Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Statistical Drift Detection Ensemble for batch processing of data streams

Joanna Komorniczak [*], Paweł Zyblewski, Paweł Ksieniewicz

Department of Systems and Computer Networks, Wroclaw University of Science and Technology, wyb. Wyspianskiego 27, Wrocław, 50370, Poland

ARTICLE INFO

Article history:
Received 24 March 2022
Received in revised form 2 June 2022
Accepted 3 July 2022
Available online 14 July 2022

MSC:
00-01
99-00

Keywords:
Data streams
Concept drift
Drift detection
Statistical drift detection
Classification

ABSTRACT

Among the difficulties being considered in data stream processing, a particularly interesting one is the phenomenon of concept drift. Methods of concept drift detection are frequently used to eliminate the negative impact on the quality of classification in the environment of evolving concepts. This article proposes *Statistical Drift Detection Ensemble* (SDDE), a novel method of concept drift detection. The method uses *drift magnitude* and *conditioned marginal covariate drift* measures, analyzed by an ensemble of detectors, whose members focus on random subspaces of the stream's features. The proposed detector was compared with *state-of-the-art* methods on both synthetic data streams and the semi-synthetic streams generated based on the real-world concepts. A series of computer experiments and a statistical analysis of the results, both for the classification accuracy and *Drift Detection errors* were carried out and confirmed the effectiveness of the proposed method.

1. Introduction

In recent years, with the dynamic development of mobile technologies and network services, the process of data transmission, analysis, and collection has been taking place almost continuously — from communication via computer networks to monitoring of weather conditions [1]. In conjunction with these events and the forecasted growth of generated data [2], the demand for methods dedicated to the processing of streaming data is growing.

The hallmark of a data stream is a potentially infinite flow of data. There are also expected difficulties in the case of stream data analysis – e.g., missing values [3] or lack of uniformity [4]. Methods dedicated to analyzing streaming data should be prepared to deal with any faults and inconveniences throughout the data inflow. They should also assume single-time access to the data [5] – one cannot allow the accumulation of a theoretically infinite amount of data in the physically limited memory of the implemented method [6].

Changes are a natural consequence of analyzing infinitely incoming data. As a result of daily physiological and seasonal cycles and changes in trends in social media, we will observe drifts both in concept and in the rate of data imbalance [7,8]. Concept drifts describe the change of posterior probability in the data stream over time. In the context of the dynamics of these changes, we can divide concept drifts into *sudden*, *incremental*, and *gradual* [9]. In the event of *sudden* drifts, the concept is changing rapidly in the span of just a few consecutive observations. In the case of *gradual* and *incremental* concept drift, the changes are more fluid, stretched over time. Correspondingly in *gradual* drifts – the change is smooth – but there are moments of coexistence of two separate concepts without a uniform transition between them. Drifts can as well be described by *recurring context* – a concept that has arisen in the past may reappear after some time period.

Another known taxonomy is the separation into *real* and *virtual* drifts. *Real* drifts, also mentioned in the literature as *class drifts* [10], are related to the shift of the decision boundary during processing. In the case of *virtual* drifts, on the other hand, despite the changes taking place in the data distribution, the decision boundary is not shifted between the classes in the recognition problem [4]. However, over the years, different definitions of *virtual* drift arose. According to the original work of Widmer et al. [11], this category of drifts was viewed as the effect of the computer model's bias changing over time [2].

The main contributions of the following work are:

- Proposing an effective concept drift detection ensemble method, analyzing *drift magnitude* and *conditioned marginal covariate drift* measures, without a need for base classifier evaluation,
- Implementation of the proposed method as well as *state-of-the-art* drift detectors for batch processing of data streams

[*] Corresponding author.
E-mail address: joanna.komorniczak@vp.pl (J. Komorniczak).

https://doi.org/10.1016/j.knosys.2022.109380
0950-7051/© 2022 Published by Elsevier B.V.

in *Python* programming language available in the GitHub repository,[1]
- Proposing three base *Drift Detection error* measures for assessing the quality of drift detection in data streams,
- Evaluation of the proposed method performance and comparison with reference drift detectors.

2. Related works

2.1. Drift detectors

Drift detectors are one of the possible approaches to minimizing the negative effect of concept drifts on the effectiveness of *recognition systems*. Detectors are usually integrated with classifiers [4], not only to have a possibility to utilize the measure of a classification error rate but also to recover its quality after the drift event. The usual procedure used in the case of detection is to replace the base classifier with a new one [12]. Effective and well-known drift detectors using classification error measurements are – among others – *Drift Detection Method* (DDM) [13] and *Early Drift Detection Method* (EDDM) [14].

DDM detects drifts based on the classification error of the base classifier and its standard deviation. It uses the assumption that the quality of the classification should increase with the time of stream processing. If otherwise, the method detects a change of concept. EDDM uses a similar method but measures the distances between upcoming classification errors and assumes that these distances should decrease with time. As for DDM – if otherwise – a concept change is signaled.

Adaptive Windowing (ADWIN) [15] uses the instances of a resizable sliding window. Two window instances – representing old and new data – are stored in memory. The specific difference between the mean values of the window data determines the moment of concept change detection. The window approach was also used in *Paired Learners* (PL) [16] method of concept drift detection. The PL algorithm uses two recognition models — the *static* one, trained with all incoming patterns, and the *reactive* one, trained only with patterns from the recent past. If the classification quality of the *static* model falls below the classification quality of the *reactive* classifier, a concept change is marked.

A *Drift Detection Method* based on the Hoeffding's inequality (HDDM) [17] monitors the performance during the learning process, using probability inequalities. The method can use two approaches — moving averages (HDDM_A), which is more suitable for sudden changes in concept, and weighted moving averages (HDDM_W), dedicated to gradual concept drifts detection.

An interesting approach, being also the starting point for the proposal presented in this article, is *Statistical Drift Detection Method* (SDDM) [18] – a drift detector dedicated to the environment with the lack of immediate access to the classification quality. In real-world applications, labels may often arrive after an extended period or not be available. Detecting drift with delay will not provide the benefits of an immediate model rebuild [6], therefore will affect classification accuracy. SDDM provides, in addition to the detection itself, information about the source of the drift and its nature. Therefore it belongs to interpretable machine learning methods. The mentioned algorithms enable the model's behaviors and predictions to be understood by humans. By using interpretable methods, we can ensure that the systems are not prejudiced against the particular types of input [19].

A series of measures have been proposed to determine the drift characteristics [10], e.g. *drift magnitude* – a measure that describes the distance between two concepts. The original measure of distance was *Hellinger Distance*. The publication by Webb et al. [20] proposes measures of *total drift magnitude*, which uses *total variation distance* [21], and two other *marginal drift magnitude* measures: *conditioned marginal covariate drift* and *posterior drift*.

The research by Micevska et al. [18] shows that the measures mentioned above become uninformative in high dimensional spaces; hence the authors decided to evaluate streams based on individual features' *concept drift magnitude*. However, reducing the analysis level to single dimensions also requires the subsequent integration of its results to obtain aggregate detection information for the entire stream.

Ensembles dedicated to minimizing the negative effect of concept drift on pattern recognition tasks have been proposed [22,23], some of which use auxiliary drift detection methods. There exist as well ensembles dedicated to the concept drift detection. The methods proposed in [24,25] integrate detections of different base drift detectors to boost their performance. Overall, the topic of ensemble-assisted drift detection has not been well explored [5].

The drift detection methods' effectiveness measures are often based on the classification quality during stream processing. The work by Bifet [26] sensibly advises not to evaluate drift detection methods using only overall classification quality. Evaluation based on the classification quality is derived from the most common and straightforward type of action performed in the case of the concept change, aimed at maintaining the quality of the classification — updating the classifier or replacing the current one with a new instance [27]. The experiment carried out in the publication mentioned above [26] shows that the reference method that does not detect changes but artificially simulates the detection can achieve better results than all of the examined detectors.

Three basic measures for the assessment of drift detectors are proposed: *Mean Time between False Alarms*, *Mean Time to Detection*, *Missed Detection Rate*, and two aggregated measures: *Average Run Length* and *Mean Time Ratio*. Before drift occurs, all detections are treated as a false alarm. After the actual drift, the first detection is considered a true alarm. These measures require ground-truth of a stream in the context of the concept changes. Obtaining such information is a significant advantage of synthetic drift injection. In the real-world streams, we cannot be sure about the moment of the concept drift and their type, which makes carrying out experimental evaluation on this type of data complex and highly inconclusive [27].

This publication proposes evaluations using a series of measures other than the ones proposed by Bifet [26]. In the case of gradual and incremental drifts, the unequivocal moment of the concept change is indistinct. In the streams used in this publication, ground truth is determined in the middle of the concept change period. All detections recognized earlier than the ground truth would be marked as a false alarm, which means that the correct detection of incremental and gradual drift, appearing in its initial phase, would be interpreted as erroneous. Such an evaluation approach significantly lowers the results for methods efficient in detecting non-sudden drift. The moment of detection in incremental and gradual drifts should depend on the detection method's sensitivity.

Additionally, according to the mentioned publication, the ideal detector's average time between false alarms should be high. In the case of several detections signaling one non-sudden drift of long duration, the value of this measure will be small; however, we consider such behavior of the detector as desirable.

[1] https://github.com/w4k2/statistical-drift-detection.

2.2. Ensemble methods for drifting data stream classification

Ensemble methods are another approach that allows for minimizing the impact of concept drift on the performance of the machine learning systems. These algorithms process the data stream in batches/data chunks or an online manner — one instance at a time [28]. In this case, the goal is to design the ensemble and the combination rule to adapt to changes due to the concept drift emergence.

The majority of batch-based methods are the development of an idea presented initially by the *Streaming Ensemble Algorithm* (SEA) [29]. This method maintains a pool of classifiers with a predetermined size, consisting of models trained on consecutive data chunks. If the limit of ensemble size is exceeded, the classifier with the lowest classification accuracy is removed. Another example is the *Accuracy Weighted Ensemble* (AWE) [30], based on mean square error and proposed by Wang et al. AWE was later extended by Brzezinski and Stefanowski, who proposed the *Accuracy Updated Ensemble* (AUE), introducing the ability to update the base models [31]. Wozniak et al. introduced the *Weighted Aging Ensemble* (WAE), which modifies AWE by allowing various classifier selection and weight calculation approaches [32]. Elwell and Polikar proposed *Learn++.NSE* [33], which modifies *Learn++* [34] by setting weight for training samples to deal with concept drift occurrence, while Gomes et al. introduced the *Adaptive Random Forest* (ARF) algorithm, which employs resampling and adaptive operators to cope with various types of concept drift [35].

Among online ensemble methods for data stream classification, we can distinguish *Online Bagging* (OB) proposed by Oza, which updates base models with the appearance of a new sample according to the Poisson distribution [36]. Later, Bifet et al. modified OB by allowing specifying the lambda value and introducing the output detection codes in the form of the *Leveraging Bagging* [37]. Gomes et al. proposed the *Streaming Random Patches* (SRP) algorithm [38], which combines *Online Bagging* with *Random Subspace* [39]. The sparse online learning algorithm was employed in the *Sparse Online Classification* (SOC) [40] framework by Wang et al.

Lastly, we can distinguish ensemble methods that combine batch-based and online approaches. An example of such an algorithm is the *Kappa Updated Ensemble* (KUE) proposed by Cano and Krawczyk [41]. KUE uses Kappa statistics for dynamic classifier weighting and selection while simultaneously allowing voting abstaining of chosen base models.

3. Methods

In order to enable the experimental evaluation of the method proposed in this paper, it was necessary to develop three essential processing blocks. The first is the *stream-learn* meta-estimator described in Section 3.1, which defines the strategy of using the drift detector in the data stream classification, allowing for effective measurement of the classification model quality in the course of processing. The second – described in Section 3.2 – is a set of evaluation metrics that effectively validate detection quality in environments with non-sudden drifts. The third – most important for this work and described in Section 3.3 – the *Statistical Drift Detection Ensemble*, which is a new method of data stream drift detection, using statistical measures of data stream properties calculated on problem subspaces, integrated into ensemble as a set of binary detectors.

3.1. Meta-estimator

Some concept change detection methods determine their decision based on the classifier's output [12]. Detectors are often integrated with classifiers to enable an up-to-date analysis of classification errors, which allows regeneration of the classification quality after a concept change. If the detector is two-state, i.e., it signalizes warnings of a concept change prior to detection, a parallel classifier can be created and trained with incoming data. If detection is confirmed, the old model is replaced with the new one [4].

Not all detectors analyzed in this paper provide warnings. Hence only confident detections are taken into account. When a concept change is marked, the classifier is restored to its initial state, and the training is restarted from patterns derived from the new concept. The described procedure is performed by implemented meta-estimator. The model passes problem instances to both the detector and the classifier during stream processing and acquires information about recognized detections.

The use of the described meta-estimator allows the evaluation of the detection quality based on the classification accuracy results and makes it possible to compare the detection methods. When the concept changes, the classification accuracy usually degenerates. Therefore if the detector cannot recognize the concept drift, the overall quality of the classification would decrease throughout stream processing. Delay of concept change detection would also harm the classification accuracy score.

3.2. Drift detection errors

Research on the reliable assessment of the recognition model's quality, in particular in the field of classification, often indicates the problem of using aggregated metrics, such as *F-score* or *Gmean*, which tend to hide some part of proper classification efficiency [42]. The apparent profit resulting from the assessment of a single criterion, in a comparative review of the effectiveness of the methods, does not allow obtaining complete information on the processing [43].

In addition, it is crucial where the non-sudden drift reference point is located on the course of a stream. By default, as indicated in the Related works section, it is defined as the equilibrium point of the dynamics of changes [26], the effects of which might be observed – by a suitable method of detection – earlier, at the first signs of drift. Theoretically, this problem could be solved by shifting the ground truth to the very beginning of the drift period. However, this would raise the question of whether it would not be necessary to treat all the detections occurring during the drift as correct. Such an approach would not measure the differences between the detectors' effectiveness. However, it would only give information about its detection and would require further modification and further complications of the metric. Additionally, in the case of non-sudden drifts, it is difficult to identify the correct beginning of the drift. We must remember that concept drifts occur in a space that is both continuous (course of the stream in which we functionally determine the probability of an object coming from a given concept) and discrete (sampling of this function by the objects that appear). Therefore, it is imperative to propose a reasonable compromise of proper drift detection for non-sudden cases.

Trying to avoid the problems mentioned above in the field with a much less structured research protocol, which is the concept drift detection, it was decided not to use or propose any aggregate metric. Batch processing of the data stream allows us to collect indexes of the chunks in which the detection occurred. In the case of synthetic streams, there is also a possibility to calculate the central point of concept drift. Therefore, in the experimental analysis presented in this paper, three easy to interpret, simple metrics defining the basic properties of drift detectors will be used:

D_1 – *closest drift* – the distance of each detection from the nearest drift, normalized as the arithmetic mean from the obtained values.

D_2 – *closest detection* – the distance of each drift from the nearest detection, normalized as the arithmetic mean from the obtained values.

R – *drift to detection ratio* – the proportions between number of drifts and number of detection. The measure is scaled to obtain the optimum value at zero. Absolute value of the scaled ratio is the final error.

Additionally, apart from the basic drift detection metric, the classification accuracy score of the final decision system will also be calculated.

3.3. Statistical Drift Detection Ensemble

This section introduces an ensemble concept drift detection method called the *Statistical Drift Detection Ensemble* (SDDE). This algorithm expands the work by Micevska et al. [18], in which the authors proposed the *Statistical Drift Detection Method* (SDDM), using statistical measures of concept drift defined by Webb et al. [10,20].

According to the taxonomies proposed by Lu et al. [27], and Hu et al. [44], two main groups of solutions can be distinguished among the supervised drift detection methods. The first one, relatively the most popular, bases the detection on the assessment of errors made by the classifier and thus empirically verifies the correctness of the currently used recognition model. The second, to which the method proposed in this paper should be assigned, departs from the performance evaluation and focuses on the analysis of statistical dependencies between the distant portions of data. Such portions should be large enough to be able to infer their distribution and, at the same time, small enough to make this inference valid and current.

The SDDE algorithm detection – as in the case of SDDM – is based on the statistical measures proposed by Webb et al. namely *Drift Magnitude* (DM) [10] and *Conditioned Marginal Covariate Drift* (CMCD) [20]. The first of these measures is defined by the distance between the concepts at time points t and u, defined as the distance between the features distributions $P(X)$ at these time points

$$DM_{t,u} = D(P_t(X), P_u(X)). \tag{1}$$

Any measure of the distance between the distributions can be used here, however, for the purposes of the SDDE algorithm, *Hellinger's distance* [45] was chosen which, for two discrete probability distributions $P = (p_1, \dots, p_k)$ and $Q = (q_1, \dots, q_k)$ is defined as follows

$$H(P, Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})^2}. \tag{2}$$

It is worth mentioning here that SDDE uses *Drift Magnitude* as defined in Web et al. from 2016 [10]. In the study from 2017 [20], the *Total Drift Magnitude* (TDM) metric was proposed, which was based on the *Total Variation Distance* [21].

Conditioned Marginal Covariate Drift is defined as the weighted sum of the distances between the conditional probability distributions for the possible values of covariate attributes for each problem class $P(X|Y)$ between the time points t and u. The weights are the average probability of occurrence of a given class $P(Y)$ at both points in time

$$\sigma_{t,u}^{X|Y} = \sum_{y \in Y}\left[\frac{P_t(y) + P_u(y)}{2}\frac{1}{2}\sum_{\bar{x} \in X}|P_t(\bar{x}|y) - P_u(\bar{x}|y)|\right]. \tag{3}$$

The first version of the SDDE algorithm also used the *Posterior Drift* (PD) measure [20], proposed by Webb et al. This metric is defined similarly to CMCD but uses covariate distributions $P(X)$ and posterior distributions $P(Y|X)$

$$\sigma_{t,u}^{Y|X} = \sum_{\bar{x} \in X}\left[\frac{P_t(\bar{x}) + P_u(\bar{x})}{2}\frac{1}{2}\sum_{y \in Y}|P_t(y|\bar{x}) - P_u(y|\bar{x})|\right]. \tag{4}$$

However, after conducting preliminary experiments, *Posterior Drift* was dropped from the pool of metrics used due to the significant number of unjustified concept drift detections.

All the above-mentioned statistical measures are calculated based on the probability density distributions estimated using *Kernel Density Estimation* (KDE). The KDE implementation according to the *scikit-learn* library [46] with default hyperparameterization was used. *Kernel Density Estimation* is sensitive to the number of problem dimensions, and a large number of features can lead to performance degradation due to the curse of dimensionality. Therefore, SDDE estimates the densities of the probability distributions in the problem subspace set $\mathcal{S} = \{s_1, \dots, s_{n_{max}}\}$, where n_{max} denotes the subspace number. Each problem subspace is created using sampling with replacement, and the *subspace_size* hyperparameter defines their size. At each given moment, we always keep two sets of KDE models. One trained on kth data chunk, denoted by $kernels_k$, and the second one built based on the chunk in which the concept drift was last detected (or on the first chunk of the stream), denoted by $kernels_b$. If a drift is detected, base kernels $kernels_b$ are replaced with current $kernels_k$. These sets correspond to the moments in time t and u, respectively.

Breaking down the problem into subspaces not only allows SDDE to avoid issues related to the large dimensionality but also allows obtaining an ensemble of detectors instead of a single one. In this case, the actual number of detectors corresponds to the number of subspaces multiplied by the number of statistical measures used, which for the proposed SDDE algorithm is equal to 2.

Due to the use of an ensemble approach to concept drift detection, the decision about drift occurrence must be considered on two levels: (i) each base detector and (ii) the entire ensemble. In the case of base detectors, the decision is made based on the difference comparison between the values of the DM_k and $CMCD_k$ statistical measures for the kth data chunk and the mean of the historical values harmonic mean – calculated over each of the base detectors for all previous data chunks – to the standard deviation of the harmonic mean, based on the three-sigma rule.

The principle of the SDDE algorithm is presented in detail by Algorithm 1 and additionally in Fig. 1.

The description of the functions used in the Algorithm 1 is as follows:

- PREPARE_SUBSPACES() – generates – using sampling with replacement – a set of subspaces with a cardinality defined by the n_{max} hyperparameter, where each subspace contains a number of problem features equal to the value of the *subspace_size* hyperparameter.
- FIT_KERNEL_DENSITY() — builds a set of KDE models on each of the subspace that allows estimating the density of probability distributions in the DS_k data chunk using the *Kernel Density Estimation* approach.

Algorithm 1 Pseudocode of the proposed SDDE algorithm.

```
Input:
    $Stream = \{\mathcal{DS}_1, \mathcal{DS}_2, \ldots, \mathcal{DS}_k, \mathcal{DS}_{k+1}, \ldots\}$ – data stream,
    $\sigma$ – value used for a single detector threshold calculation,
    $sensitivity$ – value used for ensemble detection threshold calculation,
    $n_{max}$ – number of subspaces,
    $subspace\_size$ – subspace size for each drift detector,
    $immobilizer$ – number of chunks skipped before the first detection.
Symbols:
    $\mathcal{DS}_k$ – $k$-th data chunk.
    $\mathcal{S} = \{s_1, \ldots, s_{n_{max}}\}$ – set of subspaces for each detector,
    $kernels_k = \{\mathcal{KDE}^k_1, \ldots, \mathcal{KDE}^k_{n_{max}}\}$ – set of KernelDensity models for $k$-th data chunk,
    $kernels_b = \{\mathcal{KDE}^b_1, \ldots, \mathcal{KDE}^b_{n_{max}}\}$ – set of base KernelDensity models,
    $PD_k$ – probability density estimated using $kernels_k$ on a $k$-th chunk,
    $PD^b_k$ – probability density estimated using $kernels_b$ on a $k$-th chunk.
Output:
    $drift$ – the list containing information about drift occurrence for each $\mathcal{DS}$.
$drift \leftarrow \varnothing$
$kernels_b \leftarrow \varnothing$
$\mathcal{S} =$ PREPARE_SUBSPACES($\mathcal{DS}_k$, $subspace\_size$, $n_{max}$)
$drf\_threshold = 2 * n_{max} * sensitivity$
for each $k$, $\mathcal{DS}_k = \{x^1_k, x^2_k, \ldots, x^N_k\}$ in $Stream$ do
    $kernels_k \leftarrow \varnothing$
    for each $s_i$ in $\mathcal{S} = \{s_1, \ldots, s_{n_{max}}\}$ do
        $kernels_k \leftarrow$ FIT_KERNEL_DENSITY($\mathcal{DS}_k$, $s_i$)
    end for
    if $k == 0$ then
        $kernels_b = kernels_k$
    end if
    $PD_k =$ ESTIMATE_DENSITY($\mathcal{DS}_k$, $kernels_k$)
    $PD^b_k =$ ESTIMATE_DENSITY($\mathcal{DS}_k$, $kernels^b_k$)
    $DM_k, CMCD_k =$ CALCULATE_METRICS($PD_k$, $PD^b_k$)
    if $k > immobilizer$ then
        $detection\_count_k =$ GET_DETECTIONS($DM_k$, $CMCD_k$, $\sigma$)
        if $detection\_count_k >= drf\_threshold$ then
            $drift_k \leftarrow True$
            $kernels_b = kernels_k$
        else
            $drift_k \leftarrow False$
        end if
    else
        $drift_k \leftarrow False$
    end if
end for
```

- ESTIMATE_DENSITY() — estimates the density of the probability distributions in the DS_k data chunk for each s_i subspace, using a set of previously trained *Kernel Density Estimation* models found in *kernels*.
- CALCULATE_METRICS() — calculates the values of *Drift Magnitude* and *Conditioned Marginal Covariate Drift* based on the density of probability distributions estimated using Kernel Density models built on kth data chunk and those stored from the beginning of the stream or the last detection of the concept drift occurrence.
- GET_DETECTIONS() — determines, based on the historical values of the statistical metrics, what number of base detectors identified a concept drift in the kth data chunk. The final decision to detect the drift occurrence is based on the current DM and CMCD values and the harmonic mean of the historical values over the base detectors. The sigma parameter defines to which multiple of the standard deviation of the harmonic mean the difference between the current metrics and the averaged harmonic mean is compared to indicate a concept drift occurrence.

In summary, the SDDE method builds a fixed-size pool of detectors, basing the recognition on the distribution density function of consecutive processing chunks. When the detection threshold is exceeded, i.e., a significant change between reference ($kernels_b$) and current distribution ($kernels_k$) is recognized by the ensemble, the estimated reference distributions are exchanged in each of its members, which allows updating the knowledge about the current concept. The critical element here is the batch exchange of base detectors, which are not updated at the time of individual recognition of a concept change but only after reaching consensus within the ensemble.

A simplified example of the SDDE operation is presented in Fig. 2. It shows processing on a flow of thirteen chunks ($\mathcal{DS}_1$–$\mathcal{DS}_{13}$), where the ensemble is built on six random subspaces (s_1–s_6), assuming an *immobilizer* of 2 and a *drift threshold* of 50 percent. The null chunk – where the initial reference distributions are built – is omitted from the example. Each illustration cell represents the response of a pair of detectors (*DM* and *CMCD* metrics) built on the same subspace (s_n).

Ten detections can be observed in the first chunk of processing, which is 83 percent of all detectors. However, this does not lead to a drift detection as the model is then in an immobilized state (blue area). In chunks 3 and 5, the drift is signaled by four detectors. However, it is only 33 percent of the available pool, so the ensemble is not yet reporting the drift, and each detector remains in its original state. In the sixth chunk, the critical mass of 9 detections is reached, which exceeds the *drift threshold* and leads to the recognition of a change in the concept. Each of the detectors is then updated to the distributions obtained on the $\mathcal{DS}_6$ chunk subspaces, and the meta-estimator receives information about the need to rebuild the classification model.

The presented method is – by definition – based on batch processing due to the need to estimate the probability density distributions on the given window. However, it can be adapted for online learning by replacing disjoint data chunks with a sliding window, typical for prequential analysis. This approach will allow the data stream to be processed at the level of individual problem instances while maintaining the memory of a predefined or dynamically selected number of previous samples. Such a solution will enable the estimation of the probability density distributions for online learning. However, it will be associated with a significant increase in the time complexity due to performing the drift detection after the arrival of each problem instance.

4. Experimental set-up

The following section will describe goals of the planed experiments and experimental setup of the conducted research. All experiments and methods have been implemented in *Python* programming language using the *scikit-learn* [46], *stream-learn* [47], *scikit-multiflow* [48] and *numpy* [49] libraries. The base classifier of the *meta-estimator* 3.1, to simplify calculations for extensive computer experiments, was *Gaussian Naïve Bayes Classifier*.

In experiments, both classification accuracy and chunk indexes of detection were collected in order to enable the calculation of *drift detection errors* 3.2.

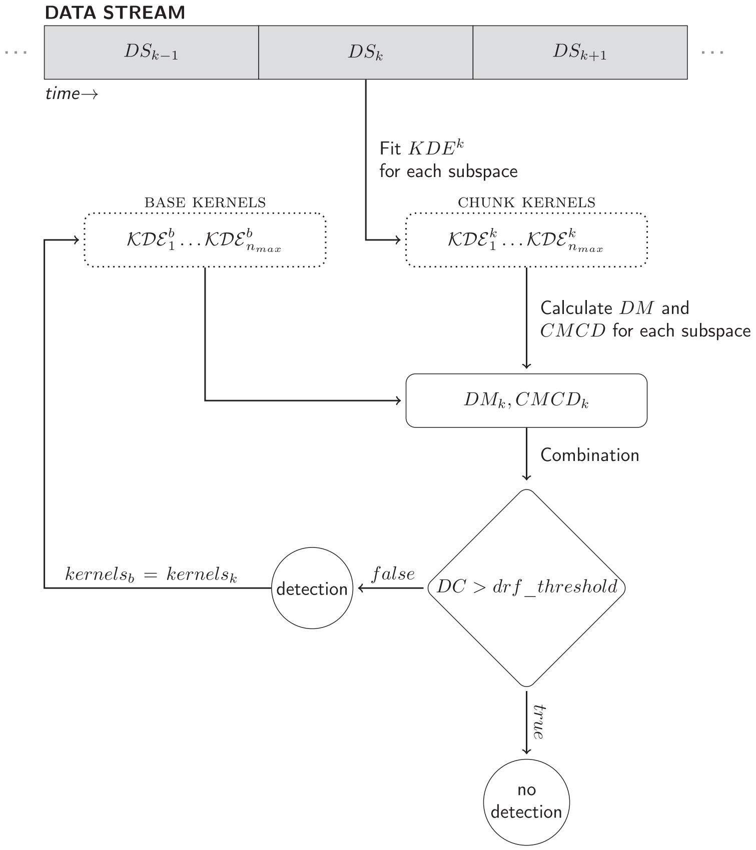


Fig. 1. The principle of SDDE processing.

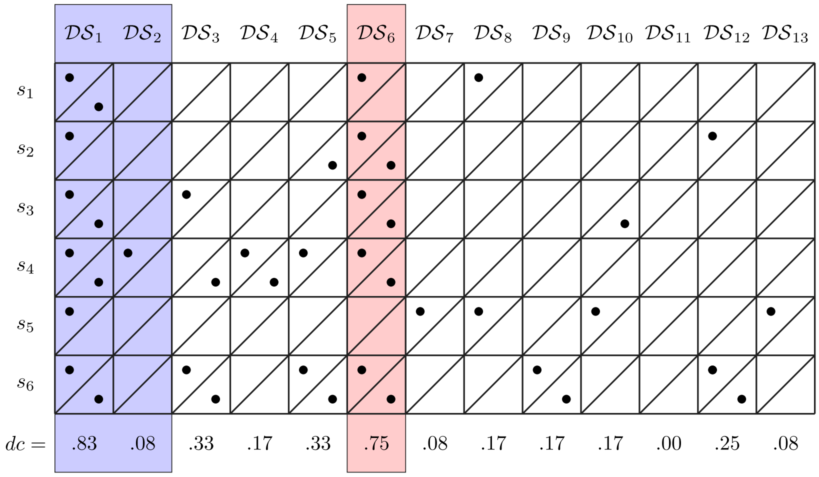


Fig. 2. The example of SDDE processing.

4.1. Experiment 1 — method optimization

The aim of the first experiment was to optimize the SDDE method's hyperparameters such as *sensitivity*, *subspace size* and the *number of detectors*. The experiment was carried out on 30 binary, balanced data streams with five non-recurring concept drifts. Three different concept drift types were taken into consideration during generation of data streams: *sudden*, *incremental* and *gradual*. The streams for hyperparameter optimization were divided into 100 data chunks, each containing 200 instances. Instances were described by 15 informative features and contained label noise at 1%. Each stream was replicated ten times with a different random state. For the purpose of this experiment synthetic data streams were generated by *StreamGenerator* from a *stream-learn* package [47].

In order to optimize the method, the following hyperparameter values were considered:

- *subspace size* containing 1, 2 or 3 random features from initial feature space;
  Research in [18] has shown that *drift magnitude* measures provide the most information for low dimensional spaces, hence low *subspace size* values were evaluated.
- *detector's sensitivity* containing 20 values form 0% to 100%, sampled from linear space;

- The *sensitivity* parameter specifies the fraction of detectors in the ensemble that must detect drift for it to be considered as an integrated decision. The threshold parameter set to zero is the equivalent of stable, deterministic drift detection on every data chunk, even when none of the detectors indicates it. Likewise, a 100 percent threshold requires all detectors to be acclaimed, making the most strict integration rule. Examining the full range of parameter values will allow for a detailed analysis of this parameter effect on an algorithm.
- *number of detectors* containing 20 values from 1 detector to 100, sampled from quadratic function.
- The increase of the *number of detectors* may potentially increase the quality of detection but as well entails an increase in memory and computational complexity. For the evaluated streams containing 15 features, increasing the number of detectors increases the chance of analyzing all features during processing. For the algorithm's usability, we are looking for the smallest parameter value that gives satisfactory results.

4.2. Experiment 2 — comparison with the reference methods

The second experiment aimed to compare the proposed SDDE method with reference drift detection algorithms. For this purpose, the following detectors have been implemented into the programming interface of *stream-learn* library:

METHOD		HYPERPARAMETERS	CITE

DDM	*Drift Detection Method*	warning level = 2.0;
		drift level = 3.0;
		skip= 30
			[13]

EDDM	*Early Drift Detection Method*	warning level = 0.95;
		drift level = 0.9;
			[14]

ADWIN	*Adaptive Windowing*	$\delta = .002$	[15]
HDDM_A	*Hoeffding Drift Detection Method* with *Bounding Moving Averages*	drift level = 0.001; warning level = 0.005	[17]

HDDM_W	*Hoeffding Drift Detection Method* with *Bounding Weighted Moving Averages*	warning level = 0.005;
		drift level = 0.001;
		$\lambda = .05$
			[17]

A total of 320 binary, balanced data streams were generated to carry out a second experiment. The streams were characterized by 3, 5, or 7 concept drifts in 200 data chunks, each containing 250 instances. The instances were described by 10, 15, or 20 informative features with label noise of 1%. The drifts were both recurring and non-recurring of the following types: *sudden*, *incremental* and *gradual*. Each of the streams was replicated ten times with a different random state. Streams were as well generated by *stream-learn* [47] package.

The proposed method – SDDE was parameterized by *sensitivity*, *number of detectors* and *subspace size* selected during analysis of Experiment 1.

4.3. Experiment 3 — real-concept data streams analysis

The purpose of the third experiment was to evaluate the performance of SDDE on streams generated from real concepts using *random projection-based concept drift injector* proposed by Komorniczak et al. [50]. The method is converting real static datasets to data streams with concept drifts of *nearest* and *cubic* types, which correspond to *sudden* and *incremental* drifts. Generator code is publicly available on *GitHub* repository.[2]

Table 1
Original number of instances and features of real-world datasets used for generating data streams.

Dataset	Samples	Features
australian	690	14
banknote	1 372	4
diabetes	768	8
wisconsin	699	9

Utilizing a generator converting real static data to data streams with concept drifts will enable calculating moments of concept change, therefore comparing detections with actual drifts. The available real-world data streams do not contain ground-truth in the context of concept drifts as well may contain drifts of mixed type [27]. The reasons mentioned above contributed to the decision to use streams generated based on real data, ensuring the moment of occurrence and type of drift instead of the original real-world data streams.

Four semi-synthetic data streams were generated based on real-world concepts. Each data stream was described by 15 attributes originating from actual static dataset features. Streams were characterized by 3, 5, and 7 drifts throughout 400 chunks of 250 instances each. The original datasets are described in Table 1.

All detection methods from *Experiment 2* were used to detect drift in the evaluated streams. Two additional pseudo-detectors, designed as the baselines for the purpose of experimental evaluation, were:

ALWAYS – method detecting drift in each chunk;
NEVER – method that never detects drift.

The *sensitivity* parameter was recalibrated for the optimal algorithm's performance of the semi-synthetic streams based on the real-world concepts.

5. Experimental evaluation

This section presents the results and critical analysis of the conducted experimental evaluation, which consisted of experiments looking for (E1) the optimal hyperparameterization of the SDDE method and the assessment of its effectiveness in the comparison with *state-of-art* methods in the context of data streams build on (*E2*) synthetic and (*E3*) based on real concepts.

5.1. Experiment 1

The first experiment aimed to select the values of the SDDE method hyperparameters for further processing, maximizing the metrics proposed in Section 3.2 for separated, overview data streams.

Figs. 3 and 4 show the context of optimization constituting a visualization that allows the analysis of the influence of hyperparameters on the values of quality assessment metrics. For the selected *size of the subspace*, it presents visualizations of heat maps showing the dependence of the six assessment criteria in the two-dimensional function of (a) the number of models in the detector ensemble pool and (b) the sensitivity of their integration function.

Fig. 3 shows the three base assessment metrics defined in Section 3.2: $D1$ – closest drift, $D2$ – closest detection and R – the ratio between the number of detections and drifts. It should be noted that while R is the dominant factor here, clearly informing about

[2] https://github.com/w4k2/ip_stream_generator/blob/master/generator.py.

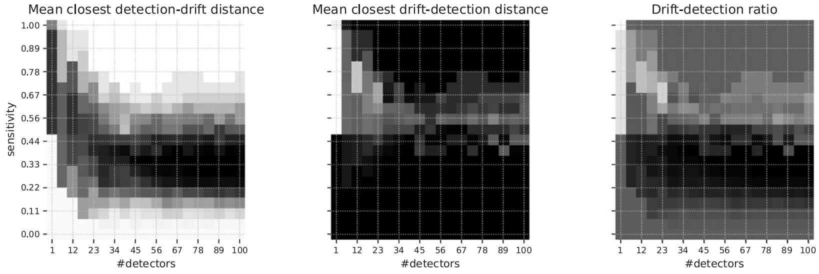


Fig. 3. Exemplary optimization results of proposed method. Number of detectors and sensitivity influence on model's drift detection errors.

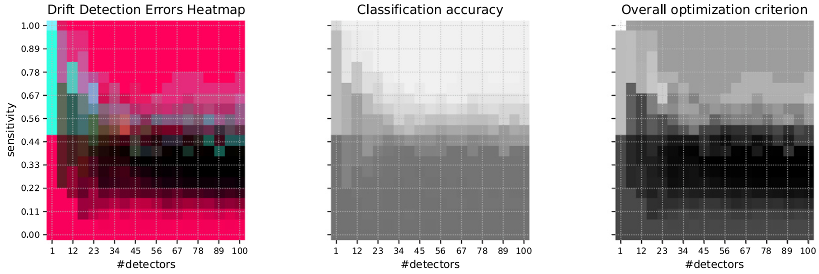


Fig. 4. Exemplary optimization results of proposed method. Aggregated drift detection errors (left), classification accuracy (center) as well as overall optimization criterion (right) consisting of combination of classification accuracy and drift detection errors.

possible indifference or hyperactivity of the detector ensemble, it is possible to achieve its optimal value by random detections with prior knowledge of the number of drifts in the stream. Therefore, the combined knowledge of many factors is necessary for the appropriate selection of suboptimal hyperparameters.

To enable the calculation of *Drift Detection error* measures in cases with the absence of drift detection for a given configuration, the detection was assumed in each data portion. In a detection task, an algorithm that is unable to detect drift carries as little information about the data as a method that detects a change of concept in each batch of data.

The combination of these factors is presented in Fig. 4. Its first cell is a color combination of RGB channels defined by the base metrics, where white represents the globally worst configuration and the most saturated colors magenta and cyan — the worst configurations according to the D_1 and D_2 criteria. Similarly, the region of the best and statistically dependent on the best configurations is represented by the black area in the illustration. The second heatmap shows the classification accuracy heat map, which is most often the primary criterion for evaluating recognition models. The last heatmap of the Figure shows the averaging of the integrated base metrics with the classification quality, constituting the basic tool for selecting hyperparameters used in this work.

In presenting the above-described visualization tool as an overview context of the drift type and size of the analyzed subspace, it is necessary to start with the assessment of the overall accuracy of models built with a specific detection strategy. The observation of Fig. 5 (left) gives a clear and intuitive observation showing that the ALWAYS rule – rebuilding the model at each batch, identical to high sensitivity SDDE detection – is optimal in terms of recognition quality. From the computational perspective, however, it is the worst and the slowest strategy, so it is necessary to select a method configuration in the bright region of the heat map. At the same time, it should not lead to redundant detection — increasing the time complexity of stream processing. Stabilization of measurements is noticeable here after reaching a sufficiently large number of detectors, the number of which strongly depends on the size of the subspace. The most stable in this context, as confirmed by the observations from [18], seems to be an independent analysis of the features, stabilizing with the number of detectors equal to the number of attributes, narrowing down the search for the optimal value of the sensitivity parameter to the range of 30%–60%.

Responses from the remaining metrics (right side of Fig. 5) confirm the observations about the stability of detection after reaching a certain number of detectors. The most straightforward and most unequivocal interpretation here is the ability to detect in the case of sudden drift, where the ground truth perfectly coincides with the point where the drift occurs, leading to a readable, broad black band representing the optimum of combined D_1, D_2 and R metrics.

In gradual and incremental drift – due to dynamics of changes – the black area is absent, replaced by an area with reduced saturation in the range of 25%–60% sensitivity. Interestingly, there is a much smaller correlation between the D_1 and D_2 metrics in the case of incremental drift, which leads to a clear magenta region which shows that we get good results in one of these metrics. However, we have to reject them due to the unsatisfactory results of the other. This suggests that the optimum in the analysis of metrics other than accuracy should be slightly less than 50% of sensitivity.

The observations made on the separate analysis of the accuracy and the proposed metrics seem to be confirmed in the case of their combination visible in Fig. 6. Between the black area of low-quality classification and the gray area of the average quality of detection, there is a narrow area of increased intensity of the heat map, in which we should look for the optimum sensitivity hyperparameter. However, strong disturbances in the read value are also clearly visible here, which may indicate that it is not possible to select one global sensitivity value, which is the optimal SDDE configuration. Such observation suggests that this parameter depends on the specific problem and should be selected individually for each data stream.

The selected basic driver in the optimization of hyperparameters for further processing was the minimization of the identified R parameter corresponding to the ratio between the number of drifts and detections while maintaining a low value of the auxiliary metrics $D1$ and $D2$, responsible for the mutual distances between the drifts and detections. A review of Fig. 7, showing

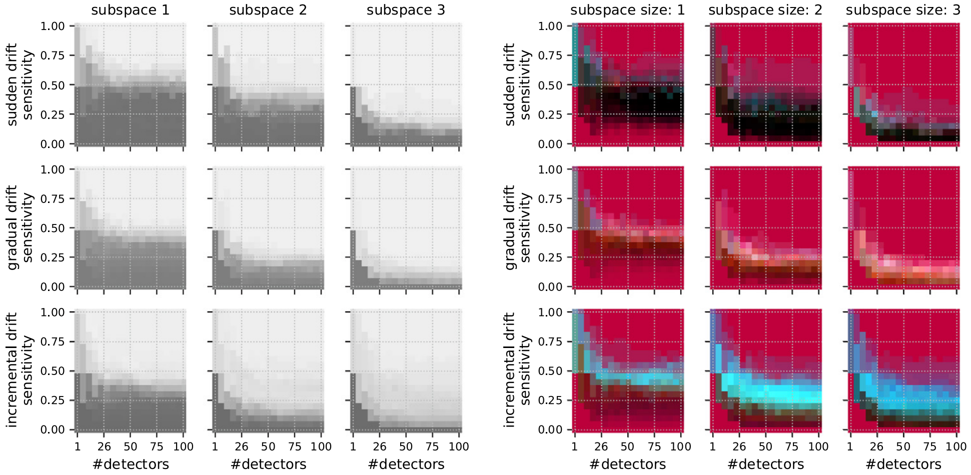


Fig. 5. Classification accuracy (left) and drift detection errors (right) for tested subspace sizes and drift types.

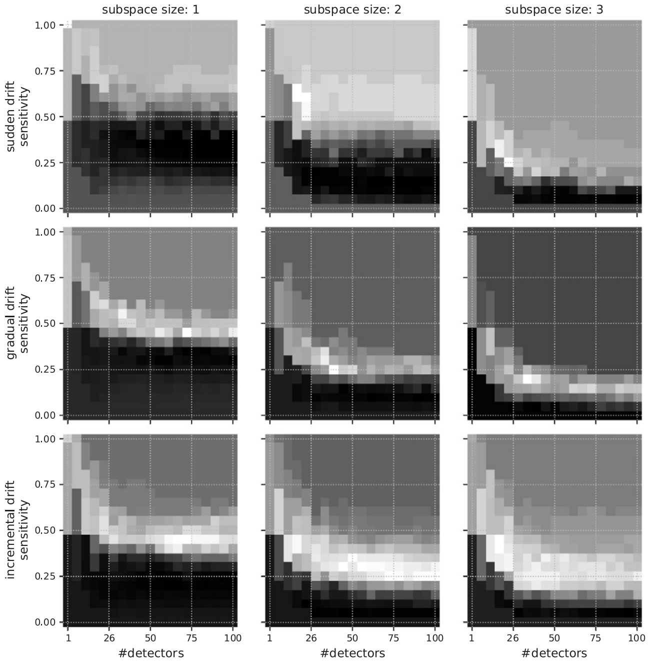


Fig. 6. Overall optimization of hyperparameters for tested subspace sizes and drift types.

the cumulative mean of the metrics as a function of detector sensitivity, made it possible to decide on a sensitivity value of 45% in the case of independent analysis of features. However, it should be noted that this choice was made on the example of synthetic streams and applies only to Experiment 2. In the case of Experiment 3, an additional calibration was made, which allowed for the selection of sensitivity of 35% as better suited to the characteristics of real concepts.

The rationale for such decisions may be found in Figs. 8 and 9, which show a time course of SDDE detections for different sensitivity values between 30%–60% in averaging ten replications. It is observable here that the value of the sensitivity parameter for synthetic streams giving almost perfect results for sudden drift (50%) is not appropriate for the other types of drift. A high value of this parameter leads to desensitization of the detector to drift and too low — to redundant detections. Therefore, the selected values of this hyperparameter represent a compromise that maximizes the ability to recognize concept changes over different classes of drift.

For streams based on real-world concepts, the conclusions remain similar. However, the optimal values of the parameter change. The *nearest* drifts, equivalent to *sudden* drifts, give satisfactory results with the tested parameter values below 45%. For *cubic* drifts, corresponding to *incremental* changes of the concept, the optimal value of the parameter is smaller, around 30 to 35%. Additionally, differences between specific data streams can be noticed. Depending on the static datasets based on which the streaming data was generated, the confidence of drift detection varies, i.e. the same sensitivity parameter value shows the different number of detections on different streams.

5.2. Experiment 2

The second experiment aimed to conduct a comprehensive overview analysis of drift detectors' ability to raise concept change alerts using examples of a variety of fully synthetic data streams. While – as is often mentioned in reviews of works in the field of data stream processing – the use of completely synthetic data is not a good strategy for the final evaluation of recognition methods and a sufficient contribution to their recommendation or rejection from practical applications, this approach allows for a clear revision of the characteristics behavior of recognition algorithms. Only in such an environment of problems is it possible to replicate streams with identical properties. The achieved results no longer depend on the difficulties of the concepts described by the data and present a possibly objectified comparison of the analyzed methods. Therefore, the results of the second experiment are presented in the paper by visualizing the location of alerts in the course of streams differing in (*i*) the number of attributes, (*ii*) drifts, and (*iii*) the type of drift of the concept.

Fig. 10 presents an exemplary visualization of experiment results. Each horizontal block shows drift detections given by consecutive methods over time, indicated by concept drifts marked on *X* axis. Successive rows within blocks corresponding to algorithms mean successive replications of streams with given characteristics, distinguishing accidental warnings from clearly visible trends. The detections of the SDDE algorithm, proposed for this paper, are marked by a yellow row. Each of the chunks in which the given detector indicated the occurrence of drift is marked with a black point in the illustration.

Fig. 11 (on the left side) shows the drift alarms for non-recurring concepts. The first visible observation in the case of this analysis is the apparent hyperactivity of the EDDM detector, leading to redundant alerts present on the entire course of

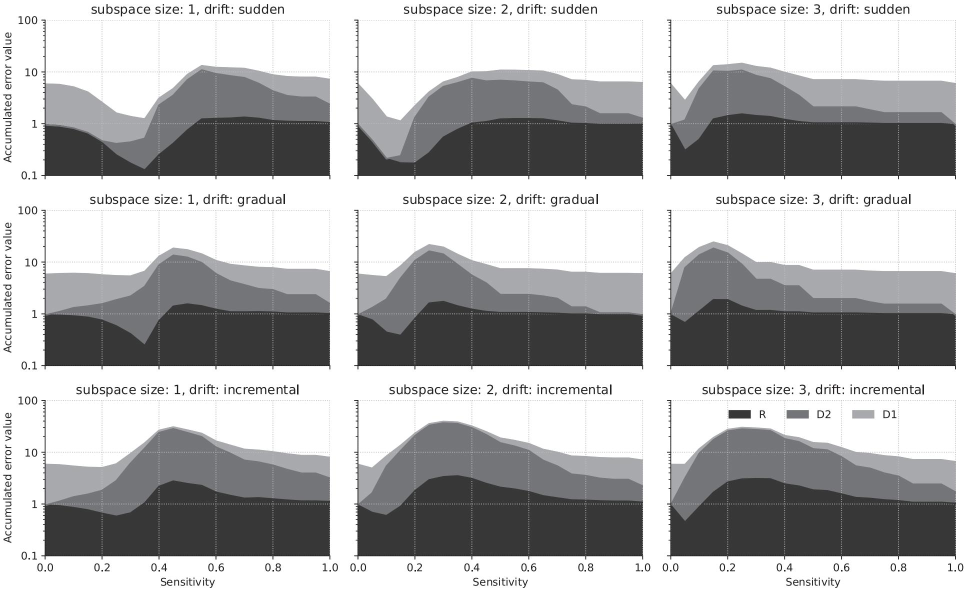


Fig. 7. Mean accumulated Drift Detection Errors (drift detection ratio, closest detection, closest drift) values for all tested number of detectors. In columns – subspace size, in rows – drift type.

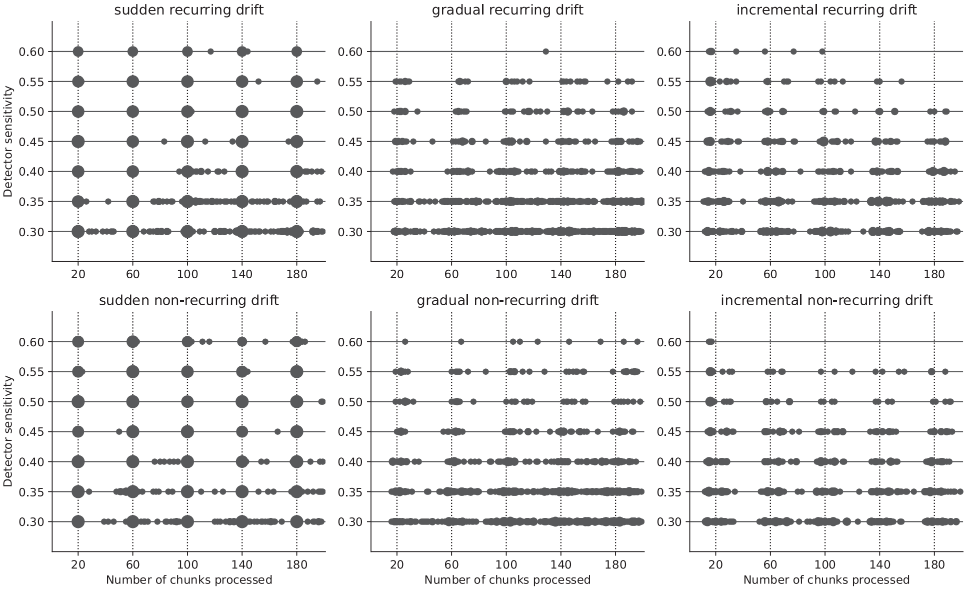


Fig. 8. Method sensitivity impact on detections number in 10 replications for synthetic streams.

the streams. There is indeed a decrease in their density at the moments of stable concepts (evident with sudden drift), but it is still a method that definitely too often indicates a change of concept. The DDM method behaves similarly, especially in the initial phase of streams. In many cases, it either quickly becomes desensitized to changes, ceasing to be useful in recognizing, or leads to continuous alerts that begin with the emergence of a new concept and finish little before the next one appears.

Interestingly, such an approach allows for a seemingly high-quality classification, but a similar one would be achieved in many cases by the abstract ALWAYS detector. Based on the conducted analysis, it seems that the wrong approach – often indicating the high usefulness of EDDM and DDM methods – is to base these conclusions on the high overall quality of recognition. It may result mainly from a frequent reconstruction of the recognition model, which in the presented strategy of updating models in chunks in which no detection takes place, allows them to gain an advantage in the overall quality of recognition, but at an apparent cost of the high time complexity.

The results for the ADWIN and SDDE methods present much better — in the case of sudden drifts, the detections often cover the line of occurrence of the drift almost perfectly. In the case of gradual drifts, they also behave quite similarly – however – spreading the detection points wider over the entire course of the ongoing change. It is sometimes alerting several times then, highlighting the transition phases of the drift and, at the same time, allowing for detection adequately earlier than the central drift point. Apparent differences between SDDE and ADWIN appear only in the case of incremental drifts, where the standard deviation of the detection distance from the drift is higher for SDDE while maintaining a uniform distribution around the central drift point, which can be interpreted as a higher ability of the proposition presented in this work to signal a drift early.

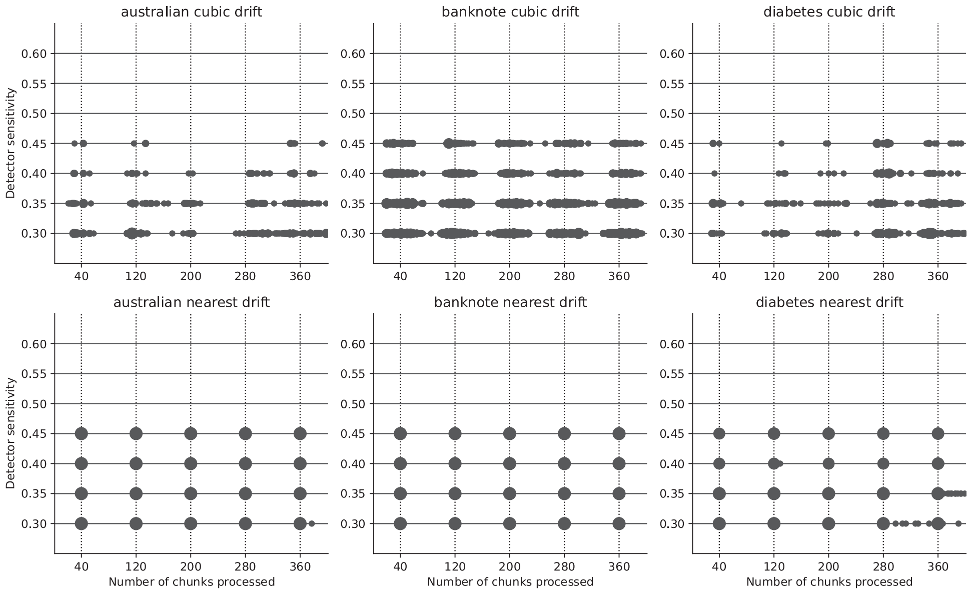


Fig. 9. Method's sensitivity impact on detections number in 10 replications for streams based on the real-world concepts.

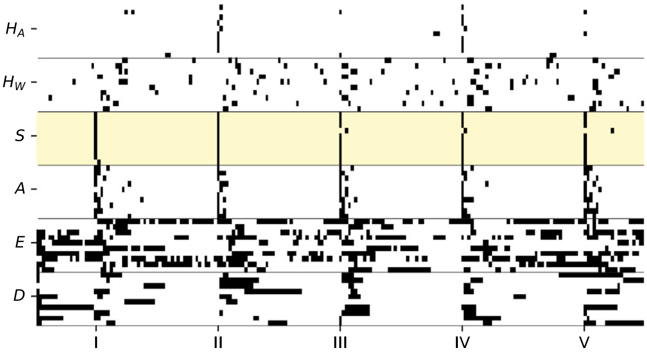


Fig. 10. Exemplary result of Experiment 2 — detection moments of evaluated methods for 10 replications of stream with 5 concept drifts.

The HDDM_A method recognizes drifts only in a few of the replications, especially in the case of streams characterized by three drifts. Additionally, in case of recurring sudden concept changes, only some of the actual ones are signaled. Generally, the method presents low sensitivity to concept drifts. The HDDM_W detector, however, is showing frequent detections, which rarely lay in the exact moment of drift but are rather signaled after a certain number of chunks describing a new concept. The results for recurrent drifts, included in Fig. 11 (on the right side), are similar. The observations seem constant, regardless of the problem's dimensionality and the number of drifts.

5.3. Experiment 3

The final experiment aimed to test the operation of the proposed method and to compare it with other drift detection methods on data streams based on real-world concepts.

The results of the experiment are presented in Figs. 12 and 13. The proposition of this paper is highlighted in a yellow color. A single black point on the plot indicates a concept drift detection in one of the repetitions of the experiment. Each of the replications of analyzed methods (DDM, EDDM, ADWIN, SDDE, HDDM_A, HDDM_W) is presented in the figure.

In every replication, the reference methods mark drifts at the same moments. Since their behavior depends only on the classification quality, the detectors are deterministic. While evaluating one stable stream, the classification quality of the deterministic *Gaussian Naive Bayes classifier* used for evaluation will be as well stable in each replication. Thus drifts will be identified at the same moment.

The proposed SDDE method does not use the classification quality during detection and has an element of randomness — the selection of a subset of the analyzed features. Experiment 1 showed that the algorithm performs best with a *subspace size* parameter set to one. Despite setting the *number of detectors* in ensemble equal to the number of features in the stream, it is not required that each attribute will be used for analysis. Duplicate features may appear as a result of randomization. For the reasons mentioned above, the SDDE method can mark detections in various moments on the same stream, depending on the selected features taken into account during the calculations. The aforementioned situation is also seen in Figs. 12 and 13.

The static datasets, based on which the streams are generated, are characterized by different difficulties in the context of the classification task. It can be noticed that the data difficulty differs in the context of concept drift detection. In the case of the *banknote* and *wisconsin* datasets, the detections of all methods occur more often than in the case of *australian* and *diabetes* datasets.

The results show that the proposed method has the potential to make more accurate detections than the reference methods. Since the drifts of type *nearest*, which is the equivalent of *sudden* type, are occurring at an unequivocal moment, the method detected accurately on almost every set tested and in every

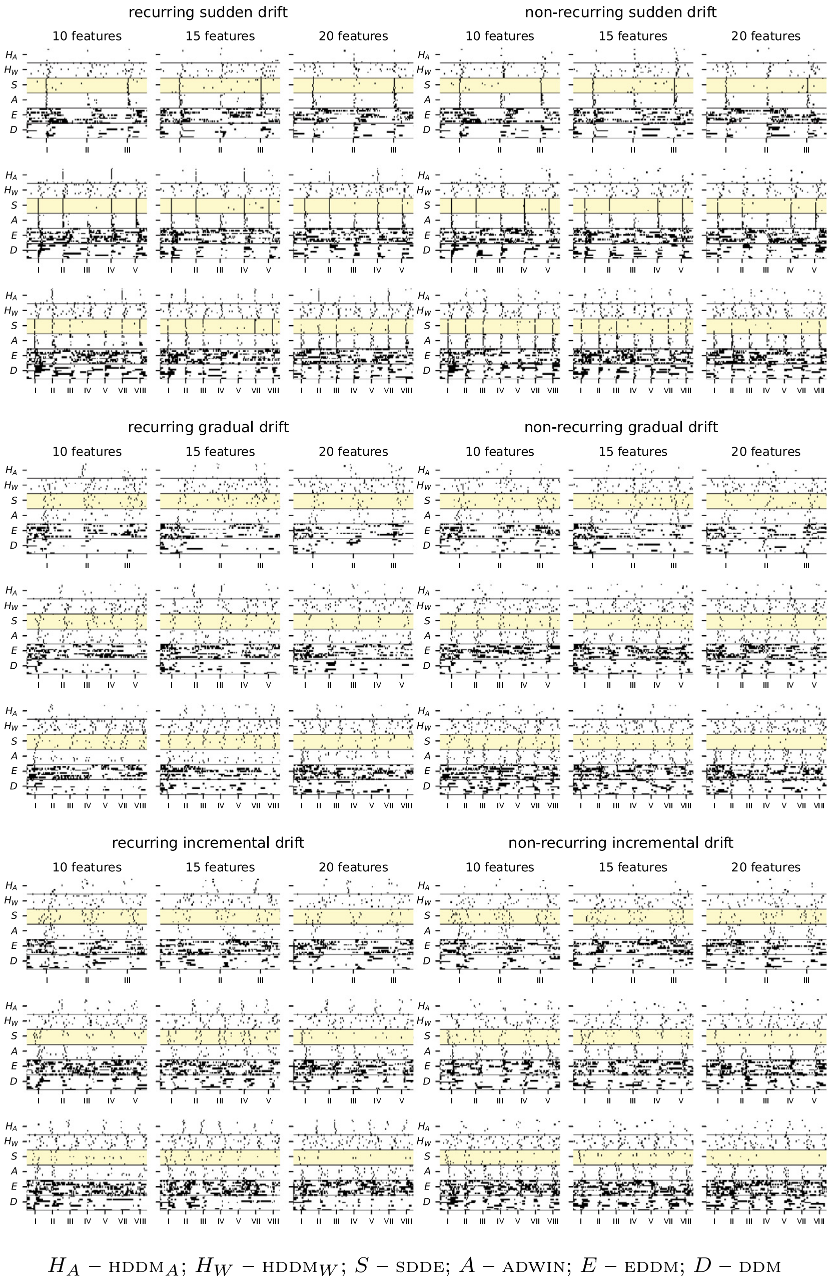


Fig. 11. Drifts detected by evaluated methods on streams with recurring (left) and non-recurring concept drift (right).

replication. Only sporadic false-positive errors appeared. On the other hand, for *cubic* drifts, corresponding to *incremental* type, we cannot determine the precise moment of drift. In the case of these drifts, reference methods often fail. Results for the proposed SDDE detector show a visible accumulation of detections in the vicinity of the drift.

Moreover, one drift can be detected several times, as the concept changes in *incremental* drifts are fluid and long-lasting. The method is also sensitive to the detection of transient concepts between the target ones and is capable of detecting an early phase of incremental drifts. This is, however, an advantageous effect. When the classification model is rebuilt to reduce the loss of recognition quality in standard detector applications, the quality will also be maintained throughout the drift duration. By rebuilding the model once, for instance, at the beginning of the *incremental* drift occurrence, the classifier would also be trained using the transition patterns lying between the two concepts, which could potentially affect classification accuracy.

The results of the comparison are also presented in Tables 2–5. The statistically best performing method was emphasized in the results. The tables are followed by *Critical Difference* diagrams with ranks obtained with Nemenyi post-hoc statistical test.

Classification results (Table 2) present all evaluated detectors and both extreme ALWAYS and NEVER methods. However, the pseudo-detector NEVER was omitted in the comparison of the drift detection quality metrics (Tables 3–5). The lack of detection makes it impossible to calculate the desired distances (D_1, D_2) and as well the proportion of drift number and detection number (R). We can consider that a method not detecting a change of concept is unprofitable and its drift detection errors will be infinite.

In terms of classification quality, the results are similar for all streams except for the streams originating from the *australian* dataset. The best results are achieved by the ADWIN, SDDE and ALWAYS methods. The NEVER method achieves the worst classification accuracy, which is due to the lack of adaptation of the base

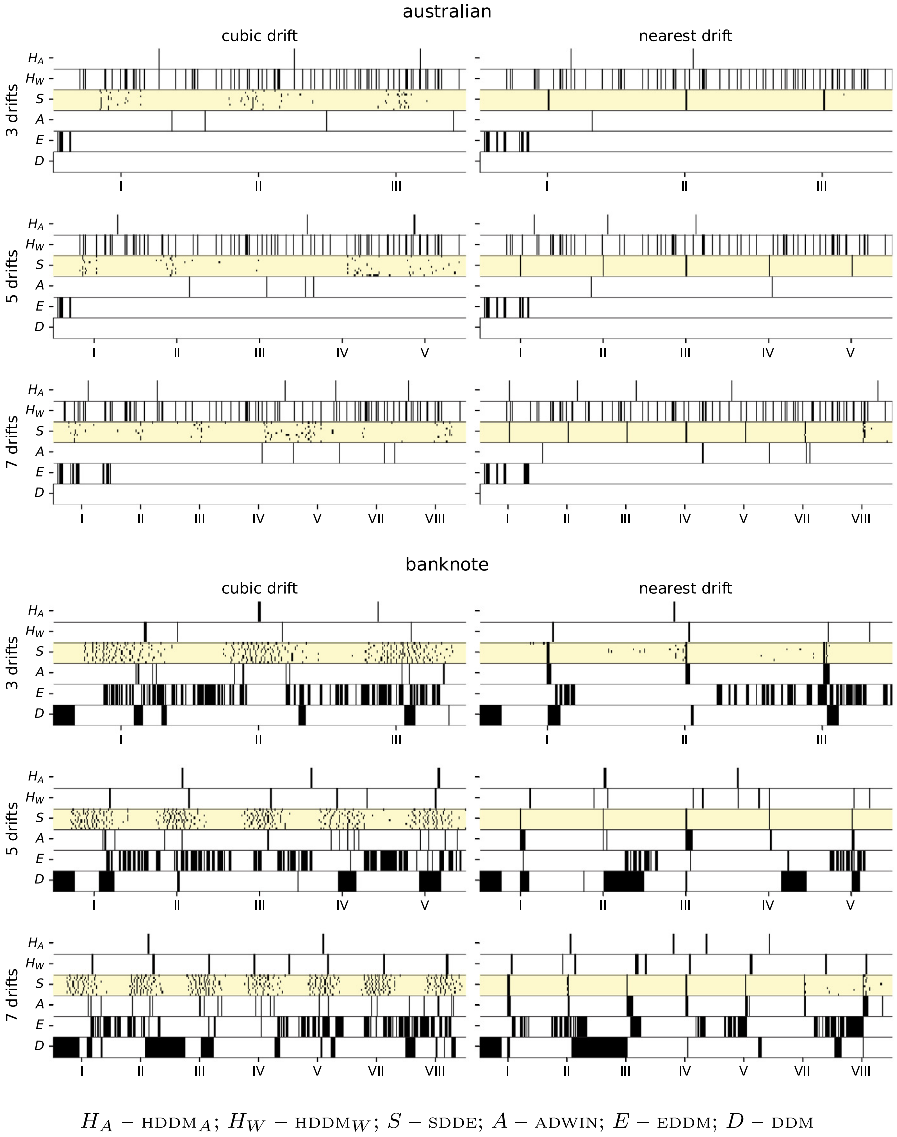


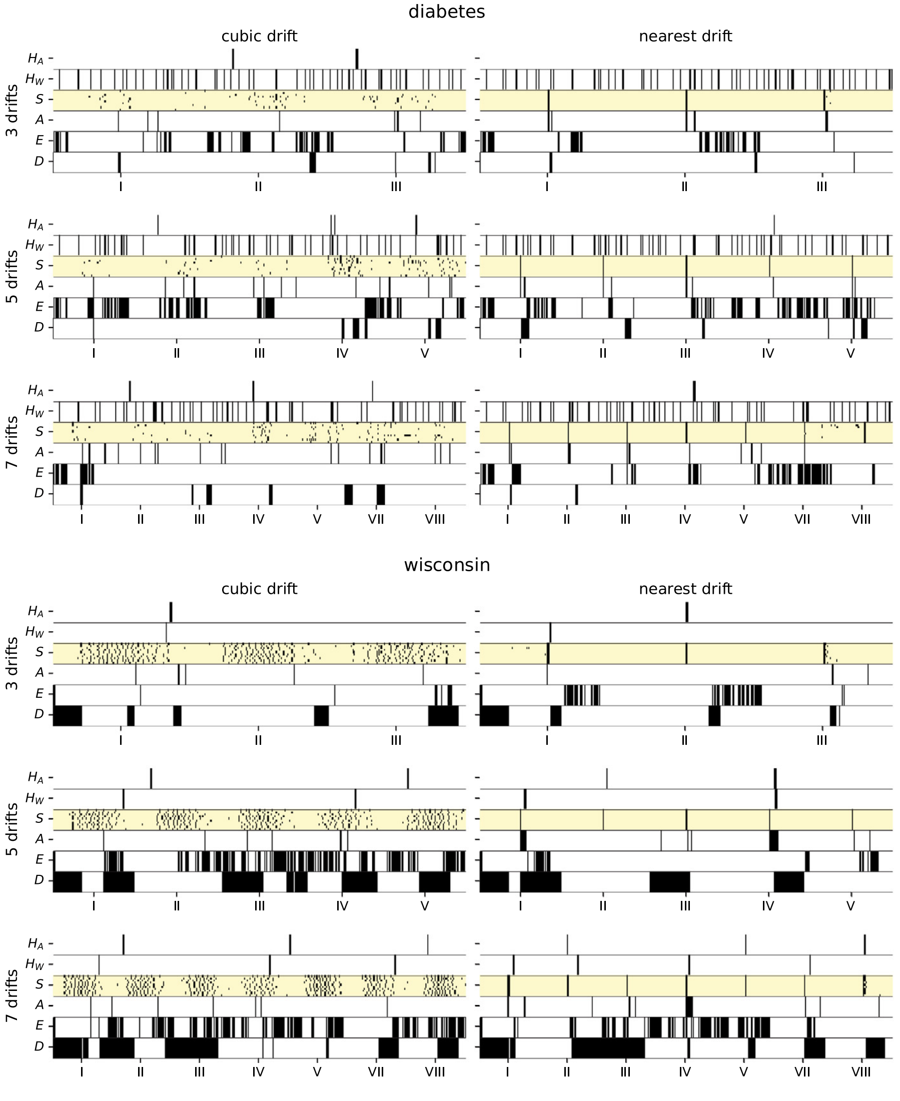
Fig. 12. Moments of detections of evaluated concept drift detection methods on real-concept data streams (*australian* and *banknote*).

classifier to changes in the concept throughout stream flow. The fact that a pseudo-detector often obtains the best results confirms the observations from the publication [26], in which a method that does not analyze data but only deterministically signals a drift every given number of patterns is able to achieve statistically better results than methods dedicated to the analysis of concept changes. Therefore confirms the above-mentioned work's conclusion that classification accuracy measures should not be used to measure drift detection effectiveness.

The *australian* set seems to be particularly difficult in terms of the classification task because the patterns derived from one concept are characterized by much noise. Only for this data stream, the best results are achieved by the detectors EDDM (for *cubic* drift) and DDM (for *nearest* drift). Both detectors, which can be noticed in the first rows of Fig. 12, detected the drifts only at the beginning of the stream analysis.

Table 3 shows the results of the D_1 (*closest drift*) measure, which indicates the average distance of each detection to the nearest drift. For most of the analyzed methods, the error values are higher for *cubic* type of drift changes. The time period of drift occurrence is wider and the exact moment of detection depends on the sensitivity of the method to concept changes. In the case of *nearest* drift types, the SDDE method for streams characterized with 5 drifts achieved ideal results — the error value is exactly zero. Statistically, the best results of this evaluation criterion in most of evaluated streams were achieved by the proposed SDDE method. The worst results were achieved by the DDM, EDDM and ALWAYS methods.

Table 4 shows the results of the D_2 (*closest detection*) measure which describes the average distance of each drift from the closest detection. The ALWAYS pseudo-detector will have zero error value in the context of this evaluation criterion. By signaling the drift in each data chunk, detection will certainly be signaled at the moment of the actual concept drift. Redundant detections are not taken into account in this evaluation measure. As with the measure D_1, the error values are often larger with *cubic* drift. This is for the same reason — the detections will be more spread over time compared to the *nearest* drift. The SDDE method achieved zero error values for the *nearest* type of drift and streams containing 5 concept changes. When disregarding the ALWAYS method from the comparison, SDDE achieves the statistically best results in the case of most of the streams with *nearest* drift. In the case of *cubic* drift, in the data stream originating from the *diabetes* dataset, the method does not perform well, and the best results are presented by HDDM$_W$ method.



H_A – HDDM$_A$; H_W – HDDM$_W$; S – SDDE; A – ADWIN; E – EDDM; D – DDM

Fig. 13. Moments of detections of evaluated concept drift detection methods on real-concept data streams (*diabetes* and *wisconsin*).

Table 2
Classification accuracy of data stream processing.

		CUBIC								NEAREST							
		D	*E*	*A*	*S*	H_W	H_A	*a*	*n*	*D*	*E*	*A*	*S*	H_W	H_A	*a*	*n*
		1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
AUSTRALIAN	3	.613 3,5:8	**.639** *all*	.598 5:8	.611 3,5:8	.590 7	.596 5,7:8	.588 –	.594 5,7	.621 3:8	**.624** *all*	.620 4:8	.585 –	.590 4,7	.604 4:5,7	.589 4	.615 4:7
	5	.617 3,5:8	**.649** *all*	.608 5:8	.618 3,5:8	.589 7:8	.603 5,7:8	.589 8	.573 –	**.619** *all*	.610 4:8	.616 2,4:8	.587 –	.592 4,7	.598 4:5,7	.589 4	.607 4:7
	7	.622 3:8	**.631** *all*	.596 5,7:8	.612 3,5:8	.595 7:8	.599 3,5,7:8	.589 8	.587 –	**.620** *all*	.617 3:8	.603 4:7	.589 –	.590 4,7	.599 4:5,7	.589 –	.611 3:7
BANKNOTE	3	.859 6,8	.880 1,5:6,8	**.890** *all*	.880 1,5:6,8	.868 1,6,8	.653 8	.883 1,2,4:6,8	.490 –	.856 2,5:6,8	.830 6,8	**.875** *all*	.869 1:2,5:8	.845 2,6,8	.542 8	.859 1:2,5:6,8	.471 –
	5	.853 6,8	.861 1,5:6,8	**.875** *all*	.866 1:2,5:6,8	.859 1,6,8	.648 8	.868 1:2,5:6,8	.479 –	.825 2,5:6,8	.799 6,8	**.850** *all*	.839 1:2,5:6,8	.804 2,6,8	.601 8	.841 1:2,4:6,8	.475 –
	7	.810 6,8	.861 1,5:6,8	**.876** *all*	.871 1:2,5:6,8	.822 1,6,8	.592 8	.875 1:2,4:6,8	.457 –	.787 6,8	.819 1,5:6,8	.843 1:2,5:6,8	.848 1:3,5:6,8	.809 1,6,8	.615 8	**.854** *all*	.465 –

(*continued on next page*)

As the number of drifts in the stream increases, the maximum error values of the D_1 and D_2 measures decrease. The mentioned measures are based on the distance between the actual drift and the detection. In case drifts occur more frequently, the reduction of the distance between detection and drift is a consequence. Therefore, drift detection methods should be compared under the same conditions, i.e., the same number and location of drifts in the stream.

Table 2 (*continued*).

		CUBIC								NEAREST							
		D	E	A	S	H_W	H_A	a	n	D	E	A	S	H_W	H_A	a	n
		1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
DIABETES	3	.594 6,8	.597 1,6:8	**.606** *all*	.592 6,8	.599 1:2,6:8	.553 8	.596 1,6,8	.550 –	.571 6,8	.594 1,3,5:6,8	.589 1,5:6,8	**.602** *all*	.588 1,6,8	.531 –	.595 1:3,5:6,8	.531 –
	5	.578 6,8	**.603** *all*	.600 1,4:8	.582 6,8	.595 1,6,8	.556 8	.597 1,4:6,8	.547 –	.586 6,8	.592 1,3,5:6,8	.591 1,6,8	.597 1:3,5:6,8	.592 1,3,6,8	.543 8	**.597** *all*	.528 –
	7	.580 2,6,8	.579 6,8	.584 1:2,4,6,8	.576 6,8	.595 1:4,6,8	.547 8	**.596** *all*	.541 –	.541 6,8	.595 1,3:6,8	.593 1,4:6,8	.588 1,6,8	.592 1,4,6,8	.529 8	**.597** *all*	.521 –
WISCONSIN	3	.959 2:3,5:6,8	.956 5:6,8	.957 2,5:6,8	**.960** 1:3,5:6,8	.948 6,8	.935 8	**.960** 1:3,5:6,8	.856 –	.954 2,6,8	.935 6,8	.959 1:2,5:6,8	**.961** *all*	.957 1:2,6,8	.837 8	.960 1:3,5:6,8	.796 –
	5	.955 5:6,8	.957 1,3,5:6,8	.956 1,5:6	.960 1:3,5:6,8	.943 6,8	.906 8	**.961** *all*	.842 –	.951 2,5:6,8	.932 6,8	.958 1:2,5:8	**.958** *all*	.945 2,6,8	.863 8	.958 1:2,5:6,8	.786 –
	7	.954 3,5:6,8	.955 1,3,5:6,8	.952 5:6,8	**.959** 1:3,5:6,8	.941 6,8	.896 8	**.959** 1:3,5:6,8	.868 –	.955 2,5:6,8	.948 5:6,8	.956 1:2,5:6,8	**.957** 1:3,5:6,8	.932 6,8	.846 8	**.957** 1:3,5:6,8	.820 –

D – DDM; E – EDDM; A – ADWIN; S – SDDE; H_W – HDDM$_W$; H_A – HDDM$_A$; a – ALWAYS; n – NEVER

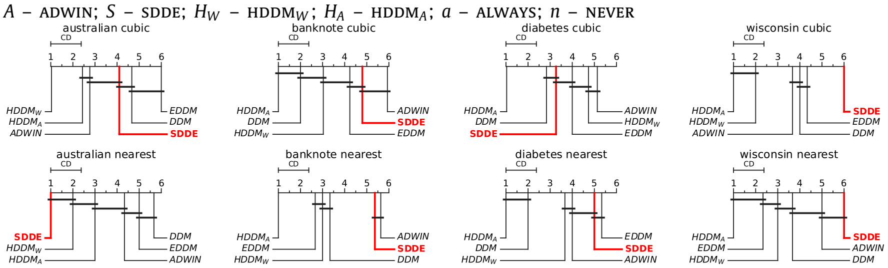
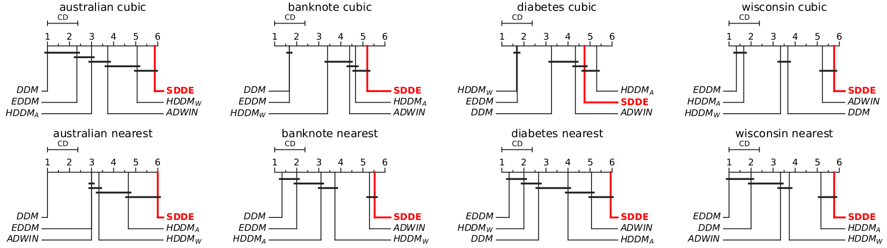


Table 3
Closest drift (D_1) error of drift detection (scaled by 10^{-2}).

		CUBIC							NEAREST						
		D	E	A	S	H_W	H_A	a	D	E	A	S	H_W	H_A	a
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
AUSTRALIAN	3	.650 –	.554 1	.553 1:2	**.123** *all*	.288 1:3,6:7	.313 1:3,7	.333 1:3	.650 –	.393 1,3	.430 1	**.015** *all*	.297 1:3,7	.155 1:3,5,7	.333 1:3
	5	.390 –	.294 1	.208 1:2,6	**.126** *all*	.197 1:3,6:7	.227 1:2	.200 1:3,6	.390 –	.170 1,5,7	.075 1:2,5:7	**.000** *all*	.207 1	.087 1:2,5,7	.200 1,5
	7	.003 –	.002 1,6	.144 1:2,6	**.094** *all*	.130 1:3,6:7	.184 1	.142 1:3,6	.270 –	.147 1,3	.148 1	**.016** *all*	.133 1:3,7	.096 1:3,5,7	.142 1:3
BANKNOTE	3	.375 –	.314 1,7	.250 1:2,5,7	.178 1:3,5,7	.278 1:2,7	**.063** *all*	.333 1	.285 2,7	.310 7	**.026** 1:2,5:7	.074 1:2,7	.113 1:2,7	.105 1:2,5,7	.333 –
	5	.135 2,6:7	.247 –	**.111** 1:2,5:7	**.109** 1:2,5:7	.116 1:2,6:7	.162 2,7	.200 2	.199 7	.191 1,7	.021 1:2,5:7	**.000** *all*	.104 1:2,6:7	.160 1:2,7	.200 –
	7	.200 –	.140 1,7	.095 1:2,5,7	.078 1:3,5,7	.114 1:2,7	**.065** *all*	.143 1	.155 –	.116 1,6:7	.031 1:2,5:7	**.019** *all*	.087 1:2,6:7	.0136 1,7	.143 1
DIABETES	3	.355 2	.423 –	**.157** 1:2,5:7	**.208** 1:2,5:7	.351 1:2	.315 1:2,5,7	.333 1:2,5	.374 2	.390 –	.034 1:2,5:7	**.014** *all*	.340 1:2	.332 1:2,5	.333 1:2,5
	5	**.114** 2:3,5,7	.201 5	.169 2,5,7	.122 2:3,5,7	.204 –	**.112** 1:3,5,7	.200 2,5	.132 2,5,7	.204 –	.028 1:2,5:7	**.000** *all*	.184 2,7	.050 1:2,5,7	.200 2
	7	0.124 2,5,7	.129 5,7	.111 1:2,5,7	.090 1:3,5,7	.151 –	**.063** *all*	.143 5	.098 2,5,7	.130 5,7	**.038** 1:2,5:7	**.026** 1:2,5:7	.136 7	.085 1:2,5,7	.143 –
WISCONSIN	3	.477 2,6	.488 –	.386 1:2,5:6	**.201** *all*	.440 1:2,6	.485 2	.333 1:3,5:6	.335 2	.429 –	.173 1:2,7	.025 1:3,7	.025 1:3,7	**.015** 1:3,5,7	.333 1:2
	5	.200 2,5:6	.226 –	**.113** 1:2,5:7	**.116** 1:2,5:7	.205 2,6	.210 2	.200 1:2,5:6	.197 2,7	.222 –	.050 1:2,5,7	**.000** *all*	.055 1:2,7	.047 1:3,5,7	.200 2
	7	.137 2,5:7	.174 6	.116 1:2,5:7	**.080** *all*	.146 2,6	.188 –	.143 2,5:6	.122 2,7	.150 –	.076 1:2,7	**.011** 1:3,5,7	.064 1:3,7	**.010** 1:3,5,7	.143 2

D – DDM; E – EDDM; A – ADWIN; S – SDDE; H_W – HDDM$_W$; H_A – HDDM$_A$; a – ALWAYS

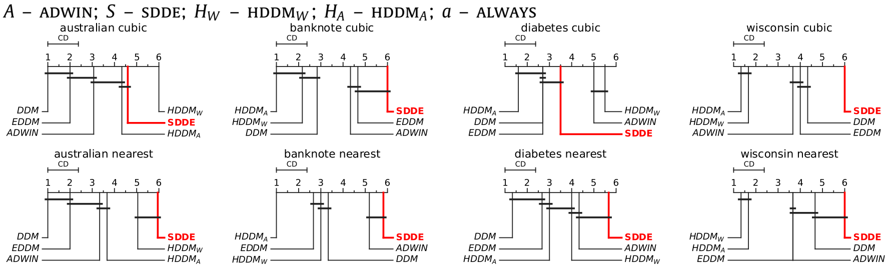


The values of the last analyzed criterion of the detection quality – R (*drift detection ratio*) – are presented in Table 5. The measure is dependent on the proportion between the number of drifts and the number of detections. The errors resulting from excessive alerts will be smaller than errors in the case of too few detections. It can be noticed, especially in the case of the

Table 4
Closest detection (D_2) error of drift detection (scaled by 10^{-3}).

		CUBIC							NEAREST						
		D	E	A	S	H_W	H_A	a	D	E	A	S	H_W	H_A	a
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
AUSTRALIAN	3	.198 –	.181 1	.052 1:2	.015 1:3,6	.000 1:3,6	.031 1:3	**.000** 1:3,5:6	.198 –	.151 1	.119 1:2	.001 1:3,5:6	.003 1:3,6	.052 1:3	**.000** *all*
	5	.199 –	.182 1	.049 1:2	.027 1:3	.002 1:4,6	.034 1:3	**.000** *all*	.199 –	.154 1	.049 1:2,6	**.000** 1:3,5:6	.003 1:3,6	.050 1,2	**.000** 1:3,5:6
	7	.198 –	.147 1	.060 1:2	.020 1:3	.002 1:4,6	.023 1:3	**.000** *all*	.198 –	.154 1	.032 1:2	.001 1:3,5:6	.001 1:3,6	.023 1:3	**.000** *all*
BANKNOTE	3	.020 6	.004 1,3,5:6	.012 1,5:6	.001 1:3,5:6	.020 6	.050 –	**.000** *all*	.004 2,5:6	.013 6	.000 1,2,5:6	.000 1:2,5:6	.004 2,6	.092 –	**.000** *all*
	5	.009 5:6	.004 1,3,5:6	.006 1,5:6	.001 1:3,5:6	.010 6	.036 –	**.000** *all*	.003 2,5:6	.009 6	.000 1:2,5:6	**.000** 1:3,5:6	.004 2,6	.054 –	**.000** 1:3,5:6
	7	.008 5:6	.003 1,5:6	.003 1:2,5:6	.001 1:3,5:6	.009 6	.049 –	**.000** *all*	.007 6	.004 1,5:6	.000 1:2,5:6	.000 1:3,5:6	.007 1,6	.038 –	**.000** *all*
DIABETES	3	.017 6	.012 1,6	.008 1:2,4,6	.031 6	.002 1:4,6	.057 –	**.000** *all*	.033 –	.011 1	.001 1:2,5	.001 1:3,5	.003 1:2	**.000** 1:5	**.000** 1:5
	5	.032 6	.008 1,6	.005 1:2,4:6	.027 –	.006 1:2,4,6	.033 –	**.000** *all*	.019 6	.004 1,6	.000 1:2,5:6	**.000** 1:3,5:6	.003 1:2,6	.115 –	**.000** 1:3,5:6
	7	.020 2,6	.161 –	.007 1:2,4,6	.018 2,6	.005 1:4,6	.033 2	**.000** *all*	.116 –	.010 1,6	.010 1:2,6	.001 1:3,5:6	.003 1:3,6	.098 1	**.000** *all*
WISCONSIN	3	.031 2,5:6	.043 5:6	.025 1:2,5:6	.002 1:3,5:6	.118 –	.116 5	**.000** *all*	.011 2:3,5:6	.020 3,5:6	.047 5:6	.000 1:3,5:6	.132 –	.089 5	**.000** *all*
	5	.010 3,5:6	.004 1,3,5:6	.025 5:6	.001 1:3,5:6	.050 6	.052 –	**.000** *all*	.019 2,5:6	.043 5:6	.012 1:2,5:6	**.000** 1:3,5:6	.049 –	.048 5	**.000** 1:3,5:6
	7	.003 2:3,5:6	.005 3,5:6	.022 5:6	.000 1:3,5:6	.034 6	.035 –	**.000** *all*	.002 2:3,5:6	.009 3,5:6	.014 5:6	.000 1:3,5:6	.025 6	.033 –	**.000** *all*

D – DDM; E – EDDM; A – ADWIN; S – SDDE; H_W – HDDM$_W$; H_A – HDDM$_A$; a – ALWAYS



stream originating from the *australian* dataset, that the methods marking too little detections – such as DDM, ADWIN and HDDM$_A$ in the case of *nearest* drift – achieve greater errors than the ALWAYS method. For other data streams in the case of *cubic* drifts, ADWIN and HDDM$_A$ achieve the statistically best results. As can be seen in Fig. 13, the proposed SDDE method repeatedly signals a single change of concept during a *cubic* drift. Multiple detections separated in time emphasize the method's sensitivity to concept changes and may have beneficial effects. However, this measure of detector evaluation increases the method's error in the context of this measure of detector evaluation. In the case of *nearest* drifts, the issue does not occur, and the proposed SDDE detector has a statistical advantage over other methods.

In summary, the proposed SDDE detector achieves satisfactory classification accuracy and *drift detection error* values for the evaluated data streams. The detector is particularly effective in detecting sudden drifts and has high sensitivity in detection of incremental drifts.

6. Conclusions and future works

This publication proposes a *Statistical Drift Detection Ensemble* (SDDE) which is a novel method for concept drift detection in evolving data streams. The method was designed for general data stream processing and applications with no or delayed information about the classification quality. The proposed detector and other *state-of-the-art* detectors known from the literature were implemented in *Python* programming language and evaluated on synthetic data streams and as well generated streams based on real-world concepts.

Reference *pseudo-detectors*, either marking detectors in each data batch or not detecting any drifts, also was a part of the evaluation. To evaluate the operation of the detectors, three metrics for assessing the quality of drift detection in data streams with available drift occurrence ground-truth were proposed. The method parameters were reviewed, and the optimal hyperparameters were selected for further evaluation.

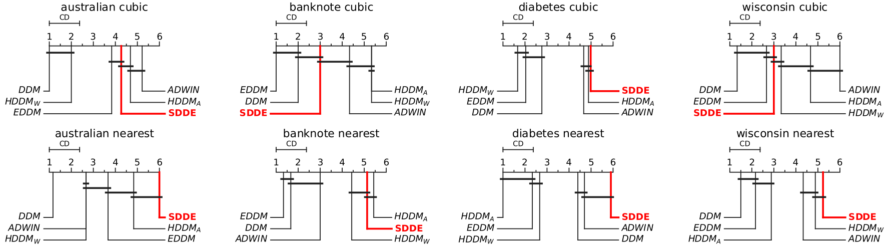
The comparison of the proposed method, *state-of-the-art* detectors, and reference extreme *pseudo-detectors* using both presented metrics and the classification quality, supported by the performed statistical tests, prove the advantage of the proposed method over other detectors. Research revealed the benefits of using metrics other than classification quality to improve drift detection quality measurement.

Future works will focus on the improvements of the SDDE method and further research under more challenging experimental conditions. The method may be equipped with an automatic mechanism of adjusting the *sensitivity* parameter, optimal for the specific data being analyzed. This parameter is critical since it greatly influences the number of detections, impacting the detection quality. Worth analyzing is also the potential of SDDE to detect concept drift occurrence in imbalanced data streams [51]. The problem of concept drift in the case of multi-label streams, which is relatively rarely tackled in the literature, is another potentially captivating research direction [52].

Table 5
Drift Detection ratio (R) error (scaled by 10^{-1}).

		CUBIC							NEAREST						
		D	E	A	S	H_W	H_A	a	D	E	A	S	H_W	H_A	a
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
AUSTRALIAN	3	.200 –	.057 1,5,7	.025 1:2,4:5,7	.060 1,5,7	.096 1,7	**.000** *all*	.099 1	.200 –	.082 1,3,5,7	.200 –	**.003** *all*	.095 1,3,7	.050 1:3,5,7	.099 1,3
	5	.400 –	.029 1,5:7	**.025** *all*	.044 1,5:7	.092 1,7	.067 1,5,7	.099 1	.400 –	.069 1,3,5,7	.150 1	**.000** *all*	.092 1,3,7	.067 1,2:3,5,7	.099 1,3
	7	.600 –	.058 1,5,7	**.040** 1:2,5,7	**.034** 1:2,5,7	.090 1,7	**.040** 1:2,5,7	.098 1	.600 –	.053 1,5,7	.040 1:2,5,7	**.005** *all*	.090 1,7	.040 1:2,5,7	.098 1
BANKNOTE	3	.094 2,7	.098 7	.075 1,2,4,7	.091 1,2,7	.040 1:4,7	**.000** *all*	.099 –	.094 2,7	.097 7	.079 1:2,7	.059 1:3,7	**.050** 1:4,7	**.050** 1:4,7	.099 –
	5	.094 2,7	.097 7	.067 1:2,4,7	.087 1:2,7	.055 1:4,7	**.017** *all*	.099 –	.095 7	.090 1,7	.074 1:2,7	**.000** *all*	.055 1:3,7	.025 1:3,5,7	.099 –
	7	.094 2,7	.095 7	.071 1:2,4,6:7	.084 1:2,7	**.056** *all*	.075 1:2,4,7	.098 –	.092 2,7	.094 7	.072 1:2,7	.032 1:3,5,7	.059 1:3,7	**.000** *all*	.098 –
DIABETES	3	.080 2,5,7	.096 7	.057 1:2,5,7	.046 1:2,5,7	.094 2,7	**.025** 1:3,5,7	.099 –	.063 2,5:7	.094 5:7	.057 1:2,5:7	**.008** *all*	.095 6:7	.099 –	.099 –
	5	.075 2,5,7	.095 7	.064 1:2,5,7	.049 1:2,5,7	.090 2,7	**.000** *all*	.099 –	.081 2,5:7	.095 6:7	.044 1:2,5:7	**.000** *all*	.091 2,6:7	.400 –	.099 6
	7	.077 5,7	.068 1,5:7	.061 1:2,5:7	**.043** *all*	.087 7	.075 1,5,7	.098 –	**.017** 2:3,5:7	.092 6:7	.042 2,5:7	**.010** 2:3,5:7	.088 2,6:7	.250 –	.098 6
WISCONSIN	3	.097 5,7	.079 1,4:5,7	**.040** *all*	.093 1,5,7	.200 –	.050 1:2,4:5,7	.099 5	.095 7	.095 7	**.000** *all*	.043 1:2,5:7	.050 1:2,7	.050 1:2,7	.099 –
	5	.097 7	.097 1,7	**.017** *all*	.088 1:2,7	.025 1:2,4,7	.025 1:2,4,7	.099 –	.096 7	.085 1,7	.075 1:2,7	**.000** *all*	.025 1:3,6:7	.067 1:3,7	.099 –
	7	.096 7	.096 1,7	**.000** *all*	.086 1:2,7	.040 1:2,4,7	.040 1:2,4,7	.098 –	.096 6:7	.093 1,6:7	.053 1:2,6:7	.030 1:3,6:7	**.000** *all*	.133 –	.098 6

D – DDM; E – EDDM; A – ADWIN; S – SDDE; H_W – HDDM$_W$; H_A – HDDM$_A$; a – ALWAYS



CRediT authorship contribution statement

Joanna Komorniczak: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Visualization. **Paweł Zyblewski:** Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft. **Paweł Ksieniewicz:** Conceptualization, Methodology, Validation, Investigation, Resources, Writing – review & editing, Visualization, Supervision.

Declaration of competing interest

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

Acknowledgments

This work was supported by the Polish National Science Centre under the grant No. 2017/27/B/ST6/01325 as well by the statutory funds of the Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Science and Technology, Poland.

References

[1] A. Bifet, Classifier concept drift detection and the illusion of progress, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh, J.M. Zurada (Eds.), Artificial Intelligence and Soft Computing, Springer International Publishing, Cham, 2017, pp. 715–725.

[2] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014).

[3] S. Hashemi, Y. Yang, Flexible decision tree for data stream classification in the presence of concept change, noise and missing values, Data Min. Knowl. Discov. 19 (1) (2009) 95–131.

[4] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, F. Herrera, A survey on data preprocessing for data stream mining: Current status and future directions, Neurocomputing 239 (2017) 39–57.

[5] R. Barros, S. Santos, A large-scale comparison of concept drift detectors, Inform. Sci. 451–452 (2018).

[6] M. Bahri, A. Bifet, J. Gama, H.M. Gomes, S. Maniu, Data stream analysis: Foundations, major tasks and tools, WIREs Data Min. Knowl. Discov. 11 (2021).

[7] P. Zyblewski, P. Ksieniewicz, M. Woźniak, Classifier selection for highly imbalanced data streams with minority driven ensemble, in: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J.M. Zurada (Eds.), Artificial Intelligence and Soft Computing, Springer International Publishing, Cham, 2019, pp. 626–635.

[8] J. Komorniczak, P. Zyblewski, P. Ksieniewicz, Prior probability estimation in dynamically imbalanced data streams, in: 2021 International Joint Conference on Neural Networks, IJCNN, IEEE, 2021, pp. 1–7.

[9] I. Zliobaite, Learning under concept drift: An overview, 2010, CoRR abs/1010.4784.

[10] G. Webb, R. Hyde, H. Cao, H.-L. Nguyen, F. Petitjean, Characterizing concept drift, Data Min. Knowl. Discov. 30 (2016).

[11] G. Widmer, M. Kubat, Effective Learning in Dynamic Environments by Explicit Context Tracking, Vol. 667, 1994.

[12] R. Barros, S. Santos, A large-scale comparison of concept drift detectors, Inform. Sci. 451–452 (2018).

[13] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: A.L.C. Bazzan, S. Labidi (Eds.), Advances in Artificial Intelligence, SBIA 2004, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 286–295.

[14] M. Baena-García, J. Campo-Ávila, R. Fidalgo-Merino, A. Bifet, R. Gavald, R. Morales-Bueno, Early drift detection method, 2006.

[15] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the 7th SIAM International Conference on Data Mining, Vol. 7, 2007.

[16] S.H. Bach, M.A. Maloof, Paired learners for concept drift, in: 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 23–32.

[17] I. Frías-Blanco, J.d. Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on hoeffding's bounds, IEEE Trans. Knowl. Data Eng. 27 (3) (2015) 810–823, http://dx.doi.org/10.1109/TKDE.2014.2345382.

[18] S. Micevska, A. Awad, S. Sakr, SDDM: An interpretable statistical concept drift detection method for data streams, J. Intell. Inf. Syst. 56 (2021).

[19] C. Molnar, Interpretable Machine Learning, 2020.

[20] G. Webb, L. Lee, F. Petitjean, B. Goethals, Understanding concept drift, 2017.

[21] D.A. Levin, Y. Peres, Markov Chains and Mixing Times, Vol. 107, American Mathematical Soc., 2017.

[22] J. Kolter, M. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts., J. Mach. Learn. Res. 8 (2007) 2755–2790.

[23] L.L. Minku, X. Yao, DDD: A new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. 24 (4) (2012) 619–633.

[24] L. Du, Q. Song, L. Zhu, X. Zhu, A selective detector ensemble for concept drift detection, Comput. J. 58 (3) (2014) 457–471.

[25] B. Maciel, S. Santos, R. Barros, A lightweight concept drift detection ensemble, 2015.

[26] A. Bifet, Classifier concept drift detection and the illusion of progress, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh, J.M. Zurada (Eds.), Artificial Intelligence and Soft Computing, Springer International Publishing, Cham, 2017, pp. 715–725.

[27] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE Trans. Knowl. Data Eng. 31 (12) (2018) 2346–2363.

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

[29] N. Street, Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in: Proceedings of the 7Th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 377–382.

[30] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '03, ACM, New York, NY, USA, 2003, pp. 226–235.

[31] D. Brzeziński, J. Stefanowski, Accuracy updated ensemble for data streams with concept drift, in: E. Corchado, M. Kurzyński, M. Woźniak (Eds.), Hybrid Artificial Intelligent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 155–163.

[32] M. Woźniak, A. Kasprzak, P. Cal, Weighted aging classifier ensemble for the incremental drifted data streams, in: Flexible Query Answering Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 579–588.

[33] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.

[34] R. Polikar, L. Upda, S.S. Upda, V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, IEEE Trans. Syst., Man, Cybern., Part C (Applications and Reviews) 31 (4) (2001) 497–508.

[35] H.M. Gomes, A. Bifet, J. Read, J.P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, T. Abdessalem, Adaptive random forests for evolving data stream classification, Mach. Learn. 106 (9) (2017) 1469–1495.

[36] N.C. Oza, S.J. Russell, Online bagging and boosting, in: International Workshop on Artificial Intelligence and Statistics, PMLR, 2001, pp. 229–236.

[37] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2010, pp. 135–150.

[38] H.M. Gomes, J. Read, A. Bifet, Streaming random patches for evolving data stream classification, in: 2019 IEEE International Conference on Data Mining, ICDM, IEEE, 2019, pp. 240–249.

[39] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844.

[40] D. Wang, P. Wu, P. Zhao, Y. Wu, C. Miao, S.C. Hoi, High-dimensional data stream classification via sparse online learning, in: 2014 IEEE International Conference on Data Mining, IEEE, 2014, pp. 1007–1012.

[41] A. Cano, B. Krawczyk, Kappa updated ensemble for drifting data stream mining, Mach. Learn. 109 (1) (2020) 175–218.

[42] D. Brzezinski, J. Stefanowski, R. Susmaga, I. Szczech, Visual-based analysis of classification measures and their properties for class imbalanced problems, Inform. Sci. 462 (2018) 242–261.

[43] D. Brzezinski, J. Stefanowski, R. Susmaga, I. Szczech, On the dynamics of classification measures for imbalanced and streaming data, IEEE Trans. Neural Netw. Learn. Syst. (2019) 1–11.

[44] H. Hu, M. Kantardzic, T.S. Sethi, No free lunch theorem for concept drift detection in streaming data classification: A review, Wiley Interdiscipl. Rev.: Data Min. Knowl. Discov. 10 (2) (2020) e1327.

[45] E. Hellinger, Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen., J. Für Die Reine Und Angew. Math. 1909 (136) (1909) 210–271.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[47] P. Ksieniewicz, P. Zyblewski, Stream-learn–open-source python library for difficult data stream batch analysis, 2020, arXiv preprint arXiv:2001.11077.

[48] J. Montiel, J. Read, A. Bifet, T. Abdessalem, Scikit-multiflow: A multi-output streaming framework, J. Mach. Learn. Res. 19 (72) (2018) 1–5.

[49] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T.E. Oliphant, Array programming with NumPy, Nature 585 (7825) (2020) 357–362.

[50] J. Komorniczak, P. Ksieniewicz, Data stream generation through real concept's interpolation, in: 30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2022, (Bruges, Belgium), October 5-7, 2022, 2022.

[51] A. Cano, B. Krawczyk, ROSE: Robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams, Mach. Learn. (2022) 1–39.

[52] G. Alberghini, S.B. Junior, A. Cano, Adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label drifting data streams, Neurocomputing (2022).