Information Sciences 654 (2024) 119821

Contents lists available at ScienceDirect

Information Sciences

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

Dynamic budget allocation for sparsely labeled drifting data streams

Gabriel J. Aguiar *, Alberto Cano

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

ARTICLE INFO

Keywords:
Data streams
Active learning
Machine learning

ABSTRACT

Learning from non-stationary data streams is inherently challenging due to their evolving nature and concept drift. Furthermore, the assumption that all instances come labeled is often impractical in real-world applications. Many strategies have been proposed to tackle learning from sparsely labeled data streams. However, they typically rely on fixed labeling budgets, which can be a limitation in the context of drifting data streams. In this study, we introduce a novel active learning strategy that dynamically manages the labeling budget to optimize its utilization and adapt promptly to concept drift. Our approach continuously monitors the data stream for concept drift, and upon detecting such drift, it dynamically increases the maximum labeling budget for a predefined time window. This adjustment provides the classifier with more flexibility to adapt to the new concept. We conducted experiments using 7 synthetic data generators encompassing various drifting scenarios and 7 real-world data streams with different labeling budgets. Our results demonstrate that offering a flexible budget to the classifier can significantly enhance performance compared to merely increasing a fixed budget. Notably, our strategy outperformed state-of-the-art active learning strategies, all while maintaining a comparable or lower number of labeled instances. Experiments are available at https://github.com/gabrieljaguiar/DBAL.

1. Introduction

Recent advancement has enabled the collection, integration, and analysis of vast amounts of data generated at high speeds. However, this creates new challenges for traditional machine learning methods, since they were originally designed to learn from static datasets. Data streams [1] are potentially unbounded sequences of data instances, with non-stationary distributions that change over time, a phenomenon referred to as concept drift [2,3]. Concept drift poses challenges for algorithms dedicated to online learning, potentially degrading predictive performance as previously acquired knowledge may become obsolete for recent instances. To address concept drift, data stream mining algorithms utilize either explicit drift detectors [2] or implicit adaptation mechanisms [4].

Considering this backdrop, plenty of algorithms were proposed in recent years to learn from data streams and tackle concept drift [3–5]. The substantial attention of the research community toward supervised data stream mining has led to the development of classification algorithms suitable for non-stationary streaming data. Nevertheless, most of these algorithms assume a fully super-

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

https://doi.org/10.1016/j.ins.2023.119821
Received 28 March 2023; Received in revised form 26 October 2023; Accepted 26 October 2023

Available online 31 October 2023

vised scenario, where class labels are assumed to be immediately available upon predicting an instance's label. Although it makes sense from the perspective of data stream mining principles, it ignores the issue of how to obtain the label for every single instance. If obtaining the true class label was an easy trivial task and we would have access to a theoretical oracle that would label every instance, there would be no need for any classification task. Consequently, many classifiers proposed in the literature, which demonstrate excellent results in fully supervised scenarios, may not be directly applicable in real-world situations with sparsely labeled data [3].

To address the challenge of learning with limited access to labels, Active Learning (AL) has emerged in the literature [6,7]. It is noteworthy that a significant portion of AL research has primarily focused on static scenarios [8]. In contrast, solutions tailored for online scenarios, where data streams evolve continuously, are relatively scarce [9–11]. Online AL solutions work by selecting only a limited number of instances for labeling under a fixed budget which is allocated approximately uniformly over the stream. Considering the evolving and drifting nature of data streams, imposing a fixed budget over the whole data stream can be seen as a constraint that may prevent the classifier ability to adapt swiftly during drift periods, where more labeled instances are crucial. Furthermore, in many cases, the choice of budget values in the literature is not grounded in experimental evidence or data-specific information but rather arbitrary values of 1%, 5%, 10%, etc. Additionally, since budget values are fixed, it is impossible to predetermine whether the classifier's performance could be significantly enhanced by small increases in the number of labeled instances. Therefore, our paper introduces a novel methodology for dynamically adjusting the budget allocation throughout the stream, allocating more budget when concept drifts are detected to facilitate quick adaptation.

Research Goal. To propose an active learning strategy that enhances classifier performance in drifting data streams by endowing the classifier with flexibility via dynamic budget management.

Overview and main contributions. We propose a novel methodology for flexible and dynamic budget management, which attempts to optimize the budget usage over the stream allocating more budget where it is necessary after a concept drift and reducing the budget in stationary periods of the stream. Our methodology increases the budget allowance when a concept drift is detected in order to help the base classifier to adapt quickly to the new concept. We compared the proposed method with state-of-the-art AL methods and with 5 base classifiers from different families applied to streams with multiple concept drifts. The analysis of the performance of the proposed method allowed us to better understand how to allocate budget on a drifting stream and also demonstrates that our method overperforms the reference active learning strategies in the majority of the scenarios. The main contributions of the paper can be summarized as follows:

- **Novel methodology.** We present a novel methodology that leverages budget flexibility over the data stream to optimize the allocation of label queries to achieve superior performance. Our approach allows the classifier to dynamically determine where to allocate the budget, enabling it to effectively adapt quickly to changing data stream characteristics.
- **Concept drift aware active learning method.** We propose a new active learning method, classifier agnostic, based on budget flexibility that optimizes budget usage taking into account concept drifts.
- **Flexibility to change.** The proposed method has a quicker response to concept drift than state-of-the-art active learning strategies due to its flexibility when allocating labeling budget. This flexibility in budget allocation is a key feature of our proposed methodology, as it enables the classifier to optimize instance labeling while maintaining budget constraints.
- **Extensive and reproducible experimental study.** We compared the proposed method in a plethora of scenarios and state-of-the-art active learning strategies using a variety of base classifiers, in order to evaluate if the dynamic allocation of budget is able to overperform traditional active learning strategies with fixed budgets.

This paper is organized as follows. Section 2 provides a background on data stream mining. Section 3 presents the foundations of active learning for data streams. Section 4 describes the proposed methodology for dynamic budget allocation based on concept drift detection. Section 5 introduces the experimental setup and methodology. Section 6 presents and analyzes the results of the experimental study. Finally, Section 7 discusses the open challenges and future directions and summarizes the conclusions.

2. Data streams

A data stream can be defined as a potentially unbounded sequence of ordered instances arriving over time at a system. Learning from data streams imposes certain constraints to the classifiers [1]. Instances will become available at given time intervals one by one (online scenario) or in chunks (block scenario). Due to its unknown and ever-expanding size, the stream cannot be stored entirely in memory, therefore only a limited number of instances may be stored. Some classifiers will keep only the most recent instances, while some store a selection of old instances and use them to remember previous concepts [4]. Statistics about the stream are also extracted since those characteristics may evolve and it is important to keep track them during the continuous learning process.

We can define a stream S as a sequence $< s_1, s_2, \ldots, s_\infty >$ of data instances. If we have a scenario where $s_i = (X, y)$, we consider it a supervised scenario, where $X = [x_1, x_2, \ldots, x_f]$ with f being the dimensionality of the feature space, and y as the class label, which may or may not be available on arrival.

The main characteristics of data streams can be summarized as follows [1]:

- **Volume.** Data streams are potentially infinite collections of data that constantly flood the system, they cannot be stored and must be processed and discarded. This imposes limitations on the computational resources.

- **Velocity.** New data is generated continuously and often in rapid bursts, leading to high-speed data streams. This obligates the classifier to work in real time. Instances should be analyzed and processed fast enough so the classification model incorporates the current state of the stream.
- **Non-stationary.** Data distribution and class boundaries are affected by the ever-evolving nature of streams. Changes in class distributions and emerging/fading of classes may happen over time. This phenomenon is known as concept drift and will be discussed further in this section.
- **Veracity.** Data streams may also be affected by noise, missing values, and injection of adversarial patterns. A fully labeled stream is almost impossible to achieve due to cost and time requirements, leading to the necessity of learning from sparsely labeled data streams.

2.1. Concept drift

Data streams experience a phenomenon called concept drift [3,12]. Each instance arrives at a time t and is randomly generated according to a probability distribution D_t. If D_t is equal to D_{t+1} for any given t, it means that we are dealing with a stationary stream. However, when we look at real-life scenarios, most data streams are subject to change, leading to the notion of non-stationary streams [5], i.e., given two instances arriving in time t and $t + C$, we observe $D_t \neq D_{t+C}$, therefore, a concept drift happened.

We should consider the following factors when analyzing concept drift:

- **Influence of the decision boundaries.** Concept drift can be arranged into two categories: virtual and real. Virtual drift does not affect the decision boundaries, because it is a change in the unconditional probability distribution $P(x)$. It still needs to be detected because it may trigger false alarms and force unnecessary adaption. On the other hand, real concept drift affects decision boundaries. Consequently, the model needs to adapt to new distribution, otherwise predictive performance will decrease.
- **Speed of change.** Regarding the speed of change we can classify drifts as incremental, gradual, and sudden. Incremental drift generates a sequence of intermediate states between the old and new concepts. Gradual drift oscillates between instances coming from both old and new concepts, with the new concept becoming more and more frequent over time. Sudden drift instantaneously switches between old and new concepts.

Therefore, building classifiers for data streams must take into account the presence of concept drift and adapt to it. There are two strategies to tackle concept drift: explicit and implicit. Explicit strategies are managed by an external tool called concept drift detectors [13,14]. They monitor characteristics of the stream or classifier performance in order to detect a change point and raise a signal when a drift is detected so that the classifier can start its adaptation process [15]. The latter approach is embedded in the classifier, which can adjust itself to new concepts [16,12]. They are usually based on sliding windows or online learners [17]. Ensemble solutions are often used for drifting streams [3].

3. Active learning for data streams

While plenty of research has been done for fully supervised data stream mining [3–5], the assumption that labels are immediately available upon prediction is simplistic in real-world applications. In practice, domain experts provide true class labels to ensure the highest level of certainty. However, this service comes at a cost, both in terms of money and time [18], and isn't feasible for every incoming instance [19]. Although some domains, like stock change or weather prediction, offer label acquisition at no additional cost, label latency remains a consideration [20].

One solution to address this challenge is Active Learning (AL) [5,21]. AL focuses on methods for selecting the most important instances according to specified criteria to enhance the learning system and optimize data stream processing costs [22]. It is not only about finding the most valuable instances for one batch of data, but additionally exploring the search space in order to find sub-spaces that are no longer valid and select instances from that region in order to ensure proper adaptation [5]. In all active learning strategies, a budget, i.e., the maximum number of instances that can be queried for labels, is a crucial component. The simplest active learning method for the streaming scenario is random selection [5], which randomly samples instances based on a predefined probabilistic distribution and decides whether to query their labels based on the available budget. Dynamic active learning strategies display more effectiveness when it comes to recovering from changes and preventing long drops in performance [23]. The most popular dynamic strategies rely on classifier (un)certainty or support functions. Uncertainty-based strategies aim to identify instances for which the classifier exhibits the highest degree of uncertainty or variability in its predictions, often selecting instances near the decision boundary [24,25]. These samples are likely to be the most informative and could potentially improve the model's performance [20]. To implement uncertainty strategies, the active learning algorithm computes the uncertainty or variability associated with each sample in the stream. There are several ways to measure uncertainty, including the prediction entropy, the prediction margin, and the disagreement between multiple models [26,27]. Some of the traditional state-of-the-art strategies can be grouped as follows:

- Random selection (`AL-R`): For each instance, a random number is drawn from a probabilistic distribution, and the decision to query the label is based on the available spending budget. This method can be naive since it does not look for more representative

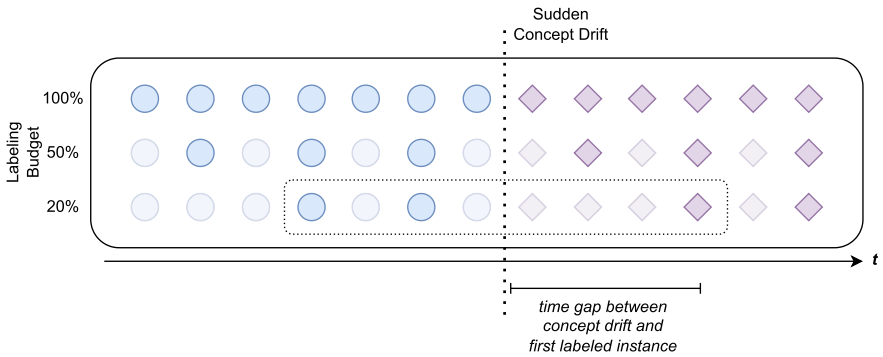


Fig. 1. An example of the problem of sparsely labeled streams in the presence of concept drift. Different shapes and colors represent different concepts and not labeled instances are transparent. The lower the budget the longer it takes to recognize a concept drift.

instances, however, it can be very useful in certain scenarios such as estimating class imbalance [4] and can be combined with more robust active learning strategies [28].

- Uncertainty-based strategies: Uncertainty-based strategies aim to select the instances that exhibit the highest level of variability or uncertainty in their predictions by the model. Therefore, the difference between them relies on how they select the threshold to determine if the label will be queried or not. The simplest strategy is to fix one value for the whole stream, which is known as Fixed Uncertainty (`AL-FU`), which can achieve good results on stationary scenarios. For drifting scenarios, dynamically changing the threshold is preferable. The Variable Uncertainty (`AL-VU`) and Random Variable Uncertainty (`AL-RVU`) strategies update the threshold for each instance, increasing it when a label is queried and decreasing it when a label is not queried. Finally, the Selective Sampling (`AL-SS`) strategy is based on sampling a number from the Bernoulli distribution and comparing it with a threshold computed with the budget and the uncertainty of the classifier.

However, even with the existing methods, concept drift is a challenge that is not completely addressed. For instance, as depicted in Fig. 1, the gap between labeled instances widens as the labeling budget diminishes, making it increasingly challenging to promptly detect concept drift. Also, certainty-based methods may fail when a concept drift happens, as a classifier may exhibit high confidence in a misclassified instance, resulting in the label not being queried. *Lughofer* et al. [29] proposed two techniques for addressing concept drift in label-scarce scenarios. The first method employs single-pass active learning filters to identify the most informative samples for supervised classification. The second technique considers the overlap between two classes in a classifier's output certainty distribution. The authors adapted the standard Page–Hinkley test (PHT) [30] to a faded version that assigns more weight to recent statistics. This modification showed good performance in drift detection. Nevertheless, it is essential to note that selecting the most relevant instances for detecting concept drift does not necessarily guarantee their relevance to a classifier. *Castellani* et al. [31] proposed a new active learning strategy to address the challenge of delayed labeling in data streams. Their approach involves adjusting the budget when a concept drift is detected and subsequently reducing it after a period of time. Nevertheless, this strategy still relies on a fixed budget, as even though the budget is temporarily increased, its new value is set below the standard budget value after the time window to compensate for the increase. Consequently, it lacks the necessary flexibility to effectively adapt the classifier in drifting stream scenarios.

Additionally, several frameworks have been proposed to tackle this issue. One approach incorporates oversampling techniques [32,33], generating new samples based on error metrics or class imbalance ratios. While these methods work effectively in addressing class imbalance, they may not account potential drifts in decision boundaries or feature drifts that may not impact significantly class ratios. Self-labeling techniques have also been employed to utilize instances not selected by the active learning strategy [34,35]. While pseudo-labels can provide significant benefits in a sparsely labeled scenario, it is crucial to remain cautious of the confirmation bias issue. This arises when the model starts relying on potentially incorrect self-created labels. Ensemble methods [36,37] have been proposed to address sparsely labeled drifting streams. These methods involve monitoring the stream with a concept drift detector and replacing some of the classifiers within the ensemble when drift is detected. In addition, ensemble or committee-based methods utilize the disagreement among base classifiers to determine whether to query the class label or not. However, one of the most powerful characteristics of ensembles is their intra-classifier diversity, which can result in excellent predictive performance but also unnecessary consumption of the labeling budget. Some of these frameworks embed mechanisms to deal with class imbalance [38,9]. Nevertheless, all these frameworks and algorithms rely on base active learning strategies, and therefore setting a fixed budget remains a challenge.

To summarize, all methods rely on fixed and uniformly distributed budgets all over the stream, despite the understanding that drifting periods demand more labeled instances for quicker adaptation. In order to tackle the concept drift problem on sparsely labeled streams, increasing the budget when a concept drift is detected may help the classifier to adapt quickly, and after the new concept is learned decrease the budget to a lower value in order to monitor possible new drifts and do not exceed the number of allowed instances to label.

4. Proposed active learning framework based on dynamic budget

When it comes to learning on a budget, the most straightforward solution to improve predictive performance is, if feasible, to increase the budget, providing the classifier access to more labeled instances. However, practical constraints such as time and cost may limit the ability to do so frequently. As a solution, we propose dynamically adjusting the labeling budget over the stream in order to optimize the learning process, particularly in scenarios involving concept drift.

Existing active learning strategies work with a budget that is fixed for the whole stream. However, it is possible to allocate the budget more effectively, querying less frequently during stationary periods and more frequently when needed, such as after detecting a concept drift. We propose a framework that is classifier agnostic and introduces a dynamic budget. The dynamic change in the budget will be triggered by concept drift, in order to help the classifier to adapt quickly to new concepts and avoid prolonged performance drops. In the following, we present our framework DBAL (Dynamic Budget Active Learning).

The framework consists of three modules:

- **Base classifier:** DBAL is classifier agnostic, therefore we can employ any incremental learning algorithm for data streams.
- **Concept drift detector:** A drift detector for monitoring the changes in the stream and trigger actions when a drift is detected.
- **Active Learning strategy:** A module for selecting the most valuable instances for the label query process under given budget constraints.

4.1. Base classifier

The classifier provides a class prediction for every single instance. The classifier must be able to return probabilistic values which will be used by the active learning strategy to decide if the label will be queried or not. If the instance label is queried, the base classifier will be updated with the instance. It is not a requirement, but it is important that the classifier should be able to learn in a purely online environment in order to achieve the best possible performance, rather than learning in batches. Some popular classifiers that can be used in our framework are Naive Bayes [39], Hoeffing Tree (HT) [40], Adaptive Random Forest (ARF) [41], and Leveraging Bagging (LB) [42].

4.2. Concept drift detector

The concept drift detector performs as an indicator of when we should increase our budget. It monitors the performance of the base classifier and the data distribution in order to raise an alert when a drift occurs. Any concept drift detector can be used such as Drift Detection Method [43] or Adaptive Windowing (ADWIN) [44]. In our framework, we used two ADWIN monitors with different settings, one more sensible used as a warning monitor which can be triggered more easily and helps prepare the classifier for a possible drift, and a second drift monitor which is used as a concept drift detector.

ADWIN (ADaptive WINdowing) [44] is a widely acclaimed drift detection method that comes with mathematical guarantees. It efficiently maintains a variable-length window of recent items, ensuring the preservation of the unchanged data distribution within the window. This window is divided into two sub-windows, namely W_0 and W_1, which are employed to assess the occurrence of any changes. ADWIN compares the averages of W_0 and W_1 to confirm their correspondence to the same distribution. If the distribution equality no longer holds, it signifies the presence of concept drift. Upon detecting drift, W_0 is replaced by W_1, and a new W_1 is initialized. To determine whether the two sub-windows correspond to the same distribution, ADWIN utilizes a significance value $\delta \in (0, 1)$.

To monitor drift, ADWIN utilizes a uni-variate distribution. In our approach, we leverage the error distribution of the base classifier to effectively monitor concept drifts, as suggested in the existing concept drift literature [44]. The error distribution is binary, and we determine the value to be added to ADWIN windows using the equation:

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

where ϕ represents the error distribution, x represents a new instance, y denotes the correct class, and L the base classifier. By employing this approach, we can effectively detect concept drifts when there are changes in the classifier error distribution. Consequently, our method is less susceptible to virtual drifts that do not impact classifier performance.

4.3. Active Learning strategy

Our framework accepts any of the Active Learning strategies from the group of stream-based selective sampling methods [45]. In our proposed strategy, we employed an uncertainty strategy [6,46] which determines if the label will be queried based on some uncertainty measures, e.g., a maximum posterior probability. The probability is compared to a threshold and if it is low enough the label will be queried as it is described in the following:

$$p(\hat{y} \mid X) = max(p(y \mid X)) \leq \theta,$$

where $p(\hat{y} \mid X)$ is the conditional class probability and θ is the threshold. The lower the θ values are, the less certain samples are going to be selected. Hence, lower values are preferred but too strict conditions will favor only instances that are close to the decision boundaries, making it blind to changes far from the decision boundary.

We also need to determine the threshold in order to select the most valuable instances. Many strategies have been published in the literature [46]. In our framework, we use one of the most reliable strategies which is the Variable Uncertainty strategy, introduced in Algorithm 1. For a given incoming instance X at time t and classifier L, we obtain the predicted label $\hat{y}$. We can obtain the posterior probabilities of the classifier L in relation to the predicted label $P_L(\hat{y} \mid X_t)$. The uncertainty metric is determined through the maximum a posteriori rule. The threshold θ that is used for decision making is updated at every step.

Algorithm 1: Variable Uncertainty active learning strategy.

Data: Instance X_t, trained classifier L, threshold step $s \in (0, 1]$
Result: Label the instance (true | false)
$\hat{y}_t \leftarrow argmax_t P_L(y \mid X_t)$;
if $P_L(\hat{y} \mid X_t) < \theta$ **then**
    $\theta \leftarrow \theta(1 - s)$;
    return true ;
else
    $\theta \leftarrow \theta(1 + s)$;
    return false ;

The main idea of this strategy is to lower the threshold value when the frequency of uncertainty instances is high, and to increase it when there is a stationary period and the model is certain about its decisions. This leads to a balanced budget spending and therefore uniformly distributed over time. This strategy exhibited reliable results according to other works [6,46], thus we employ it as our base active learning strategy. Next, we focus on proposing the best strategy to manage the budget in order to tackle drifting streams.

4.4. Dynamic Budget Active Learning (DBAL)

The proposed framework combines all three previous modules and it is described in Algorithm 2. Equation (1) illustrates the budget management function. For each incoming instance X, the actual budget spending $\hat{b}$ is checked. In the case of potentially infinite streams, it can be estimated as the ratio of labeled instances to all received instances. If the value does exceed a given budget b, initially set as the lower budget (LB) before any drift, and the active learning strategy (`QueryStrategy`) signals positively, the label is queried. Upon querying the label, the classifier is trained using the given instance and also the drift detectors are updated. After that, we assess whether a warning or a drift is detected. When a warning is detected, at time $t_{warning}$, we dynamically increase the budget to $\frac{UB}{2}$ and keep this value for a window of size w_w. After w_w queried instances, if no drift is detected, the budget returns to its original value. When a drift is detected, at time t_{drift}, the budget value is set to the upper budget limit (UB) for the subsequent w_d queried instances. This strategy allows us to temporarily expand the number of labeled instances and enable quick adaptation of the classifier to the new concept. This budget increase is restricted to the immediately following instances, and thus the number of labeling instances will eventually return close to the LB value over time. It should be noted that in a data stream containing multiple detected drifts in a short period, the number of labeled instances may differ slightly from those strategies utilizing fixed budget allocation. Our approach aims to provide the classifier with the flexibility to adapt, which implies that we may increase the budget for a certain period to allow the classifier to use it if necessary, but it won't consume it unnecessarily. This task and how these parameters affect the predictive performance and the number of labeled instances will be analyzed in the experimental evaluation.

$$B(t, t_{warning}, t_{drift}, w_w, w_d) = \begin{cases} UB, & \text{if } t_{drift} \leq t \leq t_{drift} + w_d \\ \frac{UB}{2}, & \text{if } t_{warning} \leq t \leq t_{warning} + w_w \\ LB, & \text{otherwise} \end{cases} \tag{1}$$

DBAL and PRopagate [31] both aim at addressing the challenge of limited label access in data streams under concept drift. However, there are significant differences between these two approaches. DBAL prioritizes providing the classifier more flexibility to adapt by increasing the budget for a short period of time and then returning to the original budget level. This approach ensures that the number of labeled instances is in between the lower LB and upper UB bounds by using only as much labeling as the classifier needs in that window of instances. In contrast, PRopagate dynamically changes the budget when a concept drift is detected, and then decreases it to a lower level after a time window in order to compensate for the increases. This fixed budget allocation lacks the flexibility that a classifier needs in a drifting stream. Hence, although both approaches involve adjusting the labeling budget, DBAL grants the classifier more autonomy when a drift is identified, and utilizes only the minimum amount of label queries during stable periods, while PRopagate simply reallocates the labeling budget but operates with a fixed value. Moreover, the methods of detecting concept drift in the two approaches also differ. While PRopagate was designed to tackle the delayed label problem, DBAL does not rely on delayed labeling and uses supervised concept drift detectors to identify drifts in the data stream immediately.

Algorithm 2: DBAL: Dynamic Budget Active Learning.

Data: Data Stream S, Lower budget LB, Upper budget UB,
Warning Window Size w_w, Drift Window Size w_d,
`QueryStrategy`, `WarningDetector`, `DriftDetector`
Classifier L

$\hat{b} \leftarrow 0$; $b \leftarrow LB$; $W_{idx} \leftarrow 0$; $D_{idx} \leftarrow 0$;
repeat
  $X \leftarrow S.nextInstance()$; ▷ Next instance arrives
  $\hat{y} \leftarrow L(X)$; ▷ Predict instance label
  if $\hat{b} \leq b$ ***and*** *QueryStrategy(X)* **then**
    Query the true label y of instance X;
    Update labeling expenses $\hat{b}$;
    Update `WarningDetector` using $(y, \hat{y})$;
    Update `DriftDetector` using $(y, \hat{y})$;
    Update classifier L with (X, y);
    if *WarningDetector.triggered* **then**
      $b \leftarrow \frac{UB}{2}$; ▷ Warning signal raised
      if $W_{idx} \leq w_w$ **then**
        $W_{idx}++$;
      else
        $b \leftarrow LB$;
        $W_{idx} \leftarrow 0$;
    if *DriftDetector.triggered* **then**
      $b \leftarrow UB$; ▷ Drift was detected
      if $D_{idx} \leq w_d$ **then**
        $D_{idx}++$;
      else
        $b \leftarrow LB$;
        $D_{idx} \leftarrow 0$;
until *S.end()*;

Fig. 2 provides an illustration of how the proposed method would work on a scenario in the presence of sudden concept drift. Prior to detecting the drift, the number of labeled instances is equal or below to the Lower Budget (LB) value. Once the drift is detected, the active learning strategy will have more flexibility to request additional labels in order to improve the classifier's certainty and adapt to the new concept for a period of w_d labeled instances. Consequently, the budget limit is temporarily elevated to the upper budget (UB) bound. Increasing the budget limit does not necessarily mean that the classifier will query the labels more often but it provides the flexibility and margin to do so if it estimates it is beneficial. Moreover, one also must acknowledge there is a delay between the time the concept drift happens and the time it is triggered by the drift detector, therefore it is important to detect the drift and adapt to it as quickly as possible. Furthermore, our active learning strategy is based on the confidence of the classifier, which means that the faster it adapts to the new concept, the fewer instances will need to be labeled. This creates a compensatory effect, as the budget allocation will increase for a short period of time to give the classifier some flexibility, but then returns to its original level.

In terms of memory and time costs, the Dynamic Budgeting for Active Learning (DBAL) approach does not impose any additional overhead on the base classifier or the active learning method. DBAL solely manages the budget by utilizing a concept drift detector, which operates on instances in constant time complexity ($O(1)$). Therefore, DBAL efficiently controls the budget without introducing any supplementary computational or memory expenses.

5. Experimental setup

The experimental study was designed to evaluate the performance of DBAL under different conditions and difficulties in comparison to previously published methods. The goal is to get a better understanding of which scenarios will the dynamic budget improve the performance of the classifiers and how this change will impact the number of queried instances. We will address the following research questions (RQ):

- **RQ1:** Which configuration of DBAL best performs when applied to different types of drifting streams?
- **RQ2:** How does allocating more budget after a concept drift affect the predictive performance and the number of label queries?
- **RQ3:** Can DBAL improve the predictive performance of classifiers with different learning mechanisms?
- **RQ4:** Can DBAL outperform state-of-the-art AL strategies under different types of concept drift and show robustness to drifting data?
- **RQ5:** Can DBAL outperform state-of-the-art AL strategies in real-world data streams?

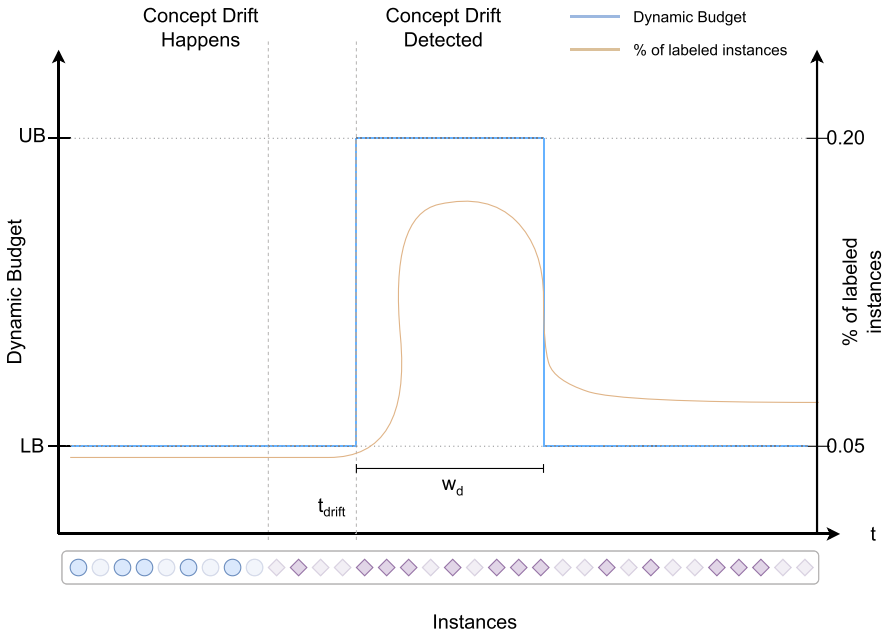


Fig. 2. Example of how DBAL manages a dynamic budget and instance labeling. The right axis represents the budget value using a blue line. The left axis represents the % of labeled instances using an orange line. Bottom: instances of the stream. Different shapes and colors represent different concepts and instances not labeled instances are transparent.

Table 1
Base classifiers used in the experiments.

Algorithm	Ensemble	Concept Drift	Reference
NB	✗	Implicit	[39]
HT	✗	Implicit	[48]
ARF	✓	Explicit	[41]
LB	✓	Explicit	[42]
SRP	✓	Explicit	[49]

5.1. Base classifiers

To evaluate the proposed methodology under a variety of scenarios we selected 5 base classifiers for data streams with different learning mechanisms. Those classifiers are presented in Table 1 with their detailed characteristics. All methods are implemented in MOA [47] and were run using the hyperparameter settings recommended by their authors.

5.2. Active learning strategies

We compared 6 active learning strategies combined with each one of the 5 base classifiers. These active learning strategies can be split into three groups: random selection, uncertainty-based, and sample-based. The first group is represented by the simplest approach `Random Selection` (AL-R), which draws a random number from a probabilistic distribution and based on the budget decides if the label will be queried or not. The second group is based on the classifier uncertainty and a threshold. The majority of methods in the literature are based on uncertainty, therefore we selected three methods: `Fixed Uncertainty` (AL-FU), `Variable Uncertainty` (AL-VU), and `Random Variable Uncertainty` (AL-RVU). The third group is represented by `Selective Sampling` (AL-SS). These methods and their specifications were presented and analyzed by *Gama* et al. [5]. We also compare our method with `PRopagate` (PRG) [31] which presents a dynamic budget approach based on two levels of fixed budgets.

5.3. DBAL experimental configuration

We evaluated the sensitivity of the parameters and their impact on the performance. Our investigation involved varying the lower budget values to assess the smallest number of instances we need to label to achieve the best performance, $LB \in \{1\%, 5\%, 10\%\}$. On the other hand, upper budget values varied to understand how much we need to increase the budget to achieve the best performance while maintaining the number of labeled instances feasible, $UB \in \{10\%, 20\%, 30\%, 40\%\}$. We varied both warning and drift windows size, $w_w \in \{25, 50, 100\}$ and $w_d \in \{50, 100, 200\}$. Considering the active learning strategies, we compared typical budget levels, $B \in \{1\%, 5\%, 10\%, 20\%, 100\%\}$.

Table 2
Specifications of the data stream generators.

Generator	Attributes	Classes	Concept drift
Agrawal	10	2	✓ Functions
Asset	5	2	✓ Functions
Hyperplane	12	2	✓ Features
Mixed	4	2	✓ Function
RandomRBF	10	2	✓ Centroids
RandomTree	10	2	✓ Tree
Sine	3	2	✓ Function

Table 3
Specifications of the real-world data streams.

Dataset	Instances	Features	Classes	Concept Drift
adult	45,222	14	2	✗
covtype1-2vsAll	267,001	54	2	✓
hepatitis	1,000,000	19	2	✓
spam	9,324	499	2	unknown
tripadvisor	18,569	30	2	✓
twitter	9,090	30	2	unknown
weather	18,159	8	2	✓

5.4. Generators

To evaluate the methodology in specific and controlled scenarios, we prepared stream generators under different drifting settings. We used 9 stream generators available in MOA [47]. Each generator was used to generate streams with 1, 2, and 3 gradual and sudden drifts. The concept drift happens in a predetermined position so that we can evaluate the behavior of the methods after changepoints. The generators are presented in Table 2, with their number of attributes, classes, and whether they can generate internal concept drifts. All generators are evaluated on a stream of $100,000$ instances.

5.5. Datasets

We evaluated 7 real-world datasets to assess the performance of the strategies on realistic benchmarks without predetermined conditions. These scenarios pose a significantly more challenging task to the classifiers and they allow us to understand how active learning strategies work under unique and challenging conditions. Table 3 presents the real-world datasets and their characteristics.

5.6. Evaluation

Algorithms were evaluated using the test-then-train model, in which each instance is first used to test and then to update the classifier (if the label is queried) in an online manner (instance by instance). The performance metrics include Accuracy and G-Mean, which are calculated over a sliding window of 500 instances. Accuracy is computed as in Eq. (2), measuring a ratio between the number of correctly classified instances and the total number of instances. This metric can give us an overview of how the classifier is performing. On the other hand, G-Mean is the combination of two metrics: sensitivity and specificity. G-Mean is the product of the two metrics as defined in Eq. (5) and works best when data classes are imbalanced. Accuracy and G-Mean are complementary metrics to have a holistic assessment of the performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

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

$$Specificity = \frac{TN}{TN + FP} \tag{4}$$

$$G - Mean = \sqrt{Sensitivity \times Specificity} \tag{5}$$

5.7. Reproducibility

The proposed method was implemented in Java using MOA [47] package and the source code of the algorithms and the experiments are publicly available on GitHub to facilitate the transparency and reproducibility of this research.[1] All algorithms use the parameter settings recommended by their authors. Detailed information about the specific parameters configuration is available on the GitHub repository. Algorithms were run on a cluster with 192 Intel Xeon cores, 6 TB RAM, and Centos 7.

[1] https://github.com/gabrieljaguiar/DBAL.

6. Results

This section presents and discusses the experimental results. First, we analyze the DBAL parameters and how they affect the classifier's performance. Second, we compare DBAL with state-of-the-art active learning strategies on a variety of synthetic stream generators with concept drift. Third, we compare the methodologies on real-world datasets.

6.1. Hyperparameter configuration

Firstly, we need to evaluate how DBAL hyperparameters selection affects its performance when dealing with drifting streams. Figs. 3 and 4 present the average accuracy of all base classifiers under different configurations of drift window size and upper budget values and for sudden and gradual drift, respectively. We can observe that increasing the drifting window size will not necessarily increase predictive performance. This shows that the base classifiers are able to adapt to the new concept and display similar performance overall. Looking at the upper budget values, we can see that higher values presented similar performance. It can be explained by which AL method we selected. AL methods that rely on uncertainty will refrain from requesting a label, even when they have an adequate budget, if the classifier's uncertainty level remains sufficiently high. Consequently, increasing the budget to 40% did not improve the performance when compared to 10% because the budget was not completely used within the period of upper

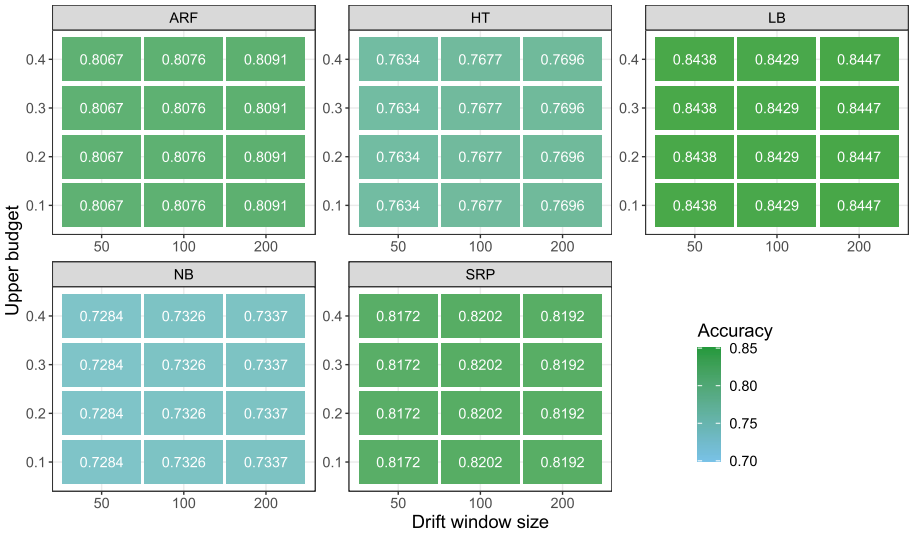


Fig. 3. Average accuracy for all classifiers for different combinations of Upper budget and drift window size for streams with sudden concept drift.

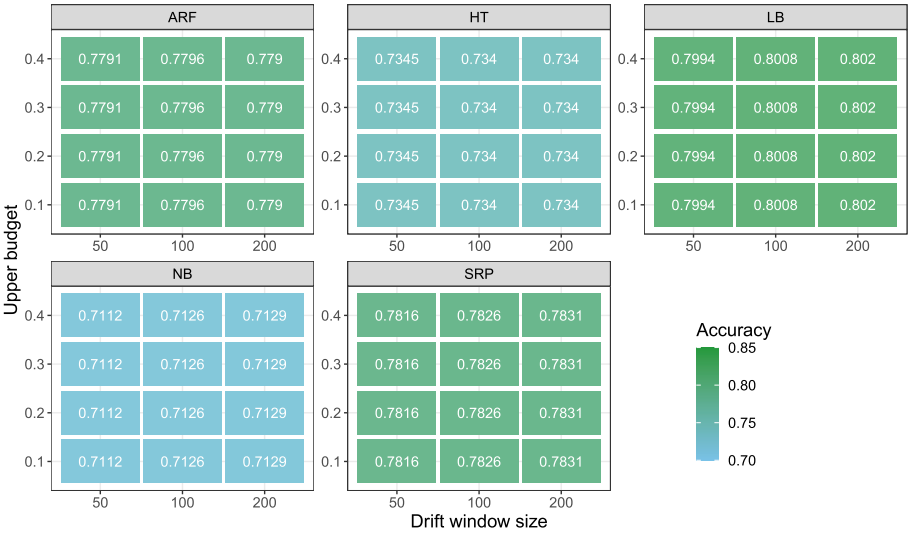


Fig. 4. Average accuracy for all classifiers for different combinations of Upper budget and drift window size for streams with gradual concept drift.

Table 4
Average percentage of labeled instances for each classifier and different drift window size values in the presence of sudden concept drift.

Classifier	Drift window size	% of labeled instances		
		$LB = 1\%$	$LB = 5\%$	$LB = 10\%$
ARF	50	1.08%	8.05%	18.18%
	100	1.08%	7.75%	18.18%
	200	1.16%	8.29%	18.18%
HT	50	1.30%	9.08%	19.80%
	100	1.38%	9.40%	19.80%
	200	1.39%	9.53%	19.80%
LB	50	1.30%	9.01%	19.02%
	100	1.34%	9.05%	19.02%
	200	1.34%	9.27%	19.02%
NB	50	1.30%	8.50%	19.28%
	100	1.38%	9.13%	19.28%
	200	1.39%	9.47%	19.28%
SRP	50	1.31%	7.63%	17.44%
	100	1.35%	7.66%	17.44%
	200	1.30%	7.61%	17.44%

Table 5
Average percentage of labeled instances for each classifier and different drift window size values in the presence of gradual concept drift.

Classifier	Drift window size	% of labeled instances		
		$LB = 1\%$	$LB = 5\%$	$LB = 10\%$
ARF	50	1.03%	7.03%	16.68%
	100	1.07%	7.04%	16.68%
	200	1.00%	7.05%	16.68%
HT	50	1.28%	8.58%	18.45%
	100	1.29%	8.88%	18.45%
	200	1.29%	8.75%	18.45%
LB	50	1.12%	7.17%	17.33%
	100	1.16%	7.30%	17.33%
	200	1.23%	7.18%	17.33%
NB	50	1.28%	8.50%	19.37%
	100	1.29%	8.72%	19.37%
	200	1.29%	8.60%	19.37%
SRP	50	1.17%	7.28%	16.75%
	100	1.18%	7.27%	16.75%
	200	1.13%	7.27%	16.75%

budget. This reflects an advantage and the flexibility of our method, offering the classifier the option to request a label only if needed. Overall, we observe that the DBAL method exhibits low sensitivity to the configuration of its hyperparameters. This is evident from the minimal variance in average accuracy across different configurations within each classifier. The observed differences in accuracy primarily arise from variations in the classifiers themselves, as expected due to their distinct learning mechanisms and predictive capabilities.

In terms of performance differences between different types of drift, it is evident that the classifiers exhibit superior performance in the presence of sudden concept drift compared to gradual concept drift. This outcome is expected since our method monitors performance, and the drop in performance is more immediate in the case of sudden drift, enabling faster adaptation. Moreover, when considering various window sizes and upper budget values, no specific configuration demonstrated superior performance for a particular type of concept drift, illustrating the robustness of our method across different drift types.

Another aspect to be investigated, is the influence of window sizes on the number of labeled instances. Tables 4 and 5 present the average number of labeled instances for each classifier under the experimented window sizes. It is possible to see that values are similar, and as expected, as the window size increases the number of queries slightly increases as well.

Furthermore, it is not only accuracy that matters, but also the speed at which the classifier can adapt to and recover from concept drift. Figs. 5 and 6 show the accuracy of each classifier over the streams with different numbers of drifts. Since the drifting point of every generator is the same we can average the accuracy, it is possible to see that on streams with 2 and 3 drifts, window sizes over 100 presented a quicker response to drift, showing that even though the overall accuracy is similar, increasing the window size can lead to quicker responses to the drift. Additionally, it is important to note the contrasting behavior of accuracy in different types of drifts. While we observe sharp drops in accuracy in the presence of sudden concept drift, we notice a smooth decrease in accuracy when confronted with gradual concept drift. However, as we previously discussed in relation to the UB parameter, we find that similar configurations of the drift window size yield comparable performances in terms of drift recovery for both types of concept drift.

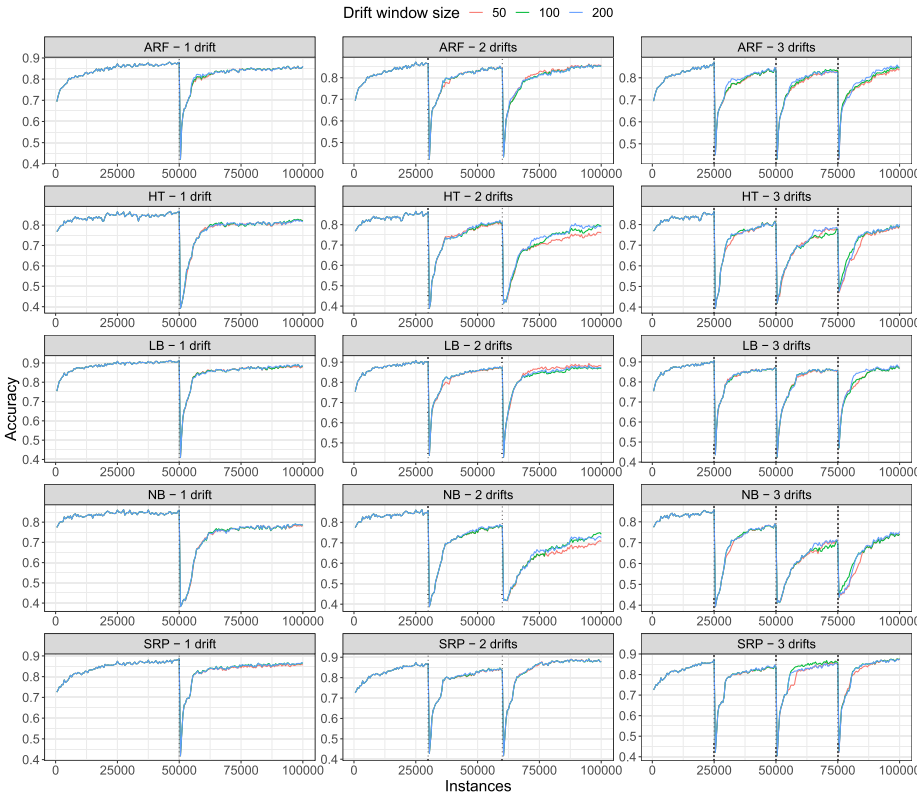


Fig. 5. Average accuracy for all classifiers under different numbers of sudden concept drifts. Each colored line represents different values of drift window size and dashed gray lines show the center point of the concept drift.

In addition to the drift window size and the upper budget value, another crucial parameter is the lower budget (LB) value. The primary objective is to set a value that is sufficiently high to detect concept drifts, yet low enough to maintain a reasonable number of queries while still achieving strong predictive performance. Figs. 7 and 8 show the relative label acquisitions value for all stream configurations we experimented. As expected, when the lower budget value is 10% the relative number of queries is higher than the other two values. Moreover, together with 5% it is possible to see by the peaks (when the upper budget takes place) that they were able to provide enough information about the stream in order to detect the concept drifts, and do not trigger many false alarms. On the other hand, when we applied 1% as our base budget value, we can see as expected that there is a gap between the moment that the drifts occur to the detection, or there is no detection as we can see in ARF row. Additionally, there are many false alarms that triggered warnings in stable periods of the stream.

When analyzing different values of LB, we can observe their impact on the delay in detecting concept drift. Reducing detection delay is a crucial and challenging task, especially when dealing with sparsely labeled data streams. In the scenario of sudden drifts, we observe a significant delay in detecting drifts when LB is below 5%. On the other hand, in the presence of gradual concept drift, the concept drift detector is triggered before the center point of the drift, as expected, but only when LB is above 5%. To summarize, utilizing LB values below 5% increases the delay in detecting both types of drift, while using 5% enables the detection of gradual drifts before the drift center point."

Tables 6 and 7 present the accuracy and G-Mean values for each evaluated value of lower budget. As was expected, since with $LB = 1\%$ drifts were not detected properly, and the number of labeled instances is significantly lower (at least 7 times lower) than other budget values. When we increase the lower budget value to 5% we can see a great improvement in predictive performance for all classifiers, but also we can see how this parameter impacts the number of queries, which increased significantly for LB and NB. This happens because a higher budget enables the detection of concept drifts more effectively, leading to an increase in budget and, consequently, a higher number of queries. The highest value of lower budget we evaluated was 10% and we could see that the number of labeled instances more than doubled relative to 5%. This shows that using lower budget values higher than that will result in a considerable amount of label queries which is not interesting in the studied scenario. Moreover, we can see that the predictive performance did not increase proportionally to the number of queried labels, but as expected for almost all classifiers it displayed better results. In summary, we can see that LB values have to be adjusted according to the amount of budget the user intends to spend. Lower values display reasonable results but may have difficulties to detect drifts, while higher values display good results but it also increases the number of labeled instances, often unnecessarily.

We showed that increasing or decreasing the drift window size did not significantly affect the overall predictive performance or the number of label queries, but it affected the drift recovery speed with windows bigger than 100 instances displaying better results.

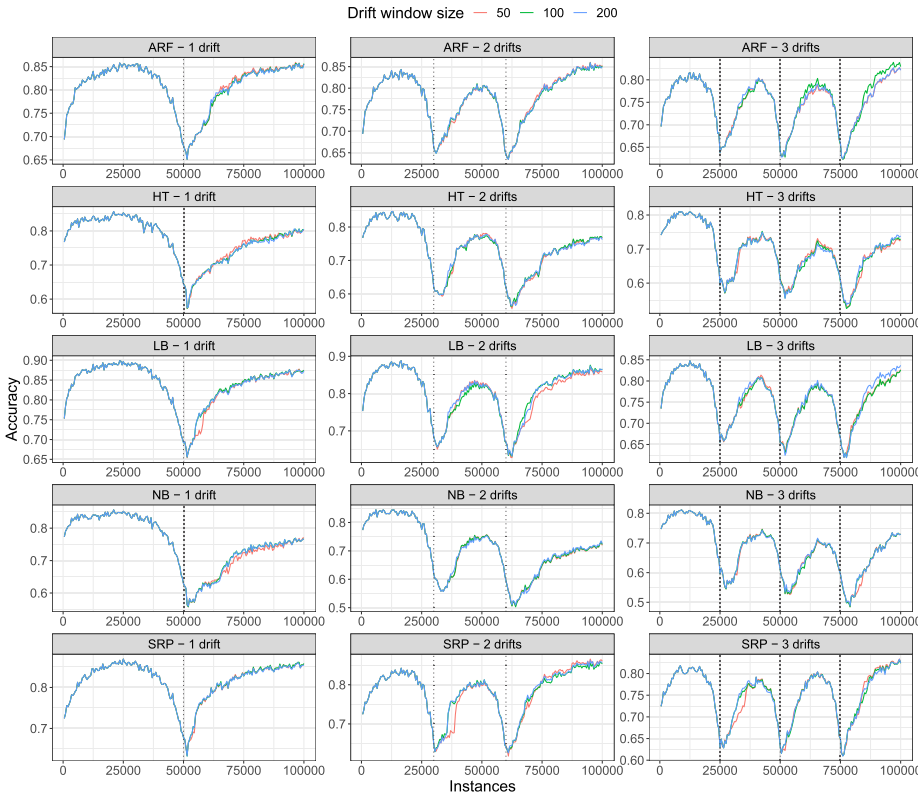


Fig. 6. Average accuracy for all classifiers under different numbers of gradual concept drifts. Each colored line represents different values of drift window size and dashed gray lines show the center point of the concept drift.

Table 6
Metrics averages of all streams with presence of sudden concept drifts for all base classifiers with all evaluated active learning strategies applied under different Lower Budget (LB) values. Inside parenthesis is presented the % of increase in relation to previous LB value.

Classifier	LB	Accuracy	G-Mean	Average % of labeled instances
ARF	1%	0.7244	0.7152	1.10%
	5%	0.8329 (↑ 14.97%)	0.8283 (↑ 15.82%)	8.03% (7.28×)
	10%	0.8661 (↑ 03.99%)	0.8635 (↑ 04.25%)	18.18% (2.26×)
HT	1%	0.6892	0.6721	1.36%
	5%	0.7942 (↑ 15.23%)	0.7824 (↑ 16.41%)	9.34% (6.88×)
	10%	0.8173 (↑ 02.91%)	0.8113 (↑ 03.69%)	19.80% (2.12×)
LB	1%	0.7712	0.7598	1.33%
	5%	0.8682 (↑ 12.57%)	0.8645 (↑ 13.76%)	9.11% (6.84×)
	10%	0.8920 (↑ 02.74%)	0.8894 (↑ 02.88%)	19.02% (2.08×)
NB	1%	0.6908	0.6799	1.36%
	5%	0.7557 (↑ 09.39%)	0.7459 (↑ 09.70%)	9.03% (6.66×)
	10%	0.7483 (↓ 01.00%)	0.7390 (↓ 01.00%)	19.28% (2.13×)
SRP	1%	0.7418	0.7336	1.32%
	5%	0.8368 (↑ 12.81%)	0.8303 (↑ 13.19%)	7.63% (5.77×)
	10%	0.8780 (↑ 04.92%)	0.8738 (↑ 05.24%)	17.44% (2.28×)

Also, by evaluating different values of upper budget we showed that this parameter did not affect predictive performance and the number of labeled instances due to the base active learning method that was applied. On the other hand, the hyperparameter that affected the performance the most was the lower budget value (LB). It affected directly the predictive performance and the number of label queries in view of the fact that a higher LB facilitates drift detection, therefore increasing performance but also the number of labeled instances.

Finally, Fig. 9 shows a real example of how the budget management works when compared to the theoretical behavior shown in Fig. 2. To create this figure, we selected HT as the classifier and the Agrawal generator. The stream contains one sudden drift that happened in instance 50,000. As we can see, there is a smaller peak before the concept drift detection, which is the warning detector

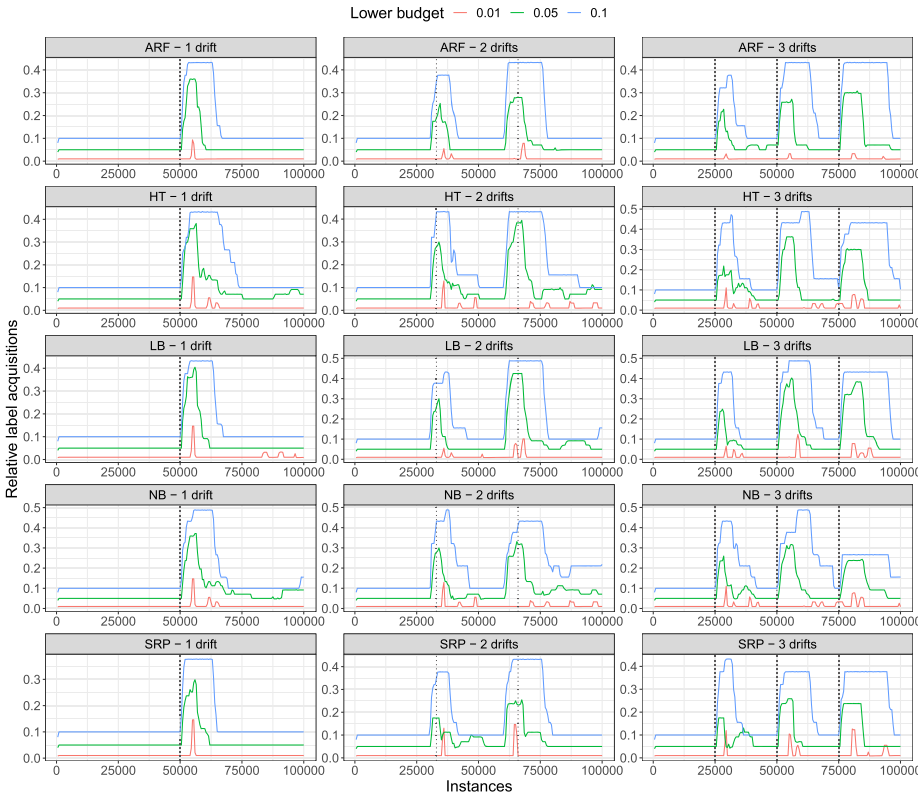


Fig. 7. Average relative label acquisitions by evaluation window for all classifiers under different numbers of sudden concept drifts. Each colored line represents different values of lower budget and dashed gray lines show where concept drift started.

Table 7
Metrics averages of all streams with presence of gradual concept drifts for all base classifiers with all evaluated active learning strategies applied under different Lower Budget (LB) values. Inside parenthesis is presented the % of increase in relation to previous LB value.

Classifier	LB	Accuracy	G-Mean	Average % of labeled instances
ARF	1%	0.7064	0.6947	1.03%
	5%	0.8039 (↑ 13.79%)	0.8002 (↑ 15.18%)	7.04% (6.82×)
	10%	0.8274 (↑ 02.92%)	0.8247 (↑ 03.05%)	16.68% (2.36×)
HT	1%	0.6741	0.6591	1.28%
	5%	0.7578 (↑ 12.42%)	0.7472 (↑ 13.36%)	8.73% (6.79×)
	10%	0.7706 (↑ 01.68%)	0.7616 (↑ 01.92%)	18.45% (2.11×)
LB	1%	0.7465	0.7383	1.17%
	5%	0.8175 (↑ 09.50%)	0.8138 (↑ 11.02%)	7.22% (6.17×)
	10%	0.8381 (↑ 02.52%)	0.8353 (↑ 02.64%)	17.33% (2.40×)
NB	1%	0.6753	0.6653	1.28%
	5%	0.7327 (↑ 08.49%)	0.7236 (↑ 08.75%)	8.60% (6.69×)
	10%	0.7287 (↓ 0.06%)	0.7209 (↓ 0.04%)	19.37% (2.25×)
SRP	1%	0.7196	0.7105	1.16%
	5%	0.8016 (↑ 11.39%)	0.7974 (↑ 12.23%)	7.27% (6.26×)
	10%	0.8261 (↑ 03.05%)	0.8234 (↑ 03.25%)	16.75% (2.30×)

triggered and leading to a quicker response to concept drift. By looking at the number of labeled instances, it is possible to say that it worked as designed since it increased the budget during the drifting window leading to a quick adaptation.

6.2. Comparison of active learning strategies on data stream generators

To evaluate how the proposed dynamic budget methodology improves the classification under limited access to labels, we compared it against other active learning strategies. According to the parameter sensitivity analysis presented before, we will use DBAL with $w_d = 100$ and $UB = 0.2$.

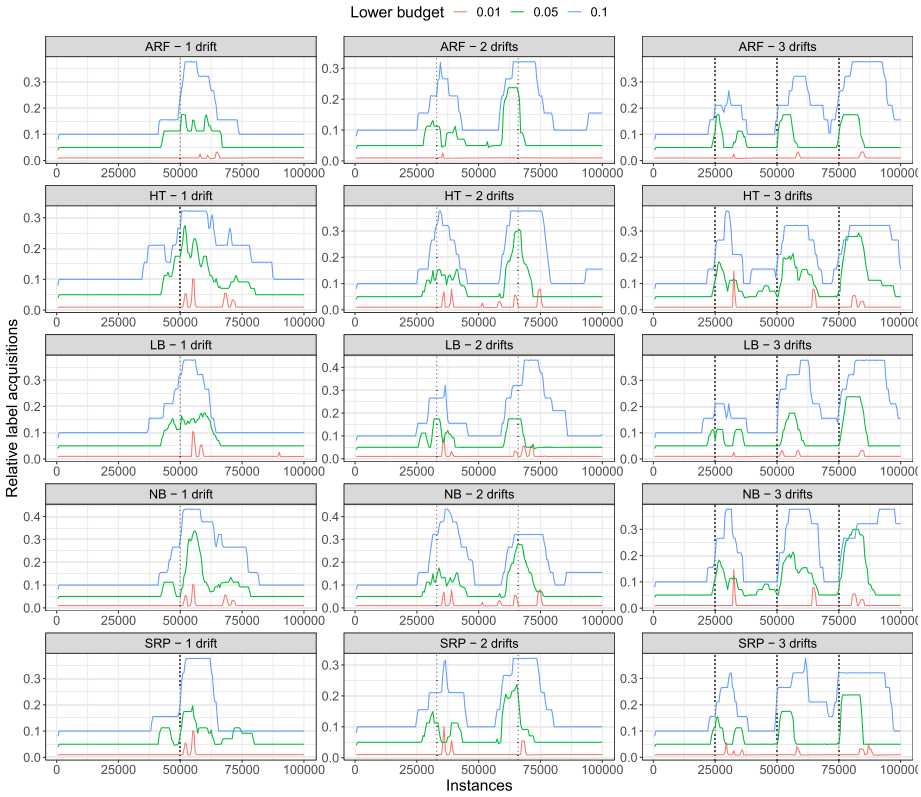


Fig. 8. Average relative label acquisitions by evaluation window for all classifiers under different numbers of gradual concept drifts. Each colored line represents different values of lower budget and dashed gray lines show the center point of the concept drift.

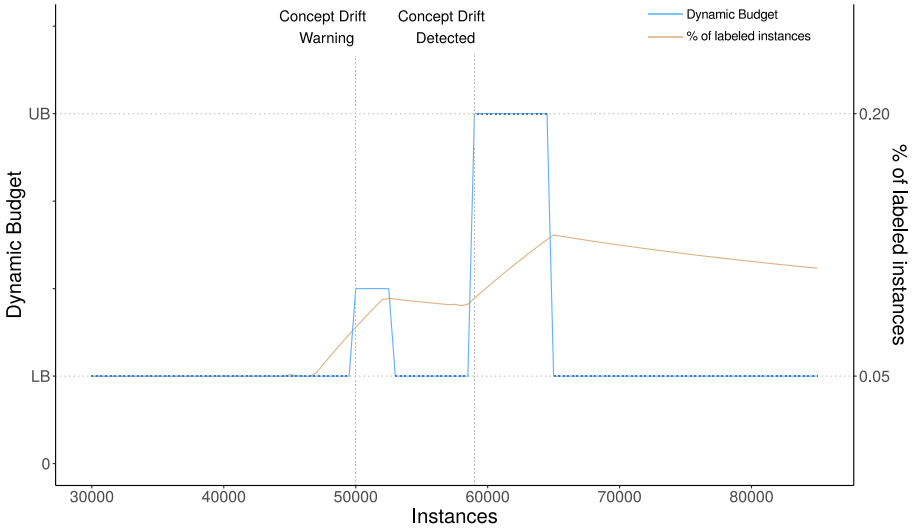


Fig. 9. Example of how DBAL manages the budget and labeled instances. Results for the HT classifier and Agrawal generator, with the presence of a sudden concept drift at the $50,000^{th}$ instance. The left axis represents the budget value using a blue line while the right axis represents the % of labeled instances using an orange line.

Tables 8 and 9 present the results for all budget values and drifting streams for HT as base classifier. Under a very restrictive budget such as 1%, we can see that DBAL can perform better than other active learning strategies for streams, with 1, 2, and 3 drifts. Even though it has a slight difference on the number of labeled instances, the predictive performance difference is close to 7% improvement in presence of sudden concept drift. However, as discussed previously, detecting drifts with 1% of labeled instances is a very difficult task, leading to worse performance on all strategies. By increasing the labeling budget to 10% we can see an increase in the performance of all classifiers, with DBAL exhibiting the best one. When specifically looking at data streams with 1

Table 8
Metrics averages of all streams with presence of sudden concept drifts for HT with all evaluated active learning strategies applied under different labeling budget values. The higher values for the given budget are highlighted.

Budget	# of drifts / AL strategy	1 drift			2 drifts			3 drifts		
		Accuracy	G-Mean	Labeled instances	Accuracy	G-Mean	Labeled instances	Accuracy	G-Mean	Labeled instances
1%	DBAL ($LB = 1\%$)	**0.7518**	**0.7388**	1.25%	**0.6659**	**0.6489**	1.42%	**0.6542**	**0.6335**	1.48%
	PRG	0.6962	0.6722	1.00%	0.6588	0.6408	1.00%	0.6162	0.5719	1.00%
	AL-VU	0.6852	0.6221	1.00%	0.6446	0.5969	1.00%	0.6117	0.5619	1.00%
	AL-RVU	0.6469	0.6225	1.00%	0.6331	0.6107	1.00%	0.5877	0.5561	1.00%
	AL-SS	0.6925	0.6731	1.00%	0.6550	0.6191	1.00%	0.6266	0.5832	1.00%
	AL-R	0.6578	0.6415	1.00%	0.6319	0.6124	1.00%	0.5951	0.5743	1.00%
	AL-FU	0.4900	0.0015	1.00%	0.4929	0.0015	1.00%	0.4883	0.0015	1.00%
10%	DBAL ($LB = 5\%$)	**0.8281**	**0.8206**	8.32%	**0.7818**	**0.7651**	9.58%	**0.7754**	**0.7635**	10.28%
	PRG	0.7850	0.7747	10.00%	0.7506	0.7372	10.00%	0.6752	0.6671	10.00%
	AL-VU	0.7814	0.7670	10.00%	0.7620	0.7484	10.00%	0.7050	0.6938	10.00%
	AL-RVU	0.7517	0.7450	10.00%	0.7166	0.7072	10.00%	0.6957	0.6847	10.00%
	AL-SS	0.8033	0.7978	10.00%	0.7579	0.7507	10.00%	0.7159	0.7061	10.00%
	AL-R	0.7488	0.7424	10.00%	0.7158	0.7091	10.00%	0.6836	0.6765	10.00%
	AL-FU	0.4896	0.0006	10.00%	0.4961	0.0006	10.00%	0.4883	0.0006	10.00%
20%	DBAL ($LB = 10\%$)	**0.8398**	**0.8346**	15.79%	**0.8129**	**0.8072**	19.23%	**0.7992**	**0.7921**	24.37%
	PRG	0.7928	0.7863	20.00%	0.7720	0.7591	20.00%	0.7101	0.7002	20.00%
	AL-VU	0.7741	0.7664	20.00%	0.7516	0.7364	20.00%	0.7224	0.7141	20.00%
	AL-RVU	0.8013	0.7950	20.00%	0.7507	0.7440	20.00%	0.7238	0.7146	20.00%
	AL-SS	0.7787	0.7749	20.00%	0.7529	0.7479	20.00%	0.7386	0.7323	20.00%
	AL-R	0.7743	0.7696	20.00%	0.7491	0.7444	20.00%	0.7166	0.7108	20.00%
	AL-FU	0.5403	0.2067	20.00%	0.5262	0.1551	20.00%	0.5288	0.1447	20.00%

Table 9
Metrics averages of all streams with presence of gradual concept drifts for HT with all evaluated active learning strategies applied under different labeling budget values. The higher values for the given budget are highlighted.

Budget	# of drifts / AL strategy	1 drift			2 drifts			3 drifts		
		Accuracy	G-Mean	Labeled instances	Accuracy	G-Mean	Labeled instances	Accuracy	G-Mean	Labeled instances
1%	DBAL ($LB = 1\%$)	**0.7165**	**0.7003**	1.26%	**0.6671**	**0.6521**	1.30%	**0.6382**	**0.6250**	1.30%
	PRG	0.6889	0.6587	1.00%	0.6528	0.6263	1.00%	0.6162	0.5719	1.00%
	AL-VU	0.6796	0.6605	1.00%	0.6539	0.6307	1.00%	0.6039	0.5483	1.00%
	AL-RVU	0.6502	0.6399	1.00%	0.6402	0.6316	1.00%	0.5956	0.5754	1.00%
	AL-SS	0.6899	0.6713	1.00%	0.6477	0.5853	1.00%	0.6214	0.5822	1.00%
	AL-R	0.6282	0.6082	1.00%	0.6190	0.5976	1.00%	0.5714	0.5482	1.00%
	AL-FU	0.4900	0.0015	1.00%	0.4929	0.0015	1.00%	0.4883	0.0015	1.00%
10%	DBAL ($LB = 5\%$)	**0.8024**	**0.7951**	8.00%	**0.7576**	**0.7463**	7.96%	**0.7123**	**0.6993**	10.67%
	PRG	0.7879	0.7805	10.00%	0.7465	0.7406	10.00%	0.6752	0.6671	10.00%
	AL-VU	0.7717	0.7583	10.00%	0.7346	0.7189	10.00%	0.6782	0.6686	10.00%
	AL-RVU	0.7507	0.7442	10.00%	0.7160	0.7097	10.00%	0.6677	0.6593	10.00%
	AL-SS	0.7876	0.7792	10.00%	0.7398	0.7327	10.00%	0.6863	0.6774	10.00%
	AL-R	0.7285	0.7205	10.00%	0.7049	0.6967	10.00%	0.6753	0.6651	10.00%
	AL-FU	0.4896	0.0007	10.00%	0.4961	0.0007	10.00%	0.4884	0.0006	10.00%
20%	DBAL ($LB = 10\%$)	**0.8027**	**0.7958**	16.60%	**0.7831**	**0.7747**	17.84%	**0.7259**	**0.7143**	20.91%
	PRG	0.7667	0.7598	20.00%	0.7545	0.7477	20.00%	0.7101	0.7002	20.00%
	AL-VU	0.7741	0.7672	20.00%	0.7507	0.7442	20.00%	0.7021	0.6941	20.00%
	AL-RVU	0.7839	0.7779	20.00%	0.7412	0.7357	20.00%	0.6865	0.6795	20.00%
	AL-SS	0.7655	0.7599	20.00%	0.7481	0.7429	20.00%	0.6987	0.6930	20.00%
	AL-R	0.7562	0.7500	20.00%	0.7384	0.7322	20.00%	0.7036	0.6958	20.00%
	AL-FU	0.5391	0.1559	20.00%	0.5304	0.1876	20.00%	0.5278	0.1914	20.00%

and 2 drifts we can see that DBAL performs better than its peers with a smaller number of labeled instances, demonstrating that querying labels in the correct timing is better than querying labels uniformly over the stream. Finally, when considering a more generous budget of 20%, DBAL achieves the best performance, and again for streams with 1 and 2 drifts, it displayed better metrics with fewer labeled instances. Moreover, one may see that for 3 sudden drifts the number of labeled instances is higher, this can be explained by the Low Budget value. With $LB = 10\%$ it is easier to trigger more warnings, therefore increasing the budget for a smaller period of time. In summary, when using DBAL combined with HT, we achieved a better performance than traditional active learning strategies for both types of drift, slightly better for sudden drift to the nature of the drift detector, and also with reasonable budget usage.

Table 10
Metrics averages of all streams with presence of sudden concept drifts for all base classifiers with all evaluated active learning strategies applied under different labeling budget values. The higher values for the given budget are highlighted.

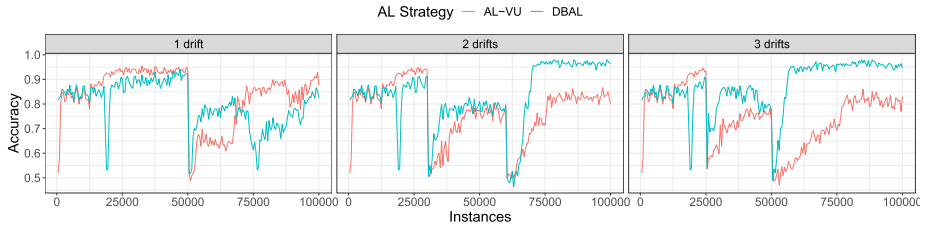
Budget	Classifier / AL strategy	ARF		LB		NB		SRP	
		Accuracy	G-Mean	Accuracy	G-Mean	Accuracy	G-Mean	Accuracy	G-Mean
1%	DBAL ($LB = 1\%$)	**0.7226**	**0.7137**	**0.7705**	**0.7593**	**0.6922**	**0.6816**	**0.7427**	**0.7346**
	PRG	0.7077	0.6935	0.7453	0.7310	0.6584	0.6372	0.7259	0.7103
	AL-VU	0.6897	0.6679	0.7426	0.7234	0.6470	0.6262	0.7240	0.6993
	AL-RVU	0.6930	0.6721	0.7398	0.7232	0.6261	0.6092	0.7182	0.6977
	AL-SS	0.7057	0.6940	0.7382	0.7255	0.6595	0.6404	0.7424	0.7301
	AL-R	0.6896	0.6759	0.7244	0.7104	0.6224	0.6070	0.6830	0.6676
	AL-FU	0.5259	0.0945	0.5781	0.2634	0.5532	0.2572	0.4905	0.0017
10%	DBAL ($LB = 5\%$)	0.8341	0.8298	**0.8662**	**0.8624**	**0.7572**	**0.7479**	0.8399	0.8333
	PRG	0.8449	0.8390	0.8621	0.8581	0.6702	0.6471	0.8475	0.8397
	AL-VU	0.8410	0.8308	0.8508	0.8454	0.6715	0.6457	0.8522	0.8440
	AL-RVU	0.8311	0.8164	0.8460	0.8366	0.6424	0.6298	0.8389	0.8302
	AL-SS	**0.8415**	**0.8333**	0.8564	0.8504	0.6613	0.6482	**0.8546**	**0.8494**
	AL-R	0.8238	0.8197	0.8486	0.8433	0.6394	0.6304	0.8212	0.8164
	AL-FU	0.5719	0.1983	0.6934	0.4641	0.5253	0.1462	0.4914	0.0007
20%	DBAL ($LB = 10\%$)	0.8661	0.8635	**0.8920**	**0.8894**	**0.7483**	**0.7390**	**0.8780**	**0.8738**
	PRG	0.8708	0.8676	0.8846	0.8820	0.6799	0.6662	0.8777	0.8743
	AL-VU	0.8696	0.8656	0.8760	0.8727	0.6809	0.6652	0.8681	0.8645
	AL-RVU	0.8605	0.8576	0.8791	0.8750	0.6505	0.6421	0.8660	0.8639
	AL-SS	**0.8667**	**0.8643**	0.8816	0.8783	0.6590	0.6489	0.8761	0.8738
	AL-R	0.8545	0.8510	0.8723	0.8675	0.6396	0.6310	0.8591	0.8563
	AL-FU	0.6097	0.3108	0.6286	0.2610	0.5715	0.3084	0.7050	0.4709

Table 11
Metrics averages of all streams with presence of gradual concept drifts for all base classifiers with all evaluated active learning strategies applied under different labeling budget values. The higher values for the given budget are highlighted.

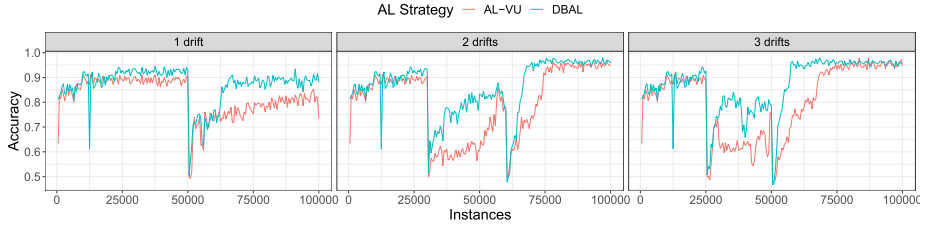
Budget	Classifier / AL strategy	ARF		LB		NB		SRP	
		Accuracy	G-Mean	Accuracy	G-Mean	Accuracy	G-Mean	Accuracy	G-Mean
1%	DBAL ($LB = 1\%$)	**0.7079**	**0.6965**	**0.7457**	**0.7366**	**0.6752**	**0.6653**	0.7120	**0.7209**
	PRG	0.6912	0.6775	0.7308	0.7175	0.6610	0.6449	**0.7164**	0.7019
	AL-VU	0.6887	0.6737	0.7287	0.7177	0.6541	0.6394	0.6833	0.7017
	AL-RVU	0.6895	0.6788	0.7160	0.7045	0.6277	0.6165	0.6827	0.6937
	AL-SS	0.6938	0.6786	0.7230	0.7125	0.6577	0.6378	0.7011	0.7142
	AL-R	0.6618	0.6449	0.6924	0.6732	0.6029	0.5855	0.6381	0.6558
	AL-FU	0.5231	0.0891	0.5900	0.2861	0.5414	0.2116	0.0017	0.4905
10%	DBAL ($LB = 5\%$)	**0.8274**	0.8000	0.8185	0.8147	**0.7339**	**0.7257**	0.7971	0.8010
	PRG	0.8142	0.8103	0.8239	0.8202	0.6748	0.6593	0.8136	0.8096
	AL-VU	0.8101	0.8052	**0.8225**	**0.8185**	0.6752	0.6622	**0.8138**	**0.8178**
	AL-RVU	0.8051	0.8010	0.8103	0.8051	0.6454	0.6349	0.8052	0.8092
	AL-SS	0.8126	**0.8087**	0.8196	0.8146	0.6614	0.6498	0.8104	0.8142
	AL-R	0.8010	0.7962	0.8190	0.8126	0.6315	0.6225	0.7978	0.8032
	AL-FU	0.5713	0.2002	0.6877	0.4530	0.5272	0.1396	0.0007	0.4914
20%	DBAL ($LB = 10\%$)	0.8274	0.8247	0.8381	0.8353	**0.7287**	**0.7209**	0.8234	0.8261
	PRG	0.8324	0.8287	0.8390	0.8363	0.6761	0.6654	0.8376	0.8350
	AL-VU	0.8301	0.8259	0.8397	0.8371	0.6771	0.6630	0.8298	0.8328
	AL-RVU	0.8254	0.8222	0.8350	0.8305	0.6486	0.6405	0.8230	0.8259
	AL-SS	0.8288	0.8258	0.8360	0.8319	0.6567	0.6471	0.8295	0.8320
	AL-R	**0.8336**	**0.8299**	**0.8485**	**0.8434**	0.6339	0.6254	**0.8390**	**0.8421**
	AL-FU	0.6040	0.2974	0.6316	0.2881	0.5681	0.2796	0.4510	0.6803

DBAL was also evaluated with 4 other base classifiers and those results are presented in Tables 10 and 11. Looking at both tables, we can see a difference between the results achieved by DBAL in the presence of sudden and gradual concept drifts. This difference is explained because sudden drifts are easier to detect with supervised drift detectors due to their instant change in the data distribution, therefore it activates the budget mechanisms at the appropriate moment.

Regarding the presence of sudden concept drift, DBAL outperformed all other active learning strategies for LB and NB for all evaluated budget values. For SRP, we can see that DBAL did not achieve the best results in a scenario with 10% of budget. As we observed in previous experiments, SRP was the classifier that on average had the lower number of labeled instances. With that in mind, it is possible to conclude that SRP shows higher certainty in certain scenarios which does not trigger querying labels. Therefore, the reason for DBAL's inferiority to its peers is more closely associated with the smaller amount of labeled instances available, rather than any deficiencies in DBAL's learning mechanism. When we look at the ARF results, we can see that `AL-SS` displayed a better performance. ARF utilizes internal drift detectors in order to manage the ensemble, therefore, it could perform better than DBAL in a scenario where the available budget was enough for the ARF internal mechanism to detect the concept drift and adapt to it.

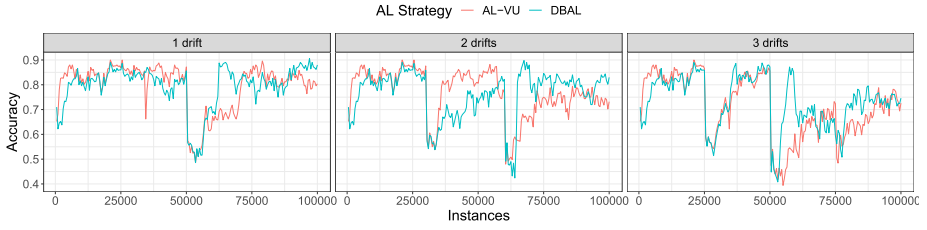


(a) 10% of labeling budget

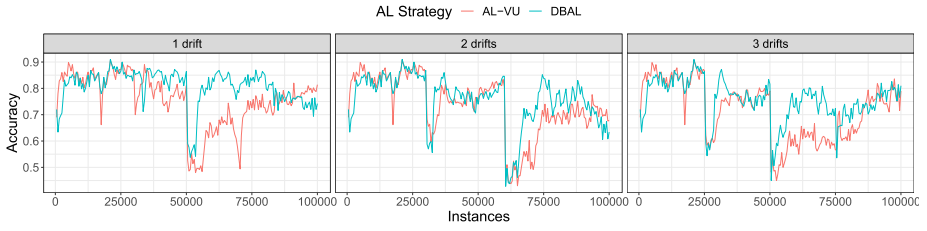


(b) 20% of labeling budget

Fig. 10. Accuracy comparison between AL-VU and DBAL with 10% and 20% of labeling budget for different number of drifts for *Agrawal* generator with HT as base classifier.



(a) 10% of labeling budget



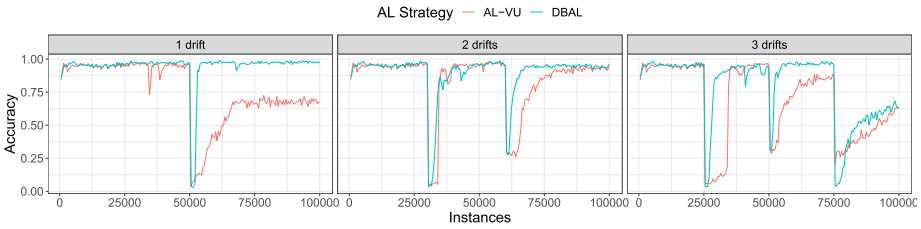
(b) 20% of labeling budget

Fig. 11. Accuracy comparison between AL-VU and DBAL with 10% and 20% of labeling budget for different number of drifts for *Hyperplane* generator with HT as base classifier.

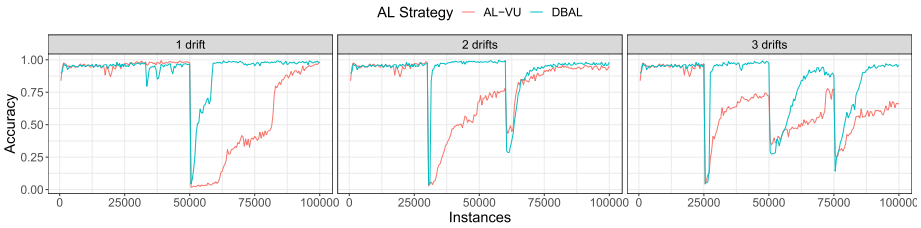
Considering the presence of gradual concept drift with NB as base classifier, DBAL presented the best performance overall. When looking at the results of ARF, LB and SRP it is possible to see that other active learning strategies could achieve better results. DBAL budget management is based on concept drift triggers, but gradual concept drift is harder to detect based on accuracy drops, therefore, the number of queries that DBAL requires on a stream with gradual concept drift is lower than with sudden drift, thus impacting its performance. It is worth mentioning that with less tight budgets, like 20% we can see that `AL-R` achieves the best results. This shows that, with bigger budgets, state-of-the-art data stream classifiers are able to deal with gradual concept drift in sparsely labeled data streams. On the other hand, when we look at very restricted budgets, which is more feasible in reality, DBAL is able to outperform all active learning strategies on streams with gradual concept drift.

Moreover, it is important to see how the accuracy changes over the data stream and how quickly recovers after a drift. For example, Figs. 10, 11, and 12 present the accuracy of DBAL and `AL-VU` with HT as base classifier for 10% and 20% of budget value for multiple generators. We can see that even if the overall accuracy values are similar, we can notice a gap in the accuracy when concept drift happens, demonstrating that DBAL can recover faster from drifts than other active learning strategies.

Furthermore, we conducted statistical tests to assess how DBAL can perform comparably or even outperform fixed-budget active learning strategies with a smaller number of labeled instances. We used the Friedman test, with a significance level of $\alpha = 0.05$. The null hypothesis is that DBAL is not better than other active learning strategies. Anytime the null hypothesis is rejected, the Nemenyi



(a) 10% of labeling budget



(b) 20% of labeling budget

Fig. 12. Accuracy comparison between AL-VU and DBAL with 10% and 20% of labeling budget for different number of drifts for *Sine* generator with HT as base classifier.

post hoc test can be applied, stating that the performance of the two approaches is significantly different if their corresponding average ranks differ by at least a Critical Difference (CD) value. When multiple algorithms are compared in this way, a graphical representation can be used to represent the results with the CD diagram, as proposed in [50].

The Nemenyi tests are presented in Figs. 13 and 14. DBAL was compared to the other evaluated active learning strategies. When looking at the diagrams we can see the great performance achieved by DBAL when combined with HT and NB. Regarding the HT, for a given budget of 10%, DBAL achieved a better ranking than all the other classifiers, and even though there are no significant differences to `AL-SS`, the number of labeled instances is lower on average than the traditional active learning strategy. With NB as base learner, for both budget values DBAL displayed a better performance than all the other strategies. Considering ARF and LB, although DBAL did not rank the best, its performance was not significantly different from the top-performing methods, even when using fewer labeled instances. Finally, using SRP as the base classifier, we can see that with 10% of budget DBAL could not achieve similar performance to what it did achieve with other base classifiers, but with 20% of budget there was no difference between DBAL and the highest ranked strategies. Upon analyzing the results categorized by budget levels, it is evident that DBAL stands out as one of the top-performing active learning strategies. It consistently outperformed its counterparts, particularly when considering a budget allocation of 1%, even with fewer labeled instances. To sum up, it can be observed that when allocating 20% of the budget to all base classifiers, DBAL attains comparable or superior performance to its peers. This is noteworthy considering that, as previously demonstrated, it employs an average of 4% to 6% fewer labeled instances. With a more limited budget, the same trend persists for three of the five assessed base classifiers.

6.3. Comparison of active learning strategies on real-world data streams

The previous experiment focused on the performance of active learning strategies in sparsely labeled drifting data streams from synthetic generators. This allowed us to scrutinize how the classifiers and strategies worked in specific and controlled situations. However, real-world datasets pose unique challenges because they are not generated in a controlled environment. It is important to highlight the differences between artificial and real-world data streams. The datasets present a combination of various learning challenges that appear with different levels of intensity or frequency. Their imbalance ratio fluctuates over time and concept drift can shift among different types at varying speeds. Artificial concept drift can be created in specific periods through changes in the generator probabilities. Real-world datasets are not subject to the clear probabilistic mechanisms that dictate the probabilities and distribution of instances in the data stream. Instead, the datasets are collected to reflect specific observations of a phenomenon. As a result, classifiers face distinct challenges, such as delays in the arrival of instances from a specific class or prolonged periods with only one class of instances. In real-world streaming datasets it is hard to determine where or if a concept drift happened without external information about the data. Therefore, since our method relies on concept drift detection, some real-world data streams will not trigger the dynamic module of DBAL and consequently, its performance will be similar to `AL-VU`.

Table 12 shows the results of the active learning strategies under different budgets for HT as base classifier. As mentioned before, we cannot control how many drifts a stream will have, and this affects directly the number of labeled instances. When running DBAL with $LB \in \{10\%, 5\%, 1\%\}$ the final number of labeled instances was superior to 10%, so in order to have a fair comparison, we compared with active learning strategies using $1\%, 5\%$ and 10% budget.

Looking at the results, it is apparent that DBAL performed remarkably well in comparison to other active learning strategies, particularly when using the most restricted budget of 1%. Out of seven real-world data streams, DBAL achieved the best or second-

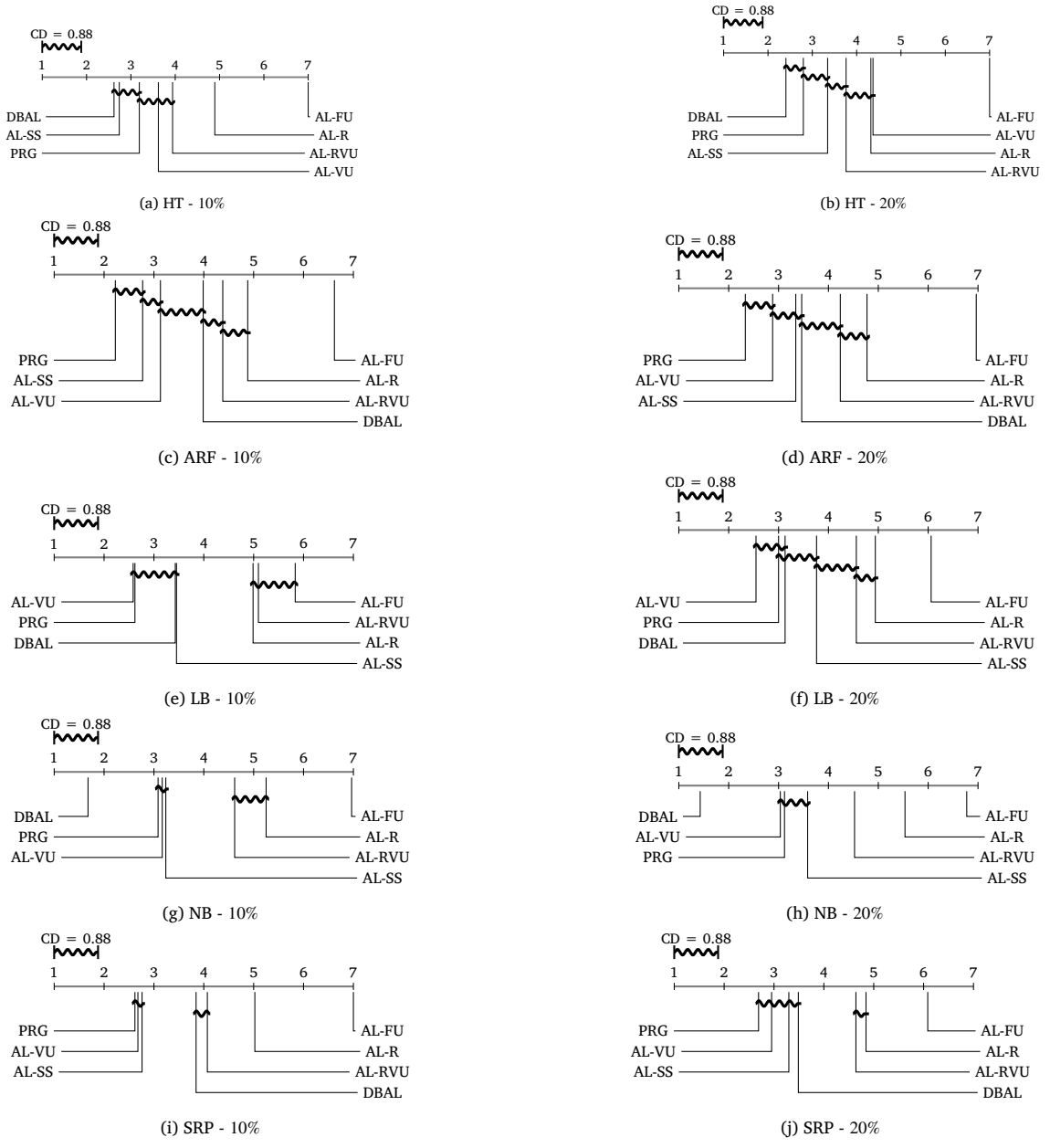


Fig. 13. Comparison of the average of metrics obtained by each active learning strategy given a base classifier according to the Nemenyi test. Groups of algorithms that are not significantly different ($\alpha = 0.05$ and $CD = 0.88$) are connected.

best G-Mean values in six datasets. However, since class imbalance is a significant challenge in real-world streams, relying on accuracy alone may lead to misleading conclusions. Nevertheless, for four of the data streams, DBAL also displayed one of the two best performances regarding accuracy. Moreover, when considering a budget value of 5%, DBAL achieved the best performance for five out of seven data streams, and was among the best and second best in all of them. This finding is consistent with previous experiments, where we observed that DBAL reaches its peak performance with a budget of 5% of the lower bound. In contrast, with a more relaxed budget of 10%, DBAL was among the best-performing strategies for four of the data streams. Although this result is still good, it is worth noting that almost all active learning strategies can handle the challenges posed by real-world data streams with a budget of 10%.

In summary, the results showed that DBAL outperformed the other active learning strategies for more restrictive budgets and obtained similar performance with more loose budgets. However, it is important to note that evaluating the proposed method's full capacity on real-world data streams can be challenging due to the absence of a clear point of concept drift. It is also worth noting that DBAL's performance on real-world data streams may differ from the synthetic generators used in the experiments. Further

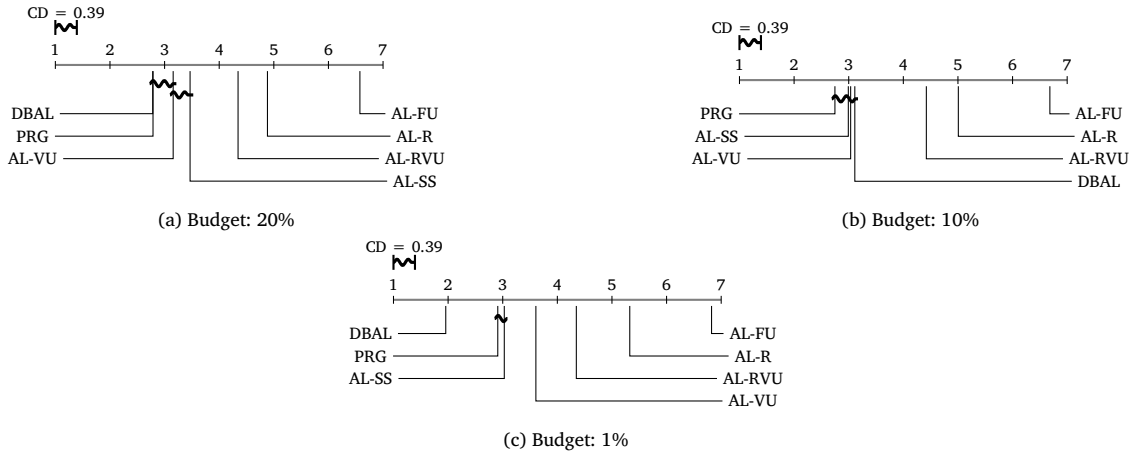


Fig. 14. Comparison of the average of metrics obtained by each active learning strategy categorized by labeling budget according to the Nemenyi test. Groups of algorithms that are not significantly different ($\alpha = 0.05$ and $CD = 0.39$) are connected.

investigations are necessary to fully explore the potential of DBAL and other dynamic budget active learning strategies in the context of real-world data streams.

6.4. Limitations

While our research has provided valuable insights into learning under limited access to class labels, it is important to acknowledge certain limitations. These limitations include the following aspects:

- **Supervised concept drift detector limitation**: Since we utilized a supervised concept drift detector that plays a crucial role in our method, it imposes restrictions on experimenting with very tight budget values that could potentially compromise the effectiveness of drift detection.
- **Multiple drifts and increasing labeled instances**: In scenarios where multiple drifts occur consecutively, there is a possibility of a rapid increase in the number of labeled instances required. This may limit the performance of our proposed approach.

It is worth mentioning that although these limitations exist, they present opportunities for future research to address and expand upon our work. By addressing these limitations, researchers can further enhance the understanding and applicability of dynamic budget allocation.

7. Conclusion and future work

In this paper, we have tackled the challenge of learning from evolving data streams with limited access to labeled data. The scenario of limited label access is common in real-world datasets, posing difficulties in adapting to concept drifts. We proposed a novel dynamic budget active learning strategy to provide the classifier with the flexibility needed to recover quickly from concept drifts. Our framework consists of monitoring the data stream with concept drift detectors and increasing the budget for a given time window, in which the base classifier will have more flexibility to query instance labels. When a concept drift is detected, we dynamically allocate more budget, during a specific time window, to ensure a prompt response from the classifier. Conversely, during stable periods, we reduce the budget allocation.

To assess the effectiveness of our dynamic budget active learning strategy, we conducted extensive experiments using five distinct base classifiers and various data streams featuring both gradual and sudden concept drifts, including real-world data streams. The results of our study demonstrate that our approach of dynamically adjusting the budget for a given time window can significantly improve the predictive performance and recovery of the classifier, outperforming traditional active learning strategies with similar budget allocations. Notably, even under tight budget constraints, such as a budget of 1%, our strategy still showed significant improvements in predictive capability for all of the base classifiers evaluated. Additionally, our study revealed that the timing of instance labeling is a crucial factor in the effectiveness of the active learning strategy, highlighting the importance of ensuring that labels are allocated appropriately in order to maximize the benefits of the proposed approach.

For future work, we plan to explore new methods to monitor concept drift using unsupervised drift detectors and investigate other features of data streams that could be used to trigger changes in budget allocation. For instance, we could consider using class imbalance as an indicator to allocate more budget, as imbalanced data can pose a challenge for classifiers. Similarly, we could use classifier error rates or noise levels as indicators to adjust the budget allocation, as these factors can impact the classifier's performance. We also plan to evaluate the proposed framework on more diverse and complex data streams to assess its generalization

Table 12
Metrics for all evaluated real-world data streams for all evaluated active learning strategies under different labeling budget values with HT as base classifier.

Labeling Budget		1%			5%			10%		
Dataset	AL	Accuracy	G-Mean	Labeled Instances	Accuracy	G-Mean	Labeled Instances	Accuracy	G-Mean	Labeled Instances
adult	DBAL	0.7519	0.0028	1.00%	0.8066	0.4910	4.99%	0.8126	0.6013	9.98%
	PRG	**0.8095**	0.5937	1.00%	0.8213	0.6136	5.00%	0.8175	0.6340	10.00%
	AL-VU	0.8068	0.5716	1.00%	0.8238	0.6541	5.00%	0.8038	0.6174	10.00%
	AL-RVU	0.8000	0.5261	1.00%	**0.8329**	**0.6999**	5.00%	0.8175	0.6511	10.00%
	AL-SS	0.8010	**0.6010**	1.00%	0.8129	0.5819	5.00%	**0.8211**	**0.6688**	10.00%
	AL-R	0.7966	0.5540	1.01%	0.8172	0.6617	4.98%	0.8145	0.6440	9.84%
	AL-FU	0.8092	0.5891	1.00%	0.7744	0.5344	0.02%	0.7944	0.5583	7.60%
covtype1-2vsAll	DBAL	0.8283	0.1775	1.64%	**0.9280**	**0.8061**	9.80%	0.9280	0.8061	9.80%
	PRG	0.8172	0.2087	1.00%	0.9096	0.6285	5.00%	0.9361	0.8605	10.00%
	AL-VU	0.8494	0.2930	1.00%	0.9131	0.6433	5.00%	0.9360	0.8607	10.00%
	AL-RVU	0.8424	0.2583	1.00%	0.9130	0.6048	5.00%	**0.9366**	**0.8554**	10.00%
	AL-SS	**0.8583**	**0.3823**	1.00%	0.9138	0.5529	5.00%	0.9337	0.8119	10.00%
	AL-R	0.8521	0.1183	0.99%	0.9124	0.6420	4.98%	0.9256	0.8326	9.97%
	AL-FU	0.8166	0.0000	0.04%	0.8166	0.0000	0.06%	0.8166	0.0000	0.13%
hepatitis	DBAL	**0.8825**	**0.8018**	1.00%	**0.8865**	**0.8006**	5.00%	0.8854	0.7769	9.99%
	PRG	0.8778	0.7911	1.00%	0.8771	0.7549	5.00%	0.8866	0.7719	10.00%
	AL-VU	0.8809	0.7501	1.00%	0.8782	0.7466	5.00%	**0.8870**	0.7679	10.00%
	AL-RVU	0.8673	0.7825	1.00%	0.8741	0.7742	5.00%	0.8833	**0.7955**	10.00%
	AL-SS	0.8773	0.7599	1.00%	0.8761	0.7510	5.00%	0.8825	0.7905	10.00%
	AL-R	0.8703	0.7949	0.99%	0.8689	0.7928	4.98%	0.8739	0.7592	9.97%
	AL-FU	0.7926	0.0000	0.01%	0.2076	0.0014	0.00%	0.7926	0.0000	0.00%
spam	DBAL	0.8568	0.5924	1.00%	0.8923	**0.6395**	4.95%	0.8922	0.6361	9.90%
	PRG	0.8443	0.5342	1.01%	0.9096	0.6495	5.01%	0.9079	0.6484	10.01%
	AL-VU	0.8497	0.5491	1.01%	**0.9006**	0.6373	5.01%	0.8998	**0.6367**	10.01%
	AL-RVU	**0.9049**	**0.6373**	1.01%	0.8849	0.6314	5.01%	0.8997	0.6284	10.00%
	AL-SS	0.8737	0.5617	1.00%	0.8795	0.6207	5.01%	0.8527	0.5566	10.01%
	AL-R	0.7972	0.5368	0.90%	0.8801	0.5976	5.17%	0.8880	0.6059	9.97%
	AL-FU	0.2518	0.0000	0.01%	0.2518	0.0000	0.01%	0.2518	0.0000	0.01%
tripadvisor	DBAL	0.7554	**0.6347**	1.00%	0.7843	**0.7260**	4.98%	**0.8193**	0.6965	9.95%
	PRG	0.7375	0.5931	1.00%	0.7481	0.1945	5.00%	0.8188	0.6566	10.00%
	AL-VU	0.7204	0.4414	1.00%	0.7966	0.5003	5.00%	0.8111	0.6988	10.00%
	AL-RVU	0.7371	0.5519	1.00%	**0.7970**	0.6865	5.00%	0.7909	0.6066	10.00%
	AL-SS	0.7358	0.0899	1.00%	0.7795	0.5447	5.00%	0.8184	**0.7251**	10.00%
	AL-R	0.7566	0.5803	0.96%	0.7839	0.5541	5.04%	0.7938	0.5819	9.85%
	AL-FU	**0.7617**	0.5030	1.00%	0.7555	0.6574	5.00%	0.8012	0.6009	10.00%
twitter	DBAL	0.7485	**0.4660**	0.99%	0.8271	**0.0896**	4.95%	0.8364	**0.0635**	9.89%
	PRG	0.8083	0.1650	1.00%	0.8368	0.0151	5.01%	0.8382	0.0160	10.00%
	AL-VU	0.7984	0.1481	1.00%	0.8368	0.0151	5.01%	0.8382	0.0160	10.00%
	AL-RVU	0.7926	0.2215	1.00%	**0.8436**	0.0205	5.01%	0.8379	0.0175	10.00%
	AL-SS	0.8007	0.0963	1.00%	0.8413	0.0368	5.01%	**0.8423**	0.0130	10.00%
	AL-R	**0.8356**	0.0828	0.88%	0.8417	0.0246	5.13%	0.8400	0.0338	9.91%
	AL-FU	0.5766	0.0699	1.00%	0.8391	0.0089	0.33%	0.1556	0.0053	0.02%
weather	DBAL	0.6775	0.3127	1.00%	**0.6868**	0.0475	4.97%	**0.7318**	**0.6259**	9.95%
	PRG	0.6842	0.0417	1.00%	0.6865	0.0597	5.00%	0.6837	0.0298	10.00%
	AL-VU	0.6737	0.1176	1.00%	0.6843	0.0044	5.00%	0.6837	0.0153	10.00%
	AL-RVU	**0.6874**	0.1457	1.00%	0.6846	0.0773	5.00%	0.6972	0.4343	10.00%
	AL-SS	0.6845	0.1841	1.00%	0.6828	0.0444	5.00%	0.6823	0.0801	10.00%
	AL-R	0.6091	**0.5979**	0.97%	0.6746	**0.3389**	5.03%	0.6786	0.1071	9.82%
	AL-FU	0.6854	0.0000	0.02%	0.6854	0.0000	0.03%	0.6831	0.0079	0.11%

capabilities. Furthermore, we aim to investigate the potential of incorporating domain knowledge or prior information to improve the performance of the proposed strategy.

CRediT authorship contribution statement

Gabriel J. Aguiar: Methodology, Software, Writing and Reviewing. **Alberto Cano:** Methodology, Writing, Reviewing.

Declaration of competing interest

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

Data availability

No data was used for the research described in the article.

References

[1] J. Gama, Knowledge Discovery from Data Streams, CRC Press, 2010.
[2] F. Bayram, B.S. Ahmed, A. Kassler, From concept drift to model degradation: an overview on performance-aware drift detectors, Knowl.-Based Syst. (2022) 108632.
[3] 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.
[4] G. Aguiar, B. Krawczyk, A. Cano, A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework, Mach. Learn. (2023).
[5] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (2014) 1–37.
[6] I. Žliobaitė, A. Bifet, B. Pfahringer, G. Holmes, Active learning with drifting streaming data, IEEE Trans. Neural Netw. Learn. Syst. 25 (2013) 27–39.
[7] X. Shen, Q. Dai, W. Ullah, An active learning-based incremental deep-broad learning algorithm for unbalanced time series prediction, Inf. Sci. 642 (2023) 119103.
[8] O. Reyes, A.H. Altalhi, S. Ventura, Statistical comparisons of active learning strategies over multiple datasets, Knowl.-Based Syst. 145 (2018) 274–288.
[9] J. Shan, H. Zhang, W. Liu, Q. Liu, Online active learning ensemble framework for drifted data streams, IEEE Trans. Neural Netw. Learn. Syst. 30 (2018) 486–498.
[10] S. Liu, S. Xue, J. Wu, C. Zhou, J. Yang, Z. Li, J. Cao, Online active learning for drifting data streams, IEEE Trans. Neural Netw. Learn. Syst. 34 (2023) 186–200.
[11] B. Krawczyk, A. Cano, Adaptive ensemble active learning for drifting data stream mining, in: IJCAI, 2019, pp. 2763–2771.
[12] M. Woźniak, P. Zyblewski, P. Ksieniewicz, Active weighted aging ensemble for drifted data stream classification, Inf. Sci. 630 (2023) 286–304.
[13] R.S.M. Barros, S.G.T.C. Santos, A large-scale comparison of concept drift detectors, Inf. Sci. 451 (2018) 348–370.
[14] R.A. Coelho, L.C.B. Torres, C.L. de Castro, Concept drift detection with quadtree-based spatial mapping of streaming data, Inf. Sci. 625 (2023) 578–592.
[15] Ł. Korycki, B. Krawczyk, Concept drift detection from multi-class imbalanced data streams, in: IEEE International Conference on Data Engineering, 2021, pp. 1068–1079.
[16] A. Cano, B. Krawczyk, ROSE: robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams, Mach. Learn. 111 (2022) 2561–2599.
[17] D. Tu, L. Chen, M. Lv, H. Shi, G. Chen, Hierarchical online NMF for detecting and tracking topic hierarchies in a text stream, Pattern Recognit. 76 (2018).
[18] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 226–235.
[19] C. Fahy, S. Yang, M. Gongora, Scarcity of labels in non-stationary data streams: a survey, ACM Comput. Surv. 55 (2022).
[20] E. Lughofer, On-line active learning: a new paradigm to improve practical useability of data stream modeling methods, Inf. Sci. 415 (2017) 356–376.
[21] D. Cacciarelli, M. Kulahci, A survey on online active learning, arXiv preprint, arXiv:2302.08893, 2023.
[22] V.E. Martins, A. Cano, S.B. Junior, Meta-learning for dynamic tuning of active learning on stream classification, Pattern Recognit. (2023) 109359.
[23] N. Cesa-Bianchi, C. Gentile, L. Zaniboni, M. Warmuth, Worst-case analysis of selective sampling for linear classification, J. Mach. Learn. Res. 7 (2006).
[24] J. Lu, P. Zhao, S.C. Hoi, Online passive-aggressive active learning, Mach. Learn. 103 (2016) 141–183.
[25] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, J. Mach. Learn. Res. 2 (2001) 45–66.
[26] D. Roth, K. Small, Margin-based active learning for structured output spaces, in: European Conference on Machine Learning, 2006, pp. 413–424.
[27] M.-F. Balcan, A. Broder, T. Zhang, Margin based active learning, in: Annual Conference on Learning Theory, 2007, pp. 35–50.
[28] 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.
[29] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, T. Radauer, Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances, Inf. Sci. 355 (2016) 127–151.
[30] H. Mouss, D. Mouss, N. Mouss, L. Sefouhi, Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system, in: 5th Asian Control Conference, vol. 2, 2004, pp. 815–818.
[31] A. Castellani, S. Schmitt, B. Hammer, Stream-based active learning with verification latency in non-stationary environments, in: International Conference on Artificial Neural Networks, 2022, pp. 260–272.
[32] G.J. Aguiar, A. Cano, An active learning budget-based oversampling approach for partially labeled multi-class imbalanced data streams, in: ACM Symposium on Applied Computing, 2023, pp. 382–389.
[33] Ł. Korycki, B. Krawczyk, Online oversampling for sparsely labeled imbalanced and non-stationary data streams, in: International Joint Conference on Neural Networks, 2020, pp. 1–8.
[34] Ł. Korycki, B. Krawczyk, Instance exploitation for learning temporary concepts from sparsely labeled drifting data streams, Pattern Recognit. 129 (2022) 108749.
[35] V. Vaquet, F. Hinder, J. Brinkrolf, B. Hammer, Combining self-labeling and demand based active learning for non-stationary data streams, arXiv preprint, arXiv:2302.04141, 2023.
[36] H. Zhang, W. Liu, L. Sun, L. Chen, Z. Ding, Q. Liu, Analyzing network traffic for protocol identification: an ensemble online active learning method, in: International Conference on Big Data and Information Analytics, 2020, pp. 167–172.
[37] B. Krawczyk, B. Pfahringer, M. Woźniak, Combining active learning with concept drift detection for data stream mining, in: IEEE International Conference on Big Data, 2018, pp. 2239–2244.
[38] X. Zhu, P. Zhang, X. Lin, Y. Shi, Active learning from data streams, in: IEEE International Conference on Data Mining, 2007, pp. 757–762.
[39] G.I. Webb, E. Keogh, R. Miikkulainen, Naive Bayes, Encycl. Mach. Learn. 15 (2010) 713–714.
[40] P. Domingos, G. Hulten, Mining high-speed data streams, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 71–80.
[41] H.M. Gomes, A. Bifet, J. Read, J.P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, T. Abdessalem, Adaptive random forests for evolving data stream classification, Mach. Learn. 106 (2017) 1469–1495.
[42] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: European Conference on Machine Learning and Knowledge Discovery in Databases, 2010, pp. 135–150.
[43] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian Symposium on Artificial Intelligence, 2004, pp. 286–295.
[44] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: SIAM International Conference on Data Mining, 2007, pp. 443–448.
[45] B. Settles, Active Learning Literature Survey, University of Wisconsin-Madison, 2009.
[46] C.C. Aggarwal, X. Kong, Q. Gu, J. Han, S.Y. Philip, Active learning: a survey, in: Data Classification, Chapman and Hall/CRC, 2014, pp. 599–634.
[47] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, T. Seidl, MOA: massive online analysis, a framework for stream classification and clustering, in: First Workshop on Applications of Pattern Analysis, PMLR, 2010, pp. 44–50.
[48] A.C. Bifet Figuerol, R. Gavaldà Mestre, Adaptive Parameter-Free Learning from Evolving Data Streams, Universitat Politècnica de Catalunya, 2009.
[49] H.M. Gomes, J. Read, A. Bifet, Streaming random patches for evolving data stream classification, in: IEEE International Conference on Data Mining, 2019, pp. 240–249.
[50] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.