C-SMOTE: Continuous Synthetic Minority Oversampling for Evolving Data Streams

Alessio Bernardo
DEIB - Politecnico di Milano
Milano, Italy
alessio.bernardo@polimi.it

Heitor Murilo Gomes
University of Waikato
Hamilton, New Zealand
heitor.gomes@waikato.ac.nz

Jacob Montiel
University of Waikato
Hamilton, New Zealand
jmontiel@waikato.ac.nz

Bernhard Pfahringer
University of Waikato
Hamilton, New Zealand
bernhard@waikato.ac.nz

Albert Bifet
University of Waikato, New Zealand
LTCI, Télécom ParisTech, France
abifet@waikato.ac.nz

Emanuele Della Valle
DEIB - Politecnico di Milano
Milano, Italy
emanuele.dellavalle@polimi.it

***Abstract*—Streaming Machine Learning (SML) studies single-pass learning algorithms that update their models one data item at a time given an unbounded and often non-stationary flow of data (a.k.a., in presence of concept drift). Online class imbalance learning is a branch of SML that combines the challenges of both class imbalance and concept drift. In this paper, we investigate the binary classification problem of rebalancing an imbalanced stream of data in the presence of concept drift, accessing one sample at a time. We propose Continuous Synthetic Minority Oversampling Technique (C-SMOTE), a novel rebalancing meta-strategy to pipeline with SML classification algorithms. C-SMOTE is inspired by the popular SMOTE algorithm but operates continuously. We benchmark C-SMOTE pipelines on ten different groups of data streams. We bring empirical evidence that models learnt with C-SMOTE pipelines outperform models trained on imbalanced data stream without losing the ability to deal with concept drifts. Moreover, we show that they outperform other stream balancing techniques from the literature.**

***Index Terms*—Streaming data, Concept Drift, Balancing, Binary Classification**

I. Introduction

Nowadays, data abound as a multitude of smart devices, such as smartphones, wearables, computers, and Internet of Things (IoT) sensors produce massive, continuous and unbounded flows of data namely *data streams*. This poses several challenges to Machine Learning (ML). First of all, 1) it is impossible to load a data stream as a whole in memory because it is infinite and, 2) it is often non-stationary, i.e. there is a constant presence of concept drift [1]. This means that the function which generates instances at time step t does not need to be the same that generates instances at time step $t+1$. ML practitioners address the first challenge by transforming the stream into a sequence of *batches* and retraining a model as new batches are available. As a consequence, another challenge arises: the need to retrain a model within strict time constrains. Moreover, dealing with concept drift requires an algorithm that can adjust quickly to changing conditions. The risk is that the model trained up to that point does not fit anymore the data received after the change. The traditional ML techniques are not designed to monitor concept drift, so their models are prone to introduce classification errors when it happens. In recent years, a different approach was introduced to tackle time, memory, and concept drift problems that affect batch methods. It is often referred to as the *Streaming Approach* (a.k.a. *Data Stream Mining* or *Online Learning*) [2]. Ideally, every time a new instance arrives, a streaming learner inspects it without saving in memory, updates the model incrementally, and it is able to predict at each moment. In this way, the data storage problem is avoided because the new instance is immediately discarded after the training phase, and the time problem is addressed by updating the model incrementally, one instance at a time, without the need of retraining it from the beginning. Moreover, this approach includes some techniques able to detect when concept drift occurs and to adapt the model accordingly to it.

In this paper, we focus on *streaming binary classification problems* involving concept drift and imbalanced data. It is a relevant topic since classification techniques tend to focus on the most representative instances, thus neglecting minority instances [3]. This may prevent or delay the discovery of any existing patterns in the minority class. Moreover, due to the concept drift occurrence, classes can swap, i.e. all the samples labeled as minority (majority) class before the concept drift occurrence get labeled as majority (minority) class after it.

The combined problems of concept drift and class imbalance are found in many real-world applications. A current example could be the diagnosis of COVID-19 disease starting from patients' symptoms. This new disease represents the minority class and it is difficult to diagnose because its symptoms are similar to a seasonal flu. At the beginning, the minority instances are rare and could be ignored, but, as time went on, they become increasingly more important and, so, the correct classification importance increases, too. Another application could be spam filtering. It is a typical classification problem involving class imbalance and concept drift. The spam class is the minority one and suffers from a higher misclassification

cost. Moreover, the spammers are actively working on how to break through the filter. It means that the adversary actions are adaptive. For example, one of the spamming behaviors is to change email content and presentation in disguise, implying a concept drift.

To address these problems, techniques to rebalance the training dataset were proposed in the batch scenario. One of the most famous and powerful [3] is the *Synthetic Minority Oversampling Technique* (SMOTE) [4]. All of them need the entire data batch to rebalance it, but, in the streaming approach, such a batch is not available.

In this paper, we investigate the following **research question**: *if one has a binary classification problem on an imbalanced data stream in presence of concept drift, should continuous rebalance the data stream before applying a SML classification algorithm?*

The main contributions of this paper are:

- C-SMOTE, a meta-strategy inspired by the well known SMOTE technique (applicable only to batches) that can be pipelined with any data stream classifier;
- an analysis of the performances of C-SMOTE when pipelined with different data stream classifiers on ten groups of dynamic and imbalanced data streams (eight real and two synthetic) showing at least an improvement of the minority class performances; and
- a comparison between C-SMOTE and four other strategies able to deal with class imbalance in the streaming scenario, namely Adaptive Random Forest with Resampling (ARF_{RE}) [5], RebalanceStream (RB) [6], undersampling-based Online Bagging (UOB) and oversampling-based Online Bagging (OOB) [7] showing that there is at least one algorithm pipelined with C-SMOTE that outperforms the state of the art one.

The remainder of this paper is organized as follows. Section II describes the investigated problem and presents techniques able to handle it. Section III describes the method proposed. Section IV introduces the related works. Section V presents our research hypotheses, introduces the datasets used in the experiments, and shows the evaluation results. Finally, Section VI discusses the conclusions and outlines directions for future research.

II. Sampling Techniques for Class Imbalance

An unequal distribution between the classes characterizes imbalance data. Since the instances contained in the minority class(es) rarely occur, the patterns for classifying these classes tend to be rare, undiscovered, or ignored. This can be a problem during the training phase: in fact, the model can analyze a sample and cleverly decide that the best thing to do is predicting the majority class without examining any of its features.

He and Garcia [3] characterize the approaches to handle class imbalance as: sampling techniques, cost-sensitive learning, kernel-based methods, and active learning methods. This work focuses on sampling techniques because they allow creating a new meta-strategy that can be run during the preprocessing phase, regardless of the streaming method chosen. The user is free to choose the SML algorithm to use. Sampling techniques change the data distribution so that standard algorithms focus on the cases that are more relevant to the user.

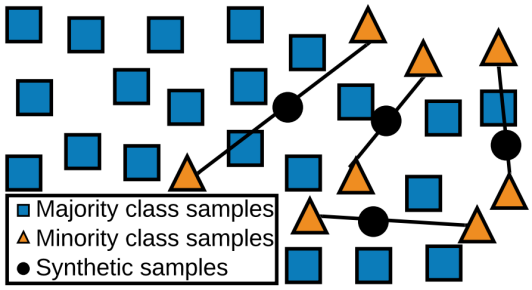

Fig. 1. SMOTE example.

Sampling techniques includes *oversampling* and *undersampling* methods. *Oversampling* methods increase the number of minority class instances through the creation of synthetic instances, until classes are balanced or nearly balanced. After the creation of new instances, the minority class, which was originally underrepresented, may exert a greater influence on learning and future predictions. *Undersampling* methods, on the other hand, aim at reducing the number of instances from the majority class by removing instances from this class. They often act in two ways, by removing noisy instances, or simply reducing instances by some heuristics or even randomly. Both methods introduce their own set of drawbacks that can worsen the learning phase [3]. In case of undersampling, removing instances from the majority class may cause important concepts loss. In case of oversampling, since data are replicated or synthetically generated, the drawback is that multiple iterations over the same instances can result in overfitting.

A popular balancing techniques is SMOTE [4], shown in Fig. 1. It is an oversampling technique, therefore it synthetically generates instances for the minority class to balance the training data. For each minority class sample x_i (orange triangle), SMOTE finds its K-nearest neighbours among the other minority class samples, it randomly chooses one $\hat{x}_i$ from them, and its distance from x_i is multiplied by a random number $\delta \in [0, 1]$. The resulting new sample x_n (black circle) is located between x_i and the selected neighbor $\hat{x}_i$.

In general SMOTE has been shown to improve classification. Moreover, it may also show drawbacks related to the way it creates synthetic samples. Specifically, SMOTE generates new samples without considering the neighbour examples, which increases the occurrence of overlapping between classes. To this end, various adaptive sampling methods have been proposed to overcome this limitation. Some representative works include the Borderline-SMOTE, Adaptive Synthetic Sampling and SMOTE+Tomek algorithms, all presented in [3].

Moreover, as a sampling technique, SMOTE caches the entire dataset in memory. This approach is against the basic

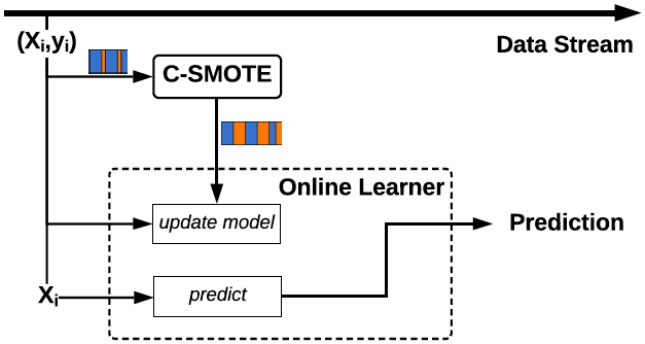

Fig. 2. Architecture of C-SMOTE meta-strategy pipelined with an Online Learner.

principles of the data stream paradigm that states that a sample can be inspected only once, as fast as possible, and then discarded. In Section III, we explain how to overcome this problem.

III. C-SMOTE

This section describes the proposed meta-strategy Continuous-SMOTE (C-SMOTE), shown in Fig. 2, inspired by the SMOTE technique. C-SMOTE is designed to rebalance an imbalanced data stream and it can be pipelined with any streaming classification techniques. C-SMOTE stands for Continuous-SMOTE, meaning that the new SMOTE version is applied continuously.

As said before, the real problem is the lack of the entire data during the rebalance phase. Moreover, it is impossible to store every new sample in memory until the data stream ends for two reasons: 1) the data streams are assumed infinite, and 2) this would be against the stream paradigm approach. The solution is to use ADWIN [8] to save only the recently-seen samples and using the minority class samples stored in the ADWIN window to apply SMOTE. ADWIN keeps a variable-length window of recently seen items and it is able to automatically detect and adapt its window to the current rate of change. ADWIN, in (1), uses a threshold called $delta$ in order to automatically configure the error with two levels, named warning and change level. The warning level is identified using $delta \times 10$, while the change level is identified using $delta$. Since $delta$ appears to the denominator, using $delta \times 10$ will produce a lower value than using $delta$. So the warning level will occur before the change one. n is the width of the window in that moment.

$$levelError = \log(\frac{2 \times \log n}{delta}) \quad (1)$$

ADWIN monitors the error over the data in the window. If the error becomes greater than a warning level, ADWIN assumes that a concept drift starts to occur and it starts collecting new samples in a new window, too. If the error becomes greater than the change level, ADWIN assumes that a concept occurred and it substitutes the old window with the new one.

In this way, the online version of SMOTE is always applied on data that are consistent with the current concept, and

Algorithm 1:
Symbols:
$minSizeMinority$: Minimum number of minority class instances to allow the rebalancement procedure;
l: Streaming learner;
t: Balance ratio to achieve;
S: Binary classification data stream;
S_0: Number of samples contained into the class 0;
S_1: Number of samples contained into the class 1;
$S_{\underline{0}}$: Number of class 0 synthetically generated samples;
$S_{\underline{1}}$: Number of class 1 synthetically generated samples;
$S_{\underline{gen}}$: Number of synthetic samples generated by each sample in W;
W: Sliding window with all the recently-seen samples;
W_{label}: Sliding window of bits stating if a sample belongs to class 0 or 1;
$adwin$: Drift detector;
$imbalanceRatio$: Imbalance ratio of the window W;

Function C-SMOTE $(minSizeMinority, l, t, S)$**:**
  $W, W_{label} \leftarrow \varnothing$
  $S_0, S_1, S_{\underline{0}}, S_{\underline{1}} \leftarrow 0$
  $S_{\underline{gen}} \leftarrow \varnothing$
  $imbalanceRatio \leftarrow 0$
  $adwin \leftarrow \varnothing$
  while $hasNext(S)$ **do**
    $X, y \leftarrow next(S)$
    $prequentialEvaluation(X, l)$
    $train(X, y, l)$
    $W \leftarrow add(X, y)$
    $updateWindows(X, y, W_{label})$
    $updateCounters(y, S_0, S_1)$
    $adwin \leftarrow add(y)$
    $checkConceptDrift(adwin, W, W_{label}, S_0, S_1, S_{\underline{0}}, S_{\underline{1}}, S_{\underline{gen}})$
    $W_{min}, S_{min}, S_{\underline{min}} \leftarrow selectMinorityClass(W, W_{label}, S_0, S_1, S_{\underline{0}}, S_{\underline{1}})$
    $W_{maj}, S_{maj}, S_{\underline{maj}} \leftarrow selectMajorityClass(W, W_{label}, S_0, S_1, S_{\underline{0}}, S_{\underline{1}})$
    if $checkMinSize(minSizeMinority, S_{min})$ **then**
      $imbalanceRatio \leftarrow ratio(S_{min}, S_{maj}, S_{\underline{min}}, S_{\underline{maj}})$
      while $t > imbalanceRatio$ **do**
        $\hat{X}, \hat{y} \leftarrow newSample(W_{min}, S_{\underline{gen}})$
        $S_{\underline{min}} \leftarrow S_{\underline{min}} + 1$
        $train(\hat{X}, \hat{y}, l)$
        $imbalanceRatio \leftarrow ratio(S_{min}, S_{maj}, S_{\underline{min}}, S_{\underline{maj}})$
      end
    end
  end
End Function

the new generated samples will be consistent to it, too. The pseudo-code of C-SMOTE is presented in Algorithm 1.

Given a stream S $\{(X_1, y_1), (X_2, y_2), ...\}$, where X_i is a feature vector, and y_i is the class label, C-SMOTE keeps two sliding windows of samples (Line 2): W contains all the samples related to the current concept, while W_{label} is an array of bits stating if the corresponding samples in W belong to the class 0 or 1. The variable $adwin$ (Line 6) is the change detector that, using the class value of the sample in input, is responsible for keeping the two windows consistent with concept drift. It is not its responsibility to adapt the model to concept drift. The pipelined streaming classifier model may or may not incorporate a change detector. C-SMOTE has also five counters (Line 3-4): S_0 and S_1, respectively, count the number of instances in each class; $S_{\underline{0}}$ and $S_{\underline{1}}$ count, respectively, the number of instances of class 0 and 1 generated by C-SMOTE; $S_{\underline{gen}}$, for each sample i in W, keeps track of how many times i is used to introduce synthetic samples. The reason why C-SMOTE keeps track of the instances of both classes is that, after a concept drift, in W, the samples of class 1 may be less than the ones of class 0. In this case, the samples of class 1 are considered as minority class and they are used by SMOTE to introduce new synthetic samples.

Every time a new sample (X, y) is available, the prequential evaluation approach [9] is applied and then, the pipelined learner l is trained (Lines 8-10). After that, the new sample (X, y) is saved in W. The function $updateWindows$ at Line 12, depending of the class value y, adds a new bit into W_{label} (0 if $y = 0$ or 1 if $y = 1$), while the function $updateCounters$ at Line 13, depending of the class value y, increments the relative counter S_0 or S_1. Then, $adwin$ is updated with the class value y of the sample in input (Line 14) and the $checkConceptDrift$ function uses $adwin$ to check the presence of a concept drift change (Line 15). If a concept drift has occurred, the function adapts and reduces W in according with the window maintained by $adwin$. Accordingly to W, it also updates W_{label} and the counters S_0, S_1. Moreover, if the instances removed from W have been used to generate some new synthetic samples, the function updates the counter of instances generated for that class $S_{\underline{0}}$ or $S_{\underline{1}}$, too. The next step identifies the real minority (Line 17) and majority (Line 18) class and saves the corresponding window and counters into, respectively, W_{min}, S_{min}, $S_{\underline{min}}$ and W_{maj}, S_{maj}, $S_{\underline{maj}}$. Then, Line 19 checks if the instances in W_{min} are at least equal to the hyperparameter $minSizeMinority$ specified by the user. If it is true, the algorithm can proceed with the rebalance phase, otherwise it waits another sample in input. If $minSizeMinority$ is equal to -1, no check is performed on the minority class size, so the $checkMinSize$ function will always return true. At Line 20, the actual $imbalanceRatio$ (2) between the number of minority and majority instances stored in W is calculated.

$$imbalanceRatio = \frac{S_{min} + S_{\underline{min}}}{S_{min} + S_{maj} + S_{\underline{min}} + S_{\underline{maj}}} \quad (2)$$

If $imbalanceRatio$ is less than the hyperparameter balance ratio to achieve t, which means that W is imbalanced, some new synthetic samples $(\hat{X}, \hat{y})$ are generated until the $imbalanceRatio$ is equal to t, which means that W is balanced now (Lines 22-25). In particular, the function $newSample$, at Line 22, is responsible to generate a new synthetic sample. This is different from the original SMOTE. Before starting to generate new instances, SMOTE calculates the number of instances to introduce for each minority sample in the batch and then it starts to execute. C-SMOTE randomly chooses one sample (X, y) from W_{min} and uses it to apply SMOTE. Then, it increments the counter $S_{\underline{gen}}$ for (X, y) and the function, in this rebalance phase, does not use anymore (X, y) to generate other synthetic instances. The only exception is when the number of synthetically generated instances needed to rebalance W is greater than the number of the minority class sample S_{min}. Only in this case, the function reuses (X, y) multiple times in the same rebalance phase. So, in our continuous version, not all the minority class instances are used to generate new instances. After that, at Line 23, $S_{\underline{min}}$ is updated, the new instance $(\hat{X}, \hat{y})$ is used to train the learner l and the new $imbalanceRatio$ is calculated.

IV. Related Work

In the literature, there are SML approaches able to learn from imbalanced data stream in presence of concept drift. They are commonly categorized into two major groups: passive versus active approaches, depending on whether an explicit drift detection mechanism is employed. Passive approaches train a model continuously without an explicit trigger reporting the drift, while active approaches determine whether a drift has occurred before taking any actions. Examples of passive approaches are RLSACP, ONN, ESOS-ELM, an ensemble of neural network, OnlineUnderOverBagging, OnlineSMOTE-Bagging, OnlineAdaC2, OnlineCSB2, OnlineRUSBoost and OnlineSMOTEBoost, while ARF_{RE}, RebalanceStream, OOB, UOB, WEOB1 and WEOB2 are considered active approaches.

RLSACP [10] is inspired from the recursive least square (RLS) filter error model. In the proposed error model, non-stationarity is handled with the forgetting factor (k) in the RLS error model while for handling class imbalance, two adaptive error weighting strategies are proposed. In the first one, error weights are adapted based on classifier results in different classes. In the second one, the number of instances in the minority and majority class are counted in a fixed window of the most recent samples and the weights are assigned accordingly.

ONN [11] is a similar approach. It is an online Multi Layer Perceptron model composed by two parts. The former is a forgetting function for handling concept drift while the latter is an error weighting function for handling class imbalance.

EONN [12] is an online ensemble neural network model composed by two layers. The first layer is a cost-sensitive neural network for handling class imbalance, while the second layer contains a method for weighting classifiers of the ensemble.

TABLE I
THE PRINCIPAL CHARACTERISTICS OF THE RELATED WORKS COMPARED TO C-SMOTE.

Method	Classifier type	Approach type	Approach for CD	Approach for Class Imbalance
RLSACP [10]	Single	Passive	Forgetting factor	Cost weight
ONN [11]	Ensemble	Passive + Active module	Forgetting factor	Cost weight
EONN [12]	Ensemble	Passive	Weighted ensemble	Cost weight
ESOS-ELM [13]	Ensemble	Passive	Weighted ensemble	Cost weight
OnlineUnderOverBagging [14]	Ensemble	Passive	Weighted ensemble	Undersampling + Oversampling
OnlineSMOTEBagging [14]	Ensemble	Passive	Weighted ensemble	SMOTE
OnlineAdaC2 [14]	Ensemble	Passive	Weighted ensemble	Cost weight
OnlineCSB2 [14]	Ensemble	Passive	Weighted ensemble	Cost weight
OnlineRUSBoost [14]	Ensemble	Passive	Weighted ensemble	Undersampling
OnlineSMOTEBoost [14]	Ensemble	Passive	Weighted ensemble	SMOTE
ARF_{RE} [5]	Ensemble	Active	ADWIN	Cost weight
RB [6]	Multiple (4)	Active	ADWIN	SMOTE
OOB [7]	Ensemble	Active	Weighted ensemble	Oversampling + Cost weight
UOB [7]	Ensemble	Active	Weighted ensemble	Undersampling + Cost weight
WEOB1/WEOB2 [15]	Ensemble	Active	Weighted ensemble	Cost weight
C-SMOTE	**Meta-strategy**	**Left to pipelined algorithm**	**Left to pipelined algorithm**	**SMOTE + ADWIN**

Another passive technique is ESOS-ELM [13]. It is an ensemble approach that, for tackling class imbalance, resamples the data using fixed weights to train each classifier with approximately equal number of majority and minority class samples. Instead, for tackling concept drift, it uses a voting weights system according to the geometric-mean (G-mean) performance metric. ESOS-ELM has also an active module to handle recurring concept drift.

OnlineUnderOverBagging, OnlineSMOTEBagging, OnlineAdaC2, OnlineCSB2, OnlineRUSBoost and OnlineSMOTEBoost [14] are the online extensions of the popular batch cost-sensitive ensemble learning algorithms UnderOverBagging, SMOTEBagging, AdaC2, CSB2, RUSBoost and SMOTEBoost respectively. The main challenge for adapting them to the online settings resides in finding a way to embed costs into online ensembles for boosting algorithms without having all the data. They reformulate the batch cost-sensitive boosting algorithms in a way that there is no normalization step at each iteration, and then to incrementally estimate the quantities embedded with the cost setting in the online learning scenario. Whereas cost sensitivity in the batch setting is achieved by different resampling mechanisms, in the online ensembles it is achieved by manipulating the parameters of the Poisson distribution for different classes.

ARF_{RE} [5] is an extension of the original ARF [16] algorithm. ARF_{RE} resamples instances based on the current class label distribution, such that it adapts the weights of the Poisson distribution to simulate a balance of the instances to the base models of the forest. To calculate the weights (3), it uses S_c and S_n that represents the number of instances from class c and the total number of instances observed on the stream, respectively.

$$weight(S_c, S_n, \lambda) = \frac{100 - \frac{S_c \times 100}{S_n}}{100} \times Poisson(\lambda) \qquad (3)$$

Effectively, if a sample is part of the minority class, its weight used during the training phase will be increased in comparison to a sample from the majority class.

RebalanceStream (RB) [6] uses ADWIN [8] to detect concept drift in the stream by training four models m_1, m_2, m_3, and m_4 in parallel: m_1 is trained with the original samples in input; m_2 uses the samples collected and rebalanced using SMOTE from the beginning to a change (when the last concept drift occurred); m_3 uses the samples collected from a warning (when the most recent concept drift started to occur) to a change; and m_4 uses the same data of m_3 but rebalanced using SMOTE. After a change, the best model among them is chosen and it is used to continue the execution according to k-statistics.

Oversampling-based Online Bagging (OOB) and undersampling-based Online Bagging (UOB) [7] are other two resampling-based ensemble methods. When class imbalance is detected, oversampling or undersampling embedded in Online Bagging [17] is triggered to either increase the chance of training minority class examples or reduce the chance of training majority class examples. If the new example (X, y) belongs to one of the minority classes ($y \in Ymin$), OOB will tune the parameter λ of Poisson distribution to $1/w_k$, which indirectly increases the number of copies of the current example for training. In other words, it combines oversampling with Online Bagging. If (X, y) belongs to one of the majority classes ($y \in Ymaj$), UOB will set λ to $(1 - w_k)$. Training examples from the majority class will be undersampled. Some performance analysis [15] on OOB and UOB show that UOB is a better choice than OOB in terms of minority-class Recall and G-mean. However, it has some weaknesses when the majority class in the data stream turns into the minority class. OOB is more robust against changes. To combine the strength of OOB and UOB, WEOB1 and WEOB2 [15], are proposed. They are based on the idea of ensembles, which train and maintain both OOB and UOB. A weight is maintained for each of them, adjusted adaptively according to their current performance measured by G-mean. It evaluates the degree of inductive bias in terms of a ratio of positive accuracy and negative accuracy. Their combined weighted vote will decide the final prediction. WEOB1 and

WEOB2 differ in the weight adjusting strategy that they use.

All the described methods with their principal characteristics are summarized and compared to C-SMOTE in Table I.

V. Experimental Evaluation

Section V-A discusses the experimental settings, Section V-B introduces the research hypotheses, while Section V-C presents and discusses the results of our experiments.

A. Experimental Settings

To empirically evaluate the new C-SMOTE meta-strategy, we use ten groups of data streams: two groups are synthetic and eight are real. Each group is composed by ten different versions of the same dataset. In case of real data streams, the original version is shuffled ten times with different seed values to artificially introduce some concept drift. In case of synthetic data streams, since the original version is already generated with some concept drift, each new version is directly generated using a different seed value. So, we performed in total 100 tests. No K-fold cross validation is performed. The characteristics of the original versions of all the data streams are summarized in Table II. For convenience, in all data streams, the *minority* class is always the class *0*, while the *majority* one is the class *1*.

The synthetic data stream SYN is generated through the RandomRBFGeneratorDrift in the MOA framework [2]. It is composed by 10^6 samples and it uses 50 centroids, a speed change of 10^{-7}, 10 numerical attributes and 2 classes. To change the imbalance level, the class ratio is randomly chosen from values between (0.6;0.4) and (0.9;0.1) and it is abruptly changed every n numbers of rows. The number of rows n between two consecutive changes is randomly sampled from a Gaussian model with mean $\mu = 25,000$ and variance $\sigma^2 = 100$. Table II reports the final imbalance ratio. SYN-CD is synthetically generated through ELKI [18]. It is composed by 10 clusters created by a normal distribution. Each one of them has different parameters, in order to introduce 3 concept drifts. The dataset has 10^5 samples, 10 numerical attributes and 2 classes. The Ann-Thyroid (ANN) data stream [19] determines whether a patient referred to the clinic is hypothyroid. It has three classes (hyper-function, subnormal functioning, not hypothyroid), but to introduce a significant imbalance level the first two classes are treated as minority class, while the last one as majority class. In the Bank Marketing (BNK) data stream [20], the classification goal is to predict if the client will subscribe a term deposit. The Credit Card Fraud Detection (FD) data stream [21] is highly imbalanced and it is usually used for outlier detection. The task is to classify if a transaction is fraudulent or not. The aim of the IEEE-CIS Fraud Detection dataset (CIS-FD) is the same of the previous one, but the data come from the IEEE Computational Intelligence Society (IEEE-CIS)[1]. The number of instances and the dimension are different, too. The Give Me Some Credit (GMSC) data stream predicts whether a loan should be granted. The Page Blocks (PB) data stream[2] classifies all the blocks of the page layout of a document that has been detected by a segmentation process. The Image Segmentation (IS) data stream[2] contains image data describing seven outdoor images (grass, path, window, cement, foliage, sky, brick face). To imbalance the data stream, all the grass images are treated as minority class, while the other ones as majority class. In the Statlog Shuttle (SLOG) data stream [19], the task is to decide what type of control of the vessel should be employed.

We evaluate the predictive performance using the prequential evaluation approach [9]. Following [3], we use metrics extracted from the computation of a confusion matrix. The model saves the instances that are correctly classified as number of True Positives (TP) and True Negatives (TN), while those that are wrongly classified are saved as number of False Positives (FP) and False Negatives (FN). In particular, we use the Recall (4) and F1-Measure (5) metrics on each class: Recall[0], F1-Measure[0] for minority class and Recall[1], F1-Measure[1] for majority class. We do not use the Accuracy (6) because it is not reliable in case of imbalanced dataset due to the impact of the least-represented, and possibly more important, examples is reduced when compared to that of the majority class.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - Measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{5}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{6}$$

We tested Adaptive Random Forest (ARF) [16], Naive Bayes (NB), Hoeffding Adaptive Tree (HAT) [22], K-Nearest Neighbor (KNN) and Temporally Augmented Classifier (SWT) [23] with ARF as base learner. We pipeline these algorithms with the new C-SMOTE meta-strategy and compare against the stand-alone versions. We also compare the ARF_{RE} [5] technique to ARF pipelined with C-SMOTE and the RB [6] strategy to SWT pipelined with C-SMOTE. We show the OOB and UOB [7] results compared to the best technique pipelined with C-SMOTE, too. Unfortunately, the implementations of the other algorithms cited in Section IV are unavailable or do not work[3].

All the experiments are made using the MOA framework [2] with default hyperparameters values for all the techniques involved. The only parameter that we set in C-SMOTE is $minSizeMinority$, the minimum number of minority class samples stored into the window to allow the rebalance procedure. We use C-SMOTE with $minSizeMinority = 10$ as it was the top performer among [10, 100, 500, 1000] in our hyper-parameter analysis.

Reported Recall r_c and F1-Measure f_c values of the class c for an algorithm a, using a group of data streams d_n with

[1] https://cis.ieee.org/

[2] https://sci2s.ugr.es/keel/imbalanced.php

[3] In a future extended version of this paper, we will implement all the unavailable methods

TABLE II
THE CHARACTERISTICS OF THE DATA STREAMS USED IN THE EXPERIMENTS. THE POSSIBLE TYPES ARE R FOR REAL AND S FOR SYNTHETIC. THE DIMENSION COLUMN REPRESENTS THE NUMBER OF NUMERIC/NOMINAL FEATURES.

Data Stream	# Instances	Dimension	Type	% Min (0)	% Maj (1)	# Drifts
SYN	1,000,000	10 / 0	S	26.23	73.77	7-18
SYN-CD	100,000	10 / 0	S	5.00	95.00	3
ANN	7,200	6 / 0	R	7.40	92.60	0
BNK	45,211	7 / 9	R	11.70	88.30	35
FD	284,807	30 / 0	R	0.17	99.83	0
CIS-FD	100,000	383 / 49	R	4.00	96.00	0
GMSC	150,000	10 / 0	R	6.68	93.32	0
PB	5,472	10 / 0	R	10.20	89.80	14
IS	2,308	20 / 0	R	14.20	85.80	0
SLOG	58,000	9 / 0	R	21.00	79.00	0

$n = [1..10]$, are calculated as follows. For each version d_i of the data stream contained in the group n, Recall r_{ci} and F1-Measure f_{ci} values are calculated. The final Recall r_c and F1-Measure f_c results are the average of all the r_{ci} and f_{ci} values, with $i = [1..10]$. To evaluate the dispersion, we calculated the standard deviation of all the r_{ci} and f_{ci} values, too.

B. Research Hypotheses

We formulate our hypotheses as follows:

- *Hp.1*: The Recall[0] and F1-Measure[0] of the minority class when applying C-SMOTE are better, in at least one algorithm, than those of the state-of-the-art.
- *Hp.2*: The Recall[1] and F1-Measure[1] of the majority class when applying C-SMOTE are better, in at least one algorithm, than those of the state-of-the-art.

C. Results and Discussion

In this section, we discuss the results presented in Tables III and IV for the minority class, and in Tables V and VI for the majority class.

We start the discussion from the minority class. In the ANN and GMSC data streams, both Recall and F1-Measure results of all the algorithms pipelined with C-SMOTE outperform the results of those without it. A high F1-Measure value means that C-SMOTE improves the Precision, too. In the BNK data stream, only the F1-Measure of the base NB technique is better than the C-SMOTE one. It means that C-SMOTE pipelined with NB improves the Recall but not the Precision w.r.t. NB. The same is observed in the CIS-FD dataset with HAT.

In the FD data stream, the base versions of ARF and HAT outperform those pipelined with C-SMOTE ones. This can be due to the extreme imbalance of the data stream, which is usually used for outlier detection. In this case, SMOTE in particular and over-sampling techniques in general, might not be suitable. It is conceivable that the minority class samples are used multiple times to generate synthetic samples, harming learning by introducing noise.

In the PB, IS, and SLOG data streams, the F1-Measure of the stand-alone KNN algorithm is better than the one of C-SMOTE pipelined with KNN, even if the C-SMOTE KNN Recall is better than the base one. The reason is that the Recall gain is smaller than the Precision loss. With high imbalanced data streams, correctly classifying a minority class sample is difficult because the majority class samples tend to be common among the k nearest neighbors.

In the SYN data stream, only the NB algorithm pipelined with C-SMOTE outperforms its baseline, while, in the SYN-CD data stream, only the HAT and NB algorithms pipelined with C-SMOTE outperform their baselines.

Comparing C-SMOTE to ARF_{RE}, the former pipelined with ARF outperforms the latter in all the experiments. The RB strategy outperforms C-SMOTE pipelined with SWT_{ARF} only in ANN data stream and only in the Recall measure. It means that in all the other cases, the C-SMOTE meta-strategy improves Precision too. Furthermore, our results prove that in case of data streams, introducing new synthetic samples as soon as the imbalance ratio changes is better rather than waiting a concept drift and introducing them all together, as RB does. Regarding OOB and UOB, for each data stream, we select the best C-SMOTE meta-strategy and we compare it against OOB and UOB. The OOB or UOB Recall results are slightly better than those of C-SMOTE, but, in most cases, the C-SMOTE F1-Measure results are better than the OOB and UOB ones. This means that C-SMOTE improves the Precision.

For the majority class results, we observe that in all the datasets tested the base versions are slightly better than the C-SMOTE ones. Comparing C-SMOTE to ARF_{RE}, the former pipelined with ARF outperforms the latter only in the F1-Measure results. RB strategy is outperformed in both Recall and F1-Measure. OOB and UOB are slightly better than the C-SMOTE results only in four data streams (SYN, BNK, CIS-FD, GMSC).

We calculate the increment/decrement inc for each data stream d, for both classes c and metrics m, and for each couple of algorithms alg (baseline and corresponding C-SMOTE version). The single $inc_{d,m,c,alg}$ for a fixed class, metric and couple of algorithms (with and without C-SMOTE) is calculated as in (7), where $res_{m,d,c,algCS}$ is the 10-run average m of the c class results achieved by an algorithm pipelined with C-SMOTE using d dataset and $res_{d,m,c,algBase}$ is the 10-run average m of the c class results achieved by the corresponding algorithm without C-SMOTE using d dataset.

$$inc_{d,m,c,alg} = \frac{res_{m,d,c,algCS} - res_{d,m,c,algBase}}{res_{d,m,c,algBase}} \times 100 \quad (7)$$

TABLE III

RECALL OF THE MINORITY CLASS RESULTS OF ALL THE DATASETS. CS MEANS THAT, THE ALGORITHM IS USED PIPELINED WITH THE C-SMOTE META-STRATEGY. RB STANDS FOR THE REBALANCESTREAM STRATEGY. RESULTS IN BOLD ARE THE BEST FOR THAT ALGORITHM, WHILE RESULTS UNDERLINED ARE THE BEST AMONG ALL THE ALGORITHMS ON THAT DATASET.

Recall[0] %															
Dataset	***CS ARF***	**ARF**	**ARF$_{RE}$**	***CS HAT***	**HAT**	***CS NB***	**NB**	***CS KNN***	**KNN**	***CS SWT$_{ARF}$***	**SWT$_{ARF}$**	**RB**	***BEST CS***	**OOB**	**UOB**
SYN	84.81 ± 1.01	**84.95 ± 1.30**	67.07 ± 3.05	82.20 ± 1.07	**82.21 ± 0.99**	**66.22 ± 0.29**	39.62 ± 2.38	**91.42 ± 0.23**	86.07 ± 1.04	82.30 ± 1.40	**84.91 ± 1.39**	71.95 ± 2.84	91.42 ± 0.23	92.42 ± 0.22	**93.81 ± 0.30**
SYN-CD	99.79 ± 0.03	**99.88 ± 0.01**	99.30 ± 0.00	**99.71 ± 0.02**	97.51 ± 0.08	**18.75 ± 13.91**	13.32 ± 12.70	97.68 ± 0.76	**98.39 ± 0.65**	99.82 ± 0.04	**99.89 ± 0.01**	99.71 ± 0.13	99.79 ± 0.03	97.36 ± 0.09	**99.95 ± 0.01**
ANN	**79.14 ± 2.26**	65.82 ± 5.45	2.60 ± 4.00	**93.95 ± 1.66**	88.39 ± 2.98	**58.43 ± 7.08**	43.41 ± 3.09	22.17 ± 1.02	22.17 ± 1.02	86.05 ± 5.86	86.05 ± 5.86	**86.69 ± 4.34**	93.95 ± 1.66	**96.95 ± 0.97**	91.14 ± 5.58
BNK	**37.80 ± 3.06**	22.11 ± 5.63	0.30 ± 0.21	**64.29 ± 1.83**	30.79 ± 4.41	**71.37 ± 2.57**	51.91 ± 0.72	**55.14 ± 0.53**	8.15 ± 0.18	**32.66 ± 1.36**	23.26 ± 2.46	29.28 ± 3.72	71.37 ± 2.57	79.16 ± 1.64	**83.79 ± 1.37**
FD	66,75 ± 2.03	**68.58 ± 2.74**	0.00 ± 0.00	64.96 ± 2.42	**65.06 ± 5.14**	**84.47 ± 0.88**	82.68 ± 0.34	**60.20 2.76**	0.39 ± 0.35	**71.61 ± 2.88**	71.38 ± 3.25	66.69 ± 4.98	71.61 ± 2.88	77.50 ± 1.08	**85.53 ± 0.68**
CIS-FD	**41.96 ± 9.84**	1.80 ± 1.93	0.36 ± 0.45	**95.88 ± 1.68**	73.54 ± 36.91	63.06 ± 1.63	**64.65 ± 0.73**	**52.72 ± 2.13**	2.61 ± 0.31	**50.30 ± 5.62**	1.45 ± 2.57	0.77 ± 1.01	63.06 ± 1.63	0.62 ± 0.11	**76.80 ± 5.63**
GMSC	**27.16 ± 1.55**	12.09 ± 0.84	0.01 ± 0.03	**58.23 ± 1.53**	18.05 ± 1.85	**78.36 ± 12.92**	6.69 ± 2.83	**44.93 ± 0.79**	2.38 ± 0.24	**12.76 ± 0.76**	9.93 ± 1.18	11.03 ± 4.09	58.23 ± 1.53	57.21 ± 0.91	**67.65 ± 3.89**
PB	**79.66 ± 1.27**	69.36 ± 1.64	15.46 ± 5.66	**79.00 ± 3.57**	44.96 ± 12.05	**66.35 ± 5.49**	49.30 ± 2.61	**75.46 ± 1.87**	55.12 ± 1.29	**76.57 ± 2.38**	70.73 ± 2.74	40.36 ± 15.74	**79.66 ± 1.27**	72.24 ± 4.33	64.33 ± 3.74
IS	**92.67 ± 1.08**	87.93 ± 2.24	55.96 ± 7.90	**92.31 ± 1.90**	52.46 ± 28.52	94.86 ± 1.97	**96.72 ± 0.59**	**96.96 ± 0.55**	95.20 ± 1.37	**93.83 ± 1.31**	91.12 ± 1.38	60.06 ± 26.52	93.83 ± 1.31	95.50 ± 0.74	**95.99 ± 0.96**
SLOG	**99.49 ± 0.08**	99.34 ± 0.10	98.60 ± 0.33	**99.27 ± 0.29**	97.76 ± 0.78	**66.22 ± 0.90**	62.86 ± 0.70	**98.31 ± 0.09**	97.52 ± 0.10	**99.28 ± 0.10**	99.11 ± 0.10	97.51 ± 0.55	**99.49 ± 0.08**	98.90 ± 0.39	97.29 ± 0.57
Total	7/10			8/10		8/10		8/10		7/10			2/10		

TABLE IV

F1-MEASURE OF THE MINORITY CLASS RESULTS OF ALL THE DATASETS. CS MEANS THAT, THAT ALGORITHM IS USED PIPELINED WITH THE C-SMOTE META-STRATEGY. RB STANDS FOR THE REBALANCESTREAM STRATEGY. RESULTS IN BOLD ARE THE BEST FOR THAT ALGORITHM, WHILE RESULTS UNDERLINED ARE THE BEST AMONG ALL THE ALGORITHMS ON THAT DATASET.

F1-Measure[0] %															
Dataset	***CS ARF***	**ARF**	**ARF$_{RE}$**	***CS HAT***	**HAT**	***CS NB***	**NB**	***CS KNN***	**KNN**	***CS SWT$_{ARF}$***	**SWT$_{ARF}$**	**RB**	***BEST CS***	**OOB**	**UOB**
SYN	87.77 ± 0.60	**88.59 ± 0.70**	79.41 ± 2.08	77.43 ± 1.73	**85.14 ± 0.80**	**60.06 ± 1.17**	50.96 ± 1.94	84.01 ± 1.02	**87.46 ± 0.69**	86.24 ± 0.84	**88.59 ± 0.71**	73.17 ± 2.50	84.01 ± 1.02	**90.25 ± 0.30**	85.96 ± 1.13
SYN-CD	99.88 ± 0.01	**99.92 ± 0.01**	99.63 ± 0.00	**99.84 ± 0.01**	98.72 ± 0.04	**29.58 ± 18.84**	21.61 ± 18.55	98.82 ± 0.39	**99.18 ± 0.33**	99.89 ± 0.02	**99.92 ± 0.01**	99.83 ± 0.06	99.88 ± 0.01	98.64 ± 0.05	**99.95 ± 0.01**
ANN	**78.06 ± 1.32**	72.69 ± 3.18	4.77 ± 7.26	**81.35 ± 2.26**	80.23 ± 2.78	**61.23 ± 4.31**	56.05 ± 2.43	35.10 ± 1.17	35.10 ± 1.17	**81.51 ± 2.93**	**81.51 ± 2.93**	79.83 ± 3.88	**81.35 ± 2.26**	81.31 ± 3.34	72.24 ± 3.58
BNK	**44.22 ± 2.36**	32.16 ± 6.11	0.60 ± 0.41	**46.68 ± 0.93**	39.26 ± 3.75	45.08 ± 1.95	**49.08 ± 0.60**	**31.26 ± 0.46**	14.30 ± 0.28	**41.09 ± 1.07**	33.87 ± 2.67	38.81 ± 3.46	45.08 ± 1.95	**56.65 ± 0.40**	51.60 ± 0.76
FD	75,40 ± 1.42	**77.04 ± 1.61**	0.00 ± 0.00	43.45 ± 2.53	**63.17 ± 3.32**	**13.29 ± 0.97**	12.00 ± 0.38	**53.10 ± 4.40**	0.76 ± 0.70	**79.21 ± 1.67**	78.70 ± 1.74	64.78 ± 3.42	**79.21 ± 1.67**	75.52 ± 3.46	8.19 ± 0.83
CIS-FD	**9.94 ± 0.43**	3.34 ± 3.53	0.68 ± 0.85	6.96 ± 0.07	**7.36 ± 0.86**	12.83 ± 0.53	**13.91 ± 0.11**	**9.83 ± 0.38**	4.39 ± 0.45	**10.25 ± 0.74**	2.72 ± 4.67	0.39 ± 1.98	**12.83 ± 0.53**	1.19 ± 0.22	9.85 ± 1.04
GMSC	**31.44 ± 1.94**	19.95 ± 1.15	0.03 ± 0.06	**24.32 ± 0.92**	11.75 ± 1.93	**14.80 ± 0.88**	9.48 ± 2.58	**18.72 ± 0.24**	4.52 ± 0.44	**19.22 ± 1.01**	16.93 ± 1.76	17.86 ± 5.22	24.32 ± 0.92	**40.49 ± 0.28**	32.21 ± 1.48
PB	**79.66 ± 1.24**	76.76 ± 1.66	26.17 ± 8.67	**58,70 ± 3.54**	51.77 ± 8.19	**59.29 ± 4.25**	50.15 ± 2.25	64.36 ± 1.08	**66.03 ± 0.92**	**78.72 ± 2.27**	77.19 ± 1.91	46.49 ± 11.93	**79.66 ± 1.24**	64.73 ± 3.81	56.23 ± 2.28
IS	**94.56 ± 0.86**	91.96 ± 1.71	70.84 ± 6.43	**86.94 ± 6.53**	47.09 ± 15.08	**65.93 ± 3.25**	64.79 ± 1.32	84.10 ± 1.02	**93.02 ± 0.95**	**95.25 ± 0.83**	94.18 ± 0.97	51.00 ± 13.88	**95.25 ± 0.83**	71.85 ± 5.21	64.32 ± 1.91
SLOG	**99.69 ± 0.04**	99.63 ± 0.05	99.27 ± 0.17	**99.18 ± 0.37**	98.35 ± 0.53	69.77 ± 0.89	**71.48 ± 0.32**	95.05 ± 0.16	**97.53 ± 0.07**	**99.56 ± 0.05**	99.49 ± 0.06	98.14 ± 0.39	**99.69 ± 0.04**	99.20 ± 0.32	98.00 ± 0.42
Total	7/10			7/10		7/10		4/10		8/10			6/10		

TABLE V

RECALL OF THE MAJORITY CLASS RESULTS OF ALL THE DATASETS. CS MEANS THAT, THAT ALGORITHM IS USED PIPELINED WITH THE C-SMOTE META-STRATEGY. RB STANDS FOR THE REBALANCESTREAM STRATEGY. RESULTS IN BOLD ARE THE BEST FOR THAT ALGORITHM, WHILE RESULTS UNDERLINED ARE THE BEST AMONG ALL THE ALGORITHMS ON THAT DATASET.

Recall[1] %															
Dataset	***CS ARF***	**ARF**	**ARF$_{RE}$**	***CS HAT***	**HAT**	***CS NB***	**NB**	***CS KNN***	**KNN**	***CS SWT$_{ARF}$***	**SWT$_{ARF}$**	**RB**	***BEST CS***	**OOB**	**UOB**
SYN	97.20 ± 0.24	97.73 ± 0.24	**99.41 ± 0.12**	90.28 ± 0.38	**96.39 ± 0.33**	82.05 ± 0.57	**94.79 ± 0.61**	91.35 ± 0.04	**96.44 ± 0.25**	97.22 ± 0.26	**97.75 ± 0.28**	91.84 ± 0.62	91.35 ± 0.04	**95.86 ± 0.22**	91.84 ± 0.42
SYN-CD	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.99 ± 0.01	99.99 ± 0.01	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
ANN	98.11 ± 0.17	98.80 ± 0.22	**99.94 ± 0.04**	97.02 ± 0.50	**97.44 ± 0.35**	97.45 ± 0.42	**99.09 ± 0.12**	99.67 ± 0.07	99.67 ± 0.07	**98.01 ± 0.20**	**98.01 ± 0.20**	97.55 ± 0.47	**97.02 ± 0.50**	96.64 ± 0.86	95.09 ± 0.79
BNK	95.63 ± 0.23	98.17 ± 0.52	**99.97 ± 0.01**	85.28 ± 0.48	**96.63 ± 0.42**	80.64 ± 2.81	**92.10 ± 0.26**	73.81 ± 0.69	**99.23 ± 0.02**	96.52 ± 0.18	**98.17 ± 0.22**	96.81 ± 0.39	80.64 ± 2.81	**86.71 ± 0.74**	81.31 ± 0.76
FD	99.98 ± 0.00	99.98 ± 0.00	**100.00 ± 0.00**	99.77 ± 0.02	**99.93 ± 0.01**	**98.11 ± 0.19**	97.93 ± 0.08	99.88 ± 0.03	**100.00 ± 0.00**	**99.98 ± 0.00**	**99.98 ± 0.00**	99.93 ± 0.01	**99.98 ± 0.00**	99.95 ± 0.02	96.67 ± 0.36
CIS-FD	74.12 ± 6.03	99.89 ± 0.09	**99.94 ± 0.08**	5.08 ± 2.52	**28.20 ± 40.58**	69.52 ± 2.19	**71.64 ± 0.62**	65.79 ± 2.83	**99.40 ± 0.09**	69.03 ± 4.23	**99.97 ± 0.04**	99.10 ± 2.59	69.52 ± 2.19	**99.88 ± 0.04**	47.74 ± 10.52
GMSC	96.67 ± 1.21	99.35 ± 0.05	**100.00 ± 0.00**	76.98 ± 1.69	**86.11 ± 3.01**	36.15 ± 14.75	**97.84 ± 1.23**	75.99 ± 0.50	**99.78 ± 0.02**	98.57 ± 0.06	**99.48 ± 0.05**	99.26 ± 0.28	76.98 ± 1.69	**91.02 ± 0.31**	81.83 ± 2.54
PB	97.68 ± 0.33	98.71 ± 0.20	**99.90 ± 0.04**	89.7 ± 1.20	**97.06 ± 1.08**	93.46 ± 0.98	**94.62 ± 0.54**	93.28 ± 0.47	**98.66 ± 0.10**	97.95 ± 0.35	**98.58 ± 0.22**	96.93 ± 1.34	**97.68 ± 0.33**	94.21 ± 0.47	92.68 ± 0.33
IS	99.44 ± 0.17	99.45 ± 0.37	**99.77 ± 0.17**	**96.45 ± 3.19**	91.02 ± 4.55	**84.46 ± 2.23**	83.05 ± 0.96	94.41 ± 0.54	**98.42 ± 0.49**	99.47 ± 0.16	**99.61 ± 0.13**	89.51 ± 4.11	**99.47 ± 0.16**	88.11 ± 2.98	82.92 ± 1.53
SLOG	99.97 ± 0.01	**99.98 ± 0.01**	**99.98 ± 0.00**	**99.75 ± 0.13**	99.71 ± 0.09	93.56 ± 1.08	**96.46 ± 0.48**	97.67 ± 0.10	**99.33 ± 0.04**	99.96 ± 0.01	**99.97 ± 0.01**	99.68 ± 0.10	**99.97 ± 0.01**	99.87 ± 0.07	99.65 ± 0.09
Total	0/10			2/10		2/10		0/10		2/10			6/10		

TABLE VI

F1-MEASURE OF THE MAJORITY CLASS RESULTS OF ALL THE DATASETS. CS MEANS THAT, THAT ALGORITHM IS USED PIPELINED WITH THE C-SMOTE META-STRATEGY. RB STANDS FOR THE REBALANCESTREAM STRATEGY. RESULTS IN BOLD ARE THE BEST FOR THAT ALGORITHM, WHILE RESULTS UNDERLINED ARE THE BEST AMONG ALL THE ALGORITHMS ON THAT DATASET.

F1-Measure[1] %															
Dataset	***CS ARF***	**ARF**	**ARF$_{RE}$**	***CS HAT***	**HAT**	***CS NB***	**NB**	***CS KNN***	**KNN**	***CS SWT$_{ARF}$***	**SWT$_{ARF}$**	**RB**	***BEST CS***	**OOB**	**UOB**
SYN	96.13 ± 0.18	**96.43 ± 0.15**	94.55 ± 0.27	92.10 ± 0.25	**95.30 ± 0.32**	84.91 ±0.64	**88.26 ± 0.72**	94.08 ± 0.09	**95.94 ± 0.16**	95.82 ± 0.19	**96.44 ± 0.15**	91.33 ± 0.68	94.08 ± 0.09	**96.63 ± 0.18**	94.72 ± 0.24
SYN-CD	99.99 ± 0.00	**100.00 ± 0.00**	99.98 ± 0.00	**99.99 ± 0.00**	99.93 ± 0.00	**97.90 ± 0.35**	97.77 ± 0.32	99.94 ± 0.02	**99.96 ± 0.02**	99.99 ± 0.00	**100.00 ± 0.00**	99.99 ± 0.00	99.99 ± 0.00	99.93 ± 0.00	**100.00 ± 0.00**
ANN	**98.22 ± 0.10**	98.04 ± 0.16	96.22 ± 0.14	**98.25 ± 0.26**	98.24 ± 0.26	97.07 ± 0.21	**97.33 ± 0.07**	96.81 ± 0.03	96.81 ± 0.03	**98.44 ± 0.19**	**98.44 ± 0.19**	98.23 ± 0.36	**98.25 ± 0.26**	98.17 ± 0.44	97.13 ± 0.41
BNK	93.82 ± 0.09	**94.17 ± 0.12**	93.79 ± 0.01	89.76 ± 0.25	**93.91 ± 0.10**	87.42 ± 1.55	**92.81 ± 0.14**	82.12 ± 0.45	**93.88 ± 0.01**	93.96 ± 0.05	**94.24 ± 0.06**	93.95 ± 0.19	87.42 ± 1.55	**91.52 ± 0.33**	88.64 ± 0.42
FD	**99.96 ± 0.00**	**99.96 ± 0.00**	99.91 ± 0.00	99.85 ± 0.01	**99.93 ± 0.01**	**99.03 ± 0.10**	98.94 ± 0.04	99.91 ± 0.01	99.91 ± 0.00	**99.97 ± 0.00**	**99.97 ± 0.00**	99.94 ± 0.00	**99.97 ± 0.00**	99.96 ± 0.01	98.30 ± 0.18
CIS-FD	83.97 ± 3.81	**98.16 ± 0.04**	**98.16 ± 0.03**	9.56 ± 4.51	**31.54 ± 41.45**	81.35 ± 1.50	**82.84 ± 0.40**	78.50 ± 2.02	**97.92 ± 0.04**	80.74 ± 2.78	**98.19 ± 0.04**	97.80 ± 1.23	81.35 ± 1.50	98.13 ± 0.02	63.62 ± 9.83
GMSC	95.76 ± 0.59	**96.62 ± 0.01**	96.54 ± 0.00	85.53 ± 1.04	**89.68 ± 1.71**	51.1 ± 13.80	**95.68 ± 0.54**	84.47 ± 0.31	**96.51 ± 0.01**	96.25 ± 0.03	**96.62 ± 0.02**	96.54 ± 0.02	85.53 ± 1.04	**93.79 ± 0.14**	88.85 ± 1.43
PB	**97.68 ± 0.17**	97.64 ± 0.17	95.36 ± 0.28	93.39 ± 0.79	**95.47 ± 0.27**	**94.74 ± 0.61**	94.43 ± 0.30	95.15 ± 0.21	**96.83 ± 0.07**	**97.65 ± 0.26**	**97.65 ± 0.17**	95.16 ± 0.25	**97.68 ± 0.17**	95.46 ± 0.42	94.22 ± 0.24
IS	**99.12 ± 0.14**	98.73 ± 0.27	96.36 ± 0.63	**97.53 ± 1.64**	91.5 ± 0.35	**91.14 ± 1.31**	90.47 ± 0.59	96.87 ± 0.25	**98.80 ± 0.18**	**99.22 ± 0.13**	99.07 ± 0.15	91.26 ± 0.36	**99.22 ± 0.13**	93.28 ± 1.64	90.33 ± 0.89
SLOG	**99.91 ± 0.01**	99.90 ± 0.01	99.80 ± 0.04	**99.78 ± 0.10**	99.55 ± 0.14	92.28 ± 0.47	**93.39 ± 0.17**	98.59 ± 0.05	**99.33 ± 0.02**	**99.88 ± 0.01**	99.86 ± 0.02	99.50 ± 0.11	**99.91 ± 0.01**	99.78 ± 0.09	99.46 ± 0.11
Total	5/10			4/10		4/10		0/10		5/10			5/10		

Then, we calculate $avgInc_{m,c,alg}$, the average among all the $inc_{d,m,c,alg}$ having the same m, c, alg. The final values are the averages among all the $avgInc_{m,c,alg}$ having the same m, c.

We observe that in the case of the minority class, the techniques pipelined with C-SMOTE outperform the others in 7 out of 10 data streams, with an average improvement of +550.61% in Recall and +166.22% in F1-Measure. In the case of the majority class, the techniques pipelined with C-SMOTE outperform the others in only 3 data streams out of 10. The average decrease of C-SMOTE results is -7.54% in Recall and -4.42% in F1-Measure. Notice that the gains in performance for the minority class is three orders of magnitude larger than the decrease in performance for the majority class. This is relevant since, as previously mentioned, when learning from imbalanced data the most relevant class is usually the minority class.

We can conclude that the *Hp.1* is validated, while the *Hp.2* is not but in general the C-SMOTE minority class performances improvement is larger than the decline in the majority class.

VI. Conclusions

In this work, we presented a new meta-strategy called C-SMOTE, inspired by the popular SMOTE technique, that continuously balances an evolving data stream one sample at time. C-SMOTE can be used as a data filter pipelined with existing streaming classification techniques. This novel meta-strategy does not require access to all data. New samples are stored in a dynamic-size window whose size is maintained in agreement with a drift detector (ADWIN). C-SMOTE uses the minority class samples contained on this window to introduce synthetic samples.

We tested the proposed meta-strategy pipelined with state-of-the-art algorithms on multiple scenarios of imbalance, concept drift and swapping between minority and majority classes. Empirical results prove that, in most of cases, C-SMOTE increases the Recall and F1-Measure performances.

We leave at the future extended version of this paper the computational costs analysis of our meta-strategy, analyzing the trade-off between the improved predictive performances and the amount of time or RAM needed to train the model. We will test other datasets with different types of concept drift, too.

For future work, our main goal is to minimize the number of instances saved into the window. Moreover, we want to improve even more the C-SMOTE performance, in particular on the majority class to validate the *Hp.2*. A potential solution is to use Borderline-SMOTE, Adaptive Synthetic Sampling, or SMOTE+Tomek. Other research venue is to adapt C-SMOTE to multiclass and regression tasks.

References

[1] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

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

[3] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[5] L. E. B. Ferreira, H. M. Gomes, A. Bifet, and L. S. Oliveira, "Adaptive random forests with resampling for imbalanced data streams," in *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*. IEEE, 2019, pp. 1–6.

[6] A. Bernardo, E. Della Valle, and A. Bifet, "Rebalancing learning on evolving data streams," in *ICDM Workshops*. IEEE, 2020, in press.

[7] S. Wang, L. L. Minku, and X. Yao, "A learning framework for online class imbalance learning," in *Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning, CIEL 2013, IEEE Symposium Series on Computational Intelligence (SSCI), 16-19 April 2013, Singapore*. IEEE, 2013, pp. 36–45.

[8] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*. SIAM, 2007, pp. 443–448.

[9] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, J. F. E. IV, F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki, Eds. ACM, 2009, pp. 329–338.

[10] A. Ghazikhani, R. Monsefi, and H. S. Yazdi, "Recursive least square perceptron model for non-stationary and imbalanced data stream classification," *Evolving Systems*, vol. 4, no. 2, pp. 119–131, 2013.

[11] ——, "Online neural network model for non-stationary and imbalanced data stream classification," *Int. J. Machine Learning & Cybernetics*, vol. 5, no. 1, pp. 51–62, 2014.

[12] ——, "Ensemble of online neural networks for non-stationary and imbalanced data streams," *Neurocomputing*, vol. 122, pp. 535–544, 2013.

[13] B. Mirza, Z. Lin, and N. Liu, "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift," *Neurocomputing*, vol. 149, pp. 316–329, 2015.

[14] B. Wang and J. Pineau, "Online bagging and boosting for imbalanced data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3353–3366, 2016.

[15] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, 2015.

[16] 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," *Mach. Learn.*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[17] N. C. Oza, "Online bagging and boosting," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, Hawaii, USA, October 10-12, 2005*. IEEE, 2005, pp. 2340–2345.

[18] E. Schubert and A. Zimek, "ELKI: A large open-source library for data analysis - ELKI release 0.7.5 "heidelberg"," *CoRR*, vol. abs/1902.03616, 2019.

[19] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

[20] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decis. Support Syst.*, vol. 62, pp. 22–31, 2014.

[21] A. D. Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 29, no. 8, pp. 3784–3797, 2018.

[22] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *Advances in Intelligent Data Analysis VIII, 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31 - September 2, 2009. Proceedings*, ser. Lecture Notes in Computer Science, N. M. Adams, C. Robardet, A. Siebes, and J. Boulicaut, Eds., vol. 5772. Springer, 2009, pp. 249–260.

[23] A. Bifet, J. Read, I. Zliobaite, B. Pfahringer, and G. Holmes, "Pitfalls in benchmarking data stream classification and how to avoid them," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I*, ser. Lecture Notes in Computer Science, H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezný, Eds., vol. 8188. Springer, 2013, pp. 465–479.