



A benchmark and survey of fully unsupervised concept drift detectors on real-world data streams

Daniel Lukats^{1,2} · Oliver Zielinski^{1,3} · Axel Hahn^{2,4} · Frederic Stahl¹

Received: 24 April 2024 / Accepted: 27 July 2024 / Published online: 27 August 2024
© The Author(s) 2024

Abstract

Concept drift detection techniques can be used to discover substantial changes of the patterns encoded in data streams in real-time. If left unaddressed, these changes can render deployed machine learning models unreliable because their training data no longer matches the patterns present in the data stream. Most algorithms proposed in the literature depend on the immediate availability of ground truth class labels. This is unrealistic for many applications due to the associated cost of labeling. Therefore, this study reviews the availability of fully unsupervised concept drift detectors, which can operate entirely without labeled data. Ten algorithms are analyzed in terms of architectural choices, core ideas and assumptions about data because they fulfilled several inclusion criteria designed to ensure faithful and reliable implementations. Seven of these algorithms are evaluated with common concept drift detection metrics on eleven real-world data streams; the remaining three performed too slow or depended on chance. Based on the results of these experiments, three concept drift detectors—*Discriminative Drift Detector*, *Image-Based Drift Detector* and *Semi-Parametric Log-Likelihood*—can be recommended depending on the desired target metric. This study further reveals issues with the evaluation metrics *Mean Time Ratio* and *lift-per-drift*. Finally, it highlights open research challenges.

Keywords Unsupervised concept drift detection · Non-stationary data analysis · Metrics for concept drift evaluation · Benchmark data streams

1 Introduction

A wealth of data is generated in real-time in the form of data streams [1, 2], e.g., in network traffic monitoring systems [3, 4], in internet-of-things networks [5] or in environmental observatories [6, 7]. Classic machine learning methods operate under the assumption that data is stationary, i.e., that the data a model is deployed on is similar to the data it was trained on. In long-running data, this assumption does

not hold, when data is generated continuously and is therefore non-stationary. Instead, patterns encoding the incoming data may change in such a manner that deployed models can no longer provide reliable predictions, giving rise to a phenomenon called *concept drift* [8].

Various methods were proposed to address the issue of concept drift, either by making predictive models themselves adaptive [9] or by detecting concept drifts with methods such as *Drift Detection Method (DDM)* [10] or *Adaptive Windowing (ADWIN)* [11] to allow manual adaptation. Most concept drift detectors proposed in the literature operate in a supervised manner; they monitor the predictive performance of a classifier deployed on the data stream. These detectors require immediate access to ground truth information about the class labels—which is unrealistic in many applications due to associated cost or limited accessibility. Any detector that detects concept drift without online access to class labels of the observed data is called *unsupervised*. In addition to detectors which do not require labeled data at any time, this term also applies to those concept drift detectors which require class labels in an offline pre-training phase

✉ Daniel Lukats
Daniel.Lukats@dfki.de

¹ Marine Perception, German Research Center for Artificial Intelligence, Marie-Curie-Straße 1, 26129 Oldenburg, Germany

² Computing Sciences, Carl von Ossietzky Universität Oldenburg, Ammerländer Heerstraße 114-118, 26129 Oldenburg, Germany

³ Leibniz Institute for Baltic Sea Research Warnemünde, Seestraße 15, 18119 Rostock, Germany

⁴ Systems Engineering for Future Mobility, German Aerospace Center, Escherstraße 2, 26121 Oldenburg, Germany

and operate without labeled data once deployed on the data stream. The key distinction between these two approaches is that the latter methods assume that labels are available for pre-training and can be used for concept drift detection [12, 13].

Both supervised and unsupervised concept drift detectors can be used in diverse real-world applications, e.g., network intrusion detection [14, 15], spam detection [16], solar irradiance forecasting [17], landslide detection [18] or predictive maintenance [19].

Although many data streams are associated with classification tasks, which pre-trained methods such as L-CODE [12] or EMAD [13] can leverage, other data streams come entirely unlabeled or are otherwise not associated with any kind of classification tasks. For this reason, this study is concerned with fully unsupervised concept drift detectors. For example, data streams from coastal observatories provide information from different sensors in real-time [6, 7]. Although no classification tasks are associated with these data streams as of now, concept drift detection is desired for these data streams to support scientific operations [20]. Various surveys addressing supervised concept drift detection are available [8, 21–23]. Barros and Santos benchmarked supervised concept drift detectors and ensembles thereof in two studies [24, 25]. Moreover, Gemaque et al. [26] provide a broader view on unsupervised methods, highlighting mostly those which require labeled data in an offline pre-training phase. A study by Suárez-Cetrulo et al. [27] reviews the state of concept drift detectors for recurring concept drift. Finally, Xiang et al. [28] review the state of deep learning methods for concept drift detection, which is sparsely covered in the other literature reviews.

Here, in addition to highlighting available methods from the literature, 7 detectors are implemented for evaluation on a choice of 11 real-world data streams. Common metrics from the literature are used in this study to evaluate the predictive performance of the detectors: The proxy metrics classifier predictive performance [29] and *lift-per-drift* [30] evaluate a concept drift detector with the help of a classifier deployed on the data stream. Classifier predictive performance is known to be flawed, as there is a bias towards more frequent detection on several real-world data streams [29]. On one data stream ground truth information about concept drifts is available enabling the use of *Mean Time Ratio*, which directly assesses the detection rate, detection time and time between false alerts [29]. This study aims to address the following research questions: Which detectors are available in the literature and suitable for implementation? Which detectors detect concept drift well? Given classifier predictive performance's bias, is *lift-per-drift* an unbiased proxy metric? Finally, implementing and benchmarking these detectors also reveals open issues with the current state of the art in both

fully unsupervised concept drift detectors and their evaluation.

Hence, this study makes the following contributions: **Firstly**, it provides implementations of 7 unsupervised concept drift detectors, made available under the 3-clause BSD license. **Secondly**, it offers extensive evaluation of these detectors on 11 real-world data streams with several metrics identifying the best performing detectors. The results of these experiments are made available alongside the source code on GitHub¹ to ensure reproducibility and enable further research. Lastly, it provides a comparison of the different metrics used and reveals a few open issues.

This paper is structured as follows: Firstly, a definition of concept drift is given and different ways how concept drifts can manifest are highlighted in Sect. 2. Then the methodology for the literature review is stated in Sect. 3. Section 4 follows with a review of the literature, highlighting typical architectural choices and introduces the implemented algorithms. The setup of the experimental evaluation is explained in Sect. 5 and the corresponding results are shown and are discussed in Sect. 6. Finally, concluding remarks are given in Sect. 8.

2 Concept drift

A concept drift denotes a change in the probability distributions governing a data stream. In contrast to outliers, which are just a few data points outside of the regular distribution of the data, a concept drift marks a longer lasting change.

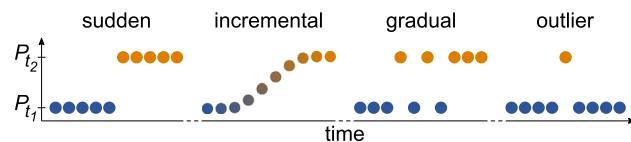
Usually, two types of concept drift are described, *real* and *virtual concept drift* [8]. *Real concept drift* means changes in the posterior distribution such that $P_{t_1}(y | X) \neq P_{t_2}(y | X)$ given $t_1 \neq t_2$, t_1 and t_2 being different points in time. X denotes the features of the data excluding the label or target feature, which is denoted y instead. In contrast to this, *virtual concept drift or covariate shift* denotes a change in the distribution of the features: $P_{t_1}(X) \neq P_{t_2}(X)$, $t_1 \neq t_2$. Supervised concept drift detectors compare a classifier's predictions \hat{y} to the true class label y . In contrast to this, the unsupervised detectors benchmarked in this study observe the features X only because real-time access to the classification ground truth is unrealistic in many scenarios (see Table 1). By virtue of operating on the feature space only, these unsupervised concept drift detectors cannot detect concept drift in the posterior distribution unless it is accompanied by a covariate shift.

Concept drifts are further characterized by the nature of the transition from one concept to another (see Fig. 1) [8]: *Sudden or abrupt concept drifts* transition from one concept to another instantly, e.g., during extreme events such as a

¹ <https://github.com/DFKI-NI/unsupervised-concept-drift-detection>.

Table 1 An overview of the different spaces observed and required by supervised, pre-trained unsupervised and fully unsupervised concept drift detectors

Type	Input	Requires y
Supervised	\hat{y} and y	Always
Pre-trained Unsupervised	X	Offline
Fully Unsupervised	X	Never

**Fig. 1** The transition from the current concept P_{t_1} to a new concept P_{t_2} can occur in different ways: Concept drifts can be sudden, incremental or gradual. In contrast to these long-lasting changes in the data, an outlier is just a single or a few data points out of the usual concept. The Figure was adapted from [8, 23]

breach in a dike or when drifting sensors are calibrated. A smooth transition over several time steps is called *incremental*. For example, temperature changes over the course of a year can be incremental. In contrast to this, *gradual concept drift* denotes transitions, during which incoming data may come from either concept. The transition from one variant of the SARS-CoV-2 coronavirus in the population is a *gradual drift*, as multiple variants can coexist at the same time. Finally, concepts may be *recurring*, as a data stream may return to a concept that was observed before, e.g., the day and night cycle in environmental data or seasonal changes over multiple years. Recurring concepts do not characterize the type of change, therefore recurring concepts may be combined with any of the kinds of concept drift.

3 Literature search

The search for literature on unsupervised concept drift detection was conducted on Google Scholar and Semantic Scholar ordering by recency. Literature prior to 2012 was not included, as no publication prior to 2012 that is referenced in the literature meets the inclusion criteria, which are given further below. Furthermore, search results were filtered by examining abstracts for any mention or relation to unsupervised concept drift detection. Search terms were constructed from the components highlighted in Table 2.

Since detectors needed to be implemented for the benchmark in this study, inclusion criteria were defined to ensure correct implementations and to keep the required work to a sensible level. Common issues impeding implementation attempts are discussed in greater detail in [31]. When algorithm descriptions are given in prose or are incomplete,

Table 2 Components of the search terms used in the literature search

1st component	2nd	3rd	4th
Unsupervised unlabeled	Concept	Drift Change Shift	Detection detector

Search terms always consisted of 4 components for a total of 12 combinations

implementing the respective algorithm is difficult and may result in many errors. Hence, these criteria filter both publications that exceed the scope of this study and those which lack sufficient detail to allow a reliable and faithful implementation:

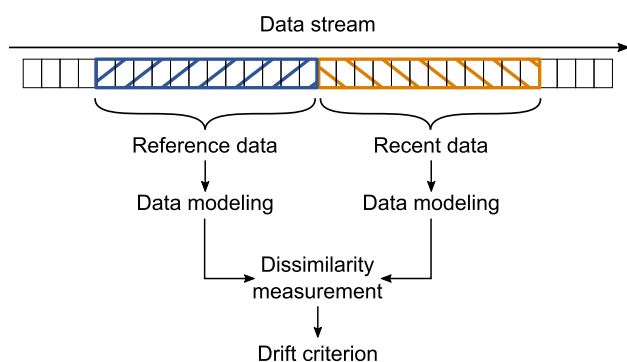
1. The paper must propose a detector. This criterion exists, since several publications related to unsupervised concept drift detection rely on existing algorithms without proposing a novel algorithm themselves. These publications typically propose concept drift detection pipelines or methods to understand the nature of the detected drift.
2. The detector must work in a fully unsupervised manner; any method that requires labels is outside the scope of this paper, as explained in the introduction (see Sect. 1).
3. The proposed algorithm must be capable of operating on multivariate data, as most data streams are multivariate, including all data streams used in this study's evaluation.
4. Owing to the real-world data streams provided in [32], detectors need to be able to perform on continuous data; many of these data streams contain continuous features, which some methods cannot operate on without modifications [33, 34].
5. The algorithm must be described either through a pseudocode listing or a single mathematical definition to ensure reproducibility, as reconstructing an algorithm from prose is prone to errors.
6. The algorithm must be described with sufficient detail, so no major architectural or algorithmic choices need to be assumed. Typical violations of this criterion include undefined symbols or a lack of information on handling edge cases such as the occasional division by zero or function inputs that are not within the function's domain [31].²

The literature search yielded 61 publications related to unsupervised concept drift detection. 51 publications violate the inclusion criteria, leaving 10 algorithms for further analysis. Table 3 highlights the distribution of inclusion criteria violations. Note that criteria were evaluated in order 1–5 and no further criteria were evaluated after a violation.

² Usually, violations of this criterion only became apparent during attempts to implement the respective detector.

Table 3 Distribution of inclusion criteria violations by the publications identified in the literature search, percentages rounded to 0 decimal places

Criterion	Rejections	% of papers
1	6	~ 10
2	21	~ 34
3	5	~ 8
4	4	~ 7
5	12	~ 20
6	3	~ 5
Total	51	~ 84

**Fig. 2** Common architecture of concept drift detectors consisting of two data windows, two independent data modeling steps, a dissimilarity measurement and finally a decision criterion (adapted from [23, 26])

4 Algorithms investigated

In this section common architectural choices of unsupervised concept drift detectors are described. Afterwards the 10 detectors that passed the inclusion criteria are introduced. In the following, the terms *algorithm* and *detector* are used interchangeably.

4.1 Architecture

Most concept drift detectors share a common architecture consisting of 3 to 4 components (cf. Fig. 2): Two data windows, one for the most recent data and one for historical reference data respectively, data modeling for each data window (optional), a dissimilarity measurement to measure the similarity or dissimilarity of the data windows and finally a drift criterion [23]. In some detectors the data modeling, dissimilarity measurement or drift criterion steps are combined in one step, e.g., when hypothesis tests are used directly on the data windows.

In the following, each component is explained briefly and common methods are given. A complete overview over the implemented detectors' approaches for these components is available in Table 4.

Table 4 Summary of the fully unsupervised concept drift detectors implemented in this study

Detector	Reference data window	Data modeling	Dissimilarity measurement	Criterion	Reset
BNDM [48]	Sliding	–	Pólya tree test	k -sample	✗
CSDDM [36]	Fixed	PCA	k -means clustering	Anderson–Darling test	✗
D3 [45]	Sliding	Artificial class labels	Classifier accuracy	Hyperparameter threshold	After drift
EDFS [49]	Sliding	Feature subspaces	Kolmogorov–Smirnov two-sample test	Empirical threshold	After drift and regularly
IBDD [46]	Fixed	–	Mean-squared deviation	Normal distribution ppf with empirical threshold	✗
NN-DVI [40]	Fixed	Neighborhood particles	Particle distance	Hyperparameter threshold	After drift
OCDD [37]	Fixed	–	Outlier detection rate	Chi-square quantiles	✗
SPLL [39]	Sliding	Gaussian mixture model	Log likelihood estimation and Mahalanobis distance	–	–
UCDD [38]	Sliding	–	k -means clustering	Beta distribution ppf	✗
UDetect [47]	Fixed	Standard deviation	Shewhart control parameters	–	✗

Full names of the detectors are provided in the text

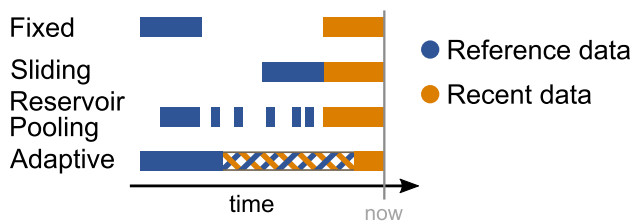


Fig. 3 Data windows are commonly maintained in one of three ways. The recent data window always contains the most recent data. A fixed reference window contains data gathered at the beginning of a concept, whereas a sliding reference window contains data that just left the recent window. Reservoir pooling represents a compromise by replacing fixed data with data from the recent window with a certain chance. Adaptive windowing incorporates all data until a concept drift occurs and dynamically determines the boundary between reference data and recent data

Data windows

All algorithms evaluated in this study deploy two data windows on the data stream. The reference window is meant to contain data from a stable concept, whereas a sliding window holding the most recent data is compared to it. If the concept changes, the recent data in the sliding window will change before the data in the reference window does, allowing concept drift detectors to discern the two samples and detect the concept drift. In general, larger sample sizes will make a detector more robust to noise but increase the time until detection, while observing smaller sample sizes has the opposite effect. Both goals are desired, as detectors should detect as soon as possible with as few false positives as possible, but they are directly opposed [29]. Therefore, choosing the sample size may require a compromise.

If the detector uses a reference window, there are two methods used by the implemented detectors and two more methods are mentioned in the literature (see Fig. 3):

1. A *fixed* reference window is filled with data at the beginning of the data stream and not adjusted afterwards [23].
2. A *sliding* reference window follows the recent data window [23].
3. *Reservoir pooling* [35] attempts to strike a balance between the previous two methods by filling a fixed window first. Then, old data is replaced with data leaving the recent data window with a certain chance.
4. Finally, *adaptive windowing* [11] is used as part of the ADWIN concept drift detector. ADWIN uses a statistical test to determine if data is stationary. The boundary between the reference data window and the recent data window is dynamically determined using the same statistical test. While there is no concept drift, the adaptive window grows. Once a concept drift is detected, only data from the recent window is kept.

Although these windows are meant to represent a stable concept, none of the publications reviewed for this study discussed how the window sizes must be chosen to address this intent. Instead, this hyperparameter must be provided by the user, so they can be drawn either from experience in the targeted application domain or are chosen arbitrarily. Additionally, no detector that passed the inclusion criteria is capable of dynamically adjusting window sizes like ADWIN [11] for example.

A few implemented detectors do not keep a window of reference data but gather sufficient data to fit models used to assess the recent data [36–38]. For all intents and purposes, these detectors can be considered detectors with fixed reference data.

Data modeling

Data modeling techniques include methods determining statistics of the data windows or estimations of the probability density functions, for example by determining the standard deviation or by creating Gaussian mixture models [39]. Some detectors model neighborhood relationships by clustering the data, e.g., by creating clusters with a k-means algorithm [38] or through a modified k-nearest-neighbors algorithm [40]. Another approach includes transforming the data with a principal component analysis to ensure the most significant features are represented [36]. Finally, although they did not satisfy the inclusion criteria for the benchmark, deep learning approaches based on (variational) autoencoders are discussed in a few publications [41–43].

Dissimilarity measurement

In order to detect a concept drift, the dissimilarity of the two data windows needs to be assessed. Some dissimilarity measurements can be applied directly on the reference data and recent data, whereas other methods assess the modeled data windows. Common methods include distance metrics such as the Mahalanobis distance, log likelihood estimation [39], density estimation [35], outlier detection rates [37] or divergence measures such as the Kullback–Leibler divergence [44]. Approaches based on deep learning often incorporate the loss [41].

Drift criterion

Finally, the detector decides if a concept drift is present in the data. Three options are available: Firstly, a concept drift can be signaled if the dissimilarity measurement exceeds a certain threshold [37, 45–47]. This threshold can be provided as a hyperparameter or it can be determined empirically on stationary data [46, 47]. Secondly, instead of deriving a threshold empirically, a probability density function can

be estimated on stationary data [38–40]. The likelihood of dissimilarity measurements can then be determined through the respective probability density function's percentage point function (ppf). If the dissimilarity measurement is in the probability density function's extremes, a concept drift is detected. Thirdly, hypothesis tests like the Kolmogorov-Smirnov two-sample test or the Page-Hinkley test can be used [36, 44, 48, 49].

Reset

Some concept drift detectors, especially those with empirically determined thresholds, must reset and adapt to the new concept after a drift. The simplest choice is to purge both data windows and perform required setup steps again by gathering sufficient data to train models or to determine thresholds. Few concept drift detectors with custom tailored reset mechanisms exist [37, 46].

4.2 Implemented algorithms

The following section conveys the architecture and core idea of each implemented detector as well as data requirements. Few authors discuss the strengths and weaknesses of their approaches. For detailed information, the reader is encouraged to consult the respective publication or the source code of this study's implementations. Table 4 gives an overview of each detector's respective choice concerning the previously mentioned components, data window management, data modeling, dissimilarity measurement, drift criterion and reset. The requirements and assumptions made by each detector are summarized in Table 5.

Although they satisfy the inclusion criteria, *EDFS*, *NN-DVI* and *UCDD* were not evaluated in the experiments. The reasons therefor are given alongside the respective detector's description below.

Bayesian nonparametric detection method (BNDM)

BNDM [48] uses a Pólya tree test to determine if the reference data and recent data originate from the same distribution. The two samples are partitioned recursively based on percentiles of a normal distribution. Then, the partitions of the samples are evaluated using the Beta function. Because the Pólya tree test is a univariate test, BNDM applies the test individually on each feature. A concept drift in a single feature is sufficient to signal a concept drift. According to the authors, the detector is best suited for sudden changes and struggles with subtle concept drifts.

The Pólya tree test assumes that the data are independent and identically distributed [50]. Furthermore, it can only operate on numerical data and demands that the data is normalized.

Clustered statistical test drift detection method (CSDDM)

CSDDM [36] combines feature reduction with clustering and a statistical test. Once a fixed reference window contains sufficient data, a setup is performed consisting of two major steps: Firstly, the detector performs a principal component analysis (PCA) to reduce the computational complexity on fixed reference data. Secondly, centroids are fitted on the transformed fixed data using a k -means algorithm.

After the setup is concluded, new incoming data are transformed according to the PCA and assigned to the centroids. Concept drifts are detected by comparing the recent and reference data in each cluster using the k -sample Anderson-Darling test [51]. Since the Anderson-Darling test is a univariate test, this test must be performed once per component, so $n \cdot m$ times given n centroids and m components.

CSDDM requires numerical data. The k -sample Anderson-Darling test typically assumes continuous independent data [51], but the implementation available in *scikit-learn* can operate on discrete data as well [52].

Discriminative drift detector (D3)

D3 [45] attempts to discern the reference data and the recent data using a linear classifier. The reference data and the recent data are assigned artificial class labels, so all reference data belongs to one class and all recent data belongs to the other. Then a classifier is trained on one part of the data with the other being used to test the predictive performance. Although a simple train/test split is mentioned in [45], the corresponding source code shared by the authors performs a k -fold cross validation. If the classifier's predictive performance exceeds a certain threshold, the two samples must be sufficiently different indicating the presence of a concept drift.

D3's requirements depend mostly on the classifier, so in most cases it is reasonable to assume that the detector would prefer to work on independent and identically distributed data. The linear classifiers proposed by the authors require numerical data, however other classifiers could enable the use on categorical data as well.

Ensemble drift detection with feature subspaces (EDFS)

EDFS [49] constructs an ensemble of ensembles of univariate Kolmogorov-Smirnov two-sample tests. Although it is meant to be used with labeled data to create meaningful feature subspaces, it can be used with random feature subspaces as well. Within each feature subspace, majority voting is used to detect concept drifts. On the ensemble level, a single subspace is sufficient to detect a concept drift.

In preliminary experiments the univariate tests rarely detected concept drift at the same time, but briefly after each other. Hence, the hypothesis tests' samples should only be

Table 5 Summary of requirements and assumptions made about data by each detector in this study; a ✓ indicates that the detector assumes independent or identically distributed data respectively

Detector	Data types	Independence	Identical distribution	Further assumptions
BNDM [48]	Numerical	✓	✓	Normalized data
CSDDM [36]	Numerical	✓		Data should not be normalized
D3 [45]	Depends on the classifier	✓	✓	
EDFS [49]	Numerical	✓	✓	Domain expert knowledge
IBDD [46]	Numerical	n/a	n/a	Normalization is mentioned but not required
NN-DVI [40]	Numerical	✓		
OCDD [37]	Depends on the outlier detector	(✓)	(✓)	
SPLL [39]	Numerical	✓	✓	
UCDD [38]	Numerical			
UDetect [47]	Numerical	✓	✓	

Owing to the inclusion criteria, all these methods operate on multivariate data, although BNDM, CSDDM and EDFs are ensembles of univariate tests. Full names are provided in the text

reset once the majority of tests in a subspace voted in favor of a concept drift. Else the detector will most likely fail to detect any drift. The authors highlight that the detector performs best for sudden concept drift, whereas longer or subtle concept drifts are harder to detect.

Since EDFs creates random feature subspaces in the absence of labels, the detector depends on chance to a large degree in this study's experiments. Preliminary tests showed a large variance in the evaluated metrics for many configurations depending mostly, if not entirely on the seed. Hence, EDFs is not included in the experiments and evaluation.

Because it is built on an ensemble of Kolmogorov-Smirnov tests, EDFs assumes independent and identically distributed numerical data [49]. Furthermore, it requires domain expert knowledge to construct the feature subspaces.

Image-based drift detector (IBDD)

IBDD³ [46] detects concept drifts based on the mean-squared deviation of the reference data and the recent data. The detection thresholds are derived empirically by calculating mean-squared deviations of permutations of the fixed reference data. If the mean-squared deviation of the recent data is not within two standard deviations of the deviations calculated on the permuted reference data, a concept drift is detected. The detector requires numerical data, which may be normalized.

³ Although its name may suggest otherwise, this detector may very well be used on non-image data.

Nearest neighbor-based density variation identification (NN-DVI)

NN-DVI [40] detects concept drifts by determining the dissimilarity of particles across two data samples. Particles are constructed by determining the nearest neighbors of each data point in the respective sample and constructing an adjacency matrix. Afterwards, the dissimilarity between the particles of the two samples is determined on the adjacent matrix using a custom distance measure. Finally, a custom statistical test is conducted to determine if a concept drift occurred. To this end the samples are permuted repeatedly to establish a baseline the current dissimilarity needs to exceed in order to detect a concept drift.

NN-DVI was not evaluated in the experiments because the straightforward implementation of the algorithm is too slow to allow thorough evaluation. The implementation relies on *scikit-learn* [52] to determine the nearest neighbors of each data point during the construction of the particles, which is not optimized for repeated calculations at each time step of a data stream.

The authors assume that data samples are independent. Furthermore, the statistical test requires independent and identically distributed data, which can be ensured by shuffling the samples, according to the authors. Lastly, the observed data must be numerical, as the proposed distance measure requires numerical data.

One-class drift detector (OCDD)

OCDD [37] assumes that an outlier detector can detect a concept drift based on the rate of outliers. The authors suggest using a one-class support vector machine [53]. This outlier detector is trained on a fixed window of reference data. Afterwards, the outlier detector predicts if incoming data are outliers or inliers. If the outlier detection rate exceeds a certain threshold, a concept drift is detected.

Requirements depend on the chosen outlier detector. The proposed one-class support vector machine assumes independent and identically distributed data [53], as do many other outlier detectors. As this study's implementation follows the suggestion of Gözüacık et al. [37], it can only operate on numerical data.

Semi-parametric log-likelihood (SPLL)

SPLL [39] is a concept drift detector based on likelihood estimation. In order to estimate log-likelihoods of the given data, a Gaussian mixture model is created on the reference data. Since the data may not support computing one covariance matrix per component of the Gaussian mixture model, the author proposes using a global covariance matrix instead. Afterwards, the log-likelihood of incoming data is estimated. Chi-squares quantiles are used to determine if a concept drift is present based on a predetermined threshold.

According to the author, the detector is best suited for detecting sudden concept drift. Furthermore, if the data cannot be clustered well, SPLL may fail to detect concept drift [39]. Since SPLL fits a Gaussian mixture model and computes a covariance matrix, the data needs to be numerical. The Chi-square test assumes independent and identically distributed data [54].

Unsupervised concept drift detector (UCDD)

UCDD [38] combines k -means clustering with the beta distribution to monitor change in a decision boundary. Two centroids are fitted on both the reference and the recent data to assign artificial class labels to the data. The resulting clusters must include data from either sample. The detector then determines the nearest data point of the other class for each data point in the recent window and assembles them in two sets. The number of items in each set is entered into the cumulative distribution function (cdf) of a beta distribution. If the resulting probability is lower than a certain threshold, a concept drift is detected.

It is crucial that the observed data can be clustered and even more importantly so that instances from both samples are present in each cluster. If one sample is not present in a cluster, the constructed set will be empty and the cdf of beta distribution will be called with one argument being 0,

which is not within the function's domain. The authors do not discuss this case, so it is unclear how to proceed. Furthermore, it is unclear how to ensure this situation does not occur or how to determine if it can occur with a given data stream.

As is the case with NN-DVI, the straightforward implementation of UCDD for this study relies on functionality from *scikit-learn* [52], in particular a k -means algorithm. During the experiments it became apparent that this study's implementation is too slow to allow thorough evaluation. The largest contributor to this issue may be repeated calculations of the clusters, which may be alleviated by using an incremental k -means algorithm.

Unsupervised change detection for activity recognition (UDetect)

UDetect [47] observes the standard deviation in the recent data on each feature. Hence, the data must be numerical. Afterwards, it determines the average standard deviation across all features and compares it to several limits. These limits are derived from fixed reference data using Shewhart control parameters. If the Shewhart limits are exceeded, a concept drift is detected. Shewhart control charts often assume independent and identically distributed data [55, 56], however this common requirement is not discussed in [47].

As the algorithm is intended for activity recognition, the authors propose using a dedicated detector for each activity with dedicated thresholds, respectively. Since no activity recognition is performed in this study, a single detector is used for global concept drift detection.

5 Experimental design and setup

In the following, the setup of the experiments is explained. Firstly, the real-world data streams and the metrics used for evaluation are introduced. Then, the workflow used to evaluate 7 of the above concept drift detectors on 11 real-world data streams is outlined. The evaluated concept drift detectors are *BNDM*, *CSDDM*, *D3*, *IBDD*, *OCDD*, *SPLL* and *UDetect*. EDFS is not evaluated because feature subspaces would be generated by chance, resulting in predictive performances that depend mostly on the seed. NN-DVI and UCDD are not evaluated, as a simple implementation using clustering algorithms from *scikit-learn* is too slow for online use.

5.1 Data streams used

Although various real-world data streams are commonly used to evaluate concept drift detectors, data with known concept drift ground truth information—which concept drifts are present, when do they begin and end—are difficult to obtain. In addition to offering an overview of existing data streams

without such ground truth information, Souza et al. propose new real-world data streams with known drift ground truth [32]. They constructed these data streams by measuring the wing beat frequency of insects inside an insect trap. Altering the temperature inside the trap causes changes in the frequency. The temperature inside the trap is a hidden variable representing the concept. By ordering the data by temperature in a specific way, abrupt, incremental and gradual concept drifts were induced. However, only the data streams with abrupt concept drift feature stationary periods as well. In the incremental and gradual data streams the temperature changes perpetually [32].

Of the 11 insects data streams Souza et al. [32] provide, only data streams with a balanced class distribution are used in this study because these streams are shorter than the imbalanced ones whilst still offering a reasonable number of samples. Instead, more experiments on other data streams and concept drift detector configurations can be conducted. In addition to the five chosen insects data streams, detectors are evaluated on six other real-world data streams without known concept drift ground truth. An overview of the real-world data streams used for evaluation is provided in Table 6.

5.2 Metrics

5.2.1 Mean time ratio

If the ground truth of concept drifts in a data stream is known, the detector's performance can be evaluated using these four metrics [29]:

1. *Missed Detection Rate (MDR)* considers a concept drift missed, if the detector does not detect a drift at any time after the drift and before the next one. The lower the *MDR* the better.
2. *Mean Time to Detection (MTD)* is used to evaluate the time it takes to detect the drift. Only valid detections are used to determine the *MTD*. A good detector will have a low *MTD*.
3. *Mean Time between False Alarms (MTFA)* evaluates the time between false detections, which occur when a concept drift is detected, but the previous concept drift was detected already. Note that the *MTFA* is undefined if less than 2 false alarms occurred. The higher the *MTFA* the better.
4. Finally, *Mean Time Ratio (MTR)* assembles the previous three metrics:

$$MTR := \frac{MTFA}{MTD} \cdot (1 - MDR). \quad (1)$$

The *MTR* respects the way the other metrics should be interpreted, as the fraction will be larger if the *MTFA* is large and the *MTD* is low. Moreover, a lower *MDR* causes a larger *MTR*. Hence, a high *MTR* is desirable and when comparing two detectors, the detector with the higher *MTR* performed better in general. Alas, it is undefined if less than 2 false alarms occurred.

Depending on the application one of the individual metrics may be more desirable than *MTR*. For example, in a data stream with very long stationary periods where concept drift occurs rarely, a high *MTFA* and therefore a low number of false alerts may be preferable over a low *MTD* accompanied by many false alerts. Furthermore, optimizing for *MTFA* likely leads to a worse *MTD* and *MDR*. In order to reduce the number of false alerts, a detector needs to become more robust to noise so larger changes are required to detect a concept drift. Naturally, this increases the time until a drift is detected and might result in a missed detection if the concept drift is subtle. Similarly, optimizing for *MTD* and *MDR* means making the detector more sensible to change, which could result in an increased number of false alerts.

These metrics can only evaluate sudden concept drift. If the evaluated concept drift is incremental or gradual and occurs over multiple time steps, multiple issues can arise; mostly because it is unclear how to match each detection with the concept drift. If the detection is delayed, should it be matched with the oldest moment of an incremental/gradual concept drift or the most recent one? Either choice may influence *MTD*, *MDR* and *MTFA*, requiring further research.

Among the data streams chosen for this study, only *INSECTS (abrupt bal.)* allows calculating *MTR* because most other data streams provide no ground truth information on concept drifts or contain no stationary periods, thus no false alerts can be determined.

5.2.2 Proxy metrics

If no ground truth information on concept drifts is available, concept drift detectors are commonly evaluated using a proxy: The accuracy or F_1 score of a classifier deployed with the detector [29]. Often, the classifier is evaluated using prequential testing or interleaved test-then-train, which means that each instance is first evaluated by the classifier and then used for further training [8]. This evaluation scheme is used in this study, too. A common assumption is that a better concept drift detection will result in a higher classifier performance. However, Bifet [29] demonstrated that naively adapting the classifier every n samples will outperform sophisticated detection methods and argued that merely comparing the classifier predictive performance is insufficient to compare concept drift detectors. In the experiments, it can be expected that detectors with smaller window sizes

Table 6 Data streams used in the experiments

Data stream	Samples	Features	Known drifts
INSECTS (abrupt bal.)	52,848	33	✓
INSECTS (gradual bal.)	24,150	33	~
INSECTS (incremental bal.)	57,018	33	~
INSECTS (incremental-abrupt bal.)	79,986	33	~
INSECTS (incremental-reoccurring bal.)	79,986	33	~
Electricity2	45,312	8	✗
NOAA Weather	18,159	8	✗
Outdoor Objects	4000	21	✗
Poker Hands	829,201	10	✗
Powersupply	29,928	2	✗
Rialto Bridge Timelapse	82,250	27	✗

All data streams are provided by [32], except for *Electricity2*, which is included in *River* [57], an online machine learning framework. In the column *known drifts*, a ✓ indicates that the exact position of each drift is known such that the metrics outlined in Sect. 5.2 can be applied. In contrast, ~ denotes that the kind of drift is known, but the data stream contains no stationary periods featuring constant concept drift. A ✗ indicates that some analysis may be available, but the exact nature of the concept drifts present is not known

and lower thresholds should achieve a higher classifier accuracy.

To account for this issue, Anderson et al. proposed *lift-per-drift* (*lpd*) [30]. *lpd* takes into account the number of times a classifier was adapted and the performance improvement over a classifier that was not adapted at all. In the default case, *lpd* is defined as

$$lpd := \begin{cases} \frac{acc_d - acc_0}{\# \text{ drifts}}, & \text{if } \# \text{ drifts} \geq 1, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where acc_d is the accuracy of a classifier enhanced by a concept drift detector, acc_0 is the accuracy of a classifier without concept drift detection and $\# \text{ drifts}$ is the number of drifts detected by the drift detector.

The larger *lpd*, the more impactful the detected concept drifts and subsequent adaptations of the classifier with respect to the number of detected drifts. Hence, the concept drift detector with a larger *lpd* can be assumed to perform better.

Anderson et al. also propose a version controlled by a hyperparameter r to adjust the required improvement each detected drift “must provide over the last (following a geometric progression)” [30]:

$$lpd := \begin{cases} \frac{(acc_d - acc_0)(1-r)}{(1-r^{\# \text{ drifts}})}, & \text{if } \# \text{ drifts} \geq 1, \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

with $0 < r < 1$, which can be simplified to the definition given in Eq. 2 for $r \rightarrow 1$. Since a hyperparameter in a metric makes the evaluation in this study significantly more complex, the default version from Eq. 2 is used.

5.3 Setup

All algorithms are implemented in Python 3.8, to allow using common data science and machine learning packages such as *numpy*, *pandas*, and *scikit-learn* [52]. The online machine learning framework *River* [57] provides classifiers used during evaluation as well as classes to handle the data streams. The full source code is available under BSD 3-clause license on GitHub,⁴ to provide public access to the implemented detectors and more importantly to support full reproducibility of the experiments and results.

Occasionally, algorithm descriptions left room for interpretation. These instances and the respective implementation choices are documented in Appendix A. For example, several publications do not mention any reset mechanism to adapt to the new concept after the detection of a concept drift. As a consequence, these detectors may signal concept drifts repeatedly until the drift cannot be detected in the data windows anymore. Since two of the metrics used to evaluate the detectors—*MTR* and *lpd*—will be adversely impacted by this behavior, adaptation is introduced if necessary by flushing the data windows and resetting all dependent information, e.g., empirically derived thresholds that determine the presence of a concept drift. The impact of this choice on the classifier’s accuracy is discussed in Sect. 6.3.

The workflow used to conduct the experiments is outlined in Algorithm 1. Every detector is deployed on every data stream with various configurations (see lines 1 to 3). A simple grid search is performed to test all permutations of the configuration parameters detailed in Appendix B. The chosen configuration parameter ranges are not meant to find the

⁴ <https://github.com/DFKI-NI/unsupervised-concept-drift-detection>.

Algorithm 1 Experiment workflow

Require: data streams, detectors, configurations

```

1: for data stream in data streams do
2:   for detector in detectors do
3:     for config in configurations[detector] do
4:       model  $\leftarrow$  detector(config)
5:       classifiers  $\leftarrow$  init_classifiers()
6:       drifts  $\leftarrow$   $\emptyset$ 
7:       predictions  $\leftarrow$   $\emptyset$ 
8:        $i \leftarrow 0$ 
9:       for  $x, y$  in data stream do
10:        predictions.add(classifiers( $x$ ))
11:        drift  $\leftarrow$  model.detect_drift( $x$ )
12:        if drift then
13:          drifts.add( $i$ )
14:          model.reset()
15:          classifiers.reset()
16:        end if
17:        classifiers.fit( $x, y$ )
18:         $i \leftarrow i + 1$ 
19:      end for
20:      metrics  $\leftarrow$  evaluate(drifts, predictions)
21:    end for
22:  end for
23: end for

```

definitive best-performing configuration, but to test different configurations to identify general trends. They are based on the default configurations proposed by the authors of the respective algorithms if a recommendation was given.

Alongside each detector four online classifiers are deployed, two Hoeffding trees [58] and two incremental naive Bayes classifiers [57], to allow proxy evaluation in the absence of concept drift ground truth information. They are initialized alongside a model of the detector parameterized with the current configuration and data structures for evaluation (see lines 4 to 7). Hoeffding trees and naive Bayes classifiers are the most common choices in the concept drift detection literature. Since these classifiers are used as means of evaluation and not evaluated themselves, no further classifiers are deployed.

On each data stream, the detected drifts and the classifiers' predictions are recorded (see lines 10 to 13). The classifiers are evaluated and trained using prequential testing (see lines 10 and 17), where each individual instance is used for evaluation before being used in training [8]. If a concept drift is detected, the detector and the classifiers reset and adapt to the new concept (lines 14 and 15). Finally, the metrics are evaluated and logged in line 20. For each classifier, the F_1 -score and accuracy are recorded. One of each is deployed independently of the detector and trained on the first 1000 samples only, as they merely provide a baseline for the calculation of lpd . The remaining Hoeffding tree and naive Bayes classifier are trained continuously and reset when the concept drift detector detects a drift by initializing a new model.

Furthermore, MDR , MTD , $MTFA$ and MTR were determined on the INSECTS (abrupt balanced) data stream.

Four evaluated detectors— $CSDDM$, $D3$, $IBDD$, $SPLL$ —are non-deterministic, as they depend on pseudo-randomness for example to fit clusters or Gaussian mixture model components. These detectors are deployed five times with different seeds to gather more informative results. The UNIX timestamp at the time of the detector's initialization is used as the seed in all experiments.

All components of this study are tested with unit tests; their interaction is further tested with integration tests. Simple function tests assure that the concept drift detectors detect artificial drifts on a synthetic data stream consisting of one feature with a single concept drift. The entire experiment data including all tested configurations, the respective seeds and recorded metrics are available alongside the source code.

6 Results and discussion

Due to the large volume of information gathered in the experiments, the results are discussed as they are presented in the following subsections. First, results are filtered to remove flawed configurations. Then, MTR results are presented and discussed, followed by the same for classifier performance. Finally, lpd results are shown and concluded by a brief explanation of the difficulties encountered during analysis of the results. Classifier accuracy and lpd are given for Hoeffding trees; results of non-deterministic detectors are averaged.

6.1 Filtering of failed configurations

In a first step, the experimental results were filtered to remove those results which featured no detection or periodic concept drift detection, i.e., detection every n time steps. Neither behavior requires intelligent detection capabilities and certainly no concept drift detector, making these results unsuitable for further analysis of the respective detector. The number of configurations that exhibited either behavior is given in Table 7 alongside the number of configurations kept for further analysis.

The most common reasons a detector did not detect any drift or triggered periodically are related to the observed sample size. In the conducted experiments, filtered detectors always had sample sizes on the low end or high end of the parameter sweep. When the sample sizes are too small, detectors are sensitive to noise. Larger sample sizes on the other hand cause periodic detection when concept drift occurs again before the detector could finish its reset by filling its data windows. Once the data windows are full, the concept drift is already perceptible by the detector. Often, there is an interaction of other parameters with the sample size in periodic detection: As the sample size decreases, a detector may

Table 7 The experimental results were filtered, as no detection and periodic detection do not require a concept drift detector; the number of filtered results is given in the respective columns

Detector	No detection	(%)	Periodic detection	(%)	Rest	(%)
BNDM	0	(0%)	150	(28%)	378	(72%)
CSDDM	0	(0%)	276	(16%)	1484	(84%)
D3	117	(6%)	21	(1%)	1841	(93%)
IBDD	0	(0%)	0	(0%)	3960	(100%)
OCDD	1	(0%)	106	(38%)	169	(62%)
SPLL	55	(7%)	10	(1%)	735	(92%)
UDetect	126	(45%)	18	(6%)	135	(49%)

The column *Rest* states the number of results kept for further analysis

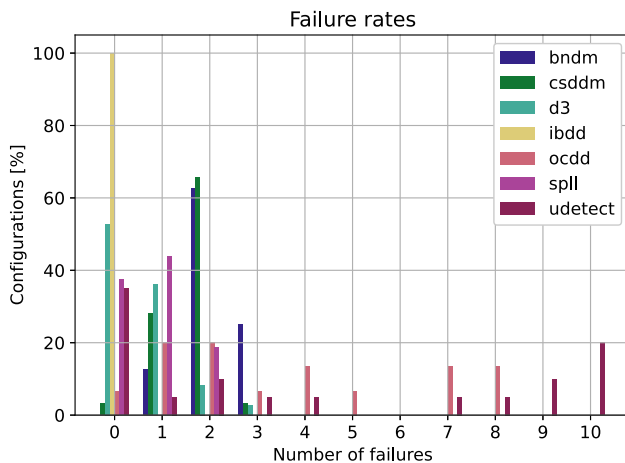


Fig. 4 The percentage of configurations that failed on $n, n \in \{0, 1, \dots, 10\}$, data streams for all detectors. A configuration is considered failed if it returned periodic drift detection or none at all for all seeds

no longer detect periodically unless a threshold or similar parameter controlling the sensitivity decreases as well.

Furthermore, some detectors failed more frequently when specific circumstances are met:

- *BNDM* does not perform well with few samples, showing periodic detection behavior more regularly.
- *CSDDM* features a parameter controlling the number of principal components of the samples used for concept drift detection. Periodic detection behavior arose more often when more than one principal component was used.
- Filtered *OCDD* configurations often featured low thresholds of 0.2 and 0.3 despite being recommended by the authors [37].
- *SPLL* failed to detect concept drifts almost exclusively on *INSECTS (incremental balanced)* when observing large samples.
- All 126 instances of *UDetect* failing to detect any drift occurred when thresholds were derived from disjoint reference windows.

Finally, the percentage of configurations that did not yield any results on at least n data streams is given in Fig. 4. Figure 4 and Table 7 show that *IBDD* is the only detector to feature no filtered configurations on any data stream. It is followed by *D3* and *SPLL*, whereas *OCDD* and *UDetect* are shown to be less reliable. The main architectural choice setting *IBDD* apart from the other detectors is the regular adaptation of its decision criterion.

As these results include repeat runs of non-deterministic detectors, the mean and standard deviation of all metrics are determined for each configuration. Although the proxy metrics accuracy and *lpd* were recorded for both a Hoeffding tree and a Naive Bayes classifier supported by the drift detector, the following evaluation considers the metrics recorded for the Hoeffding tree only for two reasons. Firstly, the correlation of the metrics recorded for the Hoeffding tree and the metrics recorded for the Naive Bayes classifier is strong on all data streams. The R^2 of the respective *lpd* values exceeds 0.95 on all data streams and 0.99 on 9 out of 11 data streams. Even though the R^2 of the accuracy values is not as high, exceeding 0.89 on 8 out of 11 data streams only, there is a positive correlation on all streams. On *NOAAWeather*, *Outdoor Objects* and *Powersupply* the R^2 is approximately 0.38, 0.64 and 0.77, respectively. Secondly, the Hoeffding tree achieves far greater accuracy than the Naive Bayes classifier on all evaluated data streams.

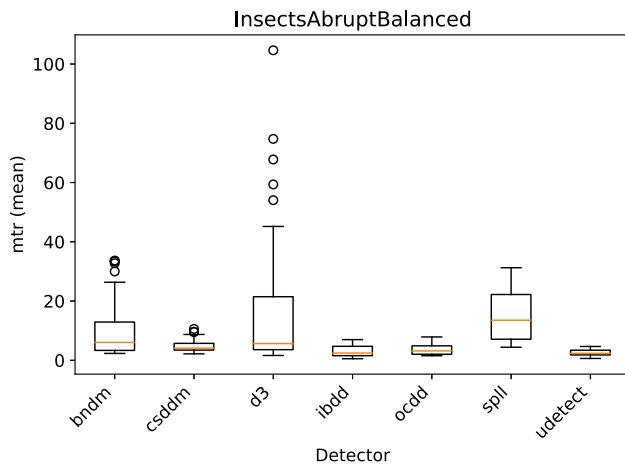
6.2 Mean time ratio

MTR is the only metric evaluating the predictive performance of concept drift detectors directly. *INSECTS (abrupt balanced)* is the only data stream in this study which can be used to determine *MTR*. Although some other *INSECTS* data streams contain specific information about abrupt concept drifts as well, they all feature incremental or gradual drift without any stationary periods. If concept drift happens constantly, any detection must be considered correct and on time, rendering recording detection times and false alerts a futile effort and therefore making it impossible to determine *MTR* on these data streams.

Table 8 Best mean *MTR* performances and the associated *MTFA*, *MTD* and *MDR* per detector on *INSECTS* (*abrupt balanced*), rounded to one decimal place

Detector	<i>MTR</i>	σ	<i>MTFA</i>	σ	<i>MTD</i>	σ	<i>MDR</i>	σ
BNDM	33.7		4927.0		146.4		0.0	
CSDDM	10.6	± 1.2	2327.7	± 298.2	219.6	± 9.2	0.0	± 0
D3	104.6	± 0.0	20664.0	± 0.0	91.6	± 28.2	0.6	± 0
IBDD	7.0	± 0.0	2450.6	± 0.0	210.7	± 0.0	0.4	± 0
OCDD	7.9		3348.3		340.0		0.2	
SPLL	31.2	± 0.0	5185.0	± 0.0	132.8	± 0.0	0.2	± 0
UDetect	4.7		1445.7		247.8		0.2	

The absolute best performance is highlighted in bold face

**Fig. 5** Although peak performances by *BNDM* and *D3* may be outliers, many configurations significantly outperform the other detectors except *SPLL*

An overview of the achieved mean *MTR* and the associated mean *MTFA*, *MTD* and *MDR* is given in Table 8. For the latter three metrics, better performing detectors are available individually. However, as *MTR* aggregates all three metrics, performing poorly on one or two metrics has a detrimental effect on *MTR*. This is further emphasized as *MTFA* encourages different behavior compared with *MTD* and *MDR*, as explained in Sect. 5.2.

In these experiments, *D3* outperforms other detectors by a large margin: Seven different configurations achieve a higher *MTR* than the second-best detector. Apart from one configuration, the *MTR* of these *D3* configurations feature standard deviations below 15. Most notably, *IBDD* is among the worst performers, despite being the least filtered detector and having the most tested configurations (cf. Appendix B.3). Figure 5 shows each detector and its respective *MTR* distribution on *INSECTS* (*abrupt balanced*) as a box plot. Although the peak performances achieved by *BNDM* and *D3* may be outliers, many other configurations significantly outperform the other detectors as well. The only exception is *SPLL*, which performs on a similar level as *BNDM*.

Table 9 Correlation of *MTR* and both accuracy and *lpd* on *INSECTS* (*abrupt balanced*) for all detectors

Detector	R^2_{acc}	R^2_{lpd}
BNDM	0.32	0.47
CSDDM	0.47	0.46
D3	−0.33	0.43
IBDD	0.04	0.18
OCDD	0.42	0.89
SPLL	−0.34	0.32
UDetect	−0.47	−0.07

Rounded to two decimal places

Because *MTR* cannot be determined on any of the other data streams, classifier accuracy and lift-per-drift are evaluated as proxy metrics. The correlation of each of these proxy metrics with *MTR* as captured by the R^2 coefficient is inconclusive (see Table 9): With only a single exception, no detector shows a strong correlation. Furthermore, for some detectors, a correlation may be positive on one metric and negative on the other. In the future, experiments on complex synthetic data streams and real-world data streams with ground truth information could shed more light on the relation of the proxy metrics with ground truth metrics.

Since no other metrics are available for further evaluation on the remaining 10 data streams, classifier accuracy and *lpd* will be used for further evaluation as well. The former is a common, albeit flawed metric used for evaluation, whereas the latter was designed with the former's shortcomings in mind.

6.3 Accuracy

Before discussing the classifier's accuracy, the impact of adding a reset mechanism to 5 detectors must be discussed. It can be argued that the impact is minor for four reasons, although quantifying the impact is difficult from the results in this study:

1. Evaluating *MTR* and *lpd* would not be possible without the modifications.

2. Detecting more regularly only improves accuracy to a certain extent on most data streams, after which the classifier's predictive performance often decreases again, likely due to underfitting. This can be seen in Fig. 6 and the Figures in Appendix C.
3. Despite not being limited by any downtime after a detection, *IBDD* only shows much better accuracy on two data streams (see Figs. 15 and 16). According to [32], the *Poker Hand* data stream is mostly ordered. A brief test with the *no-change baseline* motivated by Zliobaite et al. [59] reveals that the same holds true for the *Outdoor Objects* data stream. Predicting that the current instance will be the same class as the previous one yields an accuracy of $\sim 90\%$ despite the data stream having 40 different classes, which explains *IBDD*'s advantage: By signaling drift regularly, the classifiers are overfitted to the most recent class, which most of the time is the class of the next data point as well.
4. Finally, as argued by both Bifet and Zliobaite et al. [29, 59], accuracy is a flawed proxy for evaluation of the detected drifts.

Based on [29], it is expected that detectors achieve a higher classifier accuracy as the number of detections increases. Figure 6 exhibits a typical interaction of the number of detected drifts and the Hoeffding tree classifier's accuracy on the *Insects (abrupt balanced)* data stream, which exhibits 5 concept drifts. Similar distributions can be observed on the other data streams (see Figs. 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33—Tables 17, 18, 19, 20, 21, 22, 23 in Appendix C), too. Generally, better accuracy results are achieved by detectors that detect more frequently until a certain point, after which the classifier's predictive accuracy drops again. One likely explanation for this behavior could be that the classifiers become underfitted as they are trained on increasingly fewer data. The exception to this is the interaction of accuracy and number of detected drifts on the *OutdoorObjects* and *PokerHand* data streams (shown in Figs. 15 and 16), which are ordered. The best results for accuracy are given in Table 10. The corresponding configuration for each detector-data stream pairing are given in Appendix B.3.

With this information *IBDD* achieving the best accuracy on 5 data streams can easily be explained: It is the only detector that adapts to a new concept in a way that allows it to detect concept drifts again immediately. Hence, it can detect more frequently than any other detector. On the *INSECTS* data streams and *NOAA Weather*, most peak performances are within one percentage point of the best detector. The other data streams feature more diverse results, but often a few other detectors come close to the best detector's performance. All detectors except for OCDD achieved peak performance on at least two data streams. It can be assumed

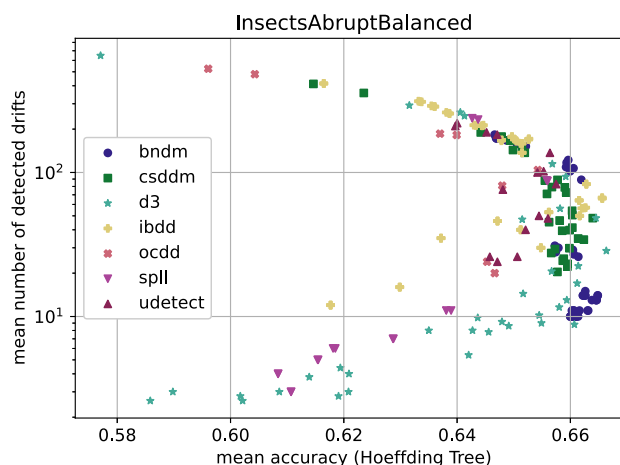


Fig. 6 Accuracy of Hoeffding tree classifiers depending on the number of concept drifts detected by the detector they are deployed with. As detectors detect more regularly, the accuracy increases until the classifier becomes underfitted. See Fig. 9 for larger resolution

that the problems presented by the different data streams require different approaches to concept drift detection. Since there is no obvious best algorithm, operators will need to compare multiple options. Overall, results could be close because classifier accuracy is the most commonly used metric during the development of new concept drift detectors and because the Hoeffding tree classifier performs well regardless of the detected concept drifts.

6.4 Lift-per-drift

As in Sect. 6.3, Fig. 7 shows the relation of the number of detected drifts and *lpd* on the *Insects (abrupt balanced)* data stream, which features 5 concept drifts. The other data streams show similar curves (see Figs. 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29 in Appendix C). While classifier accuracy favored a higher number of detected drifts and subsequent adaptations, *lpd* appears to show a bias towards fewer detections. The emphasis on different drift detection behavior is further shown in Table 11, which shows the R^2 correlation of the two proxy metrics across all data streams used in this study. On seven data streams, classifier accuracy and *lpd* are negatively correlated, 4 of those being strong correlations with the respective R^2 being less than -0.7 . The two data streams showing no correlation, *INSECTS incremental balanced* and *Rialto Bridge Timelapse*, likely possess incremental concept drift. Finally, two data streams show a weak positive correlation of classifier accuracy and *lpd*.

Anderson et al. [30] designed *lpd* so it would not be inflated by regular detection. Figures 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29 in Appendix C show that they achieved this goal, however $lpd_{r=1}$ seems biased the other way, as *lpd* is higher the lower the number of detected drifts is.

Table 10 Best accuracy achieved by a Hoeffding tree supported by the detector

Data stream	BNDM	CSDDM	D3	IBDD	OCDD	SPLL	UDetect
Elec2	–	0.842	0.853	0.845	0.820	0.847	0.837
INSECTS (abrupt bal.)	0.665	0.664	0.656	0.666	0.654	0.656	0.657
INSECTS (gradual bal.)	0.697	0.700	0.691	0.694	0.693	0.691	0.690
INSECTS (inc.-abrupt bal.)	0.681	0.682	0.679	0.683	0.670	0.673	0.682
INSECTS (incremental bal.)	0.572	0.568	0.561	0.566	0.563	0.561	0.566
INSECTS (inc.-reocc. bal.)	0.698	0.698	0.696	0.698	0.695	0.679	0.698
NOAA Weather	0.709	0.716	0.718	0.717	–	0.720	0.720
Outdoor Objects	0.616	0.758	0.812	0.784	0.799	0.767	0.787
Poker Hands	0.695	0.706	0.724	0.732	0.712	0.696	0.713
Powersupply	0.166	0.160	0.159	0.179	–	0.162	0.159
Rialto Bridge Timelapse	0.576	0.577	0.569	0.536	0.584	0.585	0.565
# best performances	2	2	2	5	0	2	2

Bold face indicates absolute best performance per data stream. Standard deviations are omitted because all standard deviations are less than 0.007

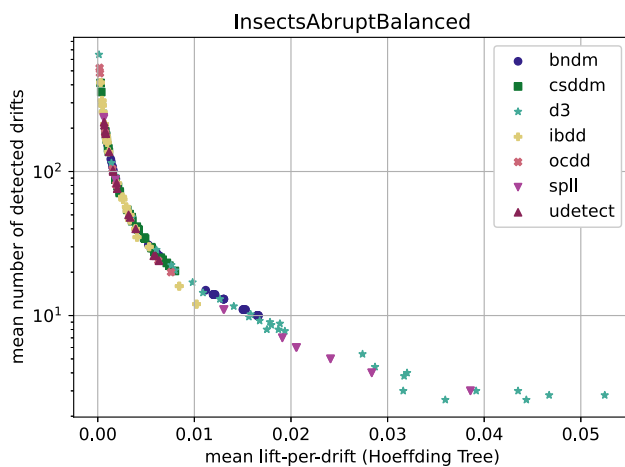


Fig. 7 Lift-per-drift of Hoeffding tree classifiers, depending on the number of concept drifts detected by the detector they are deployed with. The lower the number of detected drifts, the higher the *lift-per-drift*. *INSECTS (abrupt balanced)* exhibits 5 concept drifts. See Fig. 20 for larger resolution

The correlations in Table 11 show that the relation is not entirely direct, especially given the weak positive correlation on *NOAA Weather* and *Powersupply*.

The best results in terms of *lpd* are given in Table 12. Here, *D3* and *SPLL* are the best performing concept drift detectors, followed by *OCDD* and *UDetect*.

In depth comparison of the detectors seems difficult, given the bias towards fewer detections and adaptations, unless the number of detected drifts is very close. This rarely happened for the peak performers, but occasionally happened for other configurations. Examination of Figs. 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29 provided in Appendix C reveals that many configurations of *CSDDM*, *D3* and *SPLL* provide better concept drift detection than others on some of the data streams, as they achieve a higher *lpd* at a similar

Table 11 Correlation of accuracy and *lpd* as captured by the R^2 coefficient for all data streams

Data stream	R^2
Elec2	−0.74
INSECTS (abrupt bal.)	−0.42
INSECTS (gradual bal.)	−0.65
INSECTS (incremental-abrupt bal.)	−0.79
INSECTS (incremental bal.)	0.1
INSECTS (incremental-reoccurring bal.)	−0.68
NOAA Weather	0.43
OutdoorObjects	−0.86
Poker Hands	−0.78
Powersupply	0.31
Rialto Bridge Timelapse	−0.01

Rounded to two decimal places

number of detected drifts or a similar *lpd* at a higher number of detected drifts.⁵ These findings are supported by 6 data streams, as *INSECTS (incremental-abrupt bal.)*, *INSECTS (incremental-reocc. bal.)*, *NOAA Weather* and *Poker Hands* have no or very few close results in this regard. Furthermore, *D3* performs worse on *Powersupply*.

Finally, it is unclear what type of concept drift is present and where concept drift is located in many of the real-world data streams. This issue is not restricted to *lpd* and applies to classifier accuracy as well, although considering the number of detected drifts is a rare occurrence in the literature. Again, obtaining more real-world data streams with detailed knowledge about the position and nature of their concept drifts is a

⁵ Simply put: Because *lpd* determines the accuracy improvement per detected drift, configurations with a similar number of detected drifts or a similar *lpd* can be compared. The configuration further to the right or further up performed better, respectively.

Table 12 Best performance in terms of mean *lpd*

Data stream	BNDM	CSDDM	D3	IBDD	OCDD	SPLL	UDetect
Elec2	–	0.0037	0.0073	0.0037	0.0028	0.0033	0.0060
INSECTS (abrupt bal.)	0.0166	0.0080	0.0361	0.0223	0.0090	0.0386	0.0063
INSECTS (gradual bal.)	0.0660	0.0473	0.3091	0.1540	0.3195	0.1564	0.0289
INSECTS (incremental-abrupt bal.)	0.0152	0.0090	0.1651	0.0034	0.0154	0.0784	0.0797
INSECTS (incremental bal.)	0.0760	0.0307	0.3628	0.3565	0.0275	0.1768	0.0145
INSECTS (incremental-reocc. bal.)	0.0184	0.0109	0.3685	0.0057	0.0322	0.1004	0.0147
NOAA Weather	0.0045	0.0028	0.0142	0.0011	–	0.0549	0.0549
Outdoor Objects	0.1050	0.1143	0.1972	0.0382	0.2051	0.1073	0.0540
Poker Hands	0.00009	0.00018	0.00010	0.00003	0.00016	0.00026	0.00064
Powersupply	0.0019	0.0008	0.0024	0.0002	–	0.0217	0.0007
Rialto Bridge Timelapse	0.0025	0.0017	0.0022	0.0007	0.0026	0.0028	0.0025
# best performances	0	0	4	0	2	4	2

Bold face indicates absolute best performance per data stream. Standard deviations are omitted for better readability, but can be accessed in the study's repository

Table 13 Concepts of the SineClusters synthetic data stream

Concept	Features			
1	sin	cos	sin	cos
2	-sin	cos	sin	cos
3	-cos	-cos	sin	cos

necessity for meaningful analysis of (unsupervised) concept drift detectors, as the current modes of observation are biased or appear cumbersome and prone to error.

6.5 Verification on synthetic data

Few synthetic data streams as provided by River [57] and MOA [60] are suitable for the evaluation of unsupervised concept drift detectors. Often they introduce concept drift exclusively by changing the function generating the data stream's class label without adjusting the feature distribution. Since unsupervised concept drift detectors operate solely on the feature space, they cannot detect these changes. Others exhibit perpetual concept drift without ever offering a stationary concept. As a consequence, two new synthetic data streams are used to verify the issues highlighted before.

The first data stream is a modification of the Waveform data stream provided by River [57]. This data stream, denoted *WaveformDrift2*, generates 21 features driven by waveform functions. In addition to the original waveform function tuple h , three modified concepts $h+6$, $6-h$ and $-h$ are used. Concept drift is introduced by changing the waveform functions, which causes a concept drift in $P(X, y)$, affecting both the feature space X and the label space. The other data stream, *SineClusters* consists of four features generated by sine or cosine functions with a period of 500; the third and fourth feature are noise. There are three different classes generated by

two centroids each, which are randomly scattered in the data space. Features are assigned a class label by determining the closest centroid's class by Euclidean distance. This stream also features Gaussian noise with $\mu = 0$ and $\sigma = 0.25$. Concept drift is introduced by changing the first two functions and generating new centroids (see Table 13).

Both data streams featured abrupt concept drift every 5000 time steps to ensure that every detector had a chance to properly reset and become ready to detect again. The streams were executed with a run time of 154,987 samples, including a total of 30 concept drifts. No incremental or gradual concept drift was included, as no evaluation approach exists to leverage the availability of ground truth concept drift information in these cases.

Whereas *WaveformDrift2* has uniformly distributed features and changes each parameter's value range, *SineClusters* poses a different challenge as it changes merely the distribution of the features without adjusting their ranges. Furthermore, *SineClusters* is seasonal owing to the use of the sine and cosine functions.

Figures 30, 32 and 33 show that the same biases for accuracy and *lpd* are present in synthetic data. Merely Fig. 31 does not exhibit the same bias, but the evaluation with a Hoeffding tree classifier's accuracy remains flawed and difficult. Despite detecting more often than 30 times, many detectors achieve an accuracy similar to that of detectors which more reliably predict the true concept drifts. On *SineClusters*, configurations which detected periodically or yielded very low MTR often had sample sizes of 100 and 250, which is lower than the period of the sine functions used in the data stream.

The comparison of accuracy and *lpd* with MTR is difficult, because many detectors raised no false alarms. Consequently, the MTR remains undefined (cf. Table 14). Arguably, this shows the need for more reliable evaluation approaches.

Table 14 Best mean *MTR* performances and the associated *MTFA*, *MTD* and *MDR* per detector on the synthetic data streams *SineClusters* and *WaveformDrift2*, rounded to zero decimal places

Detector	SineClusters		MTD	MDR	WaveformDrift2		MTD	MDR
	MTR	MTFA			MTR	MTFA		
BNDM	–	–	94	0	–	–	35	0
CSDDM	84	3642	54	0	50	2102	40	0.07
D3	47	8123	171	0	–	–	29	0
IBDD	2	3438	880	0.26	58	629	11	0
OCDD	10	1871	187	0	7	1237	189	0
SPLL	–	–	251	0	–	–	6	0
UDetect	–	–	5	0.28	–	–	2	0

The absolute best performance on each data stream is highlighted in bold face

7 Open research challenges

This study reveals several open research challenges, which should be addressed in future work. Firstly, there is no simple evaluation approach available to assess the quality of concept drift detection when ground truth information is available, as MTR and MTFA are undefined when less than two false alarms are raised. Furthermore, no approach exists to evaluate data streams with incremental and gradual concept drift. Therefore, evaluation is restricted to proxy metrics even on synthetic data streams, which provide ground truth information. The MTR metrics are difficult to use to evaluate non-abrupt concept drift because it is unclear how concept drift and prediction should be matched if the concept drift lasts longer than one time step.

Secondly, most synthetic data streams provided in frameworks such as River [57] or MOA [60] are not suitable for the evaluation of (fully) unsupervised concept drift detectors, since they feature concept drift only in $P(y|X)$. This is commonly done by changing a function generating the class label without adjusting the feature generation. Unsupervised concept drift detectors cannot detect these changes, as they operate solely on the feature space X . Hence, more complex synthetic data streams are required to reliably test unsupervised concept drift detectors.

Thirdly, non-abrupt concept drift detection may require more thorough research. Many detectors evaluated in this study purge their data windows after detecting a concept drift. However, it is unclear if this behavior is suitable for incremental and gradual concept drift. If concept drift continues after a detector's reset, adaptation of other models will begin too early and may result in diminished predictive capabilities.

Fourthly, if above problem is verified, the question arises on how to combine abrupt, incremental and gradual concept drift detection in a single system. This might also include being able to discern the different types of drift.

Fifthly, analysis of experiment results showed that the size of the observed samples is one of the most impactful hyperparameters; as does a glance at the best performing

configurations given in Appendix B.3. Since most concept drift detectors construct observe samples in a similar way that is agnostic to the detector's specific ideas, methods to automatically determine sample sizes as ADWIN [11] or a modification of SPLL [61] could be worthwhile. Further research into ensuring concepts are properly represented in the reference data window may reveal new approaches to selecting window sizes. Likewise, methods to automatically derive decision thresholds from observed samples are available, but not leveraged by most concept drift detectors analyzed in this study.

Finally, this study used a grid search to evaluate different configurations. Consequently, many unsuitable configurations were tested and used up resources during evaluation. In the future, testing driven by AutoML approaches could yield better results whilst being more efficient [62]. Multi-objective optimization [63] could assist in balancing multiple opposed objectives such as fast concept drift detection, robustness to noise and good computational performance.

8 Conclusion

This study examined 61 publications related to unsupervised concept drift detection. It focused on detectors that can operate entirely without labeled data to detect substantial changes in data streams that could render predictive models unreliable or inaccurate if unaddressed. 10 publications provided a fully unsupervised algorithm and fulfilled further inclusion criteria ensuring that a reliable and faithful implementation was possible, e.g., because no major architectural choices needed to be assumed and the algorithm's description was complete. It was confirmed that these detectors adhere to a common concept drift detector architecture established in prior literature reviews [23, 26]. Furthermore, common choices for the architectural components as well as explanations of each detector's design were given.

7 out of 10 concept drift detectors were evaluated on 11 established real-world data streams [32] retrieved from the literature; the remaining 3 could not be evaluated for

Table 15 Strengths and weaknesses of the different detectors with respect to MTR, accuracy and $lpd_{r=1}$

Detector	MTR	acc	lpd	Other qualities
BNDM	++	+	o	Cannot detect changes in covariance
CSDDM	+	+	+	
D3	+	+	++	Sensitive to noise
IBDD	–	++	–	No downtime due to resets
OCDD	–	o	–	Sensitive to noise
SPLL	++	+	++	
UDetect	+	+	–	

In comparison with other the other concept drift detectors, a + indicates better performance, a – indicates worse performance and o denotes neutral performance

Table 16 Key findings concerning the metrics MTR, accuracy and $lpd_{r=1}$

	MTR	Accuracy	$lpd_{r=1}$
Proxy metric	✗	✓	✓
Requirement	Ground truth	Class labels	Class labels
Data streams	1	11	11
Bias	–	More detections	Fewer detections
Range	$[0, \infty]$	$[0, 1]$	$[-1, 1]$
Comparable on	Same data stream	Similar #detections	Similar #detections
Detectors	BNDM, D3, SPLL	IBDD, others	CSDDM, D3, SPLL

practical reasons such as a slow computational performance of this study's implementation or an excessive dependence on chance. In these experiments, the proxy metrics classifier accuracy and *lift-per-drift* (lpd) [30] were recorded to evaluate the concept drift detectors. **The key findings are summarized in Tables 15 and 16. On one data stream, Mean Time Ratio (MTR)** [29] was available to directly assess the detected concept drifts. The results reveal that *Image-Based Drift Detector* (IBDD) [46] achieves great classifier predictive performance on many data streams, although it does not perform as well when assessed with lpd and MTR. Here *Discriminative Drift Detector* (D3) [45] and *Semi-Parametric Log Likelihood* (SPLL) [39] showed the best results on most data streams, whilst also achieving good classifier predictive performance in general. Hence, these might be the most sensible detectors for initial deployment in a situation with limited knowledge about the concept drift. Finally, the results of these experiments reinforce that classifier predictive performance is biased in favor of a higher number of detected concept drifts and adaptations. This study also shows that $lpd_{r=1}$ is likewise biased, as the version used in this study evidently favors fewer detected concept drifts.

Beyond benchmarks, this study highlights a few open research challenges, which are concerned among others with the evaluation of **non-abrupt concept drift, approaches to choosing sample sizes or AutoML for concept drift detection**. While these challenges pose grander questions, some minor next steps remain for the implementations provided with this study. Firstly, the computational performance of some

detectors could be improved, e.g., by changing clustering methods to incremental versions. This would enable computational performance benchmarks as planned by Werner et al. [64]. Secondly, the best performing detectors could be included in online machine learning frameworks or concept drift detection frameworks. Finally, the implementations made available with this study could be used to create a heterogeneous ensemble of concept drift detectors.

Supplementary Information All source code used in this study, including model configurations and used random seeds, is available under <https://github.com/DFKI-NI/unsupervised-concept-drift-detection>.

Acknowledgements We thank Daphne Theodorakopoulos for the code review.

Author Contributions DL conceived the study, conducted the literature search, implemented the algorithms and experiments, conducted the experiments and analyzed the results. FS reviewed the experiment design and aided in the analysis. AH, FS and OZ provided supervision and aided with the conception of this study. DL took the lead in writing the manuscript. All authors provided critical feedback and helped shape the study, analysis and manuscript.

Funding This work was funded by VolkswagenStiftung (Volkswagen Foundation) and Niedersächsisches Ministerium für Wissenschaft und Kultur (Lower Saxony's Ministry for Science and Culture) under Niedersächsisches Vorab (grant number ZN3683, *Change Event based Sensor Sampling*).

Data availability All data generated during the experiments are available under <https://github.com/DFKI-NI/unsupervised-concept-drift-detection>. A Python3 script is available to generate all Figures from

this study. Instructions for installation and execution can be found in the repository's README file. The real-world data streams used in the experiments were published by Souza et al. [32] at <https://sites.google.com/view/uspsrepository>. The Electricity2 data stream is available from *River* [57] as well.

Declarations

Conflict of interest The authors have no Conflict of interest to declare that are relevant to the content of this article.

Ethical approval This research did not involve human participants or animals.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

Appendix A Implementation details

In some publications the algorithm description left room for implementation or was at odds with the source code provided by the authors. Furthermore, some algorithms do not adapt to the new concept after a drift. These cases and the respective choices concerning this study's implementation are outlined below. To understand some of these implementation choices, more in-depth study of the respective publication or the algorithm's source code may be required.

BNDM

The source code provided by the authors of *BNDM* uses a Matlab implementation of the Pólya tree test by Caron [65]. In this implementation the data is normalized using the median and the interquartile range. If the 25% and the 75% element have the same value, this will lead to a division by zero, which is not addressed by the authors of *BNDM* [48] nor by Caron. In this case, this study's implementation subtracts the median from the sample without adjusting its range.

The paper and Caron's Matlab implementation provide different information on the return value when the maximum recursion depth is reached. This study uses the same value as the Matlab implementation.

A reset mechanism that purges the data windows after a concept drift was added, as explained in Sect. 5.3.

CSDDM

Occasionally, one of the samples generated for evaluation with the k -sample Anderson-Darling test may be empty. As there is no information regarding this issue in the paper, the hypothesis test is skipped in this case and no concept drift is detected.

IBDD

The authors mention normalization specifically for images, by which they mean transforming the image to grayscale [46]. Since there are no further mentions of normalization, the data is not normalized in this study's implementation.

SPLL

As part of the Gaussian mixture model a covariance matrix is determined. From the publication, it is not quite clear if this covariance matrix is computed over the entire data or only the reference data. Since the Gaussian mixture model is fit on the reference data only, the implementation for this study computes the covariance matrix on the reference data only as well. A reset mechanism was added by purging the reference data window, as explained in Sect. 5.3.

UDetect

A reset mechanism that purges the data windows and resets the thresholds after a concept drift was added, as explained in Sect. 5.3.

Appendix B Configurations

Appendix B.1 Classifiers

River was used in version 0.11.1. Both the Hoeffding tree and the naive Bayes classifiers were deployed using the default configuration.

Appendix B.2 Concept drift detector configurations

BNDM

- const: 0.5, 1.0
- max_depth: 2, 3
- n_samples: 100, 250, 500, 1000
- threshold: 0.45, 0.5, 0.55

48 combinations (deterministic)

CSDDM

- confidence: 0.05, 0.01
- feature_proportion: 0.1, 0.01
- n_clusters: 2, 3
- n_samples: 100, 250, 500, 1000

32 combinations (non-deterministic → 160 runs)

D3

- n_reference_samples: 50, 125, 250, 500
- n_recent_samples_proportion: 0.1, 0.5, 1.0
- threshold: 0.6, 0.7, 0.8,

36 combinations (non-deterministic → 180 runs)

IBDD

- n_consecutive_deviations: 1, 4
- n_permutations: 10, 20, 40
- n_samples: 100, 250, 500, 1000
- update_interval: 50, 100, 250

72 combinations (non-deterministic → 360 runs)

OCDD

- n_samples: 100, 250, 500, 1000
- threshold: 0.2, 0.3, 0.4, 0.5
- outlier_detector: OneClassSVM (from sklearn)
- outlier_detector_kwargs: {'nu': 0.5, 'kernel': 'rbf', 'gamma': 'auto'}

16 combinations (deterministic)

SPLL

- n_clusters: 2, 3
- n_samples: 100, 250, 500, 1000
- threshold: 0.05, 0.005

16 combinations (non-deterministic → 80 runs)

UDetect

- disjoint_training_windows: True
 - n_samples: 50, 100, 250, 500
 - n_windows: 25, 50, 100
- disjoint_training_windows: False
 - n_samples: 100, 250, 500, 1000
 - n_windows: 50, 100, 250

12 + 15 = 24 combinations (deterministic)

Appendix B.3 Best configurations

This appendix gives the configurations for the results highlighted in Tables 8, 10 and 12.

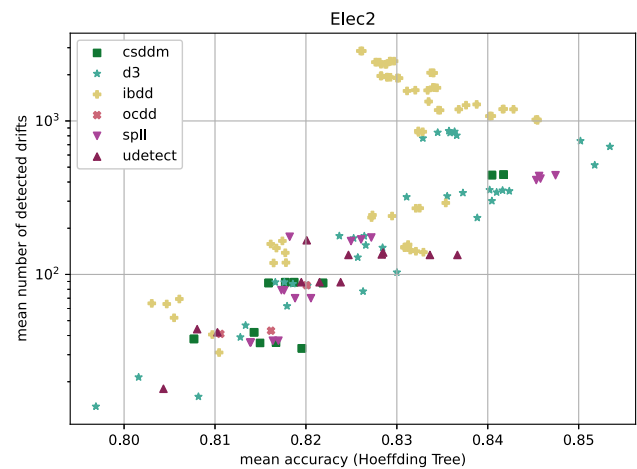


Fig. 8 Accuracy increases with number of detected drifts until a certain point

Table 17 Best performing configurations of BNDM

Data stream	Metric	n_samples	const	threshold	max_depth
INSECTS (abrupt bal.)	<i>MTR</i>	250	1.0	0.55	3
	<i>lpd</i>	1000	0.5	0.55	2
	<i>acc</i>	500	1.0	0.55	2
INSECTS (gradual bal.)	<i>lpd</i>	1000	0.5	0.55	2
	<i>acc</i>	100	0.5	0.5	2
INSECTS (incremental-abrupt bal.)	<i>lpd</i>	1000	1.0	0.55	2
	<i>acc</i>	250	0.5	0.45	2
INSECTS (incremental bal.)	<i>lpd</i>	500	1.0	0.45	3
	<i>acc</i>	500	0.5	0.45	2
INSECTS (incremental-reoccurring bal.)	<i>lpd</i>	1000	1.0	0.45	2
	<i>acc</i>	100	1.0	0.45	3
NOAA Weather	<i>lpd</i>	1000	0.5	0.45	2
	<i>acc</i>	250	0.5	0.55	3
Outdoor Objects	<i>lpd</i>	1000	0.5	0.45	2
	<i>acc</i>	100	0.5	0.45	2
Poker Hands	<i>lpd</i>	250	1.0	0.45	3
	<i>acc</i>	100	0.5	0.5	3
Powersupply	<i>lpd</i>	1000	1.0	0.55	3
	<i>acc</i>	500	0.5	0.45	3
Rialto Bridge Timelapse	<i>lpd</i>	500	1.0	0.5	3
	<i>acc</i>	100	0.5	0.55	3

Table 18 Best performing configurations of CSDDM

Data stream	Metric	n_samples	n_clusters	confidence	feature_proportion
Elec2	<i>lpd</i>	1000	2	0.01	0.1
	<i>acc</i>	100	3	0.05	0.1
	<i>MTR</i>	500	3	0.01	0.01
INSECTS (abrupt bal.)	<i>lpd</i>	1000	3	0.01	0.01
	<i>acc</i>	100	2	0.01	0.01
INSECTS (gradual bal.)	<i>lpd</i>	1000	3	0.01	0.01
	<i>acc</i>	250	2	0.05	0.01
INSECTS (incremental-abrupt bal.)	<i>lpd</i>	1000	2	0.01	0.01
	<i>acc</i>	500	2	0.01	0.1
INSECTS (incremental bal.)	<i>lpd</i>	1000	3	0.01	0.01
	<i>acc</i>	500	2	0.05	0.01
INSECTS (incremental-reoccurring bal.)	<i>lpd</i>	1000	3	0.01	0.01
	<i>acc</i>	250	3	0.05	0.01
NOAA Weather	<i>lpd</i>	1000	3	0.01	0.01
	<i>acc</i>	250	2	0.05	0.1
Outdoor Objects	<i>lpd</i>	1000	2	0.01	0.01
	<i>acc</i>	100	3	0.01	0.01
Poker Hands	<i>lpd</i>	1000	3	0.01	0.01
	<i>acc</i>	100	2	0.05	0.1
Powersupply	<i>lpd</i>	1000	3	0.01	0.01
	<i>acc</i>	1000	2	0.01	0.01
Rialto Bridge Timelapse	<i>lpd</i>	500	2	0.01	0.01
	<i>acc</i>	100	2	0.01	0.01

Table 19 Best performing configurations of D3

Data stream	Metric	n_reference_samples	recent_samples_proportion	threshold
Elec2	<i>lpd</i>	500	1.0	0.8
	<i>acc</i>	50	0.1	0.7
	<i>MTR</i>	125	1.0	0.8
INSECTS (abrupt bal.)	<i>lpd</i>	50	0.5	0.8
	<i>acc</i>	500	0.1	0.6
INSECTS (gradual bal.)	<i>lpd</i>	125	1.0	0.8
	<i>acc</i>	250	0.1	0.6
INSECTS (incremental-abrupt bal.)	<i>lpd</i>	125	1.0	0.8
	<i>acc</i>	250	0.1	0.6
INSECTS (incremental bal.)	<i>lpd</i>	500	1.0	0.6
	<i>acc</i>	500	0.1	0.6
INSECTS (incremental-reoccurring bal.)	<i>lpd</i>	125	1.0	0.8
	<i>acc</i>	125	0.5	0.6
NOAA Weather	<i>lpd</i>	500	1.0	0.8
	<i>acc</i>	250	0.5	0.6
Outdoor Objects	<i>lpd</i>	500	0.5	0.7
	<i>acc</i>	50	0.5	0.6
Poker Hands	<i>lpd</i>	500	0.5	0.8
	<i>acc</i>	50	1.0	0.6
Powersupply	<i>lpd</i>	500	1.0	0.8
	<i>acc</i>	250	0.1	0.8
Rialto Bridge Timelapse	<i>lpd</i>	500	1.0	0.8
	<i>acc</i>	125	0.1	0.6

Appendix C Figures

This appendix contains figures detailing how the number of detected drifts affects accuracy and *lift-per-drift*.

On all but two data streams, detecting more frequently increases the Hoeffding tree's accuracy until a certain point. On *Poker Hands* and *Outdoor Objects* the accuracy keeps increasing with the number of detected drifts, as the data streams are mostly ordered (cf. Sect. 6.3).

On all data streams detecting less frequently increases *lift-per-drift* measured on the Hoeffding tree classifiers in a graph that resembles a negative exponential or logarithmic behavior with a few minor outliers.

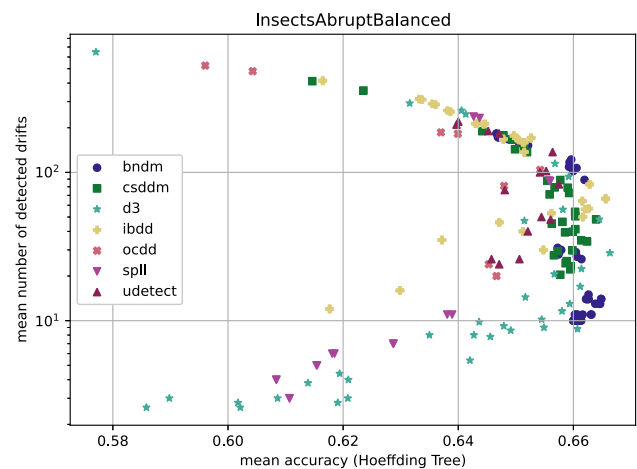
**Fig. 9** Accuracy increases with number of detected drifts until a certain point

Table 20 Best performing configurations of IBDD

Data stream	Metric	n_samples	n_consec._deviations	n_permutations	update_interval
Elec2	<i>lpd</i>	250	4	10	250
	<i>acc</i>	1000	4	10	100
	<i>MTR</i>	1000	4	10	50
INSECTS (abrupt bal.)	<i>lpd</i>	1000	4	40	50
	<i>acc</i>	250	4	40	250
INSECTS (gradual bal.)	<i>lpd</i>	1000	4	40	50
	<i>acc</i>	100	1	40	250
INSECTS (incremental-abrupt bal.)	<i>lpd</i>	1000	4	10	50
	<i>acc</i>	1000	4	40	250
INSECTS (incremental bal.)	<i>lpd</i>	1000	4	40	50
	<i>acc</i>	1000	4	40	100
INSECTS (incremental-reoccurring bal.)	<i>lpd</i>	1000	4	10	50
	<i>acc</i>	100	4	10	250
NOAA Weather	<i>lpd</i>	500	1	10	250
	<i>acc</i>	500	1	20	250
Outdoor Objects	<i>lpd</i>	1000	4	10	250
	<i>acc</i>	250	1	40	50
Poker Hands	<i>lpd</i>	100	4	40	250
	<i>acc</i>	250	1	10	50
Powersupply	<i>lpd</i>	500	4	10	250
	<i>acc</i>	500	1	10	50
Rialto Bridge Timelapse	<i>lpd</i>	1000	4	40	250
	<i>acc</i>	1000	1	20	250

Table 21 Best performing configurations of OCDD

Data stream	Metric	n_samples	threshold
Elec2	<i>lpd</i>	1000	0.5
	<i>acc</i>	500	0.5
	<i>MTR</i>	1000	0.5
INSECTS (abrupt bal.)	<i>lpd</i>	1000	0.5
	<i>acc</i>	500	0.4
INSECTS (gradual bal.)	<i>lpd</i>	500	0.5
	<i>acc</i>	500	0.4
INSECTS (incremental-abrupt bal.)	<i>lpd</i>	1000	0.5
	<i>acc</i>	500	0.3
INSECTS (incremental bal.)	<i>lpd</i>	1000	0.5
	<i>acc</i>	1000	0.5
INSECTS (incremental-reoccurring bal.)	<i>lpd</i>	1000	0.5
	<i>acc</i>	500	0.3
Outdoor Objects	<i>lpd</i>	1000	0.4
	<i>acc</i>	100	0.3
Poker Hands	<i>lpd</i>	1000	0.5
	<i>acc</i>	100	0.5
Rialto Bridge Timelapse	<i>lpd</i>	1000	0.5
	<i>acc</i>	250	0.2

Table 22 Best performing configurations of SPL

Data stream	Metric	n_samples	n_clusters	threshold
Elec2	<i>lpd</i>	1000	3	0.005
	<i>acc</i>	100	2	0.05
	<i>MTR</i>	500	2	0.05
INSECTS (abrupt bal.)	<i>lpd</i>	1000	2	0.005
	<i>acc</i>	100	3	0.005
INSECTS (gradual bal.)	<i>lpd</i>	250	2	0.005
	<i>acc</i>	100	3	0.005
INSECTS (incremental-abrupt bal.)	<i>lpd</i>	500	2	0.005
	<i>acc</i>	100	3	0.005
INSECTS (incremental bal.)	<i>lpd</i>	250	2	0.05
	<i>acc</i>	100	3	0.005
INSECTS (incremental-reoccurring bal.)	<i>lpd</i>	1000	3	0.005
	<i>acc</i>	100	3	0.05
NOAA Weather	<i>lpd</i>	250	2	0.005
	<i>acc</i>	250	2	0.005
Outdoor Objects	<i>lpd</i>	1000	2	0.005
	<i>acc</i>	100	2	0.005
Poker Hands	<i>lpd</i>	1000	3	0.005
	<i>acc</i>	100	2	0.05
Powersupply	<i>lpd</i>	1000	2	0.005
	<i>acc</i>	100	3	0.05
Rialto Bridge Timelapse	<i>lpd</i>	1000	2	0.005
	<i>acc</i>	100	2	0.005

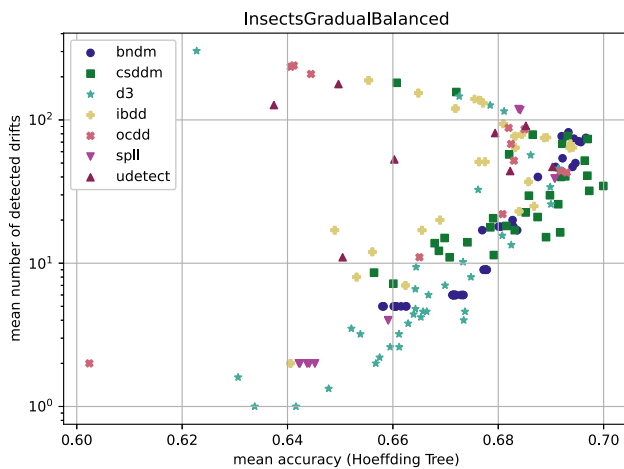
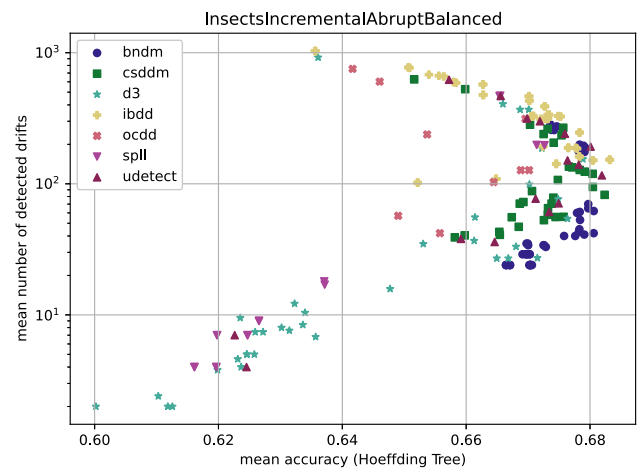
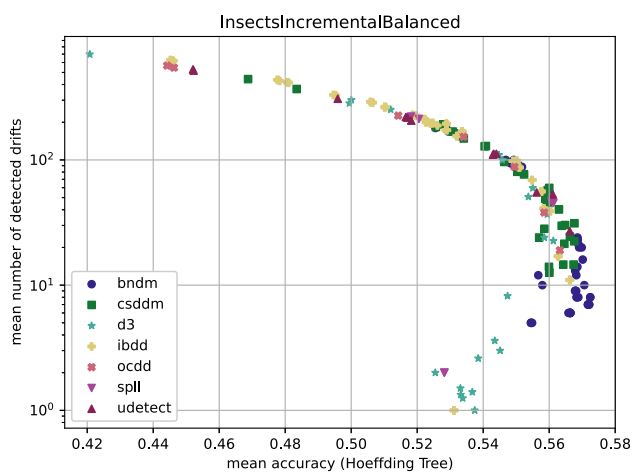
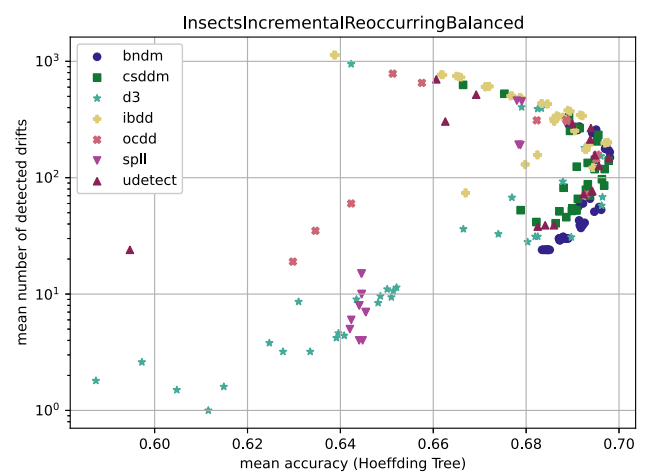
**Fig. 10** Accuracy increases with number of detected drifts until a certain point**Fig. 11** Accuracy increases with number of detected drifts until a certain point

Table 23 Best performing configurations of UDetect

Data stream	Metric	n_samples	n_windows	disjoint_training_windows
Elec2	<i>lpd</i>	1000	250	False
	<i>acc</i>	250	50	False
	<i>MTR</i>	1000	250	False
INSECTS (abrupt bal.)	<i>lpd</i>	2000	250	False
	<i>acc</i>	500	250	False
INSECTS (gradual bal.)	<i>lpd</i>	100	250	False
	<i>acc</i>	500	100	False
INSECTS (incremental-abrupt bal.)	<i>lpd</i>	250	25	True
	<i>acc</i>	500	250	False
INSECTS (incremental bal.)	<i>lpd</i>	2000	250	False
	<i>acc</i>	2000	250	False
INSECTS (incremental-reoccurring bal.)	<i>lpd</i>	500	25	True
	<i>acc</i>	500	100	False
NOAA Weather	<i>lpd</i>	50	25	True
	<i>acc</i>	50	25	True
Outdoor Objects	<i>lpd</i>	100	250	False
	<i>acc</i>	100	50	False
Poker Hands	<i>lpd</i>	100	250	False
	<i>acc</i>	100	100	False
Powersupply	<i>lpd</i>	1000	100	False
	<i>acc</i>	1000	50	False
Rialto Bridge Timelapse	<i>lpd</i>	1000	250	False
	<i>acc</i>	100	100	False

**Fig. 12** Accuracy increases with number of detected drifts until a certain point**Fig. 13** Accuracy increases with number of detected drifts until a certain point

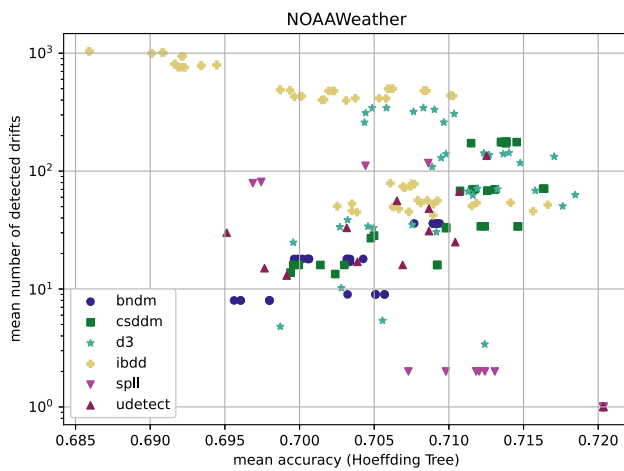


Fig. 14 Accuracy increases with number of detected drifts until a certain point

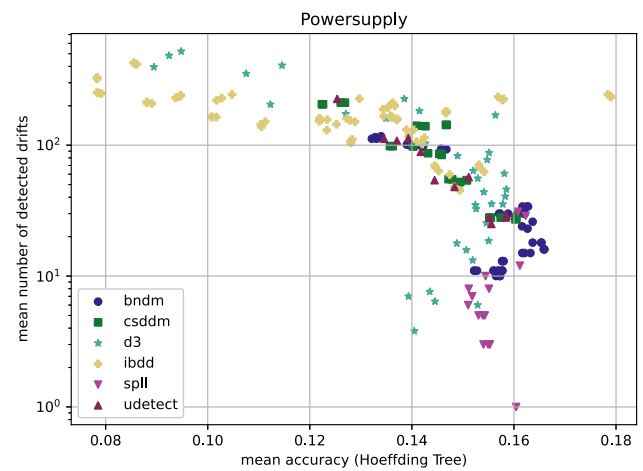


Fig. 17 Accuracy increases with number of detected drifts until a certain point

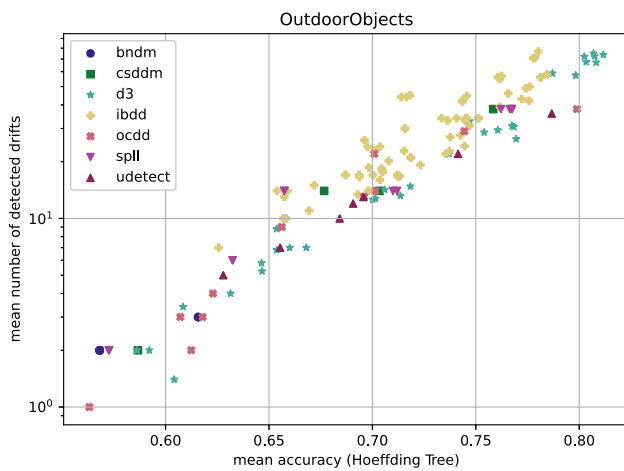


Fig. 15 Accuracy increases with number of detected drifts

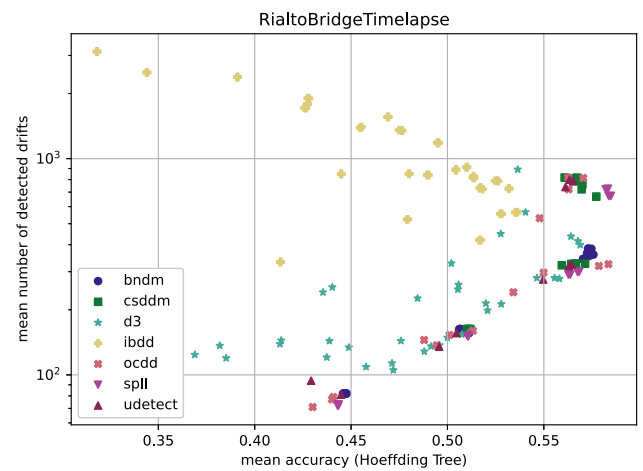


Fig. 18 Accuracy increases with number of detected drifts until a certain point

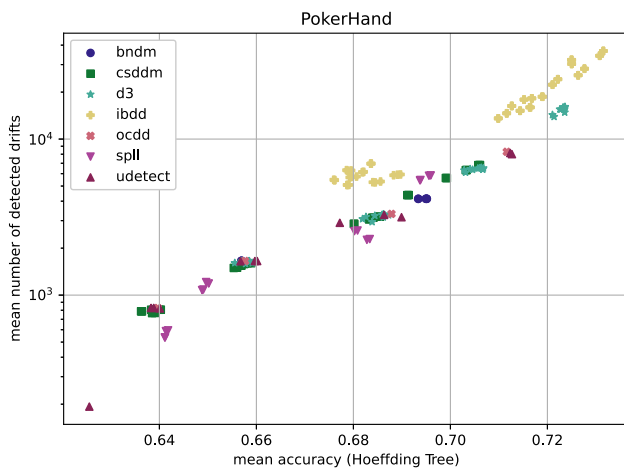


Fig. 16 Accuracy increases with number of detected drifts

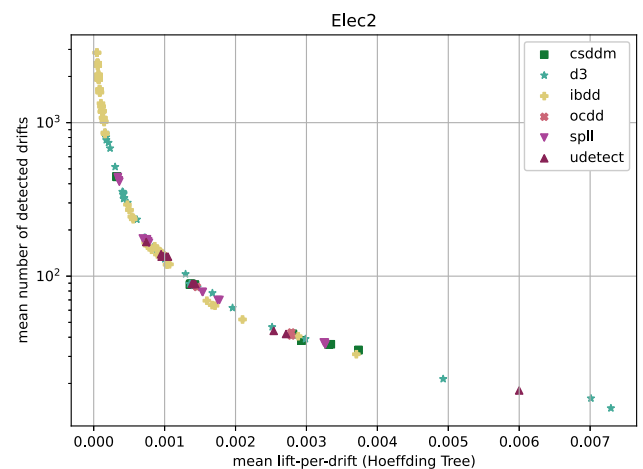


Fig. 19 Lift-per-drift increases as the number of detected drifts decreases

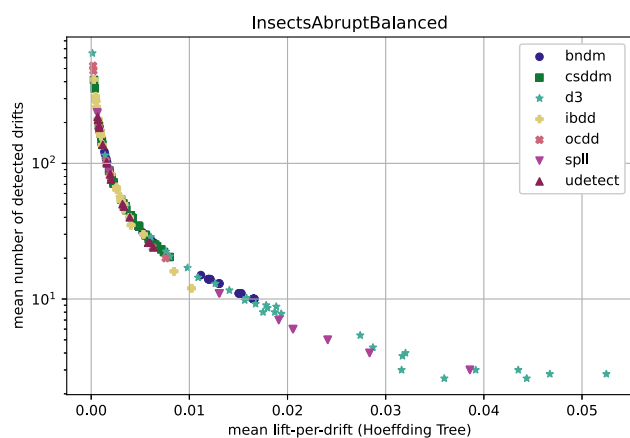


Fig. 20 *Lift-per-drift* increases as the number of detected drifts decreases

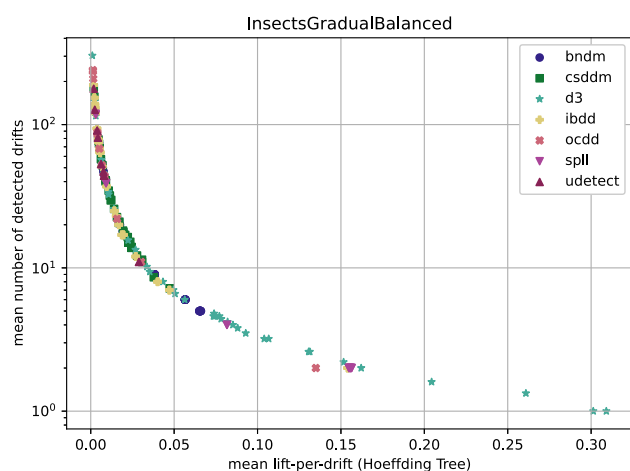


Fig. 21 *Lift-per-drift* increases as the number of detected drifts decreases

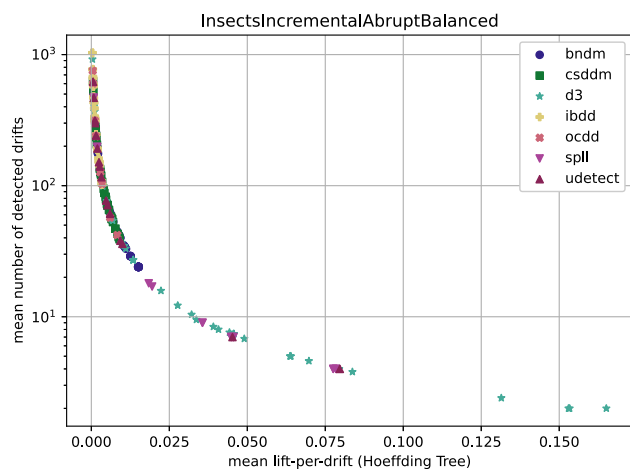


Fig. 22 *Lift-per-drift* increases as the number of detected drifts decreases

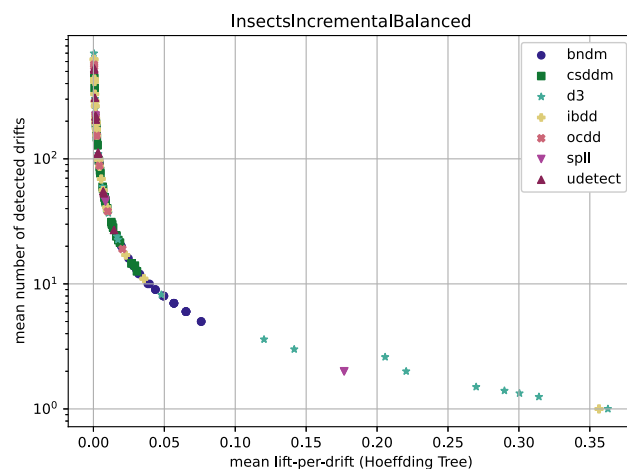


Fig. 23 *Lift-per-drift* increases as the number of detected drifts decreases

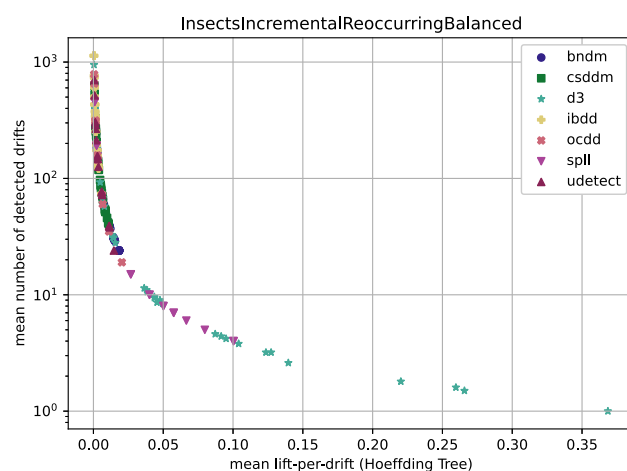


Fig. 24 *Lift-per-drift* increases as the number of detected drifts decreases

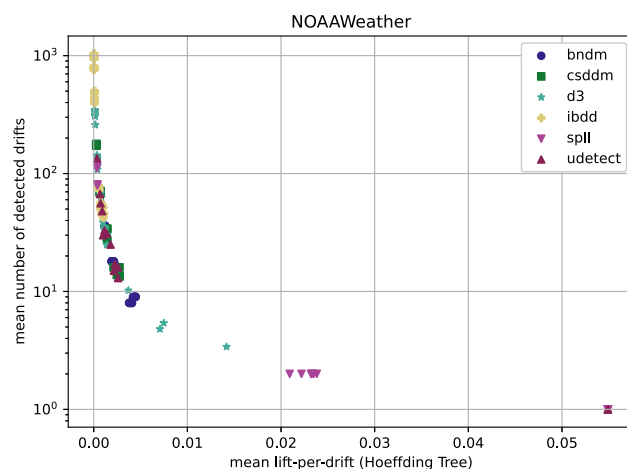


Fig. 25 *Lift-per-drift* increases as the number of detected drifts decreases

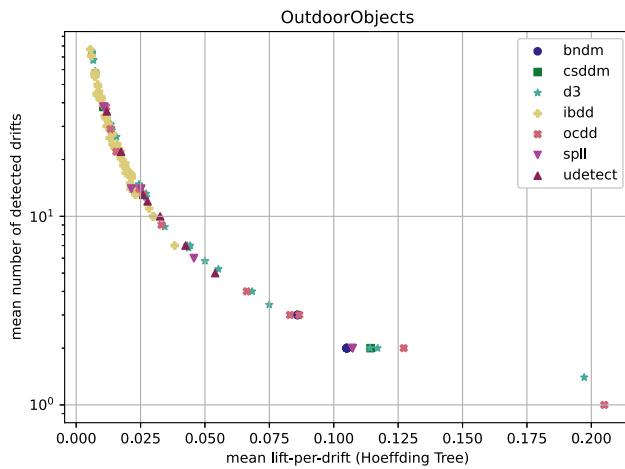


Fig. 26 *Lift-per-drift* increases as the number of detected drifts decreases

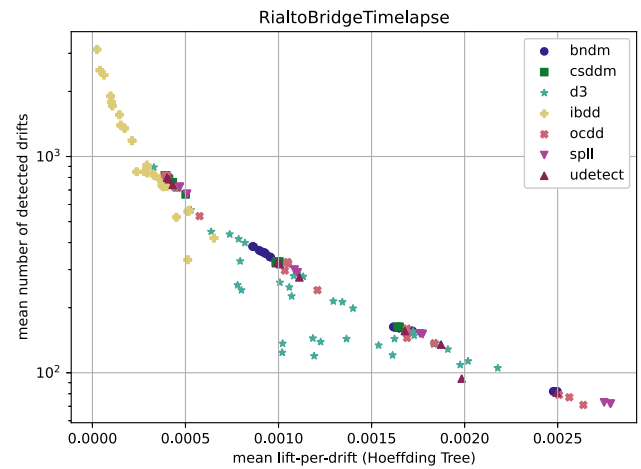


Fig. 29 *Lift-per-drift* increases as the number of detected drifts decreases

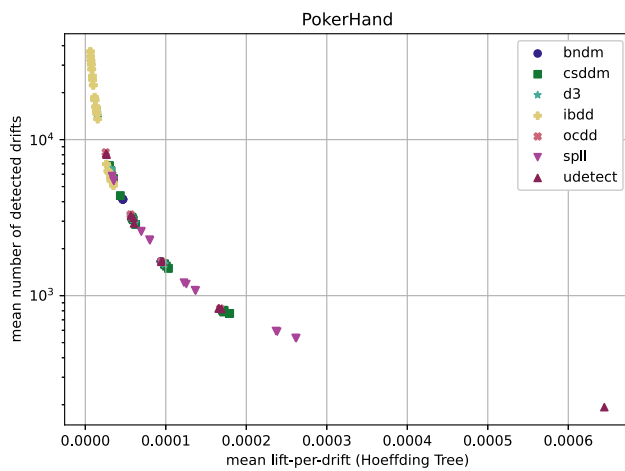


Fig. 27 *Lift-per-drift* increases as the number of detected drifts decreases

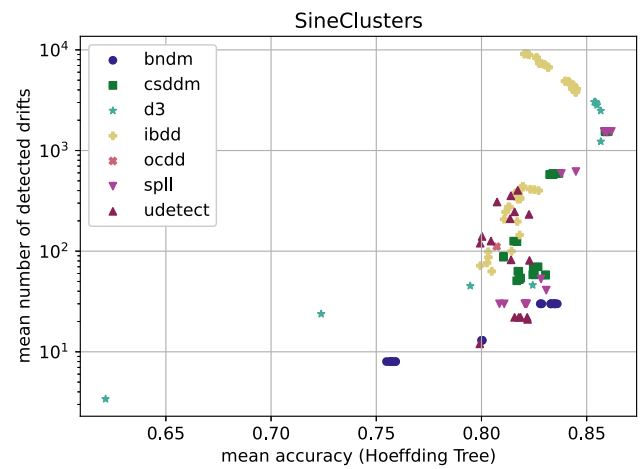


Fig. 30 *Accuracy* increases as the number of detected drifts increases. This synthetic stream has 30 concept drifts

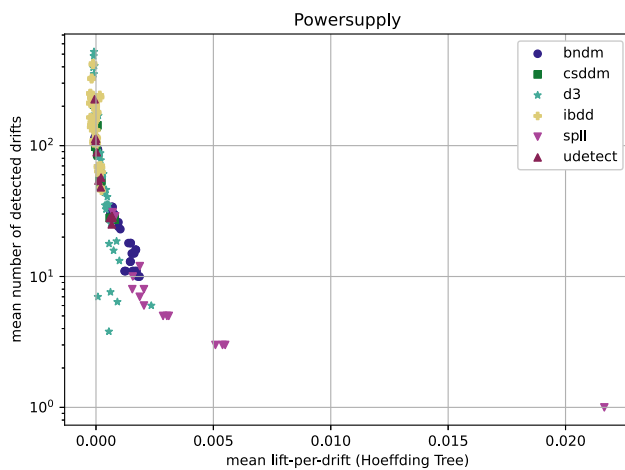


Fig. 28 *Lift-per-drift* increases as the number of detected drifts decreases

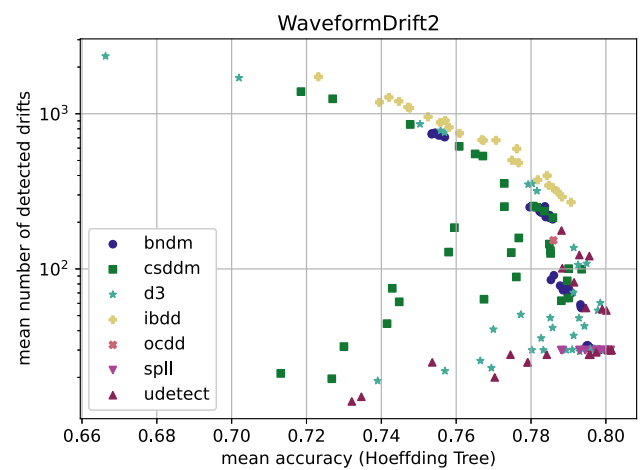


Fig. 31 *Accuracy* does not increase as the number of drifts increases. This synthetic stream has 30 concept drifts

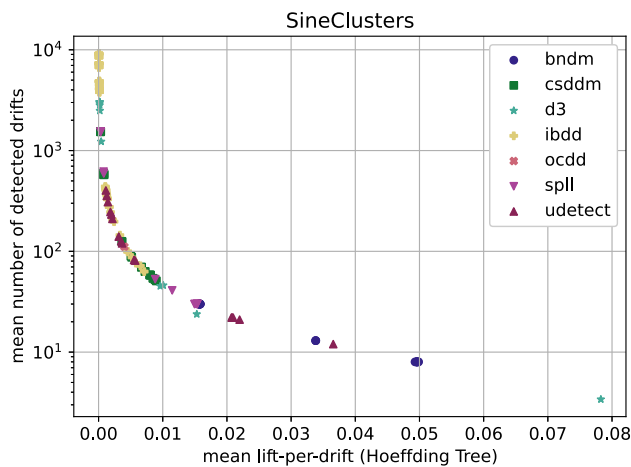


Fig. 32 *Lift-per-drift* increases as the number of detected drifts decreases. This synthetic stream has 30 concept drifts

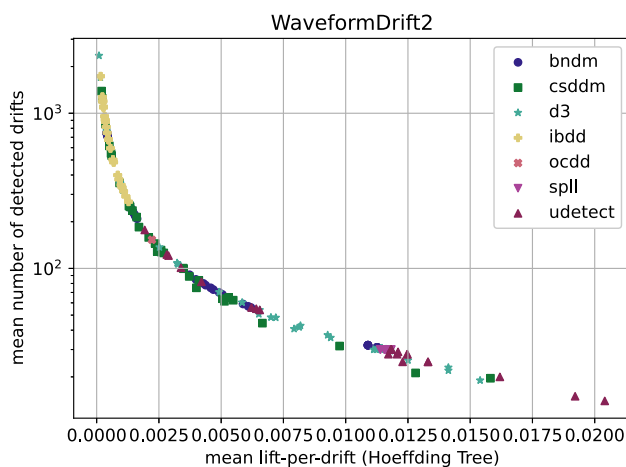


Fig. 33 *Lift-per-drift* increases as the number of detected drifts decreases. This synthetic stream has 30 concept drifts

References

- Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Data stream mining. In: Maimon, O., Rokach, L. (eds.) *Data Mining and Knowledge Discovery Handbook*, pp. 759–787. Springer, Boston (2010). https://doi.org/10.1007/978-0-387-09823-4_39
- Kreml, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., Stefanowski, J.: Open challenges for data stream mining research. *ACM SIGKDD Explor. Newsl.* **16**(1), 1–10 (2014). <https://doi.org/10.1145/2674026.2674028>
- Mukherjee, B., Heberlein, L.T., Levitt, K.N.: Network intrusion detection. *IEEE Netw.* **8**(3), 26–41 (1994). <https://doi.org/10.1109/65.283931>
- D'Alconzo, A., Drago, I., Morichetta, A., Mellia, M., Casas, P.: A survey on big data for network traffic monitoring and analysis. *IEEE Trans. Netw. Serv. Manag.* **16**(3), 800–813 (2019). <https://doi.org/10.1109/TNSM.2019.2933358>
- De Francisci Morales, G., Bifet, A., Khan, L., Gama, J., Fan, W.: IoT big data stream mining. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery And Data Mining*. KDD '16, pp. 2119–2120. Association for Computing Machinery, New York, NY, USA (2016). <https://doi.org/10.1145/2939672.2945385>
- Baschek, B., Schroeder, F., Brix, H., Riethmüller, R., Badewien, T.H., Breitbach, G., Brügg, B., Colijn, F., Doerffer, R., Eschenbach, C., Friedrich, J., Fischer, P., Garthe, S., Horstmann, J., Krasemann, H., Metfies, K., Merckelbach, L., Ohle, N., Petersen, W., Prüffrock, D., Röttgers, R., Schlüter, M., Schulz, J., Schulz-Stellenfleth, J., Stanev, E., Staneva, J., Winter, C., Wirtz, K., Wollschläger, J., Zielinski, O., Ziemer, F.: The coastal observing system for northern and arctic seas (COSYNA). *Ocean Sci.* **13**(3), 379–410 (2017). <https://doi.org/10.5194/os-13-379-2017>
- Zielinski, O., Pieck, D., Schulz, J., Thölen, C., Wollschläger, J., Albinus, M., Badewien, T.H., Braun, A., Engelen, B., Feenders, C., Fock, S., Lehnert, C., Löhms, K., Lübken, A., Massmann, G., Meyerjürgens, J., Nicolai, H., Pollmann, T., Schwalfenberg, K., Stone, J., Waska, H., Winkler, H.: The Spiekeroog coastal observatory: a scientific infrastructure at the land-sea transition zone (Southern North Sea). *Front. Mar. Sci.* **8**, 754905 (2022)
- Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. *ACM Comput. Surv.* **46**(4), 44–14437 (2014). <https://doi.org/10.1145/2523813>
- Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) *Advances in Intelligent Data Analysis VIII. Lecture Notes in Computer Science*, pp. 249–260. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-03915-7_22
- Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) *Advances in Artificial Intelligence—SBIA 2004. Lecture Notes in Computer Science*, pp. 286–295. Springer, Berlin (2004). https://doi.org/10.1007/978-3-540-28645-5_29
- Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing, pp. 443–448. Society for Industrial and Applied Mathematics, Minneapolis (2007). <https://doi.org/10.1137/1.9781611972771.42>
- Zheng, S., Zou, S., Pechenizkiy, M., Campos, C., Ipenburg, W., Harder, H.: Labelless concept drift detection and explanation. In: *NeurIPS 2019 Workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness, and Privacy* (2019)
- Castellani, A., Schmitt, S., Hammer, B.: Task-sensitive concept drift detector with constraint embedding. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 01–08 (2021). <https://doi.org/10.1109/SSCI50451.2021.9659969>
- Agrawal, A., Sazos, M., Al Durra, A., Maniatakos, M.: Towards robust power grid attack protection using LightGBM with concept drift detection and retraining. In: *Proceedings of the 2020 Joint Workshop on CPS&IoT Security And Privacy*. CPSIoTSEC'20, pp. 31–36. Association for Computing Machinery, New York, NY, USA (2020). <https://doi.org/10.1145/3411498.3419964>
- Park, S., Seo, S., Jeong, C., Kim, J.: Network intrusion detection through online transformation of eigenvector reflecting concept drift. In: *Proceedings of the First International Conference on Data Science, E-learning and Information Systems*. DATA '18. Association for Computing Machinery, New York, NY, USA (2018). <https://doi.org/10.1145/3279996.3280013>
- Henke, M., Souto, E., Santos, E.M.D.: Analysis of the evolution of features in classification problems with concept drift: application to spam detection. In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 874–877 (2015). <https://doi.org/10.1109/INM.2015.7140398>
- Wojtkiewicz, J., Katragadda, S., Gottumukkala, R.: A concept-drift based predictive-analytics framework: application for real-time solar irradiance forecasting. In: *2018 IEEE International Conference on Big Data (Big Data)*, pp. 5462–5464 (2018). <https://doi.org/10.1109/BigData.2018.8622216>

18. Suprem, A., Musaev, A., Pu, C.: Concept drift adaptive physical event detection for social media streams. In: Xia, Y., Zhang, L.-J. (eds.) *SERVICES* 2019, vol. 11517, pp. 92–105. Springer, Cham (2019)
19. Moleda, M., Momot, A., Mrozek, D.: Concept drift and avoiding its negative effects in predictive modeling of failures of electricity production units in power plants. In: 2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 1–7 (2020). <https://doi.org/10.1109/MASCOTS50786.2020.9285972>
20. Lukats, D., Berghöfer, E., Stahl, F., Schneider, J., Pieck, D., Idrees, M.M., Nolle, L., Zielinski, O.: Towards concept change detection in marine ecosystems. In: *OCEANS 2021: San Diego - Porto*, pp. 1–10 (2021). <https://doi.org/10.23919/OCEANS44145.2021.9706015>
21. Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., Ghédira, K.: Discussion and review on evolving data streams and concept drift adapting. *Evol. Syst.* **9**(1), 1–23 (2018). <https://doi.org/10.1007/s12530-016-9168-2>
22. Iwashita, A.S., Papa, J.P.: An overview on concept drift learning. *IEEE Access* **7**, 1532–1547 (2019). <https://doi.org/10.1109/ACCESS.2018.2886026>
23. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: a review. *IEEE Trans. Knowl. Data Eng.* **31**(12), 2346–2363 (2019). <https://doi.org/10.1109/TKDE.2018.2876857>
24. Barros, R.S.M., Santos, S.G.T.C.: A large-scale comparison of concept drift detectors. *Inf. Sci.* **451–452**, 348–370 (2018). <https://doi.org/10.1016/j.ins.2018.04.014>
25. Barros, R.S.M., Santos, S.G.T.C.: An overview and comprehensive comparison of ensembles for concept drift. *Inf. Fusion* **52**, 213–244 (2019). <https://doi.org/10.1016/j.inffus.2019.03.006>
26. Gemaque, R.N., Costa, A.F.J., Giusti, R., Santos, E.M.: An overview of unsupervised drift detection methods. *WIREs Data Mining Knowl. Discov.* (2020). <https://doi.org/10.1002/widm.1381>
27. Suárez-Cetrulo, A.L., Quintana, D., Cervantes, A.: A survey on machine learning for recurring concept drifting data streams. *Expert Syst. Appl.* **213**, 118934 (2023). <https://doi.org/10.1016/j.eswa.2022.118934>
28. Xiang, Q., Zi, L., Cong, X., Wang, Y.: Concept drift adaptation methods under the deep learning framework: a literature review. *Appl. Sci.* **13**(11), 6515 (2023). <https://doi.org/10.3390/app13116515>
29. Bifet, A.: Classifier concept drift detection and the illusion of progress. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science*, pp. 715–725. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59060-8_64
30. Anderson, R., Koh, Y.S., Dobbie, G.: Lift-per-drift: an evaluation metric for classification frameworks with concept drift detection. In: Mitrovic, T., Xue, B., Li, X. (eds.) *AI 2018: Advances in Artificial Intelligence. Lecture Notes in Computer Science*, pp. 630–642. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03991-2_57
31. Lukats, D., Stahl, F.: On reproducible implementations in unsupervised concept drift detection algorithms research. In: Bramer, M., Stahl, F. (eds.) *Artificial Intelligence XL. Lecture Notes in Computer Science*, pp. 204–209. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47994-6_16
32. Souza, V.M.A., Reis, D.M., Maletzke, A.G., Batista, G.E.A.P.A.: Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining Knowl. Discov.* **34**(6), 1805–1858 (2020). <https://doi.org/10.1007/s10618-020-00698-5>
33. D’Ettorre, S., Viktor, H.L., Paquet, E.: Context-based abrupt change detection and adaptation for categorical data streams. In: Yamamoto, A., Kida, T., Uno, T., Kuboyama, T. (eds.) *Discovery Science. Lecture Notes in Computer Science*. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67786-6_1
34. Balestra, C., Li, B., Müller, E.: slidSHAPs—sliding Shapley values for correlation-based change detection in time series. In: 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA) (2023). <https://doi.org/10.1109/DSAA60987.2023.10302636>
35. Bu, L., Alippi, C., Zhao, D.: A pdf-free change detection test based on density difference estimation. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(2), 324–334 (2018). <https://doi.org/10.1109/TNNLS.2016.2619909>
36. Wan, J.S.-W., Wang, S.-D.: Concept drift detection based on pre-clustering and statistical testing. *J. Internet Technol.* **22**(2), 465–472 (2021)
37. Gözüağık, Ö., Can, F.: Concept learning using one-class classifiers for implicit drift detection in evolving data streams. *Artif. Intell. Rev.* (2020). <https://doi.org/10.1007/s10462-020-09939-x>
38. Shang, D., Zhang, G., Lu, J.: Fast concept drift detection using unlabeled data. In: *Developments of Artificial Intelligence Technologies in Computation And Robotics. World Scientific Proceedings Series on Computer Engineering and Information Science*, vol. Volume 12, pp. 133–140. WORLD SCIENTIFIC, N/A (2020). https://doi.org/10.1142/9789811223334_0017
39. Kuncheva, L.I.: Change detection in streaming multivariate data using likelihood detectors. *IEEE Trans. Knowl. Data Eng.* **25**(5), 1175–1180 (2013). <https://doi.org/10.1109/TKDE.2011.226>
40. Liu, A., Lu, J., Liu, F., Zhang, G.: Accumulating regional density dissimilarity for concept drift detection in data streams. *Pattern Recognit.* **76**, 256–272 (2018). <https://doi.org/10.1016/j.patcog.2017.11.009>
41. Jaworski, M., Rutkowski, L., Angelov, P.: Concept drift detection using autoencoders in data streams processing. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) *Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science*, pp. 124–133. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61401-0_12
42. Kaminskyi, D., Li, B., Müller, E.: Reconstruction-based unsupervised drift detection over multivariate streaming data. In: 2022 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 807–813 (2022). <https://doi.org/10.1109/ICDMW58026.2022.00109>
43. Friedrich, B., Sawabe, T., Hein, A.: Unsupervised statistical concept drift detection for behaviour abnormality detection. *Appl. Intell.* **53**(3), 2527–2537 (2023). <https://doi.org/10.1007/s10489-022-03611-3>
44. Qahtan, A.A., Alharbi, B., Wang, S., Zhang, X.: A PCA-based change detection framework for multidimensional data streams: change detection in multidimensional data streams. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery And Data Mining. KDD ’15*, pp. 935–944. Association for Computing Machinery, New York, NY, USA (2015). <https://doi.org/10.1145/2783258.2783359>
45. Gözüağık, Ö., Büyükçakır, A., Bonab, H., Can, F.: Unsupervised concept drift detection with a discriminative classifier. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management. CIKM ’19*, pp. 2365–2368. Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3357384.3358144>
46. Souza, V.M.A., Chowdhury, F.A., Mueen, A.: Unsupervised drift detection on high-speed data streams. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 102–111 (2020). <https://doi.org/10.1109/BigData50022.2020.9377880>

47. Bashir, S.A., Petrovski, A., Doolan, D.: A framework for unsupervised change detection in activity recognition. *Int. J. Pervasive Comput. Commun.* **13**(2), 157–175 (2017). <https://doi.org/10.1108/IJPPCC-03-2017-0027>
48. Xuan, J., Lu, J., Zhang, G.: Bayesian nonparametric unsupervised concept drift detection for data stream mining. *ACM Trans. Intell. Syst. Technol.* **12**(1), 5–1522 (2020). <https://doi.org/10.1145/3420034>
49. Korycki, L., Krawczyk, B.: Unsupervised drift detector ensembles for data stream mining. In: 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 317–325 (2019). <https://doi.org/10.1109/DSAA.2019.00047>
50. Holmes, C.C., Caron, F., Griffin, J.E., Stephens, D.A.: Two-sample Bayesian nonparametric hypothesis testing. *Bayesian Anal.* **10**(2), 297–320 (2015). <https://doi.org/10.1214/14-BA914>
51. Scholz, F.W., Stephens, M.A.: K-sample Anderson–Darling tests. *J. Am. Stat. Assoc.* **82**(399), 918–924 (1987). <https://doi.org/10.2307/2288805>
52. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
53. Schölkopf, B., Williamson, R.C., Smola, A., Shawe-Taylor, J., Platt, J.: Support vector method for novelty detection. In: Solla, S., Leen, T., Müller, K. (eds.) *Advances in Neural Information Processing Systems*, vol. 12. MIT Press, Denver (1999)
54. Moore, D.S.: The effect of dependence on chi squared tests of fit. *Ann. Stat.* **10**(4), 1163–1171 (1982)
55. Liu, R.Y., Tang, J.: Control charts for dependent and independent measurements based on bootstrap methods. *J. Am. Stat. Assoc.* **91**(436), 1694–1700 (1996). <https://doi.org/10.2307/2291598>
56. Yao, Y., Chakraborti, S., Yang, X., Parton, J., Lewis, D., Jr., Huddnall, M.: Phase I control chart for individual autocorrelated data: application to prescription opioid monitoring. *J. Qual. Technol.* **55**(3), 302–317 (2023). <https://doi.org/10.1080/00224065.2022.2139783>
57. Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdesslem, T., Bifet, A.: River: machine learning for streaming data in Python. *J. Mach. Learn. Res.* **22**(1), 4945–4952 (2021)
58. Domingos, P., Hulten, G.: Mining high-speed data streams. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 71–80. ACM, Boston Massachusetts USA (2000). <https://doi.org/10.1145/347090.347107>
59. Žliobaite, I., Bifet, A., Read, J., Pfahringer, B., Holmes, G.: Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Mach. Learn.* **98**(3), 455–482 (2015). <https://doi.org/10.1007/s10994-014-5441-4>
60. Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., Seidl, T.: Moa: massive online analysis, a framework for stream classification and clustering. In: Diethe, T., Cristianini, N., Shawe-Taylor, J. (eds.) *Proceedings of the First Workshop on Applications of Pattern Analysis. Proceedings of Machine Learning Research*, vol. 11, pp. 44–50. PMLR, Cumberland Lodge, Windsor, UK (2010)
61. Klein, S., Verbeke, M.: An unsupervised methodology for online drift detection in multivariate industrial datasets. In: 2020 International Conference on Data Mining Workshops (ICDMW), pp. 392–399 (2020). <https://doi.org/10.1109/ICDMW51313.2020.00061>
62. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer, Cham (2019). <https://doi.org/10.1007/978-3-030-05318-5>. Accessed 2024-06-17
63. Theodorakopoulos, D., Stahl, F., Lindauer, M.: Hyperparameter Importance Analysis for Multi-Objective AutoML. [arXiv:2405.07640](https://arxiv.org/abs/2405.07640) [cs] (2024). Accessed 2024-06-17
64. Werner, E., Kumar, N., Lieber, M., Torge, S., Gumhold, S., Nagel, W.E.: Towards computational performance engineering for unsupervised concept drift detection—complexities, benchmarking, performance analysis. [arXiv:2304.08319](https://arxiv.org/abs/2304.08319) [cs] (2024). Accessed 2024-06-17
65. François Caron: Matlab/Octave package Polyatreetest (2014). <https://www.stats.ox.ac.uk/%7Ecaron/code/polyatreetest/index.html>. Accessed 2023-05-04

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.