REGULAR PAPER

AdaCC: cumulative cost-sensitive boosting for imbalanced classification

Vasileios Iosifidis[1] · Symeon Papadopoulos[2] · Bodo Rosenhahn[1] · Eirini Ntoutsi[3]

Abstract

Class imbalance poses a major challenge for machine learning as most supervised learning models might exhibit bias towards the majority class and under-perform in the minority class. Cost-sensitive learning tackles this problem by treating the classes differently, formulated typically via a *user-defined fixed* misclassification cost matrix provided as input to the learner. Such parameter tuning is a challenging task that requires domain knowledge and moreover, wrong adjustments might lead to overall predictive performance deterioration. In this work, we propose a novel cost-sensitive boosting approach for imbalanced data that *dynamically* adjusts the misclassification costs over the boosting rounds in response to model's performance instead of using a fixed misclassification cost matrix. Our method, called AdaCC, is parameter-free as it relies on the *cumulative* behavior of the boosting model in order to adjust the misclassification costs for the next boosting round and comes with theoretical guarantees regarding the training error. Experiments on 27 real-world datasets from different domains with high class imbalance demonstrate the superiority of our method over 12 state-of-the-art cost-sensitive boosting approaches exhibiting consistent improvements in different measures, for instance, in the range of [0.3–28.56%] for AUC, [3.4–21.4%] for balanced accuracy, [4.8–45%] for gmean and [7.4–85.5%] for recall.

Keywords Class imbalance · Cost-sensitive learning · Boosting · Cumulative costs · Dynamic costs

✉ Vasileios Iosifidis
iosifidis@L3S.de

Symeon Papadopoulos
papadop@iti.gr

Bodo Rosenhahn
rosenhahn@tnt.uni-hannover.de

Eirini Ntoutsi
eirini.ntoutsi@fu-berlin.de

[1] Leibniz University of Hannover, Hannover, Germany

[2] Centre of Research and Technology Hellas, Heraklion, Greece

[3] Free University of Berlin, Berlin, Germany

1 Introduction

When supervised learning models are trained on data generated from skewed class distributions, i.e., suffer from the *class imbalance* problem, their performance on the minority class can degrade significantly, even though they may have outstanding performance in terms of overall error rate or accuracy[1]. In extreme cases, the model may ignore the minority class altogether and predict always the majority class. Class imbalance is inherent in many real-world applications, e.g., medical diagnosis [28, 39], fraud detection [2, 32, 36, 41] or sentiment classification [19, 29] and could even lead to discrimination and unfairness [15–18, 20, 21, 40].

Over the years, a large body of work has been proposed for tackling the class imbalance problem. Following [46], these works can be categorized into: (i) *data-level approaches*, (ii) *model-based approaches*, and (iii) *cost-sensitive approaches*. Each category has its own limitations (and strengths). For instance, data-level approaches may discard useful information to restore balance across the different class distributions. Model-based approaches are typically designed and implemented for specific models and are therefore applicable only in limited settings. Finally, cost-sensitive methods require as input a misclassification cost matrix thus inducing additional parameters.

Here, we focus on cost-sensitive classification methods with boosting. We have chosen cost-sensitive boosting for three main reasons: (i) boosting is able to minimize the training error and at the same time, to avoid overfitting [43], (ii) boosting is a popular learning method employed in many classification systems [33], and (iii) by re-weighting the data distribution, boosting preserves more information comparing to sampling methods [46], the prevalent type of data-level methods. However, most cost-sensitive boosting methods require a fixed misclassification cost matrix provided by the user [11, 35, 46, 47]. To define such a matrix, often grid search is performed to find the best costs for the dataset at hand, a tedious and costly process. In many cases, as we also show in our experiments, grid search does not lead to optimal selection of misclassification costs. Additionally, having fixed costs during model training may lead to suboptimal learning outcomes.

In this work, we propose a new parameter-free cost-sensitive boosting approach for classification problems with high class imbalance. The proposed method, named *AdaCC*, standing for *Cumulative Cost-Sensitive Boosting*, alleviates the need for setting a fixed misclassification cost-matrix as input parameter, by leveraging the cumulative costs of the model up to the current boosting round. As we show in Sect. 4, the proposed method has proven upper bounds for the training error. We propose two variants of the method, *AdaCC1* and *AdaCC2* that differ in terms of the employed data re-weighting scheme.

We carry out a comprehensive experimental study on 27 real-world datasets and compare our method with 12 state-of-the-art cost-sensitive boosting methods as well as 5 non cost-sensitive class-imbalance methods. Our results demonstrate the superior performance of AdaCC over the state of the art in terms of AUC, balanced accuracy, geometric mean, and recall. Notably, the performance improvements are more pronounced on the minority class. This makes our method suitable for tasks where high false negative rates are critical, e.g., medical diagnosis, fraud detection, fairness-aware machine learning, etc.

Figure 1 illustrates a binary imbalanced toy dataset where the blue points (#5) belong to the minority class and the red points (#20) to the majority class. We compare the learning behaviour of the proposed method with the one by AdaBoost by observing the decision

---

[1] Note: In the binary classification case, the class with significantly more instances is the so-called *majority* class, while the other is the *minority* class.
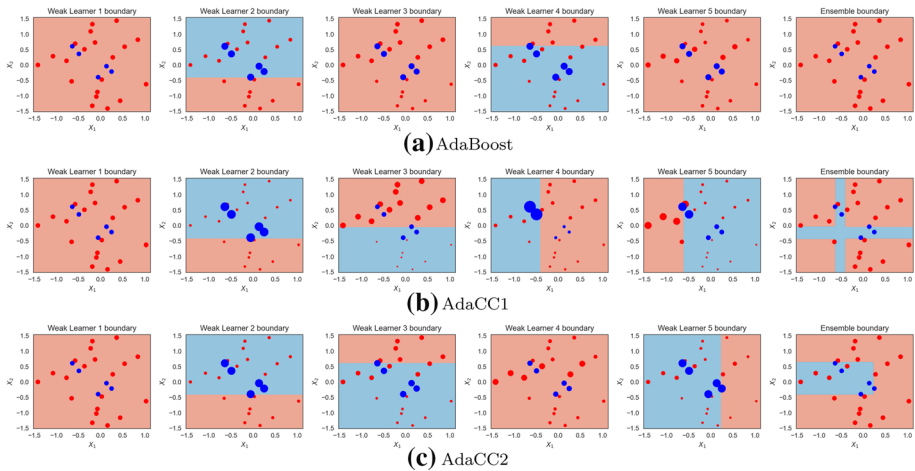
Fig. 1 Decision boundaries of AdaBoost and the two variants of the proposed AdaCC on the same imbalanced toy dataset of 5 blue and 20 red instances. The dot size is proportional to the weight allocated by each learner to the particular instance (with the exception of the last column that depicts the final ensemble), making clear that AdaCC assigns higher weight to minority class instances compared to the ones of the majority class

boundaries of the weak learners on the toy dataset as well as the ensemble boundary at the end of the training process (rightmost figure). Due to the low dimensionality of the dataset and for illustration purposes, we use a small number of $T = 5$ weak learners. Figure 1 demonstrates how weighted data distribution is affected by each weak learner as well as the decision boundary of the final ensembles. The first 5 columns correspond to the 5 weak learners while the size of the dots corresponds to the weight that each instance receives per round. The last column illustrates the decision boundary of the ensemble model. We note that the final AdaCC model manages to fit better the data distribution compared to AdaBoost, allocating a "proper" part of the feature space to the minority class.

The rest of the paper is organized as follows: Related work is summarized in Sect. 2. Basic concepts are described in Sect. 3. Our approach is introduced in Sect. 4. Evaluation setup and experimental results are presented in Sects. 5 and 6, respectively. Finally, Sect. 7 concludes our work and identifies interesting directions for future research.

2 Related work

Methods for dealing with class imbalance can be organised in three broad categories [46]: (i) data-level, (ii) model-based and (iii) cost-sensitive methods.

Data-level methods operate at the dataset level, i.e., they modify the data distribution before model training, making these methods universally applicable. In [22], the authors investigate the problem of class imbalance and the impact of re-sampling methods under the inter-dependencies of class distribution skewness, data complexity, data volume and employed models. In [30] the authors propose a combination of under- and over-sampling to equalize class distributions and measure model's performance using lift analysis. The impact of over-sampling and under-sampling under the cost curves performance metrics has been explored in [8]. The authors conclude that under-sampling is significantly more effective than over-sampling for C4.5 classifiers. In [3], the authors propose *SMOTE*, a method that

augments the minority class by interpolating new instances in local neighborhoods. In [19], the authors propose text augmentation techniques, such as distortion and semantic similarity, to increase the representation of the minority class.

Although re-sampling approaches are simple and easy to use, they come with disadvantages. For example, over-sampling may fail to "boost" existing rare cases, and adds no additional information to the dataset [8, 46]. Under-sampling on the other hand, can deteriorate the performance by removing important information from the majority class [3]. Finally, augmentation methods can amplify and propagate noise [19], leading to overall performance deterioration.

Model-based methods tackle class imbalance during training either by employing a mechanism which aims to identify rare patterns or by optimizing for a balanced-performance aware metric. *SMOTEBoost* [4] combines SMOTE [3] and AdaBoost [42] to deal with class imbalance by augmenting the minority class in each boosting round. A similar line of work is *RUSBoost* [44], which combines AdaBoost and random under-sampling of the majority class on each boosting round. *DataBoost-IM* [12] locates the hard-to-learn instances from both positive and negative classes during the training phase of AdaBoost and based on these instances, it generates synthetic data for augmentation at the end of each boosting round. Class imbalance-sensitive pruning of decision trees has been presented in [53]. The work in [50] uses a kernel alignment to optimize the decision boundary of an SVM. In [24], a class posterior re-balancing framework has been proposed to reduce imbalance while retaining classification certainty. Over the recent years, hybrid methods have also been proposed. In [49], they employ multi-set feature learning to learn discriminant features from the constructed multi-set and combine the sets with a generative adversarial network technique such that each subset has similar distribution with the original dataset. In [51], authors propose a combination of different techniques such as under/over-sampling, data transformations, misclassification costs and ensemble learning to deal with class imbalance.

The main disadvantage of model-based methods is that the inductive bias of the selected model can raise issues given an imbalanced dataset, e.g., decision tree's data fragmentation problem [14]. Additionally, they typically rely on assumptions regarding the underlying data properties or are tailored to specific classification algorithms, which makes hard their application to new domains and datasets.

Cost-sensitive methods do not optimize for overall accuracy. Instead, they try to minimize the overall misclassification costs. This class of algorithms is divided into three sub-categories [46]: (i) weighting the data space, (ii) making a specific classifier cost-sensitive, and (iii) using the Bayes risk theory to assign each instance to the class with the lowest risk. The first sub-category aims to alter data distribution by employing a misclassification cost matrix such that errors in minority class instances induce a higher loss. The very first method in this line of work is *AdaCost* [11]. Over the years many variations of AdaCost have been introduced such as: *CSB1* [47], *CSB2* [47], *RareBoost* [23], *AdaC1* [46], *AdaC2* [46], *AdaC3* [46] and *CGAda* [25–27], which differ in the following main aspects: training data weight assignments, weight update rules, and decision rules. Except RareBoost, all the aforementioned methods in this category require user parameters for the misclassification costs. An overview of these methods can be seen in Table 3. The second sub-category of cost-sensitive methods aims to make a specific classifier cost-sensitive. In [34], authors propose *AdaMEC*, a boosting classifier that uses the misclassification costs only to set thresholds to the decision boundary of AdaBoost, in contrast to the previous methods which use the misclassification costs to change the data distribution in each boosting round. CGAda and AdaMEC have also been extended in [35], namely *CGAda-Cal.* and *AdaMEC-Cal.*, by calibrating the models' scores using the Platt scaling technique [37]. In [38], a cost-sensitive k-NN classifier is introduced to

Table 1 Notations

D	Set of instances
x	Sample
A	Feature space
y	Class attribute
C	Misclassification cost vector
T	Number of weak learners
h_t	Weak learner, at round t
α_t	Weight of the weak learner, at round t
Z_t	Normalization factor, at round t
$H(\cdot)$	Ensemble
$sign(\cdot)$	Sign function
$exp(\cdot)$	Exponential function
$\mathbb{I}\{\cdot\}$	Indicator function

tackle class imbalance by using a modified distance function which takes into consideration the misclassification cost matrix. In [31], a misclassification cost matrix is used to define a cost-sensitive splitting criterion in decision trees, while in [1] the authors take into account the misclassification costs to determine the pruning criterion of a decision tree. The third sub-category uses the Bayes risk theory to assign each instance to a class with the lowest risk. Few works have been proposed in this direction, e.g., in [7] the authors swap the class labels of the leaves to minimize the misclassification cost.

For evaluation purposes, we select all the aforementioned cost-sensitive boosting methods since they are related to our contribution. In contrast to our proposed approach, however, the aforementioned cost-sensitive boosting methods assume that the misclassification costs for each class are known in advance (except RareBoost). For many applications/datasets these costs might not be available, and a costly grid search has to be performed to estimate them; however, in many cases, even grid search does not lead to optimal misclassification costs. Instead, the two variants of our approach are parameter-free and leverage the cumulative behavior of AdaBoost to dynamically adjust the misclassification costs per boosting round. Hence, our methods are applicable to any imbalanced dataset without any prior domain knowledge.

3 Preliminaries

For the sake of clarity, in Table 1 we briefly describe the employed notations. We assume a set of instances $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ consisting of n independent and identically distributed samples drawn from the joint distribution $P(A, y)$, where A denotes the feature space and y is the class attribute. For simplicity, we assume the class is binary with $y \in \{+1, -1\}$. We denote by D_+ (D_-) the set of instances belonging to the positive (negative, respectively) class. We also assume that the positive class is the minority, i.e., $|D_+| << |D_-|$. It holds that $|D_+| + |D_-| = n$.

Standard classification models treat instances of different classes equally and the performance of the induced classifier (see confusion matrix in Table 2) is measured in terms of the overall error rate (ER) as: $ER = (FP + FN)/(TP + TN + FP + FN)$. However, when the class distribution is skewed, the overall error rate is not a good indicator of model's

Table 2 Confusion matrix

	Predicted positives	Predicted negatives
Positives ($|D_+|$)	True positives (TP)	False negatives (FN)
Negatives ($|D_-|$)	False positives (FP)	True negatives (TN)

performance in all classes, but rather of the performance on the majority class. In such a case, more appropriate performance metrics should be employed (see an overview in Table 5).

Cost-sensitive models tackle the class imbalance problem by emphasizing more on the minority class through appropriate costs [14, 35, 47]. Each sample $x \in D$ is mapped to a typically *fixed* misclassification cost vector $\mathbf{C} =< C_+, C_- >$, where each sample in D_+ is associated with a fixed cost value C_+ from the misclassification cost vector $\mathbf{C}$ and each sample in D_- with a fixed cost value C_- from $\mathbf{C}$, where $C_+ > C_-$ and $C_+, C_- \in [0, \infty)$. The costs denote the misclassification costs for each class and are employed by the cost-sensitive learner during the training phase to "force" the learner to also learn minority instances. The costs, however, need to be manually set by the user, thus requiring prior domain knowledge, or to be selected via grid search [14, 46].

Boosting and AdaBoost: Boosting is an ensemble learning technique which trains a sequence of T weak learners, in order to create a strong learner. The sequential generation promotes the dependency between the weak learners and each learner learns from the mistakes of the previous learner.

AdaBoost [42], one of the most popular boosting algorithms (see Algorithm 1), adjusts in each iteration $t : 1 - T$ (the so-called *boosting round t*) the data distribution D^t based on the mistakes of the current learner h_t in order to focus in the next round $t + 1$ on the misclassified instances. In particular, the weights of the instances for the next round are updated as follows:

$$D^{t+1}(i) = \frac{D^t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{1}$$

The parameter α_t denotes the weight of the weak learner h_t in the final classification decision and is based on the error rate of the weak learner h_t:

$$\alpha_t = \frac{1}{2} \log\left(\frac{\sum\limits_{i, y_i = h_t(x_i)} D^t(i)}{\sum\limits_{i, y_i \neq h_t(x_i)} D^t(i)}\right) \tag{2}$$

The parameter Z_t is a normalization factor which is used at the end of each boosting round to make D^{t+1} a probability distribution:

$$Z_t = \sum_{i=1}^{n} D^t(i) exp\left(-\alpha_t y_i h_t(x_i)\right) \tag{3}$$

The final model is a weighted combination of the weak learners:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{4}$$

Cost-sensitive boosting approaches extend AdaBoost for class imbalance by changing the following components: (i) *weight initialization* (recall that in Adaboost all instances receive the same weight during initialization—line 1 of Algorithm 1), (ii) *distribution reweighting*

Algorithm 1 AdaBoost Algorithm

Input: $D = (x_i, y_i)_1^n$**, where** $y \in \{+1, -1\}$**,** T
Output: $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

1: Initialisation : $D^1(i) = 1/n$
2: **for** $t = 1$ to T **do**
3:     Train weak learner $h_t \rightarrow y$ using D^t
4:     Update α_t based on Eq. (2)
5:     For $i = 1, ..., n$:
6:         Update $D^{t+1}(i)$ based on Eq. (1)
           //where Z_t is the normalization factor according to Eq. 3.
7: **end for**
8: return $H(x)$ // Eq. (4)

(for AdaBoost the update is according to Eqs. (1) and (2)), and (iii) *voting schema* (for Adaboost voting is according to Eq. (4)). A detailed overview of the cost-sensitive methods and how they implement the aforementioned (i)–(iii) aspects is presented in Table 3. CGAda [25–27] employs the misclassification cost matrix only for initializing the weight distribution at the first boosting round and proceeds as standard AdaBoost thereafter. AdaCost (β_2) [11], AdaC1-C3 [46] and CSB1/2 [47] incorporate the misclassification cost matrix to change the data distribution in each boosting round. AdaMEC [34] and RareBoost [23] differ from the other cost-sensitive methods: In particular, AdaMEC does not use costs to change the data distribution but it rather shifts the decision boundary of AdaBoost to minimize the total expected loss. RareBoost does not rely on misclassification costs, rather it employs instead of a single parameter α (see Eq. (2)), two different parameters, α^+ and α^- for positive and negative predictions, respectively to update the weight distribution as well as the voting schema. RareBoost requires that $TP > FP$; however, if this assumption does not hold the algorithm's performance deteriorates [46]. CGAda-Cal. [35] and AdaMEC-Cal. [35] are not shown in Table 3 since calibration, through Platt scaling, is applied to the trained CGAda and AdaMEC models, respectively.

4 AdaCC: cumulative cost-sensitive boosting

Instead of assuming a fixed misclassification cost matrix, AdaCC dynamically adjusts the misclassification costs in each boosting round based on the performance of the model up to that round, i.e., the performance of the partial ensemble (Sect. 4.1). This way, in each boosting round AdaCC boosts the class with the highest misclassification rate. These costs are then used to update the data distribution for the next round. There are two ways to incorporate the costs in the update formula (for AdaBoost the update formula is shown in Eq. (1)): inside or outside the exponent resulting in two variations AdaCC1 (Sect. 4.2) and AdaCC2 (Sect. 4.3), respectively.

The toy example in Fig. 1 demonstrates how our approach "pays extra attention" to the minority class errors: in particular, we observe that AdaBoost, AdaCC1 and AdaCC2 misclassify the minority class (blue points) during the first boosting round $t = 1$; however, our methods assign higher weights to the minority examples on the next boosting rounds in contrast to AdaBoost, which lead to substantially different decision boundaries on the upcoming boosting rounds and also the final ensemble.

Table 3 An overview of cost-sensitive boosting methods w.r.t. cost assignment, initialization, distribution update and ensemble decision rule

Method	Costs	Initial D^1	D^{t+1} update	α_{t+1} update	Decision H(x)
AdaBoost [42]	None	$1/n$	$D^t(i)\exp(-\alpha_t y_i h_t)$	$\frac{1}{2}\log\dfrac{\sum\limits_{i,h_t(x_i)=y_i} D^t(i)}{\sum\limits_{i,h_t(x_i)\neq y_i} D^t(i)}$	$\text{Sign}(\sum_t \alpha_t h_t(x))$
CGAda [25–27]	Fixed user param	$\dfrac{C_i}{\sum_i^n C_i}$	"	"	"
AdaMEC [34]	"	$1/n$	"	"	$\text{Sign}\left(\sum\limits_{y\in\{+1,-1\}} c(y)\sum\limits_{t:h_t=y}\alpha_t h_t(x)\right)$
RareBoost [23]	None	"	$D^t(i)\exp(-\alpha_t^{+,-}y_i h_t)$*	$\alpha_t^+=\frac{1}{2}\ln\frac{TP_t}{FP_t},\ \alpha_t^-=\frac{1}{2}\ln\frac{TN_t}{FN_t}$	$\text{sign}\left(\sum\limits_{t:h_t(x)\geq 0}\alpha_t^+ h_t(x)+\sum\limits_{t:h_t(x)<0}\alpha_t^- h_t(x)\right)$
CSB1 [47]	Fixed user param	$\dfrac{C_i}{\sum_i^n C_i}$	$D^t(i)C_i\exp(-y_i h_t)$	$\frac{1}{2}\log\dfrac{\sum\limits_{i,h_t(x_i)=y_i} D^t(i)}{\sum\limits_{i,h_t(x_i)\neq y_i} D^t(i)}$	$\text{Sign}(\sum_t \alpha_t h_t(x))$
CSB2 [47]	"	"	$D^t(i)C_i\exp(-\alpha_t y_i h_t)$	"	"
AdaCost (β_2) [11]	"	"	$D^t(i)\exp(-\alpha_t y_i h_t\beta_i)$	$\frac{1}{2}\log\dfrac{1+\sum_i D^t(i)\exp(-\alpha_t y_i h_t\beta_i)}{1-\sum_i D^t(i)\exp(-\alpha_t y_i h_t\beta_i)}$	"
AdaC1 [46]	"	"	$D^t(i)\exp(-C_i\alpha_t y_i h_t)$	$\frac{1}{2}\log\dfrac{1+\sum\limits_{i,y_i=h_t(x_i)}C_i D^t(i)-\sum\limits_{i,y_i\neq h_t(x_i)}C_i D^t(i)}{1-\sum\limits_{i,y_i=h_t(x_i)}C_i D^t(i)+\sum\limits_{i,y_i\neq h_t(x_i)}C_i D^t(i)}$	"
AdaC2 [46]	"	"	$D^t(i)C_i\exp(-\alpha_t y_i h_t)$	$\frac{1}{2}\log\dfrac{\sum\limits_{i,h_t(x_i)=y_i}C_i D^t(i)}{\sum\limits_{i,h_t(x_i)\neq y_i}C_i D^t(i)}$	"
AdaC3 [46]	"	"	$D^t(i)C_i\exp(-C_i\alpha_t y_i h_t)$	$\frac{1}{2}\log\dfrac{\sum_i C_i D^t(i)+\sum\limits_{i,h_t(x_i)=y_i}C_i^2 D^t(i)-\sum\limits_{i,h_t(x_i)\neq y_i}C_i^2 D^t(i)}{\sum_i C_i D^t(i)-\sum\limits_{i,h_t(x_i)=y_i}C_i^2 D^t(i)+\sum\limits_{i,h_t(x_i)\neq y_i}C_i^2 D^t(i)}$	"
AdaCC1	Eq. (6)	$1/n$	$D^t(i)\exp(-C_i^t\alpha_t y_i h_t)$	$\frac{1}{2}\log\dfrac{1+\sum\limits_{i,y_i=h_t(x_i)}C_i^t D^t(i)-\sum\limits_{i,y_i\neq h_t(x_i)}C_i^t D^t(i)}{1-\sum\limits_{i,y_i=h_t(x_i)}C_i^t D^t(i)+\sum\limits_{i,y_i\neq h_t(x_i)}C_i^t D^t(i)}$	"

Table 3 continued

Method	Costs	Initial D^1	D^{t+1} update	α_{t+1} update	Decision $H(x)$
AdaCC2	Eq. (6)	"	$D^t(i)C_i^t \exp(-\alpha_t y_i h_t)$	$\frac{1}{2}\log\dfrac{\sum\limits_{i,h_t(x_i)=y_i} C_i^t D^t(i)}{\sum\limits_{i,h_t(x_i)\neq y_i} C_i^t D^t(i)}$	"

Our methods are shown in bold. RareBoost (*) uses α^+ (α^-) for reweighting the samples which are positively (negatively) classified

4.1 Cumulative misclassification costs

Let $t \in [1, T]$ be the current boosting round, where T is a user defined parameter indicating the number of boosting rounds. Let $H_{1:t}(x) = sign(\sum_{j=1}^{t} \alpha_j h_j(x))$ be the partial ensemble up to round t. We monitor the cumulative error of the partial ensemble and in particular, the cumulative false positive rate (FPR) and the cumulative false negative rate (FNR) defined as follows:

$$\text{FNR}_{1:t} = \frac{\sum\limits_{i, x_i \in D_+} \mathbb{I}\left\{ sign\left(\sum\limits_{j=1}^{t} \alpha_j h_j(x_i) \right) \neq y_i \right\}}{|D_+|}$$

$$\text{FPR}_{1:t} = \frac{\sum\limits_{i, x_i \in D_-} \mathbb{I}\left\{ sign\left(\sum\limits_{j=1}^{t} \alpha_j h_j(x_i) \right) \neq y_i \right\}}{|D_-|} \tag{5}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function that returns 1 if the condition within is true and 0, otherwise. The term $\text{FNR}_{1:t}$ corresponds to the error of the partial ensemble in the positive class (D_+); likewise, $\text{FPR}_{1:t}$ refers to the error in the negative class (D_-).

Based on the cumulative error rates, we define the *cumulative misclassifications costs* below in order to "bias" the weighting process for the next round towards the class with the highest misclassification rate (on the current boosting round):

$$C^t(x_i) = \begin{cases} 1 + \text{FNR}_{1:t}, & \text{if } h_t(x_i) \neq y_i, y_i = +, \text{FNR}_{1:t} > \text{FPR}_{1:t} \\ 1 + \text{FPR}_{1:t}, & \text{if } h_t(x_i) \neq y_i, y_i = -, \text{FNR}_{1:t} < \text{FPR}_{1:t} \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

where h_t is the weak learner at round t. In particular, for any misclassified instance x_i, we increase its weight using the cumulative FPR or FNR values based on its class-membership.

The costs are therefore dynamically adjusted based on the partial ensemble's cumulative behavior and the predictions of the current weak learner. In contrast to other methods that assume *fixed* misclassification costs through the boosting rounds, our method is not only *parameter-free* but it also dynamically detects which class might require extra weighting at each round. We should highlight that the cumulative misclassification costs aim to boost the class with the highest misclassification rate and not individual examples. Nonetheless, the cumulative misclassification costs affect the weights of the instances since they are used to update the data distribution. In what follows, and when it is clear from the context, we simplify the notation of $C^t(x_i)$ as C_i^t.

The two variants AdaCC1 (Sect. 4.2) and AdaCC2 (Sect. 4.3) are presented next.

4.2 AdaCC1

The first proposed algorithm, AdaCC1, modifies the weight update formula of AdaBoost (Eq. (1)) using the cumulative costs C_i^t (Eq. (6)) as follows:

$$D^{t+1}(i) = \frac{D^t(i) \exp\left(-C_i^t \alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{7}$$

The normalization factor Z_t (for Adaboost shown in Eq. (3)), in round t, is also updated to take the extra weighting factor into account:

$$Z_t = \sum_{i=1}^{n} D^t(i) \exp\left(-C_i^t \alpha_t y_i h_t(x_i)\right) \tag{8}$$

Error analysis: By unravelling Eq. (7), the following holds:

$$D^{t+1}(i) = D^1(i) \times \frac{\exp\left(-C_i^1 \alpha_1 y_i h_1(x_i)\right)}{Z_1} \times \cdots \times \frac{\exp\left(-C_i^t \alpha_t y_i h_t(x_i)\right)}{Z_t} =$$

$$= \frac{D^1(i) \exp\left(-\sum_{j=1}^{t} C_i^j \alpha_j y_i h_j(x_i)\right)}{\prod_{j=1}^{t} Z_j} \tag{9}$$

The upper bound of the training error of the final ensemble $H(x)$ can be expressed as:

$$Pr_{i \sim D^1}[H(x_i) \neq y_i] \leq \sum_{i=1}^{n} D^1(i) \exp\left(-\sum_{t=1}^{T} C_i^t \alpha_t y_i h_t(x_i)\right) = \prod_{t=1}^{T} Z_t \tag{10}$$

Therefore, the objective in each boosting round is to find the α_t that minimizes Z_t. Since Z_t is the weight summation of correctly and non-correctly classified instances at round t, following the same argumentation as in [43, 46], Eq. (8) can be expressed as:

$$\sum_{i=1}^{n} D^t(i) \exp\left(-C_i^t \alpha_t y_i h_t(x_i)\right) \leq \sum_{i=1}^{n} D^t(i) \left(\frac{1 - C_i^t y_i h_t(x_i)}{2} \exp(\alpha_t)\right.$$

$$\left. + \frac{1 + C_i^t y_i h_t(x_i)}{2} \exp(-\alpha_t)\right) \tag{11}$$

By differentiating Eq. (11) w.r.t. α_t and setting it to zero, we can estimate α_t as follows:

$$\frac{\partial}{\partial \alpha_t}\left(\sum_{i=1}^{n} D^t(i)\left(\frac{1 - C_i^t y_i h_t(x_i)}{2} \exp(\alpha_t)\right)\right.$$

$$\left. + \sum_{i=1}^{n} D^t(i)\left(\frac{1 + C_i^t y_i h_t(x_i)}{2} \exp(-\alpha_t)\right)\right) = 0 \Rightarrow$$

$$e_t^{\alpha} \sum_{i=1}^{N} D^t(i)\left(\frac{1 - C_i^t y_i h_t(x_i)}{2}\right) = e^{-\alpha_t} \sum_{i=1}^{N} D^t(i)\left(\frac{1 + C_i^t y_i h_t(x_i)}{2}\right) \Rightarrow$$

$$\alpha_t = \frac{1}{2} \log\left(\frac{\sum_{i=1}^{n} D^t(i)(1 + C_i^t y_i h_t(x_i))}{\sum_{i=1}^{n} D^t(i)(1 - C_i^t y_i h_t(x_i))}\right)$$

$$= \frac{1}{2} \log\left(\frac{1 + \sum_{i, y_i = h_t(x_i)}^{n} C_i^t D^t(i) - \sum_{i, y_i \neq h_t(x_i)}^{n} C_i^t D^t(i)}{1 - \sum_{i, y_i = h_t(x_i)}^{n} C_i^t D^t(i) + \sum_{i, y_i \neq h_t(x_i)}^{n} C_i^t D^t(i)}\right) \tag{12}$$

To ensure that α_t is non-negative, the following condition should hold, otherwise the iteration process terminates:

$$\sum_{i,\,y_i=h(x_i)} C_i^t D^t(i) > \sum_{i,\,y_i \neq h(x_i)} C_i^t D^t(i) \tag{13}$$

Time complexity: We derive the time complexity of our approach building upon the complexity of AdaBoost (c.f., Algorithm 1). AdaBoost complexity is $O(T \cdot (f + n))$, where T is the number of boosting rounds, $O(f)$ is the complexity of a weak learner (for decision stumps it is $O(n \cdot m)$ for training and $O(n)$ for testing, where m is the number of features and n the number of instances [45]), and $O(n)$ is the complexity for the weight update of the instances. Our only addition to the algorithm (computationally) is the calculation of the cumulative errors (Eq. (5)). This computation can be reduced to $O(n)$ by maintaining a vector **o** of size n over the boosting rounds which averages the decision outcomes of the weak learners in each boosting round. Note that the vector **o** is updated on each round based on the current weak learner's predictions (on the training). By doing this, we avoid spending $O(t \cdot f)$ on each boosting round t i.e., we avoid the prediction time of the partial ensemble (on the training set) on each boosting round. Therefore, the complexity of AdaCC1 is: $O(T \cdot (f + 2n)) \Rightarrow O(T \cdot (f + n))$, since 2 is a constant.

4.3 AdaCC2

The second proposed algorithm, AdaCC2, modifies the weight update formula of AdaBoost (Eq. (1)) using the cumulative costs (Eq. (6)) as follows:

$$D^{t+1}(i) = \frac{D^t(i) C_i^t \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{14}$$

Similarly to AdaCC1, the normalization factor Z_t is also updated to ensure D^{t+1} is still a probability distribution:

$$Z_t = \sum_{i=1}^{n} D^t(i) C_i^t \exp\left(-\alpha_t y_i h_t(x_i)\right) \tag{15}$$

Error analysis: Following the same logic as in Eq. (7) for AdaCC1, by unravelling Eq. (14), we obtain the following:

$$D^{t+1}(i) = \frac{D^1(i) \prod_{j=1}^{t} C_i^j \exp\left(-\alpha_j y_i h_j(x_i)\right)}{\prod_{j=1}^{t} Z_j} \tag{16}$$

Similarly to AdaCC1, the upper bound of the training error of the final ensemble $H(x)$ is given by:

$$Pr_{i \sim D^1}[H(x_i) \neq y_i] \leq \sum_{i=1}^{n} D^1(i) \prod_{t=1}^{T} C_i^t \exp\left(-\alpha_t y_i h_t(x_i)\right) = \prod_{t=1}^{T} Z_t \tag{17}$$

Following a similar to AdaCC1 rationale (Eqs. (11) and (12)), the α_t that minimizes Z_t is given by:

$$\alpha_t = \frac{1}{2} \log \left(\frac{\sum\limits_{i,\, y_i = h_t(x_i)}^{n} C_i^t D^t(i)}{\sum\limits_{i,\, y_i \neq h_t(x_i)}^{n} C_i^t D^t(i)} \right) \tag{18}$$

To ensure that α_t is non-negative, the same condition as in Eq. (13) for AdaCC1 should hold, otherwise the iteration process terminates.

Time complexity: AdaCC2 has the same time complexity as AdaCC1 since their only difference pertains to the weight estimation.

5 Evaluation setup

We compare our proposed AdaCC1 and AdaCC2 against 12 state-of-the-art cost-sensitive boosting approaches (Sect. 5.2) as well as 3 data level methods (SMOTE, Random Over-Sampling and Random Under-Sampling) and 2 model-based methods (SMOTEBoost and RUSBoost) using suitable class imbalance performance evaluation metrics (Sect. 5.1). We have experimented with a large number of real-world datasets (27), depicting various characteristics in terms of class imbalance, dimensionality and cardinality. An overview of the datasets is provided in Table 4. We have used the same pre-processing method on all datasets whenever categorical data were present i.e., one-hot encoding. The employed structures were numpy arrays for all datasets. In addition, all the classification methods which have been employed in this paper were trained on the exact same pre-processed data. The goal of our evaluation is twofold: to compare the different methods in terms of their predictive performance for both classes (Sect. 6.1), and to analyze and compare the internal behavior of our methods with the other approaches in order to understand/explain our methods' superior performance (Sect. 6.2).

For our experiments,[2] we use decision stumps, i.e., decision trees of depth 1, as weak learners for all methods. Regarding the number of weak learners T, we experiment with different numbers $T \in [25, 50, 100, 200]$. For the predictive performance experiments (Sect. 6.1), we report on the average of 10×5-fold cross validation. These results are also used for the significance test of Friedman using Bonferroni correction for validating significance on multiple datasets across various methods [5]. For the experiments on the internal behavior (Sect. 6.2), we do not perform any split rather we train on the complete datasets. By using the entire datasets for training, we avoid fluctuating values which can make the internal analysis of our methods misleading.

5.1 Performance metrics

Due to the imbalanced nature of the learning problem, we report on AUC, balanced accuracy, f1-score, gmean, TNR, and TPR. By following similar logic as [6], we also use a combined overall performance measure (OPM), which averages the aforementioned metrics, since no algorithm outperforms others in all datasets and metrics. All metrics (except AUC which

---

[2] Source code and data are available at: https://github.com/iosifidisvasileios/CumulativeCostBoosting.

Table 4 Datasets

Dataset	Features	Minority	Majority	Ratio (*Min:Maj*)	Source
Abalone	10	391	3786	1:9.68	[9]
Adult census	14	11,202	33,973	1:3.03	[9]
Bank	16	4667	35,337	1:7.57	[9]
Car eval	21	134	1594	1:11.90	[9]
Coil 2000	85	586	9236	1:15.76	[9]
Credit	23	6636	23,364	1:3.52	[9]
Eeg eye	14	6723	8257	1:1.23	[9]
Electricity	8	19,237	26,075	1:1.36	[13]
Isolet	617	600	7197	1:11.99	[9]
Letter img	16	734	19,266	1:26.25	[9]
Mammography	6	260	10,923	1:42.01	[9]
musk2	166	1017	5581	1:5.49	[9]
Optical digits	64	554	5066	1:9.14	[9]
Ozone level	72	73	2463	1:33.74	[9]
Pen digits	16	1055	9937	1:9.42	[9]
Phoneme	5	1586	3818	1:2.41	[10]
Protein hom	74	1296	144,455	1:111.46	[35]
Satimage	36	626	5809	1:9.28	[9]
Scene	294	177	2230	1:12.60	[35]
Sick euthyroid	42	293	2870	1:9.80	[35]
Skin	3	50,859	194,198	1:3.82	[9]
Spambase	53	1813	2788	1:1.54	[9]
Thyroid sick	52	231	3541	1:15.33	[35]
Us crime	100	150	1844	1:12.29	[35]
Webpage	300	981	33,799	1:34.45	[35]
Wilt	5	261	4578	1:17.54	[9]
Wine quality	11	183	4715	1:25.77	[9]

employs the confidence scores of the predictions) can be derived from the confusion matrix of Table 2 as shown in Table 5.

Due to the high amount of datasets, we cannot report on each individual dataset and therefore, similarly to [6, 35, 52], we omit individual dataset results, and report on the average across all datasets.

5.2 Competitors and parameter selection

Our main competitors are 12 cost-sensitive boosting methods, namely, AdaCost (β_2) [11], AdaC1 [46], AdaC2 [46], AdaC3 [46], AdaMEC [34], AdaMEC-Cal. [35], CGAda [25–27], CGAda-Cal. [35], CSB1 [47], CSB2 [47], and RareBoost [23]. We also employ the vanilla AdaBoost [42] to show the differences between cost-sensitive and standard boosting methods. The methods (including ours) are summarized in terms of their key characteristics in Table 3 (as already mentioned, AdaMEC-Cal. and CGAda-Cal. are excluded since they are

Table 5 Performance metrics

Metric	Definition
TPR (also Recall)	$TP/(TP + FN)$
TNR	$TN/(TN + FP)$
Balanced accuracy	$1/2 \cdot (TPR + TNR)$
$f1$-score	$2 \cdot TP/(2 \cdot TP + FP + FN)$
gmean	$\sqrt{TPR \cdot TNR}$
OPM	$1/6 \cdot (AUC + bal.acc + gmean + f1 + TPR + TNR)$

the post-processed versions of AdaMEC and CGAda, respectively). Except for the AdaBoost, RareBoost and our AdaCC1 and AdaCC2 methods, all other methods need to be initialized with the misclassification cost matrix $[C_+, C_-]$. As already discussed, finding the right costs is a tedious task requiring domain/dataset knowledge. To this end, we follow the suggestion of [35, 46] to use grid search for selecting the best class ratio for misclassification costs. In particular, for each dataset, we perform grid search on a variety of different class ratios, namely with $C_+ = 1.0$ and by varying C_- in the range $[0.1 - 1.0]$ with step 0.1. We select the class ratio which achieves the best f1-score as suggested by [35, 46]. Grid search is performed on each fold (on the training set) and each value of $T \in [25, 50, 100, 200]$; therefore, for all 10 iterations and for each different fold, the competitors are fine-tuned.[3]

We have combined the three data-level methods with a decision tree classifier. We augmented the minority class until the class-imbalance was eliminated, i.e., both classes had the same amount of instances. For the under-sampling we also removed instances from the majority class until both classes had the same amount of instances. For the model-level methods, we have used the default parameters, e.g., for SMOTEBoost we set $k = 5$ and varied the number of weak learners same as before and same for RUSBoost.

In addition, we evaluate the impact of the cumulative misclassification costs (Eq. (6)) which allows us to dynamically adjust the costs based on the performance of the partial ensemble and is central to our approach. To this end, we compare AdaCC1 and AdaCC2 with their non-cumulative counterparts, denoted by AdaN-CC1 and AdaN-CC2, respectively. The only difference is that the non-cumulative versions do not take into consideration the cumulative error of the partial ensemble, rather rely on each individual weak learner to estimate the misclassification costs for the next round. More concretely, the partial ensemble up to round t, i.e., $\sum_{j=1}^{t} \alpha_j h_j(x)$ in Eq. (5), is replaced by the corresponding weak learner in round t, i.e., $h_t(x)$.

6 Experiments

We split the experiments into two categories: (i) predictive performance (Sect. 6.1) and (ii) internal analysis (Sect. 6.2). In the first category, we compare the predictive performance of our methods against other cost sensitive boosting competitors using the metrics from Sect. 5.1. Although the aim of this work is to compare cost-sensitive boosting methods, we also

---

[3] Note: We have also used a validation set for tuning the competitors by splitting the training set into 80% training 20% validation (on each fold); however, the results were slightly worse, hence we have tuned competitors on the training set.

highlight in Table 10 the performance of data-level methods such as SMOTE [3] (where the number of neighbors $k = 5$), Random Under-Sampling (RUS) and Random Over-Sampling (ROS) combined with decision tree classifiers. Also, we employ boosting class-imbalance methods such as SMOTEBoost [4] and RUSBoost [44]. In the second category, we compare how our method differs from the others by showing the internal behavior of each method.

6.1 Predictive performance

In this section, we begin by comparing the performance of our method against the employed competitors. We continue by comparing AdaCC with its non-cumulative counterpart AdaN-CC. Note that the performance results, in terms of different evaluation metrics shown in Tables 6 and 7, are averaged over all datasets. Afterwards, we report on the ranking of each method based on the datasets. Finally, we report on the statistical significance of our results. *AdaCC versus competitors:* We begin our analysis for the main competitors in Table 6. AdaCC1 and AdaCC2 are the best in terms of balanced accuracy, gmean, recall (TPR) and OPM (AdaCC1 is also best in AUC). AdaMEC-Cal. follows with a [1.27–1.77%] relative decrease in OPM (it has very close difference with AdaCC2), [3.44–3.57%] relative decrease in balanced accuracy and [4.78–5.16%] relative decrease in gmean comparing to our best performing method (AdaCC2). The fourth performing method is CGAda-Cal. with a [1.54–2.39%] relative decrease in OPM, [4.13–4.49%] decrease in balanced accuracy and [5.82–6.48%] relative decrease in gmean comparing to our best performing method (AdaCC2). In terms of balanced accuracy, gmean and recall, AdaCC1 and AdaCC2 have the best performance. A closer look to the TPR, TNR scores shows that our approaches achieve the best performance for the minority class (higher TPR), while maintaining a moderate performance for the minority class (TNR close to average).

As expected, AdaBoost, which does not tackle imbalance, achieves the highest TNR but lowest TPR. The cost-sensitive competitors are able to produce higher TPR scores than AdaBoost, but still fail to learn the minority class effectively, e.g., AdaC1, AdaC2 and AdaC3 produce [73.3–78.25%] balanced accuracy, [65.44–75.4%] gmean and [61.04–68.21%] TPR scores which are significantly lower in contrast to our methods.

The competitive performance of AdaMEC-Cal. and CGAda-Cal. is mainly due to their high TNR and low recall. AdaMEC-Cal.'s relative difference in recall is [13.87–17.28%] lower than our approaches, and for CGAda-Cal. the relative difference is [15.7–17.98%] lower. RareBoost also calls for special mention as it performs poorly on the minority class but achieves the second best TNR scores. Its outlying behavior is probably related to its strong assumption that $TP > FP$, which cannot be always ensured.

The obtained results indicate that the cost-sensitive boosting competitors are producing higher balanced accuracy in contrast to AdaBoost but they fail to outperform our methods as indicated by balanced accuracy, gmean, recall, and AUC metrics. In addition, some competitors such as AdaCost, AdaC2, AdaC3, CSB1, and CSB2 do not improve their performance for higher values of T in contrast to other competitors. One possible reason for the sub-optimal performance of the competitors might be the non-optimal misclassification cost tuning as a result of the grid search. Our methods avoid this by dynamically adjusting misclassification costs on each boosting round based on the cumulative behavior of the model. *Cumulative versus non-cumulative:* We continue by comparing our methods, AdaCC1 and AdaCC2, with their non-cumulative counterparts, namely AdaN-CC1 and AdaN-CC2, in Table 7. By comparing AdaCC1 to AdaN-CC1 we observe a relative decrease of [16–17.47%] in balanced accuracy, [42.29–56.36%] in gmean, [37.79–63.47%] and [10.01–19.94%] in

Table 6 Results for various evaluation metrics

Method	T	Bal. Acc	Gmean	TPR (Recall)	TNR	F1Score	AUC	OPM
AdaBoost	25	70.24±2.69	56.86±5.9	43.49±5.81	**97.0±0.74**	49.62±5.23	89.86±1.24	67.85±3.25
	50	73.06±2.35	63.03±4.62	49.19±4.93	**96.93±0.52**	55.46±4.32	90.49±1.21	71.36±2.72
	100	75.05±1.79	66.24±3.05	53.17±3.69	**96.93±0.43**	58.69±3.09	90.87±1.19	73.49±1.98
	200	76.09±1.75	67.75±3.1	55.2±3.54	**96.98±0.39**	60.34±2.98	91.09±1.2	74.57±1.93
AdaCC1	25	**83.16±3.29**	82.11±6.23	79.68±8.42	86.65±3.65	55.8±4.86	**90.28±1.83**	**79.61±3.57**
	50	**84.25±2.64**	83.53±4.56	81.92±4.71	86.58±4.83	58.16±3.51	**90.98±1.6**	80.9±2.74
	100	**85.01±2.07**	84.6±2.8	81.67±3.75	88.35±2.2	60.48±2.73	**91.49±1.48**	81.93±2.02
	200	**85.21±1.85**	84.64±2.41	81.19±3.52	89.23±1.55	61.91±2.8	91.78±1.38	82.32±1.83
AdaCC2	25	82.97±2.06	82.47±2.38	80.69±5.79	85.25±3.53	56.24±3.04	89.76±1.91	79.55±1.93
	50	84.17±1.98	83.66±2.33	80.69±5.25	87.65±2.85	58.9±3.0	89.72±2.2	80.79±1.96
	100	84.46±1.95	83.71±2.42	80.54±5.04	88.38±2.82	60.08±3.16	89.16±2.58	81.05±1.99
	200	84.41±1.85	83.31±2.37	79.22±4.76	89.6±2.36	62.4±3.01	87.67±3.56	81.1±2.0
AdaMEC	25	79.65±2.89	77.71±4.03	69.35±7.33	89.95±2.4	61.54±3.13	89.08±1.19	77.88±2.58
	50	79.86±3.03	77.61±4.11	69.09±7.17	90.63±1.87	62.74±3.21	89.91±1.13	78.4±2.69
	100	79.34±2.56	76.87±3.52	67.91±5.78	90.77±1.58	64.5±3.03	90.41±1.08	78.4±2.39
	200	80.24±2.26	77.38±3.52	71.3±4.98	92.95±2.17	65.12±2.92	90.74±1.08	79.01±2.22
AdaMEC-Cal.	25	80.29±2.56	78.44±3.5	68.8±6.03	91.78±1.56	62.46±3.01	89.9±1.33	78.61±2.36
	50	81.45±2.43	79.84±3.23	70.28±5.58	92.62±1.35	64.52±2.92	90.36±1.4	79.84±2.25
	100	82.15±2.34	80.48±3.17	71.27±5.32	93.03±1.24	65.76±2.89	90.3±1.41	80.5±2.21
	200	82.37±2.19	80.48±3.07	71.3±4.98	93.44±1.11	66.65±2.74	89.72±1.47	80.66±2.12
CGAda	25	79.57±2.78	77.45±3.9	67.93±6.48	91.21±1.77	62.39±3.07	89.83±1.29	78.06±2.56
	50	80.53±2.41	78.58±3.35	68.96±5.7	92.09±1.61	64.39±2.93	90.42±1.22	79.16±2.26
	100	81.26±2.36	79.36±3.26	69.91±5.41	92.53±1.31	65.89±2.9	90.79±1.19	79.97±2.23
	200	81.44±2.37	79.27±3.38	69.81±5.4	93.08±1.21	66.95±2.8	90.99±1.24	80.26±2.24
CGAda-Cal.	25	79.86±2.61	77.93±3.66	68.58±6.11	91.14±1.68	**62.97±2.94**	89.9±1.33	78.4±2.41
	50	80.82±2.46	79.0±3.42	69.43±5.74	92.21±1.54	**64.94±2.94**	90.36±1.4	79.46±2.32
	100	81.35±2.36	79.45±3.31	70.02±5.44	92.69±1.37	**66.27±2.93**	90.3±1.41	80.01±2.28
	200	81.65±2.18	79.54±3.14	70.17±5.03	93.14±1.22	**67.27±2.76**	89.72±1.47	80.25±2.14
AdaCost	25	75.72±2.75	71.97±4.2	69.5±6.24	81.93±4.13	50.43±3.53	83.75±2.64	72.22±2.81
	50	76.43±2.74	73.87±4.42	69.62±6.87	83.23±5.19	50.7±3.52	84.3±3.25	73.03±2.83
	100	74.72±2.71	70.69±4.34	71.99±7.01	77.45±4.75	46.25±3.35	83.11±2.77	70.7±2.69
	200	75.15±2.72	70.63±4.28	74.91±6.63	75.4±5.06	45.11±3.26	81.97±2.4	70.53±2.58
CSB1	25	78.07±3.89	74.66±6.09	69.21±10.25	86.92±4.91	59.78±3.95	88.29±1.93	76.16±3.5
	50	77.8±4.68	73.71±7.52	71.52±12.18	84.07±6.65	59.01±4.75	88.05±2.41	75.69±4.2
	100	74.25±6.07	67.5±11.48	71.8±14.83	76.69±11.98	52.7±11.25	86.69±4.12	71.54±5.96
	200	70.17±7.57	59.92±16.35	75.6±16.68	64.75±19.14	42.86±9.69	84.46±6.53	66.29±7.76
CSB2	25	78.21±3.62	75.2±5.55	67.4±9.16	89.02±3.93	57.69±3.34	89.47±1.59	76.17±3.24
	50	73.97±2.52	66.2±4.99	59.17±7.38	88.77±4.14	57.11±3.83	90.35±1.43	72.6±2.74
	100	75.07±1.86	66.27±3.28	60.57±3.69	89.56±0.6	59.74±3.1	90.07±1.77	73.55±2.11
	200	76.09±1.73	67.65±2.89	62.6±3.52	89.57±0.38	61.37±2.94	90.27±1.77	74.59±1.99
AdaC1	25	73.99±2.46	69.39±4.43	61.09±6.49	86.9±3.58	52.34±3.8	85.44±2.77	71.52±2.64
	50	76.03±2.35	72.43±3.74	61.75±6.18	90.31±3.1	57.73±3.46	86.96±2.33	74.2±2.37
	100	76.57±2.42	73.01±3.79	62.56±6.42	90.59±3.22	58.97±3.42	87.31±2.34	74.84±2.42
	200	76.82±2.4	72.94±3.92	61.74±6.07	91.9±2.94	60.13±3.31	88.07±2.19	75.26±2.41
AdaC2	25	78.06±3.79	75.4±5.53	67.53±10.34	86.8±4.74	58.43±3.35	88.16±2.38	76.03±3.39
	50	76.65±3.29	72.9±5.46	61.04±9.1	92.26±4.1	59.75±3.75	88.48±2.57	75.18±3.22
	100	77.69±2.82	74.05±4.91	61.79±7.86	93.59±3.15	61.95±3.37	89.18±2.1	76.37±2.82
	200	78.25±2.46	74.22±4.21	62.03±6.75	94.48±2.3	63.04±3.15	89.92±1.88	76.99±2.4
AdaC3	25	75.71±3.84	70.31±6.81	68.21±10.45	83.21±6.31	52.32±3.68	86.51±2.49	72.71±3.48
	50	73.3±3.46	65.44±7.09	63.87±9.58	82.74±7.2	52.44±3.72	86.88±2.67	70.78±3.5
	100	74.06±3.35	66.77±7.15	61.88±8.62	86.23±6.14	55.79±3.74	87.53±2.56	72.04±3.5
	200	75.2±3.09	68.26±6.37	61.59±8.18	88.82±4.89	58.79±3.61	88.35±2.23	73.5±3.19
RareBoost	25	75.83±1.99	68.44±3.57	54.94±4.07	96.72±0.48	60.03±3.39	81.96±2.79	72.99±2.31
	50	77.51±1.83	70.96±3.14	58.03±3.6	96.84±0.39	62.82±3.09	78.64±2.78	74.16±2.09
	100	78.55±1.78	72.47±3.08	60.19±3.6	96.91±0.39	64.44±3.01	75.02±2.59	74.6±2.05
	200	79.04±1.69	73.29±2.94	61.16±3.4	96.93±0.4	65.18±2.96	71.39±2.46	74.5±1.96

Best and second best methods per different values of T are in bold and circled, respectively. Colors indicate specific values of T

Table 7 Results for various evaluation metrics for the comparison of AdaCC1/2 versus AdaN-CC1/2

Method	T	Bal. Acc	Gmean	TPR (Recall)	TNR	F1Score	AUC	OPM
AdaCC1	25	**83.16±3.29**	82.11±6.23	79.68±8.42	86.65±3.65	55.8±4.86	**90.28±1.83**	**79.61±3.57**
	50	**84.25±2.64**	83.53±4.56	81.92±4.71	86.58±4.83	58.16±3.51	**90.98±1.6**	80.9±2.74
	100	**85.01±2.07**	84.6±2.8	81.67±3.75	88.35±2.2	60.48±2.73	**91.49±1.48**	81.93±2.02
	200	**85.21±1.85**	84.64±2.41	81.19±3.52	89.23±1.55	61.91±2.8	91.78±1.38	82.32±1.83
AdaCC2	25	82.97±2.06	82.47±2.38	80.69±5.79	85.25±3.53	56.24±3.04	89.76±1.91	79.55±1.93
	50	84.17±1.98	83.66±2.33	80.69±5.25	87.65±2.85	58.9±3.0	89.72±2.2	80.79±1.96
	100	84.46±1.95	83.71±2.42	80.54±5.04	88.38±2.82	60.08±3.16	89.16±2.58	81.05±1.99
	200	84.41±1.85	83.31±2.37	79.22±4.76	89.6±2.36	62.4±3.01	87.67±3.56	81.1±2.0
AdaN-CC1	25	70.79±4.59	52.52±10.63	48.74±11.07	92.83±3.14	41.26±8.5	82.06±5.19	64.7±5.92
	50	72.57±4.47	58.7±9.7	59.45±14.94	85.68±11.41	43.97±8.48	79.56±6.36	66.66±5.58
	100	72.6±4.51	58.73±9.73	58.75±13.64	86.45±9.63	43.93±8.44	77.23±7.16	66.28±5.49
	200	72.6±4.51	58.73±9.73	58.46±13.5	86.75±8.87	43.89±8.46	76.52±7.27	66.16±5.43
AdaN-CC2	25	75.23±3.37	66.04±7.1	55.21±8.1	**95.25±2.1**	53.21±5.9	84.88±4.66	71.64±4.11
	50	76.34±3.58	67.14±7.83	61.8±8.91	**90.88±3.29**	56.06±6.06	79.81±7.98	72.0±4.45
	100	76.51±3.72	67.41±8.07	62.15±9.19	**90.86±3.28**	56.34±6.33	75.28±9.63	71.43±4.54
	200	76.5±3.72	67.41±8.07	62.15±9.19	**90.86±3.28**	56.34±6.33	74.9±9.67	71.36±4.51

Best and second best methods per different values of T are in bold and circled, respectively. Colors indicate specific values of T

AUC. There are also high (relative) differences between AdaCC2 and AdaN-CC2. These differences highlight the superiority of the cumulative costs in the reweighting procedure on each boosting round versus the non-cumulative costs.

Ranking: We also report on the ranks based on balanced accuracy across the methods in Table 8, for $T = 200$ (Tables for $T \in [25, 50, 100]$ are included in the "Appendix"). Note that Table 8 contains floats instead of integers due to the fact that in many datasets some methods produced the same balanced accuracy score.

There are some interesting observations from this table. AdaCC1 and AdaCC2 are the best and second-best in ranks with an average rank of 2.06 and 3.19 respectively, in contrast to the competitors; however, methods such as AdaMEC-Cal. and CGAda-Cal. are also achieving high ranks. Furthermore, the last row of Table 8 shows the number of datasets for which each method achieved the best performance. Our approaches, AdaCC1 and AdaCC2, have won on 10 and 8 datasets, while for the majority of datasets AdaCC1 or AdaCC2 were the best or second best methods. Similar behavior can also be observed for other values of T, where AdaCC1 and AdaCC2 achieve the best ranking scores, e.g., AdaCC1 achieves the best ranking for $T \in [25, 50, 100]$ with values 2.30, 2.33 and 2.11, respectively and AdaCC2 achieves the second best ranking with scores 2.70, 2.41 and 2.52.

In Table 10, we also compare non-cost-sensitive methods with our approach. We have used three well-known data-level methods such as SMOTE, Random Over-Sampling (ROS) and Random Under-Sampling (RUS) combined with a decision tree classifier, and also two model-based boosting methods such as SMOTEBoost and RUSBoost. As we can see, AdaCC performs better than the other methods in terms of balanced accuracy, gmean, auc and OPM. It is also visible that RUSBoost is able to maintain extremely high TPR scores; however, it under-performs in terms of TNR in contrast to AdaCC which maintains both TNR and TPR at high levels. Interestingly, by comparing Tables 6 and 10, we can observe that the non cost-sensitive methods are able to outperform several cost-sensitive methods.

Statistical significance: Finally, for the comparison of cost-sensitive methods we have performed the Friedman test ($p < 0.05$) using the Bonferroni correction [5] for comparing multiple methods across multiple datasets. The results can be seen in Table 9, in which non-significant values have been highlighted in bold. As we see, AdaCC1 and AdaCC2 are not significantly different across various values of T. AdaCC1 and AdaCC2 are significantly different compared to the other competitors. One interesting observation is that for high T, AdaMEC-Cal. and CGAda-Cal. are able to produce similar results as our methods.

6.2 Internal analysis

We begin the internal analysis by comparing our methods, AdaCC1 and AdaCC2, with their corresponding non-cumulative version, namely AdaN-CC1 and AdaN-CC2, which are introduced in Sect. 5.2. Then, we continue our analysis in which we compare our methods with competitors w.r.t. in-training instance re-weighting, α estimation, feature importance, confidence scores and decision boundaries (similar to the toy example in Fig. 1).

Cumulative versus non-cumulative costs: In Fig. 2 we compare AdaCC1/2 and AdaN-CC1/2 on the TPR and TNR values per boosting round (averaged over the datasets, $T = 200$). Figure 2a shows the in-training TPR scores over the boosting rounds. It is clear that the cumulative versions, i.e., AdaCC1 and AdaCC2, are by far better and more stable than the non-cumulative ones, AdaN-CC1 and AdaN-CC2. Figure 2b shows the in-training TNR scores over the boosting rounds. The non-cumulative versions are better than the cumulative ones. However, they exhibit high fluctuation as they rely on point-in-time estimates of

Table 8 Comparative balanced accuracy ranks across the entire set of methods and datasets (smaller values are better) for $T = 200$

	AdaBoost	AdaCC1	AdaCC2	AdaMEC	AdaMEC-Cal	CGAda	CGAda-Cal	AdaCost	CSB1	CSB2	AdaC1	AdaC2	AdaC3	RareBoost
Abalone	14.0	3.0	4.0	2.0	7.0	9.0	8.0	**1.0**	11.0	13.0	5.0	6.0	10.0	12.0
Adult	9.5	**1.0**	2.0	9.5	5.0	4.0	3.0	7.0	14.0	9.5	12.0	9.5	13.0	6.0
Bank	12.5	**1.0**	3.0	2.0	6.0	4.0	5.0	7.0	14.0	12.5	9.0	11.0	8.0	10.0
Car eval	9.5	**1.0**	3.0	9.5	5.0	6.0	4.0	14.0	12.0	9.5	2.0	9.5	13.0	7.0
Coil 2000	13.0	**1.0**	3.0	2.0	8.0	5.0	7.0	9.0	10.0	14.0	4.0	6.0	12.0	11.0
Credit	13.5	2.5	8.0	5.0	**1.0**	4.0	2.5	7.0	11.0	13.5	6.0	10.0	9.0	12.0
Eeg eye	5.0	2.0	11.0	12.0	3.0	8.0	7.0	14.0	13.0	5.0	10.0	5.0	9.0	**1.0**
Electricity	7.5	2.5	10.0	7.5	4.0	2.5	5.0	13.0	14.0	7.5	12.0	7.5	11.0	**1.0**
Isolet	9.5	**1.0**	2.0	9.5	4.0	7.0	5.0	14.0	3.0	9.5	12.0	9.5	13.0	6.0
Letter img	8.5	2.0	**1.0**	8.5	4.0	5.0	6.0	13.0	14.0	8.5	11.5	8.5	11.5	3.0
Mammography	9.5	2.0	**1.0**	9.5	3.0	5.0	6.0	4.0	14.0	9.5	12.0	9.5	13.0	7.0
musk2	8.5	3.0	2.0	8.5	4.0	6.0	5.0	11.0	14.0	8.5	12.5	8.5	12.5	**1.0**
Optical digits	8.5	**1.0**	2.0	8.5	5.0	6.0	4.0	14.0	13.0	8.5	11.5	8.5	11.5	3.0
Ozone level	11.5	2.0	3.0	11.5	4.0	6.0	5.0	**1.0**	7.0	11.5	14.0	11.5	8.0	9.0
Pen digits	8.5	2.0	**1.0**	8.5	6.0	4.0	5.0	14.0	13.0	8.5	11.5	8.5	11.5	3.0
Phoneme	9.5	5.0	4.0	7.0	2.0	1.0	3.0	12.0	14.0	9.5	11.0	6.0	13.0	8.0
Protein hom	8.5	2.0	**1.0**	8.5	3.0	4.0	6.0	14.0	13.0	8.5	11.5	8.5	11.5	5.0
Satimage	12.0	2.0	**1.0**	10.0	4.0	5.0	3.0	9.0	14.0	12.0	8.0	12.0	7.0	6.0
Scene	13.5	**1.0**	7.0	9.0	5.0	4.0	6.0	3.0	8.0	13.5	2.0	12.0	11.0	10.0
Sick euthyroid	9.5	2.0	**1.0**	9.5	7.0	4.0	6.0	12.0	14.0	9.5	3.0	9.5	5.0	13.0
Skin	8.5	5.0	4.0	8.5	2.5	1.0	2.5	13.0	14.0	8.5	11.5	8.5	11.5	6.0
Spam	8.5	4.5	2.0	8.5	4.5	6.0	3.0	14.0	13.0	8.5	11.5	8.5	11.5	**1.0**
Thyroid sick	10.0	**1.0**	2.0	8.0	4.0	6.0	5.0	3.0	14.0	10.0	12.0	10.0	13.0	7.0
Us crime	9.5	**1.0**	4.0	9.5	6.0	7.0	5.0	3.0	2.0	9.5	13.0	9.5	12.0	14.0

Table 8 continued

	AdaBoost	AdaCC1	AdaCC2	AdaMEC	AdaMEC-Cal	CGAda	CGAda-Cal	AdaCost	CSB1	CSB2	AdaC1	AdaC2	AdaC3	RareBoost
Webpage	11.5	2.0	**1.0**	3.0	5.0	7.0	6.0	8.0	9.0	11.5	14.0	10.0	13.0	4.0
Wilt	9.5	2.0	**1.0**	9.5	3.0	5.0	4.0	14.0	6.0	9.5	12.5	9.5	12.5	7.0
Wine quality	13.5	**1.0**	2.0	3.0	5.0	7.0	6.0	4.0	12.0	13.5	8.0	9.0	10.0	11.0
Avg	10.11	**2.06**	3.19	7.70	4.44	5.13	4.93	9.33	11.48	10.11	9.74	8.96	11.00	6.81
Winner	0	**10**	8	0	1	2	0	2	0	0	0	0	0	4

Best methods per dataset are in bold. Last row (winner) indicates on how many datasets a method is ranked first (best balanced accuracy score, higher values are better). Note that in some datasets, the methods have equal scores (tie); therefore, the ranks are in float format

Table 9 Friedman test: p-values for all competitors

$T =$	AdaCC1				AdaCC2			
	25	50	100	200	25	50	100	200
AdaBoost	0.0e+00	3.6e−14	6.5e−13	4.0e−11	6.6e−15	4.9e−14	8.5e−12	1.1e−08
AdaCC1	–	–	–	–	**7.7e−01**	**9.6e−01**	**7.3e−01**	**3.9e−01**
AdaCC2	**7.7e−01**	**9.6e−01**	**7.3e−01**	**3.9e−01**	–	–	–	–
AdaMEC	2.7e−03	3.7e−04	8.7e−06	7.6e−06	7.5e−03	4.5e−04	4.6e−05	3.7e−04
AdaMEC-Cal	1.2e−03	6.1e−03	**8.7e−02**	**7.6e−02**	2.9e−02	6.8e−03	1.8e−02	**4.3e−01**
CGAda	2.8e−03	3.4e−02	2.2e−02	3.1e−02	7.7e−03	3.8e−02	4.0e−02	4.7e−02
CGAda-Cal	4.7e−03	3.4e−02	2.2e−02	**8.0e−02**	1.2e−02	3.8e−02	4.0e−02	**3.0e−01**
AdaCost	2.6e−09	9.5e−09	1.0e−09	2.1e−09	1.6e−08	1.2e−08	9.5e−09	3.6e−07
CSB1	7.8e−07	1.1e−07	2.7e−12	3.3e−15	3.6e−06	1.4e−07	3.3e−11	3.0e−12
CSB2	5.7e−06	1.6e−13	5.6e−13	3.6e−11	2.3e−05	2.3e−13	7.4e−12	1.0e−08
AdaC1	2.9e−09	9.5e−09	1.0e−09	3.6e−10	1.7e−08	1.2e−08	9.5e−09	7.7e−08
AdaC2	5.7e−06	8.0e−09	8.3e−09	2.9e−08	2.3e−05	1.0e−08	6.7e−08	3.4e−06
AdaC3	1.9e−09	5.0e−12	3.4e−13	1.1e−13	1.2e−08	6.8e−12	4.6e−12	6.4e−11
RareBoost	6.976e−08	6.581e−06	1.502e−04	1.8e−04	3.6e−07	8.1e−06	6.4e−04	5.0e−03

Non-significant values ($p > 0.05$) are in bold. Colors indicate specific values of $T = [25, 50, 100, 200]$

Table 10 Results for various evaluation metrics for non-cost-sensitive class-imbalance methods

Method	T	Bal. Acc	Gmean	TPR (Recall)	TNR	F1Score	AUC	OPM
AdaCC1	25	**83.16±3.29**	82.11±6.23	79.68±8.42	86.65±3.65	55.8±4.86	**90.28±1.83**	**79.61±3.57**
	50	**84.25±2.64**	83.53±4.56	81.92±4.71	86.58±4.83	58.16±3.51	**90.98±1.6**	80.9±2.74
	100	**85.01±2.07**	84.6±2.8	81.67±3.75	88.35±2.2	60.48±2.73	**91.49±1.48**	81.93±2.02
	200	**85.21±1.85**	84.64±2.41	81.19±3.52	89.23±1.55	61.91±2.8	**91.78±1.38**	82.32±1.83
AdaCC2	25	82.97±2.06	82.47±2.38	80.69±5.79	85.25±3.53	56.24±3.04	89.76±1.91	79.55±1.93
	50	84.17±1.98	83.66±2.33	80.69±5.25	87.65±2.85	58.9±3.0	89.72±2.2	80.79±1.96
	100	84.46±1.95	83.71±2.42	80.54±5.04	88.38±2.82	60.08±3.16	89.16±2.58	81.05±1.99
	200	84.41±1.85	83.31±2.37	79.22±4.76	**89.6±2.36**	62.4±3.01	87.67±3.56	81.1±2.0
SMOTE + D.T.	1	79.41±2.11	76.53±2.93	65.24±4.17	93.58±0.7	61.31±2.91	79.35±2.14	75.9±2.34
ROS + D.T.	1	78.27±1.97	74.24±3.06	61.67±3.9	94.87±0.59	61.99±3.06	78.09±2.0	74.86±2.3
RUS + D.T.	1	82.35±1.99	82.29±2.0	82.8±3.73	81.9±2.12	50.44±2.82	82.34±2.01	77.02±2.05
RUSBoost	25	79.36±2.29	77.07±3.03	79.38±3.6	79.34±3.72	51.57±3.18	88.41±2.06	75.86±2.33
	50	76.35±3.02	71.98±4.88	**88.92±2.29**	63.79±5.93	43.98±2.93	86.99±2.86	72.0±3.05
	100	70.16±3.66	60.31±7.52	**94.67±1.7**	45.64±7.42	36.22±2.39	84.34±3.83	65.22±3.79
	200	64.61±3.94	48.61±9.69	**97.8±1.21**	31.43±8.26	30.29±1.96	81.68±4.93	59.07±4.34
SMOTEBoost	25	80.46±2.1	77.93±2.8	68.51±4.37	**92.42±0.89**	63.15±2.85	89.3±1.42	78.62±2.07
	50	81.69±2.0	79.49±2.61	71.93±4.07	**91.45±0.85**	**64.71±2.63**	89.71±1.39	79.83±1.96
	100	82.06±1.89	79.89±2.47	73.72±3.82	**90.4±0.82**	**65.14±2.48**	89.63±1.54	80.14±1.88
	200	81.91±1.82	79.58±2.43	74.46±3.72	89.37±0.81	**65.14±2.33**	89.63±1.43	80.01±1.79

Best and second best methods per different values of T are in bold and circled, respectively. Colors indicate specific values of T

(a) In-training TPR per boosting round

(b) In-training TNR per boosting round
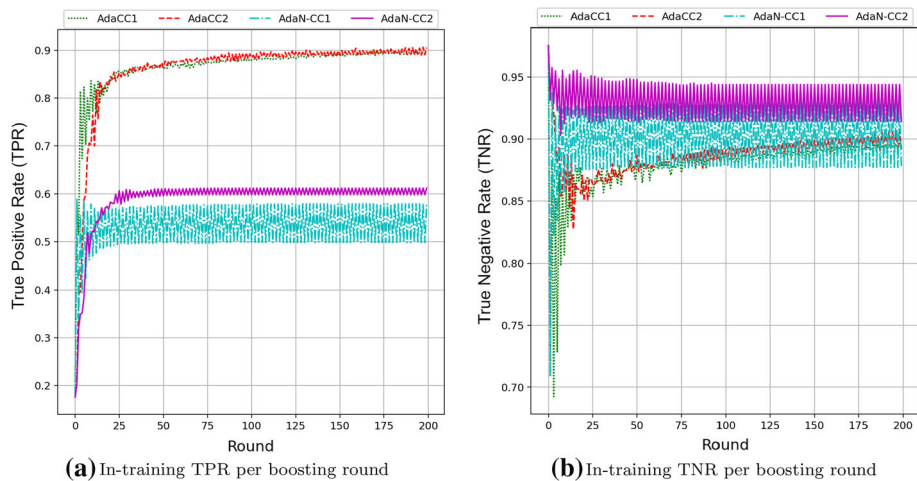
Fig. 2 Cumulative vs non-cumulative misclassification cost estimation (left:TPR, right:TNR)

misclassification costs (i.e., based on individual weak learners) comparing to the cumulative methods which rely on cumulative estimates (i.e., based on the partial ensemble). These experiments demonstrate the importance of the cumulative misclassification cost estimation for the stability of the model. Also, in terms of predictive performance, we have seen (c.f., Table 6) that the non-cumulative methods, AdaN-CC1 and AdaN-CC2, are producing significantly worse results in contrast to AdaCC1 and AdaCC2.

Model performance analysis: The experiments thus far demonstrate the superior behavior of AdaCC1 and AdaCC2, compared to state-of-the-art cost-sensitive boosting approaches. Hereafter, we explain this behavior through additional experiments on the internal behavior of the models, assessed by: (i) positive (minority) class weight assignments over the boosting rounds (Fig. 3a), (ii) alpha values over the boosting rounds (Fig. 3b), (iii) in-training balanced error over the boosting rounds (Fig. 3c), (iv) feature importance (Fig. 4) of a given dataset (mammography), (iv) confidence scores (Fig. 5), and (v) decision boundaries (Fig. 6). Moreover, AdaMEC, AdaMEC-Cal., CGAda-Cal. are omitted from these experiments (except
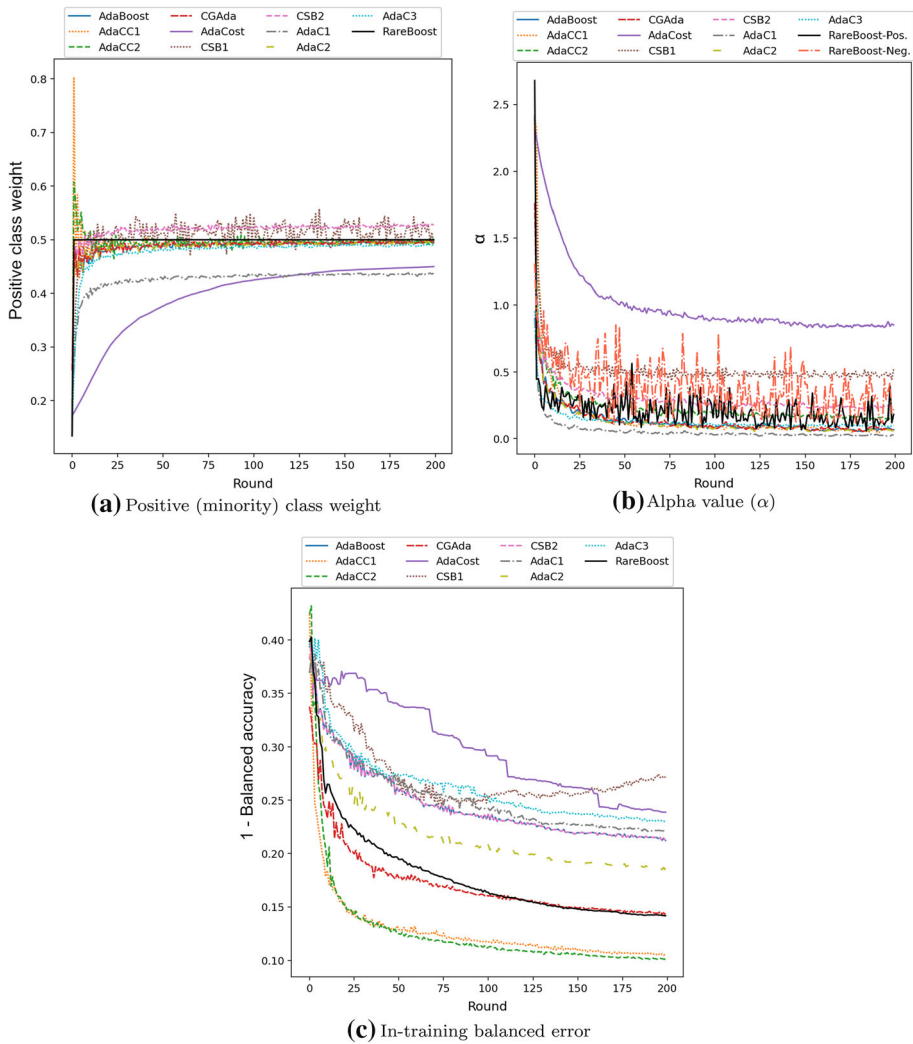
(a) Positive (minority) class weight

(b) Alpha value (α)

(c) In-training balanced error

Fig. 3 In-training behavior over the boosting rounds (for $T = 200$)

the decision boundary analysis). The reason is that AdaMEC is built on top of a trained AdaBoost model, by shifting its decision boundary towards the target class. AdaMEC-Cal. and CGAda-Cal. are calibrated versions of AdaMEC and CGAda.

In-training analysis: For in-training analysis, we set $T = 200$ and show the behavior of each method per boosting round. The weights of the minority class over the boosting rounds are shown in Fig. 3a; as we can see, AdaCC1 and AdaCC2 behave differently from the competitors by starting with very high weights during the first boosting rounds, which converge afterwards to 0.5. The other methods increase the positive weights gradually over the rounds. Our methods tackle the class imbalance problem during early boosting rounds by assigning cumulative misclassifications costs to the minority class and then proceed to reduce these costs (dynamically) as soon as the TPR scores are close to TNR scores.

In terms of α values, which control how much the weak learners contribute to the final ensemble (Fig. 3b), the methods depict a similar behavior with α decreasing over the boosting
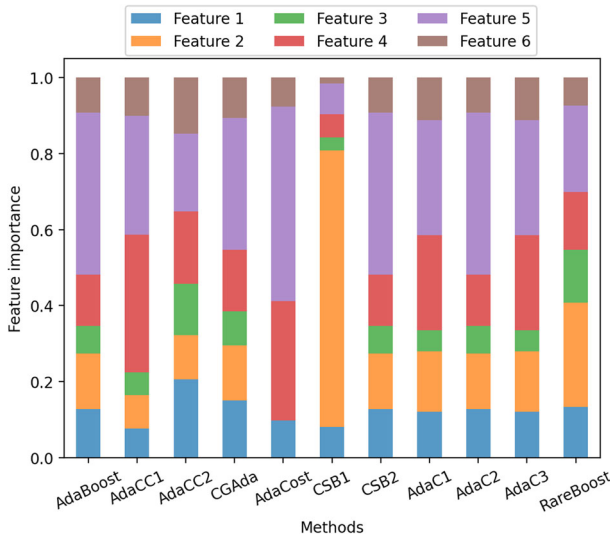
Fig. 4 Feature importance of mammography dataset (the higher, the more important the feature)

rounds. A notable exception is RareBoost which utilizes positive and negative α to estimate the weight distribution per round; thus, it is expected for its α values to fluctuate. Our methods do not differentiate from other competitors (excluding RareBoost); weak learners in the early boosting rounds (e.g., $T < 10$) are more influential to the final outcomes (higher α values).

In Fig. 3c the in-training balanced error over the boosting rounds is shown. As we can see, our methods achieve the lowest error. Moreover, AdaCC1 and AdaCC2 reduce the balanced error faster than any other method, and converge after a sufficient number of boosting rounds. The abrupt reduction of the error is directly related to the rapid increase of the positive weights in the initial boosting rounds.

Feature importance: In Fig. 4 we illustrate the feature importance for each method on the mammography dataset. We have selected this dataset since it has low dimensionality (6 features) and high class imbalance ratio (1:42). Figure 4 shows the importance of each feature which is employed by each method to make a decision (weights are normalized to be a distribution). Note that each weak learner is a decision stump which means that it selects only one feature for splitting the dataset. The feature importance is measured as follows: each ensemble consists of T weak learners and each weak learner is trained on a different data distribution. Since we have employed Decision Stumps (Decision trees of depth 1), each weak learner will use only one split; therefore, it will use only one feature. The weak learners of AdaCost, based on the data distributions which are provided (based on the model's updating strategy), do not use some features based on the splitting criterion. In addition, some models (e.g., AdaCost) may terminate their boosting rounds earlier than others based on their stopping criterion which can lead to ignoring some features. Although the feature importance does not indicate which method is the best, it shows clearly that each method utilizes differently the features based on the weighting strategy, e.g., AdaCC1 is relying more on features 4 and 5 and less on features 1 and 3 compared to AdaCC2.

Confidence analysis: In Fig. 5, we compare the confidence scores of the different methods for two ensemble sizes, $T = 25$ (Fig. 5a) and $T = 200$ (Fig. 5b), and separate them into three categories: positive (left), negative (middle) and overall (right) confidence scores. Note that
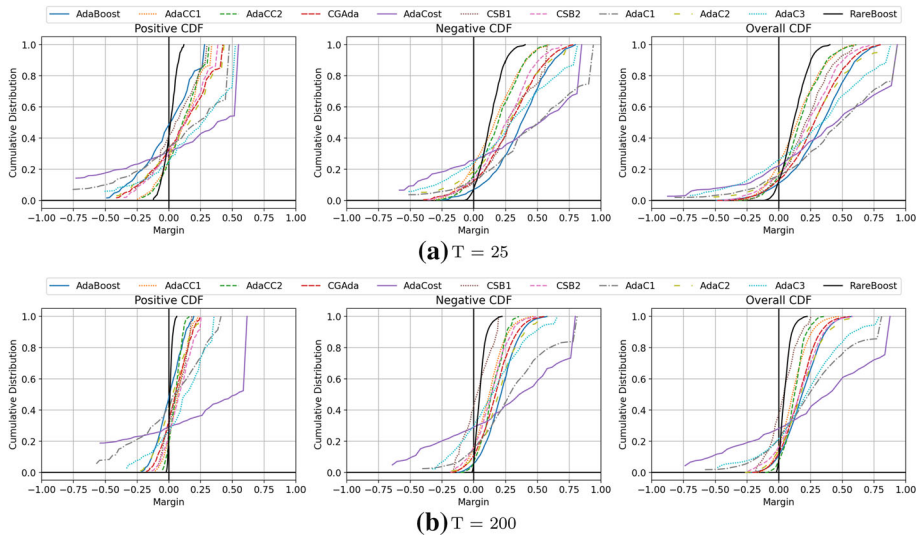
Fig. 5 Effect of boosting rounds on the confidence scores (left:positive class, middle:negative class, right :overall)

misclassified instances have confidence scores less than 0 on x-axis (values closer to 0, on x-axis, indicate lower confidence in the predictions, correct or wrong). Also, the area under the line in the range $[-1, 0]$ on the x-axis shows the proportion of misclassified instances.

At a first look at the overall confidence scores, we see that AdaCC1 and AdaCC2 are producing low misclassification rates while the area under the line in the $[-1, 0]$ range of x-axis is low. However, other methods are achieving similar results. Therefore, we need to analyze the confidence scores of each class separately since the minority (positive) class is overshadowed by the majority (negative) class. As expected, AdaBoost has the highest misclassification confidence score in positive (minority) class since it learns effectively only the negative (majority) class. AdaCC1 and AdaCC2 methods have the lowest misclassification confidence scores for $T = 25$, and reduce them even more as the number of weak learners increases, i.e., for $T = 200$. For the negative class, our approaches are able to reduce the misclassified confidence scores as the number of weak learners increases. Other competitors are able to reduce the positive misclassification confidence scores; however, their misclassification confidence scores (are under the line) for the negative (majority) class are increasing, e.g., CSB1, AdaC3. This highlights once more that the ability to adjust the weights during training is crucial to maintain good predictive performance across both classes. Note that for intermediate values of T, Figures are included in the "Appendix" as they depict this gradual behavior.

An interesting observation is that the cost-sensitive methods become less confident in the confidence of the correctly classified instances (both classes) as the number of weak learners increases. As it seems, *the more they learn*, their mistakes are reduced but they also *become less confident* in their correct decisions.

Decision boundary analysis: Finally, we generate an imbalanced dataset similar to the toy dataset in Fig. 1 of 40 instances (30 red class and 10 green class) with 2 features (for better visualization). We train each method on the same dataset and afterwards we show the decision boundaries which are learned from the training set. Since the dataset has only two features x_1 and x_2, we use a small number of weak learners ($T = 5$). In Fig. 6 we show the decision

Fig. 6 Decision boundaries of methods on the same imbalanced toy dataset of 10 blue and 30 red instances. Dot size is proportional to the weight allocated by each weak learner to the particular instance, making clear how each method assigns weights to minority class instances compared to the ones of the majority class

boundaries of all methods and how each method changes the weight distribution over the boosting rounds.

As we can see, AdaBoost gives more emphasis to the majority (red) class, while it tunes for overall classification accuracy. AdaCC1 and AdaCC2 on this particular dataset

behave similarly by properly partitioning the space, giving emphasis to minority class without deteriorating the performance on majority class (2 blue misclassified points versus 4 red misclassified points). AdaMEC and AdaMEC-Cal. cannot find, through grid search, good misclassifications costs; therefore, their behavior is similar to AdaBoost (by considering the best $C_N = 1$ which makes them behave equal to AdaBoost). The misclassification cost selection of the competitors is based upon the performance of the final ensemble while our methods dynamically adapt their misclassification costs on each boosting round. CGAda, CSB1, CSB2 and AdaC2 partition the space to allow higher recall scores; however, they misclassify 12 red points. AdaCost, AdaC1 and AdaC3 perform even worse by misclassifying 19 red points. Interestingly, RareBoost partitions the space in a safe way, e.g., it correctly classifies 5 blue points and the majority class.

7 Conclusions and future work

In this work we present a novel strategy for cost-sensitive boosting that exploits the cumulative behavior of the model to dynamically balance the misclassification costs on each boosting round.

Existing approaches require a user-defined fixed misclassification cost matrix as input. In most cases this results in additional hyperparameters which need to be optimized jointly with the basic parameters, e.g., using grid search. As grid-search does not ensure a good initialization it might hurt the model's overall predictive performance. Our methods' ability to produce consistent improvements in different measures, e.g., [0.3–28.56%] for the AUC, [3.4–21.4%] for the balanced accuracy, [4.8–45%] for gmean and [7.4–85.5%] for the recall indicate the general applicability of our method. The high recall scores demonstrate, that our method is especially helpful for domains in which low recall scores have a disastrous impact. Moreover, we have shown the superior performance of such cumulative models comparing to their non-cumulative counterparts, in terms of both predictive performance and model stability. Finally, our method comes with theoretical guarantees w.r.t. the training error and it reduces the optimization of hyper-parameters.

In the future, we will consider multi-class extensions of our method. Furthermore, we plan to investigate our method's application to the supervised online learning task. Our method's ability to dynamically adjust the misclassification costs, makes our method suitable for such a task in contrast to a recent online cost-sensitive boosting extension of AdaC2 [48].

Appendix

See Tables 11,12,13 and Fig. 7

Table 11 Comparative balanced accuracy ranks across the entire set of methods and datasets (smaller values are better) for $T = 25$

	AdaBoost	AdaCC1	AdaCC2	AdaMEC	AdaMEC-Cal	CGAda	CGAda-Cal	AdaCost	CSB1	CSB2	AdaC1	AdaC2	AdaC3	RareBoost
Abalone	14.0	3.0	5.0	2.0	9.0	8.0	10.5	4.0	10.5	**1.0**	6.0	7.0	12.0	13.0
Adult	14.0	**1.0**	2.0	3.0	7.0	6.0	8.0	5.0	10.0	13.0	12.0	4.0	9.0	11.0
Bank	14.0	2.0	**1.0**	8.0	3.0	5.0	4.0	9.0	11.0	6.0	12.0	10.0	7.0	13.0
Car eval	12.0	**1.0**	2.0	6.0	4.5	3.0	4.5	13.0	8.0	11.0	14.0	10.0	9.0	7.0
Coil 2000	14.0	5.0	2.0	7.0	10.0	9.0	6.0	11.0	12.0	**1.0**	4.0	8.0	3.0	13.0
Credit	12.5	4.0	9.0	5.0	**1.0**	3.0	2.0	7.0	8.0	12.5	6.0	11.0	10.0	14.0
Eeg eye	4.0	3.0	8.0	9.0	2.0	5.0	7.0	14.0	12.0	10.0	6.0	11.0	13.0	**1.0**
Electricity	8.5	**1.0**	7.0	5.0	3.0	6.0	4.0	12.0	14.0	8.5	11.0	10.0	13.0	2.0
Isolet	14.0	2.0	**1.0**	7.0	11.0	10.0	9.0	12.0	5.0	3.0	13.0	4.0	6.0	8.0
Letter img	9.5	**1.0**	2.0	9.5	4.0	5.0	6.0	7.0	12.0	9.5	14.0	9.5	13.0	3.0
Mammography	14.0	2.0	**1.0**	8.0	3.0	6.0	4.0	7.0	5.0	11.0	9.0	12.0	10.0	13.0
musk2	13.0	2.0	**1.0**	8.0	3.0	5.0	7.0	9.0	4.0	13.0	11.0	10.0	10.0	3.0
Optical digits	10.0	2.0	**1.0**	6.0	6.0	5.0	7.0	13.0	4.0	10.0	12.0	10.0	14.0	3.0
Ozone level	14.0	**1.0**	4.0	7.0	9.0	11.0	10.0	6.0	8.0	2.0	12.0	3.0	5.0	13.0
Pen digits	10.0	2.0	**1.0**	7.0	5.0	3.5	3.5	12.0	8.0	10.0	13.0	10.0	14.0	6.0
Phoneme	13.0	7.0	6.0	4.0	3.0	**1.0**	2.0	11.0	8.0	12.0	10.0	5.0	14.0	9.0
Protein homo	10.5	**1.0**	2.0	10.5	3.0	6.0	8.0	14.0	13.0	10.5	5.0	10.5	7.0	4.0
Satimage	14.0	**1.0**	2.0	6.0	8.5	10.0	8.5	12.0	7.0	4.0	11.0	5.0	3.0	13.0
Scene	14.0	3.0	4.0	7.0	11.0	8.0	10.0	5.0	9.0	**1.0**	6.0	2.0	12.0	13.0
Sick euthyroid	10.0	2.0	**1.0**	6.0	7.5	4.0	7.5	13.0	5.0	10.0	3.0	10.0	12.0	14.0
Skin	14.0	5.0	**1.0**	6.0	2.5	4.0	2.5	13.0	9.0	8.0	11.0	7.0	12.0	10.0
Spam	10.5	4.0	2.0	10.5	5.5	3.0	5.5	14.0	13.0	10.5	7.0	10.5	8.0	**1.0**
Thyroid sick	13.0	**1.0**	2.0	4.0	5.0	7.0	6.0	3.0	8.0	11.0	10.0	9.0	14.0	12.0
Us crime	13.0	**1.0**	2.0	5.0	6.0	10.0	7.0	4.0	3.0	11.0	8.0	12.0	9.0	14.0

Table 11 continued

	AdaBoost	AdaCC1	AdaCC2	AdaMEC	AdaMEC-Cal	CGAda	CGAda-Cal	AdaCost	CSB1	CSB2	AdaC1	AdaC2	AdaC3	RareBoost
Webpage	13.0	**1.0**	2.0	5.0	4.0	8.0	3.0	11.0	7.0	9.0	14.0	6.0	12.0	10.0
Wilt	13.0	2.0	**1.0**	8.0	4.0	9.0	5.0	12.0	10.0	3.0	14.0	6.0	7.0	11.0
Wine quality	14.0	2.0	**1.0**	4.0	6.0	9.0	7.0	10.0	8.0	3.0	11.0	5.0	12.0	13.0
Avg	12.20	**2.30**	2.70	6.43	5.61	6.28	6.09	9.74	8.57	7.94	9.81	8.17	10.00	9.15
Winner	0	**10**	**10**	0	1	1	0	0	0	3	0	0	0	2

Best methods per dataset are in bold. Last row (winner) indicates on how many datasets a method is ranked first (best balanced accuracy score, higher values are better). Note that in some datasets, the methods have equal scores (tie); therefore, the ranks are in float format

Table 12 Comparative balanced accuracy ranks across the entire set of methods and datasets (smaller values are better) for $T = 50$

	AdaBoost	AdaCC1	AdaCC2	AdaMEC	AdaMEC-Cal	CGAda	CGAda-Cal	AdaCost	CSB1	CSB2	AdaC1	AdaC2	AdaC3	RareBoost
Abalone	14.0	2.0	3.0	**1.0**	8.0	7.0	9.0	4.0	10.0	12.0	5.0	6.0	11.0	13.0
Adult	12.0	3.0	**1.0**	2.0	5.0	6.0	4.0	7.0	14.0	12.0	10.0	12.0	8.0	9.0
Bank	13.5	2.0	**1.0**	6.0	5.0	3.5	3.5	8.0	9.0	13.5	11.0	10.0	7.0	12.0
Car eval	11.0	2.0	**1.0**	8.0	3.5	5.0	3.5	13.0	7.0	11.0	9.0	11.0	14.0	6.0
Coil 2000	14.0	3.0	**1.0**	9.0	8.0	6.0	4.0	7.0	10.0	13.0	2.0	5.0	11.0	12.0
Credit	12.5	3.0	8.0	6.0	**1.0**	2.0	5.0	4.0	10.0	12.5	7.0	11.0	9.0	14.0
Eeg eye	3.0	4.0	9.0	11.0	2.0	6.0	5.0	12.0	14.0	7.0	8.0	10.0	13.0	**1.0**
Electricity	8.5	3.0	6.0	7.0	4.0	2.0	5.0	13.0	14.0	8.5	11.0	10.0	12.0	**1.0**
Isolet	12.5	**1.0**	2.0	12.5	6.0	7.0	3.0	8.0	5.0	12.5	10.0	12.5	9.0	4.0
Letter img	9.5	**1.0**	2.0	9.5	4.0	6.0	7.0	12.0	5.0	9.5	13.0	9.5	14.0	3.0
Mammography	13.5	2.0	**1.0**	4.0	3.0	6.0	5.0	7.0	9.0	13.5	8.0	12.0	10.0	11.0
musk2	11.0	3.0	2.0	9.0	6.0	5.0	7.0	8.0	4.0	11.0	13.0	11.0	14.0	**1.0**
Optical digits	9.5	2.0	**1.0**	9.5	5.5	5.5	4.0	13.0	7.0	9.5	12.0	9.5	14.0	3.0
Ozone level	13.0	**1.0**	4.0	2.0	5.0	7.0	6.0	8.0	9.0	13.0	11.0	13.0	3.0	10.0
Pen digits	9.5	2.0	**1.0**	9.5	6.0	4.0	5.0	12.0	7.0	9.5	13.0	9.5	14.0	3.0
Phoneme	9.5	6.0	**1.0**	4.0	3.0	2.0	5.0	13.0	12.0	9.5	11.0	7.0	14.0	8.0
Protein homo	9.5	2.0	**1.0**	9.5	3.0	5.0	6.0	14.0	13.0	9.5	7.0	9.5	12.0	4.0
Satimage	14.0	2.0	**1.0**	10.0	7.0	5.5	5.5	11.0	4.0	13.0	8.0	9.0	3.0	12.0
Scene	14.0	**1.0**	3.0	9.0	7.0	8.0	6.0	4.0	10.0	11.0	5.0	2.0	12.0	13.0
Sick euthyroid	9.5	2.0	**1.0**	9.5	7.0	4.0	6.0	13.0	12.0	9.5	3.0	9.5	5.0	14.0
Skin	12.5	**1.0**	6.0	2.0	4.5	3.0	4.5	14.0	9.0	12.5	10.0	7.0	11.0	8.0
Spam	8.5	6.0	2.0	8.5	4.5	3.0	4.5	14.0	13.0	8.5	11.5	8.5	11.5	**1.0**
Thyroid sick	12.5	2.0	**1.0**	6.0	4.0	7.0	5.0	3.0	8.0	12.5	9.0	11.0	14.0	10.0
Us crime	12.5	**1.0**	2.0	6.0	4.0	9.0	7.0	3.0	5.0	12.5	8.0	11.0	10.0	14.0

Table 12 continued

	AdaBoost	AdaCC1	AdaCC2	AdaMEC	AdaMEC-Cal	CGAda	CGAda-Cal	AdaCost	CSB1	CSB2	AdaC1	AdaC2	AdaC3	RareBoost
Webpage	12.5	3.0	**1.0**	2.0	6.0	9.0	7.0	8.0	5.0	12.5	14.0	10.0	11.0	4.0
Wilt	10.0	2.0	**1.0**	6.0	3.0	5.0	4.0	12.0	7.0	10.0	13.0	10.0	14.0	8.0
Wine quality	14.0	**1.0**	2.0	8.0	3.0	5.0	4.0	7.0	6.0	13.0	10.0	9.0	11.0	12.0
Avg	11.33	**2.33**	2.41	6.91	4.74	5.31	5.20	9.33	8.81	11.19	9.35	9.46	10.80	7.81
Winner	0	7	**14**	1	1	0	0	0	0	0	0	0	0	4

Best methods per dataset are in bold. Last row (winner) indicates on how many datasets a method is ranked first (best balanced accuracy score, higher values are better). Note that in some datasets, the methods have equal scores (tie); therefore, the ranks are in float format

Table 13 Comparative balanced accuracy ranks across the entire set of methods and datasets (smaller values are better) for $T = 100$

	AdaBoost	AdaCC1	AdaCC2	AdaMEC	AdaMEC-Cal	CGAda	CGAda-Cal	AdaCost	CSB1	CSB2	AdaC1	AdaC2	AdaC3	RareBoost
Abalone	14.0	2.0	3.0	9.0	6.0	7.0	8.0	**1.0**	11.0	13.0	4.0	5.0	10.0	12.0
Adult	12.0	**1.0**	2.0	3.0	4.0	6.0	5.0	9.0	14.0	12.0	10.0	12.0	8.0	7.0
Bank	12.5	**1.0**	2.0	3.0	6.0	4.0	5.0	8.0	14.0	12.5	10.0	9.0	7.0	11.0
Car eval	9.5	2.0	**1.0**	9.5	3.0	5.0	4.0	13.0	7.0	9.5	12.0	9.5	14.0	6.0
Coil 2000	13.0	**1.0**	2.0	8.0	7.0	4.0	6.0	9.0	10.0	14.0	3.0	5.0	12.0	11.0
Credit	12.5	3.0	8.0	**1.0**	2.0	4.0	5.0	6.0	11.0	12.5	7.0	10.0	9.0	14.0
Eeg eye	4.5	3.0	10.0	11.0	2.0	8.0	6.0	14.0	13.0	4.5	9.0	7.0	12.0	**1.0**
Electricity	8.0	3.0	6.0	10.0	4.0	2.0	5.0	13.0	14.0	8.0	11.0	8.0	12.0	**1.0**
Isolet	10.5	**1.0**	2.0	10.5	6.0	4.0	5.0	13.0	8.0	10.5	7.0	10.5	14.0	3.0
Letter img	8.5	2.0	**1.0**	8.5	4.0	5.0	6.0	12.0	11.0	8.5	13.5	8.5	13.5	3.0
Mammography	12.0	2.0	**1.0**	4.0	3.0	5.0	6.0	7.0	14.0	12.0	8.0	12.0	10.0	9.0
musk2	9.5	3.0	2.0	9.5	5.5	7.0	5.5	12.0	4.0	9.5	13.0	9.5	14.0	**1.0**
Optical digits	9.5	2.0	**1.0**	9.5	4.0	6.0	5.0	14.0	7.0	9.5	12.0	9.5	13.0	3.0
Ozone level	13.0	**1.0**	2.0	9.0	4.0	7.0	6.0	5.0	8.0	13.0	11.0	13.0	3.0	10.0
Pen digits	8.5	2.0	**1.0**	8.5	6.0	4.0	5.0	14.0	11.0	8.5	12.0	8.5	13.0	3.0
Phoneme	9.5	6.0	**1.0**	5.0	2.0	4.0	3.0	12.0	14.0	9.5	11.0	7.0	13.0	8.0
Protein homo	8.5	2.0	**1.0**	8.5	3.0	5.0	6.0	14.0	12.0	8.5	11.0	8.5	13.0	4.0
Satimage	13.0	2.0	**1.0**	11.0	6.0	4.0	5.0	8.0	10.0	13.0	7.0	13.0	3.0	9.0
Scene	13.5	**1.0**	6.0	3.0	7.0	9.0	8.0	2.0	10.0	13.5	4.0	5.0	12.0	11.0
Sick euthyroid	9.5	2.0	**1.0**	9.5	7.0	5.0	6.0	12.0	14.0	9.5	3.0	9.5	4.0	13.0
Skin	8.5	3.0	4.0	8.5	**1.5**	5.0	**1.5**	13.0	14.0	8.5	11.0	8.5	12.0	6.0
Spam	8.5	4.0	2.0	8.5	5.5	3.0	5.5	14.0	13.0	8.5	11.5	8.5	11.5	**1.0**
Thyroid sick	12.5	2.0	**1.0**	7.0	4.0	6.0	5.0	3.0	9.5	12.5	9.5	11.0	14.0	8.0
Us crime	12.0	**1.0**	3.0	10.0	5.0	7.0	6.0	2.0	4.0	12.0	8.0	12.0	14.0	9.0

Table 13 continued

	AdaBoost	AdaCC1	AdaCC2	AdaMEC	AdaMEC-Cal	CGAda	CGAda-Cal	AdaCost	CSB1	CSB2	AdaC1	AdaC2	AdaC3	RareBoost
Webpage	12.5	2.0	**1.0**	7.0	4.0	6.0	5.0	8.0	9.0	12.5	14.0	10.0	11.0	3.0
Wilt	9.5	2.0	**1.0**	9.5	3.0	5.0	4.0	12.0	7.0	9.5	13.0	9.5	14.0	6.0
Wine quality	13.5	**1.0**	2.0	4.0	3.0	7.0	6.0	5.0	10.0	13.5	9.0	8.0	11.0	12.0
Avg	10.69	**2.11**	2.52	7.61	4.35	5.33	5.31	9.44	10.50	10.69	9.43	9.17	11.00	6.85
Winner	0	8	**12**	1	0.5	0	0.5	1	0	0	0	0	0	4

Best methods per dataset are in bold. Last row (winner) indicates on how many datasets a method is ranked first (best balanced accuracy score, higher values are better). Note that in some datasets, the methods have equal scores (tie); therefore, the ranks are in float format
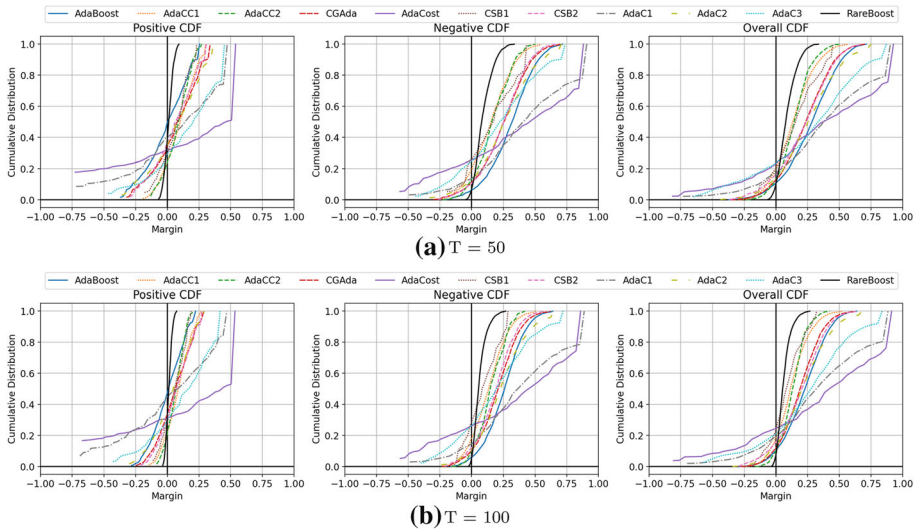
Fig. 7 Effect of boosting rounds on the confidence scores (left:positive class, middle:negative class, right:overall)

References

1. Bradford JP, Kunz C, Kohavi R, Brunk C, Brodley CE(1998) Pruning decision trees with misclassification costs. In: Nedellec C, Rouveirol C (eds) Machine learning: ECML-98, 10th European conference on machine learning, Chemnitz, Germany, April 21–23, 1998, Proceedings, Lecture notes in computer science, vol 1398. Springer, pp 131–136. https://doi.org/10.1007/BFb0026682
2. Brennan P (2012) A comprehensive survey of methods for overcoming the class imbalance problem in fraud detection. Institute of Technology Blanchardstown, Dublin
3. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357. https://doi.org/10.1613/jair.953
4. Chawla NV, Lazarevic A, Hall LO, Bowyer KW (2003) Smoteboost: improving prediction of the minority class in boosting. In: Lavrac N, Gamberger D, Blockeel H, Todorovski L (eds) Knowledge discovery in databases: PKDD 2003, 7th European conference on principles and practice of knowledge discovery in databases, Cavtat-Dubrovnik, Croatia, September 22–26, 2003, Proceedings, Lecture notes in computer science, vol 2838. Springer, pp. 107–119. https://doi.org/10.1007/978-3-540-39804-2_12
5. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
6. Ditzler G, Polikar R (2013) Incremental learning of concept drift from streaming imbalanced data. IEEE Trans Knowl Data Eng 25(10):2283–2301. https://doi.org/10.1109/TKDE.2012.136
7. Domingos PM (1999) Metacost: a general method for making classifiers cost-sensitive. In: Fayyad UM, Chaudhuri S, Madigan D (eds) Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining, San Diego, CA, USA, August 15–18, 1999. ACM, pp 155–164. https://doi.org/10.1145/312129.312220
8. Drummond C, RC Holte et al (2003) C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In: Workshop on learning from imbalanced datasets II, vol 11, pp 1–8
9. Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
10. Esprit (1991) The European strategic programme for research and development in information technology. In: Speech and natural language, proceedings of a workshop held at Pacific Grove, California, USA, February 19–22. Morgan Kaufmann. https://www.aclweb.org/anthology/H91-1007/
11. Fan W, Stolfo SJ, Zhang J, Chan PK (1999) Adacost: misclassification cost-sensitive boosting. In: Bratko I, Dzeroski S (eds) Proceedings of the sixteenth international conference on machine learning (ICML 1999). Bled, Slovenia, June 27–30. Morgan Kaufmann, , pp 97–105
12. Guo H, Viktor HL (2004) Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. SIGKDD Explor 6(1):30–39
13. Harries M, Wales NS (1999) Splice-2 comparative evaluation: electricity pricing. Citeseer

14. He H, Ma Y (2013) Imbalanced learning: foundations, algorithms, and applications. Wiley
15. Iosifidis V, Fetahu B, Ntoutsi E (2019) FAE: a fairness-aware ensemble framework. In: 2019 IEEE international conference on big data (Big Data), Los Angeles, CA, USA, December 9–12, 2019. IEEE, pp 1375–1380. https://doi.org/10.1109/BigData47090.2019.9006487
16. Iosifidis V, Ntoutsi E (2018) Dealing with bias via data augmentation in supervised learning scenarios. Jo Bates Paul D. Clough Robert Jäschke, p 24
17. Iosifidis V, Ntoutsi E (2019) Adafair: cumulative fairness adaptive boosting. In: Zhu W, Tao D, Cheng X, Cui P, Rundensteiner EA, Carmel D, He Q, Yu JX (eds) Proceedings of the 28th ACM international conference on information and knowledge management, CIKM 2019, Beijing, China, November 3–7, 2019. ACM, pp 781–790. https://doi.org/10.1145/3357384.3357974
18. Iosifidis V, Ntoutsi E (2020) FABBOO-online fairness-aware learning under class imbalance. In: Appice A, Tsoumakas G, Manolopoulos Y, Matwin S (eds) Discovery science—23rd international conference, DS 2020, Thessaloniki, Greece, October 19–21, 2020, Proceedings, Lecture notes in computer science, vol 12323. Springer, pp 159–174. https://doi.org/10.1007/978-3-030-61527-7_11
19. Iosifidis V, Ntoutsi E (2020) Sentiment analysis on big sparse data streams with limited labels. Knowl Inf Syst 62(4):1393–1432. https://doi.org/10.1007/s10115-019-01392-9
20. Iosifidis V, Roy A, Ntoutsi E(2022) Parity-based cumulative fairness-aware boosting. arXiv preprint arXiv:2201.01148
21. Iosifidis V, Zhang W, Ntoutsi E (2021) Online fairness-aware learning with imbalanced data streams. arXiv preprint arXiv:2108.06231
22. Japkowicz N, Stephen S (2002) The class imbalance problem: a systematic study. Intell Data Anal 6(5):429–449
23. Joshi MV, Kumar V, Agarwal RC (2001) Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In: Cercone N, Lin TY, Wu X (eds) Proceedings of the 2001 IEEE international conference on data mining, 29 November–2 December 2001, San Jose, CA, USA. IEEE Computer Society, pp 257–264. https://doi.org/10.1109/ICDM.2001.989527
24. Krasanakis E, Xioufis ES, Papadopoulos S, Kompatsiaris Y (2017) Tunable plug-in rules with reduced posterior certainty loss in imbalanced datasets. In: First international workshop on learning with imbalanced domains: theory and applications, LIDTA@PKDD/ECML 2017, 22 September 2017, Skopje, Macedonia. , Proceedings of machine learning research, vol 74. PMLR, pp 116–128 (2017). http://proceedings.mlr.press/v74/krasanakis17a.html
25. Landesa-Vazquez I, Alba-Castro JL (2012) Shedding light on the asymmetric learning capability of adaboost. Pattern Recognit Lett 33(3):247–255
26. Landesa-Vazquez I, Alba-Castro JL (2015) Revisiting adaboost for cost-sensitive classification. Part I: Theoretical perspective. arXiv preprint arXiv:1507.04125
27. Landesa-Vazquez I, Alba-Castro JL (2015) Revisiting adaboost for cost-sensitive classification. part ii: Empirical analysis. arXiv preprint arXiv:1507.04126
28. Laza R, Pavón R, Reboiro-Jato M, Fdez-Riverola F (2011) Evaluating the effect of unbalanced data in biomedical document classification. J Integr Bioinform. https://doi.org/10.2390/biecoll-jib-2011-177
29. Li Y, Guo H, Zhang Q, Mingyun G, Yang J (2018) Imbalanced text sentiment classification using universal and domain-specific knowledge. Knowl Based Syst 160:1–15. https://doi.org/10.1016/j.knosys.2018.06.019
30. Ling CX, Li C (1998) Data mining for direct marketing: problems and solutions. In: Agrawal R, Stolorz PE, Piatetsky-Shapiro G (eds) Proceedings of the fourth international conference on knowledge discovery and data mining (KDD-98), New York City, New York, USA, August 27–31, 1998. AAAI Press, pp 73–79. http://www.aaai.org/Library/KDD/1998/kdd98-011.php
31. Ling CX, Yang Q, Wang J, Zhang S (2004) Decision trees with minimal costs. In: Brodley CE (ed) Machine learning, proceedings of the twenty-first international conference (ICML 2004), Banff, Alberta, Canada, July 4–8, 2004, ACM international conference proceeding series, vol 69. ACM. https://doi.org/10.1145/1015330.1015369
32. Martino MD, Decia F, Molinelli J, Fernández A (2012) Improving electric fraud detection using class imbalance strategies. In: Carmona PL, Sánchez JS, Fred ALN (eds) ICPRAM 2012—proceedings of the 1st international conference on pattern recognition applications and methods, vol 2, Vilamoura, Algarve, Portugal, 6–8 February, 2012. SciTePress, pp 135–141
33. Mayr A, Binder H, Gefeller O, Schmid M (2014) The evolution of boosting algorithms-from machine learning to statistical modelling. arXiv preprint arXiv:1403.1452
34. Nikolaou N, Brown G (2015) Calibrating adaboost for asymmetric learning. In: International workshop on multiple classifier systems. Springer, pp 112–124
35. Nikolaou N, Edakunni NU, Kull M, Flach PA, Brown G (2016) Cost-sensitive boosting algorithms: do we really need them? Mach Learn 104(2–3):359–384. https://doi.org/10.1007/s10994-016-5572-x

36. Phua C, Alahakoon D, Lee VCS (2004) Minority report in fraud detection: classification of skewed data. SIGKDD Explor 6(1):50–59. https://doi.org/10.1145/1007730.1007738

37. Platt J et al (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Adv Large Margin Class 10(3):61–74

38. Qin Z, Wang AT, Zhang C, Zhang S (2013) Cost-sensitive classification with k-nearest neighbors. In: Wang M (ed) Knowledge science, engineering and management—6th international conference, KSEM 2013, Dalian, China, August 10–12, 2013. Proceedings, lecture notes in computer science, vol 8041. Springer, pp 112–131. https://doi.org/10.1007/978-3-642-39787-5_10

39. Rahman MM, Davis DN (2013) Addressing the class imbalance problem in medical datasets. Int J Mach Learn Comput 3(2):224

40. Roy A, Iosifidis V, Ntoutsi E (2021) Multi-fair pareto boosting. arXiv preprint arXiv:2104.13312

41. Sadgali I, Sael N, Benabbou F (2020) Adaptive model for credit card fraud detection. Int J Interact Mob Technol 14(3):54–65 (https://www.online-journals.org/index.php/i-jim/article/view/11763)

42. Schapire RE (1999) A brief introduction to boosting. In: Dean T (ed) Proceedings of the sixteenth international joint conference on artificial intelligence, IJCAI 99, Stockholm, Sweden, July 31–August 6, 1999. 2 Volumes, 1450 pages. Morgan Kaufmann, pp 1401–1406. http://ijcai.org/Proceedings/99-2/Papers/103.pdf

43. Schapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Mach Learn 37(3):297–336. https://doi.org/10.1023/A:1007614523901

44. Seiffert C, Khoshgoftaar TM, Hulse JV, Napolitano A (2010) Rusboost: a hybrid approach to alleviating class imbalance. IEEE Trans Syst Man Cybern Part A 40(1):185–197. https://doi.org/10.1109/TSMCA.2009.2029559

45. Su J, Zhang H (2006) A fast decision tree learning algorithm. In: Proceedings, the twenty-first national conference on artificial intelligence and the eighteenth innovative applications of artificial intelligence conference, July 16–20, 2006, Boston, MA, USA. AAAI Press, pp 500–505. http://www.aaai.org/Library/AAAI/2006/aaai06-080.php

46. Sun Y, Kamel MS, Wong AKC, Wang Y (2007) Cost-sensitive boosting for classification of imbalanced data. Pattern Recognit 40(12):3358–3378. https://doi.org/10.1016/j.patcog.2007.04.009

47. Ting KM (2000) A comparative study of cost-sensitive boosting algorithms. In: Langley P (ed) Proceedings of the seventeenth international conference on machine learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29–July 2, 2000. Morgan Kaufmann, pp 983–990

48. Wang B, Pineau J (2016) Online bagging and boosting for imbalanced data streams. IEEE Trans Knowl Data Eng 28(12):3353–3366. https://doi.org/10.1109/TKDE.2016.2609424

49. Wu F, Jing X, Shan S, Zuo W, Yang J (2017) Multiset feature learning for highly imbalanced data classification. In: Singh SP, Markovitch S (eds) Proceedings of the thirty-first AAAI conference on artificial intelligence, February 4–9, 2017, San Francisco, CA, USA. AAAI Press, pp 1583–1589. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14570

50. Wu G, Chang EY (2003) Class-boundary alignment for imbalanced dataset learning. In: ICML work on learning from imbalanced data sets II, pp 49–56

51. Yin J, Gan C, Zhao K, Lin X, Quan Z, Wang Z (2020) A novel model for imbalanced data classification. In: The thirty-fourth AAAI conference on artificial intelligence, AAAI 2020, the thirty-second innovative applications of artificial intelligence conference, IAAI 2020, the tenth AAAI symposium on educational advances in artificial intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020. AAAI Press, pp 6680–6687. https://aaai.org/ojs/index.php/AAAI/article/view/6145

52. Yin QY, Zhang JS, Zhang CX, Liu SC (2013) An empirical study on the performance of cost-sensitive boosting algorithms with different levels of class imbalance. Math Probl Eng

53. Zadrozny B, Elkan C (2001) Learning and making decisions when costs and probabilities are both unknown. In: Lee D, Schkolnick M, Provost FJ, Srikant R (eds) Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26–29, 2001. ACM, pp 204–213. http://portal.acm.org/citation.cfm?id=502512.502540

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Vasileios Iosifidis works as a data scientist in the industry. He received his Diploma (2014) and M.Sc. (2016) in Computer Science from Department of Computer Engineering and Informatics, Patras, Greece. He obtained his Ph.D. from Leibniz University, Hanover, Germany (2020). His current research interests include data mining, supervised learning, fairness-aware learning, class-imbalance, deep learning, sentiment analysis and adaptive learning.

Symeon Papadopoulos is a Senior Researcher with the Information Technologies Institute (ITI), Centre for Research and Technology Hellas (CERTH), Thessaloniki, Greece. He holds an electrical and computer engineering diploma from the Aristotle University of Thessaloniki, a Professional Doctorate in Engineering from the Technical University of Eindhoven, a Master's in Business Administration from the Blekinge Institute of Technology and a Ph.D. in Computer Science from the Aristotle University of Thessaloniki. His research interests lie at the intersection of multimedia understanding, social network analysis, information retrieval, big data management and artificial intelligence. Dr. Papadopoulos has co-authored more than 40 papers in refereed journals, 10 book chapters and 130 papers in international conferences, 3 patents, and has edited two books. He has participated in and coordinates a number of relevant EC FP7, H2020 and Horizon Europe projects in the areas of media convergence, social media and artificial intelligence. He is leading the Media Analysis, Verification and Retrieval Group (MeVer, https://mever.iti.gr), and is a co-founder of the Infalia Private Company, a spinout of CERTH-ITI.

Bodo Rosenhahn studied Computer Science (minor subject Medicine) at the University of Kiel. He received the Dipl.-Inf. and Dr.-Ing. degrees from the University of Kiel in 1999 and 2003, respectively. From 10/2003 till 10/2005, he worked as post doc at the University of Auckland (New Zealand), funded with a scholarship from the German Research Foundation (DFG). In 11/2005–08/2008 he worked as senior researcher at the Max-Planck Insitute for Informatics in Saarbruecken. Since 09/2008 he is Full Professor at the Leibniz University Hannover, heading a group on automated image interpretation.

Eirini Ntoutsi is a full professor at the Bundeswehr University Munich and Research Institute Cyber Defence and Smart Data (CODE), since August 2022. Before her appointment in Munich, she was a full professor at the Freie University Berlin, an associate professor at the Leibniz University Hannover, and a postdoctoral researcher at the Ludwig Maximilian University of Munich. She obtained her Ph.D. (2008) from the University of Piraeus, Athens and she holds an MSc (2003) and a Diploma (2001) from the Department of Computer Engineering and Informatics, Patras, Greece. Her research interests are within the broad field of Artificial Intelligence and Machine Learning where she develops intelligent algorithms for adaptive learning, responsible AI, and generative AI.