Cost-sensitive learning for imbalanced data streams

Lucas Loezer
Graduate Program in Informatics (PPGIa)
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
loezerl@ppgia.pucpr.br

Fabrício Enembreck
Graduate Program in Informatics (PPGIa)
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
fabricio@ppgia.pucpr.br

Jean Paul Barddal
Graduate Program in Informatics (PPGIa)
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
jean.barddal@ppgia.pucpr.br

Alceu de Souza Britto Jr.
Graduate Program in Informatics (PPGIa)
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
alceu@ppgia.pucpr.br

ABSTRACT

The data imbalance problem hampers the classification task. In streaming environments, this becomes even more cumbersome as the proportion of classes can vary over time. Approaches based on misclassification costs can be used to mitigate this problem. In this paper, we present the Cost-sensitive Adaptive Random Forest (CSARF) and compare it to the Adaptive Random Forest (ARF) and ARF with Resampling (ARF_{RE}) in six real-world and six synthetic data sets with different class ratios. The empirical study analyzes two misclassification costs strategies of the CSARF and shows that the CSARF obtained statistically superior w.r.t. the average recall and average F1 when compared to ARF.

CCS CONCEPTS

• **Machine learning → Machine learning algorithms**; • **Cost-sensitive learning**; • **Online learning settings**; • **Ensemble**;

KEYWORDS

cost-sensitive, ensemble, data stream, imbalanced datasets, adaptive random forest

ACM Reference Format:
Lucas Loezer, Fabrício Enembreck, Jean Paul Barddal, and Alceu de Souza Britto Jr.. 2020. Cost-sensitive learning for imbalanced data streams. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.1145/3341105.3373949

1 INTRODUCTION

Among the existing challenges in the classification task, we focus on problems that involve skewed class distributions. Datasets with class imbalance are those where one class outnumbers other classes. Therefore, it is usual to classifiers to classify instances from the majority class correctly while presenting issues to classify instances from the minority class. This phenomenon can be categorized as intrinsic if the class imbalance is natural such as in the following applications: fraud detection [3], oil spill detection [38], abnormal activity classification [22] and breast cancer malignancy classification [28]. Otherwise, if the occurrence of imbalance does not originate from the problem itself, such as the failure of a sensor in a system or the mishandling of the data by any reason, it is said to be extrinsic. Regardless of the imbalance origin, the minority class (positive class) is very often more relevant than the majority class (negative class), as rare examples involve high costs if not correctly detected.

Table 1: Cost Matrix

	Negative	Positive
Predicted Negative	cost(0,0) or TN	cost(0,1) or FN
Predicted Positive	cost(1,0) or FP	cost(1,1) or TP

The problem of imbalanced data is present in both batch and data stream databases. The difference is that in the streaming environments, the classification task becomes even more difficult as no information can be retrieved from the whole dataset, as only data from the past can be taken into account and there is no guarantee that the data behavior observed thus far shall be maintained. In this environment, the classification task consists in building a model capable of categorizing an instance x^t, which arrives at timestamp t and belongs to a data stream S, in a given class [25]. Due to the evolutionary behavior of a data stream, the proportion between classes may vary over time, and new imbalances may arise [28].

Different solutions have already been presented by researchers to reduce the impact of class imbalance in the classification task. Among them, there are solutions focused on synthesizing new data samples (oversampling) or removing instances (undersampling) to reduce the inequality between the positive and negative classes when training a classifier [2, 39]. There are also approaches using ensembles of classifiers that aim to reduce the over-representation of the majority class by dividing the main problem into balanced sub-problems [11, 31]. Finally, some approaches involve misclassification costs during the decision-making stage of a classifier [24, 40].

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/10.1145/3341105.3373949

The solutions based on misclassification costs assign penalties for classifying from a class in another. Hence, it is possible to employ higher costs to the positive classes to make the classifier consider it during prediction and training steps. If the classification task is binary, the costs for each class can be represented by a cost matrix for two classes, as denoted in Table 1. Mathematically, we let cost(i, j) be the involved cost in classifying an example x in i when its ground-truth class is j. For the classifier to induce a class considering the misclassification costs, the better prevision for the example x must be the class i which minimizes the sum of the probabilities and costs for the actual class of x (Equation 1) [18].

$$L(x, i) = \sum_{j} P(j|x) \times \text{cost}(i, j) \quad (1)$$

In this paper, we study the attribution of sensitivity to misclassification costs in an ensemble of adaptive trees, developed to data stream classification, as known as Adaptive Random Forest (ARF) [26]. We propose the Cost-sensitive Adaptive Random Forest (CSARF), which employs two variations to consider misclassification costs in its architecture. The solutions previous to CSARF to handle imbalanced problems are described in Section 2. The CSARF architecture details are listed in Section 3, and its empirical analysis in Section 4, thus highlighting the competitive results it yields when compared to state-of-the-art approaches. This work's conclusions are presented in Section 5.

2 RELATED WORK

The cost matrix can be applied either to (i) directly influence the output probabilities of a classifier (threshold), or (ii) to oversample the instances with higher misclassification costs (sampling). In [16], authors proposed the MetaCost algorithm, which assigns cost sensitivity to a model, training it in a re-labeled dataset, such that the new labels are given by an ensemble of classifiers that uses a cost matrix. In terms of ensembles, the Balanced Random Forest [8] builds each decision tree using a stratified sample of the entire dataset. A similar approach is the Easy Ensemble [32], which also divides the classification problem into balanced sub-problems. Furthermore, each of its base classifiers is the result of the iteration of Adaboost [36] over each balanced sub-problems. More recently, the authors in [29] proposed the creation of an ensemble using cost-sensitive trees. The weighted majority vote is given only by the best base classifiers selected by a genetic algorithm. The same authors proposed a new version which excludes the ensemble size limitation as a parameter [30].

In batch environments, we have the facility to be able to query how many times the dataset is needed. Therefore, the solutions mentioned above are feasible to deal with imbalanced data. However, when working with potentially infinite and imbalanced data streams, such techniques must be adapted to meet the needs of a model for data stream mining, i.e., working within the limits of processing time and memory usage [4].

Focusing on the classification of data streams, the Selective Recursive Approach [9] (SERA) consumes the data stream as successive batches and reuses the positive examples from previous batches to decrease the degree of unbalance. Therefore, it deals with concept drift by re-training a model using a balanced set with the positive instances seen so far. The Learn++.CDS [15] and Learn++.NIE [14] are two variations for unbalanced data streams from Learn++.NSE [19]. Both variations maintain the Learn++.NSE feature of interpreting a data stream as successive batches. The CDS variation uses SMOTE [7] to resample instances that are classified incorrectly in a batch. A new classifier is trained in the enriched batch and may replace a current ensemble member if it does not satisfy the efficiency condition for the enriched batch. The NIE variation aims to train each of the classifiers in balanced subsets. The final prediction of both variations is given by the weighted combination of the votes of each classifier member of the ensemble.

To handle imbalanced data streams, authors in [23] proposed two neural networks that assign misclassification costs in its objective functions. And one of them has an adaptive cost matrix capable of changing the weights involved whenever a concept drift occurs. In [24], the same authors employ an error minimization function that assigns high costs to the minority class in the neural network. The authors in [41], use a cost matrix that is optimized by a genetic algorithm during the training step.

More recently, a new version of Adaptive Random Forest (ARF) for imbalanced data streams was introduced. The ARF with Resampling (ARF_{RE}) combines weights with Poisson's distribution output for a given instance [5]. As a result, ARF_{RE} changes the chance of an example being used in the training phase based on its class distribution (Equation 2) and harverst it by resampling it following a Poisson distribution with parameter λ. The higher the incidence of a c class (S_c), in a data stream with S_n of observed examples, the lower the weight assigned to the instances which belong to this class.

$$weight(S_c, S_n, \lambda) = \frac{100 - \frac{S_c \times 100}{S_n}}{100} \times p, \textbf{ s.t. } p \sim Poisson(\lambda) \quad (2)$$

Sampling strategies are common as an approach to handle imbalanced data streams. However, oversampling the minority instances or undersampling the majority can result in a slower training phase or information loss, respectively. There are solutions such as ARF_{RE}, which applies both techniques without reproducing its drawbacks. Nonetheless, it is unknown the impact of resampled data streams in concept drift detection because the classifier interprets the problem in a different way of its nature [5]. For these reasons, the use of the cost matrix approach to influence the decision-making of a classifier is an option to handle imbalanced data streams.

3 APPLYING MISCLASSIFICATION COSTS TO ADAPTIVE RANDOM FOREST

Ensembles allow the use of different imbalance-driven strategies to reduce the impact of class-skewed data in the classification task as seen in Easy Ensemble, Balanced Random Forest and MetaCost. However, not all imbalanced ensemble-based approaches work or adapt to data stream environments due to a variety of factors, e.g., extensive computational calculations and the need for analyzing the entire dataset. Therefore, we opted for cost-based strategies to mitigate the impact of imbalance on model construction.

The Adaptive Random Forest (ARF) is an algorithm proposed in [26] for the data stream environments. Using the main features

present in the popular Random Forest (RF) [6], ARF builds an ensemble of incremental decision trees capable of dealing with potentially infinite data flows and concept drift. In practice, ARF induces a set of Hoeffding Trees [17] using Online Bagging [34] and a limited and randomly selected number of features to reproduce the RF characteristics. Besides, it uses concept drift detectors for each tree inside the ensemble. Thus, as soon as there is a suspicion of concept drift in the data stream, ARF starts to train new classifiers to replace the current classifier if the change is confirmed. The final decision of the ensemble is given by the weighted majority vote using the accuracy of each base classifier as the weight.

Even though ARF is a state-of-the-art algorithm in the data stream classification, there was no exploratory study of its performance in imbalanced environments by its authors in [26]. In this paper, we present an analysis of ARF performance using imbalanced data sets along with the proposal of a new version of ARF that is sensitive to misclassification costs.

Hereafter, we propose the Cost-sensitive Adaptive Random Forest (CSARF - Algorithm 1), which is a variant of ARF planned to handle class imbalance in data stream classification tasks. To achieved this, the following modifications were made in ARF:

(1) The assignment of weights to each internal tree using Matthews Correlation Coefficient (MCC) instead of accuracy;
(2) The addition of a sliding window to observe the classes distribution over time;
(3) The modification in the learning process (sampling), in order to ensure that all trees train with the instances belonging to the minority class.
(4) The assignment of cost sensitivity with two different strategies: local and global. Details on these strategies are described below.

The evaluation using prequential [20] of the ARF trees prioritizes the participation of classifiers with better performances in the weighted majority vote. However, the use of accuracy (Equation 3) in imbalanced environments is ineffective in evaluating the performance of a classifier. For example, if a dataset contains only 100 minority instances within a set with 10,000 examples, the base accuracy obtained by a model correctly classifying solely the majority class would be of 99%. Therefore, in the CSARF, the base classifiers are weighted according to their performance in the MCC metric (Line 12 of Algorithm 1). The metric MCC [33] (Equation 4) is calculated using the confusion matrix, and it is, among the metrics derived from the confusion matrix, the most informative for binary problems since it incorporates all four information when evaluating a classifier (TP, TN, FP, and FN). By considering the proportion of each class in its formula, it assigns higher values to the classifiers that perform well in both classes [10].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \tag{4}$$

Changes #2 and #3 relate to the use of Online Bagging (OB) to resampling the training set. The OB process used in ARF follows the $Poisson(\lambda = 6)$ distribution to sample the instances for training its decision trees. By not differentiating between classes, there is a probability that minority instances will not be presented for some classifiers of the ensemble. During the training stage, CSARF alleviates this problem by verifying if the recurrent instance is positive

Algorithm 1 Cost-sensitive Adaptive Random Forest. **Symbols**: m: Maximum features evaluated per split; n: Total number of trees ($n = |T|$); δ_w: Warning threshold; δ_d: drift threshold; w_{size}: Sliding window size; cost: Cost matrix; $c(\cdot)$: Change detection method; S: Data stream; B: Set of background trees; $W(t)$: Tree t weight; I_w: Sliding window; C: Cost matrix; $P(\cdot)$: Learning performance estimation function.

```
1:  function CSARF(m, n, δ_w, δ_d, w_SIZE, COST)
2:      T ← CreateTrees(n)
3:      W ← InitWeights(n)
4:      I_w ← w_size
5:      C ← cost
6:      B ← ∅
7:      while HasNext(S) do
8:          (x, y) ← next(S)
9:          I_w ← (x, y)
10:         for all t ∈ T do
11:             ŷ ← predict(t, x)
12:             W(t) ← P(W(t), ŷ, y)
13:             RFTreeTrain(m, t, x, y)
14:             if C(δ_w, t, x, y) then
15:                 b ← CreateTree()
16:                 B(t) ← b
17:             end if
18:             if C(δ_d, t, x, y) then
19:                 t ← B(t)
20:             end if
21:         end for
22:         for all b ∈ B do
23:             RFTreeTrain(m, b, x, y)
24:         end for
25:     end while
26: end function
```

Algorithm 2 RFTreeTrain. **Symbols**: λ: Fixed parameter to Poisson distribution. GP: Grace Period before recalculating heuristics for split test.

```
1:  function RFTreeTrain(m, t, x, y)
2:      k ← Poisson(λ = 6)
3:      if k > 0 OR isPositive(x, y) then
4:          l ← FindLeaf(t, x)
5:          UpdateLeafCounts(l, x, k)
6:          if InstancesSeen(l) ≥ GP then
7:              AttemptSplit(l)
8:              if DidSplit(l) then
9:                  CreateChildren(l, m)
10:             end if
11:         end if
12:     end if
13: end function
```

through the sliding window with w_{size} (Line 4 of Algorithm 1) size, and then, it uses the minority example even if the value obtained of k in $Poisson(\lambda = 6)$ is equal to zero (Line 3 of Algorithm 2).

Formally, the weighted majority vote of a B size ensemble for an x instance is computed by associating a w_j weight with a probability that an instance belongs to the i class given by the j classifier:

$$C(x) = \arg\max_i \sum_{j=1}^{B} w_j p_{ij} \tag{5}$$

The fourth change is based on the use of the cost matrix by changing the weighted majority vote (Equation 5) in two ways using the cost framework $L(x, i)$ (Equation 1). Therefore, the two variations of CSARF are as follows:

- **CSARF-local**: This variation uses the misclassification costs to influence the output of the base classifiers before the combination of votes. Thus, the opinion of the classifiers is influenced, and this may or may not modify the vote obtained after the combination. Mathematically, CSARAF-local computes the majority vote by associating a weight w_j (MCC performance) to the i class that minimizes the sum of the alternative probabilities of the current x class (Equation 6):

$$C(x) = \arg\max_i \sum_{j=1}^{B} w_j \times \left[\arg\min_i \sum_i L(x, i) \right] \tag{6}$$

- **CSARF-global**: The "global" variation is based on changing the probabilities after combining the classifiers. Influencing thus, not the individual opinions of each decision tree, but the final combination of their opinions. Mathematically, CSARF-global after computing the majority vote chooses the i class that minimizes the sum of the alternative probabilities of the current x class (Equation 7):

$$C(x) = \arg\min_i \sum_i \sum_j cost(i, j) \times \left[\sum_{k=1}^{B} w_k p_{jk} \right] \tag{7}$$

3.1 Misclassification cost heuristic

The most common method for constructing a cost matrix is to set a fixed misclassification cost for each class. The values can be chosen through an exhaustive grid search, thus testing various costs to achieve the desired performance. This strategy works in a batch environment, but in data streams, one must consider the variation of class distribution and that the potentially infinite universe of a data stream is unknown.

Strategies that use class distribution as information are alternatives to the use of fixed penalties. In [35], the authors defined weights for each class according to their distribution in the database (Equation 8).

$$classweight(i) = \frac{\#samples}{\#classes \times \#samples(i)} \tag{8}$$

That is, if class A is n times more frequent than class B, it will have its cost divided by n while class B will have its weight multiplied by n. The authors employed this cost to perform majority class under-sampling and minority class oversampling. Yet, in CSARF, the costs are used to influence the classifier's output probabilities. Even though we do not have information about the distribution of a data stream, this method can be applied to the sliding window in CSARF. As data is consumed, the distribution of classes may vary, and as a result, the misclassification costs change as well.

4 EMPIRICAL EVALUATION

In the previous section we showed two approaches to take into account the cost matrix inside the Adaptive Random Forest learning scheme. In order to evaluate these approaches, the framework Massive Online Analysis (MOA) [4] was chosen to implement the tests as the ARF is already available in this environment.

4.1 Experimental Setup

The experimental protocol consists of evaluating whether the null hypothesis that there is no significant difference between the CSARF and the ARF for the imbalanced data sets is true.

We analyze the performance of the models using the prequential or test-then-train strategy, ensuring that the model first tests and then trains with each new instance of the data stream [21]. The first 1,000 examples of each data set were used only to train the model. In order to evaluate the CSARF with similar approaches, we included the ARF_{RE} [5] in the experimental protocol. All classifiers parameters are given in Table 2. Furthermore, these learning algorithms were evaluated in six real data sets with different levels of imbalance and more six synthetic data sets built by two generators:

Santander - santd[1]. Dataset consisting of examples that identify whether a customer will carry out a transaction in the future, regardless of the amount of money involved. It contains 76,020 instances, 370 attributes and a class distribution [96.04%, 3.95%].

Pozzolo Credit Card Fraud Detection - pozzl[2]. Contains credit card transactions from Europe. The 284,807 examples, with 30 attributes, contain only 492 fraudulent transactions, with a distribution of [99.82%, 0.17%].

Electricity - elect[27]. Contains data collected from the Australian electricity market, where prices fluctuate according to market demand and supply every 5 minutes. This dataset contains 45,312 instances described by 8 attributes, and labels determine whether the price has gone up or down. Its distribution of classes is [42.45%, 57.54%].

Give Me Some Credit - credt[3]. The data set available in the Kaggle competition aims to determine the probability of a customer becoming defaulted within the next two years. This data set contains 150,000 instances described by 10 attributes, and its class distribution is of [93.316%, 6.684%].

Airlines - airli[4]. The goal in this data set is to determine the probability of a flight to be delayed. It contains 539,383 instances, 7 attributes and a distribution of [55.46%, 44.54%].

Private Credit Score - privt. A private dataset that represents a credit score scenario. It contains 97,226 instances, 130 attributes, and a class distribution of [26.07%, 73.92%].

Agrawal Generator - agrwl. This generator is capable of synthesizing a binary classification task containing 9 attributes. In

[1] https://www.kaggle.com/c/santander-customer-transaction-prediction/
[2] https://www.kaggle.com/mlg-ulb/creditcardfraud
[3] https://www.kaggle.com/c/GiveMeSomeCredit/
[4] http://kt.ijs.si/elena_ikonomovska/data.html

Table 2: ARF, CSARF and ARF_{RE} parameters

	Value
ensembleSize	10
mfeatureMode	percent
mfeaturePerTreeSize	80%
lambda	6
imbalanceWindowSize	10,000

addition, it allows adding noisy data between attributes according to a uniform random distribution [1]. We parameterize it to generate 1,000,000 instances with the following imbalance variations: **agrwl90** ([90%, 10%]), **agrwl95** ([95%, 5%]), **agrwl99** ([99%, 1%]).

SEA Generator - sea. As the Agrawal Generator, SEA Generator is able to synthesize a binary classification task containing three continuous attributes and employ a noise percentage of the data. The generated versions have 10% noise and are defined as: **sea90** ([90%, 10%]), **sea95** ([95%, 5%]), **sea99** ([99%, 1%]) [37].

The ensemble votes for each instance are used to calculate the performance of the model in the evaluation metrics. Because it is a task of classification of unbalanced data, the use of metric accuracy lacks information to evaluate the classifier as explained previously. Therefore, metrics that consider performance in the minority class, such as the AUC-PR (Area under the precision-recall curve), stand out [12].

The precision (Equation 9) and the recall (Equation 10) are two metrics constructed from the confusion matrix. The first allows us to analyze how many of the positive examples detected are truly positive, while the second makes it possible to evaluate how many positive examples have been detected in their completeness. One can combine both precision and recall using F1 score (Equation 11), that is a harmonic mean between precision and recall for a given class. It is also possible to combine both metrics via the calculation of the Area Under the Precision-Recall Curve (AUC-PR). One can calculate a precision and recall pair for each decision boundary between the positive and negative class and thus construct a curve. The single value obtained through the area below this curve (AUC-PR) provides visualization of the performance of a classifier without the performance in the majority class interfering with the result obtained. The advantages of AUC-PR[5] on the popular AUROC metric are detailed in [12].

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

We evaluated the three models (ARF, CSARF, and ARF_{RE}) in 12 datasets using the Average Recall, AUC-PR, and Average F1 Score. The results are arranged in table 3, where the CSARF-g corresponds to the *global* variation and CSARF-l to the *local* variation.

[5]The library for python scikit-learn made it possible to evaluate the performance of the models in the AUC-PR metric.

Table 3: Results for the experiment

Data set	CLF	Avg. Recall	Avg. F1	AUC-PR
santd	ARF	50.13	49.30	11.83
santd	ARF_{RE}	55.53	47.66	5.34
santd	CSARF-g	**70.38**	52.73	11.90
santd	CSARF-l	65.85	**58.78**	**13.43**
pozzl	ARF	86.36	90.09	**80.95**
pozzl	ARF_{RE}	89.70	72.46	62.66
pozzl	CSARF-g	**91.95**	71.80	80
pozzl	CSARF-l	87.41	**90.47**	79.70
elect	ARF	88.75	88.96	37.49
elect	ARF_{RE}	88.85	88.95	37.46
elect	CSARF-g	**88.94**	88.85	**37.51**
elect	CSARF-l	88.83	**88.98**	37.50
credt	ARF	56.17	58.89	33.90
credt	ARF_{RE}	74.53	64.87	31.95
credt	CSARF-g	**75.45**	62.51	**34.04**
credt	CSARF-l	70.36	**67.85**	33.15
airli	ARF	61.76	61.79	59.30
airli	ARF_{RE}	**62.32**	**62.35**	**60.54**
airli	CSARF-g	61.04	61.06	58.22
airli	CSARF-l	61.42	61.46	58.78
privt	ARF	63.91	65.72	58.23
privt	ARF_{RE}	70.33	**70.40**	58.33
privt	CSARF-g	**71.47**	69.49	58.32
privt	CSARF-l	69.14	69.57	**58.50**
agrwl90	ARF	83.82	88.27	**89.64**
agrwl90	ARF_{RE}	**93.44**	87.07	88.35
agrwl90	CSARF-g	92.67	82.93	89.47
agrwl90	CSARF-l	89	**89.37**	87.96
agrwl95	ARF	68.46	75.77	79.42
agrwl95	ARF_{RE}	88.62	80.23	70.63
agrwl95	CSARF-g	**90.83**	75.83	**81.22**
agrwl95	CSARF-l	81.29	**84.71**	76.97
agrwl99	ARF	50.11	49.97	11.52
agrwl99	ARF_{RE}	50.36	50.36	1.11
agrwl99	CSARF-g	**63.08**	**54.64**	**11.73**
agrwl99	CSARF-l	50.60	50.91	3.63
sea90	ARF	53.36	53.97	34.25
sea90	ARF_{RE}	**84.46**	**70.11**	**34.75**
sea90	CSARF-g	83.02	69.27	34.29
sea90	CSARF-l	63.90	64.05	34.42
sea95	ARF	50.21	49.23	19.06
sea95	ARF_{RE}	**83.04**	**61.86**	**20.13**
sea95	CSARF-g	76.16	61.28	19.07
sea95	CSARF-l	52.98	53.91	19.18
sea99	ARF	50	49.75	3.22
sea99	ARF_{RE}	54.97	**52.43**	**3.42**
sea99	CSARF-g	**59.42**	52.18	3.12
sea99	CSARF-l	50.03	49.83	3.11

4.2 Empirical Analysis

From the obtained results, some interesting points are highlighted w.r.t. the two strategies of assigning misclassification costs.

Predictive models may exhibit different behaviors for data sets with similar levels of imbalances but different class separability. In *santd*, we note that both versions of CSARF performed better than ARF and ARF_{RE} with significant gains. However, in the *pozzl* experiment, ARF showed an interesting performance for a problem where 99.82% of the examples are from the majority class. Even so, the imbalance data-driven versions were notable because both (ARF_{RE} and CSARF) were able to perform better than ARF on the average recall. In this case, the gain scale is smaller than the one observed for the *santd*, but it is significant. It is also worthy of highlighting that ARF alone performed better than its variants in the AUC-PR metric with slightly higher performance than CSARF versions.

When evaluating CSARF and ARF_{RE} on low imbalance data, i.e., *elect* and *airli*; we note that the performance gain is low. For *airli*, CSARF versions performed even worse than the original ARF version. However, as this is not a problem with a high degree of imbalance, the increase in the minority class rating rate is not as significant. Yet, when the cost involved in classifying one class into another is high, using a classifier that can distinguish the minority class from the others is crucial. This is clear in *credit* and *privt* experiments, where CSARF and ARF_{RE} performed better, especially CSARF-g with better average recall and AUC-PR results in data set *credit*.

The evaluation of the models with synthetic generators showed that CSARF-g and CSARF-l perform better than other approaches when the degree of imbalance intensifies. The exception is ARF_{RE}, which stands out with the best results in almost all unbalanced scenarios using the SEA generator.

To verify if there is no statistically significant difference between the CSARF, ARF_{RE}, and the ARF is true, the Nemenyi test was performed for each performance metric following the instructions of [13]. The critical distance chart contains the method ranking, where the lowest placement represents the best performing algorithm. For a confidence level of 95%, the results suggest that CSARF-g is the best among the other approaches evaluated in average recall and that there is no significant difference between the last two ranks (CSARF-l and ARF) and the first ones (CSARF-g and ARF_{RE}) (Figure 1). One can verify that both CSARF-g and ARF_{RE} performed better than the ARF in this metric and there's a significant difference between CSARF-g and ARF. In the average F1 metric (Figure 2), CSARF-l stands out with better performance and meaningfully difference when compared with ARF. In this scenario, there is no significant difference between the three last ranks (ARF_{RE}, CSARF-g and ARF). However, when evaluating the classifiers in the AUC-PR metric (Figure 3), we noticed that there is no significant difference between the performances of the evaluated methods. Even so, both versions of CSARF ranked first in evaluating the classifiers in each of the metrics. Even if the performance gain is apparently low, in imbalanced datasets the gain in minority class prediction is significant given its rarity.

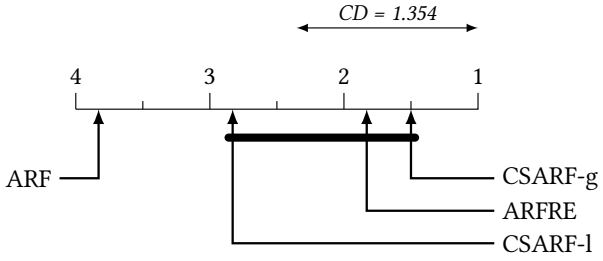

Figure 1: Critical differences chart for the Average Recall obtained by all the CSARF variations, ARF and ARF_{RE}.

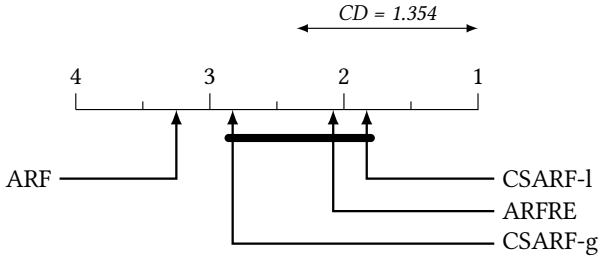

Figure 2: Critical differences chart for the Average F1 obtained by all the CSARF variations, ARF and ARF_{RE}.

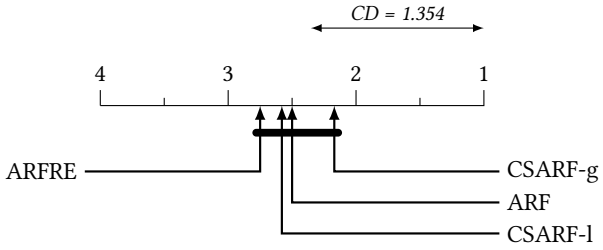

Figure 3: Critical differences chart for the AUC-PR obtained by all the CSARF variations, ARF and ARF_{RE}.

5 CONCLUSION AND FUTURE WORK

In this paper, an empirical study was presented on real and synthetic imbalanced datasets. The used datasets present different classification problems with different levels of imbalance. We compared the performance of the traditional Adaptive Random Forest, and one of its variants that perform data resampling, against our proposed method that is based on misclassification cost. The comparative analysis between ARF, ARF_{RE}, and CSARF made it possible to evaluate the beneficial impact of the use of strategies based on misclassification costs.

For future works, the objective is to analyze the CSARF performance for non-binary classification tasks that also present imbalance. In future work, the use of dynamic misclassification costs that accompany the relevant changes of a data stream should be also be assessed, both to change the classifier's decision and to restate the minority instances.

ACKNOWLEDGMENT

This work was financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), through the Fundação Araucária program to support the scientific and technological development of the state of Paraná.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. 1993. Database mining: A performance perspective. *IEEE transactions on knowledge and data engineering* 5, 6 (1993), 914–925.

[2] Gustavo EAPA Batista, Andre CPLF Carvalho, and Maria Carolina Monard. 2000. Applying one-sided selection to unbalanced datasets. In *Mexican International Conference on Artificial Intelligence*. Springer, 315–325.

[3] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J Christopher Westland. 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 50, 3 (2011), 602–613.

[4] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. 2010. Moa: Massive online analysis. *Journal of Machine Learning Research* 11, May (2010), 1601–1604.

[5] Luis E Boiko, Heitor Gomes, Albert Bifet, and Luiz S Oliveira. 2019. Adaptive Random Forests with Resampling for Imbalanced data Streams. *International Joint Conference on Neural Networks (IJCNN)* (2019).

[6] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[8] Chao Chen, Andy Liaw, Leo Breiman, et al. 2004. Using random forest to learn imbalanced data. *University of California, Berkeley* 110 (2004), 1–12.

[9] Sheng Chen and Haibo He. 2009. Sera: selectively recursive approach towards nonstationary imbalanced stream data mining. In *2009 International Joint Conference on Neural Networks*. IEEE, 522–529.

[10] Davide Chicco. 2017. Ten quick tips for machine learning in computational biology. *BioData mining* 10, 1 (2017), 35.

[11] Andrea Dal Pozzolo, Reid Johnson, Olivier Caelen, Serge Waterschoot, Nitesh V Chawla, and Gianluca Bontempi. 2014. Using HDDT to avoid instances propagation in unbalanced and evolving data streams. In *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 588–594.

[12] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 233–240.

[13] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* 7, Jan (2006), 1–30.

[14] Gregory Ditzler and Robi Polikar. 2010. An ensemble based incremental learning framework for concept drift and class imbalance. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[15] Gregory Ditzler and Robi Polikar. 2013. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* 25, 10 (2013), 2283–2301.

[16] Pedro Domingos. 1999. Metacost: A general method for making classifiers cost-sensitive. In *KDD*, Vol. 99. 155–164.

[17] Pedro Domingos and Geoff Hulten. 2000. Mining high-speed data streams. In *Kdd*, Vol. 2. 4.

[18] Charles Elkan. 2001. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, Vol. 17. Lawrence Erlbaum Associates Ltd, 973–978.

[19] Ryan Elwell and Robi Polikar. 2009. Incremental learning of variable rate concept drift. In *International Workshop on Multiple Classifier Systems*. Springer, 142–151.

[20] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. 2009. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 329–338.

[21] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. 2013. On evaluating stream learning algorithms. *Machine learning* 90, 3 (2013), 317–346.

[22] Xingyu Gao, Zhenyu Chen, Sheng Tang, Yongdong Zhang, and Jintao Li. 2016. Adaptive weighted imbalance learning with application to abnormal activity recognition. *Neurocomputing* 173 (2016), 1927–1935.

[23] Adel Ghazikhani, Reza Monsefi, and Hadi Sadoghi Yazdi. 2013. Online cost-sensitive neural network classifiers for non-stationary and imbalanced data streams. *Neural computing and applications* 23, 5 (2013), 1283–1295.

[24] Adel Ghazikhani, Reza Monsefi, and Hadi Sadoghi Yazdi. 2014. Online neural network model for non-stationary and imbalanced data stream classification. *International Journal of Machine Learning and Cybernetics* 5, 1 (2014), 51–62.

[25] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. 2017. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 23.

[26] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning* 106, 9-10 (2017), 1469–1495.

[27] Michael Harries and New South Wales. 1999. Splice-2 comparative evaluation: Electricity pricing. (1999).

[28] Bartosz Krawczyk, Mikel Galar, Łukasz Jeleń, and Francisco Herrera. 2016. Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. *Applied Soft Computing* 38 (2016), 714–726.

[29] Bartosz Krawczyk, Michal Wozniak, and Gerald Schaefer. 2011. Improving minority class prediction using cost-sensitive ensembles. In *16th Online World Conference on Soft Computing in Industrial Applications*.

[30] Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. 2014. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing* 14 (2014), 554–562.

[31] Ryan N Lichtenwalter and Nitesh V Chawla. 2009. Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 53–75.

[32] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. 2009. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39, 2 (2009), 539–550.

[33] Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405, 2 (1975), 442–451.

[34] Nikunj C Oza. 2005. Online bagging and boosting. In *2005 IEEE international conference on systems, man and cybernetics*, Vol. 3. Ieee, 2340–2345.

[35] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. 2018. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018).

[36] Robert E Schapire. 1999. A brief introduction to boosting. In *Ijcai*, Vol. 99. 1401–1406.

[37] W Nick Street and YongSeog Kim. 2001. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 377–382.

[38] Konstantinos Topouzelis. 2008. Oil spill detection by SAR images: dark formation detection, feature extraction and classification algorithms. *Sensors* 8, 10 (2008), 6642–6659.

[39] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. 2007. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*. ACM, 935–942.

[40] Bianca Zadrozny, John Langford, and Naoki Abe. 2003. Cost-Sensitive Learning by Cost-Proportionate Example Weighting.. In *ICDM*, Vol. 3. 435.

[41] Chong Zhang, Kay Chen Tan, and Ruoxu Ren. 2016. Training cost-sensitive deep belief networks on imbalance data problems. In *2016 international joint conference on neural networks (IJCNN)*. IEEE, 4362–4367.