Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(11): 6577–6609.
DOI: 10.3934/era.2025291
Received: 25 August 2025
Revised: 28 October 2025
Accepted: 31 October 2025
Published: 05 November 2025

Research article

AD-DES: An adaptive dual dynamic ensemble selection for imbalanced data streams

Ziyan Mo[1], Li Deng[1,*], Bo Wei[2,3,*], Jiakai Chen[2] and Aixi Chen[1]

[1] School of Science, Zhejiang Sci-Tech University, Hangzhou 310018, China
[2] School of Computer Science and Technology (School of Artificial Intelligence), Zhejiang Sci-Tech University, Hangzhou 310018, China
[3] Longgang Research Institute, Zhejiang Sci-Tech University, Longgang 325000, China

* **Correspondence:** Email: lideng75@zstu.edu.cn, weibo@zstu.edu.cn.

Abstract: Classification of imbalanced data streams in nonstationary environments poses a significant challenge in machine learning. Ensemble learning has demonstrated efficacy in managing imbalanced data streams and concept drift. However, most existing methods develop different strategies for each aspect, overlooking their relationships and interactions, which hinders the expected performance of these strategies. To address this issue, an adaptive dual dynamic ensemble selection (AD-DES) method was proposed for classifying imbalanced data streams with concept drift. First, an adaptive equalization resampling (AER) strategy was proposed to obtain balanced data chunks, which can help to reduce the risk of overfitting or insufficient sampling caused by excessive data imbalance. Following that, the data chunk obtained by the AER strategy was stored to balance the subsequent ones. Second, a dual dynamic ensemble selection (D-DES) strategy was introduced to perform two rounds of selection in the classifier pool to obtain the optimal ensemble model. Finally, an adaptive drift detector in AD-DES was integrated, which is beneficial for the model to adjust to newly emerging concepts in nonstationary environments. The research findings demonstrated that AD-DES outshines 9 comparison algorithms in terms of classification precision and robustness across 10 synthetic datasets and 5 real-world datasets featuring diverse forms of concept drift.

Keywords: ensemble classifier; class imbalance; concept drift; data streams; ensemble learning

1. Introduction

Classification of imbalanced data streams in nonstationary environments is a fundamental yet challenging task in machine learning [1]. Nonstationary environments, characterized by constantly changing data distributions, pose significant challenges for the development and deployment of predictive

models [2]. One of the most prevalent forms of such challenges is concept drift, characterized by the target variable's generation distribution evolving over time [3]. The occurrence of concept drift may render predictive models based on old data outdated, thereby resulting in poor performance on the new data and necessitating continuous adaptation to maintain their accuracy. Another critical challenge is class imbalance, characterized by a significant disparity in the number of instances between the minority and majority classes [4]. The presence of class imbalance may cause predictive models to favor the latter, leading to less attention on the former. What's more, the problem becomes more complex when they both occur simultaneously [5].

To cope with concept drift, researchers have explored a range of adaptation techniques, which typically fall into two categories: passive and active detection approaches [3,6]. Passive detection approaches rely on the dynamic adjustment of ensemble models to adapt to concept drift, without actively monitoring changes in real-time [7]. In this way, passive detection approaches often result in delayed responses to concept drift. In contrast, active detection approaches continuously monitor the data distributions, enabling the models to adjust accordingly before their performance deteriorates significantly [8–10]. However, setting predefined thresholds for active detection approaches can be challenging, because inappropriate parameter settings may lead to missed drift events or excessive false positives.

To deal with class imbalance, various methods have been introduced successively, such as oversampling and undersampling [11]. These methods can improve classifier effectiveness on imbalanced data streams to some extent. However, they often suffer from two issues: overfitting and underfitting. The primary cause of these issues is the highly imbalanced data streams, which make the resampled data chunks generated by a single preprocessing method poorly reflect the original data distribution. In addition, theoretically, it is helpful for different data chunks to adaptively select the optimal preprocessing methods. Therefore, developing a preprocessing strategy is crucial for solving this problem. This strategy can automatically adjust resampling strategies in response to changes in class distribution within the data streams.

For the classification of imbalanced data streams with concept drift, the common method integrates ensemble learning algorithms with data preprocessing approaches [12]. As a representative of ensemble learning algorithms, dynamic ensemble selection (DES) selects an optimal subset of base classifiers tailored to the specific local region (competence subregion) of the test instances, rather than using the entire set of base classifiers [13]. Considering its dynamic adjustment characteristics, several DES-based approaches have been proposed successively and achieved better results [14]. However, most existing DES-based algorithms have two main shortcomings. First, they are mostly independent of other strategies and tend to overlook the subtle relationships between the DES strategy and the data resampling strategy. For example, DES-ICD [15] employs oversampling to balance data chunks but lacks discussion on how the resampling process might affect the DES strategy. Similarly, DSCB [11] combines undersampling, oversampling, and weighted bagging to handle class imbalance but lacks an explicit mechanism to coordinate resampling with ensemble adaptation. Such decoupled designs may restrict the overall adaptability of the models under nonstationary and highly imbalanced environments. Second, most existing DES-based algorithms have not fully explored competence subregions across different scenarios, potentially leading to less accurate classifier selection when the data distribution is highly complex or non-uniform. As a result, the generalization ability and prediction performance of these models may be negatively affected.

1.1. Objective and contributions

To cope with the two aforementioned shortcomings, an adaptive dual dynamic ensemble selection (AD-DES) method is proposed for binary classification of imbalanced data streams in nonstationary environments. The main contributions of this paper can be summarized as follows:

1) The adaptive equalization resampling (AER) strategy, which can be adaptively adjusted based on the imbalance ratio (IR) of the dataset, is proposed to obtain balanced data chunks. This strategy can help to reduce the risk of overfitting or insufficient sampling caused by excessive data imbalance.
2) In AD-DES, the balanced data chunk obtained by the AER strategy is stored to balance the subsequent ones. In addition, these stored data chunks are utilized within the DES strategy to optimize its performance in nonstationary environments.
3) A dual dynamic ensemble selection (D-DES) strategy is introduced, which captures the feature information surrounding the test instances to define different types of competence subregions. Based on this, the classifier pool undergoes two rounds of selection to obtain the optimal ensemble model.
4) The adaptive drift detector in AD-DES is utilized to dynamically adjust the size of the classifier pool and stored data chunks. It is beneficial for the model to adapt to newly emerging concepts in data streams with concept drift.

The rest of this article is organized as follows. Section 2 reviews related works. Section 3 describes the proposed AD-DES in detail. Section 4 presents the experimental setup. The experimental results and analysis are presented in Section 5. Section 6 presents conclusions.

2. Related works

This section introduces relevant research addressing class imbalance and concept drift, and further discusses learning strategies tailored to nonstationary data streams.

2.1. Concept drift

Concept drift denotes changes in the underlying distribution of the target variable over time [3], leading to evolving relationships between input data and output labels and rendering previously trained models ineffective. More specifically, concept drift refers to the evolving nature of the joint probability distribution, that is, $\exists t : P_t(X,Y) \neq P_{t+1}(X,Y)$ [16]. Based on Bayes' theorem, concept drift can be grouped into the following four types [2, 17].

1) Data Distribution Drift $P(X)$: This type of drift is regarded as a virtual drift, reflecting temporal changes in the distribution of input data.
2) Prior Probability Drift $P(Y)$: This is classified as a virtual drift, which involves changes in the relative frequency of class labels.
3) Conditional Probability Drift $P(X|Y)$: This is also classified as a virtual drift, which occurs when the feature generation processes for different classes change.
4) Posterior Probability Drift $P(Y|X)$: This type of drift represents a real drift, characterized by shifts in the decision boundary caused by changes in the posterior probability distribution.

As mentioned above, only the posterior probability drift $P(Y|X)$ is regarded as a real drift. In this case, temporal shifts in the target class distribution can adversely affect the predictive capability of the

model [2]. The other three types of drift, i.e., $P(X)$, $P(Y)$ and $P(X|Y)$, are referred to virtual drifts [18], which can also influence the classifier's decision-making process to some extent. It should be pointed out that these four types of drift rarely occur in isolation. They often appear simultaneously in data streams, posing significant challenges for classifier training [19].

2.2. Imbalanced data classification

Class imbalance refers to a situation in classification tasks where there is a disproportionate distribution of instances among different classes [4]. Under the circumstances, classifiers trained on imbalanced datasets are biased toward the majority class in binary classification problems, leading to subpar performance when predicting the minority class. To address this issue, several approaches have been introduced one after another, which are usually categorized into two types: oversampling and undersampling techniques [11].

In oversampling approaches, additional minority class instances are generated to balance the class distribution. Among them, the synthetic minority oversampling technique (SMOTE) is extensively employed in imbalanced data classification tasks [20]. In the SMOTE, new instances are created through linear interpolation between existing instances of the minority class. Based on the SMOTE, Han et al. proposed the borderline-SMOTE method [21], which emphasizes oversampling minority class instances located near the classification boundary to avoid overfitting during model training. Bunkhumpornpat et al. introduced the safe-level-SMOTE based on the SMOTE to reduce inter-class conflict [22], where the newly generated minority class instances do not overlap with those of the majority class. Furthermore, He et al. presented a novel adaptive synthetic (ADASYN) sampling approach [23]. In ADASYN, minority class instances are assigned adaptive weights, which in turn influence the quantity of synthetic instances created per instance. This ensures that more synthetic instances based on the harder-to-learn ones are generated to compensate for the skewed distribution of instances. To further address the limitations of the SMOTE, such as oversampling noisy or uninformative instances and increasing the risk of class overlap, Soltanzadeh et al. proposed the range-controlled SMOTE (RCSMOTE) method [24]. This method improves both the instance categorization strategy and the synthetic instance generation process, thereby defining safe oversampling regions for each feature dimension. In addition, a novel oversampling method was proposed, termed the SMOTE with natural neighbors (NaNSMOTE) [25], which dynamically adjusts both the sampling distance and the number of neighbors according to data complexity.

For undersampling approaches, random undersampling (RUS) is a well-known method for the data resampling process [26]. This method achieves data balance by randomly eliminating instances from the majority class. Based on RUS, several undersampling methods have been developed sequentially to reduce the risk of discarding valuable information. For instance, Kumar et al. proposed the undersampled K-means algorithm (USKM) [27], where the imbalanced data is clustered by intelligently removing noise and instances with less information from the majority class. Ng et al. introduced a diversified sensitivity-based undersampling method (DSUS) [28], which leverages the distribution characteristics of the training data to promote diversity during the resampling process. Expanding upon clustering strategies, Lin et al. proposed two undersampling methods utilizing clustering algorithms [29], wherein the majority class instances are divided into clusters corresponding in quantity to the minority class instances. The centers of the clusters along with their closest neighbors are then selected as representative majority instances. Similarly, Tsai et al. introduced the cluster-based instance selection (CBIS) method [30]. This method groups similar majority class instances into "subclasses" and then refines the dataset by filtering

out less representative instances from each group to achieve a balanced distribution. In recent years, an equalization ensemble method (EASE) was proposed by Ren et al. [31]. In EASE, the equalization undersampling strategy is utilized to obtain balanced data chunks, with the purpose of mitigating the impact of class imbalance on individual classifiers.

Although the aforementioned methods alleviate class imbalance to some extent, they commonly rely on a single resampling strategy. As the degree of imbalance increases, the inherent limitations of oversampling and undersampling become more pronounced, ultimately degrading model performance. Therefore, the primary objective of this study is to develop a method capable of dynamically adjusting the resampling approach according to the imbalance level, and even integrating multiple strategies to achieve flexible and adaptive balancing.

2.3. Learning in nonstationary environments

With mobile internet technology rapidly evolving, data is often produced in the form of streams in many practical applications (e.g., e-commerce and social media), which is known as data streams [32]. It is very common for class imbalance and concept drift to occur simultaneously in data streams. Although several existing approaches can effectively address the problems of class imbalance and concept drift separately, it remains challenging to deal with both of them in data streams [33]. To address this challenge, several algorithms have been proposed successively, which can generally be categorized into the stream-based methods and the chunk-based methods.

In the stream-based methods, instances are processed sequentially. For instance, Wang et al. first introduced a drift detection method for online class imbalance problems (DDM-OCI) in data streams [9]. In DDM-OCI, concept drift is detected by continuously monitoring the recall of minority class instances. Based on DDM-OCI, Wang and Abraham presented a novel framework [34], named linear four rates (LFR), in which concept drift is detected to pinpoint data points belonging to new concepts. Subsequently, oversampling-based online bagging (OOB) and undersampling-based online bagging (UOB) were proposed to solve the classification problem in imbalanced data streams [35]. In the two algorithms, a dynamic mechanism is employed to adjust the quantity of resampled instances according to a Poisson distribution. Based on the SMOTE algorithm, a continuous SMOTE (C-SMOTE) method was developed to improve the classification of imbalanced data streams affected by concept drift [36]. As a pipeline-based rebalancing meta-strategy grounded in SML classification algorithms, the C-SMOTE can be flexibly integrated with various data stream classifiers. Malialis et al. introduced an adaptive rebalancing (AREBA) algorithm, which adaptively modifies queue lengths to maintain a balance among instances [37]. Besides, Cano and Krawczyk proposed a robust online self-adjusting ensemble (ROSE) algorithm for the classification of imbalanced data in nonstationary environments [38]. In the ROSE, adaptive bagging and sliding window strategies are employed to enhance the visibility of minority class instances, thereby facilitating the construction of skew-insensitive classifiers.

Unlike the stream-based methods, chunk-based methods process multiple instances simultaneously, which is beneficial for having higher computational efficiency. Based on this, they are better suited for handling large datasets. As a representative of chunk-based methods, Learn++ family algorithms, in which penalty constraints and the SMOTE are incorporated into the Learn++.NSE algorithm, were proposed for learning in nonstationary environments [39]. Gomes et al. introduced the adaptive random forests (ARF) algorithm [40]. In the ARF algorithm, a resampling method and adaptive operators are effectively integrated to cope with diverse forms of concept drift. Furthermore, Zyblewski et al.

presented an innovative framework designed for classifying imbalanced data streams [41], where a dynamic ensemble selection method is employed to enhance the diversity of base classifiers. Jiao et al. proposed a DES for imbalanced data streams with concept drift (DES-ICD) algorithm [15]. In DES-ICD, a novel oversampling technique (AnnSMOTE) is employed to generate balanced data chunks during data preprocessing. Subsequently, different subsets of classifiers are selected for test instances to achieve better prediction performance. However, DES-ICD uses the most recent data chunk as the validation dataset, which may be suboptimal in imbalanced data stream scenarios, since the latest chunk can be highly skewed and may lack sufficient samples in competence subregions. Moreover, the validation dataset in DES-ICD is neither preprocessed nor influenced by other adaptive strategies, thereby limiting the potential interaction between the resampling and ensemble selection mechanisms. Klikowski and Woźniak introduced the deterministic sampling classifier with weighted bagging (DSCB) algorithm to address the classification of imbalanced data streams [11]. In DSCB, undersampling and oversampling techniques are integrated to construct a weighted bagging model, and a time decay factor is employed to dynamically adjust the influence of stored historical data chunks on the current learning process. Nevertheless, the resampling process and ensemble weighting in DSCB are loosely coupled, and there is no explicit mechanism ensuring their mutual adaptation or feedback adjustment, which may limit its responsiveness to complex concept drifts. In our preliminary work, the adaptive bagging-based dynamic ensemble selection (AB-DES) algorithm was proposed [7]. In AB-DES, a novel dual adaptive sampling bagging strategy is utilized to update base classifiers and enhance the algorithm's ability to adapt to concept drift.

Existing methods have demonstrated certain effectiveness in handling imbalanced data streams with concept drift; however, there remains considerable room for improvement. Specifically, the potential synergy between DES and resampling strategies has not been fully exploited, and many DES-based algorithms exhibit limited exploration of competence subregions under varying distributional scenarios. Therefore, this study aims to address these limitations to further enhance model performance. In addition, existing drift detectors require refinement to more accurately monitor concept drift and enable timely and appropriate adaptation.

3. AD-DES

For the binary classification of imbalanced data streams with concept drift, an adaptive dual dynamic ensemble selection (AD-DES) method is proposed in this paper. Specifically, AD-DES has four parts, which are the adaptive equalization resampling (AER) strategy, classifier pool management, dual dynamic ensemble selection (D-DES) strategy, and adaptive drift detection. The AER strategy, which is adaptively adjusted according to the imbalance ratio, is used to obtain balanced data chunks. Subsequently, the balanced data chunk is stored to balance the subsequent ones, which are then utilized to create (update) the base classifiers effectively. After that, a D-DES strategy is introduced to adaptively choose the optimal base classifiers for prediction. Meanwhile, the adaptive drift detection mechanism is employed to dynamically adjust the size of the classifier pool and the number of stored data chunks, ensuring the model's robustness and accuracy in nonstationary environments. Figure 1 illustrates the overall structure of AD-DES.

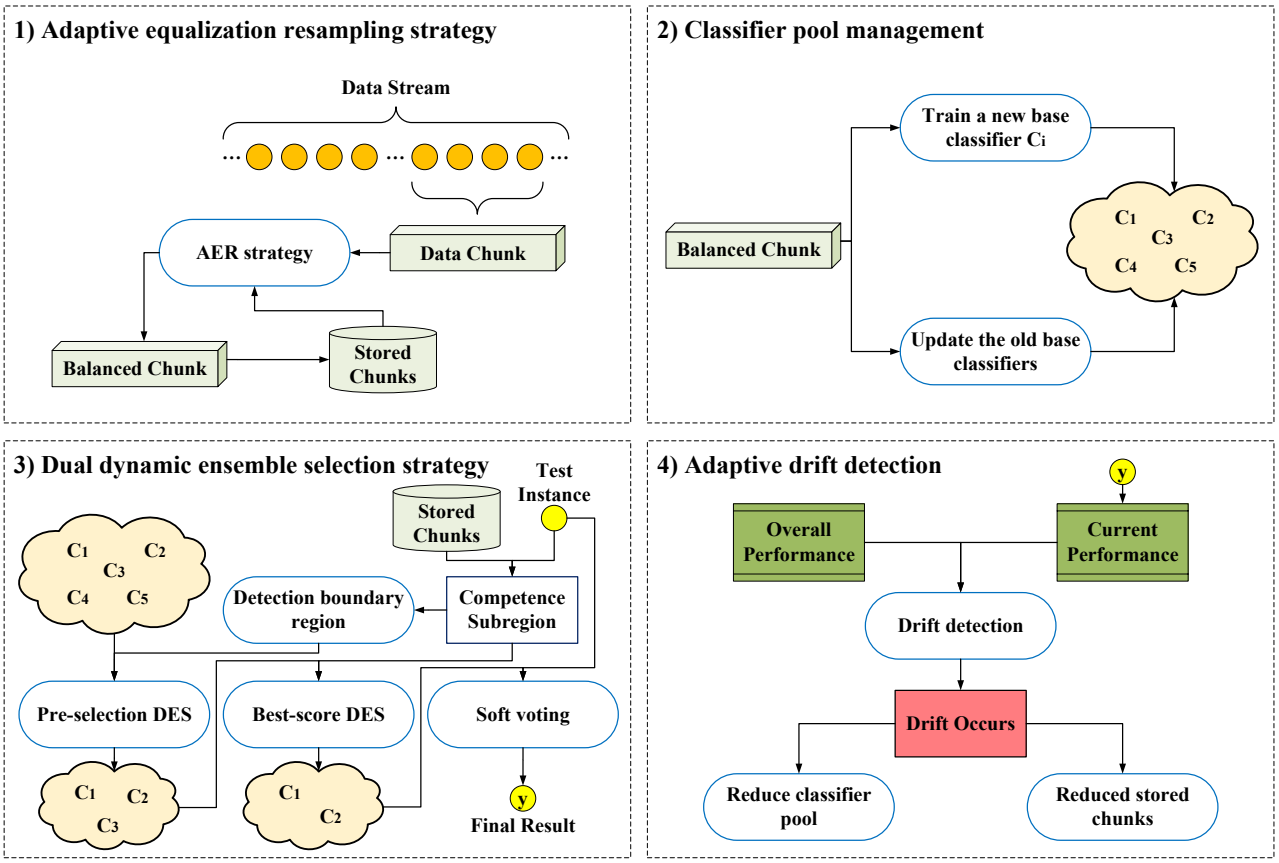


Figure 1. Framework of AD-DES. 1) Each data chunk is adaptively balanced via hybrid resampling using current and stored chunks. 2) A new base classifier is trained, and old ones are updated to maintain diversity. 3) Two-level dynamic selection combines pre-selection and best-score selection with soft voting for final prediction. 4) Drift detection compares overall and current performance, adaptively reducing the classifier pool and stored chunks when drift occurs.

3.1. Adaptive equalization resampling strategy

In this subsection, the adaptive equalization resampling (AER) strategy is proposed to tackle classification challenges under conditions of significant class imbalance. In the AER strategy, three kinds of instances, including stored instances and those obtained by oversampling and undersampling, are dynamically combined according to the current imbalance ratio (IR) of each data chunk, with the purpose of obtaining balanced data chunks. The IR is typically used to reflect the relative proportions of instances in the majority and minority classes, indicating the degree of class imbalance [11]. The IR is defined in Eq (3.1).

$$\mathrm{IR} = \frac{N_{\mathrm{maj}}}{N_{\mathrm{min}}} \tag{3.1}$$

where N_{maj} and N_{min} denote the amount of instances in the majority and minority classes, respectively. The higher the IR, the greater the degree of class imbalance.

To better accommodate learning in data streams, the degree of class imbalance is categorized into three levels in this study, following the common practice of imbalance-level partitioning in previous studies [42, 43], with the goal of enhancing the generalizability of AD-DES, as shown in Eq (3.2). Based on this, different resampling methods are employed to process data chunks with different levels of class imbalance.

$$\begin{cases} \text{Slight Imbalance}, & 1 < \text{IR} \leq \alpha \\ \text{Moderate Imbalance}, & \alpha < \text{IR} \leq \beta \\ \text{High Imbalance}, & \text{IR} > \beta \end{cases} \tag{3.2}$$

where α and β are threshold parameters for classifying different degrees of class imbalance.

Algorithm 1: Adaptive equalization resampling

Input: D_t: current data chunk, S: max size of stored data chunks, D_s: stored data chunks
Output: D_t^B: a balanced data chunk

1 $D_t^{\text{maj}} \leftarrow$ extract the majority class instances from D_t
2 $D_t^{\text{min}} \leftarrow$ extract the minority class instances from D_t
3 $D_s^{\text{min}} \leftarrow$ extract the minority class instances from D_s
4 IR $\leftarrow$ calculate the imbalance ratio of D_t
5 **if** $t = 0$ **then**
6 $\quad D_t^{U\text{maj}} \leftarrow$ undersample D_t^{maj}
7 $\quad D_t^B \leftarrow D_t^{U\text{maj}} \cup D_t^{\text{min}}$
8 **end**
9 **else**
10 $\quad$ **if** $1 < \text{IR} \leq \alpha$ **then**
11 $\quad\quad D_t^{U\text{maj}} \leftarrow$ undersample D_t^{maj}
12 $\quad\quad D_t^B \leftarrow D_t^{U\text{maj}} \cup D_t^{\text{min}}$
13 $\quad$ **end**
14 $\quad$ **else if** $\text{IR} \leq \beta$ **then**
15 $\quad\quad D_t^{U\text{maj}} \leftarrow$ undersample D_t^{maj}
16 $\quad\quad D_t^{O\text{min}} \leftarrow$ oversample D_t^{min}
17 $\quad\quad D_t^B \leftarrow D_t^{U\text{maj}} \cup D_t^{O\text{min}}$
18 $\quad$ **end**
19 $\quad$ **else if** $\text{IR} > \beta$ **then**
20 $\quad\quad D_t^{U\text{maj}} \leftarrow$ undersample D_t^{maj}
21 $\quad\quad D_t^{O\text{min}} \leftarrow$ oversample D_t^{min}
22 $\quad\quad D_t^B \leftarrow D_t^{U\text{maj}} \cup D_t^{O\text{min}} \cup D_s^{\text{min}}$
23 $\quad$ **end**
24 **end**
25 **return** D_t^B

Algorithm 1 presents the pseudocode of the AER strategy. In lines 10–13, the data chunk exhibits slight imbalance. Since there are sufficient minority class instances, a balanced data chunk can be easily

achieved through undersampling the majority class instances. When the data chunk exhibits moderate imbalance, as shown in lines 14–18, the difference in class sizes is relatively small. In such a case, the AER strategy can simultaneously utilize both undersampling and oversampling methods to achieve balance. As a result, the AER strategy can effectively mitigate the issues of overfitting and underfitting caused by relying on a single method. For highly imbalanced data chunks, as illustrated in lines 19–23, resampling methods alone are insufficient for achieving balance, so it is crucial to effectively utilize the stored instances. Specifically, the AER strategy applies moderate undersampling and oversampling to the imbalanced data chunk, while the remaining imbalance is rectified by incorporating the latest stored instances, which were obtained through the resampling methods in the previous AER strategy.

Overall, the AER strategy seamlessly integrates oversampling, undersampling, and stored instances, reducing the risk of overfitting or insufficient sampling caused by excessive data imbalance. Notably, the stored instances both balance future data chunks and determine incoming instances' competence subregions. What is more, the way of utilizing the latest stored instances to balance subsequent data chunks is beneficial for enriching the training dataset, which can help to create (update) the base classifiers effectively.

3.2. Classifier pool management

To establish an effective ensemble model in nonstationary environments, a classifier pool is maintained in AD-DES, as shown in Figure 1(2). Specifically, the AER strategy generates a balanced data chunk (D_t^B), which is then used to train a novel base classifier before being added to the existing classifier pool (P). As a note, any suitable online classifier can be used as the base classifier based on specific requirements. Furthermore, the current balanced data chunk (D_t^B) is fed into the classifier pool to update the existing base classifiers, thereby enhancing the adaptability of AD-DES to data streams exhibiting concept drift. Moreover, the oldest base classifier is removed when the classifier pool reaches its maximum capacity to preserve its effectiveness. The detailed process for classifier pool management is presented in Algorithm 2.

Algorithm 2: Classifier pool management

Input: D_t^B: a balanced data chunk, M: the max size of the base classifier pool, P: un-updated base classifier pool

Output: P: updated base classifier pool

for *each base classifier* C_i *in* P **do**
  update C_i based on D_t^B
end
$C_j \leftarrow$ train a new base classifier on D_t^B
$P \leftarrow P \cup C_j$
if $|P| > M$ **then**
  Remove the oldest base classifier from P
end
return P

3.3. Dual dynamic ensemble selection strategy

In this subsection, a dual dynamic ensemble selection (D-DES) strategy is introduced. This strategy involves two rounds of DES. First, an initial round of DES is applied to the classifier pool to select those classifiers that perform well in the boundary regions, a process referred to as pre-selection DES. Subsequently, a second round of DES is conducted on the pre-selection classifiers to identify the highest-scoring ones, which is termed best-score DES. Specifically, the D-DES strategy consists of four main steps.

First, the validation dataset is constructed from all stored data chunks. Notably, these chunks are generated by the AER strategy and dynamically adjusted based on feedback from an adaptive drift detector, as described in Section 3.4. This approach ensures interconnection among various strategies within AD-DES, enabling quicker adaptation to concept drift. Specifically, fewer data chunks are retained for the validation dataset when the data stream becomes unstable, which helps candidate base classifiers adapt to new concepts. Conversely, when the data stream stabilizes, more data chunks are retained for the validation dataset, which allows richer historical information to be utilized for selecting the base classifiers.

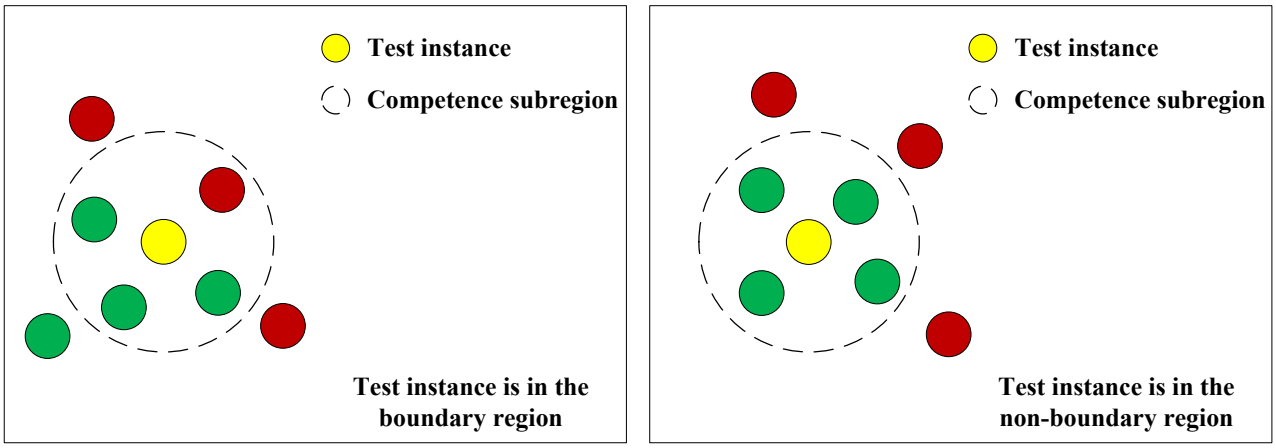
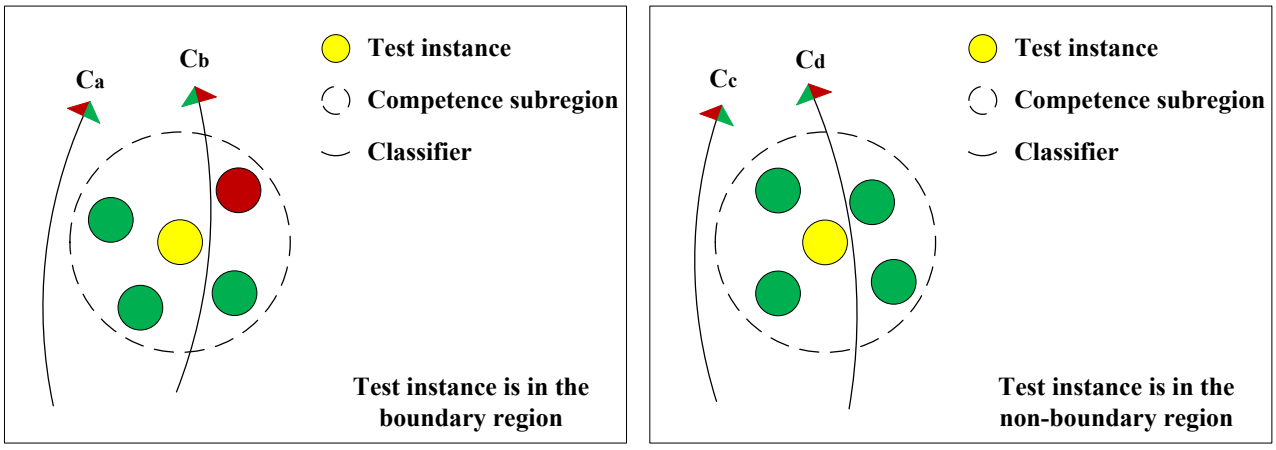


Figure 2. Two types of competence subregions.

Second, the K nearest instances of the test instance within the validation dataset are identified to form the competence subregion. Since these K instances have the closest properties to the test instance, the competence subregion is used to select an optimal subset of base classifiers, with the purpose of constructing a high-performance ensemble model for specific characteristics of the test instance. Furthermore, the competence subregion is categorized into two types in the D-DES strategy. Two types of competence subregions are illustrated in Figure 2, where the yellow circle represents the test instance, red (green) circles denote instances from different classes, and dashed lines delineate the competence subregion. If the competence subregion contains instances of both the majority and minority classes, it is classified as a boundary region, as shown in Figure 2(a). Such boundary regions denote the intersection of instances with different labels, where base classifiers with strong discriminative capabilities are needed to resolve ambiguity. If the competence subregion consists solely of instances of a single class, it is termed a non-boundary region, as shown in Figure 2(b). Under these circumstances, the test instance

resides within a stable and homogeneous area dominated by one class. This kind of division plays a critical role in selecting base classifiers, as different base classifiers are suited for different characteristics of competence subregions.

Third, the behavior of base classifiers is analyzed in these distinct types of competence subregions, and the pre-selection DES is performed on the classifier pool. Figure 3 illustrates potential scenarios of base classifiers, with the continuous curves representing base classifiers. For instance, two scenarios of base classifiers near a boundary region are depicted in Figure 3(a), where the classifier C_a does not intersect the competence subregion, while the classifier C_b does. The classifier C_b is prioritized in the D-DES strategy although the classification accuracies of both classifiers have reached 75%. This is because C_b is capable of accurately classifying at least one pair of instances belonging to different classes, whereas C_a, which assigns all instances to the same class, may lack genuine classification ability. Furthermore, two scenarios of base classifiers near a non-boundary region are depicted in Figure 3(b), where the classifier C_c does not cross the competence subregion, while the classifier C_d does. The classifier C_c is prioritized in the D-DES strategy since its accuracy has reached 100%. In summary, when a test instance falls within a boundary region, base classifiers that cross the competence subregion are selected to distinguish between instances of different classes. For a test instance located in a non-boundary region, base classifiers that do not intersect the competence subregion are selected to accurately classify all instances of the same class. Specifically, when no base classifiers meet the selection criteria, all available classifiers are included in the ensemble model.



(a) (b)

Figure 3. Classifiers in different scenarios.

Finally, the best-score DES is performed on the subset of base classifiers obtained in the previous step using the competence subregion. From the subset of classifiers, the highest-performing classifiers are selected for predicting the test instance. Considering the potential presence of instances of different classes within the competence subregion, the G-mean metric was utilized to assess model performance in our work. Furthermore, a soft voting strategy is employed to decide the final result of the test instance, which can enhance prediction reliability by incorporating the confidence levels of multiple base classifiers. The soft voting strategy is described in Eq (3.3).

$$\hat{y} = \arg\max_{y=0,1} \sum_{C_i \in E_{\text{sub}}} C_i(y) \tag{3.3}$$

where $\hat{y}$ denotes the final prediction result, E_{sub} represents the ensemble of selected base classifiers based on their highest evaluation scores, and $C_i(y)$ indicates the predicted probability of class y by the i-th base classifier.

The detailed implementation procedure of the D-DES strategy is described in Algorithm 3.

Algorithm 3: Dual dynamic ensemble selection

Input: x: a test instance, P: the classifier pool, D_s: stored data chunks
Output: $\hat{y}$: the final prediction result

$\psi \leftarrow K$ nearest instances of x in D_s
$P' \leftarrow \emptyset$
$scores \leftarrow \emptyset$
if ψ *is boundary region* **then**
  for *each base classifier* C_i *in* P **do**
    $\Phi \leftarrow$ instances in ψ correctly classified by C_i
    $L_{min} \leftarrow$ the minority instances in Φ
    $L_{maj} \leftarrow$ the majority instances in Φ
    if $|L_{min}| \geq 1$ *and* $|L_{maj}| \geq 1$ **then**
      $P' \leftarrow P' \cup C_i$
    end
  end
end
else
  for *each base classifier* C_i *in* P **do**
    $\Phi \leftarrow$ instances in ψ correctly classified by C_i
    if $|\Phi| = |\psi|$ **then**
      $P' \leftarrow P' \cup C_i$
    end
  end
end
if $|P'| = 0$ **then**
  $P' \leftarrow P$
end
for C_i *in* P' **do**
  $y_i \leftarrow C_i$ predict result on ψ
  $score \leftarrow$ calculate G-mean value based on y_i
  $scores \leftarrow scores \cup score$
end
$E_{sub} \leftarrow$ select base classifiers with the highest G-mean values according to *scores*
$\hat{y} \leftarrow$ determine the final prediction result according to Eq (3.3)
return $\hat{y}$

3.4. Adaptive drift detection

As typically passive approaches, traditional methods generally rely on incremental learning of classifiers and dynamic adjustment of ensemble models, continuously adapting to new data in response to potential shifts in nonstationary environments. In this way, these methods can cope with the gradual concept drift effectively, however, they may fail to adapt to sudden (significant) concept drift.

To overcome this limitation, an adaptive drift detection mechanism is integrated into AD-DES to proactively identify shifts in data streams. When concept drift is detected, the sizes of the classifier pool and stored data chunks are reduced to facilitate adaptation to newly emerging concepts. Conversely, during stable periods, these sizes are gradually increased to enhance the adaptability of the model by utilizing richer historical information.

Algorithm 4: Adaptive drift detection

Input: G_s: stored performance scores, T: the max size of stored performance scores, G_t: the performance score of the current data chunk, P: the classifier pool, M: the max size of the classifier pool, D_s: stored data chunks, S: the max size of the stored data chunks

Output: $N'(P)$: the updated number of classifiers, $N'(D_s)$: the updated number of stored data chunks

$ratio \leftarrow$ determine the degree of concept drift according to Eq (3.4)
if $ratio \geq \Delta$ **then**
    $N'(P) \leftarrow$ determine the updated number of classifiers according to Eq (3.6)
    $N'(D_s) \leftarrow$ determine the updated number of stored data chunks according to Eq (3.7)
end
else
    $N'(P) \leftarrow |P| + 1$
    $N'(D_s) \leftarrow |D_s| + 1$
end
$N'(P) \leftarrow \min(N'(P), M)$
$N'(D_s) \leftarrow \min(N'(D_s), S)$
$G_s \leftarrow G_s \cup G_t$
if $|G_s| > T$ **then**
    Remove the oldest performance score from G_s
end
return $N'(P), N'(D_s)$

Specifically, the performance scores of the latest T data chunks (G_s) are monitored to detect concept drift. Specifically, the G-mean is employed as the performance metric to capture changes in imbalanced data streams. The performance score of the current data chunk (G_t) and the average of G_s are used to calculate the degree of concept drift, which is quantified using the *ratio* metric, as demonstrated in Eq (3.4). Concept drift is then detected based on the criterion specified in Eq (3.5).

$$ratio = \frac{\text{mean}(G_s)}{G_t} \tag{3.4}$$

$$\text{Concept Drift} = \begin{cases} \text{Detected}, & \text{if } ratio \geq \Delta \\ \text{Not Detected}, & \text{otherwise} \end{cases} \tag{3.5}$$

where Δ represents the threshold parameter for detecting concept drift. When drift is identified, both base classifiers and data chunks are dynamically scaled based on drift magnitude, as specified in Eqs (3.6) and (3.7):

$$N'(P) = \max\left(1, \left\lceil \frac{|P|}{e^{ratio-1}} \right\rceil\right) \tag{3.6}$$

$$N'(D_s) = \max\left(1, \left\lceil \frac{|D_s|}{e^{\pi\cdot(ratio-1)\cdot ratio}} \right\rceil\right) \tag{3.7}$$

where $N'(P)$ represents the number of classifiers retained in the updated classifier pool, and $N'(D_s)$ denotes the number of stored data chunks after adjustment.

The detailed pseudocode for the adaptive drift detection mechanism is provided in Algorithm 4.

3.5. Computational complexity analysis

This subsection analyzes the computational complexity of AD-DES, which mainly consists of the training and prediction phases. Since the adaptive drift detection component contributes minimal overhead, it is considered negligible in this analysis.

In the training phase, the worst-case scenario occurs when all three components (i.e., RUS, SMOTE, and stored data chunks) are involved simultaneously to generate a balanced dataset. Given a data chunk of size N, the computational complexity of RUS is $O(N)$. The complexity of generating synthetic instances is $O(D\cdot N\log(N))$ in the SMOTE method, where D represents the size of features in the dataset. The computational complexity of stored data chunks is $O(1)$ since the latest data chunks are utilized, which can be considered negligible. Furthermore, assuming that training a single base classifier requires $C_{train}(N)$, the overall training complexity for all M classifiers in the pool is $O(M\cdot C_{train}(N))$. In the training phase, the overall computational complexity is expressed as $O(N+D\cdot N\log(N)+M\cdot C_{train}(N))$.

In the prediction phase, the computational complexity of identifying the competence subregion is $O(S\cdot N\log(K))$, where S represents the size of the stored data chunks (i.e., the validation dataset), and K is the size of the competence subregions. Based on a single base classifier, the computational complexity of prediction is denoted as $C_{predict}$. Therefore, selecting the most competent classifiers has a complexity of $O(M\cdot C_{predict}(K))$. Finally, the classifier ensemble model is employed for predictions, which involves a complexity of $O(M\cdot C_{predict}(N))$. As a result, the total complexity is $O(S\cdot N\log(K)+M\cdot C_{predict}(K)+M\cdot C_{predict}(N))$ in the prediction phase.

According to the above analysis, the overall computational complexity of AD-DES, combining both the training and prediction phases, is shown as Eq (3.8).

$$O\left(\underbrace{N+D\cdot N\log(N)+M\cdot C_{train}(N)}_{\text{training}}+\underbrace{S\cdot N\log(K)+M\cdot C_{predict}(K)+M\cdot C_{predict}(N)}_{\text{prediction}}\right) \tag{3.8}$$

4. Experimental setup

This section presents the experimental framework, including benchmark datasets, comparative algorithms, evaluation metrics, and parameter configurations. A comprehensive analysis of the experimental results will be provided in the following section.

4.1. Benchmark datasets

The benchmark datasets utilized in the experiments are summarized in Table 1, which includes 10 synthetic datasets and 5 real-world datasets. The synthetic datasets exhibit three types of concept drift: sudden, gradual, and incremental. Except for the Gaussian and Hyper Plane datasets, the concept drift in the synthetic datasets occurs at the midpoint of the data stream, resulting in an interchange between the minority and the majority classes.

Table 1. Information of datasets.

Dataset	Instances	Features	Drift Type	IR	Source
Sine (sudden)	50,000	2	*sudden*	50/1	synthetic
Sine (gradual)	50,000	2	*gradual*	50/1	synthetic
SEA (sudden)	50,000	3	*sudden*	50/1	synthetic
SEA (gradual)	50,000	3	*gradual*	50/1	synthetic
Mixed (sudden)	50,000	4	*sudden*	50/1	synthetic
Mixed (gradual)	50,000	4	*gradual*	50/1	synthetic
Tree (sudden)	50,000	5	*sudden*	50/1	synthetic
Tree (gradual)	50,000	5	*gradual*	50/1	synthetic
Gaussian	50,000	2	*incremental*	20/1	synthetic
Hyper Plane	48,530	10	*incremental*	20/1	synthetic
Electricity	16,912	7	*unknown*	20/1	real-world
Weather	18,159	8	*unknown*	7/3	real-world
GMSC	150,000	10	*unknown*	14/1	real-world
CovType35	45,247	54	*unknown*	35/9	real-world
CovType36	53,121	54	*unknown*	35/17	real-world

1) Sine [44]: The Sine dataset consists of two features, each ranging from 0 to 1, with a classification boundary defined by a sine function. Data points positioned beneath the sine curve are classified as positive instances. Concept drift is introduced by reversing the class labels.
2) SEA [45]: The SEA dataset consists of three features, with values ranging from 0 to 10. The first two features are used for classification, while the third serves as noise. Instances are randomly generated and classified based on a predefined threshold, with concept drift introduced by altering this threshold.
3) Mixed [44]: The Mixed dataset consists of two binary features (with values 0 or 1) and two numerical features uniformly distributed between 0 and 1. Classification rules are based on various combinations of these features, with concept drift introduced through changes in these combinations.

4) Tree: The Tree dataset is generated using a decision tree model, where classification labels are determined by branch nodes and partitioning rules. This dataset exhibits only $P(y)$ drift.
5) Gaussian [46]: The Gaussian dataset consists of two classes generated from two-dimensional Gaussian distributions that exhibit incremental drift throughout the data stream. The mean coordinates of the two classes gradually shift from [5,0] and [7,0] to [-5,0] and [-3,0], demonstrating a smooth drift process.
6) Hyper Plane [47]: The Hyper Plane dataset generates data points based on the geometry of a hyperplane, where a point x satisfies the Eq $f(x) = \sum_{i=1}^{d-1} a_i \cdot \frac{x_i + x_{i+1}}{x_i}$. The dimensionality d and the coefficients a_i define the decision boundary. Concept drift is introduced through overlap or misclassification between the two initially separable classes.
7) Electricity [48]: The Electricity dataset captures electricity load demand and price data from the New South Wales electricity market, sampled at 30-minute intervals. Class labels are determined based on changes in electricity prices.
8) Weather [49]: The Weather dataset encompasses 50 years of weather data from Bellevue, with the goal of predicting rainfall on the following day based on historical weather features.
9) GMSC [40]: The GMSC dataset, obtained from Kaggle's "Give Me Some Credit" project, includes various financial characteristics of users, with the aim of predicting future credit defaults.
10) CovType [47]: The CovType dataset, provided by the U.S. Forest Service (USFS), contains forest cover type data with 54 features across seven classes, labeled from 1 to 7. To generate two binary classification datasets, classes 3, 5, and 6 are selectively grouped, with class 3 treated as the majority class and classes 5 and 6 treated as the minority classes, resulting in the datasets CovType35 and CovType36.

4.2. Comparison algorithms

To validate the effectiveness of AD-DES, seven chunk-based algorithms and two stream-based algorithms are employed to compare their performance in nonstationary environments. The competing algorithms selected for comparative evaluation consist of the following:

Chunk-based algorithms:

1) Learn++.NSE (2011) [50]: It incrementally trains a dedicated base classifier on each arriving data chunk and aggregates their predictions through an adaptive weighted voting mechanism. The weight assigned to each classifier during prediction decreases based on its performance, and classifiers that fall below a predefined performance threshold are eliminated.
2) ARF (2017) [40]: ARF is an ensemble learning model based on decision trees, designed to address various types of concept drift. When drift is detected, underperforming trees are either reset or replaced with new ones.
3) DWMIL (2017) [46]: DWMIL is an incremental learning method that adaptively updates the weights of classifiers to minimize the impact of underperforming ones. Furthermore, the integration of bagging strategies helps mitigate the influence of noise, thereby enhancing the overall model's robustness.
4) ROS-U (2021) [41]: ROS-U employs stratified bagging to train classifiers and incorporates preprocessing and dynamic ensemble selection methods to classify imbalanced data streams.
5) DES-ICD (2022) [15]: DES-ICD uses the most recent data chunk as the validation dataset to

ensure a quick response during concept drift while maintaining model performance. Additionally, the integration of the AnnSMOTE enhances the algorithm's ability to handle class imbalance more effectively.

6) DSCB (2022) [11]: DSCB combines undersampling and oversampling methods to process imbalanced data streams. Furthermore, the use of weighted bagging improves the diversity of the base classifiers.
7) EMRIL (2024) [51]: EMRIL introduces reinforcement learning into ensemble pool management to effectively construct an optimal base classifier subset. This technique demonstrates remarkable performance in handling data streams with severe concept drift and pronounced class imbalance.

Stream-based algorithms:

1) DDM-OCI (2013) [9]: This algorithm continuously tracks minority class recall as a key indicator for concept drift detection. Notably, all classifiers are reset upon detecting drift to better adapt to new concepts.
2) RE-DI (2019) [52]: RE-DI integrates static and dynamic classifiers, adjusting the weights of base classifiers to place greater emphasis on new concepts.

4.3. Evaluation metrics

In balanced classification tasks, the accuracy of a model is commonly used as the evaluation metric. However, the accuracy of the model tends to be dominated by the majority class under class imbalance, which can obscure its true performance. In contrast, the G-mean and the recall serve as more robust indicators for evaluating the model's effectiveness on the minority class, making them more suitable for comprehensive evaluation in imbalanced data scenarios. Accordingly, the G-mean and the recall are adopted as performance measures throughout our experiments, with their calculation methods presented in Eqs (4.1) and (4.2), respectively.

$$\text{G-mean} = \sqrt{\text{Recall} \times \text{Specificity}} \tag{4.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

where Specificity represents the ability of the model to correctly classify negative instances, calculated as Specificity $= \frac{\text{TN}}{\text{TN+FP}}$. TP, FN, FP, and TN represent true positives, false negatives, false positives, and true negatives, respectively, which can be derived from the confusion matrix.

In addition, to enable fair comparisons across different algorithms, we record the prediction scores of each algorithm for every data chunk in the data stream. The final comparison results are then derived by averaging these scores of each algorithm across all data chunks in the data stream.

4.4. Experimental settings

All algorithmic implementations in our experiments were developed using Python 3.8. The experiments were conducted on a system equipped with a 24-core 13th Gen Intel(R) Core(TM) i9-13980HX CPU, 16 GB of RAM, and the Windows 11 operating system. To ensure fairness, AD-DES and 9

comparison algorithms employed the Hoeffding Tree Classifier from the Scikit-multiflow library [53] as the base learner, with identical parameter settings, including a maximum depth of 15, a Gini Index split criterion, and a grace period of 200 instances.

In the experiments, the chunk size for all chunk-based approaches was set to 500. In AD-DES, the SMOTE and RUS were utilized as the oversampling and undersampling methods, respectively, while α was set to 3 and β was set to 9. The candidate classifier pool (M) was limited to a maximum size of 20, and the number of stored data chunks (S) was capped at 10. For the D-DES strategy, the competence subregion size (K) was configured as 7. In the drift detection mechanism, the drift detection threshold (Δ) was set to 1.2, and the maximum size of the drift detection window (T) was set to 7, while G-mean was employed to track the variations in model performance. To ensure a fair evaluation, the main hyperparameters of all comparison algorithms were set according to the recommended or optimal configurations reported in their original publications, while keeping other experimental settings consistent across all algorithms. For each dataset, all experiments were independently repeated ten times, and the average results were recorded.

5. Experimental results and analysis

In this section, five groups of experiments were conducted on 10 synthetic datasets and 5 real-world datasets to comprehensively evaluate the performance of the AD-DES algorithm. First, the classification performance of AD-DES was compared with that of 9 state-of-the-art algorithms in nonstationary environments. Second, the changes in the number of base classifiers and stored data chunks within AD-DES were visualized when following the detection of concept drift, aiming to visually demonstrate the effectiveness of the dynamic adjustment mechanism. Third, the robustness of AD-DES on 8 datasets with different imbalance ratios was analyzed. Fourth, the sensitivity analysis of the main parameters in AD-DES was investigated. Finally, the analysis of the main strategies in AD-DES was conducted.

5.1. Comparison of AD-DES and state-of-the-art algorithms

In this subsection, the performance of AD-DES is compared with that of 9 competing methods based on the G-mean value and the recall value.

The average G-mean values of the proposed AD-DES and these 9 comparison algorithms are presented in Table 2. The best results are shown in bold. It can be seen that AD-DES achieved the best performance on 11 out of all 15 datasets, with an average ranking of 1.267. Specifically, AD-DES demonstrated superior overall performance compared with DSCB, which ranks second and relies solely on balancing strategies for classifier training. This superiority may stem from the inclusion of an adaptive drift detection mechanism, which enables AD-DES to more rapidly adapt to potential concept drift. Notably, on the two datasets with incremental drift (Gaussian and Hyper Plane), the time-varying concepts hindered DSCB's ability to rapidly adapt to emerging concepts, thus impacting its classification performance. For the DDM-OCI and ARF algorithms, which also utilize active drift detection mechanisms, they exhibited similar performance across most of these 15 datasets. They could quickly adapt to concept drift by resetting all (or most) classifiers, but faced the risk of losing valuable information due to minor concept drift. By contrast, AD-DES actively adjusted the amount of outdated information and discarded it based on the degree of drift, thereby demonstrating a stronger adaptability to concept drift.

Table 2. Average G-mean values (%) of different algorithms.

Dataset	DDM-OCI	RE-DI	Learn++.NSE	ARF	DWMIL	ROS-U	DES-ICD	DSCB	EMRIL	AD-DES
Sine (sudden)	88.13 (6)	69.81 (7)	61.22 (8)	90.75 (4)	90.93 (3)	37.81 (10)	92.88 (2)	90.31 (5)	46.92 (9)	**94.62 (1)**
Sine (gradual)	75.23 (6)	66.73 (7)	65.71 (8)	87.64 (5)	90.86 (3)	35.17 (10)	91.41 (2)	88.03 (4)	44.53 (9)	**93.56 (1)**
SEA (sudden)	90.57 (6)	89.71 (7)	66.95 (10)	92.77 (4)	93.82 (2)	68.58 (9)	90.94 (5)	93.24 (3)	78.96 (8)	**95.31 (1)**
SEA (gradual)	87.06 (7)	90.05 (6)	69.81 (10)	92.12 (4)	94.62 (2)	74.86 (8)	91.15 (5)	93.35 (3)	71.09 (9)	**95.51 (1)**
Mixed (sudden)	80.79 (6)	63.75 (7)	60.52 (8)	86.93 (4)	88.68 (2)	17.90 (10)	82.08 (5)	86.94 (3)	45.23 (9)	**91.07 (1)**
Mixed (gradual)	82.99 (4)	57.74 (8)	62.18 (7)	82.79 (5)	87.34 (2)	15.75 (10)	80.96 (6)	86.36 (3)	46.87 (9)	**89.27 (1)**
Tree (sudden)	79.33 (5)	68.06 (7)	53.06 (9)	56.42 (8)	82.07 (4)	32.01 (10)	84.14 (3)	**91.54 (1)**	69.21 (6)	91.03 (2)
Tree (gradual)	83.01 (3)	65.95 (6)	61.81 (9)	62.14 (8)	82.74 (5)	28.75 (10)	82.95 (4)	**91.89 (1)**	63.68 (7)	90.73 (2)
Gaussian	74.97 (4)	26.03 (9)	52.00 (7)	15.53 (10)	77.54 (2)	48.30 (8)	75.23 (3)	69.60 (5)	62.99 (6)	**78.96 (1)**
Hyper Plane	51.44 (5)	1.88 (8)	35.57 (7)	1.03 (10)	60.23 (2)	1.61 (9)	47.02 (6)	52.36 (4)	57.49 (3)	**61.26 (1)**
Electricity	63.34 (4)	37.63 (9)	55.71 (8)	36.77 (10)	68.20 (2)	65.59 (3)	62.32 (5)	61.87 (6)	57.61 (7)	**68.59 (1)**
Weather	66.04 (7)	66.71 (5)	62.43 (10)	67.38 (4)	66.12 (6)	68.89 (3)	63.56 (9)	**72.34 (1)**	65.17 (8)	69.56 (2)
GMSC	65.07 (5)	33.08 (10)	46.50 (8)	33.55 (9)	68.77 (3)	69.15 (2)	53.22 (7)	68.24 (4)	64.68 (6)	**70.48 (1)**
CovType35	95.37 (3)	94.23 (5)	93.24 (7)	95.50 (2)	91.85 (8)	91.12 (9)	93.86 (6)	94.92 (4)	85.80 (10)	**95.62 (1)**
CovType36	78.46 (6)	72.86 (9)	80.73 (5)	81.21 (4)	75.43 (8)	76.14 (7)	84.25 (3)	**87.11 (1)**	59.75 (10)	85.80 (2)
AvgRank	5.133 (5)	7.333 (7)	8.067 (10)	6.067 (6)	3.600 (3)	7.867 (9)	4.733 (4)	3.200 (2)	7.733 (8)	**1.267 (1)**

Table 3. Average recall values (%) of different algorithms.

Dataset	DDM-OCI	RE-DI	Learn++.NSE	ARF	DWMIL	ROS-U	DES-ICD	DSCB	EMRIL	AD-DES
Sine (sudden)	80.34 (6)	78.44 (8)	71.32 (9)	90.59 (5)	91.01 (4)	55.12 (10)	92.92 (2)	91.77 (3)	79.64 (7)	**93.67 (1)**
Sine (gradual)	66.30 (9)	74.59 (7)	73.54 (8)	88.82 (4)	89.38 (3)	55.40 (10)	**91.63 (1)**	87.47 (5)	79.80 (6)	91.30 (2)
SEA (sudden)	90.79 (6)	88.47 (8)	77.01 (9)	93.60 (4)	95.94 (2)	49.39 (10)	93.34 (5)	93.82 (3)	90.21 (7)	**96.05 (1)**
SEA (gradual)	89.29 (6)	88.56 (7)	78.00 (9)	93.16 (5)	96.34 (2)	59.73 (10)	93.68 (3)	93.49 (4)	80.46 (8)	**96.50 (1)**
Mixed (sudden)	85.18 (5)	72.27 (8)	71.23 (9)	87.24 (4)	88.43 (3)	50.79 (10)	83.77 (6)	88.73 (2)	75.92 (7)	**89.88 (1)**
Mixed (gradual)	73.77 (7)	68.20 (9)	73.59 (8)	86.66 (3)	86.97 (2)	50.32 (10)	84.34 (5)	86.32 (4)	75.73 (6)	**86.99 (1)**
Tree (sudden)	85.45 (6)	63.90 (9)	66.58 (7)	64.12 (8)	86.50 (4)	31.06 (10)	86.81 (3)	86.30 (5)	87.87 (2)	**88.27 (1)**
Tree (gradual)	81.11 (6)	59.72 (9)	67.14 (7)	62.04 (8)	87.01 (2)	15.55 (10)	85.66 (4)	85.92 (3)	81.78 (5)	**92.75 (1)**
Gaussian	61.46 (4)	10.78 (9)	29.61 (7)	5.15 (10)	66.87 (2)	24.56 (8)	63.96 (3)	51.39 (5)	43.75 (6)	**69.24 (1)**
Hyper Plane	37.94 (4)	0.45 (10)	15.67 (7)	1.03 (9)	54.12 (2)	1.14 (8)	28.36 (6)	31.86 (5)	45.99 (3)	**55.12 (1)**
Electricity	58.46 (3)	18.78 (9)	39.86 (8)	17.92 (10)	62.23 (2)	52.83 (4)	49.71 (6)	45.10 (7)	50.82 (5)	**64.69 (1)**
Weather	76.06 (5)	65.64 (8)	51.52 (9)	50.85 (10)	80.67 (3)	76.85 (4)	80.97 (2)	66.07 (7)	66.61 (6)	**81.19 (1)**
GMSC	51.41 (5)	13.44 (9)	24.21 (8)	12.23 (10)	52.61 (3)	52.24 (4)	36.11 (7)	52.85 (2)	45.67 (6)	**60.90 (1)**
CovType35	95.73 (2)	94.52 (4)	93.04 (7)	93.85 (6)	94.74 (3)	88.35 (9)	92.48 (8)	94.48 (5)	86.03 (10)	**97.06 (1)**
CovType36	74.63 (6)	71.00 (9)	75.14 (5)	73.04 (7)	81.08 (4)	72.09 (8)	87.07 (2)	84.06 (3)	52.58 (10)	**89.37 (1)**
AvgRank	5.333 (5)	8.200 (9)	7.800 (8)	6.867 (7)	2.733 (2)	8.333 (10)	4.200 (3)	4.200 (3)	6.267 (6)	**1.067 (1)**

Table 3 presents the average recall values obtained by AD-DES and these 9 comparison algorithms. The best results are shown in bold. It can be seen from Table 3 that AD-DES achieved the best performance on 14 out of all 15 datasets, with an average ranking of 1.067. For the Sine dataset with gradual concept drift, DES-ICD achieved the best performance, while the AD-DES achieved the second. Analysis shows that AD-DES has superior ability to identify the minority class instances, which is primarily attributed to the unique preprocessing strategy for handling imbalanced data. By adaptively selecting appropriate data balancing methods, each base classifier can learn better from the minority class instances, thereby enhancing the overall robustness of AD-DES. Notably, DWMIL performed well on most of the 15

datasets, consistently maintaining a high ranking. Furthermore, DES-ICD and DSCB exhibited nearly identical performance across all 15 datasets, both sharing an average ranking of third place.

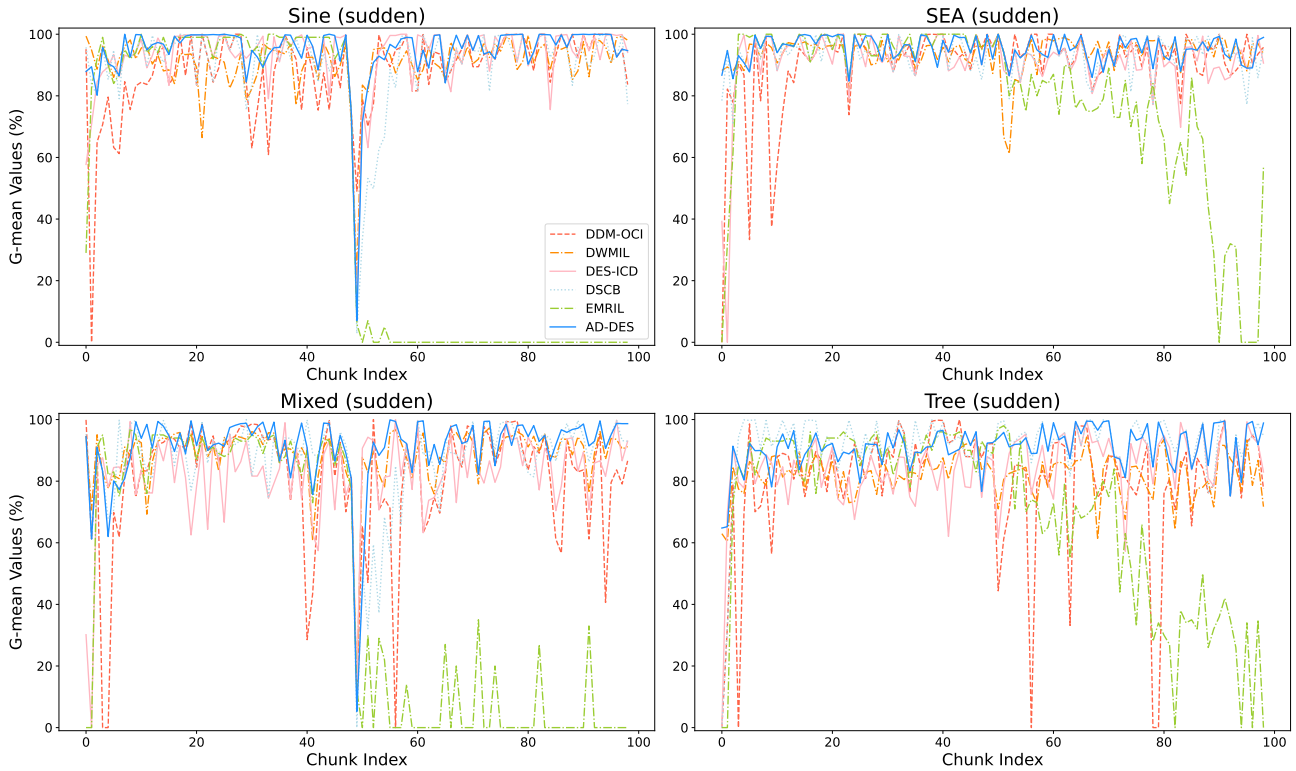


Figure 4. Variation trends of G-mean values for different algorithms across 4 datasets with sudden drift types.

In order to more intuitively demonstrate the advantages of AD-DES, Figures 4 and 5 present the trend changes in G-mean results obtained by several representative competitive methods and AD-DES across 8 synthetic datasets with sudden and gradual concept drift, respectively. In the two figures, the x-axis represents the specific data chunk index, with each chunk containing 500 instances, while the y-axis indicates the G-mean values corresponding to the data chunks. From Figures 4 and 5, it can be seen that the performance of the DDM-OCI algorithm was initially unsatisfactory on the SEA and Tree datasets with virtual drift. However, its performance improved significantly as more data chunks arrived. This improvement is primarily due to the limited amount of data available for learning during the early stages of the data streams. In contrast, with the application of the AER and the D-DES strategy, AD-DES demonstrated relatively stable and superior performance from the outset. Compared with virtual drift, real drift has a more pronounced influence on the model performance, as reflected in the Sine and Mixed datasets. On these datasets, most algorithms exhibited more drastic fluctuations in performance, especially around the midpoint of the datasets. In comparison, AD-DES showed a faster recovery in performance. This phenomenon is primarily due to the adaptive drift detection strategy integrated into AD-DES, which enables the model to adapt to new concepts quickly. Although DES-ICD demonstrated a slightly faster recovery speed at the midpoint of certain datasets, this was primarily due to its use of the latest data chunk as the validation dataset. However, this approach was less favorable for maintaining stable performance in the long-term. In contrast, AD-DES achieved more consistent

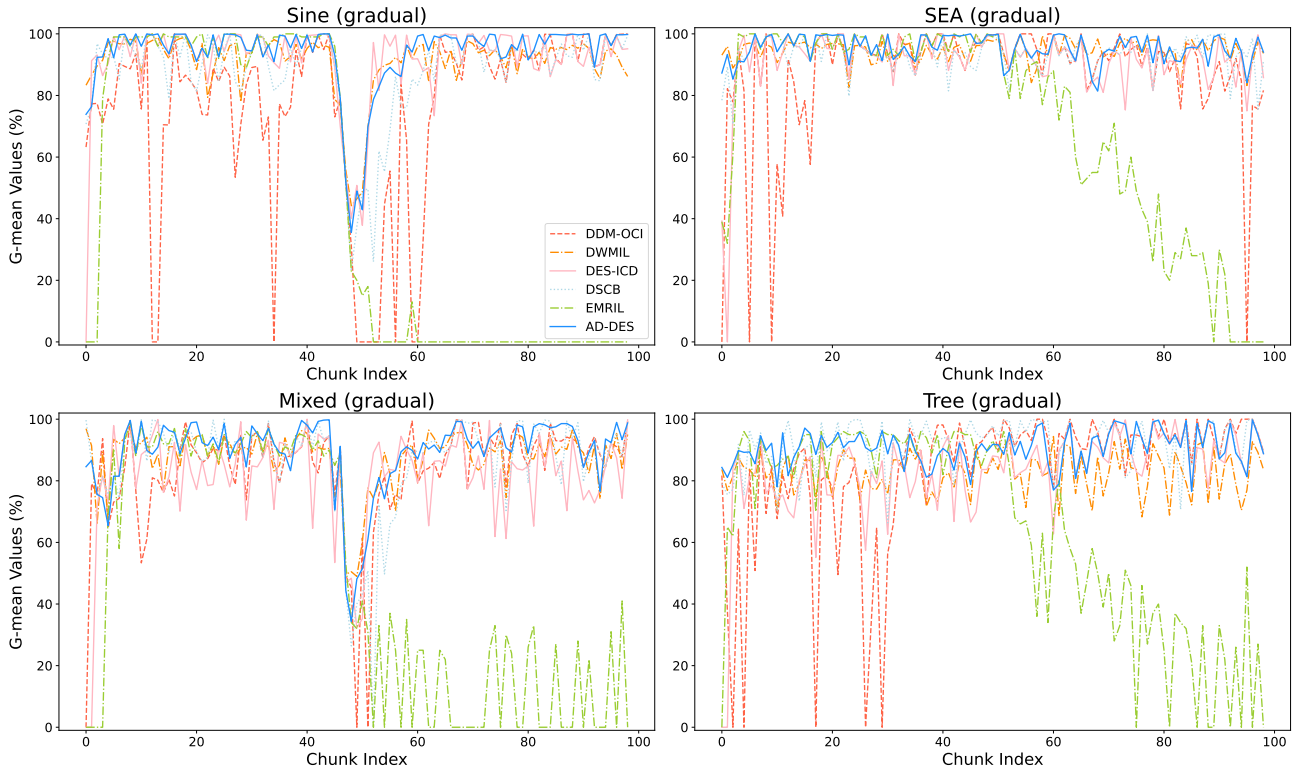


Figure 5. Variation trends of G-mean values for different algorithms across 4 datasets with gradual drift types.

and stable performance across the entire data stream, making it better suited for handling evolving data environments. In addition, it should be noted that while EMRIL exhibited outstanding performance during the early stages of the data streams, it struggled to adapt to concept drift involving the interchange of majority and minority classes over time. This limitation led to a marked drop in classification accuracy during the latter half of the streams—particularly when class reversal occurred—and was a major factor contributing to its relatively low overall ranking.

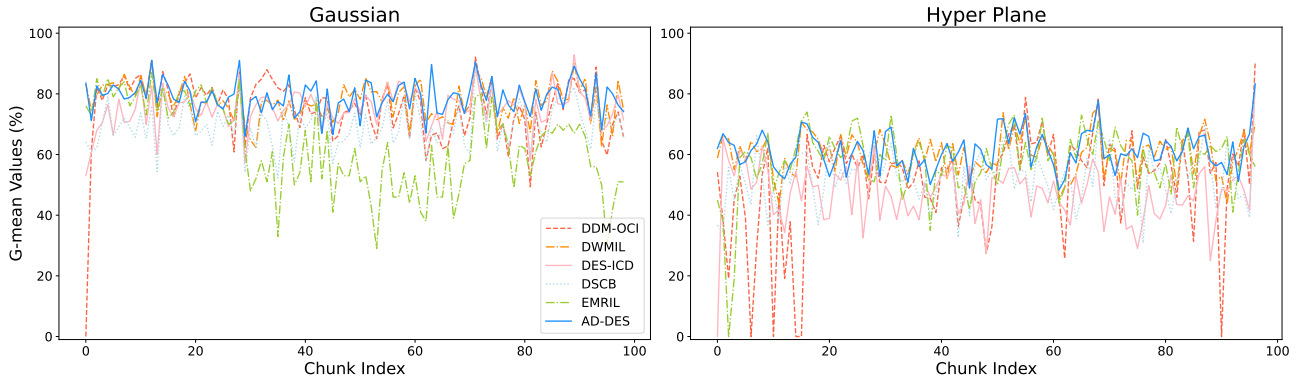


Figure 6. Variation trends of G-mean values for different algorithms across datasets with incremental drift types.

Furthermore, Figure 6 depicts the variations in G-mean values of AD-DES and several representative

algorithms across 2 synthetic datasets with incremental drift (i.e., Gaussian and Hyper Plane). It is evident that the incremental drift poses a greater challenge to the model performance compared to sudden (gradual) drift. Nonetheless, the synergistic integration of multiple strategies enables AD-DES to maintain good and stable performance, even in these high-dimensional and complex datasets.

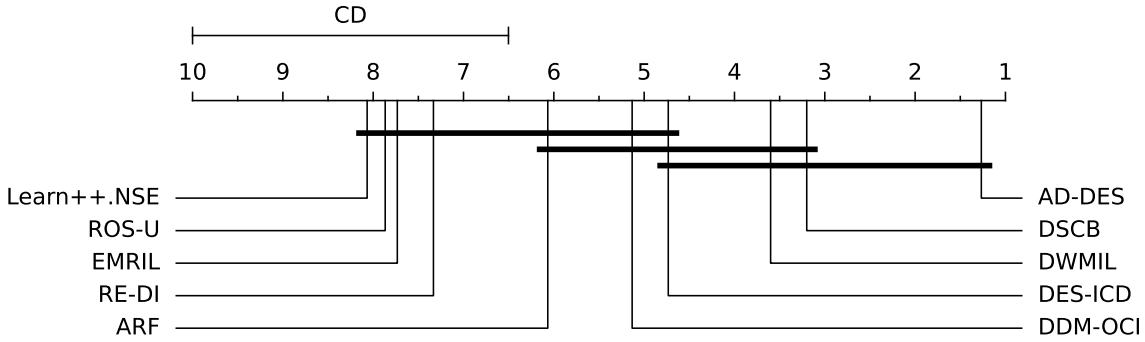


Figure 7. Nemenyi statistical significance test of G-mean values.

To provide a more comprehensive comparison between AD-DES and 9 comparison algorithms, a Friedman test and Nemenyi test were conducted on the G-mean values obtained by all these 10 algorithms. The results of the Friedman test show that the null hypothesis was rejected at a significance level of 0.05, indicating a significant performance difference between AD-DES and the 9 competitors, with a p-value of 0.00 and a chi-square statistic of 79.20. What is more, Figure 7 presents the experimental results achieved by AD-DES and 9 comparison algorithms in terms of a post-hoc Nemenyi test. In this figure, lower ranks indicate better performance, with a critical difference (CD) of 3.498. It is evident that AD-DES significantly outperforms all comparison algorithms, except for the DSCB, DWMIL, and DES-ICD algorithms.

In summary, these experimental results clearly demonstrate that AD-DES outperforms 9 comparison algorithms and exhibits superior adaptability in nonstationary environments.

5.2. Visual analysis of adaptive drift detection

To validate the adaptive drift detection mechanism in AD-DES more intuitively, Figure 8 illustrates the variation in the number of base classifiers and stored data chunks on these 15 datasets. In particular, for most of the time, the number of base classifiers and stored data chunks remained unchanged (at the predefined maximum) on 8 synthetic datasets with sudden (gradual) concept drift. This is because slight drifts were filtered by the adaptive drift detection mechanism, ensuring the optimal utilization of available data resources. However, significant concept drifts, such as those observed in the Sine (Mixed) dataset around the 50th chunk, have not been overlooked. These concept drifts represented a complete reversal of the concepts associated with the two classes, rendering the existing model ineffective. For severe drifts, the number of base classifiers and stored data chunks were promptly reduced by using the drift detection mechanism, which enabled the model to forget outdated concepts and rapidly adapt to new ones. For smaller yet still notable fluctuations, the number of base classifiers and stored data chunks were moderately decreased to balance the trade-off between discarding outdated concepts and retaining useful data. As a result, the effectiveness and rationality of the adaptive drift detection mechanism were validated effectively.

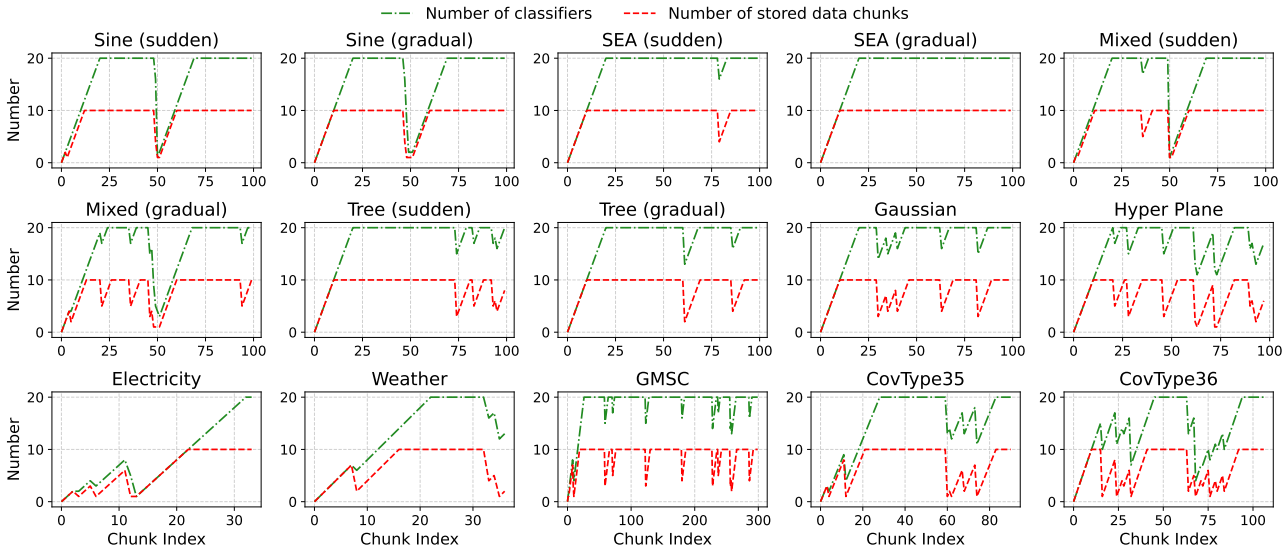


Figure 8. Variation trends in the number of classifiers and stored data chunks across 15 datasets.

Furthermore, it can be seen that the number of base classifiers and stored data chunks were periodically reduced on the Gaussian (Hyper Plane) dataset, as shown in Figure 8. This phenomenon reflected the characteristics of incremental drift. Especially, fluctuations in the model performance on the Hyper Plane dataset were more pronounced than on the Gaussian dataset, highlighting the greater learning challenges it presents.

Moreover, the specific types of concept drift have not been detected on 5 real-world datasets, as shown in Figure 8. Under the circumstances, the adaptive drift detection mechanism in AD-DES remained effective, where sudden drifts were flexibly handled within these datasets. For instance, the number of classifiers was decreased more slowly than the number of stored data chunks on the GMSC dataset with slight drift. This is because classifiers, refined through multiple iterations, were considered as more valuable resources and designed to persist longer than individual stored chunks. In contrast, the number of classifiers was promptly reduced by using the drift detector to adapt to the new concept when severe drifts were detected on the CovType36 dataset.

In conclusion, the adaptive drift detection mechanism enabled AD-DES to exhibit tailored adaptation processes to various types of concept drift across the 15 datasets. Therefore, the adaptive drift detection mechanism demonstrated superior robustness compared with those passive strategies.

5.3. Analysis of the impact of imbalance ratios on AD-DES

To further evaluate the performance of AD-DES under different imbalance ratios, five imbalance ratios, i.e., 2, 3.5, 10, 20, and 50, were generated for 8 synthetic datasets with sudden (gradual) drift. Under the circumstances, the average G-mean values achieved by AD-DES and 9 comparison algorithms were compared and analyzed, as shown in Figure 9. In particular, the value displayed above each algorithm represents its stability, with the calculation method described in Eq (5.1).

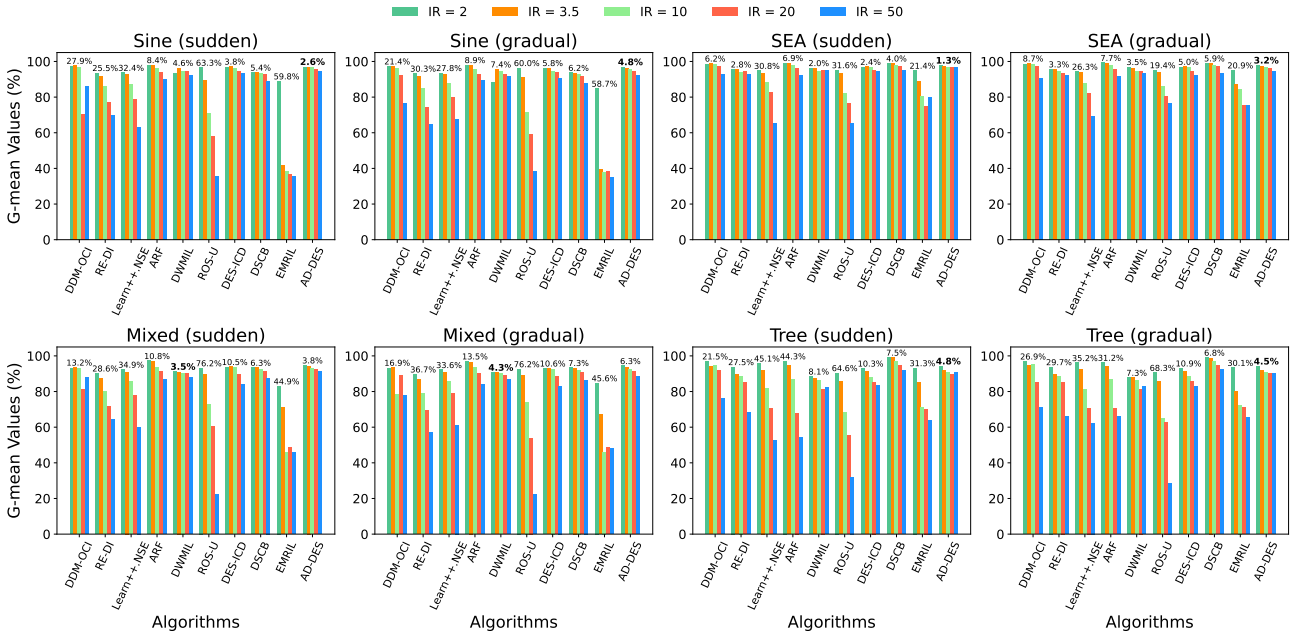


Figure 9. Performance analysis of 10 algorithms under different imbalance ratios.

$$\text{Stability} = \frac{\max f(\text{IR}) - \min f(\text{IR})}{\max f(\text{IR})} \tag{5.1}$$

where $\text{IR} \in \{2, 3.5, 10, 20, 50\}$, and $f(\text{IR})$ denotes the average G-mean value achieved by an algorithm under a given imbalance ratio. A lower stability value indicates greater robustness, and the smallest stability value for each dataset is highlighted in bold.

As shown in Figure 9, the performance of all 10 algorithms decreased as the imbalance ratio increased. However, AD-DES exhibited a much smaller decline in the performance compared with the other 9 algorithms on these 8 synthetic datasets with the increase of imbalance ratios. For these 8 synthetic datasets with severe imbalance (e.g., IR = 50), AD-DES significantly outperformed the 9 comparison algorithms. Furthermore, the stability values indicate that AD-DES achieved superior robustness compared to the 9 competitors on 6 out of 8 datasets. For the Mixed dataset, AD-DES ranked the second and DWMIL ranked the first in terms of the stability, while the overall performance of the former was better than the latter.

Overall, as these datasets were transitioned from slight to severe imbalance, the performance decline of AD-DES was limited to less than 5% in most cases. This phenomenon demonstrates that AD-DES is highly robust and better equipped to handle highly imbalanced datasets compared to 9 comparison algorithms.

5.4. Sensitivity analysis of key parameters

In this subsection, the influence of key parameters on the performance of AD-DES was explored, with the goal of identifying the optimal parameter settings for the algorithm. The key parameters in AD-DES include the maximum size of the classifier pool (M), the maximum number of stored data

chunks (S), the competence subregion size (K), the classification threshold parameters α and β of the IR, the drift detection threshold (Δ), and the maximum size of the drift detection window (T). To ensure the reliability of the analysis, extensive experiments were conducted on all 15 datasets, which were categorized into three groups based on drift types (i.e., sudden drift, gradual drift, and other types of drift). It should be noted that incremental drift was grouped with the other drift types due to the limited availability of datasets with incremental drift. For each experiment, only the parameter under investigation was adjusted, while all other conditions remained unchanged. In particular, these parameter settings were defined as follows: $M \in \{1, 5, 10, 15, 20, 25, 30\}$, $S \in \{3, 5, 8, 10, 15, 20\}$, $K \in \{1, 3, 5, 7, 9\}$, $\alpha \in \{2, 3, 4\}$, $\beta \in \{8, 9, 10\}$, $\Delta \in \{1, 1.1, 1.2, 1.3\}$, and $T \in \{3, 5, 7, 9\}$.

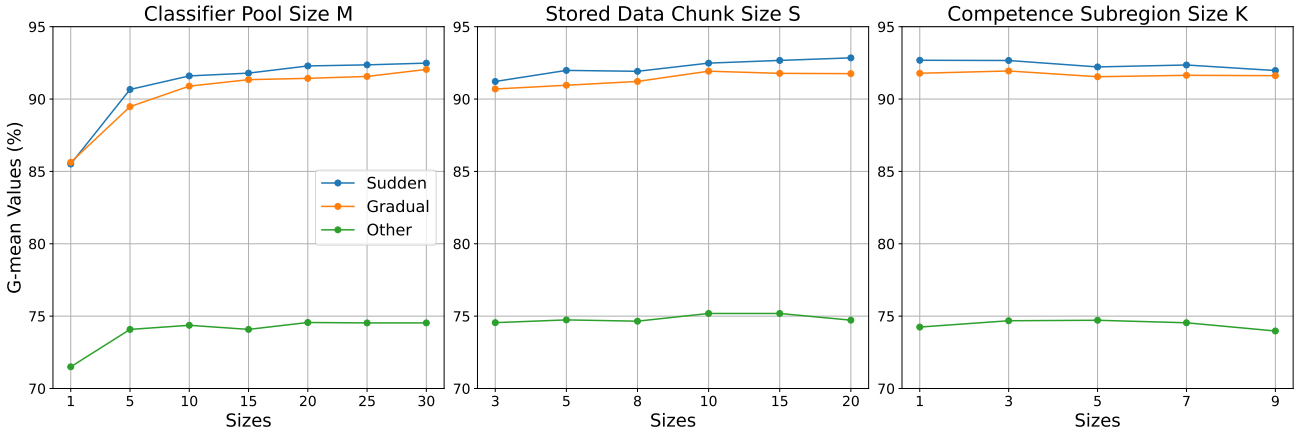


Figure 10. Average G-mean values of AD-DES under the influence of different parameters (M, S, and K) across 3 types of datasets.

Figure 10 presents the average G-mean values achieved by AD-DES under different values of three key parameters (M, S, and K). The experimental results were reported based on three types of drift, exhibiting the impact of each parameter on model performance. From the performance trends related to the maximum capacity of the classifier pool (M), it can be seen from Figure 10 that AD-DES has shown significant improvements in performance with the increase of M across all 15 datasets. What is more, the performance of AD-DES tended to stabilize when M reached 20. This suggests that a larger size of the classifier pool can enhance the classification accuracy of AD-DES. For the maximum number of stored data chunks (S), there was a slight improvement in the classification performance of AD-DES with the increase of S. This phenomenon could be mainly attributed to the adaptive drift detection mechanism where the model was allowed to learn from a greater variety of instances. However, excessively large S can lead to the accumulation of outdated instances, which may hinder the model's ability to adapt to new concepts. Regarding the size of the competence subregion (K), the overall performance of AD-DES with different values of K remained relatively stable across all 15 datasets. However, for the datasets with other types of drift, excessive noise would be introduced when the value of K is larger, while useful information may be excluded when the value of K is smaller. Both scenarios can negatively impact the model's overall performance.

In addition, Figure 11 presents the average G-mean values obtained by AD-DES under 9 combinations of the two key parameters (α and β), where darker colors indicate better performance and the highest values were clearly highlighted. As shown in the figure, the configuration with α set to 3 and β set to 9

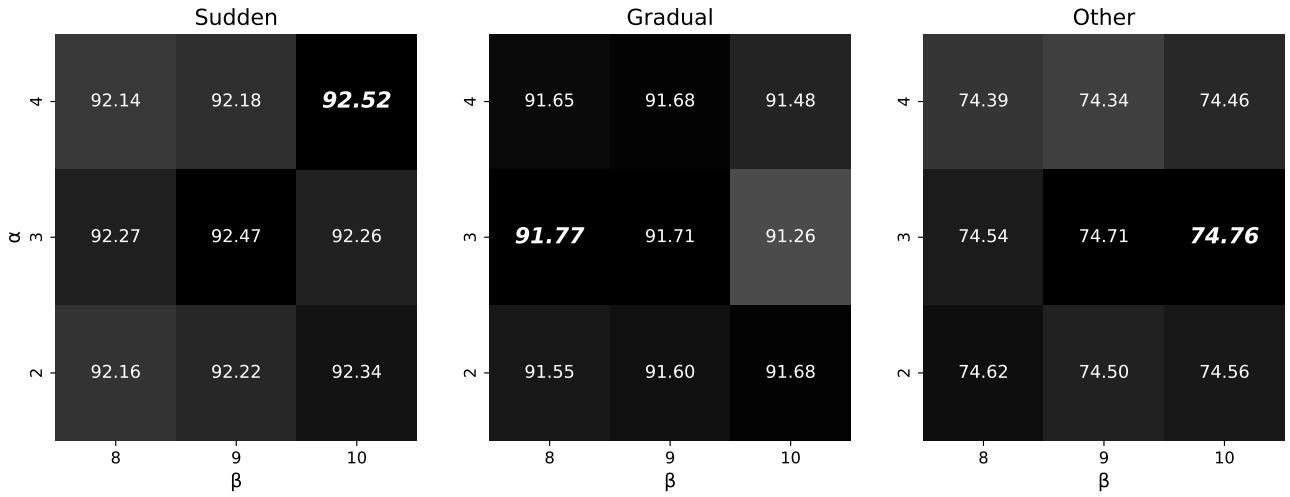


Figure 11. Average G-mean values of AD-DES under the combined influence of different parameters (α and β) across 3 types of datasets.

achieved the most balanced and robust performance across the three scenarios (Sudden, Gradual, and Other). It differed from the optimal score in each scenario by only about 0.05%, while consistently maintaining the second-best position. Overall, the performance differences among different parameter combinations are minor, indicating that the proposed AD-DES method is not highly sensitive to variations in α and β. This suggests that although the exact choice of these parameters may slightly affect local performance, their impact on the overall classification effectiveness is marginal. These results verify the stability of the AER strategy and confirm that the proposed thresholds provide a reasonable trade-off between adaptability and stability.

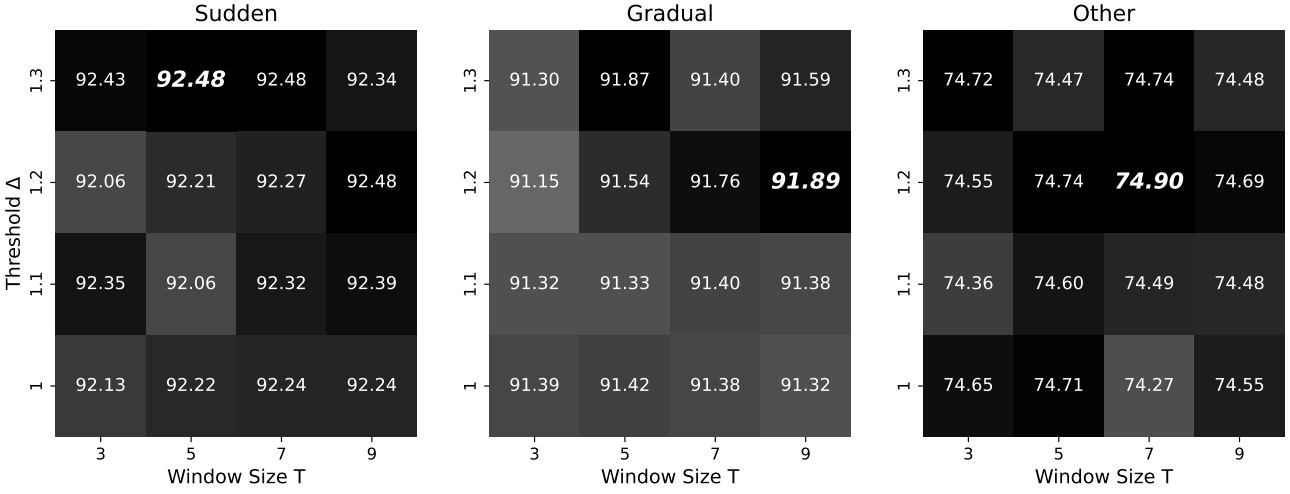


Figure 12. Average G-mean values of AD-DES under the combined influence of different parameters (Δ and T) across 3 types of datasets.

Furthermore, the average G-mean values obtained by AD-DES with 16 combinations of two key parameters (Δ and T) are illustrated in Figure 12. As shown in Figure 12, the optimal performance of AD-DES was achieved when Δ was approximately 1.2. In particular, smaller values of Δ made the

model overly sensitive to minor fluctuations since the valuable classifiers and stored data chunks were frequently discarded. For the maximum size of the drift detection window T, AD-DES with smaller values of T achieved better performance on the datasets with sudden drift, where the model was allowed to adapt to the new concept quickly. Conversely, AD-DES with larger values of T achieved better performance on the datasets with gradual drift, where the model was allowed to adapt to evolving concepts more effectively. For the datasets with other types of concepts, AD-DES with a moderate value of T obtained stable performance. Notably, AD-DES achieved the optimal results across all 15 datasets when T was set to approximately 7.

In summary, AD-DES consistently demonstrated excellent performance and robust adaptability across a wide range of parameter settings although variations in parameters influenced the performance of the model differently.

5.5. Analysis on the effectiveness of AD-DES components

To evaluate the contribution of different components to the overall performance of AD-DES, three variants of AD-DES were developed to conduct comparative experiments on all 15 datasets, where the sizes of data chunks were set to 300, 500, and 800, respectively. The three variants of AD-DES are described as follows:

1) AD-DES*wr*: The AER strategy in AD-DES was replaced by random undersampling for balancing the data chunks.
2) AD-DES*wd*: The DES technique in AD-DES was replaced by combining all base classifiers for predicting the results of test instances.
3) AD-DES*wf*: The drift detection mechanism in AD-DES was removed and the model was updated passively without adaptive responses to the concept drift.

Table 4. Average G-mean values (%) of AD-DES and its variants.

	Chunk Size = 300				Chunk Size = 500				Chunk Size = 800			
Dataset	AD-DES*wr*	AD-DES*wd*	AD-DES*wf*	AD-DES	AD-DES*wr*	AD-DES*wd*	AD-DES*wf*	AD-DES	AD-DES*wr*	AD-DES*wd*	AD-DES*wf*	AD-DES
Sine (sudden)	93.16	93.09	91.24	**93.57**	93.98	93.25	90.43	**94.36**	92.71	90.78	88.49	**94.01**
Sine (gradual)	92.21	91.53	90.90	**92.27**	92.66	91.61	88.92	**93.28**	92.36	91.76	87.03	**92.62**
SEA (sudden)	93.75	94.37	94.11	**94.44**	94.93	93.81	94.98	**95.67**	95.32	93.92	95.96	**96.16**
SEA (gradual)	94.22	94.55	94.38	**94.87**	94.53	94.57	95.04	**95.25**	95.34	96.18	95.88	**96.34**
Mixed (sudden)	88.05	87.70	88.24	**89.70**	89.36	87.43	86.23	**90.40**	87.86	85.98	86.15	**89.46**
Mixed (gradual)	85.12	84.10	85.25	**85.55**	87.01	85.66	86.36	**89.83**	87.40	86.78	85.10	**87.68**
Tree (sudden)	81.84	88.37	88.99	**89.59**	87.11	89.99	90.08	**90.51**	88.23	88.19	91.59	**91.74**
Tree (gradual)	82.83	88.26	88.55	**89.14**	88.18	90.34	90.48	**90.67**	88.60	88.65	90.71	**90.93**
Gaussian	79.63	**82.27**	80.67	81.17	78.81	**80.21**	77.88	78.29	76.90	**79.77**	73.22	74.57
Hyper Plane	**64.08**	61.73	60.35	61.90	**63.73**	62.31	60.61	61.62	**62.97**	61.98	61.25	61.83
Electricity	67.54	67.41	66.62	**68.11**	65.54	65.15	67.38	**68.07**	65.21	64.12	66.30	**67.76**
Weather	66.65	66.29	66.46	**67.23**	66.74	66.84	67.97	**68.79**	65.47	65.91	65.40	**67.26**
GMSC	68.21	62.58	67.69	**69.73**	67.11	63.17	70.18	**70.83**	66.35	63.02	**70.52**	69.86
CovType35	95.70	95.65	96.70	**96.91**	93.50	93.77	94.97	**95.57**	93.85	92.88	95.40	**95.68**
CovType36	85.97	74.69	86.84	**86.88**	84.14	72.80	84.70	**85.09**	83.07	70.80	83.54	**83.67**

Table 4 presents the average G-mean values of AD-DES and its three variants under different sizes of data chunks. Bold values indicate the best performance for a specific size of data chunk, while underlined values highlight the best performance across all sizes of data chunks. From Table 4, it can be seen that AD-DES consistently demonstrates significant advantages on 13 out of these 15 datasets. In

particular, AD-DES*wd* achieved the best performance on the Gaussian dataset, where the type of drift was smooth and continuous. In AD-DES*wd*, all base classifiers were integrated to help maintain the overall distribution of historical data. In contrast, using the DES strategy in AD-DES has introduced bias by prioritizing locally optimal classifiers. On the Hyper Plane dataset, characterized by inherent nonstationarity and complexity, stored data chunks were utilized to balance subsequent data chunks, which hindered the learning about new concepts. As a result, AD-DES*wr* achieved the best performance by training classifiers only on the most recent data chunks. For AD-DESwf, the results generally show a decline in performance compared with the complete AD-DES. Without adaptive responses to concept drift, the model failed to timely adjust to the evolving data distribution, particularly in the Sine and Mixed datasets. This passive updating strategy caused the accumulated outdated knowledge to interfere with the learning of new concepts, leading to lower G-mean values in most cases. These results confirm the importance of the drift detection mechanism in maintaining the robustness of AD-DES.

Furthermore, experimental results indicated that four methods have achieved better results on datasets with incremental drift (real-world datasets) when the size of the data chunk was smaller, as shown in Table 4. This was because larger data chunks exacerbated within-chunk drift, which would negatively impact classifier performance. Conversely, four methods have achieved better results on relatively stable synthetic datasets when the size of the data chunk was larger, where the classifiers would be able to obtain valuable information.

In conclusion, the three strategies are crucial for the performance of AD-DES to cope with the imbalanced data streams in nonstationary environments. They can substantially enhance the model's overall performance to a large extent.

6. Conclusions

Learning from imbalanced data streams in nonstationary environments presents a significant challenge in machine learning. To address this issue, we propose an adaptive dual dynamic ensemble selection (AD-DES) method for data streams. First, the adaptive equalization resampling (AER) strategy dynamically adjusts the resampling methods according to the IR of each data chunk. This enables AD-DES to maintain stable and reliable performance across datasets with a wide range of imbalance ratios. Second, the dual dynamic ensemble selection (D-DES) strategy refines the classifier pool through a two-stage filtering process, effectively selecting the most competent base classifiers. Finally, an adaptive drift detector dynamically adjusts the number of base classifiers and stored data chunks based on the detected drift intensity, allowing the model to adapt rapidly to evolving data distributions. Notably, AD-DES explicitly bridges the resampling, ensemble selection, and drift detection strategies through a shared chunk-based mechanism. Specifically, all stored data chunks are used to construct the validation dataset, with their number dynamically controlled by the drift detector and the resampling process, while simultaneously supporting class balance maintenance and competence evaluation. Experimental results on 10 synthetic datasets and 5 real-world datasets with different types of concept drift show that AD-DES consistently outperforms 9 state-of-the-art algorithms in terms of classification accuracy, robustness, and adaptability. These findings highlight that AD-DES provides an effective and generalizable solution for dynamic learning in nonstationary and imbalanced environments.

Our future work will be devoted to three aspects: (i) considering extending AD-DES to multi-class classification tasks in nonstationary environments; (ii) optimizing the algorithm's structure to reduce

computational costs; and (iii) exploring theoretically justified decay functions to further strengthen the theoretical foundation of the adaptive drift detection mechanism.

Acknowledgments

This study is funded by the Basic Public Welfare Research Project of Zhejiang Province (LGF22F020020), the National Natural Science Foundation of China (61806204, 21062367), the Scientific Research Starting Foundation of Zhejiang Sci-Tech University (20032309-Y), and the Research Fund Project of Zhejiang Sci-Tech University Longgang Research Institute (LGYJY2023003).

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare that there is no conflict of interest in the submission of this article, which has been approved for publication by all authors.

References

1. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, *ACM Comput. Surv.*, **46** (2014), 1–37. https://doi.org/10.1145/2523813
2. S. Wang, L. L. Minku, X. Yao, A systematic study of online class imbalance learning with concept drift, *IEEE Trans. Neural Networks Learn. Syst.*, **29** (2018), 4802–4821. https://doi.org/10.1109/TNNLS.2017.2771290
3. G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: A survey, *IEEE Comput. Intell. Mag.*, **10** (2015), 12–25. https://doi.org/10.1109/MCI.2015.2471196
4. H. He, E. A. Garcia, Learning from imbalanced data, *IEEE Trans. Knowl. Data Eng.*, **21** (2009), 1263–1284. https://doi.org/10.1109/TKDE.2008.239
5. X. Guo, Y. Yin, C. Dong, G. Yang, G. Zhou, On the class imbalance problem, in *2008 Fourth International Conference on Natural Computation*, IEEE, **4** (2008), 192–201. https://doi.org/10.1109/ICNC.2008.871
6. X. Wang, Q. Kang, M. Zhou, L. Pan, A. Abusorrah, Multiscale drift detection test to enable fast learning in nonstationary environments, *IEEE Trans. Cybern.*, **51** (2020), 3483–3495. https://doi.org/10.1109/TCYB.2020.2989213
7. B. Wei, J. Chen, L. Deng, Z. Mo, M. Jiang, F. Wang, Adaptive bagging-based dynamic ensemble selection in nonstationary environments, *Expert Syst. Appl.*, **255** (2024), 124860. https://doi.org/10.1016/j.eswa.2024.124860
8. A. Pesaranghader, H. L. Viktor, E. Paquet, Mcdiarmid drift detection methods for evolving data streams, in *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, (2018), 1–9. https://doi.org/10.1109/IJCNN.2018.8489260

9. S. Wang, L. L. Minku, D. Ghezzi, D. Caltabiano, P. Tino, X. Yao, Concept drift detection for online class imbalance learning, in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, IEEE, (2013), 1–10. https://doi.org/10.1109/IJCNN.2013.6706768
10. S. Ren, B. Liao, W. Zhu, K. Li, Knowledge-maximized ensemble algorithm for different types of concept drift, *Inf. Sci.*, **430** (2018), 261–281. https://doi.org/10.1016/j.ins.2017.11.046
11. J. Klikowski, M. Woźniak, Deterministic sampling classifier with weighted bagging for drifted imbalanced data stream classification, *Appl. Soft. Comput.*, **122** (2022), 108855. https://doi.org/10.1016/j.asoc.2022.108855
12. I. Czarnowski, Weighted ensemble with one-class classification and over-sampling and instance selection (wecoi): An approach for learning from imbalanced data streams, *J. Comput. Sci.*, **61** (2022), 101614. https://doi.org/10.1016/j.jocs.2022.101614
13. R. M. Cruz, R. Sabourin, G. D. Cavalcanti, Dynamic classifier selection: Recent advances and perspectives, *Inf. Fusion*, **41** (2018), 195–216. https://doi.org/10.1016/j.inffus.2017.09.010
14. A. H. Ko, R. Sabourin, A. S. Britto Jr, From dynamic classifier selection to dynamic ensemble selection, *Pattern Recognit.*, **41** (2008), 1718–1731. https://doi.org/10.1016/j.patcog.2007.10.015
15. B. Jiao, Y. Guo, D. Gong, Q. Chen, Dynamic ensemble selection for imbalanced data streams with concept drift, *IEEE Trans. Neural Networks Learn. Syst.*, **35** (2022), 1278–1291. https://doi.org/10.1109/TNNLS.2022.3183120
16. J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, *IEEE Trans. Knowl. Data Eng.*, **31** (2018), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857
17. T. R. Hoens, N. V. Chawla, Learning in non-stationary environments with class imbalance, in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, (2012), 168–176. https://doi.org/10.1145/2339530.2339558
18. Y. Lu, Y. M. Cheung, Y. Y. Tang, Adaptive chunk-based dynamic weighted majority for imbalanced data streams with concept drift, *IEEE Trans. Neural Networks Learn. Syst.*, **31** (2019), 2764–2778. https://doi.org/10.1109/TNNLS.2019.2951814
19. W. Liu, H. Zhang, Z. Ding, Q. Liu, C. Zhu, A comprehensive active learning method for multiclass imbalanced data streams with concept drift, *Knowl.-Based Syst.*, **215** (2021), 106778. https://doi.org/10.1016/j.knosys.2021.106778
20. N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: synthetic minority oversampling technique, *J. Artif. Intell. Res.*, **16** (2002), 321–357. https://doi.org/10.1613/jair.953
21. H. Han, W. Y. Wang, B. H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, in *International Conference on Intelligent Computing*, Springer, (2005), 878–887. https://doi.org/10.1007/11538059_91
22. C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in *Advances in Knowledge Discovery and Data Mining. PAKDD 2009. Lecture Notes in Computer Science()*, Springer, Berlin, Heidelberg, **5476** (2009), 475–482. https://doi.org/10.1007/978-3-642-01307-2_43

23. H. He, Y. Bai, E. A. Garcia, S. Li, Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, IEEE, (2008), 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969
24. P. Soltanzadeh, M. Hashemzadeh, Rcsmote: Range-controlled synthetic minority oversampling technique for handling the class imbalance problem, *Inf. Sci.*, **542** (2021), 92–111. https://doi.org/10.1016/j.ins.2020.07.014
25. J. Li, Q. Zhu, Q. Wu, Z. Fan, A novel oversampling technique for class-imbalanced learning based on smote and natural neighbors, *Inf. Sci.*, **565** (2021), 438–455. https://doi.org/10.1016/j.ins.2021.03.041
26. M. A. Tahir, J. Kittler, K. Mikolajczyk, F. Yan, A multiple expert approach to the class imbalance problem using inverse random under sampling, in *Multiple Classifier Systems. MCS 2009. Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, **5519** (2009), 82–91. https://doi.org/10.1007/978-3-642-02326-2_9
27. N. S. Kumar, K. N. Rao, A. Govardhan, K. S. Reddy, A. M. Mahmood, Undersampled k-means approach for handling imbalanced distributed data, *Prog. Artif. Intell.*, **3** (2014), 29–38. https://doi.org/10.1007/s13748-014-0045-6
28. W. W. Ng, J. Hu, D. S. Yeung, S. Yin, F. Roli, Diversified sensitivity-based undersampling for imbalance classification problems, *IEEE Trans. Cybern.*, **45** (2014), 2402–2412. https://doi.org/10.1109/TCYB.2014.2372060
29. W. C. Lin, C. F. Tsai, Y. H. Hu, J. S. Jhang, Clustering-based undersampling in class-imbalanced data, *Inf. Sci.*, **409** (2017), 17–26. https://doi.org/10.1016/j.ins.2017.05.008
30. C. F. Tsai, W. C. Lin, Y. H. Hu, G. T. Yao, Under-sampling class imbalanced datasets by combining clustering analysis and instance selection, *Inf. Sci.*, **477** (2019), 47–54. https://doi.org/10.1016/j.ins.2018.10.029
31. J. Ren, Y. Wang, M. Mao, Y. M. Cheung, Equalization ensemble for large scale highly imbalanced data classification, *Knowl.-Based Syst.*, **242** (2022), 108295. https://doi.org/10.1016/j.knosys.2022.108295
32. 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. https://doi.org/10.1016/j.inffus.2017.02.004
33. M. Woźniak, P. Zyblewski, P. Ksieniewicz, Active weighted aging ensemble for drifted data stream classification, *Inf. Sci.*, **630** (2023), 286–304. https://doi.org/10.1016/j.ins.2023.02.046
34. H. Wang, Z. Abraham, Concept drift detection for streaming data, in *2015 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Killarney, (2015), 1–9. https://doi.org/10.1109/IJCNN.2015.7280398
35. S. Wang, L. L. Minku, X. Yao, Resampling-based ensemble methods for online class imbalance learning, *IEEE Trans. Knowl. Data Eng.*, **27** (2014), 1356–1368. https://doi.org/10.1109/TKDE.2014.2345380

36. A. Bernardo, H. M. Gomes, J. Montiel, B. Pfahringer, A. Bifet, E. D. Valle, C-smote: Continuous synthetic minority oversampling for evolving data streams, in *2020 IEEE International Conference on Big Data (Big Data)*, IEEE, (2020), 483–492. https://doi.org/10.1109/BigData50022.2020.9377768
37. K. Malialis, C. G. Panayiotou, M. M. Polycarpou, Online learning with adaptive rebalancing in nonstationary environments, *IEEE Trans. Neural Networks Learn. Syst.*, **32** (2020), 4445–4459. https://doi.org/10.1109/TNNLS.2020.3017863
38. A. Cano, B. Krawczyk, Rose: robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams, *Mach. Learn.*, **111** (2022), 2561–2599. https://doi.org/10.1007/s10994-022-06168-x
39. G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, *IEEE Trans. Knowl. Data Eng.*, **25** (2012), 2283–2301. https://doi.org/10.1109/TKDE.2012.136
40. H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, et al., Adaptive random forests for evolving data stream classification, *Mach. Learn.*, **106** (2017), 1469–1495. https://doi.org/10.1007/s10994-017-5642-8
41. P. Zyblewski, R. Sabourin, M. Woźniak, Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams, *Inf. Fusion*, **66** (2021), 138–154. https://doi.org/10.1016/j.inffus.2020.09.004
42. R. Zhu, Y. Guo, J. H. Xue, Adjusting the imbalance ratio by the dimensionality of imbalanced data, *Pattern Recognit. Lett.*, **133** (2020), 217–223. https://doi.org/10.1016/j.patrec.2020.03.004
43. R. Kumari, J. Singh, A. Gosain, Impact of class imbalance ratio on ensemble methods for imbalance problem: A new perspective, *J. Intell. Fuzzy Syst.*, **45** (2023), 10823–10834. https://doi.org/10.3233/JIFS-223333
44. J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in *Advances in Artificial Intelligence – SBIA 2004. SBIA 2004. Lecture Notes in Computer Science()*, Springer, Berlin, Heidelberg, **3171** (2004), 286–295. https://doi.org/10.1007/978-3-540-28645-5_29
45. W. N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, (2001), 377–382. https://doi.org/10.1145/502512.502568
46. Y. Lu, Y. M. Cheung, Y. Y. Tang, Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift, in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, (2017), 2393–2399. https://doi.org/10.24963/ijcai.2017/333
47. G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, (2001), 97–106. https://doi.org/10.1145/502512.502529
48. J. Z. Kolter, M. A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, *J. Mach. Learn. Res.*, **8** (2007), 2755–2790. https://jmlr.org/papers/v8/kolter07a.html
49. A. Liu, J. Lu, G. Zhang, Diverse instance-weighting ensemble based on region drift disagreement for concept drift adaptation, *IEEE Trans. Neural Networks Learn. Syst.*, **32** (2020), 293–307. https://doi.org/10.1109/TNNLS.2020.2978523

50. R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, *IEEE Trans. Neural Networks*, **22** (2011), 1517–1531. https://doi.org/10.1109/TNN.2011.2160459

51. M. Usman, H. Chen, Emril: Ensemble method based on reinforcement learning for binary classification in imbalanced drifting data streams, *Neurocomputing*, **605** (2024), 128259. https://doi.org/10.1016/j.neucom.2024.128259

52. H. Zhang, W. Liu, S. Wang, J. Shan, Q. Liu, Resample-based ensemble framework for drifting imbalanced data streams, *IEEE Access*, **7** (2019), 65103–65115. https://doi.org/10.1109/ACCESS.2019.2914725

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

AIMS Press