Neurocomputing 442 (2021) 125–145

Contents lists available at ScienceDirect

Neurocomputing

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

# A novel semi-supervised ensemble algorithm using a performance-based selection metric to non-stationary data streams

Shirin Khezri [a], Jafar Tanha [b,*], Ali Ahmadi [c], Arash Sharifi [d]

[a] Department of Computer Engineering and Information Technology, Payame Noor University (PNU), Tehran, Iran
[b] Computer and Electrical Engineering Department, University of Tabriz, Tabriz, Iran
[c] Department of Computer Engineering, K.N. Toosi University of Technology, Iran
[d] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

ARTICLE INFO

*Article history:*
Received 19 August 2020
Revised 26 January 2021
Accepted 12 February 2021
Available online 2 March 2021
Communicated by Zidong Wang

*Keywords:*
Data streams
Ensemble learning
Semi-supervised learning
Concept drift

ABSTRACT

In this article, we consider the semi-supervised data stream classification problems. Most of the semi-supervised learning algorithms suffer from a proper selection metric to select from the newly-labeled data points through the training procedure. These approaches mainly employ the probability estimation of the underlying base learners to their predictions as a selection metric, which is not optimal in many cases. Handling different kinds of concept drifts is another issue in data streams. Considering these issues, we propose a novel Semi-Supervised Ensemble algorithm using a Performance-Based Selection metric to data streams, named SSE-PBS. The proposed selection metric is based on a pseudo-accuracy and energy regularization factor. We show that SSE-PBS improves classification performance and handles different kinds of concept drifts. The proposed algorithm can also employ any kind of incremental base learners. In the experiments, we report the results of two base learners on synthetic and real-world datasets. The experiments show that SSE-PBS significantly improves the classification performance of the used underlying base learners. Furthermore, we compare the results to the state-of-the-art supervised and semi-supervised approaches in data streams. The results further show that SSE-PBS outperforms the other methods when there is a small portion of labeled instances.

© 2021 Elsevier B.V. All rights reserved.

## 1. Introduction

Currently, a large number of high-speed data streams is produced by the technology improvement in many applications and practical domains, like social networks, sensor networks, and network traffic and monitoring [15,55]. Therefore, mining from these kinds of data streams is interesting and challenging. The key idea here is how to exploit hidden structures and patterns from continuous data streams employing machine learning approaches [24]. Meanwhile, increasing the large scale, high dimensional, and evolving data in many real-world and practical domains indicates the need for development of various machine learning algorithms for data stream classification tasks. In [48], the state-of-the-art data stream approaches are addressed regarding the clustering and classification tasks in this field. This research addresses a general approach for stream mining and explains the differences between traditional data mining and data stream mining. Different challenges related to data stream mining are presented in various scopes in [54]. This study addresses several stream mining approaches for classification and clustering tasks.

The main challenge in learning from data streams is the concept drift issue. When concept drift occurs, the data distribution changes over time [1,12]. For example, in credit card fraud detection, when a new fraud case is found which has not been seen before, the classifier may not be able to detect it correctly [46]. These changes gradually decrease the classification performance of the constructed model. Recently, several methods have been presented to deal with the concept change problem in data streams, see [26,40,62,29]. The main concern of these approaches is to update the constructed model such that the new model is compatible with the newest data chunk. Two main methods have been addressed to data streams: explicit [2] and implicit approaches [41]. In explicit approaches, the model is equipped with a change detection method and retrains when a trigger has been flagged. However, in implicit approaches, classifiers are updated at regular intervals regardless of whether a change has occurred or not.

Recently, ensemble methods are used to handle data stream classification tasks. These approaches are considered as an implicit method in data steams mining. They tend to be more accurate than

* Corresponding author.
*E-mail addresses:* sh.khezri@pnu.ac.ir (S. Khezri), tanha@tabrizu.ac.ir (J. Tanha), ahmadi@kntu.ac.ir (A. Ahmadi), a.sharifi@srbiau.ac.ir (A. Sharifi).

https://doi.org/10.1016/j.neucom.2021.02.031
0925-2312/© 2021 Elsevier B.V. All rights reserved.

single classifiers because of the reduction in the error variance. The ensemble model is continuously reformed by adding a new classifier, removing the oldest or the weakest classifier, and increasing or decreasing the classifiers weights using some metrics, such as the accuracy in the current chunk. Hence, they employ a natural mechanism to omit the abolished knowledge in a non-stationary environment, see [40,29,19,53,68,51].

As mentioned earlier, different kinds of approaches have been proposed to handle concept drift in data streams, but most of these methods have been originally designed for labeled data stream tasks. However, achieving labeled data is not a straightforward task in many domains. In many cases, the number of labeled data is limited, because data streams are originally high-speed and huge, therefore hard to label them. Typically, labeling is a tedious, expensive, and time-consuming task in many fields. Hence, it requires expert supervision, special devices, and costly experiments in many applications. Meanwhile, supervised machine learning algorithms cannot directly deal with these kinds of tasks. In this case, semi-supervised learning can be a proper solution. The semi-supervised learning algorithms employ the labeled and unlabeled data to build a strong classifier. The goal of semi-supervised learning is to exploit the hidden pattern of the unlabeled data and add it to the obvious knowledge of labeled data in order to reinforce the classifier [61]. Semi-supervised learning can be addressed as a classification or clustering task [61]. Recently, several semi-supervised approaches have been proposed for learning from limited labeled data streams [65,35]. Examples are cluster-based [47,32,44,17], EM-based [10], iterative-based [65,35], and EVL-based approaches [22,63]. In this article, we propose a new ensemble-based semi-supervised learning framework to deal with the semi-supervised data stream classification tasks.

Most recently, several algorithms are addressed to handle the semi-supervised classification tasks in non-stationary environments, see [10,65]. Although, several semi-supervised methods have been presented to limited labeled data for non-stationary data streams, there are still several challenging problems to overcome. First, most of the proposed semi-supervised approaches have employed cluster-based algorithms which suffer from the lack of a proper similarity criterion. Second, many algorithms are based on a fixed-size chunk approach, and their classification performance typically relies on the chunk size. However, defining the chunk size is not a straightforward task. Using too small chunk size, makes the classifiers more flexible to changes but it increases computational costs and may lead to poor classification performance in a stable period. While too large chunk size leads to more accurate classifiers in a stable period but it results in slow adaptation to drifts. Third, the data selection metric based on a high confidence score in iterative semi-supervised techniques is not a reasonable method because the confidence score is a criterion that indicates a specific sample's classification accuracy instead of a metric that can evaluate the potential of a sample's contribution in training procedure for improving the constructed classification model.

Two main difficulties in employing an ensemble approach for semi-supervised data streams are: (1) the weighting factor for each component classifier in order to decide whether to update or remove it and (2) the selection metric to select informative examples from the unlabeled data at each chunk. Therefore, to formulate an ensemble algorithm, we need a novel weighting factor for each component classifier and a suitable selection metric to select from the newly-labeled data. To handle these issues, we address a novel Semi-Supervised Ensemble algorithm using a Performance-Based Selection metric, named SSE-PBS, which employs a chunk-based ensemble model and incremental base learners to achieve accurate predictions to deal with various kinds of concept drifts. Our proposed algorithm updates its component classifiers incrementally instead of using batch static learners. Since the components are incrementally updated, we show that the model performs well in dealing with sudden changes. Besides, the algorithm is not highly dependent on chunk size and can process smaller chunks without decreasing its classification accuracy. On the other hand, SSE-PBS can react to gradual drift by weighting its component classifiers based on their prediction error rates. Furthermore, our proposed algorithm does not rely on the confidence of the classifier as a selection criterion to select unlabeled data for improving classification performance. Therefore, the use of confidence score as the selection metric in semi-supervised training procedure may lead to inaccurate estimation of actual distribution, thereby resulting in performance degradation, even if the selected samples are classified correctly. Instead, we use a new method to evaluate the ability of candidate unlabeled data points to improve classification performance. In particular, only unlabeled samples that can participate for improving classification accuracy are selected for updating the classification model. Our main contributions in this article are presented as follows:

- We develop a new ensemble-based semi-supervised learning framework that can employ any incremental base learner.
- We employ a new metric for the evaluation of component classifiers in terms of their weighting factors.
- We propose a novel selection metric to select from the newly-labeled data based on the classification performance.
- We formulate the final metric for evaluating the performance of a model based on pseudo-accuracy and energy regularization.

In fact, SSE-PBS is a hybrid approach that handles gradual and sudden changes by weighting classifiers based on their error rates. It assigns the highest possible weight to the latest classifier and updates the classifiers incrementally after each chunk. In order to show the effectiveness of the proposed metric, we develop another version of SSE which uses a confidence score instead of performance as a selection metric. We then employ several experiments on synthetic and real-world datasets to address the performance of the SSE-PBS algorithm. Our experimental results indicate that SSE-PBS gives better classification performance than the state-of-the-art semi-supervised learning algorithms when there is a limited number of labeled examples. We further observe that SSE-PBS significantly improves the classification performance of supervised learning algorithms. Statistical tests verify our results in the experiment.

The rest of this paper is presented as follows. Section 2 gives the literature review. Section 3 presents the proposed semi-supervised ensemble algorithm for data streams. The experiments are addressed in Section 4. Section 5 gives the experimental results. Finally, Section 6 presents the conclusion.

## 2. Literature review

Concept drift detection is the main challenge in data stream classification problems. Various algorithms have been addressed to deal with concept drift in non-stationary data streams, see [26,40,62,29]. These approaches can be categorized into explicit and implicit approaches.

Explicit approaches are an effective way to deal with concept drift since they avoid the unnecessary updating of the current classifier. Explicit approaches mainly use a detection method and retrain when a trigger has been flagged. Most of the methods rely on an indicator, called detector, which monitors certain features of data streams over time, see [43,2]. The most common indicators are classification errors [52], statistical distribution [56], and density estimation [45]. One of the explicit approaches is the adaptive

size time window (ADWIN) [3]. The size of the window is tuned adaptively to the current concept drift. Generally, if concept drift is detected, the window size will decrease to remove the outdated samples; otherwise, the window size will be extended to contain more samples.

In implicit approaches, classifiers are updated at regular intervals regardless of whether a change has occurred or not. In other words, the classifier naturally adapts itself to changes. Weighted examples [37] and fixed-size time windows [3] are two examples of implicit approaches. In weighted examples, smaller weights are assigned to old samples based on their age. In [41], a weighted Naive Bayes scheme is proposed to handle concept drift, called WNB-CD. By weighting data instances based on their importance and gradually forgetting the out-of-date data points, WNB-CD can adaptively adjust itself to the current concept. In fixed-size time windows, the size of the window should be determined by the user. It is a tuning parameter that has a huge impact on the performance of the resulting model. Having a small window helps the learner to adapt to drifts faster, but it decreases the accuracy of the model in stable periods. On the other hand, a large window leads to higher accuracies in stable periods while the model reacts more slowly to drifts.

Recently, ensemble approaches are extensively used in data stream classification problems. These approaches can also be considered as implicit approaches. They tend to be more accurate than single classifiers because of the reduction in the error variance. An ensemble model is continuously reconstructed by adding a new classifier, removing the weakest classifier or the oldest, and amending the weights of the component classifiers. Hence, they have a natural method to handle gradual drifts, see [40,29,19], while methods based on drift detectors are more suitable for coping with abrupt drift. Thus, in some cases, ensemble algorithms are equipped with drift detectors. For example, Leveraging bagging [6], ADWIN online bagging [7], and Adaptive Random Forest (ARF) [30] use the ADWIN algorithm [3].

Two popular kinds of ensembles in non-stationary data streams are: online ensembles which are trained incrementally by processing each incoming examples and ensembles based on processing chunks (blocks or batches) of data. In online ensemble algorithms, a pool of online classifiers is used through the training procedure. These classifiers are updated incrementally and adapt to changes with arriving each single instance, see [6,7,38,30,53,68,51,39,49,9].

In [50], an online version of bagging is introduced, named Oza. In Oza, the component classifiers are incremental learners that combine their decisions using a simple majority voting algorithm. The sampling, crucial to batch bagging, is performed incrementally by presenting each example to a component k times, where k is defined by the Poisson distribution. [6] introduced a modification of Oza's algorithm, called Leveraging Bagging, which aims to add more randomization to the input and output of the base classifiers. To cope with evolving data streams, Oza and Leveraging Bagging are equipped with ADaptive WINdowing (ADWIN) drift detector [3], which in this paper are called OzaAdw [7] and Lev respectively.

Following this approach, in [30], another incremental streaming classifier is presented for evolving data streams, called adaptive random forest (ARF) algorithm. ARF is an extension of the classical Random Forest algorithm [11], which combines the trait of batch algorithms with dynamic update methods to handle evolving data streams. ARF uses a theoretically authentic re-sampling method based on online bagging [50] and an updated adaptive strategy to deal with evolving data streams. This adaptive strategy is based on using a drift monitor per tree to track warnings and drifts, and train the new trees in the background (when a warning is detected) before replacing them (when a drift is detected).

In [39], an improvement of online ensembles is proposed using boosting with the abstaining option, called OAbst. Instead of usual voting, the classifiers contribute to the final decision if their confidence level on an arriving instance exceeds a certain threshold; otherwise, they abstain.

The chunk-based ensemble algorithms are constructed based on the following strategy: a new classifier is trained using each arriving data chunk, and then the available component (base) classifiers are evaluated using the newest chuck. Next, the weakest base classifier will be replaced with the new classifier trained on the most recent chunk. The final decision is achieved by the weighted majority voting method. There are several approaches that implement a different strategy to weigh their classifiers in the pool, such as weighting based on quality score [57], prediction error [64], and accuracy [21].

In the chunk-based ensemble approach, adjusting the chunk size is crucial. However, a large chunk may construct a better classifier in a stable period, but may lead to a slow adaptation to drifts when the data includes more than one concept. On the other hand, a trained model on a small chunk will react quickly to changes, but in a stable period might give a weaker classification model and would increase computational costs [40,13].

Most recently, there is a group of hybrid techniques that combines the data chunk scheme with an incremental component. [13] proposed the Accuracy Updated Ensemble (AUE2), which combines a chunk-based ensemble model with incremental base learners. This method reacts to various kinds of concept drifts [13].

Similarly, in [14], Kappa Updated Ensemble approach (KUE) is proposed to handle data steam mining tasks. KUE is a combination of online components and chunk-based ensemble approaches that uses Kappa statistic for dynamic weighting and selection of base classifiers. In order to achieve higher diversity among base learners, each of them is trained using a different subset of features and new instances with a given probability based on the Poisson distribution. Furthermore, the ensemble model is updated by new classifiers only when they contribute positively for improving the quality of the model. This is similar to the AUE2 algorithm [13]. However, it uses the Kappa statistic instead of accuracy to drive the competence of the classifiers.

As mentioned above, various approaches have been proposed to deal with concept drift in data streams, however, in most of them it is assumed that incoming data is fully labeled. Dealing with the concept drift is a main challenging task in non-stationary data streams. Moreover, in practice, obtaining labeled data is an expensive and difficult task. Therefore, with limited access to labeled data, handling the concept drift will be even more challenging. Recently, several semi-supervised approaches have been proposed for learning from limited labeled data streams, see cluster-based [47,32,44,66,17], EM-based [10], iterative-based [65,35], and EVL-based approaches [22,63].

In [47], a cluster-based classification model is proposed, called ReaSC. On each limited labeled chunk, K clusters are constructed using a semi-supervised clustering algorithm based on EM. The summary of statistical information of each cluster preserved as a micro-cluster. The micro-clusters are used as a classification model, and classification is done with k-nearest clusters. To handle the concept drift, a pool of component classifiers is constructed. With arriving each data chunk, a new model is constructed, and the ensemble classifier is updated by selecting the best L models from the L + 1 models, based on their accuracies on the labeled portion of the data chunk.

Recently, a semi-supervised pool and accuracy-based stream classification algorithm, called SPASC, is addressed in [32]. It maintains a pool of weighted cluster-based classifiers, each of which corresponds to one specific concept and is built upon an EM algorithm. A new classifier is built when the performance of all classifiers in the model fails to achieve a specific performance. Hence, when a recurring concept drift occurs, this method can adapt to

it using the ensembles of classifiers. However, SPASC is not suitable for handling new and non-recurring drifts. In [66], a semi-supervised classification method for data streams is proposed using BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [71] and a local structure mapping (SCBELS). In SCBELS, the local structure mapping strategy [28] is utilized to compute local similarity around each sample and combined with the semi-supervised method [32] to detect concept drift.

In [44], a semi-supervised learning algorithm is proposed on limited labeled data streams to address concept drift. In this approach, a decision tree is constructed as a classifier. Through the tree creation procedure, a clustering algorithm based on k-means is employed to generate concept clusters. Meanwhile, the unlabeled data is labeled by the k-means algorithm. Next, the concept change is monitored by comparing the old and current concept clusters. Another semi-supervised algorithm is proposed to deal with concept drift data streams, including labeled and unlabeled data in [10]. This algorithm monitors concept drift according to comparing the distribution difference of sequential data chunks. A new classifier is built using the EM algorithm on the recent data when drift occurs; otherwise, the current classifier remains without any changes. In [17], an online semi-supervised learning approach is presented by modeling concepts with a set of micro-clusters. These micro-clusters are dynamically maintained to capture the evolving concepts with error-based representative learning.

Self-training and co-training algorithms are iterative methods for semi-supervised learning [61]. In [35], a novel semi-supervised approach is presented to data stream classification problems, named STDS. This algorithm is an iterative approach that is based on an extended self-training approach and an incremental base learner. In STDS to exploit similarity information along with the classifier prediction, an ensemble of clustering algorithms is used to select a set of high-confidence predictions in order to improve the selection metric. Moreover, drift detection is based on comparing Kullback–Leibler (KL) divergence of sequential data chunks. Similarly, another semi-supervised algorithm, named the Streaming Co-Forest algorithm (SCo-F), is proposed to deal with classification constraints in limited labeled data streams [65]. SCo-F merges the traditional co-training approach [8] with the extended Random Forest algorithm to data streams [11]. Initially, random trees are trained using subsets of labeled data in the first chunk based on the bootstrapping method. Next, at each arriving data chunk, SCo-F employs the ADWIN algorithm [4] as a drift detector. If a change is detected, the worst classifiers are replaced by new ones; otherwise, the model is left without any change. Finally, each base learner is updated using the high confidence newly labeled instances selected by the other random trees.

In some semi-supervised classification approaches, one of the most reasonable assumptions is named extreme verification latency (EVL) which provides only unlabeled data after initialization. In other words, the labeled data are unavailable after initialization (or provided very sporadically).

One of the semi-supervised classifiers based on the EVL, assumption is COMPOSE [20]. The main scenario behind COMPOSE is the core support extraction (CSE), which is extracting the most representative instances. The unlabeled instances are labeled previously by the semi-supervised classifier. Next, for each class, $\alpha$-shape is constructed, then compacted, and the geometric center of each class is extracted. These instances are used as the labeled data for the next chunk. Likewise, AMANDA algorithm relies on the EVL assumption [22]. In the CSE step of AMANDA the unlabeled data are weighted by the kernel density estimation method, and the denser instances are served as the labeled data in the next chunk.

Even though several semi-supervised approaches have been proposed to limited labeled data streams in non-stationary environments, as mentioned above, there are still several challenging problems to overcome. First, most of the proposed semi-supervised approaches have mainly used cluster-based approaches and suffer from the lack of a proper similarity criterion. Second, most algorithms are based on the fixed-size chunk, and their classification performance mainly relies on the chunk size. Defining the chunk size is not a straightforward task. Using too small chunk sizes, the classifier reacts quickly to changes but would increase computational costs and may lead to poor classification performance in a stable period. While using too large chunk sizes lead to more accurate classifiers in a stable period, but results in slow adaptation to drifts when there is more than one concept. Third, the data selection metric based on a high confidence score in the iterative semi-supervised techniques is not a reasonable method, because the confidence score is a criterion to compute the accuracy of a classifier on a specific sample, and it is not a criterion to evaluate the potential of the sample to improve a model.

To address these problems, a new semi-supervised ensemble algorithm is proposed that considers the assumption of a chunk-based ensemble algorithm and incremental base learner to achieve accurate predictions to deal with various kinds of concept drift. Furthermore, a new selection criterion according to performance is proposed in which only those unlabeled data have been selected, which can help to improve classification accuracy.

## 3. Ensemble-based semi-supervised data streams framework

In this section, we initially give the definition of the concept drift and the semi-supervised data streams setting, then we discuss the proposed ensemble-based algorithm, the Confidence-Based Selection metric (CBS), and the Performance-Based Selection metric (PBS). Finally, a variation of the semi-supervised ensemble algorithm is addressed, called SSE-CBS.

### 3.1. The definition of concept drift

The traditional machine learning approaches are effective when the data distribution remains stationary over time. Nevertheless, in data streams, the distribution of data may change over time. This change leads to the concept drift problem, see [25,18,42,62]. Hence, employing traditional machine learning algorithms may not be a proper method to deal with data stream classification tasks.

Basically, the concept drift is the main challenge in data stream classification. Let $X$ be a set of input variables and $y$ be a target variable with $P_j$ as the data distribution at time $t_j$. When the data distribution changes over time, the concept drift occurs through a training procedure, i.e.:

$$\exists X : P_{t_0}(X, y) \neq P_{t_1}(X, y), \tag{1}$$

where $P_{t_0}$ is the joint distribution between a set of input variable $X$ and target variable $y$ in the time point $t_0$.

When concept drift occurs, three cases may happen as follows [27]:

- Prior probabilities of classes $p(y)$ may change;
- The classes conditional probabilities $p(X|y)$ may change;
- The posterior probabilities of classes $p(y|X)$ may change;

### 3.2. The setting of semi-supervised data streams

Suppose $D = \{(x_i, y_i)\}_1^{\infty}$ be an infinite sequence of data points $(x_i, y_i)$ for data streams, where $x_i$ is an $n$-dimensional vector and

$y_i \in \{-1, +1\}$. Note that the data streams are originally infinitive. Therefore, we require to divide the data into chunks with the same size, $D_1, D_2, \cdots, D_i, \cdots$, in order to construct the classification model [48].

Note that each chunk $D_i$ includes a set of limited labeled examples $L_i$ and a large set of unlabeled data points $U_i$ where $l \ll u$ that $l$ and $u$ are the numbers of labeled and unlabeled instances, respectively.

### 3.3. The proposed ensemble-based algorithm

Most of the data stream classification algorithms are typically able to cope with specific concept drifts when fully labeled data is available. However, in practice, labeled data is rare and difficult to achieve. Meanwhile, the type of drift is not known in many practical domains.

The main goal of this article is to propose an ensemble algorithm that can handle a variety of changes in the environment when labeled data is limited. To achieve this goal, we propose an ensemble semi-supervised algorithm that combines the weighting mechanism based on the accuracy of chunk-based ensemble algorithms with the incremental nature of online base learner algorithms. We call the resulting algorithm SSE-PBS, which stands for Semi-Supervised Ensemble using Performance-Based Selection metric.

The proposed algorithm builds a set of weighted base learners and assigns the class label to unlabeled data points using the majority weighted voting strategy over a combination of component outputs. When a data chunk arrives, a new classifier is trained using labeled data. The weakest classifier (classifier with minimum weight) is replaced by the new classifier. The efficiency of each classifier is estimated by measuring its error rate based on the labeled data of the latest chunk. Afterward, a pseudo-label is assigned to unlabeled data using the ensemble of component classifiers, and the confidence score of each sample is measured. Then, a subset of pseudo-labeled data is selected based on a selection metric, and they are added to labeled data in the current chunk to update the remaining ensemble members. Fig. 1 gives an overview of our proposed method.

Two main difficulties in employing the ensemble approach for semi-supervised data streams are: (1) the weighting factor for each component classifier in order to whether update or remove it and (2) a selection metric to select informative examples from the unlabeled data at each chunk. Therefore, to formulate the proposed ensemble algorithm, we need a novel weighting factor for each component classifier and a suitable selection metric to select from the newly-labeled data. In the following sections, we address these issues and propose two methods.

#### 3.3.1. The SSE-PBS algorithm

As mentioned earlier, to formulate the proposed ensemble algorithm, we need a novel weighting factor for each component classifier.

Based on AUE2 [13], we use the following weighting factor for each component classifier where for each incoming chunk $D_i, \alpha_{ij}$ is the weight of learner $h_j, j \in \{1, \cdots, K\}$. This weighting factor is measured by predicting the error rate on data chunk $D_i$ as shown in Eq. 4.

$$MSE_{ij} = \frac{1}{L_i} \sum_{x,y \in D_i} \left(1 - f_y^j(x)\right)^2 \tag{2}$$

$$MSE_r = \sum_y p(y)(1 - p(y))^2 \tag{3}$$

$$\alpha_{ij} = \frac{1}{MSE_r + MSE_{ij} + \epsilon}, \tag{4}$$

where $f_y^j(x)$ represents the probability of the assigned class of $y$ to sample $x$ using classifier $h_j$, the value of $MSE_{ij}$ demonstrates the estimation error of learner $h_j$ on the labeled data of current chunk $L_i$, and the $MSE_r$ is the mean square error of a classifier with random prediction.

As shown in Eq. 4, a small value $\epsilon$ is added to the denominator to avoid division by zero. The main goal of the weighting formula presented in Eq. 4 is to combine the accuracy of classifiers and the current class distribution. Besides assigning weight to each component classifier in the ensemble approach, a new $h_{new}$ classifier is trained with the proportion of labeled data in the current chunk. Since $h_{new}$ is trained using the latest chunk, it behaves like a perfect classifier, and its weight is calculated according to Eq. 5.

$$\alpha_{new} = \frac{1}{MSE_r + \epsilon}. \tag{5}$$

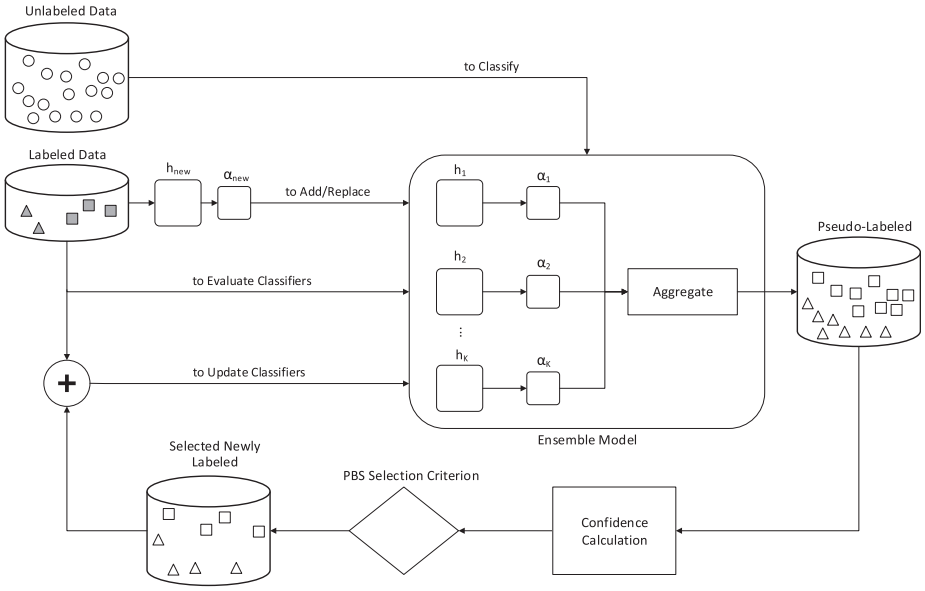



**Fig. 1.** An overview of the proposed SSE-PBS approach.

The prediction error of $h_{new}$ on $D_i$ is ignored within calculating its weight, $\alpha_{new}$, while it is considered in calculating the weight of existing ensemble components. This approach is based on the assumption that the newest chunk is the best representative of the current data distribution.

Since $h_{new}$ is built on the newest chunk, it is considered the best available classifier. If the number of components in the ensemble is less than $k$ ($k$ is the ensemble size), the newest classifier $h_{new}$ is added to the ensemble model; otherwise, the weakest component will be replaced by $h_{new}$. Next, the pseudo-label of the unlabeled data is predicted using ensemble component classifiers (Eq. 6), and their confidence scores are calculated using Eq. 7. Afterward, a subset of newly labeled data is selected based on the proposed novel selection metric, named Performance-Based Selection metric (PBS), see Section 3.5. Then, this subset is added to the labeled data in the current chunk $L_i$ for updating the remaining ensemble members incrementally.

As mentioned earlier, we use Eq. 6 to assign pseudo-label to unlabeled data as follows:

$$m_{MAP} = argmax_{m\in M}\sum_{j=1}^{K}\alpha_{ij} \times P_j(y_m|x) \tag{6}$$

furthermore, Eq. 7 to assign confidence to the predictions as:

$$Confidence(x) = \frac{\sum_{j=1}^{K}\alpha_{ij} \times P_j(y_{m_{MAP}}|x)}{\sum_{m=1}^{M}\left(\sum_{j=1}^{K}\alpha_{ij} \times P_j(y_m|x)\right)}, \tag{7}$$

where $M$ is the number of classes, $P_j(y_m|x)$ represents the probability of assigning class $y_m$ to sample $x$ by the classifier $h_j$, $m_{MAP}$ gives the assigned class label to the unlabeled sample $x$ by the ensemble classifier, and $Confidence(x)$ shows the confidence score of the assigned class label to sample $x$, which is obtained from the weighted probability of the majority class divided by the sum of the weighted probability of all existing classes.

Our proposed ensemble semi-supervised approach adapts fast after sudden changes because its component classifiers are updated after each new coming chunk. Furthermore, on data streams with gradual or incremental changes, the proposed approach may lead to an accurate prediction model because the weights of its classifiers are computed based on their error rates, and the highest possible weight is assigned to the latest classifier.

Algorithm 1 represents the proposed algorithm according to the formulations as mentioned above.

**Algorithm 1:** SSE-PBS algorithm

**Input:** $S$: data streams, $K$: number of ensemble components, $L_i$ and $U_i$: Labeled and Unlabeled data in chunk $D_i$, respectively and $L_i \ll U_i$;
**Output**: $E$: Ensemble of $K$ weighted incremental classifiers;
$E \longleftarrow \varnothing$;
**For** each chunk $D_i$
  $h_{new} \leftarrow$new component classifier built on $L_i$;
  $\alpha_{new} \leftarrow \frac{1}{(MSE_r+\epsilon)}$;
  **For** all classifiers $h_j \in E$ do
    Apply $h_j$ on Labeled data $L_i$ to derive $MSE_{ij}$ based on Eq. 2;
    Compute weight $\alpha_{ij}$ based on Eq. 4;
  end for
  if $|E| < K$ then
    $E \leftarrow E \cup h_{new}$;
  else
    Substitute least accurate classifier in $E$ with $h_{new}$;
  end if
  **For** each $x_u \in U_i$ do
    Assign pseudo-label to each unlabeled data $x_u$ using $E$;
    Calculate the confidence score for each prediction using Eq. 7;
  end for
  Split the unlabeled data into $N$ Bins $b_n : (1 \leqslant n \leqslant N)$ using Algorithm2;
  **For** each $b_n$
    Train a model $C_{b_n}$ using $L_i \cup b_n$;
    Evaluate $C_{b_n}$ based on Eq. 13 and let $f_{b_n} = f(C_{b_n})$;
  end for
  Find $b_{best} = argmax_{b_n \in B}(f_{b_n})$;
  $h_{new} \longleftarrow$ Incrementally Updated$(h_{new}, b_{best})$;
  $L_i = L_i \cup b_{best}$;
  **For** all classifiers $h_j \in E \setminus h_{new}$ do
    $h_j \longleftarrow$ Incrementally Updated$(h_j, L_i)$;
  end For
**End For**

To handle the second issue in the semi-supervised framework, most of the current approaches trust the probability estimation of the underlying base learner to assign confidence to the newly-labeled data. These approaches then select a set of high-confidence predictions based on the probability estimation of the underlying base learners. Skimming over this gives us the feeling that this technique is acceptable and may improve the performance of the constructed classification model. However, in the next section we analyze the confidence-based selection metric and show that it may result in an inaccurate estimation of the underlying data distribution.

Inspired by [70], we propose a novel performance-based selection criterion in data streams. The proposed approach connects the data selection and the accuracy of the classification model. Using this metric, an unlabeled sample is selected based on its potential to improve future models instead of its prediction accuracy.

### 3.4. The confidence-based selection metric (CBS)

In this section, we show that the data selection based on the classification confidence may cause an inferior estimation of the data distribution. We then address the disadvantages of this selection metric.

Basically, in semi-supervised learning algorithms, the confidence score is calculated based on the probability estimation of the underlying base learner. Therefore, selecting the unlabeled data with a high confidence score often leads to the selection of those samples to which the current model has already been adapted [73,69,60]. Updating with such data leads to boosting the ability of the current model for classifying what it already can predict correctly without reducing the estimation bias due to a lack of labeled data.

One of the strengths of semi-supervised learning is that the enormous number of unlabeled samples can be used to make a correct estimation of the distribution of $p(x)$. The question that needs to be answered is whether the selected unlabeled samples based on their confidence score match the proper distribution for $p(x)$ or not. In confidence-based selection metrics, the unlabeled data

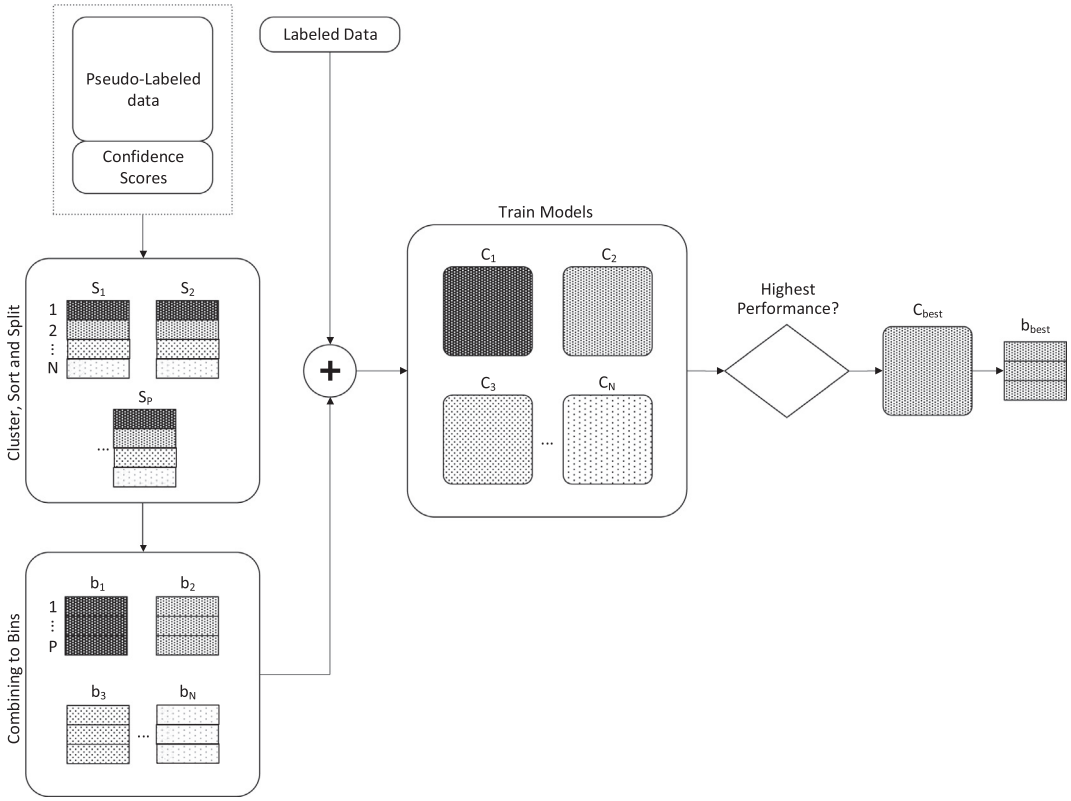



**Fig. 2.** An overview of the proposed PBS selection criterion.

with high confidence score is typically distributed in the specific input area i.e., the areas far away from the decision boundary, instead of being distributed through the whole space [67,59,58]. Consequently, the $p(x)$ obtained from the selected data is not accurate. Furthermore, for a specific class such as $c$, maybe most of the examples are positioned in a low confidence classified region. Therefore, no unlabeled example is chosen from this class. This may lead to a biased estimation of the underlying prior probability of class $c, p(c)$. Although the biased estimation of $p(x)$ or $p(c)$ does not inevitably cause a reduction in efficiency, it does create an unpredictable classification model.

The confidence score is a criterion to compute the classification correctness on a specific sample, instead of a criterion to evaluate the potential participation of the sample to improve a model. According to the discussion above, the use of confidence score as the selection criterion may lead to an inaccurate estimation of the real distribution leading to performance degradation, even if the selected samples are correctly labeled.

Here, we employ a new approach for sample selection using a performance-based approach.

### 3.5. The performance-based selection metric (PBS)

In this section, we introduce a new selection metric based on performance for semi-supervised data streams, named PBS. The proposed selection metric is based on evaluating the potential of candidate data to enhance the performance of the classification model. In this approach, the most informative unlabeled samples that give a higher classification accuracy are selected for updating the model.

The proposed selection method consists of the following steps. Initially, the unlabeled data is divided into some clusters by a clustering algorithm. In each cluster, the samples are sorted ascendingly based on their confidence scores. Afterward, a fraction of each cluster falls into the modules called "Bin"s. In the second step, an objective function is used to determine the ability of each bin to improve the performance of the classification model. Then, the best bin from the unlabeled data is selected and added to the labeled data in the current chunk to update the existing component classifiers. The general overview of the proposed Performance-Based Selection metric is presented in Fig. 2.

The next two sections present the proposed approach steps in more detail.

#### 3.5.1. The partitioning layout

In this step, the unlabeled data is split into several bins of the same size to determine the candidates. First, a model is trained based on the examples of each bin. The samples belonging to each bin must be selected from the whole input area. The confidence score is used as a measurement of how well the sample adapts to the current model. Then, a clustering algorithm, such as k-means, is used to divide the input space into several clusters, and a fraction of each cluster falls into the bins; therefore, the examples in each bin are sampled from the entire input space. Algorithm 2 gives more details.

**Algorithm 2:** Partitioning Schema

1: **Input**: Unlabeled data in a chunk;
2: **Output**: Bins;
3: Set the number of Bins and initialize each Bin with the empty set: $b_n = \varnothing \quad (1 \leqslant n \leqslant N)$;
4: Split unlabeled data to $P$ subspace $S_1, \ldots, S_P$ with a clustering algorithm such k-means;

(*continued on next page*)

5: **For** each subspace $S_p(1 \leqslant p \leqslant P)$:
6: Sort unlabeled example $x$ that $x \in S_p$ according to its confidence score (Eq. 7) ascending;
7: Add each unlabeled example $x$ to one of the $N$ bins that $b_n(1 \leqslant n \leqslant N)$ accepts the examples whose confidence scores are within the range from top $\frac{(n-1)}{N}\%$ to $\frac{(n)}{N}\%$;

*3.5.2. The model evaluation metric*

Our main goal in the proposed approach is to recognize those unlabeled samples that may improve the classification performance i.e., informative data points. The question here is how the accuracy of the classification model can be measured on labeled and unlabeled data. To handle this difficulty, we employ two terms which are pseudo-accuracy and energy regularization as follow.

*3.5.2.1. Pseudo-accuracy.* The goal of this criterion is to compute the classification accuracy on the labeled and unlabeled data. The calculation of accuracy on the labeled data is clear. In the case of unlabeled data, a pseudo-label is assigned to each unlabeled data according to the weighted majority voting strategy of the components predictions in the ensemble model, and then the pseudo-accuracy is calculated based on the following formulation.

To measure the performance of model $C$, a discriminative function $d_C$ is formulated according to Eq. (8), which can distinguish the proper class from other classes.

$$d_C(y_m|x_i) = log(P_C(y_m|x_i)) - log\left[\frac{1}{M-1}\sum_{y\in\{y_1,\ldots,y_M\}, y\neq y_m} P_C(y|x_i)\right]. \quad (8)$$

If $x_i$ is a labeled sample, $y_m$ represents its class; otherwise, it gives the pseudo-label that is assigned using Eq. 6. If $d_C(y_m|x_i)$ is negative, a misclassification has occurred; otherwise, the classification is probably correct. Afterward, we assign the obtained value $d_C(y_m|x_i)$ into a sigmoid function to estimate the pseudo-accuracy, which is formulated as Eq. 9.

$$f_{acc}(C) = \sum_{i=1}^{l+u} \frac{1}{1+exp(-d_C(y_m|x_i)+1)}. \quad (9)$$

*3.5.2.2. Energy regularization.* Since the pseudo-label may be different from the real class label, the pseudo-accuracy criterion is adjusted using another criterion. According to the fact that samples which are close to each other inside the input space are typically similar and should have the same class label, an energy-based factor is added to the performance evaluation [74], which is described as follows:

$$f_{eng}(C) = \sum_{i,j=1}^{l+u} S_{ij} \sum_{m=1}^{M} \left[P_C(y_m|x_i) - P_C(y_m|x_j)\right]^2, \quad (10)$$

where $S_{ij}$ shows the similarity of the samples $x_i$ and $x_j$. The most common criterion of similarity is the Gaussian kernel function, which is defined as Eq. 11.

$$S_G(x_i, x_j) = exp\left(-\frac{d(x_i, x_j)^2}{2\sigma^2}\right). \quad (11)$$

In the Gaussian kernel, the similarity between two points is determined only based on their Euclidean distance and is not influenced by their surroundings. For example, consider two moons dataset as in Fig. 3. The $S_G(a,b)$ is equal to $S_G(a,c)$, because their Euclidean distance is the same i.e., $d(a,b) = d(a,c)$, while based on the clustering assumption, the similarity between $a$ and $c$ should be more than the similarity between $a$ and $b$ because in a specific radius, points $a$ and $c$ have more shared neighbors than points $a$ and $b$ [72]. However, as it is difficult to determine the neighborhood radius, a new similarity criterion has been proposed based on the number of shared-neighbors, *NSN*, in terms of the k-nearest neighbor approach which gives the data points local density. Our proposed method is based on the observation that two points are more similar if they have more shared neighbors. The proposed similarity criterion is defined as follows:

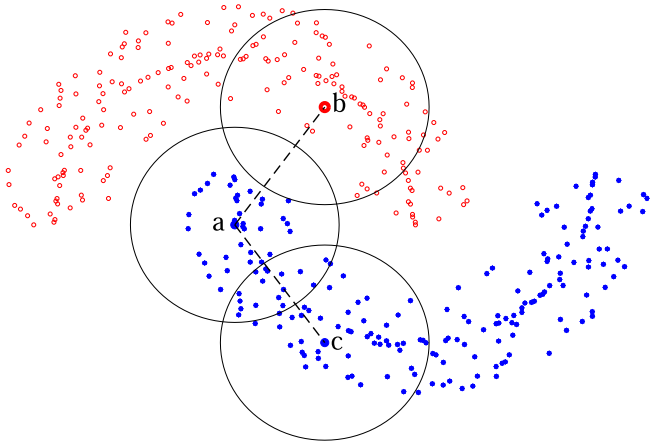



**Fig. 3.** The two moons dataset.

$$S_{ij} = Similarity(x_i, x_j) = exp\left(-\frac{d(x_i, x_j)^2}{NSN_{ij}+1}\right), \quad NSN_{ij} = |N_i \bigcap N_j|, \quad (12)$$

where $N_i$ and $N_j$ represent the set of data in the k-nearest neighborhood of the points $x_i$ and $x_j$, respectively, and $NSN_{ij}$ is the number of shared neighbors of two points $x_i$ and $x_j$ in the k-NN graph. The final metric for evaluating the performance of the model $C$, which is a linear combination of two aforementioned factors, is presented as follows:

$$f(C) = a_1 f_{acc}(C) - a_2 f_{eng}(C), \quad (13)$$

where $a_1$ and $a_2$ are the weights of $f_{acc}(C)$ and $f_{eng}(C)$, which in our experiments are set to 0.75 and 0.25, respectively.

*3.6. A variation of semi-supervised ensemble algorithm: SSE-CBS*

In order to show the effectiveness of the proposed metric, we have developed another version of SSE, which uses a confidence score instead of performance as a selection metric. It is called a Semi-Supervised Ensemble algorithm with a Confidence-Based Selection metric (SSE-CBS). The confidence-based selection metric has been previously explained in detail in Section 3.4.

The SSE framework in both SSE-PBS and SSE-CBS is the same. The only difference between them is the selection metric for choosing the newly labeled data. In confidence-based selection metrics, those instances with labeling confidence exceeding a certain threshold are added to the labeled data in the current chunk to update the remaining ensemble members.

## 4. Experiments

In this section, we conduct several experiments in order to evaluate the proposed algorithm. We then compare the performance of the SSE-PBS algorithm with supervised methods and the state-of-

the-art semi-supervised algorithms employing various datasets, synthetic [5] and real-world [23]. Experiments are designed to answer the following research questions:

- Does the proposed algorithm lead to improvements in the accuracy of the non-stationary semi-supervised data streams?
- Can the proposed algorithm effectively exploit information from the unlabeled data when there is a limited number of labeled data?
- Is the proposed algorithm independent of the base learner?

The first experiment shows a comparison between the state-of-the-art supervised algorithms and the proposed algorithm. The second experiment compares the performance of the proposed algorithm to the state-of-the-art semi-supervised approaches, where the labeled data are 2%, 5%, and 10%. In the next experiment, we show how SSE-PBS can properly deal with concept drift.

### 4.1. Base learners

We employ Hoeffding Tree and Na've Bayes as the component classifiers with their default setting in MOA. However, the proposed approach can be employed as a framework in which other incremental learning algorithms can be used as the base learner as well.

### 4.2. Experimental setup

The classification performance of the SSE-PBS algorithm is compared to the state-of-the-art methods for data stream classification problems, including supervised and semi-supervised, single and ensemble, with and without explicit drift detection. These algorithms include AUE2 [13], KUE [14], OAbst. [39], Oza [50], OzaAdw[7], Lev [6], ARF [30], SPASC [32], SCBELS [66], SCo-F [65], STDS [34], COMPOSE [20], AUE-LL, WNB-CD [41], Naive Bayes (NB), Hoeffting Tree (HT), and SSE-CBS.

AUE2 and KUE are selected as the supervised ensemble algorithms on fully labeled data that use incremental elements in chunk-based ensembles. OAbst. is chosen as a supervised online ensemble which excludes weak classifiers in its decision-making process. SSE-PBS is also compared to the Bagging variants including online bagging (Oza), online bagging Adwin (OzaAdw), leveraging bagging (Lev), and Adaptive Random Forest (ARF). The difference between AUE-LL and AUE2 is that AUE-LL uses a small portion of labeled data. SPASC and SCBELS are cluster-based ensemble classifiers. Sco-F algorithm is chosen as a chunk-based semi-supervised ensemble algorithm that uses a concept drift detector. STDS is selected as a semi-supervised single classifier with a concept drift detector. COMPOSE is employed as a single semi-supervised classifier under EVL assumption. WNB-CD is used as a single supervised classifier that employs an implicit approach for concept drift handling. Additionally, Na've Bayes (NB) and Hoeffding Tree (HT) algorithms are selected as algorithms without any drift handling mechanism. Finally, SSE-CBS is a variation of SSE-PBS, which uses a confidence score as the selection metric instead of performance.

To make a fair comparison, we set the same parameter values for all algorithms in this study. For ensemble approaches, the number of component classifiers is set to ten. The accuracy is measured using the chunk evaluation method, which works similar to the test-then-train paradigm used in [36], but employs chunks instead of single examples. This approach reads incoming instances without processing them until they build a data chunk of size $D$. The chunk size is set to $D = 500$ for all datasets. For each dataset, 20 percent of the data is kept as the test set, and the rest is used as training data. In each chunk, a fixed portion of examples is labeled. In order to evaluate the proposed algorithm, we employ 2%, 5%, and 10% of the labeled data.

The number of bins is set to 5 in all experiments. Furthermore, in SSE-PBS, the parameters $a_1, a_2$, and $\epsilon$ are set to 0.75, 0.25, and 0.0001, respectively. Furthermore, we select $p = 20\%$ of unlabeled examples with confidence predictions above a threshold $T = 90\%$ in each chunk, and they are added to the original labeled data for updating existing classifiers in SCo-F, STDS, and SSE-CBS algorithms.

### 4.3. Statistical test

To verify if the results are statistically significant, we perform a non-parametric test using the methodology presented in [16]. For the statistical test, we employ the Friedman test with $\alpha = 0.05$ and the null hypothesis (when there is no statistical difference between the given algorithms). This statistical test ranks the algorithms based on their performance for each dataset separately. The best algorithm gets the rank of 1, the second-best rank 2, and so on, as shown in Table 10.

Let $r_i^j$ be the rank of the j-th of $k$ algorithms on the i-th of $N$ datasets. The Friedman test computes the average ranks of the algorithms as $R_j = \frac{1}{N}\sum_i r_i^j$. The Friedman's statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{14}$$

is distributed according to $\chi_F^2$ with $k-1$ degrees of freedom. Iman and Davenport [33] showed that Friedman's $\chi_F^2$ is undesirably conservative and derived a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \tag{15}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

If the null hypothesis of Friedman's test is rejected, then we proceed with Holm's post hoc test [31] to identify the differences. The Holm's method sequentially checks the hypotheses ordered by their performance. The ordered p-values are denoted by $p_1 < p_2 < \ldots < p_{k-1}$. Then each $p_i$ is compared to $\frac{\alpha}{k-i}$. If $p_i$ is less than $\frac{\alpha}{k-i}$, then the corresponding hypothesis is rejected. The test statistic for comparing the i-th and j-th classifiers is

$$Z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}}, \tag{16}$$

which is called Z-score. The Z-score is used to find the corresponding probability from the normal distribution and compare it to the corresponding value of $\alpha$.

### 4.4. Datasets

In the experiments, we evaluate the performance of the algorithms on 22 standard data stream benchmarks, including 4 real-world [23] and 18 synthetic, which are generated in MOA [5] with different characteristics shown in Table 1.

### 4.5. Parameter settings

In order to find the tuned bin numbers in the experiments, we use different bin numbers for the $SEA_S$ data streams. Fig. 4 shows the results of this experiment. As shown, when we set the bin numbers to 5, then SSE-PBS with HT as the base learner achieves the best classification performance. Therefore, we set the bin numbers to 5 in the rest of the experiments in this study, which means

**Table 1**
Specification of the used datasets.

| Type | DataSets | #Inst | #Attr | #Cls | %Noise | #Drift | Drift Type |
|---|---|---|---|---|---|---|---|
| Synthetic | $SEA_S$ | 1000000 | 3 | 2 | 10% | 3 | Sudden |
| | $SEA_F$ | 2000000 | 3 | 2 | 10% | 9 | Sudden |
| | $Hyp_S$ | 1000000 | 10 | 2 | 5% | 1 | Gradual |
| | $Hyp_F$ | 1000000 | 10 | 2 | 5% | 1 | Gradual |
| | $RBF_B$ | 1000000 | 20 | 4 | 0% | 2 | Blips |
| | $RBF_{GR}$ | 1000000 | 20 | 4 | 0% | 4 | Gradual Recurring |
| | $RBF_{ND}$ | 1000000 | 20 | 4 | 0% | 0 | No drift |
| | $LED_{GS}$ | 1000000 | 24 | 10 | 10% | 2 | Gradual Sudden |
| | $LED_{ND}$ | 10000000 | 24 | 10 | 20% | 0 | No drift |
| | $Rtree_R$ | 1000000 | 10 | 2 | 5% | 5 | Recurrent |
| | $Rtree_G$ | 1000000 | 10 | 2 | 5% | 1 | Gradual |
| | $Rtree_{FG}$ | 1000000 | 10 | 2 | 5% | 1 | Fast Gradual |
| | $Rtree_{ND}$ | 1000000 | 10 | 2 | 5% | 0 | No drift |
| | $Agrawal_S$ | 1000000 | 9 | 2 | 10% | 4 | Sudden |
| | $Agrawal_G$ | 1000000 | 9 | 2 | 10% | 1 | Gradual |
| | $Wave_G$ | 1000000 | 40 | 3 | 10% | 1 | Gradual |
| | $Wave_{ND}$ | 1000000 | 40 | 3 | 10% | 0 | No drift |
| | $Stagger_G$ | 1000000 | 3 | 2 | 10% | 1 | Gradual |
| Real | *Elec* | 45312 | 8 | 2 | – | – | Unknown |
| | *Cov* | 581012 | 54 | 7 | – | – | Unknown |
| | *Poker* | 1000000 | 10 | 10 | – | – | Unknown |
| | *Airlines* | 539000 | 7 | 2 | – | – | Unknown |

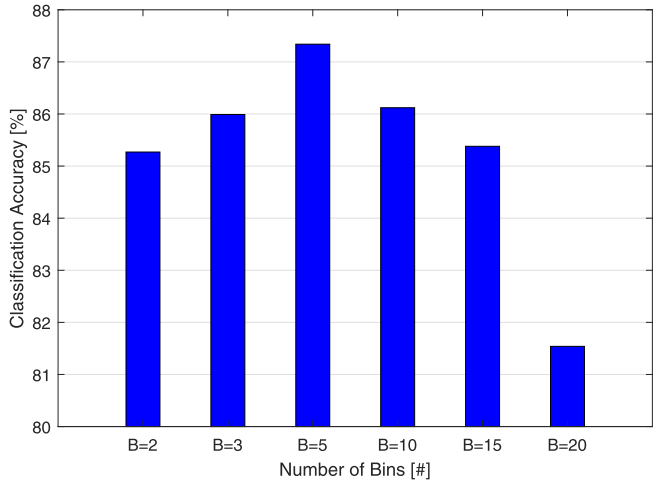



**Fig. 4.** Sensitivity of the proposed algorithm to bin numbers.

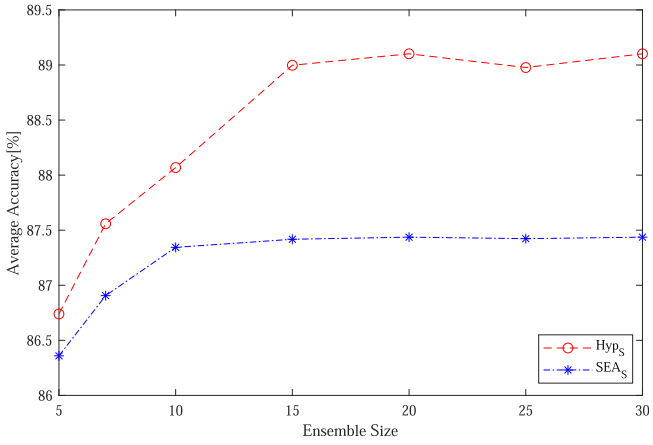



**Fig. 5.** Impact of the ensemble size on accuracy when the base learner is HT.

20% unlabeled examples are added to the labeled data in each chunk based on the proposed performance-based selection metric.

In the second experiment, we tune the number of the component classifiers in the proposed ensemble approach. Fig. 5 depicts the accuracy of the proposed algorithm when the number of base learners varies from 5 to 30 on $SEA_S$ and $Hyp_S$ datasets. Based on these results, we use 10 component classifiers in our experiments.

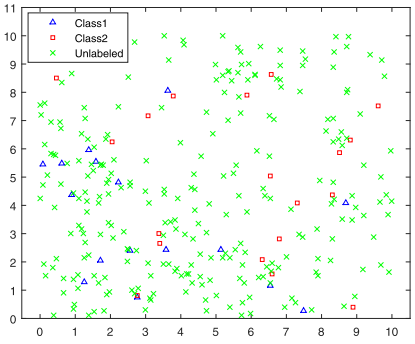
### 4.6. SSE-PBS performance evaluation

In this experiment, we randomly sample 300 data points of the $SEA_S$ data streams to evaluate SSE-PBS algorithm. We use only 10% labeled data and employ HT as the base learner. The error rate is 12.66%, which is quite acceptable considering 10% noise in the dataset. As shown in Fig. 6b, SSE-PBS algorithm exploits knowledge from the unlabeled examples and improves the classification performance. The classification performance of SSE-PBS is indeed similar to the original fully labeled data, see 6.c. This observation indicates that the SSE-PBS algorithm can properly exploit knowledge from the unlabeled instances and improves the classification performance effectively.

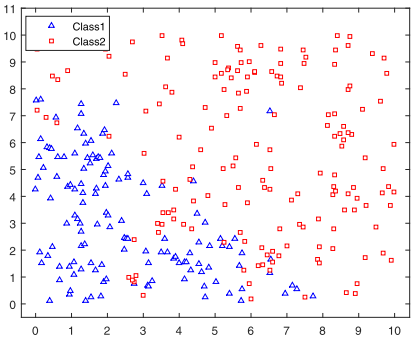
## 5. Results

In this section, we report the results of the various experiments to compare the classification performance of SSE-PBS to the state-of-the-art semi-supervised methods and supervised algorithms employing different datasets: synthetic and real-world. The experiments show the impact of employing unlabeled data in order to improve the classification performance using different base learners.

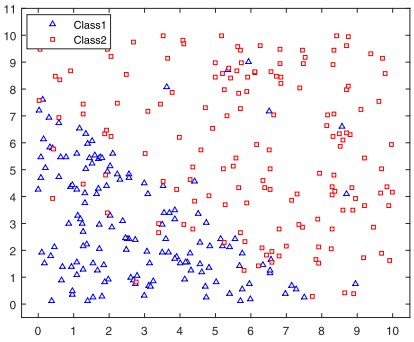
Our experiments include the performance comparison and rank between SSE-PBS, supervised, and semi-supervised algorithms when the percentage of labeled data is 2%, 5%, and 10% using two incremental classifiers as the base learners, HT and NB. The results are shown in Tables 2–9. In all tables, the average ranks of the classifiers according to the Friedman's test are reported. The results indicate that using HT in comparison to NB as the base learner in SSE-PBS generally leads to higher classification performance.




(a) Labeled and Unlabeled data.




(b) Newly-labeled data.




(c) Original fully labeled data.

**Fig. 6.** Classification performance of SSE-PBS on $SEA_S$ dataset using HT as the base learner.

Table 10 shows the rank of each semi-supervised algorithm according to the Friedman's test when their base learner is HT and 2% labeled data is available in each chunk. Tables 10–16 present the pairwise comparison between SSE-PBS and state-of-the-art methods on all datasets based on the outcomes of the Holm's statistical test. Finally, Figs. 11a–d present the results over all processed instances for four selected datasets.

### 5.1. SSE-PBS with HT as the base learner

In the first experiment, we compare the proposed approach to the supervised methods to show how SSE-PBS can effectively exploit information from the unlabeled data. Table 2 shows the classification performance of SSE-PBS and the state-of-the-art supervised algorithms with HT as the base learner. As shown, Table 2 includes three sections: supervised with limited labeled data (LL), semi-supervised with limited labeled and unlabeled data (LU), and supervised with fully labeled data(FL). The first two columns show the results for the supervised classifiers HT and AUE-LL using 10% labeled data. The next three columns show the results of SSE-PBS using labeled and unlabeled data where the labeled data in each chunk varies from 2%, 5%, and 10%, respectively. Finally, the last seven columns show the classification performance of the supervised ensemble classifiers including Oza, OzaAdw, Lev, ARF, KUE, OAbst, and AUE2 with fully labeled data when their base learner is Hoeffding Tree.

The results show that the difference between supervised methods and the SSE-PBS algorithm decreases while the number of labeled examples increases. We further observe that SSE-PBS outperforms all supervised algorithms and significantly improves the classification performance nearly on all datasets. We have boldfaced the best results of each part separately.

Additionally, in Fig. 7, we present box plots that show the average classification accuracy of the algorithms over all datasets when their base learner is HT. As shown, SSE-PBS gives the best classification performance.

In the second experiment, we compare our proposed algorithms to state-of-the-art semi-supervised methods when there is only 10% labeled data. Table 3 shows the classification performance of the methods. The results indicate that SSE-PBS and SSE-CBS give higher classification performance than the other semi-supervised algorithms on 16 out of 22 datasets and the best average rank and the highest average accuracy. The classification performance of SSE-PBS is close to the classifiers that are trained with fully labeled for nearly all datasets.

#### 5.1.1. Sensitivity to the number of labeled data

To show the sensitivity of SSE-PBS to the number of labeled examples, we run a set of experiments with 2% and 5% labeled data in each chunk.

Tables 4 and 5 show the comparison results between SSE-PBS and state-of-the-art semi-supervised algorithms when their base learner is HT in which the proportions of labeled data is 2% and 5%, respectively.

Despite the most approaches in semi-supervised data streams such as Sco-F and STDS, which employ the confidence score as their selection metric, the proposed algorithm does not rely on the classification confidence as the selection criterion. Instead, it uses a Performance-Based Selection algorithm (PBS) to evaluate the ability of candidate data points to improve classification performance. In particular, the only unlabeled samples that help to improve classification accuracy are selected for retraining the classification model because confidence score is a criterion that measures the accuracy of a given sample's classification, rather than

**Table 2**
Accuracy of SSE-PBS vs. supervised classifiers-setting (Base learner = HT).

| Datasets | Supervised Algorithms LL | | Semi-Supervised LU | | | Supervised Algorithms FL | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HT | AUE-LL | SSE-PBS$_{2\%}$ | SSE-PBS$_{5\%}$ | SSE-PBS$_{10\%}$ | Oza | OzaAdw | Lev | ARF | KUE | OAbst. | AUE2 |
| $SEA_S$ | 77.42 | **82.81** | 82.89 | 85.53 | **87.34** | 88.17 | 88.03 | 87.09 | 88.93 | 88.98 | **91.66** | 89.19 |
| $SEA_F$ | 76.98 | **82.81** | 83.13 | 85.57 | **87.22** | 88.37 | 87.31 | 86.68 | 89.35 | 88.35 | **91.22** | 88.72 |
| $Hyp_S$ | 77.31 | **84.93** | 81.28 | 86.64 | **88.06** | 83.36 | 88.61 | 85.36 | 85.38 | 86.4 | **89.12** | 88.43 |
| $Hyp_F$ | 76.42 | **83.12** | 80.97 | 85.3 | **87.48** | 83.69 | 89.28 | 89.51 | 88.09 | 88.83 | **91.38** | 89.46 |
| $RBF_B$ | 68.32 | **80.31** | 81.72 | 84.37 | **85.95** | 93.08 | 95.29 | 95.28 | **98.08** | 94.68 | 93.7 | 94.77 |
| $RBF_{GR}$ | 64.54 | **79.14** | 80.81 | 83.39 | **86.21** | 92.56 | 94.81 | 94.74 | **97.38** | 97.26 | 93.18 | 94.43 |
| $RBF_{ND}$ | 69.82 | **78.39** | 70.33 | 75.35 | **79.06** | 91.37 | 89.41 | 92.24 | 94.55 | **94.75** | 92.54 | 93.33 |
| $LED_{GS}$ | 61.25 | **62.9** | 60.12 | 63.75 | **65.26** | 67.62 | 53.15 | 66.74 | 52.59 | 53.17 | **71.16** | 67.58 |
| $LED_{ND}$ | 47.32 | **48.41** | 45.64 | 48.57 | **50.46** | 51.23 | 51.21 | 50.64 | 51.11 | **51.96** | 51.47 | 51.26 |
| $Rtree_R$ | 60.22 | **68.29** | 77.02 | 79.56 | **81.71** | 80.19 | **83.71** | 86.37 | 80.07 | 81.89 | 79.58 | 83.69 |
| $Rtree_G$ | 69.61 | **77.04** | 85.83 | 88.73 | **90.97** | 86.36 | 91.93 | 91.45 | 53.94 | 92.41 | 90.67 | **93.49** |
| $Rtree_{FG}$ | 60.77 | **68.92** | 74.71 | 76.64 | **78.02** | 79.65 | 81.26 | **84.77** | 79.61 | 75.88 | 81.53 | 80.79 |
| $Rtree_{ND}$ | 73.92 | **80.11** | 86.22 | 89.18 | **90.27** | 86.57 | 95.73 | **97.74** | 89.54 | 95.35 | 91.32 | 95.34 |
| $Agrawal_S$ | 71.94 | **80.75** | 84.17 | 86.92 | **88.31** | 82.91 | 88.66 | 86.22 | 87.04 | 89.35 | 90.16 | **90.31** |
| $Agrawal_G$ | 68.76 | **76.83** | 80.96 | 82.90 | **85.43** | 82.66 | 86.42 | 84.21 | 84.51 | **89.68** | 89.02 | 88.58 |
| $Wave_G$ | 68.39 | **77.51** | 78.22 | 80.25 | **82.05** | 81.45 | **85.51** | 85.01 | 83.42 | 85.15 | 83.97 | 85.27 |
| $Wave_{ND}$ | 69.15 | **76.67** | 79.34 | 81.33 | **83.37** | 81.57 | **85.52** | 84.97 | 83.6 | 85.29 | 83.88 | 85.33 |
| $Stagger_G$ | 61.08 | **65.89** | 67.10 | 69.83 | **72.24** | 74.52 | 75.11 | 74.88 | 74.81 | 75.09 | **75.86** | 75.03 |
| *Elec* | 61.31 | **70.09** | 70.70 | 74.04 | **76.79** | 77.34 | 76.28 | 76.08 | 76.86 | 76.9 | **80.48** | 77.32 |
| *Cov* | 61.62 | **67.93** | 71.05 | 75.94 | **79.93** | 80.4 | 84.26 | **88.26** | 86.24 | 87.53 | 81.34 | 85.2 |
| *Poker* | 57.26 | **60.77** | 60.67 | 63.87 | **65.12** | 61.13 | 63.17 | 87.25 | 68.98 | **90.37** | 62.03 | 66.1 |
| *Airlines* | 56.43 | **58.89** | 62.32 | 63.58 | **65.21** | 66.39 | 62.38 | 63.1 | 62.43 | 62.56 | 65.82 | **67.37** |
| **Avg. Acc.** | 66.36 | **73.30** | 74.69 | 77.78 | **79.84** | 80.03 | 81.63 | **83.57** | 79.84 | 83.27 | 82.78 | 83.23 |
| **Avg. Rank** | 11.73 | **10.45** | 10.14 | 8.05 | **6.36** | 6.55 | 4.36 | 4.59 | 5.55 | 3.77 | 3.5 | **2.95** |

**Table 3**
Accuracy of semi-supervised classifiers (base learner = HT, Labeled data = 10%).

| Datasets | Semi-Supervised Algorithms | | | | | | |
|---|---|---|---|---|---|---|---|
| | COMPOSE | STDS | SPASC | SCBELS | Sco-F | SSE-CBS | SSE-PBS |
| $SEA_S$ | 59.73 | 84.35 | 79.9 | **87.37** | 86.96 | 86.09 | 87.34 |
| $SEA_F$ | 59.98 | 83.65 | 79.24 | 85.23 | 87.04 | 85.88 | **87.22** |
| $Hyp_S$ | 58.28 | 84.74 | 69.05 | 85.74 | 78.02 | 87.13 | **88.06** |
| $Hyp_F$ | 58.19 | 84.89 | 68.23 | 86.11 | 79.97 | 86.53 | **87.48** |
| $RBF_B$ | 60.82 | 83.89 | 79.33 | 80.41 | 83.22 | 84.57 | **85.95** |
| $RBF_{GR}$ | 60.56 | 83.63 | 78.02 | 81.17 | 83.08 | 85.28 | **86.21** |
| $RBF_{ND}$ | 58.34 | 73.52 | 72.52 | 76.79 | 71.94 | 77.43 | **79.06** |
| $LED_{GS}$ | 46.62 | 62.85 | 55.21 | 59.77 | 52.83 | 64.91 | **65.26** |
| $LED_{ND}$ | 40.55 | 48.02 | 46.93 | 48.61 | **55.68** | 50.78 | 50.46 |
| $Rtree_R$ | 61.22 | 71.67 | 70.21 | 73.02 | 74.78 | **81.88** | 81.71 |
| $Rtree_G$ | 66.16 | 79.79 | 77.08 | 79.56 | 80.15 | 90.83 | **90.97** |
| $Rtree_{FG}$ | 60.65 | 74.22 | 67.81 | 72.74 | 73.09 | **78.15** | 78.02 |
| $Rtree_{ND}$ | 68.72 | 82.72 | 80.63 | 81.29 | 83.38 | 88.39 | **90.27** |
| $Agrawal_S$ | 68.91 | 84.21 | 79.88 | 84.35 | 85.86 | 88.08 | **88.31** |
| $Agrawal_G$ | 61.73 | 79.87 | 78.92 | 79.61 | **86.11** | 84.22 | 85.43 |
| $Wave_G$ | 60.79 | 78.65 | 77.11 | 80.16 | 80.37 | **82.58** | 82.05 |
| $Wave_{ND}$ | 61.68 | 79.23 | 79.92 | **84.17** | 79.42 | 81.71 | 83.37 |
| $Stagger_G$ | 60.91 | 67.12 | **72.28** | 68.33 | 68.14 | 70.44 | 72.24 |
| *Elec* | 60.15 | 72.81 | 57.11 | 65.45 | 62.36 | 75.24 | **76.79** |
| *Cov* | 51.82 | 74.44 | 67.82 | 69.26 | 66.05 | 78.21 | **79.93** |
| *Poker* | 51.09 | 61.32 | 59.87 | 63.01 | 65.74 | **65.85** | 65.12 |
| *Airlines* | 55.64 | 61.13 | 59.71 | 61.98 | **66.42** | 63.44 | 65.21 |
| **Avg. Acc.** | 58.75 | 75.31 | 70.76 | 75.19 | 75.03 | 78.98 | **79.84** |
| **Avg.Rank** | 6.95 | 4.27 | 5.59 | 3.86 | 3.64 | 2.14 | **1.5** |

a criterion that assesses the potential of a sample's participation in training for improving a model. Therefore, using confidence score as the selection criterion may lead to inaccurate estimation of actual distribution and result in performance degradation, even if the selected samples are classified correctly. Our experiments support the proposed idea and show the impact of the proposed approach.

In the first experiment, we only use 2% labeled data to show the ability of the proposed algorithms to handle the semi-supervised stream classification tasks. Table 4 gives the average classification performance of the state-of-the-art approaches when their base learner is HT. As shown, SSE-PBS and SSE-CBS improve the classification performance on 14 out of 22 datasets. We further observe that the proposed approaches give the best average ranks and the highest average accuracies.

In the second experiment, 5% labeled data is used to handle the semi-supervised stream classification tasks. Table 5 shows classification performance of the semi-supervised methods. The results

**Table 4**
Accuracy of semi-supervised classifiers (base learner = HT, Labeled data = 2%).

| Datasets | Semi-Supervised Algorithms | | | | | | |
|---|---|---|---|---|---|---|---|
| | COMPOSE | STDS | SPASC | SCBELS | Sco-F | SSE-CBS | SSE-PBS |
| $SEA_S$ | 54.11 | 79.58 | 76.29 | 82.88 | 82.17 | 82.46 | **82.89** |
| $SEA_F$ | 54.05 | 77.11 | 74.64 | 79.46 | **85.71** | 82.72 | 83.13 |
| $Hyp_S$ | 52.77 | 79.73 | 64.74 | **83.49** | 58.16 | 81.14 | 81.28 |
| $Hyp_F$ | 53.65 | 78.65 | 64.2 | **81.14** | 58.61 | 80.36 | 80.97 |
| $RBF_B$ | 54.86 | 74.63 | 75.2 | 77.01 | 69.32 | 80.63 | **81.72** |
| $RBF_{GR}$ | 54.61 | 76.28 | 73.4 | 78.11 | 65.03 | 80.01 | **80.81** |
| $RBF_{ND}$ | 52.02 | 64.19 | 69.5 | **70.53** | 57.01 | 69.46 | 70.33 |
| $LED_{GS}$ | 40.47 | 57.42 | 51.77 | 55.11 | 51.44 | 58.97 | **60.12** |
| $LED_{ND}$ | 36.66 | 43.13 | 42.55 | 44.8 | **52.85** | 46.67 | 45.64 |
| $Rtree_R$ | 58.2 | 67.67 | 64.84 | 68.48 | 70.12 | **77.29** | 77.02 |
| $Rtree_G$ | 62.58 | 75.65 | 74.17 | 73.74 | 76.76 | **86.21** | 85.84 |
| $Rtree_{FG}$ | 56.51 | 68.11 | 64.32 | 66.25 | 67.83 | 72.12 | **74.72** |
| $Rtree_{ND}$ | 65.31 | 78.94 | 76.02 | 76.18 | 77.69 | 83.96 | **86.23** |
| $Agrawal_S$ | 64.25 | 76.97 | 76.35 | 75.54 | 80.41 | 83.4 | **84.17** |
| $Agrawal_G$ | 58.08 | 73.29 | 74.13 | 74.47 | 75.47 | **81.08** | 80.96 |
| $Wave_G$ | 55.42 | 72.46 | 73.15 | 74.44 | 77.46 | 76.01 | **78.22** |
| $Wave_{ND}$ | 57.21 | 72.72 | 74.56 | 78.28 | **78.66** | 76.93 | 78.34 |
| $Stagger_G$ | 55.36 | 62.42 | **67.48** | 62.13 | 62.81 | 65.64 | 67.1 |
| *Elec* | 54.71 | 67.68 | 50.38 | 59.69 | 61.11 | 69.87 | **70.70** |
| *Cov* | 44.19 | 68.82 | 64.11 | 65.6 | 65.38 | 69.36 | **71.05** |
| *Poker* | 45.92 | 56.17 | 55.79 | 57.3 | **62.58** | 60.95 | 60.67 |
| *Airlines* | 51.64 | 57.33 | 56.61 | 56.66 | 60.64 | **61.51** | 61.32 |
| **Avg. Acc.** | 53.75 | 69.5 | 66.55 | 70.06 | 68.06 | 73.94 | **74.69** |
| **Avg.Rank** | 6.95 | 4.41 | 5.18 | 3.82 | 3.73 | 2.27 | **1.64** |

**Table 5**
Accuracy of semi-supervised classifiers (base learner = HT, Labeled data = 5%).

| Datasets | Semi-Supervised Algorithms | | | | | | |
|---|---|---|---|---|---|---|---|
| | COMPOSE | STDS | SPASC | SCBELS | Sco-F | SSE-CBS | SSE-PBS |
| $SEA_S$ | 55.82 | 80.54 | 77.39 | 85.09 | **86.29** | 84.65 | 85.53 |
| $SEA_F$ | 55.91 | 79.21 | 77.41 | 82.23 | **86.73** | 84.23 | 85.57 |
| $Hyp_S$ | 54.27 | 81.64 | 66.28 | 84.64 | 72.69 | 85.52 | **86.64** |
| $Hyp_F$ | 55.53 | 81.19 | 65.57 | 84.19 | 74.84 | 84.13 | **85.30** |
| $RBF_B$ | 56.38 | 79.11 | 77.61 | 79.13 | 81.24 | 82.09 | **84.37** |
| $RBF_{GR}$ | 56.63 | 79.89 | 75.08 | 79.31 | 81.35 | 80.94 | **83.39** |
| $RBF_{ND}$ | 54.78 | 70.82 | 70.61 | 74.86 | 70.89 | 72.21 | **75.35** |
| $LED_{GS}$ | 42.66 | 60.62 | 53.2 | 56.76 | 52.3 | 61.78 | **63.75** |
| $LED_{ND}$ | 37.03 | 44.13 | 43.97 | 46.05 | **55.59** | 48.88 | 48.57 |
| $Rtree_R$ | 59.81 | 69.69 | 67.59 | 70.59 | 72.09 | **79.81** | 79.56 |
| $Rtree_G$ | 64.83 | 78.11 | 75.86 | 75.82 | 80.07 | **88.76** | 88.73 |
| $Rtree_{FG}$ | 59.32 | 71.35 | 66.76 | 68.91 | 70.09 | 74.75 | **76.64** |
| $Rtree_{ND}$ | 67.45 | 81.33 | 79.59 | 79.48 | 80.57 | 86.95 | **89.19** |
| $Agrawal_S$ | 66.54 | 81.73 | 78.38 | 81.25 | 82.74 | **87.01** | 86.92 |
| $Agrawal_G$ | 59.52 | 75.81 | 76.43 | 76.76 | 77.73 | **82.92** | 82.9 |
| $Wave_G$ | 58.03 | 76.21 | 75.20 | 77.58 | 78.04 | 78.53 | **80.25** |
| $Wave_{ND}$ | 59.16 | 76.26 | 77.22 | **81.82** | 79.47 | 79.14 | 81.33 |
| $Stagger_G$ | 58.12 | 64.91 | **69.86** | 64.76 | 65.6 | 68.22 | 69.83 |
| *Elec* | 56.68 | 68.87 | 55.09 | 63.26 | 62.07 | 72.55 | **74.04** |
| *Cov* | 45.47 | 69.23 | 66.22 | 68.05 | 65.79 | 75.12 | **75.94** |
| *Poker* | 48.21 | 59.12 | 57.49 | 60.64 | **64.39** | 63.93 | 63.87 |
| *Airlines* | 53.82 | 59.97 | 57.97 | 59.32 | **64.24** | 63.14 | 63.58 |
| **Avg. Acc.** | 55.73 | 72.26 | 68.67 | 72.75 | 72.95 | 76.6 | **77.78** |
| **Avg. Rank** | 6.95 | 4.36 | 5.55 | 4.05 | 3.23 | 2.27 | **1.59** |

show that SSE-PBS and SSE-CBS perform better than other semi-supervised algorithms on 15 out of 22 datasets. It is also shown that SSE-PBS gives the best average classification accuracy.

Furthermore, in Fig. 8, we present three box plots that show average classification accuracy over all datasets with different proportions of labeled data using HT as the base learner. As shown, SSE-PBS outperforms state-of-the-art semi-supervised algorithms.

### 5.2. SSE-PBS with NB as the base learner

As mentioned earlier, the proposed algorithm is a framework that can use any incremental classifier as its base learner, while ARF and Sco-F can use only HT as the base learner. Therefore, in this experiment, we use Naive Bayes as the base learner to evaluate the proposed algorithm. Table 6 shows the performance comparison between SSE-PBS and state-of-the-art supervised algorithms with Naive Bayes (NB) as the base learner. This table includes three sections: supervised with limited labeled data (LL), semi-supervised with limited labeled and unlabeled data (LU), and supervised with fully labeled (FL). The first three columns show the results for supervised classifiers including NB, WNB-CD, and AUE-LL using 10% labeled data. The next three columns show the results of SSE-PBS using labeled and unlabeled data where the labeled data in each chunk is 2%, 5%, and 10%, respectively. Finally,

**Table 6**
Accuracy of SSE-PBS vs. supervised classifiers (base learner = NB).

| Datasets | Supervised Algorithms LL | | | Semi-Supervised LU | | | Supervised Algorithms FL | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | NB | WNB-CD | AUE-LL | SSE-PBS$_{2\%}$ | SSE-PBS$_{5\%}$ | SSE-PBS$_{10\%}$ | Oza | OzaAdw | Lev | OAbst. | AUE2 |
| $SEA_S$ | 74.02 | 78.11 | **81.93** | 83.21 | 85.21 | **87.53** | 83.79 | 86.19 | 86.58 | **89.09** | 88.87 |
| $SEA_F$ | 74.78 | 77.98 | **82.61** | 80.77 | 84.87 | **85.03** | 85.37 | 86.73 | 86.91 | **89.56** | 87.96 |
| $Hyp_S$ | 75.65 | 78.24 | **83.56** | 80.89 | 85.65 | **87.91** | 77.62 | 89.45 | **89.92** | 88.82 | 89.04 |
| $Hyp_F$ | 74.31 | 77.87 | **82.03** | 81.23 | 85.09 | **87.17** | 77.13 | 88.53 | 88.79 | **91.02** | 88.96 |
| $RBF_B$ | 65.43 | 75.09 | **80.93** | 77.97 | 80.11 | **83.13** | 60.72 | 78.82 | 79.7 | 90.92 | **93.56** |
| $RBF_{GR}$ | 60.52 | 70.65 | **79.53** | 78.44 | 80.76 | **84.78** | 58.24 | 72.57 | 75.59 | 90.62 | **93.11** |
| $RBF_{ND}$ | 68.67 | 70.32 | **75.14** | 77.98 | 79.94 | **81.53** | 71.49 | 71.97 | 71.96 | 89.95 | **92.06** |
| $LED_{GS}$ | 59.25 | **61.53** | 60.95 | 60.02 | 63.57 | **66.32** | 54.48 | 54.53 | 54.58 | **70.88** | 68.21 |
| $LED_{ND}$ | 46.41 | 47.09 | **49.03** | 42.75 | 48.09 | **50.88** | 51.19 | 51.24 | 51.21 | **55.01** | 52.32 |
| $Rtree_R$ | 61.09 | 65.48 | **67.05** | 76.2 | 78.08 | **79.89** | 79.52 | 80.56 | **82.18** | 79.03 | 82.17 |
| $Rtree_G$ | 71.11 | 73.55 | **75.24** | 84.11 | 86.62 | **89.91** | 86.6 | 88.21 | 87.85 | 89.05 | **91.38** |
| $Rtree_{FG}$ | 65.74 | 68.25 | **68.56** | 73.74 | 75.36 | **77.09** | 78.48 | 78.81 | **79.97** | 79.80 | 79.55 |
| $Rtree_{ND}$ | 74.23 | 77.63 | **78.87** | 84.32 | 87.19 | **88.84** | 86.07 | 92.3 | **95.05** | 87.66 | 94.28 |
| $Agrawal_S$ | 73.28 | 78.01 | **79.52** | 82.62 | 85.05 | **87.09** | 84.49 | 86.51 | 84.17 | **87.92** | 87.25 |
| $Agrawal_G$ | 71.44 | 74.29 | **75.65** | 80.18 | 82.74 | **84.05** | 81.42 | 83.68 | 82.29 | 85.1 | **86.47** |
| $Wave_G$ | 70.22 | 72.62 | **76.31** | 76.24 | 78.12 | **80.20** | 82.69 | 83.01 | **83.84** | 81.02 | 83.80 |
| $Wave_{ND}$ | 73.09 | 74.17 | **76.52** | 79.13 | 80.63 | **82.17** | 81.08 | 82.96 | 81.85 | 81.75 | **84.88** |
| $Stagger_G$ | 63.56 | 65.15 | **65.53** | 65.71 | 68.83 | **70.75** | 71.48 | 72.59 | 72.31 | **72.61** | 72.18 |
| *Elec* | 59.11 | 64.21 | **70.11** | 69.12 | 72.43 | **75.58** | 68.36 | 68.05 | 67.92 | **77.00** | 76.24 |
| *Cov* | 60.35 | 63.57 | **66.43** | 65.24 | 70.11 | **73.52** | 68.83 | 81.38 | 81.44 | 78.25 | **84.12** |
| *Poker* | 56.76 | 58.78 | **59.33** | 59.66 | 62.87 | **65.09** | 60.59 | 61.45 | 61.43 | 61.13 | **67.25** |
| *Airlines* | 56.94 | 58.21 | **59.03** | 59.49 | 61.15 | **63.71** | 64.55 | **66.35** | 63.66 | 64.68 | 65.80 |
| **Avg. Acc.** | 66.18 | 69.58 | **72.45** | 73.59 | 76.48 | **78.73** | 73.37 | 77.54 | 77.69 | 80.95 | **82.25** |
| **Average Rank** | 10.73 | 9.55 | **7.73** | 7.91 | 5.77 | **4.05** | 7.09 | 4.32 | 4.36 | 2.68 | **1.82** |

**Table 7**
Accuracy of semi-supervised classifiers (base learner = NB, Labeled data = 10%).

| Datasets | Semi-Supervised Algorithms | | | | | |
|---|---|---|---|---|---|---|
| | COMPOSE | SPASC | SCBELS | STDS | SSE-CBS | SSE-PBS |
| $SEA_S$ | 58.89 | 79.9 | **87.37** | 84.18 | 86.21 | 87.53 |
| $SEA_F$ | 59.13 | 79.24 | **85.23** | 82.07 | 84.88 | 85.03 |
| $Hyp_S$ | 57.88 | 69.05 | 85.74 | 84.51 | 86.56 | **87.91** |
| $Hyp_F$ | 58.05 | 68.23 | 86.11 | 83.77 | 85.92 | **87.17** |
| $RBF_B$ | 60.34 | 79.33 | 80.41 | 80.42 | 82.43 | **83.13** |
| $RBF_{GR}$ | 59.65 | 78.02 | 81.17 | 80.19 | 83.1 | **84.78** |
| $RBF_{ND}$ | 58.11 | 72.52 | 76.79 | 73.34 | 80.67 | **81.53** |
| $LED_{GS}$ | 45.78 | 55.21 | 59.77 | 62.39 | 65.27 | **66.32** |
| $LED_{ND}$ | 41.51 | 46.93 | 48.61 | 47.45 | 50.32 | **50.88** |
| $Rtree_R$ | 61.53 | 70.21 | 73.02 | 72.09 | **80.01** | 79.89 |
| $Rtree_G$ | 64.36 | 77.08 | 79.56 | 78.97 | **90.13** | 89.91 |
| $Rtree_{FG}$ | 60.43 | 67.81 | 72.74 | 71.72 | 74.49 | **77.09** |
| $Rtree_{ND}$ | 68.13 | 80.63 | 81.29 | 80.67 | **89.04** | 88.84 |
| $Agrawal_S$ | 68.31 | 79.88 | 84.35 | 83.32 | **87.15** | 87.09 |
| $Agrawal_G$ | 61.52 | 78.92 | 79.61 | 78.77 | 81.12 | **84.05** |
| $Wave_G$ | 60.88 | 77.11 | 80.16 | 76.81 | 79.42 | **80.20** |
| $Wave_{ND}$ | 62.88 | 79.92 | **84.27** | 78.03 | 80.63 | 82.17 |
| $Stagger_G$ | 59.61 | **72.28** | 68.33 | 66.16 | 68.67 | 70.75 |
| *Elec* | 60.12 | 57.11 | 65.45 | 72.32 | 74.76 | **75.58** |
| *Cov* | 50.43 | 67.82 | 69.26 | 72.55 | 71.56 | **73.52** |
| *Poker* | 52.66 | 59.87 | 63.01 | 61.01 | 64.71 | **65.09** |
| *Airlines* | 55.47 | 59.71 | 61.98 | 59.88 | **63.82** | 63.71 |
| **Avg. Acc.** | 58.44 | 70.76 | 75.19 | 74.12 | 77.77 | **78.74** |
| **Avg. Rank** | 5.95 | 4.73 | 2.91 | 3.95 | 2.09 | **1.05** |

the last five columns show the supervised ensemble classifiers including Oza, OzaAdw, Lev, OAbst, and AUE2 with fully labeled data when their base learner is Naive Bayes. The results show that the difference between the supervised methods and SSE-PBS decreases while the number of labeled examples increases. We have boldfaced the best results of each part separately.

As shown in Table 6, SSE-PBS outperforms the supervised methods nearly on all datasets. It is also observed that SSE-PBS significantly improves the classification performance. Furthermore, we compare the performance of SSE-PBS to the models that have been trained on fully-labeled data. These results indicate that the performance of SSE-PBS is close to the fully-labeled trained models.

Additionally, in Fig. 9, we present box plots that show the average classification accuracy of the algorithms over all datasets when their base learner is NB. As shown, SSE-PBS gives acceptable results.

In the second experiment, the performance of SSE-PBS is compared to the other semi-supervised methods when there is only 10% labeled data. Table 7 shows the classification performance of the compared methods when the base learner is NB. The results

**Table 8**
Accuracy of semi-supervised classifiers (base learner = NB, Labeled data = 2%).

| Datasets | Semi-Supervised Algorithms | | | | | |
|---|---|---|---|---|---|---|
| | COMPOSE | SPASC | SCBELS | STDS | SSE-CBS | SSE-PBS |
| $SEA_S$ | 53.34 | 76.29 | 82.88 | 77.83 | 81.66 | **83.21** |
| $SEA_F$ | 53.21 | 74.64 | 79.46 | 76.67 | 78.91 | **80.77** |
| $Hyp_S$ | 52.32 | 64.74 | **83.49** | 78.45 | 80.03 | 80.89 |
| $Hyp_F$ | 53.56 | 64.2 | 81.14 | 77.78 | 80.11 | **81.23** |
| $RBF_B$ | 54.54 | 75.2 | 77.01 | 74.32 | 77.25 | **77.97** |
| $RBF_{GR}$ | 54.32 | 73.4 | 78.11 | 74.37 | 77.47 | **78.44** |
| $RBF_{ND}$ | 51.76 | 69.5 | 70.53 | 65.43 | 77.33 | **77.98** |
| $LED_{GS}$ | 40.61 | 51.77 | 55.11 | 55.36 | 59.54 | **60.02** |
| $LED_{ND}$ | 36.41 | 42.55 | **44.80** | 42.24 | 41.43 | 42.75 |
| $Rtree_R$ | 57.78 | 64.84 | 68.48 | 67.75 | **76.73** | 76.20 |
| $Rtree_G$ | 60.63 | 74.17 | 73.74 | 74.39 | **84.25** | 84.11 |
| $Rtree_{FG}$ | 55.89 | 64.32 | 66.25 | 65.70 | 70.49 | **73.74** |
| $Rtree_{ND}$ | 64.81 | 76.02 | 76.18 | 76.87 | **84.45** | 84.32 |
| $Agrawal_S$ | 64.19 | 76.35 | 75.54 | 77.92 | 81.14 | **82.62** |
| $Agrawal_G$ | 57.79 | 74.13 | 74.47 | 73.62 | 78.05 | **80.18** |
| $Wave_G$ | 56.61 | 73.15 | 74.44 | 72.19 | **76.93** | 76.24 |
| $Wave_{ND}$ | 59.27 | 74.56 | 78.28 | 72.85 | 77.56 | **79.13** |
| $Stagger_G$ | 55.2 | **67.48** | 62.13 | 61.03 | 65.83 | 65.71 |
| *Elec* | 54.82 | 50.38 | 59.69 | 64.59 | 68.33 | **69.12** |
| *Cov* | 43.35 | 64.11 | **65.60** | 64.44 | 63.91 | 65.24 |
| *Poker* | 45.75 | 55.79 | 57.3 | 54.11 | 58.76 | **59.66** |
| *Airlines* | 51.26 | 56.61 | 56.66 | 55.06 | **59.11** | 59.09 |
| **Avg. Acc.** | 53.52 | 66.55 | 70.06 | 68.32 | 72.69 | **73.57** |
| **Avg. Rank** | 5.95 | 4.32 | 2.86 | 4.09 | 2.32 | **1.45** |

**Table 9**
Accuracy of semi-supervised classifiers (base learner = NB, Labeled data = 5%).

| Datasets | Semi-Supervised Algorithms | | | | | |
|---|---|---|---|---|---|---|
| | COMPOSE | SPASC | SCBELS | STDS | SSE-CBS | SSE-PBS |
| $SEA_S$ | 55.71 | 77.39 | 85.09 | 80.25 | 84.56 | **85.21** |
| $SEA_F$ | 55.75 | 77.41 | 82.23 | 78.98 | 81.76 | **84.87** |
| $Hyp_S$ | 54.94 | 66.28 | 84.64 | 81.22 | 84.38 | **85.65** |
| $Hyp_F$ | 55.16 | 65.57 | 84.19 | 80.43 | 83.18 | **85.09** |
| $RBF_B$ | 56.88 | 77.61 | 79.13 | 79.12 | 80.03 | **80.11** |
| $RBF_{GR}$ | 56.15 | 75.08 | 79.31 | 78.45 | 80.69 | **80.76** |
| $RBF_{ND}$ | 54.74 | 70.61 | 74.86 | 70.27 | 79.22 | **79.94** |
| $LED_{GS}$ | 42.21 | 53.2 | 56.76 | 58.18 | 63.04 | **63.57** |
| $LED_{ND}$ | 37.16 | 43.97 | 46.05 | 44.25 | 47.87 | **48.09** |
| $Rtree_R$ | 60.32 | 67.59 | 70.59 | 70.12 | **78.43** | 78.08 |
| $Rtree_G$ | 62.58 | 75.86 | 75.82 | 76.67 | **87.68** | 86.62 |
| $Rtree_{FG}$ | 58.32 | 66.76 | 68.91 | 68.54 | 72.09 | **75.36** |
| $Rtree_{ND}$ | 66.51 | 79.59 | 79.48 | 78.54 | 84.32 | **87.19** |
| $Agrawal_S$ | 66.72 | 78.38 | 81.25 | 80.97 | **85.78** | 85.05 |
| $Agrawal_G$ | 60.07 | 76.43 | 76.76 | 76.08 | 79.8 | **82.74** |
| $Wave_G$ | 59.08 | 75.2 | 77.58 | 74.74 | **78.16** | 78.12 |
| $Wave_{ND}$ | 60.6 | 77.22 | **81.82** | 75.71 | 78.42 | 80.63 |
| $Stagger_G$ | 57.43 | **69.86** | 64.76 | 64.05 | 65.87 | 68.83 |
| *Elec* | 56.25 | 55.09 | 63.26 | 67.11 | 71.21 | **72.43** |
| *Cov* | 47.76 | 66.22 | 68.05 | 66.41 | 69.02 | **70.11** |
| *Poker* | 48.89 | 57.49 | 60.64 | 56.32 | 61.44 | **62.87** |
| *Airlines* | 53.76 | 57.97 | 59.32 | 57.1 | **62.72** | 61.15 |
| **Avg. Acc.** | 55.77 | 68.67 | 72.75 | 71.07 | 75.44 | **76.48** |
| **Avg. Rank** | 5.95 | 4.45 | 3 | 4.23 | 2.05 | **1.32** |

show that SSE-PBS and SSE-CBS give higher classification performance than the other semi-supervised algorithms on 18 out of 22 datasets. It is also observed that the classification performance of SSE-PBS is close to fully labeled classifiers for nearly all datasets. Additionally, SSE-PBS has the best average rank and the highest mean accuracy.

*5.2.1. Sensitivity to the number of labeled data*

To show the sensitivity of SSE-PBS to the number of labeled examples, we run a set of experiments with 2% and 5% labeled data in each chunk. Tables 8 and 9 show the comparison results between SSE-PBS and state-of-the-art semi-supervised algorithms when their base learner is NB.

In the first experiment, 2% labeled data is used to evaluate the classification performance of the used methods. Table 8 gives the average classification performance of state-of-the-art approaches. As shown, SSE-PBS and SSE-CBS outperforms the other algorithms on 18 out of 22 datasets. We further observe that SSE-PBS achieves the best rank and the highest average accuracy.

In the second experiment, the performance of SSE-PBS is compared to the other algorithms when there is 5% labeled data. Table 9

**Table 10**
Statistical rank (Friedman's test)-base learner = HT, Labeled data = 2%.

| Datasets | Semi-Supervised Algorithms | | | | | | |
|---|---|---|---|---|---|---|---|
| | COMPOSE | STDS | SPASC | SCBELS | Sco-F | SSE-CBS | SSE-PBS |
| $SEA_S$ | 7 | 5 | 6 | 2 | 4 | 3 | 1 |
| $SEA_F$ | 7 | 5 | 6 | 4 | 1 | 3 | 2 |
| $Hyp_S$ | 7 | 4 | 5 | 1 | 6 | 3 | 2 |
| $Hyp_F$ | 7 | 4 | 5 | 1 | 6 | 3 | 2 |
| $RBF_B$ | 7 | 5 | 4 | 3 | 6 | 2 | 1 |
| $RBF_{GR}$ | 7 | 4 | 5 | 3 | 6 | 2 | 1 |
| $RBF_{ND}$ | 7 | 5 | 3 | 1 | 6 | 4 | 2 |
| $LED_M$ | 7 | 3 | 5 | 4 | 6 | 2 | 1 |
| $LED_{ND}$ | 7 | 5 | 6 | 4 | 1 | 2 | 3 |
| $Rtree_R$ | 7 | 5 | 6 | 4 | 3 | 1 | 2 |
| $Rtree_G$ | 7 | 4 | 5 | 6 | 3 | 1 | 2 |
| $Rtree_{FG}$ | 7 | 3 | 6 | 5 | 4 | 2 | 1 |
| $Rtree_{ND}$ | 7 | 3 | 6 | 5 | 4 | 2 | 1 |
| $Agrawal_S$ | 7 | 4 | 5 | 6 | 3 | 2 | 1 |
| $Agrawal_G$ | 7 | 6 | 5 | 4 | 3 | 1 | 2 |
| $Wave_G$ | 7 | 6 | 5 | 4 | 2 | 3 | 1 |
| $Wave_{ND}$ | 7 | 6 | 5 | 3 | 1 | 4 | 2 |
| $Stagger_G$ | 7 | 5 | 1 | 6 | 4 | 3 | 2 |
| *Elec* | 6 | 3 | 7 | 5 | 4 | 2 | 1 |
| *Cov* | 7 | 3 | 6 | 4 | 5 | 2 | 1 |
| *Poker* | 7 | 5 | 6 | 4 | 1 | 2 | 3 |
| *Airlines* | 7 | 4 | 6 | 5 | 3 | 1 | 2 |
| **Avg. Rank** | 6.95 | 4.41 | 5.18 | 3.82 | 3.73 | 2.27 | **1.64** |

**Table 11**
Holm's test: SSE-PBS vs. semi-supervised (learner = HT, %Label = 2%).

| i | Classifier | Avg. rank | Z-score | P-value | $\alpha$/(k-i) |
|---|---|---|---|---|---|
| 1 | COMPOSE | 6.95 | −8.15243740940620 | 3.56661557076978E−16 | 0.008 |
| 2 | SPASC | 5.18 | −5.43495827293747 | 5.48092571162087E−08 | 0.010 |
| 3 | STDS | 4.41 | −4.25277808362621 | 2.11134782122402E−05 | 0.013 |
| 4 | SCBELS | 3.82 | −3.34695170480330 | 0.000817054437191702 | 0.017 |
| 5 | Sco-F | 3.73 | −3.20877479955913 | 0.001333018566914120 | 0.025 |
| 6 | SSE−CBS | 2.27 | −0.96723833670921 | 0.333424901520396000 | 0.050 |

**Table 12**
Holm's test: SSE-PBS vs. semi-supervised (learner = HT, %Label = 5%).

| i | Classifier | Avg. rank | Z-score | P-value | $\alpha$/(k-i) |
|---|---|---|---|---|---|
| 1 | COMPOSE | 6.95 | −8.22920235676408 | 1.88464170339055E−16 | 0.008 |
| 2 | SPASC | 5.55 | −6.07978383074361 | 1.20344679222331E−09 | 0.010 |
| 3 | STDS | 4.36 | −4.25277808362621 | 2.11134782122402E−05 | 0.013 |
| 4 | SCBELS | 4.05 | −3.77683541000739 | 0.000158833616890992 | 0.017 |
| 5 | Sco-F | 3.23 | −2.51789027333826 | 0.011806009400732500 | 0.025 |
| 6 | SSE-CBS | 2.27 | −1.04400328406708 | 0.296483870222749000 | 0.050 |

**Table 13**
Holm's test: SSE-PBS vs. semi-supervised (learner = HT, %Label = 10%).

| i | Classifier | Avg. rank | Z-score | P-value | $\alpha$/(k-i) |
|---|---|---|---|---|---|
| 1 | COMPOSE | 6.95 | −8.29061431465038 | 1.12665054430035E−16 | 0.008 |
| 2 | SPASC | 5.59 | −6.20260774651621 | 5.55351374190192E−10 | 0.010 |
| 3 | STDS | 4.27 | −4.17601313626834 | 2.96662695127916E−05 | 0.013 |
| 4 | SCBELS | 3.86 | −3.54654056793377 | 0.000390324640664697 | 0.017 |
| 5 | Sco-F | 3.64 | −3.20877479955913 | 0.001333018566914120 | 0.025 |
| 6 | SSE-CBS | 2.14 | −0.90582637882291 | 0.365027756488288000 | 0.050 |

shows the classification performance of the used methods. The results show that SSE-PBS and SSE-CBS give higher classification performance than the other semi-supervised algorithms on 20 out of 22 datasets. SSE-PBS also achieves the best average accuracy on all datasets.

Furthermore, in Figs. 10 and 11, we present three box plots that show the average classification accuracy over all semi-supervised algorithms with different proportions of labeled data using NB as the base learner. As shown, SSE-PBS gives the best results over all datasets.

**Table 14**
Holm's test: SSE-PBS vs. semi-supervised (learner = NB, %Label = 2%).

| i | Classifier | Avg. rank | Z-score | P-value | α/(k-i) |
|---|---|---|---|---|---|
| 1 | COMPOSE | 5.95 | −7.97764734385126 | 1.4914880734548E−15 | 0.010 |
| 2 | SPASC | 4.32 | −5.08796619485625 | 3.6192386316403E−07 | 0.013 |
| 3 | STDS | 4.09 | −4.68021977505941 | 2.8656755542659E−06 | 0.017 |
| 4 | SCBELS | 2.86 | −2.49966283440673 | 0.01243115551396900 | 0.025 |
| 5 | SSE-CBS | 2.32 | −1.54234515314458 | 0.12298974198073700 | 0.050 |

**Table 15**
Holm's test: SSE-PBS vs. semi-supervised (learner = NB, %Label = 5%).

| i | Classifier | Avg. rank | Z-score | P-value | α/(k-i) |
|---|---|---|---|---|---|
| 1 | COMPOSE | 5.95 | −8.20811271156252 | 2.2469224280645E−16 | 0.010 |
| 2 | SPASC | 4.45 | −5.54889693027877 | 2.8747758406931E−08 | 0.013 |
| 3 | STDS | 4.23 | −5.15887861569048 | 2.4843330288490E−07 | 0.017 |
| 4 | SCBELS | 3 | −2.97832167503781 | 0.00289831628184782 | 0.025 |
| 5 | SSE-CBS | 2.05 | −1.29415168022476 | 0.19561303335947700 | 0.050 |

**Table 16**
Holm's test: SSE-PBS vs. semi-supervised (learner = NB, %Label = 10%).

| i | Classifier | Avg. rank | Z-score | P-value | α/(k-i) |
|---|---|---|---|---|---|
| 1 | COMPOSE | 5.95 | −8.13720029072829 | 4.0452320134382E−16 | 0.010 |
| 2 | SPASC | 4.73 | −5.97437145528417 | 2.3097897441902E−09 | 0.013 |
| 3 | STDS | 3.95 | −4.59157924901662 | 4.3990453926113E−06 | 0.017 |
| 4 | SCBELS | 2.91 | −2.74785630732655 | 0.00599862919188068 | 0.025 |
| 5 | SSE-CBS | 2.09 | −1.29415168022476 | 0.19561303335947700 | 0.050 |

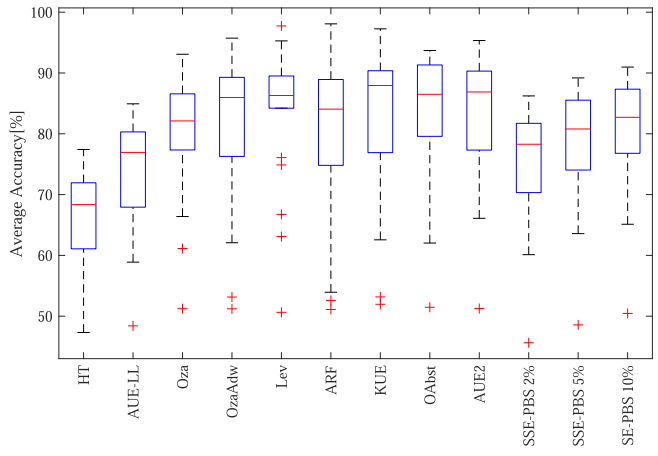



**Fig. 7.** The average accuracy of SSE-PBS and Supervised algorithms with HT as learner over all datasets.

### 5.3. Statistical analyses

In this section, we first analyze the results of the semi-supervised algorithms reported in Table 4, according to the Friedman's test when their base learner is HT and 2% labeled data is available in each chunk. The rank of each algorithm for each dataset is shown in Table 10. Furthermore, average ranks are reported in the last row. Here, the Friedman's test is used to check whether the measured average ranks are significantly different from the mean rank $R_j = 4$ expected under the null-hypothesis. Then, according to Eqs.14 and 15, we compute $\chi_F^2$ and $F_F$ as follows:

$$\chi_F^2 = \frac{12\times22}{7\times8}\left[(6.95^2 + 4.41^2 + 5.18^2 + 3.82^2 + 3.73^2 + 2.27^2 + 1.64^2) - \frac{7\times8^2}{4}\right] = 89.24$$

$$F_F = \frac{21\times89.24}{22\times6-89.24} = 43.83$$

Now, with 7 algorithms and 22 datasets, $F_F$ follows $F-distribution$ with 7–1 = 6 and (7–1)(22–1)=126 degrees of freedom. The critical value of $F(6,126)$ for $\alpha = 0.05$ is 2.17, hence, it rejects the null-hypothesis. Next, we apply the Holm's test, see Table 11.

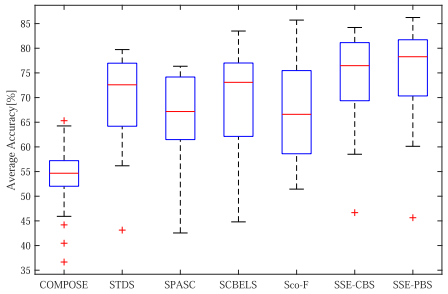
Table 11 shows the results based on the Holm's test. As shown, the Holm procedure rejects all hypotheses except for the last one, since the corresponding $p$-values are smaller than the adjusted $\alpha$'s. Therefore, we conclude that SSE-PBS is significantly different from COMPOSE, SPASC, STDS, SCBELS, and Sco-F. We further observe that SSE-PBS outperforms them when their base learner is HT, and 2% labeled data is available in each chunk.

In the second test, we analyze the results of Table 5. Table 12 shows the results based on the Holm's test. As can be seen, the Holm procedure rejects all hypotheses except for the last one, since the corresponding $p$-values are smaller than the adjusted $\alpha$'s. Hence, it is concluded that SSE-PBS is significantly different from COMPOSE, SPASC, STDS, SCBELS, and Sco-F, and it outperforms them when their base learner is HT and 5% labeled data is available in each chunk.

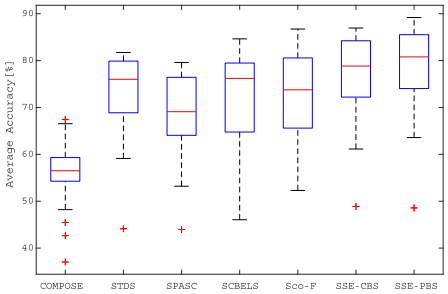
Table 13 shows the results of the analysis in Table 3 based on the Holm's test. As shown, the Holm procedure rejects all hypotheses except for the last one. Therefore, we conclude that SSE-PBS is significantly different from COMPOSE, SPASC, STDS, SCBELS, and Sco-F, and it significantly outperforms them when their base learner is HT, and 10% labeled data is available in each chunk.

Table 14 shows the results of the analysis in Table 8 based on the Holm's test. As can be seen, the Holm procedure rejects all hypotheses, and SSE-PBS is significantly different from COMPOSE, SPASC, STDS, and SCBELS. It is also shown that SSE-PBS outperforms them when their base learner is NB, and 2% labeled data is available in each chunk.

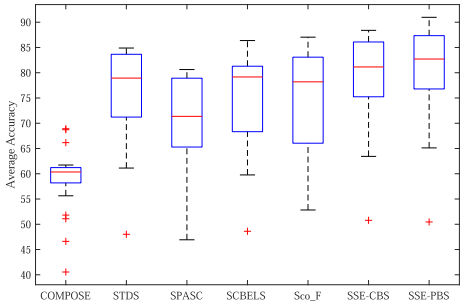
Table 15 gives the results of the analysis in Table 9 based on the Holm's test. In this test, the Holm procedure rejects all hypotheses




(a) 2% Labeled data.




(b) 5% Labeled data.




(c) 10% Labeled data.

**Fig. 8.** The average accuracy of semi-supervised algorithms over all datasets with respect to different proportions of labeled data using HT as the base learner.

except for the last one, and shows that SSE-PBS is significantly different from COMPOSE, SPASC, STDS, and SCBELS. We further observe that SSE-PBS outperforms the other methods when their base learner is NB, and 5% labeled data is available in each chunk.

Finally, Table 16 shows the results of the analysis in Table 7 based on the Holm's test. As shown, the Holm procedure rejects all hypotheses except for the last one, since the corresponding $p$-values are smaller than the adjusted $\alpha$'s. Hence, it is concluded that SSE-PBS is significantly different from COMPOSE, SPASC, STDS, and SCBELS and outperforms the other approaches when their base learner is NB, and 10% labeled data is available in each chunk.

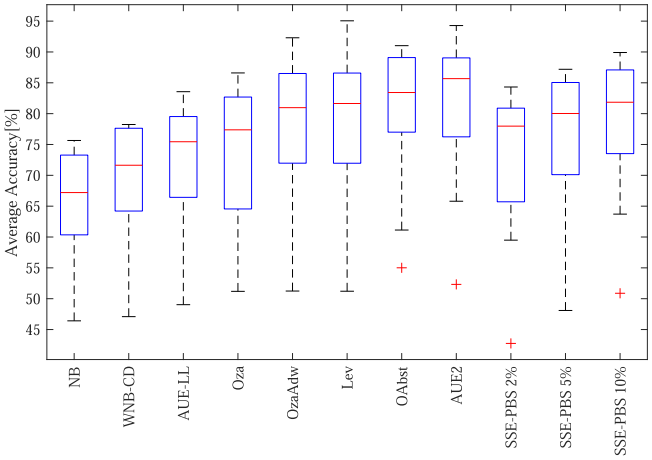



**Fig. 9.** The average accuracy of SSE-PBS and supervised algorithms with NB as learner over all datasets.

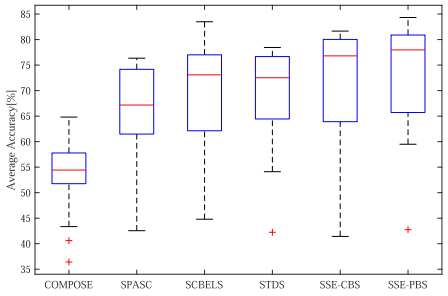
### 5.4. Visual comparison

To display the performance of the proposed algorithm in the case of concept drift, several graphical plots are presented, see Fig. 11a–d. The results are for 10% labeled data in each chunk and the Hoeffting tree as the base learner. The most representative figures are shown, which focus on the properties of SSE-PBS. The measured accuracy on each arriving data chunk is depicted on the y-axis and the number of processed training samples on the x-axis. For a fair comparison, AUE2, AUE-LL, and SSE-CBS and SSE-PBS algorithms are included, which are representatives for three groups FL, LL, and LU.

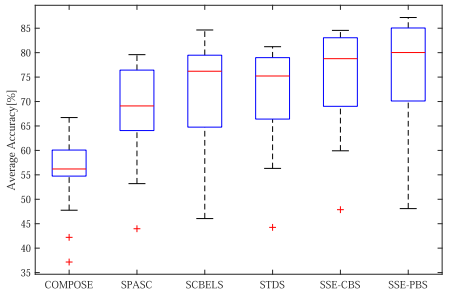
Fig. 11a shows the performance of the algorithms on the $SEA_S$ dataset with sudden drifts. The drifts in every 250 k examples are obvious on the accuracy plot, where decreased accuracy can be observed. Since SSE-PBS incrementally updates its component classifiers after each chunk, the fast adaptation after sudden drifts is achieved. According to the results, we observe that SSE-PBS outperforms the other algorithms except for AUE2, which uses fully labeled data. AUE2 outperforms the others because it is a supervised algorithm, and it has been trained with fully labeled data which means it has more information than the others, and the conditions for comparison are not the same to make a fair judgment.

Next, we investigate the $Hyp_F$ dataset containing incremental drift. As shown in Fig. 11b, the best results belong to AUE2 and SSE-PBS. It seems SSE-PBS can react properly to gradual drift by updating weights of classifiers according to their prediction error.

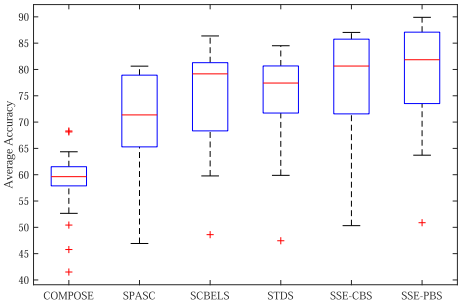
Fig. 11c shows the accuracy plot for the $LED_{GS}$ dataset. In this dataset, we combine a complicated change by blending two gradually drifting streams. At first, there is a gradual drift, after 500 k samples, suddenly another gradual drift occurs. The decreasing accuracy demonstrates that handling merged gradual drifts is a difficult task.




(a) 2% Labeled data.

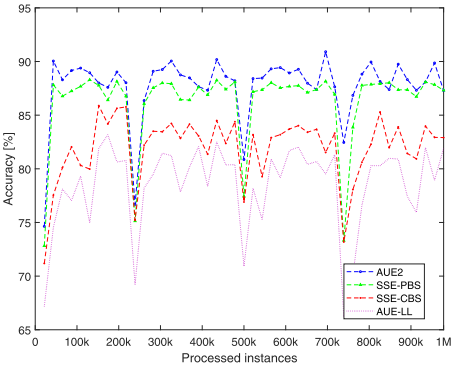



(b) 5% Labeled data.

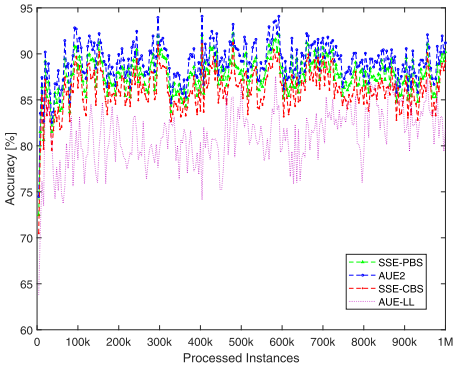



(c) 10% Labeled data.

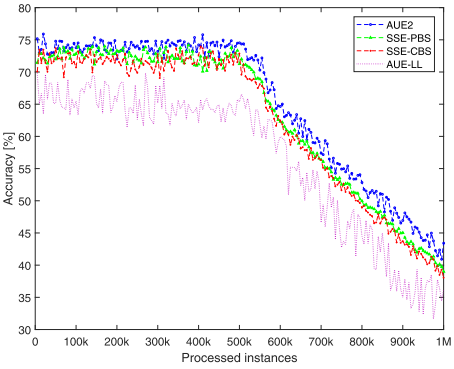
**Fig. 10.** The average accuracy over all datasets with respect to different proportions of labeled data using NB as base learner.



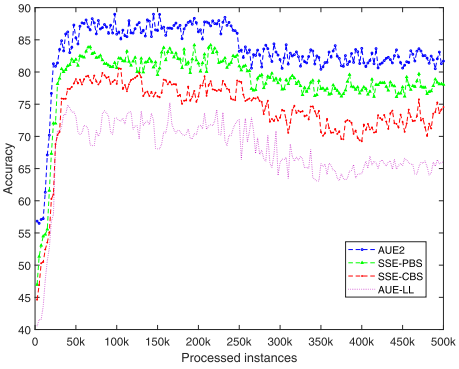

(a) $SEA_S$ dataset




(b) $Hyp_F$ dataset




(c) $LED_{GS}$ dataset




(d) $Cov$ dataset

**Fig. 11.** Average classification performance with 10% labeled data in each chunk on different datasets.

Finally, we employ the *Cov* dataset. Fig. 11d gives the results. As shown, SSE-PBS presents the best performance except for AUE2, which uses fully labeled data. Furthermore, observing the classification performance in Fig. 11d, it is concluded that this dataset probably contains gradual change.

## 6. Conclusion

In this article, we proposed a novel ensemble-based approach to semi-supervised data stream, named SSE-PBS. The proposed algorithm handles various kinds of concept drifts when there is a limited labeled data. SSE-PBS handles the gradual and sudden drifts by combining the weighting mechanism based on the accuracy of the chunk-based ensemble algorithms with the incremental nature of online base learner algorithms.

The main challenge in semi-supervised approaches is how to find a set of informative data points from the unlabeled data. Most of the current approaches, employ the probability estimation of the base learner to select from the newly-labeled data. However, this is not the optimal way in many cases. Our proposed algorithm does not rely on the confidence of the classifier as a selection criterion to select newly labeled data for improving classification performance. Instead, we used a new selection algorithm to evaluate the ability of candidate unlabeled data to improve classification performance. In particular, only unlabeled samples that can help to enhance the classification performance are selected for retraining the classification model. The confidence score is a criterion that measures the accuracy of a given sample, rather than a criterion that assesses the potential of a sample's participation in training for improving the current model. Therefore, the use of confidence score as the selection criterion in the semi-supervised training procedure may lead to inaccurate estimation of actual distribution and thereby resulting in performance degradation, even if the selected samples are classified correctly.

Therefore, we employed a new selection metric based on the classification performance and formulated the proposed selection metric in terms of pseudo-accuracy and energy regularization.

To evaluate the performance of the proposed algorithm, we set up a variety of experiments on synthetic and real-world datasets. Our experimental results indicate that our selection metric exploits informative unlabeled data and improves the classification performance properly. We further showed that SSE-PBS can effectively handle the different concept drifts. We also observed that the proposed algorithm can handle the data stream classification task with very limited labeled data, e.g., 2% labeled data. We showed that SSE-PBS can employ any incremental base learner. Finally, we demonstrated that our proposed method outperforms the other state-of-the-art semi-supervised algorithms using statistical tests.

In the future, we plan to improve the robustness of the model to noise. Furthermore, we would like to deal with concept drift in imbalanced limited labeled data streams.

## 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.

## References


[1] C.C. Aggarwal, Data streams: an overview and scientific applications, in: Scientific Data Mining and Knowledge Discovery, Springer, 2009, pp. 377–397.
[2] R.S.M. Barros, S.G.T.C. Santos, A large-scale comparison of concept drift detectors, Inf. Sci. 451 (2018) 348–370.
[3] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, 2007, pp. 443–448.
[4] A. Bifet, R. Gavaldà, Adaptive learning from evolving data streams, in: International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 249–260..
[5] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Moa: Massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[6] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2010, pp. 135–150.
[7] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 139–148..
[8] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, ACM, 1998, pp. 92–100.
[9] H.R. Bonab, F. Can, Goowe: geometrically optimum and online-weighted ensemble classifier for evolving data streams, ACM Trans. Knowl. Discovery Data (TKDD) 12 (2018) 1–33.
[10] H. Borchani, P. Larrañaga, C. Bielza, Classifying evolving data streams with partially labeled data, Intell. Data Anal. 15 (2011) 655–670.
[11] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[12] D. Brzeziński, Mining data streams with concept drift. Ph.D. thesis. PhD thesis, MS thesis, Dept. of Computing Science and Management, Poznan University of Technology, Poznan Google Scholar, 2010..
[13] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: the accuracy updated ensemble algorithm, IEEE Trans. Neural Networks Learn. Syst. 25 (2014) 81–94.
[14] A. Cano, B. Krawczyk, Kappa updated ensemble for drifting data stream mining, Mach. Learn. 109 (2020) 175–218.
[15] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, X. Tong, Textflow: towards better understanding of evolving topics in text, IEEE Trans. Visualiz. Comput. Graphics 17 (2011) 2412–2421.
[16] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[17] S.U. Din, J. Shao, J. Kumar, W. Ali, J. Liu, Y. Ye, Online reliable semi-supervised learning on evolving data streams, Inf. Sci. (2020).
[18] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: a survey, IEEE Comput. Intell. Mag. 10 (2015) 12–25.
[19] Y. Dong, N. Japkowicz, Threaded ensembles of autoencoders for stream learning, Comput. Intell. 34 (2018) 261–281.
[20] K.B. Dyer, R. Capo, R. Polikar, Compose: a semisupervised learning framework for initially labeled nonstationary streaming data, IEEE Trans. Neural Networks Learn. Syst. 25 (2014) 12–26.
[21] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Networks 22 (2011) 1517–1531.
[22] R.S. Ferreira, G. Zimbrão, L.G. Alvim, Amanda: semi-supervised density-based adaptive model for non-stationary data with extreme verification latency, Inf. Sci. 488 (2019) 219–237.
[23] A. Frank, A. Asuncion, et al., Uci machine learning repository, 2010 (2011). http://archive. ics. uci. edu/ml 15, 22..
[24] M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, Mining data streams: a review, ACM Sigmod Record 34 (2005) 18–26.
[25] J. Gama, Knowledge Discovery from Data Streams, Chapman and Hall/CRC, 2010.
[26] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surveys (CSUR) 46 (2014) 44.
[27] J. Gao, W. Fan, J. Han, P.S. Yu, A general framework for mining concept-drifting data streams with skewed distributions, in: Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, 2007, pp. 3–14.
[28] J. Gao, W. Fan, J. Jiang, J. Han, Knowledge transfer via multiple model local structure mapping, in, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 283–291.
[29] H.M. Gomes, J.P. Barddal, F. Enembreck, A. Bifet, A survey on ensemble learning for data stream classification, ACM Comput. Surveys (CSUR) 50 (2017) 23.
[30] 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.
[31] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70.
[32] M.J. Hosseini, A. Gholipour, H. Beigy, An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams, Knowl. Inf. Syst. 46 (2016) 567–597.
[33] R.L. Iman, J.M. Davenport, Approximations of the critical region of the fbietkan statistic, Commun. Stat. Theory Methods 9 (1980) 571–595.
[34] S. Khezri, J. Tanha, A. Ahmadi, A. Sharifi, Stds: self-training data streams for mining limited labeled data in non-stationary environment. Appl. Intell. 1–20..
[35] S. Khezri, J. Tanha, A. Ahmadi, A. Sharifi, Stds: self-training data streams for mining limited labeled data in non-stationary environment, Appl. Intell. (2020) 1–20.
[36] R.B. Kirkby, Improving hoeffding trees. Ph.D. thesis. The University of Waikato, 2007..
[37] R. Klinkenberg, Learning drifting concepts: example selection vs. example weighting, Intell. Data Anal. 8 (2004) 281–300.
[38] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.

[39] B. Krawczyk, A. Cano, Online ensemble learning with abstaining classifiers for drifting and noisy data streams, Appl. Soft Comput. 68 (2018) 677–692.
[40] 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.
[41] B. Krawczyk, M. Wozniak, Weighted naive bayes classifier with forgetting for drifting data streams, in: 2015 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2015, pp. 2147–2152.
[42] P. Kulkarni, R. Ade, Incremental learning from unbalanced data with concept class, concept drift and missing features: a review, Int. J. Data Min. Knowl. Manage. Process 4 (2014) 15.
[43] L.I. Kuncheva, Change detection in streaming multivariate data using likelihood detectors, IEEE Trans. Knowl. Data Eng. 25 (2011) 1175–1180.
[44] P. Li, X. Wu, X. Hu, Mining recurring concept drifts with limited labeled streaming data, in: Proceedings of 2nd Asian Conference on Machine Learning, 2010, pp. 241–252.
[45] A. Liu, J. Lu, F. Liu, G. Zhang, Accumulating regional density dissimilarity for concept drift detection in data streams, Pattern Recogn. 76 (2018) 256–272.
[46] D. Malekian, M.R. Hashemi, An adaptive profile based fraud detection framework for handling concept drift, in: 2013 10th International ISC Conference on Information Security and Cryptology (ISCISC), IEEE, 2013, pp. 1–6.
[47] M.M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K.W. Hamlen, N.C. Oza, Facing the reality of data stream classification: coping with scarcity of labeled data, Knowl. Inf. Syst. 33 (2012) 213–244.
[48] H.L. Nguyen, Y.K. Woon, W.K. Ng, A survey on data stream clustering and classification, Knowl. Inf. Syst. 45 (2015) 535–569.
[49] M.K. Olorunnimbe, H.L. Viktor, E. Paquet, Dynamic adaptation of online ensembles for drifting data streams, J. Intell. Inf. Syst. 50 (2018) 291–313.
[50] N.C. Oza, Online bagging and boosting, in: 2005 IEEE International Conference on Systems, Man and Cybernetics, IEEE, 2005, pp. 2340–2345..
[51] A. Pesaranghader, H. Viktor, E. Paquet, Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams, Mach. Learn. 107 (2018) 1711–1743.
[52] A. Pesaranghader, H.L. Viktor, Fast hoeffding drift detection method for evolving data streams, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2016, pp. 96–111.
[53] L. Pietruczuk, L. Rutkowski, M. Jaworski, P. Duda, How to adjust an ensemble size in stream data mining?, Inf. Sci. 381 (2017) 46–54.
[54] B.R. Prasad, S. Agarwal, Stream data mining: platforms, algorithms, performance evaluators and research trends, Int. J. Database Theory Appl. 9 (2016) 201–218.
[55] T. Sakaki, M. Okazaki, Y. Matsuo, Earthquake shakes twitter users: real-time event detection by social sensors, in: Proceedings of the 19th International Conference on World Wide Web, ACM, 2010, pp. 851–860.
[56] P. Sobolewski, M. Woźniak, Scr: simulated concept recurrence–a non-supervised tool for dealing with shifting concept, Expert Syst. 34 (2017) e12059.
[57] W.N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 377–382..
[58] J. Tanha, Mssboost: a new multiclass boosting to semi-supervised learning, Neurocomputing 314 (2018) 251–266.
[59] J. Tanha, A multiclass boosting algorithm to labeled and unlabeled data, Int. J. Mach. Learn. Cybern. 10 (2019) 3647–3665.
[60] J. Tanha, M. van Someren, H. Afsarmanesh, Semi-supervised self-training for decision tree classifiers, Int. J. Mach. Learn. Cybern. 8 (2017) 355–370.
[61] J. Tanha, et al., Ensemble approaches to semi-supervised learning. PhD thesis, 2013. ISBN: 978-90-5335-669-2, SIKS..
[62] A. Tsymbal, The problem of concept drift: definitions and related work, Computer Science Department, Trinity College Dublin 106, 2004.
[63] M. Umer, C. Frederickson, R. Polikar, Learning under extreme verification latency quickly: fast compose, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2016, pp. 1–8..
[64] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 226–235..
[65] Y. Wang, T. Li, Improving semi-supervised co-forest algorithm in evolving data streams, Appl. Intell. (2018) 1–15.
[66] Y.M. Wen, S. Liu, Semi-supervised classification of data streams by birch ensemble and local structure mapping, J. Comput. Sci. Technol. 35 (2020) 295–304.
[67] J. Wu, L. Li, W.Y. Wang, Reinforced co-training, 2018. arXiv preprint arXiv:1804.06035..
[68] T. Zhai, Y. Gao, H. Wang, L. Cao, Classification of high-dimensional evolving data streams via a resource-efficient online ensemble, Data Min. Knowl. Disc. 31 (2017) 1242–1265.
[69] M.L. Zhang, Z.H. Zhou, Cotrade: confident co-training with data editing, IEEE Trans. Syst., Man, Cybern. B (Cybern.) 41 (2011) 1612–1626.
[70] R. Zhang, A.I. Rudnicky, A new data selection principle for semi-supervised incremental learning, in: 18th International Conference on Pattern Recognition (ICPR'06), IEEE, 2006, pp. 780–783..
[71] T. Zhang, R. Ramakrishnan, M. Livny, Birch: a new data clustering algorithm and its applications, Data Min. Knowl. Disc. 1 (1997) 141–182.
[72] X. Zhang, J. Li, H. Yu, Local density adaptive similarity measurement for spectral clustering, Pattern Recogn. Lett. 32 (2011) 352–358.
[73] Z.H. Zhou, M. Li, Tri-training: exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (2005) 1529–1541.
[74] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 912–919..

**Shirin Khezri** received B.Sc. degree in computer science from Tabriz University in 2007, M.Sc. degree in artificial intelligence from IAU Qazvin branch in 2010, and Ph.D. degree in artificial intelligence from IAU science and research branch in 2020. Her research interests include data stream mining, ensemble classifiers, and semi-supervised classification algorithms. She is currently a faculty member at Payame Noor University, Mahabad, Iran.

**Jafar Tanha** was born in Bonab, Iran. He received the B. Sc and M.Sc degree in computer science from the university of AmirKabir (Polytechnic), Tehran, Iran, in 1999 and 2001, respectively, and the Ph.D degree in computer science-Artificial Intelligence from the University of Amsterdam (UvA), Amsterdam, The Netherlands, in 2013. He joined at INL institute, Leiden, The Netherland, as a researcher, from 2013 to 2015. Since 2015, he has been with the Department of Computer Engineering, Payame-Noor University, Tehran, Iran, where he was an Assistance Professor. He has held lecturing positions at the Iran university of Science & Technology, Tehran, Iran, in 2016. His current position is the assistant professor at the University of Tabriz, Tabriz, Iran. His main areas of research interest are machine learning, pattern recognition, and document analysis.

**Ali Ahmadi** received the Postdoctoral fellow in Learning Systems and their Hardware Implementation, COE Program from Research Center for Nano-devices and Systems from Hiroshima University, Hiroshima, Japan, in 2004–2007, Ph.D. and M.SC. degree from University of Osaka Prefecture, Osaka, Japan, in 2004 and 2001 respectively, and B.SC. in Electrical Engineering from Amirkabir University of Technology, Tehran, Iran, 1990. He is an Associate Professor in K. N. Toosi university of Technology, Tehran, Iran. His current research interests include Interactive learning, Semantic data mining and information fusion, image search engine, virtual reality and artificial life, and machine vision and image processing.

**Arash Sharifi** received B.S. degree in computer hardware engineering from IAU South Tehran Branch, M.S degree and Ph.D. degree in artificial intelligence from IAU science and research branch, in 2007 and 2012 respectively. He is currently head of computer engineering department of SRBIAU. His current research interests include image processing, machine learning and deep learning.