A Semi-supervised Based Framework for Data Stream Classification in Non-Stationary Environments

Arthur Costa Gorgônio, Anne Magály de P. Canuto
Dept. of Informatics and Applicated Mathematics (DIMAp)
Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
arthurgorgonio@ppgsc.ufrn.br, anne@dimap.ufrn.br

Karliane M. O. Vale, Flavius L. Gorgônio
Dept. of Computing and Technology (DCT)
Federal University of Rio Grande do Norte (UFRN)
Caicó, Brazil
karliane@dct.ufrn.br, flavius@dct.ufrn.br

***Abstract*—Semi-supervised learning (SSL) is a paradigm that has been continuously used in data classification tasks in datasets that do not have enough labeled instances to train a supervised model with a minimum acceptable accuracy. In this context, data stream classification in dynamic environments appears as a natural application for this approach, because changes in data distribution contribute to decrease the performance of the classification algorithms. In this paper, we have proposed a framework, refered to as Dynamic Data Stream Learning (DyDaSL), that implements an auto-adaptive classifier ensemble – which is able to evaluate and replace classifiers with decreasing performance. This platform uses the FlexCon-C method, which is a variant of the self-training SSL algorithm that adjusts a confidence threshold dynamically, in each iteration, to define which instances will be labeled. Experimental tests on synthetic and real datasets show that the proposed approach obtains better results than traditional approaches using four evaluation metrics: accuracy, F-score, precision, and recall.**

***Index Terms*—Semi-supervised learning, data stream classification, concept drift.**

I. Introduction

Recently, with advances in hardware and software technologies, data analysis has became a daily task. With this, it is possible, for instance, when performing the processing of a massive number of data, that machines can develop these analysis in a shorter time, when compared to humans. Hence, the Machine Learning area provides techniques that use a history of data to perform data analysis and, based on this analysis, using the obtained model to make decisions [1]. In this area, the learning task can be broadly divided into supervised, unsupervised (or clustering) and semi-supervised learning.

Supervised learning is used when all instances in a dataset have labels which are previously known, obtaining a classification model. On the other hand, unsupervised learning is used when there is no label, but it is still possible to cluster the dataset into groups. There is, therefore, high homogeneity among instances of the same group and high heterogeneity among instances of different groups. Finally, the semi-supervised learning (SSL) approach combines the previous two into one, using labeled and unlabeled instances to generate a model. This approach has been increasingly used in real scenarios, since the real world datasets usually have a small amount of labeled instances, which may not be enough to train a supervised model with a minimum acceptable accuracy [2], [3]. Therefore, the SSL approach uses all of the labeled instances to train a classifier with this subset. Then, the built model is applied to label the unlabeled subset and some newly labeled instances are selected to be included in the labeled set. This whole process is repeated until all unlabeled instances are labeled or a stopping criterion is reached. In this paper, we will be working with the semi-supervised learning approach.

When a classification algorithm is applied to a training set and a model is generated, it is able to perform the classification of the remaining data. However, it is true only in a stationary data stream context, since the analyzed environment does not change. In addition, it is well known that the training step can be very sensitive to some aspects, including: training set, selected classification algorithm and data distribution.

In a non-stationary (or dynamic) data stream context, we can apply two machine learning strategies: online learning and batch learning [4]. In the first case, instances are continuously received; and in the second one, n instances are grouped to generate data conglomerates. Furthermore, in dynamic environments, data distribution can be affected by some factors such as number of instances, concept drift, among others, and it is called data stream. One of these factors is the *concept drift*, which occurs when data distribution changes over time and this corresponds to changes in the underlying stream patterns. In this case, the performance of the classification algorithms decreases substantially [5].

Concept drift occurs in non-stationary data streams and a model needs to adapt to perform the classification task [6]. There are four types of concept drift: abrupt, gradual, incremental and recurring. In the abrupt case, the analyzed data changes the distribution very quickly, this type is the easiest one to identify. In the gradual type, the data distribution gradually changes in a transition window. In the incremental type, the learned concept slowly evolves over time, similarly to

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

the gradual type. Finally, in the recurring type, changes occur periodically in the data distribution. In addition, noise and outliers should not be confused with drifts because they can occur in all contexts [7]. One attempt to detect concept drifts is through the use of classifier ensembles, since they can surpass the over-fitting delivered by a single classifier. Nevertheless, building those models is a complex computational task [8].

Once semi-supervised methods have shown very positive results in applications of different domains in stationary environments [9], it is expected that they can also obtain promising results in the data stream context. In this paper, we will investigate the performance of a semi-supervised method, named Flexible Confidence with Classifier (FlexCon-C) [10], which is an extension of a self-training method, in a non-stationary data streams context. The main aim of this investigation is to identify the concept drift by training a classifier in each iteration of the analyzed method which will be used to classify the new instances trying to detect when the drift occurs.

This paper is divided into seven sections as follows. Section II describes the theoretical aspects related to the subject of this paper, while Section III presents some related work. In Section IV, the proposed architecture is described. The experiment methodology is described in Section V, then our results are presented in Section VI. Finally, in Section VII we present the conclusions and suggestions for future works.

II. Theoretical Aspects

In this section, we present the theoretical aspects related to the subject of this paper, focusing mainly on semi-supervised learning (Section II-A) and classifier ensembles (Section II-B).

A. Semi-supervised Learning

Sometimes, it is impossible to obtain a dataset which is fully labeled or has a large proportion of the labeled data to train supervised models. This activity usually requires a huge amount of time and/or specialists to classify all the dataset instances. Real world databases usually have a small percentage of labeled data and, in such cases, it is not useful to apply the supervised learning because this approach requires a fully labeled dataset to train the model with high accuracy. A solution to this problem is the use of semi-supervised learning (SSL).

A popular semi-supervised algorithm is self-training, which uses the classifier predictions to improve its performance in classification tasks, considering that the labeled subset is increased in each iteration of the labeling process [9]. In order to improve the performance and/or functioning of the self-training algorithm, extensions have been proposed in the literature [11]–[13].

Recently, a self-training method with the flexible confidence threshold, called (FlexCon-C), was proposed [10]. It applies a confidence threshold as a selection criterion in the labeling process [10]. Basically, it defines the minimum acceptable confidence to be used in the labeling process. In other words, the unlabeled instances whose confidence is higher than the confidence threshold are selected for labeling and possibly for inclusion in the labeled set.

The SSL algorithms consider some basic assumptions. Let D be a dataset composed by $l + u$ instances, as being divided into two subsets: i) a subset of l labeled instances $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, where $\mathbf{x}_i$ corresponds to an instance in L and y_i corresponds respectively to the label of each $\mathbf{x}_i$, and ii) a subset of u unlabeled instances $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$, where $\mathbf{x}_j$ corresponds to an instance in U. In addition, in most cases, $|U| \gg |L|$. Algorithm 1 shows the general idea of the FlexCon-C, whose approach follows the following steps: i) split a dataset into labeled (L) and unlabeled (U); ii) train the classifier (f), with supervised learning; iii) classify the unlabeled instances in U using the classifier f; iv) select the unlabeled instances from U that satisfy a selection criteria, label and move them to selected set (S); v) adjust confidence value according to Algorithm 2; vi) move the labeled instances from S to L.

Algorithm 1: Pseudo-code of the FlexCon-C

Require: L $\leftarrow \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and U $\leftarrow \{\mathbf{x}_j\}_{j=l+1}^{l+u}$
1: e $\leftarrow$ 0;
2: $L_0 \leftarrow$ L;
3: **while** U $\neq \emptyset$ **do**
4: e $\leftarrow$ e + 1;
5: Train classifier f using labeled instances L with supervised learning;
6: Classify unlabeled data U using f;
7: Remove the subset of the instances S from U, so that confidence rate in $conf(\mathbf{x}) \geq conf(t_e)$;
8: $conf(t_e) \leftarrow$ Adjust Confidence Value (L_0, S, $conf(t_e)$)
9: Move all instances from S to L;
10: **end while**
11: **return** f;

As it can be observed in Algorithm 1, FlexCon-C has two important aspects; i) it changes the confidence threshold dynamically in each iteration; and ii) it is a wrapper-based method in which the output of a classifier is used to change the confidence threshold.

To satisfy the first aspect of FlexCon-C, a confidence threshold updating equation was defined, which is presented in Eq. (1). In this equation, the new confidence threshold ($conf(t_{e+1})$) is defined using the current confidence threshold ($conf(t_e)$) and a change rate (cr). Nevertheless, the correct updating depends on the classifier accuracy (acc), a minimum acceptable precision (mp) and an acceptable variation (ϵ).

$$conf(t_{e+1}) = \begin{cases} conf(t_e), & \text{if } mp - \varepsilon < acc < mp + \varepsilon \\ conf(t_e) - cr, & \text{if } acc \geq mp + \varepsilon \\ conf(t_e) + cr, & \text{if } acc \leq mp - \varepsilon \end{cases} \quad (1)$$

Algorithm 2 describes the confidence threshold updating strategy, which can be briefly described in four steps: i) train

a classifier f using the initial labeled data (L_0); ii) measure the accuracy of f when applying f on the L_0 set; iii) train another classifier f' using the selected instances S; iv) apply the f' on L_0; v) change the iteration threshold based on the accuracy of the f', according to Eq. (1).

Algorithm 2: Adjust Confidence Value (L_0, S, $conf(t_e)$)

$cr \leftarrow 0.05$
$\epsilon \leftarrow 0.01$
Train f using L_0 with supervised learning;
$mp \leftarrow$ accuracy of f in the L_0 set;
Train f' using select instances in S with supervised learning;
Apply f' in each instance of L_0 set;
if accuracy of $f' \leq mp - \epsilon$ **then**
    return $conf(t_e) + cr$
else if accuracy of $f' \geq mp + \epsilon$ **then**
    return $conf(t_e) - cr$
else
    return $conf(t_e)$
end if

In order to satisfy the first aspect of FlexCon-C, it is important to emphasize that it uses two types of predictions, namely the predictions generated in the first iteration, and the others generated in each iteration, with the newly included instances. These predictions will be compared using the following criteria:

1) An instance is classified with the same label by both predictions and both confidence values are higher than the confidence threshold;
2) An instance is classified with the same label by both predictions, but just one of confidence value is higher than the confidence threshold;
3) An instance is classified with different labels, but both confidence values are higher than the confidence threshold;
4) An instance is classified with different labels and just one of confidence values are higher than the confidence threshold;
5) The threshold is changed to the maximum confidence value of the present iteration.

The following criteria will be evaluated if, and only if, the previous criteria select no instances to be labeled [10], [14].

B. Classifier Ensembles

Recent advances in the applicability of classification algorithms in a wide range of tasks have encouraged their use. Nevertheless, there is a perception that no classifier is considered completely satisfactory for a particular task. Therefore, the idea of combining different methods to improve performance has emerged as a very promising possibility [10]. Combining classifiers explores the idea that different classifiers can provide additional information concerning patterns to be classified, improving the effectiveness of the recognition process as a whole [15].

In these cases, the reliability of the classification process is increased by classifier ensembles. In other words, an ensemble is used when it is required to maximize the accuracy of the process, and it combines several predictions regarding the same instance into one single output. An ensemble of classifiers can be configured in different ways: it can be a homogeneous or a heterogeneous ensemble; it has a varied number of classifiers; and it uses different strategies for selecting input instances (bagging, boosting, stacking) and combining their outputs (through simple, majority voting or use of a meta-classifier) [16].

An ensemble is called homogeneous when all the base classifiers are of the same type. On the other hand, when the ensemble has more than one classifier type, it is called heterogeneous. Also, in an ensemble approach it is possible to combine a lot of outputs from base-classifiers into a single output, and a meta classifier uses the outputs of the base classifiers to select the label of the instance. This strategy is preferred instead of voting, because it uses more than one information from each classifier to predict an instance.

Finally, in an ensemble algorithm we can define an abstract fusion model called *oracle*. This model is the most precise than others, because it always selects the classifiers which predict the correct label of the instance [17].

III. Related Work

In this section we will focus on the studies related to the main subjects of this paper, an ensemble approach in the context of data streams [18]–[20], concept drift detection [21], [22] and semi-supervised learning [10], [14]. All of these works are sorted by publication year.

In the studies related to concept drift, an adaptive Random Forest was proposed in [20]. In the cited work, the authors modified one of the most used ensemble generation methods, random forest, to perform the classification of data stream in non-stationary environments using the batch strategy. The drift adaptation strategy trains a background tree to replace the primary tree when this model identifies a drift. Finally, the proposed model was compared to other state-of-the-art algorithms and the proposed approach obtained good classification performance.

Still in the context of concept drift, a drift detection based on active learning was presented in [21]. The authors proposed a new method, called DDAL (drift detection based on active learning), which selects the most significant instances to measure density variation in order to identify drift occurrences. Another drift detection method was developed in [22], which is based on a re-sampling scheme and a paired student t-test to identify a drift. This approach generated a sub-window and compared to the last sub-window generated using a paired t-test, this is a drift detection module, because when the t-test presents a significance, it is possible to assume that the distributions are different, and therefore, there was *drift*.

In the context of semi-supervised classification tasks in data stream, SPASC – an ensemble of cluster-based classifiers – is intended to recognize recurring concept drifts of data streams, proposed in [18]. The main idea of this paper is to use a pool of active classifiers, with each classifier being representative of one single concept. At first, a batch of instances is classified by this algorithm. Thereafter, some of these instances are labeled and this partially labeled batch is used to update the classifiers of the pool. In [19], a framework for imbalanced stream classification was proposed. They focused on correctly classifying the minority class of the selected data streams. The strategy of the classification task was to update a classifier using newly instances of the stream.

The FlexCon-C algorithm was originally proposed in [10]. Along with this algorithm, two other self-training extensions were proposed that use different strategies to update the confidence threshold. All proposed methods were compared to the original self-training method in an empirical analysis. In the reported empirical analysis, 20 datasets were investigated with different percentages of initially labeled instances, applied to three classification algorithms (Naïve Bayes, DT and RIPPER). The FlexCon-C algorithm has a confidence threshold parameter that is updated in each iteration, and an analysis of this parameter was the study object in [14]. The authors analyzed the effect of the confidence threshold parameter in FlexCon-C.

Similar to [18] and [20], in this paper we use a batch learning strategy to train a semi-supervised framework in non-stationary data streams. We propose to use an ensemble with a small number of base classifiers which, when their performance decreases, are replaced by more accurate classifiers. This approach differs from the others because we use the FlexCon-C algorithm and, in each iteration, a new classifier trained with the current batch is assumed to be the most reliable one (the oracle). Finally, we use this oracle to evaluate the ensemble performance and its respective base classifiers in order to evaluate the need to make any changes on the classifier ensemble.

IV. The Proposed Approach

As previously mentioned in this work, we develop a framework, named DyDaSL (Dynamic Data Stream Learning), which uses the FlexCon-C method to train a classifier ensemble, in a semi-supervised batch learning process, applied to a dynamic data stream context.

Figure 1 presents the DYDaSL workflow. Initially, each dataset is divided into batches – to simulate the stream data environment – that will be used as input data. Each batch is divided so that 75% of the dataset is included in the training set and 25% is reserved for testing. Moreover, we split the training dataset into labeled and unlabeled. In the first batch, three classifiers are trained to compose the classifier ensemble and, in the next batches, one classifier (the oracle) – used to detect a drift – is trained in order to evaluate the ensemble performance in the current stream. Then, we use the oracle prediction in the current batch to evaluate the need to replace any base classifier of the ensemble. If this classifier ensemble achieves a minimum acceptable accuracy in the current batch, it is not updated. Otherwise, the base classifier with worst performance (in relation to accuracy) will be replace by the oracle. The whole process is repeated and a new oracle is trained with the current batch. After processing the last batch, the trained model will be returned and the training step is completed.

As it can be observed, an oracle classifier is used and we assume that this oracle is more confident when compared to the ensemble or the base classifiers outputs, since it is trained with the i^{th} batch. Then, we can state that this is a passive detection drift approach, once the current batch is not actively analyzed by an algorithm/method. However, a classifier is trained to assess whether or not there has been a drift.

V. Experimental Methodology

This section describes the main aspects of the empirical study conducted in this paper, aiming to evaluate the performance of the proposed framework.

In order to perform a comparative analysis, five methods from data stream classification widely used in the literature were selected, which are:

1) Hoeffding Trees (HT) [23];
2) Active Classifier (AC) [24];
3) Oza Boost Adwin (OBA) [25];
4) Accuracy Weight Ensemble (AWE) [26]; and
5) Anticipative Dynamic Adaptation to Concept Change (ADACC) [27].

The first method is a single Decision Tree that performs the classification task in data streams. The second one is based on an active learning model and the last three methods are ensemble-based methods.

All these compared methods are available in the "RMOA" package, an API for Massive Online Analysis (MOA) framework [28]. Four classification algorithms were used as base classifiers of the FlexCon-C method, which are: Naïve Bayes (NB) [29], Decision Tree (DT) [30], RIPPER [31] and *k*-NN [32]. These algorithms were available in the "RWeka" package, an API for WEKA [33].

In order to evaluate the performance of the analyzed methods, the following metrics are used: Accuracy, Precision, Recall and F-Score. Table I shows the datasets characteristics, number of instances, features and classes for each dataset. These datasets, which combine both real and synthetic data, were obtained in [28], [34], [35].

In order to simulate a data stream environment, a conversion of the static datasets into dynamic data is required. Therefore, subsets with *n* instance are grouped to generate data conglomerates (batches), and the same batch was used to train all analyzed methods. In this paper, two batch sizes are used: 500 and 5000 instances.

When a new batch is received, it will be divided into training and testing subsets, using a 75% and 25% proportion of instances. The next step is to split the training subset into ten stratified folds for cross-validation (each model is trained

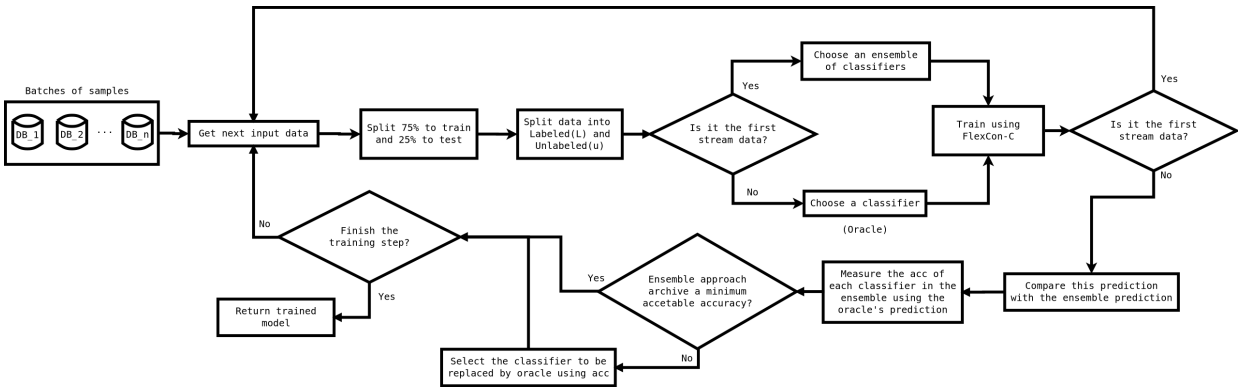


Fig. 1. Workflow of the DyDaSL method

TABLE I
DESCRIPTION OF THE DATASETS USED IN THE EXPERIMENTS

Datasets	Number of instances	Features	Classes
Real datasets			
Adult	32,561	14	2
Airlines	539,383	7	2
Electricity	45,312	8	2
Synthetic datasets			
GearS2C2D	200,000	2	2
UG2C3D	200,000	3	2

TABLE II
RESULTS FOR ADULT DATASET

batch size = 500				
Method	Accuracy	F-Score	Precision	Recall
DyDaSL	0.73 ± 0.08	0.58 ± 0.12	0.57 ± 0.15	0.59 ± 0.09
HT	$\mathbf{0.82 \pm 0.04}$	$\mathbf{0.73 \pm 0.06}$	$\mathbf{0.77 \pm 0.06}$	$\mathbf{0.70 \pm 0.06}$
AC	0.57 ± 0.25	0.55 ± 0.17	0.56 ± 0.20	0.56 ± 0.12
OBA	0.38 ± 0.21	0.33 ± 0.18	0.29 ± 0.21	0.52 ± 0.06
AWE	0.76 ± 0.02	0.43 ± 0.01	0.38 ± 0.01	0.50 ± 0.00
ADACC	0.76 ± 0.02	0.43 ± 0.02	0.38 ± 0.08	0.03 ± 0.01
batch size = 5000				
DyDaSL	$\mathbf{0.80 \pm 0.02}$	0.70 ± 0.06	$\mathbf{0.73 \pm 0.08}$	0.67 ± 0.06
HT	0.56 ± 0.30	0.50 ± 0.28	0.48 ± 0.33	0.62 ± 0.11
AC	0.24 ± 0.01	0.19 ± 0.00	0.12 ± 0.00	0.50 ± 0.00
OBA	0.52 ± 0.26	0.39 ± 0.13	0.35 ± 0.17	0.50 ± 0.00
AWE	0.79 ± 0.04	$\mathbf{0.74 \pm 0.05}$	$\mathbf{0.73 \pm 0.03}$	$\mathbf{0.74 \pm 0.07}$
ADACC	0.54 ± 0.26	0.33 ± 0.12	0.27 ± 0.13	0.50 ± 0.02

with nine folds and validated with the remaining fold) and the testing subset is applied to all models. As FlexCon-C is a semi-supervised approach, it is necessary to use only a small portion of labeled data in this context. For such, only 5% of the training instances remain with the original label, maintaining the stratification of the original data.

For the supervised approaches, we usually selected the default options. However, the following parameters for AWE were changed: the maximum number of classifiers is set up to 10, the chunk size is the same as the training set, and this method generates 100 classifiers to select 10. On the other hand, for the semi-supervised approach, all parameters were set to the same values in [10].

As explained earlier, we need to define the minimum acceptable precision to decide if the classifier ensemble will be updated. In this work, the minimum acceptable accuracy was set up to 70% of the oracle prediction.

VI. EXPERIMENTAL RESULTS

In this section, we describe the results obtained for each method, separated by dataset, using four metrics for evaluation: Accuracy, F-Score, Precision, and Recall, and their respective standard derivations. In addition, we also analyze the behavior of the methods in different batch sizes. Tables II to VI show these results organized by batch size. The highest result for each metric is highlighted in bold in each column, and the light gray shaded cells indicate when our approach achieves a better result compared to other considered methods.

Furthermore, we present a critical difference diagram for each dataset and batch size. Critical difference diagrams are easier to analyze and show clearly which algorithm is better than the others. It uses a post-hoc Friedman test, the Nemenyi test. The leftmost method obtained the lowest ranks and the rightmost method obtained the highest ranks, when the results are ranked. Moreover, methods not covered by a line of critical difference, are statistically different, otherwise the null hypothesis of the Friedman test cannot be refuted. Figures VI-A to VI-E show the critical difference diagram for each dataset, separated by batch size.

A. Adult dataset

Table II presents the results obtained in Adults dataset, for each metric analyzed, using both batch sizes of 500 and 5000 instances. As can be seen from this table, the DyDaSL approach was better than the other approaches in 70% (14 of 20) and in 85% (17 of 20) of cases, respectively.

Therefore, when the methods were trained and tested with batches of 500 instances, the HT method was better than all methods, in addition to being statistically significant compared with all others approaches (Figures 2a and 2b). However, when the batch size grew to 5000, the DyDaSL method obtained the superior results in accuracy and precision metrics. Furthermore, in a statistical point of view, the DyDaSL was similar to the AWE method, but both methods were statistically superior to others.

B. Airline dataset

Table III presents the results obtained in the Airline dataset, for each metric analyzed, using both batch sizes of 500 and

(a) Batch size = 500 (b) Batch size = 5000

Fig. 2. Critical Difference for Adult Dataset

TABLE III
RESULTS FOR AIRLINES DATASET

Method	Accuracy	F-Score	Precision	Recall
batch size = 500				
DyDaSL	0.60 ± 0.08	0.48 ± 0.09	0.46 ± 0.13	0.52 ± 0.05
HT	$\mathbf{0.66 \pm 0.07}$	$\mathbf{0.60 \pm 0.06}$	$\mathbf{0.61 \pm 0.07}$	$\mathbf{0.60 \pm 0.06}$
AC	0.55 ± 0.12	0.53 ± 0.11	0.53 ± 0.13	0.55 ± 0.05
OBA	0.49 ± 0.12	0.37 ± 0.10	0.30 ± 0.13	0.51 ± 0.02
AWE	0.55 ± 0.13	0.35 ± 0.05	0.28 ± 0.07	0.50 ± 0.00
ADACC	0.55 ± 0.13	0.35 ± 0.05	0.28 ± 0.07	0.50 ± 0.00
batch size = 5000				
DyDaSL	0.55 ± 0.11	0.48 ± 0.12	0.46 ± 0.16	0.53 ± 0.04
HT	$\mathbf{0.67 \pm 0.04}$	$\mathbf{0.64 \pm 0.03}$	$\mathbf{0.64 \pm 0.03}$	$\mathbf{0.63 \pm 0.04}$
AC	0.45 ± 0.12	0.30 ± 0.06	0.22 ± 0.06	0.50 ± 0.00
OBA	0.50 ± 0.13	0.33 ± 0.06	0.26 ± 0.08	0.50 ± 0.00
AWE	0.54 ± 0.11	0.36 ± 0.06	0.28 ± 0.07	0.50 ± 0.01
ADACC	0.55 ± 0.12	0.35 ± 0.05	0.27 ± 0.06	0.50 ± 0.00

5000 instances. As can be seen from this table, the DyDaSL approach was better than others in 65% (13 of 20) and in 75% (15 of 20) of cases, respectively. However, we can see that the HT method achieved the best results in all evaluated metrics, regardless the batch size. Consequently, this method was statistically superior to others approaches – Figures 3a and 3b.

C. Electricity dataset

Table IV presents the results obtained in the Electricity dataset, for each metric analyzed, using both batch sizes of 500 and 5000 instances. As can be seen from this table, the DyDaSL approach was better in 60% (12 of 20) of cases when using batch size of 500. On the other hand, when using batch size of 5000, the DyDaSL was the best method in 100% (20 of 20) of cases. Figures 4a and 4b compare the relevance of the methods in a statistical point of view and we can observe that DyDaSL was statistically significant because it had a better ranking than other three methods using batch size of 500 and better than five others methods while using 5000.

D. Gears2C2D dataset

Table V presents the results obtained in Gears2C2D dataset, for each metric analyzed, using both batch sizes of 500 and

TABLE IV
RESULTS FOR ELECTRICITY DATASET

Method	Accuracy	F-Score	Precision	Recall
Batch size = 500				
DyDaSL	0.74 ± 0.11	0.71 ± 0.13	0.72 ± 0.15	0.71 ± 0.12
HT	$\mathbf{0.80 \pm 0.10}$	$\mathbf{0.78 \pm 0.13}$	$\mathbf{0.79 \pm 0.14}$	$\mathbf{0.77 \pm 0.10}$
AC	0.78 ± 0.10	$\mathbf{0.78 \pm 0.10}$	0.78 ± 0.10	$\mathbf{0.77 \pm 0.10}$
OBA	0.51 ± 0.15	0.39 ± 0.18	0.32 ± 0.22	0.54 ± 0.11
AWE	0.58 ± 0.08	0.36 ± 0.03	0.29 ± 0.04	0.50 ± 0.00
ADACC	0.57 ± 0.08	0.37 ± 0.07	0.30 ± 0.09	0.50 ± 0.03
Batch size = 5000				
DyDaSL	$\mathbf{0.73 \pm 0.08}$	$\mathbf{0.72 \pm 0.10}$	$\mathbf{0.73 \pm 0.10}$	$\mathbf{0.72 \pm 0.09}$
HT	0.51 ± 0.15	0.45 ± 0.20	0.42 ± 0.27	0.56 ± 0.11
AC	0.46 ± 0.11	0.37 ± 0.16	0.31 ± 0.21	0.53 ± 0.08
OBA	0.49 ± 0.08	0.33 ± 0.04	0.25 ± 0.05	0.50 ± 0.00
AWE	0.57 ± 0.04	0.36 ± 0.02	0.28 ± 0.02	0.50 ± 0.00
ADACC	0.52 ± 0.08	0.36 ± 0.05	0.27 ± 0.07	0.50 ± 0.00

(a) Batch size = 500 (b) Batch size = 5000

Fig. 4. Critical Difference for Electricity Dataset

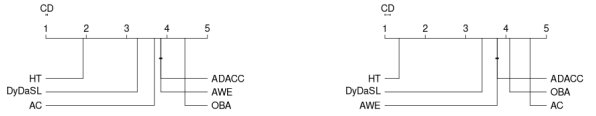
5000 instances. As can be seen from this table, DyDaSL was the best method in 80% (16 of 20) and in 100% (20 of 20), except when using batch size of 500 with the AC method. These results are confirmed from the statistical point of view in Figure 5a.

E. Ug2C3D dataset

Table VI presents the results obtained in Ug2C3D dataset, for each metric analyzed, using both batch sizes of 500 and 5000 instances. As can be seen from this table, the DyDaSL approach was better than others in 80% (16 of 20) and in 100%

TABLE V
RESULTS FOR GEARS2C2D DATASET

Method	Accuracy	F-Score	Precision	Recall
Batch size = 500				
DyDaSL	0.94 ± 0.04	$\mathbf{0.95 \pm 0.03}$	$\mathbf{0.95 \pm 0.03}$	0.94 ± 0.04
HT	0.53 ± 0.11	0.38 ± 0.15	0.31 ± 0.19	0.53 ± 0.10
AC	$\mathbf{0.95 \pm 0.02}$	$\mathbf{0.95 \pm 0.02}$	$\mathbf{0.95 \pm 0.02}$	$\mathbf{0.95 \pm 0.02}$
OBA	0.57 ± 0.16	0.44 ± 0.22	0.38 ± 0.26	0.57 ± 0.15
AWE	0.50 ± 0.02	0.33 ± 0.01	0.25 ± 0.01	0.50 ± 0.00
ADACC	0.50 ± 0.03	0.33 ± 0.03	0.25 ± 0.04	0.50 ± 0.02
Batch size = 5000				
DyDaSL	$\mathbf{0.97 \pm 0.02}$	$\mathbf{0.97 \pm 0.02}$	$\mathbf{0.97 \pm 0.03}$	$\mathbf{0.97 \pm 0.02}$
HT	0.50 ± 0.00	0.33 ± 0.00	0.25 ± 0.00	0.50 ± 0.00
AC	0.50 ± 0.00	0.33 ± 0.00	0.25 ± 0.00	0.50 ± 0.00
OBA	0.50 ± 0.03	0.34 ± 0.05	0.25 ± 0.05	0.50 ± 0.03
AWE	0.50 ± 0.00	0.33 ± 0.00	0.25 ± 0.00	0.50 ± 0.00
ADACC	0.50 ± 0.00	0.33 ± 0.00	0.25 ± 0.00	0.50 ± 0.00



(a) Batch size = 500 (b) Batch size = 5000

Fig. 3. Critical Difference for Airline Dataset

(a) Batch size = 500 (b) Batch size = 5000

Fig. 5. Critical Difference for Gears2C2D Dataset

TABLE VI
RESULTS FOR UG2C3D DATASET

Method	Accuracy	F-Score	Precision	Recall
Batch size = 500				
DyDaSL	0.91 ± 0.08	0.91 ± 0.08	0.92 ± 0.08	0.91 ± 0.08
HT	0.79 ± 0.23	0.73 ± 0.30	0.70 ± 0.34	0.79 ± 0.23
AC	$\mathbf{0.94 \pm 0.06}$	$\mathbf{0.94 \pm 0.06}$	$\mathbf{0.95 \pm 0.06}$	$\mathbf{0.94 \pm 0.06}$
OBA	0.57 ± 0.23	0.56 ± 0.31	0.51 ± 0.35	0.67 ± 0.23
AWE	0.50 ± 0.02	0.33 ± 0.01	0.25 ± 0.01	0.50 ± 0.00
ADACC	0.50 ± 0.03	0.33 ± 0.03	0.25 ± 0.04	0.50 ± 0.02
Batch size = 5000				
DyDaSL	$\mathbf{0.94 \pm 0.07}$	$\mathbf{0.94 \pm 0.07}$	$\mathbf{0.94 \pm 0.08}$	$\mathbf{0.94 \pm 0.07}$
HT	0.57 ± 0.17	0.43 ± 0.23	0.36 ± 0.26	0.57 ± 0.17
AC	0.50 ± 0.00	0.33 ± 0.00	0.25 ± 0.00	0.50 ± 0.00
OBA	0.53 ± 0.11	0.37 ± 0.15	0.29 ± 0.17	0.53 ± 0.11
AWE	0.50 ± 0.00	0.33 ± 0.00	0.25 ± 0.00	0.50 ± 0.00
ADACC	0.52 ± 0.09	0.36 ± 0.13	0.28 ± 0.15	0.52 ± 0.09

(a) Batch size = 500 (b) Batch size = 5000

Fig. 6. Critical Difference for UG2C3D Dataset

(20 of 20) of cases, respectively. Comparing all the methods, DyDaSL was the best except using batch size of 500 with AC method. Figure 6b confirms these results using a statistical analysis.

In general, DyDaSL was better than the other methods in 77.5% (31 of 40), in 70% (28 of 40), in 80% (32 of 40), in 90% (36 of 40), in 90% (36 of 40) for Adult, Airlines, Electricity, Gears2C2D and UG2C3D dataset, respectively. Therefore, DyDaSL, the framework proposed in this work, is a valid approach because it uses only 5% of labeled data and obtains positive results when compared with supervised approaches in 81.5% of cases (163 of 200).

VII. CONCLUSIONS

In this paper we presented a new semi-supervised classification approach for a non-stationary data stream environment. This framework, named DyDaSL, aims to investigate the performance of FlexCon-C method applied to dynamic data stream.

In order to perform a comparative analysis, we carried out experiments with five datasets, using two batches size of instances with 500 and 5000, and five supervised methods: HT, AC, OBA, AWE and ADACC. Four metrics were selected to be analyzed: Accuracy, F-Score, Precision and Recall. Therefore, in order to run DyDaSL as a semi-supervised learning, each batch had 5% of data labeled.

According to the obtained results, the proposed framework had performance equal or higher than all supervised approaches in 81.5%. It is important to emphasize that the DyDaSL performed better than the others methods in 80% (4 of 5) of datasets with batch size of 5000.

By analyzing the statistical tests, we could observe that the performances of the several methods are statistically significant. Then, we observed that DyDaSL were statistically superior in 80% (8 of 10) of the analyzed cases. However, we conclude that the proposed framework is a valid approach, because it had performance equal or better than supervised approaches analyzed in this work.

Future works can investigate other percentages of initially labeled data or evaluate the performance of this framework using other ensemble settings. Also, it is possible to apply DyDaSL combined with others semi-supervised learning algorithms.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] T. M. Mitchell, *Machine Learning*. Boston, Massachusetts, London: McGraw-Hill, 1997.

[2] A. Bouchachia, "Learning with partional supervision," in *Machine Learning*. Hershey PA 17033: Information Science Reference, 2012, vol. 3, pp. 1880–1888.

[3] M. F. A. Hady and F. Schwenker, "Semi-supervised learning," in *Handbook on Neural Information Processing*. Berlin Heidelberg: Springer, 2013, pp. 215–239.

[4] I. A. D. Gunn, A. Arnaiz-González, and L. Kuncheva, "A taxonomic look at instance-based stream classifiers," *Neurocomputing*, 02 2018.

[5] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 23:1–23:36, Mar. 2017.

[6] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woniak, "Ensemble learning for data stream analysis," *Inf. Fusion*, vol. 37, no. C, pp. 132–156, Sep. 2017.

[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014.

[8] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "A survey of classification methods in data streams," in *Data Streams*. Springer, 2007, pp. 39–59.

[9] X. Zhu and A. B. Goldberg, *Introduction to semi-supervised learning*, 3rd ed. San Rafael, California: Morgan & Claypool Publishers, 2009.

[10] K. M. O. Vale, A. M. P. Canuto, A. M. Santos, F. L. Gorgônio, A. M. Tavares, A. C. Gorgônio, and C. T. Alves, "Automatic adjustment of confidence values in self-training semi-supervised method," *International Joint Conference on Neural Networks*, 2018.

[11] F. M. Rodrigues, A. M. Santos, and A. M. P. Canuto, "Using confidence values in multi-label classification problems with semi-supervised learning," *International Joint Conference on Neural Networks*, pp. 1 – 8, 2013.

[12] F. M. Rodrigues, A. M. P. Canuto, and A. M. Santos, "Confidence factor and feature selection for semi-supervised multi-label classification methods," *International Joint Conference on Neural Networks*, pp. 864 – 871, 2014.

[13] Y. Tao, D. Zhang, S. Cheng, and X. Tang, "Improving semi-supervised self-training with embedded manifold transduction," *Transactions of the Institute of Measurement and Control*, vol. 40, no. 2, pp. 363–374, 2018.

[14] A. C. Gorgônio, C. T. Alves, A. J. F. Lucena, F. L. Gorgônio, K. M. O. Vale, and A. M. P. Canuto, "Analysis of the threshold variation of the flexcon-c algorithm for semi-supervised learning," in *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional*. SBC, 2018, pp. 775–786.

[15] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*, 2nd ed. John Wiley & Sons, 2014.

[16] C. C. Aggarwal, *Data classification: algorithms and applications*, 1st ed. Chapman and Hall/CRC, 2014.

[17] L. I. Kuncheva, "A theoretical study on six classifier fusion strategies," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 281–286, Feb. 2002.

[18] M. J. Hosseini, A. Gholipour, and H. Beigy, "An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams," *Knowl. Inf. Syst.*, vol. 46, no. 3, pp. 567–597, Mar. 2016.

[19] W. Zhang and J. Wang, "A hybrid learning framework for imbalanced stream classification," in *2017 IEEE International Congress on Big Data (BigData Congress)*, June 2017, pp. 480–487.

[20] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, Oct 2017.

[21] A. F. J. Costa, R. A. S. Albuquerque, and E. M. dos Santos, "A drift detection method based on active learning," in *2018 International Joint Conference on Neural Networks*, ser. IJCNN 2018. Rio de Janeiro, Brazil: IEEE, 2018.

[22] X. Wang, Q. Kang, M. Zhou, and S. Yao, "A multiscale concept drift detection method for learning from data streams," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Aug 2018, pp. 786–790.

[23] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 97–106.

[24] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with evolving streaming data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 597–612.

[25] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 359–364.

[26] H. Wang, W. Fan, P. S. Yu, and 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*, 2003, pp. 226–235.

[27] G. Jaber, A. Cornuéjols, and P. Tarroux, "A new on-line learning method for coping with recurring concepts: The adacc system," in *Neural Information Processing*, M. Lee, A. Hirose, Z.-G. Hou, and R. M. Kil, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 595–604.

[28] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, 2010.

[29] G. Dimitoglou, J. A. Adams, and C. M. Jim, "Comparison of the c4.5 and a naive bayes classifier for the prediction of lung cancer survivability," *Journal of Computing*, vol. 4, no. 8, 2012.

[30] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. New York: CHAPMAN & HALL/CRC, 1984.

[31] W. W. Cohen, "Fast effective rule induction," in *Machine Learning*, ser. ML95. Tahoe City, California, USA: Morgan Kaufmann Publishers, 1995, pp. 115–123.

[32] C. G. Atkeson, A. W. Moore, and S. Schaau, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1, pp. 11–73, Feb 1997.

[33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[34] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[35] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2015, pp. 873–881.