

## The fraud loss for selecting the model complexity in fraud detection

Simon Boge Brant & Ingrid Hobæk Haff

To cite this article: Simon Boge Brant & Ingrid Hobæk Haff (2022): The fraud loss for selecting the model complexity in fraud detection, Journal of Applied Statistics, DOI: [10.1080/02664763.2022.2070137](https://doi.org/10.1080/02664763.2022.2070137)

To link to this article: <https://doi.org/10.1080/02664763.2022.2070137>



© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



[View supplementary material](#)



Published online: 25 May 2022.



[Submit your article to this journal](#)



Article views: 556



[View related articles](#)



[View Crossmark data](#)

# The fraud loss for selecting the model complexity in fraud detection

Simon Boge Brant and Ingrid Hobæk Haff

Department of mathematics, University of Oslo, Oslo, Norway

## ABSTRACT

Statistical fraud detection consists in making a system that automatically selects a subset of all cases (insurance claims, financial transactions, etc.) that are the most interesting for further investigation. The reason why such a system is needed is that the total number of cases typically is much higher than one realistically could investigate manually and that fraud tends to be quite rare. Further, the investigator is typically limited to controlling a restricted number  $k$  of cases, due to limited resources. The most efficient manner of allocating these resources is then to try selecting the  $k$  cases with the highest probability of being fraudulent. The prediction model used for this purpose must normally be regularised to avoid overfitting and consequently bad prediction performance. A loss function, denoted the fraud loss, is proposed for selecting the model complexity via a tuning parameter. A simulation study is performed to find the optimal settings for validation. Further, the performance of the proposed procedure is compared to the most relevant competing procedure, based on the area under the receiver operating characteristic curve (AUC), in a set of simulations, as well as on a credit card default dataset. Choosing the complexity of the model by the fraud loss resulted in either comparable or better results in terms of the fraud loss than choosing it according to the AUC.

## ARTICLE HISTORY


Received 4 October 2021  
Accepted 20 April 2022


## KEYWORDS

Fraud detection;  
hyperparameter; penalised  
regression; boosting;  
cross-validation

## 1. Introduction

Fraud detection has cropped up as a term in statistical research at least since the early 1990s. In the excellent review article by Bolton and Hand [1], the authors identify some of the characteristics that make statistical fraud detection a distinct field of the statistical literature, and not just a special case of binary classification, or some other well-understood problem class. The goal of statistical fraud detection, is to create a system that automatically selects a subset of all cases, (insurance claims, financial transactions, etc.), that are the most interesting cases for further investigation. This is necessary because there is typically a much higher number of claims than one realistically could investigate manually, and because fraud is typically quite rare. In simple terms, statistical fraud detection can

**CONTACT** Simon Boge Brant  [simonbb@math.uio.no](mailto:simonbb@math.uio.no)

 Supplemental data for this article can be accessed here. <https://doi.org/10.1080/02664763.2022.2070137>

© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group  
This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way.

be thought of as binary classification, potentially with highly imbalanced classes, depending on the type of fraud. By imbalanced classes we mean, in this case, that there are very few occurrences of one of the two possible outcomes. In other words, the vast majority of financial transactions or insurance claims are legitimate, and fraud is, relatively speaking, a rare occurrence.

In fraud detection applications, the investigator is often required to efficiently allocate limited resources. This amounts to selecting a restricted number of cases that are most likely to be fraudulent, or most worthy of investigation. In order to achieve this, a model should be fitted to recorded data of previously investigated cases, and then used to predict the probability of fraud on new cases. The set of cases to be investigated should subsequently be determined from the predicted probabilities. In this respect, we have a precise notion of what a good or bad model is for this purpose, namely one that lets us pick a certain number of cases, such that as many as possible of these are actual cases of fraud. Given the application, we term this notion *fraud loss*. However, we acknowledge that it has been studied in other contexts under different names – notably by Cléménçon and Vayatis [5], where they refer to the problem as *finding the best instances*, or *classification with a mass constraint*.

The problem of minimising fraud loss, or *finding the best instances* is equivalent to maximising a measure known as the *precision at  $k$*  in the field of information retrieval. This is discussed amongst others by Robertson and Zaragoza [18], Joachims [13], and Boyd *et al.* [2]. A related problem is that of local bipartite ranking, where the aim is to find the best pairwise ranking of a subset of the data. In the language of Cléménçon and Vayatis [5], the focus is then not only on *finding* the best instances, but to *rank the best instances*. The goal in this setting is not only to select the  $k$  most relevant instances, but also to rank them as well as possible. In the context of fraud, this corresponds to selecting  $k$  cases such that as many as possible are positive, and that if they are ordered according to their predicted probability of fraud, the greatest possible number of the selected positive cases are ranked higher than the selected negative cases.

There are several suggestions for how to estimate a model in order to solve these and related problems. Boyd *et al.* [2] propose an estimation criterion that should result in model that finds the best instances. This criterion involves minimising a hinge-type loss function over all pairs of observations with opposite outcome. The number of optimisation problems to solve in the estimation procedure is then twice the number of pairs. Rudin [19] proposes an estimation framework that concentrates on the top ranked cases, called the  *$P$ -norm push*. Her method is inspired by the *RankBoost* algorithm of Freund *et al.* [10] for minimising the ranking loss, extending it *RankBoost* to a more general class of loss functions. Eban *et al.* [7] propose estimation methods aiming to maximise different measures relevant to ranking, such as the area under the precision-recall curve and the recall at a fixed precision. They do this by approximating the false positive and the true positive rates in the objective function. Their aim is to construct the objective function in such a way that the derivatives have more or less the same complexity as those of the log-likelihood function of a logistic regression model. This makes the method a lot more scalable than those of Rudin [19] and Boyd *et al.* [2].

As opposed to the above mentioned papers, we will not focus on the estimation of the model parameters directly, but rather on choosing the complexity of the model, via a tuning parameter. More specifically, we will consider maximisation of the likelihood function of

the statistical model with regularisation, using penalised methods, or boosting. Different values of the regularisation parameters will then result in models of varying complexity. The model is to be used for a very specific purpose, namely to make predictions in order to select the  $k$  most likely cases of fraud among a new set of cases. Therefore, it seems reasonable to try to choose the regularisation parameters that are optimal for this particular application. In that context, we define a loss function, which, broadly speaking, is the number of non-fraudulent cases among the  $k$  selected.

An important question is how to estimate the out of sample value of the fraud loss function in order to select tuning parameters. There are a number of different out of sample validation techniques based on the training data. They all involve fitting models to different subsets of the data, and evaluating the error on the data points that are left out. As the application is somewhat special, standard settings and techniques may not be adequate. For instance, predicted class labels will depend on an empirical quantile of the predicted probabilities, which might require subsamples of a certain size to be stable. Therefore we will investigate what the best strategies for the out of sample validation are.

The paper is organised as follows. Section 2 defines the models that are used, whereas Section 3 describes the actual problem. Further, the approach for selecting the model complexity with fraud loss is presented in Section 4. In Section 5, the properties of the proposed method is evaluated in a simulation study, and the method is further tested on real data in Section 6. Finally, Section 7 provides some concluding remarks.

## 2. Models

In order to detect fraud, one needs to find a model that produces predicted probabilities of fraud, or at least predicted scores with the same ordering as the probabilities. In what follows,  $Y$  denotes the outcome and  $\mathbf{X}$  represents the  $p$ -dimensional random vector of covariates. Although it could be useful to describe  $Y$  as a categorical or an ordinal variable in some cases, we will concentrate on the binary case, where  $Y$  takes either the value 1 if the case is fraudulent or 0 if it is not.

In many cases, for instance if the covariates are noisy or high-dimensional, the predictive accuracy of the model can be improved by shrinking the predicted probabilities towards the common value  $\frac{1}{n} \sum_{i=1}^n y_i$ , which is the estimate of the marginal probability  $\Pr(Y = 1)$ . Our aim is to select the model complexity in such a way that we are able to choose the  $k$  most likely cases of fraud, for some specific  $k$ , based on the predictions from the resulting model, as we will describe in Section 3.

We will in all cases consider models where

$$\log \left( \frac{\Pr(Y = 1 | \mathbf{X} = \mathbf{x})}{\Pr(Y = 0 | \mathbf{X} = \mathbf{x})} \right) = \eta(\mathbf{x}).$$

The model  $\eta(\mathbf{x})$  could be a linear function of the covariates, or an additive model where each component is a regression tree. This specification implies that the conditional probabilities of an event will take the form

$$p(\mathbf{x}_i) = \Pr(Y_i = 1 | \mathbf{X}_i = \mathbf{x}_i) = \frac{\exp(\eta_i(\mathbf{x}_i))}{1 + \exp(\eta_i(\mathbf{x}_i))}. \quad (1)$$

The fraud indicators  $Y_i$  are assumed to be conditionally independent and Bernoulli-distributed with probabilities  $p(\mathbf{x}_i)$ , given  $\mathbf{X}_i = \mathbf{x}_i$ . That leads to a binary regression model with a logit link function with the log-likelihood function

$$ll = \sum_{i=1}^n \log(p_i^{y_i} (1 - p_i)^{1-y_i}) = \sum_{i=1}^n (y_i \eta_i - \log(1 - p_i)),$$

where  $p_i$  is a shorthand for  $p(\mathbf{x}_i)$ .

As mentioned earlier, we wish to do shrinkage on the model. In the case of a parametric linear model  $\eta(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^t \mathbf{x}$ , this would correspond to shrinking the regression coefficients ( $\hat{\beta}_0, \hat{\boldsymbol{\beta}}$ ) towards zero, and for a tree model to reducing the number of trees, the depth of each tree, and possibly also the weights assigned to the leaf nodes. One option in the case of the linear model is to only look for solutions where  $(\hat{\beta}_0, \hat{\boldsymbol{\beta}})$  is within a certain euclidian distance of the origin. This regularised estimator is the maximiser of the penalised log-likelihood function  $ll(\beta_0, \boldsymbol{\beta}) - \lambda \sum_{j=1}^p \hat{\beta}_j^2$  and is the solution to

$$\operatorname{argmax}_{\hat{\beta}_0, \hat{\boldsymbol{\beta}}} \sum_{i=1}^n y_i \hat{\eta}_i - \log(1 - \hat{p}_i) - \lambda \sum_{j=1}^p \hat{\beta}_j^2, \quad (2)$$

This is often referred to as ridge regression [12], or  $L_2$ -penalised logistic regression, since it assigns a penalty to the squared  $L_2$ -norm of the parameters. Similar estimators such as the lasso [20] or the elastic net [21] can be constructed by considering different norms, i.e.  $L_1$  norm and a convex combination of  $L_1$  and squared  $L_2$  norm, respectively.

We will also consider additive models of the form

$$\eta(\mathbf{x}) = f_0 + \sum_{j=1}^M f_j(\mathbf{x}), \quad (3)$$

where  $f_0$  is a constant function and each  $f_j(\mathbf{x}) = \sum_{t=1}^T c_t \mathcal{I}(\mathbf{x} \in R_t)$  is a regression tree, that for a partition  $\{R_t\}_{t=1}^T$  of  $\mathbb{R}^p$  takes a constant value  $c_t$  for all  $\mathbf{x}$  in each  $R_t$ . These models are typically fitted by gradient boosting [11], an iterative procedure where one starts with a constant, then adds one component at a time by maximising a local Taylor approximation of the likelihood around the current model. Usually, this update is scaled down by a factor, i.e. multiplied by some number  $\nu \in (0, 1)$ , in order to avoid stepping too far and move past an optimum of the likelihood function. Several highly efficient implementations to fit such models exist, such as LightGBM [15], CatBoost [6] and XGBoost [3]. The flexibility that such models offer, especially if the individual trees are allowed to be complex, will easily lead to models that capture too much of the variability in the training data. In order to avoid this, and get a stable model, we will control the total number  $M$  of components in the model, and constrain the complexity  $T$  of each tree.

### 3. Problem description

An informal description of the fraud detection problem was given in the introduction. Here, the problem will be defined and explained in more detail. Formally, we can describe

our version of a fraud detection problem as follows. We divide the total dataset into a training set of size  $n_{tr}$  and a test set of size  $n_{te}$ . Further, the  $(p + 1)$  dimensional random vectors  $(Y_i, \mathbf{X}_i)$ ,  $i = 1, \dots, n_{tr} + n_{te}$  are assumed to be iid, and the main interest is the conditional distribution of  $Y_i$  given  $\mathbf{X}_i$ . In what follows,  $n$  will denote for the size of a sample, regardless of whether the sample in question is the test set or the training set, unless it is unclear from the notation which one is referred to.

Since the goal of the investigation is to uncover fraud, there should be as many actual cases of fraud as possible among the ones selected for investigation. This amounts to producing  $n$  predictions  $\{\widehat{Y}_i\}_{i=1}^n$ , such that  $k$  of these have the value 1. Therefore, the minimiser of the loss function is a model that minimises

$$\mathcal{L}^{fraud} = \sum_{i=1}^n (1 - Y_i) \widehat{Y}_i, \text{ s.t. } \sum_{i=1}^n \widehat{Y}_i = k.$$

This is equivalent to minimising the classification error

$$\mathcal{L}^{class} = \sum_{i=1}^n |Y_i - \widehat{Y}_i| = \sum_{i=1}^n (1 - Y_i) \widehat{Y}_i + \sum_{i=1}^n Y_i (1 - \widehat{Y}_i),$$

under the same constraint. Since  $\sum_{i=1}^n \widehat{Y}_i = k$ , we have

$$\mathcal{L}^{fraud} = k - \sum_{i=1}^n Y_i \widehat{Y}_i \quad \text{and} \quad \mathcal{L}^{class} = k + \sum_{i=1}^n Y_i - 2 \sum_{i=1}^n Y_i \widehat{Y}_i,$$

so the two must have the same minimiser.

The idea is to minimise the expected value of  $\mathcal{L}_{te}^{fraud}$ , which is the fraud loss for the test set. It would therefore be illuminating to know what the minimiser is, i.e. what it is that one is attempting to estimate. Let  $P_i = Pr(Y_i = 1 | \mathbf{X}_i)$ . then we can write

$$\begin{aligned} \mathbb{E} \left( \mathcal{L}_{te}^{fraud} \right) &= \mathbb{E} \left( \sum_{i=1}^n (1 - Y_i) \widehat{Y}_i \right) = \mathbb{E} \left( \mathbb{E} \left( \sum_{i=1}^n (1 - Y_i) \widehat{Y}_i \mid \mathbf{X}_{te} \right) \right) \\ &= \mathbb{E} \left( \sum_{i=1}^n (1 - P_i) \widehat{Y}_i \right), \end{aligned}$$

The minimiser of the above expectation over all vectors  $\widehat{\mathbf{Y}}$ , having all elements equal to zero, except exactly  $k$  elements that are equal to one, is the vector  $\widehat{\mathbf{Y}}^*$  satisfying

$$\widehat{Y}_i^* = \mathcal{I}(P_i \geq P_{(n-k+1)}),$$

where  $P_{(1)} \leq \dots \leq P_{(n)}$  are the conditional fraud probabilities  $P_i$  for the test set, sorted in ascending order. This is an indicator of whether  $P_i$  is among the  $k$  largest in the test sample. Thus, a quite natural approach is to fit a model for the regression function  $p(\mathbf{x}) = Pr(Y =$

$1|\mathbf{X} = \mathbf{x})$ , resulting in the predicted probabilities  $\widehat{p}_i$ , and then use the prediction

$$\widehat{Y}_i = \mathcal{I}(\widehat{p}_i \geq \widehat{p}_{(n-k+1)}). \quad (4)$$

#### 4. Selecting the model complexity

We want to find the model that minimises the fraud loss function. This is done by fitting a sequence of models, either by solving (2) for a sequence of values of the penalty parameter  $\lambda$ , or by fitting the additive tree model (3) with via gradient boosting, where each model has a different number  $M$  of components. Choosing the best of these corresponds to selecting a value of  $\lambda$  in the former case, and  $M$  in the latter case. However, the main interest is not the best possible fit to the training data, but rather the best possible performance on a new dataset. This may be determined by this estimating the relative out of sample performance of each model via repeated cross validation, or bootstrap validation. As mentioned in the introduction, we will study how different validation schemes perform. Both bootstrap validation and cross-validation involve splitting the data in different subsets, and repeatedly using some of the data for fitting, and the rest to evaluate the fitted models. Since the model is evaluated on datasets whose size  $n_{eval}$  is not in general the same as the size of the test set,  $n_{te}$ , we let the number of selected observations be  $k = \lceil n_{eval}\tau \rceil$ , i.e. the nearest integer to  $n_{eval}\tau$ , where  $\tau = \frac{k}{n_{te}} \in (0, 1)$  is the proportion of the cases in the test set we want to select.

We evaluate the classification error using cross-validation with  $L$  folds and  $D$  repetitions. The data are split into different folds by randomly assigning observations to each fold, thus creating  $D$  sequences of  $L$  non-overlapping subsets of the integers  $\{1, 2, \dots, n\}$ , which we denote as  $\{A_l^{(d)}\}_{l=1}^L$ ,  $d = 1, \dots, D$ . The cross-validation a statistic is given by

$$\widehat{\mathcal{L}}_{CV}^{fraud}(\tau) = \frac{1}{D} \sum_{d=1}^D \frac{1}{L} \sum_{l=1}^L \frac{1}{|A_l^{(d)}|} \frac{\sum_{i \in A_l^{(d)}} (1 - y_i) \widehat{y}_i^{l,d}(\tau)}{\sum_{i \in A_l^{(d)}} \widehat{y}_i^{l,d}(\tau)},$$

with  $\widehat{y}_i^{l,d}(\tau) = \mathcal{I}(\widehat{p}_i^{l,d} \geq \widehat{p}_{(n_{eval} - \lceil n_{eval}\tau \rceil + 1)}^{l,d})$ ,

as described in expression (4) of Section 3.

An alternative to cross-validation is bootstrap validation, where one for each fold draws observations from the training set with replacement, usually as many as the number of training observations. The left-out observations from each fold are then used for validation. The probability that a specific observation is left out of a bootstrap fold when the bootstrap folds are of the same size as the training set is equal to  $(1 - \frac{1}{n})^n$ , which when  $n$  increases converges to  $e^{-1} \approx 0.368$ . This means that the validation sets on average will contain a little over a third of the total training data.

In a standard binary classification setting, one can compute a statistic that mimics a leave one out cross-validation error as

$$\widehat{\text{Err}}^{(1)}(\tau) = \frac{1}{n} \sum_{i=1}^n \frac{\sum_{b=1}^B I_i^b L(y_i, \widehat{y}_i^{*b}(\tau))}{\sum_{b=1}^B I_i^b} \quad \text{or} \quad \widehat{\text{Err}}^{(1)}(\tau) = \frac{\sum_{i=1}^n \sum_{b=1}^B I_i^b L(y_i, \widehat{y}_i^{*b}(\tau))}{\sum_{i=1}^n \sum_{b=1}^B I_i^b},$$

where  $I_i^b$  is an indicator of whether the  $i$ th observation is *not* included in the  $b$ th bootstrap sample, and  $\hat{y}_i^{*b}$  is the prediction obtained for observation  $i$  from the  $b$ th model fit, the former expression stemming from Efron and Tibshirani [9] and the latter from Efron [8]. According to Efron and Tibshirani [9] these will be close for larger values of  $B$ . The bootstrap statistic we will use is based on the latter, and is given by

$$\widehat{\mathcal{L}}_{BOOT}^{fraud}(\tau) = \sum_{i=1}^n \frac{\sum_{b=1}^B I_i^b (1 - y_i) \hat{y}_i^{*b}(\tau)}{\sum_{b=1}^B \sum_{i=1}^n I_i^b \hat{y}_i^{*b}(\tau)}.$$

## 5. Simulation study

In order to study how well selecting the model complexity via a tuning parameter based on the fraud loss works, we will perform a simulation study. First, we examine and compare different setups of the selection approach. Subsequently, we make a comparison to the most relevant alternative approach, based on the AUC.

### 5.1. Covariates

We want the synthetic datasets to possess many of the same characteristics as real datasets from fraud detection applications. The common traits that we want to replicate, at least to some degree, are correlated covariates, with margins of different types, some continuous, some discrete, and an imbalance in the marginal distribution of the outcome.

In order to draw the covariates, we use the following procedure. First, we draw a sample of a random vector, from a multivariate distribution with uniform margins, i.e. a copula [17]. After simulating observations from the copula, each of the margins are transformed to one of the distributions listed in Table 1. The copula we will use is the  $t_{2,R}$ -copula. Data are then simulated by drawing from a  $p$ -variate  $t_{2,R}$ -distribution, i.e. a standard  $t$ -distribution with a  $p \times p$  correlation matrix  $\mathbf{R}$  and 2 degrees of freedom. Then, the margins are transformed via the (univariate)  $t_2$ -quantile function, which makes the margins uniform, but still dependent.

We specify the correlation matrix  $\mathbf{R}$  of the multivariate  $t$ -distribution by drawing a matrix from a uniform distribution over all positive definite correlation matrices, using the algorithm described by Joe [14]. When looking at comparisons across a number of different datasets, we always keep the correlation matrix  $\mathbf{R}$  fixed. Setting the correlation matrix by simulating it via an algorithm is just a pragmatic way of specifying a large correlation matrix, while ensuring that it is positive semidefinite. As for the correlation matrix,

**Table 1.** List of the 17 different marginal distributions used to simulate covariates.

Family	Parameters	Values
Bernoulli	$p$	0.2, 0.4, 0.6, 0.8
Beta	$(\alpha, \beta)$	(1, 2), (2, 1), (2, 2)
Gamma	$(\alpha, \beta)$	(1, 3), (3, 1), (3, 3)
Normal	$(\mu, \sigma)$	(0, 1)
Student's t	$\nu$	3, 4, 6
Poisson	$\lambda$	1, 3, 5



the distributions for the margins are also drawn randomly from a list, and these are also kept fixed whenever we consider comparisons across different datasets.

## 5.2. Probabilities

Given the simulated covariates  $\mathbf{x}$ , probabilities  $p(\mathbf{x})$  are computed, and the binary outcomes  $y_i$  are simulated from Bernoulli( $p(\mathbf{x}_i)$ ) distributions, where  $i = 1, \dots, n$ . The probabilities follow the form (1) with  $\eta(\mathbf{x}_i) = \beta_0 + f(\mathbf{x})$ , where the model,  $f(\mathbf{x})$ , is either the linear  $f(\mathbf{x}) = \boldsymbol{\beta}^t \mathbf{x}$ , or an additive model of the form (3), where the components are trees. Unless otherwise stated, we simulate datasets with  $p = 100$  covariates. In the model with a linear predictor, we let 15 of the 100 covariate effects be non-zero, and let these take values in the range  $(-0.77, 0.62)$ , with an average absolute value of 0.34.

In order to specify a tree model, we draw a covariate matrix  $\tilde{\mathbf{X}}$ , and a response to be used only for constructing the trees. The response follows the model  $BE_1 - (1 - B)E_2$ , where  $B$  is a Bernoulli(0.5) variable, and  $E_1, E_2$  are exponentially distributed with parameters  $\lambda_1 = 0.2$  and  $\lambda_2 = 0.1$ , respectively. We then fit an additive tree model to this, based on 15 of the 100 covariates. The resulting model is used to generate datasets, keeping the model fixed across the different datasets.

In both models, the parameter  $\beta_0$  can change from dataset to dataset, as the average probability  $p_0$ , which is an estimate of the marginal probability of fraud  $\Pr(Y = 1)$ , should be kept fixed. To achieve this, we solve the equation

$$\frac{1}{n} \sum_{i=1}^n p(\mathbf{x}_i) = p_0$$

numerically, given the model  $f(\mathbf{x})$  and the covariate values  $\mathbf{x}_i$ ,  $i = 1, \dots, n$ .

## 5.3. Data drawn from a logistic regression model

We first consider penalised logistic regression. We draw two datasets from a logistic regression model, one for estimation, and one for testing, by the method described previously, both with  $n = 1000$  observations. The coefficient  $\beta_0$  is first set so that  $p_0 = 0.2$ , which is in the higher range compared to a typical fraud detection setting. In order to select the value of the regularisation parameter  $\lambda$ , we compare bootstrap validation, and repeated cross-validation with 10, 5, 3, or 2 folds. We also try drawing the folds stratified on the outcome, so that each fold has the same proportion of positive outcomes as the training sample. All the experiments are repeated for  $S = 100$  simulated datasets.

In order to make the comparison fair, we balance the number of bootstrap folds and the number of repetitions for the cross-validation, so that the computational complexity is roughly the same, assuming that the computational complexity of fitting a model is proportional to the number of observations. All validation procedures are adjusted so that they are comparable to 10-fold cross-validation without repetition. Since this involves fitting models to 10 different datasets of .9 times the size of the total, we use 9 folds in the bootstrap validation, and 2, 4, and 9 repetitions of 5-, 3-, and 2-fold cross-validation, respectively. We also do the same, with double the number of repetitions across 10-, 5-, 3- and 2- fold cross-validation, and with 18 bootstrap folds.

The relative average fraud loss (RAFL), compared to the minimum over the alternatives for  $\lambda$  used in each simulation  $s = 1, 2, \dots, S$ ,

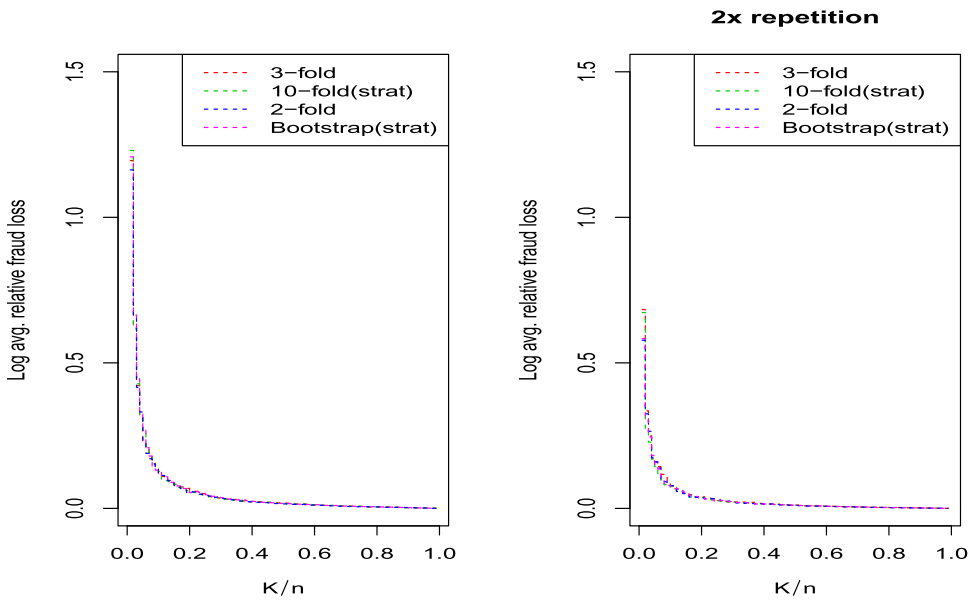
$$RAFL(\tau) = \frac{\sum_{s=1}^S \sum_{i=1}^n \widehat{y}_{i,s}^{\lambda_{CV}} (1 - y_{i,s}) / \sum_{i=1}^n \widehat{y}_{i,s}^{\lambda_{CV}}}{\sum_{s=1}^S \sum_{i=1}^n \widehat{y}_{i,s}^{\lambda_*} (1 - y_{i,s}) / \sum_{i=1}^n \widehat{y}_{i,s}^{\lambda_*}},$$

is computed for each simulation for  $\tau = 0.01, 0.02, \dots, 0.98, 0.99$ , and reported in Table 2, where  $\widehat{y}_{i,s} = \widehat{y}_{i,s}(\tau)$ , as explained in Section 4. As expected, doubling the number of repetitions leads to better performance. Looking at the different validation schemes, it seems like there is a tendency that cross-validation with 2 or 3 folds is the better option, and that there is a slight advantage to stratification, but this does not seem very conclusive. Based only on these results, we would therefore suggest using 2-fold cross-validation, and to stratify on  $y$ , at least if there are so few observations where  $y = 1$ , that there is a risk of getting folds where all observations have a negative outcome.

In Figure 1, the logarithm of the average relative fraud loss is plotted as a function of the proportion of cases that are selected, for a selection of the validation procedures. From

**Table 2.**  $RAFL(\tau)$  averaged over the different values of  $\tau$ , data simulated from linear model, model fitted using a linear model. 1000 observations with 100 covariates, both in training and test sets.

Notes	10-fold cv	5-fold cv	3-fold cv	2-fold cv	Bootstrap
	1.0764	1.0766	1.0753	1.0725	1.0763
stratified	1.0745	1.0767	1.0730	1.0743	1.0768
2x repeat	1.0385	1.0366	1.0364	1.0366	1.0389
2x repeat, stratified	1.0396	1.0384	1.0355	1.0354	1.0366



**Figure 1.** Plot of  $\log(RAFL(\tau))$  against the fraction  $\tau = k/n$  of the observations that are selected, for a selection of the methods for setting the tuning parameter. Logistic regression models were used both to simulate the data, and to make predictions.

**Table 3.**  $RAFL(\tau)$  averaged over the different values of  $\tau$ , data simulated from linear model, fitted additive tree model. 1000 observations with 100 covariates, both in training and test sets.

Notes	10-fold cv	5-fold cv	3-fold cv	2-fold cv	Bootstrap
stratified	1.2060	1.1779	1.1617	1.1424	1.1578
2x repeat	1.2123	1.1716	1.1534	1.1403	1.1496
2x repeat, stratified	1.0586	1.0507	1.0526	1.0541	1.0464
	1.0552	1.0523	1.0499	1.0515	1.0472

these, we see that there is not a very large difference between the different types. Further, it is hardest to select the top  $k$  cases for lower values of  $\tau$ , and thus of  $k$ , as expected.

We repeat this experiment for the same datasets, but instead of estimating the probabilities using a penalised logistic regression model, we use an additive tree model fitted by boosting. The penalty parameter we are trying to select is now the total number of components of the additive model, i.e.  $M$  in (3). Corresponding results to those in Table 2 and Figure 1 are shown in Table 3 and in Figure 1 in the supplementary material, respectively. Compared to the ridge regression fits, the relative average fraud loss is now greater, meaning that the selected model differs more in size, compared to the minimum for each value of  $\tau$ . The best alternative now seems to be the bootstrap variants, with stratified 3-fold cross-validation being the closest contender.

#### 5.4. Data drawn from an additive tree model

Next, we do a similar experiment, but drawing the data from a model where the linear predictor is replaced with an additive tree model, as previously outlined. We first estimate models using penalised logistic regression. The results of this are summarised in Table 2 in the supplementary material, and are very similar to previous results.

We also estimate additive tree models, the results for which are summarised in Table 4. In addition, we here also fit models to data simulated from the same type of model, but where  $p_0 = 0.05$ , i.e. more severely imbalanced, which is not uncommon in fraud detection. The results of the latter are summarised in Table 5. In the first case, 2-fold cross-validation gave the best results, with stratified 3-fold cross-validation in second place. Curiously, the error does not seem to be smaller when doubling the number of repetitions, which could suggest that it is easier to select the best model for the data simulated from a more complex model. For the latter case, 2-fold cross validation seems to be the best option, based on the aggregate relative average fraud both when the marginal probability of  $Y = 1$  is  $p_0 = 0.2$  and when  $p_0 = 0.05$ . It could be argued that real data are most likely not similar to a logistic regression model with a *linear* predictor, but rather more complex, and therefore that repeated 2-fold cross-validation seems like the more reliable option, overall.

**Table 4.**  $RAFL(\tau)$  averaged over the different values of  $\tau$ , data simulated from additive tree model, fitted additive tree model. 1000 observations with 100 covariates, both in training and test sets.

Notes	10-fold cv	5-fold cv	3-fold cv	2-fold cv	Bootstrap
stratified	1.0463	1.0450	1.0452	1.0454	1.0450
2x repeat	1.0469	1.0474	1.0457	1.0451	1.0454
2x repeat, stratified	1.0467	1.0464	1.0456	1.0435	1.0451
	1.0473	1.0453	1.0464	1.0450	1.0452

**Table 5.**  $RAFL(\tau)$  averaged over the different values of  $\tau$ , data simulated from additive tree model, fitted additive tree model, with  $p_0 = 0.05$ , fitted additive tree model. 1000 observations with 100 covariates, both in training and test sets.

Notes	10-fold cv	5-fold cv	3-fold cv	2-fold cv	Bootstrap
stratified	1.0194	1.0197	1.0195	1.0194	1.0194
2x repeat	1.0193	1.0196	1.0195	1.0189	1.0194
2x repeat, stratified	1.0198	1.0196	1.0193	1.019	1.0197
	1.0195	1.0196	1.0196	1.0191	1.0193

### 5.5. Comparisons with an alternative approach

Next, we want to compare our method to the estimator obtained when setting the penalty parameter using the AUC (area under the receiver operating characteristic curve) as a criterion, here estimated by the Wilcoxon type statistic

$$\widehat{AUC} = \frac{\sum_{i=1}^n \sum_{j=1}^n (1 - y_i) y_j \mathcal{I}(\hat{p}(\mathbf{x}_j) > \hat{p}(\mathbf{x}_i))}{\sum_{i=1}^n y_i \sum_{i=1}^n (1 - y_i)}.$$

The AUC is a popular measure for judging how good a binary regression model is at discrimination. Further, it is related to ranking, and is therefore a natural alternative to fraud loss. In fact, the AUC can on a population level be seen to be equivalent to the probability that an observation with  $Y = 0$  will be given a lower probability than one with  $Y = 1$ . Hence, if one model has a higher AUC than another, then the aforementioned probability will be highest for the model with the highest AUC [4]. Symbolically, this can be written as

$$AUC(\hat{p}) = P(\hat{p}(x_i) \geq \hat{p}(x_j) | Y_i = 1, Y_j = 0).$$

Likelihood, or likelihood-based measures such as Akaike’s information criterion (AIC) are also commonly used to set tuning parameters, but we will here disregard these. The reason for this is both that we do not see them as particularly relevant for the problem, as we are not interested in finding the model that best fits all the data, and also that the log-likelihood often explodes numerically when some of the probabilities become very close to 1. In our experiments, this happened often when we evaluated the likelihood for the data that were not used for estimation in a cross-validation procedure.

We start with a simulation study using the same simulation models as previously discussed. The  $RAFL(\tau)$ , averaged over values of  $\tau$  from 0.01 to 0.99, for models where the penalty parameter is chosen using repeated 2-fold cross-validation of the AUC and the fraud loss, respectively, are reported in Table 6 for  $p_0 = 0.2$  and in Table 7 for  $p_0 = 0.05$ . In addition, we report an aggregate over a smaller range of values of  $\tau = 0.16, 0.17, \dots, 0.25$  for  $p_0 = 0.2$  and  $\tau = 0.03, 0.04, \dots, 0.10$  for  $p_0 = 0.05$ , which are the most interesting values in the fraud setting. When looking at the aggregate over  $\tau$  from 0.01 to 0.99, it seems that it is advantageous to use the fraud loss when estimating an additive tree model, whatever the data generating model is. This is also the case when considering an aggregate over the smaller ranges of  $\tau$ -values, and in addition, the fraud loss works slightly better when fitting penalised logistic regression models to data simulated from an additive tree model. In Figure 2 and 3, the average difference in fraud loss for the two estimators is plotted as a function of  $\tau$ . It seems that in neither case, one or the other estimator is consistently better across the entire grid  $\tau = 0.01, 0.02, \dots, 0.98, 0.99$ .

**Table 6.**  $RAFL(\tau)$  averaged over the different values of  $\tau$ , comparison of AUC and fraud loss. 1000 observations with 100 covariates, both in training and test sets where  $p_0 = 0.2$ .

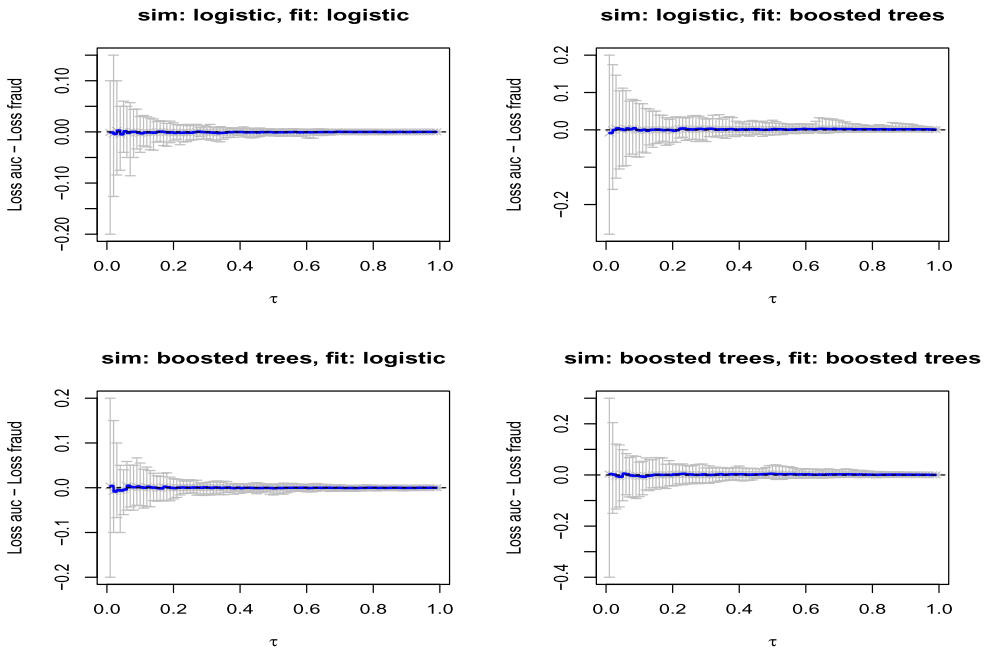
Simulation model	Logistic	Logistic	Additive trees	Additive trees
Estimation model	Logistic	Additive trees	Logistic	Additive trees
Average over all $\tau = 0.01, 0.01, \dots 0.99$ :				
auc	1.0335	1.0763	1.0186	1.0482
fraud	1.0371	1.0741	1.0195	1.0442
auc, 2x repeat	1.0337	1.0730	1.0183	1.0470
fraud, 2x repeat	1.0350	1.0721	1.0186	1.0454
Average over $\tau = 0.16, 0.17, \dots 0.24, 0.25$ :				
auc	1.0349	1.0626	1.0228	1.0583
fraud	1.0356	1.0591	1.0232	1.0531
auc, 2x repeat	1.0338	1.0610	1.0228	1.0570
fraud, 2x repeat	1.0359	1.0601	1.0225	1.0545

**Table 7.**  $RAFL(\tau)$  averaged over the different values of  $\tau$ , comparison of AUC and fraud loss. 1000 observations with 100 covariates, both in training and test sets where  $p_0 = 0.05$ .

Simulation model	Logistic	Logistic	Additive trees	Additive trees
Estimation model	Logistic	Additive trees	Logistic	Additive trees
Average over all $\tau = 0.01, 0.2, \dots 0.99$ :				
auc	1.0108	1.0185	1.0135	1.022
fraud	1.0137	1.0181	1.0131	1.0194
auc 2x repeat	1.0108	1.0185	1.0128	1.022
fraud 2x repeat	1.0121	1.0175	1.0126	1.0198
Average over $\tau = 0.03, 0.04, \dots 0.09, 0.10$ :				
auc	1.0549	1.089	1.0427	1.0664
fraud	1.0638	1.0879	1.0413	1.0617
auc 2x repeat	1.0551	1.0891	1.0411	1.0655
fraud 2x repeat	1.0593	1.0855	1.0397	1.0635

We repeat the experiment, but now we expand the datasets so that the number of covariates is  $p = n = 1000$ . These are simulated in the same way as for  $p = 100$ . We again simulate data both from a logistic regression model, and from an additive tree model, and scale the number of covariates the response depends on with the dimension of the covariate matrix, so that it, in both cases, it depends on 150 covariates. For the logistic regression model, the 150 non-zero effects take values in the interval  $(-0.67, 0.85)$ , and the average absolute value of these is 0.198. The results for all four combinations of data-generating model and estimated model, which are shown in Tables 2 and 3 and Figures 2 and 3 in the supplementary material, are similar to the ones for  $p = 100$ .

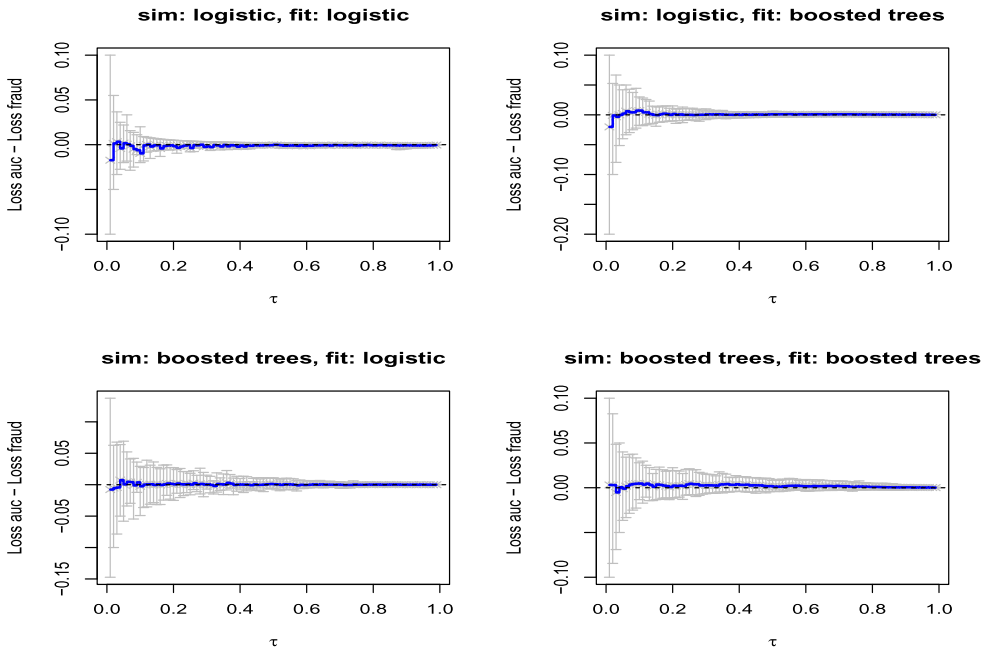
We repeat the experiment a final time, now in a context where  $p > n$ . Specifically, we let  $p = 4000$ , while keeping the number of observations at  $n = 1000$ . Everything is scaled up in the same way as when the number of covariates was changed to  $p = 1000$ , with the exception of the correlation matrix  $R$ , which we now construct from one  $(1000) \times (1000)$  correlation matrix that we stack diagonally, giving us a block-diagonal correlation matrix where  $4 \times 10^6$  out of the  $16 \times 10^6$  entries are non-zero. We see in Tables 8 and 9 that for the case when  $p_0 = 0.2$ , the models selected using the fraud loss did somewhat better aggregated over all values of  $\tau$ , except when estimating an additive tree model on data from a logistic regression model. For the aggregate over the smaller range of  $\tau$ , the models



**Figure 2.** Plot of the difference in  $RAFL(\tau)$  when selecting the model according to the AUC and fraud loss, respectively. Data are simulated from a logistic regression model or additive tree model, with  $n = 1000$  observations,  $p = 100$  covariates, and  $p_0 = 0.2$ . The grey bars correspond to empirical 80% confidence intervals.

selected using the fraud loss were only better when estimating a logistic regression model to data simulated from a logistic regression model. For the case where  $p_0 = 0.05$ , the models selected using the fraud loss were better aggregated over all  $\tau$ -values when the data were simulated from an additive tree model. For the aggregate over the smaller range of  $\tau$ , they were also better when estimating a penalised logistic regression model to data simulated from a logistic regression model. The difference in fraud loss for the models selected using the two approaches is also displayed in Figures 4 and 5 in the same way as for  $p = 100$  and  $p = 1000$ . In some of the cases, it is still hard to see a large clear difference between the two models. However, for  $p_0 = 0.2$ , the models chosen using the fraud loss seem to do better for smaller values of  $\tau$  fitting a logistic regression model to data simulated from an additive tree model. For  $p_0 = 0.05$ , the performance of the two methods was similar when the data were simulated from the same type of model as the estimation model, whereas the fraud loss based method worked better than the AUC based one when the a penalised regression model was fitted to data simulated from a tree model, and somewhat worse when a tree model was fitted to data simulated from a logistic regression model, especially for  $\tau$  up to around 0.1.

All in all, setting the tuning parameter by cross-validating the fraud loss, seems to work quite well, compared to using the AUC, especially for values of  $\tau$  smaller than or close to the marginal probability  $p_0$  of  $Y = 1$ . This is good, as one could argue that these are the most interesting values in most fraud detection applications. The fraud loss does not outperform the AUC in all cases, such as for the boosted tree models for  $p > n$ , or the penalised



**Figure 3.** Plot of the difference in  $RAFL(\tau)$  when selecting the model according to the AUC and fraud loss, respectively. Data are simulated from a logistic regression model or additive tree model, with  $n = 1000$  observations,  $p = 100$  covariates, and  $p_0 = 0.05$ . The grey bars correspond to empirical 80% confidence intervals.

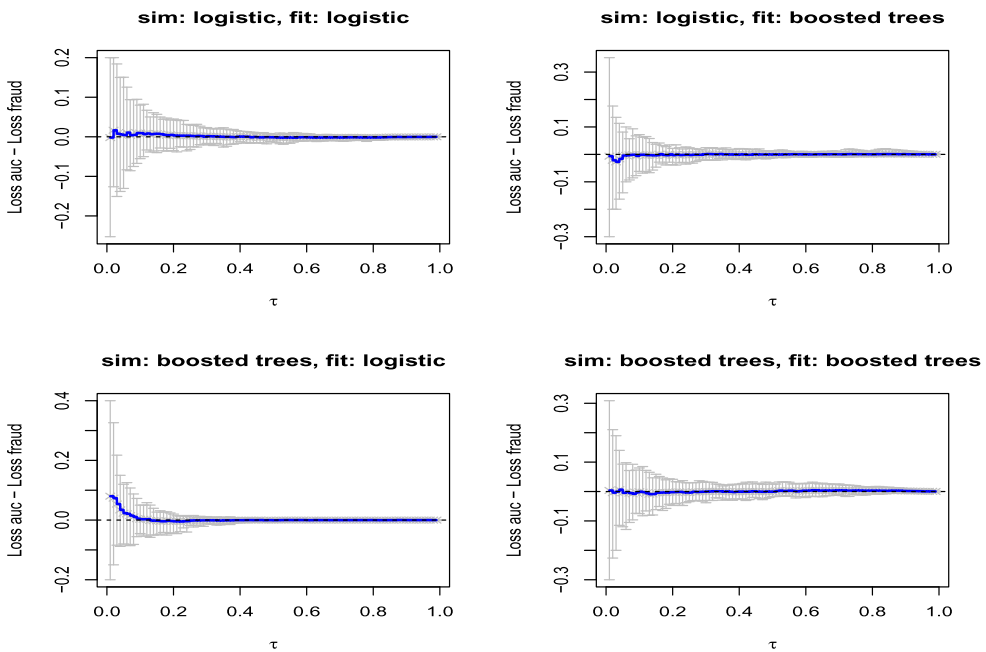
**Table 8.**  $RAFL(\tau)$  averaged over the different values of  $\tau$ , comparison of AUC and fraud loss.  $n = 1000$  observations with  $p = 4000$  covariates, both in training and test sets where  $p_0 = 0.2$ .

Simulation model	Logistic	Logistic	Additive trees	Additive trees
Estimation model	Logistic	Additive trees	Logistic	Additive trees
Average over all $\tau = 0.01, 0.02, \dots, 0.99$ :				
auc	1.0415	1.0526	1.0229	1.0453
fraud	1.0400	1.0577	1.0182	1.0457
auc, 2x repeat	1.0412	1.0520	1.0229	1.0443
fraud, 2x repeat	1.0388	1.0547	1.0187	1.0438
Average over $\tau = 0.16, 0.17, \dots, 0.24, 0.25$ :				
auc	1.0538	1.0605	1.0187	1.0507
fraud	1.0498	1.0655	1.0223	1.0585
auc, 2x repeat	1.0532	1.0567	1.0189	1.0491
fraud, 2x repeat	1.0468	1.0591	1.0239	1.0530

regression models when the data are simulated from a logistic regression model, and  $p < n$ . A possible explanation for the first case could be that when estimating the probabilities is hard, i.e. that  $p > n$ , and the data generating model is complex, then trying to adapt the model locally to  $\tau$  could introduce instability. For the latter case, it could be that the estimation problem is so simple that we more or less recover the data generating model, and therefore, there might not be too much to gain from adapting the model to a specific  $\tau$ .

**Table 9.**  $RAFL(\tau)$  averaged over the different values of  $\tau$ , comparison of AUC and fraud loss. 1000 observations with 4000 covariates, both in training and test sets where  $p_0 = 0.05$ .

Simulation model	Logistic	Logistic	Additive trees	Additive trees
Estimation model	Logistic	Additive trees	Logistic	Additive trees
Average over all $\tau = 0.01, 0.2, \dots 0.99$ :				
auc	1.0235	1.0207	1.0148	1.0188
fraud	1.0233	1.0227	1.0108	1.0185
auc 2x repeat	1.0199	1.0198	1.0162	1.0197
fraud 2x repeat	1.0225	1.0219	1.0108	1.0188
Average over $\tau = 0.03, 0.04, \dots 0.09, 0.10$ :				
auc	1.0895	1.0762	1.0389	1.0551
fraud	1.0759	1.0863	1.0284	1.0576
auc 2x repeat	1.0787	1.0745	1.0423	1.0582
fraud 2x repeat	1.0727	1.0845	1.0279	1.0572



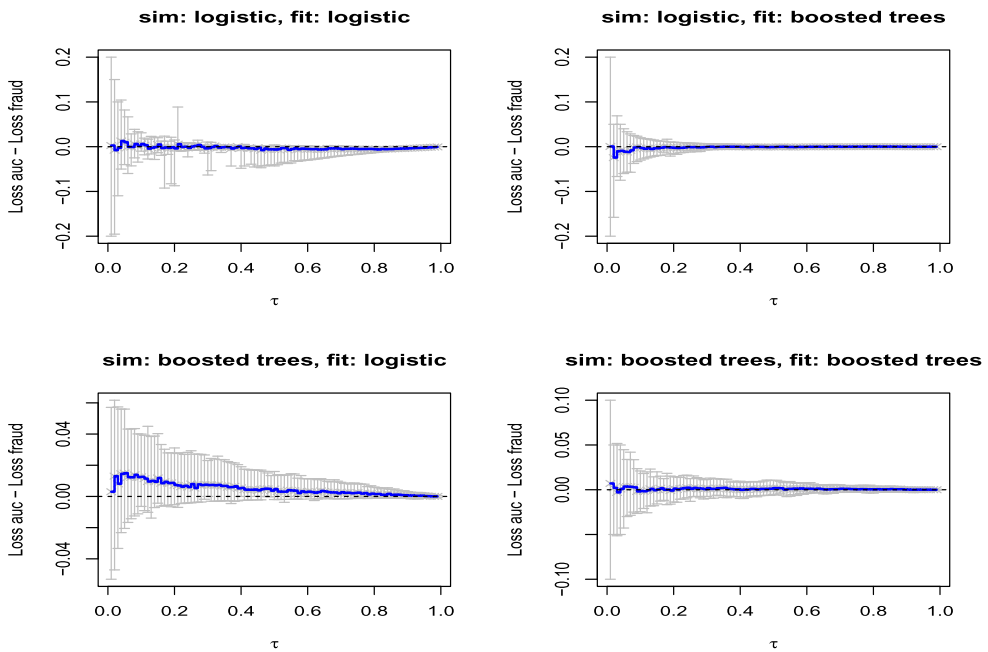
**Figure 4.** Plot of the difference in  $RAFL(\tau)$  when selecting the model according to the AUC and fraud loss, respectively. Data are simulated from a logistic regression model or additive tree model, with  $n = 1000$  observations,  $p = 4000$  covariates, and  $p_0 = 0.2$ . The grey bars correspond to empirical 80% confidence intervals.

### 6. Illustration on credit card default data

In this section, we will consider credit card default data, that are published on Kaggle by Mishra [16]. These are not fraud detection data, as no such real data are publicly available. However, they share many of the same characteristics as fraud detection data, in particular the binary outcome and class imbalance.

The dataset consists of 307511 observations representing credit card clients, with a binary outcome variable (default – no default), where roughly 8% are recorded as defaults,





**Figure 5.** Plot of the difference in  $RAFL(\tau)$  when selecting the model according to the AUC and fraud loss, respectively. Data are simulated from a logistic regression model or additive tree model, with  $n = 1000$  observations,  $p = 4000$  covariates, and  $p_0 = 0.05$ . The grey bars correspond to empirical 80% confidence intervals.

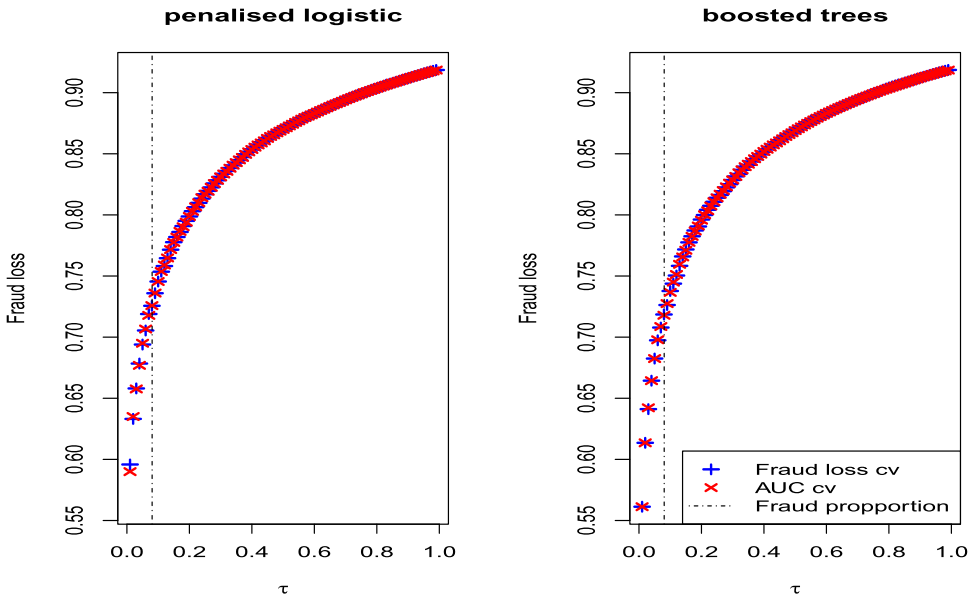
as well as 122 covariates. There are too many covariates to list all of them in detail, but they include information such as the clients gender, the size of the credit card loan, whether the client owns a car, whether the client provided various information, the clients age, and different credit score values. A full overview of all the covariates is available on Kaggle [16]. Some of the covariates are categorical, for example one of the covariates is a record of the education level of the client, which has 5 categories. We recode all such covariates as  $L-1$  binary variables, where  $L$  is the number of categories of the variable. After we have recoded the data in this way, the dataset now consists of  $p = 230$  covariates. There are some missing entries in the dataset, which we chose to impute using the median value. The data are split randomly into a training and a test set of roughly the same size.

On the training data, we then fitted a logistic regression model with a ridge-type penalty, and a boosted tree model where each tree is allowed to have a maximum depth of 3, and chose the number of trees, and the size of the penalty using repeated 2-fold cross-validation of the fraud loss and the AUC, with 9 repetitions. For each value of  $\tau = k/n$  on  $0.01, \dots, 0.99$ , we set the tuning parameter, either the number of components in the tree model or the penalty parameter in the penalised logistic regression, by cross-validating the fraud loss. In addition, we also set tuning parameters by cross-validating the AUC. Then, we evaluated the chosen models on the test data.

The results are summed up in Table 10 and Figure 6, where the fraud loss is plotted as a function of  $\tau$  for both models, and both of the ways of setting the tuning parameter. As we can see from Figure 6, there is little difference between the two ways of choosing the

**Table 10.**  $RAFL(\tau)$  averaged over different values of  $\tau$ , with aggregated fraud loss (i.e. the numerator of  $RAFL(\tau)$ ) given in parentheses. Comparison of AUC and fraud loss on the credit default dataset from Kaggle.

Estimation model	Logistic	Additive trees
Average over all $\tau = 0.01, 0.02, \dots, 0.99$ :		
auc	1.00045 (0.8474)	1.00104 (0.8450)
fraud	1.00057 (0.8475)	1.00097 (0.8449)
Average over $\tau = 0.05, 0.06, \dots, 0.12, 0.13$ :		
auc	1.0015 (0.7338)	1.0042 (0.7252)
fraud	1.0011 (0.7335)	1.0036 (0.7248)



**Figure 6.** Plot of the fraud loss when selecting the model according to the AUC and fraud loss, respectively on the credit default dataset from Kaggle.

tuning parameters for this dataset. Further, Table 10 shows that the performance is almost identical when we average over all of the values for  $\tau$ . For the smaller range from  $\tau = 0.05$  to  $\tau = 0.13$ , i.e. in closer proximity to the marginal balance of the data, the models where we chose the tuning parameter by cross-validating the fraud loss were slightly better than the model found by cross-validating the AUC for the additive tree model, and quite similar for the ridge model. Finally, the fraud loss is smaller for the fitted tree models than for the ridge ones, using both approaches, which suggests that the tree models are the most appropriate for these data.

### 7. Concluding remarks

Statistical fraud detection consists in creating a system, that automatically selects a subset of the cases that should be manually investigated. However, the investigator is often limited to controlling a restricted number  $k$ , or proportion  $\tau$ , of cases. In order to allocate the

resources in the most efficient manner, one should then try to select the cases that have the highest probability of being fraudulent. Prediction models that are used for this purpose, must typically be regularised to avoid overfitting. In this paper, we propose a loss function, the fraud loss, for selecting the complexity of the prediction model via a tuning parameter. More specifically, we suggest an approach where either a penalised logistic regression model, or an additive tree model is fitted by maximising the log-likelihood of a binary regression model with a logit-link, and the tuning parameter is set by minimising the fraud loss function.

In a simulation study, we have investigated different ways of selecting the model complexity with the fraud loss, taking the out-of-sample performance into account, either by cross-validation or bootstrapping. Based on this, repeated cross-validation with few folds seems to be the most favourable. In particular, we have opted for repeated 2-fold cross validation without stratification. Still, we recognise that stratification might be necessary if there are very few cases of fraud in the training data.

Then, we carried out a larger simulation study, where we compared the performance of setting tuning parameters by cross-validating the fraud loss, to cross-validating the AUC. In these simulations, we saw that the fraud loss in some cases gave better results than the AUC, especially when  $p \leq n$ , and in some cases similar results. We saw a tendency that the fraud loss can work better when the proportion of the cases we select is smaller than or close to the marginal probability of fraud. We have also illustrated our approach on a dataset of credit card defaults, where we made the same comparison as in the second round of simulations. In this example, the fraud loss and the AUC performed quite similarly, but the fraud loss did somewhat better for the additive tree models, which also resulted in lower fraud loss than the ridge models, suggesting that the tree models were the most adequate for these data.

We have focussed on two particular estimation methods and corresponding definitions of the model complexity. The first is maximising the logistic log-likelihood function, subject to ridge regularisation, where the penalty parameter is the one to be chosen. The second is boosting for an additive tree model, where the number of trees is the focus. The first could however be easily adapted to the other types of regularisation, such as the lasso or the elastic net. For the second, one might define the complexity in terms of for instance the size of each tree. One could also imagine using the fraud loss to select the complexity for other types of binary classification models.

Further, one might search for new divergences to optimise, that put more emphasis on estimating the higher probabilities accurately, which is similar to the work of Rudin [19], Boyd *et al.* [2] and Eban *et al.* [7]. Another alternative is to adapt regression trees to the problem of picking a certain number of cases. This might be done by fitting small trees directly combined with bagging, or by pruning a decision tree to minimise the fraud loss after growing the tree using a standard splitting criterion.

## Acknowledgements

The authors would also like to thank Riccardo De Bin, for his useful input, and participation in discussions. We also thank the referees and Associate Editor for their help to improve this paper with their constructive comments and suggestions.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

This work is funded by The Research Council of Norway centre Big Insight [Project 237718].

## References

- [1] R.J. Bolton and D.J. Hand, *Statistical fraud detection: A review*, Stat. Sci. 17 (2002), pp. 235–249.
- [2] S. Boyd, Corinna. Cortes, Mehryar. Mohri, and Ana. Radovanovic, *Accuracy at the top*, in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., Curran Associates, Inc., 2012, pp. 953–961. Available at <http://papers.nips.cc/paper/4635-accuracy-at-the-top.pdf>.
- [3] T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system*, Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 785–794.
- [4] S. Cléménçon, G. Lugosi, and N. Vayatis, *Ranking and empirical minimization of  $u$ -statistics*, Ann. Stat. 36 (2008), pp. 844–874.
- [5] S. Cléménçon and N. Vayatis, *Ranking the best instances*, J. Mach. Learn. Res. 8 (2007), pp. 2671–2699.
- [6] A.V. Dorogush, V. Ershov, and A. Gulin, *Catboost: Gradient boosting with categorical features support*, arXiv preprint arXiv:1810.11363, 2018.
- [7] E.E.T. Eban, M. Schain, A. Mackey, A. Gordon, R.A. Saurous, and G. Elidan, *Scalable learning of non-decomposable objectives*, arXiv preprint arXiv:1608.04802, 2016.
- [8] B. Efron, *Estimating the error rate of a prediction rule: Improvement on cross-validation*, J. Am. Stat. Assoc. 78 (1983), pp. 316–331.
- [9] B. Efron and R. Tibshirani, *Improvements on cross-validation: The 632+ bootstrap method*, J. Am. Stat. Assoc. 92 (1997), pp. 548–560.
- [10] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer, *An efficient boosting algorithm for combining preferences*, J. Mach. Learn. Res. 4 (2003), pp. 933–969.
- [11] J.H. Friedman, *Greedy function approximation: A gradient boosting machine*, Ann. Stat. 29 (2001), pp. 1189–1232.
- [12] A.E. Hoerl and R.W. Kennard, *Ridge regression: Biased estimation for nonorthogonal problems*, Technometrics 12 (1970), pp. 55–67.
- [13] T. Joachims, *A support vector method for multivariate performance measures*, Proceedings of the 22nd International Conference on Machine Learning, ACM, 2005, pp. 377–384.
- [14] H. Joe, *Generating random correlation matrices based on partial correlations*, J. Multivar. Anal. 97 (2006), pp. 2177–2189.
- [15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, *Lightgbm: A highly efficient gradient boosting decision tree*, in *Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.
- [16] A. Mishra, *Fraud detection dataset*. Available at <https://www.kaggle.com/mishra5001/credit-card>, 2019. Accessed: 01/02-2022.
- [17] R.B. Nelsen, *An Introduction to Copulas*, Springer Science & Business Media, 2007.
- [18] S. Robertson and H. Zaragoza, *On rank-based effectiveness measures and optimization*, Inf. Retr. Boston. 10 (2007), pp. 321–339.
- [19] C. Rudin, *The  $p$ -norm push: A simple convex ranking algorithm that concentrates at the top of the list*, J. Mach. Learn. Res. 10 (2009), pp. 2233–2271.
- [20] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. R. Stat. Soc. Ser. B (Methodol.) 58 (1996), pp. 267–288.
- [21] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, J. R. Stat. Soc. Ser. B (Stat. Methodol.) 67 (2005), pp. 301–320.