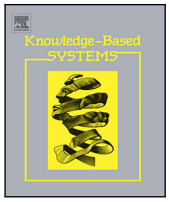
Knowledge-Based Systems 289 (2024) 111535

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



A comprehensive analysis of concept drift locality in data streams

Gabriel J. Aguiar, Alberto Cano *

Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA

ARTICLE INFO

Keywords:
Data streams
Multi-class
Concept drift
Machine learning

ABSTRACT

Adapting to drifting data streams is a significant challenge in online learning. Concept drift must be detected for effective model adaptation to evolving data properties. Concept drift can impact the data distribution entirely or partially, which makes it difficult for drift detectors to accurately identify the concept drift. Despite the numerous concept drift detectors in the literature, standardized procedures and benchmarks for comprehensive evaluation considering the locality of the drift are lacking. We present a novel categorization of concept drift based on its locality and scale. A systematic approach leads to a test bed of 2760 data stream benchmarks, reflecting various difficulty levels following our proposed categorization. We conduct a comparative assessment of 9 state-of-the-art drift detectors across diverse difficulties, highlighting their strengths and weaknesses for future research. We examine how drift locality influences the classifier performance and propose strategies for different drift categories to minimize the recovery time. Lastly, we provide lessons learned and recommendations for future concept drift research. Our benchmark data streams and experiments are publicly available at https://github.com/gabrieljaguiar/locality-concept-drift.

1. Introduction

Modern data sources continuously generate information characterized by both volume and velocity, flooding learning systems with a constant flow of data. This scenario is commonly referred to as data streams [1,2]. Traditional classification methods, designed for static data, struggle to keep up with the ever-changing characteristics of these incoming instances [1,3]. Given the dynamic nature of data streams, it becomes essential for learning methods to adapt and acquire knowledge about emerging concepts over time. This phenomenon is known as concept drift [4], and it can manifest in various ways, including shifts in class distribution and decision boundaries [5], and the emergence of new features or classes [6]. If not detected and addressed effectively, concept drift can significantly degrade predictive performance, as knowledge learned from older concepts may not be useful anymore to classify recent instances [7].

In recent years, the issue of concept drift has garnered significant attention within the research community across various domains, including sensors, robotics, system monitoring, and anomaly detection [8]. Current research in this field is tackling increasingly complex challenges. These challenges include accurately detecting concept drift within unstructured and noisy datasets [9], providing understandable explanations for concept drift [10], and effectively responding to drift by adapting relevant knowledge [11]. When we extend these concerns to scenarios involving multiple classes, we encounter a complex and perplexing scenario that actually occurs in many real-life applications. Detecting concept drift in such contexts becomes exceptionally demanding, as we must account for the evolving nature of multiple classes [6,12]. In addition to the challenges previously mentioned, it is important to note that the location of concept drift within the feature space significantly influences both the performance of classifiers and the effectiveness of drift detection methods [6,13]. However, there is a lack of studies that evaluate drift detectors under varying degrees of drift locality or provide benchmark datasets to support research in this crucial area.

Motivation. While the literature offers numerous concept drift detectors, there remains a notable absence of standardized procedures and benchmarks for a comprehensive assessment of these methods when considering the locality of concept drifts. Specifically, there is a lack of dedicated benchmarks suitable for evaluating drift detectors across a diverse spectrum of challenges, particularly those tied to the locality of concept drift. An in-depth experimental comparison of state-of-the-art drift detectors, applied to a diverse set of challenges, would provide valuable insights into the performance of these detectors under various conditions. Furthermore, current research tends to concentrate on particular subsets of drift detectors and specific data challenges, typically limited to binary class data. These studies often fall short in offering insightful knowledge regarding the essential aspects of concept drift that should be taken into consideration. For that reason, we propose a

* Corresponding author.
E-mail addresses: aguiargj@vcu.edu (G.J. Aguiar), acano@vcu.edu (A. Cano).

https://doi.org/10.1016/j.knosys.2024.111535
Received 12 December 2023; Received in revised form 15 February 2024; Accepted 15 February 2024
Available online 17 February 2024

comprehensive and reproducible study to gain insights and evaluate the performance of various drift detectors. This study spans a diverse range of difficulties, allowing us to analyze how the locality and magnitude of a concept drift impact its detection.

Overview and main contributions. This paper presents a comprehensive and reproducible study for benchmarking and evaluating the impact of the locality and magnitudes of a concept drift on the classifier or drift detector. We systematically identify critical challenges within this domain and leverage them to create a set of benchmark problems that encompass various difficulties, guided by a novel concept drift categorization. Furthermore, we conduct a comparative evaluation of nine state-of-the-art drift detectors across this wide range of difficulty. This analysis not only identifies the top-performing detectors considering the proposed difficulties but also sheds light on their specific strengths and weaknesses, providing valuable insights for future research in drift detection. The main contributions of this paper are:

- **Concept drift locality categorization.** We present a novel categorization that organizes concept drift based on the number of affected classes and the magnitude of the change. This categorization empowers us and future researchers to evaluate the effect of various concept drift levels when proposing new drift detectors or classifiers. This categorization guides the creation of benchmark problems, ranging from scenarios affecting only one class to those that transform the entire data stream. The comprehensive evaluation set consists of 2760 data stream benchmarks.
- **Drift locality impact evaluation.** We present a comprehensive study designed to assess the influence of concept drift locality on its detection, encompassing both binary and multi-class data streams.
- **Comparison between state-of-the-art drift detectors.** We conduct an extensive, comprehensive, and reproducible comparative study among 9 state-of-the-art drift detectors with different detection mechanisms based on the proposed framework and a set of benchmarks to assess their performance and behavior under a plethora of difficulties.
- **Recommendations and open challenges.** Based on the results from the experimental study, we derive recommendations seeking insights into the strengths and weaknesses of the top-performing drift detectors. These recommendations aim to provide a comprehensive understanding of the detectors' capabilities. Additionally, we identify open challenges within the domain of learning from data streams impacted by concept drift, outlining directions for future research.

This paper is structured as follows: Section 2 provides the theoretical foundation and discusses related work. Section 3 introduces our proposed concept drift locality categorization and presents the benchmark problems. Section 4 outlines the experimental setup, while Section 5 presents the results. Section 6 delves into the lessons learned, and Section 7 offers the conclusion and outlines directions for future work.

2. Background and related work

This section reviews the background and related work. We provide an overview of the literature on data streams and concept drift detection.

2.1. Data streams

A data stream refers to a potentially unbounded sequence of ordered instances that arrive over time within a system. Learning from data streams imposes specific limitations on classifiers [3]. We can define a stream, denoted as S, as a sequence $\langle s_1, s_2, s_3, \dots, s_\infty \rangle$, where $s_i = (X, y)$. This stream can be handled either one instance at a time (online scenario) or in batches (block scenario). Data streams exhibit four primary characteristics [2,5]: (i) Volume, (ii) Velocity, (iii) Veracity, and (iv) Non-stationarity, which present challenges to classifiers that must adapt accordingly.

Data streams are susceptible to concept drift [14,15]. Each instance arrives at a specific time, denoted as t, and is generated based on a probabilistic distribution denoted as $\phi^t(X, y)$, where X represents the feature vector and y denotes the class label. If all instances in the stream are generated by the same probability distribution, the data is considered stationary, indicating a single and stable underlying concept. However, in real-world applications, data rarely adheres to stationary assumptions [16]. On the other hand, when two distinct instances arriving at times t and $t + C$ are generated by $\phi^t(X, y)$ and $\phi^{t+C}(X, y)$ respectively, and if $\phi^t \neq \phi^{t+C}$, indicates the occurrence of a concept drift. This phenomenon impacts various aspects of a data stream and, as such can be examined from multiple viewpoints. When analyzing and comprehending concept drift, the following factors come into consideration [5,6]:

- **Influence of the decision boundaries.** Firstly, it is necessary to consider how concept drift affects the learned decision boundaries, distinguishing between real and virtual concept drifts. Virtual drift produces a change in the unconditional probability distribution $P(x)$, without affecting the learned decision boundaries. Although virtual drift does not impair learning models, its detection is necessary to avoid false alarms and prevent unnecessary, costly adaptations. In contrast, real concept drift modifies the decision boundaries, making them worthless to the current concept. Detecting and adapting to real concept drift is crucial for preserving predictive performance.
- **Speed of changes.** There are three types of concept drift [17] regarding their speed: (i) incremental; (ii) gradual; and (iii) sudden concept drifts as illustrated in Fig. 1. Incremental drift generates a sequence of intermediate states between the old and new concept, while gradual drift oscillates between instances coming from both old and new concepts, with the new concept becoming more and more frequent over time. Finally, sudden drift instantaneously switches between old and new concepts, leading to an instant degradation of the underlying learning algorithm.
- **Recurrence.** Changes in the stream can be either unique or recurring as illustrated in Fig. 1. In the latter case, the previously seen concept may reemerge over time, allowing us to recycle previously learned knowledge. The past knowledge can be used as an initialization point for the drift recovery.
- **Presence of noise.** Noise can take the form of sporadic, insignificant variations within a stream that can be disregarded, or substantial corruption within the features or class labels that need to be dealt with in order to prevent the input of misleading or adversarial data into the classifier [18]. Drift detectors must ignore noise.
- **Locality**. The literature distinguishes between global and local concept drifts [13]. The former impacts the entire data stream, while the latter pertains to specific parts of it, such as individual clusters of instances or subsets of classes. However, this classification is too shallow and a more in-depth and detailed discussion regarding the concept drift's locality will be introduced in this manuscript.

To address the challenges posed by concept drift, two approaches are commonly employed: (i) implicit and (ii) explicit. Implicit methods manage drift adaptation through intrinsic mechanisms integrated within the classifier, assuming its capability to self-adjust to new instances reflecting the most recent concept while gradually discarding outdated information [8,19]. These approaches involve establishing appropriate learning and forgetting rates, utilizing adaptive sliding windows, or continually tuning hyperparameters. Conversely, explicit

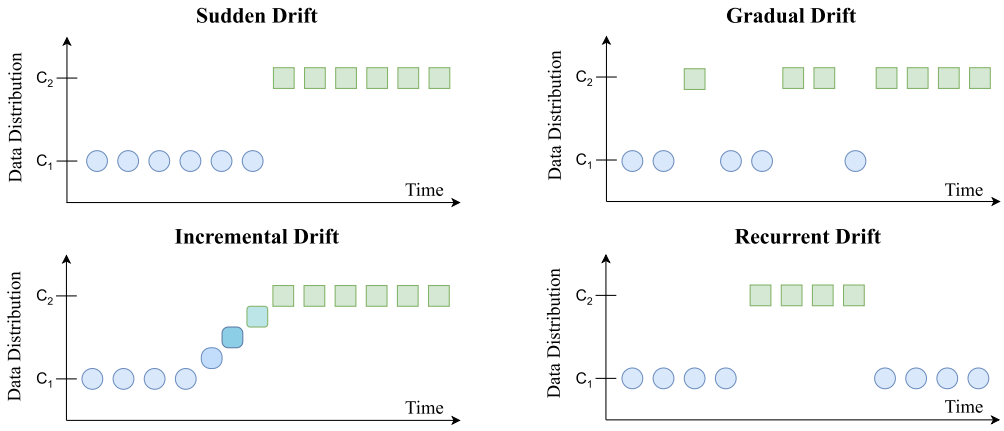


Fig. 1. Different types of concept drift.

approaches assign drift adaptation to an external tool known as a drift detector [8,15]. Drift detectors continuously monitor stream properties (e.g., statistics) or classifier performance (e.g., error rates). They raise a warning signal when there are indications of impending drift and trigger an alarm signal when concept drift has occurred.

2.2. Drift detectors

In recent years, a plethora of explicit drift detectors have been introduced [20]. As defined by Lu et al. [15], drift detectors function by extracting critical features from both historical and newly arrived data and subsequently subjecting them to dissimilarity tests. Like classifiers, drift detection methods fall into three categories: supervised (or error rate-based), semi-supervised, and unsupervised (or data distribution-based) [15]. The primary distinction among these categories is the point at which drift is identified. While supervised drift detectors pinpoint alterations in class boundaries, unsupervised detectors focus on tracking shifts in data distribution [4].

Among supervised drift detectors, the most widely adopted group relies on evaluating classifier error or accuracy using labeled instances. This category of drift detectors can be further divided into three distinct types: (i) change detection-based detectors, (ii) statistical-based detectors, and (iii) window-based detectors.

- **Change detection-based detectors**. In the first category, we find detectors like Page Hinkley [21], CUSUM [21], and Geometric Moving Average [22]. These detectors utilize cumulative sums to trigger an alarm when a significant change in input data occurs. While they are computationally efficient, their performance heavily depends on the selection of hyperparameters.
- **Statistical-based detectors.** The second group comprises detectors such as the Drift Detector Method (DDM) [1] and its variants like Early DDM (EDDM) [23], Reactive DDM (RDDM) [11], DDM based on Hoeffding's bound (HDDM) [24], Statistical Drift Detection Method (SDDM) [25], Fast Hoeffding Drift Detection Method (FHDDM) [26,27], McDiarmid Drift Detection Methods (MDDM) [28], Wilcoxon Rank Sum Test Drift Detector (WSTD) [29], and EWMA for Drift Detection (ECDD) [30]. These methods compute statistical features over time based on error rates. They operate under the assumption that as long as the data distribution remains stationary, the learner's error rate will decrease with an increasing number of analyzed samples. When the error rate exceeds a predefined threshold, an alert is triggered.
- **Window-based detectors.** Finally, a popular category of supervised detectors relies on metrics calculated within subwindows of a data stream. An example is the ADaptative WINdow (ADWIN) [31], which employs an adaptive sliding window based on Hoeffding's inequality. ADWIN manages a sliding window divided into two sub-windows representing old and new data and dynamically adjusts its size, expanding during periods of stability and contracting in the presence of drift. ADWIN signals a drift when the average between the two windows surpasses a predefined threshold. This approach has inspired the development of several other detectors such as Kolmogorov–Smirnov Windowing (KSWIN) [32], Statistical Test of Equal Proportions (STEPD) [33], optimized versions of ADWIN [34,35] and others [26,36].

In addition to detectors based on classifier accuracy, there are also supervised trainable detectors that employ traditional machine learning classifiers to detect concept drift. Examples of such detectors include DCS [37], the Restricted Boltzmann Machine (RBM-IM) [6], the Complexity Drift Detector (C2D) [38] and QuadCDD [39]. Concept drift detectors that can predict the type/speed of the drift have also been proposed in order to promote an informed adaptation of the classifer [40]. Moreover, Meta-Learning has been employed for drift detection [38,41,42]. Besides adapting to concept drift, Halstead et al. [43] presents a new algorithm to identify and correct errors in concept drift adaptation. It is important to note that there is a notable absence of drift detectors specifically designed for multi-class scenarios, which come with their own unique characteristics and challenges.

Unsupervised drift detectors are primarily focused on identifying disparities in unlabeled data without the need for added supervision. These detectors typically employ a distance metric to quantify the dissimilarity between the distribution of historical data and newly arrived data [44,45]. When this dissimilarity is statistically significant, it triggers a process to update the learning model. In this category, we find detectors such as Statistical Change Detection for multi-dimensional data (SCD) [46], the PCA-based change detection framework (PCA-CD) [47], Equal Density Estimation (EDE) [48], Least Squares Density Difference-based Change Detection Test (LSDD-CDT) [49], among others [15,50]. More advanced unsupervised methods aim to precisely locate where the drift occurred in the feature space [51]. These detectors focus on spatial searches employing various dissimilarity measures [10,52]. They tackle concept drift at its root, addressing distribution drift. Typically, these algorithms require users to define both a historical time window and a new data window. A common approach is to use two sliding windows, with the historical time window fixed while sliding the new data window [53,54]. Moreover, Cerqueira et al. [55] presents a detector based on the student-teacher learning mechanism.

As aforementioned, numerous drift detectors have been proposed in the literature, highlighting the need for comparisons to understand how each one behaves in different complex scenarios. While Lu et al. [15] and Suárez-Cetrulo et al. [8] reviewed the state-of-the-art in data stream learning with a focus on drift detectors, however they lacked an empirical comparison between these detectors. Additionally, the locality of the concept drift is often neglected [6,13]. In a different

Table 1
Comparison of most commonly applied, drift detectors and types of concept drift by contributions in related works.

Reference	Drift detectors	Drift type
Gama et al. [1]	DDM	Sudden; Gradual
Baena-Garcıa et al. [23]	DDM, EDDM	Sudden; Gradual
Frias-Blanco et al. [24]	DDM, ADWIN, $HDDM_W$, ECDD	Sudden; Gradual
Gonçalves et al. [56]	DDM, EDDM, ADWIN, ECDD,STEPD, PH, NB, PL, DOF	Sudden; Gradual
Barros and Santos [20]	DDM, EDDM, ADWIN, $HDDM_W$,$HDDM_A$, ECDD, SEQDRIFT, SEED, STEPD, FHDDM, FTDD, RDDM, WSTD	Sudden; Gradual
Santos et al. [57]	DDM, EDDM, ADWIN, $HDDM_W$,$HDDM_A$, ECDD, SEQDRIFT, STEPD, FHDDM, RDDM, and WSTD	Sudden; Gradual
Babüroğlu et al. [58]	DDM, EDDM, ADWIN, $HDMM_W$,$HDDM_A$, ECDD, SEQDRIFT, SEED, STEPD, FHDDM, FTDD, RDDM, WSTD, GMA, PH	Sudden
Korycki and Krawczyk [6]	FHDDM, RDDM, WSTD, PerfSim,DDM-OCI, RBM-IM	Sudden; Gradual; Incremental
Poenaru-Olaru et al. [59]	DDM, EDDM, ADWIN, $HDDM_W$,$HDDM_A$	Sudden; Gradual
Mahgoub et al. [60]	DDM, EDDM, ADWIN, $HDDM_W$,$HDDM_A$, STEPD, SEQDRIFT, RDDM, SEED, GMA, PH, EWMA, CUSUM	Sudden; Gradual; Incremental
Sakurai et al. [61]	DDM, EDDM, $HDDM_W$, $HDDM_A$	Sudden; Gradual; Incremental

approach, Barros and Santos [20] conducted an extensive empirical comparison and evaluation of concept drift detection methods across various data stream configurations. Moreover, Table 1 presents a summary of works that empirically compare drift detectors and the type of concept drift evaluated. However, a notable gap exists in the literature concerning concept drift detection in data streams with multiple classes and how the locality of the drift influences its detection.

3. Proposed categorization of concept drift locality

As discussed in various studies [6,13,62,63], local data difficulty factors and the number of classes exert a substantial influence on the capabilities of classifiers when handling data streams. Furthermore, these factors also play a major role in concept drift detection. Notably, the performance of standard drift detectors can vary significantly based on the underlying distribution of the stream, especially when they lack strategies to address these factors. This can result in either excellent performance or suboptimal outcomes.

However, the current body of literature addressing the categorization and characterization of concept drifts in streaming classification problems often overlooks two crucial factors: the number of classes affected and local data difficulties. It is important to note that a significant portion of concept drift research primarily focuses on binary classification and global shifts in data distribution, or perturbations that primarily impact the minority class, as observed in the imbalanced scenario [62]. Given the substantial impact that local data difficulty factors and the number of classes affected can have on non-stationary data streams, the existing categorizations of concept drift may fall short in evaluating concept drift detection in dynamic environments. They may fail to encompass all the nuances that make a concept drift particularly challenging or solvable. Therefore, we propose an extended concept drift categorization that explicitly incorporates considerations of locality and the number of classes affected when assessing concept drift. This expanded framework should pave the way for further research in the development of classifiers and drift detectors tailored for multi-class data streams in non-stationary environments, enabling systematic studies under different relevant drifting conditions.

3.1. Categorization of concept drift locality

First, we formally define the proposed categorization. Consider a bounded d-dimensional attribute space X and output space y and a given posterior probability distribution $\phi^t(X, y)$ at time t, which can be expanded as $\phi^t_{c_1}(X, y) \cup \phi^t_{c_2}(X, y) \cdots \cup \phi^t_{c_n}(X, y)$ where c_i represents the distribution for a given class and n the number of classes. As previously defined if $\phi^t(X, y) \neq \phi^{t+C}(X, y)$ a concept drift has occurred and falls into one of the four categories:

- **Single-Class Local Concept Drift**: Given the drifted distribution $\phi^{t+C}(X, y)$ exists **only one** $i \in [0, n]$ that $\phi^{t+C}_{c_i}(X, y) = \hat{\phi}_{c_i}(X, y)$.
- **Single-Class Global Concept Drift**: Given the drifted distribution $\phi^{t+C}(X, y)$ exists **only one** $i \in [0, n]$ that $\phi^{t+C}_{c_i}(X, y) = \psi_{c_i}(X, y)$.
- **Multi-Class Local Concept Drift**: Given the drifted distribution $\phi^{t+C}(X, y)$ exists **more than one** $i \in [0, n]$ that $\phi^{t+C}_{c_i}(X, y) = \hat{\phi}_{c_i}(X, y)$
- **Multi-Class Global Concept Drift**: Given the drifted distribution $\phi^{t+C}(X, y)$ exists **more than one** $i \in [0, n]$ that $\phi^{t+C}_{c_i}(X, y) = \psi_{c_i}(X, y)$.

where $\hat{\phi}_{c_i}(X, y)$ represents a new distribution that partially equal with ϕ_{c_i} for a given class c_i, and $\psi_{c_i}(X, y)$ represents an entirely new probabilistic distribution for a given class c_i.

Figs. 2 and 3 present a visual representation of the four defined categories. Local drifts occur when the new distribution is partially equal to the previous concept. For example, when we have two clusters and only one moves. In our experiments, we considered local changes, changes that affect less than 50% of the original distribution. This type of concept drift can be exceedingly subtle yet has the potential to compromise predictive accuracy in specific regions of the feature space. Consequently, it poses a challenge for detection and resolution. On the other hand, global concept drift involves a complete transformation of the data distribution, often resulting in less subtle changes. However, the ease of drift detection depends on the behavior of the new distribution. It is worth noting that learning mechanisms and concept drift detection strategies can differ significantly in scenarios with multiple classes [6,63]. Thus, when assessing classifier performance or concept drift detection, we must also account for the number of classes affected by the concept drift. This proposed categorization enables us to analyze concept drift detection while considering intricacies that have not received extensive attention in the existing literature. Additionally, it empowers the development of novel classifiers and drift detectors capable of handling a spectrum of concept drifts, ranging from subtle to noticeable.

3.2. Benchmarks

To explicitly assess the performance of classifiers and drift detectors in data streams featuring the concept drift categories outlined earlier, we introduce a set of drift difficulties corresponding to each category within our proposed framework. These difficulties were implemented using two widely recognized data stream generators, namely Random RBF and Random Tree, as established in the literature [5]. The Random RBF algorithm generates N centroids randomly in the feature space. Using a mean value μ and a standard deviation σ, infinite instances can be generated and assigned to the nearest centroid. To introduce concept

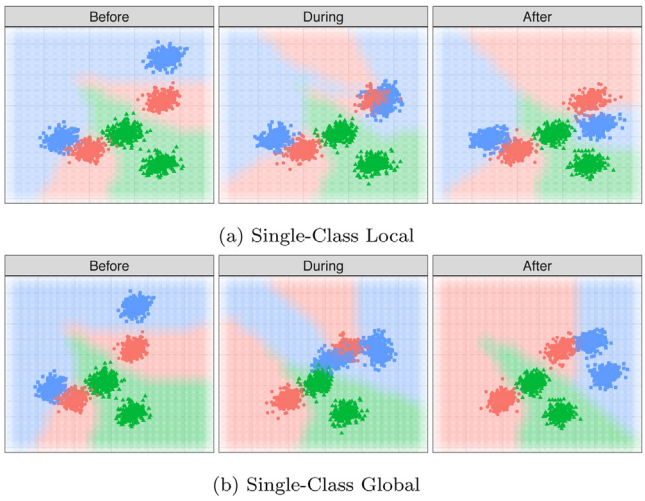


Fig. 2. Illustrative data distribution before, during and after a Single-Class Local (a) and Global (b) concept drift. Each color represents one class and background color represents decision boundaries.

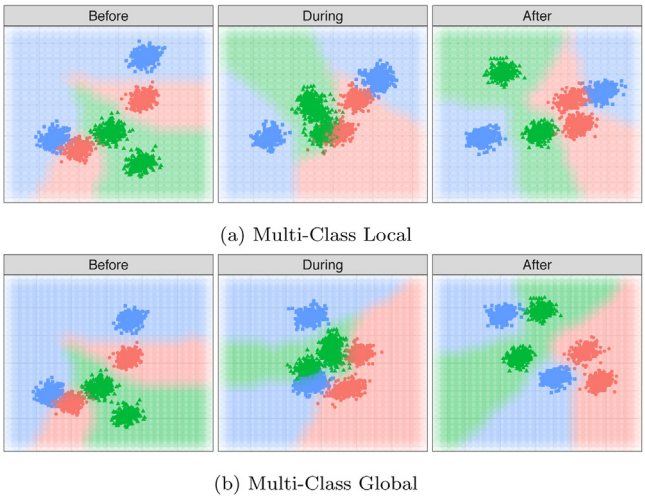


Fig. 3. Illustrative data distribution before, during and after a Multi-Class Local (a) and Global (b) concept drift. Each color represents one class and background color represents decision boundaries.

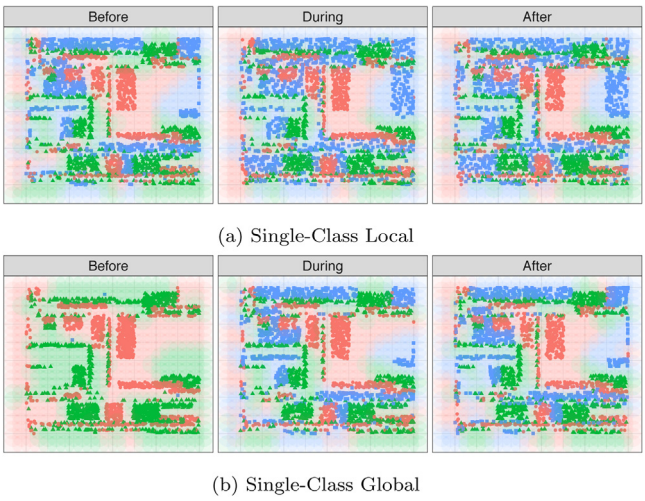


Fig. 4. Data distribution before, during and after a Single-Class Local (a) and Global (b) `emerging_branch` concept drift. Each color represents one class and background color represents decision boundaries.

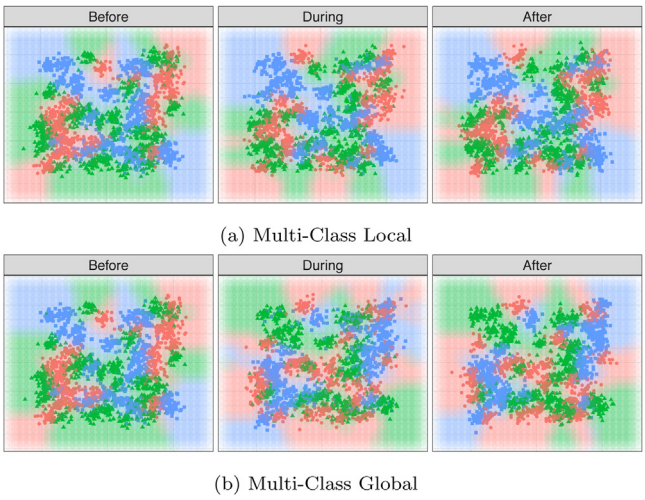


Fig. 5. Data distribution before, during and after a Multi-Class Local (a) and Global (b) `swap_cluster` concept drift with all classes affected. Each color represents one class and background color represents decision boundaries.

drifts, centroids can be moved or the values of μ and σ modified. For the Random Tree algorithm, the generation of instances requires defining the number of features and the desired depth of the tree. With this information, the algorithm randomly generates splits in a decision tree, which is used later to assign labels to randomly generated instances in the feature space. Concept drift can be induced by either completely rebuilding the tree or modifying its leaves and internal nodes.

Table 2 presents a comprehensive overview of the proposed data difficulties along with their respective categorizations. For each difficulty setting, we generated data streams encompassing a range of class counts, including $\{2, 3, 5, 10\}$ classes, and varying feature dimensions of $\{2, 5, 10\}$. Additionally, we introduced three distinct types of drifts: Sudden, Gradual, and Incremental. In the context of Multi-Class drifts, we considered scenarios where drifts affected subsets of classes, including $\{\{2\}, \{2, 3\}, \{2, 3, 5\}, \{2, 3, 5, 10\}\}$, depending on the number of classes. Hence, by systematically exploring all possible permutations of drift speed, drift type, number of classes, and features, we generated a total of 2760 distinct data stream benchmarks. To elaborate, each Single-Class difficulty resulted in 48 data streams for difficulties without incremental drift and 64 with the presence of incremental drift. For Multi-Class concept drifts, we obtained 96 data streams for difficulties without incremental drift and 128 with incremental drifts, as we now also account for the number of classes affected by the drift. This theoretically sums up to 2976 data streams; however, certain scenarios like `reappearing_cluster` and `class_emerging` exclude the possibility of all classes being affected, as this would result in no data. Moreover, Table 3 displays a summary of the specifications of each drift category. Figs. 4 and 5 present a detailed example visual representation of the single and multi-class concept drifts.

Finally, to ensure reproducibility and facilitate the utilization of our proposed categorization and benchmarks, all the data streams used in our experiments have been made publicly available for future research.[1]

4. Experimental setup

The experiments are designed to assess the performance of drift detectors across a diverse range of drift challenges. We also investigate

[1] https://github.com/gabrieljaguiar/locality-concept-drift

Table 2
Concept drift locality benchmark specifications. S: Sudden, G: Gradual, I: Incremental.

Generator	Impact on the class space	Impact on the feature space	Difficulty	Description	Drift speed
RBF	Single-Class	Local	emerging_cluster	A new subcluster emerge	S/G
			reappearing_cluster	Part of the clusters of ONE class disappears for a period of time	S/G
			splitting_cluster	Part of the clusters of ONE class splits into two clusters that move to another direction	S/G/I
			merging_cluster	Part of the clusters of ONE class clusters are merged in the midpoint between them	S/G/I
			moving_cluster	Part of the clusters of ONE class center moves to another position	S/G/I
		Global	reappearing_cluster	All clusters of ONE class disappear for a period of time	S/G
			splitting_subcluster	All subclusters of ONE class splits into two subclusters that move to another direction	S/G/I
			merging_cluster	All subclusters of ONE class of one class merge	S/G/I
			moving_cluster	All clusters of ONE class moves to another position	S/G/I
			class_emerging	A new class appear	S/G
	Multi-Class	Local	emerging_cluster	A new emerging subcluster for N classes appears	S/G
			reappearing_cluster	Part of the clusters of N classes disappears for a period of time	S/G
			splitting_cluster	Part of the clusters of N classes splits into two clusters that move to another direction	S/G/I
			merging_clusters	Part of the clusters are merged in the midpoint between them for N classes	S/G/I
			moving_cluster	Part of the clusters of N classes center moves to another position	S/G/I
			swap_cluster	Part of the clusters of N classes swap position	S/G
		Global	reappearing_cluster	All clusters of N classes disappear for a period of time	S/G
			splitting_cluster	All clusters of N classes splits into two clusters that move to another direction	S/G/I
			merging_cluster	All clusters of N classes center merges	S/G/I
			moving_cluster	All clusters of N classes moves to another position	S/G/I
			swap_cluster	All clusters of N classes swaps position	S/G
RT	Single-Class	Local	emerging_branch	A new branch appears	S/G
			prune_regrowth_branch	Part of the branches of ONE class are pruned and then regrowth this branches	S/G
			prune_growth_new_branch	Part of the branches of ONE class are pruned and then other branches grow	S/G
		Global	prune_regrowth_branch	All of the branches of ONE class are pruned and then regrowth this branches	S/G
			prune_growth_new_branch	All of the branches of ONE class are pruned and then other branches grow	S/G
			class_emerging	A new class appear	S/G
	Multi-Class	Local	emerging_branch	A new branch appears for N classes	S/G
			prune_regrowth_branch	Part of the branches of N classes are pruned and then regrowth this branches	S/G
			prune_growth_new_branch	Part of the branches of N classes are pruned and then other branches grow	S/G
			split_node	Part of the leaves of N classes are splitted to generate two leafs of two different classes	S/G
			swap_leaves	Part of the leaves of N classes swaps with another class	S/G
		Global	prune_regrowth_branch	All of the branches of N classes are pruned and then regrowth this branches	S/G
			prune_growth_new_branch	All of the branches of N classes are pruned and then other branches grow	S/G
			split_node	All of the leaves of N classes are splitted to generate two leafs of two different classes	S/G
			swap_leaves	All of the leaves of N classes swaps with another class	S/G

Table 3
Summary of the proposed benchmark problems.

Drift category	Drift speed	# of classes	# of features	# of instances	# of benchmark streams
No Drift	–	{2, 3, 5, 10}	{2, 5, 10}	100,000	24
Single-Class Local	{1, 1000, 5000, 10000}	{2, 3, 5, 10}	{2, 5, 10}	100,000	384
Single-Class Global	{1, 1000, 5000, 10000}	{2, 3, 5, 10}	{2, 5, 10}	100,000	384
Multi-Class Local	{1, 1000, 5000, 10000}	{2, 3, 5, 10}	{2, 5, 10}	100,000	1092
Multi-Class Global	{1, 1000, 5000, 10000}	{2, 3, 5, 10}	{2, 5, 10}	100,000	876

how the locality and the number of affected classes influence drift detection. The primary goal is to gain insights into the performance of various drift detectors and how each type of drift challenge affects their effectiveness. Additionally, we examine the impact of each challenge on classifiers that rely on explicit drift detectors. To address these objectives, we formulate the following research questions (RQ):

- **RQ1**: Which concept drift detector demonstrates the most effective detection performance across all scenarios?
- **RQ2**: Which concept drift detector excels in detection performance for each distinct type of scenario?
- **RQ3**: How does each category of concept drift influence the performance of the drift detector?
- **RQ4**: Which specific difficulty presents the most challenging scenario?
- **RQ5**: How does the number of classes and features affect the concept drift detection?
- **RQ6**: How does each scenario impact the performance of classifiers utilizing the best-performing drift detector?

4.1. Drift detectors

To assess the performance of drift detectors across various types of concept drift, we chose 9 state-of-the-art supervised drift detectors displayed in Table 4. We made these selections based on their widespread use in the literature and their demonstrated effectiveness in numerous scenarios. It is important to note that we included at least one representative from each group of supervised drift detectors mentioned in Section 2. Our deliberate choice of these detectors was strategic, with the primary objective of comprehensively assessing the behavior, strengths, and weaknesses within each group. Our work is centered on understanding how the locality and magnitude of a concept drift impact its detection. Our focus extends beyond merely identifying the best-performing concept drift, aiming instead to delve into the nuanced dynamics of detection performance.

To monitor drift, we leverage the error distribution of the classifier to effectively monitor concept drifts, as suggested in the existing concept drift literature [31]. The error distribution is binary, and we

Table 4
Drift detectors used in the experiments.

Acronym	Name	Reference
ADWIN	ADaptive Windowing	Bifet and Gavalda [31]
DDM	Drift Detection Method	Gama et al. [1]
ECDD	EWMA for Drift Detection	Ross et al. [30]
EDDM	Early Drift Detection Method	Baena-Garcıa et al. [23]
FHDDM	Fast Hoeffding Drift Detection Method	Pesaranghader and Viktor [26]
FHDDMS	Stacking Fast Hoeffding Drift Detection Method	Pesaranghader et al. [27]
HDDM	Drift Detection Method based on Hoeffding bound	Frias-Blanco et al. [24]
KSWIN	Kolmogorov–Smirnov Windowing	Raab et al. [32]
PH	Page Hinkley	Page [21]
RDDM	Reactive Drift Detection Method	Barros et al. [11]
STEPD	Statistical Test of Equal Proportions	Nishida and Yamauchi [33]

determine the value to update the drift detector using the equation:

$$\phi = \begin{cases} 1, & \text{if } L(x) = y \\ 0, & \text{if } L(x) \neq y \end{cases} \tag{1}$$

where ϕ represents the error distribution, x represents a new instance, y denotes the correct class, and L the classifier. By employing this approach, we can effectively detect concept drifts when there are changes in the classifier error distribution.

4.2. Classifier

To evaluate the performance of the drift detectors, we opted to use Hoeffding Tree (HT) [64] as our classifier. This classifier is known for its exceptional predictive performance and lacks inherent drift detection or adaptation mechanisms, ensuring an unbiased assessment of concept drift detection.

4.3. Performance evaluation

Given that each data stream in our experiments is characterized by a single known drift point, we employed an evaluation process similar to that used in classification tasks. Consequently, we extracted True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values according to the following definitions:

- **TP**: When a concept drift detected at time t and $t \in [i_d, i_d + R]$.
- **TN**: When no alert is raised and there is no known concept drift.
- **FP**: When a concept drift alert is raised at t and $t \notin [i_d, i_d + R]$.
- **FN**: When no alert is raised within the range of detection R.

where i_d is the instance where the concept drift happened, and R is the range of detection, with $R = 5000$ in our experiments. This value of R was defined based on 5% the total size of the stream.

Subsequently, we calculated three performance metrics: Precision (Eq. (2)), Recall (Eq. (3)), and F1-Score (Eq. (4)). Precision assesses the ratio between false alarms and accurate drift alerts, while recall measures the ratio between correct drift detection and undetected drifts. The F1-Score combines both of these metrics. These three metrics in combination provide insights into the behavior of each concept drift detector.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 - Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{4}$$

Furthermore, for each accurate drift alarm, we calculate the average detection delay [65] as described in Eq. (5), where D represents the instance position of the drift, and D_d indicates the instance position when the drift alarm was triggered. This metric assesses the speed of drift detection, and it is measured considering the number of instances between D_d and D.

$$Delay = \frac{\sum(D_d - D)}{TP} \tag{5}$$

4.4. Reproducibility

The source code of the algorithms, the experiments and the data streams are publicly available on GitHub to facilitate the transparency and reproducibility of this research.[2] All results, interactive plots and tables are available on the website[3]. All the experiments, generators and drift detectors were implemented using Python 3.8 and the `river` [66] package. Experiments were run on a GNU/Linux cluster with 192 Intel Xeon cores, 6 TB RAM, and Centos 7.

5. Results

This section presents and discusses the experimental results. Firstly, we conducted a comparative analysis of all drift detectors, irrespective of the specific scenarios, to determine which drift detector exhibits the best overall performance both with and without concept drift. Secondly, we investigate deeper into each scenario to gain insights into how the locality and the proposed data difficulties influence drift detection. Additionally, we evaluated the influence of the number of classes on drift detection in each scenario. In sequence, we assessed how these proposed difficulties affect the classifier's performance and the impact of using a classifier in combination with an explicit drift detector. Finally, we assess the performance of the drift detectors in real-world multi-class benchmark problems.

5.1. Overall aggregated comparison for all scenarios

To address **RQ1**, we first evaluated the performance of all classifiers with a focus on data streams that either had or did not have concept drift, irrespective of how the concept drift affected the feature space or class space. Table 5 presents the average results for 24 data streams without concept drift. In these cases, as the stream remains stationary, we display only the average True Negatives and False Positives values. Notably, only 3 drift detectors did not raise any alert in this scenario, highlighting the sensitivity of explicit drift detectors to changes in the error distribution, even when data distribution remains stable. Moreover, ECDD, EDDM, and STEPD exhibited a high level of sensitivity to any change, resulting in a high number of False Positives. In contrast, DDM and ADWIN displayed the lowest values of false alerts, demonstrating their stability over time.

Table 6 presents the results considering data streams with concept drift. It is interesting to note that EDDM exhibited the lowest delay

[2] https://github.com/gabrieljaguiar/locality-concept-drift

[3] Interactive plots and tables for all experiments are available at https://gabrieljaguiar.github.io/comprehensive-concept-drift/

Table 5
Comparison between drift detectors considering scenarios without concept drift.

Drift detector	TN	FP
ADWIN	0.00	3.79
DDM	0.17	**2.71**
ECDD	**0.46**	904.58
EDDM	0.00	62 914.04
FHDDM	1.00	284.00
FHDDMS	0.00	1058.00
HDDM	0.00	41.83
KSWIN	0.08	10.25
PH	0.00	12.46
RDDM	0.00	27.96
STEPD	0.00	4628.13

Table 6
Comparison between drift detectors considering all evaluated drift difficulties.

Drift detector	Precision	Recall	F-1	Delay
ADWIN	**7.40%**	34.50%	**12.19%**	2192
DDM	1.82%	5.45%	2.73%	2168
ECDD	0.03%	36.48%	0.06%	235
EDDM	0.00%	95.18%	0.00%	15
FHDDM	4.68%	62.43%	8.7%	1771
FHDDMS	1.50%	74.31%	2.9%	1087
HDDM	1.65%	76.61%	3.23%	1271
KSWIN	3.32%	42.54%	6.16%	1866
PH	5.18%	72.19%	9.66%	1867
RDDM	0.89%	25.99%	1.72%	2027
STEPD	0.01%	**95.80%**	0.03%	516

Table 7
Comparison between drift detectors considering single class local concept drifts.

Drift detector	Precision	Recall	F1	Delay
ADWIN	**5.48%**	22.40%	8.80%	2728
DDM	1.88%	5.47%	2.80%	2287
ECDD	0.03%	29.17%	0.07%	201
EDDM	0.00%	**97.40%**	0.00%	4
FHDDM	4.24%	46.61%	7.8%	2045
FHDDMS	1.46%	61.20%	2.9%	1223
HDDM	1.69%	67.45%	3.30%	1359
KSWIN	2.91%	31.51%	5.32%	1991
PH	5.21%	64.06%	**9.63%**	2176
RDDM	1.13%	31.77%	2.18%	2016
STEPD	0.02%	91.15%	0.04%	649

Table 8
Comparison between drift detectors considering single class global concept drifts.

Drift detector	Precision	Recall	F1	Delay
ADWIN	**8.15%**	39.06%	**13.48%**	1909
DDM	2.21%	6.25%	3.27%	2286
ECDD	0.02%	26.30%	0.05%	315
EDDM	0.00%	**91.41%**	0.00%	11
FHDDM	4.84%	53.13%	8.9%	1550
FHDDMS	1.65%	66.67%	3.2%	1074
HDDM	1.81%	69.01%	3.52%	1321
KSWIN	3.30%	34.11%	6.02%	1879
PH	5.87%	75.78%	10.90%	1737
RDDM	1.01%	28.39%	1.96%	1982
STEPD	0.01%	93.75%	0.02%	517

among all evaluated drift detectors and the second-highest recall, although this came at the cost of raising numerous drift alerts, leading to a precision of 0%. A similar pattern is observed for ECDD and STEPD. When comparing DDM and its variations, their detection performance is quite similar, with HDDM displaying the best F1-Score and the shortest detection delay, highlighting the efficiency of Hoeffding's inequality. In terms of F1-Score, ADWIN and PH displayed the best detection performance, with PH having a shorter detection delay.

In summary, ADWIN, DDM, FHDDM, and PH stood out as strong performers in both evaluated scenarios. They exhibited caution when dealing with stationary streams and demonstrated good detection performance when concept drift occurred. On the other hand, EDDM, ECDD, and STEPD could detect concept drifts quickly but raised a significant number of false alarms in the process.

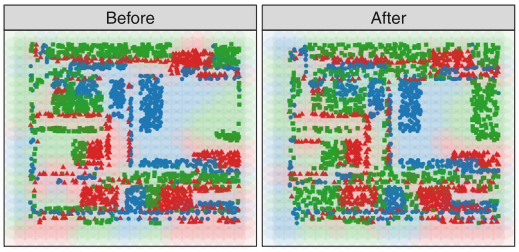
5.2. Detailed comparison within each scenario

Tables 7 and 8 present the metrics for Single-Class Local and Global concept drift scenarios. When we analyze the detection performance, it is evident that ADWIN and PH consistently exhibited the best detection performance for both local and global concept drifts, while EDDM, ECCD, and STEPD displayed the least favorable results. Comparing the differences in results between local and global concept drift, we observe that precision values remained relatively stable. However, recall values increased for most drift detectors, with a 17% increase for ADWIN and a 10% increase for PH. This indicates an inverse correlation between false alarms and the magnitude of the concept drift. On the other hand, the detection rate improved from local to global drifts. This is understandable since global concept drifts have a more substantial impact on data distribution, making them easier to detect due to their significant influence.

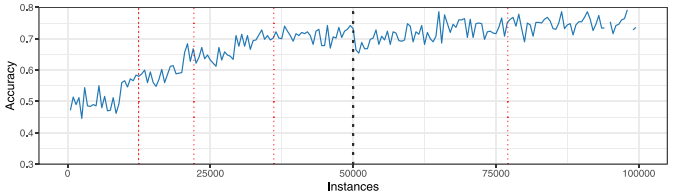
When considering Multi-Class concept drift scenarios, with results presented in Tables 9 and 10, ADWIN and PH again demonstrated the best results for both scenarios. Similar to the observations in Single-Class concept drift, when we compare local and global drifts, we notice that global drifts were detected more effectively due to their more substantial impact on classifier performance. Furthermore, it is noteworthy that Multi-Class Local drifts exhibited higher recall values than their single-class counterparts. This is expected since a larger portion of the data distribution is influenced by the drift in multi-class scenarios, making it easier to detect and leading to improved recall values.

Additionally, Figs. 6 to 9 illustrate four examples of the data distribution, accuracy of the classifier, and the moments when the ADWIN (the most effective overall) signaled drifts. These visualizations provide insight into the ease or difficulty in detecting specific scenarios. In the case of Local drifts (Figs. 6 and 8), the drift alerts were signaled when no actual drift occurred, indicating that detecting Local drifts is challenging due to their subtle nature. Conversely, in Global drifts (Figs. 7 and 9), a more distinct alteration in the data distribution was observable, making it easier to identify and leading to more noticeable changes in classifier accuracy. Both Single and Multi scenarios exhibited false alarms, where drift detection alerts were raised despite no actual drift occurring during those periods.

To address **RQ2**, ADWIN and PH consistently demonstrated superior detection performance across all evaluated scenarios. Conversely, EDDM and STEPD, while achieving high recall values, also triggered numerous false alarms, resulting in lower precision. It is worth noting that DDM consistently exhibited low values of both recall and precision in all scenarios, indicating a more conservative approach. Furthermore, when comparing the behavior of drift detectors in the four scenarios, a hierarchy of difficulty emerges, from easiest to hardest detection as follows: Multi-Class Global, Single-Class Global, Multi-Class Local, Single-Class Local. This order of difficulty corresponds closely to the impact on data distribution, with a more pronounced impact leading to a more significant change in error distribution. Another noteworthy observation is that in all evaluated drift detectors, scenarios with more localized drifts tended to generate a higher number of false alarms. This highlights the need for the development of drift detectors that are less sensitive to error distribution, particularly in scenarios with local drifts. In conclusion, our analysis provides insights into how each scenario affected the performance of drift detectors, addressing RQ3.

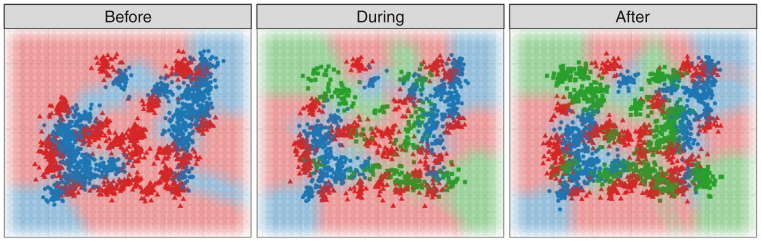


(a) Data distribution before and after a concept drift where new boundaries for the green class are created.

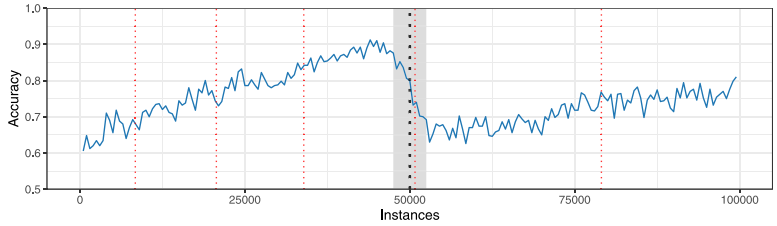


(b) Hoeffding Tree accuracy over time for a specific stream experiencing concept drift. The gray rectangle denotes the duration of the drift. Red dashed lines represent the drift alarms triggered by ADWIN.

Fig. 6. Single-Class Local sudden drift data distribution and drift alerts over time considering the `emerging_branch` difficulty and ADWIN as drift detector.



(a) Data distribution before, during and after a concept drift where a new class (green) emerges.



(b) Hoeffding Tree accuracy over time for a specific stream experiencing concept drift. The gray rectangle denotes the duration of the drift. Red dashed lines represent the drift alarms triggered by ADWIN.

Fig. 7. Single-Class Global drift data distribution and drift alerts over time considering the `class_emerging` difficulty and ADWIN as drift detector.

Table 9
Comparison between drift detectors considering multi class local concept drifts.

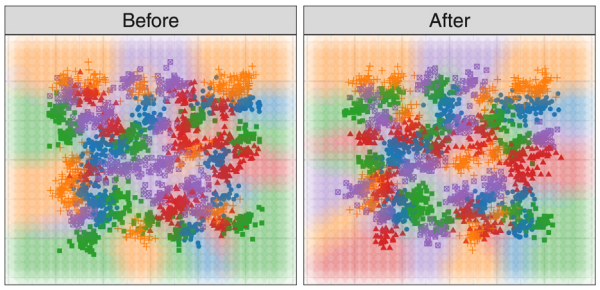
Drift detector	Precision	Recall	F1	Delay
ADWIN	**7.19%**	30.40%	**11.63%**	2360
DDM	1.58%	4.67%	2.36%	2036
ECDD	0.04%	40.29%	0.07%	192
EDDM	0.00%	**95.51%**	0.00%	21
FHDDM	4.60%	64.10%	8.6%	1830
FHDDMS	1.48%	77.11%	2.9%	1092
HDDM	1.67%	80.68%	3.28%	1207
KSWIN	3.31%	44.32%	6.16%	1873
PH	5.18%	70.88%	9.66%	1956
RDDM	0.92%	26.37%	1.77%	2083
STEPD	0.02%	96.70%	0.03%	508

Table 10
Comparison between drift detectors considering multi class global concept drifts.

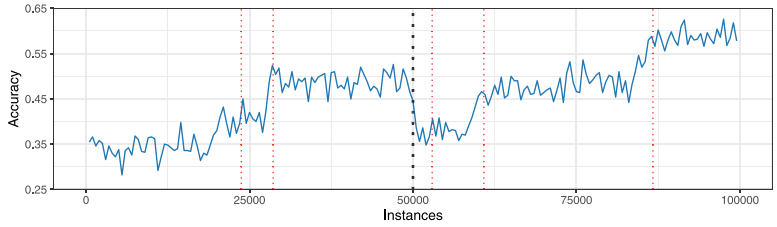
Drift detector	Precision	Recall	F1	Delay
ADWIN	**7.95%**	42.92%	**13.41%**	2033
DDM	1.94%	6.05%	2.93%	2195
ECDD	0.03%	39.38%	0.06%	277
EDDM	0.00%	**95.43%**	0.00%	14
FHDDM	4.86%	71.35%	9.1%	1699
FHDDMS	1.48%	79.91%	2.9%	1039
HDDM	1.56%	78.88%	3.06%	1299
KSWIN	3.48%	48.86%	6.49%	1819
PH	4.91%	75.80%	9.22%	1706
RDDM	0.72%	21.92%	1.39%	1977
STEPD	0.01%	97.60%	0.03%	469

5.3. Comparison between difficulties

In this experiment, our aim is to assess the varying levels of difficulty encountered by each drift detector when dealing with different concept drift scenarios. We seek to analyze which scenarios pose more significant challenges and which ones are relatively easier to detect for each drift detector. To accommodate space limitations within the manuscript, we will primarily focus on presenting the results for the

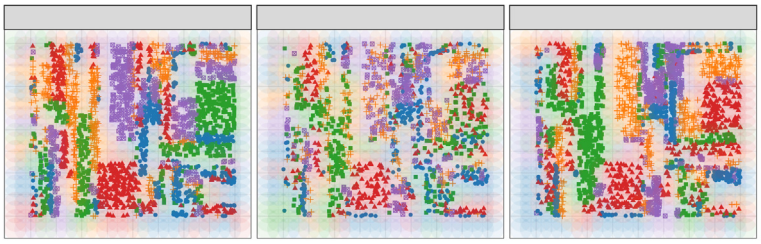


(a) Data distribution before and after a concept drift that swaps positions of some clusters of all classes.

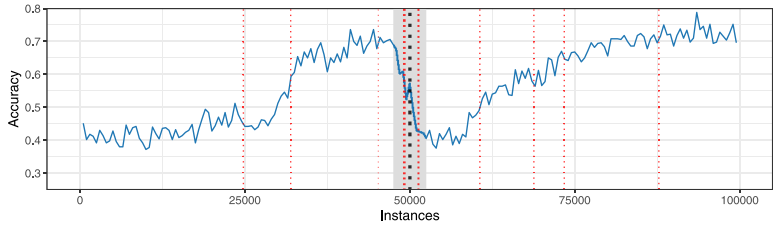


(b) Hoeffding Tree accuracy over time for a specific stream experiencing concept drift. The gray rectangle denotes the duration of the drift. Red dashed lines represent the drift alarms triggered by ADWIN.

Fig. 8. Multi-Class Local drift data distribution and drift alerts over time considering the `swap_cluster` difficulty and ADWIN as drift detector.



(a) Data distribution before, during and after a concept drift that swaps positions of all classes.



(b) Hoeffding Tree accuracy over time for a specific stream experiencing concept drift. The gray rectangle denotes the duration of the drift. Red dashed lines represent the drift alarms triggered by ADWIN.

Fig. 9. Multi-Class Global drift data distribution and drift alerts over time considering the `swap_leaves` difficulty and ADWIN as drift detector.

top-performing drift detectors, ADWIN and Page Hinkley. The complete set of results is accessible on our website for a more detailed examination.

When considering Single-Class Local drifts, Table 11 provides an overview of the performance metrics for each difficulty as observed with both drift detectors. It is worth noting that, for both detectors, the `reappearing_cluster` scenario appeared to be the easiest to detect. This is because the disappearance and reappearance of data segments tend to induce substantial changes in error distribution, making them relatively clear for the drift detectors, even if they occur in specific regions of the feature space. Conversely, when faced with more complex drift scenarios, ADWIN displayed its worst performance in detecting the `emerging_branch` difficulty, while PH struggled with the `moving_cluster` scenario. The differences in their performance can be attributed to their distinct detection mechanisms. ADWIN employs a two-window approach that relies on detecting a significant difference between the two windows to identify concept drift. Consequently, when the classifier rapidly adapts to the new data distribution, ADWIN may encounter challenges in detecting the drift. On the other hand, PH utilizes a cumulative approach, allowing it to handle such scenarios more effectively. Additionally, difficulties that inherently involve incremental changes, such as those related to moving, merging, and splitting clusters, displayed a higher detection delay in general.

Furthermore, Table 12 provides an overview of the results for each of the Single-Class Global difficulties. Notably, for both ADWIN and PH, the `class_emerging` difficulty appeared to be the least challenging to detect and exhibited the lowest detection delay. This observation aligns with the logic that the emergence of a new class results in a sudden drop in accuracy, as a novel class is being introduced and evaluated. Conversely, the most challenging difficulty to detect was the `moving_cluster`, followed by `splitting_cluster` and `merging_cluster`. These three difficulties are inherently incremental in

Table 11
Comparison of detection performance for Single-Class Local drift difficulties by ADWIN and Page Hinkley. Bold values represent the highest value of that column (easiest detection) and underscored values represent the lowest value (harder detection).

Difficulty	Precision		Recall		F1		Delay	
	ADWIN	PH	ADWIN	PH	ADWIN	PH	ADWIN	PH
emerging_branch	3.83%	5.24%	14.58%	**70.83%**	6.06%	9.76%	2176	1975
emerging_cluster	5.61%	5.41%	25.00%	62.50%	9.16%	9.95%	3364	2249
merging_cluster	5.15%	5.39%	20.83%	62.50%	8.26%	9.92%	2914	2285
moving_cluster	5.03%	4.94%	20.83%	58.33%	8.10%	9.11%	2930	2050
prune_growth_new_branch	4.47%	5.09%	16.67%	66.67%	7.05%	9.45%	2257	2067
prune_regrowth_branch	7.14%	4.94%	22.92%	64.58%	10.89%	9.17%	2479	2261
reappearing_cluster	**7.35%**	**5.56%**	**37.50%**	66.67%	**12.29%**	**10.26%**	2550	2179
splitting_cluster	4.95%	5.16%	20.83%	60.42%	8.00%	9.51%	2936	2368

nature, which means they do not create an abrupt change in the learning curve and error rate. Consequently, this incremental nature makes them harder to detect promptly. Comparing the results to those of Single-Class Local drifts, an improvement in recall values is noticeable. This suggests that, for almost all difficulties, detecting Global drifts was generally easier due to their broader impact on the data distribution.

Analyzing the metrics for Multi-Class Local drifts in Table 13, ADWIN achieved the best results when encountering the `prune_growth_new_branch` difficulty, while PH exhibited its best performance with `swap_clusters`. Notably, results for difficulties that do not involve incremental drifts were quite similar to the best results. This observation emphasizes the challenge that drift detectors face in effectively detecting incremental drifts, even when multiple classes are affected. Another aspect worth noting is the larger performance gap between ADWIN and PH in the context of multi-class drifts compared to single-class drifts. This difference can be attributed to the higher precision that ADWIN demonstrates in multi-class drift scenarios. When we make comparisons with their Single-Class Local counterparts, it becomes evident that multi-class drifts generally yield better detection results, as expected due to their greater impact on data distribution.

The results for Multi-Class Global drifts are presented in Table 14. ADWIN displayed its best detection performance when confronted with `prune_growth_new_branch` drifts, but it also achieved the satisfactory F1 values for other difficulties, such as `prune_regrowth_branch` and `swap_leaves`. The poorest results for ADWIN once again correlated with difficulties involving incremental drift. PH, on the other hand, exhibited its best performance with `reappearing_cluster`, although its performance with other difficulties was quite similar and generally less favorable than that of ADWIN. In comparison to Multi-Class Local drifts, precision values remained similar, but recall values increased, indicating improved detection even with a notable number of false alarms.

To address **RQ4**, it becomes evident that sudden changes in data distribution, like new data appearing in the feature space or the emergence of a new class, were the easiest and quickest to be detected. Conversely, difficulties that inherently involved incremental changes displayed higher values of False Negatives, resulting in lower recall and less effective detection. It is important to note that this observation is correlated with how the base learner operates. Consequently, when evaluating drift detectors with other base learners or employing unsupervised drift detectors, the ranking of difficulties in terms of complexity may vary.

5.4. Impact of number of classes and number of features

Furthermore, the benchmark dataset used in the experiments included streams with varying configurations in terms of the number of classes and features. Therefore, we assessed how these different specifications influenced drift detection while considering each drift categorization to address **RQ5**. Fig. 10 presents a boxplot showing the F1 score for the top 2 performing detectors, i.e., ADWIN and Page Hinkley in each drift category, comparing each configuration regarding

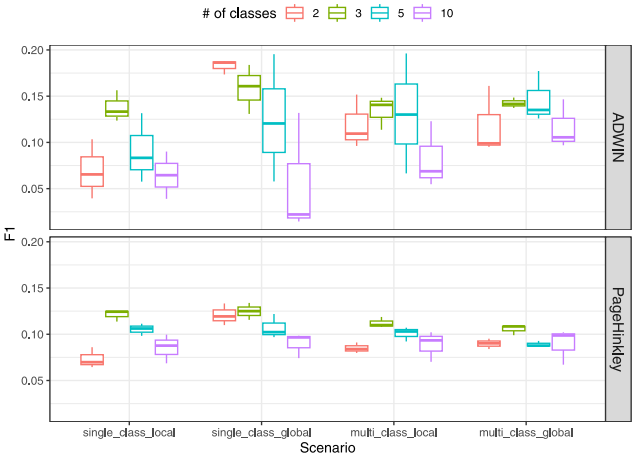


Fig. 10. F1 Score for each categorization of concept drift. Each color represents streams with different number of classes.

numbers of classes, and Fig. 11 considering different feature space dimensionality.

Firstly, it is evident that as the number of classes in the stream increases, the detection of concept drifts becomes more challenging, regardless of how the stream is affected. Moreover, when considering Single-Class Global drifts, a noticeable drop in performance is observed. This is expected in this specific scenario because when only one class is affected, the more stationary the other classes are, the more localized the change becomes, making it difficult to detect. On the other hand, when multiple classes are affected globally, the increase in the number of classes did not have a significant impact on drift detection. The scenario that posed the most challenge for both drift detectors was Single-Class Local drift with only 2 classes. In a binary setting where only a portion of the classes is affected, it is similar to an imbalanced scenario, and these changes may not significantly impact the classifier's error.

When considering different feature space configurations, we can see that this factor did not have a substantial impact on Page Hinkley's performance. However, it did affect ADWIN, particularly in the presence of Single-Class Local and Multi-Class Local drifts. Additionally, it is worth noting that ADWIN's performance exhibits higher variance, while Page Hinkley demonstrates consistent performance across various stream configurations.

5.5. Impact on classifier performance

Finally, in addition to assessing how the proposed scenarios and difficulties influence concept drift detection, we also examined their impact on the performance of the utilized classifier, the Hoeffding Tree, when used in conjunction with the best-performing drift detector. The primary objective of this experiment is to ascertain how each

Table 12
Comparison of detection performance for Single-Class Global drift difficulties by ADWIN and Page Hinkley. Bold values represent the highest value of that column (easiest detection) and underscored values represent the lowest value (harder detection).

Difficulty	Precision		Recall		F1		Delay	
	ADWIN	PH	ADWIN	PH	ADWIN	PH	ADWIN	PH
class_emerging_rbf	10.07%	**7.68%**	**58.33%**	**93.75%**	17.18%	**14.20%**	1104	1698
class_emerging_rt	**11.06%**	6.55%	52.08%	85.42%	**18.25%**	12.17%	1198	1280
merging_cluster	5.24%	5.17%	20.83%	60.42%	8.37%	9.52%	2895	2130
moving_cluster	4.37%	4.32%	20.83%	54.17%	7.22%	8.00%	2885	2089
prune_growth_new_branch	9.60%	5.68%	39.58%	77.08%	15.45%	10.59%	2378	1689
prune_regrowth_branch	9.21%	5.54%	43.75%	83.33%	15.22%	10.39%	1722	1487
reappearing_cluster	9.62%	6.76%	58.33%	91.67%	16.52%	12.59%	2124	1896
splitting_cluster	4.50%	5.22%	18.75%	60.42%	7.26%	9.60%	2985	1899

Table 13
Comparison of detection performance for Multi-Class Local drift difficulties by ADWIN and Page Hinkley. Bold values represent the highest value of that column (easiest detection) and underscored values represent the lowest value (harder detection).

Difficulty	Precision		Recall		F1		Delay	
	ADWIN	PH	ADWIN	PH	ADWIN	PH	ADWIN	PH
emerging_branch	**11.53%**	5.35%	45.37%	76.85%	18.39%	10.01%	2474	1644
emerging_cluster	6.82%	5.15%	27.78%	64.81%	10.95%	9.54%	3016	2329
merging_cluster	4.65%	5.14%	19.44%	64.81%	7.50%	9.52%	2783	2032
moving_cluster	4.66%	4.86%	20.37%	63.89%	7.59%	9.03%	2799	2042
prune_growth_new_branch	11.40%	5.37%	**49.07%**	80.56%	**18.50%**	10.08%	1983	1618
prune_regrowth_branch	4.37%	3.88%	15.00%	58.33%	6.77%	7.27%	2650	1970
reappearing_cluster	11.37%	5.69%	48.33%	73.33%	18.41%	10.56%	2696	1749
split_node	3.32%	4.50%	12.04%	61.11%	5.21%	8.38%	1777	2387
splitting_cluster	4.82%	5.58%	20.37%	67.59%	7.80%	10.31%	3013	2213
swap_cluster	7.76%	**5.97%**	42.59%	**85.19%**	13.12%	**11.15%**	1995	1842
swap_leaves	8.23%	5.23%	35.19%	78.70%	13.33%	9.80%	1675	1835

Table 14
Comparison of detection performance for Multi-Class Global drift difficulties by ADWIN and Page Hinkley. Bold values represent the highest value of that column (easiest detection) and underscored values represent the lowest value (harder detection).

Difficulty	Precision		Recall		F1		Delay	
	ADWIN	PH	ADWIN	PH	ADWIN	PH	ADWIN	PH
merging_cluster	4.37%	4.53%	20.37%	60.19%	7.20%	8.43%	2917	1889
moving_cluster	4.93%	4.30%	27.78%	66.67%	8.37%	8.08%	3148	1885
prune_growth_new_branch	**11.52%**	5.19%	63.89%	87.04%	**19.52%**	9.80%	1666	1540
prune_regrowth_branch	10.19%	5.14%	55.00%	90.00%	17.19%	9.72%	1865	1485
reappearing_cluster	9.31%	**5.51%**	58.33%	85.00%	16.06%	**10.34%**	1797	1705
split_node	7.24%	4.79%	25.93%	67.59%	11.31%	8.94%	2129	2061
splitting_cluster	4.97%	4.62%	20.37%	58.33%	7.99%	8.55%	2987	2177
swap_cluster	8.76%	5.12%	**69.44%**	**91.67%**	15.56%	9.70%	1847	1263
swap_leaves	9.79%	5.13%	57.41%	86.11%	16.73%	9.68%	1657	1608

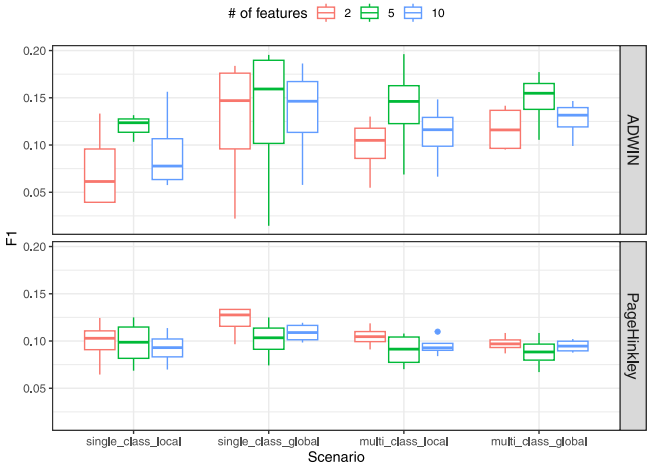


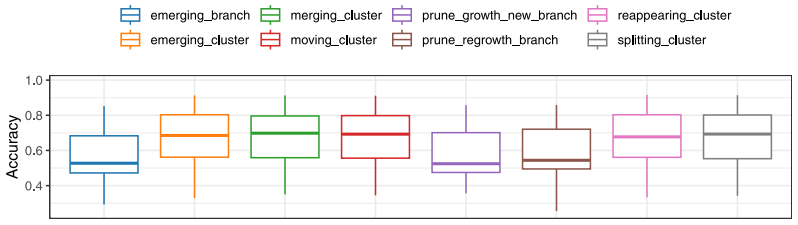
Fig. 11. F1 Score for each categorization of concept drift. Each color represents streams with different number of features.

of the proposed scenarios affects classification performance and to determine whether combining them with a drift detector enhances overall performance.

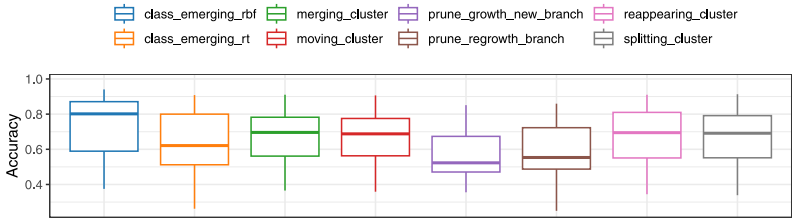
Figs. 12 and 13 present the accuracy corresponding to each drift difficulty. Analyzing the average accuracy enables identification of the drifts causing the most significant drop in accuracy, given that, prior to the drift, all data streams from the same generator are equal.

In the Single-Class scenario, particularly focusing on local drifts, incremental changes such as emerging, merging, moving, and splitting clusters proved to be the most challenging to detect. However, classifiers exhibited a capacity for self-adaptation and showed their best accuracy levels. The emergence of new branches posed the most complex scenario in terms of predictive performance. Furthermore, streams generated with Random Tree generator displayed lower accuracy compared to those with RBF. In terms of global changes, the results were similar to local changes, except for `class_emerging`, which exhibited higher accuracy values. This is likely due to the reduction in the number of classes for the majority of the stream, leading to improved accuracy.

Within the Multi-Class local and global scenarios, performance aligned closely with what was observed in Single-Class drifts. Incremental drifts demonstrated the highest accuracy levels, whereas `emerging_branch` and `swap_leaves` presented the most challenging difficulties. Once again, the Random Tree generator proved more

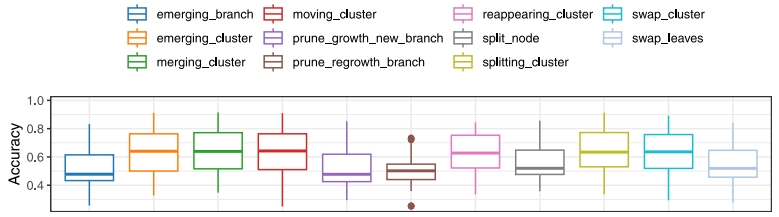


(a) Single-Class Local

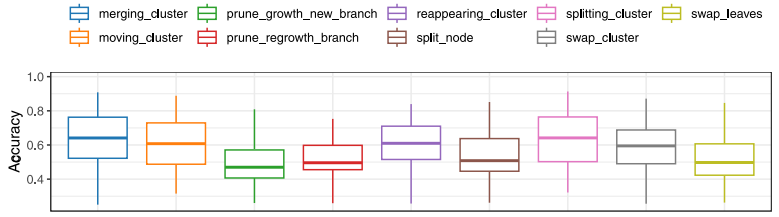


(b) Single-Class Global

Fig. 12. Boxplot of HT accuracy for each of the proposed difficulties for Single-Class Local (a) and Global (b).



(a) Multi-Class Local



(b) Multi-Class Global

Fig. 13. Boxplot of HT accuracy for each of the proposed difficulties for Multi-Class Local (top) and Global (bottom).

challenging to learn from than RBF, which could be attributed to the distribution of clusters. As feature space dimensions increased, the RBF model facilitated easier learning of class boundaries.

Furthermore, we conducted a comparison to understand the impact of integrating the best-performing drift detector, i.e. ADWIN, on the classifier's performance. For this analysis, we compared the standard classifier with two modified versions: Drift/Warning Retraining (HT-DW) and Adaptive Hoeffding Tree (AHT). The former initiates retraining of the classifier upon the detector signaling a warning and completely replaces the classifier when a drift is detected. In contrast, the latter employs a drift detector for each branch of the tree, replacing only the affected branch when a drift signal is detected, while maintaining the overall structure of the tree. This differentiates HT-DW's global approach from AHT's localized strategy in addressing concept drift within the classifier structure.

Table 15 displays the average accuracy for each drift category across the three classifiers. Overall, the AHT strategy, focusing on addressing drift locally, improved the accuracy of the standard Hoeffding Tree by an average of 4.03%. This improvement underscores the robustness of the Hoeffding Tree in adapting to new concepts, exhibiting good accuracy values even without a drift detector. On the other hand, despite the utilization of a drift detector, completely retraining the classifier resulted in decreased accuracy across all evaluated scenarios. Notably, the more localized the drift category, the more pronounced the reduction in accuracy.

Additionally, in Figs. 14 to 17, the accuracy of each classifier under the `prune_growth_new_branch`, `reappearing_cluster`, `reappearing_cluster` and `reappearing_cluster` difficulties is displayed, including sudden and gradual drifts. These figures serve as an illustration of the classifiers' behavior across different drift categories. Across Single-Class drifts, both local and global, AHT consistently demonstrated superior accuracy. Even in the presence of drift, HT showcased a remarkable self-adaptation ability, recovering swiftly. Conversely, the retraining strategy resulted in notably poorer performance. Notably, the sole scenario where retraining displayed an advantage was in the context of Multi-Class Global drifts. This specific category provoked a significant change in data distribution, evident

Table 15
Accuracy of HT and AHT for evaluated drift categories.

Drift category	HT	AHT	HT-DW
Single-Class Local	66.88% ± 0.148	70.71% ± 0.127 (↑ 3.83%)	50.13% ± 0.135 (↓ 16.75%)
Single-Class Global	68.16% ± 0.148	71.61% ± 0.132 (↑ 3.44%)	52.55% ± 0.157 (↓ 15.61%)
Multi-Class Local	60.78% ± 0.147	65.18% ± 0.126 (↑ 4.39%)	44.92% ± 0.121 (↓ 15.87%)
Multi-Class Global	58.80% ± 0.144	63.28% ± 0.128 (↑ 4.48%)	45.52% ± 0.122 (↓ 13.28%)

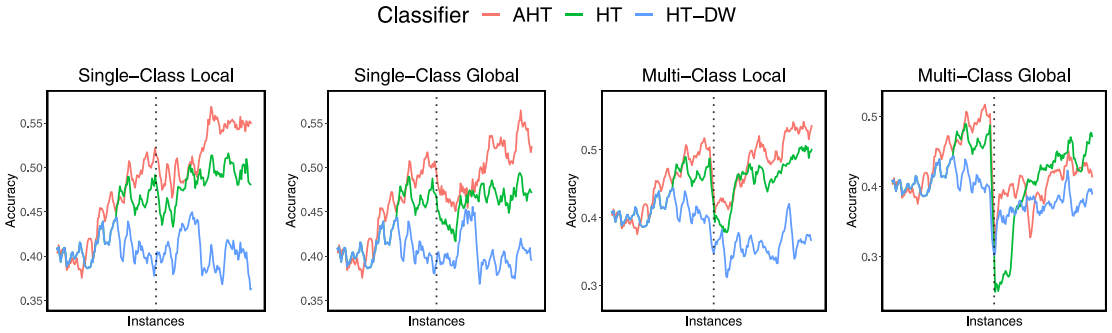


Fig. 14. Accuracy curve for AHT, HT and HT-DW on `prune_growth_new_branch` data stream in the presence of a sudden concept drift. Dashed vertical line indicates where the concept drift happened.

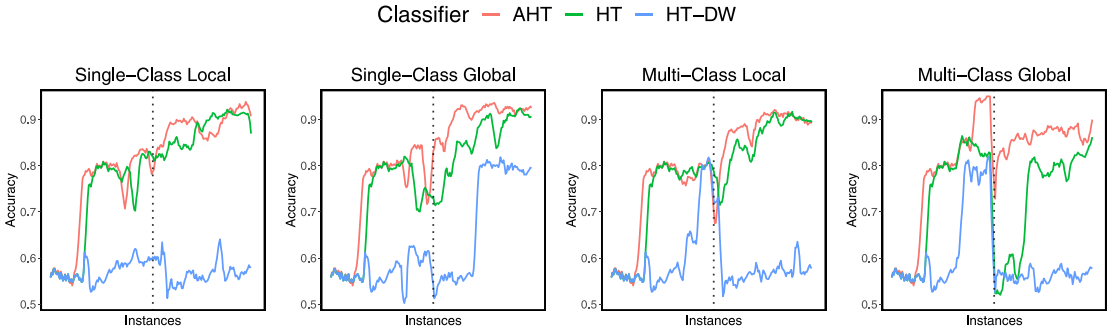


Fig. 15. Accuracy curve for AHT, HT and HT-DW on `reappearing_cluster` data stream in the presence of a sudden concept drift. Dashed vertical line indicates where the concept drift happened.

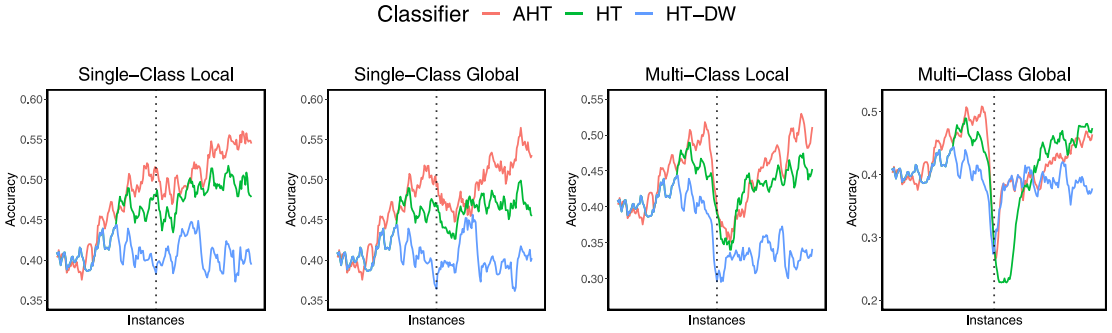


Fig. 16. Accuracy curve for AHT, HT and HT-DW on `prune_growth_new_branch` data stream in the presence of a gradual concept drift. Dashed vertical line indicates where the concept drift happened.

from the notable accuracy drop. Given the entirely new data distribution, retraining the classifier proved a valid method for recovering from these drifts. However, by the end of the stream, the plain HT exhibited better accuracy compared to both drift-adapted classifiers.

5.6. Comparison in real-world scenarios

Real-world data streams pose unique challenges to classifiers, including the latency with which instances from a specific class arrive or extended periods during which instances from only one class appear. This configuration of data streams introduces additional challenges for streaming classifiers.

In the context of drift detectors, many real-world problems do not provide information about when a concept drift occurs, making evaluation challenging. However, Souza et al. [67] addressed this issue by creating 8 data streams with the presence of Multi-Class Global concept drifts. They had prior knowledge of the characteristics and patterns of changes to adequately evaluate new adaptive algorithms. Table 16 presents a summary of the real-world benchmark problems.

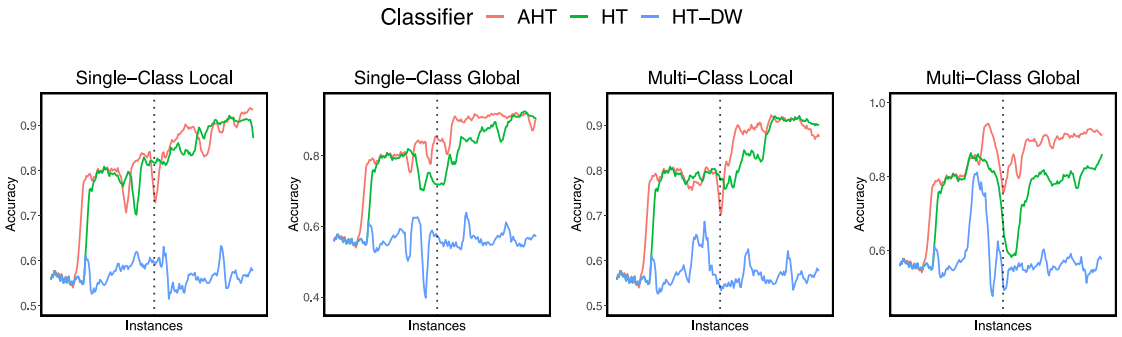


Fig. 17. Accuracy curve for AHT, HT and HT-DW on `reappearing_cluster` data stream in the presence of a gradual concept drift. Dashed vertical line indicates where the concept drift happened.

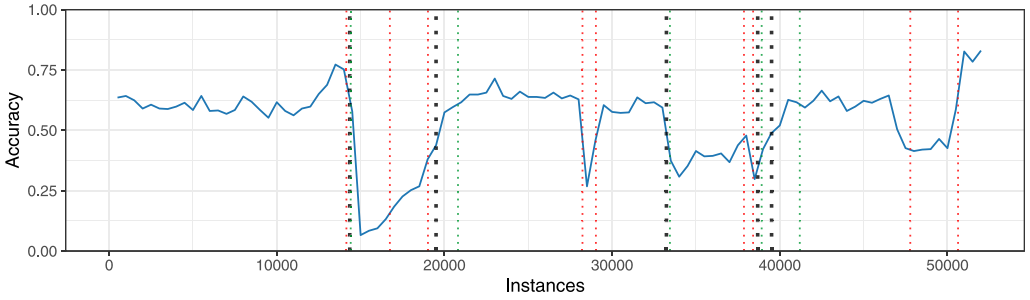


Fig. 18. Hoeffding Tree accuracy over time for Abrupt (balanc.) stream experiencing concept drift. The gray dashed line denotes the change points. Red dashed lines represent the FP drift alarms triggered by ADWIN and Green dashed lines represent the TP drift alarms.

Table 16
INSECTS datasets specifications.

Datasets	# of instances	Drift points
Abrupt (balanc.)	52,848	14,352; 19,500; 33,240; 38,682; 39,510
Abrupt (imb.)	355,275	83,859; 128,651; 182,320; 242,883; 268,380
Gradual (balanc.)	24,150	14,028
Gradual (imb.)	143,323	58,159
Incremental(balanc.)	79,986	26,568; 53,364
Incremental (imb.)	452,044	150,683; 301,365
Reoccurring (balanc.)	79,986	26,568; 53,364
Reoccurring (imb.)	452,044	150,683; 301,365

As the change point is known, we were able to evaluate the drift detectors using the same methodology employed in the generated benchmarks. Table 17 presents the overall values of F1 and Delay, along with the average ranking of each drift detector. The results closely resembled those observed in scenarios with generated data streams. ADWIN exhibited the best performance, achieving the top average ranking and the highest F1 value in five of the datasets, ranking second in the remaining three. In addition to ADWIN, PH and FHDDM also demonstrated commendable results in the real-world datasets. On the other hand, ECDD and EDDM presented the least favorable outcomes, with nearly zero correct detections and elevated false positive values.

Furthermore, Fig. 18 displays the accuracy curve, change points, and instances where ADWIN triggered alarms for concept drifts in the Abrupt balanced dataset. While ADWIN accurately identified all concept drifts, it also produced numerous false positives. In a scenario where prompt adaptation is crucial upon detecting a concept drift, such a high rate of false positives is undesirable.

Moreover, Table 18 provides the average accuracy of the assessed classifiers with ADWIN as the drift detector. AHT exhibited superior accuracy compared to HT-DW, although the margin was not as substantial as observed in the generated scenarios. This can be attributed to the numerous false alarms raised by ADWIN, triggering retraining when unnecessary. Furthermore, the incorporation of a drift detector to the HT classifier enhanced its performance by an average of 4.11%, showcasing the robustness of HT to concept drifts even without an explicit mechanism. Fig. 19 illustrates the accuracy curve for each classifier across all datasets. For balanced datasets, AHT and HT-DW consistently exhibit higher accuracy throughout the entire stream, whereas in imbalanced scenarios, defining the best-performing classifier is less straightforward.

6. Lessons learned and recommendations

In order to summarize the knowledge we extracted through the experimental evaluation, this section presents the lessons learned and recommendations for future researchers.

Locality matters. The locality of concept drift plays a pivotal role in the accuracy and adaptability of detectors when detecting changes in data streams. When dealing with the detection of concept drift, understanding its locality provides insights into how changes within the data stream, affecting either the entire stream or specific parts of it, influence the classification system. For instance, in scenarios where drift affects only particular sections, such as individual clusters or subsets of classes, it is critical to discern the precise areas within the data stream that undergo these changes. Different sections might require varied adaptations to maintain predictive performance. Moreover, different types of drift, local or global, might require different strategies to handle them effectively. Understanding these subtleties can significantly impact the accuracy and performance of the drift detection and adaptation methods used in learning models.

Handling high number of classes. The learning task becomes more challenging for both classifiers and drift detectors as the number of classes increases. Our study indicated that when the impact is localized due to fewer classes being affected, the detection of the change becomes more difficult. Hence, drift detectors capable of handling a larger number of classes play a crucial role in effective drift detection. Furthermore, future research should aim at detecting drifts within

Table 17
Comparison between drift detectors considering real-world datasets.

Dataset	ADWIN		DDM		ECDD		EDDM		FHDDM		FHDDMS	
	F1	Delay	F1	Delay	F1	Delay	F1	Delay	F1	Delay	F1	Delay
Abrupt (balanc.)	**47.62%**	698	36.36%	390	0.00%	0	0.04%	2	31.58%	1576	23.26%	1415
Abrupt (imb.)	**25.81%**	327	0.00%	0	0.00%	0	0.00%	1.2	17.78%	294	6.41%	326.4
Gradual (balanc.)	33.33%	3196	**40.00%**	3628	0.00%	0	0.00%	0	28.57%	92	14.29%	22
Gradual (imb.)	**12.50%**	473	0.00%	0	0.00%	0	0.00%	0	5.71%	744	2.04%	42
Incremental (balanc.)	15.38%	138	0.00%	0	0.00%	0	0.01%	4	16.00%	29	7.14%	12.5
Incremental (imb.)	**8.16%**	320	0.00%	0	0.00%	0	0.00%	2	5.41%	13	1.58%	133.5
Reoccurring (balanc.)	**17.39%**	314	0.00%	0	0.00%	0	0.01%	12.5	8.33%	13	3.23%	5.5
Reoccurring (imb.)	3.85%	522	**10.00%**	141	0.00%	0	0.00%	1.5	2.41%	790	1.43%	1635
Avg. Rank	**1.75**	3.75	6.25	7.50	9.62	9.50	8.87	8.50	3.37	4.37	5.62	5.00

Dataset	HDDM		KSWIN		PH		RDDM		STEPD	
	F1	Delay	F1	Delay	F1	Delay	F1	Delay	F1	Delay
Abrupt (balanc.)	16.67%	558	38.10%	1156	35.71%	257	15.79%	669	0.14%	700
Abrupt (imb.)	3.90%	400	15.09%	590	11.94%	91	2.06%	344	0.05%	188
Gradual (balanc.)	11.76%	1388	18.18%	34	22.22%	2706	14.29%	2950	0.05%	4
Gradual (imb.)	2.02%	961	0.00%	0	6.45%	724	0.00%	0	0.02%	14
Incremental (balanc.)	6.35%	515	**17.39%**	18	10.26%	599	3.85%	1681	0.03%	12
Incremental (imb.)	1.82%	1822	7.27%	1865	4.21%	119	0.00%	0	0.01%	16
Reoccurring (balanc.)	3.70%	75	9.52%	6.5	12.90%	1258	0.00%	0	0.02%	0
Reoccurring (imb.)	1.55%	1398	5.88%	1882	4.35%	645	1.28%	5	0.02%	114
Avg. Rank	6.00	**3.25**	3.13	4.00	3.38	4.63	7.63	5.75	7.88	7.00

Table 18
Accuracy of HT and AHT for each INSECTS dataset.

Dataset	HT	AHT	HT-DW
Abrupt (balanc.)	53.64%	61.94% (↑ 08.30%)	**63.14%** (↑ 09.51%)
Abrupt (imb.)	**66.12%**	66.00% (↓ 00.12%)	60.02% (↓ 06.10%)
Gradual (balanc.)	60.22%	61.84% (↑ 01.62%)	**65.22%** (↑ 05.00%)
Gradual (imb.)	57.62%	**63.42%** (↑ 05.80%)	61.99% (↑ 04.37%)
Incremental (balanc.)	57.38%	64.09% (↑ 06.71%)	**64.99%** (↑ 07.62%)
Incremental (imb.)	**64.54%**	62.66% (↓ 01.88%)	57.61% (↓ 06.93%)
Reoccurring (balanc.)	53.17%	65.20% (↑ 12.03%)	**68.04%** (↑ 14.87%)
Reoccurring (imb.)	62.92%	**63.33%** (↑ 00.41%)	57.55% (↓ 05.38%)
Average	59.45%	**63.56%** (↑ 04.11%)	62.32% (↑ 02.87%)

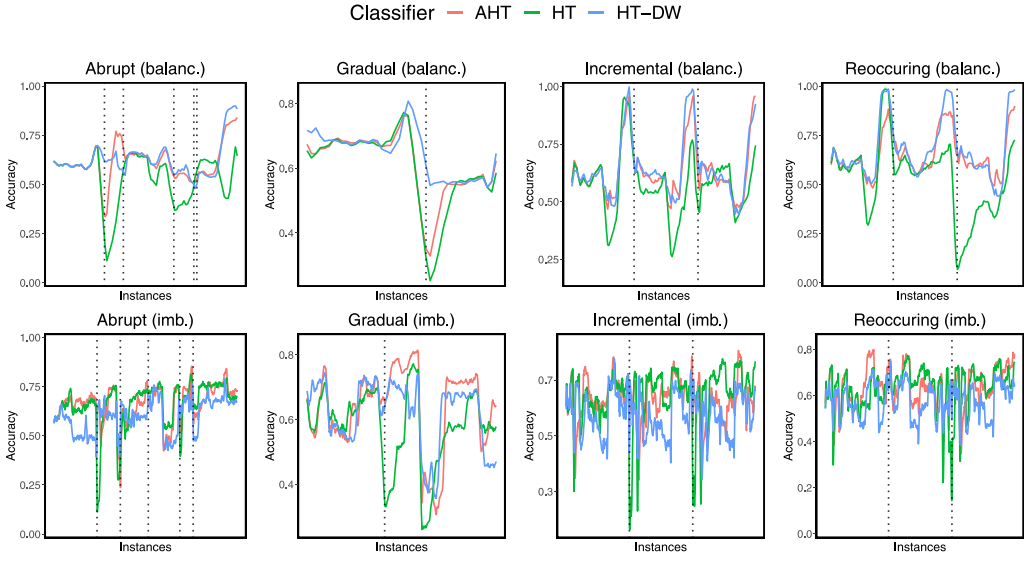


Fig. 19. Accuracy curve for AHT, HT and HT-DW on real-world INSECTS dataset. Dashed vertical line indicates where the concept drift happened.

individual classes, a strategy that could facilitate the adaptation of classifiers to new concepts.

False alarms. Drift detectors that rely on error rates often generate numerous false alarms. Given their univariate nature, even positive changes in the error rate, such as when a new branch is created in an incremental tree or a classifier establishes a new boundary, can trigger alarms in certain drift detectors, which is counterproductive. In more complex scenarios, such as localized changes, false alarms can mislead the classifier, resulting in poorer performance. For future

research, it is essential that drift detectors strike a better balance between detection and false alarms by examining not only the error curve but also statistics related to data distribution.

The common strategy to address concept drift involves replacing the classifier once a drift is identified. While this approach works well with ensembles due to their varied classifiers, our experiments have shown that when handling drifts that only affect specific boundaries or when the classifier is capable of self-adaptation, avoiding complete replacement of the established boundaries results in better performance. Furthermore, retraining the classifier for each identified drift may lead to redundant work due to the high occurrence of false alarms, even with state-of-the-art drift detectors. This emphasizes the necessity for drift detectors capable of pinpointing which class or segment of the data stream was affected by the drift. This, in turn, facilitates a more precise adaptation of the classifier rather than a complete replacement.

Retraining is not (always) the best option. The common strategy to address concept drift involves replacing the classifier once a drift is identified. While this approach works well with ensembles due to their varied classifiers, our experiments have shown that when handling drifts that only affect specific boundaries or when the classifier is capable of self-adaptation, avoiding complete replacement of the established boundaries results in better performance. Furthermore, retraining the classifier for each identified drift may lead to redundant work due to the high occurrence of false alarms, even with state-of-the-art drift detectors. This emphasizes the necessity for drift detectors capable of pinpointing which class or segment of the data stream was affected by the drift. This, in turn, facilitates a more precise adaptation of the classifier rather than a complete replacement.

7. Conclusion and future work

In this paper, we presented a comprehensive study focusing on benchmarking and evaluating the impact of concept drift, specifically concerning its locality and magnitude, on classifiers and drift detectors. We introduced a novel categorization of concept drift, considering its locality and scale. Through a systematic approach, we identified significant challenges within this domain, formulated, and evaluated a set of 2760 data stream benchmarks that encompass various levels of difficulty, guided by our proposed categorization. Additionally, we conducted a comparative evaluation of 9 state-of-the-art drift detectors across a diverse range of difficulties. Each drift detector was thoroughly assessed across various isolated and combined scenarios. This analysis not only identifies the top-performing detectors but also sheds light on their specific strengths, providing valuable insights for future research in drift detection. Moreover, we evaluated how the locality of drift influences the base-classifier's performance, gaining knowledge into the most effective approaches for addressing different categories of concept drift to minimize recovery time. This comprehensive study offered insights into different and yet unexplored categorizations of concept drift, providing an understanding of how drift detectors and classifiers perform across numerous difficulties. All benchmark problems and evaluation methodologies are publicly available, enabling future researchers to design efficient approaches, drift detectors, and classifiers to handle local concept drifts effectively.

Our future work aims to explore the influence of concept drift locality on data streams characterized by imbalanced class distributions. Additionally, our plans include an investigation into the performance of unsupervised and semi-supervised drift detectors within the proposed scenarios. Furthermore, we seek to develop classifiers with the capacity to handle concept drift locally, avoiding the necessity for retraining or complete replacement.

CRediT authorship contribution statement

Gabriel J. Aguiar: Writing – original draft, Software, Data curation. **Alberto Cano:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code and data are shared on Github and the link is provided in the manuscript.

References

[1] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian Symposium on Artificial Intelligence, 2004.
[2] M. Bahri, A. Bifet, J. Gama, H.M. Gomes, S. Maniu, Data stream analysis: Foundations, major tasks and tools, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. (2021).
[3] J. Gama, Knowledge Discovery from Data Streams, CRC Press, 2010.
[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. (2014).
[5] G. Aguiar, B. Krawczyk, A. Cano, A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework, Mach. Learn. (2023).
[6] Ł. Korycki, B. Krawczyk, Concept drift detection from multi-class imbalanced data streams, in: IEEE International Conference on Data Engineering, 2021.
[7] A.D. Viniski, J.P. Barddal, A. de Souza Britto Jr., F. Enembreck, H.V.A. de Campos, A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start, Expert Syst. Appl. (2021).
[8] A.L. Suárez-Cetrulo, D. Quintana, A. Cervantes, A survey on machine learning for recurring concept drifting data streams, Expert Syst. Appl. (2023).
[9] N. Lu, J. Lu, G. Zhang, R.L. De Mantaras, A concept drift-tolerant case-base editing technique, Artificial Intelligence (2016).
[10] A. Liu, Y. Song, G. Zhang, J. Lu, Regional concept drift detection and density synchronized drift adaptation, in: International Joint Conference on Artificial Intelligence, 2017.
[11] R.S. Barros, D.R. Cabral, P.M. Gonçalves Jr., S.G. Santos, RDDM: Reactive drift detection method, Expert Syst. Appl. (2017).
[12] E.B. Gulcan, F. Can, Unsupervised concept drift detection for multi-label data streams, Artif. Intell. Rev. (2023).
[13] J. Gama, G. Castillo, Learning with local drift detection, in: International Conference on Advanced Data Mining and Applications, 2006.
[14] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, M. Woźniak, Ensemble learning for data stream analysis: A survey, Inf. Fusion (2017).
[15] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE Trans. Knowl. Data Eng. (2018).
[16] A.R. Masegosa, A.M. Martínez, D. Ramos-López, H. Langseth, T.D. Nielsen, A. Salmerón, Analyzing concept drift: A case study in the financial sector, Intell. Data Anal. (2020).
[17] G.I. Webb, R. Hyde, H. Cao, H.L. Nguyen, F. Petitjean, Characterizing concept drift, Data Min. Knowl. Discov. (2016).
[18] B. Krawczyk, A. Cano, Online ensemble learning with abstaining classifiers for drifting and noisy data streams, Appl. Soft Comput. (2018).
[19] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: A survey, IEEE Comput. Intell. Mag. (2015).
[20] R.S.M. Barros, S.G.T.C. Santos, A large-scale comparison of concept drift detectors, Inform. Sci. (2018).
[21] E.S. Page, Continuous inspection schemes, Biometrika (1954).
[22] S. Roberts, Control chart tests based on geometric moving averages, Technometrics (2000).
[23] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: International Workshop on Knowledge Discovery from Data Streams, 2006.
[24] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Díaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on hoeffding's bounds, IEEE Trans. Knowl. Data Eng. (2014).
[25] S. Micevska, A. Awad, S. Sakr, SDDM: an interpretable statistical concept drift detection method for data streams, J. Intell. Inf. Syst. (2021).
[26] A. Pesaranghader, H.L. Viktor, Fast hoeffding drift detection method for evolving data streams, in: European Conference on Machine Learning and Knowledge Discovery in Databases, 2016.
[27] A. Pesaranghader, H. Viktor, E. Paquet, Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams, Mach. Learn. (2018).
[28] A. Pesaranghader, H.L. Viktor, E. Paquet, Mcdiarmid drift detection methods for evolving data streams, in: International Joint Conference on Neural Networks, 2018.

[29] R.S.M. de Barros, J.I.G. Hidalgo, D.R. de Lima Cabral, Wilcoxon rank sum test drift detector, Neurocomputing (2018).
[30] G.J. Ross, N.M. Adams, D.K. Tasoulis, D.J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recognit. Lett. (2012).
[31] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: SIAM International Conference on Data Mining, 2007.
[32] C. Raab, M. Heusinger, F.-M. Schleif, Reactive soft prototype computing for concept drift streams, Neurocomputing (2020).
[33] K. Nishida, K. Yamauchi, Detecting concept drift using statistical testing, in: International Conference on Discovery Science, 2007.
[34] H. Moharram, A. Awad, P.M. El-Kafrawy, Optimizing ADWIN for steady streams, in: ACM/SIGAPP Symposium on Applied Computing, 2022.
[35] P.M. Grulich, R. Saitenmacher, J. Traub, S. Breß, T. Rabl, V. Markl, Scalable detection of concept drifts on data streams with parallel adaptive windowing, in: International Conference on Extending Database Technology, 2018.
[36] D.T.J. Huang, Y.S. Koh, G. Dobbie, R. Pears, Detecting volatility shift in data streams, in: IEEE International Conference on Data Mining, 2014.
[37] F. Pinagé, E.M. dos Santos, J. Gama, A drift detection method based on dynamic classifier selection, Data Min. Knowl. Discov. (2020).
[38] J. Komorniczak, P. Ksieniewicz, Complexity-based drift detection for nonstationary data streams, Neurocomputing (2023).
[39] P. Wang, H. Yu, N. Jin, D. Davies, W.L. Woo, QuadCDD: A quadruple-based approach for understanding concept drift in data streams, Expert Syst. Appl. (2024).
[40] H. Yu, J. Li, J. Lu, Y. Song, S. Xie, G. Zhang, Type-LDD: A type-driven lite concept drift detector for data streams, IEEE Trans. Knowl. Data Eng. (2023).
[41] B. Halstead, Y.S. Koh, P. Riddle, M. Pechenizkiy, A. Bifet, Combining diverse meta-features to accurately identify recurring concept drift in data streams, ACM Trans. Knowl. Discov. Data (2023).
[42] G. Aguiar, A. Cano, Enhancing concept drift detection in drifting and imbalanced data streams through meta-learning, in: IEEE International Conference on Big Data, 2023.
[43] B. Halstead, Y.S. Koh, P. Riddle, R. Pears, M. Pechenizkiy, A. Bifet, G. Olivares, G. Coulson, Analyzing and repairing concept drift adaptation in data stream classification, Mach. Learn. (2022).
[44] A. Łapiński, B. Krawczyk, P. Ksicnicwicz, M. Woźniak, An empirical insight into concept drift detectors ensemble strategies, in: IEEE Congress on Evolutionary Computation, 2018.
[45] P. Sobolewski, M. Woźniak, Comparable study of statistical tests for virtual concept drift detection, in: International Conference on Computer Recognition Systems, 2013.
[46] X. Song, M. Wu, C. Jermaine, S. Ranka, Statistical change detection for multi-dimensional data, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007.
[47] A.A. Qahtan, B. Alharbi, S. Wang, X. Zhang, A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015.
[48] F. Gu, G. Zhang, J. Lu, C.-T. Lin, Concept drift detection based on equal density estimation, in: International Joint Conference on Neural Networks, 2016.
[49] L. Bu, C. Alippi, D. Zhao, A pdf-free change detection test based on density difference estimation, IEEE Trans. Neural Netw. Learn. Syst. (2016).
[50] J. Xuan, J. Lu, G. Zhang, Bayesian nonparametric unsupervised concept drift detection for data stream mining, ACM Trans. Intell. Syst. Technol. (2020).
[51] Y.-N. Wan, B.P. Jaysawal, J.-W. Huang, Unsupervised concept drift detection using dynamic crucial feature distribution test in data streams, in: International Conference on Technologies and Applications of Artificial Intelligence, 2022.
[52] A. Liu, J. Lu, F. Liu, G. Zhang, Accumulating regional density dissimilarity for concept drift detection in data streams, Pattern Recognit. (2018).
[53] N. Lu, G. Zhang, J. Lu, Concept drift detection via competence models, Artificial Intelligence (2014).
[54] Ö. Gözüaçık, A. Büyükçakır, H. Bonab, F. Can, Unsupervised concept drift detection with a discriminative classifier, in: 28th ACM International Conference on Information and Knowledge Management, 2019.
[55] V. Cerqueira, H.M. Gomes, A. Bifet, L. Torgo, STUDD: A student–teacher method for unsupervised concept drift detection, Mach. Learn. (2023).
[56] P.M. Gonçalves Jr., S.G. de Carvalho Santos, R.S. Barros, D.C. Vieira, A comparative study on concept drift detectors, Expert Syst. Appl. (2014).
[57] S.G. Santos, R.S. Barros, P.M. Gonçalves Jr., A differential evolution based method for tuning concept drift detectors in data streams, Inform. Sci. (2019).
[58] E.S. Babüroğlu, A. Durmuşoğlu, T. Dereli, Novel hybrid pair recommendations based on a large-scale comparative study of concept drift detection, Expert Syst. Appl. (2021).
[59] L. Poenaru-Olaru, L. Cruz, A. van Deursen, J.S. Rellermeyer, Are concept drift detectors reliable alarming systems? A comparative study, in: IEEE International Conference on Big Data, 2022.
[60] M. Mahgoub, H. Moharram, P. Elkafrawy, A. Awad, Benchmarking concept drift detectors for online machine learning, in: International Conference on Model and Data Engineering, 2022, pp. 43–57.
[61] G.Y. Sakurai, J.F. Lopes, B.B. Zarpelão, S. Barbon Junior, Benchmarking change detector algorithms from different concept drift perspectives, Future Internet (2023).
[62] D. Brzezinski, L.L. Minku, T. Pewinski, J. Stefanowski, A. Szumaczuk, The impact of data difficulty factors on classification of imbalanced and concept drifting data streams, Knowl. Inf. Syst. (2021).
[63] M. Lango, J. Stefanowski, What makes multi-class imbalanced problems difficult? An experimental study, Expert Syst. Appl. (2022).
[64] G. Holmes, R. Kirkby, B. Pfahringer, Stress-testing hoeffding trees, in: European Conference on Principles and Knowledge Discovery in Databases, 2005.
[65] A. Liu, J. Lu, Y. Song, J. Xuan, G. Zhang, Concept drift detection delay index, IEEE Trans. Knowl. Data Eng. 35 (5) (2022) 4585–4597.
[66] J. Montiel, M. Halford, S.M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H.M. Gomes, J. Read, T. Abdessalem, et al., River: machine learning for streaming data in python, J. Mach. Learn. Res. (2021).
[67] V.M. Souza, D.M. dos Reis, A.G. Maletzke, G.E. Batista, Challenges in benchmarking stream learning algorithms with real-world data, Data Min. Knowl. Discov. (2020).