Decision Support Systems 74 (2015) 57–66

Contents lists available at ScienceDirect

Decision Support Systems

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

Cost-sensitive Global Model Trees applied to loan charge-off forecasting

Marcin Czajkowski [a,*], Monika Czerwonka [b], Marek Kretowski [a]

[a] Faculty of Computer Science, Bialystok University of Technology, Wiejska 45a, 15-351 Bialystok, Poland
[b] Collegium of Management and Finance, Warsaw School of Economics, Al. Niepodleglosci 162, 02-554 Warsaw, Poland

ARTICLE INFO

Article history:
Received 17 April 2014
Received in revised form 10 February 2015
Accepted 30 March 2015
Available online 9 April 2015

Keywords:
Cost-sensitive regression
Model trees
Evolutionary algorithms
Asymmetric costs
Loan charge-off forecasting

ABSTRACT

Regression learning methods in real world applications often require cost minimization instead of the reduction of various metrics of prediction errors. Currently in the literature, there is a lack of white box solutions that can deal with forecasting problems where under-prediction and over-prediction errors have different consequences. To fill this gap, we introduced the Cost-sensitive Global Model Tree (*CGMT*), which applies a fitness function that minimizes an average misprediction cost. Proposed specialized genetic operators improve searching for optimal tree structure and cost-sensitive linear regression models in the leaves. Experimental validation is performed on loan charge-off data. It is known to be a difficult forecasting problem for banks due to the asymmetric cost structure. Obtained results show that specialized evolutionary algorithm applied to model tree induction finds significantly more accurate predictions than tested competitors. Decisions generated by the *CGMT* are simple, easy to interpret, and can be applied directly.

1. Introduction

A lot of real world problems are cost-sensitive, which means that different types of prediction errors are not equally costly [5]. As a result, typical minimization of prediction errors is not the best scenario. A cost-sensitive term encompasses all types of learning where cost is considered [50,19], and different types of costs (e.g., costs of attributes, cost of instances, and costs of errors) can be distinguished. The current research focuses on a single cost for decision making, however, multiple costs are also investigated [35].

For example, in medical diagnoses, there are several types of costs that can be minimized, such as the cost of misclassification (e.g., overlooking an ill patient can be fatal in contrast to a false positive test) or the cost of treatment (e.g., financial or risk). When speculating on stock exchange, investors directly compare future gains and losses and usually give more weight to losses. Researchers show that potential gains need to be approximately twice as large to offset potential losses [51]. As a consequence, investors tend to realize their gains more often than their losses as they sell winning stocks more readily. There are many other examples for such asymmetry, such as in bankruptcy prediction [57], behavioral finances [45], expected stock returns [2], criminal justice settings [6], physician prognostic behavior [1], product recommendations [32], and so on.

Cost-sensitive regression is still not adequately addressed in the data mining literature, as most existing research in this area deals with classification problems. Conventional algorithms usually operate with symmetric loss functions and minimize absolute or squared errors that do not distinguish differences between under-prediction and over-prediction, as each is weighted equally (the cost of under-prediction and over-prediction is equal). There is a need for solutions with asymmetric loss functions that can successfully forecast cost-sensitive regression problems. Such models also minimize absolute or squared errors, however, the under-predicted and over-predicted instances have different weights that depend on the costs.

Our study makes several important contributions to the literature. First, we propose a new method called the Cost-sensitive Global Model Tree (*CGMT*), which extends the cost-neutral solution called *GMT* [17]. By applying evolutionary algorithms (*EA*) in the model tree induction, we managed to successfully search for optimal tree structure and cost-sensitive regression models in the leaves under different asymmetric loss functions. What is more, *CGMT* predictions on loan charge-off forecasting data as one of the cost-sensitive problems faced by banks are significantly more accurate than the results of their tested counterparts.

Next, the hierarchical tree structure, in which appropriate tests from consecutive nodes are sequentially applied, closely resembles a human way of decision making. Therefore, the *CGMT* prediction model is natural and easy to understand and interpret, which is extremely important in financial forecasting. Finally, we propose an improvement of existing algorithms in terms of their performance. In solutions proposed in [5] and [56], the linear tuning function is calculated by the heuristic approach (hill climbing algorithm). We managed to find a direct minimization that returns the exact value of an adjusted regression model. The proposed approach significantly extends upon previously performed research on cost-sensitive extensions for *GMT* [16]. In

* Corresponding author.
E-mail address: m.czajkowski@pb.edu.pl (M. Czajkowski).

http://dx.doi.org/10.1016/j.dss.2015.03.009
0167-9236/

particular, proposed solution can work with any convex function. New specialized variants of genetic operators were proposed, the fitness function was improved and mode detailed experimental analysis was performed.

The rest of the paper is organized as follows. The next section presents background information and Section 3 proposes the *CGMT* approach. The experimental evaluation is performed in Section 4 and the paper is concluded in the last section, in which future work is also discussed.

2. Background

In this section, we want to present some background information about decision trees for regression and asymmetric loss function as well as the problem of loan charge-off forecasting.

2.1. Regression and model trees

The most common predictive tasks in data mining are classification and regression, and the decision tree [42] is one of the most frequently applied prediction techniques. Tree-based approaches are easy to understand, visualize, and interpret. Their similarity to the human reasoning process makes them a powerful tool [29] among data analysts.

The problem of learning the optimal model tree is known to be NP-complete [39]. Consequently, practical decision-tree inducers are based on heuristics such as the greedy approach, where locally optimal decisions are made in each node. This process is known as recursive partitioning [41]. Two main variants of decision trees can be distinguished by the type of problem they are applied to. Tree predictors can be used to classify existing data (classification trees) or to approximate real-valued functions (regression trees). In each leaf, the classification tree assigns a class label (usually a majority class of all instances that reach that particular leaf), while the regression tree holds a constant value (usually an average value for the target attribute). The model tree can be seen as an extension of the regression tree. The most important difference is that the constant value in each leaf of the regression tree is replaced in the model tree by the linear (or nonlinear) regression function. An example of classification, regression, and model tree induced by the top-down greedy approach is illustrated in Fig. 1. The color of each region in the classification tree represents a different class. In regression and model trees, the height of each region corresponds to the value of the prediction function.

One of the first and probably the most well-known top-down regression tree solutions is the *CART* system [7]. The algorithm searches for a locally optimal split that minimizes the sum of squared residuals and builds a piecewise constant model with each terminal node fitted with the training sample mean. Other solutions have managed to improve the prediction accuracy by replacing single values in the leaves with more advanced models. The *M*5 system [52] induces a tree that contains multiple linear models in the leaves, and thus the tree is similar to a piecewise linear function. All these solutions are fast and generally efficient in many practical problems, but they usually produce locally optimal solutions.

To limit the negative effects of greedy induction, multiple authors have proposed various techniques [36,54]. However, the true global approach for decision tree induction was possible with evolutionary computation. In the literature, there are attempts to apply the evolutionary approach for the induction of decision trees [4], but only a few solutions concern the regression problem. In *TARGET* [20], the authors propose to evolve a *CART*-like regression tree with simple operators and the Bayesian Information Criterion (*BIC*) [43] as a fitness function. Evolutionary induced regression trees with linear models in the leaves were proposed in a solution called *E-Motion* [3]. The authors applied a standard 1-point crossover and two different mutation strategies (shrinking and expanding). The algorithm optimizes the tree error

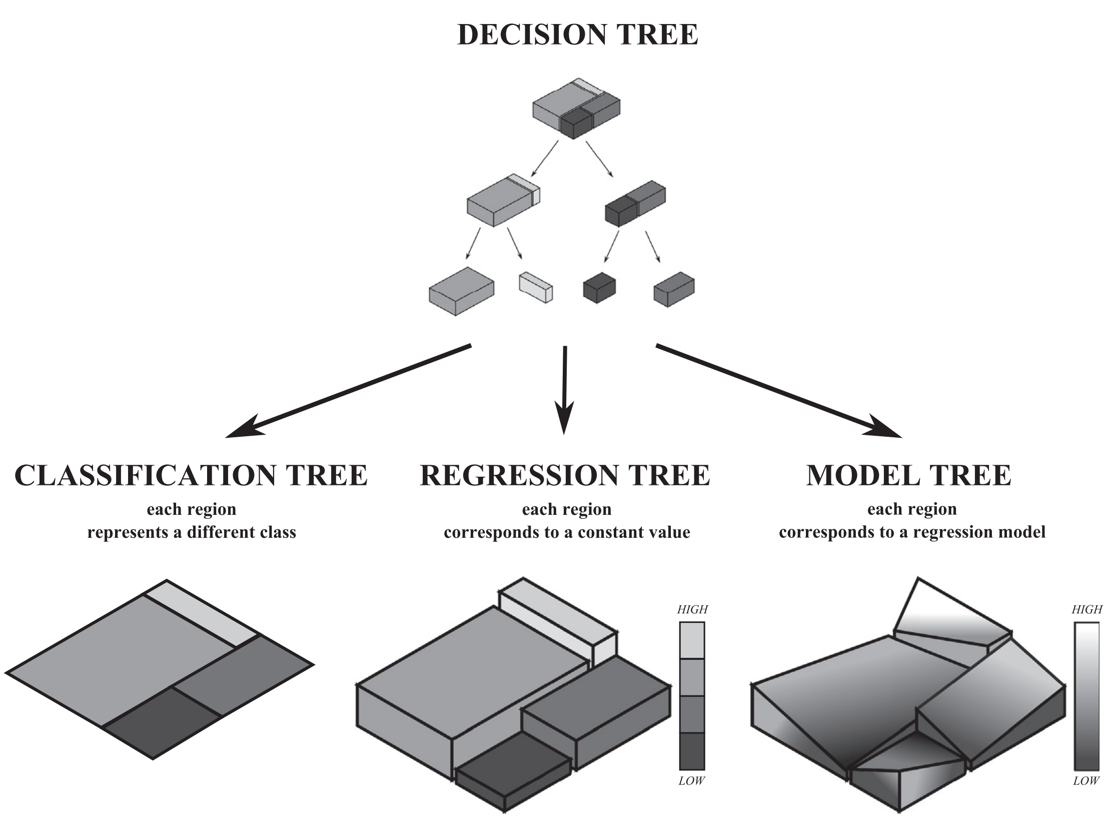

Fig. 1. An example of top-down induction of classification, regression, and model tree.

(*RMSE*, *MAE*) and size (number of leaves) using a weight formula or lexicographic analysis as a fitness function. A more complex solution called *GMT* is proposed in [15,17], where specialized *EA* is used to induce model trees as well as the linear regression models in the leaves.

Most of the systems that perform global searches in the field of candidate solutions compete successfully with popular greedy methods. Evolutionary induced trees are usually smaller and have higher prediction accuracy [4]. The price for better and more accurate predictions is the slow induction time of decision trees, but this can be partially mitigated by parallel implementations or combining *EA* with memetic algorithms [11,14].

Although, the forecasting performance of regression and model trees may be slightly lower than more complex solutions like random forests or neural networks, the explanation ability of the model trees and the transparency of predictions are significantly higher.

2.2. Asymmetric loss function

The loss function is a part of every forecasting model. Let dependent variable y be predicted based on a vector of independent variables (X), and let $f(X)$ be a function of the regressors minimizing a particular loss function $L(y, f(X))$. Typical examples of loss functions are the squared error, such as used in least squares methods:

$$L(y, f(X)) = (y - f(X))^2 \quad (1)$$

and the absolute error:

$$L(y, f(X)) = |y - f(X)|. \quad (2)$$

Both loss functions are symmetric in that for every y and k, the $L(y + k, f(X)) = L(y - k, f(X))$. Symmetric loss functions dominate in statistics and data mining. The Nobel Laureate Clive Granger stated that the obvious problem with the choice of the right loss function is that it is a symmetric function, whereas actual loss functions are often asymmetric [24].

The pioneer in the field of asymmetric costs in prediction was Varian [53], who proposed the *LinEx* loss function. This loss function, which was approximately exponential on one side and linear on the other, became a popular alternative to least squares procedures. It was later extended to the asymmetric linear quadratic loss functions [10] presented in Fig. 2.

However, in machine learning, there are not many propositions of algorithms that handle asymmetric costs. In the most recent survey of cost-sensitive decision tree induction algorithms [34], the regression or model trees were not mentioned, and only the problem of classification was discussed. In the induction of cost-sensitive classification trees, three techniques are popular:

- Transforming the traditional decision tree into a cost-sensitive one. This requires incorporating costs into splitting criteria and pruning [8];
- Post hoc tuning (e.g., MetaCost [18] or cost instance weighting [47]). Both methods are universal and transform traditional algorithms into cost-sensitive;
- Using the evolutionary approach for cost-sensitive decision tree induction [30].

The vast majority of data mining algorithms are applied only to classification problems [48], while cost-sensitive regression is not really studied outside the field of statistics [5]. There are, however, few papers in the literature that deal with regression (e.g., in [12], the authors propose a modified back-propagation neural network (*NN*) that applies the *LinLin* cost function). As for the regression and model trees, there are two different techniques: the application of *EA*s for model trees [16] and post hoc tuning [5,56]. There also exist other regression solutions, but they are focused on different types of costs, such as feature evaluation costs [23] and rare extreme values [49].

Post hoc tuning methods [5] for regression are similar to the ones in cost-sensitive classification. The solution proposed by Bansal et al. [5] minimizes average misprediction cost post hoc by adjusting the final prediction by a certain value. The algorithm is tested for traditional regression methods under an asymmetric cost structure. This system was later extended [56] and introduced polynomial functions as a model adjustment. This way, the algorithm could work with any convex cost function, such as *LinLin* and *QuadQuad*.

The first cost-sensitive extensions for the global induction of model trees were introduced in [16]. Authors have successfully extended the evolutionary algorithm for model tree induction with an asymmetric *LinLin* loss function that minimizes the average misprediction cost. Two variants of mutation operators that search for cost-sensitive linear regression models in the leaves were proposed. The first operator adjusted the regression model by a certain amount, like in the post hoc tuning method [5]. The second operator calculated a new regression model based on a subset of instances in the node.

2.3. Loan charge-off forecasting

Loan loss reserves are determined by banks based on their predictions of future loan charge-off amounts. This data is characterized by asymmetric costs on misprediction errors because the under-prediction of loan charge-off is more costly than over-prediction. This problem was also evaluated in the papers [5,56,16].

Let's assume that the bank needs to predict the future loan charge-off (y) using a vector of independent variables (X). If the regression model in the bank over-predicts its future loan charge-off ($f(X) > y$), the worst that could happen is a reduction in the bank's income because extra funds will remain in the loan loss reserves and in some cases, it will also result in a lower credit score from financial analysts. Under-prediction ($f(X) < y$) means that the bank did not prepare sufficient provisions for its loan losses and does not have enough reserves. It causes regulatory problems and a significant downturn of its credit rating, which is much more dangerous to the bank.

As it was pointed out in [5], it is important to penalize under-predictions more heavily than over-predictions by discouraging banks from having less than adequate amounts as reserves. This requires using asymmetric loss functions in prediction models like *LinLin* and *QuadQuad*.

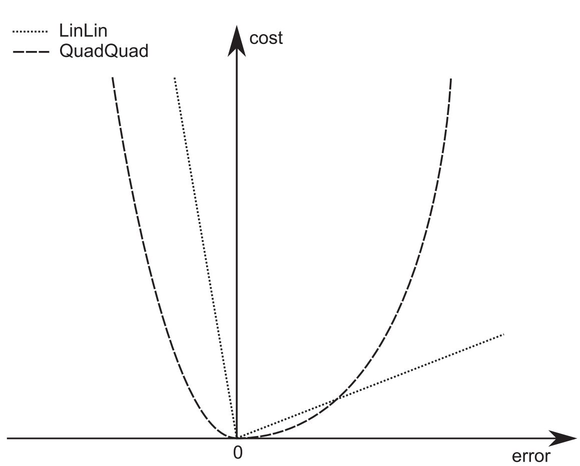

Fig. 2. An example of *LinLin* and *QuadQuad* cost functions.

3. Cost-sensitive Global Model Tree

In this section, we present how to induce the cost-sensitive model tree with the evolutionary approach. At first, we briefly describe a solution called Global Model Tree (*GMT*) [17]. Next, we illustrate how to efficiently convert this cost-neutral model tree inducer into the cost-sensitive one and propose our solution, the Cost-sensitive Global Model Tree (*CGMT*). We also refer to the solutions proposed in [16,5,56] and show how they can be improved.

3.1. Global Model Tree schema

Evolutionary algorithms [37] belong to a family of meta-heuristic methods. They represent techniques for solving a wide variety of difficult optimization problems. *GMT* follows a traditional framework of *EA* inspired by biological mechanisms of evolution. The algorithm (see Fig. 3) operates on individuals that compose a current population. Each individual represents a candidate solution to the target problem. Individuals are assessed using a fitness function that measures their performance. Next, individuals with higher fitness usually have a higher probability of being selected for reproduction. Genetic operators such as mutation and crossover influence individuals, thereby producing new offspring. This guided random search (offspring usually inherits some traits from its ancestors) is stopped when some convergence criteria is satisfied.

Model trees are represented in their actual form as traditional univariate trees, so each split in the internal node is based on a single attribute. If the attribute is nominal, at least one value is associated with each branch (inner disjunction). In case of a continuous-valued attribute, the typical inequality tests are applied. To predict instances associated with each leaf, a multivariate linear regression model [40] is constructed. Initial individuals are induced using greedy strategies on a random subsample of the training data, and the tests in internal nodes are searched on a random subset of attributes. Each tree leaf contains a multiple linear regression model using the standard regression technique [40] that is constructed with instances associated with that leaf. A dependent variable (y) is explained by the linear combination of multiple independent variables $X = \{x_1, x_2, \ldots, x_m\}$:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \ldots + \beta_q * x_q, \tag{3}$$

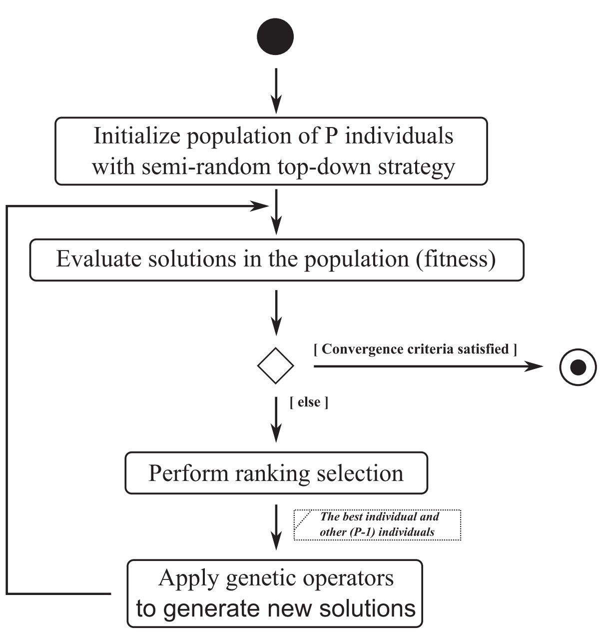

Fig. 3. The GMT process diagram.

where q is the number of independent variables and $\beta_{0 \ldots q}$ are fixed coefficients that minimize the sum of the squared residuals of the model.

Tree-based representation requires developing specialized genetic operators corresponding to classical mutation and crossover. Application of the operators can modify the tree structure, tests in internal nodes, and models in the leaves. The crossover operator attempts to combine elements of two existing individuals (parents) to create a new solution. An example of crossover where two individuals exchange all subtrees is illustrated in Fig. 4. The mutation operator makes random changes in some places of selected individuals. Several variants of crossover and mutations were proposed [13,15,17]:

- Modify the test in internal nodes (shift threshold, replace tested attribute);
- Prune the internal node and transform it into the leaf with a new multivariate linear regression model;
- Expand the leaf into the internal node;
- Replace one of the following: subtree, branch, node, or test between two affected individuals;
- Modify linear regression models in the leaves (add, remove, or change attributes).

The selection mechanism is based on the ranking linear selection [37] with the *elitist strategy*, which copies the best individual founded so far to the next population.

Selection acts as a force that increases the quality of the population. For this reason, the mechanism of selection usually requires a quantitative measure to assess individuals in the population. Such an objective function is called a fitness function and in practice, it is a very important and sensitive element of the evolutionary system. It drives the evolutionary search process by measuring how good a single individual is in terms of meeting the problem objective. In the context of decision trees, a direct minimization of the prediction performance measured on the learning set leads to the over-fitting problem and poor performance on unseen testing instances. Therefore, there is a need to balance the predictive error and the complexity of the tree. In *GMT*, the authors adapt the Bayesian Information Criterion (*BIC*) [43] as a fitness function. The *BIC* fitness is equal to:

$$Fit_{BIC}(T) = -2 * ln(L(T)) + ln(n) * k(T), \tag{4}$$

where $L(T)$ is the tree T maximum of likelihood function, $k(T)$ is the complexity term, and n equals the number of instances. The function $L(T)$ is common for regression models [21] and equals:

$$ln(L(T)) = -0.5n * [ln(2\pi) + ln(SS_e(T)/n) + 1]. \tag{5}$$

The term $SS_e(T)$ is the sum of squared residuals of the tree T (on the training set).

3.2. Cost-sensitive transformation

There are a few steps to transform the regular model tree into a cost-sensitive one. First, the authors adapt the average misprediction cost (*AMC*) [5] as a measurement for assessing the performance of the proposed method. Consider a dependent variable, y, that is predicted based on a vector of independent variables x. The regression method learns the prediction model, $f : x \rightarrow y$ from the training data $S = \{< x_i, y_i > | i = 1, 2, \ldots, N\}$. *AMC* can be defined as:

$$AMC = \frac{1}{N} \sum_{1}^{N} C(f(x_i) - y_i), \tag{6}$$

where $C(e)$ is a function that characterizes the cost and e is a prediction error. *AMC* measurements can be viewed as a more general version of

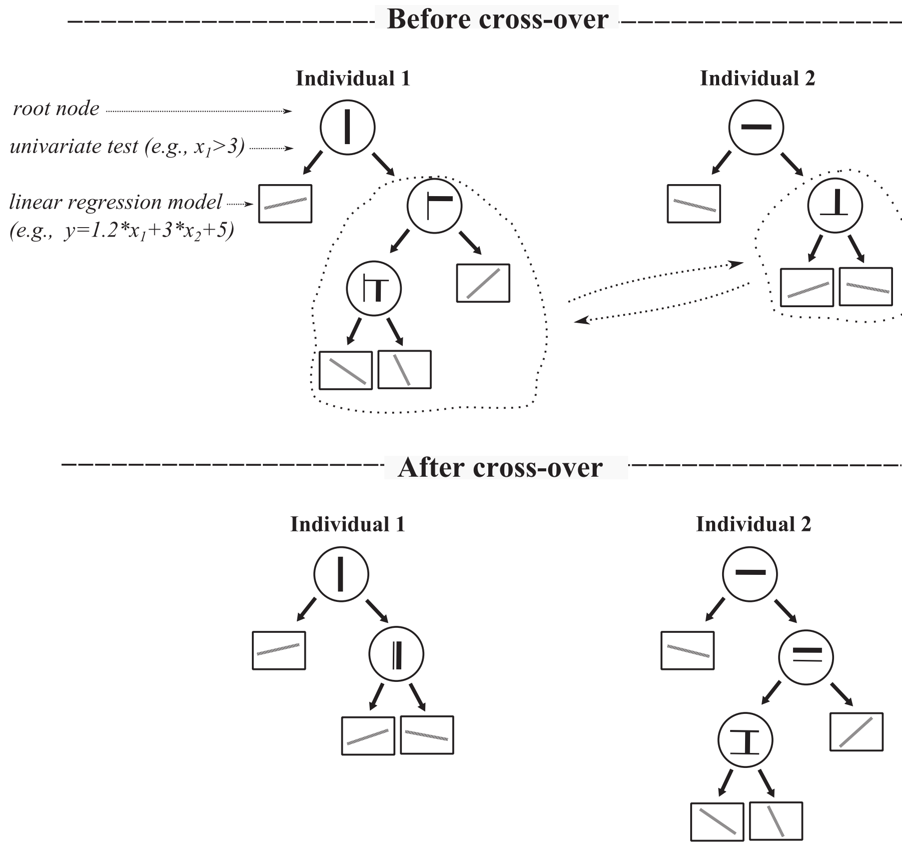

Fig. 4. Crossover between two individuals and their resulting offspring.

the mean absolute error (*MAE*). When the costs for under-prediction and over-prediction are equal, the loss function is symmetric (Eq. (2)) and *AMC* = *MAE*.

In *CGMT*, we adapted the cost-sensitive *BIC*-like fitness function proposed in [16]. The authors replaced $SS_e(T)$ from Eq. (5) with *AMC* from Eq. (6). This way, the fitness function will measure how good a single individual is in terms of meeting the cost-sensitive problem objective. The complexity term, $k(T)$, works as a penalty for over-parameterization of the tree *T*. The penalty term:

$$k(T) = 2 * (Q(T) + M(T)), \tag{7}$$

where $Q(T)$ is the number of internal nodes and $M(T)$ is the number of attributes that build models in the leaves. In this paper, we updated the parameter $k(T)$ in comparison to [16]. Performed experiments showed that this higher value of complexity term works better with *CGMT* because it reduces the tree size and improves the generalizability.

With the all cost-sensitive transformations proposed in this paper, the final form of the fitness function for *CGMT* is as follows:

$$Fit_{\mathrm{CGMT}}(T) = n[\ln(2\pi) + \ln(AMC(T)/n) + 1] + 2\ln(n) * (Q(T) + M(T)), \tag{8}$$

where in the first element of equation depends on the tree average misprediction cost measured on the learning set and the second part reflects the tree complexity.

To allow *CGMT* to work with any convex cost function (e.g., *LinLin*, *QuadQuad*, *LinEx*, and *SquarEx*) we had to develop a universal way to search for cost-sensitive models in the leaves of the tree. We proposed an adjustment of regression models and different variants of mutation operator together with modified fitness function, thereby efficiently converting cost-neutral model trees into cost-sensitive ones.

3.2.1. Model adjustment for any convex cost function

One of the ways to modify the regression model in the leaf into a cost-sensitive one is to shift it toward a smaller *AMC* error. We will denote the desired shift as θ. This strategy works well in the post hoc tuning methods proposed in [5] and later in [56]. To find regression model adjustment, the authors used a heuristic approach such as the hill climbing algorithm. A similar idea was proposed in [16], where each regression model was shifted by a certain amount using *EA*.

For convex cost functions like *QuadQuad*, *LinEx*, and *SquarEx*, θ cannot be calculated directly and needs to be approximated. In *CGMT*, we simply checked between which two instances the *AMC* error was minimized. Next, the algorithm selects θ randomly as long as two instances are on the different sides of the regression line (one instance should be under-predicted and the other should be over-predicted). After calculating the adjustment, the new regression model in the leaf is equal to:

$$f_{new}(x) = f(x) + \theta, \tag{9}$$

where θ is the shift. Fig. 5 shows an example of actual $f_0(x)$ and the shifted regression model $f_1(x)$.

In *CGMT*, the adjustment of the regression model in the leaf is applied after any modifications of the regression model. Changes in regression models can be caused either by mutation or crossover operators.

3.2.2. Model adjustment for LinLin

Finding adjustment θ with *EA* is time consuming as besides searching for the optimal tree structure, splits in internal nodes, and regression models in the leaves, *EA* also looks for the value that each model should be shifted. However, in case of *LinLin* cost function, the adjustment θ can be calculated directly.

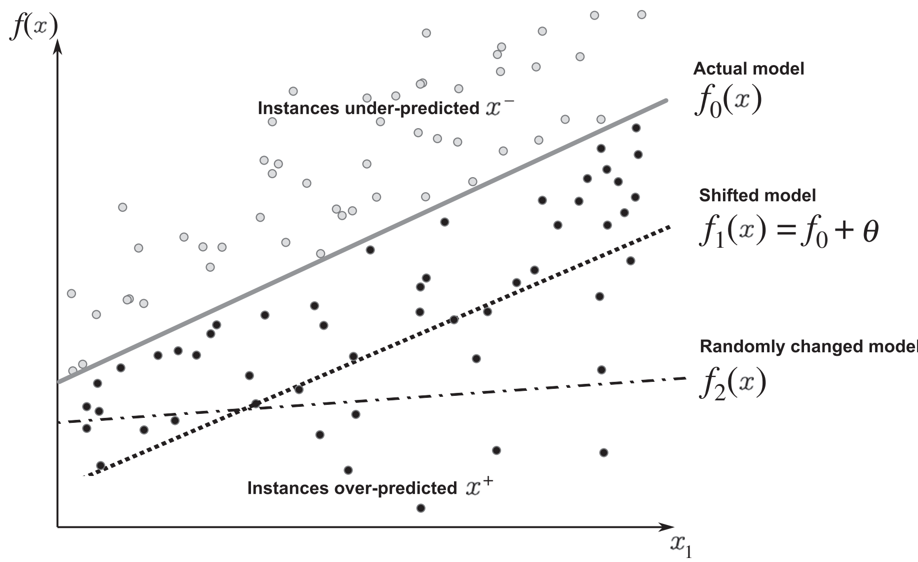

Fig. 5. An example of the linear regression model $f_0(x)$ shifted by θ: $f_1(x)$ and the new, randomly changed model: $f_2(x)$.

Let x^- denote under-predicted instances and x^+ denote over-predicted instances by an actual regression model in the leaf. The cost of under-prediction and over-prediction equals C^- and C^+, respectively. If the AMC^+ denotes the average misprediction cost of over-predicted instances (x^+) and AMC^- the under-predicted ones (x^-), then:

$$AMC = AMC^+ + AMC^-. \tag{10}$$

In *LinLin* loss function, which minimizes the absolute error (Eq. (2)), the problem of searching for θ is reduced to finding appropriate regression quantile that will split instances according to their costs and cardinality [28]. In our case, we know the weights given to positive (over-predicted) and negative (under-predicted) instances. To minimize Eq. (10), we need to have equal weight differences between over-predicted and under-predicted instances; therefore, we need to solve:

$$N^+ * C^+ - N^- * C^- = 0, \tag{11}$$

where N^+ and N^- denote the number of over-predicted and under-predicted instances in the adjusted model, respectively. Considering that $N^+ + N^- = N$, we can estimate the number of instances that should be over-predicted and under-predicted:

$$\hat{N}^+ = \frac{n * C^-}{C^- + C^+} \qquad \hat{N}^- = \frac{n * C^+}{C^- + C^+}. \tag{12}$$

The final step is to calculate model adjustment, which equals:

$$\theta = y_k - f(x_k), \tag{13}$$

where k is the N^+-th instance in order starting from the most under-predicted instances to the most over-predicted. The instance k can also be calculated as the N^--th one in order, sorted from the most over-predicted instances to the most under-predicted, as this would return the same instance. However, the estimated value $\hat{N}^+$ from Eq. (12) does not have to be an integer; therefore, we have to consider two variants:

- If $\hat{N}^+$ is an integer, then $N^+ = \hat{N}^+$,
- If $\hat{N}^{X+}$ is not an integer, then: $N^+ = \begin{cases} floor\left(\hat{N}^+\right) & if \quad C^- > C^+ \\ ceil\left(\hat{N}^+\right) & if \quad C^- < C^+ \end{cases}.$

With knowledge of the exact value of over-predicted and under-predicted instances in the adjusted model, we can calculate the shift from Eq. (13). It can be observed that when $N^+ = \hat{N}^+$, the θ has the same value in the range $< y_k - f(x_k), \quad y_{k+1} - f(x_{k+1}) >$.

3.2.3. Random model changes

Shifting regression models in the leaves to make them cost-sensitive is one of the options. Another one is to modify the coefficients in linear regression models and leave the search for the optimal one to the *EA*. We randomly selected a single coefficient and modified it (increased or decreased it) by a random percent from 0 % to 100 %. Next, we adjusted the regression model with the above-mentioned algorithm. Such a technique, together with a mutation variant that added, removed, or changed the attributes in the models in the leaves, allowed us to find completely new and cost-sensitive regression models. It can also work with any convex cost function like *LinLin*, *QuadQuad*, *LinEx*, *SquarEx*, etc. Fig. 5 illustrates actual $f_0(x)$ and the new, randomly modified regression model denoted as $f_2(x)$.

3.3. Discussion of the improvements

In the paper, we have extended the *GMT* solution [17] to work as a cost-sensitive learner. The original *GMT* is cost-neutral algorithm with symmetric loss function and therefore, couldn't be applied to the data with different prediction costs. To transform *GMT* to the cost-sensitive learner, the new fitness function that involve costs of the under-predictions and over-prediction was defined. Next, new variants of mutations were proposed that improve the search of the cost-sensitive regression models in the leaves. Modifications of the initialization the individuals in the population was also performed as only the long dipolar strategy [16] was applied to search for the splits in the internal nodes. Experimental results presented in the next section show that there is a huge gap between *GMT* and *CGMT* when cost-sensitive data is analyzed.

In the literature, there is also an attempt to transform *GMT* to the cost-sensitive learner denoted as GMT_{CS} [16]. There are, however, significant differences between proposed *CGMT* solution and the GMT_{CS}. In particular:

- An evolutionary search of the regression model adjustments for the *LinLin* cost function in GMT_{CS} was replaced by the analytical formula. This increased the convergence of evolutionary algorithms and significantly improved the prediction accuracy of the regression models;
- New mutation operators that perform random changes in the regression models in the leaves were introduced. This allowed *CGMT* to

explore a larger solution space and fit more accurate cost-sensitive linear regression models into the instances in the leaves;

- The previous GMT_{CS} algorithm works only with the *LinLin* cost function. The *CGMT* solution works with any convex cost functions like *LinLin*, *QuadQuad*, *LinEx*, *SquarEx*, etc.;
- The fitness function of the *CGMT* was improved to increase the algorithm's generalization ability;
- More detailed experimental analysis of loan charge-off predictions was performed, and *LinLin* and *QuadQuad* loss functions were investigated. Presented results in the next section show that *CGMT* managed to significantly outperform (P $value < 0.0001$) in terms of error reduction the GMT_{CS} solution.

4. Experiments

In this section, we present an experimental validation of the proposed approach called *CGMT* and its competitors on a loan charge-off forecasting problem.

4.1. Datasets

The proposed solution is applied to a loan charge-off forecasting problem that is characterized by asymmetric costs, as over-prediction is less costly than under-prediction. We focus on financial institutions in the USA from 2004 to 2010 and use the data from Wharton Research Data Services [55]. The average number of banks analyzed in this study was 7, 695 — from 6, 992 to 8, 315. In each quarter of the year, the banks needed to forecast the amount of loan charge-off.

In the experiments, we preprocessed the data in the same way as other authors that have dealt with this forecasting problem [5,56,16]. The instances with missing values were removed and the natural logarithm transformation was performed on the dependent variable to reduce the extent of skewness.

Each quarter of the year was predicted separately with 14 attributes related to the current financial information of the bank. To predict the loan charge-off in the following quarter, the attributes presented in Table 1 were used. From the table, we can observe that one of the independent variables is the loan charge-off of the previous quarter. One can also notice that no e.g., macroeconomic variables and unemployment information were used. These attributes could have had an impact on the amount of charge-off loans, however, to compare *CGMT* with previous research, we used the same set of independent variables.

From 28 quarters, 27 datasets were generated, as for the newest quarter the predicted charge-off value for *I* 2011 was not accessible in *WRDS*. Each algorithm was trained on a single (training) quarter and then tested on the following (testing) quarter. In this way, 26 independent training and testing sets were used in the experiments.

4.2. Setup

We followed the experimental validation performed in [56]. Thanks to Prof. Huimin Zhao, who provided us with the source code of his tuning methods denoted as *BSZ* [5] and *Linear* [56], we were able to confront the prediction performance of all tested algorithms. We attached the average misprediction cost (*AMC*) for three base regression models: standard least-squares linear regression (*LR*), the *M*5 model tree [52], and the back-propagation neural network (*NN*) [44], which were also tested in [5,56,16].

In all reported experiments, the algorithms ran with their default settings. Algorithms *LR*, *M*5, and *NN* were tested using the *Weka* system [25]. The settings of *CGMT* were as recommended in *GMT* [17]: the population size was 50, the probability of the mutation single node was 0.8, and the probability of the crossover between two individuals was 0.2.

We also presented the results of the cost-neutral *GMT* algorithm and proposed the *CGMT* solution. Tests were performed for two different cost functions; *LinLin* and *QuadQuad*, and different cost rations (*CR*) for under-prediction to over-prediction were as follows: 10:1, 20:1, 50:1, and 100:1. For the results with *LinLin* cost function, we also included the GMT_{CS} results [16] (the algorithm does not work with the *QuadQuad* loss function). Table 2 illustrates an overview of the performed experiments, algorithms, and tested settings.

Table 1
The variable codes, names, and definitions from WRDS used in the experimental evaluation. [55]

No.	Code	Name	Definition
1	RCFD1400	Total loans and leases, gross	The aggregate gross book value of total loans together with valuation reserves.
2	RCFD1403	Total loans and lease finance receivables: nonaccrual	Includes: the outstanding balances of loans; lease financing receivables status; all restructured loans; lease financing receivables (all with nonaccrual status).
3	RCFD1407	Total loans and lease financing receivables: past due 90 days or more and still accruing	Includes: all restructured loans and leases; loans and lease financing receivables on which payment is due and unpaid for 90 days or more.
4	RCFD2143	Intangible assets	Includes the unamortized amount of intangible assets.
5	RCFD2170	Total assets	The sum of all asset items.
6	RCFD3163	Goodwill	Includes the amount (book value) of unamortized goodwill.
7	RCFD3200	Subordinated notes and debentures	Includes the amount of outstanding subordinated notes and debentures (including mandatory convertible debt).
8	RCFD3210	Equity capital, total	The sum of “Perpetual Preferred Stock and Related Surplus”, “Common Stock”, “Surplus”, “Undivided Profits and Capital Reserves”, “Cumulative Foreign Currency Translation Adjustments” less “Net Unrealized Loss on Marketable Equity Securities”.
9	RIAD4010	Interest and fee income on loans, total	Includes the total of interest and fee income and similar charges levied against all assets classified as loans in condition reports, including fees on overdrafts.
10	RIAD4079	Total noninterest income	Includes the sum of “Income from Fiduciary Activities”, “Service Charges on Deposit Accounts in Domestic Offices”, “Trading Gains (Losses) and Fees from Foreign Exchange Transactions”, “Other Foreign Transaction Gains (Losses)”, “Gains (Losses) and Fees from Assets Held in Trading Accounts”, “Service Charges on Deposit Accounts” and “Other Noninterest Income”.
11	RIAD4180	Expense of federal funds purchased and securities sold under agreements to repurchase	Includes the gross expense of all liabilities included in “Federal Funds Purchased and Securities Sold Under Agreements to Repurchase”.
12	RIAD4340	Net income (loss)	Includes the “Net Income (Loss)” for the period.
13	RCFDA223	Risk-weighted assets (net of allowances and other deductions)	The amount of the bank's risk-weighted assets net of all deductions.
14	RIAD4635	Charge-offs on allowance for loan and lease losses	The amount of gross charge-offs on loans and leases during the calendar year-to-date.

Table 2
An overview of *GMT* experimental validation.

Original	Algorithms		Cost Settings	
	Tuned BSZ	Tuned Linear	Ratio	Function
LR	LR_{BSZ}	LR_{Linear}	10	LinLin
M5	$M5_{BSZ}$	$M5_{Linear}$	20	QuadQuad
NN	NN_{BSZ}	NN_{Linear}	50	
GMT	GMT_{BSZ}	GMT_{Linear}	100	
GMT_{CS}	–	–		
CGMT	$CGMT_{BSZ}$	$CGMT_{Linear}$		

4.3. Comparison results

Table 3 summarizes the *AMC* results for *CGMT*, three state-of-the-art regression solutions, and the cost-neutral model tree denoted as *GMT*. In Table 3 we also show the impact of two post hoc cost-sensitive tuning methods: *BSZ* [5] and *Linear* [56]. In the experiments, four cost ratio (*CR*) and two different loss functions (*LinLin* and *QuadQuad*) are applied. We report an average value that algorithms achieved on 26 independent testing sets. In case of *CGMT*, the *AMC* result equals an average value of 20 runs; therefore, the standard deviation is included.

From the results in Table 3, we see a huge difference between cost-neutral and cost-sensitive algorithms. Post hoc tuning methods significantly reduce *AMC*, and the *Linear* variant is almost always better than *BSZ*. We can observe that the best performance of *CGMT* competitors is achieved by GMT_{Linear} and NN_{Linear}. The poor performance of the *M5* model tree is a result of over-fitting and the biases of the linear models in the leaves for outliers in the analyzed datasets. It is especially visible when the *QuadQuad* loss function is applied. An additional study of *M5* performance showed that errors on a few under-predicted instances was responsible for such poor results (on one of the instances, the *AMC* was near 1, 000, 000, which equals under-prediction of the loan charge-off for *QuadQuad* loss function with $Cost\ \ ratio = 100$). When those outliers were eliminated, *M5* managed to obtain slightly better results than *LR*. In Table 3, we can see that the post hoc tuning of *CGMT* (algorithms $CGMT_{BSZ}$ and $CGMT_{Linear}$) has no impact on the results. It proves that there is no more space for improvement for *CGMT* with examined tuned methods.

The average *AMC* reduction between the best post hoc tuned algorithm from Table 3 and *CGMT* varies between 11.8 % and 28.3 %. The real improvement is much higher because the results are on a natural logarithm scale that varies between 44.6 % and 99.9 %. We believe that such error reduction could not be ignored by the institutions struggling with the loan charge-off forecasting. The *QuadQuad* loss function and cost ratio equals 100, and the *AMC* for NN_{Linear} equals 14, 831, 216, 212 ($e^{23.4}$), where for *CGMT*, the *AMC* is almost four orders of magnitude smaller (9, 530, 426).

It can be expected that searching for cost-sensitive models directly during induction results in finding better solutions. The first attempt of *CS*-extensions for *GMT* applied to *LinLin* (algorithm GMT_{CS}) [16] managed to decrease the *AMC* and outperformed all tuned base regression models under every cost ratio. However, our algorithm goes even further — it significantly decreases the *AMC* in comparison to all tested solutions. The *CGMT* solution can work with any convex function (in contrast to GMT_{CS}). The comparison results are enclosed in Table 4. The average cost reduction between the GMT_{CS} and *CGMT* is in the range of 9.4 % to 15.4 % when the cost values are on a natural log scale. Therefore, the real cost reduction on the original scale is in the range of 26.7 % to 55.5 %. Unfortunately, we could not compare GMT_{CS} with *CGMT* using the *QuadQuad* loss function. However, the *AMC* reduction for *CGMT* is expected to be higher than for GMT_{CS}.

The Wilcoxon signed rank test between *CGMT* and GMT_{CS} proved that the differences between the algorithms are statistically significant ($P\ value < 0.0001$). In addition, we have also tested within *Weka* software two nonlinear regression models: *RBF Network* [9] that implements a normalized Gaussian radial basis function network and *SMOreg* [46] that is the support vector machine for regression (with polynomial kernel). Both tuned algorithms perform similar to GMT_{Linear}, and NN_{Linear} and together with the rest of tested methods, they were statistically worse than the *CGMT* solution.

Table 3
Average misprediction cost (*AMC*) for *CGMT*, *GMT*, and three popular regression algorithms with and without post hoc tuning.

Algorithm	*CR*	*LinLin* loss function			*QuadQuad* loss function		
		No tuning	BSZ	Linear	No tuning	BSZ	Linear
LR	10:1	7.41	3.78	3.81	22.39	12.64	11.80
M5	10:1	7.29	4.16	3.88	139.03	129.60	97.21
NN	10:1	8.16	3.69	3.57	23.74	11.11	10.43
GMT	10:1	7.07	3.81	3.65	19.93	10.99	9.91
CGMT	10:1	2.98	2.98	2.98	8.75	8.77	8.74
± (stdev)	–	0.16	0.16	0.16	1.30	1.31	1.30
LR	20:1	14.06	4.84	4.42	42.59	17.94	15.73
M5	20:1	13.78	5.47	4.86	162.22	137.21	94.43
NN	20:1	15.60	4.62	4.26	45.37	15.11	13.56
GMT	20:1	13.66	5.07	4.27	39.91	14.66	13.13
CGMT	20:1	3.43	3.43	3.43	10.54	10.53	10.54
± (stdev)	–	0.19	0.19	0.19	2.33	2.33	2.33
LR	50:1	34.02	6.24	5.23	103.19	28.40	22.64
M5	50:1	33.23	6.03	6.44	231.73	154.80	97.72
NN	50:1	37.92	5.69	5.09	110.34	22.09	18.62
GMT	50:1	32.96	6.40	6.11	95.86	22.93	19.93
CGMT	50:1	4.02	4.02	4.02	13.27	13.22	13.27
± (stdev)	–	0.25	0.25	0.25	2.43	2.41	2.43
LR	100:1	67.27	7.06	5.85	204.19	40.41	30.10
M5	100:1	65.66	7.24	7.94	347.60	177.92	107.73
NN	100:1	75.12	6.50	5.80	218.42	29.03	23.42
GMT	100:1	64.15	7.03	5.98	180.43	25.09	22.41
CGMT	100:1	4.46	4.46	4.46	16.08	16.01	16.07
± (stdev)	–	0.36	0.36	0.36	2.91	2.90	2.91

5. Discussion

In Tables 3 and 4, we showed the average error results for 26 datasets. To prove that the *CGMT* algorithm is consistent in reducing the *AMC* error, we present, in detail, the performance of the top three algorithms: NN_{Linear}, GMT_{CS}, and *CGMT*. In the first experiment in Tables 3 and 4, the *LinLin* loss function was used, and the cost ratio was equal to 10. With these settings, the *AMC* reduction for *CGMT* was the lowest due to a small penalty for wrong predictions. It is, however, still significant. The results illustrated in Fig. 6 show that the proposed solution is almost always better (and never worse) than NN_{Linear} and GMT_{CS} on all 26 tested datasets. The cost values are on a natural log scale, so, the real differences between *CGMT* and other solutions are much higher.

We can also observe that the results of all algorithms strongly depend on the quarter of the year. One of the reasons for poor predictions at the beginning of each year is that the banks changed the cost-ratio for this particular *I* quarter. The new year is always a difficult time to forecast financial data, as all institutions need to publish their annual reports, thus revealing their annual macroeconomic data. In

Table 4
Average misprediction cost (*AMC*) for GMT_{CS} and *CGMT*. Algorithms used the *LinLin* loss function.

Algorithm	$CS = 10$	$CS = 20$	$CS = 50$	$CS = 100$
GMT_{CS}	3.29	3.85	4.66	5.27
CGMT	2.98	3.43	4.02	4.46

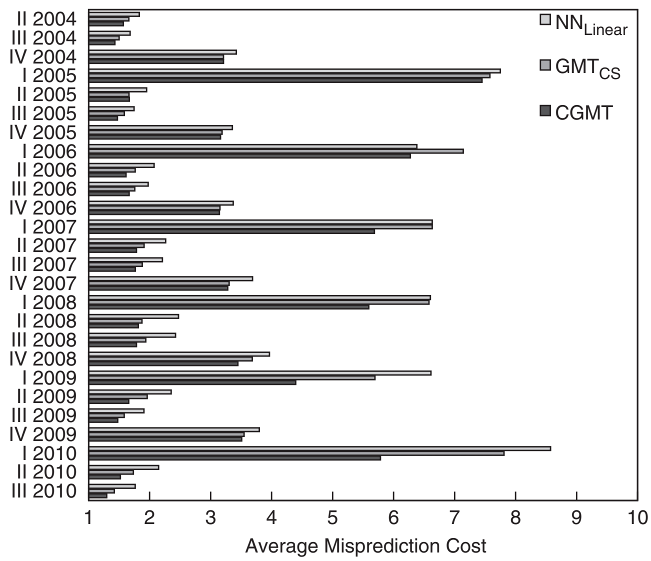

Fig. 6. Average misprediction cost for NN_{Linear}, GMT_{CS}, and *CGMT* on datasets with a cost ratio equal to 10. Algorithms use the *LinLin* loss function.

these uncertain times, banks increase their reserves to minimize the risk of not covering their loan losses. However, seasonality affects quarterly data for accounting as well as for economic reasons. For example, in [33], the authors show that loan provisions are often delayed to the *IV* fiscal quarter when the audit occurs. This factor would explain the constant bad performance of algorithms in the *IV* quarter, especially when prediction models are built using the previous values of *Total Loan Charge-offs* (RIAD4635).

Besides prediction accuracy, there is one more important factor in loan charge-off forecasting problems. Business analysts need a solution that can provide new insights into underlying process that they then try to understand. Fig. 7 illustrates an output tree for *CGMT* for the first quarter of 2004 with a *LinLin* loss function and cost ratio equal to 10.

The output tree of the *CGMT* algorithm is compact and can be easily analyzed and interpreted, even in this form. Internal node tests consider previous values of *Total Loan Charge-offs* (RIAD4635) and *Total Loans Not Accruing* (RCFD1403). Models in the leaves also consider *Noninterest Income* (RIAD4079) and *Loans* 90 + *Days Late* (RCFD1407). To calculate the real value of the searched loan charge-off, the exponential value of prediction should be taken as the loan charge-off is on a natural log scale. In addition, all attributes (except for RIAD4635, which is on a natural log scale) represent the amount in thousands of dollars. Therefore, for example, the first split checks if the loan charge-off is greater than $391, 505, and if that is true, the predicted loan charge-off for the tested bank can be calculated from the equation:

$$y = e^{0.94*RIAD4635+1.43}. \tag{14}$$

The *AMC* of this particular tree equals 1.545 and the tree has five linear models with an average of 1.8 attributes in each linear model. For this particular dataset (*I* 2004), the $M5_{Linear}$ output tree has an *AMC* equal to 1.80, and the output tree is larger (19 leaves) and uses an average of 12.1 attributes (from available 14) in each linear model. To top it all off, the post hoc *Linear* tuning method must be applied; therefore, each prediction of new instances must be multiplied by 0.83, and a value of 1.88 must be added. In this way, the *M*5 algorithm, which is generally considered as a "white box" solution becomes a model that is really difficult to understand, analyze, and interpret.

A small number of leaves and attributes in the linear models is crucial to understanding the relationships in the data. It should be noticed that the whole tree (tests in internal nodes and models in the leaves) was designed to solve a cost-sensitive problem in the learning phase by *EA*. The fitness function of *CGMT*, with its penalty term for overgrown (number of leaves) and complicated (number of attributes in models) trees, keeps output solutions as small and simple as possible. Although induced trees by the *CGMT* system are different for each quarter, there are similarities and patters between them that could be helpful for bank analysts. In all performed experiments with different cost ratios and loss functions, the trees induced by *CGMT* have an average of 8.5 leaves with 1.47 attributes in each linear model. The output trees for *CGMT* are in general three times smaller and have eight times fewer attributes in linear regression models than the *M*5 solution.

Finally, the evolutionary approaches are not the fastest ones, and the *CGMT* is not an exception. The average training time for the proposed solution was from a few minutes to an hour for each dataset which, considering the amount of data, is not a bad result. The execution of the *CGMT* solution on test instances is very fast.

6. Conclusion and future works

In this paper, we proposed a cost-sensitive solution for model trees. Specialized evolutionary algorithms that can work with any convex cost

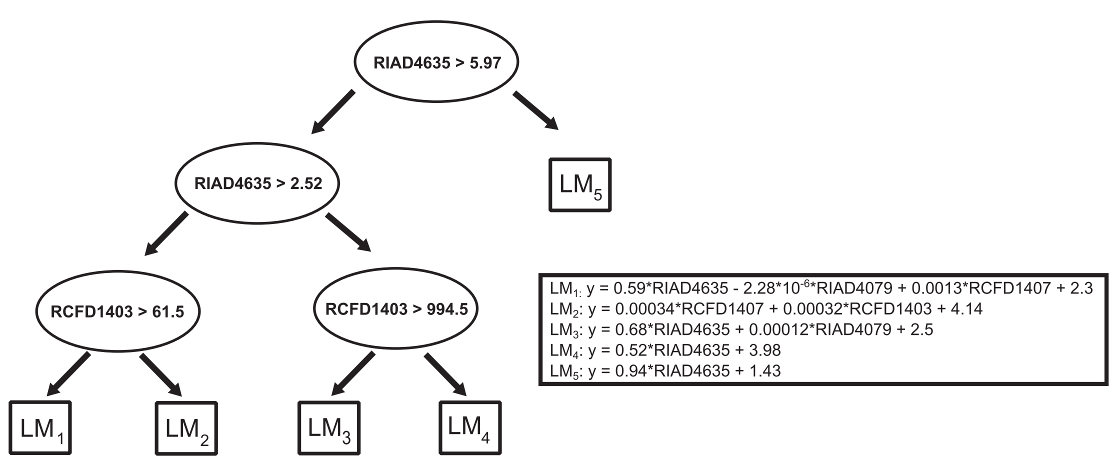

Fig. 7. The induced *CGMT* for the first quarter of 2004, with cost ratio equal to 10 and *LinLin* loss function.

function incorporated into the training process successfully dealt with forecasting problems where under-prediction and over-prediction errors had different consequences. With the memetic operators and cost-sensitive fitness function, we managed to efficiently evolve simple model trees that minimized the average misprediction cost.

Experimental validation was performed on 26 independent datasets and over 200,000 instances were tested. We proved that the proposed solution, *CGMT*, is significantly better than all post hoc tuned methods, as well as other evolutionary approaches, such as GMT_{CS}. The true reduction of cost error varies between 26.7% and 99.9%. However, the improvement was not only in the prediction accuracy. Equally if not more important is the fact that the decisions and models induced by *CGMT* can have direct applicability. They are simple, can be easily understood, and consider costs of errors during the learning phase.

We see many promising directions for future research. In particular, we are focused on extending *CGMT* to work with multiple costs, such as the cost of attributes. We also want to apply *CGMT* to different real-life forecasting problems.

Acknowledgments

The authors want to thank Huimin Zhao, Atish P. Sinha, and Gaurav Bansel for providing the implementation of *BSZ* and *Linear* tuning methods. This project was funded by the Polish National Science Centre allocated on the basis of decision no. 2013/09/N/ST6/04083.

References

[1] M. Alexander, N.A. Christakis, Bias and asymmetric loss in expert forecasts: a study of physician prognostic behavior with respect to patient survival, Journal of Health Economics 27 (4) (2008) 1095–1108.
[2] K. Aretz, S.N. Bartram, P.F. Pope, Asymmetric loss functions and the rationality of expected stock returns, International Journal of Forecasting 27 (2) (2011) 413–437.
[3] R.C. Barros, D.D. Ruiz, M. Basgalupp, Evolutionary model trees for handling continuous classes in machine learning, Information Sciences 181 (2011) 954–971.
[4] R.C. Barros, M.P. Basgalupp, A.C. Carvalho, A.A. Freitas, A survey of evolutionary algorithms for decision-tree induction, IEEE Transactions on Systems, Man, and Cybernetics Part C 42 (3) (2012) 291–312.
[5] G. Bansal, A.P. Sinha, H. Zhao, Tuning data mining methods for cost-sensitive regression: a study in loan charge-off forecasting, Journal of Management Information Systems 25 (3) (2008) 317–338.
[6] R. Berk, Asymmetric loss functions for forecasting in criminal justice settings, Journal of Quantitative Criminology 27 (1) (2011) 107–123.
[7] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth Int, Group, 1984.
[8] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, C.E. Brodley, Pruning Decision Trees with Misclassification Costs, Proc. of ECML, Springer 1998, pp. 131–136.
[9] M.D. Buhmann, Radial Basis Functions: Theory and Implementations, Cambridge University, 2003.
[10] M. Cain, C. Janssen, Real estate price prediction under asymmetric loss, Annals of the Institute of Statistical Mathematics 47 (3) (1995) 401–414.
[11] X.S. Chen, Y.S. Ong, M.H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 591–607.
[12] S.F. Crone, S. Lessmann, R. Stahlbock, Utility Based Data Mining for Time Series Analysis: Cost-sensitive Learning for Neural Network Predictors, Proc. of UDBM, Chicago 2005, pp. 59–68.
[13] M. Czajkowski, M. Kretowski, An Evolutionary Algorithm for Global Induction of Regression Trees with Multivariate Linear Models, Proc. of ISMIS, Lecture Notes in Artificial Intelligence, 6804 2011, pp. 230–239.
[14] M. Czajkowski, M. Kretowski, Does memetic approach improve global induction of regression and model trees? Lecture Notes in Computer Science 7269 (2012) 174–181.
[15] M. Czajkowski, M. Kretowski, An evolutionary algorithm for global induction of regression and model trees, International Journal of Data Mining, Modelling and Management 5 (3) (2013) 261–276.
[16] M. Czajkowski, M. Czerwonka, M. Kretowski, Cost-Sensitive Extensions for Global Model Trees, Application in Loan Charge-Off Forecasting, Proc. of ICSS, Advances in Intelligent Systems and Computing, 400 2014, pp. 315–324.
[17] M. Czajkowski, M. Kretowski, Evolutionary induction of global model trees with specialized operators and memetic extensions, Information Sciences 288 (2014) 153–173.
[18] P. Domingos, MetaCost: A General Method for Making Classifiers Cost-sensitive, Proc. of KDD, ACM Press 1999, pp. 155–164.
[19] C. Elkan, The Foundations of Cost-Sensitive Learning, Proc. of IJCAI 2001, pp. 973–978.
[20] G. Fan, J.B. Gray, Regression tree analysis using target, Journal of Computational and Graphical Statistics 14 (1) (2005) 206–218.
[21] P. Gagne, C.M. Dayton, Best regression model using information criteria, Journal of Modern Applied Statistical Methods 1 (2002) 479–488.
[23] R. Goetschalckx, K. Driessens, Parsimonious Linear Model Trees, Proc. of ICML Workshop on Machine Learning and Games, 2010.
[24] C.W.J. Granger, Forecasting in Business and Economics, 2nd ed. Academic Press, New York, 1989.
[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA Data Mining Software: An Update, SIGKDD Explorations 11 (1) (2009).
[28] R.W. Koenker, Quantile Regression, Cambridge University Press, 2005.
[29] S.B. Kotsiantis, Decision trees: a recent overview, Artificial Intelligence Review, 39, Springer Netherlands, 2013. 261–283.
[30] M. Kretowski, GrześM. , Evolutionary Induction of Cost-Sensitive Decision Trees, Proc. of ISMIS, Lecture Notes in Artificial Intelligence, 4203 2006, pp. 121–126.
[32] Y. Lee, P.J. Hu, T. Cheng, Y. Hsieh, A cost-sensitive technique for positive-example learning supporting content-based product recommendations in B-to-C e-commerce, Decision Support Systems 53 (1) (2012) 245–256.
[33] C. Liu, S.G. Ryan, J. Wahlen, Differential valuation implications of loan loss provisions across banks and fiscal quarters, The Accounting Review 72 (1) (1997) 133–146.
[34] S. Lomax, S. Vadera, A survey of cost-sensitive decision tree induction algorithms, ACM Computing Surveys 45 (2) (2013) 16.
[35] G. Ma, E. Song, C. Hung, L. Su, D. Haung, Multiple costs based decision making with back-propagation neural networks, Decision Support Systems 52 (3) (2012) 657–663.
[36] D. Malerba, F. Esposito, M. Ceci, A. Appice, Top-down induction of model trees with regression and splitting nodes, IEEE Transaction on PAMI 26 (5) (2004) 612–625.
[37] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed. Springer, 1996.
[39] G.E. Naumov, NP-completeness of problems of construction of optimal decision trees, Soviet Physics Doklady 36 (4) (1991) 270–271.
[40] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in C, Cambridge University Press, 1988.
[41] L. Rokach, O.Z. Maimon, Top-down induction of decision trees classifiers — a survey, IEEE Transactions on Systems, Man, and Cybernetics Part C 35 (4) (2005) 476–487.
[42] L. Rokach, O.Z. Maimon, Data mining with decision trees: theory and application, Machine Perception Artificial Intelligence 69 (2008).
[43] G. Schwarz, Estimating the dimension of a model, The Annals of Statistics 6 (1978) 461–464.
[44] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, Parallel Distributed Processing, MIT Press, Cambridge, MA, 1986. 318–362.
[45] H. Shefrin, M. Statman, The disposition to sell winners too early and ride losers too long: theory and evidence, Journal of Finance 40 (1985) 777–790.
[46] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, Improvements to the SMO Algorithm for SVM Regression, IEEE Transactions on Neural Networks (2000) 1188–1193.
[47] K. Ting, An instance-weighting method to induce cost-sensitive trees, IEEE Transactions on Knowledge and Data Engineering 14 (3) (2002) 659–665.
[48] L. Torgo, R. Ribeiro, Utility-based regression, Proc. of ECML/PKDD, Lecture Notes in Computer Science, 4702 2007, pp. 597–604.
[49] L. Torgo, R. Ribeiro, Precision and Recall for Regression, Proc. of DS, Berlin, Heidelberg 2009, pp. 332–346.
[50] P. Turney, Types of cost in inductive concept learning, Proc. of ICML Workshop on Cost-Sensitive Learning. Stanford, CA, 2000.
[51] A. Tversky, D. Kahneman, Loss aversion in riskless choice: a reference dependent model, Quarterly Journal of Economics 106 (1991) 1039–1061.
[52] J. Quinlan, Learning with Continuous Classes, Proc. of AI, World Scientific 1992, pp. 343–348.
[53] H.R. Varian, A Bayesian Approach to Real Estate Assessment, Studies in Bayesian Econometrics and Statistics: In honor of L.J. Savage (North-Holland), Amsterdam 1974. 195–208.
[54] D. Vogel, O. Asparouhov, T. Scheffer, Scalable look-ahead linear regression trees, Proc. of ACM SIGKDD, ACM Press New York 2007, pp. 757–764.
[55] Wharton Research Data Services, http://wrds-web.wharton.upenn.ed.
[56] H. Zhao, A.P. Sinha, G. Bansal, An extended tuning method for cost-sensitive regression and forecasting, Decision Support Systems 51 (2011) 372–383.
[57] H. Zhao, A.P. Sinha, W. Ge, Effects of feature construction on classification performance: an empirical study in bank failure prediction, Expert Systems with Applications 36 (2) (2009) 2633–2644.

Marcin Czajkowski obtained his Ph.D. with honors in Computer Science from Bialystok University of Technology, in 2015. He has published papers in multiple peer-reviewed journals and conferences. His current research interests are data mining, machine learning and bio-inspired computation.

Monika Czerwonka is an assistant professor in the Institute of Finance at Warsaw School of Economics. She received her M.A. in 2003 and her Ph.D. in economics from Warsaw School of Economics in 2007. Her fields of specialization are: Behavioral Finance, Capital Markets, Socially Responsible Investing, and Cultural Finance.

Marek Kretowski is an associate professor in the Faculty of Computer Science, Bialystok University of Technology. He received his Ph.D. in 2002 from the University of Rennes I and Bialystok University of Technology. In 2009 he obtained his habilitation degree from the Institute of Computer Science, Polish Academy of Science. His current research interests are data mining, evolutionary computation and biomedical application of computer science.