

Example-dependent cost-sensitive adaptive boosting

Yuri Zelenkov

National Research University Higher School of Economics, 20 Myasnikskaya, Moscow101000, Russian Federation



ARTICLE INFO

Article history:

Received 13 August 2018
Revised 17 May 2019
Accepted 5 June 2019
Available online 6 June 2019

Keywords:

Cost-sensitive learning
Example-dependent cost-sensitive classifier
Classifier calibration
Adaptive boosting
AdaBoost
Fraud detection

ABSTRACT

Intelligent computer systems aim to help humans in making decisions. Many practical decision-making problems are classification problems in their nature, but standard classification algorithms often not applicable since they assume balanced distribution of classes and constant misclassification costs. From this point of view, algorithms that consider the cost of decisions are essential since they are more consistent with the requirements of real life. These algorithms generate decisions that directly optimize parameters valuable for business, for example, the costs savings. But despite on practical value of cost-sensitive algorithms, the little number of works study this problem concentrating mainly on the case when the cost of a classifier error is constant and does not depend on a specific example. However, many real-world classification tasks are example-dependent cost-sensitive (ECS), where the costs of misclassification vary between examples and not only within classes. Existing methods of ECS learning include just modifications of the simplest models of machine learning (naive Bayes, logistic regression, decision tree). These models produce promising results, but there is a need for further improvement in performance that can be achieved by using gradient-based ensemble methods. To break this gap, we present the ECS generalization of AdaBoost. We study three models which differ by the ways to introduce cost into the loss function: inside the exponent, outside the exponent, and both inside and outside the exponent. The results of the experiments on three synthetic and two real datasets (bank marketing and insurance fraud) show that example-dependent cost-sensitive modifications of AdaBoost outperform other known models. Empirical results also show that critical factors influencing the choice of the model are not only the distribution of features, which is typical for cost-insensitive and class-dependent cost-sensitive problems but also the distribution of costs. Next, since the outputs of AdaBoost are not well calibrated posterior probabilities, we check three approaches to calibration of classifier scores: Platt scaling, isotonic regression, and ROC modification. The results show that calibration not only significantly improves the performance of specific ECS models but allows making better capabilities of original AdaBoost. Obtained results provide new insight regarding the behavior of the cost-sensitive model from a theoretical point of view and prove that the presented approach can significantly improve the practical design of intelligent systems.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

In practice, decision making often comes down to the problem of classification. The responsible person must determine to what known class the particular object belongs, and the assigned class label defines possible scenarios of actions. For example, the loan manager first analyzes the data of the borrower to determine the level of risk (for example, low, medium or high) and then selects the terms of the loan agreement based of risk level assigned. Similar tasks arise in all other areas of human activity.

Since the number of factors influencing a decision can be huge and their relationships are complicated, computer methods are widely used to solve the classification problem. But practition-

ers often face problems that cannot be solved by standard algorithms, since these methods assume a balanced class distribution and equal misclassification costs (He & Garcia, 2009). In real life, the most typical situation is when the number of examples of one class is much smaller (10 or more times) than the number of instances of another. Moreover, the minor class includes those objects whose identification is of particular interest (Liu & Zhou, 2006; Zadrozny & Elkan, 2001a): insurance fraudsters (Abdallah, Maarof, & Zainal, 2016), dishonest borrowers (Abellan & Castellano, 2017), fraudulent credit card transactions (Sahin, Bulkan, & Duman, 2013), patients with a specific diagnosis (Sun, Kamel, Wong, & Wang, 2007), etc. Besides, it is evident that potential financial losses depend on the type of classifier error. Approval of a loan to a fraudster leads to higher losses than the denial to a bona fide borrower.

E-mail address: yzelenkov@hse.ru

There are three main categories of approaches to imbalanced data sets classification: data processing level methods, classifier level methods, and cost-sensitive methods. The data processing level methods adjust the ratio of two classes to form balanced data sets by re-sampling strategies. Elkan (2001) argue that these strategies have little effect on the classifiers generated by the standard Bayesian methods and decision trees. He and Garcia (2009) review these methods, including Random Oversampling (extension of the dataset by repeating examples of the minor class or generating of objects similar to the instances of the minor class), and Random Undersampling (reducing the major class objects). Classifier level methods deal with the setting of classes weights as meta-parameters of the classifier.

While sampling methods attempt to balance distributions by considering the representative proportions of classes, cost-sensitive learning methods consider the costs associated with classifier decisions (Elkan, 2001; He & Garcia, 2009). The main idea is not to minimize the number of errors but to reduce the total cost of errors. For practitioners, taking into account the costs of classifier decisions is an additional advantage, since it allows moving from unclear for them quality characteristics (F-score, ROC AUC ...) to estimates expressed in business terms, for example, system cost, cost reduction, etc.

Many researchers consider the cost-sensitive learning problem, and most often they believe that the cost of a classifier error does not depend on a specific example and is constant for each type of error (false negatives and false positives). This approach can be called a Class-dependent Cost-Sensitive (CCS) classification problem (Elkan, 2001). Many well-known classification algorithms are adapted for this case, e.g., decision trees (Sahin et al., 2013), their ensembles (Krawczyk, Woźniak, & Schaefer, 2014), in particular AdaBoost (Nikolaou, Edakunni, Kull, Flach, & Brown, 2016) and gradient boosting (Xia, Liu, & Liu, 2017), support vector machines (Park, Luo, Parhi, & Netoff, 2011), neural networks (Zhou & Liu, 2006) etc.

But many real-world classification problems are Example dependent Cost-Sensitive (ECS) since the misclassification cost varies between examples and not only within classes (Bahnsen, Aouada, & Ottersten, 2014a). For example, potential losses in the case of a loan to a fraudster depend on the amount requested.

Lenarcik & Piasta first formulated the ECS problem in 1998 (Lenarcik & Piasta, 1998), but so far there is a relatively small amount of studies developing algorithms to solve it. Zadrozny and Elkan (2001b) adapted MetaCost (Domingos, 1999) algorithm to ECS problem, Brefeld, Geibel, and Wyszotzki (2003) proposed cost-sensitive SVM for non-separable classes, there are also example-dependent cost-sensitive implementations of logistic regression (Bahnsen et al., 2014a), decision tree (Bahnsen, Aouada, & Ottersten, 2015a) and ensembles of random trees (Bahnsen, Aouada, & Ottersten, 2015b). These studies prove the significance of ECS methods for practice but consider the only simplest base models. The goal of our work is to expand the list of available algorithms by generalizing Adaptive Boosting (AdaBoost) for example-dependent cost-sensitive problem.

AdaBoost outperforms other algorithms because it trains an ensemble of weak classifiers moving in the direction of the negative gradient of the loss function. Therefore, each new estimator attempts to correct the error of its predecessors. We consider three methods, which differ in the way of including the cost of the classifier error in the loss function. Since the outputs from AdaBoost are not well calibrated posterior probabilities (Niculescu-Mizil & Caruana, 2005), we also consider three methods of probability calibration based on Platt scaling (Platt, 1999), isotonic regression (Zadrozny & Elkan, 2001a) and ROC modification (Hernández-Orallo, Flach, & Ferri, 2012). We tested proposed ECS AdaBoost on three synthetic data sets, as well as on two real problems of bank

Table 1
Cost matrix.

	Actual positive	Actual negative
Predict positive	C_{TP}	C_{FP}
Predict negative	C_{FN}	C_{TN}

marketing and auto insurance and compared the results with state-of-the-art methods based on decision trees (Bahnsen et al., 2015a; 2015b) and with original AdaBoost with calibration (Nikolaou et al., 2016). Results of experiments prove that proposed generalization of AdaBoost outperform existing algorithms.

The contribution of our work can be summarized as follows. In the theoretical part, in addition to an in-depth review of cost-sensitive classification problem, we concentrate on two aspects. The first one is how the incorporation of cost into the loss function impacts the preservation of class asymmetry; the second one is the problem of probability calibration since original AdaBoost is trying to fit a logit of the true probabilities instead of true probabilities themselves. In the practical part, we present variants of the example-dependent cost-sensitive AdaBoost designed on base theoretical analysis and show that empirical results are consistent with the theory. It proves that the proposed approach has significant potential to be used in intelligent and expert systems supporting the decision-making process.

2. Related works review

2.1. Cost-sensitive classification problem

We consider the problem of binary classification, for the clear presentation, we establish the notation that will be used below. Considering a given training data set S with N examples (i.e., $|S|=N$), we define $S = \{(\mathbf{x}_i, y_i)\}$, $i = 1, \dots, N$, where $\mathbf{x}_i \in X$ is an instance in k -dimensional feature space $X = \{x_1, x_2, \dots, x_k\}$, and $y_i \in Y = \{-1, 1\}$ is a class label associated with instance \mathbf{x}_i . The label $y_i = 1$ corresponds to the minor class.

Usually, C_{FN} and C_{FP} denote the cost of misclassifying a positive and negative example respectively. C_{TP} and C_{TN} are the cost of true positives and true negatives respectively. It leads to a cost matrix presented in Table 1. For CCS problem the elements of cost matrix are constant for all examples. For ECS problem the cost matrix varies between examples. Therefore, in the second case, we have to use augmented instances of the training dataset $\{\mathbf{x}_i, C_{TP_i}, C_{FP_i}, C_{FN_i}, C_{TN_i}\}$.

Conceptually, the cost of mislabeling an example should always be greater than the cost of labeling it correctly, i.e., $C_{FN} > C_{TP}$ and $C_{FP} > C_{TN}$. Elkan (2001) calls it 'reasonableness' condition. Very often it is believed that the cost of correct classification is zero, i.e. $C_{TP} = C_{TN} = 0$, and $C_{FN} > 0$, $C_{FP} > 0$.

2.2. AdaBoost and class dependent cost-sensitive classification

AdaBoost (Freund & Schapire, 1997) is an ensemble learning technique which constructs a strong classifier f_{AB} from a weighted vote of multiple weak classifiers G_m usually implemented as a decision stump

$$f_{AB}(x) = \text{sign} \left[\sum_{m=1}^M \beta_m G_m(x) \right]. \quad (1)$$

Here M is a number of weak classifiers or the number of rounds of learning. It is accomplished through gradient descent with respect to the exponential loss function

$$L_{AB} = \sum_{i=1}^N \exp(-y_i f_{AB}(x_i)). \quad (2)$$

According to Hastie, Tibshirani, and Friedman (2009), on each iteration, one must solve

$$(\beta_m, G_m) = \operatorname{argmin}_{\beta, G} \sum_{i=1}^N w_i^m \exp(-\beta y_i G(x_i)) \quad (3)$$

with $w_i^m = \exp(-y_i f_{m-1}(x_i))$. The solution can be obtained in two steps. First, gradient $G_m(x_i)$ is the classifier that minimizes the weighted error rate in predicting y . Second, optimal step β_m is computed through a line search along direction $G_m(x_i)$, what causes the weights for the next iteration to be $w_i^{m+1} = w_i^m \cdot \exp(-\beta_m y_i G_m(x_i))$. Since a weighted distribution guides each base learner over training examples, that leads it to focus on the mistakes of its predecessors.

Friedman, Hastie, and Tibshirani (2000) prove that the boosting loss L_{AB} is minimized by the symmetric logistic transform ($y = 1|x$) at

$$f_{AB}(x) = \frac{1}{2} \ln \frac{P(y = 1|x)}{P(y = -1|x)}. \quad (4)$$

Eq. (4) follows from the minimum of

$$E[P(y = 1|x)e^{-f_M(x_i)} + P(y = -1|x)e^{f_M(x_i)}]. \quad (5)$$

Nikolaou et al. (2016) note that there are, in general, two strategies to make AdaBoost cost-sensitive. First one related to change the training phase of the algorithm by specifying an alternative lost function to Eq. (2) and deriving what the corresponding weights and their update rule should be. Many authors applied this approach to CCS problem, Nikolaou et al. (2016) present the list of 13 distinct variants of this method (Fan, Stolfo, Zhang, & Chan, 1999; Masnadi-Shirazi & Vasconcelos, 2007; Sun et al., 2007; Ting, 2000; Viola & Jones, 2002; and Landesa-Vázquez & Alba-Castro, 2012). Nikolaou et al. (2016) analyze these approaches using four theoretical frameworks: Bayesian decision theory, the functional gradient descent view, margin theory, and probabilistic modeling. They conclude that only two algorithms from this group satisfy all requirements, namely asymmetric AdaBoost (AsymAda, Viola & Jones, 2002) and cost-generalized AdaBoost (CGAda, Landesa-Vázquez & Alba-Castro, 2012). One of the main drawbacks of other methods is that weights updating rule is not derived from explicitly specified loss function; their authors manually modify the update rule without regard to it.

According to the second strategy, the classifiers set is trained with the original AdaBoost, but decision rule is modified in a cost-respecting manner. It is AdaBoost with Minimum Expected Cost (AdaMEC) rule (Ting, 2000), in the generalized formulation of Nikolaou et al. (2016) the decision rule is the following:

$$f_M(x) = \operatorname{sign}[\hat{p}(y = 1|x) - C(y)]. \quad (6)$$

Here $C(y) = C_{FP}/(C_{FP} + C_{FN})$ is the cost of the decision of CCS problem (here the cost of correct classification is zero, i.e. $C_{TP} = C_{TN} = 0$) and \hat{p} is probability estimation. Eq. (6) reduces to the original AdaBoost decision rule of Eq. (1) when the task is symmetric, i.e., $C_{FP} = C_{FN}$.

Sun et al. (2007) note that there are three ways to introduce cost into the weight update formula of AdaBoost: inside the exponent, outside the exponent, and both inside and outside the exponent. Methods that use these approaches when explicitly specifying the loss function are CSAda (Masnadi-Shirazi & Vasconcelos, 2007) and CGAda (Landesa-Vázquez & Alba-Castro, 2012).

In CSAda Masnadi-Shirazi and Vasconcelos (2007) suggest an alternative asymmetric boosting loss, which included the class dependent cost of misclassification C defined in Table 1

$$L_{CS} = \sum_{i=1}^N \exp(-y_i f_{CS}(x_i)C). \quad (7)$$

If $C_{TP} = C_{TN} = 0$ then this loss function is minimized at

$$\begin{aligned} f_{CS}(x) &= \frac{1}{C_{FP} + C_{FN}} \ln \frac{P(y = 1|x)C_{FN}}{P(y = -1|x)C_{FP}} \\ &= \frac{1}{C_{FP} + C_{FN}} \left(f_{AB}(x) - \frac{1}{2} \ln \frac{C_{FP}}{C_{FN}} \right). \end{aligned} \quad (8)$$

Landesa-Vázquez and Alba-Castro (2012) derived CGAda from the loss function

$$L_{CG} = \sum_{i=1}^N C \cdot \exp(-y_i f_{CG}(x_i)), \quad (9)$$

which has minimum at (Landesa-Vázquez & Alba-Castro, 2015)

$$f_{CG}(x) = \frac{1}{2} \ln \frac{P(y = 1|x)C_{FN}}{P(y = -1|x)C_{FP}} = f_{AB}(x) - \frac{1}{2} \ln \frac{C_{FP}}{C_{FN}}. \quad (10)$$

So, in the last case, the optimal cost-sensitive classifier $f_{CG}(x)$ can be expressed as a threshold on the cost-insensitive optimal predictor $f_{AB}(x)$. For CSAda relation between cost-insensitive and cost-sensitive predictors is more complex (see Eq. (8)).

Another significant difference between these two methods is that CSAda does not preserve the class asymmetry when $y_i f_M(x_i) > 0$ (Landesa-Vázquez & Alba-Castro, 2015; Nikolaou et al., 2016). It is easy to see that if $C_1 > C_2$ then $\exp(-y_i f_M(x_i)C_1) > \exp(-y_i f_M(x_i)C_2)$ when $y_i f_M(x_i) < 0$ and $\exp(-y_i f_M(x_i)C_1) < \exp(-y_i f_M(x_i)C_2)$ otherwise. It can be a critical issue because in practice very often misclassification of positive example can lead to a bigger loss than misclassification of negative example. That is why Nikolaou et al. (2016) excluded CSAda from the list of methods that consistent with the rules of their theoretical perspectives.

Viola and Jones (2002) introduce the ratio of false negatives and false positives cost k in loss function in their AsymAda algorithm:

$$L_{AA} = \sum_{i=1}^N \exp(-y_i f_{CG}(x_i)) \exp(y_i \sqrt{k}).$$

It is equivalent to pre-weighting of each example by $\exp(y_i \sqrt{k})$. However, although this algorithm meets all the requirements of Nikolaou et al. (2016), it cannot be generalized to the case when the cost of correct classification is not zero.

Results discussed above are very important for understanding the cost-sensitive classification. But CCS methods are of limited use since the cost of misclassification almost always varies between the examples in real practical problems.

Particular attention should be paid to violation of class asymmetry in the case when cost introduced inside the exponent (Eq. (7)). This technique of cost incorporation should produce worse results in the case of both the CCS and the ECS tasks.

2.3. Example-dependent cost-sensitive classification

Domingos (1999) described MetaCost algorithm, the central idea of which is to change the label of each training example to be its optimal label, and then to learn a classifier that predicts these new labels. Zadrozny and Elkan (2001b) adapted this approach to the ECS problem. Each example x is associated with a cost $C(i, j, x)$ of predicting class i for x when a true class of x is j . The label to assign to x is the class i that leads to the lowest expected cost

$$\sum_j \hat{p}(y = j|x)C(i, j, x). \quad (11)$$

Applying MetaCost requires knowledge of conditional probability $\hat{p}(j|x)$ for each training example. Almost always, these probabilities are not given as a part of the training dataset. Instead, training data can be used to learn a classifier that estimates $\hat{p}(y = j|x)$ for each training example x and each class j .

Zadrozny and Elkan (2001b) used the single decision tree to estimate these probabilities instead of bagging as in the original paper of Domingos (1999).

In the two-class case, the optimal prediction is class $y = 1$ if and only if the expected cost of this prediction is less than or equal to the expected cost of predicting class $y = -1$ (Elkan, 2001), i.e.

$$\begin{aligned} & \hat{p}(y_i = -1|x_i)C_{FP_i} + \hat{p}(y_i = 1|x_i)C_{TP_i} \\ & \leq \hat{p}(y_i = -1|x_i)C_{TN_i} + \hat{p}(y_i = 1|x_i)C_{FN_i}. \end{aligned} \quad (12)$$

Eq. (12) reduces to Eq. (11) when true positive and true negative costs are zero. Later, the approach on the base of Eq. (12) was used by Bahnsen et al. (2015b) in their Bayes minimum risk model with the combination of different classifiers.

Among other models adapted for the ECS problem, we can mention the support vector machine (Brefeld et al., 2003), logistic regression (Bahnsen et al., 2014a), but the most excellent attention of the researchers is attracted to the decision trees.

There are three main ways to incorporate cost information into decision trees training process (Mac Aodha & Brostow, 2013). The first option is to alter how the data is sampled, e.g., Zadrozny, Langford, and Abe (2003) propose cost-proportionate rejection sampling with aggregation. The second option is to modify the class distribution at each node (Ting, 2002). The last option is to create a new impurity measure that is designed specifically for the example cost-sensitive case (Bahnsen et al., 2015a; Mac Aodha & Brostow, 2013).

Bahnsen et al. (2015a) propose an example-dependent cost-sensitive decision trees (CSDT) algorithm that considers the example-dependent costs during the training and pruning of a tree. Instead of using traditional splitting criteria such as Gini, entropy or misclassification, the cost of each tree node is calculated, and the gain of using each split evaluated as the decrease in the total cost of the algorithm. After the tree is constructed, it is pruned by using a cost-based pruning criterion.

In their next work (Bahnsen et al., 2015b) these authors propose few ECS ensemble algorithms on the base of CSDT. Their approach consists of creating different example-dependent CSDT's on random subsamples of the training set and then combining them using various methods (bagging, random forest, random patches, and pasting). They also propose two new cost-sensitive combination approaches: cost-sensitive weighted voting and cost-sensitive stacking. According to their results, the voting ensemble of cost-sensitive random patches (CSRPP) demonstrates the best results on five practical datasets.

Reviewed works made a significant contribution to the ECS problem, but they consider the generalization of basic models of machine learning and most straightforward methods of their combination, namely averaging and voting. Ensembles based on gradient descent show more accuracy in cost-insensitive problems so that we can expect similar improvement in ECS tasks. From this point of view, the most obvious candidate to the generalization is AdaBoost; details will be present in Section 3.

2.4. Probability calibration

Some approaches (e.g., presented by Eqs. (6), (11) and (12)) to the cost-sensitive classification suggest a modification of the results of the model based on posterior probabilities. However, the outputs from AdaBoost are not well calibrated posterior probabilities, Niculescu-Mizil and Caruana (2005) empirically show that as the number of steps of boosting increases, the predicted values are pushed away from marginal values and tend to collect on either side of the decision surface. Because boosting can be viewed as an additive logistic regression model (Friedman et al., 2000), a consequence of this is that the predictions made by boosting are trying

to fit a logit of the true probabilities, as opposed to the true probabilities themselves.

The procedure of converting classifier scores to actual probability estimates is called calibration. For the symmetric classification task, Niculescu-Mizil and Caruana (2005) show that once properly calibrated, AdaBoost produced better probability estimates than any other model examined. Authors correct output of AdaBoost using three different approaches. The first approach is to directly apply a logistic correction implied by the framework of Friedman et al. (2000) and consists of getting back the conditional probability from Eq. (4). The second calibration method is Platt scaling (Platt, 1999) that consist of finding parameters A and B for a sigmoid mapping $p(y = 1|f) = 1/(1 + \exp(Af + B))$, such that likelihood of the data is maximized. The final approach is isotonic regression (Zadrozny & Elkan, 2001a). Among the three methods, Platt scaling produces the best probability estimates on small sample sizes, closely followed by isotonic regression.

For the class-sensitive classification problem, Nikolaou and Brown (2015) compared the performance of the original AdaBoost calibrated with Platt scaling to that of few CCS modifications of boosting. According to their results, the performance of calibrated original AdaBoost is on par with that of other models on low-dimensional datasets, but on higher-dimensional datasets, calibrated AdaBoost clearly outperforms all other methods.

Later Nikolaou et al. (2016) suggested that authors of all CCS AdaBoost algorithms analyzed in their paper share a common flaw: they assume that AdaBoost produces well-calibrated probability estimation. So, they tested the performance of three calibrated CCS modifications of AdaBoost that consistent with their theoretical frameworks (namely, AdaMEC, AsymAda and CGAda). Platt scaling was used for calibration. Results show that once calibrated, these tree algorithms perform equivalently, and outperform all others. The final recommendation of Nikolaou et al. (2016) is to use calibrated AdaMEC, i.e., the original AdaBoost algorithm with a shifted decision threshold (Eq. (6)), and calibrated probability estimates.

Another approach to calibration was proposed by Hernández-Orallo et al. (2012), in which calibrated probabilities are extracted after modifying the ROC curve in such a manner that it becomes convex. Bahnsen et al. (2015a, 2015b) apply this calibration to ECS classifiers based on logistic regression and decision tree in combination with decision rule presented by Eq. (12). Their results show that calibration improves the performance of ECS classifiers too.

Note, that in case of solving of ECS problem with the help of AdaBoost we also have to concern calibration of outputs, but Eq. (12) should be used instead of Eq. (6) to estimate the threshold.

2.5. Cost-sensitive model quality measures

Standard quality measures such as misclassification rate or F-score, assume the same cost for the different misclassification errors. But in case of ECS problem, costs of prediction of two classifiers with equal misclassification rate but different numbers of false positives and false negatives are not the same since at least $C_{FP_i} \neq C_{FN_i}$. Therefore, Bahnsen et al. (2015a) propose the measure that considers the actual costs $C_i = [C_{TP_i}, C_{FP_i}, C_{TN_i}, C_{FN_i}]$ of each example i . Here we will use the same approach, modifying it for labels $y_i \in \{-1, 1\}$.

Let Z be a set of N examples as established in Section 2.1, and each example is represented by augmented feature vector $\mathbf{z}_i = [\mathbf{x}_i, C_{TP_i}, C_{FP_i}, C_{FN_i}, C_{TN_i}]$ and $y_i \in \{-1, 1\}$ is a class label associated with instance \mathbf{x}_i . A classifier f which predicts label $f(\mathbf{z}_i)$ for each element i is trained using the set Z . Then the absolute value

of the cost of using f on Z is

$$C(y, f(z)) = \sum_{i=1}^N C(y_i, f(z_i)),$$

$$C(y_i, f(z_i)) = \frac{1}{4} [(1 + y_i)((1 - f(z_i))C_{FN_i} + (1 + f(z_i))C_{TP_i}) + (1 - y_i)((1 + f(z_i))C_{FP_i} + (1 - f(z_i))C_{TN_i})]. \quad (13)$$

However, the total cost does not provide enough information for comparing performance on various problems. In Bahsen et al. (2015a) a savings measure was proposed. They defined the savings of using an algorithm as the cost of the algorithm versus the cost of using no algorithm at all. By evaluating the cost of classifying all examples as the class with the lowest cost $C_{base} = \min \{C(y, -1), C(y, 1)\}$, the cost improvement can be expressed as the cost savings as compared with C_{base}

$$S(y, f(z)) = \frac{C_{base} - C(y, f(z))}{C_{base}}. \quad (14)$$

Here $C(y, -1)$ refers to the cost when classifier predicts all examples of Z as belonging to the class $y = -1$, and similarly, $C(y, 1)$ match the case when the predicted labels of all examples of Z are $y = 1$.

3. Example-dependent cost-sensitive AdaBoost algorithms

According to Sun et al. (2007), there are three ways to introduce cost into the AdaBoost: inside the exponent, outside the exponent, and both inside and outside the exponent; they named these CCS approaches AdaC1, AdaC2, and AdaC3 respectively. In the next sections, we adopt these three approaches to example-driven cost-sensitive AdaBoost (EDAB), these will be EDAB.C1, EDAB.C2 and EDAB.C3 algorithms.

3.1. EDAB.C1 algorithm

Following the approach proposed by Masnadi-Shirazi and Vasconcelos (2007), let us define loss function as $L = \sum_{i=1}^N \exp[-y_i \cdot f(z_i) \cdot C(y_i, f(z_i))]$ where the cost of misclassification of i th example $C(y_i, f(z_i))$ is given by Eq. (13). According to Eq. (1), the basis functions are the individual classifiers $G_m(z) \in \{-1, 1\}$. Using the presented loss function, one must solve

$$(\beta_m, G_m) = \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^N \exp[-y_i (f_{m-1}(z_i) + \beta G(z_i)) \times C(y_i, f_{m-1}(z_i) + \beta G(z_i))]$$

for the classifier G_m and corresponding coefficient β_m to be added at each step. Let

$$C(y_i, f(z_i) + \beta G(z_i)) = C(y_i, f(z_i)) - \beta G(z_i)D(z_i, y_i), \quad (15)$$

$$D(z_i, y_i) = \frac{1}{4} [(1 + y_i)(C_{FN_i} - C_{TP_i}) + (1 - y_i)(C_{TN_i} - C_{FP_i})].$$

For simplicity let $f_m = f_m(z_i)$, $G = G(z_i)$, $C_i(f) = C(y_i, f(z_i))$ and $D_i = D(z_i, y_i)$. So, this can be expressed as

$$(\beta_m, G_m) = \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^N w_i^{(m)} \times \exp[-y_i \beta G(C_i(f_{m-1}) - (f_{m-1} + \beta G)D_i)] \quad (16)$$

with $w_i^{(m)} = \exp(-y_i f_{m-1} C(f_{m-1}))$.

Following standard AdaBoost algorithm, the solution to this equation can be obtained in two steps. First, for any value of $\beta > 0$, the solution of Eq. (16) for $G_m(z)$ is

$$G_m = \underset{G}{\operatorname{argmin}} \sum_{i=1}^N w_i^{(m)} I(y_i \neq G(z_i)),$$

here $I(\cdot) \in [0, 1]$ is indicator function.

Second, β_m is defined from a minimum of function Eq. (16). Unlike the original AdaBoost in the case of example-dependent cost-sensitive classification, it is impossible to obtain an explicit formula for β_m . However, Eq. (16) can be minimized using numerical methods, for example, the Nelder-Mead simplex algorithm. The approximation is then updated

$$f_m = f_{m-1} + \beta_m G_m,$$

which causes the weights for the next iteration to be

$$w_i^{(m+1)} = w_i^{(m)} \exp[-y_i \beta_m G_m(C_i(f_{m-1}) - (f_{m-1} + \beta_m G_m)D_i)] \quad (17)$$

Eq. (17) follows from

$$(\beta_{m+1}, G_{m+1}) = \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^N \exp[-y_i (f_{m-1} + \beta_m G_m + \beta G) \times C_i(f_{m-1} + \beta_m G_m + \beta G)].$$

Note that in EDAB.C1 algorithm we introduce the cost inside the exponent. According to the analysis presented in Section 2.2, this should violate the class asymmetry, and we can expect that this algorithm will produce worse results than other models on the base of AdaBoost.

3.2. EDAB.C2 algorithm

Yet another possible presentation of loss function for ECS problem is $L = \sum_{i=1}^N C(y_i, f(z_i)) \cdot \exp(-y_i \cdot f(z_i))$. This loss function preserves class asymmetry, so we can expect that EDAB.C2 algorithm will produce more accurate predictions than EDAB.C1. A similar approach is used by Landesa-Vázquez and Alba-Castro (2012) in their class-dependent cost-generalized algorithm (CGAda).

Repeating logic presented in Section 3.1, we derive the formula that should be used instead of Eq. (16) for the classifier G_m and corresponding coefficient β_m for step m

$$(\beta_m, G_m) = \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^N w_i^{(m)} \left[1 - \beta G \frac{D_i}{C_i(f_{m-1})} \right] \exp[-y_i \beta G] \quad (18)$$

with weight $w_i^{(m)} = C(f_{m-1}) \cdot \exp(-y_i f_{m-1})$. Weight updating rule in that case is

$$w_i^{(m+1)} = w_i^{(m)} \left[1 - \beta_m G_m \frac{D_i}{C_i(f_{m-1})} \right] \exp[-y_i \beta_m G_m]. \quad (19)$$

3.3. EDAB.C3 algorithm

Third possible loss function is $L = \sum_{i=1}^N C(y_i, f(z_i)) \cdot \exp[-y_i \cdot f(z_i) \cdot C(y_i, f(z_i))]$. In that case, formula that should be used instead of Eq. (16) to find the classifier G_m and corresponding coefficient β_m for step m is

$$(\beta_m, G_m) = \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^N w_i^{(m)} \left[1 - \frac{(f_{m-1} + \beta G)D_i}{C(f_{m-1})} \right] \exp[-y_i \beta G] \quad (20)$$

Corresponding weight update rule is

$$w_i^{(m+1)} = w_i^{(m)} \left[1 - \frac{(f_{m-1} + \beta_m G_m)D_i}{C_i(f_{m-1})} \right] \exp[-y_i \beta_m G_m]. \quad (21)$$

The complete description of the proposed algorithms is presented in Table 2.

Table 2
Example-driven cost-sensitive AdaBoost algorithms.

<ol style="list-style-type: none"> 1. Given augmented training set $(z_i, y_i)_{i=1, \dots, (z_{N_i}, y_{N_i})}$ where $z_i = [x_i, C_{TP_i}, C_{FP_i}, C_{FN_i}, C_{TN_i}]$, x_i is vector of features, C_{xy} is cost of classification, and $y_i \in \{-1, +1\}$ is the class label of example z_i. 2. Initialize the examples weights $w_i = 1/N$, $i = 1, 2, \dots, N$. 3. For $m = 1$ to M: <ol style="list-style-type: none"> a. Fit a classifier $G_m(z)$ to the training data using weights w_i. b. Compute coefficient β_m minimizing function defined by <ul style="list-style-type: none"> • Eq. (16) for EDAB.C1 algorithm; • Eq. (18) for EDAB.C2 algorithm; • Eq. (20) for EDAB.C3 algorithm. c. Update weights according rule defined by <ul style="list-style-type: none"> • Eq. (17) for EDAB.C1 algorithm; • Eq. (19) for EDAB.C2 algorithm; • Eq. (21) for EDAB.C3 algorithm. 4. Output $f_{EDAB}(x) = \text{sign}[\sum_{m=1}^M \beta_m G_m(z)]$.

4. Experiments

In this section, we apply the proposed algorithms to ECS task on the base of three synthetic and two real datasets. First, we evaluate the performance of proposed EDAB algorithms and compare it against the state-of-art-methods namely the cost-sensitive decision tree (CSDT, Bahnsen et al., 2015a), the voting ensemble of cost-sensitive random patches (CSRP, Bahnsen et al., 2015b) and original cost-incentive AdaBoost (AB). We use the stump (i.e., decision tree with max depth 1) as the basic classifier for the algorithms EDAB.C1 - EDAB.C3. Next, we apply three calibration methods to predictions for each classification problem. These calibration methods are Platt (Platt scaling, Platt, 1999), Iso (isotonic regression, Zadrozny & Elkan, 2001a) and ROC (ROC modification, Hernández-Orallo et al., 2012). For a prediction on the base of calibrated classifiers, we use rule defined by Eq. (12).

The CSDT and CSRP algorithms and ROC calibration method were trained using the implementations of *costcla*¹ software library, AB algorithm, Platt and Iso calibrations were trained using the *scikit-learn* library (Pedregosa et al., 2011).

4.1. Synthetic datasets

First, we tested all the algorithms on three synthetic data sets whose properties are presented in Table 3. These unbalanced datasets were created as follows. Twenty thousand examples were generated for each dataset using function `make_circles`, `make_moons` and `make_blobs` from the library *scikit-learn* (Pedregosa et al., 2011). Each dataset includes two features x_1 and x_2 and target variable y . Then, for each instance of the positive class, we have generated a random number n uniformly distributed in $[0, 20]$, and selected examples with $n < 3$. It should ensure that the proportion of positive instances is less than 15%. The sizes of the obtained data sets, the percentage of positive examples as well as a distribution of the target variable are presented in Table 3. The costs of false positives and false negatives errors were calculated for each instance as (index i for simplicity is omitted)

$$C_{FP} = (|x_1| + |x_2|) \cdot \mathbb{N}(0.05, 0.1); C_{FN} = (|x_1| + |x_2|) \cdot \mathbb{N}(0.05, 0.8),$$

here $\mathbb{N}(a, b)$ is a normal distribution, a and b being boundaries of its 95% confidence interval.

Bahnsen et al. (2015a) introduce the classification problem cost characteristic $b_i = (C_{FN_i} - C_{TP_i}) / (C_{FP_i} - C_{TN_i})$ and its mean μ_b and variance σ_b respectively. In the case of cost-insensitive classification problem $\mu_b = 1$, $\sigma_b = 0$. For class-dependent cost-sensitive problems $\mu_b \neq 1$, $\sigma_b = 0$, for example-dependent cost-sensitive problems $\sigma_b \neq 0$. Table 3 lists values of μ_b and σ_b for all synthetic datasets.

Each synthetic problem was divided into training and test datasets in the ratio of 0.67: 0.33. Table 4 presents the results of checking the models on the test data. For every algorithm, Table 4 lists the mean time of training compared with the time of training of original AdaBoost, the value of cost savings computed by Eq. (14) and value of the F1 score. The results of the algorithm that provides the maximum cost savings are highlighted in bold.

As expected, algorithms based on decision trees (CSDT and CSRP) are not able to solve the problem presented by the *Circles* dataset. Also, entirely predictable, that modifications of the EDAB.C2 algorithm (with or without calibration) demonstrate the best performance on synthetic datasets (see discussion of asymmetry preservation in Section 2.2). However, EDAB.C3 does not converge on the *Blobs* dataset, since an overflow occurred while minimizing the functions Eq. (20). It is also worth noting that the cal-

¹ <http://albahnsen.github.io/CostSensitiveClassification/index.html>

Table 3
Summary of synthetic datasets.

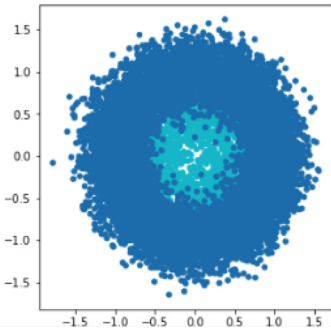
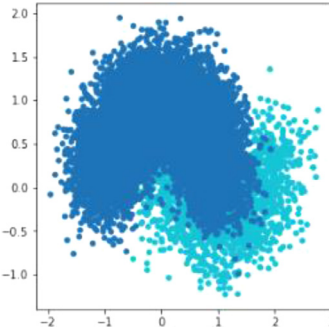
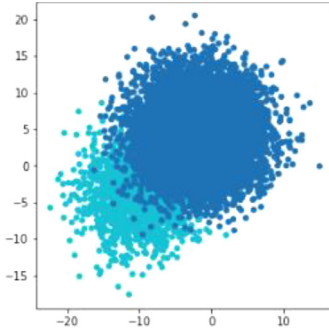
Dataset	Circles	Moons	Blobs
Example distribution			
Examples	11,499	11,464	11,522
% positives	13.04	12.77	13.21
μ_b	10.866	12.933	12.394
σ_b	161.51	253.07	197.32

Table 4
Performance on synthetic datasets.

Algorithm	Circles			Moons			Blobs		
	Time	Savings	F1	Time	Savings	F1	Time	Savings	F1
EDAB.C1	44,5	0,593	0,609	11,5	0,907	0,659	4,7	0,748	0,695
EDAB.C1-ROC		0,586	0,557		0,912	0,611		0,810	0,685
EDAB.C1-Platt		0,558	0,558		0,914	0,604		0,766	0,607
EDAB.C1-Iso		0,580	0,586		0,920	0,649		0,809	0,691
EDAB.C2	44,8	0,557	0,590	23,0	0,925	0,748	20,1	0,868	0,777
EDAB.C2-ROC		0,582	0,554		0,932	0,680		0,863	0,697
EDAB.C2-Platt		0,548	0,557		0,916	0,617		0,859	0,695
EDAB.C2-Iso		0,616	0,619		0,931	0,678		0,847	0,669
EDAB.C3	26,6	0,538	0,612	15,75	0,912	0,726	N/A	N/A	N/A
EDAB.C3-ROC		0,566	0,548		0,929	0,668		N/A	N/A
EDAB.C3-Platt		0,545	0,556		0,920	0,625		N/A	N/A
EDAB.C3-Iso		0,586	0,583		0,928	0,682		N/A	N/A
CSDT	N/A	N/A	N/A	5,0	0,792	0,560	9,4	0,821	0,612
CSDT-ROC		N/A	N/A		0,795	0,497		0,829	0,584
CSDT-Platt		N/A	N/A		0,793	0,487		0,824	0,583
CSDT-Iso		N/A	N/A		0,796	0,514		0,824	0,588
CSRP	N/A	N/A	N/A	40,9	0,620	0,456	34,6	0,768	0,468
CSRP-ROC		N/A	N/A		0,703	0,448		0,743	0,596
CSRP-Platt		N/A	N/A		0,672	0,441		0,728	0,609
CSRP-Iso		N/A	N/A		0,675	0,445		0,712	0,586
AB	1,0	0,436	0,714	1,0	0,784	0,739	1,0	0,769	0,850
AB-ROC		0,582	0,565		0,922	0,669		0,854	0,689
AB-Platt		0,596	0,568		0,922	0,658		0,861	0,690
AB-Iso		-0,712	0,208		0,628	0,459		0,560	0,567

Table 5
Summary of real datasets.

Dataset	Bank marketing	Insurance fraud
Examples	37,931	8400
% positives	12.62	8.79
μ_b	9.306	6.248
σ_b	19.90	1.67
No of features	32	31

ibrated original AdaBoost outperform CSDT and CSRP, this is consistent with the results of Nikolaou et al. (2016) for the class-dependent problem.

4.2. Real datasets

Next, we tested all the algorithms on two real data sets, the description of which is presented in Table 5. The first dataset is *Bank marketing* that is included in the *costcla* library. It contains

data of Portuguese bank clients who received an offer to open a long-term deposit account. The dataset contains features such as age, job, marital status, education, average yearly balance, current loan status and the label indicating whether or not the client accepted the offer (Bahnsen, Stojanovic, Aouada, & Ottersten, 2014b). The peculiarity of this dataset is in the fact that the costs of true positives are greater than zero (i.e., $C_{TP_i} > 0$) since the bank incurs the administrative expenses of contacting the client in that case.

The second dataset is *Insurance fraud*, which contains data of clients of a Russian company that provides car insurance services. This dataset includes features such as the type of policyholder (individual or company), the address of the policyholder and the owner of the car, the type of insurance policy, conditions and coefficients, the model and age of the insured vehicle, the time elapsed from the contract signing to the time of the insured event, the number of payments etc. The target variable indicates whether or not the insurance claim was recognized as an attempt of fraud.

Table 6 lists the same data as Table 4 but for real datasets. In this case, the EDAB.C3 algorithm produces best results, and

Table 6
Performance on real datasets.

Algorithm	Bank marketing			Insurance fraud		
	Time	Savings	F1	Time	Savings	F1
EDAB.C1	27,4	0,298	0,343	2,4	0,184	0,252
EDAB.C1-ROC		0,498	0,298		0,179	0,262
EDAB.C1-Platt		0,486	0,277		0,173	0,262
EDAB.C1-Iso		0,499	0,296		0,170	0,249
EDAB.C2	26,6	0,301	0,345	5,1	0,079	0,267
EDAB.C2-ROC		0,497	0,297		0,141	0,255
EDAB.C2-Platt		0,487	0,277		0,160	0,260
EDAB.C2-Iso		0,495	0,295		0,168	0,247
EDAB.C3	17,1	0,304	0,347	4,0	0,184	0,252
EDAB.C3-ROC		0,500	0,299		0,179	0,262
EDAB.C3-Platt		0,487	0,277		0,172	0,261
EDAB.C3-Iso		0,501	0,296		0,170	0,249
CSDT	5,7	0,471	0,266	178,2	0,162	0,259
CSDT-ROC		0,472	0,267		0,162	0,257
CSDT-Platt		0,471	0,267		0,166	0,257
CSDT-Iso		0,354	0,251		0,164	0,260
CSRP	37,4	0,468	0,258	416,8	0,146	0,256
CSRP-ROC		0,490	0,280		0,149	0,256
CSRP-Platt		0,479	0,274		0,167	0,257
CSRP-Iso		0,495	0,284		0,170	0,260
AB	1,0	-0,354	0,258	1,0	0,014	0,045
AB-ROC		0,498	0,298		0,130	0,281
AB-Platt		0,497	0,294		0,133	0,278
AB-Iso		0,495	0,284		0,170	0,260

the performance of the calibrated original AdaBoost is comparable with the performance of CST and CSRP.

5. Discussion

5.1. Performance of example-dependent cost-sensitive AdaBoost

The first issue that should be discussed is the relative performance of various implementations of EDAB algorithms. As discussed above, loss function defines properties of the classifier, e.g., the introduction of costs inside the exponent violates asymmetry preservation (Section 2.2).

From this point of view, it is interesting to compare the distribution of decision of various EDAB models. Table 4 presents the test data on synthetic datasets. More in-depth insight can be obtained through a confusion matrix. Also, we introduce a new confusion matrix which represents the costs associated with various elements of the original matrix. These data for Moons dataset are presented in Table 7. In each cell, the numerator represents the value of the confusion matrix; denominator corresponds to the associated costs.

As follows from Table 7 no single model provides a better combination of false positive and false negative faults. Decisions of each model are shifted in one direction or another. Moreover, the results in Table 4 show that the better value of Savings (Eq. (14)) very often corresponds to the lower F1 score. It is why traditional metrics of classification quality are not applicable in case of ECS problem.

EDAB.C2 algorithm outperforms the EDAB.C1 for both Savings and F1 scores (see Table 4). As follows from Table 7, this is achieved by shifting the decision boundary toward increasing false negatives values. It significantly reduces the cost of false positives with slightly grows of cost of false negative. Further improvement in the Savings is achieved after calibration EDAB.C2 via ROC modification (Bahnsen et al., 2015b; Hernández-Orallo et al., 2012). Note that ROC modification in this particular case reduces the C_{FP} and C_{FN} simultaneously. At the same time, the number of errors and correct solutions of the algorithm EDAB.C2-ROC takes an intermediate position between the results of EDAB.C1 and EDAB.C2, so, its F1 score is reduced comparing with EDAB.C2.

Among the ECS modifications of AdaBoost, EDAB.C3 is the best to identify negative examples, but at the same time, its capability to detect positive objects is the worst. Thus, this model produces the most significant value of C_{FN} , which can be a problem in practice, since the most significant financial losses are associated with positive objects.

To visualize the cost distribution of incorrectly detected examples, we propose a diagram that can be constructed in the following way. The false negative cost space is discretized into ten bins. For each bin, the prediction error rate and costs associated with the incorrectly classified examples are plotted. Fig. 1 presents this diagram for the Moon dataset. The top chart on Fig. 1 gives the distribution of error rate depending on the cost of false negatives. Other charts show the distribution of actual values of negative, positive and total costs of misclassification for various models. Labels on the horizontal axis correspond to the average value of the C_{FN} in the bins.

We can conclude from Fig. 1 that the errors rate of EDAB.C2 model generally tends to reduce with growing C_{FN} . For the algorithms EDAB.C1 and EDAB.C3, the error rate decreases to the center of the C_{FN} range, but then it starts to grow again. It is a consequence of the asymmetry violation (see Section 2.2) since, in the loss function of both these models, the cost is inside the exponent. Calibration increases the error in the entire C_{FN} range but reduces the total cost of the classifier decisions.

Confusion matrix and associated costs for the Insurance fraud dataset are presented in Table 8. Fig. 2 shows the cost distribution. Unlike the previous example, in this case, the decision boundary in the transition from EDAB.C1 to EDAB.C2 shifts towards increasing the false positives, which leads to an increase in cost. Calibration of EDAB.C2 again helps to receive a model that is mean between EDAB.C1 and EDAB.C2. Models EDAB.C1 and EDAB.C3, in this case, are identical and they produce the best results. Note, that EDAB.C1 and EDAB.C3 show the more or less uniform distribution of error rate over the range of C_{FN} . At the same time, the error rate of EDAB.C2 and EDAB.C2-ROC first decreases, and then again grows in the field of large C_{FN} .

We should conclude that the EDAB models demonstrate the opposite behavior in two considered cases. To explain this let us analyze the cost distribution in both datasets. As follows from Fig. 3, distribution of C_{FN} in the Moons dataset is unimodal, while the distribution of C_{FN} in the Insurance fraud dataset is bimodal. Besides, distributions of costs have very different levels of asymmetry; this impacts on the predictive capabilities of the models. The skew-

Table 7
Confusion matrix with associated costs for Moons dataset (test data 3784 examples).

		Predicted							
		EDAB.C1		EDAB.C2		EDAB.C2-ROC		EDAB.C3	
		Negative	Positive	Negative	Positive	Negative	Positive	Negative	Positive
Actual	Negative	2917 0	400 7789	3098 0	219 4496	2964 0	353 4316	3195 0	122 3778
	Positive	41 3886	426 0	57 4844	410 0	45 4219	422 0	131 7200	336 0

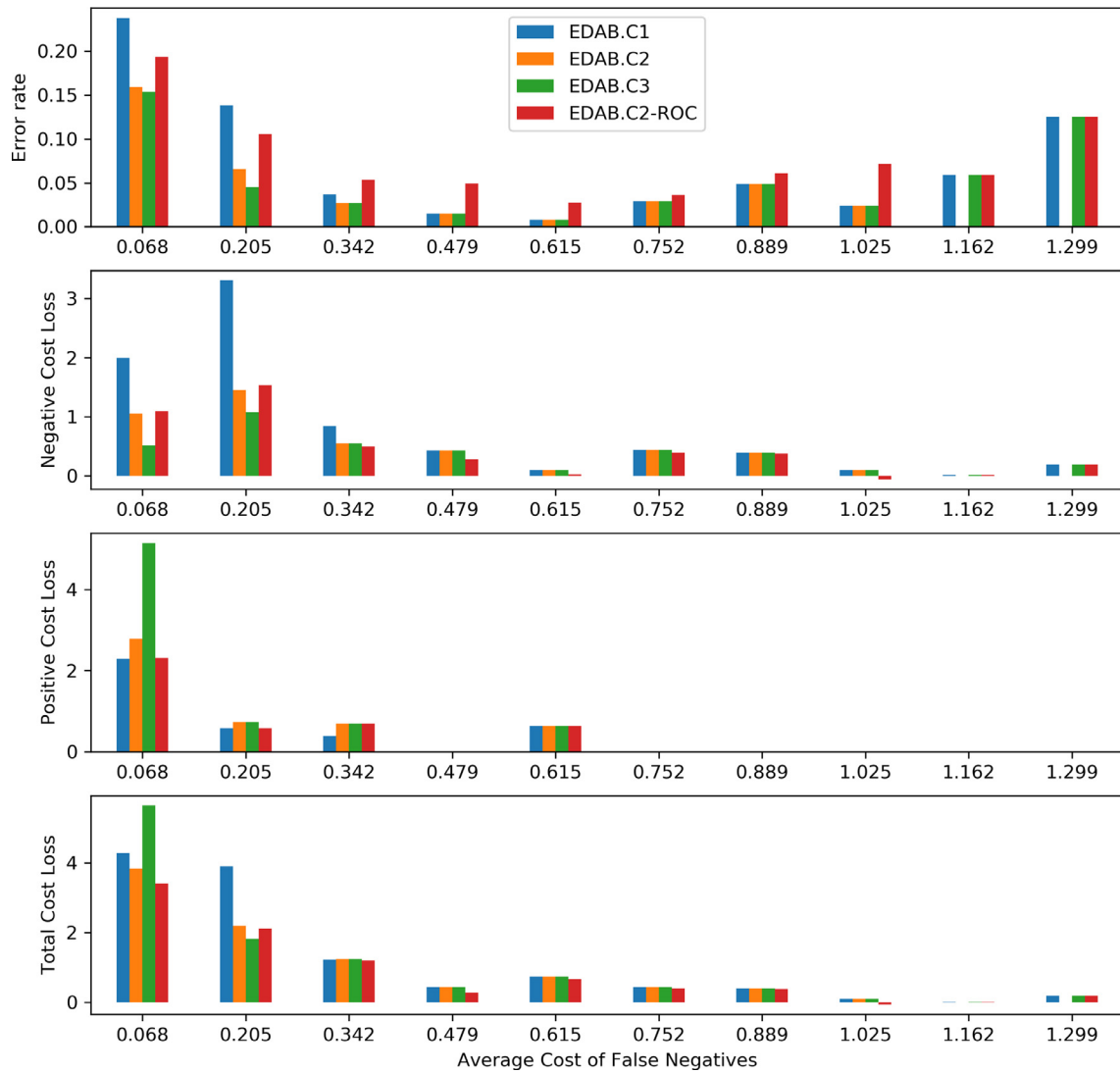


Fig. 1. Distribution of decision costs for *Moons* dataset.

Table 8
Confusion matrix with associated costs for *Insurance fraud* dataset (test data 2770 examples).

		Predicted							
		EDAB.C1		EDAB.C2		EDAB.C2-ROC		EDAB.C3	
		Negative	Positive	Negative	Positive	Negative	Positive	Negative	Positive
Actual	Negative	2356	177	1856	677	2039	494	2356	177
	Positive	0	12155	0	45155	0	30846	0	12155
		65164	60	42086	141	50547	107	65164	60

ness (third standardized moment) of costs in the *Moons* dataset is $S_{C_{FP}} = 1.83$ and $S_{C_{FN}} = 1.28$, in the *Insurance fraud* datasets respectively $S_{C_{FP}} = 1.93$ and $S_{C_{FN}} = 1.77$.

So, we can conclude that models with asymmetry preservation (*EDAB.C2*) show better performance on datasets with unimodal cost distribution. It is consistent with theoretical analysis of Landesa-Vázquez and Alba-Castro (2015) and Nikolaou et al. (2016) for CCS problem. In our experiments, models with the violation of asymmetry (*EDAB.C1* and *EDAB.C3*) showed better performance on datasets with bimodal cost distribution. This fact requires further theoretical analysis and confirmation.

Also, we should note, that in the case of bimodal cost distribution training time of EDAB algorithms significantly better than time of decision tree-based algorithms (see Table 6).

Summing up the performance analysis, we would like to highlight the following. First, AdaBoost-based algorithms produce better results both in terms of accuracy and computational time than existing ECS methods on all dataset tested. It proves the practical implications of the proposed algorithms. Second, as we noted in the theoretical discussion, it is impossible to explicitly represent the ECS classifier as a combination of a cost-insensitive optimal predictor and a threshold, as done for the CCS model (see Eqs. (8) and (10)). However, two proposed presentations, namely confusion matrix with associated costs and graph of the decision cost distribution, help to analyze the shift of the threshold for ECS algorithms. We can see that no single rule that can help to predict the behavior of the threshold for each model. Moreover, the cost

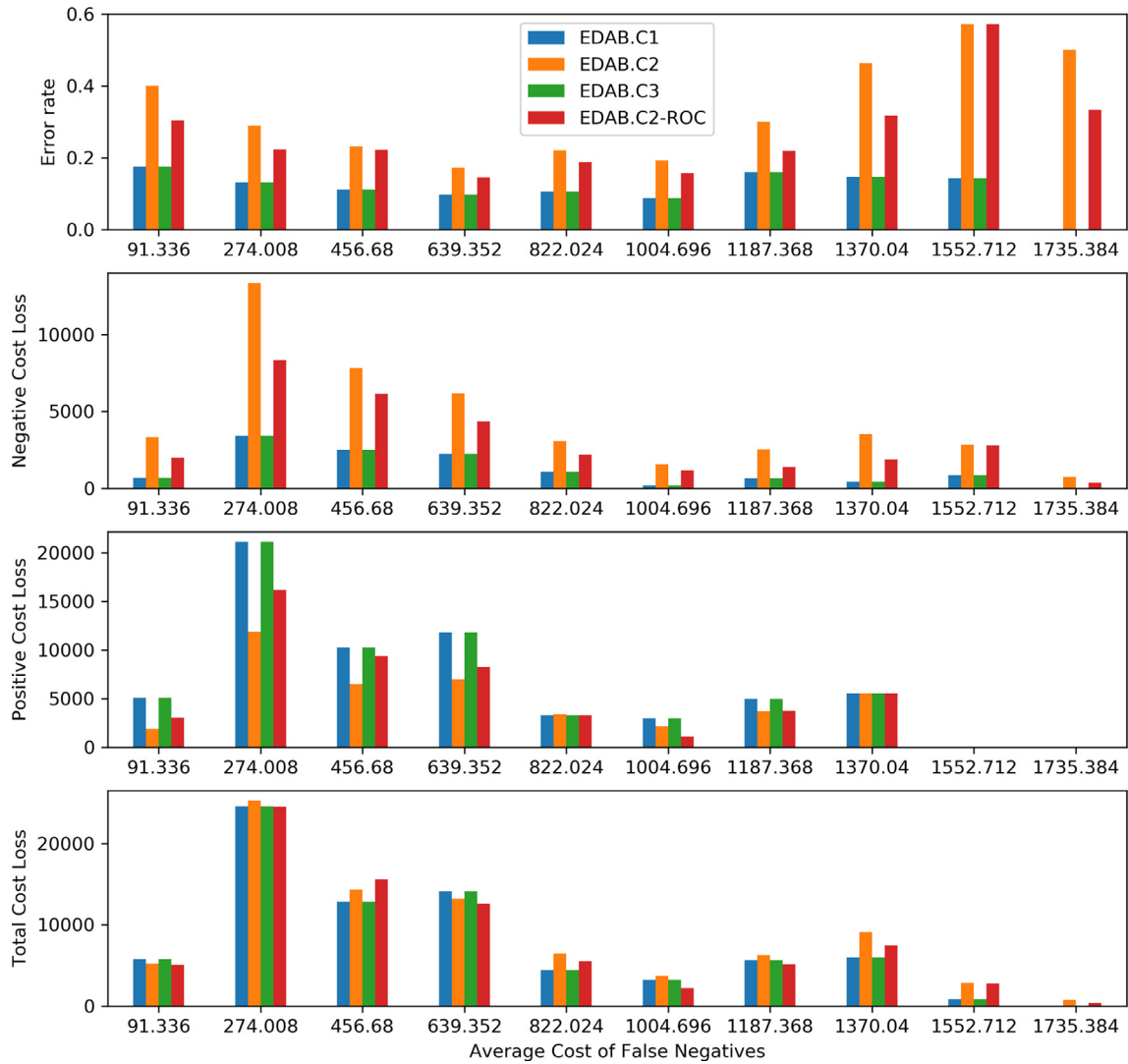


Fig. 2. Distribution of decision costs for Insurance fraud dataset.

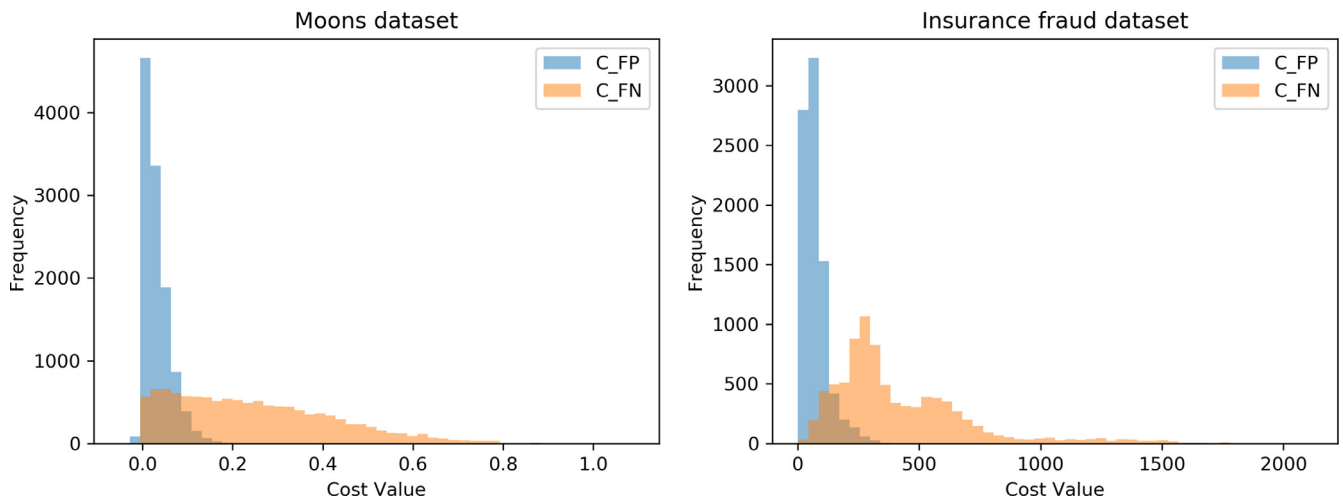


Fig. 3. Cost distribution in Moons and Insurance fraud datasets.

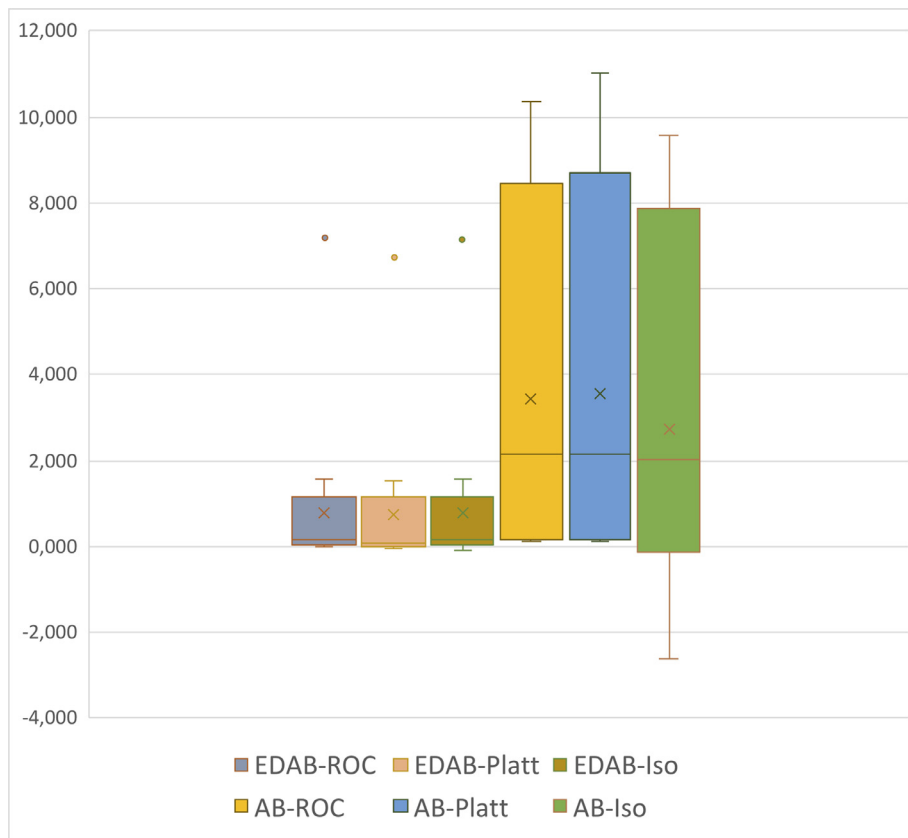


Fig. 4. Relative performance improvement after calibration.

distribution also impacts model performance and should be considered in the system design.

5.2. Impact of calibration

The next issue to be discussed is the impact of calibration on model performance for the ECS problem. We can see from Tables 4 and 6 that in most cases calibration improves performance of all considered models. Fig. 4 presents summary data for all datasets about relative performance improvement for two groups of algorithms, namely the EDAB family introduced here, and original AdaBoost.

The calibration has an especially strong effect on the properties of the original AdaBoost. On synthetic datasets with unimodal cost allocation, calibrated AdaBoost outperforms methods based on decision trees (CSDT and CSRP). On a data set with a non-zero C_{TP} (*Bank marketing*) and with bimodal cost allocation (*Insurance fraud*), it shows performance comparable to these methods. Among the three considered methods, Platt scaling produced the best results for original AdaBoost, closely followed by ROC modification. Isotonic regression in one case (*Circles* dataset) sharply deteriorated the quality, in all other cases, it produces results comparable with other methods. Thus, calibrated original AdaBoost can be used for solving ECS tasks; this is consistent with the results of Nikolaou et al. (2016) obtained for CCS problem.

As for the specialized ECS models (EDAB family, CSDT, CSRP), all calibration methods also increase productivity, but to a lesser extent than for the original AdaBoost. It can be explained by the fact that these models have a sufficiently good initial tuning for the ECS problem and the possibilities to increase their performance are limited. The only exception is the use of calibration to the algorithms EDAB.C1 and EDAB.C3 in the case of *Insurance fraud*, that

led to a worsening of the results. We can assume that this is also caused by the bimodal distribution of costs in the data set.

6. Conclusion and future works

Cost-sensitive learning has prime importance for practice since the optimal decision in real life often depends on costs relating to a specific situation. However, the main research flow that studies this problem concerns mainly on the case when the misclassification costs depend on the class. Known methods of example-dependent cost-sensitive learning include just modifications of the basic models of machine learning and the most straightforward rules of their combinations. From this point of view the theoretical contribution of our work can be summarized as follows:

- We present generalizations of AdaBoost that based on an explicit definition of the loss function and deriving the corresponding formula for weights update;
- We highlight the limitations of the algorithm based on the introduction of costs into the exponent since it violates class asymmetry in training dataset;
- We prove that calibration improves the performance of ECS AdaBoost models; moreover, it radically enhances the performance of original AdaBoost when solving cost-sensitive problems;
- We show that the critical factors influencing the choice of the model are not only the distribution of features, what is typical for cost-insensitive and class-dependent cost-sensitive tasks, but also the distribution of costs. So, no single ECS AdaBoost model produces the best performance on all datasets.

The main practical contribution of presented work is that the example-driven cost-sensitive AdaBoost allows to significantly improve the ECS problem solving comparing with existing methods

among which the cost-sensitive modifications of simple models of machine learning and the original AdaBoost with output calibration. Note that the proposed algorithms are superior to other ECS methods not only in performance but also in computation time. So, these algorithms potentially can be used in a new generation of expert and intelligent systems that will produce more accurate results.

Regarding future research, there are few critical issues to be investigated. First of all, it is necessary to study how the bi-modality (or, more general, multi-modality) of cost distribution impacts the performance of ECS models since this problem is ECS-specific and new for the research community. Here we can identify three possible directions.

First one concerns the example-dependent cost-sensitive learning in general; we need to understand how various models cope with multimodality. Significant efforts in this area should focus on the underlying machine learning models (e.g., decision tree); it also should shed light on the expectable behavior of their homogeneous and heterogeneous combinations.

The second direction should concentrate on AdaBoost-based models proposed here; it is necessary to find out how the multimodality affects the violation of the class asymmetry, i.e., assess the suitability of various loss function options. This direction can develop independently of the previous one because the proposed algorithms use the standard decision tree as the basic classifier. The cost is taken into account by modifying the ensemble generation rule.

In the third direction, it is necessary to investigate as many as possible practice problems to find out how often non-unimodal costs distributions appear in real life. Also, there are virtually no publicly available datasets regarding cost-sensitive problems today. It is one of the reasons that prevent the development of this area of research.

The last but not least research direction can focus on ensembles generated on the base of cost-sensitive underlying models. In the present work, we combined standard decision trees with the help of cost-sensitive ensemble generation rule. There are two other possible variants. First is to produce ensemble integrating cost-sensitive underlying models on base of the standard rules. Secondly, it is possible to combine cost-sensitive loss functions both on the level of basic classifiers as on the level of the ensemble simultaneously. It is a non-explored area that promises exciting results.

Conflicts of interest

There is no conflict of interest.

Credit authorship contribution statement

Yuri Zelenkov: Conceptualization, Formal analysis, Methodology, Software, Validation, Visualization, Writing - original draft.

References

- Abdallah, A., Maarof, M. A., & Zainal, A. (2016). Fraud detection system: A survey. *Journal of Network and Computer Applications*, 68, 90–113.
- Abellán, J., & Castellano, J. G. (2017). A comparative study on base classifiers in ensemble methods for credit scoring. *Expert Systems with Applications*, 73, 1–10.
- Bahnsen, A. C., Aouada, D., & Ottersten, B. (2014a). Example-dependent cost-sensitive logistic regression for credit scoring. In *Proceedings of the thirteenth international conference on machine learning and applications (ICMLA)*, 2014 (pp. 263–269). IEEE.
- Bahnsen, A. C., Stojanovic, A., Aouada, D., & Ottersten, B. (2014b). Improving credit card fraud detection with calibrated probabilities. In *Proceedings of the SIAM international conference on data mining* (pp. 677–685). Society for Industrial and Applied Mathematics.
- Bahnsen, A. C., Aouada, D., & Ottersten, B. (2015a). Example-dependent cost-sensitive decision trees. *Expert Systems with Applications*, 42(19), 6609–6619.
- Bahnsen, A. C., Aouada, D., & Ottersten, B. (2015b). Ensemble of example-dependent cost-sensitive decision trees. arXiv:1505.04637.
- Brefeld, U., Geibel, P., & Wyszotzki, F. (2003). Support vector machines with example dependent costs. In N. Lavrac, D. Gamberger, H. Blockeel, & L. Todorovski (Eds.), *Machine learning: ECML 2003* (pp. 23–34). Springer.
- Domingos, P. (1999). MetaCost: A general method that making classifiers cost sensitive. In *Proceedings of the fifth international conference on knowledge discovery and data mining* (pp. 155–164). ACM Press.
- Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the international joint conference on artificial intelligence*: 17 (pp. 973–978).
- Fan, W., Stolfo, S. J., Zhang, J., & Chan, P. K. (1999). AdaCost: Misclassification cost-sensitive boosting. In *Proceedings of the ICML* (pp. 97–105).
- Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119–139.
- Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2), 337–407.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning* (2nd ed). Springer.
- He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9), 1263–1284.
- Hernández-Orallo, J., Flach, P., & Ferri, C. (2012). A unified view of performance metrics: Translating threshold choice into expected classification loss. *Journal of Machine Learning Research*, 13, 2813–2869.
- Krawczyk, B., Woźniak, M., & Schaefer, G. (2014). Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14, 554–562.
- Landesa-Vázquez, I., & Alba-Castro, J. L. (2012). Shedding light on the asymmetric learning capability of AdaBoost. *Pattern Recognition Letters*, 33(3), 247–255.
- Landesa-Vázquez, I., & Alba-Castro, J. L. (2015). Revisiting AdaBoost for cost-sensitive classification. Part I: Theoretical perspective. arXiv:1507.04125v1.
- Lenarcik, A., & Piasta, Z. (1998). Rough classifiers sensitive to costs varying from object to object. In *Proceedings of the international conference on rough sets and current trends in computing* (pp. 222–230). Berlin, Heidelberg: Springer.
- Liu, X. Y., & Zhou, Z. H. (2006). The influence of class imbalance on cost-sensitive learning: An empirical study. In *Proceedings of the sixth international conference on data mining, 2006 ICDM'06* (pp. 970–974). IEEE.
- Mac Aodha, O., & Brostow, G. J. (2013). Revisiting example dependent cost-sensitive learning with decision trees. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013 (pp. 193–200). IEEE.
- Masnadi-Shirazi, H., & Vasconcelos, N. (2007). Asymmetric boosting. In *Proceedings of the twenty-fourth international conference on machine learning* (pp. 609–619). ACM.
- Niculescu-Mizil, A., & Caruana, R. (2005). Obtaining calibrated probabilities from boosting. In *Proceedings of the UAI* (p. 413).
- Nikolaou, N., & Brown, G. (2015). Calibrating AdaBoost for asymmetric learning. In *Proceedings of the international workshop on multiple classifier systems* (pp. 112–124). Springer.
- Nikolaou, N., Edakunni, N., Kull, M., Flach, P., & Brown, G. (2016). Cost-sensitive boosting algorithms: Do we really need them. *Machine Learning*, 104(2–3), 359–384.
- Park, Y., Luo, L., Parhi, K. K., & Netoff, T. (2011). Seizure prediction with spectral power of EEG using cost-sensitive support vector machines. *Epilepsia*, 52(10), 1761–1770.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12, 2825–2830.
- Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3), 61–74.
- Sahin, Y., Bulkan, S., & Duman, E. (2013). A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15), 5916–5923.
- Sun, Y., Kamel, M. S., Wong, A. K., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12), 3358–3378.
- Ting, K. M. (2000). A comparative study of cost-sensitive boosting algorithms. In *Proceedings of the ICML* (pp. 983–990).
- Ting, K. M. (2002). An instance-weighting method to induce cost-sensitive trees. In *Proceedings of the IEEE transactions on knowledge and data engineering*, 2002: 2 (p. 3).
- Viola, P., & Jones, M. (2002). Fast and robust classification using asymmetric Adaboost and a detector cascade. In *Proceedings of the NIPS*.
- Xia, Y., Liu, C., & Liu, N. (2017). Cost-sensitive boosted tree for loan evaluation in peer-to-peer lending. *Electronic Commerce Research and Applications*, 24, 30–49.
- Zadrozny, B., & Elkan, C. (2001a). Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the ICML: Vol. 1* (pp. 609–616).
- Zadrozny, B., & Elkan, C. (2001b). Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 204–213). ACM.
- Zadrozny, B., Langford, J., & Abe, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the ICDM*, 2003: 2 (p. 5).
- Zhou, Z. H., & Liu, X. Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1), 63–77.