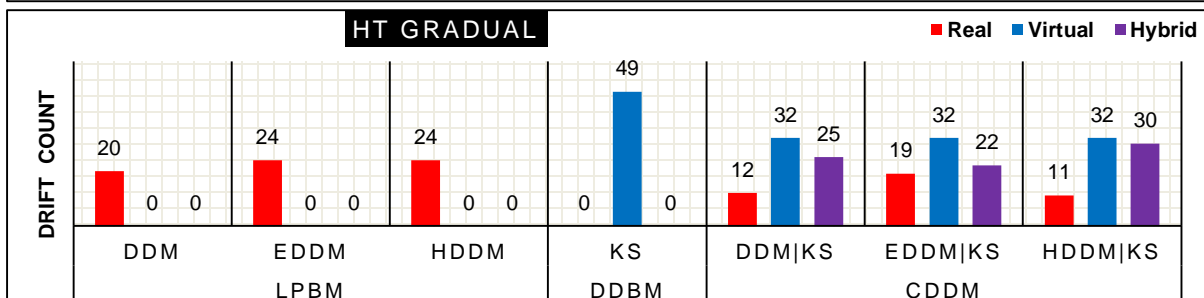
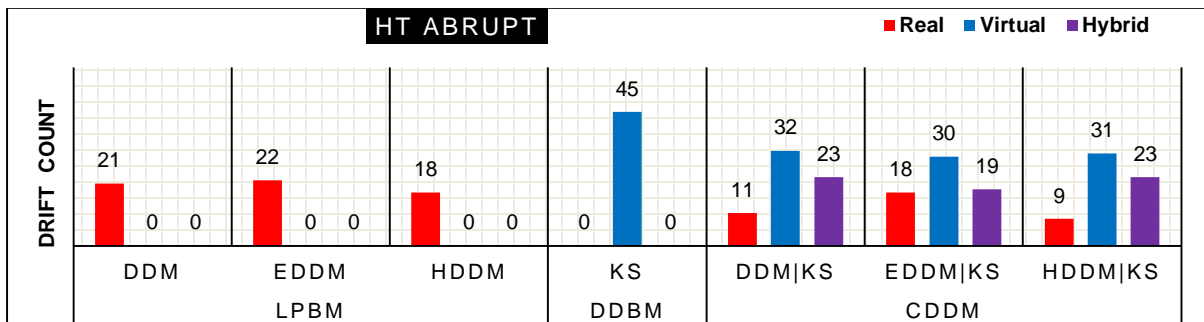


Figure 4b. Drift Type Differentiation: LPBM vs DDBM vs CDDM with HT as Base Learner

**Model Rebuild/Retrain Request**

The CDDM technique achieves improved model update (rebuild/retrain) performance when compared with LPBM and DDBM techniques. As shown in Fig. 5a and Fig. 5b, the

CDDM technique makes lesser and more precise model rebuild and retrain requests respectively. The CDDM technique performs consistently better across all the drift detection methods and for both abrupt and gradual drift types.

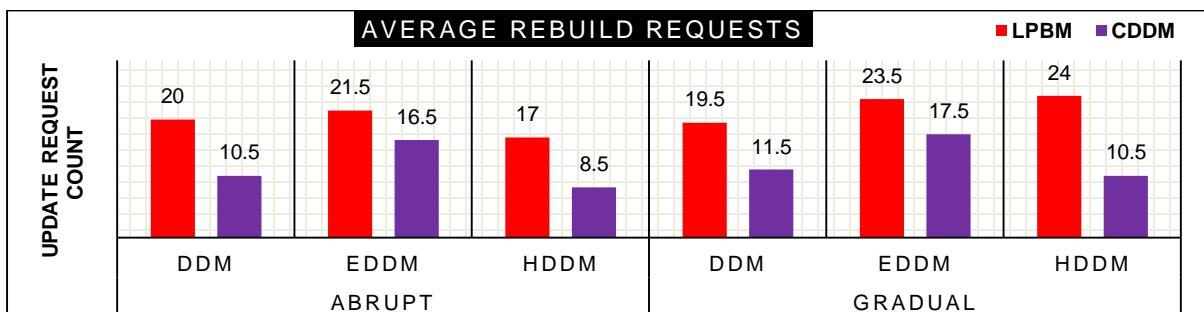


Figure 5a. Average Model Rebuild Requests: LPBM vs CDDM



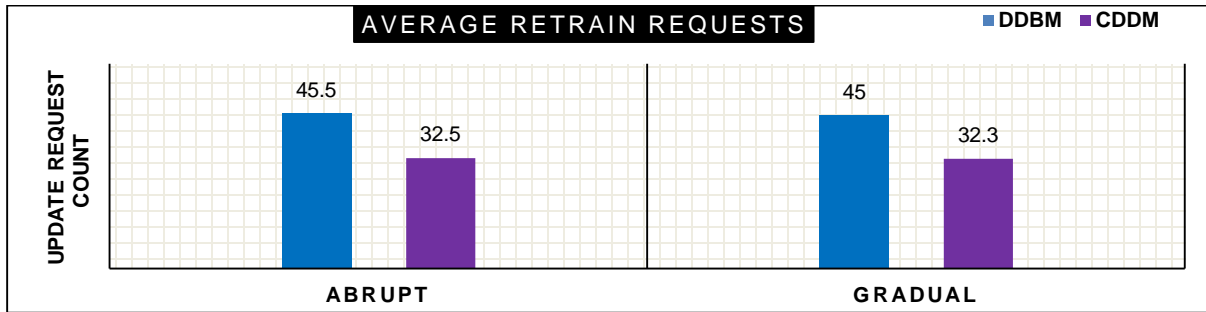


Figure 5b. Average Model Retrain Requests: DDBM vs CDDM

**Execution Time**

In comparison as shown in Fig. 6, the LPBM techniques are computationally cheaper than DDBM techniques. But considering functionally, model training and rebuild time will make LPBM techniques costlier. The

training phase of the LPBM technique was not considered for execution time evaluation. The CDDM technique was implemented as a sequential (CDDM-S) and parallel (CDDM-P) process for comparison. The parallel technique executes quicker than the sequential method and is on par with DDBM methods.

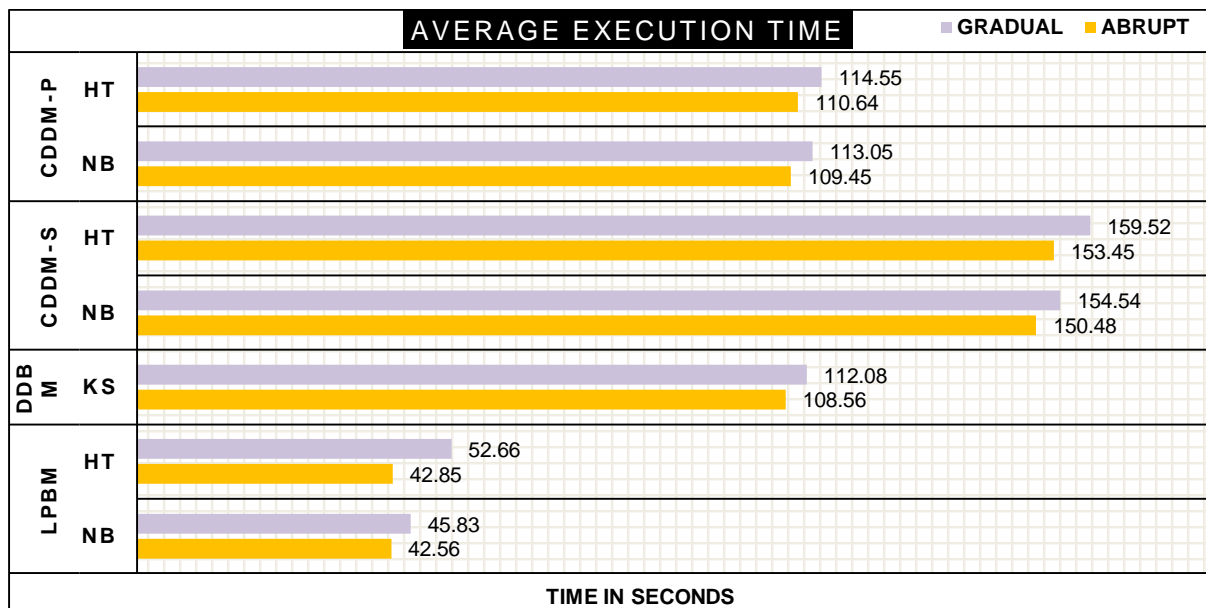


Figure 6. Average Execution Time: LPBM vs DDBM vs CDDM-S vs CDDM-P

**Conclusion**

Online learning methods that operate on nonstationary data need to be evaluated constantly. The drift occurs due to dynamism and uncertainty in the real world and is detected by monitoring data distribution or learner performance with its own set of methods. Past methods detect drift incoherently i.e., the methods do not share drift status. This disconnection lead to unnecessary

model updates which was an expensive task when operating on streaming data settings. To overcome this a novel hybrid concept drift detector ensemble was proposed and utilizes a novel connected drift detection method which connects heterogeneous detection methods. The connected technique is capable of monitoring both distribution dissimilarity and learner error rate concurrently. Experiments were conducted for individual and connected techniques separately and the latter performed



better in aspects of detecting and differentiating drift into the pure real, pure virtual, and hybrid drift. Further, the connected technique is capable of better model update performance by making a precise rebuild/retain decision. To reduce computation time this work utilized a parallel processing framework that executes the detection methods concurrently. In the future extension of this work, a novel drift adaptation technique that can handle pure and hybrid drift will be proposed. Further, the idea of preserving drift occurrence data enables intelligent and early drift prediction.

## References

Agrahari, S. and Singh, A.K. Concept Drift Detection in Data Stream Mining: A literature review. *Journal of King Saud University-Computer and Information Sciences* 2021.

Baena-Garcia, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldá, R. and Morales-Bueno, R.: Early drift detection method. In *Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams (IWKDDs'06)* Berlin, Germany 2006; (6): 77-86.

Barros, R.S.M. and Carvalho Santos, S.G.T. A large-scale comparison of concept drift detectors. *Information Sciences* 2018; 451: 348-370.

Barros, R.S.M. and Carvalho Santos, S.G.T. An overview and comprehensive comparison of ensembles for concept drift. *Information Fusion* 2019; 52: 213-244.

Dehghan, M., Beigy, H. and ZareMoodi, P. A novel concept drift detection method in data streams using ensemble classifiers. *Intelligent Data Analysis* 2016; 20(6): 1329-1350.

Ditzler, G., Roveri, M., Alippi, C. and Polikar, R. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine* 2015; 10(4): 12-25.

Du, L., Song, Q., Zhu, L. and Zhu, X. A selective detector ensemble for concept drift detection. *The Computer Journal* 2015; 58(3): 457-471.

**Acknowledgement:** The authors wish to thank all their colleagues, friends and family who supported and provided insights into this research work.

**Funding Statement:** The authors received no specific funding for this study.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

Frias-Blanco, I., del Campo-Ávila, J., Ramos-Jimenez, G., Morales-Bueno, R., Ortiz-Diaz, A. and Caballero-Mota, Y. Online and non-parametric drift detection methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering* 2014; 27(3): 810-823.

Gama, J., Medas, P., Castillo, G. and Rodrigues, P. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, Springer, Berlin, Heidelberg, 2004; 286-295.

Goldenberg, I. and Webb, G.I. Survey of distance measures for quantifying concept drift and shift in numeric data. *Knowledge and Information Systems* 2019; 60(2): 591-615.

Goncalves Jr, P.M., de Carvalho Santos, S.G., Barros, R.S. and Vieira, D.C. A comparative study on concept drift detectors. *Expert Systems with Applications* 2014; 41(18): 8144-8156.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M. and Ghédira, K. Discussion and review on evolving data streams and concept drift adapting. *Evolving systems* 2018; 9(1): 1-23.

Kifer, D., Ben-David, S. and Gehrke, J. Detecting change in data streams. In *VLDB 4* 2004; 180-191.

Korycki, L. and Krawczyk, B.: Unsupervised drift detector ensembles for data stream mining. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)* 2019; 317-325.



Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J. and Wozniak, M. Ensemble learning for data stream analysis: A survey. *Information Fusion* 2017; 37: 132-156.

Krempl, G., Zliobaite, I., Brzezinski, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M. and Stefanowski, J. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter* 2014; 16(1): 1-10.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. and Zhang, G. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering* 2018; 31(12): 2346-2363.

Maciel, B.I.F., Santos, S.G.T.C. and Barros, R.S.M. A lightweight concept drift detection ensemble. In *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2015; 1061-1068.

Massey Jr, F.J. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 1951; 46(253): 68-78.

Moreno-Torres, J.G., Raeder, T., Alaiz-Rodriguez, R., Chawla, N.V. and Herrera, F. A unifying view on dataset shift in classification. *Pattern recognition* 2012; 45(1): 521-530.

Oliveira, G., Minku, L.L. and Oliveira, A.L. Tackling Virtual and Real Concept Drifts: An Adaptive Gaussian Mixture Model Approach. *IEEE Transactions on Knowledge and Data Engineering* 2021.

Patil, M.M. Handling concept drift in data streams by using drift detection methods. In *Data Management Analytics and Innovation*, Springer, Singapore 2019; 155-166.

Perez, J.L.M., Barros, R.S. and Santos, S.G. Statistical tests ensemble drift detector. In *IEEE Symposium Series on Computational Intelligence (SSCI)* 2020; 1021-1028.

Ramirez-Gallego, S., Krawczyk, B., García, S., Wozniak, M. and Herrera, F. A survey on data pre-processing for data stream mining: Current

status and future directions. *Neurocomputing* 2017; 239: 39-57.

Sebastiao, R. and Gama, J. A study on change detection methods. In *Progress in artificial intelligence. 14th Portuguese conference on artificial intelligence, EPIA 2019*; 12-15.

Sobolewski, P. and Wozniak, M. Comparable study of statistical tests for virtual concept drift detection. In *Proceedings of the 8th International Conference on Computer Recognition Systems CORES*, Springer, Heidelberg 2013; 329-337.

Tran, D.H., Gaber, M.M. and Sattler, K.U. Change detection in streaming data in the era of big data: models and issues. *ACM SIGKDD Explorations Newsletter* 2014; 16(1): 30-38.

Tsymbol, A. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin* 2004; 106(2): 58.

Wang, Z. and Wang, W. Concept drift detection based on Kolmogorov-Smirnov test. In *Artificial Intelligence in China*, Springer, Singapore 2020; 273-280.

Webb, G.I., Hyde, R., Cao, H., Nguyen, H.L. and Petitjean, F. Characterizing concept drift. *Data Mining and Knowledge Discovery* 2016; 30(4): 964-994.

Widmer, G. and Kubat, M. Learning in the presence of concept drift and hidden contexts. *Machine learning* 1996; 23(1): 69-101.

Wozniak, M., Ksieniewicz, P., Cyganek, B. and Walkowiak, K. Ensembles of heterogeneous concept drift detectors-experimental study. In *IFIP International 31 on Computer Information Systems and Industrial Management*, Springer, Cham, 2016; 538-549.

Zliobaite, I., Pechenizkiy, M. and Gama, J. An overview of concept drift applications. *Big data analysis: new algorithms for a new society* 2016; 91-114.

