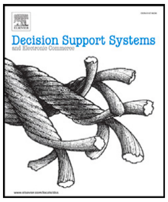
Interpretable cost-sensitive regression through one-step boosting

Decision Support Systems 175 (2023) 114024

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss



Interpretable cost-sensitive regression through one-step boosting

Thomas Decorte [a], Jakob Raymaekers [c], Tim Verdonck [b,d,*]

[a] University of Antwerp, Department of Mathematics, Middelheimlaan 1, Antwerp 2020, Belgium
[b] University of Antwerp - imec, Department of Mathematics, Middelheimlaan 1, Antwerp 2020, Belgium
[c] Maastricht University, Department of Quantitative Economics, P.O. box 616, 6200 MD, Maastricht, The Netherlands
[d] KU Leuven, Department of Mathematics, Celestijnenlaan 200B, Leuven 3001, Belgium

ARTICLE INFO

Dataset link: https://github.com/ThomasDecorteUA/Cost_Sensitive_Regression, https://www.cs.toronto.edu/delve/data/datasets.html, https://archive.ics.uci.edu/ml/datasets.php, https://geodacenter.github.io/data-and-lab//KingCounty-HouseSales2015/

Keywords:
Cost-sensitive regression
Asymmetric costs
Boosting
Interpretability
Data mining

ABSTRACT

In most practical prediction problems, such as regression and classification, the different types of prediction errors are not equally costly in the decision-making process. Although there exist numerous real-world cost-sensitive regression problems, ranging from loan charge-off forecasting to house price predictions, the literature on cost-sensitive learning mainly focuses on classification and only a few solutions are proposed for regression problems. These regressions are typically characterized by an asymmetric cost structure, where over- and underpredictions of a similar magnitude face vastly different costs. In this paper, we present a one-step boosting method (OSB) for cost-sensitive regression. The proposed methodology leverages a secondary learner to incorporate cost-sensitivity into an already trained cost-insensitive regression model. The secondary learner is defined as a linear function of certain variables deemed interesting for cost-sensitivity. These variables do not necessarily need to be the same as in the already trained model. An efficient optimization algorithm is achieved through iteratively reweighted least squares using the asymmetric cost function. The obtained results become interpretable through bootstrapping, enabling decision makers to distinguish important variables for cost-sensitivity as well as facilitating statistical inference. Applying different cost functions and various initial cost-insensitive learning methods on several public datasets consistently yields a significant reduction in the average misprediction cost, illustrating the excellent performance of our approach.

1. Introduction

Predictive models are increasingly being used to optimize decision-making. These modern-day applications of predictive data mining techniques, such as classification and regression, are often of a cost-sensitive nature [1–3]. This implies that different deviations from a correct prediction (over- and underprediction errors) entail different consequences and costs, possibly based on contextual information. In many applications, the goal is therefore to minimize the cost incurred through decisions rather than the prediction error.

A classical example of such a cost-sensitive classification problem is fraud detection, where the cost resulting from a false positive is very different from that of a false negative [4]. A typical example for cost-sensitive regression can be found in stock or inventory management, where keeping 100 extra units in stock without selling them involves different costs compared to failing to meet demand by 100 units. Whilst the prediction errors are the same in magnitude, the costs related to the decision-making process are considerably different for under- or overpredictions. Traditional linear regression models are not suitable for this example since they assume that the costs for over- and underpredictions are equal. There exist numerous other application areas in which these cost-sensitive prediction problems appear. Some of the more popular application domains are summarized in Table 1.

Although many cost-sensitive learning methods for classification problems have been proposed in literature, there exist only very few solutions for regression problems [5]. In general, two approaches have been proposed to obtain cost-sensitive predictions in regression problems.

The first approach works by integrating costs within the learning objective of the predictive model and hence deal with the asymmetric costs during model learning. This approach is sometimes called the direct approach or *predict-and-optimize* approach [6–8]. The second approach is a post hoc approach or *predict-then-optimize* approach: first, a predictive model is built to maximize its predictive power and then decisions are made based on the model's predictions and the costs associated with these decisions. Hence, in this approach costs are only considered in a post hoc manner. In Vanderschueren et al.

[*] Corresponding author at: University of Antwerp - imec, Department of Mathematics, Middelheimlaan 1, Antwerp 2020, Belgium.
E-mail addresses: Thomas.Decorte@uantwerpen.be (T. Decorte), j.raymaekers@maastrichtuniversity.nl (J. Raymaekers), Tim.Verdonck@uantwerpen.be (T. Verdonck).

https://doi.org/10.1016/j.dss.2023.114024
Received 14 October 2022; Received in revised form 27 May 2023; Accepted 3 June 2023
Available online 10 June 2023

Table 1
Overview of common application areas of cost-sensitive learning together with references.

Cost-sensitive learning application domains	
Application domain	References
Healthcare	[3,13,14]
Forecasting	[15–17]
Customer churn/retention	[18–20]
Real estate price prediction	[21,22]
Loan charge-off forecasting	[9,10,23]
Risk prediction	[24,25]
Fraud detection	[26–28]

[8], it is shown that the best results in a classification setting were obtained when incorporating cost-sensitivity in a post hoc manner. Moreover, post hoc cost-sensitive regression models can cope with various cost structures without modifying the underlying learning method. Recently, Bansal et al. [9] and Zhao et al. [10] proposed post hoc tuning approaches to obtain cost-sensitive predictions in regression problems.

In this paper, we build on these proposals and present a more general strategy to incorporate cost-sensitivity through a one-step boosting of the cost-insensitive regression model, which minimizes the average misprediction cost under an asymmetric cost structure. A secondary learner, defined as a linear function that optimizes the average misprediction cost given a certain cost function, is used to achieve this goal. The combination of the initial regression model and the secondary learner then delivers cost-sensitive predictions. Optimal coefficients for the linear function are efficiently obtained based on the iteratively reweighted least-squares method [11]. Motivated by the low computational cost of the algorithm, a bootstrapping approach is presented to obtain confidence intervals for statistical inference. This makes the cost-sensitive regression interpretable and may provide important insights into how taking costs into account may alter the trained model or which variables are important to the cost-sensitivity of the problem. It is worth noting that, in cases where the initial regression model is trained as a linear model, the proposed one-step linear boosting is the same as training with the cost function directly. In these cases our approach is equivalent to direct (predict-and-optimize) methods leveraging a linear regression. We performed an empirical evaluation of our algorithm on several public datasets using a number of different cost functions and initial regression methods. The results show a significant reduction in the average misprediction cost on all real cases. Comparing our method to other post hoc techniques [9,10,12] and direct approaches based on LightGBM and neural network implementations demonstrated a performance improvement in average misprediction cost for our one-step boosting method.

The rest of this paper is organized as follows. We start in Section 2 with an overview of the relevant literature concerning cost-sensitive learning in a regression setting. In Section 3, the developed method for cost-sensitivity through one-step boosting, denoted as the OSB approach, is introduced and in Section 4 we present the empirical evaluation. Finally, concluding remarks and potential directions for future research are provided in Section 5.

2. Related work

Before presenting our interpretable cost-sensitive regression through one-step boosting (OSB), we briefly summarize relevant work on cost-sensitive regression.

2.1. Cost-sensitive regression

In regression problems, cost-sensitivity typically originates from asymmetric costs between over- and underpredictions or from a situation-specific cost function related to the prediction errors. Just like in classification problems, cost-sensitivity in regression problems is quite a common occurrence in real-world applications [9,21]. Despite the frequency of asymmetric costs in regression problems, it has received relatively little attention in literature. Current literature suggests two general avenues to deal with cost-sensitivity in regression problems.

The first approach focuses on the introduction of optimal predictors and the adjustment of regression techniques in order to cope with cost-sensitivity directly [29,30] (*predict-and-optimize* approach). This stream of literature has led to the introduction of new cost functions (see Section 3.4) and the statistical derivations of their optimal predictors in linear regression [22,31]. In order to obtain these optimal predictors for certain cost functions new optimization techniques such as series expansions or numerical approximations have been introduced [32–36]. For neural networks, similar approaches have been proposed to accommodate for the asymmetric cost structure [29,37,38]. Other approaches extend specific models such as the Cost-sensitive Global Model Tree (CGMT), which builds on the cost-neutral solution called Global Model Tree (GMT) by introducing cost-sensitive linear regression models in the leaves and leveraging memetic operators [23]. For time-series forecasting, a cost-sensitive strategy of this nature was presented by Van Calster et al. [15].

The second approach introduces cost-sensitivity in a post hoc manner [9,10] (*predict-then-optimize* approach). In parallel with the post hoc approach commonly used in classification, the main idea is that a generic, cost-insensitive regression model f is trained and then, in a next step, adjusted to account for the asymmetric cost structure. The initial model f can be any regression model ranging from a simple linear regression to a complex neural network. In the method introduced by Bansal et al. [9], the post hoc step consists of the tuning of predictions from the regression model f by adding a constant term α to them in order to obtain the adjusted model $f' = f + \alpha$. It is in this tuning step that the asymmetric cost function is used to find the optimal α with minimal average misprediction cost, thus the cost-sensitive loss

$$\sum_{i=1}^{n} \rho((f(\boldsymbol{x}_i) - y_i) + \alpha) \tag{1}$$

with $f(\boldsymbol{x}_i)$ the predictions from f and y_i the true responses is minimized. To achieve this a hill climbing algorithm is designed which searches for the optimal α, thus only convex cost functions with regards to the average misprediction cost are considered [9]. In Zhao et al. [10] an extended post hoc tuning method for cost-sensitive regression is proposed. It extends upon the first method of Bansal et al. [9] by using a polynomial function, rather than a constant, to adjust the predictions and make them cost-sensitive. This generalization transforms the predictions using a new function g to obtain

$$\sum_{i=1}^{n} \rho(g(f(\boldsymbol{x}_i)) - y_i) \tag{2}$$

where g is fitted as a polynomial function of the predictions. The first method then becomes a special case of the extended post hoc tuning where the polynomial function only adds a constant. Comparing the performance of the evaluated polynomial functions shows that linear functions of the predicted values perform best, and any higher order polynomial tends to overfit [10]. Hence, the suggested model using the linear tuning function is then defined as $\sum_{i=1}^{n} \rho((\alpha + \beta f(\boldsymbol{x}_i)) - y_i)$ with α and β the parameters of the linear tuning function. The linear tuning function also clearly illustrates how the extended tuning method is a generalization from the earlier work of Bansal et al. [9]. Following these studies, [12] introduced a local probabilistic reframing method for post hoc cost-sensitive regression. Rather than using a global function or method to adjust all predicted values identically to the new cost function, such as in Bansal et al. [9],Zhao et al. [10], local reframing leverages enriched soft regression models for instance-dependent optimal predictions. The method converts a traditional regression model to a soft regression model with two parameters (mean and variance) using enrichment methods [12,39]. The predicted values are then transformed based on the conditional density function to new contexts or cost functions generating instance-dependent optimal cost-sensitive predictions. Compared to the global methods of Bansal et al. [9],Zhao et al. [10], local reframing adjusts every instance differently to the cost function or cost context. Post hoc approaches to cost-sensitivity have several advantages over the direct methods, such as model reuse, flexibility to a changing environment, applicability to multiple problems and simplicity in implementation.

These works are an indication of the possibilities related to the post hoc introduction of costs for regression models [9,10,12]. In this paper we expand upon the idea of making a trained regression model cost-sensitive in a post hoc, global manner.

3. One-step boosting for cost-sensitive regression

Building on the earlier studies in post hoc cost-sensitivity for regression problems [9,10,12], we present a more general strategy to incorporate cost-sensitivity into already trained cost-insensitive models. Our one-step boosting (OSB) method focuses on post hoc global reframing or adjusting. First, we describe some preliminaries followed by our proposed approach. Then, we also present an efficient algorithm for our OSB approach. Finally, we describe various cost functions that can be used within our methodology.

3.1. Preliminaries

Consider a multivariate independent vector of variables $\boldsymbol{x}$ with size m and a univariate and continuous dependent (or response) variable y. Datasets or samples of size n observations can then be denoted as $S = \{\langle \boldsymbol{x_i}, y_i \rangle | i = 1, 2, \dots, n\}$. A regression model then tries to approximate the true function $f : X \to Y$ from a certain training sample S. This approximation or estimation is typically obtained by using a predictive learning method (such as a linear regression, decision tree or neural network). Predictions can then also be derived for unlabeled and/or new instances. In other words, for instances $\boldsymbol{x_i}$ of which the corresponding y_i is not known, the regression model yields a prediction $\hat{y}_i$. The performance of a regression model is often measured by the difference between these predictions and the true responses of training instances. Costs in regression problems are then associated with the prediction errors expressed in a cost (or loss) function. The prediction error or residual e_i of instance i from the training sample S is defined as $e_i = \hat{y}_i - y_i$ with y_i and $\hat{y}_i$ respectively the true and predicted response. Hence, an overprediction is represented by a positive residual, whereas an underprediction results in a negative residual. These residuals incur a certain cost based on a cost function ρ. The cost function ρ takes the true response y_i of an instance and $\hat{y}_i$ the predicted response as inputs and then outputs a certain cost, making the function of the form $\rho : \mathbb{R}^2 \to \mathbb{R}^+$. In regression, the model needs to be trained to minimize $E[\rho]$, which in the finite-sample S boils down to minimizing: $\frac{1}{n}\sum_{i=1}^{n} \rho(\hat{y}_i, y_i)$. This can be further simplified by assuming that the cost function only depends on the residual e_i:

$$\frac{1}{n}\sum_{i=1}^{n} \rho(e_i) \tag{3}$$

Throughout this paper, we will denote cost functions using this simplification where a residual e_i equals the difference between $\hat{y}_i$ and y_i. Note that some cost functions cannot be defined in terms of the residuals. Eq. (3) represents the average misprediction cost based on the cost function ρ, which is often used to illustrate the performance of a certain model on that cost function [9,10].

3.2. Methodology

In essence, the method of Bansal et al. [9] can also be seen as a special case of one-step boosting. More specifically, starting with a model f that is trained with an arbitrary loss function (such as the least squares), the new response is then constructed as follows: $(f(x_i) - y_i)$, which equals the residuals of the existing model. Using these residuals, a 'weak learner' (i.e. a constant α in the case of Bansal et al. [9]) is leveraged to improve on the existing model (see Eq. (1)). Therefore, this methodology can be described as a boosting approach of the original regression model f using a constant α to obtain cost-sensitive predictions. In the extended proposal of Zhao et al. [10], we can then no longer speak of boosting, as it involves the transformation of the predictions by leveraging a tuning function rather than the introduction of a 'weak learner'. Instead of the generalization by Zhao et al. [10], we propose a generalization of Bansal et al. [9] for post hoc cost-sensitivity that continues to leverage the boosting framework [40]. Later we will also propose an efficient algorithm and tools for statistical inference, leading to interpretable results.

First, as with other post hoc methods, a regression model f is trained without the consideration of the asymmetric cost structure. This regression model can be fit using any learning algorithm such as linear regression, random forest or a neural network. Next, we want to use a secondary learner f_1 to incorporate cost-sensitivity into the previously trained model f. Given an asymmetric cost function ρ, the optimization of the learner f_1 would be

$$\frac{1}{n}\sum_{i=1}^{n} \rho(f_1(\boldsymbol{x_i}) - (f(\boldsymbol{x_i}) - y_i)), \tag{4}$$

as to minimize the average misprediction cost of the final prediction. For an overview of suitable and popular cost functions and their properties, we refer to Section 3.4. Eq. (4) can be further simplified to $\frac{1}{n}\sum_{i=1}^{n} \rho(f_1(\boldsymbol{x_i}) - y_i^*)$, where $y_i^* = f(\boldsymbol{x_i}) - y_i$ denotes the residuals of the trained learner f. Since the boosting occurs in a single step, the procedure is denoted one-step boosting. This can be seen as an extension of the method of Bansal et al. [9], where a constant function $\alpha \in \mathbb{R}$ for f_1 is used, see Eq. (1). Instead of a constant function, we propose to utilize a linear function of the predictors with an intercept $f_1(\boldsymbol{x_i}) = \alpha + \boldsymbol{\beta x_i}$.

In order to find the optimal coefficients of the linear function f_1, we need to minimize Eq. (4), which is a standard M-estimation problem. M-estimation originated from the research field of robust statistics (see Huber [41],Hampel et al. [42],Huber [43],Rousseeuw and Leroy [44] for more information), but we consider it more generally without necessarily plugging in robust ρ functions. Considering the earlier simplification in Eq. (3), it can be seen that minimizing the expected value of any cost function leads to an M-estimator minimizing the average misprediction cost $\frac{1}{n}\sum_{i=1}^{n} \rho(e_i)$. When this concept is applied in the boosting step of our method, we obtain an M-estimator minimizing $\frac{1}{n}\sum_{i=1}^{n} \rho(r_i)$ with $r_i = (\alpha + \boldsymbol{\beta x_i}) - y_i^*$ the difference between y_i^*, the residuals from the trained learner f from the first step, and the predictions from f_1 of these residuals. If the cost function ρ is then (almost everywhere) differentiable, we can efficiently solve for f_1 by applying iteratively reweighted least squares (IRLS) [11]. More specifically, when the condition of an (almost everywhere) differentiable cost function is met, then optimizing with respect to β is M-estimation with a new response variable. With the differentiated cost function ψ we obtain

$$\sum_{i=1}^{n} \frac{\psi(r_i)}{r_i} r_i \boldsymbol{x_i} = 0 \tag{5}$$

with $r_i = (\alpha + \boldsymbol{\beta x_i}) - y_i^*$ and y_i^* the residuals of the trained learner f. Note that the cost function ρ needs to be convex to guarantee a unique solution. This restriction still allows for a wide applicability of the proposed OSB method, since most cost functions, both in practice and literature, tend to be convex (e.g., see Fig. 1). The optimal coefficients of the linear function f_1 in combination with the trained model f from the first step then give cost-sensitive predictions with relation to the cost function ρ.

3.3. Algorithm

The OSB algorithm (for detailed overview in pseudo-code, see Algorithm 1) takes five inputs: the residuals of the initial regression model y_i^*, a convex cost function ρ, the maximum number of iterations k, a precision parameter ϵ and a training sample S. Once a cost-insensitive regression model f has been trained on a sample $S = \{\langle \boldsymbol{x_i}, y_i \rangle | i = 1, 2, \ldots, n\}$, the residuals $y_i^* = f(\boldsymbol{x_i}) - y_i$ serve as the primary input to the cost-sensitive boosting step. If the cost function satisfies the assumptions detailed in the previous Section 3.2 (convexity and almost everywhere differentiability), then convergence to the global optimum is guaranteed. The values for the maximum number of iterations k and the precision parameter ϵ are user-defined. The last input is the training data used in order to make the regression model cost-sensitive. Here, we opt to use the same training sample S as in the first step. However, it should be noted that this is not a requirement as any variables deemed useful for cost-sensitivity can be used. In practice, there can be cases where preferably a specific set of variables, not used in the initial regression f, should be used to introduce cost-sensitivity. Thus, the second boosting step does not necessarily need to happen with the same variables as the training of the initial regression model. The only requirement for the sample in the second step is that there are observations of the desired variables available for every residual y_i^*. Furthermore, we assume that the training sample S in the pseudo-code contains a column of 1's to represent the intercept of the linear function f_1.

Algorithm 1: One-Step Boosting for Cost-sensitive Regression

```
Input: Residuals $\{y_i^*\}_{i=1}^N$ of trained cost-insensitive regression
       model $f$
       Cost function $\rho$
       Training sample $S$ with instances $\{\boldsymbol{x_i}\}_{i=1}^N$ with $m$ variables
       Maximum number of iterations $k$
       Precision parameter $\epsilon$
  Initialization
  $PreviousCost = \frac{1}{n}\sum_{i=1}^{n} \rho(y_i^*)$  ▷ Calculate average misprediction cost of $y_i^*$
  Start Iteratively Reweighted Least Squares
  for $j$ in range (1:k) do
    if $j = 1$ then
        $r_i = \boldsymbol{x_i} * \boldsymbol{\beta_0} - y_i^*$
    end
    $w_i = \frac{\psi(r_i)}{r_i}$  ▷ Calculate weights from $\psi$ see equation (5)
    $\boldsymbol{\beta} \leftarrow LinearRegression(\sqrt{w_i}\boldsymbol{x_i}, \sqrt{w_i}y_i^*)$  ▷ Obtain coefficients of linear function $f_1$
    $r_i = \boldsymbol{x_i} * \boldsymbol{\beta} - y_i^*$  ▷ Obtain residuals from predictions
    $CurrentCost = \frac{1}{n}\sum_{i=1}^{n} \rho(r_i)$  ▷ Average misprediction cost of $r_i$
    if $(PreviousCost - CurrentCost) > (\epsilon * \sigma_{(y_i^*)})$ then
      $PreviousCost = CurrentCost$
      continue  ▷ Go to next iteration
    else
        break  ▷ Exit for loop and stop iteration
    end
  end
Output: Optimal coefficients $\boldsymbol{\beta}$ of the linear function $f_1$
```

In the initialization step the algorithm calculates the average misprediction cost using the starting residuals, denoted as $PreviousCost$. Thus, the $PreviousCost$ variable represents the average misprediction cost of the cost-insensitive regression model f on the cost function ρ. Next, the iteratively reweighted least squares (IRLS) is applied in search of the optimal coefficients. The IRLS is iterated until convergence of the average misprediction cost is obtained (with a maximum of k runs). Due to the nature of IRLS in combination with the convex cost function, the algorithm will continue to find a lower cost which is why convergence up until a certain precision is used. When the difference between the costs of the current and previous iteration is not larger than the standard deviation of the starting residuals multiplied by the user-defined factor of precision ϵ (in our implementation $1e^{-6}$), then the algorithm has sufficiently converged to the optimum. In our implementation convergence happens typically within 10 to 30 iterations of the algorithm. The user-defined k maximum number of iterations is therefore mainly implemented as a fail-safe stop for potentially badly specified cost functions. In the first iteration, when the coefficients $\boldsymbol{\beta}$ of the linear regression are not available, the variables r_i are calculated using a vector of zeros for the coefficients $\boldsymbol{\beta}$ (represented by $\boldsymbol{\beta_0}$ in the pseudo-code). Hence, the r_i variables in the first iteration are equal to the negative residuals y_i^* of the initial regression f. These r_i residual variables are subsequently used in the weight function ψ (derivative of the cost function divided by r_i). In a following step the weights are leveraged in a linear regression applied to the pairs of instances $(\sqrt{w_i}\boldsymbol{x_i}, \sqrt{w_i}y_i^*)$ resulting in the coefficients $\boldsymbol{\beta}$ for that iteration. In the last step, the final residuals are calculated of the OSB ensemble learner followed by a convergence check using the precision parameter ϵ and the standard deviation $\sigma_{(y_i^*)}$ of the initial residuals. When the algorithm converges, the last $\boldsymbol{\beta}$ will be the optimal coefficients for the linear function f_1.

Note that in cases where f is trained as a linear model, the one-step linear boosting is the same as training with the cost function ρ directly. Hence, when f is a linear model, the linear boosting step for cost-sensitivity is equivalent to applying a direct (predict-and-optimize) approach with a linear regression. This property gives the OSB approach a distinct advantage over other post hoc approaches. Furthermore, due to the boosting framework, the model f_1 can provide insight into how taking costs into account can alter a regression model, provided f_1 is not a complex learner. As we use a linear function for f_1, we can leverage bootstrapping to make confidence intervals of the coefficients. The confidence intervals provide insights into the significance of the coefficients related to the cost-sensitivity, making our OSB method interpretable. Moreover, the interpretation of the linear model is fairly straightforward, which proves to be useful in practical settings. Another advantage of using a linear model for f_1 is the fast training time, amplifying the ease of use of the OSB algorithm in practical settings as it speeds up the process of finding a correct cost function and allows for a quick cost function calibration, which is especially useful in dynamic environments where costs change frequently.

3.4. Cost functions

In practice, any cost function ρ is possible based on the context. For example, a Huber loss function [45]:

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(\hat{y} - y)^2, & \text{for } |\hat{y} - y| \le \delta \\ \delta(|\hat{y} - y| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{6}$$

However, it is when overpredictions and underpredictions face different costs, yielding asymmetric costs in the cost function $\rho(e) \neq \rho(-e)$, that cost-sensitivity becomes relevant. In traditional algorithms (e.g. least-squares linear regression) cost functions (or loss functions) are symmetric such as the squared error loss function $\rho(e) = e^2$, meaning that under- or overpredictions face the same costs $\rho(e) = \rho(-e)$. These conventional methods using symmetric cost functions for optimization are thus insufficient to deal with cost-sensitive problems. Granger [46] introduced several cost functions and suggested to add a constant bias term to the predictor to account for the generalized cost functions. These introduced cost functions are the LinLin (asymmetric linear) (8) and QuadQuad (asymmetric quadratic) functions (7), often considered as generalizations of the absolute error loss $\rho(e) = |e|$ and squared error (quadratic) loss functions $\rho(e) = e^2$ respectively. In Eqs. (7) and (8) the parameters a and b are two positive constants that regulate the asymmetry of the cost function.

$$\rho(e) = \begin{cases} a(e)^2, & \text{if } e \geq 0 \\ b(e)^2, & \text{if } e < 0 \end{cases} \tag{7}$$

$$\rho(e) = \begin{cases} a|e|, & \text{if } e \geq 0 \\ b|e|, & \text{if } e < 0 \end{cases} \tag{8}$$

Christoffersen and Diebold [33,34] indicate that in general (e.g. when non-normality is present) closed-form solutions, such as the well-known solution for the least squares estimator $\hat{\beta} = (X^T X)^{-1} X^T y$, do not exist for the LinLin and QuadQuad cost functions. However, the predictor in these cost functions for which a closed form solution does not exist, can be approximated using numerical solutions [33, 34]. Furthermore, the severe difference in costs between over- and underpredictions in the real estate assessment of taxable properties lead to the introduction of the LinEx loss function by Varian [22]. This cost function is approximately linear for underpredictions and approximately exponential for overpredictions (see Fig. 1(c)) with $a \neq 0$ and $b > 0$:

$$\rho(e) = b(\exp(ae) - ae - 1) \tag{9}$$

From Eq. (9) it can be deduced that the parameter b is a scaling parameter and parameter a controls the asymmetry and shape of the cost function. Hence, if parameter a is negative then an ExLin function (exponential for underprediction and linear for overprediction see Fig. 1(d)) would be obtained. Unlike the LinLin and QuadQuad cost functions the LinEx function has general closed-form solutions under certain conditions [31]. Even under non-normality closed-form solutions for the LinEx cost function exist or the solution can be approximated using series expansion [32,35]. In later literature several expansions upon the LinEx cost function have been introduced [30]. Also specific case studies have been done, such as Cain and Janssen [21] who applied the LinEx cost functions to a real estate price prediction problem and compared the results with LinLin and QuadQuad cost functions.

In Fig. 1 the different cost functions discussed above are shown. Both the LinLin and QuadQuad functions have 1 and 5 for the values of a and b respectively. The LinEx and ExLin functions are made with $a = 0.4$ ($a = -0.4$ for ExLin) and $b = 1$. As can be seen the costs under the asymmetric cost functions can increase rapidly for certain prediction errors, resulting in sub-optimal predictions from regression models that optimize symmetric errors during training.

4. Empirical evaluation

In this section we will illustrate the benefits of using our one-step boosting method to introduce cost-sensitivity into a regression model. First, we will illustrate in detail the practical implementation of our OSB approach. Then, the results from our empirical validation on various real datasets will be discussed. Next, an example will be included to illustrate the output of the algorithm as well as the possible interpretation that can follow from bootstrapping OSB. Furthermore, a comparison with the methods of Bansal et al. [9],Zhao et al. [10], Hernandez-Orallo [12] and a predict-and-optimize approach based on a LightGBM implementation will be made.

4.1. Implementation

As described in Section 3, various cost functions to incorporate cost-sensitivity can be used in our OSB method. In our implementation we evaluate the algorithm with the LinLin, QuadQuad and ExLin cost functions. To make the results more comparable, the underpredictions are always considered to be more costly than the overpredictions. Hence, in the LinLin and QuadQuad cost functions the parameter b is larger than the parameter a (see Eqs. (7),(8)), illustrated in cost ratios of the form $a : b$. In the cost functions we keep the cost for overpredictions a equal to 1, meaning that overpredictions are not more costly than their intrinsic value in that cost function, whilst we evaluate different costs b for underpredictions. Note that the ExLin cost function has a negative value for a in the tables as it would be used in the earlier introduced formula. Due to the more extreme nature of the ExLin cost function, we will only evaluate it for one specific cost structure. Following Algorithm 1, the only requirements are that these cost functions are (almost everywhere) differentiable and convex to achieve a unique optimum, which the evaluated cost functions meet.

The algorithm takes, next to the cost function, two other main inputs: the residuals of the initial regression model f and a training sample with predictors. For simplicity in implementation, we use the same training sample of predictors as was used in the training of the initial regression model f. Note that this is not necessary as a different set of variables can be used. To generate the residuals y_i^* several base regression methods are used as f. The use of different regression methods in the first step of the boosting method allows us to validate that OSB functions with any regression method. The evaluated base regression methods f are: the least-squares linear regression (LR), linear model trees [10,35] (MT), neural network (NN), random forest (RF), and LightGBM (LGBM) [47]. The algorithm is implemented in Python, including all of the first step base regression methods. The default parameters for the regression methods f of the used Python packages are retained. The scikit-learn (both LinearRegression and RandomForestRegressor) [48], TensorFlow (Keras) [49], linear-tree,[1] and LightGBM [47] packages are used for the LR, RF, NN, MT and LGBM methods respectively. The code implementation of our algorithm is made available on GitHub.[2]

Similarly to Bansal et al. [9],Zhao et al. [10], we use the average misprediction cost as a measure of performance for the cost-sensitive regression problems. A 2 × 5 cross-validation with the average misprediction cost is run for every evaluated combination of dataset, base regression method f and cost function ρ with certain cost ratio. Thus, generating both robust training performance measures of the complete boosting method (step 1 and boosting step), as well as robust testing performance measures to evaluate the generalization of the model to new data.

4.2. Data

Experiments were run generating average misprediction costs for four different regression datasets: Abalone, Bank (8FM), House (8L), and KC House. The first three of these datasets are made available on the DELVE.[3] (Data for Evaluating Learning in Valid Experiments) repository of the University of Toronto or on the UCI repository[4] The last dataset, KC House, is made available by the Center for Spatial Data Science[5] at the University of Chicago. As the main goal of the experiment is to evaluate OSB, only minimal preprocessing is done to each dataset. This mainly included the deletion of certain variables such as ID or date. Furthermore, certain variables were characterized by skewed distributions (e.g. price of a house). Logarithmic transformations are then used to reduce or even remove this skewness. Table 2 contains the relevant information for each dataset.

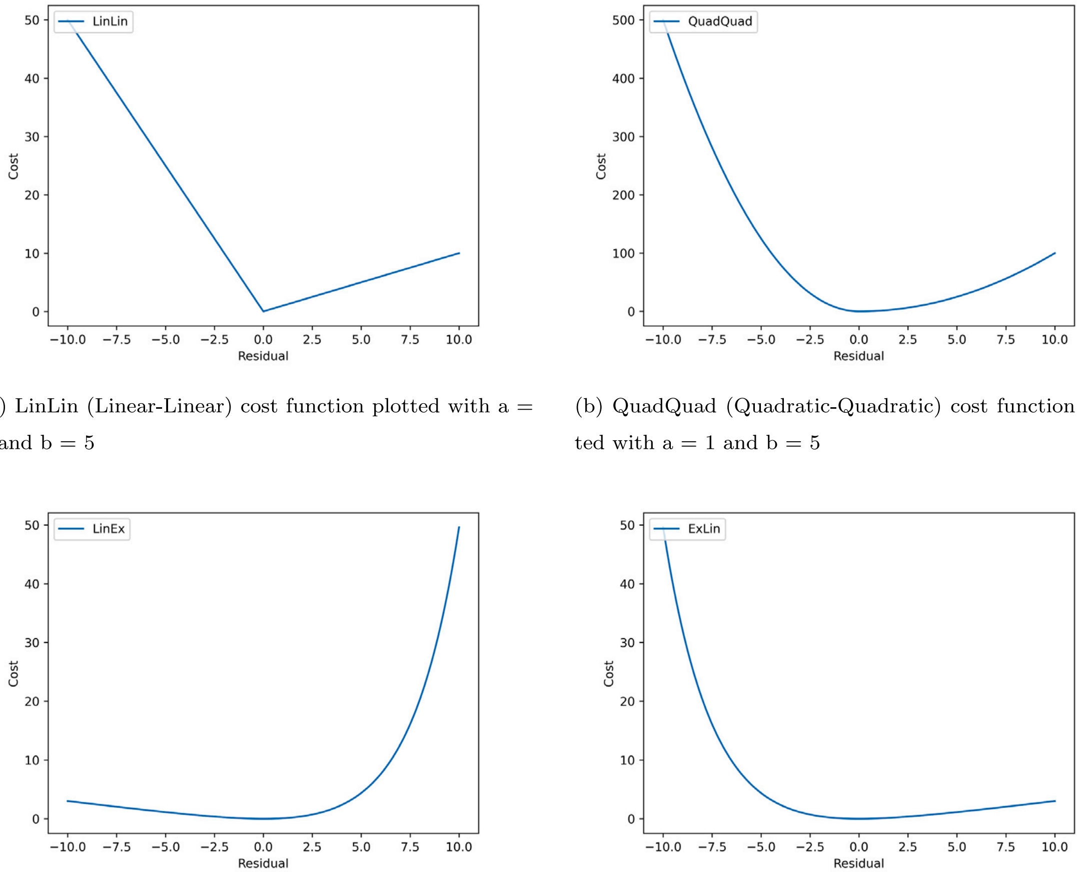
[1] https://github.com/cerlymarco/linear-tree

[2] https://github.com/ThomasDecorteUA/Cost_Sensitive_Regression

[3] https://www.cs.toronto.edu/∼delve/data/datasets.html

[4] https://archive.ics.uci.edu/ml/datasets.php

[5] https://geodacenter.github.io/data-and-lab//KingCounty-HouseSales2015/



(a) LinLin (Linear-Linear) cost function plotted with a = 1 and b = 5

(b) QuadQuad (Quadratic-Quadratic) cost function plotted with a = 1 and b = 5

(c) LinEx (Linear-Exponential) cost function plotted with a = 0.4 and b = 1

(d) ExLin (Exponential-Linear) cost function plotted with a = -0.4 and b = 1

Fig. 1. Plots of various cost functions.

Table 2
Datasets used in Section 4 and related appendices. We show the size, the number of attributes besides the response, the mean of the response and the standard deviation of the response.

Datasets for empirical evaluation				
Dataset	Size	Attributes	μ Response	σ Response
Abalone	4177	8	2.25	0.32
Bank (8FM)	8192	8	0.16	0.15
House (8L)	22784	8	10.52	0.72
KC House	21613	9	13.05	0.51

4.3. Results

The results of the OSB method for the different datasets are summarized in Tables 3 and 4 for the LightGBM initial regression and the NN base regression method respectively. For results on other base regression methods we refer to Appendix A.1. The tabulated results contain the name of the dataset in column 1 and indicate the specific cost function (the LinLin, QuadQuad and ExLin cost functions are represented as LL, QQ and EL respectively) as well as the cost ratio (expressed as $a : b$) in the second column (CF $a : b$). The reported numbers in columns 3,4,5, and 6 represent the average misprediction costs from the 2 × 5 cross-validation. The column "Initial" refers to the average misprediction cost of the base regression method f, i.e. the costs when only the first step of the algorithm is applied and no cost-sensitivity is considered. The "OSB" results then contain the average misprediction costs after the entire one-step boosting algorithm is applied. A percentage decrease between both average misprediction cost measures is added to aid in interpretation of the results. These results are made available both for the train and test data following the 2 × 5 cross-validation.

The OSB method always yields significantly better predictions in every cost function, dataset, and cost ratio, except for one case where there is no substantial improvement (Bank(8FM) with EL $-3 : 2$). In this particular case, cost-sensitivity is not useful as the cost function is badly tuned for the residuals and the predictions of the base regression method are already very good (see also Section 4.4). As expected, the initial costs are, across all results, higher than the post hoc cost. It is clearly seen that there is always a significant decrease between both costs. As the cost ratios increase and the initial costs follow, a much more gentle increase in post hoc costs can be observed. For example, the initial costs between a LinLin cost function with $b = 10$ and $b = 20$ is almost double, whereas the post hoc costs only increase with a small amount. Aside from the one case, this trend is true for every cost function, cost ratio, dataset, and base regression method f. Note that the average misprediction costs are in each evaluated cost function calculated using the residuals. Therefore, the scale of the costs follows the scale of the residuals and hence also the scale of the response variable.

On the testing data, a similar decrease in costs is observed as on the training data. Even when we compare the different base regression methods f, the decrease in costs (percentage decrease) remains similar across most datasets and cost functions.

Using NN as a base regression (see Table 4), we obtain higher costs on most dataset/cost function combinations (except for the Abalone

Table 3
Performance of OSB for cost-sensitivity with a LightGBM initial regression (LGBM) expressed in the average misprediction cost on training and testing data using 2 × 5 cross-validation on various datasets with different cost functions and cost ratios.

One-step Boosting with LightGBM regression							
Dataset	CF $a : b$	Initial train	OSB train	% Train	Initial test	OSB test	% Test
Abalone	LL 1:10	0.600	0.233	61.1	0.822	0.344	58.2
Abalone	LL 1:20	1.146	0.272	76.3	1.570	0.421	73.2
Abalone	LL 1:50	2.782	0.324	88.3	3.814	0.555	85.5
Abalone	LL 1:100	5.510	0.367	93.3	7.553	0.676	91.1
Abalone	QQ 1:10	0.118	0.051	57.0	0.224	0.099	56.2
Abalone	QQ 1:20	0.227	0.065	71.5	0.431	0.130	69.9
Abalone	QQ 1:50	0.554	0.088	84.1	1.052	0.186	82.3
Abalone	QQ 1:100	1.100	0.110	90.0	2.087	0.246	88.2
Abalone	EL -3:2	0.200	0.182	9.0	0.402	0.340	15.6
Bank (8FM)	LL 1:10	0.087	0.036	58.6	0.120	0.053	55.6
Bank (8FM)	LL 1:20	0.166	0.042	74.9	0.229	0.065	71.6
Bank (8FM)	LL 1:50	0.404	0.049	87.9	0.556	0.082	85.3
Bank (8FM)	LL 1:100	0.800	0.053	93.3	1.102	0.098	91.1
Bank (8FM)	QQ 1:10	0.003	0.001	52.3	0.005	0.003	50.4
Bank (8FM)	QQ 1:20	0.005	0.002	68.5	0.010	0.003	65.7
Bank (8FM)	QQ 1:50	0.012	0.002	83.0	0.024	0.005	79.9
Bank (8FM)	QQ 1:100	0.023	0.002	89.7	0.047	0.006	86.9
Bank (8FM)	EL -3:2	0.004	0.004	0.3	0.008	0.008	0.9
House (8L)	LL 1:10	1.303	0.483	62.9	1.463	0.561	61.7
House (8L)	LL 1:20	2.487	0.563	77.4	2.793	0.664	76.2
House (8L)	LL 1:50	6.040	0.666	89.0	6.784	0.817	88.2
House (8L)	LL 1:100	11.961	0.744	93.8	13.436	0.947	93.3
House (8L)	QQ 1:10	0.609	0.220	63.9	0.790	0.294	62.8
House (8L)	QQ 1:20	1.178	0.272	76.9	1.529	0.372	75.6
House (8L)	QQ 1:50	2.886	0.354	87.7	3.746	0.505	86.5
House (8L)	QQ 1:100	5.732	0.426	92.6	7.441	0.633	91.5
House (8L)	EL -3:2	2.334	0.742	68.3	5.311	0.988	81.4
KC House	LL 1:10	1.278	0.539	57.8	1.347	0.570	57.7
KC House	LL 1:20	2.440	0.626	74.4	2.571	0.665	74.1
KC House	LL 1:50	5.926	0.727	87.7	6.245	0.779	87.5
KC House	LL 1:100	11.736	0.800	93.2	12.367	0.868	93.6
KC House	QQ 1:10	0.467	0.237	49.3	0.520	0.265	49.7
KC House	QQ 1:20	0.889	0.301	66.2	0.991	0.338	65.9
KC House	QQ 1:50	2.157	0.396	81.7	2.404	0.450	81.3
KC House	QQ 1:100	4.269	0.474	88.9	4.759	0.545	88.6
KC House	EL -3:2	0.970	0.802	17.3	1.116	0.899	19.4

dataset). The cause of these higher costs can be found in the NN model performance on these datasets compared to the LGBM performance. The NN performs worse than the LGBM both on training and testing data, leading to worse initial predictions. This difference in performance is clearly observed in the initial costs. Even then, the secondary learner f_1 in the OSB algorithm succeeds in minimizing this difference quite well, as can be seen in the post hoc costs. Although a lower performance for the NN base regression is obtained, the percentage decrease in costs remains similar to the LGBM case. Note that the NN base regression was run using default parameters and hence better results could be obtained by doing advanced hyperparameter tuning. Furthermore, when the base regression overfits the training data, then also the boosting step will give a worse generalization. These conclusions follow quite logically from the boosting nature of the algorithm. Similarly, in Zhao et al. [10] the linear regression base method achieved the worst performance on the training data comparative to the MT and NN base models. However, the linear regression then outperformed both on the testing data, indicating the importance of an appropriate model complexity for each problem [10]. In our case one base regression model always outperforms the other on most datasets, yet the conclusion that the initial model needs an appropriate model complexity and fit to the data remains the same.

In Appendix A.1 we include simulations with other base regression methods, but similar results are obtained. The same conclusions also hold when overpredictions are more costly than underpredictions.

4.4. Inference on cost-sensitivity

The example below shows the result of applying the OSB method on the KC House dataset using the LinLin cost function with b = 10, a = 1 and the LGBM base regression method. After completing the cost-sensitive boosting step of the algorithm, we obtain the following linear function:

$$\begin{aligned}\hat{y}_{OSB} = 0.467 + 0.003x_1 - 0.006x_2 - 0.050x_3 + 0.010x_4 - 0.007x_5 \\ -0.028x_6 + 0.008x_7 + 0.001x_8 - 0.001x_9\end{aligned} \qquad (10)$$

While the coefficients of the variables may appear relatively small, it is important to note that the response variable is log-scaled. The coefficients and confidence intervals, obtained through bootstrapping, can then be used to explain the impact and significance of the independent variables on the introduction of cost-sensitivity in the model.

In Table 5 the confidence intervals for the coefficients in the linear model can be found from bootstrapping the boosting step. The significant variables are indicated in bold. Thus, from Table 5 we can deduce that the variables *Waterfront*, *Age* and *Age_Rnvtd* are not significant and hence are not relevant to the introduction of cost-sensitivity into the trained base regression model. In addition, we can see that some variables, such as *Floors* and *Condition*, have a negative coefficient, indicating that the number of *Floors* the property has and the *Condition* of the house (integer variable between 1-14) have a negative impact on the predictions, given the other predictors. Whilst the positive coefficients of variables, such as *Grade*, have the opposite effect, given the other predictors as fixed. As can be seen, these results lead to insights relating to the variables as well as the specific cost function for the decision-making process behind the cost-sensitive model.

In the case where cost-sensitivity might not be useful, the algorithm will only give a small improvement or possibly no improvement (all β of the linear function are zero). This might occur when the predictions of the base regression method are already very good or when the cost

Table 4
Performance of OSB for cost-sensitivity with a neural network base regression (NN) expressed in the average misprediction cost on training and testing data using 2 × 5 cross-validation on various datasets with different cost functions and cost ratios.

One-step Boosting with NN regression							
Dataset	CF *a* : *b*	Initial train	OSB train	% Train	Initial test	OSB test	% Test
Abalone	LL 1:10	0.852	0.318	62.7	0.854	0.326	61.8
Abalone	LL 1:20	1.628	0.366	77.5	1.631	0.381	76.6
Abalone	LL 1:50	3.955	0.433	89.1	3.962	0.466	88.2
Abalone	LL 1:100	7.834	0.484	93.8	7.847	0.546	93
Abalone	QQ 1:10	0.242	0.095	60.9	0.245	0.102	58.5
Abalone	QQ 1:20	0.467	0.118	74.8	0.472	0.130	72.4
Abalone	QQ 1:50	1.142	0.154	86.5	1.153	0.184	84
Abalone	QQ 1:100	2.266	0.186	91.8	2.288	0.245	89.3
Abalone	EL -2:3	0.449	0.338	24.6	0.455	0.355	21.9
Bank (8FM)	LL 1:10	0.138	0.075	45.8	0.140	0.072	48.7
Bank (8FM)	LL 1:20	0.263	0.111	57.7	0.266	0.104	61.1
Bank (8FM)	LL 1:50	0.639	0.210	67.1	0.647	0.188	70.9
Bank (8FM)	LL 1:100	1.265	0.368	70.9	1.282	0.322	74.9
Bank (8FM)	QQ 1:10	0.006	0.003	56.7	0.006	0.003	55.5
Bank (8FM)	QQ 1:20	0.012	0.003	72	0.012	0.004	71
Bank (8FM)	QQ 1:50	0.029	0.004	85.3	0.030	0.005	84.6
Bank (8FM)	QQ 1:100	0.058	0.005	91.3	0.059	0.005	90.8
Bank (8FM)	EL -2:3	0.008	0.007	3.4	0.008	0.008	2.5
House (8L)	LL 1:10	1.800	0.677	62.4	1.812	0.687	62.1
House (8L)	LL 1:20	3.439	0.788	77.1	3.463	0.804	76.8
House (8L)	LL 1:50	8.358	0.934	88.8	8.415	0.958	88.6
House (8L)	LL 1:100	16.556	1.049	93.7	16.669	1.086	93.5
House (8L)	QQ 1:10	1.122	0.427	62	1.137	0.440	61.3
House (8L)	QQ 1:20	2.168	0.534	75.4	2.196	0.554	74.8
House (8L)	QQ 1:50	5.304	0.704	86.7	5.374	0.741	86.2
House (8L)	QQ 1:100	10.531	0.858	91.9	10.669	0.916	91.4
House (8L)	EL -2:3	8.011	1.521	81	7.153	1.549	21.6
KC House	LL 1:10	1.381	0.771	44.2	1.389	0.729	47.5
KC House	LL 1:20	2.632	0.908	65.5	2.646	0.851	67.8
KC House	LL 1:50	6.384	1.441	77.4	6.419	1.294	79.8
KC House	LL 1:100	12.639	2.234	82.3	12.707	1.933	84.8
KC House	QQ 1:10	0.552	0.279	49.3	0.561	0.292	47.9
KC House	QQ 1:20	1.048	0.353	66.4	1.066	0.376	64.7
KC House	QQ 1:50	2.539	0.461	81.8	2.582	0.516	80
KC House	QQ 1:100	5.024	0.551	89	5.107	0.661	87.1
KC House	EL -2:3	1.231	0.947	23	1.343	0.992	26.1

Table 5
Confidence intervals following the 100 times bootstrapping of the second cost-sensitive step in the algorithm using the KC House data, a LinLin cost function with $b = 10$ and $a = 1$ and the LGBM base regression method.

Confidence Intervals of KC House Dataset			
Variable	Prediction	Lower limit CI	Upper limit CI
Intercept	0.467	0.458	0.475
Bedrooms	0.003	0.002	0.004
Bathrooms	−0.006	−0.008	−0.004
Floors	−0.050	−0.051	−0.048
Waterfront	0.010	−0.017	0.021
View	−0.007	−0.009	−0.005
Condition	−0.028	−0.030	−0.027
Grade	0.008	0.007	0.010
Age	0.001	−0.001	0.001
Age_Rnvtd	−0.001	−0.001	0.001

Table 6
Performance expressed in average misprediction cost using 2 × 5 cross-validation on the Bank (8FM) testing data and the ExLin cost function with LGBM base regression.

One-Step Boosting - Bank (8FM) Testing Data				
Cost function	Cost ratio	Initial cost	post hoc cost	% Decrease
ExLin	−3:2	0.00807	0.00800	0.9
ExLin	−20:2	0.41645	0.36610	12.1

function is badly tuned for the residuals. An example of this can be found in Table 6, which is an extended example from a result out of Table 3. This illustrative example is computed on the Bank (8FM) training dataset using the ExLin cost function. To properly interpret the results of the Bank (8FM) dataset, one has to take into account that the response variable values range between 0 and 0.802. Two different calibrations of the ExLin cost function are shown, namely one with $a = -3$ and $b = 2$ and one where $a = -20$ and $b = 2$. The second calibration is thus a lot more extreme considering the exponential nature of the ExLin cost function.

Under both calibrations of the cost function, the underlying regression method already delivers extremely good results. In the −3 : 2 cost function calibration, the boosting step does not find any significant improvement and all $\boldsymbol{\beta}$ are close to zero. Even in the extreme case, the boosting step only improves performance, based on average misprediction cost, by a relatively small percentage (12.1%) compared to the results obtained on other datasets with less extreme cost functions. Furthermore, the confidence intervals of the variables in the −20 : 2 calibration (many containing zero) as well as the coefficients (many close to zero) indicated that most of the variables are not significant in reducing costs. Thus, cost-sensitive boosting may not be necessary in this particular case with this cost function. This type of results can be helpful to distinguish for which cost functions and datasets cost-sensitivity needs to be incorporated and for which it can be disregarded.

4.5. Comparison with other methods

We now compare the OSB method with the most relevant competing alternatives. In particular, we compare OSB with the method introduced by Bansal et al. [9] (BSZ), the extended tuning method of Zhao et al. [10] (BSZ-EXT), a direct method (Direct) based on the

Table 7
Performance of the OSB, the method introduced by [9] (BSZ), the extended version by [10] (BSZ-EXT), the local reframing of [12] (HER) and a direct cost-sensitive approach based on a LGBM implementation. All post hoc methods use LGBM as an initial regression. The performance is expressed in the average misprediction cost (multiplied by 100) on testing data of various datasets using 2 × 5 cross-validation on different cost functions calibrations. The cost ratios $a : b$ always sum up to one as described in Section 4.5.

Comparison of methods on testing data with LGBM base regression							
Dataset	Cost function	Initial	Direct	BSZ	BSZ-EXT	HER	OSB
Abalone	LL 1:10	7.413	4.134	3.704	3.684	3.830	**3.140**
Abalone	LL 1:20	7.414	2.453	2.437	2.417	2.677	**2.025**
Abalone	LL 1:50	7.415	1.399	1.293	1.283	1.658	**1.096**
Abalone	LL 1:100	7.415	0.956	0.778	0.764	1.178	**0.679**
Abalone	QQ 1:10	2.018	1.156	1.108	1.101	1.160	**0.898**
Abalone	QQ 1:20	2.034	0.875	0.798	0.791	0.877	**0.621**
Abalone	QQ 1:50	2.044	0.609	0.486	0.480	0.594	**0.369**
Abalone	QQ 1:100	2.048	0.480	0.323	0.319	0.444	**0.246**
Bank (8FM)	LL 1:10	1.091	0.491	0.561	0.515	0.509	**0.489**
Bank (8FM)	LL 1:20	1.092	**0.312**	0.370	0.332	0.336	**0.312**
Bank (8FM)	LL 1:50	1.092	**0.166**	0.202	0.177	0.189	0.172
Bank (8FM)	LL 1:100	1.092	0.115	0.124	0.108	0.124	**0.101**
Bank (8FM)	QQ 1:10	0.046	**0.020**	0.026	0.024	0.024	0.023
Bank (8FM)	QQ 1:20	0.046	**0.014**	0.019	0.017	0.017	0.016
Bank (8FM)	QQ 1:50	0.046	**0.007**	0.011	0.010	0.011	0.009
Bank (8FM)	QQ 1:100	0.046	**0.003**	0.008	0.007	0.007	0.006
House (8L)	LL 1:10	13.328	8.958	6.711	6.704	6.964	**5.075**
House (8L)	LL 1:20	13.331	5.794	4.360	4.354	4.779	**3.154**
House (8L)	LL 1:50	13.333	3.351	2.298	2.295	2.883	**1.593**
House (8L)	LL 1:100	13.333	2.218	1.384	1.383	2.013	**0.931**
House (8L)	QQ 1:10	7.209	4.300	3.969	3.964	4.189	**2.656**
House (8L)	QQ 1:20	7.312	3.484	2.899	2.895	3.214	**1.759**
House (8L)	QQ 1:50	7.378	2.591	1.804	1.802	2.219	**0.980**
House (8L)	QQ 1:100	7.401	2.153	1.227	1.225	1.690	**0.618**
KC House	LL 1:10	12.247	5.093	**5.031**	**5.031**	5.213	5.189
KC House	LL 1:20	12.249	4.349	**3.130**	**3.130**	3.307	3.173
KC House	LL 1:50	12.258	1.990	1.567	1.567	1.735	**1.535**
KC House	LL 1:100	12.257	1.113	0.901	0.901	1.059	**0.863**
KC House	QQ 1:10	4.733	2.536	**2.348**	**2.348**	2.430	2.412
KC House	QQ 1:20	4.724	1.812	**1.586**	**1.586**	1.677	1.612
KC House	QQ 1:50	4.719	1.868	0.892	0.892	0.983	**0.883**
KC House	QQ 1:100	4.717	0.872	0.560	0.560	0.647	**0.540**

Table A.8
Performance of OSB, the method introduced by [9] (BSZ), the extended version by [10] (BSZ-EXT), and the local reframing of [12] (HER). All post hoc methods use MT as an initial regression. The performance is expressed in the average misprediction cost (multiplied by 100) on testing data of various datasets using 2 × 5 cross-validation on different cost functions calibrations. The cost ratios $a : b$ always sum up to one as described in Section 4.5.

Comparison of methods on testing data with MT base regression						
Dataset	Cost function	Initial	BSZ	BSZ-EXT	HER	OSB
Abalone	LL 1:10	7.329	3.488	3.485	3.575	**2.814**
Abalone	LL 1:20	7.327	2.159	2.156	2.312	**1.724**
Abalone	LL 1:50	7.326	1.059	1.058	1.243	**0.863**
Abalone	LL 1:100	7.326	0.592	0.591	0.774	**0.500**
Abalone	QQ 1:10	1.984	1.030	1.029	1.072	**0.847**
Abalone	QQ 1:20	1.997	0.706	0.706	0.759	**0.567**
Abalone	QQ 1:50	2.006	0.397	0.397	0.456	**0.325**
Abalone	QQ 1:100	2.009	0.247	0.247	0.304	**0.211**
Bank (8FM)	LL 1:10	1.175	0.559	0.523	0.506	**0.477**
Bank (8FM)	LL 1:20	1.175	0.350	0.326	0.320	**0.290**
Bank (8FM)	LL 1:50	1.175	0.174	0.164	0.166	**0.140**
Bank (8FM)	LL 1:100	1.175	0.099	0.094	0.101	**0.078**
Bank (8FM)	QQ 1:10	0.052	0.027	0.026	0.027	**0.025**
Bank (8FM)	QQ 1:20	0.052	0.019	0.018	0.018	**0.017**
Bank (8FM)	QQ 1:50	0.052	0.011	0.010	0.011	**0.009**
Bank (8FM)	QQ 1:100	0.052	0.007	**0.006**	0.007	**0.006**
House (8L)	LL 1:10	15.268	7.584	7.583	7.858	**5.544**
House (8L)	LL 1:20	15.267	4.895	4.894	5.262	**3.377**
House (8L)	LL 1:50	15.267	2.547	2.547	3.032	**1.645**
House (8L)	LL 1:100	15.267	1.503	1.503	2.030	**0.933**
House (8L)	QQ 1:10	9.358	5.064	5.063	5.318	**3.317**
House (8L)	QQ 1:20	9.493	3.659	3.658	3.995	**2.156**
House (8L)	QQ 1:50	9.58	2.233	2.233	2.653	**1.170**
House (8L)	QQ 1:100	9.61	1.488	1.489	1.948	**0.724**
KC House	LL 1:10	12.46	**5.071**	**5.071**	5.264	5.236
KC House	LL 1:20	12.461	**3.143**	**3.143**	3.312	3.189
KC House	LL 1:50	12.461	1.569	1.569	1.696	**1.531**
KC House	LL 1:100	12.461	0.900	0.900	1.008	**0.855**
KC House	QQ 1:10	4.852	**2.396**	**2.396**	2.471	2.461
KC House	QQ 1:20	4.842	**1.613**	**1.613**	1.691	1.638
KC House	QQ 1:50	4.835	0.903	0.903	0.972	**0.890**
KC House	QQ 1:100	4.833	0.565	0.565	0.625	**0.539**

LGBM implementation in Python [47] directly optimizing the actual cost function, and the local reframing introduced by Hernandez-Orallo [12] (HER). We compare these methods on the datasets introduced in Section 4.2, and we focus on the LinLin and QuadQuad cost functions as these are easier to interpret. We present and discuss the methods using LGBM as initial regression model in the main text. The results when using other initial regression methods are given in Appendix A.1, but the same conclusions hold.

In the implementation of BSZ-EXT we only applied a linear tuning as higher order tuning functions tend to overfit [10]. We also use the same parameter of precision for adjusting the coefficients as proposed in Zhao et al. [10] for both the BSZ and BSZ-EXT methods. In order to objectively compare with the local reframing method, we use the same setup as in the empirical validation of Hernandez-Orallo [12]. As a result, the parameters a and b in each cost function now sum up to one, meaning that a 1 : 100 cost function relates to $a = 1/101$ and $b = 100/101$. Note that this does not change the interpretation or outcomes of the comparative study. Furthermore, we assume that the density function for local reframing follows a normal distribution. We utilize the univariate k-nearest comparison ($uKNC$) enrichment (with $k = 10$) to transform the crisp initial regression model (LGBM) to a soft regression model for local reframing, since this yielded the best results in the study of Hernandez-Orallo [12] on the LinLin and QuadQuad cost functions. The predict-and-optimize approach based on the LGBM implementation uses the specified cost function directly in the LGBM model training. To make the comparison more realistic some hyperparameter tuning is done for each dataset and cost function combination using a grid search (with the searched values between brackets) for the learning rate (0.0001,0.001,0.01,0.1,0.5,1,1.5,2), maximum depth (−1,5,10,20), number of leaves (20,40,60,80,100) and the minimal number of data points in one leaf (1,5,10,15,25).

The results of the comparative study are shown in Table 7. The first two columns describe the dataset and cost function. The other columns represent the various methods as defined earlier with the average misprediction cost per dataset and cost function multiplied by 100 for clarity. In the "Initial Cost" column we can see that the average misprediction costs do not rise significantly when increasing the cost ratio. This can be attributed to the cost ratios summing up to one, hence making the increase in cost ratio only have a marginal effect on the average misprediction cost without a cost-sensitive method. However, the increase in cost ratio makes the overpredictions considerably more attractive in each step.

In general our OSB approach performs very well and outperforms the alternative methods in most cases. Significant improvements on the methods of Bansal et al. [9],Zhao et al. [10] are obtained on almost all cost functions and datasets, except for the KC House dataset. Furthermore, the results from the extended tuning method (BSZ-EXT) are, as expected, always better or very similar as those of BSZ. On the KC House dataset we see that both methods (BSZ and BSZ-EXT) are almost equivalent. Here, the polynomial tuning function primarily consisted out of a large constant and a small or no linear tuning coefficient rendering both approaches equivalent. The difference in average misprediction costs of cases where BSZ-EXT or BSZ outperforms OBS is rather small and our approach remains relatively competitive. Comparing the OSB approach to the local reframing (HER) shows that both methods are relatively close in average misprediction cost on certain dataset/cost function combinations such as Bank(8FM) with

Table A.9
Performance of OSB, the method introduced by [9] (BSZ), the extended version by [10] (BSZ-EXT), a direct implementation using NN, and the local reframing of [12] (HER). All post hoc methods use NN as an initial regression. The performance is expressed in the average misprediction cost (multiplied by 100) on testing data of various datasets using 2 × 5 cross-validation on different cost functions calibrations. The cost ratios $a : b$ always sum up to one as described in Section 4.5.

Comparison of methods on testing data with NN base regression							
Dataset	Cost function	Initial	Direct	BSZ	BSZ-EXT	HER	OSB
Abalone	LL 1:10	7.439	3.663	3.559	3.558	3.658	**2.789**
Abalone	LL 1:20	7.434	2.376	2.201	2.201	2.375	**1.684**
Abalone	LL 1:50	7.432	1.267	1.063	1.063	1.276	**0.841**
Abalone	LL 1:100	7.431	0.762	0.596	0.596	0.795	**0.503**
Abalone	QQ 1:10	2.043	1.577	1.051	1.051	1.095	**0.822**
Abalone	QQ 1:20	2.061	1.020	0.720	0.719	0.777	**0.542**
Abalone	QQ 1:50	2.072	0.750	0.400	0.400	0.468	**0.300**
Abalone	QQ 1:100	2.076	0.633	0.245	0.245	0.313	**0.193**
Bank (8FM)	LL 1:10	1.296	0.517	0.569	0.549	0.534	**0.474**
Bank (8FM)	LL 1:20	1.301	0.320	0.362	0.348	0.343	**0.281**
Bank (8FM)	LL 1:50	1.305	0.162	0.184	0.180	0.183	**0.131**
Bank (8FM)	LL 1:100	1.306	0.107	0.108	0.106	0.114	**0.071**
Bank (8FM)	QQ 1:10	0.059	0.027	0.029	0.029	0.028	**0.023**
Bank (8FM)	QQ 1:20	0.059	0.021	0.021	0.020	0.020	**0.015**
Bank (8FM)	QQ 1:50	0.060	0.014	0.012	0.012	0.012	**0.008**
Bank (8FM)	QQ 1:100	0.060	0.016	0.008	0.008	0.008	**0.005**
House (8L)	LL 1:10	16.281	7.535	7.856	7.750	8.167	**6.160**
House (8L)	LL 1:20	16.294	4.816	5.042	4.967	5.434	**3.797**
House (8L)	LL 1:50	16.303	2.512	2.605	2.568	3.072	**1.900**
House (8L)	LL 1:100	16.306	1.484	1.529	1.509	2.022	**1.106**
House (8L)	QQ 1:10	10.228	5.336	5.452	5.353	5.731	**4.148**
House (8L)	QQ 1:20	10.342	3.786	3.890	3.820	4.239	**2.848**
House (8L)	QQ 1:50	10.415	2.316	2.334	2.302	2.743	**1.720**
House (8L)	QQ 1:100	10.441	1.501	1.532	1.522	1.963	**1.197**
KC House	LL 1:10	12.919	5.277	5.239	5.254	5.431	**5.356**
KC House	LL 1:20	12.930	3.289	3.254	3.267	3.429	**3.247**
KC House	LL 1:50	12.936	1.671	1.627	1.635	1.764	**1.568**
KC House	LL 1:100	12.939	0.965	0.938	0.943	1.053	**0.876**
KC House	QQ 1:10	5.222	2.637	**2.589**	2.626	2.745	2.609
KC House	QQ 1:20	5.214	1.777	1.745	1.770	1.879	**1.741**
KC House	QQ 1:50	5.210	1.019	0.980	0.995	1.081	**0.957**
KC House	QQ 1:100	5.208	0.632	0.617	0.628	0.696	**0.594**

Table A.10
Performance of OSB, the method introduced by [9] (BSZ), the extended version by [10] (BSZ-EXT), and the local reframing of [12] (HER). All post hoc methods use RF as an initial regression. The performance is expressed in the average misprediction cost (multiplied by 100) on testing data of various datasets using 2 × 5 cross-validation on different cost functions calibrations. The cost ratios $a : b$ always sum up to one as described in Section 4.5.

Comparison of methods on testing data with RF base regression						
Dataset	Cost function	Initial	BSZ	BSZ-EXT	HER	OSB
Abalone	LL 1:10	7.319	4.711	4.623	4.923	**4.491**
Abalone	LL 1:20	7.309	3.638	3.556	4.088	**3.563**
Abalone	LL 1:50	7.303	2.570	**2.494**	3.263	2.634
Abalone	LL 1:100	7.301	1.978	**1.925**	2.824	2.117
Abalone	QQ 1:10	2.003	1.388	1.361	1.457	**1.148**
Abalone	QQ 1:20	2.017	1.151	1.122	1.265	**0.927**
Abalone	QQ 1:50	2.027	0.880	0.854	1.053	**0.699**
Abalone	QQ 1:100	2.030	0.716	0.692	0.925	**0.565**
Bank (8FM)	LL 1:10	1.144	0.756	0.708	0.723	**0.698**
Bank (8FM)	LL 1:20	1.144	0.592	0.550	0.582	**0.545**
Bank (8FM)	LL 1:50	1.145	0.421	**0.386**	0.443	0.404
Bank (8FM)	LL 1:100	1.145	0.322	**0.296**	0.368	0.319
Bank (8FM)	QQ 1:10	0.052	0.036	0.034	0.035	**0.032**
Bank (8FM)	QQ 1:20	0.053	0.030	0.028	0.030	**0.026**
Bank (8FM)	QQ 1:50	0.053	0.023	0.021	0.023	**0.020**
Bank (8FM)	QQ 1:100	0.053	0.019	0.017	0.020	**0.016**
House (8L)	LL 1:10	13.337	8.917	8.830	9.004	**8.573**
House (8L)	LL 1:20	13.322	6.944	6.847	7.441	**6.297**
House (8L)	LL 1:50	13.313	4.858	4.771	5.925	**4.480**
House (8L)	LL 1:100	13.309	3.623	3.552	5.115	**3.441**
House (8L)	QQ 1:10	7.425	5.298	5.246	5.474	**3.745**
House (8L)	QQ 1:20	7.523	4.485	4.428	4.833	**2.989**
House (8L)	QQ 1:50	7.586	3.492	3.435	4.105	**2.211**
House (8L)	QQ 1:100	7.608	2.845	2.791	3.658	**1.759**
KC House	LL 1:10	13.191	**6.411**	6.421	6.669	6.542
KC House	LL 1:20	13.208	**4.225**	4.237	4.788	4.352
KC House	LL 1:50	13.219	**2.259**	2.269	3.122	2.431
KC House	LL 1:100	13.223	**1.369**	1.376	2.329	1.500
KC House	QQ 1:10	5.646	**3.123**	3.125	3.268	3.155
KC House	QQ 1:20	5.652	**2.243**	2.246	2.487	2.284
KC House	QQ 1:50	5.655	**1.372**	1.376	1.726	1.418
KC House	QQ 1:100	5.656	**0.920**	0.924	1.327	0.966

the QuadQuad cost function. However, in other cases OSB clearly outperforms local reframing as on the House (8L) dataset. The obtained results for local reframing could be improved by potentially using more neighbors for the $uKNC$ enrichment method or even using a different enrichment method. The performance of the initial regression method can also have an impact on these results. The direct approach based on LGBM does not always deliver competitive results. Due to the small grid search, the optimal hyperparameters were not found in some cases or perhaps the LGBM is not the best implementation for a direct approach on certain dataset/cost function combinations. Nevertheless, on the Bank (8FM) dataset it outperformed all other implementations for most cost functions, indicating that good results can be achieved with proper hyperparameter tuning and a good fit on the data. More extensive tuning on the other datasets might lead to better results. However, such extensive tuning is challenging and costly in terms of computation time.

The average computation time across all simulated results (every dataset/cost function combination) of OSB and the methods of Bansal et al. [9],Zhao et al. [10],Hernandez-Orallo [12] is 165 ms, 162 ms, 281 ms and 8557 ms respectively. The higher computation time for the local reframing method can be attributed to the need for looping over the entire test set as well as the calculations for the various t values (See Proposition 7.7 in Hernandez-Orallo [12]). The computation time of the direct method on average is 10 min and 59.2 s, which is considerably more than the other methods and can be attributed to the grid search for the hyperparameters. These results stress the fast computational time of our OSB approach. All experiments were run on a Thinkpad T495 workstation, configured with AMD Ryzen 7 PRO 3700U 2.30 GHz and 16 GB of RAM, running the Windows 10 operating system.

In general, we can conclude that our OSB method performs very well compared to other post hoc implementations, as it outperformed in most cases and otherwise remains relatively competitive to the best performing method. Our OSB approach can even outperform a direct optimization approach. Another important advantage is its fast computation time due to an efficient algorithm. Moreover, the obtained results become easy to interpret through bootstrapping.

5. Conclusion

In this paper we have shown the importance of cost-sensitivity in regression problems. Since many real-world applications of regression problems often face asymmetric costs, we have proposed a one-step boosting algorithm (OSB) to incorporate cost-sensitivity into a cost-insensitive regression. The OSB method allows for more realistic cost structures to be implemented in a post hoc manner, rather than adjusting or modifying the underlying method or objective function. This post hoc introduction of cost-sensitivity entails several advantages. It can be applied to a wide range of cost functions and, in essence, can extend any base regression to a cost-sensitive regression, enabling model reuse. Moreover, it allows for the evaluation of multiple cost functions without having to make model changes, and thus enabling the practitioner to build and validate cost functions on the fly. This proves an even bigger advantage in dynamic environments where cost structures tend to change frequently.

Building on earlier studies that leverage a tuning function to introduce cost-sensitivity post hoc, the OSB algorithm illustrates a more general strategy to post hoc cost-sensitivity. The proposed method uses a linear function in the boosting step, optimized on the asymmetric cost

Table A.11
Performance of OSB, the method introduced by [9] (BSZ), the extended version by [10] (BSZ-EXT), and the local reframing of [12] (HER). All post hoc methods use LR as an initial regression. The performance is expressed in the average misprediction cost (multiplied by 100) on testing data of various datasets using 2 × 5 cross-validation on different cost functions calibrations. The cost ratios a : b always sum up to one as described in Section 4.5.

Comparison of methods on testing data with LR base regression						
Dataset	Cost function	Initial	BSZ	BSZ-EXT	HER	OSB
Abalone	LL 1:10	7.889	3.808	3.807	4.023	**2.962**
Abalone	LL 1:20	7.888	2.354	2.354	2.644	**1.796**
Abalone	LL 1:50	7.887	1.134	1.134	1.443	**0.891**
Abalone	LL 1:100	7.887	0.627	0.627	0.911	**0.512**
Abalone	QQ 1:10	2.336	1.205	1.205	1.315	**0.987**
Abalone	QQ 1:20	2.354	0.822	0.822	0.944	**0.668**
Abalone	QQ 1:50	2.366	0.455	0.455	0.578	**0.387**
Abalone	QQ 1:100	2.370	0.277	0.277	0.391	**0.263**
Bank (8FM)	LL 1:10	1.426	0.682	0.642	0.618	**0.533**
Bank (8FM)	LL 1:20	1.426	0.429	0.403	0.395	**0.319**
Bank (8FM)	LL 1:50	1.426	0.215	0.206	0.207	**0.149**
Bank (8FM)	LL 1:100	1.426	0.124	0.120	0.126	**0.083**
Bank (8FM)	QQ 1:10	0.078	0.041	0.039	0.042	**0.035**
Bank (8FM)	QQ 1:20	0.078	0.028	0.027	0.029	**0.023**
Bank (8FM)	QQ 1:50	0.078	0.016	0.016	0.017	**0.012**
Bank (8FM)	QQ 1:100	0.078	0.010	0.010	0.011	**0.008**
House (8L)	LL 1:10	19.707	10.098	10.098	10.115	**7.379**
House (8L)	LL 1:20	19.706	6.402	6.402	6.660	**4.202**
House (8L)	LL 1:50	19.706	3.276	3.276	3.696	**1.764**
House (8L)	LL 1:100	19.705	1.904	1.903	2.382	**0.896**
House (8L)	QQ 1:10	15.968	8.646	8.646	8.577	**5.758**
House (8L)	QQ 1:20	16.171	6.139	6.138	6.225	**3.634**
House (8L)	QQ 1:50	16.302	3.634	3.632	3.902	**1.827**
House (8L)	QQ 1:100	16.347	2.357	2.356	2.714	**1.034**
KC House	LL 1:10	12.784	**5.207**	**5.207**	5.397	5.336
KC House	LL 1:20	12.784	3.227	3.227	3.402	**3.223**
KC House	LL 1:50	12.783	1.608	1.608	1.758	**1.548**
KC House	LL 1:100	12.783	0.921	0.921	1.048	**0.858**
KC House	QQ 1:10	5.097	**2.518**	**2.518**	2.600	2.567
KC House	QQ 1:20	5.086	**1.693**	**1.693**	1.782	1.697
KC House	QQ 1:50	5.078	0.945	0.945	1.028	**0.915**
KC House	QQ 1:100	5.076	0.590	0.590	0.664	**0.553**

function using iteratively reweighted least-squares, in order to make the base regression method cost-sensitive. The evaluated results indicate a significant reduction in average misprediction cost across various cost functions, datasets, and base regression methods. Furthermore, following the use of a linear function in the boosting step, the results can be made interpretable through the use of bootstrapping due to the fast convergence of the iteratively reweighted least-squares procedure. The secondary learner can then provide insights into how taking costs into account may alter the trained model. In addition, the use of a linear function as the secondary learner leads to easy and understandable inferencing, fast cost function calibration, quick evaluation of multiple cost functions and simplicity in implementation. The OSB algorithm is even equivalent to direct (predict-and-optimize) approaches in cases where the initial model is a linear regression. This aspect as well as the interpretability of the results and potential use of a different set of variables give the OSB approach unique advantages over other post hoc methods.

Future research can expand upon OSB in several ways. First, other cost functions can be evaluated in the algorithm to further validate results in potentially more specific settings. Second, the introduced cost functions can be expanded to cost functions of the type $w(x)\rho(x)$, where the costs also depend on certain variables from the sample. This extension of the method will allow for more complicated cost structures to be developed, similarly to those in classification problems. Third, the one-step boosting method for cost-sensitivity can be more extensively compared to methods where cost-sensitivity is introduced by model adjustment. Comparing these two approaches to cost-sensitivity can lead to interesting results regarding precision, ease of use, and relevance for practitioners. Fourth, in our empirical evaluation we applied the same training sample in the boosting step as was used in the base regression method. However, the boosting approach allows the use of a select number of variables or even a completely different set of variables to introduce cost-sensitivity. Depending on the specific problem and cost setting certain combinations can be evaluated further. Finally, we introduced and evaluated the proposed boosting method from a data-science and statistical viewpoint. Evaluating the algorithm in practice on a specific problem with a dynamic and changing environment can provide interesting results for the future use of the boosting approach to incorporate cost-sensitivity in regression problems.

CRediT authorship contribution statement

Thomas Decorte: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft. **Jakob Raymaekers:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Supervision. **Tim Verdonck:** Methodology, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The preprocessed data that support the findings of this study are available at https://github.com/ThomasDecorteUA/Cost_Sensitive_Regression. These data, namely the Abalone, Bank (8FM), House (8L), and KC House datasets, were derived from the following resources available in the public domain: the Abalone, Bank (8FM) and House (8L) datasets are made available on the DELVE (https://www.cs.toronto.edu/delve/data/datasets.html) (Data for Evaluating Learning in Valid Experiments) repository of the University of Toronto as well as on the UCI repository (https://archive.ics.uci.edu/ml/datasets.php). The last dataset, KC House, is made available by the Center for Spatial Data Science (https://geodacenter.github.io/data-and-lab//KingCounty-HouseSales2015/) at the University of Chicago. The link to the github code is included in the manuscript as a footnote, namely the second footnote in section 4.1.

Acknowledgments

The authors want to thank Professor Dr. José Hernández-Orallo for providing the implementation of his local reframing method.

Appendix. Extra results output

A.1. Results comparison with other base regression methods

In Tables A.8, A.9, A.10 and A.11 are the results for the comparison of OSB, the method introduced by Bansal et al. [9] (BSZ), the extended version by Zhao et al. [10] (BSZ-EXT) and the local reframing of Hernandez-Orallo [12] (HER) using the RF random forest, MT model tree, NN neural network and the LR linear regression as initial models. In these comparisons we only leverage a direct approach for the neural network implementation, as a direct approach is not always available for other methods or in the case of a linear regression is equivalent to our method as highlighted earlier. The setup is the same as described in Section 4.5, meaning that the costs sum up to one. Thus a cost function of the form 1 : 10 has $a = 1/11$ and $b = 10/11$ as cost ratio values. The average misprediction cost is in each table multiplied by 100 for clarity. The conclusions for the different base regression methods for the OSB method are very comparable as those discussed earlier in Section 4.3.

As can be seen, the base regression does not have a large influence. Of course the performance of our method does depend on the fit of the base regression as an overfit will lead to worse results in the subsequent boosting step, due to the poor performance in the test predictions. Here, this can be seen in some cases where the model clearly is overfit or has a bad fit with the data such as with the LR regression on the House (8L) dataset, which is also illustrated by the worse performance on the initial costs. This can be attributed to the non linear relationship between the variables and response. The conclusions for the comparison with other methods on various base regression methods are consistent with the earlier results discussed in Section 4.5, again stressing the consistency of the introduced OSB approach.

References

[1] E. Alfaro, N. García, M. Gámez, D. Elizondo, Bankruptcy forecasting: An empirical comparison of AdaBoost and neural networks, Decis. Support Syst. 45 (1) (2008) 110–122.

[2] A.C. Bahnsen, D. Aouada, B. Ottersten, Example-dependent cost-sensitive logistic regression for credit scoring, in: 2014 13th International Conference on Machine Learning and Applications, IEEE, 2014, pp. 263–269.

[3] H. Wang, Z. Cui, Y. Chen, M. Avidan, A.B. Abdallah, A. Kronzer, Predicting hospital readmission via cost-sensitive deep learning, IEEE/ACM Trans. Comput. Biol. Bioinform. 15 (6) (2018) 1968–1978.

[4] S. Höppner, B. Baesens, W. Verbeke, T. Verdonck, Instance-dependent cost-sensitive learning for detecting transfer fraud, European J. Oper. Res. 297 (1) (2022) 291–300.

[5] C. Elkan, The foundations of cost-sensitive learning, in: International Joint Conference on Artificial Intelligence, Vol. 17 (1), Lawrence Erlbaum Associates Ltd, 2001, pp. 973–978.

[6] P. Donti, B. Amos, J.Z. Kolter, Task-based end-to-end model learning in stochastic optimization, Adv. Neural Inf. Process. Syst. 30 (2017).

[7] B. Wilder, B. Dilkina, M. Tambe, Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33 (01), 2019, pp. 1658–1665.

[8] T. Vanderschueren, T. Verdonck, B. Baesens, W. Verbeke, Predict-then-optimize or predict-and-optimize? An empirical evaluation of cost-sensitive learning strategies, Inform. Sci. 594 (2022) 400–415.

[9] G. Bansal, A.P. Sinha, H. Zhao, Tuning data mining methods for cost-sensitive regression: a study in loan charge-off forecasting, J. Manage. Inf. Syst. 25 (3) (2008) 315–336.

[10] H. Zhao, A.P. Sinha, G. Bansal, An extended tuning method for cost-sensitive regression and forecasting, Decis. Support Syst. 51 (3) (2011) 372–383.

[11] P. Huber, J. Wiley, W. InterScience, Robust Statistics, Wiley New York, 1981.

[12] J. Hernandez-Orallo, Probabilistic reframing for cost-sensitive regression, ACM Trans. Knowl. Discov. Data (TKDD) 8 (4) (2014) 1–55.

[13] L. Zhang, J. Huang, L. Liu, Improved deep learning network based in combination with cost-sensitive learning for early detection of ovarian cancer in color ultrasound detecting system, J. Med. Syst. 43 (2019) 1–9.

[14] S. Qiu, R.B. Chinnam, A. Murat, B. Batarse, H. Neemuchwala, W. Jordan, A cost sensitive inpatient bed reservation approach to reduce emergency department boarding times, Health Care Manag. Sci. 18 (2015) 67–85.

[15] T. Van Calster, F.V.d. Bossche, B. Baesens, W. Lemahieu, Profit-oriented sales forecasting: a comparison of forecasting techniques from a business perspective, 2020, arXiv preprint arXiv:2002.00949.

[16] C.h. Tsai, L.c. Chang, H.c. Chiang, Forecasting of ozone episode days by cost-sensitive neural network methods, Sci. Total Environ. 407 (6) (2009) 2124–2135.

[17] G. Arminger, N. Götz, Asymmetric Loss Functions for Evaluating the Quality of Forecasts in Time Series for Goods Management Systems, Univ., SFB 475, 1999.

[18] S. Höppner, E. Stripling, B. Baesens, S. vanden Broucke, T. Verdonck, Profit driven decision trees for churn prediction, European J. Oper. Res. 284 (3) (2020) 920–933.

[19] A.C. Bahnsen, D. Aouada, B. Ottersten, A novel cost-sensitive framework for customer churn predictive modeling, Decis. Anal. 2 (1) (2015) 1–15.

[20] K. Coussement, Improving customer retention management through cost-sensitive learning, Eur. J. Mark. 48 (3/4) (2014) 477–495.

[21] M. Cain, C. Janssen, Real estate price prediction under asymmetric loss, Ann. Inst. Statist. Math. 47 (3) (1995) 401–414.

[22] H.R. Varian, A Bayesian Approach to Real Estate Assessment, in: Studies in Bayesian Econometric and Statistics in Honor of Leonard J. Savage, North Holland, 1975, pp. 195–208.

[23] M. Czajkowski, M. Czerwonka, M. Kretowski, Cost-sensitive global model trees applied to loan charge-off forecasting, Decis. Support Syst. 74 (2015) 57–66.

[24] Y. Hu, B. Feng, X. Mo, X. Zhang, E. Ngai, M. Fan, M. Liu, Cost-sensitive and ensemble-based prediction model for outsourced software project risk prediction, Decis. Support Syst. 72 (2015) 11–23.

[25] N. Ghatasheh, H. Faris, R. Abukhurma, P.A. Castillo, N. Al-Madi, A.M. Mora, A.M. Al-Zoubi, A. Hassanat, Cost-sensitive ensemble methods for bankruptcy prediction in a highly imbalanced data distribution: A real case from the Spanish market, Prog. Artif. Intell. 9 (2020) 361–375.

[26] Y. Sahin, S. Bulkan, E. Duman, A cost-sensitive decision tree approach for fraud detection, Expert Syst. Appl. 40 (15) (2013) 5916–5923.

[27] A.C. Bahnsen, A. Stojanovic, D. Aouada, B. Ottersten, Cost sensitive credit card fraud detection using Bayes minimum risk, in: 2013 12th International Conference on Machine Learning and Applications, Vol. 1, IEEE, 2013, pp. 333–338.

[28] S. Nami, M. Shajari, Cost-sensitive payment card fraud detection based on dynamic random forest and k-nearest neighbors, Expert Syst. Appl. 110 (2018) 381–392.

[29] S.F. Crone, Training artificial neural networks for time series prediction using asymmetric cost functions, in: Proceedings of the 9th International Conference on Neural Information Processing, 2002, Vol. 2, ICONIP'02, IEEE, 2002, pp. 2374–2380.

[30] A. Basu, R. Thompson, P.W. Laud, Life testing and reliability estimation under asymmetric loss, in: Survival Analysis: State of the Art, Springer, 1992, pp. 3–10.

[31] A. Zellner, Bayesian estimation and prediction using asymmetric loss functions, J. Amer. Statist. Assoc. 81 (394) (1986) 446–451.

[32] G.A. Christodoulakis, Financial forecasts in the presence of asymmetric loss aversion, skewness and excess kurtosis, Finance Res. Lett. 2 (4) (2005) 227–233, http://dx.doi.org/10.1016/j.frl.2005.08.002, URL https://www.sciencedirect.com/science/article/pii/S1544612305000541.

[33] P.F. Christoffersen, F.X. Diebold, Optimal prediction under asymmetric loss, Econom. Theory 13 (6) (1997) 808–817.

[34] P.F. Christoffersen, F.X. Diebold, Further results on forecasting and model selection under asymmetric loss, J. Appl. Econometrics 11 (5) (1996) 561–571.

[35] M. Niglio, Multi-step forecasts from threshold ARMA models using asymmetric loss functions, Stat. Methods Appl. 16 (3) (2007) 395–410.

[36] J. Liu, E. Zio, Weighted-feature and cost-sensitive regression model for component continuous degradation assessment, Reliab. Eng. Syst. Saf. 168 (2017) 210–217.

[37] S.F. Crone, S. Lessmann, R. Stahlbock, Utility based data mining for time series analysis: Cost-sensitive learning for neural network predictors, in: Proceedings of the 1st International Workshop on Utility-Based Data Mining, 2005, pp. 59–68.

[38] J. Yao, C.L. Tan, Time dependent directional profit model for financial time series forecasting, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, Vol. 5, IEEE, 2000, pp. 291–296.

[39] J. Hernández-Orallo, Soft (Gaussian CDE) regression models and loss functions, 2012, arXiv preprint arXiv:1211.1043.

[40] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (2) (1990) 197–227.

[41] P.J. Huber, Robust estimation of a location parameter, in: Breakthroughs in Statistics, Springer, 1992, pp. 492–518.

[42] F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, W.A. Stahel, Robust Statistics: the Approach Based on Influence Functions, Vol. 196, John Wiley & Sons, 2011.

[43] P.J. Huber, Robust Statistics, Vol. 523, John Wiley & Sons, 2004.

[44] P.J. Rousseeuw, A.M. Leroy, Robust Regression and Outlier Detection, Vol. 589, John wiley & sons, 2005.

[45] P.J. Huber, Robust estimation of a location parameter, Ann. Math. Stat. 35 (1) (1964) 73–101, http://dx.doi.org/10.1214/aoms/1177703732.

[46] C.W. Granger, Prediction with a generalized cost of error function, J. Oper. Res. Soc. 20 (2) (1969) 199–207.

[47] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Adv. Neural Inf. Process. Syst. 30 (2017) 3146–3154.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, URL https://www.tensorflow.org/ Software available from tensorflow.org.

Thomas Decorte is a Ph.D. candidate at the University of Antwerp. He obtained a Master in Commercial Engineering from the KU Leuven in 2020. His research interests include robust and cost-sensitive learning, causal modeling, and data-driven sports analytics.

Jakob Raymaekers is an Assistant Professor at the Department of Quantitative Economics of Maastricht University. His research interests relate to the development of

machine learning and statistics for anomaly detection, with applications to business. His research has been published in journals such as Journal of Computational and Graphical Statistics, Journal of Machine Learning Research, Machine Learning, Technometrics and Journal of Multivariate Analysis.

Tim Verdonck is a Professor in Statistics and Data Science at the University of Antwerp - imec and KU Leuven. He is chairholder of the BNP Paribas Fortis Chair in Fraud Analytics, the Allianz Chair in Prescriptive Business Analytics in Insurance and the BASF Chair in Robust Predictive Analytics. His research concentrates on data science with a focus on finance and insurance problems. Recent research has been published in journals such as Applied Soft Computing, Computational Statistics & Data Analysis, Decision Support Systems, European Journal of Operational Research and Information Sciences.