Tree-based Cost Sensitive Methods for Fraud Detection in Imbalanced Data

Guillaume Metzler[1,2,⋆], Xavier Badiche[2], Brahim Belkasmi[2], Elisa Fromont[3], Amaury Habrard[1], and Marc Sebban[1]

[1] Univ. Lyon, UJM-Saint-Etienne, CNRS, Institut d'Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France
guillaume.metzer,amaury.habrard,marc.sebban@univ-st-etienne.fr
[2] Blitz Business Service, 265 Rue Denis Papin, Villefontaine 38090, France
gmetzler,xbadiche,bbelkasmi@blitzbs.com
[3] Univ. Rennes 1, IRISA/Inria, 35042 Rennes Cedex, France.
efromont@irisa.fr

Abstract. Bank fraud detection is a difficult classification problem where the number of frauds is much smaller than the number of genuine transactions. In this paper, we present cost sensitive tree-based learning strategies applied in this context of highly imbalanced data. We first propose a cost sensitive splitting criterion for decision trees that takes into account the cost of each transaction and we extend it with a decision rule for classification with tree ensembles. We then propose a new cost-sensitive loss for gradient boosting. Both methods have been shown to be particularly relevant in the context of imbalanced data. Experiments on a proprietary dataset of bank fraud detection in retail transactions show that our cost sensitive algorithms allow to increase the retailer's benefits by 1,43% compared to non cost-sensitive ones and that the gradient boosting approach outperforms all its competitors.

Keywords: Cost Sensitive Learning · Imbalance Learning · Binary Classification

1 Introduction and Related Work

Imbalanced data are ubiquitous in many real world applications, e.g. in medical domains [13], bank transactions [2,16] or industrial processes [1]. Supervised machine learning tasks are challenging in this context because algorithms struggle to focus on the important class (e.g. fraud, disease, failure, etc.) which is under-represented in the data. Classical approaches tend to tackle the problem by rebalancing the data [7] or optimizing different performance measures than the classical accuracy [15] which would otherwise lead to predict all instances in the over-represented classes. Ensemble methods such as random forests [5] or boosting algorithms [9,17] have been shown to be particularly successful in this context because they can combine local decisions taken in areas where the imbalance is (made) less prominent.

In some application domains such as fraud/anomaly detection, additional precise information can be given to favor one class from other ones in the learning process. This can for example be done by cost-sensitive learning approaches [8] which can take into account user preferences (in terms of the importance of the classes or of the attributes). For example, [2] proposes a cost-sensitive decision tree stacking algorithm to tackle a fraud detection task in a banking context. The authors provide a cost matrix that assigns costs to each decision made by the model and define an optimization function that takes this matrix into account. In Decision Tree learning the splitting criterion is made according to these costs method and not according to the usual impurity measures (such as entropy or Gini). This allows them to better target the rare classes. [17] presents a cost-sensitive version of the Adaboost boosting algorithm [9] and also shows its relevance in this imbalanced scenario. [18] gives a general study of the cost-sensitive learning methods in the context of imbalanced data. They categorise the methods into two sets: those which fix the error costs in each class and apply it *a posteriori* to make a decision, and those which tune *a priori* the cost matrix depending on the local distribution of the data (this category seems more successful). [16] tackles the problem of credit card fraud detection. The approach, similar to [2] and to the first one we present in this paper, proposes to induce decision trees by splitting each node according to a cost matrix associated to each example. However, as in [16], they focus on the actual money losses losses but not on possible benefits of better classifications and they apply their method to ranking problems. Other methods like [13] have focused on the cost of the attributes (here in the context of medical data). They consider that acquiring the exact value of a given attribute is costly and try to find a good compromise between the classification errors and the total feature acquisition costs.

In this paper, we also propose different cost sensitive tree-based learning approaches in the highly imbalanced context of bank fraud detection. The first approach, similar to [2] and [16] and presented in Section 3, uses a cost sensitive splitting criterion for decision trees that takes into account the costs (as well as the benefits) of each transaction. But it differs from [2] and [16] in the combinaison strategy for building an ensemble method. The second one presented in Section 4 is a new cost-sensitive proper loss [6] for gradient boosting. Section 2 presents our notations, the gain/cost matrix we are working with and the associated weighted miss-classification loss we want to optimize. The experiments and results are presented in Section 5. We illustrate the different methods using both the retailer margin and the F-Measure (F_1) as performance measures. The experiments are made on a proprietary dataset of the *Blitz* company. We finally conclude in Section 6.

2 Notations and Problem Formulation

2.1 Notations

We focus in this paper on binary supervised classification problems. Let $S = (\mathbf{X}, \mathbf{Y}) = ((\boldsymbol{x}_1, y_1), ...(\boldsymbol{x}_m, y_m))$ be a set of m training instances where $\boldsymbol{x}_i \in \mathbb{R}^d$

and $\boldsymbol{y_i} \in \{0,1\}^m$ are their corresponding labels. The notation x_i^j is used to denote the value of the j^{th} variable of the instance i. The label 0 will be used for the negative or majority class (i.e. the genuine transactions) and the label 1 will be used for the positive or rare class (i.e. the fraudulent transactions). We will further denote by S_+ the set of m_+ positive examples and S_- the set of m_- negative examples (here $m_- >> m_+$). We will also note $\hat{y}_i$ the label predicted by our learned model for the instance i. We use the notation p for the predicted probability that an example belongs to the minority class, F will be used to design the learned model, i.e. $p_i = F(x_i)$ is the probability that the transaction is fraudulent. A threshold is then used to get its label.

2.2 Problem Formulation

Our goal is to maximize the profits of the retailers by predicting, online, with decision trees [4] which transactions, made by a customer are genuine or not. While training the trees offline, the company might like to introduce some costs assigned to the training examples, according to the adequacy between the actual label of the transaction and the predicted one (see Table 1). For instance, the retailers will gain money by accepting a genuine transaction, i.e. $c_{TN_i} > 0$, where TN stands for *True Negative* or genuine transactions correctly classified. However, if the retailers accept a fraudulent one, they will loose the amount of the transaction $c_{FN_i} < 0$, where FN stands for *False Negative* or fraudulent transaction predicted as a genuine one.

Table 1. Cost Matrix associated to each example of the training set.

	Predicted Positive (fraud)	Predicted Negative (genuine)
Actual Positive (fraud)	c_{TP_i}	c_{FN_i}
Actual Negative (genuine)	c_{FP_i}	c_{TN_i}

In this paper, we use a similar approach as the one presented in [2]. However, instead of only minimizing the money loss due to an acceptation of a fraudulent transaction, we rather focus on maximizing the retailers profits, i.e. we aim at maximizing the loss function L defined as follows:

$$L(y \mid \hat{y}) = \sum_{i=1}^{m} \left[y_i(\hat{y}_i c_{TP_i} + (1-\hat{y}_i)c_{FN_i}) + (1-y_i)(\hat{y}_i c_{FP_i} + (1-\hat{y}_i)c_{TN_i}) \right]. \tag{1}$$

Talking about profits instead of classical "costs" is more meaningful for the retailers. Furthermore, if we simply focus on the error made by the algorithm, a correctly classified instance will have no influence on the learned model.

In the next section, we show how this loss function can be optimized while learning decision trees.

3 Cost Sensitive Decision Trees

A classic decision tree induction algorithm proceeds in a top-down recursive divide-and-conquer manner. At each step, the best (according to a given criterion) attribute A is chosen as a new test (internal node) in the tree and the set of examples is splitted according to the outcome of the attribute for each of the instance (there is one child node v per possible outcomes for a given attribute). Then, the same procedure is applied recursively to each new created subset of examples S_v until reaching a given stopping criterion. Classification trees (e.g. [4]) usually split the nodes according to an "impurity" measure. One such measure is the Gini index of a set of m instances $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ defined as follows: $Gini = 1 - \sum_{k=1}^{C} p_k^2$, where p_k denotes the probability to belong to the class k and C is the number of classes ($C = 2$ in our case). In this paper, the splitting criterion is based on the cost matrix defined above. We do not want to minimize an impurity but to *maximize the retailer profits* according to the cost matrix.

3.1 Splitting criterion and label assignment

Our splitting criterion $\mathit{\Gamma}_S$ on a given set of training instances S of size m (as defined in Section 2) is:

$$\mathit{\Gamma}_S = \sum_{i \in S_-} \left(\frac{m_+}{m} c_{FP_i}(\boldsymbol{x}_i) + \frac{m_-}{m} c_{TN_i}(\boldsymbol{x}_i) \right) + \sum_{i \in S_+} \left(\frac{m_+}{m} c_{TP_i}(\boldsymbol{x}_i) + \frac{m_-}{m} c_{FN_i}(\boldsymbol{x}_i) \right), \tag{2}$$

where the first term corresponds to the profits due to genuine transactions and the second to the fraudulent transactions.

Note that this quantity depends on the amount of the transaction of each example in S through the costs c. The best attribute A is the one which *maximizes* the quantity:

$$(\frac{1}{n+\varepsilon}) \sum_{v \in Children(A)} \mathit{\Gamma}_{S_v} - \mathit{\Gamma}_S.$$

Note that this quantity is very similar to the splitting criterion used to minimize to minimize the Gini impurity up to the number of examples in the parent node. We simply take the opposite of the classical gain and divide it by the number of instances in the parent node, so that this criterion becomes convex.

The values $\mathit{\Gamma}_{S_v}$ are computed using equation (2) on each set S_v. It differs from the splitting criterion used in [2] where the splits minimize the cost of wrongly accepting or blocking the transactions.

Once the induction tree stopping criterion is reached (ours is defined in Section 5), a class label is associated to each leaf of the tree. For the sake of clarity we introduce the following notations:

$\gamma_0(l)$: the average profit associated to the leaf l if all the instances are predicted as genuine:

$$\gamma_0(l) = \frac{1}{|l|}\left(\sum_{i:\boldsymbol{x}_i \in l \cap S_-} c_{TN_i} + \sum_{i:\boldsymbol{x}_i \in l \cap S_+} c_{FN_i}\right),$$

$\gamma_1(t)$: the average profit associated to the leaf l if all instances are predicted as frauds:

$$\gamma_1(l) = \frac{1}{|l|}\left(\sum_{i:\boldsymbol{x}_i \in l \cap S_-} c_{FP_i} + \sum_{i:\boldsymbol{x}_i \in l \cap S_-} c_{TP_i}\right),$$

where $|l|$ denotes the number of examples in the leaf l and $i : \boldsymbol{x}_i \in l \cap S$ denotes the index i of the example x_i both in leaf l and in the set S.

A leaf is assigned the label 1 if $\gamma_1 > \gamma_0$, i.e. all the transactions in a given leaf are predicted fraudulent if the associated average profit is greater than one associated when all instances are predicted genuine.

Note that this strategy can be easily extended to ensembles of trees [5]. In this case, a standard decision rule consists in applying a majority vote over the whole set of the T learned decision trees. However, this decision rule does not take into account the probability score that can be associated to each tree prediction using the class distribution of the examples in the leaf $l(\boldsymbol{x}_i)$ (as in [16]). Following this idea, we suggest here to label an instance as positive if the average $\bar{\gamma}_1(\boldsymbol{x})$ of the average profits $\gamma_1(l^j(\boldsymbol{x}_i))$ over the T trees is greater than $\bar{\gamma}_0(\boldsymbol{x})$, where $l^j(\boldsymbol{x}_i)$ is the leaf of the j^{th} tree containing $\boldsymbol{x}_i$:

$$\bar{\gamma}_1(\boldsymbol{x}) = \frac{1}{T}\sum_{j=1}^{T}\gamma_1(l(\boldsymbol{x})) \geq \frac{1}{T}\sum_{j=1}^{T}\gamma_0(l(\boldsymbol{x})) = \bar{\gamma}_0(\boldsymbol{x}).$$

4 Cost Sensitive Gradient Boosting

In this section, we briefly present the gradient boosting framework introduced in [11]. Then we present a proper cost-sensitive loss function in order to implement it in a gradient boosting algorithm in an efficient way.

4.1 Generalities about Gradient Boosting

Gradient boosting has been shown to be very efficient to deal with classification problems, and a very good candidate to address issues due to imbalance data [3,12]. Unlike the well known Adaboost algorithm [9], gradient boosting performs an optimization in the *function* space rather than in the *parameter* space. At each iteration, a weak learner f_t is learned using the *residuals* (or the errors) obtained

by the linear combination of the previous models. The linear combination F_t at time t is defined as follows:

$$F_t = F_{t-1} + \alpha_t f_t, \tag{3}$$

where F_{t-1} is the linear combination of the first $t-1$ models and α_t is the weight given to the t^{th} weak learner. The weak learners are trained on the residuals $r_i = y_i - F_{t-1}(\boldsymbol{x}_i)$ of the current model. These residuals are given by the negative gradient, $-g_t$, of the used loss function L with respect to the current prediction $F_{t-1}(\boldsymbol{x_i})$:

$$r_i = g_t = -\left[\frac{\partial L(y, F_{t-1}(\boldsymbol{x}_i))}{\partial F_{t-1}(\boldsymbol{x}_i)}\right].$$

Once the residuals r_i are computed, the following optimization problem is solved:

$$(f_t, \alpha_t) = \underset{\alpha, f}{argmin} \sum_{i=1}^{m} (r_i - \alpha f(\boldsymbol{x}_i))^2.$$

Finally, the update rule (3) is applied.

4.2 Cost sensitive loss for gradient boosting

In this section we aim to use the framework presented in [6] to give a proper formulation of our loss function of Eq (1) in the context of a boosting algorithm, using the gain matrix presented in Table 1.

Using a Bayes rule for classification [8], an instance i is predicted fraudulent if $\gamma_1 > \gamma_0$, i.e:

$$p_i c_{TP_i} + (1 - p_i) c_{FP_i} - p_i c_{FN_i} - (1 - p_i) c_{TN_i} > 0,$$

where p_i denotes the probability of the instance to be a genuine transaction. It gives us a threshold over which the transaction is declined (or predicted fraudulent):

$$p_i > \frac{c_{TN_i} - c_{FP_i}}{c_{TP_i} - c_{FN_i} + c_{TN_i} - c_{FP_i}} = s_i$$

Using the threshold s_i, our cost-weighted miss-classification loss can be rewritten as:

$$L(y \mid p) = -\frac{1}{m} \sum_{i=1}^{m} (y_i c_{TP_i} + (1 - y_i) c_{FP_i}) \mathbb{1}_{p_i > s_i} + (y_i c_{FN_i} + (1 - y_i) c_{TN_i}) \mathbb{1}_{p_i \leq s_i}. \tag{4}$$

Then, following the framework presented in [6], $L(y \mid p)$ can be rewritten as follows:

$$L(y \mid p) = \frac{1}{m} \sum_{i=1}^{m} \xi_i \left[y_i (1 - s_i) \mathbb{1}_{p_i \leq s_i} + (1 - y_i) s_i \mathbb{1}_{p_i > s_i} \right] - \frac{1}{m} \sum_{i=1}^{m} (y_i c_{TP_i} + (1 - y_i) c_{TN_i}), \tag{5}$$

where $\mathbb{1}$ is an indicator function and where we use the fact that $s = s\mathbb{1}_{p>s} + s\mathbb{1}_{p\leq s}$ and set $\xi_i = c_{TN_i} - c_{FP_i} + c_{TP_i} - c_{FN_i}$ which is positive in our context. In fact, $c_{TN} > c_{FP}$, we earn more if we correctly classify a genuine transaction. Furthermore, if we accept a fraudulent transaction then we loose money and we earn nothing if we declined it, i.e. $0 = c_{TP} > c_{FN}$.
The first part of equation (4.2), which we will note L_{s_i} corresponds to the cost-sensitive loss introduced in [6] with $s_i \in [0, 1]$. Each term of the sum is multiplied by a constant ξ_i which depends on the data. The second term represents the maximum that our loss can reach if the predictions were perfect. Note that this second term does not depend on p_i. Therefore, we want to minimize:

$$\underset{p\in[0,1]}{argmin}\ \mathbb{E}_y[L(y \mid p)] = \underset{p\in[0,1]}{argmin}\ \mathbb{E}_y\left[\frac{1}{m}\sum_{i=1}^{m}\xi_i L_{s_i}(y_i \mid p_i)\right].$$

However it has been shown that in the context of Boosting, it is more convenient to use an exponential approximation [11]. We adapt it to consider the output of a prediction model F directly in our approach as follows [4]: :

$$\ell_{s_i} = (1 - s_i)y_i e^{-F(\boldsymbol{x}_i)} + s_i(1 - y_i)e^{F(\boldsymbol{x}_i)}.$$

Solving $\dfrac{\partial \mathbb{E}_{\mathbf{Y}}[\ell_{s_i}]}{\partial F(x_i)} = 0$, we obtain the link function ψ_i between p_i and $\hat{F}_i = F(\boldsymbol{x}_i)$:

$$p_i = \psi_i(\hat{F}_i) = \frac{1}{1 + \dfrac{1 - s_i}{s_i}e^{-2\hat{F}_i}},$$

and its inverse ψ_i^{-1} is given by:

$$e^{\hat{F}_i} = \left(\frac{1 - s_i}{s_i}\right)^{1/2}\left(\frac{p_i}{1 - p_i}\right)^{1/2}. \tag{6}$$

The way to transform the output of a boosting model into a probability (the calibration process) plays a key role in the performance of the predictive algorithm. It has been shown that we can achieve at least the same performance with well calibrated boosting model than with one which is cost sensitive [14]. However, we think that our cost sensitive approach gives us a good transformation of the output of the model into a probability.

It is worth noticing that we can make use of equation (5) to provide a smooth approximation of the indicator function, such that:

$$\mathbb{1}_{p_i>s_i} \leq \left(\frac{1 - s_i}{s_i}\right)^{1/2}\left(\frac{p_i}{1 - p_i}\right)^{1/2} = e^{\hat{F}_i}.$$

[4] note that it exists a direct link between a predicted probability and the output of a model (see Section 3 of [10] and Section 4 of [6] for further details

Note that: $p_i > s_i \iff \psi_i(\hat{F}_i) > s_i \iff e^{\hat{F}_i} > 1 \iff \hat{F}_i > 0$. So it is enough to check the sign of the score to predict the label of each transaction. Finally we are minimizing an upper bound $\tilde{L}$ of L:

$$L(y \mid p) \leq \tilde{L}(y \mid F) = \frac{1}{m} \sum_{i=1}^{m} (1 - s_i) y_i e^{-\hat{F}_i} + s_i (1 - y_i) e^{\hat{F}_i}.$$

To use it in a gradient boosting algorithm, it remains to compute the first and second order derivative of $\tilde{L}$ for each instance i with respect to $\hat{F}_i$. They are given by:

$$\frac{\partial \tilde{L}}{\partial \hat{F}_i} = \xi_i \left[-(1 - s_i) y_i e^{-\hat{F}_i} + s_i (1 - y_i) e^{\hat{F}_i} \right],$$

and

$$\frac{\partial^2 \tilde{L}}{\partial \hat{F}_i^2} = \xi_i \left[(1 - s_i) y_i e^{-\hat{F}_i} + s_i (1 - y_i) e^{\hat{F}_i} \right].$$

5 Experiments

In this section, we evaluate the decision rule presented in Section 3 and the loss function presented in Section 4. We compare the results of our method on the retailer profits compared to using a classic Random Forest (RF) algorithm based on the Gini impurity criterion (baseline). The experiments are performed on a private dataset own by the *Blitz* company which can not be entirely described and made available for privacy reasons.

5.1 Dataset and experiments

The *Blitz* dataset consists of 10 months of bank transactions of a retailer. The first six months are used as the training set ($1,663,009$ transactions) and the four remaining ones as test set ($1,012,380$ transactions). The data are described by 17 features and are labeled as *fraud* or *genuine*. The Imbalance Ratio (IR) of the dataset is equal to 0.33%.

The first series of experiments compares the random forest baseline (**RF**) to the tree ensemble algorithm which uses the decision rule presented in Section 3 ($\mathbf{RF_X}$). We made different variants of the decision rule:

1. $\mathbf{RF_{maj}}$: each leaf is labeled according to the majority class of the examples that fall into the leaf, thus the output of each tree is in {0,1}. The voting criterion is detailed below.
2. $\mathbf{RF_{maj-mar}}$: each leaf is labeled to maximize the profit (also called margin) over the set of all examples in the leaf (the label is 0 if $\gamma_0 > \gamma_1$ and 1 otherwise). We then use a majority vote to predict the label of each transaction.
3. $\mathbf{RF_{mean-mar}}$: this model is the one described in Section 3

For each tree ensemble algorithm, we have used 24 trees with the same maximum depth. Furthermore, for the models $\mathbf{RF}, \mathbf{RF_{maj}}$ and $\mathbf{RF_{maj-mar}}$, the ensemble classifies a transaction as "fraud" if at least 9 trees agree on this positive class (this threshold is coherent with the one currently used in the *Blitz* company).

The second series of experiments is dedicated to the analysis of the gradient boosting approaches. We compare our approach $\mathbf{GB_{margin}}$, presented in Section 4.2, with three gradient boosting algorithms which aim to minimize the logistic loss:

1. $\mathbf{GB_{tune-Pre}}$: the threshold has been chosen so that we have the same precision on the validation set as the model **RF** in the training phase.
2. $\mathbf{GB_{tune-mar}}$: the threshold has been chosen to maximize the margin on the validation dataset.
3. $\mathbf{GB_{tune-F1}}$: the threshold has been chosen to maximize the F-Measure F_1 on the validation dataset.

For each of these three experiments we needed a validation data set to choose the optimal threshold over which a transaction is predicted as fraudulent. For this purpose, the training set is splitted in two sets, the first one is used to train the model and the other as a validation set, to find the best threshold for the given criterium we want to optimize. To do so, the first four months of transactions constitute the training set, the two remaining months are used as the validation set. Finally these three experiments have been conducted on the same *training/validation* set and the **R** software[5].

For privacy reasons, the explicit expressions of $c_{TP_i}, c_{FP_i}, c_{TN_i}, c_{FN_i}$ of the cost matrix can not be given. Note that they are simple functions of the amount M of the transaction. For example, we define c_{FP_i} as follows $c_{FP_i} = h(M) - \zeta$, where ζ is a parameter used to translate in financial terms, the dissatisfaction of the customer whose transaction has been declined.

5.2 Results

To measure the performance of each algorithm, we measure the gap between the maximum profits, i.e. the profits obtained if no errors are made, and the profits given by the algorithms. We use classic performance measures that are often used in an imbalanced setting such as the Precision, Recall and F_1-Measure defined as follows:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \quad F_1 = \frac{2\ Precision \times Recall}{Precision + Recall}.$$

All the experiments have been conducted with the same cost matrix where $\zeta = 5$. The results are presented in Table 2. We first notice that we get a reduction of the gap of profits of 1.43%, with the gradient boosting model $\mathbf{GB}_{margin}$ compared to the baseline **RF**. To give an idea to the reader, having a gap of 1%

[5] https://www.r-project.org/ and using the package **XGBoost**.

Table 2. Gap to the maximal margin of each algorithm. In this table, the value of ζ was set to 5. The results are separated into two groups: Random Forest models and Gradient Boosting models.

Experiments	Gap max profits	Precision	Recall	$\mathbf{F}_1$
RF	2.99%	68.1%	5.66%	10.5%
$\mathbf{RF_{maj}}$	2.88%	73.8%	4.71%	8.86%
$\mathbf{RF_{maj-mar}}$	1.81%	30.2%	10.6%	15.7%
$\mathbf{RF_{mean-margin}}$	1.87%	30.3%	9.52%	14.5%
$\mathbf{GB_{tune-Pre}}$	3.01%	**61.0%**	6.49%	11.7%
$\mathbf{GB_{tune-mar}}$	2.26%	19.1%	**16.6%**	**17.8%**
$\mathbf{GB_{tune-F1}}$	2.70%	45.4%	9.24%	15.4%
$\mathbf{GB_{margin}}$	**1.56%**	18.8%	13.3%	15.6%

to the maximum profits represents a loss of $43,000$ euros. So, by reducing the gap of 1.43% we increase the profits of the retailer by $60,000$ euros.

Regarding the Random Forest models, we note that the proposed approaches are able to improve the profits of the retailer compared to the model **RF**. However, we note that $\mathbf{RF_{maj}}$, which uses the number of examples and their label to predict the class of the examples in the leaf, gives similar performance as **RF** even if it is built differently. This means that the way to label the leaves has, at least, the same importance as the way to build the trees. The models $\mathbf{RF_{maj-mar}}$ and $\mathbf{RF_{mean-mar}}$ which directly use the notion of average profits in each leaf are the two models that give the best results, in terms of both profits, recall and F-Measure even if the precision is reduced. This is explained by the fact that refusing a genuine transaction will have small impact on the margin of the retailers while accepting a fraudulent transaction will represent a loss for the retailers that is close to the amount of the transaction. Using only our proposed method of Random Forest algorithm, we are able to reduce the gap of 1.18.

If we focus now on gradient boosting models, we first note that the model $\mathbf{GB_{tune-Pre}}$ is the one with the highest precision. On the other hand, the other models have a significantly smaller precision but exhibits a higher recall: by maximizing the margin they actually try to find the most fraudulent transactions. As mentioned previously, our cost-sensitive approach $\mathbf{GB_{margin}}$ is the one that achieves the best results in terms of margin. But it has also the worst precision (18.8%) for the reason given in the previous paragraph. This model provides also better results than $\mathbf{GB_{tune-mar}}$ which emphasizes the interest of a cost-sensitive approach compared to a simple classification model. However, we note that the model $\mathbf{GB_{tune-F1}}$ is not the one achieving the best F-Measure at test time. Let us also note that F_1 score remains low for each presented algorithms. We think that low values are observed because of the complexity of the data and the problem. Frauds are rare and spread in the all data set.

In a second part, we want to analyze the effect of the parameter ζ. Indeed, some retailers, for marketing reasons, do not want to refuse the transaction of good customers, i.e. they prefer to have a higher precision on their predictions.

A simple way to take this into account in our model is to artificially increase the value of ζ. Figure 1 shows the impact of the parameter ζ on the *Precision*, *Recall* and F_1, while the Gap Margin is still evaluated with $\zeta = 5$. Recall that some retailers do not want to refuse the payment of the good customers, however, the model which maximized is the one with the lowest precision (16.6%). So, it can be interesting to propose to the retailers several models using different values of ζ and give them the choice of the compromise between profits and precision. We first notice that the higher the value of ζ, the higher the precision and the smaller the recall. However, we see that it possible to reach a precision which is twice superior than the $\mathbf{GB_{margin}}$ one by setting $\zeta = 20$ and the gap will still be low with a value of 1.94%.

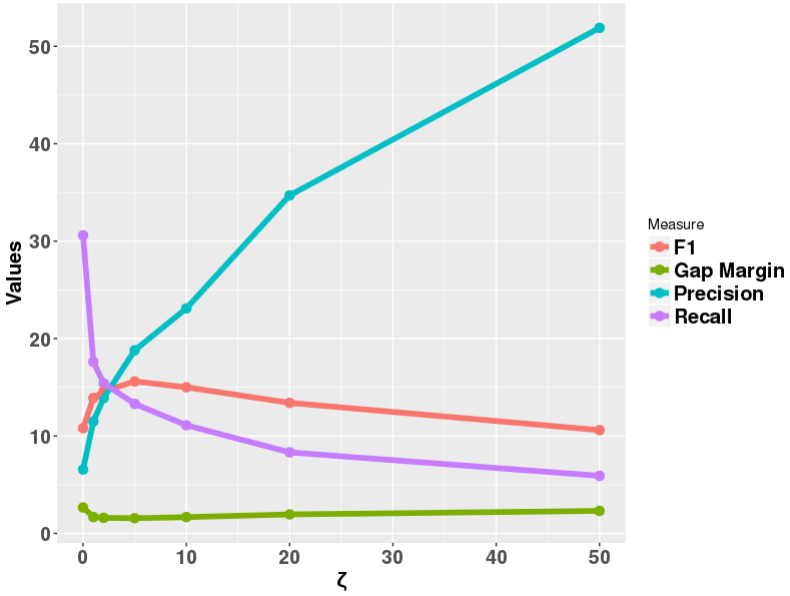

Fig. 1. Study of the influence of the parameter ζ in the definition of the gain of false positive C_{FP_i}. We illustrate the behaviour of the Precision, Recall and F_1 according to ζ. We also represent the gap to the maximum margin with respect to ζ, but we set $\zeta = 5$ to compute the gap.

6 Conclusion

We have presented different cost sensitive tree-based learning strategies to detect frauds in imbalanced retail transaction data. The first strategy is a tree ensemble algorithm which uses a new decision rule which tries to directly optimize the retailer profit. The second one is a gradient boosting algorithm which optimizes

a new cost-sensitive loss function. Experiments show that our cost sensitive algorithms allow to increase the retailer's benefits by 1,43% compared to non cost-sensitive ones and that the gradient boosting approach outperforms all its competitors. We plan to focus on how to combine different types of models that may capture different modalities of the data. Furthermore, due to our industrial context, we also want to work on the notion of *concept drift* and study how the distribution of frauds is evolving in order to take it into account in our models. This opens the door to the development of new domain adaptation methods.

References

1. Aggarwal, C.C.: Outlier Analysis. Springer (2013)
2. Bahnsen, A.C., Villegas, S., Aouada, D., Ottersten, B., Correa, A.M., Villegas, S.: Fraud detection by stacking cost-sensitive decision trees. DSCS (2017)
3. Beygelzimer, A., Hazan, E., Kale, S., Luo, H.: Online gradient boosting. In: Advances in Neural Information Processing Systems (NIPS), pp. 2458–2466 (2015)
4. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA (1984)
5. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (Oct 2001)
6. Buja, A., Stuetzle, W., Shen, Y.: Loss functions for binary class probability estimation and classification: Structure and applications, manuscript, available at www-stat.wharton.upenn.edu/ buja (2005)
7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. J. Artif. Int. Res. **16**(1), 321–357 (Jun 2002)
8. Elkan, C.: The foundations of cost-sensitive learning. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2. pp. 973–978. IJCAI'01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
9. Freund, Y., Schapire, R.E.: A short introduction to boosting. In: In Proceedings of the Sixteenth IJCAI. pp. 1401–1406. Morgan Kaufmann (1999)
10. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics **28**, 2000 (1998)
11. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29**, 1189–1232 (2000)
12. Li, P., Burges, C.J.C., Wu, Q.: Mcrank: Learning to rank using multiple classification and gradient boosting. In: Proceedings of the 20th International Conference on Neural Information Processing Systems. pp. 897–904. NIPS'07 (2007)
13. Ling, C., Sheng, V., Yang, Q.: Test strategies for cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering **18**(8), 1055–1067 (aug 2006)
14. Nikolaou, N., Edakunni, N., Kull, M., Flach, P., Brown, G.: Cost-sensitive boosting algorithms: Do we really need them? Machine Learning **104**(2), 359–384 (Sep 2016)
15. Puthiya Parambath, S., Usunier, N., Grandvalet, Y.: Optimizing f-measures by cost-sensitive classification. In: NIPS, pp. 2123–2131 (2014)
16. Sahin, Y., Bulkan, S., Duman, E.: A cost-sensitive decision tree approach for fraud detection. Expert Syst. Appl. **40**(15), 5916–5923 (Nov 2013)
17. Sun, Y., Kamel, M.S., Wong, A.K., Wang, Y.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition **40**(12), 3358 – 3378 (2007)
18. Thai-Nghe, N., Gantner, Z., Schmidt-Thieme, L.: Cost-sensitive learning methods for imbalanced data. In: The 2010 International Joint Conference on Neural Networks (IJCNN). IEEE (jul 2010)