Credit scoring using neural networks and SURE posterior probability calibration

Matthieu Garcin, Samuel Stéphan

Credit scoring using neural networks and SURE posterior probability calibration *†

Matthieu Garcin[1] and Samuel Stéphan ‡[1,2]

[1]*Léonard de Vinci Pôle Universitaire, Research center, 92916 Paris La Défense, France. E-mail address: matthieu.garcin@m4x.org*

[2]*SAMM, Université Paris 1 Panthéon-Sorbonne, 90 rue de Tolbiac, 75013 Paris cedex 13, France. E-mail address: stephansamuel22@gmail.com*

September 26, 2023

*This work was supported by the Caisse des dépôts et consignations.

†The authors would like to thank Vivien Brunel for useful discussions.

‡Corresponding author

Abstract

In this article we compare the performances of a logistic regression and a feed forward neural network for credit scoring purposes. Our results show that the logistic regression gives quite good results on the dataset and the neural network can improve a little the performance. We also consider different sets of features in order to assess their importance in terms of prediction accuracy. We find that temporal features (i.e. repeated measures over time) can be an important source of information resulting in an increase in the overall model accuracy. Finally, we introduce a new technique for the calibration of predicted probabilities based on Stein's unbiased risk estimate (SURE). This calibration technique can be applied to very general calibration functions. In particular, we detail this method for the sigmoid function as well as for the Kumaraswamy function, which includes the identity as a particular case. We show that the SURE calibration technique is able to calibrate the predicted probabilities as well as the classical Platt method.

Keywords – Deep learning, credit scoring, calibration, SURE

1 Introduction

Credit scoring aims to measure the risk for a bank to grant a loan to an applicant. Depending on the value of the returned score, the bank will accept or not to grant the loan. This score is generally computed by a model which has been fitted on a database containing past information of the consumer behavior and its corresponding credit profile. Typical descriptors of consumer behavior include loan information (amount, maturity, type of interest rate, nature of the loan) and the borrower characteristics (age, marital status, profession, monthly income, personal savings, number of actual loans). The target, also known as dependent feature, corresponds to whether or not the customer has defaulted on its loan. The default is subject to a formal definition given by the Basel committee which states that the bank is facing a default event if the counterpart past due is more than 90 days. Thus, the feature to be predicted is coded as 1 if the borrower did default and 0 otherwise.

Correct risk assessment is an important aspect of banking activities. To ensure robust estimation of the risks, the regulator has framed several rules to follow. These rules are edited by the Basel Committee on Banking Supervision (BCBS) which is the primary global standard setter for the prudential regulation of banks. Several updates have been made by the regulator to fit the development of banking activities over the years. Currently, banks are subject to Basel III agreements that aim to ensure that banks have a rigorous approach to risk and capital management related to their activities. The Basel framework lets the choice for each financial institution to manage its credit risk assessment through standard or internal methods. In the latter case, the bank uses a modeling framework for estimating the risk parameters. In the Internal Rating Based (IRB) approach, the bank estimates only the probabilities of default (PD). In the advanced IRB, the bank has to estimate the expected credit loss using the following parameters: the Loss Given Default (LGD), Exposure at Default (EAD), Maturity of exposures (M), and the PDs. As its name

indicates, the PD risk parameter consists in estimating the likelihood that each loan is going to be repaid. This is typically what is evaluated in credit scoring applications.

From a practical point of view, the estimation of the PDs corresponds to a classification problem. The specificity of this problem is that we want the classifier to output a probability of the event while in numerous other applications we only want the outcome. Traditional machine learning models such as logistic regression and linear discriminant analysis are well suited for this task (Altman, 1968; Steenackers & Goovaerts, 1989). Despite their simplicity, they are probably the most widespread models for credit scoring applications since they are very well understood tools and easy to use (Anderson, 2007; Dumitrescu et al., 2020). Furthermore, they are implemented in most statistical software and the computation of the final prediction is straightforward from the features coefficients. Today, a large area of research in credit scoring consists in the development of new scoring techniques. This literature is motivated by the limitations of the standard techniques. Indeed, in their simplest design, logistic regression and linear discriminant analysis exploit only linear interactions. Thus, the rise of performances of credit scoring models has been initiated by the use of ensemble methods that enable nonlinearities and provide high generalization capabilities (Baesens et al., 2003; Finlay, 2011; Paleologo et al., 2010; Wang et al., 2011). Other recent advances include the use of hybrid methods (Huang et al., 2007; Lee et al., 2002; Zhu et al., 2018) and deep learning which have first achieved promising results in the field of computer vision (Albanesi & Vamossy, 2019; Kvamme et al., 2018; LeCun et al., 1998; Tai & Huyen, 2019; Tygert et al., 2016; Voulodimos et al., 2018). Nevertheless, credit scoring remains a field where deep learning has trouble asserting itself because of strong regulatory requirements (Alonso & Carbó, 2020; Bücker et al., 2022; Gunnarsson et al., 2021; Hjelkrem & Lange, 2023; Lessmann et al., 2015).

In this paper, we investigate how powerful are these deep neural networks through a feed-forward architecture and we compare the results with a standard logistic regression. We show that the use of a deep neural network leads, on our dataset, to a slight improvement in the forecast compared to the logistic regression. We also investigate the properties of our dataset which has the particularity to contain a mix of static features and temporal features. To the best of our knowledge, no previous study on credit scoring has ever explored the ability of a model to provide accurate results depending on the static/dynamic nature of the data. The procedure we use is as follows: we split the dataset into static features set and dynamics features set. We then evaluate the models on three sets: static, dynamic, and all features, and finally compare the results. We give evidence that temporal data should not be neglected in credit scoring applications. In our dataset, dynamic features drastically improved the results compare to feeding the model only with static features.

Another important aspect of credit scoring application is to ensure that the estimated probabilities are close to the true probabilities. This can be analyzed in terms of cost. On one hand, machine learning models assume all estimated probabilities to have the same cost. On the other hand, in many decision-making applications, not having an accurate probability of belonging to the target class can be costly. For instance, it can be the choice of a treatment for a patient in

medicine, a driving decision in a self-driving car application, or an investment decision in business. A common approach consists in post-processing the probabilities of a classifier in order to get calibrated outputs. Calibration can be performed using parametric and non-parametric techniques. Non-parametric techniques include histogram binning (Zadrozny & Elkan, 2001), isotonic regression (Zadrozny & Elkan, 2002), similarity binning averaging (Bella et al., 2009), and adaptive calibration of predictions (X. Jiang et al., 2012). All these techniques consist in binning the samples and assigning them a calibrated probability. We argue that probabilities calibrated by these techniques can be highly biased. Indeed, each bin requires a big enough number of instances in order to have a low variance estimate of its average default rate, thus leading to a unique estimated PD for all the instances in this bin. Choosing the optimal number of bins, balancing bias and variance, is not obvious as well. Parametric methods include Platt scaling (Platt, 1999), beta calibration (Kull et al., 2017), asymmetric Laplace method (Bennett, 2003), and piecewise logistic regression (Zhang & Yang, 2004). Since these parametric approaches rely on the assumption that the probabilities follow a particular distribution, they are subject to model mismatch. In this paper, we propose a novel parametric approach for probability calibration. We use Stein's Unbiased Risk Estimate (SURE) as a proxy to minimize the Mean Squared Error (MSE) between the estimated probability and the true probability. The calibrated density is then the one minimizing the estimated MSE. This technique has been used intensively in the field of signal processing for image denoising. One advantage of this approach is that we do not rely on predefined bins which are biased by nature. Besides, our method offers the possibility to use a custom calibration function to prevent from model mismatch. We empirically show that this new technique is as accurate as the standard Platt method for the calibration of the PDs.

The major contribution of this article to the credit scoring literature is the introduction of a new methodology to calibrate the predicted probability to the hidden true probabilities using the SURE approach. In addition, we investigate the efficiency of models based on deep learning versus traditional models when estimating individual PDs. We also evaluate the impact of using time series data in credit scoring models.

The article is organized as follows. Section 2 describes the data used for the application, exposes the evaluation of the models and the feature importance assessment, and presents the models. In Section 3, we demonstrate the necessity of calibration in machine learning and we propose a new framework to calibrate posterior predicted probabilities. Section 4 compares empirically the models and assesses the efficiency of the proposed calibration approach.

2 Experimental setting

In this section, we describe how we design the experiment. First, we present the dataset on which models have been fitted. We precise what preprocessing steps have been applied and how we split the data in order to assess the performance of our model. Next, we explain how we organized the features such that we were able to assess the importance of static and dynamic features in the

performance. Then, we discuss the choice of a proper evaluation metric in credit scoring. Indeed, credit scoring is an area in which the event to predict is difficult because of class imbalance. This characteristic should drive the choice of the measure to get an unbiased measure of performance. Finally, we present the models we use for the application.

2.1 The dataset

In every machine learning application, the quality of data is of primary importance. Ideally, a credit scoring application would include quantitative data such as financial ratios describing the borrower's financial health and its past credit history, including potential default information. Indeed, defaults have been shown to be persistent over time with the reasoning that a borrower who has already defaulted is more likely to stay in this current state (Albanesi & Vamossy, 2019). Qualitative data are useful as well, such as the education level, the type of product granted, or the presence of collateral to secure the loan.

In this study, we use a Taiwanese credit dataset publicly available on the UCI machine learning repository[1] and already used in the machine learning literature (C. Jiang et al., 2023; Yeh & Lien, 2009). This dataset consists of anonymized default payments realized on revolving credit granted in Taiwan. It gathers information of 30 000 credit card users among which 22% default next month. We can notice that the dataset is imbalanced between defaulters and non-defaulters with an imbalance ratio of 3.51%. This is a common characteristic of credit scoring applications which makes the modeling difficult. The dataset includes classic features of a credit scoring dataset such as age, sex, level of education, marital status, and the credit limit of individuals. It also contains time series data observed from April 2005 to September 2005: the monthly bill statement, the monthly paid amount, and an indicator of the revolving credit delay payment. We expect these last dynamic features to be highly informative since it gives a trajectory of the individual account which may reveal interesting patterns for defaulters. Our target is the dummy variable indicating whether the client defaults next month (No = 0, Yes = 1). To complete the dataset, we have created 6 additional features, one for each month of observation, consisting of the ratio of the bill amount by the credit limit. We expect this set of new features to improve the model's ability to learn the default drivers, since a high bill statement relative to the credit limit may imply payment difficulties.

We apply the classical preprocessing steps to the features in order to facilitate training. We standardize the distribution of continuous data to have a mean value of 0 and a standard deviation of 1. This typical operation is done in order to speed up the learning process. We then split the dataset into a shuffled and stratified train, a validation, and a test set representing respectively 60% - 20% - 20% of the data available. We use the train and validation sets to train each model and tune their hyperparameters while we use the test set only once to assess the out-of-sample performance. This last set allows us to estimate an unbiased performance of the classifier.

[1] https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

2.2 Feature importance assessment

In this study, we are interested in assessing the importance of different sets of features instead of each feature individually. Indeed, we want to assess the importance of the nature of data in the model's performance. In particular, we expect times series to be highly informative since they reflect the behavior of the borrower. Thus, we defined three subsamples of features which are detailed in Table 1. Note that all dynamic data are time series available from April 2005 to September 2005.

Static features	Amount of given credit (NT dollar), Gender, Marital status, Age (year).
Dynamic features	History of past payment tracked via past monthly payment records, Repayment status, Amount on bill statement, Amount of previous payment, Ratio of bill statement relative to the credit limit
All features	Static features + Dynamic features

Table 1: Features sets

We expect time series data to be highly informative since it gives a dynamic view of the borrower's credit activity. Moreover, classical scoring models often incorporate an aggregated vision of the borrower's behavior instead of a detailed monthly vision. We think that the industry could benefit from the use of time series data in their internal credit scoring models in terms of predictive accuracy. Using this particular setting, we are able to train each model on different sets of features and evaluate their inner performances on the corresponding independent testing sets. Then, we are able to identify the most informative set of features by picking the one that achieves the highest performance on the test set.

2.3 Choosing the right evaluation metric

Credit scoring is typically an area where class imbalance can be severe because risk management aims to ensure that the number of defaulters stays much smaller than non-defaulters. Such characteristics make learning difficult. This is why several techniques have been introduced. Data-level techniques consist in resampling the dataset in order to balance the classes. These techniques include random over-sampling, random under-sampling, and the creation of synthetic samples via the SMOTE algorithm (Chawla et al., 2002; H. He & Garcia, 2009). However, a recent comparison of oversamplers tends to demonstrate that oversampling doesn't improve the performance of classifiers on imbalanced datasets (Kovács, 2019). Therefore we choose not to use these techniques in this study. Other strategies for balancing the classes consist in rendering the model cost-sensitive by adding a cost to misclassified instances of both positive and negative class. This improves the model's ability to correctly classify positive samples. Such methods include weighting and thresholding (Sheng & Ling, 2006; Sun et al., 2009). In this article, our choice goes to weighting the loss function as detailed in section 2.4.1. This choice is motivated by the fact that being a model-level

technique, it doesn't change the structure of the data. Therefore, the original statistical properties of the dataset are preserved which is preferable from a regulatory point of view. We also applied thresholding to the predicted probabilities as explained below.

The class imbalance also makes the choice of the appropriate evaluation metric quite challenging. Indeed, standard measures can't be applied because they tend to be biased toward the majority class. Among these evaluation metrics, we can cite, for instance, the accuracy, which is defined as the sum of true positives and true negatives over the whole dataset. The accuracy is not a good choice of metric since a classifier biased toward the negative class will always reach a high accuracy. In this study, we propose an evaluation based on the precision, the recall and the F1 score. Let's consider a classical binary classification problem with a model fitted on a training set. We want to evaluate this model on a test set. We define $y_i \in \{0, 1\}$ as the true label of the i^{th} instance. The label y_i equals 1 if the instance is tagged as positive and 0 if tagged as negative. The model outputs a quantity $\widehat{p_i} \in [0, 1]$ which is often interpreted in the literature as the probability of a given instance of belonging to the positive class. For a given threshold $\tau \in [0, 1]$, the predicted label is defined as $\widehat{y_i} = 1$ if $\widehat{p_i} > \tau$ and $\widehat{y_i} = 0$ otherwise. Given these notations, we can compute the number of True Positives ($TP = \sum_{i=1}^{N} \mathbb{1}_{(y_i=1 \cap \widehat{y_i}=1)}$), True Negatives ($TN = \sum_{i=1}^{N} \mathbb{1}_{(y_i=0 \cap \widehat{y_i}=0)}$), False Positives ($FP = \sum_{i=1}^{N} \mathbb{1}_{(y_i=0 \cap \widehat{y_i}=1)}$), and False Negatives ($FN = \sum_{i=1}^{N} \mathbb{1}_{(y_i=1 \cap \widehat{y_i}=0)}$). Then we can compute the recall, the precision, and the F1 score:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

The F1 score is a combination of the precision and the recall. Maximizing this metric thus leads to balance the effect of false negative and false positive, giving an unbiased estimate of how the model is performing.

Another well-known measure for model evaluation is the ROC curve. This curve is obtained by computing the specificity and the corresponding sensitivity for various probability thresholds of the classifier. We then plot the sensitivity against the specificity. One convenient measure associated with the ROC curve is the area under the curve (AUC), with AUC > 0.5 meaning that the model is performing better than a random classification. These metrics are also not good candidates for the evaluation of imbalanced datasets classifiers. The reason is that the ROC curve is used to assess the overall performance in discriminating the positive class and the negative class while most of the time, in imbalanced learning, we are only focusing on the correct classification of positive samples. This generally leads to an overconfident estimate of how well the classifier is performing because it doesn't take into account that the classifier is biased toward the negative class containing more samples. Instead, we can rely on the Precision-Recall (PR) curve which focuses on the performance of the model in classifying positive samples only (Davis & Goadrich, 2006; Saito & Rehmsmeier,

2015). This curve is obtained by plotting the precision and recall for various probability thresholds. As in the ROC case, we can compute the corresponding AUC of the PR curve (Boyd et al., 2013).

We consider the recall, the precision and, the F1 score to evaluate the performance of our classifiers for two reasons. First, our dataset is imbalanced as it contains 22% of defaulters. Second, in this imbalanced framework, our preference goes to a better classification of positive samples, in order to diminish the risk for the lender. These measures being based on the confusion matrix, the attribution of classes is ultimately done by applying a threshold value to all samples. Most software considers a default probability threshold of 0.5 which is not recommended for imbalanced datasets (Lipton et al., 2014; Provost, 2000; Q. Zou et al., 2016). We choose to tune the threshold value such that we achieve the best F1 score in a simple manner. The procedure is as follows: we move the decision threshold for the predicted probabilities of the training set. For each of these thresholds, we assign a class to each sample and compute the corresponding F1 score. We then apply this threshold to the test set to compute the out-of-sample F1 score. We also consider a global measure of performance such as the AUC-PR and display the AUC-ROC for comparison purposes.

2.4 Models

2.4.1 Logistic regression

The logistic regression will be our baseline model since it is still one of the most widely used in the banking industry for credit scoring (Anderson, 2007). To take into account the slight imbalance of classes (i.e. 22/78), we trained the model using the balanced cross-entropy loss (Xie & Tu, 2015). We consider the learning set $\{(x_i, y_i) \in (\mathcal{X} \times \mathcal{Y}) | i = 1, ..., N\}$ where y_i are the labels, x_i the inputs, and N the total number of instances. We want to estimate a function $f(x|\theta)$ which maps the inputs x_i to the output y_i, where θ is a set of parameters to be optimized on the training set. We denote $\widehat{p_i} = f(x|\theta)$ the model's output and n^+ the number of positives samples. We estimate θ by minimizing the loss function over all the instances:

$$\mathcal{L}(y_i, \widehat{p_i}) = -[\alpha y_i \log(\widehat{p_i}) + (1 - \alpha)(1 - y_i) \log(1 - \widehat{p_i})], \tag{1}$$

where $\alpha = \frac{n^+}{\mathrm{N}}$. Intuitively, the term α accounts for class imbalance insofar as mispredictions on both the positive and the negative class are penalized.

2.4.2 Feed forward neural network

We specify a neural network with a feed forward architecture for predicting the occurrence of defaults. The advantage of such a model over traditional methods is that we can easily modify the network in order to perfectly scale the problem. This is stated in the universal approximation theorem (Hornik, 1991):

> "A single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units."

Moreover, it has been shown that it is more efficient in terms of predictive performance to build a multi-layer neural network since it is able to learn deeper representations (Bengio & LeCun, 2007). Convolutional neural networks for instance are able to learn complex concepts such as edges or geometric shapes of an image. Also, multi-layer neural networks need exponentially fewer neurons than shallow networks to learn data representations.

Our network architecture is composed of an input layer, three hidden layers of 60 hidden units each, and an output unit which uses a sigmoid activation function in order to output a probability estimate. We introduce nonlinearities in our network with the help of the ReLU activation function defined by:

$$\text{ReLU}(x) = \max(0, x).$$

We used the ReLU activation function for several reasons. First, because the ReLU is fast to compute compared to other activation functions such as the hyperbolic tangent or the sigmoid. Its gradient computation is very simple (either 0 or 1) which helps backpropagating the signal through each layer and has the effect of speeding up the training. Therefore, in a ReLU feed forward network, not all neurons are activated at the same time. Indeed, a proportion of the neurons output zero so that the network is less complex than its full architecture and the computations are more time efficient than in a neural network using tanh or sigmoid activations. This particular choice of activation function is also motivated by the literature because it decreases the training time and can approximate any continuous function (Krizhevsky et al., 2012).

We train the network using the balanced cross-entropy loss presented in equation (1). This common framework makes it possible to compare fairly the performance of the two models since their prediction abilities will come from their intrinsic architecture.

The weights of the network have been optimized using the Adam algorithm with a batch size of 256 samples (Kingma & Ba, 2015). Indeed, a recent study comparing various optimization methods has shown that using Adam optimizer, in the context of training a neural network, improves the performance compared to classical optimizers like SGD or Nesterov (Choi et al., 2019). Based on the literature, we initialize carefully the weights using He initialization scheme (Glorot & Bengio, 2010; K. He et al., 2015). This approach is widely used by practitioners to ensure that gradients don't vanish or explode during training.

3 Probability calibration

We first review the existing literature on calibration. We present the original use of SURE and we connect this approach to the machine learning area. Then, we detail our SURE calibration framework and we give a pseudo-code to enhance the comprehension and allow reproducibility.

We conclude this section by giving evidence of why our framework should be preferred to binning and we report the evaluation measure we used to estimate the calibration error.

3.1 Literature review

Ideally, we would like machine learning models to output accurate probabilities in the sense that they reflect the real unobserved probabilities, that is $\mathbb{P}(\widehat{Y} = 1) = p$, where $\widehat{Y}$ is a class prediction, and $p \in [0, 1]$ is the true probability. This is exactly the purpose of calibration techniques, which aim to map the predicted probabilities to the true ones in order to reduce the probability distribution error of the model. Probability calibration is very important in many real-world classification tasks. Indeed, most of the time, classification models are evaluated globally, using synthetic measures, without taking care of how the error is distributed. Thus, one can't identify the confidence of the predicted probabilities. Calibration is overriding for instance in medicine (Cearns et al., 2019; Chen et al., 2018), for self-driving cars (Feng et al., 2019), or in finance (Bequé et al., 2017). Indeed, reliable probabilities are preferable to take accurate decisions and reduce the risks.

In credit scoring applications, having accurate probabilities is a matter of concern in the perspective of improving risk assessment. Good calibration can result in significant gains for financial institutions since they will be able to correctly assess the risk related to each borrower (Blöchlinger & Leippold, 2006). Besides, predicted probabilities that correctly match the empirical distribution lead to better risk management and anticipation when it comes to model different portfolio scenarios and evaluate expected losses (Bequé et al., 2017).

Usually, calibration of predicted probabilities is performed using histogram binning (Zadrozny & Elkan, 2001), Platt scaling (Platt, 1999), or isotonic regression (Zadrozny & Elkan, 2002). Concretely, histogram binning consists in dividing the samples into equal bins and assigning them a calibrated probability. But due to the fact that $\widehat{P}$ is a continuous random variable, the calibration cannot be computed using finitely many samples without introducing a bias. In the case of Platt scaling and isotonic regression, the method consists in fitting a model to regress the predicted probabilities on the real labels. This way, we obtain a continuous estimation of the true probability. We can also mention the quasi-moment-matching methods based on the cumulative accuracy profile (CAP) that consists in minimising the MSE of the errors defined as the difference between the empirical CAP curve and its equivalent parameter (Tasche, 2009). The calibration model is estimated on the validation set and is then used to predict default probabilities on the test set. Recently, calibration has been rediscovered with the deep learning booming trend. Indeed, deep learning models are no exception to the rule and are even more than traditional methods subject to model uncertainty due to their complex architectures (Antorán et al., 2020). That's why measuring calibration errors and developing calibration techniques for deep neural networks have become an important research topic these recent years (Guo et al., 2017; Nixon et al., 2019; Pan et al., 2020).

In this paper, we are introducing a new calibration method based on SURE. This method has

originally been developed to estimate the MSE[2] of any nonlinear differentiable estimator whose input observations are assumed to be Gaussian (Stein, 1981). We think that this approach is well suited for probabilities calibration since it allows to measure an unobserved error. In our case, this error is the difference between the real unobserved probability and the estimated one. This technique has been used intensively in signal processing, mostly as a denoising technique. One of the major applications in this field was to select the optimal threshold of noisy wavelet coefficients in order to recover a signal (Donoho & Johnstone, 1995; Garcin & Guégan, 2016). The SURE method also appears in the machine learning literature for model selection. Many modern machine learning algorithms require shrinkage through a regularization parameter to get models that generalize well to new data. In this case, minimizing the SURE can be an approach to find this optimal regularization parameter (Abadie & Kasy, 2019; Efron et al., 2004; Li & Zhu, 2008; Tibshirani & Taylor, 2011; H. Zou et al., 2007). Currently, most practitioners use cross-validation as an estimate of the model's MSE because it doesn't require normality assumptions and it's easy to implement. However, research has shown that both methods give robust results (Abadie & Kasy, 2019).

In the deep learning area, the SURE method is used for neural network based denoising algorithms. It has been proposed to train a deep denoiser network only on noisy training (Soltanayev & Chun, 2018). The model outperformed the classical non-deep learning based denoisers. This approach has been extended for learning with correlated pairs of noisy images and compressed sensing (Metzler et al., 2020; Zhussip et al., 2019).

In this work, we calibrate the output of our models using our SURE framework. This choice is motivated by the fact that we want to minimize the error depending on the unobserved true probabilities. Thus cross-validation cannot be applied since we never observe the true probabilities. Furthermore, we argue that our SURE framework gives a better estimation of the calibration error than histogram binning. Indeed, in histogram binning, the choice of the number of bins introduces a bias due to the unequal number of samples falling in each bin. Finally, we compare our method to Platt scaling which is a parametric method in the same vein as our SURE framework.

3.2 Platt scaling

In this subsection, we give a brief overview of the Platt scaling method which is the standard approach for probabilities calibration. This approach consists in using the predicted probabilities of a classifier as an input of a logistic regression. Thus, probabilities are modified by the sigmoid function below:

$$\widetilde{p_i} = \frac{1}{1 + e^{-(\theta_1 \widehat{p_i} + \theta_2)}}, \tag{2}$$

where $\widehat{p_i}$ is the predicted probability of observation i, $\widetilde{p_i}$ is the calibrated probability and θ_1, θ_2 the parameters of the logistic regression. These parameters are to be estimated so as to make the

[2]Called risk in the literature about signal processing, without any link with the finance-related notion of risk introduced in this paper.

calibrated probability close to the true probability p_i. However, p_i is not observed and Platt's method puts forward the labels y_i instead. Thus, we estimate θ_1 and θ_2 by maximizing the likelihood, in the calibrated probability, of observed data, namely the labels. The labels can take values 0 or 1. Thus, it can be interpreted as the realisation of a Bernoulli random variable Y such that $Y \sim Ber(\widetilde{p_i})$. Given this general framework, we can write the likelihood of the observations:

$$L(\theta_1, \theta_2) = \prod_{i=1}^{N} \left(\frac{1}{1 + e^{-(\theta_1 \widehat{p_i} + \theta_2)}} \right)^{y_i} \left(1 - \frac{1}{1 + e^{-(\theta_1 \widehat{p_i} + \theta_2)}} \right)^{1-y_i}.$$

This equation is often transformed in log-likelihood for calculus convenience:

$$\mathcal{L}(\theta_1, \theta_2) = \sum_{i=1}^{N} y_i \log \left(\frac{1}{1 + e^{-(\theta_1 \widehat{p_i} + \theta_2)}} \right) + (1 - y_i) \log \left(1 - \frac{1}{1 + e^{-(\theta_1 \widehat{p_i} + \theta_2)}} \right).$$

The log-likelihood is also known as Binary Cross Entropy (BCE). We can maximize this quantity by applying gradient-based algorithms such as steepest descent or quasi-Newton to find an estimate of the parameters θ_1 and θ_2.

3.3 A SURE framework for calibration

3.3.1 General framework

In this framework, we consider the predicted probabilities $\widehat{\boldsymbol{p}} = (\widehat{p_i})_{i \in [\![1,N]\!]}$ of a model to be noisy measurements of the true probabilities. We make the assumption that the relation between the corrupted probabilities and the true probabilities is as follows:

$$\widehat{\boldsymbol{p}} = \boldsymbol{p} + \boldsymbol{\varepsilon},$$

where $\boldsymbol{p} = (p_i)_{i \in [\![1,N]\!]}$ denotes the true probabilities and $\boldsymbol{\varepsilon} = (\varepsilon_i)_{i \in [\![1,N]\!]}$, the noise, is a vector of i.i.d. Gaussian variables of mean 0 and variance σ^2.

We also define the denoising function $G_\theta(.)$ parametrized by $\boldsymbol{\theta}$ and differentiable. This function takes in input the noisy probabilities and output the denoised probabilities that is $\widetilde{p_i} = G_\theta(\widehat{p_i})$. Besides, we restrict the number of parameters to two, that is $\boldsymbol{\theta} = (\theta_1, \theta_2)$. This seems reasonable since Platt method also consists in two parameters. The function $G_\theta(.)$ is increasing, so that the order of the denoised probabilities is respected: if $\widehat{p_i} < \widehat{p_j}$, the calibration must not change the classification and thus $\widetilde{p_i} < \widetilde{p_j}$.

Our goal is to select optimally the parameters $\boldsymbol{\theta}$ in order to obtain an estimate $\widetilde{\boldsymbol{p}} = (\widetilde{p_i})_{i \in [\![1,N]\!]}$ as close as possible to $\boldsymbol{p}$. This kind of problem is typically achieved by minimizing the MSE between

the estimated probability and the true probability:

$$r(\widetilde{\boldsymbol{p}}) = \frac{1}{N}\sum_{i=1}^{N}(\widetilde{p_i} - p_i)^2.$$

However, in practice, we don't have access to $\boldsymbol{p}$. The SURE method proposed by Stein (Stein, 1981) allows us to overcome this difficulty by providing an approximation of the MSE, defined by:

$$SURE(\boldsymbol{\theta}) = -N\sigma^2 + \sum_{i=1}^{N}(G_\theta(\widehat{p_i}) - \widehat{p_i})^2 + 2\sigma^2\sum_{i=1}^{N}\frac{\partial G_\theta(\widehat{p_i})}{\partial \widehat{p_i}}. \tag{3}$$

The SURE loss is an unbiased estimate of the MSE under the assumption of Gaussian noise. Since we want an unbiased estimator, we expect the mean calibrated probability to be equal to the empirical event frequency that is $\frac{1}{N}\sum_{i=1}^{N} G_\theta(\widehat{p_i}) = \frac{1}{N}\sum_{i=1}^{N} y_i$. This leads to the following minimization program:

$$(\mathcal{P}) = \begin{cases} \min\limits_{\boldsymbol{\theta}} & \text{SURE}(\boldsymbol{\theta}) \\ \text{s.t.} & C(\boldsymbol{\theta}) = 0 \end{cases}$$

with $C(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} G_\theta(\widehat{p_i}) - \frac{1}{N}\sum_{i=1}^{N} y_i$.

3.3.2 Choice of the calibration function

In this experiment, we tried the sigmoid and Kumaraswamy repartition functions as denoisers. Both are increasing and differentiable functions on the interval $[0, 1]$. This properties make them suitable for the calibration of probabilities.

The sigmoid function defined in equation (2) is the one we used in the experiment. This choice seems natural since it is also the one used in Platt scaling. Thus, it makes sense to use it for fair comparison of both methods. This distribution has the particularity to assume normally distributed probabilities within each class with same variance. This strong property can reduce the efficiency of the approach which motivates the choice of another function (Kull et al., 2017).

The Kumaraswamy distribution has been presented first for simulations purposes in hydrological data modeling (Kumaraswamy, 1980). This function is of the same family of the beta law and it has the advantage to be explicitly differentiable. We define the cumulative Kumaraswamy distribution function as:

$$G_\theta(\widehat{p_i}) = 1 - (1 - \widehat{p_i}^{\theta_1})^{\theta_2},$$

with $\theta_1 > 0, \theta_2 > 0$.

This function is much more flexible than the sigmoid one because it can map the predicted probabilities to the sigmoid, the inverse sigmoid, and the identity shapes. In particular, if probabilities are already calibrated, the Kumaraswamy can recover the identity while the sigmoid does not, independently to the method employed. The ability to recover different shapes makes the Kumaraswamy suitable for a large variety of models that output different uncalibrated probability distributions.

3.3.3 Estimation of the noise variance

In the above framework we assumed the knowledge of σ^2 which is not true in practice. We want to estimate σ from the data such that it stays constant over the optimization. We have made the assumption that $\widehat{p_i} = p_i + \varepsilon_i$, with $\varepsilon_i \sim N(0, \sigma^2)$. Besides, the labels y_i can be equal to 0 or 1. Thus each observation y_i can be interpreted as the realisation of a Bernoulli random variable such that $y_i \sim Ber(p_i)$. Hence we have $\mathbb{P}(y_i = 1) = p_i$ and $\mathbb{P}(y_i = 0) = 1 - p_i$.

We want to minimize the spread between the estimated probability $\widehat{p_i}$ and the true one p_i. Thus, taking the expectation:

$$\mathbb{E}[(\widehat{p_i} - y_i)^2] = \mathbb{E}[\widehat{p_i}^2] + \mathbb{E}[y_i^2] - 2\mathbb{E}[\widehat{p_i} y_i],$$

where:

$$\begin{aligned}
\mathbb{E}[\widehat{p_i}^2] &= \mathbb{E}[(p_i + \varepsilon_i)^2] = p_i^2 + \sigma^2 \\
\mathbb{E}[y_i^2] &= 0^2\mathbb{P}(y_i = 0) + 1^2\mathbb{P}(y_i = 1) = p_i \\
\mathbb{E}[\widehat{p_i} y_i] &= \mathbb{E}[(p_i + \varepsilon_i) y_i] = \mathbb{E}[(p_i + \varepsilon_i)]\mathbb{E}[y_i] = p_i^2.
\end{aligned}$$

In the last calculus, we assume the independence between $\widehat{p_i}$ and y_i to conclude, which is obviously not guaranteed. However, this strong assumption gives a simple expression of σ^2 and leads to satisfying empirical results. We end up with:

$$\mathbb{E}[(\widehat{p_i} - y_i)^2] = p_i^2 + \sigma^2 + p_i - 2p_i^2 = p_i(1 - p_i) + \sigma^2 = V(y_i) + \sigma^2.$$

Rearranging the terms, we have the theoretical expression for σ^2: $\sigma^2 = \mathbb{E}[(\widehat{p_i} - y_i)^2] - V(y_i)$, which can be estimated by the following equation:

$$\widehat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(\widehat{p_i} - y_i)^2 - \frac{1}{N}\sum_{i=1}^{N}(y_i - \overline{y})^2,$$

where $\overline{y}$ is the mean of the labels y_i.

We also note that, alternatively, σ^2 can be cross-validated in order to optimize a criterion. This

technique has the advantage of giving a robust estimate of σ that will generalize well on the unseen data. We made several trials that suggests that using the sigmoid function as a denoiser, we can retrieve exactly the Platt calibration map by cross-validating σ^2.

3.3.4 Numerical approximation framework

A popular practice for solving constrained nonlinear optimization problems is to use the Sequential Quadratic Programming (SQP) approach. This method consists in approximating the solution of the original problem by a quadratic subproblem, solving it and using the solution to compute the next iterate. From the original system $(\mathcal{P})$ describing our problem, we can derive the Lagrangian of that function which is composed of the original objective function and the constraint. We define the Lagrangian of the SURE function as follows:

$$\mathcal{L}(\theta, \lambda) = SURE(\boldsymbol{\theta}) + \lambda C(\boldsymbol{\theta}),$$

where $\lambda \in \mathbb{R}$ is the Lagrange multiplier. We also define the quadratic approximation of the Lagrangian near $\boldsymbol{\theta}$ for the current iterates $\boldsymbol{\theta_k}$ employing a second-order Taylor expansion as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \lambda_k) \approx \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k) + \nabla_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)^T (\boldsymbol{\theta} - \boldsymbol{\theta_k}) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta_k})^T \nabla^2_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)(\boldsymbol{\theta} - \boldsymbol{\theta_k})$$

where $\nabla \mathcal{L}(.)$ is the gradient of the Lagrangian and $\nabla^2 \mathcal{L}(.)$ is the Hessian of the Lagrangian. In order to simplify the notations, we set $d_\theta = (\boldsymbol{\theta} - \boldsymbol{\theta_k})$. We also linearize the constraint as follows:

$$C(\boldsymbol{\theta}) \approx C(\boldsymbol{\theta_k}) + \nabla_{\boldsymbol{\theta_k}} C(\boldsymbol{\theta_k})^T d_\theta$$

where $\nabla C(.)$ is the gradient of the constraint. The challenge is now to define the quadratic subproblem such that it approximates well the original problem $(\mathcal{P})$. The SQP method consists in using directly the Lagrangian of the initial problem as objective function of the subproblem subject to the constraint (Bonnans et al., 2006; Gill et al., 1981; Wright & Nocedal, 1999). The quadratic subproblem is thus:

$$(\mathcal{P}_2) = \begin{cases} \min\limits_{d_{\boldsymbol{\theta}}} & \nabla_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)^T d_{\boldsymbol{\theta}} + \frac{1}{2} d_{\boldsymbol{\theta}}^T \nabla^2_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k) d_{\boldsymbol{\theta}} \\ \text{s.t.} & C(\boldsymbol{\theta_k}) + \nabla_{\boldsymbol{\theta_k}} C(\boldsymbol{\theta_k})^T d_\theta = 0. \end{cases}$$

The advantage of this new formulation $(\mathcal{P}_2)$ of the initial problem $(\mathcal{P})$ is that it takes into account the nonlinearity of the constraint in the Lagrangian while at the same time the constraint is linearized. We can now write the KKT conditions for the quadratic subproblem. We denote the optimal multiplier of $(\mathcal{P}_2)$ by d_λ (that is d_λ is the multiplier associated to the Lagrangian formed by the quadratic objective function and the constraint approximation) (Bonnans et al., 2006). Then we apply the first order conditions to this new function which gives the following system:

$$\begin{pmatrix} \nabla^2_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k) & \nabla_{\boldsymbol{\theta_k}} C(\boldsymbol{\theta_k}) \\ \nabla_{\boldsymbol{\theta_k}} C(\boldsymbol{\theta_k})^T & 0 \end{pmatrix} \begin{pmatrix} d_\theta \\ d_\lambda \end{pmatrix} = - \begin{pmatrix} \nabla_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k) \\ C(\boldsymbol{\theta_k}) \end{pmatrix}, \tag{4}$$

where $\nabla^2_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)$ is supposed to be positive definite on the tangent space of the constraints, and $\nabla_{\boldsymbol{\theta_k}} C(\boldsymbol{\theta_k})$ has full row rank. Solving this system for each iteration k allows us to realize simultaneously our two-sided objective: minimizing the objective function and satisfying the constraint which is important to ensure the convergence of the algorithm. The most straightforward approach for finding the solution (d_θ, d_λ) of the system (4) is to invert the first matrix also known as KKT matrix. We finally end up with the following Newton step of the parameters for the next iterate:

$$\begin{cases} \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + d_\theta \\ \lambda_{k+1} = \lambda_k + d_\lambda. \end{cases}$$

The explicit form of the gradient of SURE and of the constraint is given in appendix A and B. The Hessian of SURE is given in appendix C. Finally, for completeness of this framework, we provide the explicit calculus of the derivatives for the sigmoid and Kumaraswamy cumulative distribution functions in appendix D and E.

Any practical implementation of Newton's algorithm requires safeguards to ensure the convergence of the algorithm. To ensure that the Newton-SQP method produces iterates that converge to the optimum, it is recommended to initialize the starting values $(\boldsymbol{\theta_0}, \lambda_0)$ so that they are feasible i.e. starting values for which the constraint is saturated. In this framework, we initialize the starting values of $\boldsymbol{\theta}$ using the Newton-Raphson algorithm directly applied to the constraint. We then use these feasible points to find a good starting value for λ_0. Indeed, if $\boldsymbol{\theta_0}$ is close enough to the optimum $\boldsymbol{\theta_\star}$, then we can infer from the first-order conditions applied to the Lagrangian that the optimal value $\lambda_\star$ is:

$$\begin{aligned} &\nabla_{\boldsymbol{\theta_\star}} SURE(\boldsymbol{\theta_\star}) + \lambda_\star \nabla_{\boldsymbol{\theta_\star}} C(\boldsymbol{\theta_\star}) = 0 \\ &\Leftrightarrow \lambda_\star = - \left[\nabla_{\boldsymbol{\theta_\star}} C(\boldsymbol{\theta_\star})^T \nabla_{\boldsymbol{\theta_\star}} C(\boldsymbol{\theta_\star}) \right]^{-1} \nabla_{\boldsymbol{\theta_\star}} C(\boldsymbol{\theta_\star})^T \nabla_{\boldsymbol{\theta_\star}} SURE(\boldsymbol{\theta_\star}). \end{aligned}$$

Hence, a good starting value for λ_0 is $\lambda_0 = - \left[\nabla_{\boldsymbol{\theta_0}} C(\boldsymbol{\theta_0})^T \nabla_{\boldsymbol{\theta_0}} C(\boldsymbol{\theta_0}) \right]^{-1} \nabla_{\boldsymbol{\theta_0}} C(\boldsymbol{\theta_0})^T \nabla_{\boldsymbol{\theta_0}} SURE(\boldsymbol{\theta_0})$. Another important safeguard concerns the Hessian matrix. A sufficient condition for the Newton-Lagrange method to find a local minimizer is that the Hessian of Lagrangian is positive definite. Indeed, if this assumption is satisfied, then the Newton step is a descent direction and the algorithm converges to the optimum. If the Hessian is not positive definite a popular method consists in defining $H = \nabla^2_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k) + E_k$ where H is an approximation of the Hessian and E_k is a general diagonal matrix being equal to zero if $\nabla^2_{\boldsymbol{\theta_k}} \mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)$ is positive definite and chosen to make H positive definite if not. Different strategies can be used to find the diagonal matrix E_k. In this framework we used (Gill & Murray, 1974) approach which is based on a modified Cholesky factorization. Finally, we stress that one can improve the method by using a line-search to ensure the stability of the algorithm (Bonnans et al., 2006). This common practice consists in applying

an adaptative step length α to the descent direction instead of the pure Newton step where α is implicitly set to 1. The update formula of the parameters now becomes $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k d_\theta$. In this article, we haven't implemented this technique since our algorithm is fast and give precise enough results.

The pseudo-code described in Algorithm 1 summarizes the procedure of the proposed SURE calibration.

Algorithm 1 Newton's method for equality constraint

Initialization: K=20, θ_0 = (random, random), tol = 1e-4
$\widehat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(\widehat{p}_i - y_i)^2 - \frac{1}{N}\sum_{i=1}^{N}(y_i - \overline{y})^2$
Find a feasible point $\boldsymbol{\theta_0}$ using Newton-Raphson algorithm
for $k = 0$ to K **do**
  if $||\nabla_{\boldsymbol{\theta_k}}\mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)|| \leq tol$ **then**
    break;
  end if
  if all the eigen values of $\nabla^2_{\boldsymbol{\theta_k}}\mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)$ are non-negative **then**
    $H = \nabla^2_{\boldsymbol{\theta_k}}\mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)$
  else
    $H = \nabla^2_{\boldsymbol{\theta_k}}\mathcal{L}(\boldsymbol{\theta_k}, \lambda_k) + E_k$
  end if
  Solve the linear system (4) replacing $\nabla^2_{\boldsymbol{\theta_k}}\mathcal{L}(\boldsymbol{\theta_k}, \lambda_k)$ by H
  $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + d_\theta$
  $\lambda_{k+1} = \lambda_k + d_\lambda$
end for

3.4 Calibration evaluation

Measuring the calibration error is important to get an estimate of the model uncertainty. In this study, we want to compare the calibrated probabilities of the Platt method to our SURE method. Most of the time, calibration models are evaluated using a reliability diagram that compares the average predicted probability to the observed proportion of events in each quantile. We denote the average predicted probability of the sample B_m as APP(B_m) and the corresponding observed event frequency as OEF(B_m):

$$\text{OEF}(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m} \mathbb{1}(y_i = 1), \qquad \text{APP}(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m} \widehat{p}_i,$$

where B_m is the set of indices of samples whose predicted probabilities fall into the interval $I_m = (\frac{m-1}{M}, \frac{m}{M}]$, for m $\in [\![1, \ldots, M]\!]$ and M is the total number of bins. Note that these two measures are similar to the accuracy and the confidence introduced in the multiclass context (Guo et al., 2017).

The key idea of this representation is to visualize the quality of prediction for each quantile of risk. First, predicted probabilities and their corresponding labels are sorted in ascending order. Then, the sample is divided into M bins. Finally, the OEF and APP are computed on each bin and reported on the diagram. If the OEF equals the APP (i.e. the points are on the diagonal line) the model is considered to be perfectly calibrated. However, this representation can be misleading since there exists a trade-off between the number of bins chosen and the APP estimated. Indeed, an insufficient number of samples in some bins can result in an inefficient estimation. Even with a large number of observations, it is not always obvious to assert the quality of the calibration. Figure 1 illustrates this point by showing a reliability diagram with different values of M. The left plot shows the reliability diagram with 10 bins while the right plot uses 50 bins.

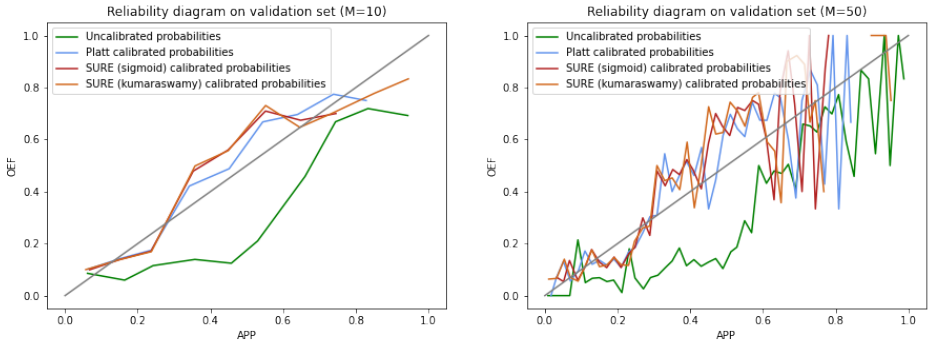

Figure 1: Example of reliability diagrams using the predicted probabilities of the logistic regression on the Taiwan credit dataset.

Both plots are using the same data. The left plot would make us believe that the calibration is almost perfect for the Platt method, whereas it is not so obvious in the right plot.

Synthetic measures of the calibration based on the reliability diagram also allow judging the quality of calibrated probabilities. Recently, the Expected Calibration Error (ECE) has been proposed to summarize the calibration benefits (Naeini et al., 2015). This measure is computed as the average of the difference between the OEF and the APP over the bins:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\text{OEF}(B_m) - \text{APP}(B_m)|,$$

where N is the total number of data points. Alternatively, one may also want to measure the worst possible gap between the ideal calibration and the estimated confidence which is possible through the Maximum Calibration Error (MCE):

$$\text{MCE} = \max_{m \in 1,\ldots,M} |\text{OEF}(B_m) - \text{APP}(B_m)|,$$

(Naeini et al., 2015). ECE and MCE suffer from the same limitations as the reliability diagram (Nixon et al., 2019). For instance, the distribution of predicted probabilities is often left-

skewed. This has the effect of putting more weight on certain bins densely populated which affects the ECE. Therefore, choosing the number of bins amounts to a bias-variance tradeoff. Indeed, when M is large, the number of points in each bin becomes small leading to an increase in the variance while the bias tends to decrease.

Because of the above limitations, we decided to use another classical way to evaluate the calibration. It consists in using the full distribution of the calibrated probabilities and comparing them to the true output. This is the purpose of the Binary Cross-Entropy (BCE) and the Brier Score (BS) (Brier, 1950). The BCE and the BS have been computed as follows:

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\widetilde{p}_i) + (1-y_i)\log\left(1-\widetilde{p}_i\right),$$

$$\text{BS} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widetilde{p}_i)^2.$$

As we can see, the BS is similar to the MSE often used to measure the error in regression problems. In this particular setting, it can be interpreted as the mean prediction error. The BCE comes from the specification of the likelihood in binary classification problems as stated in section 3.2. It is also used as a loss function for training binary classifiers. Additionally, we compute the Mean Default Rate (MDR) for each method:

$$\text{MDR} = \left(\frac{1}{N}\sum_{i=1}^{N}\widetilde{p}_i\right) \times 100.$$

This last measure is useful to control the resulting bias of the calibrated probabilities. Indeed, a perfectly calibrated classifier should produce probabilities that have the same mean default rate as the true observed default rate.

4 Empirical results

In this section, we report the results obtained for the logistic regression and the neural network on the different sets of features. The models are optimized by minimizing the BCE and the optimal threshold $\tau^\star$ is chosen to maximize the F1 score. Both models are evaluated using the F1 score and we also report the precision and the recall to diagnose the possible model weaknesses and the ROC-AUC and PR-AUC to spot the effect of imbalanced learning. We then move on to the calibration results. We compare the results of the Platt method to our SURE method.

4.1 Comparison of models

For both the logistic regression and the feed forward neural network, we use the same learning and evaluation procedure. First, the model is trained on the learning set and at the same time evaluated on the validation set. We implement the early stopping rule such that the model stops learning when validation performance starts to diverge according to the performance on the training set. Once the stopping rule is reached, the model's parameters are saved (Prechelt, 1998). The remaining model is then used on the test set to assess its performance on unseen data.

As we can see in both Table 2 and Table 3, it seems that the two models do not suffer from overfitting. Indeed, for all different sets of features, the training and testing results are very close and even better when looking at the logistic regression testing results in Table 2. Thus, we can infer that our early stopping procedure has been successful to prevent overfitting and both models can generalize well with the data provided. As stated in Section 2.3, the AUC-ROC is not a good choice for assessing the performance of our models. Indeed, since we are in an imbalanced class problem, it produces good results for both models because it does not focus only on the positive class. The AUC-PR and F1 score are more reliable indicators of the model performance for predicting consumer default. On average, the observed gap between the AUC-ROC and the F1 score is 0.2, which is quite high. On the feature side, an interesting thing to remark is that the set of dynamic features provides better results than the static ones. Indeed, the testing F1 scores (i.e. results obtained on unseen data) are improved by around 0.10 if we consider the logistic regression and by around 0.15 considering the feed forward neural network. Thus we can infer that dynamic data contains more informative features for customer default prediction.

	Static features			Dynamic features			All features		
	Training	Validation	Testing	Training	Validation	Testing	Training	Validation	Testing
F1	0.40	0.40	0.40	0.51	0.53	0.53	0.51	0.52	0.53
Recall	0.70	0.70	0.71	0.48	0.51	0.52	0.48	0.51	0.51
Precision	0.27	0.28	0.28	0.54	0.54	0.55	0.54	0.54	0.55
AUC-ROC	0.62	0.63	0.63	0.71	0.73	0.73	0.72	0.74	0.73
AUC-PR	0.30	0.32	0.31	0.49	0.50	0.51	0.50	0.51	0.51
Loss	0.23	0.23	0.23	0.21	0.21	0.21	0.21	0.21	0.21

Table 2: Logistic regression performance on the different sets.

We can also notice that the feed forward neural net seems to be unstable on static data since its F1 score, AUC-ROC, AUC-PR values are worse than for the logistic regression. We suspect that the difference is due to both the set of features and the model complexity. Indeed, on one hand, the feed forward neural network learns complex data interactions on the static features which appear to be uninformative for the given task. On the other hand, the logistic regression can be seen as a feed forward neural network without hidden layers, thus leading to a simpler model with fewer parameters. The resulting performance of the logistic regression is better because this model doesn't learn these complex interactions. This result can be connected to the adversarial attack

	Static features			Dynamic features			All features		
	Training	Validation	Testing	Training	Validation	Testing	Training	Validation	Testing
F1	0.40	0.38	0.39	0.56	0.53	0.54	0.56	0.52	0.55
Recall	0.82	0.81	0.81	0.62	0.53	0.55	0.61	0.51	0.54
Precision	0.27	0.25	0.25	0.53	0.53	0.53	0.53	0.53	0.55
AUC-ROC	0.59	0.60	0.60	0.77	0.77	0.77	0.78	0.76	0.77
AUC-PR	0.28	0.30	0.28	0.54	0.54	0.53	0.54	0.53	0.53
Loss	0.23	0.23	0.23	0.20	0.20	0.20	0.19	0.20	0.20

Table 3: Neural network performance on the different sets.

notion which exists in computer vision (Goodfellow et al., 2014; Szegedy et al., 2014). The general idea is to fool the model by introducing a little perturbation noise into the input data which doesn't change the image perception for a human. This perturbation causes the deep learning model to produce false predictions.

When we investigate the results for the dynamic and the whole feature sets, we can observe that the feed forward neural network reaches better testing results than the logistic regression. This improvement comes from the model architecture. With the hidden layers being fully connected to each other, the network learns deeper representations of the inputs by automatically creating feature interactions. These representations allow the network to learn more complex patterns than the logistic regression, which explains why the results are improved. Even if there is an undeniable improvement of the performance, we can notice that results are very close between Table 2 and Table 3. For example, the F1 score is superior by only 0.02 for the neural network which is not high. Here, we want to stress the fact that appropriate measures should be used to evaluate the model depending on the use case. For the same dataset, several studies have shown that a neural network is a more accurate model while our results suggest only little improvement (Imtiaz & Brimicombe, 2017; Yeh & Lien, 2009). However, other studies in the field of credit scoring tend to confirm our results. Indeed, other authors found that deep neural networks do not outperform their shallower counterparts while they are computationally considerably more expensive to build and train (Alonso & Carbó, 2020; Gunnarsson et al., 2021; Lessmann et al., 2015).

4.2 Evaluation of the calibration

We present the results obtained by the different calibration techniques. We used the same procedure for training and testing as standard papers on calibration. First, we fit the model on the predicted probabilities of the validation set and use the learned parameters to calibrate the predicted probabilities of the test set (Guo et al., 2017; Pan et al., 2020; Platt, 1999). The measures realized on the test set are thus considered unbiased estimates of the calibration error. Beyond the SURE calibration technique, we also try a form of model stacking which consists in combining two models in order to improve the overall accuracy. The rationale behind this approach is that each model learns its own nonlinearities such that we can benefit from combining models. Here

we combine the Platt method to our SURE approach in two ways. We build a first model using Platt's model output probabilities as an input of the SURE model. The second model is the exact reverse: we feed the Platt model by the SURE output probabilities. The first observation that can be done about the results gathered in Tables 4 and 5 is that uncalibrated probabilities do not reflect the true probabilities. Indeed, the mean probability goes from 45.26% for the Neural network to 47.04% for the logistic regression whereas the true empirical default rate is 22.13%. This emphasizes the need for probability calibration.

	Mean default rate		BCE		BS	
	Validation	Testing	Validation	Testing	Validation	Testing
Uncalibrated	0.4673	0.4704	0.610	0.613	0.210	0.211
Platt	0.2213	0.2244	0.460	0.459	0.144	0.143
SURE (sigmoid)	0.2213	0.2240	0.461	0.461	0.145	0.144
SURE (Kumaraswamy)	0.2213	0.2241	0.465	0.465	0.146	0.145
Platt + SURE (sigmoid)	0.2213	0.2243	0.455	0.451	0.143	0.141
SURE (sigmoid) + Platt	0.2213	0.2248	0.454	0.451	0.142	0.140
Platt + SURE (Kumaraswamy)	0.2213	0.2244	0.460	0.459	0.144	0.143
SURE (Kumaraswamy) + Platt	0.2213	0.2247	0.456	0.453	0.143	0.141

Table 4: Logistic regression calibration.

	Mean default rate		BCE		BS	
	Validation	Testing	Validation	Testing	Validation	Testing
Uncalibrated	0.4526	0.4548	0.580	0.584	0.194	0.194
Platt	0.2213	0.2237	0.437	0.436	0.137	0.136
SURE (sigmoid)	0.2213	0.2237	0.437	0.436	0.137	0.136
SURE (Kumaraswamy)	0.2213	0.2240	0.439	0.439	0.138	0.137
Platt + SURE (sigmoid)	0.2213	0.2243	0.438	0.436	0.137	0.136
SURE (sigmoid) + Platt	0.2213	0.2245	0.438	0.436	0.137	0.136
Platt + SURE (Kumaraswamy)	0.2213	0.2236	0.437	0.436	0.137	0.136
SURE (Kumaraswamy) + Platt	0.2214	0.2247	0.438	0.437	0.137	0.137

Table 5: Neural network calibration.

Regarding the performance of the various calibration methods, we report that the Platt and SURE methods lead to a sharp improvement of uncalibrated probabilities. When we investigate both methods separately, our SURE method appears to be as good as Platt scaling. The good performance of the SURE method remains valid if we replace the sigmoid function by the Kumaraswamy one. Finally, when we combine both methods trough stacking, the results stay stable.

5 Conclusion

In this paper, we have investigated classification models for determining the probability that a consumer will default on its credit card. The obtained results give us some guidelines for this kind of prediction problem. First, the class imbalance should be identified in order to choose an appropriate measure of performance. In the case of a severe class imbalance, it is recommended to use a measure that relies on positive class detection, such as the F1 score, to have an unbiased estimate of the model's true performance. Secondly, temporal features, if available, must be added to the model as they can significantly improve its learning ability and performance. Depending on the frequency of the data, it could be interesting to perform pre-processing steps in order to extract useful information such as descriptive statistics of that series. This has been left for further studies. On the contrary, one should take care of uninformative features which can lead to false data representations and decreased performance when learning with a complex model such as a neural network. Thirdly, we stress out the fact that real-world applications should output predicted probabilities that reflect the true unobservable probabilities in order to diminish uncertainty. Our results show that it can be done accurately and efficiently by using our SURE calibration method.

References

Abadie, A., & Kasy, M. (2019). Choosing among regularized estimators in empirical economics. *The review of economics and statistics*, *101*(5), 743–762.

Albanesi, S., & Vamossy, F. D. (2019). Predicting consumer default: A deep learning approach. *arXiv preprint*. https://doi.org/10.2139/ssrn.3445152

Alonso, A., & Carbó, J. (2020). *Machine learning in credit risk: Measuring the dilemma between prediction and supervisory cost* (Working Paper No. 2032). Banco de Espana.

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The journal of finance*, *23*(4), 589–609.

Anderson, R. (2007). *The credit scoring toolkit: Theory and practice for retail credit risk management and decision automation*. Oxford university press.

Antorán, J., Urquhart Allingham, J., & Hernández-Lobato, J. M. (2020). Depth uncertainty in neural networks. *arXiv preprint*, arXiv 2006.08437.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society*, *54*(6), 627–635.

Bella, A., Ferri, C., Hernández-Orallo, J., & Ramírez-Quintana, M. J. (2009). Similarity-binning averaging: A generalisation of binning calibration., In *In international conference on intelligent data engineering and automated learning*.

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large-scale kernel machines*, *34*(5), 1–41.

Bennett, P. N. (2003). Using asymmetric distributions to improve text classifier probability estimates, In *Proceedings of the 26th annual international acm sigir conference on research and development in information retrieval*.

Bequé, A., Coussement, K., Gayler, R., & Lessmann, S. (2017). Approaches for credit scorecard calibration: An empirical analysis. *Knowledge-based systems*, *134*, 213–227.

Blöchlinger, A., & Leippold, M. (2006). Economic benefit of powerful credit scoring. *Journal of banking and finance*, *30*(3), 851–873.

Bonnans, J. F., Gilbert, J. C., Lemaréchal, C., & Sagastizábal, C. A. (2006). *Numerical optimization: Theoretical and practical aspects*. Springer Science; Business Media.

Boyd, K., Eng, K. H., & Page, C. D. (2013). Area under the precision-recall curve: Point estimates and confidence intervals, In *Joint european conference on machine learning and knowledge discovery in databases*.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, *78*(1), 1–3.

Bücker, M., Szepannek, G., Gosiewska, A., & Biecek, P. (2022). Transparency, auditability, and explainability of machine learning models in credit scoring. *Journal of the Operational Research Society*, *73*(1), 70–90.

Cearns, M., Hahn, T., Clark, S., & Baune, B. T. (2019). Machine learning probability calibration for high-risk clinical decision-making. *Australian and New Zealand journal of psychiatry*, *54*, 123–126.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Chen, W., Sahiner, B., Samuelson, F., Pezeshk, A., & Petrick, N. (2018). Calibration of medical diagnostic classifier scores to the probability of disease. *Statistical methods in medical research*, *27*(5), 1394–1409.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., & Dahl, G. E. (2019). On empirical comparisons of optimizers for deep learning. *arXiv preprint*, arXiv 1910.05446.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves, In *Proceedings of the 23rd international conference on machine learning*.

Donoho, D. L., & Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American statistical association*, *90*(432), 1200–1244.

Dumitrescu, E. I., Hué, S., & Hurlin, C. (2020). Machine learning or econometrics for credit scoring: Let's get the best of both worlds. *Expert systems with applications*.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The annals of statistics*, *32*(2), 407–499.

Feng, D., Rosenbaum, L., Glaeser, C., Timm, F., & Dietmayer, K. (2019). Can we trust you? on calibration of a probabilistic object detector for autonomous driving. *arXiv preprint*, arXiv 1909.12358.

Finlay, S. (2011). Multiple classifier architectures and their application to credit risk assessment. *European journal of operational research*, *210*(2), 368–378.

Garcin, M., & Guégan, D. (2016). Wavelet shrinkage of a noisy dynamical system with non-linear noise impact. *Physica D: nonlinear phenomena*, *325*, 126–145.

Gill, P. E., & Murray, W. (1974). Newton-type methods for unconstrained and linearly constrained optimization. *Mathematical Programming*, *7*(1), 311–350.

Gill, P. E., Murray, W., & Wright, M. H. (1981). *Practical optimization.* Society for Industrial; Applied Mathematics.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks, In *Proceedings of the 13th international conference on artificial intelligence and statistics.*

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint*, arXiv 1412.6572.

Gunnarsson, B. R., Vanden Broucke, S., Baesens, B., Óskarsdóttir, M., & Lemahieu, W. (2021). Deep learning for credit scoring: Do or don't? *European Journal of Operational Research*, *295*(1), 292–305.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks, In *Proceedings of the 34th international conference on machine learning.*

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE transactions on knowledge and data engineering*, *21*(9), 1263–1284.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, In *Proceedings of the ieee international conference on computer vision.*

Hjelkrem, L. O., & Lange, P. E. D. (2023). Explaining deep learning models for credit scoring with shap: A case study using open banking data. *Journal of Risk and Financial Management*, *16*(4), 221.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, *4*(2), 251–257.

Huang, C. L., Chen, M. C., & Wang, C. J. (2007). Credit scoring with a data mining approach based on support vector machines. *Expert systems with applications*, *33*(4), 847–856.

Imtiaz, S., & Brimicombe, A. J. (2017). A better comparison summary of credit scoring classification. *International journal of advanced computer science and applications*, *8*(7), 1–4.

Jiang, C., Lu, W., Wang, Z., & Ding, Y. (2023). Benchmarking state-of-the-art imbalanced data learning approaches for credit scoring. *Expert systems with applications*, *213*, 118878.

Jiang, X., Osl, M., Kim, J., & Ohno-Machado, L. (2012). Calibrating predictive model estimates to support personalized medicine. *Journal of the American medical informatics association*, *19*(2), 263–274.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization, In *Proceedings of international conference on learning representations.*

Kovács, G. (2019). An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, *83*, 105662.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097–1105.

Kull, M., Silva Filho, T., & Flach, P. (2017). Beta calibration: A well-founded and easily implemented improvement on logistic calibration for binary classifiers. *Artificial intelligence and statistics*, 623–631.

Kumaraswamy, P. (1980). A generalized probability density function for double-bounded random processes. *Journal of hydrology*, *46*(1-2), 79–88.

Kvamme, H., Sellereite, N., Aas, K., & Sjursen, S. (2018). Predicting mortgage default using convolutional neural networks. *Expert systems with applications*, *102*, 207–217.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition, In *Proceedings of the ieee*.

Lee, T. S., Chiu, C. C., Lu, C. J., & Chen, I. F. (2002). Credit scoring using the hybrid neural discriminant technique. *Expert systems with applications*, *23*(3), 245–254.

Lessmann, S., Baesens, B., Seow, H., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring. *European Journal of Operational Research*, *247*, 124–136.

Li, Y., & Zhu, J. (2008). L1-norm quantile regression. *Journal of computational and graphical statistics*, *17*(1), 163–185.

Lipton, Z. C., Elkan, C., & Naryanaswamy, B. (2014). Optimal thresholding of classifiers to maximize f1 measure. *Machine learning and knowledge discovery in databases*, *8725*, 225–239.

Metzler, C. A., Mousavi, A., Heckel, R., & Baraniuk, R. G. (2020). Unsupervised learning with stein's unbiased risk estimator. *arXiv preprint*, arXiv 1805.10531.

Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning, In *Proceedings of the aaai conference on artificial intelligence*.

Nixon, J., Dusenberry, M., Zhang, L., Jerfel, G., & Tran, D. (2019). Measuring calibration in deep learning. *arXiv preprint*, arXiv 1904.01685.

Paleologo, G., Elisseeff, A., & Antonini, G. (2010). Subagging for credit scoring models. *European journal of operational research*, *201*(2), 490–499.

Pan, F., Ao, X., Tang, P., Lu, M., Liu, D., Xiao, L., & He, Q. (2020). Field-aware calibration: A simple and empirically strong method for reliable probabilistic predictions, In *Proceedings of the web conference 2020*.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, *10*(3), 61–74.

Prechelt, L. (1998). Automatic early stopping using cross validation: Quantifying the criteria. *Neural networks*, *11*(4), 761–767.

Provost, F. (2000). Machine learning from imbalanced data sets 101, In *Proceedings of the aaai 2000 workshop on imbalanced data sets*.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, *10*, 3.

Sheng, V. S., & Ling, C. X. (2006). Thresholding for making classifiers cost-sensitive, In *Proceedings of the 21st national conference on artificial intelligence*.

Soltanayev, S., & Chun, S. Y. (2018). Training and refining deep learning based denoisers without ground truth data. *arXiv preprint*, arXiv 1803.01314.

Steenackers, A., & Goovaerts, M. (1989). A credit scoring model for personal loans. *Insurance: mathematics and economics*, *8*(1), 31–34.

Stein, C. M. (1981). Estimation of the mean of a multivariate normal distribution. *The annals of statistics*, *9*(6), 1135–1151.

Sun, Y., Wong, A. K., & Kamel, M. S. (2009). Classification of imbalanced data: A review. *Internatioknal journal of pattern recognition and artificial intelligence*, *23*(4), 687–719.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *arXiv preprint*, arXiv 1312.6199.

Tai, L. Q., & Huyen, G. T. (2019). Deep learning techniques for credit scoring. *Journal of economics, business and management*, *7*(3), 93–96.

Tasche, D. (2009). Estimating discriminatory power and pd curves when the number of defaults is small. *arXiv preprint*, arXiv 0905.3928.

Tibshirani, R. J., & Taylor, J. (2011). The solution path of the generalized lasso. *The annals of statistics*, *39*(3), 1335–1371.

Tygert, M., Bruna, J., Chintala, S., LeCun, Y., Piantino, S., & Szlam, A. (2016). A mathematical motivation for complex-valued convolutional networks. *Neural computation*, *28*(5), 815–825.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*. https://doi.org/10.1155/2018/7068349

Wang, G., Hao, J., Ma, J., & Jiang, H. (2011). A comparative assessment of ensemble learning for credit scoring. *Expert systems with applications*, *38*(1), 223–230.

Wright, S., & Nocedal, J. (1999). *Numerical optimization*. Springer Science.

Xie, S., & Tu, Z. (2015). Holistically-nested edge detection, In *Proceedings of the ieee international conference on computer vision*.

Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, *36*(2), 2473–2480.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown, In *Proceedings of the 7th acm sigkdd international conference on knowledge discovery and data mining*.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates, In *Proceedings of the 8th acm sigkdd international conference on knowledge discovery and data mining*.

Zhang, J., & Yang, Y. (2004). Probabilistic score estimation with piecewise logistic regression, In *Proceedings of the 21st international conference on machine learning*.

Zhu, B., Yang, W., Wang, H., & Yuan, Y. (2018). A hybrid deep learning model for consumer credit scoring, In *2018 international conference on artificial intelligence and big data*.

Zhussip, M., Soltanayev, S., & Young Chun, S. (2019). Extending stein's unbiased risk estimator to train deep denoisers with correlated pairs of noisy images. *arXiv preprint*, arXiv 1902.02452.

Zou, H., Hastie, T., & Tibshirani, R. (2007). On the "degrees of freedom" of the lasso. *The annals of statistics*, *35*(5), 2173–2192.

Zou, Q., Xie, S., Lin, Z., Wu, M., & Ju, Y. (2016). Finding the best classification threshold in imbalanced classification. *Big data research*, *5*, 2–8.

A Gradient of SURE

$$\nabla_{\boldsymbol{\theta}} SURE(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial SURE}{\partial \theta_1} \\ \frac{\partial SURE}{\partial \theta_2} \end{pmatrix}$$

$$\begin{cases} \dfrac{\partial SURE}{\partial \theta_1} = 2\sum_{i=1}^{N}(G_{\boldsymbol{\theta}}(\widehat{p_i}) - \widehat{p_i})\dfrac{\partial G_\theta(\widehat{p_i})}{\partial \theta_1} + 2\sigma^2 \sum_{i=1}^{N} \dfrac{\partial^2 G_\theta(\widehat{p_i})}{\partial \widehat{p_i} \partial \theta_1} \\ \dfrac{\partial SURE}{\partial \theta_2} = 2\sum_{i=1}^{N}(G_{\boldsymbol{\theta}}(\widehat{p_i}) - \widehat{p_i})\dfrac{\partial G_\theta(\widehat{p_i})}{\partial \theta_2} + 2\sigma^2 \sum_{i=1}^{N} \dfrac{\partial^2 G_\theta(\widehat{p_i})}{\partial \widehat{p_i} \partial \theta_2} \end{cases}$$

B Gradient of the constraint

$$\nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial C}{\partial \theta_1} \\ \frac{\partial C}{\partial \theta_2} \end{pmatrix}$$

$$\begin{cases} \dfrac{\partial C}{\partial \theta_1} = \dfrac{1}{N}\sum_{i=1}^{N} \dfrac{G_\theta(\widehat{p_i})}{\partial \theta_1} \\ \dfrac{\partial C}{\partial \theta_2} = \dfrac{1}{N}\sum_{i=1}^{N} \dfrac{G_\theta(\widehat{p_i})}{\partial \theta_2} \end{cases}$$

C Hessian of SURE

$$\nabla^2_{\boldsymbol{\theta}} SURE(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial^2 SURE}{\partial \theta_1^2} & \frac{\partial^2 SURE}{\partial \theta_1 \partial \theta_2} \\ \frac{\partial^2 SURE}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 SURE}{\partial \theta_2^2} \end{pmatrix}$$

$$
\begin{cases}
\dfrac{\partial^2 SURE}{\partial \theta_1^2} = 2\displaystyle\sum_{i=1}^{N}\left((G_\theta(\widehat{p_i})-\widehat{p_i})\dfrac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_1^2}+\left(\dfrac{\partial G_\theta(\widehat{p_i})}{\partial \theta_1}\right)^2\right)+2\sigma^2\displaystyle\sum_{i=1}^{N}\dfrac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_1^2} \\
\dfrac{\partial^2 SURE}{\partial \theta_2^2} = 2\displaystyle\sum_{i=1}^{N}\left((G_\theta(\widehat{p_i})-\widehat{p_i})\dfrac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_2^2}+\left(\dfrac{\partial G_\theta(\widehat{p_i})}{\partial \theta_2}\right)^2\right)+2\sigma^2\displaystyle\sum_{i=1}^{N}\dfrac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_2^2} \\
\dfrac{\partial^2 SURE}{\partial \theta_2\partial \theta_1} = \dfrac{\partial^2 SURE}{\partial \theta_1\partial \theta_2} = 2\displaystyle\sum_{i=1}^{N}\left((G_\theta(\widehat{p_i})-\widehat{p_i})\dfrac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_1\partial \theta_2}+\dfrac{\partial G_\theta(\widehat{p_i})}{\partial \theta_1}\dfrac{\partial G_\theta(\widehat{p_i})}{\partial \theta_2}\right)+2\sigma^2\displaystyle\sum_{i=1}^{N}\dfrac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_1\partial \theta 2}
\end{cases}
$$

D Derivatives of the sigmoid function

$$
\begin{aligned}
G_\theta(\widehat{p_i}) &= \frac{1}{1+e^{-(\theta_1\widehat{p_i}+\theta_2)}} \\
\frac{\partial G_\theta(\widehat{p_i})}{\partial \widehat{p_i}} &= \theta_1 G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i})) \\
\frac{\partial G_\theta(\widehat{p_i})}{\partial \theta_1} &= \widehat{p_i} G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i})) \\
\frac{\partial G_\theta(\widehat{p_i})}{\partial \theta_2} &= G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i})) \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_1} &= G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i}))+\theta_1\widehat{p_i}G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i}))(1-2G_\theta(\widehat{p_i})) \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_2} &= \theta_1 G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i}))(1-2G_\theta(\widehat{p_i})) \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_1^2} &= \widehat{p_i}^2 G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i}))(1-2G_\theta(\widehat{p_i})) \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_2^2} &= G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i}))(1-2G_\theta(\widehat{p_i})) \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_1\partial \theta_2} &= \frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_2\partial \theta_1} = \widehat{p_i} G_\theta(\widehat{p_i})(1-G_\theta(\widehat{p_i}))(1-2G_\theta(\widehat{p_i})) \\
\frac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_1^2} &= \frac{\partial G_\theta(\widehat{p_i})}{\partial \theta_1}(1-2G_\theta(\widehat{p_i}))+\widehat{p_i}\left(\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_2^2}+\frac{\partial G_\theta(\widehat{p_i})}{\partial \theta_1}\theta_1\left[(1-G_\theta(\widehat{p_i}))(1-2G_\theta(\widehat{p_i}))+G_\theta(\widehat{p_i})(4G_\theta(\widehat{p_i})-3)\right]\right) \\
\frac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_2^2} &= \frac{\partial G_\theta(\widehat{p_i})}{\partial \theta_2}\theta_1\left((1-G_\theta(\widehat{p_i}))(1-2G_\theta(\widehat{p_i}))+G_\theta(\widehat{p_i})(4G_\theta(\widehat{p_i})-3)\right) \\
\frac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_1\partial \theta_2} &= \frac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\partial \theta_2\partial \theta_1} = \frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_2^2}+\theta_1\frac{\partial G_\theta(\widehat{p_i})}{\partial \theta_1}\left((1-G_\theta(\widehat{p_i}))(1-2G_\theta(\widehat{p_i}))+G_\theta(\widehat{p_i})(4G_\theta(\widehat{p_i})-3)\right)
\end{aligned}
$$

E Derivatives of the Kumaraswamy cumulative distribution

$$
\begin{aligned}
G_\theta(\widehat{p_i}) &= 1-(1-\widehat{p_i}^{\theta_1})^{\theta_2} \\
\frac{\partial G_\theta(\widehat{p_i})}{\partial \widehat{p_i}} &= \theta_1\theta_2\widehat{p_i}^{\theta_1-1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-1} \\
\frac{\partial G_\theta(\widehat{p_i})}{\partial \theta_1} &= \theta_2\log(\widehat{p_i})\widehat{p_i}^{\theta_1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-1} \\
\frac{\partial G_\theta(\widehat{p_i})}{\partial \theta_2} &= -\log(1-\widehat{p_i}^{\theta_1})(1-\widehat{p_i}^{\theta_1})^{\theta_2} \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\theta_1} &= \theta_2\left(\widehat{p_i}^{\theta_1-1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-1}(1+\theta_1\log(\widehat{p_i}))-\theta_1\widehat{p_i}^{2\theta_1-1}(\theta_2-1)(1-\widehat{p_i}^{\theta_1})^{\theta_2-2}\log(\widehat{p_i})\right) \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\theta_2} &= \theta_1\widehat{p_i}^{\theta_1-1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-1}\left(1+\theta_2\log(1-\widehat{p_i}^{\theta_1})\right) \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_1^2} &= \theta_2\left(\log(\widehat{p_i})\right)^2\widehat{p_i}^{\theta_1}\left((1-\widehat{p_i}^{\theta_1})^{\theta_2-1}-(\theta_2-1)(1-\widehat{p_i}^{\theta_1})^{\theta_2-2}\widehat{p_i}^{\theta_1}\right) \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_2^2} &= -\left(\log(1-\widehat{p_i})\right)^2(1-\widehat{p_i}^{\theta_1})^{\theta_2} \\
\frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_1\theta_2} &= \frac{\partial^2 G_\theta(\widehat{p_i})}{\partial \theta_2\theta_1} = \log(\widehat{p_i})\widehat{p_i}^{\theta_1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-1}(1+\theta_2\log(1-\widehat{p_i}^{\theta_1})) \\
\frac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\theta_1^2} &= \theta_2\Bigg(\widehat{p_i}^{\theta_1-1}\log(\widehat{p_i})\left((1-\widehat{p_i})^{\theta_2-1}-(\theta_2-1)\widehat{p_i}^{\theta_1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-2}\right)(1+\theta_1\log(\widehat{p_i})) \\
&\quad +\widehat{p_i}^{\theta_1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-1}\log(\widehat{p_i})-(\theta_2-1)\log(\widehat{p_i})\widehat{p_i}^{2\theta_1-1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-2} \\
&\quad +\theta_1\log(\widehat{p_i})\left(2(1-\widehat{p_i}^{\theta_1})^{\theta_2-2}-\widehat{p_i}^{\theta_1}(\theta_2-2)(1-\widehat{p_i}^{\theta_1})^{\theta_2-3}\right)\Bigg) \\
\frac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\theta_2^2} &= \theta_1\widehat{p_i}^{\theta_1-1}\left(\log(1-\widehat{p_i}^{\theta_1})(1-\widehat{p_i}^{\theta_1})^{\theta_2-1}(2+\theta_1\log(1-\widehat{p_i}^{\theta_1})\right) \\
\frac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\theta_1\theta_2} &= \frac{\partial^3 G_\theta(\widehat{p_i})}{\partial \widehat{p_i}\theta_2\theta_1} = \widehat{p_i}^{\theta_1-1}\left((1-\widehat{p_i})^{\theta_2-1}(1+\theta_1\log(\widehat{p_i})-\theta_1(\theta_2-1)\log(\widehat{p_i})\widehat{p_i}^{\theta_1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-2}\right)(1+\theta_2\log(1-\widehat{p_i}^{\theta_1})) \\
&\quad -\theta_1\theta_2\widehat{p_i}^{\theta_2-1}(1-\widehat{p_i}^{\theta_1})^{\theta_2-1}\frac{\log(\widehat{p_i})\widehat{p_i}^{\theta}}{1-\widehat{p_i}^{\theta_1}}
\end{aligned}
$$