Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-021-00790-4

Cost-sensitive ensemble learning: a unifying framework

George Petrides[1,2] · **Wouter Verbeke**[2]

Received: 15 July 2020 / Accepted: 4 August 2021
© The Author(s) 2021

Abstract
Over the years, a plethora of cost-sensitive methods have been proposed for learning on data when different types of misclassification errors incur different costs. Our contribution is a unifying framework that provides a comprehensive and insightful overview on cost-sensitive ensemble methods, pinpointing their differences and similarities via a fine-grained categorization. Our framework contains natural extensions and generalisations of ideas across methods, be it AdaBoost, Bagging or Random Forest, and as a result not only yields all methods known to date but also some not previously considered.

Keywords Cost-sensitive learning · Class imbalance · Classification · Misclassification cost

1 Introduction

The task of supervised machine learning is given a set of recorded observations and their outcomes to predict the outcome of new observations. Standard classification techniques aim for the highest overall accuracy or, equivalently, for the smallest total error, and include among others support vector machines, Bayesian classifiers, logistic regression, decision tree classifiers such as CART (Breiman et al. 1984) and C4.5 (Quinlan 1993), and ensemble methods which build several classifiers and aggregate

Responsible editor: Grigorios Tsoumakas.

This work was supported by Secur'IT, the platform dedicated to information security launched by Innoviris, the Brussels Region Research funding agency.
The main body of this work was finished while the first author's main affiliation was VUB.

✉ George Petrides
firstname.lastname@uib.no

1 University of Bergen, Bergen, Norway

2 Vrije Universiteit Brussel (VUB), Brussels, Belgium

Published online: 28 September 2021

their predictions such as Bagging (Breiman 1996), AdaBoost (Freund and Schapire 1997) and Random Forests (Breiman 2001).

Of particular interest in certain domains are *binary* classifiers which deal with cases where only two classes of outcomes are considered, such as fraudulent and legitimate credit card transactions, respondents and non-respondents to a marketing campaign, patients with and without cancer, intrusive and authorised network access, and defaulting and repaying debtors to name a few. In most of these cases, one of the classes is a small minority and consequently traditional classifiers might classify all of its members as belonging to the majority class without any significant overall accuracy loss. The severity of this *class imbalance* becomes more noticeable when failing to correctly predict a minority class member is more costly than doing so with a member of the majority class, as the case often is.

A remedy to the undesirable situation just described are classifiers which, instead of accuracy, take misclassification costs into account and are thus termed *cost-sensitive*. We illustrate this idea in the credit card fraud detection framework: accepting a fraudulent transaction as legitimate incurs a cost equal to its amount. Conversely, requiring an additional security check for a transaction (such as contacting the card owner) incurs an overhead cost. The job of a cost-sensitive classifier is to find a cost-minimising balance between overhead costs and fraud costs.

1.1 Related work

An increased research interest in cost-sensitive learning that spanned more than a decade was witnessed in the mid-nineties, including among others (Knoll et al. 1994; Pazzani et al. 1994; Bradford et al. 1998; Ting 1998; Ting and Zheng 1998b; Domingos 1999; Fan et al. 1999; Ting 2000b; Turney 2000; Elkan 2001; Zadrozny and Elkan 2001a; Viola and Jones 2001; Ting 2002; Zadrozny et al. 2003; Ling et al. 2004; Sheng and Ling 2006; Sun et al. 2007). In recent years, a renewed interest is observed (Coussement 2014; Bahnsen et al. 2015; Nikolaou et al. 2016; Xia et al. 2017; Choi et al. 2017; Petrides et al. 2020; Lawrance et al. 2021), partly attributed to practitioners starting to realise the potential of using cost-sensitive models for their businesses by considering real-world monetary costs.

The most recent articles providing an overview of these methods differ in the fraction of the classifier spectrum they cover. Often, cost-sensitive learning is reviewed within surveys on learning on imbalanced datasets as an approach towards treating class imbalance in any domain by artificially introducing costs (He and Garcia 2009; Prati et al. 2009; Sun et al. 2009; Galar et al. 2011). Employing a cost minimisation point of view, Lomax and Vadera (2013) review cost-sensitive classifiers based on decision trees. Overall, cost-sensitive boosting methods receive more attention than other methods such as weighting, altered decisions and cost-sensitive node splitting.

1.2 Our contribution

Our primary contribution in this article is a unifying framework of binary ensemble classifiers that, by design or after slight modification, are cost-sensitive with respect

to misclassification costs. It is presented in terms of combinable components that are either directly extracted from the existing literature or indirectly via natural extensions and generalisations we have identified. A notable example of such an extension are ways in which costs can influence the aggregation of the outputs of the individual models in any ensemble, as done in AdaBoost (Sect. 3.4.3). As such, our work goes one step further than being a mere survey. The advantages of our approach include that

(a) by abstracting the core ideas behind each classifier, we are able to provide generic descriptions that allow for a fine-grained categorisation with respect to the way costs influence the final decision,
(b) it makes the similarities and differences between methods easier to recognise (see for example Table 1 and the equivalence proven in Theorem 1),
(c) it clearly indicates the types of costs (constant or record-dependent) that are applicable for each method,
(d) combining the framework components in all possible ways not only yields all methods known to date, but also some not previously considered (see for example Tables 2 and 3),
(e) the framework components are generic enough to be instantiated with different classifiers, including Random Forests (see for example Table 2), and
(f) it highlights research directions that can lead to new cost-sensitive methods (see Sect. 5).

1.3 Outline

We give a brief introduction to decision tree classifiers, ensemble methods and cost-sensitive learning in Sect. 2, before presenting our framework of cost-sensitive components in Sect. 3. In Sect. 4 we discuss the road towards the state of the art, and we end the paper with our conclusions and directions for future work in Sect. 5.

2 Preliminaries

Most of this section's material is provided with the intention of making the article as self contained as possible. We begin with the basics of decision tree classifiers which play a central role in this work. Readers familiar with these and ensemble methods can proceed to Sect. 2.3 for an introduction to cost-sensitive learning.

A dataset is a collection of records which consist of a number of characteristics, often referred to as *attributes* or *features*. A record's outcome or *class* is what is of importance and needs to be predicted. Classifiers are trained using a set of records together with their known class in order to be able to predict the class of other records for which it is unknown.

In this work our interest lies in the binary class case where there are only two possibilities for the class. In binary imbalanced datasets, it is customary to call records within the minority class *positive* and within the majority class *negative*. Throughout

this article, the class of positive records will be denoted by 1 and that of negative ones by 0.

Distinction is made between the different predictions a classifier makes. *True Positive* (TP) and *False Negative* (FN) denote a positive record correctly classified and misclassified respectively, and *True Negative* (TN) and *False Positive* (FP) are the equivalents for a negative record.

2.1 Decision tree classifiers

Decision tree classifiers are greedy algorithms that try to partition datasets according to their records' outcomes through a series of successive splits. For each split, the attribute that partitions the records the best according to some metric is chosen, and splitting ends after each partition contains records of only one class, or when further splits do not improve the situation.

Starting from the initial set of all records, or the tree's *root node*, splits create *branches* in the tree which are labelled by the attribute used for splitting. Split sets are known as *parent nodes*, sets obtained after a split are called *children nodes*, and sets that are no longer split are called *leaf nodes*.

Usually, the next step after tree growing is *pruning*, done by removing nodes which do not improve accuracy. Pruning is a process starting from the bottom of the tree and going up and serves the purpose of reducing *over-fitting*, that is the effect of the tree's quality of predictions not generalising beyond the dataset used for training.

In the final tree, each leaf node is assigned the class with the highest frequency among its records, and every record reaching the node will be predicted as having that class.

For certain parts of a decision tree algorithm, such as the node splitting step, it is necessary to know the probabilities that a record reaching a node t is positive (P_{t_+}) and negative (P_{t_-}). Since $P_{t_-} = 1 - P_{t_+}$, it in fact suffices to know one of the two. Let N_t, N_t^+ and N_t^- respectively denote the sets of all, the positive, and the negative records at node t (when no subscript is specified we will be referring to the root node of the tree, or the set of all records). Then,

$$P_{t_+} = |N_t^+|/|N_t|.$$

As probabilities based on frequency counts can be unreliable due to high bias and variance, they can be calibrated using methods like *Laplace Smoothing* as suggested by Pazzani et al. (1994) ($P_{t_+} = \left(|N_t^+| + 1\right) / \left(|N_t| + 2\right)$), *m-estimation* (Cestnik 1990; Zadrozny and Elkan 2001a, b) ($P_{t_+} = \left(|N_t^+| + b \cdot m\right) / \left(|N_t| + m\right)$, where $b = |N^+|/|N|$ and $b \cdot m \approx 10$), and *Curtailment* (Zadrozny and Elkan 2001a, b) (each node with less than m records, m as before, gets assigned the probability of its closest ancestor with at least m records) and combinations of the latter with any of the former two.

Examples of decision tree classifiers include the widely used CART (Breiman et al. 1984) and C4.5 (Quinlan 1993), briefly described below.

2.1.1 CART

CART (*Classification and Regression Trees*, Breiman et al. (1984)) uses the Gini index as a splitting measure during tree growing. More specifically, the attribute chosen for splitting is the one that maximises the following value, known as *gain*:

$$1 - P_{t_+}^2 - P_{t_-}^2 - \sum_{i=1}^{k} \frac{|N_{t_i}|}{|N_t|} \left(1 - P_{t_{i+}}^2 - P_{t_{i-}}^2\right), \tag{1}$$

where t_1 to t_k are the children nodes of tree node t.

The pruning method used by CART is *cost complexity pruning*. Given a tree T and $\alpha \in \mathbb{R}$ the aim is to find the subtree of T with the smallest error approximation, which is its error on the training data (the number of wrong predictions over the number of correct ones) plus α times the number of its leaf nodes. Starting from the tree to be pruned and $\alpha = 0$, a finite sequence of subtrees and increasing αs is obtained in this way. The tree chosen as the pruned tree is the one with the smallest error approximation on a separate validation set (a set not used for anything else).

By design, CART can take record weights w as input during training and use them to modify the probabilities used in calculating the gain (1) as

$$P_{w_{t_+}} = W_t^+ / W_t,$$

where W_t^+ and W_t respectively denote the sum of weights of all positive and all records at node t. Moreover, minimum weight is considered instead of minimum error for pruning, and each leaf node is assigned the class with the largest total weight among its records.

Remark 1 It is not clear if and how weighted probabilities can be calibrated.

2.1.2 C4.5

The splitting measure of C4.5 (Quinlan 1993) is an extension of Entropy which is a normalised version known as *gain ratio*:

$$\frac{P_{t_+} \log_2 P_{t_+} + P_{t_-} \log_2 P_{t_-} - \sum_{i=1}^{k} \frac{|N_{t_i}|}{|N_t|} \left(P_{t_{i+}} \log_2 P_{t_{i+}} + P_{t_{i-}} \log_2 P_{t_{i-}}\right)}{\sum_{i=1}^{k} \frac{|N_{t_i}|}{|N_t|} \log_2 \frac{|N_{t_i}|}{|N_t|}},$$

where t_1 to t_k are the children nodes of tree node t. C4.5 employs *reduced error pruning*.

Ting (1998, 2002) showed how to implement the weighted CART design in C4.5.

2.2 Ensemble methods

In ensemble methods, several models are trained and their outcomes combined to give the final outcome, usually as a simple or weighted majority vote, the difference lying

on whether each model's vote weighs the same (as in Bagging and Random Forests) or may weigh differently (as in AdaBoost). Probabilities are usually combined by taking their average.

Some of the most important ensemble methods are briefly described below.

2.2.1 Bagging

The idea of Bagging (Breiman 1996) is to build several models (originally CART models, though in principle there is no restriction) on samples of the data. If the sampled sets are of equal size as the original data, they are called *bootstraps*.

1. Sample with replacement a number of uniformly random and equally sized sets from the training set.
2. For each sampled set, build a model producing outcomes or probabilities.
3. A record's final outcome (respectively probability P_+) is the majority vote on its outcome (respectively the average of its probabilities) from all models.

2.2.2 Random Decision Forests

As originally defined by Ho (1995, 1998), Random Decision Forests differs from Bagging in that it samples subsets of the attribute set instead of the data to build decision trees. Here we will abuse terminology slightly and consider Random Decision Forests as a special case of Bagging that builds *Random Feature Trees*, a name we give to decision trees that are grown on a random subset of the attribute set.

2.2.3 Boosting

Boosting refers to enhancing the predictive power of a "weak" classifier by rerunning it several times, each time focusing more on misclassified records.

AdaBoost (Freund and Schapire 1997) is the most notable example of Boosting in which the focus on each record is in terms of weights: misclassified records after a round get increased weights and correctly classified ones get decreased weights.

1. Assign weight $w = 1$ to each record in the training set.
2. Normalise each record's weight by dividing it by the sum of the weights of all records.
3. Build a model using the weighted records and obtain each record's outcome $h \in \{0, 1\}$ and the model's total error ε as the sum of weights of all misclassified records.
4. Update each record's weight as $w' = w \cdot e^{-\alpha y_* h_*}$, where $\alpha = \frac{1}{2} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$, and h_* and y_* are h and y, the record's true class, mapped from $\{0, 1\}$ to $\{-1, 1\}$.
5. Repeat steps 2 to 4 as required.
6. A record's final outcome is the weighted majority vote on its outcome from all models, the weights being the αs.

A generalised version of AdaBoost with different α was proposed by Shapire and Singer (1999). Niculescu-Mizil and Caruana (2005) investigated methods for obtaining

reliable probabilities from AdaBoost, something that by default it is incapable of doing, through calibrating S, the normalised sum of weighted model votes. These are *Logistic Correction* (Friedman et al. 2000) ($P_{lc} = 1/(e^{-2(2\cdot S-1)} + 1)$, where the name was coined by Niculescu-Mizil and Caruana (2005)), *Platt Scaling* (Platt 1999; Zadrozny and Elkan 2002) ($P_{ps} = 1/(e^{A\cdot S+B} + 1)$, where A and B maximise P_{ps} on a validation set with classes mapped from {0, 1} to $\{(|N^+| + 1)/(|N^+| + 2), 1/(|N^-| + 2)\}$) and *Isotonic Regression* (Robertson et al. 1988; Zadrozny and Elkan 2002) (essentially an application of the PAV algorithm (Ayer et al. 1955): (1) sort training records according to their sum S, (2) initialise each record's probability as 0 if negative and 1 if positive, (3) whenever a record has higher probability than its successor, replace the probability of both by their average and consider them as one record thus forming intervals, and (4) a record's probability is the one of the interval its sum S falls in).

2.2.4 Random Forests

Random Forests (Breiman 2001) is in fact Bagging confined to tree classifiers with *Random Input Selection*, which at each splitting step choose the best attribute out of a small randomly chosen subset of all attributes, and are not pruned.

2.3 Cost-sensitive learning

Cost-sensitive (CS) learning refers to aiming at minimising costs related to the dataset instead of error, typically via these costs influencing the classification process in some way. In this work we only consider misclassification costs, though other types exist, such as the cost of attribute acquisition and obtaining attribute values that are missing.

Traditionally, different costs (or benefits) assigned to each type of classification are given in the form of a *Cost Matrix*:

$$CM = \begin{bmatrix} C_{TP} & C_{FN} \\ C_{FP} & C_{TN} \end{bmatrix}$$

In the sequel, we only consider misclassification costs that are higher than costs of correct classification, and by letting $C'_{TN} = C'_{TP} = 0$, $C'_{FP} = C_{FP} - C_{TN}$ and $C'_{FN} = C_{FN} - C_{TP}$, we can reduce our attention to only misclassification costs, even when the other costs are non-zero (Elkan 2001).

Costs can be either constant for all records of a class, often called *class-dependent*, or vary per record which we will call *record-dependent*. For instance, in credit card fraud detection, false positive costs C_{FP} are equal to overhead costs and can be the same for all transactions, whereas false negative costs C^i_{FN} depend on the individual transactions i and are equal to the corresponding amount.

2.3.1 CS decisions

A cost-insensitive classifier would label a record as positive if $P_+ > P_-$ or equivalently if $P_+ > 0.5$. As explained by Elkan (2001), this decision can be made cost-sensitive

by using the minimum expected cost (MEC) criterion, that is by labelling a record as positive if $C_{FP} \cdot P_- < C_{FN} \cdot P_+$, or equivalently if $P_+ > T_{cs}$, where T_{cs} is the cost-sensitive threshold

$$T_{cs} = \frac{C_{FP}}{C_{FP} + C_{FN}}. \tag{2}$$

Note that $T_{cs} = 0.5$ corresponds to the case of equal misclassification costs, $C_{FN} > C_{FP}$ implies $T_{cs} < 0.5$ and $C_{FN} < C_{FP}$ implies $T_{cs} > 0.5$.

Remark 2 The case of record-dependent costs can be treated by considering a distinct threshold T_{cs}^i per record i, as first observed by Zadrozny and Elkan (2001b).

Thresholding (Sheng and Ling 2006), instead of using the theoretical threshold (2), looks for the best threshold T_{thr} among all probabilities obtained from the training set by computing the total costs for each on a validation set and choosing the one with the lowest.

2.3.2 CS data sampling

To induce decision making using the threshold T_{cs} of (2) instead of 0.5 when the cost ratio is constant, we can *under-sample* the negative training records by only sampling $|N^-| \cdot \frac{C_{FP}}{C_{FN}}$ out of $|N^-|$ (Elkan 2001). Equivalently, we can *over-sample* the positive training records by duplicating existing ones or by synthesising new records to reach a total of $|N^+| \cdot \frac{C_{FN}}{C_{FP}}$ instead of $|N^+|$. Sampling and duplicating can either be random or targeted according to some rule. Naturally, any combination of these techniques that yields a positive-negative ratio equal to

$$r_{cs} = \frac{|N^+|}{|N^-|} \cdot \frac{C_{FN}}{C_{FP}} \tag{3}$$

is possible, and we shall call it *hybrid-sampling*.

Remark 3 If $\frac{C_{FN}}{C_{FP}} > \frac{|N^-|}{|N^+|}$ then sampling turns the positive class into the majority. Under-sampling reduces the size of training data and consequently model training time at the cost of losing potentially useful data. On the other hand, over-sampling makes use of all data but leads to increased training times, and record duplication entails the risk of over-fitting.

One method for synthesising new records, thus avoiding the risk of over-fitting is the *Synthetic Minority Oversampling Technique (SMOTE)* (Chawla et al. 2002), which over-samples positive records by creating new ones that are nearest neighbours (roughly speaking, that have the closest similarity attribute-wise) to existing ones:

1. Choose a positive record and find some (say k) of its nearest neighbours.
2. For each nearest neighbour, find its per attribute distance d_a with the positive record.
3. Create a new positive record with attributes those of the positive record minus a random fraction of d_a.

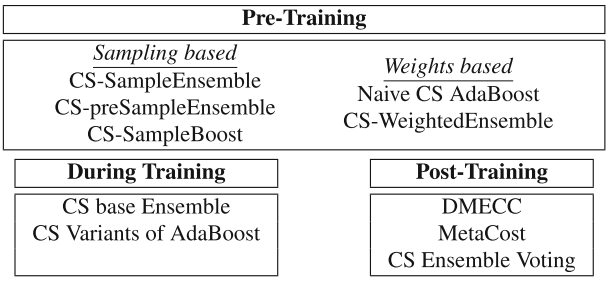


Fig. 1 The main categorisation of cost-sensitive ensemble methods with respect to the point that costs influence the classification process. For a non-exhaustive list of pre- and during-training methods see Table 2, and for post-training methods see Table 3

4. Repeat as required, keeping k fixed.

An alternative to sampling or synthesising records to reach the ratio r_{cs} in (3) is *Cost-Proportionate Rejection (CPR) Sampling* (Zadrozny et al. 2003), which is also applicable when costs are record-dependent, and where a sampled record is accepted with probability proportional to its cost:

1. Sample with replacement a uniformly random set from the training set.
2. Create a new training set that includes each of the sampled set's elements with probability $C_{FN}/\max\{C_{FN}, C_{FP}\}$ if positive or $C_{FP}/\max\{C_{FN}, C_{FP}\}$ if negative.

2.3.3 CS record weights

Ting (1998) was the first to explicitly incorporate costs in the weights w_+ of the positive and w_- of the negative classes used in weighted classifiers, followed by normalisation:

$$w_+ = C_{FN} \text{ and } w_- = C_{FP}. \tag{4}$$

Remark 4 We observe that record-dependent costs can be easily taken into account by replacing C_{FN} by C^i_{FN} and C_{FP} by C^i_{FP}. Clearly, equal costs yield equal weights.

3 Cost-sensitive ensemble methods

Cost-sensitive ensemble methods can be divided into three main categories, depending on when costs influence the classification process: before training at the data level (Sect. 3.1), during training at the algorithm level (Sect. 3.2), and after training at the decision level (Sect. 3.4). Figure 1 provides a summary. Naturally, combinations of these are possible and yield what we shall call *hybrid* methods.

The descriptions we provide are brief and sometimes slightly modified from the original ones in order to unify and generalise them, to incorporate costs when they are not explicitly mentioned, and, where possible, to include record-dependent costs when the original descriptions were given with only constant costs in mind. They are also

base-classifier independent, meaning that apart from decision trees that we focus on in this paper, other classifiers such as logistic regression and support vector machines can be used as well.

3.1 Pre-training methods

Pre-training methods employ either cost-sensitive data sampling or record weights, and as a result models need to be retrained in case the costs change.

3.1.1 Sampling based

CS-SampleEnsemble By this we describe the class of ensembles that use some cost-sensitive sampling method to modify the training set before training each base classifier. This is a generalisation of the concept used in Costing (Zadrozny et al. 2003) where subsets of the training set are obtained by CPR sampling. The other examples found in the literature are Balanced Random Forest (Chen et al. 2004) which uses equally sized sets (originally meant to be balanced and thus cost-insensitive) to build Random Input Selection Tree models (Sect. 2.2.4), EasyEnsemble (Liu et al. 2008), an ensemble of ensembles where under-sampling is used to obtain a number of equally sized sets and build AdaBoost models, and SMOTEbagging and UnderBagging (Wang and Yao 2009) which respectively use SMOTE and random undersampling. It can be seen as Bagging (Sect. 2.2.1) with modified sampling step:

1. Using a cost-sensitive sampling method, sample a number of sets from the training set.

Remark 5 Although the original definition of Costing did not include models that also produce probabilities, we could not find any reason to exclude them. Costing reduces to Bagging when costs are equal, as CPR-sampling first randomly samples the data with replacement. Other sampling methods however can sample with replacement at most one of the classes.

CS-preSampleEnsemble We propose this as the class of ensembles that use some cost-sensitive sampling method to first modify the training set before sampling subsets from it in the manner of Bagging. It can be viewed as Bagging (Sect. 2.2.1) with the following different steps:

0. Modify the training set by means of a cost-sensitive sampling method.
1. Sample with replacement a number of uniformly random and equally sized sets from the modified training set.

Remark 6 Using CS-undersampling has the disadvantage of producing modified sets of a rather small size. Also, when $C_{FP} = C_{FN}$, CS-preSampleEnsemble reduces to Bagging.

CS-SampleBoost By this we describe the class of AdaBoost variants that modify the weight of each record by using some sampling method on the training set. The examples found in the literature are SMOTEBoost (Chawla et al. 2003) and RUSBoost

(Seiffert et al. 2010) which respectively use SMOTE and random undersampling. The steps different to AdaBoost (Sect. 2.2.3) are:

2a. Modify the training set by means of a cost-sensitive sampling method.
2b. Normalise each record's weight in the modified set by dividing it by the sum of weights of all records in it.
3. Build a model using the modified set of weighted records and obtain for the initial set each record's outcome $h \in \{0, 1\}$ and the model's total error ε as the sum of weights of all misclassified records.
5. Repeat steps 2a to 4 as required.

Remark 7 When $C_{FP} = C_{FN}$, CS-SampleBoost reduces to AdaBoost.

3.1.2 Weights based

Naive CS AdaBoost Mentioned by Viola and Jones (2001), Zadrozny et al. (2003) and Masnadi-Shirazi and Vasconcelos (2011), its only difference to AdaBoost are the cost-dependent initial weights in Step 1.

1. Assign to each positive and negative record in the training set weights as in (4).[1]

Remark 8 When $C_{FP} = C_{FN}$, Naive CS AdaBoost reduces to AdaBoost.

CS-WeightedEnsemble By this we describe the class of special cases of Bagging where the models built are weighted and the weights initialised as in (4). It is a generalisation of Weighted Random Forest (Chen et al. 2004), a variant of Random Forests that builds weighted Random Input Selection Tree models (see Sect. 2.2.4). Weighted CART and C4.5 can also be used. We only describe the steps that are different to Bagging (Sect. 2.2.1):

0. Assign to each positive and negative record in the training set weights as in (4).
2. For each sampled set, normalise the weights and build a weighted model.
3. A record's final outcome is the weighted majority vote on its outcome from all models, the weights being the average record weights at the tree nodes reached by the record.

3.2 During-training methods

In during-training methods, costs directly influence the way base classifiers are built, which therefore have to be rebuilt when costs change.

3.2.1 CS base ensemble

Cost-insensitive ensemble methods such as Bagging can be made cost-sensitive by employing base classifiers whose decisions based on maximising accuracy are replaced by decisions based on minimising misclassification costs. Restricting our attention in

[1] In Viola and Jones (2001) the costs are actually $w_+ = \sqrt{C_{FN}/C_{FP}}$ and $w_- = \sqrt{C_{FP}/C_{FN}}$.

this paper to binary decision trees, the possibilities are cost-sensitive node splitting and/or tree pruning.

CS node Splitting By replacing the impurity measure (such as Entropy or the Gini index) by a cost measure, node splitting in a decision tree is made cost-sensitive.

An example is Decision Trees with Minimal Costs (Ling et al. 2004) that do cost-minimising splitting and labelling without pruning. In detail, during tree-growing, a node t is labelled according to $P_{t_+} > T_{cs}$. The costs C_t at this node are $\sum_{i \in N_t^+} C^i_{FN}$ if the node is labelled as negative and $\sum_{i \in N_t^-} C^i_{FP}$ otherwise. The attribute selected for node-splitting is the one that instead of maximising the gain value maximises $C_t - \sum_{i=1}^{k} C_{t_i}$, where C_{t_i} to C_{t_k} are the costs at the children nodes of node t, calculated the same way as C_t.

CS pruning By replacing the accuracy measure by a cost measure, tree pruning becomes cost-sensitive. Examples include Reduced Cost Pruning (Pazzani et al. 1994) and Cost-Sensitive Pruning (Bradford et al. 1998) which respectively modify the reduced error pruning of C4.5 and the cost-complexity pruning of CART to calculate costs instead of errors, and Knoll et al. (1994) where both are done.

A hybrid example are Cost Sensitive Decision Trees (Bahnsen et al. 2015) that do the same cost-minimising splitting, labelling and pruning mentioned above, with emphasis on record-dependent costs.

3.3 CS variants of AdaBoost

CS variants of AdaBoost typically use the misclassification costs to update the weights of misclassified records differently per class. They include UBoost[2] (Ting and Zheng 1998a), AdaCost (Fan et al. 1999), AdaUBoost (Karakoulas and Shawe-Taylor 1999), Asymmetric AdaBoost (AssymAB, (Viola and Jones 2001), CSB0 (Ting 2000a, b), CSB1 and CSB2 (Ting 2000a), AdaC1, AdaC2 and AdaC3 (Sun et al. 2007), and Cost-Sensitive AdaBoost (CSAB, (Masnadi-Shirazi and Vasconcelos 2011)). Their steps different to AdaBoost are:

1. Assign to each record in the training set weight according to Table 1.
4. Update each record's weight according to Table 1.
6. A record's final outcome is the weighted majority vote on its outcome from all models, the weights being according to Table 1.

Remark 9 It is not immediately clear how record-dependent costs can be used in CSAB.

All these have been theoretically analysed in (Nikolaou et al. 2016) together with Naive CS AdaBoost (Sect. 3.1.2) and AdaMEC (Sect. 3.4.3) from different viewpoints, with the conclusion that only the latter two and AsymAB have solid foundations, while calibration improves performance.

[2] Its only difference to its predecessor, *Boosting* (Ting and Zheng 1998b), are the cost-based initial weights.

Table 1 Details of the CS-variants of AdaBoost

Method	Weight initialisation	Weight update	α	Vote's weight	Other details	Reduction to AdaBoost
UBoost	$w_+ = C_{FN}$ $w_- = C_{FP}$	$w' = w \cdot e^{-\alpha y_* h_*}$	$\frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$	$\alpha\,(W_+ C_{FN}$ $-W_- C_{FP})$	W_+ and W_- are the total positive and negative weights at the tree node reached by the record	×
AdaCost	$w_+ = C_{FN}$ $w_- = C_{FP}$	$w' = w \cdot e^{-\alpha y_* h_* \beta}$	$\frac{1}{2}\ln\left(\frac{1-r}{1+r}\right)$	α	record cost adjustment function β : $\beta_{TP} = (1 - C_+)/2,\ \beta_{TN} = (1 - C_-)/2,$ $\beta_{FN} = (1 + C_+)/2,\ \beta_{FP} = (1 + C_-)/2,$ $r = \sum w \cdot y_* \cdot h_* \cdot \beta$	×
AdaUBoost	$w_+ = C_{FN}$ $w_- = C_{FP}$	$w' = w \cdot e^{-\alpha y_* h_* \left(\frac{C_{FN}}{C_{FP}}\right)^y}$	$\frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$	α		if $C_{FN} = C_{FP}$
AsymAB	$w_+ = \sqrt[2m]{\frac{C_{FN}}{C_{FP}}} \cdot \frac{1}{\lvert N\rvert}$ $w_- = \sqrt[2m]{\frac{C_{FP}}{C_{FN}}} \cdot \frac{1}{\lvert N\rvert}$	$w'_+ = \sqrt[2m]{\frac{C_{FN}}{C_{FP}}} \cdot w_+ \cdot e^{-\alpha h_*}$ $w'_- = \sqrt[2m]{\frac{C_{FP}}{C_{FN}}} \cdot w_- \cdot e^{\alpha h_*}$	$\frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$	α	m : #times steps 2 to 4 are repeated in step 5	if $C_{FN} = C_{FP}$
CSBj	$w_+ = C_{FN}$ $w_- = C_{FP}$	TP &TN: $w' = w \cdot e^{-\alpha_j}$, FN: $w'_+ = C_{FN} \cdot w_+ \cdot e^{\alpha_j}$, FP: $w'_- = C_{FP} \cdot w_- \cdot e^{\alpha_j}$	$\frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$	$\alpha \cdot C_h$	$\alpha_0 = 0, \alpha_1 = 1, \alpha_2 = \alpha,$ $C_h \in \{C_{FN}, C_{FP}\}$ is the misclassification cost associated with the record's outcome h	if $j = 2$and $C_{FN} = 1 = C_{FP}$
AdaC1	$w = 1$	$w'_+ = w_+ \cdot e^{-\alpha h_* C_+}$ $w'_- = w_- \cdot e^{\alpha h_* C_-}$	$\frac{1}{2}\ln\left(\frac{1+r_t-r_f}{1-r_t+r_f}\right)$	α	$r_t = \sum_{y=h} w_+ \cdot C_+ + \sum_{y=h} w_- \cdot C_-,$	if $C_{FN} = 1 = C_{FP}$
AdaC2	$w = 1$	$w'_+ = C_{FN} \cdot w_+ \cdot e^{-\alpha h_*}$ $w'_- = C_{FP} \cdot w_- \cdot e^{\alpha h_*}$	$\frac{1}{2}\ln\left(\frac{r_t}{r_f}\right)$	α	$r_f = \sum_{y\neq h} w_+ \cdot C_+ + \sum_{y\neq h} w_- \cdot C_-,$	if $C_{FN} = C_{FP}$
AdaC3	$w = 1$	$w'_+ = C_+ \cdot w_+ \cdot e^{-\alpha h_* C_+}$ $w'_- = C_- \cdot w_- \cdot e^{\alpha h_* C_-}$	$\frac{1}{2}\ln\left(\frac{r_t+r_f+r_{2t}-r_{2f}}{r_t+r_f-r_{2t}+r_{2f}}\right)$	α	$r_{2t} = \sum_{y=h} w_+ \cdot C_+^2 + \sum_{y=h} w_- \cdot C_-^2,$ $r_{2f} = \sum_{y\neq h} w_+ \cdot C_+^2 + \sum_{y\neq h} w_- \cdot C_-^2$	if $C_{FN} = 1 = C_{FP}$
CSAB	$w_+ = \frac{1}{\lvert N^+\rvert}$ $w_- = \frac{1}{\lvert N^-\rvert}$	$w'_+ = w_+ \cdot e^{-\alpha h_* C_{FN}}$ $w'_- = w_- \cdot e^{\alpha h_* C_{FP}}$	$2b \cdot C_{FN} \cdot \cosh(C_{FN}\alpha) +$ $2d \cdot C_{FP} \cdot \cosh(C_{FP}\alpha)$ $= C_{FN} \cdot e^{-C_{FN}\alpha} \sum w_+ +$ $C_{FP} \cdot e^{-C_{FP}\alpha} \sum w_-$	α	$b = \sum_{h=0} w_+,\ d = \sum_{h=1} w_-$	if $C_{FN} = 1 = C_{FP}$ and $\lvert N^+\rvert = \lvert N^-\rvert$

As in AdaBoost (Sect. 2.2.3), the model's total error ε is the sum of weights of all misclassified records, h_* and y_* are the record's outcome h and true class y mapped from $\{0, 1\}$ to $\{-1, 1\}$, and weights are normalised after initialisation and updating. C_+ and C_- are C_{FN} and C_{FP} scaled within $(0, 1]$

3.4 Post-training methods

In post-training methods, misclassification costs influence the classification step. Thus, when not used to build hybrid models, they offer the advantage of not having to retrain models when costs change. Some of these methods are only applicable if the costs of the records to be predicted are known at the time of prediction, thus when this is not the case, the unknown costs need to be somehow estimated.

3.4.1 Direct minimum expected cost classification

Direct minimum expected cost classification (DMECC) bases the final decision of a classifier producing probabilities on a threshold $T \in \{T_{cs}, T_{thr}\}$ (see Sect. 2.3.1).

1. Build a model producing probabilities.
2. A record's outcome is obtained according to $P_+ > T$.

One possibility is to apply DMECC to an ensemble producing probabilities, as done in *Calibrated AdaBoost* (Nikolaou and Brown 2015), where AdaBoost probabilities are obtained using Platt Scaling (see Sect. 2.2.3) and $T = T_{cs}$.

Another possibility we have identified is to use DMECC to obtain a cost-sensitive outcome (instead of the default cost-insensitive one) from the base classifiers in any ensemble, when these are capable of producing probabilities, leading us to propose *DMECC-Ensemble* and *DMECC-AdaBoost*, which do so respectively in Bagging and AdaBoost.

Remark 10 If T_{cs} is constant for all records then DMECC-Ensemble and DMECC-AdaBoost should be equivalent to CS-SampleEnsemble and CS-SampleBoost respectively (excluding CPR-sampling), though relying on probabilities instead of data-sampling. This follows from the equivalence of DMECC with threshold T_{cs} and CS-sampling shown by Elkan (2001) and discussed in Sect. 2.3.2.

Remark 11 DMECC with threshold T_{cs} is only applicable if the costs of the records to be predicted are known at the time of prediction (as T_{cs} depends on them).

3.4.2 MetaCost

As originally proposed, MetaCost (Domingos 1999) relabels the training set using the predictions obtained by Bagging with DMECC and re-uses it to train a single classifier. We generalise this concept to use the predictions of any cost-sensitive classifier in the direction of (Ting 2000b) where AdaMEC is used (see Sect. 3.4.3 below) and CSB0.

1. Replace each training record's outcome by the one obtained from a cost-sensitive model.
2. Build a single (cost-insensitive) model using the relabelled records.
3. A record's outcome is its outcome from the new model.

Remark 12 MetaCost reduces cost-sensitive ensemble models to single models, which are typically more explainable but less capable of capturing all the data characteristics. As observed by Ting (2000b), these single models often perform worse than the ensemble models.

3.4.3 CS ensemble voting

Costs can also be taken into account in an ensemble during weighted majority voting.

Cost-sensitive weights for model votes In certain AdaBoost variants, such as Naive CS AdaBoost, ε is calculated on cost-based record weights and hence results in a cost-sensitive α, which serves as the weight of the model's vote in weighted majority voting. We observe that it is in fact straightforward to mimic this for any ensemble as follows:

1. Assign to each positive and negative record in the training set weights as in (4).
2. For each model in the ensemble compute $\alpha = f(\varepsilon)$, where f is some function and ε is the sum of weights of all misclassified records from the training or a validation set.

Possibilities for the function f include

$$f(\varepsilon)=\ln\left((1-\varepsilon)/\varepsilon\right),\quad f(\varepsilon)=1-\varepsilon,\quad f(\varepsilon)=e^{(1-\varepsilon)/\varepsilon}\text{ and }f(\varepsilon)=\left((1-\varepsilon)/\varepsilon\right)^2 \tag{5}$$

where the latter two providing a right-skewed distribution.

MEC-voting By MEC-Voting we shall refer to the generalisation to any ensemble of the idea behind *AdaBoost with minimum expected cost criterion* (Ting 2000a, b), or *AdaMEC* as coined by Nikolaou and Brown (2015), which is AdaBoost with modified Step 6:

6. A record's final outcome is the weighted majority vote on its outcome from all models, the weights being the product of α and the misclassification cost associated with the outcome.

Majority threshold adjustment (MTA) The outcome of weighted majority voting is positive if the sum of positive votes is greater than 0.5. Alternative cost-sensitive *majority thresholds* that can be used are T_{cs} (Sect. 2.3.1) and T_{mthr} which we define as the one that yields the least costs on a validation set as done in Thresholding described in Sect. 2.3.1.

Theorem 1 *MEC-Voting and MTA using T_{cs} are equivalent.*

Proof The weighted sums of positive votes with and without MEC-Voting in an ensemble of m models are respectively $S_1 = \frac{\sum_{i:h_i=1}\alpha_i C_{h_i}}{\sum_{i=1}^{m}\alpha_i C_{h_i}}$ and $S_2 = \frac{\sum_{i:h_i=1}\alpha_i}{\sum_{i=1}^{m}\alpha_i}$, where $C_{h_i} \in \{C_{FN}, C_{FP}\}$ is the (non-zero) misclassification cost associated with the record's outcome h_i from model i.

If $S_1 = 0$ then $S_2 = 0$ and the theorem holds trivially. Otherwise, S_1 can be expressed in terms of S_2:

$$S_1 = \frac{\sum_{i:h_i=1}\alpha_i C_{h_i}}{\sum_{i:h_i=1}\alpha_i C_{h_i}+\sum_{i:h_i=0}\alpha_i C_{h_i}} = \frac{1}{1+\frac{\sum_{i:h_i=0}\alpha_i C_{h_i}}{\sum_{i:h_i=1}\alpha_i C_{h_i}}} = \frac{1}{1+\frac{C_{FP}}{C_{FN}}\left(\frac{\sum_{i:h_i=0}\alpha_i}{\sum_{i:h_i=1}\alpha_i}+1-1\right)}$$
$$= \frac{1}{1+\frac{C_{FP}}{C_{FN}}\left(\frac{\sum_{i=1}^{m}\alpha_i}{\sum_{i:h_i=1}\alpha_i}-1\right)} = \frac{1}{1+\frac{C_{FP}}{C_{FN}}\left(\frac{1}{S_2}-1\right)}.$$

Table 2 The basic ensembles and a (non-exhaustive) list of pre- and during-training methods derived from our framework (independent of base classifier), with abbreviations

#	Abbrv.	Name	#	Abbrv.	Name
1.	bg	Bagging (Bg)	19.	upbg	Under-preSampleEnsemble - Bg *
2.	rf	Random Forests (RF)	20.	cprpbg	CPR-preSampleEnsemble - Bg *
3.	rdf	Random Decision Forests (RDF)	21.	opbg	Over-preSampleEnsemble - Bg *
4.	wbg	weightedEnsemble - Bg *	22.	uprf	Under-preSampleEnsemble - RF *
5.	wrf	weightedEnsemble - RF	23.	cprprf	CPR-preSampleEnsemble - RF *
6.	wrdf	weightedEnsemble - RDF *	24.	oprf	Over-preSampleEnsemble - RF *
7.	ab	AdaBoost - AB	25.	uprdf	Under-preSampleEnsemble - RDF *
8.	ncsab	Naive CS AB	26.	cprprdf	CPR-preSampleEnsemble - RDF *
9.	ac*i*	AdaC*i*, $i \in \{1, 2, 3\}$	27.	oprdf	Over-preSampleEnsemble - RDF *
10.	acost	AdaCost	28.	ubg	Under-SampleEnsemble - Bg
11.	aub	AdaUBoost	29.	cprbg	CPR-SampleEnsemble - Bg
12.	csa	CSAB	30.	obg	Over-SampleEnsemble - Bg
13.	csb*i*	CSB*i*, $i \in \{0, 1, 2\}$	31.	urf	Under-SampleEnsemble - RF *
14.	asb	Asymmetric AB	32.	cprrf	CPR-SampleEnsemble - RF *
15.	usb	Under-SampleBoost	33.	orf	Over-SampleEnsemble - RF *
16.	cprb	CPR-SampleBoost *	34.	urdf	Under-SampleEnsemble - RDF *
17.	dab	DMECC AB*	35.	cprrdf	CPR-SampleEnsemble - RDF *
18.	dm-	DMECC-Ensemble*	36.	ordf	Over-SampleEnsemble - RDF *

Novel ones identified in this work are indicated by an asterisk *. The DMECC-Ensemble prefix *dm-* is applicable to all non-AdaBoost ensembles in the list

Solving this equality for S_2 and using the fact that a record's outcome is positive if $S_1 > 0.5$ and negative otherwise, we obtain $S_2 > \frac{C_{FN}}{C_{FN}+C_{FP}}$, which is MTA using T_{cs} as required. □

Remark 13 The equivalence of Theorem 1 was shown specifically for AdaMEC by Nikolaou et al. (2016).

Remark 14 Both MEC-Voting and MTA using T_{cs} are only applicable if the costs of the records to be predicted are known at the time of prediction.

3.5 Hybrid methods

Table 3 gives an overview of how post-training methods can be combined with pre- or during training methods to yield hybrid methods.

4 Towards determining the state-of-the-art in cost-sensitive learning

A natural question to ask is which, if any, of the described framework components can be considered as state-of-the-art. Obtaining an indication on this requires a rigorous

Table 3 Overview of the post-training methods combinable with each type of pre- and during-training method according to the probability calibration used

Method type	Probability calibration	Ensemble's output		DMECC		MTA*	Meta cost
		Avg(pr)	wtMaj(cls)*	T_{cs}	T_{thr}		
	None	×	✓	×	×	✓	✓
AdaBoost & variants	Logistic correction	×	✓	✓	✓	✓	✓
	Platt scaling	×	✓	✓	✓	✓	✓
	Isotonic regression	×	✓	✓	✓	✓	✓
CS-weighted ensemble	None	✓	✓	✓	✓	✓	✓
	None	✓	✓	✓	✓	✓	✓
Others	Laplace smoothing	✓	✓	×	✓	✓	✓
	m-estimation	✓	✓	×	✓	✓	✓
	Curtailment	✓	✓	×	✓	✓	✓

There are four main components, namely DMECC, type of the ensemble's output, MTA and MetaCost, and their combination yields different hybrid models. Considering all combinations is a novelty of this work, and so are the two components indicated by an asterisk *. Avg(pr) denotes the average probability of the models in the ensemble, and wtMaj(cls) the class predicted by their weighted majority, considering both equal and cost-sensitive weights $\alpha = f(\varepsilon)$, the latter given by all functions f specified in (5)

experimental comparison over a range of datasets, often referred to as *benchmarking*. Such a benchmarking would be most useful if it considers sufficiently many publicly available datasets in order to allow reproducibility and the updating of the benchmarking via the inclusion of newly proposed methods. As already mentioned in the Introduction (Sect. 1.1), there are two main uses of cost-sensitive learning, which should be considered separately.

The first use is for treating class-imbalance alone, in which case the misclassification costs do not necessarily have to be derived from the dataset or application domain and can be randomly assigned. Typically, different pairs of constant (class-dependent) costs are tried out in search for the one that gives the best results according to the metric of choice, which should be suitable for imbalanced datasets. A benchmarking can therefore be performed on a selection of the many imbalanced datasets already publicly available, and different sub-cases can depend on the level of imbalance. Although such a benchmarking will give an indication on the framework components that are best suited for treating class imbalance, in order to provide a complete picture it needs to part of a more general benchmarking that includes cost-insensitive methods specifically aiming at treating imbalance (see for example (Galar et al. 2011) for an overview) as well.

The second use is for minimising misclassification costs that are derived from the application domain or dataset, irrespective of the level of imbalance. In some cases, determining these costs is straightforward (such as the amount of a donation or transaction), whereas in others it can be non-trivial. For instance, the provider of the dataset considered by Petrides et al. (2020) had to calculate the costs using their domain knowledge and experience, and the costs considered by Lawrance et al.

(2021) depend on a parameter whose estimation first requires to design and carry-out appropriate experiments.

Typically, the evaluation measure depends on these costs, often simply being their sum. This, however, might not be sufficient in certain cases, which include fraud detection and direct marketing, and an additional evaluation measure might be necessary. For instance, in fraud detection we are in practice interested in models that do not disrupt the operation of the business, thus our attention should be restricted to models that not only achieve the lowest costs, but also a realistic False Positive Rate (FPR). In the case of credit cards in particular, a FPR of at most 3% should be within the capacity of investigating agents. In direct marketing scenarios, contacting potential respondents to a request (such as for making a donation or a purchase) may incur costs (such as for postage). Thus, the application of a model assumes that a budget that provides the capacity to contact all those the model predicts as respondents is readily available, which might not always be the case. For this reason it would be more appropriate to also look at the *return on investment* (ROI), given as the net profit over expenditure. These two examples suggest that a benchmarking should probably be done per application domain, something not unusual (see for instance (Lessmann et al. 2015) for a benchmark in the domain of credit scoring, albeit not focusing on cost-sensitive methods and using private datasets).

The main obstacle preventing such a benchmarking is the absence of sufficiently many publicly available datasets in general, let alone per application domain. The alternative of using publicly available datasets having no attribute from which misclassification costs can be derived, and assigning random costs to them (as done for just treating class imbalance we discussed above), might give misleading results in the absence of a clear business case. Moreover, constant costs are quite rare in domains such as direct marketing and fraud detection, and assigning random record-dependent costs is a difficult task, mainly because the distribution from which they should be taken is unknown. We hope that practitioners will receive this as an open call to make more datasets publicly available in order to facilitate advances in the field, from which they can benefit as well.

For the interested reader, in Appendix A, we provide four examples of cost-minimisation applications of cost-sensitive learning using publicly available datasets.

5 Conclusions and future work

In this paper we have described and categorised available cost-sensitive methods with respect to misclassification costs by means of a unifying framework that also allowed us to identify new combinations and extend ideas across classifiers. This was our main contribution, which clarifies the picture and should aid further developments in the domain and serve as a baseline to which newly proposed methods should be compared.

As our work has identified, some possibilities for new cost-sensitive ensemble methods can arise by developing a novel approach in any of the following domains: (a) cost-sensitive base classifiers such as decision trees with cost-sensitive node splitting and pruning, (b) cost-sensitive sampling, and using costs to (c) specify record weights,

(d) update weights in AdaBoost variants, and (e) specify classifier weights for ensemble voting.

Worth exploring are how *Ensemble Pruning* (Zhou 2012) and *Stacking* (Wolpert 1992) (which instead of using the outputs of all the ensemble members for voting or averaging, respectively first choose a subset of them, or use them to train a second model) can be made cost-sensitive for inclusion in the post-training methods. It would also be interesting to investigate whether *Gradient Boosting* (Friedman 2001), another representative of the boosting principle, and its popular variant *XGBoost* (Chen and Guestrin 2016) can have cost-sensitive variants, in particular with record-dependent costs, apart from being combined with post-training methods.

Another avenue for future research is to examine cost-sensitivity with respect to other types of costs as mentioned by Turney (2000), particularly costs of attribute acquisition and costs of obtaining missing values in the data (Turney 1995; Ling et al. 2004), that are important in many real world applications.

Acknowledgements The authors are grateful for the useful recommendations provided by anonymous reviewers of the present and earlier versions of this work.

Funding Open access funding provided by University of Bergen (incl Haukeland University Hospital).

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

A Example applications

Here we briefly describe four examples of how cost-sensitive learning can be applied in real world scenarios where the main task is cost-minimisation. They are all based on publicly available datasets.

A.1 Datasets

Direct marketing

The first dataset is on a donations to charity campaign in the United States and is available on-line as part of the 1998 KDD cup.[3] The dataset contains 483 attributes

[3] https://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html.

related to people contacted by mail and asked to donate to a charity, including the donation amount (where 0 indicates no donation). It is a highly imbalanced set whose positive records (the donors) are 5.1% of it.

Churn prediction

The second dataset is on churn prediction in the Telecom industry and is available on-line at Kaggle.[4] It comprises 7043 customer records of 21 attributes, including whether they churned or not. It is a moderately imbalanced set whose positive records (the churners) constitute 26.54% of it.

Credit card fraud detection

The third dataset is on credit card fraud detection and is available on-line at Kaggle.[5] In total, two days of credit card transactions are given (284,807 records of 31 attributes). For obvious security and user privacy reasons, all the attributes names and values have been scrambled and transformed into numerical ones by the owners, apart from the transaction's amount, the time (in seconds) elapsed between the transaction and the first transaction in the dataset, and whether the transaction was fraudulent or not. It is an extremely imbalanced set whose positive records (the fraudulent transactions) are a mere 0.17% of it.

Credit scoring

The fourth and final dataset is on credit scoring and is available on-line on GitHub.[6] It contains an ID and 20 anonymised features related to 18,918 accepted loans given to private clients through 3 different channels after some screening, including whether the loan is in default or not. The level of imbalance varies per channel, with positive records (the loans in default) respectively constituting 5.6%, 15% and 34% of the total number in each. More information about the dataset can be found in Petrides et al. (2020), where it was introduced.

A.2 Data preparation

The datasets were segmented into three disjoint subsets respecting the global proportion of positive records: a training set used for building models, a validation set used for determining parameters such as T_{thr} and ε used for specifying model weights, and a test set used for model evaluation. Table 4 provides the details.

Direct marketing

The data is provided in two parts. Following the competition guidelines, we used the first part for training and validation and the second one for evaluation. As the

[4] https://www.kaggle.com/blastchar/telco-customer-churn.

[5] https://www.kaggle.com/mlg-ulb/creditcardfraud/.

[6] https://github.com/vub-dl/data-csl-pdcs.

Table 4 Overview of the 4 datasets

#	Domain	Set	Size		+ves (%)	#Attributes		Training (C_{FP}, C_{FN})
			#	%		Total	Used	
1.	Direct marketing	Total	191,779	100	5.1	483	31	(0.68, 14.45),
		Training	65,000	33.9	5.1		(8)	(0.68, $amt - 0.68$)
		Validation	30,412	15.9	5			
		Test	96,367	50.2	5			
2.	Churn prediction	Total	7043	100	26.54	21	20	(1, 6),
		Training	4508	64	26.77			(122.35, 893.09)
		Validation	1127	16	26.44			
		Test	1408	20	25.85			
3.	Credit card	Total	284,807	100	0.17	31	31	(1, *amt*), (1, 119.68),
	Fraud detection	Training	110,852	38.9	0.18			(2, *amt*), (2, 119.68),
		Validation	33,934	11.9	0.18			(5, *amt*), (5, 119.68)
		Test	140,021	49.2	0.15			
4.	Credit scoring	Total	18,918	100	16.9	21	18	(*exp*, *exl*), ($\overline{exp}$,$\overline{exl}$)
		Channel 1	14,179	75	15			
		Channel 2	2831	15	34			
		Channel 3	1908	10	5.6			

For the first one, within brackets is the number obtained after attribute engineering

amount of attributes the dataset contains is relatively large, we followed the selection and engineering approach of Georges and Milley (2000), the competition winners, to reduce their number significantly and speed up model training.

Churn prediction

There was no restriction on the splitting. The redundant customer ID attribute was dropped. Although the dataset does not have predefined costs, it does have an attribute from which they can be derived, namely *Monthly Costs*. Assuming a (realistic) campaign where customers contacted are offered 2 months for free if they renew their telephone subscription, C^i_{FP} can be set to two times *amt*, the amount charged monthly, and C^i_{FN} to twelve times *amt* (equivalent to a whole year of lost profits from the churning customers).

Credit card fraud detection

Credit card fraud models are typically built on past transactions in order to detect fraud amongst new transactions. With this in mind, we used the first day for training and validation and the other for evaluation. All attributes were used.

Credit scoring

Apart from its features, the dataset provides an indication of which records to include in the test set, which we followed. The redundant customer ID attribute was dropped, as were the channel and *days_ late*, the latter being highly correlated to the status of loans. The missing values of the *FICO_Score* variable were left untreated. Following Petrides et al. (2020), we trained on the entire dataset, and evaluated per channel.

A.3 Misclassification costs

Models were trained using both the class and record-dependent dataset-specific cost pairs (C_{FP}, C_{FN}) that are mentioned below. However, T_{cs}, T_{thr} and ε used for model weights were always computed using the record-dependent costs.

Direct marketing

(0.68, 14.45) and (0.68, $amt - 0.68$), where 0.68 is the cost of contacting a person, 14.45 is the average donation amount of the donors in the training dataset minus 0.68, and *amt* is the actual record-dependent donation's amount for positive records. To be able to use DMECC and MTA with T_{cs} and MEC-Voting (see Remarks 11 and 14), we estimated *amt* for negative records as the would-have-been donation if the record was in fact positive via a linear regression, again following Georges and Milley (2000).

Churn prediction

(1, 6), where 1 to 6 is the ratio of C^i_{FP} over C^i_{FN} as derived from the campaign we set up in Sect. A.2.

Credit card fraud detection

(1, *amt*), (1, 119.68), (2, *amt*), (2, 119.68), (5, *amt*) and (5, 119.68), where 119.68 is the average amount of fraudulent transactions in the training data, *amt* is the actual record-dependent transaction's amount, and 1, 2 and 5 are the (class-dependent) overhead costs we consider in the absence of a clear idea of what they are.

Credit scoring

(*exp*,*exl*) and ($\overline{exp}$,$\overline{exl}$), where *exp* and *exl* are the record-dependent profit and loss estimates per loan as included in the dataset, and the overlines indicate their (class-dependent) averages in each training set. For more information on how these non-trivial estimates were made by the loan provider, see Petrides et al. (2020).

A.4 Evaluation

We evaluate models based on their cost saving performance using the record-dependent misclassification costs we defined per dataset. For reference, we also include *True* and

False positive rates (TPR and FPR), and *area under the ROC curve* (AUC (Provost and Fawcett 2001)). These are standard accuracy-based measures, and while the former two depend on the decision threshold used, the latter evaluates performance across all possible thresholds.

Direct marketing

The total net profit is computed as the sum of donations by people contacted minus a cost of 0.68 per person contacted. As discussed in Sect. 4, we also look at ROI. For example, the trivial model of targeting everyone on the list yields a net profit of 10,560.08, but only if the budget to cover the associated 65,529.56 in postage costs is available, leading to a ROI of 0.16. Next, consider a naive CS AdaBoost model with net profit 9282.59 that has ROI 0.59. Even though the profit is about 1300 less, the ROI is almost 4 times larger, meaning that only one fourth of the trivial model's budget is needed.

Churn prediction

We calculated Cost%, the percentage by which costs are reduced due to the campaign as compared to having no campaign and losing all churners, with the assumption that all contacted churners accept the offered 2 months of free subscription and are retained.

Credit card fraud detection

We used *amt* as C^i_{FN} and computed the percentage of the sum of the amounts of all fraudulent transactions that is saved by a model by correctly detecting fraud, which we denote by TotF%. As discussed in Sect. 4, we restrict our attention those that achieve a realistic FPR of at most 3%.

Credit scoring

Following Petrides et al. (2020), by first obtaining its estimated profit (the sum of *exp* of all negative records minus the sum of *exl* of all positive ones), we calculated each model's MaxProfit% as the percentage of the maximum possible estimated profit (achieved by only accepting loans not in default) the model's profit constitutes, separately per channel.

A.5 Experiments

We implemented our framework in the software package R (R Core Team 2021), using a combination of existing and our own implementations. All ensembles we built consist of 100 rpart trees (the implementation of CART in R), for Random Forests we allowed the growing of deep trees, whereas decision stumps (trees of depth 2) were grown for AdaBoost and variants. All experiments were run three times using a different splitting of the dataset into training and validation sets (see Sect. A.2), and the results were averaged.

Table 5 The 10 models with the highest profit in the Direct Marketing Dataset, using the abbreviations from Tables 2 and 3

#	Method	Training (C_{FP}, C_{FN})	Probability calibration	Ensemble's output	DMECC/ MTA	α	Profit	ROI	TPR (%)	FPR (%)	AUC
1.	dm-rdf		m-estim.	wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	**13,769.75**	0.28	69.37	75.40	0.458
2.	dm-rf		Laplace	wtMaj(cls)		$e^{\frac{1-\varepsilon}{\varepsilon}}$	13,729.00	0.37	59.97	56.93	0.519
3.	dm-rf			wtMaj(cls)		$e^{\frac{1-\varepsilon}{\varepsilon}}$	13,700.18	0.37	59.03	55.97	0.519
4.	dm-rf			wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	13,698.68	**0.38**	58.80	**55.67**	0.520
5.	dm-rf		Laplace	wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	13,683.40	0.37	59.73	56.63	0.519
6.	dm-rf		Laplace	wtMaj(cls)		$1-\varepsilon$	13,679.26	0.37	60.13	57.20	0.519
7.	rdf			Avg(pr)	T_{cs}		13,674.27	0.28	69.30	75.03	**0.603**
8.	dm-rdf		m-estim.	wtMaj(cls)		$e^{\frac{1-\varepsilon}{\varepsilon}}$	13,662.86	0.28	**69.70**	76.03	0.458
9.	dm-rf			wtMaj(cls)		$1-\varepsilon$	13,662.41	0.37	59.17	56.20	0.519
10.	dm-rdf		Laplace	wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	13,654.57	0.30	63.40	69.77	0.458
*	rpart		m-estim.				94.60		0.20	0.10	0.594

We observe that their ranking according to the ROI column would be different. All DMECC-ensembles happen to use T_{cs}, and the best performing cost-insensitive model (*) is included for reference

Table 6 The 10 models with the highest Cost% in the Churn Prediction Dataset, using the abbreviations from Tables 2 and 3

#	Method	Training (C_{FP}, C_{FN})	Probability calibration	Ensemble's output	DMECC/ MTA	α	Cost%	TPR (%)	FPR (%)	AUC
1.	acost	(1, 6)	Platt	wtMaj(cls))	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	**70.84**	90.70	44.00	0.834
2.	asb	(1, 6)		wtMaj(cls)		$\log(\frac{1-\varepsilon}{\varepsilon})$	70.84	90.40	44.40	**0.839**
3.	asb	(1, 6)	Logistic	wtMaj(cls)		$\log(\frac{1-\varepsilon}{\varepsilon})$	70.84	90.40	44.40	0.839
4.	ab		Logistic	wtMaj(cls)	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	70.77	90.10	44.20	0.838
5.	dm-bg			wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	70.61	91.80	47.63	0.811
6.	dm-bg		Laplace	wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	70.58	**92.00**	**48.20**	0.811
7.	dm-bg			wtMaj(cls)		$e^{\frac{1-\varepsilon}{\varepsilon}}$	70.57	**92.00**	**48.20**	0.812
8.	asb	(1, 6)	Platt	wtMaj(cls)	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	70.57	90.10	44.40	0.839
9.	ab		Platt	wtMaj(cls)	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	70.55	90.10	44.60	0.838
10.	dm-bg		Laplace	wtMaj(cls)		1	70.53	91.80	47.87	0.810
*	ab		Platt	wtMaj(cls)		$\log(\frac{1-\varepsilon}{\varepsilon})$	51.04	53.00	10.60	0.838

All DMECC-ensembles happen to use T_{cs}. The best performing cost-insensitive model (*) is included for reference

Table 7 The 5 models with the highest TotF% and FPR respectively at most 1%, 2% and 3% in the Credit Card Fraud Detection Dataset, using the abbreviations from Tables 2 and 3

#	Method	Training (C_{FP}, C_{FN})	Probability calibration	Ensemble's output	DMECC/ MTA	α	TotF%	FPR (%)	TPR (%)	AUC
1.	dm-upbg	(5, 119.31)		wtMaj(cls)		$\log(\frac{1-\varepsilon}{\varepsilon})$	**80.67**	0.97	75.83	0.889
2.	dm-oprdf	(5, 119.31)	m-estim.	wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	80.42	0.87	70.93	0.891
3.	dm-oprdf	(1, 119.31)	Laplace	wtMaj(cls)		$\log(\frac{1-\varepsilon}{\varepsilon})$	79.86	**0.83**	74.23	0.889
4.	dm-cprbg	(5, 119.31)		wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	79.52	0.97	75.67	0.884
5.	dm-cprbg	$(5, amt)$	m-estim.	wtMaj(cls)	T_{cs}	$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	77.26	0.87	**76.13**	**0.959**
1.	dm-uprdf	(5, 119.31)	Laplace	wtMaj(cls)		1	**88.69**	1.83	68.23	**0.891**
2.	dm-cprrdf	(5, 119.31)	Laplace	wtMaj(cls)		1	88.20	1.73	68.40	0.877
3.	dm-cprrdf	(5, 119.31)	Laplace	wtMaj(cls)		$\log(\frac{1-\varepsilon}{\varepsilon})$	88.15	1.63	**68.90**	0.880
4.	dm-cprrdf	(5, 119.31)		wtMaj(cls))		1	88.15	1.63	68.23	0.886
5.	dm-cprrdf	(5, 119.31)		wtMaj(cls))		$1-\varepsilon$	88.15	**1.60**	68.23	0.887
1.	dm-oprdf	(2, 119.31)	m-estim.	wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	**95.64**	**2.53**	**77.73**	**0.899**
2.	dm-cprrdf	(5, 119.31)	m-estim.	wtMaj(cls)		1	94.86	2.70	70.77	0.859
3.	dm-cprrdf	(5, 119.31)	m-estim.	wtMaj(cls)		$1-\varepsilon$	94.86	2.63	70.77	0.860
4.	dm-uprdf	(5, 119.31)	m-estim.	wtMaj(cls)		1	94.52	2.70	70.13	0.870
5.	dm-uprdf	(5, 119.31)	m-estim.	wtMaj(cls)		$1-\varepsilon$	94.52	2.67	70.13	0.873
*	rpart						48.34	0.10	72.00	0.903

All DMECC-ensembles use T_{cs} except one that uses T_{thr}. The best performing cost-insensitive model (*) is included for reference

A.6 Analysis of experimental results

Tables 5, 6, 7 and 8 show the best-performing models per dataset as obtained from our experiments. The main observation is that in all datasets we obtain cost-sensitive models that are superior to the trivial and cost-insensitive models, indicating the positive effect cost-sensitive learning can have in these domains, even in the presence of class imbalance.

The small number of datasets considered prevent general conclusions about model performance, both per domain and as a whole. Despite this, by looking at these results we can make a few general observations, whose validity should be verified on other such datasets from real-world domains (c.f. our discussion in Sect. 4).

1. With very few exceptions, almost all of the reported models are hybrid models.
2. DMECC-ensembles are well-performing overall.
3. Weighted majority voting with cost-sensitive model weights seems to be a reasonable approach.
4. Random Decision Forests and Random Forests variants, enabled via our framework, emerge as promising cost-sensitive classifiers next to AdaBoost and Bagging variants.

Table 8 The 5 models with the highest MaxProfit% for Channels 1, 2 and 3 (separated by horizontal lines) in the Credit Scoring Dataset, using the abbreviations from Tables 2 and 3

#	Method	Training (C_{FP}, C_{FN})	Probability calibration	Ensemble's output	DMECC/MTA	M	TotF%	FPR (%)	TPR (%)	AUC
1.	dm-rf		m-estim.	wtMaj(cls)	T_{cs}	$\exp(\frac{1-\varepsilon}{\varepsilon})$	**71.95**	4.90	20.30	**0.740**
2.	cprdrf	(exp, exl)	Laplace	wtMaj(cls)	T_{cs}	1	71.84	6.13	**24.70**	0.731
3.	dm-rf		m-estim.	wtMaj(cls)	T_{cs}	$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	71.84	5.07	20.57	**0.740**
4.	dm-rf		m-estim.	wtMaj(cls)		$\exp(\frac{1-\varepsilon}{\varepsilon})$	71.75	**4.60**	19.33	**0.740**
5.	dm-rf		m-estim.	wtMaj(cls)		$\left(\frac{1-\varepsilon}{\varepsilon}\right)^2$	71.74	**4.60**	19.27	**0.740**
*	rpart						70.29	1.00	3.40	0.717
1.	ab		Logistic	wtMaj(cls)	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	**53.23**	8.00	28.53	**0.750**
2.	ncsab	(exp, exl)	Platt	wtMaj(cls)	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	53.00	7.47	22.40	0.745
3.	ncsab	(exp, exl)	Logistic	wtMaj(cls)	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	53.00	7.20	22.07	0.745
4.	ncsab	(exp, exl)		wtMaj(cls)	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	52.97	**7.17**	21.83	0.745
5.	asb	(exp, exl)		wtMaj(cls)		$\log(\frac{1-\varepsilon}{\varepsilon})$	52.70	9.03	**31.20**	0.745
*	rpart		m-estim.				23.11	5.60	15.37	0.695
1.	wrf	(exp, exl)		wtMaj(cls)	T_{cs}	1	**89.72**	**0.73**	17.73	**0.699**
2.	cprrf	$(\overline{exp}, \overline{exl})$	Laplace	wtMaj(cls)	T_{cs}	1	89.65	1.37	**21.90**	0.681
3.	cprrf	$(\overline{exp}, \overline{exl})$	Laplace	wtMaj(cls)	T_{cs}	$1-\varepsilon$	89.65	1.37	**21.90**	0.682
4.	cprrf	$(\overline{exp}, \overline{exl})$	Laplace	wtMaj(cls)	T_{cs}	$\log(\frac{1-\varepsilon}{\varepsilon})$	89.65	1.37	**21.90**	0.682
5.	cprrf	$(\overline{exp}, \overline{exl})$		wtMaj(cls)	T_{cs}	1	89.46	1.50	**21.90**	0.681
*	rpart						87.96	0.23	2.07	0.724

All DMECC-ensembles happen to use T_{cs}. The best performing cost-insensitive model (*) per channel is included for reference

References

Ayer M, Brunk H, Ewing G, Reid W, Silverman E (1955) An empirical distribution function for sampling with incomplete information. Ann Math Stat 26(4):641–647

Bahnsen AC, Aouada D, Ottersten B (2015) Example-dependent cost-sensitive decision trees. Expert Syst Appl 42:6609–6619

Bradford J, Kuntz C, Kohavi R, Brunk C, Brodley C (1998) Pruning decision trees with misclassification costs. In: European conference on machine learning ECML, LNCS, vol 1398, pp 131–136

Breiman L (1996) Bagging predictors. Mach Learn 26(2):123–140

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees, 1st edn. Routledge. https://doi.org/10.1201/9781315139470

Cestnik B (1990) Estimating probabilities: a crucial task in machine learning. In: European conference on artificial intelligence ECAI, pp 147–149

Chawla N, Bowyer K, Hall L, Kegelmeyer W (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Chawla N, Lazarevic A, Hall L, Bowyer K (2003) SMOTEboost: improving prediction of the minority class in boosting. In: Knowledge discovery in databases PKDD, LNAI, vol 2838, pp 107–119

Chen T, Guestrin CE (2016) XGBoost: a scalable tree boosting system. In: ACM SIGKDD International conference on knowledge discovery and data mining KDD, pp 785–794

Chen C, Liaw A, Breiman L (2004) Using random forest to learn imbalanced data. Statistics Technical Report 666, University of California, Berkley
Choi S, Kim YJ, Briceno S, Mavris D (2017) Cost-sensitive prediction of airline delays using machine learning. In: IEEE/AIAA digital avionics systems conference DASC, pp 1–8
Coussement K (2014) Improving customer retention management through cost-sensitive learning. Eur J Mark 48(3/4):477–495
Domingos P (1999) Metacost: a general method for making classifiers cost-sensitive. In: ACM SIGKDD International conference on knowledge discovery and data mining KDD, pp 155–164
Elkan C (2001) The foundations of cost-sensitive learning. In: International joint conference on artificial intelligence IJCAI, pp 973–978
Fan W, Stolfo SJ, Zhang J, Chan PK (1999) AdaCost: misclassification cost-sensitive boosting. In: International conference on machine learning ICML, pp 97–105
Freund Y, Schapire R (1997) A decision–theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139
Friedman J (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29(5):1189–1232
Friedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting. Ann Stat 28(2):337–374
Galar M, Fernández A, Barrenechea E, Bustince H, Herrera F (2011) A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans Syst Man Cybern Part C Appl Rev 42(4):463–484
Georges J, Milley AH (2000) Kdd'99 competition: knowledge discovery contest. ACM SIGKDD Explor Newsl 1(2):79–84
He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
Ho TK (1995) Random decision forests. In: International conference on document analysis and recognition, vol 1, pp 278–282
Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844
Karakoulas G, Shawe-Taylor J (1999) Optimising classifiers for imbalanced training sets. Adv Neural Inf Process Syst 11:253–259
Knoll U, Nakhaeizadeh G, Tausend B (1994) Cost-sensitive pruning of decision trees. In: European conference on machine learning ECML, LNCS, vol 784, pp 383–386
Lawrance N, Petrides G, Guerry MA (2021) Predicting employee absenteeism for cost effective interventions. Decis Support Syst 147:113539
Lessmann S, Baesens B, Seow HV, Thomas L (2015) Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. Eur J Oper Res 247(1):124–136
Ling C, Yang Q, Wang J, Zhang S (2004) Decision trees with minimal costs. In: International conference on machine learning ICML, pp 64–71
Liu X, Wu J, Zhou Z (2008) Exploratory undersampling for class-imbalance learning. IEEE Trans Syst Man Cybern Part B Cybern 39(2):539–550
Lomax S, Vadera S (2013) A survey of cost-sensitive decision tree induction algorithms. ACM Comput Surv 45(2):Article 16
Masnadi-Shirazi H, Vasconcelos N (2011) Cost-sensitive boosting. IEEE Trans Pattern Anal Mach Intell 33(2):294–309
Niculescu-Mizil A, Caruana R (2005) Obtaining calibrated probabilities from boosting. In: Conference on uncertainty in artificial intelligence UAI, pp 413–420
Nikolaou N, Brown G (2015) Calibrating adaboost for asymmetric learning. In: Multiple classifier systems MCS, LNCS, vol 9132, pp 112–124
Nikolaou N, Edakunni N, Kull M, Flach P, Brown G (2016) Cost-sensitive boosting algorithms: do we really need them? Mach Learn 104(2):359–384
Pazzani M, Merz C, Murphy P, Ali K, Hume T, Brunk C (1994) Reducing misclassification costs. In: International conference on machine learning ICML, pp 217–225
Petrides G, Moldovan D, Coenen L, Guns T, Verbeke W (2020) Cost-sensitive learning for profit-driven credit scoring. J Oper Res Soc. https://doi.org/10.1080/01605682.2020.1843975
Platt J (1999) Probabilistic outputs for support vector machines and comparison to regularised likelihood methods. Adv Large Margin Classif 10(3):61–74
Prati R, Batista G, Monard M (2009) Data mining with imbalanced class distributions: concepts and methods. In: Indian international conference on artificial intelligence IICAI, pp 359–376

Provost F, Fawcett T (2001) Robust classification for imprecise environments. Mach Learn 42(3):203–231
Quinlan R (1993) C4.5: programs for machine learning. Morgan Kaufmann, London
R Core Team (2021) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna. https://www.R-project.org/
Robertson T, Wright F, Dykstra R (1988) Order restricted statistical inference. Wiley, New York
Seiffert C, Khoshgoftaar T, Hulse JV, Napolitano A (2010) RUSBoost: a hybrid approach to alleviating class imbalance. IEEE Trans Syst Man Cybern Part A Syst Hum 40(1):185–197
Shapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Mach Learn 37(3):297–336
Sheng V, Ling C (2006) Thresholding for making classifiers cost-sensitive. In: National conference on artificial intelligence AAAI, vol 1, pp 476–481
Sun Y, Kamel M, Wong A, Wang Y (2007) Cost-sensitive boosting for classification of imbalanced data. Pattern Recognit 40:3358–3378
Sun Y, Wong A, Kamel M (2009) Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell 23(4):687–719
Ting K (1998) Inducing cost-sensitive trees via instance weighting. In: European symposium on principles of data mining and knowledge discovery PKDD, LNCS, vol 1510, pp 139–147
Ting K (2000a) A comparative study of cost-sensitive boosting algorithms. In: International conference on machine learning ICML, pp 983–990
Ting K (2000b) An empirical study of metacost using boosting algorithms. In: European conference on machine learning ECML, LNAI, vol 1810, pp 413–425
Ting K (2002) An instance-weighting method to induce cost-sensitive trees. IEEE Trans Knowl Data Eng 14(3):659–665
Ting K, Zheng Z (1998a) Boosting cost-sensitive trees. In: International conference on discovery science DS, LNAI, vol 1532, pp 244–255
Ting K, Zheng Z (1998b) Boosting trees for cost-sensitive classifications. In: European conference on machine learning ECML, LNAI, vol 1398, pp. 190–195
Turney P (1995) Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. J Artif Intell Res 2:369–409
Turney P (2000) Types of cost in inductive concept learning. In: International conference on machine learning ICML, pp 15–21
Viola P, Jones M (2001) Fast and robust classification using assymetric adaboost and a detector cascade. Adv Neural Inf Process Syst 14:1311–1318
Wang S, Yao X (2009) Diversity analysis on imbalanced data sets by using ensemble models. In: IEEE Symposium on computational intelligence and data mining CIDM, pp 324–331
Wolpert DH (1992) Stacked generalisation. Neural Netw 5:241–259
Xia Y, Liu C, Liu N (2017) Cost-sensitive boosted tree for loan evaluation in peer-to-peer lending. Electron Commerce Res Appl 24:30–49
Zadrozny B, Elkan C (2001a) Learning and making decisions when costs and probabilities are both unknown. In: ACM SIGKDD International conference on knowledge discovery and data mining, pp 204–213
Zadrozny B, Elkan C (2001b) Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: International conference on machine learning ICML, pp 609–616
Zadrozny B, Elkan C (2002) Transforming classifier scores into accurate multiclass probability estimates. In: ACM SIGKDD International conference on knowledge discovery and data mining, pp 694–699
Zadrozny B, Langford J, Abe N (2003) Cost-sensitive learning by cost-proportionate example weighting. In: IEEE International conference on data mining ICDM, pp 435–442
Zhou Z (2012) Ensemble methods foundations and algorithms. CRC Press, Boca Raton

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