Arabian Journal for Science and Engineering (2022) 47:10605–10621
https://doi.org/10.1007/s13369-022-06653-4

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Disposition-Based Concept Drift Detection and Adaptation in Data Stream

Supriya Agrahari[1] · Anil Kumar Singh[1]

Received: 26 September 2021 / Accepted: 30 January 2022 / Published online: 8 March 2022

Abstract

The change in data distribution over time (known as concept drift) makes the classification process complex because of the discrepancy between current and incoming data distribution. A plethora of drift detection methods often focus on the early identification of concept drift. Along with the drift, other deformities like noise and blips are also present in the data stream. These deformities may be damaged the underlying learning system by forcing adaptation to false drift. Thereby unnecessary update performs in the learning model that leads to decrease in learner's accuracy. The existing drift detection methods are not capable of differentiating between actual and false drift. The paper proposes DBDDM, a disposition-based drift detection method, to overcome the issue of false drift. In this paper, we utilize the approximate randomization test to find the frequency of consecutive drift and compare the obtained frequency with the threshold to determine the actual drift. DBDDM compares with the several state-of-the-art methods using synthetic and real-time datasets. It exhibits a maximum increase in accuracy of 24% and 28% with a rise of 2.50 and 1.91 average ranks using Naive Bayes and the Hoeffding tree classifier, respectively.

Keywords Concept drift · Data-stream mining · Disposition based drift detection method (DBDDM) · Learning model

1 Introduction

In the growing era of technologies, a tremendous amount of streaming data generates from various applications. The streaming data may have unstable distributions [1]. The change in distribution with respect to time is known as concept drift. The data distribution change needs to be analyzed because it adds a concept drift problem in the data stream of infinite length. The concept (or context) refers to target values or classes. The data stream instances are continuous; thus, we have a short period (or single-pass) to look into the data. Sometimes, it is difficult to analyze these instances because of their characteristics like varying speed, timely ordered, and rapidly changing distribution [2].

The drift detection method is associated with the learning model [3]. The detector is used to find the significant change in the concept, whereas the learning model (or classification model) is used to forecast the outcomes of data instances (or examples). An increase in false alarm rate, decrease in accuracy of learning model, increase in classification error rate, etc., commonly identify the concept change in the data stream. Sometimes the accuracy of the classifier or predictor is degraded even the concept is stable for a long time [3]. So, it is necessary to examine such conditions in the data stream.

Concept drift detection requires in various applications such as fraud detection, cyber-security, gas sensor analysis, medical information, churn prediction, weather forecasting, etc. The applications generally use the learning model to predict the incoming data patterns. Due to concept drift, the learning model eventually becomes obsolete because it trains using old data instances. The distribution of incoming data instances changes over time. Hence, the learning model needs to be retrained using the current data distribution to minimize adverse situations like malicious activities, disasters, medical emergencies, etc.

The drift is classified as a sudden (or abrupt), gradual, incremental, recurring, blip, noise, etc., in terms of speed of change (see Fig. 1). The speed of change denotes the transition period between consecutive concepts [4]. Sudden drift occurs when an incoming data instance suddenly originates a new concept, i.e., the point of change from an old class to

✉ Supriya Agrahari
supriyaagrahari@mnnit.ac.in

Anil Kumar Singh
ak@mnnit.ac.in

[1] Computer Science and Engineering Department, Motilal Nehru National Institute of Technology, Allahabad, India

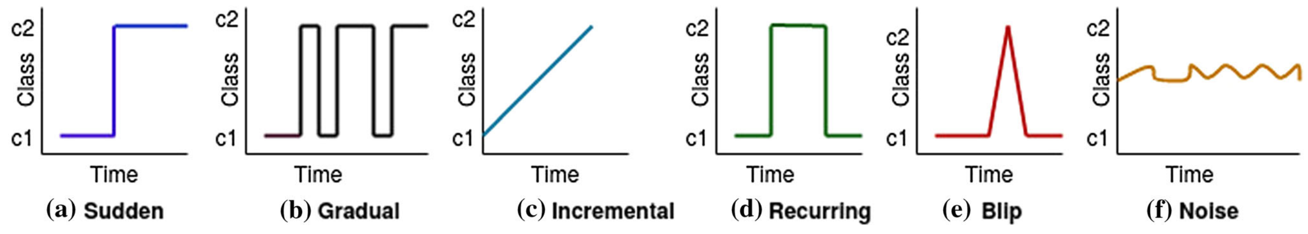


Fig. 1 Different drift types

a new class is considered sudden drift. In the case of gradual drift, a context change occurs gradually in the data instances. Therefore, the occurrence time of gradual drift is more than the sudden drift. The incremental drift is seen when the data instances slowly change their values concerning time. Recurrent drift happens if the same concept is seen after some time interval like cyclic phenomena. Blip represents quick and sudden change (or rare event) in concept. It can refer as an outlier in a stationary distribution. Noise signifies an unexpected change in the distribution of data instances, and it should be filtered out proficiently.

In the streaming environment, a wide range of methods evolves to address the concept drift problems. Generally, the concept drift detection methods are categorized according to their working behavior. Some researchers consider the distribution-based drift detection methods as the most accurate drift detectors because it is capable of representing the corresponding confidence intervals and directly addresses the root causes of concept drift [5–7]. Although these kinds of drift detection methods have made noticeable achievements, it still encounters some restrictions.

- *Delusion of false drift*: Several drift detectors are designed to identify different types of drifts. While detecting the drift, other deformities are also present in the data stream, such as blips and noise. Blips stand for rare random changes in the data stream. It should be neglected and not mistaken for a drift. Noise stands for significant corruption in the target values or attributes values, and it should be filtered out to avoid the classifier's feeding adversarial or inaccurate information. These deformities can cause delusion of drift [8]. Such drift is considered as false drift. Thereby unnecessary update is performed in the learning model. In this scenario, the detection of false drift remains an issue in the data stream. Hence, it is required to find the actual drift as per the change in data distribution to improve the learning model's performance.
- *Classification problem*: The binary-class and multi-class classification problems may exist in a real-time environment. These problems are independent of the presence or absence of concept drift. In multi-class classification problems, the existing methods may not be able to admeasure the differences between the prediction of learning models when they do not correctly predict the same example via different target values. As a result, the performance of a learning model is degraded. In this way, the classification problem with concept drift becomes more challenging in data streams. Many drift detection methods are built to handle binary-class classification problems [9–11], still some methods do not address multi-class classification problems [12].
- *Drift handling*: There are various drift detection methods built to detect different types of drifts present in the data stream (see Fig. 1). The performance of most of the existing drift detectors is better either for sudden drift or gradual drift, but not for both [13].

The proposed work aims to develop a concept drift detection method that efficiently analyzes the change in the context of data and finds actual drift. Most of the existing drift detectors monitor some characteristics of currently incoming data instances. They generally define some threshold to find the drift, if new incoming instances are significantly different. But the change in concept may occur due to deformities of data, and a single concept change is not sufficient to decide actual concept drift. Thus, the variables are introduced to determine the continuous change in context in the streaming environment.

In order to deal with adversarial drift in the data stream, the paper proposes a Disposition-Based concept Drift Detection and adaptation Method, DBDDM. The distribution-based drift detector method performs an approximate random test to find considerable change in two-windows data instances using the absolute mean difference as a statistical measure. The proposed method performs the statistical significance analysis based on hypothesis testing to determine the drift. There are two significant levels defined: warning level and drift level. Both the levels are based on two variables, namely drift_count and Flag. As per the defined threshold for significance levels, the window size varies to reduce the events of misclassification error. The principal contributions of the paper are:

- We develop the drift detection method DBDDM, which is based on two-window analysis. The method detects the changes using a random test as a statistical test. The hypothesis test is used to detect whether the concept change occurs or not. It performs incremental concept drift detection and adaptation in a non-stationary environment.

- The proposed method introduces a variable Flag to design robust machine learning methods for adversarial concept drift detection. The Flag identifies the consecutive drifts to overcome the problem of noise and blip, which creates a delusion of concept drift.
- To verify the statistical significance of the performance of DBDDM and the compared methods using NB and HT classifier, we utilize the *Friedman* test with *Nemenyi-post-hoc* analysis. It shows that DBDDM is significantly better than DDM, ECDD with NB classifier, and ADWIN, ECDD, SEED, and SEQDRIFT2 with HT classifier.
- We experimentally evaluate the proposed method using various synthetic datasets which contains sudden and gradual drifts. The results show that DBDDM detects sudden and gradual drift efficiently.
- DBDDM is a distribution-independent and model-independent window-based approach. In addition to this, it also deals with binary-class and multi-class classification problems.
- We have conducted an ablation study on hyperparameters to understand the impact of the change in current window size and the number of possible shuffling of concatenated window data instances in the proposed model. We experimentally show how the varying size of hyperparameters impacts the accuracy of the learning model.

The rest of the paper is organized as follows: the categorization of drift detection and adaptation methods (Sect. 2.1), and discussion of related research work (Sect. 2.2) along with analysis (Sect. 2.2.1) is present in related work (Sect. 2). Preliminaries are defined in Sect. 3. Section 4 presents a discussion on the proposed method, DBDDM, with workflow, algorithms, and phases (Sect. 4.2). Section 5 illustrates experimental analysis, which contains the description of datasets (Sect. 5.1), experimental experiment, and parameter setting (Sect. 5.2). Result evaluation is given in Sect. 6, where Sect. 6.1 contains experimental results and analysis, and statistical comparison of methods presents in Sect. 6.2. Finally, the conclusion is in the last section.

2 Related Work

This section focuses on categorization of existing drift detection and adaptation methods (Sect. 2.1), followed by some recent work and research related to the proposed work (Sect. 2.2). Further, research analysis related to concept drift presents in Sect. 2.2.1.

2.1 Categorization of Drift Detection and Adaptation Methods

Drift detection and adaptation methods distinguish into two parts based on literature: passive and active approaches. Passive approaches do not depend on drift occurrence in the data examples. It updates the learning model whenever new data instances come into existence. Whereas the active approaches [14] of drift detectors categorize as (1) statistical analysis-based methods generally deal with statistical computations like mean, median, skewness, kurtosis, etc., to detect drift. (2) In sequential analysis-based methods, the data instances are analyzed one by one to find the drift. It requires more data instances of a new concept. (3) Window analysis-based methods usually use two windows, i.e., fixed and adaptive windows, to identify the drift. The fixed window uses a specific length of the window for drift detection. At the same time, the adaptive window refers to the dynamic adjustment of the window. The adaptive window size depends on drift occurrences, i.e., the window size gets shrink whenever the drift is detected; otherwise, it expands. The further categorization of drift detection methods [15] is discussed in Table 1.

2.2 Research Related to Concept Drift

In this section, we discuss various existing drift detectors. One of the recent works Mahdi et al. [36] focuses on concept drift detection in the presence of multiple classes. Due to multi-class in the data stream, there are high costs in memory consumption and run time. It develops a hybrid block-based ensemble (HBBE), an approach that combine online drift detection methods. In addition to this an online drift detector for K-class problem (ODDK) is built, that contains a pair of base learners to detect drifts. This method is constructed for K-class problems with block-based weighting to deal with various types of drifts. It calculates diversity using a new technique for the K-class problem and is able to find sudden, gradual, and recurring drifts.

Mahdi et al. [37] proposes a drift detector KAPPA, which is designed to detect sudden drift. KAPPA measures quickly drop in incorrect predictions. So, it is more useful than using error rate or accuracy that only introduces small changes. The competence of a classifier is evaluated by measuring the inter-rater agreement between correct predictions. PH test is considered and compared with a threshold to detect drift. On the other hand, Mehmood et al. [38] focuses on concept drift detection in the field of smart city applications. Change or shifting in ground truth generally rebuilds the predictive model in the analytical tasks. It uses PHT, DDM, EDDM, and ADWIN methods for concept drift handling.

Heusinger and Schleif [39] proposes a concept drift detection method based on Minimum Enclosing Ball (MEB),

Table 1 Categorization of drift detection methods

Categories	Keypoints	Methods
Error rate-based methods	They are based on the classification error of learning model to detect drift	DDM [16], LLDD [17], STEPD [18], FW-DDM [19,20], HDDM [4], EDDM [21], DELM, EWMA [22], ADWIN [23,24], ECDD [22], SEED [25], SEQDRIFT2 [26]
Significance analysis-based methods	They consider some prior assumptions (or hypothesis), i.e., null hypothesis and alternative hypothesis. These assumptions are used to detect drift	TSMSD-EWMA [27], HCDTs [28], HLFR [29], DBSCAN [30]
Data distribution-based methods	They determine the considerable change in the data samples distribution to identify the concept drift	LDD-DSDA [19,20], EDE [31], CM [2,6], SCD [32], LSDD-CDT [33,34], LSDD-INC [33,34], STEPD [35], Wilcoxon-rank-sum test [3]

which can process the higher-dimensional data very fast. It is a window-based method that keeps all the data points of the current window. When new data points come into existence, it removes old data points. The method checks that if there are any data points outside the ball, these data points belong to another concept. The method works for binary-class classification problem.

Misra et al. [40] presents Fourier Inspired Windows for Concept Drift detection (FIWCD), a Fourier analysis-based mechanism to determine the window length. It has a buffer window in which it contains large and small overlapping recent window data instances. If the model values of the adjacent buffer window diverge beyond a threshold, the concept change is detected.

Mahdi et al. [41] proposes diversity measure as a new drift detection method (DMDDM). The method reacts quickly to concept change in less time and it consumes less memory. It combines diversity measures, and disagreement measures with the Page-Hinkley test to detect drift. It analyzes the diversity of a classifier's pair using the fading factor.

The learning under adversarial concept drift is focused by Korycki and Krawczyk [8]. It finds the valid drifts and adversarial drifts. A novel approach, Robust Restricted Boltzmann Machine Drift Detector, is introduced to handle adversarial instances. It uses an improved gradient method which makes the method more robust to adversarial concept drift. Further, a novel measure, Relative Loss of Robustness, is used to evaluate the performance of the drift detector.

One of the baseline methods for concept drift detection is DDM [16]. The method considers the binomial distribution of the data stream to detect the drift. It measures the error rate of available data. The method detects sudden and gradual drift. DDM considers two levels for drift detection, namely warning level (i.e., concept drifts may have happened) and drift level (i.e., drift is confirmed). The condition for warning level and drift level define as $(p_i + s_i \geq p_{\min} + (2 * s_{\min}))$ and $(p_i + s_i \geq p_{\min} + (3 * s_{\min}))$, respectively. Here, p_i and s_i are the probability of error rate (denotes that data instances are not classified correctly) and standard deviation, respectively.

An adaptive sliding window mechanism considers in ADWIN [24]. A successive method, i.e., ADWIN2, is also proposed by the author. The drift is detected when the average distribution difference of the two consecutive windows is more considerable than the predefined threshold. ADWIN2 [23] overcomes the limitation of ADWIN by detecting the slow-gradual drift and consuming less memory and time.

An exponentially weighted moving average chart-based method is ECDD [22]. It performs the classification of data samples to detect drift. It uses a feedback mechanism and examines the false-positive rates in a controlled way. The paper claims that it has only O(1) overhead for the classifier.

The rate of change in concept is a focused area in SEED [25]. The method performs drift detection in the first phase and volatility detection (i.e., rate of change in concept) in the next phase. A window mechanism is used by the method. SEED is based on block compression and finds the actual cut point in the first phase. The next phase utilizes the cut points and their relative location to infer whether there is a change in the rate at which cut points happened. The method checks consecutive blocks and merges them if they are homogeneous. For drift detection, two samples mean values evaluate with a specific allowable false-positive rate.

SEQDRIFT2 [26] is based on SEQDRIFT1 [42] which can be seen as an improved variant of ADWIN. It is a sliding window-based method and provides memory management using reservoir sampling. Bernstein Bound is used to find the change between population and sample mean. Further, hypothesis testing performs to analyze the concept drift.

A statistical analysis-based method is STEPD [35], which uses two classifiers to analyze the predictive accuracy of the learning model. It considers two accuracies, i.e., current accuracy and overall accuracy. These accuracies compare with equal proportions for statistical analysis. The method contains two significance levels for warning and drift conditions. WSTD uses Wilcoxon-rank-sum test [3] and is based on STEPD. It modifies the statistical test, which is used to signal the warnings and drifts. Compared to STEPD, it limits the size of the older window. The method works better for abrupt drift as compare to gradual drift.

The literature extensively uses all the above-discussed methods. They are a well-balanced selection of both older and newer drift detectors. These methods use different strategies to detect drift and are based on the windowing of data instances to some extend.

2.2.1 Analysis of Related Work

HBBE and ODDK address the multi-class classification problems. The detectors consume less memory. HBBE can be fast to detect single drift, whereas ODDK detects multiple drifts in less time. On the other hand, KAPPA drift detector is limited to detecting the sudden concept change. Mehmood et al. [38] defines the limitation of the approach as that its evaluation is based on a few existing detectors and real-time predictive models. As a result, it is not easy to provide well-established directions on the particular settings or application domains. Heusinger and Schleif propose a drift detection method that is capable of quickly processing the higher-dimensional data. But it is limited to detecting the binary-class classification problems only. Another method, DMDDM, detects the sudden drift and works for binary-class classification problems.

DDM cannot detect the slow-gradual drift and considers the binomial distribution of the data stream to detect drift. The performance of DDM is usually deteriorated when the concepts are stable for a longer time or a very large concept is present. It does not aid the noisy data, whereas ADWIN2 overcomes the constraint of ADWIN concerning time and memory. Still, it works only for single-dimensional data. ECDD is based on EWMA and work only for two-class classification problems. The SEED method works on user-defined thresholds and is limited to specific drift detection. Here, Hoeffding inequality is used along with Bonferroni correction as present in ADWIN. SEQDRIFT2 is an improved variant of ADWIN and has a better false-positive rate than ADWIN and EWMA. Still, it requires to define a false positive rate. Instead of the test of equal proportions used in STEPD, WSTD utilizes the Wilcoxon rank-sum statistical test. It constraints the size of the older window.

3 Preliminaries

3.1 Data Stream

The data stream is a continuous flow of varying volumes and velocities of data. The incoming data distribution may change concerning time. It causes a drift in the data stream. The data stream D_s can be defined as sequences of samples $\{S_1, S_2, \ldots, S_p, \ldots\}$, and these samples contain a collection of data instances or examples. The labeled examples represent as $\{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$, where X is input attribute vector, y is target or class values, and n is the number of examples.

3.2 Concept Drift

Suppose at time stamp t_u and t_v, the data distribution of ith and jth instance is $P(X_i, y_i)_{t_u}$ and $P(X_j, y_j)_{t_v}$, respectively, where $P(X_i, y_i)$ is the joint probability distribution of ith instance of data sample. Thus, $P(X_i)_{t_u} \; ! = \; P(X_j)_{t_v}$ (or $P(X_i, y_i)_{t_u} \mathrel{!=} P(X_j, y_j)_{t_v}$) defines the condition of concept drift. It shows that the distribution of input attribute vector itself (or the distribution of target class value with respect to input attribute vector) changes over time.

4 Proposed Work

This section presents a discussion on the proposed method, DBDDM, with workflow (Fig. 2), algorithms (Algorithms 1, 2, and 3), and phases (Sect. 4.2).

4.1 Overview of Proposed Method

The proposed drift detection method is based on the windowing of data instances to check the concept change (or drift) in the data stream. In this method, we use two windows, namely the anchored and current windows. The anchored and current windows contain initial and recent data instances, respectively. The specified size of data instances is grouped and stored in these windows. This process is the same for each new incoming window. For concept drift detection, the method learns from the change in $P(X)$ and uses the exact test to analyze whether the data distribution is stable with time. The exact test is a statistical test and performs a random test to compare the distribution corresponding to the two independent samples. The significance of the exact test is determined by hypothesis testing, which is based on null and alternate hypotheses. Further, this paper uses two significance levels, namely warning level and drift level. These levels are determined by two thresholds, i.e., drift_count and Flag.

4.2 Disposition-Based Concept Drift Detection and Adaptation Method (DBDDM)

The working of the proposed method discusses in three different phases: Initial Phase, Drift Detection Phase, and Model Update Phase. The pseudocode describes in Algorithms 1, 2, and 3, and the workflow of a block diagram depicts in Fig. 2. The algorithmic parameters and their mnemonics are present in Table 2. The meaning of these parameters explains whenever used first time in the paper.

4.2.1 Initial Phase

In the real-time scenario, the instances of data stream arrive one by one with uniform or non-uniform velocity. These instances are stored in the window. The stored data is read in sequence for processing. Here, the window behaves as a FIFO queue, which is dynamic. The initial window acts as anchored window W_a and the new incoming window denotes as current window W_c. The data examples of W_a use to build the base learning model, and the learning model predicts the target values of incoming data examples. As per the predicted target values, the performance measure of the learning model is evaluated prequentially (interleaved-test-then-train). Here, we consider the mean accuracy as a performance measure. The prediction results compare with the actual target values. If they are similar, '1' is set as True Positive; otherwise, '0' is considered as True Negative. The mean accuracy is defined in Eq. 1:

$$\text{Mean Accuracy}_i = \frac{\text{TP}_i + \text{TN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i} \quad (1)$$

here i, TP, TN, FP, and FN denote ith window of the data stream, True Positives (correct positive prediction), True Negatives (correct negative prediction), False Positives (incorrect positive prediction), and False Negatives (incorrect negative prediction), respectively. Whenever a new current window comes into existence, the mean accuracy is calculated. By evaluating the mean accuracy, the method analyzes the behavior of incoming data instances with the existing learning model. It shows how the learner predicts correctly, and our method detects the change in context efficiently.

4.2.2 Drift Detection Phase

In order to perform drift detection, the proposed method analyzes the distribution change between two window instances by the drift_check() and drift_detector() methods consecutively. The primary criteria for drift detection are that the size of the anchored window (W_a) should be equal to the incoming current window size (W_c). In DBDDM, the drift detection (see Algorithms 2 and 3) and the prediction of outcome by

Algorithm 1: DBDDM

```
Input: Data Stream $D_s$; Anchored Window $W_a$; Current
       Window $W_c$; Base Classifier $C_B$; Warning level constant
       $\alpha$; Drift level constant $\beta$; Flag; drift_count.
Output: Classification Accuracy; No. of Drift Detected; Time to
        Detect Drift.
1  Initialize $W_c$, Flag, $C_B$, $\alpha = 0.50$, $\beta = 0.75$;
2  while Stream_has_more_data do
3      if sizeof($W_c$) != FULL   then
4          Add current data instance of $D_s$ into window;
5          Compute mean accuracy;
6      else
7          if Drift_Check($W_a$,$W_c$, drift_count) then
8              drift_count = drift_count + 1;
9              if (drift_count $< \theta$) and (Flag $< \theta$) then
   //Warning Level
10                 Set $W_c = \alpha * W_c$;
11                 Set Flag = Flag + 1;
12                 Evaluate the model using incoming data instances;
13             else
   //Drift Level
14                 Set $W_c = \beta * W_c$;
15                 Evaluate and retrain the model using current
   distribution of data instances;
16                 Reset();
17             end if
18         else
19             Set Flag = Flag - 1;
20             Evaluate the model using incoming data instances;
21         end if
22     end if
23 Compute classification accuracy;
24 function Reset()
25     Set Flag = 0;
26     Set drift_count = 0;
27     Forget the old information from the buffer space;
28 end function
29 end while
```

Algorithm 2: Drift_Check(W_a, W_c, drift_count)

```
Input: Anchored Window $W_a$; Current Window $W_c$; drift_count,
       Array p[ ].
1  if (SizeOf($W_a$)==SizeOf($W_c$))
2      if (len(p) == 0) or (drift_count $< \theta$) then
3          Set p value = drift_detector($W_a$,$W_c$);
4          Append p value in p;
5          return True;
6      end if
7      if drift_count $>= \theta$ then
8          Set p value = drift_detector($W_a$,$W_c$);
9          Append p value in p;
10         return True;
11     else
12         return False;
13     end if
14 end if
```

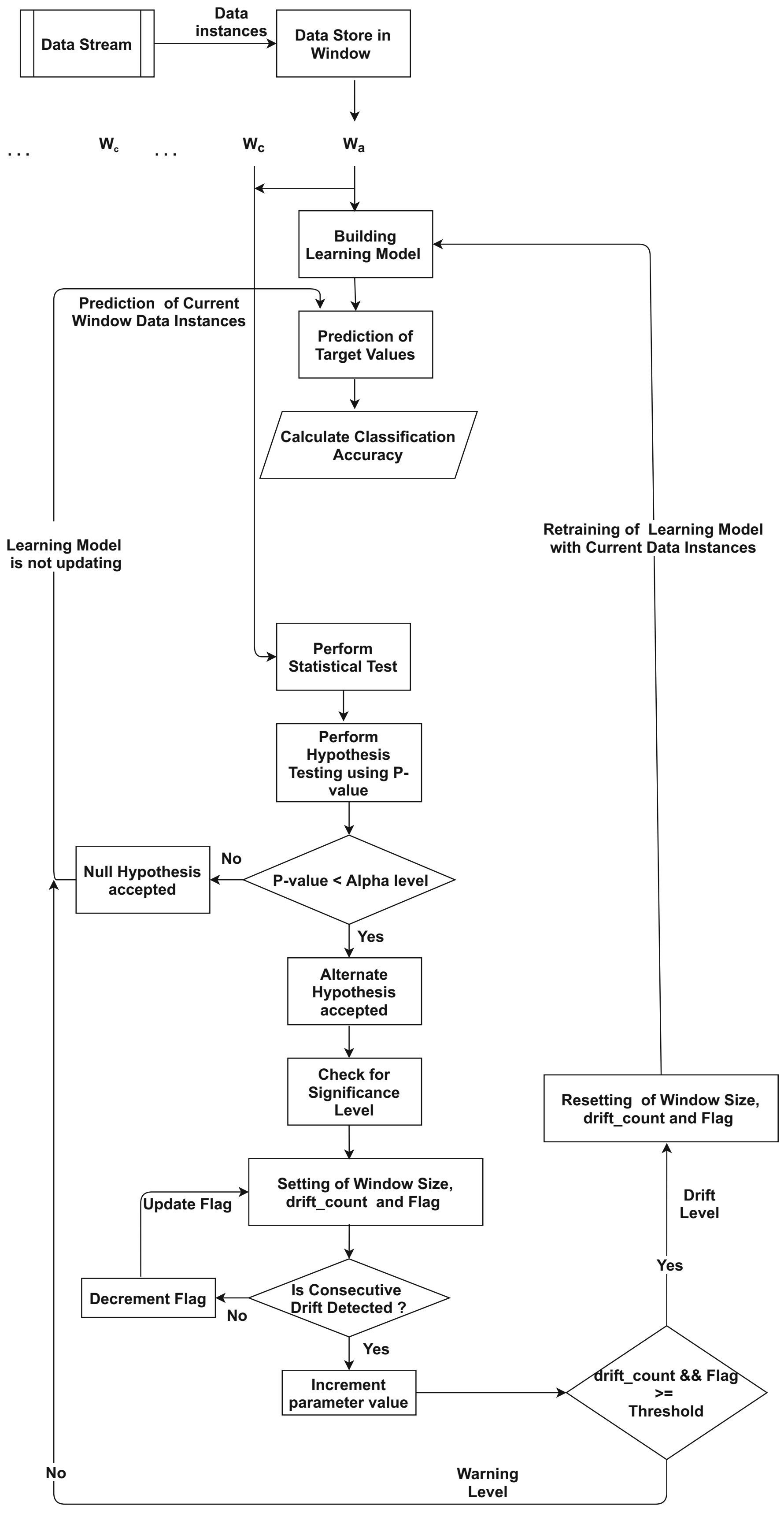

Data Stream
Data instances
Data Store in Window
. . .
W_c
. . .
W_c
W_a
Building Learning Model
Prediction of Current Window Data Instances
Prediction of Target Values
Calculate Classification Accuracy
Learning Model is not updating
Retraining of Learning Model with Current Data Instances
Perform Statistical Test
Perform Hypothesis Testing using P-value
Null Hypothesis accepted
No
P-value < Alpha level
Yes
Alternate Hypothesis accepted
Check for Significance Level
Resetting of Window Size, drift_count and Flag
Setting of Window Size, drift_count and Flag
Update Flag
Drift Level
Yes
Decrement Flag
Is Consecutive Drift Detected ?
No
Yes
Increment parameter value
drift_count && Flag >= Threshold
No
Warning Level

Fig. 2 Workflow diagram of DBDDM

Table 2 Parameters and their Mnemonics

Parameters	Symbols
Data stream	D_s
Base Classifier	C_B
Anchored Window	W_a
Current Window	W_c
Concatenated Window	W_{con}
Warning level window size	$\alpha * W_c$
Drift level window size	$\beta * W_c$
Absolute mean difference of two subsequent window	D_i

Algorithm 3: drift_detector(W_a, W_c)

```
   Input: Anchored Window $W_a$; Current Window $W_c$;
          Concatenated Window $W_{con}$; alpha level; Array $D_a$.
   Output: Hypothesis accepted; Hypothesis rejected.
1  Initialize alpha level is 0.05;
2  Perform exact test using absolute mean difference of $W_a$ and $W_c$;
3  Compute absolute mean difference of [0 to $len(W_{con})/2$] and
   [$len(W_{con})/2 + 1$ to $len(W_{con})$];
4  for calculate p value until range(len($W_{con}$) * 2) do
5      Add value in $D_a$ when absolute mean difference of $W_a$ and
       $W_c$ <= absolute mean difference of multiple shuffle windows of
       [1 to $len(W_{con})/2$] and [$len(W_{con})/2 + 1$ to $len(W_{con})$];
6  Set p value = $\sum_{i=1}^{D_a}(D_a)_i$ / len($D_a$);
7  end for
8  if p value < alpha level then
9          return p value;
10 else
11         return False;
12 end if
```

the learning model (see Algorithm 1) perform simultaneously. Here, the prequential analysis of the learning model follows supervised learning, whereas the concept drift detection phase is based on unsupervised learning.

For drift detection, we use the absolute mean difference as a statistical measure to perform statistical analysis. The analysis determines whether the distribution of two windows' data instances is the same. Here, the anchored window (W_a) and current window (W_c) at time stamp t_u and t_v are represented as: $W_a = \{X_{1,t_u}, X_{2,t_u}, \ldots, X_{n,t_u}\}$ and $W_c = \{X_{1,t_v}, X_{2,t_v}, \ldots, X_{n,t_v}\}$, where n denotes number of instances in a particular window and X contains m attributes, i.e., $X = \{a_i, a_{i+1}, \ldots, a_m\}$. The change in distribution between the data examples of both windows is evaluated in following steps. In the first step, the absolute mean difference (D_i) is calculated to determine the divergence between the data examples of two windows (see Eq. 2) and it becomes a base measure for further randomized calculations.

$$D_i \longleftarrow |\mu(W_a) - \mu(W_c)| \tag{2}$$

In the second step, we build a concatenated window (W_{con}) by concatenating the data examples of W_a and W_c (see Eq. 3) to perform the random test.

$$W_{con} \longleftarrow \{X_{1,t}, X_{2,t}, \ldots, X_{n,t}, X_{1,t+1}, X_{2,t+1}, \ldots, X_{n,t+1}\} \tag{3}$$

In third step, the data instances of W_{con} are shuffled and then divided into two-part, i.e., [1 to len(W_{con})/2] and [len(W_{con})/2 + 1 to len(W_{con})]. The absolute mean difference of shuffled windows is calculated. Further, the random test is performed len(W_{con})∗2 times based on disposition (or shuffling) of the data instances of W_{con}. This random sample uses to make a statistical inference. The absolute mean difference of shuffled windows compares with the base difference (D_i). The result stores in the array D_a (see Eq. 4).

$$D_a = \begin{cases} 1, & \text{if } D_i \leq \left|\mu\left(\left[1 : \frac{\text{len}(W_{con})}{2}\right]\right) - \mu\left(\left[\frac{\text{len}(W_{con})}{2} + 1 : \text{len}(W_{con})\right]\right)\right| \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

In obtained result, the values '1' and '0' denote non-favorable and favorable conditions, respectively. The favorable condition shows that the absolute mean difference of original data instances of W_a and W_c is similar to shuffled data instances ([1 to len(W_{con})/2] and [len(W_{con})/2 + 1 to len(W_{con})]). The non-favorable conditions show that original data samples' absolute mean difference is dissimilar to shuffle data samples. The above procedure performs the random test and compares the considerable change in the distribution of two independent windows. This procedure is simple, and there is no requirement of the mathematical assumption.

In addition to this, the significance of the random test is analyzed by hypothesis testing. The hypothesis testing is based on the null hypothesis H_0 and alternate hypothesis H_a. The hypothesis testing uses to check the distribution of test statistics, which is performed by calculating all possible random shuffling of data examples. For experimental purpose,

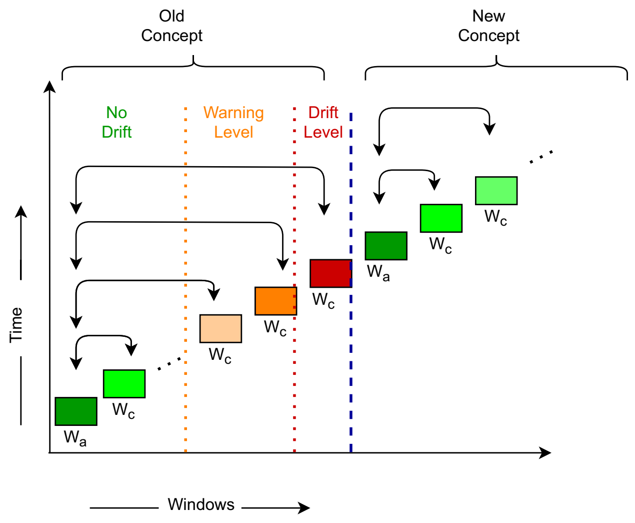


Fig. 3 Depiction of windows during concept evolution

the null hypothesis (H_o) and alternate hypothesis (H_a) is defined as below:

H_o: $W_a = W_c$

H_a: W_a != W_c

Here, W_a and W_c contain two independent window's data instances. If the data instances distribution of W_a is similar to W_c, the null hypothesis (H_o) is accepted. At the same time, the dissimilarity exists between two windows' data instances distribution in the case of the alternate hypothesis (H_a).

When a statistical test is performed, the p value is used to determine the significance of outcomes in relation to the null hypothesis. The p value is the frequency, which is based on random data samples with favorable conditions or non-favorable conditions. It defines that the test statistic would be at least as extreme as we observed if H_o is true. The p value helps to find the firmness of evidence to support the null hypothesis. The p value is calculated in Eq. 5.

$$p \text{ value} = \frac{\left(\sum_{i=1}^{\text{len}(D_a)} D_{a_i}\right)}{len(D_a)} \tag{5}$$

The alpha level is set to 0.05 for evaluation purposes, which shows 95% of the confidence interval. If the p value is less than the alpha level, the null hypothesis is rejected; otherwise, the null hypothesis is accepted. The rejection of the null hypothesis considers as a drift in the data stream.

Further, the method discards the current window W_c to accommodate new incoming current window data instances. It compares the distribution change between the anchored window W_a and the new incoming current window W_c (see Fig. 3). The anchored window is considered a true concept for the subsequent drift detection because the successive current windows may have some distorted information. As a result, the false drift is encountered by the detector. So, we compare each new incoming current window with the anchored window to find a better change in concept until it reaches the drift level, i.e., the actual drift is detected. After that, the old concept's anchored window data information removes from the buffer space, and the new concept window considers as a new anchored window. In this way, new incoming current windows W_c compare with new anchored window W_a. This process repeats until the stream gets exhausted.

4.2.3 Model Update Phase

In a streaming environment, the change in data distribution over time may result in the inaccurate prediction of the learning model. Sometimes, the other deformities like noise (additional meaningless information) and blips (sudden and short change in concept) are also present in the data stream. Due to these deformities, the concept drift detection methods misinterpret an adversarial change in data as a drift. Such a type of drift is considered false drift. In this case, an increase in false drift decreases the learner's accuracy and other performance measures. Hence, it is essential to find the actual drift in incoming data patterns and incorporate it with the classification model. The model adaptation and forgetting mechanism are performed as defined in Algorithm 1. The process of updating the learning model as per the change in the distribution of current data instances is considered a model adaptation, whereas removing unuseful old information to accommodate new data information is known as a forgetting mechanism.

In order to find actual drift and update the learning model accordingly, the proposed method considers two parameters, namely drift_count and Flag. These parameters use to set the warning level, drift level, and dynamic window size. The introduced parameter Flag is used to restrict the event like noise and blips, i.e., it neglects the small changes in the data stream. The drift_count is used to count the number of drift. Here, the warning level is signaled when the value of drift_count and Flag is less than the threshold; otherwise, the drift level is indicated. The warning level signifies that the drift may occur or false drift is detected. At the same time, the actual drift is detected in the drift level. The condition for both levels is defined in Eqs. 6 and 7, respectively.

$$\text{Condition for warning level}: (\text{drift\_count} < \theta)\&\&(\text{Flag} < \theta) \tag{6}$$

$$\text{Condition for drift level}: (\text{drift\_count} \geq \theta)\&\&(\text{Flag} \geq \theta) \tag{7}$$

In the case of warning level (see Eq. 8), the window size is set to three-by-fourth (or α = 3/4) of the current window size, and in drift level (see Eq. 9), the window size is half (or

β = 1/2) of the current window size.

$$\text{Warning level window size}: W_c = \alpha * W_c \quad (8)$$

$$\text{Drift level window size}: W_c = \beta * W_c \quad (9)$$

As per the above conditions, Table 3 exhibits a scenario of actual drift detection by DBDDM. In this scenario, the parameters initialize at time-stamp t. In the case of drift, the drift_count and Flag increment by one, and the window size is set to warning level (see Eq. 8). Whereas there is no change in drift_count and window size, a decrement occurs in Flag (till Flag > 0; otherwise, Flag remains set to 0) in case of no drift condition. Each time drift_count and Flag compare with the threshold (θ) to define the significance levels.

The warning level shows a lower confidence level and considers drift may have happened. Warning level verifies the actual drift or false drift. In this way, the actual drift is detected whenever the drift level is reached (see Eq. 7), and then the resetting of the parameter's value is performed. Finally, the method retrains the learning model as per the new concept in the data stream.

5 Experimental Analysis

This section illustrates datasets description (Sect. 5.1), and experimental environment and parameter setting (Sect. 5.2). Further, ablation studies are discussed in Sect. 5.2.1.

5.1 Datasets

For experimental analysis purposes, the methods are evaluated using four synthetic and four real-time datasets (see Table 4). The datasets contain binary as well as multi-class target values. For analysis purposes, all attribute values are taking into consideration.

5.1.1 Synthetic Datasets

- *LED dataset*: It uses to predict digits that are shown in the LED display of the seven-segment. This multivariate dataset has 10% noise. It has 24 attributes, which are categorical data. The LED display contains the representation of the attributes in the form of 0 or 1. It shows whether the reciprocal light is on. The 10% noise represents that there is a 10% probability of inverted value for each attribute vector. The change in attributes value denotes a drift.
- *SINE dataset*: There are two contexts in the dataset, i.e., Sine1, where $y_i = sin(x_i)$ and Sine2, where $y_i = 0.5 + 0.3 * sin(3\pi x_i)$. The concept drift is detected by reversing the condition of above context.
- *Agrawal dataset*: The dataset contains information about people who want to take the loan. They classify into group A and group B. The dataset has attributes like age, salary, education level, house value, zip code, etc. It has ten functions, but only five functions use to generate the dataset. The attribute value is numeric as well as nominal. Here, the concept drift occurs abruptly and gradually.

5.1.2 Real Time Datasets

- *Airlines dataset*: The dataset has two target values. It determines whether there is a flight delay. The analysis is based on attributes like flight, airport to and from, time, days of the week, and length.
- *Spam Assassin dataset*: The dataset has 500 attributes based on e-mail messages. All attributes values are binary. It indicates whether a word is present in the e-mail. It depicts that does a gradual change occurs in spam messages with time?
- *Forest cover dataset*: The dataset covers 30 $*$ 30 m cells in the area of US Forest Service (USFS), Region 2. It has 54 attributes, where 44 attributes are binary values, and 10 attributes are numerical values. It illustrates various features like elevation, vegetation appearances, disappearances, etc. It is a normalized dataset.
- *Usenets dataset*: Dataset combines usenet1 and usenet2 to build new dataset, i.e., Usenets. It is a collection of twenty news groups. The user sequentially labels the messages according to their interest. There are 99 attributes in both datasets.

5.2 Experimental Environment and Parameter Setting

DBDDM build with the Scikit-multiflow framework, a machine learning package for streaming data in Python. For experimental purposes, the window size is taken as 1000, and the alpha level is set to 0.05. Here, alpha level is a significance level, and represented as α. It shows the probability of rejecting the H_o, when it is true. The proposed method uses a significance level of 0.05, which specifies a 5% risk of deducing that a difference exists when no actual difference is present. The Naive Bayes (NB) and Hoeffding Tree (HT) are used as base classifiers for evaluation. NB is a probabilistic classifier, which is based on Bayes' theorem. It is a simple and most effective classifier. At the same time, HT is an incremental decision tree and learns from massive data streams. The experimental parameter settings, i.e., window size, alpha level, and classifiers, are the same for all the compared methods.

Table 3 Scenario of actual drift detection in DBDDM

Time-stamps	Parameters			Drift	No drift
	Drift_count	Flag	Current window size (W_c)		
t	0	0	No change	–	Y
$t + 1$	1	1	Warning level	Y	–
$t + 2$	2	2	Warning level	Y	–
$t + 3$	2	1	No change	–	Y
$t + 4$	2	0	No change	–	Y
$t + 5$	2	0	No change	–	Y
$t + 6$	3	1	Warning level	Y	–
$t + 7$	4	2	Warning level	Y	–
$t + 8$	5	3	Warning level	Y	–
$t + 9$	6	4	Warning level	Y	–
$t + 10$	7	5	Drift level	Y	–
Resetting of parameters					
t+11	0	0	No change	–	Y

Table 4 Datasets description

Datasets	Number of attributes	Number of examples	Target classes
LED	24	20,000	10
Sine	2	20,000	2
Agrawal (Abr)	9	20,000, 50,000, 100,000	2
Agrawal (Grad)	9	20,000, 50,000, 100,000	2
Airlines	7	539,383	2
Forest cover	54	581,012	2
Spam Assassin	500	9324	2
Usenets	99	3000	2

5.2.1 Ablation Studies

The ablation studies on the hyperparameters are introduced with the proposed approach specifically in the current window size W_c and number of shuffling performed in the concatenated window data instances W_{con}. The various window size W_c is taken into consideration, where $W_c = \{100, 250, 500, 750, 1000\}$ and the different values of $\text{len}(W_{con})$ are taken as $\text{len}(W_{con})/2$, $\text{len}(W_{con})$, $\text{len}(W_{con})*2$, $\text{len}(W_{con})*3$, $\text{len}(W_{con})*4$. As a result, the high performing values are considered for the experimental analysis.

The $\text{len}(W_{con})$ is related to random test for concept drift detection. A random test is performed for concept drift detection. In this test, the considerable distribution changes between two windows are determined by shuffling the windows data instances. When we shuffle the data instances during the test, we see all possible behavior of windows data instances (which can become quite large, i.e., $\text{len}(W_{con})!$ times). Still, it creates an additional burden for computation and requires high memory. Hence, while a random test requires that we see all possible shuffled instances, we perform 'approximate shuffled tests' by conducting many resamples. Thus, the parameter setting is requisite to obtain the highest performance of the proposed method with lowering the computational complexity. In Tables 5 and 6, we demonstrate the behavior of the proposed method with NB and HT Classifier for different values of $\text{len}(W_{con})$ by considering two parameters the accuracy and number of drift detected. Tables 5 and 6 show that the less permutation (or test case) gives low performance in terms of accuracy because fewer combinations of samples are available for the analysis. In comparison, more number of permutations requires more memory and computation time. In addition to this, it shows the decline in performance in terms of accuracy. When we perform shuffling $\text{len}(W_{con} * 2)$ times, it offers the best performance and comparatively requires less memory and time with both the classifiers. In addition to this, Tables 7 and 8 show that the increment in window size is directly proportional to the learning model's performance, and window size W_c =1000 gives better performance with most datasets. For brevity, beyond 1000, window size is not shown in the table. Thus, the performance of DBDDM is superior with W_c = 1000 and $\text{len}(W_{con} * 2)$ times shuffling of concatenated window data instance for almost all the datasets. So, it is taken into consideration for evaluation purposes.

In addition to this, we consider two parameters, i.e., Flag and drift_count, for drift detection. The threshold (θ) for both parameters is limited to five. The current window size is set according to two significant levels, i.e., three-by-forth and a half for warning level and drift level, respectively. The size of window data instances is restricted for improvement in drift detection. The forgetting mechanism also uses to remove the old information from buffer space.

6 Result Evaluation

This section discusses experimental results and analyses (Sect. 6.1), and statistical comparison of methods (Sect. 6.2).

6.1 Experimental Results and Analyses

The experiment performs with synthetic and real-time datasets. In synthetic datasets, there are variations in the size of the dataset and induced drift types. The proposed method DBDDM compares with state-of-the-art methods using Naive Bayes and Hoeffding Tree classifier. The mean accuracy (see Eq. 1) of each window is used to find classification accuracy (or average mean accuracy) at the end of the data stream as defines in Eq. 10.

$$\text{Classification accuracy} = \frac{\sum_{i=1}^{n} (\text{Mean accuracy})_i}{\text{No. of Data chunks}} \tag{10}$$

Here, No. of Data Chunks describe the number of partitions of the overall data stream. For experimentation, each dataset is divided into thirty chunks to calculate classification accuracy.

The performance results of the proposed method using the Naive Bayes classifier in terms of the classification accuracy on datasets are demonstrated in Table 9 and discussed as follows. In the case of the LED gradual drift dataset, the classification accuracy of the proposed method is around 2% more compared to the highest performing detector. For finding the gradual drift in the Sine dataset, it shows a decline in accuracy around 1.5%. At the same time, three Agrawal datasets for gradual drift encompass 20 K, 50 K, 100 K data instances and exhibit approx 24%, 24%, and 22% increase in accuracy, respectively. The other three Agrawal datasets for abrupt drift enclose 20 K, 50 K, 100 K data examples and manifest near 9%, 19%, and 18% increase in accuracy, respectively. For Airlines and Usenets datasets, it has a marginal decrease in accuracy from the compared methods. It has around a 12% increase in accuracy for the Forest cover dataset, and there is a marginally increase in classification accuracy for the Spam Assassin dataset.

The behavior of the proposed method using the Hoeffding base classifier in terms of classification accuracy (see Table 10) is as follows. For finding the gradual drift in LED and Sine datasets, it exhibits a significant increase in accuracy. Around 28%, 22%, and 20% increase in accuracy demonstrated in the Agrawal dataset for gradual drift using 20K, 50K, and 100K data examples, respectively. The other three Agrawal datasets for abrupt drift enclose 20K, 50K, and 100K data examples and reveal 7%, 15%, and 10% increase in accuracy, respectively. The marginally decrease in accuracy is seen for Airlines, Spam Assassin, and Usenets datasets. The Forest cover dataset exhibits around a 20% increase in accuracy. The above observations show that the performance of the proposed method is better with synthetic and real-time datasets using both classifiers.

6.2 Statistical Comparison of Methods

To verify the statistical significance of the performance of DBDDM and the compared methods using NB and HT classifier, we utilize *Friedman* test with *Nemenyi-post-hoc* analysis ([43]). *Friedman* test is based on the null hypothesis H_0, which defines that the equivalent methods share the same rank. In this test, we compare eight methods using twelve datasets. The methods are arranged in order best to worst as per their classification accuracy and assigned a rank (from 1 to k). If the classification accuracy of methods is similar, the average of their ranks assigns to them. The average rank of methods demonstrates in Fig. 4. As a result, the top rank method is the best performing method with overall datasets.

In case of rejection of H_0, *Nemenyi-post-hoc* analysis is conducted. It defines that the performance of the two methods is considerably different if the corresponding average ranks differ by at least critical difference (CD). The following equation computes CD:

$$\text{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{11}$$

Here, q_α, k, and N denote critical value, no. of methods, and no. of datasets, respectively. The value of critical difference is obtained as 3.031 by equation 11. This test suggests that DBDDM is a top-ranked method. It is significantly superior to DDM, ECDD with NB classifier (see Fig. 5), and ADWIN, ECDD, SEED, and SEQDRIFT2 with HT classifier (see Fig. 6).

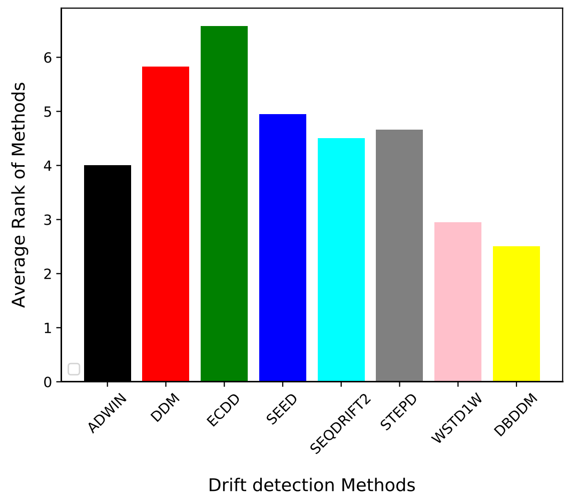
7 Conclusion

The paper proposes DBDDM, an adaptive disposition-based concept drift detection method. Here, we utilize the approx-

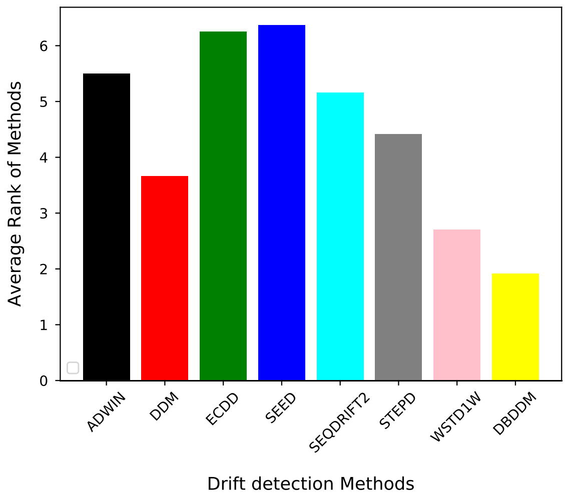
Table 5 Classification accuracy and number of drift detected as per different values of len(W_{con}) based on NB classifier

Dataset	Parameters	len(W_{con})/2	len(W_{con})	len($W_{con} * 2$)	len($W_{con} * 3$)	len($W_{con} * 4$)
LED (Grad-20K)	Accuracy	73.33	73.24	**73.58**	73.33	73.33
	No. of drift detected	1	1	2	1	1
Sine (Grad-20K)	Accuracy	83.33	83.26	**83.36**	83.13	83.04
	No. of drift detected	70	46	64	76	72
Agrawal (Abr-20K))	Accuracy	72.81	73.37	**74.94**	72.05	72.87
	No. of drift detected	19	70	19	73	11
Agrawal (Abr-50K)	Accuracy	80.90	83.47	**85.96**	83.12	84.76
	No. of drift detected	96	92	74	114	77
Agrawal (Abr-100K)	Accuracy	79.84	80.21	84.81	**85.61**	82.8
	No. of drift detected	49	76	56	81	62
Agrawal (Grad-20K)	Accuracy	84.92	84.53	**88.38**	85.90	79.71
	No. of drift detected	25	28	24	21	89
Agrawal (Grad-50K)	Accuracy	82.39	83.59	**86.61**	78.72	79.36
	No. of drift detected	78	96	88	79	69
Agrawal (Grad-100K)	Accuracy	85.56	82.16	**88.09**	83.74	87.90
	No. of drift detected	72	83	51	89	94
Airlines	Accuracy	61.77	61.78	**61.88**	61.64	61.80
	No. of drift detected	19	17	18	28	18
Forest cover	Accuracy	77.98	78.38	**80.21**	78.86	77.86
	No. of drift detected	19	17	28	18	15
Spam Assassin	Accuracy	91.64	90.57	**91.64**	**91.64**	91.63
	No. of drift detected	0	1	1	0	0
Usenets	Accuracy	70.82	68.74	**71.59**	69.01	69.38
	No. of drift detected	20	18	13	11	16

Bold signifies the highest accuracy of the proposed method DBDDM with a particular dataset using NB classifier by observing the different shuffling of concatenated window data instances



(a) Average rank of methods with NB Classifier



(b) Average rank of methods with HT Classifier

Fig. 4 Average rank of different methods based on Friedman test using **a** NB and **b** HT classifiers

Table 6 Classification accuracy and number of drift detected as per different values of $\text{len}(W_{\text{con}})$ based on HT Classifier

Dataset	Parameters	$\text{len}(W_{\text{con}})/2$	$\text{len}(W_{\text{con}})$	$\text{len}(W_{\text{con}}*2)$	$\text{len}(W_{\text{con}}*3)$	$\text{len}(W_{\text{con}}*4)$
LED (Grad-20K)	Accuracy	73.12	73.33	**73.67**	73.04	73.37
	No. of drift detected	1	1	1	3	1
Sine (Grad-20K)	Accuracy	86.41	89.51	**92.27**	88.85	85.07
	No. of drift detected	39	43	72	88	63
Agrawal (Abr-20K))	Accuracy	70.36	70.89	72.66	**72.75**	71.82
	No. of drift detected	21	27	32	57	16
Agrawal (Abr-50K)	Accuracy	82.26	83.74	**85.75**	84.50	84.41
	No. of drift detected	105	77	74	70	69
Agrawal (Abr-100K)	Accuracy	78.07	82.62	**83.37**	83.26	84.16
	No. of drift detected	78	85	83	60	50
Agrawal (Grad-20K)	Accuracy	87.38	90.50	**93.68**	89.38	87.04
	No. of drift detected	63	24	27	57	17
Agrawal (Grad-50K)	Accuracy	79.28	87.03	**91.34**	82.87	84.84
	No. of drift detected	72	88	67	57	64
Agrawal (Grad-100K)	Accuracy	86.13	89.99	**93.61**	91.69	84.06
	No. of drift detected	57	75	92	80	74
Airlines	Accuracy	62.92	63.07	**63.12**	63.08	63.06
	No. of drift detected	15	19	19	20	21
Forest cover	Accuracy	87.51	87.53	**88.88**	86.63	87.64
	No. of drift detected	19	19	18	20	21
Spam Assassin	Accuracy	90.31	90.32	**91.34**	90.10	90.10
	No. of drift detected	0	1	1	0	0
Usenets	Accuracy	67.44	69.58	**72.16**	72.05	70.15
	No. of drift detected	12	12	13	13	16

Bold signifies the highest accuracy of the proposed method DBDDM with a particular dataset using HT classifier by observing the different shuffling of concatenated window data instances

Table 7 Classification accuracy as per different values of current window (W_c) based on NB classifier

Dataset	Parameters	$W_c = 100$	$W_c = 250$	$W_c = 500$	$W_c = 750$	$W_c = 1000$
LED (Grad-20K)	Accuracy	69.49	71.23	73.35	73.24	**73.58**
Sine (Grad-20K)	Accuracy	83.46	83.49	**83.51**	82.99	83.36
Agrawal (Abr-20K)	Accuracy	74.64	70.48	67.79	66.16	**74.94**
Agrawal (Abr-50K)	Accuracy	84.63	87.32	**88.58**	88.32	85.96
Agrawal (Abr-100K)	Accuracy	86.94	87.62	87.88	87.77	**88.11**
Agrawal (Grad-20K)	Accuracy	87.13	87.25	87.97	88.34	**88.38**
Agrawal (Grad-50K)	Accuracy	85.58	88.02	87.88	**88.54**	88.38
Agrawal (Grad-100K)	Accuracy	85.68	88.20	87.32	88.04	**88.09**
Airlines	Accuracy	61.54	62.09	**62.15**	61.97	61.88
Forest cover	Accuracy	**82.68**	81.13	79.62	79.09	80.21
Spam Assassin	Accuracy	90.38	91.48	91.38	91.17	**91.64**
Usenets	Accuracy	68.81	68.42	66.21	69.47	**71.59**

Bold signifies the highest accuracy of the proposed method DBDDM with a particular dataset using NB classifier by observing the different values of the current window

Table 8 Classification accuracy as per different values of current window (W_c) based on HT Classifier

Dataset	Parameters	$W_c = 100$	$W_c = 250$	$W_c = 500$	$W_c = 750$	$W_c = 1000$
LED (Grad-20K)	Accuracy	71.23	72.08	72.65	73.20	**73.67**
Sine (Grad-20K)	Accuracy	89.43	87.89	89.90	91.19	**92.27**
Agrawal (Abr-20K)	Accuracy	73.45	**75.20**	72.75	73.05	72.66
Agrawal (Abr-50K)	Accuracy	90.18	91.18	92.69	92.26	**94.14**
Agrawal (Abr-100K)	Accuracy	87.01	92.47	90.28	92.16	**93.34**
Agrawal (Grad-20K)	Accuracy	89.39	91.73	91.93	92.12	**93.68**
Agrawal (Grad-50K)	Accuracy	84.92	89.76	**92.39**	92.27	92.34
Agrawal (Grad-100K)	Accuracy	89.53	90.92	90.98	**94.00**	93.61
Airlines	Accuracy	62.00	62.31	62.59	62.88	**63.12**
Forest cover	Accuracy	84.87	85.71	86.38	86.47	**88.88**
Spam Assassin	Accuracy	89.43	90.55	90.28	90.38	**91.34**
Usenets	Accuracy	65.42	67.86	68.56	70.47	**72.16**

Bold signifies the highest accuracy of the proposed method DBDDM with a particular dataset using HT classifier by observing the different values of the current window

Table 9 Comparison of classification accuracy between proposed method and existing methods using NB classifier

Dataset	ADWIN	DDM	ECDD	SEED	SEQDRIFT2	STEPD	WSTD1W	DBDDM
LED (Grad-20K)	63.99	71.57	68.57	56.97	61.99	67.42	71.45	**73.58**
Sine (Grad-20K)	84.18	**84.84**	84.32	84.12	84.57	84.55	84.76	83.36
Agrawal (Abr-20K))	64.26	63.67	62.42	64.25	64.04	64.56	64.73	**74.94**
Agrawal (Abr-50K))	65.64	64.27	62.93	65.48	65.65	65.27	65.71	**85.96**
Agrawal (Abr-100K)	66.08	64.85	62.97	66.02	66.14	65.48	65.79	**84.81**
Agrawal (Grad-20K)	63.25	63.13	61.98	63.64	62.63	63.54	63.56	**88.38**
Agrawal (Grad-50K)	65.34	64.49	62.64	65.19	65.41	64.91	65.31	**86.61**
Agrawal (Grad-100K)	65.93	64.69	62.85	65.86	65.96	65.25	65.57	**88.09**
Airlines	66.70	65.35	63.66	**66.71**	66.60	65.73	**66.71**	61.88
Forest cover	67.73	67.14	67.39	67.32	67.68	67.62	68.18	**80.21**
Spam Assassin	**91.87**	89.34	88.39	90.90	89.70	91.42	91.80	**91.64**
Usenets	68.41	71.01	**72.75**	68.65	66.31	71.95	71.58	71.59
Average rank	4.00	5.83	6.58	4.95	4.50	4.66	2.95	**2.50**

Bold signifies the highest accuracy of a method with a particular dataset using NB classifier

Table 10 Comparison of classification accuracy between proposed method and existing methods using HT classifier

Dataset	ADWIN	DDM	ECDD	SEED	SEQDRIFT2	STEPD	WSTD1W	DBDDM
LED (Grad-20K)	63.09	70.73	67.59	55.60	60.97	65.64	69.06	**73.67**
Sine (Grad-20K)	85.54	86.82	85.22	85.49	86.69	86.01	86.79	**92.27**
Agrawal (Abr-20K))	64.50	65.21	64.27	64.64	64.63	65.27	65.76	**72.66**
Agrawal (Abr-50K)	65.88	70.01	65.40	65.39	67.29	66.47	69.89	**85.75**
Agrawal (Abr-100K)	66.63	73.09	66.96	65.71	68.70	67.06	70.52	**83.37**
Agrawal (Grad-20K)	63.62	65.27	63.26	63.84	63.37	64.26	64.98	**93.68**
Agrawal (Grad-50K)	65.48	69.20	65.69	65.04	66.83	66.02	68.58	**91.34**
Agrawal (Grad-100K)	66.35	73.48	66.43	65.47	68.49	66.89	70.22	**93.61**
Airlines	66.70	65.35	63.66	**66.71**	66.60	65.73	**66.71**	63.12
Forest cover	67.73	67.14	67.39	67.32	67.68	67.62	68.18	**88.88**
Spam Assassin	**91.87**	89.34	88.39	90.90	89.70	91.42	91.80	91.34
Usenets	68.41	71.01	**72.75**	68.65	66.31	71.95	71.58	72.16
Average rank	5.50	3.66	6.25	6.37	5.16	4.41	2.70	**1.91**

Bold signifies the highest accuracy of a method with a particular dataset using HT classifier

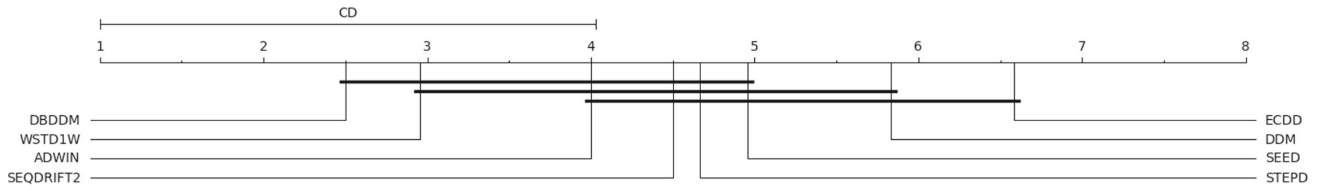


Fig. 5 Critical distance (CD) diagram based on classification accuracy of methods with NB classifier (see Table 9)

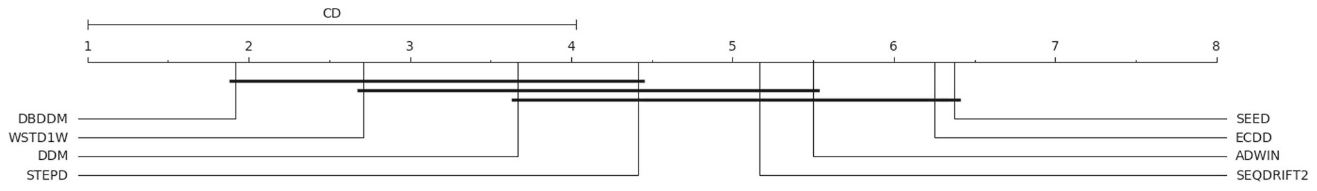


Fig. 6 Critical distance (CD) diagram based on classification accuracy of methods with HT classifier (see Table 10)

imate randomization test to determine the frequency of consecutive drift and compare the obtained frequency with the threshold to determine the actual drift. The approximate random test is performed to find a significant difference between window data instances. Due to deformities in the stream, the learning model may encounter false drifts, which is a delusion of drift, not an actual drift. The parameter Flag is introduced to count the frequency of consecutive drift. The frequency restricts the false drift situations and efficiently detects the actual drift. The proposed work detects sudden and gradual drift incrementally as per the reported experiments. In the real-time environment, the DBDDM also deals with multi-dimensionality of data as well as binary-class and multi-class classification problems.

The proposed method is compared with seven state-of-the-art concept drift detection methods using synthetic and real-time datasets. DBDDM is the top-ranked method in terms of classification accuracy with Naive Bayes (NB) and Hoeffding Tree (HT) base classifier in the reported experiments. A statistical significance test, $Friedman$ test with $Nemenyi$-$post$-hoc analysis, is performed with proposed and existing methods. It shows that the DBDDM is significantly better than DDM, ECDD with NB classifier, and ADWIN, ECDD, SEED, and SEQDRIFT2 with HT classifier. We have conducted ablation studies to understand the impact of the change in current window size and the number of possible shuffling of concatenated window data instances. Interestingly, the smaller window size, less shuffling, and more shuffling of concatenated window data instances degrade the learning model's accuracy.

In future work, DBDDM can be utilized in different domains of applications. Further, the adaptive window size and threshold identification can be achieved for automatic parameter settings instead of a fixed value of parameters. The other future direction is that the handling of concept drifts under missing values, which depicts the incompleteness of features. On the other hand, the incremental regularization or dimensionality reduction techniques can be applied with the concept drift detection method to minimize the computation resources in terms of memory and time.

References

1. Gama, J.; Žliobaite, I.: A survey on concept drift adaptation. In: 2015 International Joint Conference on Neural Networks (IJCNN). ACM Computing Survey, pp. 1–37 (2014)
2. Lu, J.; Liu, A.; Song, Y.; Zhang, G.: Data-driven decision support under concept drift in streamed big data. Complex Intell. Syst. **6**(1), 157–163 (2020)
3. de Barros, R.S.M.; Hidalgo, J.I.G.; de Lima Cabral, D.R.: Wilcoxon rank sum test drift detector. Neurocomputing **275**, 1954–1963 (2018)
4. Frías-Blanco, I.; del Campo-Ávila, J.; Ramos-Jimenez, G.; Morales-Bueno, R.; Ortiz-Díaz, A.; Caballero-Mota, Y.: Online and non-parametric drift detection methods based on Hoeffding's bounds. IEEE Trans. Knowl. Data Eng. **27**(3), 810–823 (2014)
5. Shao, J.; Ahmadi, Z.; Kramer, S.: Prototype-based learning on concept-drifting data streams. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 412–421 (2014)
6. Lu, N.; Lu, J.; Zhang, G.; De Mantaras, R.L.: A concept drift-tolerant case-base editing technique. Artif. Intell. **230**, 108–133 (2016)
7. 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)
8. Korycki, Ł.; Krawczyk, B. Adversarial concept drift detection under poisoning attacks for robust data stream mining (2020). arXiv preprint arXiv:2009.09497
9. Hulten, G.; Spencer, L.; Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 97–106 (2001)
10. Masud, M.M.; Gao, J.; Khan, L.; Han, J.; Thuraisingham, B.: A multi-partition multi-chunk ensemble technique to classify concept-drifting data streams. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp. 363–375 (2009)

11. Abdulsalam, H.; Skillicorn, D.B.; Martin, P.: Classification using streaming random forests. IEEE Trans. Knowl. Data Eng. **23**(1), 22–36 (2010)
12. Yu, S.; Abraham, Z.; Wang, H.; Shah, M.; Wei, Y.; Príncipe, J.C.: Concept drift detection and adaptation with hierarchical hypothesis testing. J. Frank. Inst. **356**(5), 3187–3215 (2019)
13. Brzezinski, D.; Stefanowski, J.: Reacting to different types of concept drift: the accuracy updated ensemble algorithm. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 81–94 (2013)
14. Pesaranghader, A.; Viktor, H.L.: Fast Hoeffding drift detection method for evolving data streams. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 96–111 (2016)
15. Agrahari, S.; Singh, A.K.: Concept drift detection in data stream mining: a literature review. J. King Saud Univ. Comput. Inf. Sci. (2021). https://doi.org/10.1016/j.jksuci.2021.11.006
16. Gama, J.; Medas, P.; Castillo, G.; Rodrigues, P.: Learning with drift detection. In: Brazilian Symposium on Artificial Intelligence, Springer, pp. 286–295 (2004)
17. Gama, J.; Castillo, G.: Learning with local drift detection. In: International Conference on Advanced Data Mining and Applications, Springer, pp. 42–55 (2006)
18. Nishida, K.: Learning and Detecting Concept Drift. Information Science and Technology (2008)
19. Liu, A.; Song, Y.; Zhang, G.; Lu, J.: Regional concept drift detection and density synchronized drift adaptation. In: IJCAI International Joint Conference on Artificial Intelligence (2017)
20. Liu, A.; Zhang, G.; Lu, J.: Fuzzy time windowing for gradual concept drift adaptation. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, pp. 1–6 (2017)
21. Baena-Garcıa, M.; del Campo-Ávila, J.; Fidalgo, R.; Bifet, A.; Gavalda, R.; Morales-Bueno, R.: Early drift detection method. Fourth Int. Workshop Knowl. Discov. Data Streams **6**, 77–86 (2006)
22. Ross, G.J.; Adams, N.M.; Tasoulis, D.K.; Hand, D.J.: Exponentially weighted moving average charts for detecting concept drift. Pattern Recognit. Lett. **33**(2), 191–198 (2012)
23. Bifet, A.: Adaptive learning and mining for data streams and frequent patterns. ACM SIGKDD Explor. Newsl. **11**(1), 55–56 (2009)
24. Bifet, A.; Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, pp. 443–448 (2007)
25. Huang, D.T.J.; Koh, Y.S.; Dobbie, G.; Pears, R.: Detecting volatility shift in data streams. In: 2014 IEEE International Conference on Data Mining, pp. 863–868. https://doi.org/10.1109/ICDM.2014.50 (2014)
26. Pears, R.; Sakthithasan, S.; Koh, Y.S.: Detecting concept change in dynamic data streams. Mach. Learn. **97**(3), 259–293 (2014)
27. Raza, H.; Prasad, G.; Li, Y.: EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments. Pattern Recognit. **48**(3), 659–669 (2015)
28. Alippi, C.; Boracchi, G.; Roveri, M.: Hierarchical change-detection tests. IEEE Trans. Neural Netw. Learn. Syst. **28**(2), 246–258 (2016)
29. Yu, S.; Abraham, Z.: Concept drift detection with hierarchical hypothesis testing. In: Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, pp. 768–776 (2017)
30. Miyata, Y.; Ishikawa, H.: Concept drift detection on data stream for revising DBSCAN cluster. In: Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics, pp. 104–110 (2020)
31. Gu, F.; Zhang, G.; Lu, J.; Lin, C.T.: Concept drift detection based on equal density estimation. In: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 24–30 (2016)
32. Song, X.; Wu, M.; Jermaine, C.; Ranka, S.: Statistical change detection for multi-dimensional data. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 667–676 (2007)
33. 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 (2016)
34. Bu, L.; Zhao, D.; Alippi, C.: An incremental change detection test based on density difference estimation. IEEE Trans. Syst. Man Cybern. Syst. **47**(10), 2714–2726 (2017)
35. Nishida, K.; Yamauchi, K.: Detecting concept drift using statistical testing. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) Discovery Science, pp. 264–269. Springer, Berlin (2007)
36. Mahdi, O.A.; Pardede, E.; Ali, N.: A hybrid block-based ensemble framework for the multi-class problem to react to different types of drifts. Cluster Comput. **24**(3), 2327–2340 (2021)
37. Mahdi, O.A.; Pardede, E.; Ali, N.: kappa as drift detector in data stream mining. Procedia Comput. Sci. **184**, 314–321 (2021)
38. Mehmood, H.; Kostakos, P.; Cortes, M.; Anagnostopoulos, T.; Pirttikangas, S.; Gilman, E.: Concept drift adaptation techniques in distributed environment for real-world data streams. Smart Cities **4**(1), 349–371 (2021)
39. Heusinger, M.; Schleif, F.M.: reactive concept drift detection using coresets over sliding windows. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, pp. 1350–1355 (2020)
40. Misra, S.; Biswas, D.; Saha, S.K.; Mazumdar, C.: Applying Fourier inspired windows for concept drift detection in data stream. In: 2020 IEEE Calcutta Conference (CALCON), IEEE, pp. 152–156 (2020)
41. Mahdi, O.A.; Pardede, E.; Ali, N.; Cao, J.: Diversity measure as a new drift detection method in data streaming. Knowl. Based Syst. **191**, 105227 (2020)
42. Sakthithasan, S.; Pears, R.; Koh, Y.S.: One pass concept change detection for data streams. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 461–472. Springer, Berlin (2013)
43. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)