OnlineSIRUOS: An Inverse Random Under and Oversampling, Heterogeneous Ensemble, and Meta-Learning Approach for Imbalanced Data Stream Classification

Vinicios Cainã dos Santos Coelho[1 [0000−0001−9173−4576]], Alceu de Souza Britto Jr.[1 [0000−0002−3064−3563]], and Jean Paul Barddal[1 [0000−0001−9928−854X]]

Programa de Pós-Graduação em Informática (PPGIa), Pontifícia Universidade Católica do Paraná (PUCPR), Curitiba, Brasil
{vinicios.coelho, alceu, jean.barddal}@ppgia.pucpr.br

Abstract. Data streams are potentially unbounded data sequences that are made available rapidly and over time. Due to their pervasiveness, mining data streams has become a major scientific and practical issue. Scenarios involving data streams present multiple challenges, including the requirement for single-pass processing due to constraints on computational resources and the necessity to respond to concept drift over time. Another common trait of several streaming scenarios is class imbalance, that is, a class, often of interest, is majorly outnumbered by others, thus hardening the learning process. This paper introduces Online Stacking Inverse Random Under and Over Sampling (OnlineSIRUOS). This ensemble-based approach combines meta-learning, sampling, and heterogeneous components to address class-imbalanced data stream classification. We evaluated our proposal against existing work tailored for class imbalance in data streams using synthetic and real-world datasets. Experimental results show that our proposal achieves competitive F1 scores in different imbalance ratios and is less computationally intensive than its competitors in processing time and memory consumption. The results also show that our proposal is particularly well-suited for highly imbalanced data streams.

Keywords: Data Stream Classification · Class Imbalance · Ensemble Learning

1 Introduction

The immense amount of data generated daily creates a special machine learning scenario known as data stream mining. In data stream mining, algorithms must always be available for prediction and up-to-date with current data without over-consuming computational resources. Given these constraints, several algorithms have been adapted from traditional batch learning to streaming variants, including monolithic and ensemble-based methods [18].

Regarding the most common task in machine learning, i.e., classification, a recurring challenge is class imbalance, which occurs when the number of instances associated with each class is disproportionate. Even though this phenomenon may occur in different degrees, there are scenarios in which class ratios are so imbalanced that a classifier might overfit the majority class while ignoring the minority class.

Multiple classifiers have been developed focusing on streaming scenarios to address the class imbalance, including ensembles, sampling algorithms, cost-sensitive learning, and algorithm modification approaches [15]. The most prominent algorithms include *CSARF* [21] and *ARFRE* [16], which are Adaptive Random Forest [18] variants. However, they present significant computational resource usage. With this in mind, we propose an algorithm named *Online Stacking Inverse Random Under and Over Sampling* (OnlineSIRUOS) that combines essential data stream mining concepts to address imbalanced data stream classification while presenting reduced memory consumption and processing time.

The contribution of the present work is two-fold: i) a novel algorithm (OnlineSIRUOS) inspired by the work of [33], which incorporates three key components: ensemble learning, inverse random under and oversampling, and meta-learning to address the challenges of class imbalance in online data streams; and ii) a robust experimental analysis to validate the effectiveness of OnlineSIRUOS compared to existing works. Synthetic and real-world data are used in the experiments to evaluate the performance of the proposed algorithm across various scenarios.

This paper is divided as follows. Section 2 describes the core concepts of data stream classification and class imbalance. Section 3 describes related works on imbalanced data stream classification. Section 4 introduces our proposal, which is experimentally assessed in Section 5. Finally, Section 6 concludes this paper and discusses envisioned future works.

2 Data Stream Classification and Class Imbalance

Data streams are potentially unbounded data sequences $S = (s^1, s^2, s^3, ..., s^\infty)$ where every item (s^t) is an independent instance composed of $(\boldsymbol{x}^t, y^t)$ that is made available at a timestamp t. In particular, $\boldsymbol{x}^t$ represents a d-dimensional vector, and $y_t \in Y$ represents its corresponding class. In classification, the interest lies in creating and updating a predictive model $f : \boldsymbol{x} \rightarrow Y$ over time that is as accurate as possible.

Streaming data poses several challenges compared to batch scenarios, including mainly data availability, processing time, and memory consumption. According to [6], to consider an algorithm apt to work with data streams, it has to meet the following requirements:

1. Process one instance at a time and only once.
2. Use a limited amount of memory.
3. Use a limited amount of time.
4. Be ready to predict new examples at any time.

5. Be able to detect and adapt to concept drifts, which are changes in the data distribution that occur over time [30].

In this paper, we are particularly interested in data stream classification scenarios in which classes are unequally represented, thus giving rise to a phenomenon called "class imbalance". In class imbalance scenarios, the less frequent class is often the most significant and complex to identify. For instance, in an image classification problem that aims to discern between benign and malignant tumors, the latter are much less frequent than the former and also more significant.

The level of imbalance of a task can be measured with an imbalance ratio (IR). Considering a binary problem, the class with fewer instances is called **positive class** and its counterpart **negative class**. The IR can be calculated by dividing the number of majority class instances by the number of minority class instances [34]. In practice, the higher the IR, the more likely classifiers are to overfit the majority class and underfit the minority one. This occurs since classifiers are designed to minimize the overall error rate, thus considering the classification error of all classes equally [31].

3 Related Works

Different proposals have been developed to handle class imbalance in data stream scenarios. In this section, we categorize the existing algorithms according to the taxonomy given in [15], i.e., ensemble algorithms, sampling algorithms, cost-sensitive learning, and algorithm modification. It is noteworthy to highlight that several algorithms fall under different categories; thus, these are not mutually exclusive.

3.1 Ensemble

Ensemble learning focuses on decreasing bias and variance by training and combining multiple weak learners into a strong one. A relevant aspect of ensemble learning is that its members must be diverse. Thus, different techniques for training each component have been proposed, including bagging, boosting, random subspaces, and random forests [19]. Regarding the prediction step, members' votes are often combined using majority vote, weighted majority vote, or using selection methods [19].

Focusing on imbalanced data streams, different ensemble techniques have been proposed. One example is the *OnlineAdaC2* [31], which has been inspired by AdaC2 [28]. OnlineAdaC2 is based on boosting, and it uses different classification error costs to assign different weights to the instances, thus strengthening the training on instances of the minority class. OzaBoost has also adopted the OnlineAdaC2 rationale [26], which samples instances initially according to $Poisson(\lambda = 1)$ and then adjusts the training weights according to the misclassification rates of other classifiers to prioritize hard-to-classify instances. In addition, to identify and handle concept drifts, OzaBoosts' implementations, e.g., in MOA [5] and River [22], are coupled with the ADWIN drift detector [4].

3.2 Sampling

Sampling techniques target balancing the number of instances of each class used for training. An interesting trait of sampling algorithms is that they are classifier-agnostic, and thus, they can be coupled with any learning algorithm [25]. There are three categories of sampling approaches: oversampling, undersampling, and hybrid. Oversampling regards the balancing of classes via the generation of instances from the minority class, while undersampling seeks class balance by removing cases that belong to the majority class. Finally, hybrid algorithms combine both of the approaches above.

In the context of imbalanced data stream classification, a relevant work is the *Adaptive Random Forest with Resampling* (ARFRE) [16], which is a variant of the original Adaptive Random Forest (ARF) [18]. In ARF, instances are sampled for training in each tree according to Poisson($\lambda = 6$), resulting in each instance being selected for training at least once, approximately 99% of the time. In ARFRE, this process is adjusted so that instances are resampled and reweighted considering class imbalance according to Equation 1, where I_{y_i} represents the number of instances of class y_i observed thus far, and $I = \sum_{y_j \in Y} I_{y_j}$ represents the total number of instances observed regardless of the class.

$$\text{Weight}(\boldsymbol{x}, y) = \frac{100 - \frac{I_y \times 100}{I}}{100} \times \text{Poisson}(\lambda) \tag{1}$$

With this sampling-based change, minority class instances are selected more often than their original counterparts, thus avoiding the overfitting issue towards the majority class.

3.3 Cost-sensitive

Cost-sensitive learning uses classification errors to enhance classifiers during training or prediction. In practice, different types of errors are coupled with costs (weights) that are used to penalize each classifier so that classes with more errors are prioritized during training.

A relevant example of cost-sensitive learning in imbalanced data stream classification is the *Cost-sensitive Adaptive Random Forest* (CSARF) [21], which, similarly to ARFRE, is also an adaptation of ARF. However, unlike its counterparts, it uses the costs of misclassifications to handle class imbalance. CSARF adopts a sliding window to check the class distribution over time. This sliding window is relevant to compute the imbalance ratio and, consequently, the costs coupled with each class. Next, CSARF guarantees that all its members are trained with minority class samples, regardless of whether they can correctly predict such samples. Even though the original paper depicts that the prediction step of CSARF assumes the Matthews Correlation Coefficient (MCC) to weigh in members' votes, the implementation in [5] has been adjusted by the authors to assume the F1-score. Finally, costs are used during prediction using either a local or a global approach. The local approach associates each ensemble classifier prediction with their respective costs, which come from user-given cost matrices

before the weighted majority vote strategy is used on all votes from all classifiers in the ensemble. In the global approach, the votes of each base classifier in the ensemble are combined using the majority vote first. Then, the cost matrix is applied to the probability vector representing the combination of these votes.

3.4 Algorithm Modification

Algorithm modification regards changes in the inner parts of a classifier so that it favors the classification of the minority class. One of the main drawbacks of this approach is that the modification made to a specific classifier may not be reproducible to another classifier. Examples of algorithm modifications include OnlineAdaC2 and CSARF, which were discussed in the sections above.

4 Proposal

This section describes our proposal, the Online Stacking Inverse Random Under and Over Sampling (OnlineSIRUOS) algorithm. OnlineSIRUOS combines stacking, inverse random under and oversampling, and ensemble learning to address imbalanced data stream classification while presenting reduced memory consumption and processing time. First, we describe SIRUS [33], an algorithm for batch imbalanced classification that inspired OnlineSIRUOS' design. Next, we detail OnlineSIRUOS and its training and test steps.

4.1 Preliminaries on SIRUS

Stacking and Inverse Random UnderSampling (SIRUS) [33] is an ensemble-based algorithm for batch classification that uses sampling and meta-learning to deal with class imbalance. SIRUS assumes an inverse random undersampling process [29] that consists of creating B groups of instances from the training dataset. However, in contrast to Bagging [10], it maintains all instances from the minority class and undersamples elements from the majority class until the imbalance ratio (IR) of each group is the inverse of the original set. We denote the original dataset's IR as $\frac{W}{Z}$, where Z and W represent the cardinality of the sets of instances from each class. Therefore, the IR of the undersampled groups follows Equation 2, where W_i represents the number of instances of the majority class randomly selected for the i-th group.

$$\frac{Z}{W_i} \approx \frac{W}{Z} \tag{2}$$

Each data group created is used to train one of the B sets of classifiers. This also means that each classifier in the ensemble will only be trained on one data group. Also, it is important to note that SIRUS has heterogeneous ensembles built with the following rule: B sets of classifiers will each have one classifier of each type from R defined to compose the ensemble. Consequently, assuming $|R|$ as the number of algorithms to form each group, the ensemble will

have $|B| \times |R|$ classifiers since each group has $|R|$ members. Another relevant trait of SIRUS is meta-learning, which combines the votes cast by the ensemble members. In particular, SIRUS uses stacking [32], in which the outputs of the ensemble members are used as predictive features of another learner responsible for combining these predictions into a final vote.

4.2 OnlineSIRUOS

Online Stacking Inverse Random Under and Over Sampling (OnlineSIRUOS) is inspired by the SIRUS algorithm. OnlineSIRUS is composed of $|B|$ groups of classifiers, and each group contains $|R|$ heterogeneous classifiers.

One of the significant traits of OnlineSIRUOS is the sampling process, which is tailored to be performed incrementally. In practice, OnlineSIRUOS introduces an inverse random under and oversampling process for which the class imbalance is accounted. In practice, the ratio between the number of instances already observed for training and that belong to the same class of a new instance (I_y) and the total number of instances used for training ($I = \sum_{y_j \in Y} I_{y_j}$) is used to adjust the sampling process with Poisson($\lambda = 6$)[1] according to Equation 3. The result of the aforementioned equation (w) indicates the number of times an instance will be used for training in each ensemble group.

$$w(\lambda', y) = \left\lceil \left(1 - \frac{I_y}{\sum_{y_j \in Y} I_{y_j}} \right) \times \text{Poisson}\left(\lambda = \lambda'\right) \right\rceil \tag{3}$$

This process is similar to the resampling process used in ARFRE (see Section 3.2). However, the sampling process is used per group rather than for ensemble members. The rationale behind this process is to avoid overfitting the majority class via oversampling the minority class and undersampling the majority class.

The second major trait of OnlineSIRUOS is the meta-learning component, in which stacking is applied. In practice, any classifier can be used as the stacking meta-learner, and this is represented as Ω in the remainder of this paper. We also work under the assumption that Ω is an online learner, which means that it can be updated over time.

The training step of OnlineSIRUOS is detailed in Algorithm 1. The training step starts by defining an empty array (Y') to store the predictions of each ensemble member that is later used for updating the meta-learner and incrementing the class counts (I) (lines 1 and 2, respectively). Next, the algorithm loops over each group of members $b_i \in B$ (lines 3 to 10). For each group, the resampling process described in Equation 3 determines the number of times an instance will be used for training (line 4). Next, the algorithm loops over the ensemble members in the group (lines 5 to 9), in which each member is updated (line 7), and its prediction is cast and appended to the array of predictions Y' (line 8). Finally, after the loops are over and all ensemble members' forecasts

[1] Even though λ is a user-given parameter, we have assumed $\lambda = 6$ given the empirical observations in [18].

Algorithm 1 OnlineSIRUOS training step

Require: $(\boldsymbol{x}, y)$: a labeled instance for training
Require: ϵ: the ensemble composed of B groups with R members each
Require: Ω: the meta-learner
Require: I: an array for class counting
Require: λ: sampling parameter (defaults to 6)
1: $Y' \leftarrow \emptyset$ ▷ Array of predictions used for meta-learning
2: $I_y \leftarrow I_y + 1$
3: **for** $b_i \in B$ **do**
4: $\quad w \leftarrow \left\lceil \left(1 - \frac{I_y}{\sum_{y_j \in Y} I_{y_j}}\right) \times \text{Poisson}\,(\lambda = \lambda') \right\rceil$ ▷ Equation 3
5: $\quad$ **for** $r_j \in b_i$ **do**
6: $\quad\quad \epsilon_{i,j} \leftarrow \epsilon[b_i][r_j]$
7: $\quad\quad$ Update $\epsilon_{i,j}$ w times with $(\boldsymbol{x}, y)$
8: $\quad\quad Y' \leftarrow Y' \cup \{\epsilon_{i,j}(\boldsymbol{x})\}$
9: $\quad$ **end for**
10: **end for**
11: Update Ω with (Y', y) ▷ Updates the meta-learner

are appended to Y', this array is used to update the meta-learner as predictive features with the corresponding ground-truth label y (line 11).

OnlineSIRUOS' test step follows the process reported in Algorithm 2. This step receives as input an unlabeled instance for prediction $(\boldsymbol{x})$, the ensemble ϵ, and the meta-learner Ω. First, Y' is initialized as an empty array to store the ensemble members' votes (line 1). Next, the algorithm loops over all groups (lines 2 to 7) and the corresponding members (lines 3 to 6), obtaining their predictions and storing them in the array (line 5). Once all predictions are obtained, they are input to the meta-classifier, yielding the final prediction $\hat{y}$ (line 8).

Algorithm 2 OnlineSIRUOS test step

Require: $\boldsymbol{x}$: an unlabeled instance for prediction
Require: ϵ: the ensemble composed of B groups with R members each
Require: Ω: the meta-learner
1: $Y' \leftarrow \emptyset$
2: **for** $b_i \in B$ **do**
3: $\quad$ **for** $r_j \in b_i$ **do**
4: $\quad\quad \epsilon_{i,j} \leftarrow \epsilon[b_i][r_j]$
5: $\quad\quad Y' \leftarrow Y' \cup \{\epsilon_{i,j}(\boldsymbol{x})\}$
6: $\quad$ **end for**
7: **end for**
8: $\hat{y} \leftarrow \Omega(Y')$
9: **return** $\hat{y}$

5 Analysis

In this section, we describe the experimental protocol adopted to assess our proposal against state-of-the-art algorithms for imbalanced data stream classification and discuss the results obtained.

5.1 Experimental Protocol

The method used as a validation scheme for the classifiers tested was *prequential* [17]. In prequential validation, each arriving instance is first used for testing and then for training. Our experimentation tests included both synthetic and real-world datasets. Synthetic streams included binary classification problems created using Asset Negotiation [14], Agrawal [2], Random Trees [13], and SEA [27] experiments that are made available as part of the Massive Online Analysis (MOA) framework [5]. Synthetic data streams were created with different class ratios, i.e., 50%-50%, 70%-30%, 80%-20%, 90%-10%, 95%-05%, 99%-01%, 99.5%-0.5%, to verify how suitable different algorithms are under different levels of class imbalance. Our experimental protocol also included drifting scenarios, in which the class ratios shifted amongst the class ratios above, i.e., from 50%-50% to 99.5%-0.5%, with a change at every 200 thousand instances. The rationale behind this experiment is to verify how the proposed algorithm and its competitors behave under imbalancedness drifting scenarios.

Furthermore, all synthetic streams had a length $U = 1,000,000$, and performance metrics were extracted every 10% of the experiment. Different datasets were gathered with varying class ratios in terms of real-world datasets. Table 1 gives the datasets and their main characteristics.

Table 1: Real-world datasets used in experimentation.

Name	**# of Instances**	**# of Attributes**	**# of Classes**	**IR**
Bank Marketing [23, 24]	45,211	16	2	7.55
Forest Covertype [7] [8] [20]	581,012	54	7	103.13
CSDS-1 [3]	315,539	176	2	6.71
CSDS-2 [3]	50,401	37	2	47.51
CSDS-3 [3]	97,226	152	2	2.84
Kaggle's Give me Loan	150,000	10	2	13.96
IntelLabSensors [9]	2,313,153	5	58	21,898
IntelLabSensors-1vsAll [9]	2,313,153	5	2	52.74
IntelLabSensors-1to3vsAll [9]	2,313,153	5	2	15.93
IntelLabSensors-1to5vsAll [9]	2,313,153	5	2	11.82
IntelLabSensors-1to9vsAll [9]	2,313,153	5	2	5.84
KDD99 [1] [11]	4,898,431	41	2	4.04

Algorithms. Our experimentation included Adaptive Random Forest (ARF), Adaptive Random Forest with Resampling (ARFRE), Kappa Updated Ensemble

(KUE), Leveraging Bagging, Cost-sensitive Adaptive Random Forest (CSARF), OnlineAdaC2, and Online Under-overbagging (OnlineUOB). All ensembles were set to use 100 Hoeffding Trees. The only exception was OnlineSIRUOS, in which 50 groups of Hoeffding Trees and Naive Bayes were used (thus totaling 100 ensemble members), and the meta-classifier Ω was set to Naive Bayes, given its simplicity.

Evaluation metrics and hypothesis testing. Given the interest in assessing algorithms in class-imbalanced scenarios, the assessment was conducted using the F1-score, the harmonic mean of precision and recall. Additionally, algorithms were evaluated according to the processing time (in seconds) and memory consumption (in GB-Hours). Finally, the results were organized and processed by the combination of *Friedman* and *Nemenyi* tests following the protocol described at [12] to identify whether significant differences amongst methods were observable. The results are reported using Critical Distance (CD) charts.

Code availability. Finally, we highlight that the source code for our proposal implementation and experimentation scripts are made available at <**link omitted due to blind-review policy**>.

5.2 Discussion

In this section, we discuss the results obtained during experimentation. First, we divide the discussion of the results according to the experiment conducted: (i) considering all datasets and then focusing on (ii) synthetic experiments with high class imbalance ratios, (iii) synthetic datasets with drifting class imbalance ratios, and (iii) real-world datasets.

Analysis considering all experiments In this section, we analyze the results obtained across all the experiments, i.e., considering synthetic data streams with different class imbalance ratios, synthetic data streams in which the class imbalance drifts, and real-world datasets. We summarize the results in Table 2, in which we report the average macro F1-Score, CPU Time (in seconds), and RAM-Hours (in GB-Hour) obtained across all experiments.

The results show that the proposed algorithm OnlineSIRUOS achieves the highest average macro F1-Scores and is the best-ranked algorithm, followed by Adaptive Random Forest with Resampling (ARFRE), Leveraging Bagging (LevBag), and CSARF. In particular, using Friedman + Nemenyi hypothesis tests showed no statistical difference between these algorithms. In contrast, they surpass the others, i.e., Kappa Updated Ensemble (KUE), Adaptive Random Forest (ARF), OnlineUnderOverBagging (OUOBagging), and OnlineAdaC2.

Considering processing time, the fastest algorithm is OnlineUnderOverBagging (OUOBagging), followed by our proposal (OnlineSIRUOS), Kappa Updated Ensemble (KUE), and OnlineAdaC2. Similarly, as before, Friedman + Nemenyi testing showed that these algorithms are statistically faster than Leveraging Bagging (LevBag), ARFRE, ARF, and CSARF.

Table 2: Average results obtained in all experiments. Results in bold depict the best-ranked group of results per metric, i.e., Macro F1-Score, processing time, and RAM-Hours.

Algorithm	Macro F1-Score (%)	Processing Time (s)	RAM-Hours (GB-Hour)
ARF	77.91	2533.74	1.46×10^{0}
LevBag	**79.85**	917.02	1.70×10^{-1}
ARFRE	**79.55**	1160.28	3.55×10^{-1}
KUE	80.27	**477.29**	4.00×10^{-2}
OUOBagging	72.90	**142.50**	$\mathbf{1.57 \times 10^{-3}}$
CSARF	**80.22**	2571.95	1.48×10^{0}
OnlineAdaC2	56.04	**896.30**	1.55×10^{-1}
OnlineSIRUOS	**80.92**	**254.04**	$\mathbf{8.19 \times 10^{-3}}$

Focusing on memory consumption rates, we observe that OnlineUnderOverBagging (OUOBagging) is the least consuming algorithm, followed by OnlineSIRUOS and Kappa Updated Ensemble (KUE). In particular, the hypothesis test showed that amongst all algorithms, OUOBagging and OnlineSIRUOS are significantly less memory-consuming than KUE, ARFRE, OnlineAdaC2, LevBag, ARF, and CSARF.

Considering all evaluation metrics, OnlineSIRUOS is a fierce competitor, ranked amongst the best-performing algorithms in all traits. Even though this analysis provides an interesting overview of the results obtained and highlights the efficiency of the proposed method against the state-of-the-art, we further delve into the results on scenarios with synthetic data with high imbalance ratios, synthetic data with drifting imbalance ratios, and real-world datasets.

Analysis considering synthetic datasets and high imbalance ratios Since our proposal has been tailored to focus on imbalanced data streams, a specific analysis of such scenarios must be included. Therefore, in this section, we narrow our analysis towards synthetic experiments with high-class imbalance ratios, i.e., 90%-10%, 95%-05%, 99%-01%, and 99.5%-0.5% to check how our proposal compares to existing works in the literature. The results for such scenarios are given in Table 3, where average macro F1-scores, processing time, and memory consumption rates are given.

These results depict that our proposal, OnlineSIRUOS, figures again as the best-performing algorithm in Macro F1-Score, a significant result considering the challenging aspects of highly imbalanced data streams. Considering the result of hypothesis testing, we observe that CSARF, KUE, and LevBag follow OnlineSIRUOS and that these algorithms outperform others significantly. The processing time and memory consumption results do not differ significantly from those observed earlier, i.e., OnlineSIRUOS achieves competitive results, losing to OUOBagging, yet with no statistically significant differences.

Analysis considering synthetic datasets with drifting class imbalance ratios In this section, we assess the behavior of the proposed method and its

Table 3: Average results obtained in highly imbalanced synthetic data streams. Results in bold depict the best-ranked group of results per metric, i.e., Macro F1-Score, processing time, and RAM-Hours.

Algorithm	Macro F1-Score	Processing time (s)	RAM-Hours (GB-Hour)
ARF	66.21	2171.39	8.77×10^{-1}
LevBag	71.59	936.49	1.33×10^{-1}
ARFRE	70.87	606.77	9.04×10^{-2}
KUE	**74.68**	594.44	4.65×10^{-2}
OUOBagging	58.97	**143.88**	$\mathbf{1.84 \times 10^{-3}}$
CSARF	**74.46**	2213.22	8.59×10^{-1}
OnlineAdaC2	65.48	622.11	6.43×10^{-2}
OnlineSIRUOS	**75.93**	**185.65**	$\mathbf{4.10 \times 10^{-3}}$

counterparts from the literature in scenarios where the class imbalance ratio drifts over time. The results obtained are reported in Table 4, in which we observe that ARF achieves the highest macro F1-scores, followed by ARFRE, KUE, and OnlineSIRUOS and that there is no significant statistical difference amongst them. We highlight that in opposition to what has been observed in the previous scenarios, ARF and ARFRE were ranked as the best-performing algorithms, mainly due to their ability to detect and adapt to concept drifts using background learning, i.e., a strategy in which background trees are learned preemptively and swapped when a drift is flagged. This is a relevant trait that is not observed in its contenders. Focusing on processing time and memory consumption, we observe again OUOBagging and OnlineSIRUOS leading the ranks when compared to their counterparts.

Table 4: Average results obtained in synthetic experiments with drifting imbalance ratios. Results in bold depict the best-ranked group of results per metric, i.e., Macro F1-Score, processing time, and memory consumption.

Algorithm	Macro F1-Score	Processing time (s)	RAM-Hours (GB-Hour)
ARF	**90.41**	3576.24	1.67×10^{0}
LevBag	87.38	1331.61	2.53×10^{-1}
ARFRE	**88.93**	2186.98	6.99×10^{-1}
KUE	**88.76**	592.22	4.86×10^{-2}
OUOBagging	83.86	**120.28**	$\mathbf{8.87 \times 10^{-4}}$
CSARF	85.26	3577.41	1.65×10^{0}
OnlineAdaC2	61.74	1443.84	2.63×10^{-2}
OnlineSIRUOS	**88.40**	**313.47**	$\mathbf{1.52 \times 10^{-2}}$

Analysis considering real-world datasets Focusing on real-world datasets, we summarize the results obtained in Table 5, in which the average results for Macro F1-Score, (b) processing time, and (c) RAM-Hours are given with the

respective standard deviations. Here, we observe that the best-ranked algorithm in terms of macro F1-scores is ARFRE, closely followed by the proposed algorithm OnlineSIROUS. Considering Friedman + Nemenyi's statistical tests, we concluded that there are no statistical differences between ARFRE, OnlineSIRUOS, and CSARF. Concerning processing time, we observe that OnlineUnderOverBagging (OUOBagging) is the fastest algorithm, followed by Kappa Updated Ensemble (KUE), Leveraging Bagging (LevBag) and the OnlineSIRUOS. Considering the hypothesis tests, we see no differences between OnlineSIRUOS, OUOBagging, and ARFRE. Finally, the results for memory consumption depict the proposed algorithm (OnlineSIRUOS) as the best-ranked algorithm, followed by OnlineUnderOverBagging (OUOBagging) and ARFRE, and no statistically significant differences are observed among them.

Table 5: Average results obtained in real-world datasets. Results in bold depict the best-ranked group of results per metric, i.e., Macro F1-score, processing time, and RAM-Hours.

Algorithm	**Macro F1-Score**	**Processing time (s)**	**RAM-Hours (GB-Hour)**
ARF	74.66	322.62	3.75×10^{-2}
LevBag	**74.33**	**220.89**	1.48×10^{-2}
ARFRE	**74.91**	259.87	$\mathbf{7.67 \times 10^{-3}}$
KUE	72.33	**148.79**	$\mathbf{1.15 \times 10^{-2}}$
OUOBagging	73.23	**150.37**	$\mathbf{1.73 \times 10^{-3}}$
CSARF	**74.25**	357.82	6.83×10^{-2}
OnlineAdaC2	35.76	254.05	4.40×10^{-3}
OnlineSIRUOS	**73.20**	**234.36**	$\mathbf{4.40 \times 10^{-3}}$

6 Conclusion

Class imbalance is a significant challenge for data stream mining. There are several gaps to be filled due to the varied and complex behavior of data streams when there are unequal examples from different classes. In this paper, inspired by the batch SIRUS algorithm, we introduced OnlineSIRUOS, a novel algorithm for data stream classification that relies on (i) ensemble learning, (ii) inverse random under and oversampling, and (iii) meta-learning.

We experimented with our proposal against existing works using synthetic and real-world data. Considering the F1-score, our proposal, OnlineSIRUOS, achieves significant results across all scenarios. Still, such a conclusion becomes even more evident in highly imbalanced scenarios, i.e., when the minority class is represented by 10% or less of the instances. Our proposal is also well-ranked regarding computational resources, losing only to Online UnderOverBagging, which has significantly lower F1-score rates. One of the gaps observed concerned drifting class imbalance scenarios, i.e., scenarios in which the class ratios evolve

over time, in which OnlineSIRUOS achieved good results; however, they are still below ARF and ARFRE. This is a relevant aspect since ARF and ARFRE use background learning, a technique in which learners are preemptively trained and swapped when a drift is flagged. This process speeds up drift recovery and could be beneficial to our proposal in the future.

Acknowledgments. Acknowledgments will be included in the camera-ready version to maintain the integrity of the double-blind review process.

J.P. Barddal would like to thank CAPES-COFECUB for partially funding this research under grant 88887.987125/2024-00 and Fundação Araucária under grant number 188/2025.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29. p. 81–92. VLDB '03, VLDB Endowment (2003)
2. Agrawal, R., Imielinski, T., Swami, A.: Database mining: a performance perspective. IEEE Transactions on Knowledge and Data Engineering **5**(6), 914–925 (1993)
3. Barddal, J.P., Loezer, L., Enembreck, F., Lanzuolo, R.: Lessons learned from data stream classification applied to credit scoring. Expert Systems with Applications **162**, 113899 (2020)
4. Bifet, A., Gavaldà, R.: Learning from Time-Changing Data with Adaptive Windowing, pp. 443–448
5. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research **11**(52), 1601–1604 (2010)
6. Bifet, A., Kirkby, R.: Data stream mining a practical approach (2009)
7. Bifet, A., Pfahringer, B., Read, J., Holmes, G.: Efficient data stream classification via probabilistic adaptive windows. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. p. 801–806. SAC '13, Association for Computing Machinery, New York, NY, USA (2013)
8. Bifet, A., Read, J., Žliobaitė, I., Pfahringer, B., Holmes, G.: Pitfalls in benchmarking data stream classification and how to avoid them. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 465–479. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
9. Bodik, P., Thibaux, R., Paskin, M., Madden, S., Guestrin, C., Hong, W.: (Jun 2004), `http://db.csail.mit.edu/labdata/labdata.html`
10. Breiman, L.: Bias, variance, and arcing classifiers. Tech. rep., Tech. Rep. 460, Statistics Department, University of California, Berkeley (1996)
11. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: SDM (2006)
12. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**(1), 1–30 (2006)
13. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 71–80. KDD '00, Association for Computing Machinery, New York, NY, USA (2000)

14. Enembreck, F., Ávila, B.C., Scalabrin, E.E., Barthès, J.P.: Learning drifting negotiations. Appl. Artif. Intell. **21**(9), 861–881 (Oct 2007)
15. Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F.: Foundations on Imbalanced Classification, pp. 19–46. Springer International Publishing, Cham (2018)
16. Ferreira, L.E.B., Gomes, H.M., Bifet, A., Oliveira, L.S.: Adaptive random forests with resampling for imbalanced data streams. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–6 (July 2019)
17. Gama, J., Sebastiao, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 329–338. KDD '09, Association for Computing Machinery, New York, NY, USA (2009)
18. Gomes, H.M., Bifet, A., Read, J., Barddal, J.P., Enembreck, F., Pfharinger, B., Holmes, G., Abdessalem, T.: Adaptive random forests for evolving data stream classification. Mach. Learn. **106**(9–10), 1469–1495 (Oct 2017)
19. Gomes, H.M., Barddal, J.P., Enembreck, F., Bifet, A.: A survey on ensemble learning for data stream classification. ACM Comput. Surv. **50**(2) (mar 2017)
20. Kosina, P., Gama, J.: Very fast decision rules for multi-class problems. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. p. 795–800. SAC '12, Association for Computing Machinery, New York, NY, USA (2012)
21. Loezer, L., Enembreck, F., Barddal, J.P., de Souza Britto, A.: Cost-sensitive learning for imbalanced data streams. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. p. 498–504. SAC '20, Association for Computing Machinery, New York, NY, USA (2020)
22. Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T., Bifet, A.: River: machine learning for streaming data in python. J. Mach. Learn. Res. **22**(1) (jan 2021)
23. Moro, S., Laureano, R., Cortez, P.: Using data mining for bank direct marketing: An application of the crisp-dm methodology. In: et al., P.N. (ed.) Proceedings of the European Simulation and Modelling Conference - ESM'2011. pp. 117–121. EUROSIS, Guimaraes, Portugal (Oct 2011)
24. Moro, S., Cortez, P., Rita, P.: A data-driven approach to predict the success of bank telemarketing. Decision Support Systems **62**, 22–31 (2014)
25. Nguyen, H.M., Cooper, E.W., Kamei, K.: A comparative study on sampling techniques for handling class imbalance in streaming data. In: The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems. pp. 1762–1767 (2012)
26. Oza, N.C.: Online bagging and boosting. In: 2005 IEEE International Conference on Systems, Man and Cybernetics. vol. 3, pp. 2340–2345 Vol. 3 (2005)
27. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 377–382. KDD '01, Association for Computing Machinery, New York, NY, USA (2001)
28. Sun, Y.: Cost-Sensitive Boosting for Classification of Imbalanced Data. Ph.D. thesis, CAN (2007), aAINR34548
29. Tahir, M.A., Kittler, J., Yan, F.: Inverse random under sampling for class imbalance problem and its application to multi-label classification. Pattern Recognition **45**(10), 3738–3750 (2012)
30. Tsymbal, A.: The problem of concept drift: definitions and related work (2004)
31. Wang, B., Pineau, J.: Online bagging and boosting for imbalanced data streams. IEEE Transactions on Knowledge and Data Engineering **28**(12), 3353–3366 (2016)

32. Wolpert, D.H.: Stacked generalization. Neural Networks **5**(2), 241–259 (1992)
33. Zhang, Y., Liu, G., Luan, W., Yan, C., Jiang, C.: An approach to class imbalance problem based on stacking and inverse random under sampling methods. In: 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC). pp. 1–6 (2018)
34. Zhu, R., Guo, Y., Xue, J.H.: Adjusting the imbalance ratio by the dimensionality of imbalanced data. Pattern Recognition Letters **133**, 217–223 (2020)