ORIGINAL RESEARCH
published: 23 December 2021
doi: 10.3389/fdata.2021.715320

Weighting Methods for Rare Event Identification From Imbalanced Datasets

*Jia He and Maggie X. Cheng**

Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, United States

In machine learning, we often face the situation where the event we are interested in has very few data points buried in a massive amount of data. This is typical in network monitoring, where data are streamed from sensing or measuring units continuously but most data are not for events. With imbalanced datasets, the classifiers tend to be biased in favor of the main class. Rare event detection has received much attention in machine learning, and yet it is still a challenging problem. In this paper, we propose a remedy for the standing problem. Weighting and sampling are two fundamental approaches to address the problem. We focus on the weighting method in this paper. We first propose a boosting-style algorithm to compute class weights, which is proved to have excellent theoretical property. Then we propose an adaptive algorithm, which is suitable for real-time applications. The adaptive nature of the two algorithms allows a controlled tradeoff between true positive rate and false positive rate and avoids excessive weight on the rare class, which leads to poor performance on the main class. Experiments on power grid data and some public datasets show that the proposed algorithms outperform the existing weighting and boosting methods, and that their superiority is more noticeable with noisy data.

Keywords: imbalanced dataset, bias, classification, machine learning, rare event

OPEN ACCESS

Edited by:
Fan Jiang,
University of Northern British Columbia Canada, Canada

Reviewed by:
Hideitsu Hino,
Institute of Statistical Mathematics (ISM), Japan
Carson Leung,
University of Manitoba, Canada

****Correspondence:***
Maggie X. Cheng
maggie.cheng@iit.edu

Specialty section:
This article was submitted to Data Mining and Management, a section of the journal Frontiers in Big Data

Received: *26 May 2021*
Accepted: *04 November 2021*
Published: *23 December 2021*

Citation:
He J and Cheng MX (2021) Weighting Methods for Rare Event Identification From Imbalanced Datasets. Front. Big Data 4:715320. doi: 10.3389/fdata.2021.715320

1 INTRODUCTION

In this paper, we study the problem of learning with an imbalanced dataset. In classification, this is also called rare events problem, in which there are thousands of times fewer *yes* cases than *no* cases. The yes cases are called events. Usually the events are what we are interested in, which may have very few occurrences while the nonevent cases are abundant. This is typical in network monitoring applications, where data representing events are only a tiny portion of the entire dataset. For instance, we may have a fault or anomaly observed in 1 month's worth of data while all other observations are nonevents. Using machine learning approach forevent detection and identification would require training a machine learning algorithm with these data, but the scarce representation of events in the dataset makes learning the rare event difficult.

It has been reported in the statistics literature that rare events are difficult to predict [see King and Zeng (2001) and others]. In Weiss (2004), it is pointed out that with imbalanced datasets, the learning algorithms are biased in favor of the class priors. The statistical procedure to predict the event, such as Logistic Regression, often underestimates the probability of the rare events. Such procedures will have a high overall prediction accuracy mainly due to the correct predictions on the large number of nonevent cases, but the *recall* metrics, defined as the fraction of true positives that have been successfully predicted, is extremely low. Such performance does not serve the purpose of event detection, since what we are interested in is the event. For event detection in a networked system, having a missed detection on important events has more detrimental effect than having a false alarm. Oftentimes we are willing to improve the detection rate even if it will generate more false positives. However, using the standard classifiers, the cost associated with misclassification on either class has the same contribution in the cost function.

To address the problem, we need to give more importance to the rare class in the cost function. This can be done either by directly changing the number of examples in each class in the training set, i.e., by sampling (Cahyana et al., 2019), or by changing the weights of the classes, which also changes the distribution of the classes. A review article (Li and Mao, 2014) provided an in-depth study of data sampling strategies and weight-modification strategies from the Bayesian classification point of view. In addition, the number of classes is also not limited to two classes. It can be extended to multiple-classes (Tanha et al., 2020).

Sampling has the merit of simplicity but it also has limitations. First, there are two forms of rarity as pointed out in Weiss (2004): absolute rarity and relative rarity. While relative rarity can be corrected by under-sampling the main class, absolute rarity can only resort to oversampling the rare class. However there is an issue with this approach—in case there are outliers in the data, the oversampled outliers will ruin the prediction performance. Moreover, under-sampling may also cause loss of information. It is necessary to consider an alternative way to address the rarity issue.

In this paper, we focus on weighting methods. Unlike the previous weighting methods that use the population information or the relative rarity in the sample to decide the weights, we develop algorithms to compute the weights during the training process. The weighting algorithms can work with any classifier. In this paper, we use Logistic Regression, Random Forest, and Support Vector Machine to demonstrate its effectiveness. The proposed algorithms have the advantages of not relying on unknown population information, and being able to improve the prediction performance of the rare class with a controllable tradeoff with the main class. Most importantly, this is the best approach to deal with absolute rarity, which poses great challenges to other methods.

In practice, a user does not have to choose between a weighting method and a sampling method. An ensemble approach that combines sampling techniques and weighting techniques can achieve the best of the two worlds. For instance, (Guo and Viktor, 2004) combines generating synthetic data and boosting procedures to improve the predictive accuracies of both the majority and minority classes. An improvement on the weighing method also contributes to the ensemble techniques.

The rest of the paper is organized as follows: in **Section 2**, we cover the preliminaries for classification with imbalanced dataset; in **Section 3**, we propose two weighting algorithms, DiffBoost, and AdaClassWeight. They can address both forms of rarity by adaptively adding weights and combining weighting with boosting; and subsequently in **Section 4**, we explain how to train a classifier under the computed weights; in **Section 5**, we provide performance results for the proposed algorithm, along with comparison with related methods; and in **Section 6**, we conclude the paper with outlook for future work.

2 CLASSIFICATION WITH IMBALANCED DATASETS

Among many others (Japkowicz et al., 1995; Liu et al., 2006; Ertekin et al., 2007), using weights is a fundamental approach to address the data imbalance problem in classification. We focus on the use of class weights in this paper, in which we add class weights to the loss function, making it more expensive to have a classification error in the rare class. This is done by assigning the rare class a larger weight and the main class a smaller weight. Weighting is also considered a type of cost-sensitive learning method (Pazzani et al., 1994). In cost-sensitive learning, the cost associated with misclassifying a rare class outweighs the cost of correctly classifying the main class. In King and Zeng (2001), weights are decided based on sample distribution in the population: the rare class weight $w^+ = \tau/\bar{y}$ and the main class weight $w^- = (1-\tau)/(1-\bar{y})$, where τ is the fraction of the rare class in the population, and $\bar{y}$ is the fraction in the sample, respectively. In some applications, population information may be straightforward to know, such as in political activities (King and Zeng, 2001). However, in most other applications, we do not know the class distribution in the population. For convenience, many resort to using the sample information, i.e., in the training set if there are N^+ examples in the rare class, and N^- examples in the main class, the weight would be N^-/N^+ for the rare class and 1 for the main class. This method, as we will see later in this paper, has the disadvantage of not considering the absolute rarity, and also not having control over the trade off between the false positive rate and the false negative rate. In case we need to prioritize the rare class, we cannot improve the performance on the rare class further since the fixed weights only reflect the ratio of the examples in the sample.

Similar to the class-weighted methods, there are previous work that use individual weights in the classification algorithms. We briefly review some existing work that are developed based on the idea of introducing a cost for each individual example. The weight update rule was first introduced in AdaBoost (Freund and Schapire, 1995) to force the classifier to be biased towards the minority class.

Given the number of iterations T, and training data $\{(x_i, y_i), i = 1, \ldots, N\}$, the AdaBoost algorithm computes the sample weight distribution D_t at the t-th iteration. h_t is the classifier trained under weight distribution D_t. The weight update rule in AdaBoost is given by

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t h_t(x_i) y_i\right)}{Z_t}, \quad (1)$$

where α_t is a weight update parameter that needs to be computed in each iteration. Z_t is normalization factor defined as

$$Z_t = \sum_i D_t(i)\exp\left(-\alpha_t h_t(x_i) y_i\right). \tag{2}$$

Based on the weight update rule of AdaBoost, later works AdaC1, AdaC2, and AdaC3 from Sun et al. (2007), CSB1 and CSB2 from Ting (2000), and Adacost from Fan et al. (1999) were developed by associating a cost $C_i \geq 0$ with individual examples in **Eq. 1**. Examples from the minority class are associated with larger costs than those from the majority class.

- AdaC1 modifies **Eq. 1** by introducing C_i inside the exponent,

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t C_i h_t(x_i) y_i\right)}{Z_t}. \tag{3}$$

- AdaC2 adds a cost C_i outside the exponent of **Eq. 1**,

$$D_{t+1}(i) = \frac{C_i D_t(i)\exp\left(-\alpha_t h_t(x_i) y_i\right)}{Z_t}. \tag{4}$$

- AdaC3 can be considered as a combination of AdaC1 and AdaC2, in which C_i is included both inside and outside the exponent,

$$D_{t+1}(i) = \frac{C_i D_t(i)\exp\left(-\alpha_t C_i h_t(x_i) y_i\right)}{Z_t}. \tag{5}$$

- AdaCost also uses a cost inside the exponent of **Eq. 1**, however, instead of directly using cost item C_i, it defines a cost adjustment function $\Gamma_{1_{(h_t(x_i),y_i)}}$ based on C_i,

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t \Gamma_{1_{(h_t(x_i),y_i)}} h_t(x_i) y_i\right)}{Z_t}, \tag{6}$$

where $\Gamma_{1_{(h_t(x_i),y_i)}}$ can be set as $\Gamma_+ = -0.5C_i + 0.5$ if classified correctly, and $\Gamma_- = 0.5C_i + 0.5$ otherwise.

All the aforementioned methods involve using a cost. A common drawback of them is that one must manually determine the "optimal" cost, which is predetermined. Arbitrarily selected costs can result in poor classification performance as shown in **Section 5**. However, there is no better algorithm than exhaustive search to decide the costs. This motivates an algorithm that computes the weights solely from the data and does not depend on any hyper-parameter.

3 THE PROPOSED WEIGHTING METHODS

Throughout this paper, we assume the rare class is the positive class. Let N^+ be the number of examples in the rare class, and N^- the number of examples in the main class, and $N^+ \ll N^-$. The class weights are denoted as w^+ for the rare class and w^- for the main class.

Through numerous tests, it is observed that what makes it difficult to learn from an imbalanced dataset is not the relative ratio of the rare class to the main class, rather it is the small number of examples in the rare class. In other words, the absolute rarity matters much more than the relative rarity. This is especially true for Logistic Regression, as the logistic model is often estimated by using maximum likelihood estimation and inherently has the "small-sample bias" issue (King and Zeng, 2001). The simple ratio-based algorithm that uses $w^+/w^- = N^-/N^+$ will only use the ratio information regardless of the sample size and is deemed unable to find the optimal weights. For a dataset with $(N^+, N^-) = (200, 2000)$, a weight ratio of $w^+/w^- = 10$ gives too much weight to the rare class, leading to overfitting the rare class; and for a dataset with $(N^+, N^-) = (2, 10)$, using a weight ratio of $w^+/w^- = 5$ is not enough.

TABLE 1 | Spam Data. The number of examples in the training set is $(N^+, N^-) = (200, 2000)$.

	Training		Test	
	Recall	Precision	Recall	Precision
$w^+/w^- = 2$	0.819	0.804	0.776	0.752
$w^+/w^- = 5$	0.910	0.716	0.870	0.676
$w^+/w^- = 7.5$	0.92	0.668	0.887	0.627
$w^+/w^- = 10$	0.924	0.491	0.897	0.440

TABLE 2 | Spam Data. The number of examples in the training set is $(N^+, N^-) = (2, 10)$.

	Training		Test	
	Recall	Precision	Recall	Precision
$w^+/w^- = 5$	0.800	0.559	0.580	0.239
$w^+/w^- = 7.5$	0.950	0.388	0.620	0.223
$w^+/w^- = 10$	1.000	0.382	0.640	0.206
$w^+/w^- = 12$	1.000	0.382	0.660	0.183

We use two experiments on the Spam data to demonstrate the effect of the absolute rarity. In the first experiment, the training set has a total of 2,200 examples, with $(N^+, N^-) = (200, 2000)$. We compare results using a sequence of weights: $w^+/w^- \in \{2, 5, 7.5, 10\}$. **Table 1** shows the results. It is noted that for the simple ratio-based algorithm using $w^+/w^- = 10$, while the recall is the highest, the precision is the lowest for both training and test, and the sum of recall and precision is the lowest. Therefore, the results support the claim that $w^+/w^- = 10$ leads to overfitting on the rare class. On the other hand, the algorithm with $w^+/w^- = 5$ has competitive performance, with a close-to-the-highest recall and the highest sum of recall and precision, indicating that $w^+/w^- = 5$ is the best among the four options.

For the second experiment, the training set has a total of 12 examples, with $(N^+, N^-) = (2, 10)$. We used a sequence of weights: $w^+/w^- \in \{5, 7.5, 10, 12\}$, and the results are shown in **Table 2**. It is noted that simply setting $w^+/w^- = 5$ according to the ratio of the examples in the training set is not enough to have a high recall. The algorithm with $w^+/w^- = 12$ has the highest recall, and the algorithm with $w^+/w^- = 10$ has a close-to-the-highest recall and the highest sum for recall and precision.

The experiments verified that it is the number of examples in the rare class that determines the error. Therefore, it is necessary to look beyond the ratio and develop an algorithm to find the class weights. We present two algorithms, Differentiated Boosting (DiffBoost) and Adaptive Class Weights (AdaClassWeight).

3.1 A Boosting Style Weighting Method: DiffBoost

We define two index sets $I^+ = \{i: y_i = 1\}$, and $I^- = \{i: y_i = -1\}$. We use + to denote the rare class and − to denote the main class. In the algorithm DiffBoost, we compute the weights iteratively under an overarching boosting framework.

Algorithm 1. DiffBoost.

```
Input: $\{(x_i, y_i), i = 1, \ldots, N\}$, $x_i \in \chi$, $y_i \in \{-1, 1\}$
Initialization: for $t = 1$,
  $w_t^+ = w_t^- = 1$
  $D_t(i) = \frac{1}{N^+}, \forall i \in I^+$
  $D_t(i) = \frac{1}{N^-}, \forall i \in I^-$
1 for $t \leftarrow 1$ to T
2     do
3         Train the classifier $h_t : \chi \to \{-1, 1\}$ under class weights $w_t^+$ and $w_t^-$
4         Compute the weighted class error:
            $\epsilon_t^+ = \sum_{i \in I^+} D_t(i) \cdot \mathbb{1}_{(h_t(x_i) \neq y_i)}$
            $\epsilon_t^- = \sum_{i \in I^-} D_t(i) \cdot \mathbb{1}_{(h_t(x_i) \neq y_i)}$
5         Compute $\alpha_t = \max\left\{0, \frac{1}{2} \ln \frac{1-\epsilon_t^+}{\epsilon_t^+}\right\}$
6         Update class weights:
            $w_{t+1}^+ = w_t^+ \exp(\epsilon_t^+)$
            $w_{t+1}^- = w_t^- \exp(\epsilon_t^-)$
7         Update sample weights:
            $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t^+}, \forall i \in I^+$
            $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t^-}, \forall i \in I^-$
            where $Z_t^+ = \sum_{i \in I^+} D_t(i)\exp(-\alpha_t y_i h_t(x_i))$,
            and $Z_t^- = \sum_{i \in I^-} D_t(i)\exp(-\alpha_t y_i h_t(x_i))$
8 Final prediction: $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

The classifier h_t maps an element in the feature space χ to a label. Z_t^+ and Z_t^- are chosen such that $\sum_{i \in I^+} D_{t+1}(i) = 1$, and $\sum_{i \in I^-} D_{t+1}(i) = 1$, and therefore $D_{t+1}(i)$, $i \in I^+$ will be a distribution, and $D_{t+1}(i)$, $i \in I^-$ will be a distribution. This is the invariant of the algorithm, as it holds from initialization until the algorithm terminates.

Although the algorithm seems to operate symmetrically on both the rare class and the main class, it actually boosts the performance of the rare class more than the main class. This is due to the fact that $N^+ \ll N^-$, and therefore $D_t(i)$ takes larger values for $i \in I^+$ than for $i \in I^-$. Initially, the main class is doing well due to the large number of examples, so we have $\epsilon+ > \epsilon-$ after the first iteration, and then w^+ becomes larger than w^-. The algorithm starts to behave in favor of the rare class.

This boosting algorithm has the property that as the iteration number T increases, the training error for the rare class monotonically decreases. We outline the proof in the following. The training error of the rare class is given by $\frac{1}{N^+} \sum_{i \in I^+} 1_{(H(x_i) \neq y_i)}$.

Theorem 1. $\frac{1}{N^+} \sum_{i \in I^+} 1_{(H(x_i) \neq y_i)}$ *asymptotically converges to zero as* $T \to \infty$.

The proof of Theorem 1 is straightforward from Lemma 1 and Lemma 2.

Lemma 1. *The training error of the rare class has the following bound:*

$$\frac{1}{N^+} \sum_{i \in I^+} 1_{(H(x_i) \neq y_i)} \leq \prod_{t=1}^{T} Z_t^+.$$

Proof. Let $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$, i.e., the final prediction $H(x) = \mathrm{sign}(f(x))$. From the update rule of $D_{t+1}(i)$, and using telescoping, we have that $\forall i \in I^+$,

$$\begin{aligned}
&D_{T+1}(i) \\
&= D_T(i) \cdot \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T^+} \\
&= D_1(i) \cdot \frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1^+} \cdots \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T^+} \\
&= \frac{1}{N^+} \cdot \frac{\exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right)}{\prod_{t=1}^{T} Z_t^+} \\
&= \frac{1}{N^+} \cdot \frac{\exp(-y_i f(x_i))}{\prod_{t=1}^{T} Z_t^+}
\end{aligned}$$

Next, we show that the training error of the rare class is bounded from above by $\prod_{t=1}^{T} Z_t^+$.

Recall that $H(x_i) = \mathrm{sign}(f(x_i))$.

- If y_i and $f(x_i)$ have the same sign, then $1_{(H(x_i) \neq y_i)} = 0$, and $0 < \exp(-y_i f(x_i)) < 1$, thus $1_{(H(x_i) \neq y_i)} \leq \exp(-y_i f(x_i))$.
- If y_i and $f(x_i)$ have different signs, $1_{(H(x_i) \neq y_i)} = 1$, and $\exp(-y_i f(x_i)) > 1$, thus $1_{(H(x_i) \neq y_i)} \leq \exp(-y_i f(x_i))$ still holds.

Combining both cases, we have

$$\begin{aligned}
\frac{1}{N^+} \sum_{i \in I^+} 1_{(H(x_i) \neq y_i)} &\leq \frac{1}{N^+} \sum_{i \in I^+} \exp(-y_i f(x_i)) \\
&= \sum_{i \in I^+} D_{T+1}(i) \prod_{t=1}^{T} Z_t^+ \\
&\overset{(a)}{=} \prod_{t=1}^{T} Z_t^+
\end{aligned}$$

(a) is due to that $\sum_{i \in I^+} D_{T+1}(i) = 1$, which is the invariant of the algorithm.

Lemma 2. $\prod_{t=1}^{T} Z_t^+$ *decreases monotonically as the iteration number T increases.*

Proof.

$$\begin{aligned}
Z_t^+ &= \sum_{i \in I^+} D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \\
&= \sum_{i \in I^+, h_t(x_i) = y_i} D_t(i) \exp(-\alpha_t y_i h_t(x_i)) + \sum_{i \in I^+, h_t(x_i) \neq y_i} D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \\
&= e^{-\alpha_t} \sum_{i \in I^+, h_t(x_i) = y_i} D_t(i) + e^{\alpha_t} \sum_{i \in I^+, h_t(x_i) \neq y_i} D_t(i) \\
&= e^{-\alpha_t}(1 - \epsilon_t^+) + e^{\alpha_t} \epsilon_t^+
\end{aligned}$$

Plugging in $\alpha_t = \max\left\{0, \frac{1}{2} \ln \frac{1-\epsilon_t^+}{\epsilon_t^+}\right\}$, we have the following,

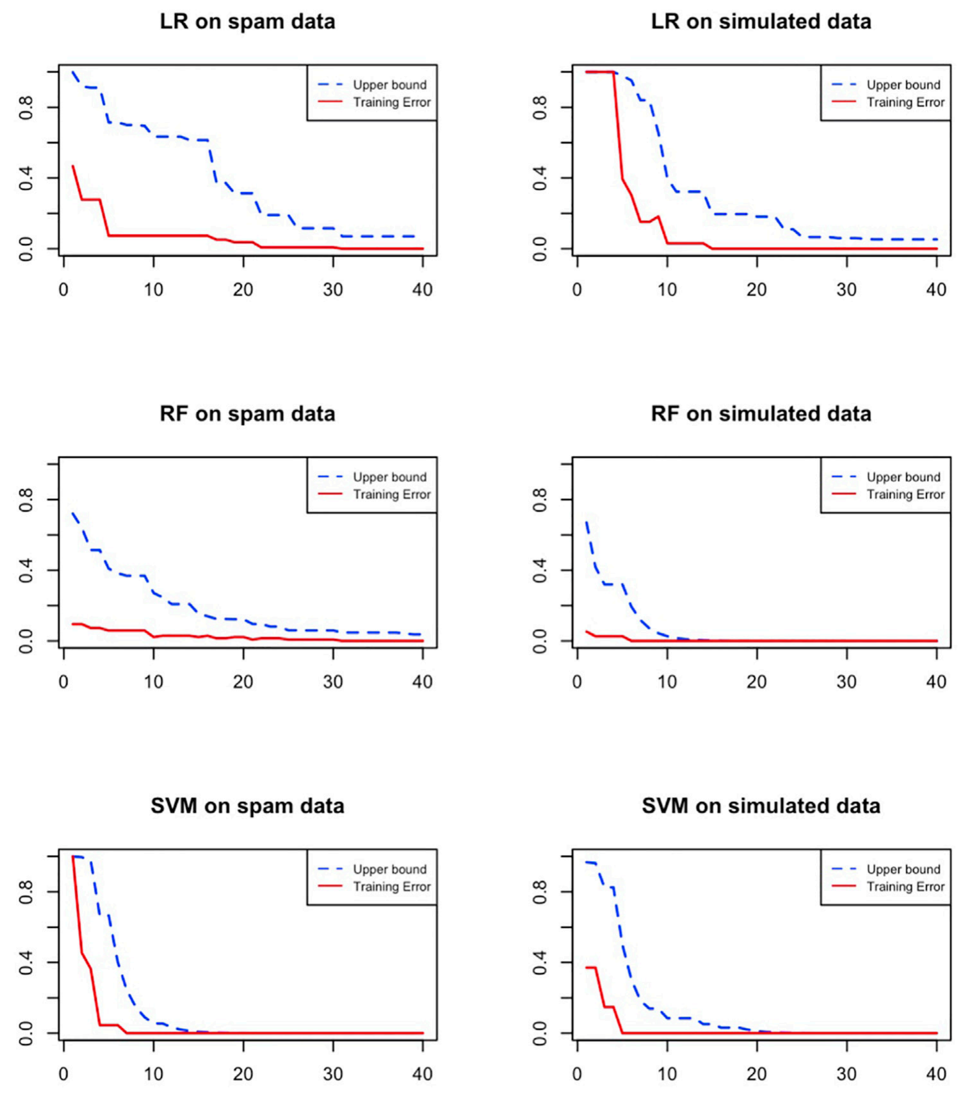

FIGURE 1 | The convergence of the DiffBoost algorithm. Training error of the rare class asymptotically converges to zero as its upper bound decreases monotonically.

- When $\epsilon_t^+ < \frac{1}{2}$, $\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t^+}{\epsilon_t^+}$,

$Z_t^+ = \sqrt{\frac{\epsilon_t^+}{1-\epsilon_t^+}}\,(1-\epsilon_t^+) + \sqrt{\frac{1-\epsilon_t^+}{\epsilon_t^+}}\,\epsilon_t^+ < 1.$

- When $\epsilon_t^+ \geq \frac{1}{2}$, $\alpha_t = 0$ (i.e., this iteration has no contribution to the final prediction), and $Z_t^+ = 1$.

Therefore $Z_t^+ \leq 1, \forall t$. Thus, $\prod_{t=1}^{T} Z_t^+$ decreases monotonically with T.

The following analysis shows that $Z_t^+ = 1$ is only a transient state and the algorithm will quickly pass this state and enter into an exponential decrease state.

- The applicable scenario for the weighting algorithm is when data is extremely imbalanced and the rare class has very few data points. Under this condition, it is reasonable to assume that the initial weighted class error $\epsilon_t^+ > \epsilon_t^-$. During the iteration, ϵ_t^+ will decrease, and we can stop the iteration if $\epsilon_t^+ < \epsilon_t^-$ has been achieved in less than T iterations.
- When $\epsilon^+ \geq \frac{1}{2}$, $\alpha_t = 0$, $D_{t+1}(i) = D_t(i)$ is unchanged, w_{t+1}^+ and w_{t+1}^- both increase, however w_{t+1}^+ increases faster since $\epsilon_t^+ > \epsilon_t^-$. Therefore, in the next iterations ϵ_{t+1}^+ will be decreasing. It continues to decrease until eventually ϵ_t^+ becomes less than $\frac{1}{2}$, so that the upper bound starts to decrease again.
- When $\epsilon^+ < \frac{1}{2}$, $Z_t^+ = 2\sqrt{\epsilon_t^+(1-\epsilon_t^+)}$. Let $\gamma_t = \frac{1}{2} - \epsilon_t^+$, we have $Z_t^+ = \sqrt{1-4\gamma_t^2} \leq e^{-2\gamma_t^2}$, and $\prod_t Z_t^+ \leq e^{-2\sum_t \gamma_t^2}$. In this case, the upper bound decreases exponentially.

Since the upper bound of $\frac{1}{N^+}\sum_{i\in I^+} 1_{(H(x_i)\neq y_i)}$ decreases monotonically with T, and $Z_t^+ = 1$ is a transient state and will eventually transform to $Z_t^+ < 1$, we conclude that the training error of the rare class asymptotically converges to zero as $T \to \infty$.

Figure 1 shows the upper bound of the training error and the actual training error vs the iteration number by using DiffBoost. Three algorithms, Logistic Regression (LR), Random Forest (RF) and Support Vector Machine (SVM) are tested on two datasets. The Spam dataset is from UCI machine learning repository (Dua and Graff, 2017). The simulated data are generated using a function. We generate a total of 1,832 examples with three predictors X_1, X_2 and X_3. Each predictor variable follows a Gaussian distribution, $X_1 \sim N(0, 1)$, $X_2 \sim N(1, 2)$ and $X_3 \sim N(-2, 1.5)$. The target function is $f(X_1, X_2, X_3) = X_1 + X_2 + X_3 - 5(X_1^2 + X_2^2 + X_3^2) + X_1^3 + X_2^3 + X_3^3$. The response variable $Y = f(X_1, X_2, X_3) + e$, with error term $e \sim N(0, 3)$. Binary labels are assigned as $y_i = 1$ if $y_i \geq 0$ and $y_i = -1$ otherwise.

The theory and simulation both show that DiffBoost has excellent converging property. However, it still has one issue—ittakes as many iterations to predict a new response as it takes to train the classifier. This problem is inherent to the boosting style algorithms. AdaBoost (Freund and Schapire, 1997) has the same issue. The parameters learned from each iteration as well as α_t must be saved, as the final prediction $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$ requires α_t and all the parameters used by $h_t(x)$ for $t = 1, \ldots, T$. In the next section, we propose an algorithm that takes only one shot to predict. Training may take many iterations, and can take place off-line, but once we have learned the class weights, prediction takes only one shot. This algorithm will be suitable for real-time applications.

3.2 Adaptively Computing Class Weights

Algorithm 2. AdaClassWeight.

```
   Input: $\{(x_i, y_i), i = 1, \ldots, N\}, x_i \in \chi, y_i \in \{-1, 1\}$
   Initialization: $w_t^+ = w_t^- = 1$ for $t = 1$
1  for $t \leftarrow 1$ to T
2      do
3          Train the classifier $h_t : \chi \to \{-1, 1\}$ under $w_t^+$ and $w_t^-$
4          Compute the unweighted class error rates:
               $\epsilon_t^+ = \frac{1}{N^+} \sum_{i \in I^+} \mathbb{1}_{(h_t(x_i) \neq y_i)}$
               $\epsilon_t^- = \frac{1}{N^-} \sum_{i \in I^-} \mathbb{1}_{(h_t(x_i) \neq y_i)}$
5          Update class weights:
               $w_{t+1}^+ = w_t^+ \exp(\epsilon_t^+)$
               $w_{t+1}^- = w_t^- \exp(\epsilon_t^-)$
6  Final prediction: $H(x) = h_T(x)$
```

The algorithm, called AdaClassWeight, starts with an unweighted classifier and adaptively increases the weight of the rare class until a stopping criterion is met. Unlike the DiffBoost algorithm, the AdaClassWeight algorithm only uses the parameters of the classifier learned in the final iteration to make a new prediction. DiffBoost would require all the learned parameters in the past T iterations in order to make a new prediction. AdaClassWeight also differs from DiffBoost in the way that the class error rates ϵ_t^+ and ϵ_t^- are updated. DiffBoost uses the weighted class error rates while AdaClassWeight uses the unweighted class error rates. If there is misclassification within a class, the unweighted class error rate satisfies $0 < \epsilon_t \leq 1$, so the class weight will increase in the next iteration. However, the class with a larger error rate will increase more, thus to get better classification results in the next iteration.

Remark. For the implementation of AdaClassWeight and DiffBoost, we can inject a stopping criterion during the iteration to avoid unnecessary large number of iterations. We can stop whenever the desired error rate for the rare class has been achieved, i.e., we can stop when $\epsilon_t^+ < \epsilon_t^-$, or when an absolute error threshold has reached, e.g., $\epsilon_t^+ < 0.001$.

4 INCORPORATING CLASS WEIGHTS INTO CLASSIFICATION ALGORITHMS

We show how the class weights w^+ and w^- computed from DiffBoost and AdaClassWeight are used with Logistic Regression, Random Forest, and Support Vector Machine.

4.1 Weighted Logistic Regression

Logistic Regression models are usually fit by the maximum likelihood method (Cox, 1958; Hastie et al., 2009). The log-likelihood for N observations is given by:

$$L(\beta) = \sum_{i \in I^+} \log(p_i) + \sum_{i \in I^-} \log(1 - p_i), \tag{7}$$

where L is the log-likelihood function, β is the vector of parameters, which is estimated by maximizing the log likelihood, and p_i is the probability of the i-th example being in the rare class. To assign a class weight to each class, we modify $L(\beta)$ as follows:

$$L(\beta) = w^+ \sum_{i \in I^+} \log(p_i) + w^- \sum_{i \in I^-} \log(1 - p_i), \tag{8}$$

where w^+ and w^- are class weights assigned to the positive class and the negative class, respectively.

4.2 Weighted Random Forest

Random Forest uses decision trees as building blocks (Breiman, 2001). At each split, a predictor x_j and its corresponding cut point are chosen to minimize the misclassification error, which in practice is replaced by the Gini index (Hastie et al., 2009).

$$\text{Gini index} = 1 - \sum_{i=1}^{2} \left(\frac{n_i}{\sum_{j=1}^{2} n_j} \right)^2, \tag{9}$$

where n_i is the number of training observations of class i in the node under consideration.

To incorporate class weights into Random Forest, the weighted Gini index is given as follows:

$$\text{Weighted Gini index} = 1 - \sum_{i=1}^{2} \left(\frac{w_i n_i}{\sum_{j=1}^{2} w_j n_j} \right)^2, \tag{10}$$

where w_i is the weight for class i, and $i \in \{1, 2\}$.

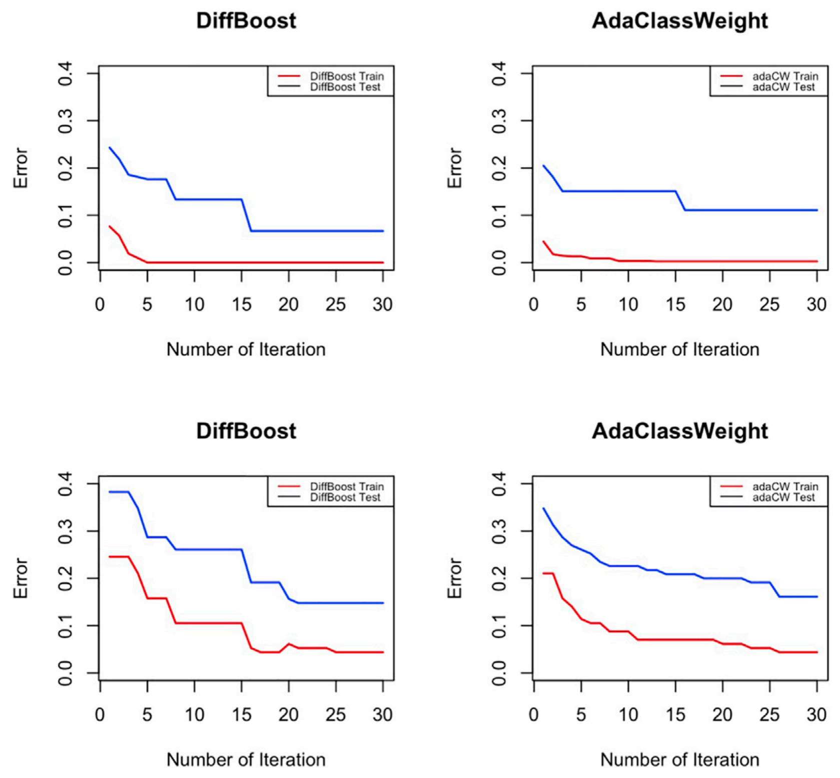

FIGURE 2 | Testing error of the rare class decreases along with training error. In the top row, IEEE 39-bus power system data is used; In the bottom row, Spam data is used.

4.3 Weighted Support Vector Machine

Support vector classifier is a maximum-margin classifier. The classification problem can be expressed as the following optimization problem (Hastie et al., 2009):

$$\min_{\beta_0,\beta}\left\{w\sum_{i=1}^{N}(1-y_i f(x_i))+\frac{1}{2}\|\beta\|^2\right\}, \tag{11}$$

where y_i is the i-th observation, and $f(x_i)$ is the classification result for the i-th data point x_i. The classifier $f()$ is a function of parameters β and β_0. Let $\phi(x_i)$ be the transformed feature vector for x_i, then $f(x_i) = \beta^T\phi(x_i) + \beta_0$. Function $\phi()$ can be the identity function for linear classification problems. The parameters β and β_0 are estimated by solving the optimization problem in **(11)**. Since both y_i and $f(x_i)$ take values in $\{-1, +1\}$, $y_i f(x_i) = -1$ only when there is a classification error. Classification error is penalized by using the same weight w on both the positive class and the negative class. To incorporate class weights into SVM, we assign different weights to different classes as follows (Veropoulos et al., 1999; Batuwita et al., 2013):

$$\min_{\beta_0,\beta}\left\{w^+\sum_{i\in I^+}(1-y_i f(x_i))+w^-\sum_{i\in I^-}(1-y_i f(x_i))+\frac{1}{2}\|\beta\|^2\right\}, \tag{12}$$

where w^+ and w^- are weights applied to the positive class and the negative class, respectively. Chang and Lin (2011) used a different formulation for the optimization problem, but the two formulations are equivalent. The equivalence of the two formulations can be found in Hastie et al. (2009).

TABLE 3 | Testing error ϵ^+_{test} for the 39-bus power system data.

Methods	Testing error		
	LR	RF	SVM
AdaBoost	0.186	—	—
$\frac{w^+}{w^-} = \frac{N^-}{N^+}$	0.097	0.2	0.139
$\frac{w^+}{w^-} = 1$	0.132	0.289	0.674
DiffBoost	0.067	0.089	0.025
AdaClassWeight	0.09	0.133	0.118

5 EXPERIMENTS

Through experiments on real datasets, we show the excellent performance of the proposed algorithms, and compare them with other algorithms: AdaBoost (Freund and Schapire, 1997), a simple weighting method that uses the ratio of positive cases and negative cases in the sample to compute weights, i.e., $\frac{w^+}{w^-} = \frac{N^-}{N^+}$, and the unweighted method. We compare the algorithms in three aspects: 1) testing error, 2) the Receiver Operating Characteristic (ROC) curve, and 3) sensitivity to noise. All classifiers are given

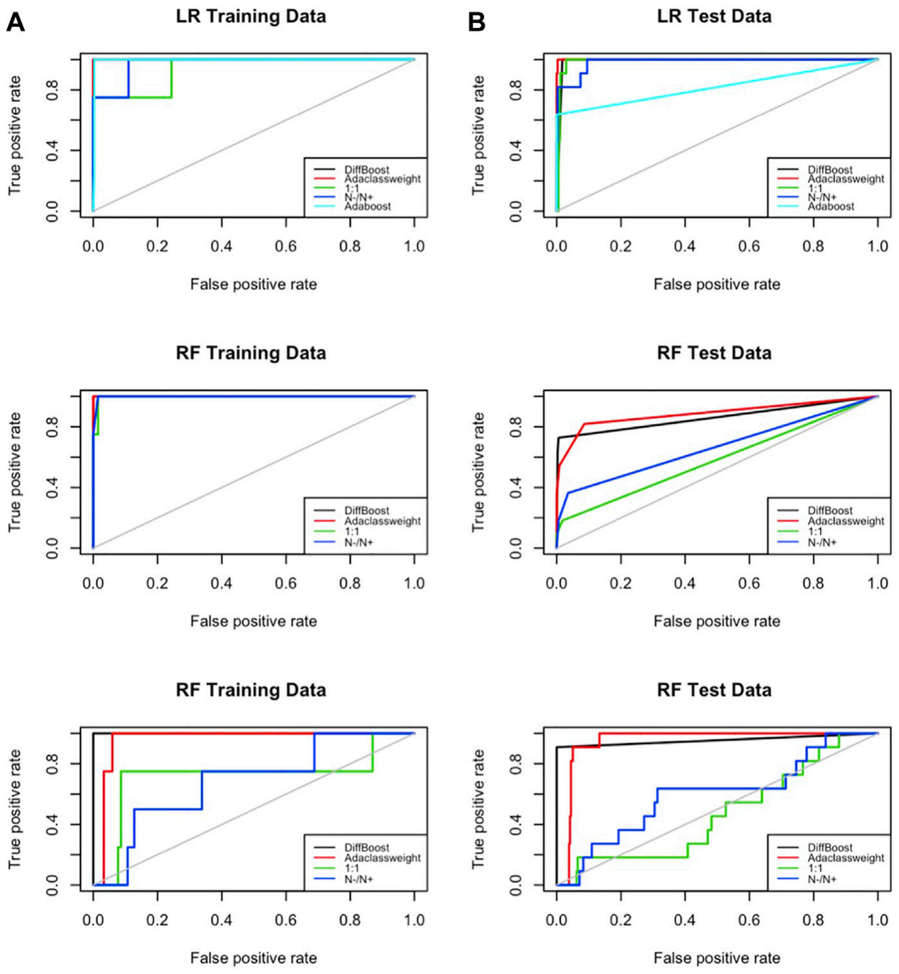

FIGURE 3 | ROC on training data and testing data. Tests are done on the IEEE 39-bus power system data. DiffBoost and AdaClass Weight both have a larger area under the curve than other algorithms.

the same input. Finally, we compare the rare class performance of our algorithms with the group of algorithms that were developed from AdaBoost.

5.1 Testing Error

Training classifiers under differentiated weights significantly reduces the training error of the rare class. Will this also translate to a reduced error on the testing data? In this experiment we test if reducing training error on the rare class leads to overfitting, which means we will get an increased testing error. **Figure 2** shows an overall trend of decreasing for both testing error and training error, so the improvement on training error on the rare class is also an improvement on the testing error. The algorithms start with an unweighted algorithm ($w^+ = w^- = 1$ at initialization), and then through iterations as the training error decreases, the testing error also decreases. Results in **Figure 2** are obtained from running Logistic Regression on the IEEE 39-bus dataset and the Spam dataset. Other classifiers show similar results.

Comparison with other methods on testing error are shown in **Table 3**. We consider three classifiers: LR, RF, and SVM. When we train a weak classifier under weights, LR can take both individual weights and class weights, but RF and SVM can only take class weights. Since AdaBoost produces one weight per example and does not give class weights, we can only obtain results for AdaBoost when using LR as the weak classifier. This experiment demonstrates that DiffBoost and AdaClassWeight both outperform existing methods in improving the rare class performance. The improvement for SVM is the most significant as the error rate is improved from 0.674 to 0.025.

5.2 Receiver Operating Characteristic

It is reasonable to expect that the weighting algorithm will improve the prediction accuracy of the rare class at the expense of the main class. In this experiment, we will find out how much tradeoff exists between the rare class and the main class. We evaluate the performance of DiffBoost and AdaClassWeight in terms of the true positive rate and the false positive rate, and we compare them with 1) the simple weighting algorithm that uses the ratio to decide weight, i.e., $\frac{w^+}{w^-} = \frac{N^-}{N^+}$, 2) the classical boosting algorithm AdaBoost, and

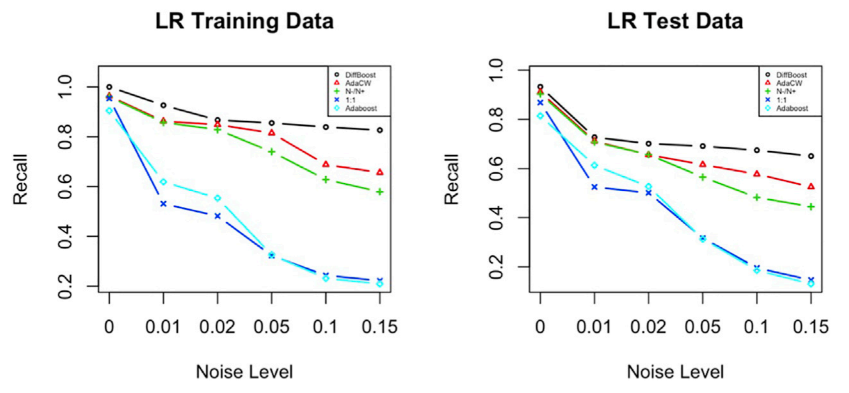

FIGURE 4 | Sensitivity to noise for both training data and testing data. Test is done on the IEEE 39-bus power system data. While all algorithms are impacted by noise, the proposed algorithms perform better.

TABLE 4 | Spam data. From left to right: the recall on test data under various cost ratios, the precision corresponding to the highest recall, training time, and test time.

	Recall with different cost ratios C_+: C_-							Precision	Training Time(s)	Test Time(s)
	1:1	1.5:1	2:1	2.5:1	5:1	7.5	10:1			
Adacost	0.633	0.669	0.698	0.878	0.698	0.92	0.775	0.1	1.41	0.004
AdaC1	0.633	0.824	0.885	0.896	0.917	0.99	0.99	0.076	3.51	0.004
AdaC2	0.633	0.9	0.92	0.96	0.98	0.99	0.99	0.076	2.73	0.005
AdaC3	0.594	0.881	0.94	0.824	0.824	0.775	0.91	0.33	2.1	0.004
DiffBoost	0.931	0.931	0.931	0.931	0.931	0.931	0.931	0.392	1.35	0.003
AdaClassWeight	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.403	0.74	0.003

3) the unweighted algorithm (labeled as 1:1 in figures). AdaBoost is not designed to address the data imbalance problem, but it does use boosting to improve prediction performance without distinction of classes. We would like to see how differentiated boosting performs in terms of ROC compared to AdaBoost and others.

The ROC curve is the plot of true positive rate versus false positive rate for different cut-off points of a parameter. If increased true positive rate is at the cost of the increased false positive rate, the curve would go along the 45° line (the gray line in **Figure 3**). Otherwise, if the false positive rate does not go up proportionally, it would stay above the 45° line. On the ROC plot, the *area under the curve* is used to compare algorithms. The one with the largest area under the curve is considered the best.

Results from using IEEE 39-bus system data (**Figure 3**) show that the areas under the curves for the two proposed algorithms are the two largest compared to all other algorithms. This means the false positive rate did not go up proportionally. This is because there are a large number of examples in the main class, and adding appropriate weight on the rare class does not affect the performance of the main class too much.

5.3 Sensitivity to Noise

We care about sensitivity to noise because added weight will also amplify noise if noise happens to be on the rare class. In this section we test if the weighting algorithms DiffBoost and AdaClassWeight are sensitive to noise. We test the algorithms on the IEEE 39-bus dataset and present the results from using Logistic Regression. Results from using the other two classifiers are similar.

Noise is added into data as class noise, i.e., we flip a certain percentage of labels in each class. We define the percentage of flipped labels as the noise level, which takes values in (0, 1, 2, 5, 10, 15%). The results show that DiffBoost and AdaClassWeight can perform better than the ratio-based weighting algorithm, better than the AdaBoost algorithm, and much better than the unweighted algorithm (see **Figure 4**). The superiority of the two proposed algorithms is more evidenced as the noise level increases. Although the performance of DiffBoost and AdaClassWeight is also impacted by noise, the impact is smaller than other algorithms since the performance drop is not very steep.

5.4 Comparison With the Boosting Algorithms

The boosting algorithms AdaC1, AdaC2, AdaC3, and Adacost mentioned in **Section 2** are developed from AdaBoost and all have similar ROC curves as AdaBoost (see **Figure 3**). Next we show another aspect of these algorithms: all methods require a preset cost C_i, $\forall i$, and the classification performance is extremely sensitive to this hyper-parameter. We use the Spam data to demonstrate. The Spam data has a total of 4,597 examples with 2,785 non-spam emails and 1,812 spam emails. We randomly select 1,506 examples from the original dataset. The ratio between non-spam examples and spam examples is 12 : 1, and 50–50% split is used for training and test within each class.

To see the effects of different cost settings on classifiers, we compare the results of Adacost, AdaC1, AdaC2, and AdaC3 based on different cost ratios C_+: C_-, and report the results on the test data (see **Table 4**). First of all, the highest recalls of the four algorithms occur at very different locations even for the same dataset. This indicates it is not a simple task to assign cost ratios. In **Table 4** the highest recall is highlighted for each algorithm and the precision under the same cost ratio is also reported. Although these recalls are very impressive, the precisions are extremely low. If an algorithm simply predicts all cases as the rare class, it can achieve recall 1.0, but the precision is close to zero. This is observed with these boosting algorithms. In comparison, AdaClassWeight has the highest precision while achieving a high recall. It is also observed that the sum of recall and precision is the highest by the two proposed algorithms.

The proposed algorithms also outperform the boosting algorithms in algorithm complexity. For the boosting algorithms, the cost ratio is a hyper-parameter. The optimal value for the hyper-parameter is obtained by a grid search. In the experiment, we performed a grid search for the cost ratio $C_+/C_- \in$ (Pazzani et al., 1994; King and Zeng, 2001) with step size 0.5. The training time would be much longer had we used a finer grid. In comparison, the proposed algorithms adaptively find the weights without a preset hyper-parameter. DiffBoost and AdaClassWeight terminate after T iterations, or before T iterations if a desired tradeoff between the positive class and the negative class have been found, so the algorithms are bounded to T iterations. The proposed algorithms have lower complexity as shown in **Table 4**.

6 CONCLUSION AND OUTLOOK

We have studied the problem of classifying rare events in imbalanced datasets, in which the rare class examples are significantly fewer than the main class examples. We focused on designing weighting algorithms to compute class weights during the training phase. DiffBoost and AdaClassWeight are general weighting algorithms and can be used in junction with any classifier. It has been tested with Logistic Regression, Random Forest, and Support Vector Machine, and has been applied to several datasets. The experimental results show that they improve the prediction accuracy of the rare class with a controlled tradeoff in the main class. The ROC curves of the proposed algorithms have larger "area under the curve" than the simple ratio-based algorithm, the original unweighted algorithm, and the AdaBoost algorithm as well as other boosting algorithms based on AdaBoost. It also has the advantage of being able to focus on the rare class, giving it a much higher accuracy in case we need to prioritize the rare class, with a controllable tradeoff. Multiclass classification for more than two classes using differential class weights is an easy extension from this work, which will be addressed in the future work.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://github.com/jhe58/Identification-of-Rare-Events-from-Imbalanced-Datasets.

AUTHOR CONTRIBUTIONS

JH and MC designed the algorithms and performed analysis. JH implemented the algorithms and conducted the experiments. MC provided overall guidance and supervision for problem setup, research methods, and reporting. All authors contributed to the article and approved the submitted version.

FUNDING

The authors gratefully acknowledge the support of NSF awards 1854077, 1936873, and 2027725. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the NSF, or the US government.

REFERENCES

Batuwita, R., and Palade, V. (2013). "Class Imbalance Learning Methods for Support Vector Machines," in *Imbalanced Learning: Foundations, Algorithms, and Applications*. Editors H. He and Y. Ma (IEEE), 83–99. doi:10.1002/9781118646106.ch5

Breiman, L. (2001). Random Forests. *Machine Learn.* 45, 5–32. doi:10.1023/a:1010933404324

Cahyana, N., Khomsah, S., and Aribowo, A. S. (2019). "Improving Imbalanced Dataset Classification Using Oversampling and Gradient Boosting," in 2019 5th International Conference on Science in Information Technology (ICSITech) (IEEE), 217–222. doi:10.1109/icsitech46713.2019.8987499

Chang, C.-C., and Lin, C.-J. (2011). Libsvm. *ACM Trans. Intell. Syst. Technol.* 2, 1–27. doi:10.1145/1961189.1961199

Cox, D. R. (1958). The Regression Analysis of Binary Sequences. *J. R. Stat. Soc. Ser. B Methodol.* 20, 215–232. doi:10.1111/j.2517-6161.1958.tb00292.x

Dua, D., and Graff, C. (2017). *UCI Machine Learning Repository*.

Ertekin, S., Huang, J., and Giles, C. (2007). "Active Learning for Class Imbalance Problem," in Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval-SIGIR'07, 823–824. doi:10.1145/1277741.1277927

Fan, W., Stolfo, S. J., Zhang, J., and Chan, P. K. (1999). Adacost: Misclassification Cost-Sensitive Boosting. *ICML* 99, 97–105.

Freund, Y., and Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* 55, 119–139. doi:10.1006/jcss.1997.1504

Freund, Y., and Schapire, R. E. (1995). "A Desicion-Theoretic Generalization of On-Line Learning and an Application to Boosting," in European Conference on Computational Learning Theory (Springer), 23–37. doi:10.1007/3-540-59119-2_166

Guo, H., and Viktor, H. L. (2004). Learning from Imbalanced Data Sets with Boosting and Data Generation. *SIGKDD Explor. Newsl.* 6, 30–39. doi:10.1145/1007730.1007736

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. 2nd Edition. Springer.

Japkowicz, N., Myers, C., and Gluck, M. (1995). "A novelty Detection Approach to Classification," in Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1-IJCAI'95, 518–523.

King, G., and Zeng, L. (2001). Logistic Regression in Rare Events Data. *Polit. Anal.* 9, 137–163. doi:10.1093/oxfordjournals.pan.a004868

Li, Q., and Mao, Y. (2014). A Review of Boosting Methods for Imbalanced Data Classification. *Pattern Anal. Applic.* 17, 679–693. doi:10.1007/s10044-014-0392-8

Liu, Y., An, A., and Huang, X. (2006). "Boosting Prediction Accuracy on Imbalanced Datasets with Svm Ensembles," in *Advances in Knowledge Discovery and Data Mining*. Editors W. K. Ng, M. Kitsuregawa, J. Li, and K. Chang (Berlin, Heidelberg: Springer Berlin Heidelberg), 107–118. doi:10.1007/11731139_15

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., and Brunk, C. (1994). "Reducing Misclassification Costs," in Proceedings of the Eleventh International Conference on International Conference on Machine Learning-ICML'94, 217–225. doi:10.1016/b978-1-55860-335-6.50034-9

Sun, Y., Kamel, M. S., Wong, A. K. C., and Wang, Y. (2007). Cost-sensitive Boosting for Classification of Imbalanced Data. *Pattern Recognition* 40, 3358–3378. doi:10.1016/j.patcog.2007.04.009

Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., and Asadpour, M. (2020). Boosting Methods for Multi-Class Imbalanced Data Classification: an Experimental Review. *J. Big Data* 7, 1–47. doi:10.1186/s40537-020-00349-y

Ting, K. M. (2000). "A Comparative Study of Cost-Sensitive Boosting Algorithms," in Proceedings of the 17th International Conference on Machine Learning (Citeseer), 983–990.

Veropoulos, K., Campbell, C., and Cristianini, N. (1999). "Controlling the Sensitivity of Support Vector Machines," in Proceedings of the International Joint Conference on AI, 55–60.

Weiss, G. M. (2004). Mining with Rarity. *SIGKDD Explor. Newsl.* 6, 7–19. doi:10.1145/1007730.1007734

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.