A Model-Agnostic Heuristics for Selective Classification

Andrea Pugnana[1], **Salvatore Ruggieri**[2]

[1] Scuola Normale Superiore, Pisa, Italy
[2] University of Pisa, Pisa, Italy
andrea.pugnana@sns.it, salvatore.ruggieri@unipi.it

Abstract

Selective classification (also known as classification with reject option) conservatively extends a classifier with a selection function to determine whether or not a prediction should be accepted (i.e., trusted, used, deployed). This is a highly relevant issue in socially sensitive tasks, such as credit scoring. State-of-the-art approaches rely on Deep Neural Networks (DNNs) that train at the same time both the classifier and the selection function. These approaches are model-specific and computationally expensive. We propose a model-agnostic approach, as it can work with any base probabilistic binary classification algorithm, and it can be scalable to large tabular datasets if the base classifier is so. The proposed algorithm, called SCROSS, exploits a cross-fitting strategy and theoretical results for quantile estimation to build the selection function. Experiments on real-world data show that SCROSS improves over existing methods.

Introduction

There is a pressing demand for a trustworthy AI (Wing 2021). For instance, the EU Regulatory framework proposal on AI (European Parliament and the Council 2021) rules that high-risk AI systems will be subject to strict obligations before deployment. One such obligation is to ensure "a high level of robustness, security and accuracy". High-risk AI systems refer to socially sensitive domains, including: healthcare, where predictions might be used to predict treatments; justice, where predictions can assess the risk of recidivism; hiring, where predictions can determine rankings of candidates; credit scoring, where predictions are routinely used to estimate the probability of repaying a debt. In all these cases, AI is typically framed as a probabilistic binary classifier (see e.g., (Dastile, Çelik, and Potsane 2020) for credit scoring), and the predictions are used to score or rank people. However, the predictive performance of classifiers is typically not homogeneous over the data distribution. Identifying sub-populations with low performance could be helpful for debugging and monitoring purposes, especially in high-risk scenarios. A direction toward improving robustness and accuracy is to lift from the canonical framework of binary classification to the selective classification one. Selective classification (also known as classification with reject option) (Chow 1970; Pietraszek 2005) conservatively extends a classifier with a selection function (reject option/strategy) to determine whether or not a prediction should be accepted. The selection function assesses the classifier prediction's confidence, robustness, and trustworthiness. If a sufficiently high level is not reached, the selective classifier abstains from producing a prediction – e.g. a risk score in a credit evaluation application. Levels can also inform the user about the degree of confidence in the classifier prediction, e.g., by associating a rating label to the prediction (e.g., high confident, average confident, low confident). Selective classification has been extensively studied from a theoretical side (El-Yaniv and Wiener 2010; Franc and Průša 2019). State-of-the-art practical approaches and tools, however, are model-specific, e.g., they are tailored to DNNs, as e.g., in the case of SelectiveNet (Geifman and El-Yaniv 2019) and Self-Adapting Training (SAT) (Huang, Zhang, and Zhang 2020), and focused/experimented mainly on image datasets.

Our contribution consists of a model-agnostic heuristics that lifts any base probabilistic binary classifier to a selective (probabilistic binary) classifier. The proposed algorithm, called SCROSS, adopts a cross-fitting approach. Training data is split into K folds. For each fold, we train a classifier on the remaining $K-1$ folds and test it on the fold. The confidence of predictions over all the folds is used to estimate the bounds on prediction confidence for which the base classifier should abstain. Estimating the bounds relies on theoretical results on quantile estimation (Knight and Bassett 2003) that guarantee a second-order improvement over sample quantile estimation. We present extensive experimentation on eight real-world datasets, comparing SCROSS with the standard PLUGIN baseline, which is model-agnostic but requires a validation set, and with the state-of-the-art approaches SelectiveNet and SAT.

Related Work

Selective classification trades off the error rate with the rejection rate. Two main models are considered.

In the cost model (Chow 1970), the goal is to minimize the expected loss defined in terms of misclassification and rejection costs. The Bayes optimal classifier requires posterior probabilities, which can be estimated on a validation set by the *plug-in* rule (Herbei and Wegkamp 2006). Other methods are based on estimates of the ROC curve (Tortorella

2005) on a validation set. Discriminative methods learn the selection function directly without the probability estimations (Bartlett and Wegkamp 2008).

An alternative model, which does not require defining costs, is the bounded-improvement model (Pietraszek 2005). Here, the selection function is evaluated based on the probability mass of the accepted region (coverage) and the average loss over such a region (selective risk). Optimal strategies maximize coverage for a maximum target risk or minimize risk for a minimum target coverage (Geifman and El-Yaniv 2017). An optimal strategy for noisy free data was proposed by (El-Yaniv and Wiener 2010), while (Franc and Průša 2019) deal with the case of known distributions. They also establish the equivalence of cost-based and bounded-improvement models.

Most approaches for selective classification are model-specific in that they try and build the classifier and the selection function concurrently. Examples include methods for SVM (Fumera and Roli 2002), boosted decision trees (Cortes, DeSalvo, and Mohri 2016), DNNs (Geifman and El-Yaniv 2017, 2019; Liu et al. 2019; Huang, Zhang, and Zhang 2020). A general taxonomy of existing methodologies can be found in (Hendrickx et al. 2021).

A closely related field is uncertainty estimation, e.g., as considered in conformal prediction (Shafer and Vovk 2008). In such a paradigm, model predictions are paired with uncertainty sets/intervals with statistical guarantees (Angelopoulos and Bates 2021). Another related line of research is algorithmic triage (a.k.a. learning to defer to an expert), where the learning algorithm accounts for the performances of human decision makers to whom rejected instances are passed to (Okati, De, and Gomez-Rodriguez 2021).

In this paper, we adhere to the bounded-improvement model, take a model-agnostic view of the problem, and specialize to the case of probabilistic binary classifiers.

Selective Classification

Consider random variables $(\mathbf{X}, Y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a feature space and $\mathcal{Y} = \{0, 1, \ldots, n_{\mathcal{Y}}\}$ is a finite label space. The joint distribution of $(\mathbf{X}, Y) \sim \mathcal{D}$ is unknown, but we can observe one or more datasets of i.i.d. realizations $S_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. A *classifier* is a function $h : \mathcal{X} \to \mathcal{Y}$ that maps features to classes defined from the hypothesis space $\mathcal{H}$. For a given loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and a dataset of observations (the training set), the supervised classification problem consists of finding a hypothesis $h \in \mathcal{H}$ that minimizes the *risk* function:

$$R(h) = \int_{\mathcal{X} \times \mathcal{Y}} l(h(\mathbf{x}), y) d\mathcal{D}(\mathbf{x}, y) = \mathbb{E}_{\mathcal{D}}[l(h(\mathbf{X}), Y)]$$

The *empirical risk* counterpart over a dataset of observations S_n distinct from the training set (the test set) is used to estimate $R(h)$:

$$\hat{r}(h|S_n) = \frac{1}{n} \sum_{i=1}^{n} l(h(\mathbf{x}_i), y_i)$$

The canonical setting is extended to model situations where predictions of classifiers are not sufficiently reliable, and rejecting to predict is preferable. A *selective classifier* is a pair (h, g), where h is a classifier and $g : \mathcal{X} \to \{0, 1\}$ is a *selection function*, which determines when to accept/abstain from using h as follows:

$$(h, g)(\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if } g(\mathbf{x}) = 1 \\ \text{abstain} & \text{otherwise.} \end{cases}$$

A soft selection approach (see (Geifman and El-Yaniv 2017)) consists of defining g in terms of a *confidence function* $m_h : \mathcal{X} \to [0, 1]$ (the subscript highlights that m_h depends on h) and a threshold θ of minimum confidence for accepting:

$$g(\mathbf{x}) = \mathbb{1}(m_h(\mathbf{x}) > \theta) \tag{1}$$

A good confidence function should rank instances based on descending loss, i.e., if $m_h(\mathbf{x}_i) \leq m_h(\mathbf{x}_j)$ then $l(h(\mathbf{x}_i), y_i) \geq l(h(\mathbf{x}_j), y_j)$. The *coverage* of a selective classifier is $\phi(g) = E_{\mathcal{D}}[g(\mathbf{X})]$, i.e., the expected probability mass of the accepted region. The *selective risk* is the risk over the accepted region:

$$R(h, g) = \frac{\mathbb{E}_{\mathcal{D}}[l(h(\mathbf{X}), Y) g(\mathbf{X})]}{\phi(g)} = \mathbb{E}_{\mathcal{D}}[l(h(\mathbf{X}), Y) | g(\mathbf{X})]$$

The inherent trade-off between risk and coverage is summarized by the *risk-coverage curve* (El-Yaniv and Wiener 2010). Empirical coverage and empirical selective risk are respectively defined as $\hat{\phi}(g|S_n) = \frac{\sum_{i=1}^n g(\mathbf{x}_i)}{n}$ and:

$$\hat{r}(h, g|S_n) = \frac{\frac{1}{n} \sum_{i=1}^n l(h(\mathbf{x}_i), y_i) g(\mathbf{x}_i)}{\hat{\phi}(g|S_n)}$$

The selective classification problem can be framed by fixing an upper bound to the selective risk and then looking for a selective classifier that maximizes coverage. (Geifman and El-Yaniv 2017) show how to convert this framing into an alternative one, where a lower bound c for coverage is fixed (*target coverage*), and then the problem looks for a selective classifier that minimizes risk. We will consider such an alternative formulation. For g as in (1), the *selective classification problem* can be stated as:

$$\min_{\theta \in [0,1]} R(h, g|\mathcal{S}) \quad \text{s.t.} \quad \phi(g) \geq c \tag{2}$$

State of the art approaches include *SelectiveNet* (Geifman and El-Yaniv 2019) and *SAT* (*Self-Adaptive Training*) (Huang, Zhang, and Zhang 2020). They are specifically designed for DNNs. *SelectiveNet* builds at the same time the classifier h and the selection function g by optimizing a loss function parametric in the target coverage c. Furthermore, g is a soft selection function as in (1), with m_h set to the softmax function, and θ determined on a validation set (not used for training) by estimating the $100(1 - c)$ percentile of the distribution of m_h. SAT incorporates model predictions into the training process by using a convex combination of labels and predictions. For the selective classification task, SAT closely resembles the *Deep Gamblers* approach (Liu et al. 2019), which models the reject option as an additional class label in an extension of the cross-entropy loss function. g is a soft selection function, with m_h equal to the predicted probability of the additional class label. The threshold θ is computed on a validation set. Differently from *SelectiveNet*, the target coverage c is not part of the loss function, and then the calculation of θ does not require re-training.

The SCROSS Algorithm

We focus our analysis on probabilistic binary classifiers, where $\mathcal{Y} = \{0, 1\}$ and $h(\mathbf{x}) = \mathbb{1}(s(\mathbf{x}) > 1/2)$. Here, the *score* $s(\mathbf{x}) \in [0, 1]$ is an estimate of the posterior probability $P_{\mathcal{D}}(Y = 1|\mathbf{X} = \mathbf{x})$. We assume that the algorithm and hyperparameters for fitting h are given. Apart from this assumption, our approach for lifting h to a selective classifier (h, g) will be completely agnostic to the specific algorithm and hyperparameters. Specifically, we adopt a soft selection function $g(\mathbf{x}) = \mathbb{1}(m_h(\mathbf{x}) > \theta)$, with the canonical choice of softmax as confidence function:

$$m_h(\mathbf{x}) = \max\{s(\mathbf{x}), 1 - s(\mathbf{x})\}.$$

Our approach aims at determining a threshold θ for the selective classification problem (2) that is a better estimate of the $100(1 - c)$ percentile of m_h than the one adopted by the state-of-the-art methods, namely the empirical quantile of a sample. The estimate will be "better" in two respects.

First, we adopt a subsample quantile estimator with theoretical guarantees to improve the variance of a full-sample quantile estimator. The theoretical backbone of our approach is supported by the results by (Knight and Bassett 2003), reported next for completeness.

Theorem 1 ((Knight and Bassett 2003), Theorem 3) *Given a random sample distributed according to F, satisfying some regularity conditions, and K non-overlapping subsamples of it, let $\hat{q}(\alpha)$ be the empirical α-quantile estimator of F over the whole sample, and $\bar{q}(\alpha)$ a weighted average of the empirical quantile estimators of F over the subsamples. For $t \in [0, 1]$, let us define the linear combination $\tilde{q}(\alpha) = t\hat{q}(\alpha) + (1 - t)\bar{q}(\alpha)$. The variance of $\tilde{q}(\alpha)$ is minimized for $t = 1/\sqrt{2}$, $K = 2$ and equally sized subsamples.*

The sample quantile $\hat{q}(\alpha)$ is known to be asymptotically normal. A weighted average of subsample quantiles $\bar{q}(\alpha)$ is first-order equivalent to $\hat{q}(\alpha)$. The above theorem states the conditions for minimizing the variance of (the second-order term of) any linear combination of the two estimators.

As a second enhancement, we prevent setting apart a validation set from the training set, as done in state-of-the-art methods, and then we will be able to fit the classifier h on the whole available training set. Since the goal is to estimate quantiles over the (unknown) population of confidence values of the classifier h, we approximate it by using a cross-fitting strategy, similar to the double ML approach (Chernozhukov et al. 2018).

Let us describe our approach. Given a training set $\mathcal{S}$, we partition $\mathcal{S}$ into K non overlapping stratified folds $\mathcal{S}_1, \ldots, \mathcal{S}_K$, and we define $\mathcal{S}_{-k} = \mathcal{S} \setminus \mathcal{S}_k$ for $k = 1, \ldots, K$. For each k, we train a base probabilistic classifier h_k over $\mathcal{S}_{-k}$. We then compute the scores $s_k(\mathbf{x})$ for $\mathbf{x} \in \mathcal{S}_k$, and the confidence values $m_{h_k}(\mathbf{x}) = \max\{h_k(\mathbf{x}), 1 - h_k(\mathbf{x})\}$. Let us define $m_{h_k}(\mathcal{S}_k) = \{m_{h_k}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{S}_k\}$, and $m(\mathcal{S}) = \cup_{k=1}^{K} m_{h_k}(\mathcal{S}_k)$. Notice that $m(\mathcal{S})$ is an approximation of the (unknown) population of confidence values of the classifier h, since each h_k is built on $K - 1$ stratified folds of the entire training set $\mathcal{S}$ which, in turn, is a set of i.i.d. realizations from $\mathcal{D}$. For a target coverage c, we set $\hat{\theta}$ as as the $100(1-c)$ percentile of the empirical distribution of $m(\mathcal{S})$. According to Theorem 1, to minimize the variance of the quantile estimator, we then randomly partition $m(\mathcal{S})$ in two subsets of equal size and we compute the empirical $100(1 - c)$ percentiles over each of them, denoting such percentiles as $\bar{\theta}_1$ and $\bar{\theta}_2$ respectively. The final estimator $\tilde{\theta}$ is:

$$\tilde{\theta} = \frac{1}{\sqrt{2}}\hat{\theta} + (1 - \frac{1}{\sqrt{2}})(\frac{1}{2}\bar{\theta}_1 + \frac{1}{2}\bar{\theta}_2) \tag{3}$$

Algorithm 1 shows the details of the approach, which we name SCROSS for Selective CROSS-Classification. Notice that the final classifier h is built at line 13 on the full training set $\mathcal{S}$. This is not strictly required, e.g., in the case of an already trained classifier.

Algorithm 1: SCROSS.fit()

Input : $\mathcal{S} = \{\mathbf{X}, \mathbf{y}\}$ - training set, h - base classification algorithm, c - target coverage K - number of folds

Output: (h, g) - selective classifier

1 $\mathcal{S}_1, \ldots, \mathcal{S}_K \leftarrow$ StratifiedKFold($(\mathbf{X}, \mathbf{y}), K$) `// partitioning`
2 **for** $k \in 1, \ldots, K$ **do** `// for each fold`
3 $\quad \mathcal{S}_{-k} = \mathcal{S} \setminus \mathcal{S}_k$ `// training data`
4 $\quad h_k \leftarrow h.fit(\mathcal{S}_{-k})$ `// train` k^{th} `classifier`
5 $\quad s_k = h_k.score(\mathcal{S}_k)$ `// score data`
6 $\quad m_k = \max\{s_k, 1 - s_k\}$ `// compute confidence`
7 $m \leftarrow \cup_{k=1}^{K} m_k$ `// store all the scores`
8 $\hat{\theta} \leftarrow$ perc$(m, 100(1 - c))$ `// calculate` $\hat{\theta}$
9 $m_1, m_2 \leftarrow$ KFold$(m, 2)$ `// partitioning`
10 $\bar{\theta}_1 \leftarrow$ perc$(m_1, 100(1 - c))$ `// calculate` $\bar{\theta}_1$,
11 $\bar{\theta}_2 \leftarrow$ perc$(m_2, 100(1 - c))$ `// calculate` $\bar{\theta}_2$
12 $\tilde{\theta} = \frac{1}{\sqrt{2}}\hat{\theta} + (1 - \frac{1}{\sqrt{2}})(\frac{1}{2}\bar{\theta}_1 + \frac{1}{2}\bar{\theta}_2)$ `// final` $\bar{\theta}$ `estimate`
13 $h \leftarrow h.fit(\mathcal{S})$ `// classifier`
14 $g \leftarrow$ **lambda** $\mathbf{x} : \mathbb{1}(\max\{1 - h.score(\mathbf{x}), h.score(\mathbf{x})\} \geq \bar{\theta})$ `// selection function`
15 **return** (h, g)

Experiments

We experiment our proposal[1] on eight datasets and compare it with a bounded-improvement version of the plug-in rule (Herbei and Wegkamp 2006) (PLUG-IN), the SelectiveNet method (Geifman and El-Yaniv 2019) (SELNET), and the Self Adaptive Training method (Huang, Zhang, and Zhang 2020) (SAT). PLUG-IN uses the softmax function as the confidence function, as in our approach, but it computes the θ parameter as the $100(1 - c)$ percentile on a validation set, as in the case of SELNET and SAT. Unlike those two approaches, PLUG-IN is a model-agnostic method. Thus, PLUG-IN is a natural baseline to evaluate the contribution of our cross-fitting approach.

Settings. For tabular data, we train both SELNET and SAT using a ResNet structure (Gorishniy et al. 2021), while we use LightGBM (Ke et al. 2017) as a base classifier for both SCROSS and PLUGIN. We also evaluate how the performances of SCROSS are affected by the base classifier

[1]Python source code and experimental notebooks are available at https://github.com/andrepugni/SCross.

choice. For the image data (the CatsVsDogs dataset), we compare all the methods on the same VGG architecture provided by (Liu et al. 2019) as the base classifier. For SCROSS and PLUGIN we train the network using cross-entropy loss; for SAT we use the Self Adaptive Loss and all the parameters as in the selective implementation experiment of (Huang, Zhang, and Zhang 2020); for SELNET we use as loss function $\mathcal{L} = \alpha\mathcal{L}_{(h,g)} + (1-\alpha)\mathcal{L}_v$, where $\mathcal{L}_{(h,g)} = \hat{r}(h,g|S_m) + \lambda(\max(0, c - \hat{\phi}(g)))^2$, $\mathcal{L}_v = R_{emp}(v)$ with v the auxiliary head of SelectiveNet, and with parameters α and λ set to 0.5 and 32, as in (Geifman and El-Yaniv 2019). For SELNET and SAT approach, we set 300 epochs in training, Stochastic Gradient Descent as an optimizer, a learning rate of .1 decreased by a factor .5 every 25 epochs, as in the original papers. All the parameters of base classifiers are left as the default ones. Regarding SCROSS, we fix $K = 5$ unless otherwise specified. The validation sets for SELNET and SAT consist of 10% of the training set and 2,000 instances, respectively, as in their original implementations. Experiments were run on a machine with 96 cores equipped with Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz and two NVIDIA RTX A6000, OS Ubuntu 20.04.4, programming language Python 3.8.12.

Evaluation metrics. The metrics used for performance evaluation include the empirical coverage, the empirical accuracy (1 minus the empirical selective risk), and the elapsed training time. They are computed on a test set (25% of available data) separated from the training set (75%). The split into training and test set is time-based when a timestamp feature is available and stratified random otherwise. Performances are shown in the format "mean $\pm$ stdev", where the mean and the standard deviation refer to 1,000 bootstrap runs over the test set, as proposed in (Rajkomar et al. 2018). We follow the approach in (Demsar 2006) to test whether differences across multiple classifiers are statistically significant at .001 significance level.[2]

Datasets. We briefly summarize the main characteristics of the experimental datasets (7 tabular datasets and 1 image dataset) and the pre-processing steps performed. A Jupyter notebook is provided for each dataset, including all the preprocessing details.

Adult. We dropped from the raw census data (Dua and Graff 2017) the instances with missing values, because some base classifiers do not deal with them. The categorical features were one-hot encoded for SCROSS and PLUG-IN, while SELNET and SAT learn embeddings over them. The final training set contains 30,162 instances and 10 features (55 after one-hot encoding). The test set size is 15,060.

Lending. The Lending Club dataset[3] regards loans in an online platform. We considered instances whose loan status is either *Fully Paid* or *Charged Off* and used them as class labels. We removed redundant and non-informative features and split instances as follows: the ones up to May 2017 are included in the training set; the ones from June 2017 up to 2020 are included in the test set. Categorical features were processed as for the Adult dataset. The final training set contains 1,364,697 instances and 18 features (65 after one-hot encoding). The test set size is 445,912.

CSDS1/2/3. The datasets CSDS1, CSDS2 and CSDS3 from (Barddal et al. 2020) regard defaults in repaying a loan: within six months for CSDS1 (data span over 15 months), within 2 months for CSDS2 (data span over 25 months), and within three months for CSDS3 (data span over 16 months). All features are anonymized. We removed those with missing values. We then used the time feature to split the raw data into training and test set: for CSDS1, the threshold was June 2017; for CSDS2, November 2017; and for CSDS3, November 2014. The training set of CSDS1 consists of 230,409 instances and 155 features (test set size is 76,939). For CSDS2, the training set contains 37,100 instances and 35 features (test set size is 12,533). For CSDS3, the training set contains 71,177 instances and 144 features (test set size 23,288).

GiveMe. The GiveMeSomeCredit dataset[4] aims at predicting the financial distress of a borrower within two years. As for CSDS datasets, we removed features with missing values. The training and the test set were obtained by stratified random sampling, and they contain 112,500 and 37,500 instances, respectively, and 12 features.

UCICredit. This dataset from (Dua and Graff 2017) concerns whether or not a credit card holder will default in the next six months (Yeh and Lien 2009). There are no missing values. Training and test sets were obtained by stratified random sampling. The training set includes 22,500 instances (7,500 for the test set) and 23 features.

CatsVsDogs. This dataset[5] containing a collection of cats and dogs images. The task here is to distinguish between the two species. The training and test sets were split as described in (Liu et al. 2019). There are no missing values. The training set contains 20,000 images, each one of 64x64 pixels. The test set consists of 5,000 images.

Comparing approaches. Table 1 compares the empirical coverage, the selective accuracy, and the elapsed training time metrics (mean $\pm$ stdev) for SCROSS, PLUGIN, SELNET, and SAT. For each dataset, we consider different target coverages c, ranging from .75 to .99. A value is shown in bold if the classifier is statistically significantly better than others in the same row (i.e., empirical coverage closest to the target c, highest empirical selective accuracy, and shortest elapsed training time, respectively). In case of ties among classifiers, we report in bold all values of such classifiers. Before discussing the results of Table 1, a clarification should be pointed out about the limitation of a fully fair comparison. First, SCROSS and PLUGIN can be compared using the same base classifiers because they are both model-agnostic. Similarly, SELNET and SAT can be compared to each other because they are end-to-end learning, namely they fit the classifier and the selection function together. It is not possible, instead, to use a same trained base classifier for all of the four methods, as SELNET and SAT would

[2]We rely on the Python package *autorank* (Herbold 2020) for statistical analysis.

[3]https://www.kaggle.com/wordsforthewise/lending-club

[4]https://www.kaggle.com/c/GiveMeSomeCredit

[5]https://www.kaggle.com/competitions/dogs-vs-cats

		Empirical Coverage				Empirical Selective Accuracy				Elapsed Training Time			
	c	SCROSS	PLUGIN	SELNET	SAT	SCROSS	PLUGIN	SELNET	SAT	SCROSS	PLUGIN	SELNET	SAT
Adult	.99	.993 ± .001	.992 ± .001	**.991 ± .001**	**.990 ± .001**	.871 ± .003	**.872 ± .003**	.847 ± .003	.845 ± .003	1.84	**0.21**	2,059	1,088
	.95	**.950 ± .002**	.947 ± .002	.946 ± .002	.941 ± .002	.888 ± .003	**.888 ± .003**	.856 ± .003	.845 ± .003	1.84	**0.21**	2,065	1,088
	.90	.904 ± .003	**.900 ± .003**	**.899 ± .003**	.893 ± .003	.902 ± .003	**.903 ± .003**	.875 ± .003	.845 ± .004	1.84	**0.21**	2,067	1,088
	.85	**.850 ± .003**	.846 ± .003	.840 ± .004	**.850 ± .003**	**.919 ± .003**	**.920 ± .003**	.894 ± .003	.845 ± .004	1.84	**0.21**	2,065	1,088
	.80	**.802 ± .004**	.794 ± .004	.786 ± .004	.828 ± .004	.934 ± .003	**.936 ± .003**	.902 ± .003	.846 ± .004	1.84	**0.21**	2,059	1,088
	.75	**.746 ± .004**	.743 ± .004	.741 ± .004	.798 ± .004	.950 ± .003	**.950 ± .003**	.923 ± .003	.845 ± .004	1.84	**0.21**	2,072	1,088
Lending	.99	.994 ± .001	.995 ± .001	**.992 ± .001**	1.00 ± .000	**.899 ± .001**	**.899 ± .001**	.873 ± .001	.876 ± .001	13.21	**2.44**	8,933	7,668
	.95	**.969 ± .001**	.970 ± .001	.979 ± .001	1.00 ± .000	**.910 ± .001**	.909 ± .001	.880 ± .001	.876 ± .001	13.21	**2.44**	8,935	7,668
	.90	.938 ± .001	.939 ± .001	**.918 ± .001**	1.00 ± .000	**.924 ± .001**	.923 ± .001	.891 ± .001	.876 ± .001	13.21	**2.44**	8,916	7,668
	.85	**.908 ± .001**	.909 ± .001	.931 ± .001	1.00 ± .000	**.938 ± .001**	.937 ± .001	.900 ± .001	.876 ± .001	13.21	**2.44**	8,920	7,668
	.80	**.879 ± .001**	.880 ± .001	.897 ± .001	1.00 ± .000	**.951 ± .001**	.951 ± .001	.918 ± .001	.876 ± .001	13.21	**2.44**	8,911	7,668
	.75	.849 ± .001	.850 ± .001	**.821 ± .001**	1.00 ± .000	**.963 ± .001**	**.963 ± .001**	.940 ± .001	.876 ± .001	13.21	**2.44**	8,903	7,668
GiveMe	.99	.990 ± .001	**.990 ± .001**	**.991 ± .001**	.992 ± .001	**.942 ± .002**	.942 ± .002	.939 ± .002	.937 ± .002	1.64	**0.27**	2,420	1,364
	.95	**.950 ± .002**	.949 ± .002	.948 ± .002	.988 ± .001	**.956 ± .002**	.956 ± .002	.953 ± .002	.939 ± .002	1.64	**0.27**	2,628	1,364
	.90	**.902 ± .002**	.903 ± .002	**.901 ± .002**	.984 ± .001	**.967 ± .001**	.967 ± .001	.964 ± .001	.940 ± .002	1.64	**0.27**	2,283	1,364
	.85	**.852 ± .002**	.856 ± .002	.853 ± .002	.961 ± .001	**.973 ± .001**	.973 ± .001	.969 ± .001	.945 ± .002	1.64	**0.27**	2,281	1,364
	.80	**.800 ± .003**	.807 ± .003	.806 ± .003	.955 ± .002	**.978 ± .001**	.977 ± .001	.971 ± .001	.946 ± .002	1.64	**0.27**	2,278	1,364
	.75	.748 ± .003	.756 ± .003	**.750 ± .003**	.950 ± .002	**.981 ± .001**	.980 ± .001	.975 ± .001	.946 ± .002	1.64	**0.27**	2,276	1,364
UCICredit	.99	.993 ± .001	**.993 ± .002**	**.988 ± .002**	1.00 ± .000	.814 ± .005	**.814 ± .005**	.813 ± .005	.813 ± .005	1.31	**0.19**	1,719	872
	.95	**.951 ± .003**	**.948 ± .003**	**.949 ± .003**	1.00 ± .000	**.827 ± .005**	.826 ± .005	.818 ± .005	.813 ± .005	1.31	**0.19**	1,677	872
	.90	.906 ± .004	**.897 ± .004**	**.904 ± .004**	.992 ± .002	**.839 ± .005**	.838 ± .005	.829 ± .005	.814 ± .005	1.31	**0.19**	1,684	872
	.85	.848 ± .005	**.851 ± .005**	.846 ± .005	.992 ± .002	**.855 ± .005**	.849 ± .005	.848 ± .005	.814 ± .005	1.31	**0.19**	1,675	872
	.80	.797 ± .005	**.803 ± .005**	.806 ± .005	.992 ± .002	**.867 ± .005**	.863 ± .005	.859 ± .005	.814 ± .005	1.31	**0.19**	1,691	872
	.75	**.750 ± .005**	**.751 ± .006**	.770 ± .005	.992 ± .002	**.875 ± .005**	.872 ± .005	.867 ± .005	.814 ± .005	1.31	**0.19**	1,681	872
CSDS1	.99	.980 ± .001	.980 ± .001	.980 ± .001	**.986 ± .001**	**.863 ± .002**	.863 ± .002	.862 ± .002	.861 ± .002	6.13	**1.06**	4,933	2,888
	.95	.917 ± .001	.918 ± .001	.924 ± .001	**.934 ± .001**	**.875 ± .002**	**.875 ± .002**	.873 ± .002	.872 ± .002	6.13	**1.06**	5,158	2,888
	.90	.849 ± .002	.853 ± .002	.859 ± .002	**.869 ± .002**	**.885 ± .002**	.885 ± .002	.882 ± .002	.882 ± .002	6.13	**1.06**	5,146	2,888
	.85	.790 ± .002	.792 ± .002	.795 ± .002	**.805 ± .002**	**.892 ± .002**	**.892 ± .002**	.890 ± .002	.889 ± .002	6.13	**1.06**	5,151	2,888
	.80	.736 ± .002	.737 ± .002	**.741 ± .002**	.740 ± .002	**.899 ± .002**	.898 ± .002	.897 ± .002	.896 ± .002	6.13	**1.06**	5,155	2,888
	.75	.682 ± .002	.681 ± .002	.680 ± .002	**.690 ± .002**	.904 ± .002	**.904 ± .002**	.903 ± .002	.902 ± .002	6.13	**1.06**	5,152	2,888
CSDS2	.99	.983 ± .002	**.984 ± .002**	.933 ± .003	1.00 ± .000	.982 ± .002	.982 ± .002	**.984 ± .002**	.982 ± .002	1.33	**0.23**	669	351
	.95	.916 ± .003	.927 ± .003	.905 ± .003	**.934 ± .003**	.985 ± .002	.983 ± .002	**.985 ± .002**	.984 ± .002	1.33	**0.23**	669	351
	.90	.834 ± .004	.851 ± .004	.814 ± .004	**.875 ± .003**	.985 ± .002	.984 ± .002	**.986 ± .002**	.984 ± .002	1.33	**0.23**	669	351
	.85	.761 ± .004	.773 ± .004	.747 ± .004	**.812 ± .004**	.986 ± .002	.985 ± .002	**.986 ± .002**	.986 ± .002	1.33	**0.23**	669	351
	.80	.687 ± .005	.715 ± .005	.682 ± .005	**.761 ± .004**	.987 ± .002	.986 ± .002	**.987 ± .002**	.986 ± .002	1.33	**0.23**	671	351
	.75	.620 ± .005	.651 ± .005	.616 ± .005	**.703 ± .005**	**.987 ± .002**	.986 ± .002	.987 ± .002	.987 ± .002	1.33	**0.23**	658	351
CSDS3	.99	**.991 ± .001**	.992 ± .001	.992 ± .001	.995 ± .001	**.816 ± .003**	.816 ± .003	.809 ± .003	.810 ± .003	5.25	**0.80**	1,316	696
	.95	.955 ± .002	.957 ± .002	**.949 ± .002**	.954 ± .002	**.827 ± .003**	.826 ± .003	.819 ± .003	.821 ± .003	5.25	**0.80**	1,318	696
	.90	.911 ± .002	.909 ± .002	.906 ± .002	**.904 ± .002**	.841 ± .003	**.841 ± .003**	.835 ± .003	.836 ± .003	5.25	**0.80**	1,315	696
	.85	.865 ± .003	.859 ± .003	**.850 ± .003**	.862 ± .003	.855 ± .003	**.856 ± .003**	.849 ± .003	.849 ± .003	5.25	**0.80**	1,306	696
	.80	.818 ± .003	**.808 ± .003**	**.808 ± .003**	.824 ± .003	.869 ± .003	**.871 ± .003**	.862 ± .003	.859 ± .003	5.25	**0.80**	1,319	696
	.75	.769 ± .003	**.758 ± .003**	.758 ± .003	.777 ± .003	.883 ± .003	**.884 ± .003**	.874 ± .003	.872 ± .003	5.25	**0.80**	1,316	696
CatsVsDogs	.99	**.992 ± .002**	.997 ± .001	.923 ± .004	.913 ± .004	.972 ± .003	.964 ± .003	.980 ± .003	**.988 ± .002**	18,638	**3,240**	3,535	3,767
	.95	**.960 ± .003**	.979 ± .003	.855 ± .006	.867 ± .005	.982 ± .002	.963 ± .003	.988 ± .002	**.990 ± .002**	18,638	**3,240**	3,581	3,767
	.90	**.922 ± .004**	.948 ± .004	.809 ± .006	.819 ± .006	.989 ± .002	.962 ± .003	**.993 ± .002**	.990 ± .002	18,638	**3,240**	3,581	3,767
	.85	**.889 ± .005**	.912 ± .004	.795 ± .006	.777 ± .006	**.993 ± .002**	.962 ± .003	**.992 ± .002**	.990 ± .002	18,638	**3,240**	3,573	3,767
	.80	**.846 ± .005**	.869 ± .005	**.758 ± .007**	.729 ± .007	.994 ± .002	.960 ± .003	**.995 ± .002**	.990 ± .002	18,638	**3,240**	3,587	3,767
	.75	.794 ± .006	.822 ± .006	**.720 ± .007**	.682 ± .007	.994 ± .002	.959 ± .004	**.997 ± .002**	.990 ± .002	18,638	**3,240**	3,580	3,767
	#	19/48	11/48	17/48	13/48	27/48	16/48	9/48	2/48	0/48	48/48	0/48	0/48

Table 1: Performance metrics for SCROSS and baselines (1,000 bootstrap runs over the test set, results as mean ± stdev).

re-train it differently. For image data, where the base classifier is a VGG architecture for all four methods, we choose to set a standard cross-entropy loss in training for SCROSS and PLUGIN (see settings), which alleviates but do not solve the problem raised. For tabular data, we choose to adopt as base classifier a gradient boosting tree classifier, namely LightGBM, which is known to outperform DNNs in general (Grinsztajn, Oyallon, and Varoquaux 2022; Shwartz-Ziv and Armon 2022). However, the selective accuracy of all four methods for $c = .99$ is very close in Table 1. This means that the performances of the LightGBM base classifier and of the ResNet trained by SELNET and SAT are very close

		Empirical Coverage					Empirical Selective Accuracy				
	c	$K=2$	$K=3$	$K=5$	$K=7$	$K=10$	$K=2$	$K=3$	$K=5$	$K=7$	$K=10$
Adult	.99	.992 ± .001	**.991 ± .001**	.993 ± .001	.992 ± .001	.992 ± .001	.872 ± .003	**.872 ± .003**	.871 ± .003	**.872 ± .003**	**.872 ± .003**
	.95	**.950 ± .002**	.949 ± .002	**.950 ± .002**	**.951 ± .002**	**.951 ± .002**	**.888 ± .003**	**.888 ± .003**	**.888 ± .003**	.887 ± .003	.887 ± .003
	.90	.906 ± .003	**.904 ± .003**	**.904 ± .003**	.904 ± .003	.904 ± .003	.901 ± .003	**.902 ± .003**	**.902 ± .003**	**.902 ± .003**	**.902 ± .003**
	.85	.849 ± .003	.850 ± .003	.850 ± .003	**.850 ± .003**	**.850 ± .003**	**.920 ± .003**	.920 ± .003	.919 ± .003	.919 ± .003	.919 ± .003
	.80	.798 ± .004	**.801 ± .004**	.802 ± .004	.802 ± .004	**.801 ± .004**	**.935 ± .003**	.934 ± .003	.934 ± .003	.934 ± .003	.934 ± .003
	.75	.745 ± .004	**.747 ± .004**	.746 ± .004	.746 ± .004	.747 ± .004	**.950 ± .003**	.949 ± .003	.950 ± .003	.949 ± .003	.949 ± .003
Lending	.99	.995 ± .001	.995 ± .001	**.994 ± .001**	.995 ± .001	.994 ± .001	**.899 ± .001**	**.899 ± .001**	**.899 ± .001**	**.899 ± .001**	**.899 ± .001**
	.95	.970 ± .001	.969 ± .001	**.969 ± .001**	.970 ± .001	.970 ± .001	**.910 ± .001**	.910 ± .001	**.910 ± .001**	.910 ± .001	.910 ± .001
	.90	.939 ± .001	.938 ± .001	**.938 ± .001**	.938 ± .001	.938 ± .001	.923 ± .001	.924 ± .001	**.924 ± .001**	.924 ± .001	.924 ± .001
	.85	.909 ± .001	.908 ± .001	**.908 ± .001**	.908 ± .001	.908 ± .001	.937 ± .001	.937 ± .001	**.938 ± .001**	.937 ± .001	**.938 ± .001**
	.80	.880 ± .001	.879 ± .001	.879 ± .001	.879 ± .001	**.878 ± .001**	.950 ± .001	.951 ± .001	.951 ± .001	.951 ± .001	**.951 ± .001**
	.75	.850 ± .001	.849 ± .001	.849 ± .001	.849 ± .001	**.849 ± .001**	.962 ± .001	.963 ± .001	.963 ± .001	.963 ± .001	**.963 ± .001**
GiveMe	.99	**.990 ± .001**	**.990 ± .001**	.990 ± .001	.990 ± .001	**.990 ± .001**	.942 ± .002	.942 ± .002	**.942 ± .002**	**.942 ± .002**	**.942 ± .002**
	.95	.948 ± .002	.949 ± .002	**.950 ± .002**	.950 ± .002	**.950 ± .002**	**.957 ± .002**	.957 ± .002	.956 ± .002	.956 ± .002	.956 ± .002
	.90	**.901 ± .002**	**.901 ± .002**	.902 ± .002	**.901 ± .002**	.902 ± .002	**.967 ± .001**	.967 ± .001	.967 ± .001	.967 ± .001	**.967 ± .001**
	.85	.853 ± .002	.852 ± .002	**.852 ± .002**	.852 ± .002	.853 ± .002	**.973 ± .001**	.973 ± .001	**.973 ± .001**	.973 ± .001	.973 ± .001
	.80	.797 ± .003	.799 ± .003	**.800 ± .003**	**.800 ± .003**	**.801 ± .003**	**.978 ± .001**	**.978 ± .001**	.978 ± .001	.978 ± .001	.978 ± .001
	.75	.744 ± .003	.746 ± .003	.748 ± .003	.747 ± .003	**.749 ± .003**	**.981 ± .001**	**.981 ± .001**	**.981 ± .001**	**.981 ± .001**	**.981 ± .001**
UCICredit	.99	**.992 ± .002**	.993 ± .001	.993 ± .001	.993 ± .001	.993 ± .001	**.814 ± .005**	.814 ± .005	.814 ± .005	.814 ± .005	.814 ± .005
	.95	**.950 ± .003**	**.951 ± .003**	**.951 ± .003**	.953 ± .003	.953 ± .003	**.827 ± .005**	**.827 ± .005**	**.827 ± .005**	.826 ± .005	.826 ± .005
	.90	**.903 ± .004**	.906 ± .004	.906 ± .004	.907 ± .004	**.903 ± .004**	**.840 ± .005**	.839 ± .005	.839 ± .005	.839 ± .005	**.840 ± .005**
	.85	.849 ± .005	**.851 ± .005**	.848 ± .005	.853 ± .005	.849 ± .005	**.855 ± .005**	.854 ± .005	**.855 ± .005**	.854 ± .005	**.855 ± .005**
	.80	.795 ± .005	.795 ± .005	.797 ± .005	**.801 ± .005**	.797 ± .005	**.867 ± .005**	**.867 ± .005**	**.867 ± .005**	.866 ± .005	**.867 ± .005**
	.75	.745 ± .005	.745 ± .005	**.750 ± .005**	.754 ± .005	**.751 ± .005**	.875 ± .005	**.876 ± .005**	.875 ± .005	.875 ± .005	.875 ± .005
CSDS1	.99	.979 ± .001	.979 ± .001	.980 ± .001	**.981 ± .001**	.981 ± .001	**.863 ± .002**	**.863 ± .002**	**.863 ± .002**	.862 ± .002	.863 ± .002
	.95	.917 ± .001	**.918 ± .001**	.917 ± .001	.917 ± .001	.917 ± .001	**.875 ± .002**	**.875 ± .002**	.875 ± .002	.875 ± .002	.875 ± .002
	.90	**.850 ± .002**	.849 ± .002	.849 ± .002	.848 ± .002	.849 ± .002	**.885 ± .002**	.885 ± .002	.885 ± .002	.885 ± .002	.885 ± .002
	.85	**.792 ± .002**	.791 ± .002	.790 ± .002	.790 ± .002	.790 ± .002	.892 ± .002	.892 ± .002	**.892 ± .002**	**.892 ± .002**	**.892 ± .002**
	.80	**.739 ± .002**	.737 ± .002	.736 ± .002	.736 ± .002	.736 ± .002	.899 ± .002	**.899 ± .002**	**.899 ± .002**	**.899 ± .002**	**.899 ± .002**
	.75	**.684 ± .002**	.682 ± .002	.682 ± .002	.682 ± .002	.682 ± .002	**.904 ± .002**	**.904 ± .002**	**.904 ± .002**	**.904 ± .002**	**.904 ± .002**
CSDS2	.99	.983 ± .002	.983 ± .002	.983 ± .002	.984 ± .002	**.984 ± .002**	**.983 ± .002**	**.983 ± .002**	**.982 ± .002**	**.983 ± .002**	**.983 ± .002**
	.95	.902 ± .003	.915 ± .003	.916 ± .003	**.919 ± .003**	.918 ± .003	**.985 ± .002**	.985 ± .002	.985 ± .002	.984 ± .002	.985 ± .002
	.90	.791 ± .004	.824 ± .004	.834 ± .004	**.836 ± .004**	**.836 ± .004**	**.986 ± .002**	.986 ± .002	.985 ± .002	.985 ± .002	.985 ± .002
	.85	.686 ± .005	.736 ± .004	.761 ± .004	**.765 ± .004**	.764 ± .004	**.987 ± .002**	.986 ± .002	.986 ± .002	.986 ± .002	.986 ± .002
	.80	.597 ± .005	.657 ± .005	.687 ± .005	**.692 ± .005**	.691 ± .005	**.988 ± .002**	.987 ± .002	.987 ± .002	.987 ± .002	.987 ± .002
	.75	.521 ± .005	.586 ± .005	.620 ± .005	.623 ± .005	**.624 ± .005**	**.987 ± .002**	**.988 ± .002**	**.987 ± .002**	**.988 ± .002**	**.988 ± .002**
CSDS3	.99	**.990 ± .001**	**.990 ± .001**	**.991 ± .001**	.991 ± .001	**.991 ± .001**	**.817 ± .003**	**.817 ± .003**	.816 ± .003	.816 ± .003	**.817 ± .003**
	.95	.953 ± .002	**.953 ± .002**	.955 ± .002	.955 ± .002	.955 ± .002	**.828 ± .003**	**.828 ± .003**	.827 ± .003	.827 ± .003	.827 ± .003
	.90	**.907 ± .002**	.907 ± .002	.911 ± .002	.910 ± .002	.910 ± .002	**.842 ± .003**	**.842 ± .003**	.841 ± .003	.841 ± .003	.841 ± .003
	.85	**.861 ± .003**	.862 ± .003	.865 ± .003	.865 ± .003	.863 ± .003	**.856 ± .003**	**.856 ± .003**	.855 ± .003	.855 ± .003	.856 ± .003
	.80	**.813 ± .003**	.815 ± .003	.818 ± .003	.817 ± .003	.817 ± .003	**.870 ± .003**	.870 ± .003	.869 ± .003	.869 ± .003	.869 ± .003
	.75	**.766 ± .003**	.768 ± .003	.769 ± .003	.769 ± .003	.768 ± .003	**.884 ± .003**	.883 ± .003	.883 ± .003	.883 ± .003	.883 ± .003
CatsVsDogs	.99	.997 ± .001	.992 ± .002	.992 ± .002	**.991 ± .002**	**.990 ± .002**	.971 ± .003	.971 ± .003	.972 ± .003	.972 ± .003	**.972 ± .003**
	.95	.976 ± .003	.963 ± .003	.960 ± .003	**.950 ± .004**	.948 ± .004	.978 ± .003	.980 ± .003	**.982 ± .002**	.982 ± .002	.982 ± .002
	.90	.956 ± .003	.926 ± .004	.922 ± .004	**.901 ± .005**	.889 ± .005	.983 ± .002	.987 ± .002	.989 ± .002	.991 ± .002	**.992 ± .002**
	.85	.938 ± .004	.884 ± .005	.889 ± .005	**.855 ± .005**	.793 ± .006	.988 ± .002	.992 ± .002	.993 ± .002	.994 ± .002	**.994 ± .002**
	.80	.923 ± .004	.833 ± .006	.846 ± .005	**.774 ± .006**	.718 ± .007	.990 ± .002	.994 ± .002	.994 ± .002	.994 ± .002	**.995 ± .002**
	.75	.909 ± .005	**.775 ± .006**	.794 ± .006	.691 ± .007	.658 ± .007	.991 ± .002	.994 ± .002	.994 ± .002	.995 ± .002	**.996 ± .002**
	#	15/48	12/48	12/48	15/48	16/48	32/48	19/48	19/48	10/48	23/48

Table 2: Performance metrics for SCROSS by varying K (1,000 bootstrap runs over the test set, results as mean ± stdev).

to each other when considering zero or minimal rejection area. Other base classifiers, including the ResNet architecture adopted in SELNET and SAT, are also experimented, and results will be described later on.

Let us now turn on the result in Table 1. Regarding empirical coverage, SCROSS is the closest to target coverage in 19 out of 48 cases, followed by SELNET (17 out of 48), SAT (13 out of 48) and PLUGIN (11 out of 48). The largest violations occur for the CSDS1 and CSDS2 datasets, with SAT performing the best in such two cases. A possible reason is the high imbalance of the two datasets (12% and 2% of positive rate, respectively), which leads to poor extreme quantile estimation. On the other hand, SAT performs poorly on Lending, GiveMe and UCICredit, with considerable viola-

		Empirical Coverage				Empirical Selective Accuracy			
	c	**Logistic**	**ResNet**	**RandomForest**	**XGBoost**	**Logistic**	**ResNet**	**RandomForest**	**XGBoost**
Adult	.99	.989 ± .001	**.991 ± .001**	.989 ± .001	.992 ± .001	.850 ± .003	.847 ± .003	.841 ± .004	**.872 ± .003**
	.95	**.950 ± .002**	.949 ± .002	**.950 ± .002**	**.951 ± .002**	.863 ± .003	.861 ± .003	.854 ± .004	**.886 ± .003**
	.90	**.901 ± .003**	.895 ± .003	**.899 ± .003**	.903 ± .003	.879 ± .003	.877 ± .003	.871 ± .003	**.901 ± .003**
	.85	**.851 ± .003**	.847 ± .003	.857 ± .003	.857 ± .003	.894 ± .003	.893 ± .003	.885 ± .003	**.916 ± .003**
	.80	**.800 ± .004**	**.800 ± .004**	.808 ± .004	**.801 ± .004**	.910 ± .003	.906 ± .003	.899 ± .003	**.933 ± .003**
	.75	**.752 ± .004**	.753 ± .004	.755 ± .004	**.749 ± .004**	.923 ± .003	.921 ± .003	.915 ± .003	**.949 ± .003**
Lending	.99	**.992 ± .001**	.994 ± .001	.996 ± .001	.994 ± .001	.849 ± .001	.885 ± .001	.877 ± .001	**.903 ± .001**
	.95	**.958 ± .001**	.969 ± .001	.970 ± .001	.970 ± .001	.862 ± .001	.894 ± .001	.887 ± .001	**.913 ± .001**
	.90	**.917 ± .001**	.937 ± .001	.927 ± .001	.939 ± .001	.878 ± .001	.907 ± .001	.904 ± .001	**.927 ± .001**
	.85	**.878 ± .001**	.906 ± .001	.893 ± .001	.909 ± .001	.894 ± .001	.919 ± .001	.918 ± .001	**.941 ± .001**
	.80	**.842 ± .001**	.875 ± .001	.861 ± .001	.880 ± .001	.911 ± .001	.931 ± .001	.933 ± .001	**.955 ± .001**
	.75	**.809 ± .001**	.845 ± .001	.824 ± .001	.853 ± .001	.927 ± .001	.944 ± .001	.949 ± .001	**.966 ± .001**
GiveMe	.99	.991 ± .001	.999 ± .001	**.991 ± .001**	.990 ± .001	.936 ± .002	.934 ± .002	.939 ± .002	**.941 ± .002**
	.95	**.949 ± .002**	.752 ± .003	**.952 ± .002**	.949 ± .002	.943 ± .002	.949 ± .002	.953 ± .002	**.955 ± .002**
	.90	**.901 ± .002**	.710 ± .003	.905 ± .002	**.902 ± .002**	.946 ± .002	.952 ± .002	.962 ± .001	**.966 ± .001**
	.85	**.849 ± .002**	.653 ± .003	.858 ± .002	**.849 ± .002**	.949 ± .002	.956 ± .002	.969 ± .001	**.973 ± .001**
	.80	**.799 ± .003**	.586 ± .003	**.801 ± .003**	**.799 ± .003**	.952 ± .002	.958 ± .002	.974 ± .001	**.978 ± .001**
	.75	**.750 ± .003**	.525 ± .003	.767 ± .003	**.749 ± .003**	.955 ± .002	.959 ± .002	.976 ± .001	**.981 ± .001**
UCICredit	.99	**.992 ± .002**	.994 ± .001	.995 ± .001	**.989 ± .002**	.787 ± .005	**.817 ± .005**	.811 ± .005	.808 ± .005
	.95	**.952 ± .003**	.957 ± .003	.957 ± .003	.955 ± .003	.801 ± .005	**.826 ± .005**	.823 ± .005	.818 ± .005
	.90	**.898 ± .004**	.915 ± .004	.906 ± .004	**.903 ± .004**	.823 ± .005	.837 ± .005	**.838 ± .005**	.836 ± .005
	.85	**.852 ± .005**	.907 ± .004	**.850 ± .005**	**.851 ± .005**	.841 ± .005	.839 ± .005	**.852 ± .005**	.850 ± .005
	.80	**.798 ± .005**	.795 ± .005	**.800 ± .005**	.797 ± .005	.852 ± .005	.861 ± .005	.859 ± .005	**.861 ± .005**
	.75	.740 ± .005	**.748 ± .005**	.759 ± .006	.745 ± .006	.858 ± .005	.871 ± .005	.865 ± .005	**.872 ± .005**
CSDS1	.99	**.986 ± .001**	.984 ± .001	.985 ± .001	.980 ± .001	.861 ± .002	.861 ± .002	.854 ± .002	**.862 ± .002**
	.95	.923 ± .001	.932 ± .001	**.932 ± .001**	.921 ± .001	.873 ± .002	.872 ± .002	.865 ± .002	**.874 ± .002**
	.90	.855 ± .002	**.878 ± .002**	.871 ± .002	.851 ± .002	.882 ± .002	.880 ± .002	.873 ± .002	**.884 ± .002**
	.85	.793 ± .002	.809 ± .002	**.820 ± .002**	.792 ± .002	**.891 ± .002**	.889 ± .002	.879 ± .002	**.891 ± .002**
	.80	.736 ± .002	.749 ± .002	**.765 ± .002**	.735 ± .002	.896 ± .002	.896 ± .002	.885 ± .002	**.897 ± .002**
	.75	.684 ± .002	.691 ± .002	**.700 ± .002**	.681 ± .002	.902 ± .002	.901 ± .002	.889 ± .002	**.903 ± .002**
CSDS2	.99	.978 ± .002	.968 ± .002	.980 ± .002	**.984 ± .002**	.983 ± .002	**.983 ± .002**	.983 ± .002	.982 ± .002
	.95	.902 ± .003	.865 ± .003	.926 ± .003	**.928 ± .003**	.984 ± .002	**.985 ± .002**	.984 ± .002	.983 ± .002
	.90	.819 ± .004	.760 ± .004	.853 ± .004	**.860 ± .004**	.985 ± .002	**.986 ± .002**	.985 ± .002	.984 ± .002
	.85	.744 ± .004	.664 ± .005	**.798 ± .004**	.794 ± .004	.986 ± .002	**.987 ± .002**	.986 ± .002	.984 ± .002
	.80	.672 ± .005	.584 ± .005	.724 ± .004	**.729 ± .005**	.986 ± .002	**.988 ± .002**	.986 ± .002	.984 ± .002
	.75	.605 ± .005	.508 ± .005	.619 ± .005	**.669 ± .005**	.986 ± .002	**.988 ± .002**	.987 ± .002	.985 ± .002
CSDS3	.99	**.990 ± .001**	**.990 ± .001**	.992 ± .001	.991 ± .001	.811 ± .003	.808 ± .003	.810 ± .003	**.812 ± .003**
	.95	**.951 ± .002**	**.949 ± .002**	.963 ± .002	.953 ± .002	**.823 ± .003**	.820 ± .003	.818 ± .003	.823 ± .003
	.90	**.901 ± .002**	.896 ± .003	.918 ± .002	.903 ± .002	**.838 ± .003**	.835 ± .003	.831 ± .003	**.838 ± .003**
	.85	.847 ± .003	.845 ± .003	.860 ± .003	**.852 ± .003**	.854 ± .003	.851 ± .003	.847 ± .003	**.855 ± .003**
	.80	.795 ± .003	**.795 ± .003**	.816 ± .003	**.805 ± .003**	**.869 ± .003**	.865 ± .003	.859 ± .003	.868 ± .003
	.75	.744 ± .003	**.751 ± .003**	.772 ± .003	.756 ± .003	**.882 ± .003**	.876 ± .003	.873 ± .003	**.882 ± .003**
	#	25/42	8/42	12/42	17/42	5/42	7/42	2/42	30/42

Table 3: Performance metrics for SCROSS with different base classifiers over tabular datasets $k = 5$, and 1,000 bootstrap runs over the test set, results as mean ± stdev.

tions from target coverages. Regarding selective accuracy, SCROSS performs better in 27 out of 48 cases. PLUGIN obtains performances close to SCROSS on all the tabular datasets. SELNET and SAT obtain better performances on the image dataset (in 5 out of 6 target coverages): this is partly due to the fact that SELNET and SAT cover consider-

ably fewer instances than the target coverage. Regarding the elapsed training time, PLUGIN is, as one would expect, the fastest method. SCROSS require K times the elapsed time of PLUG-IN. Notice, however, that the K-fold approach of SCROSS can be easily parallelized. Deep learning methods require extensive training time for tabular data. A special mention is needed for the image dataset, where the additional training of a sub-network for the selective function in SELNET and SAT is moderate if compared to the training of the sub-network for the classifier (as in PLUGIN). In such a case, SCROSS is the slowest approach since it requires training K models. Finally, notice that, since SCROSS, PLUGIN, and SAT can be trained only once for all the target coverages, the reported training time is insensitive of c. SELNET, instead, needs specific training for each target coverage as the loss function directly takes it into account. *In summary*, for tabular data, SCROSS outperforms its direct model-agnostic competitor PLUGIN as per coverage and accuracy, and with an elapsed training time that is, in absolute terms, moderate. Concerning SELNET and SAT, there is a clear improvement as per selective accuracy and elapsed training time, while not improving coverage over imbalanced datasets. For image data, SCROSS shows better empirical coverage, but worse selective accuracy and elapsed training time.

Impact of K in SCROSS. We investigate how the performances of SCROSS depend on the number K of folds in Algorithm 1. Recall that so far we set $K = 5$. Table 2 shows the empirical coverage and selective accuracy for different values of K. We omit the elapsed training time as it monotonically increases with K. Results show that the empirical coverage and selective accuracy are contrasting objectives, as one would expect. $K = 7$ and $K = 10$ reach the best coverages, partly mitigating the low empirical coverage problem for the imbalanced datasets CSDS1 and CSDS2. Instead, for CatsVsDogs, smaller values of K achieve minimal target coverage, while larger values do not. Empirical selective accuracy is almost insensitive to K for tabular datasets, except for CSDS2 and CSDS3, for which lower K's are preferable. On the contrary, larger K's perform better for the image dataset. *In summary*, the default value $K = 5$ for SCROSS trades off the contrasting metrics of empirical coverage, selective accuracy, and elapsed training time.

Impact of base classifier in SCROSS. Let us now consider the impact of different base classifiers in the SCROSS approach. We experiment with Logistic Regression (Logistic), ResNet (ResNet) trained using cross-entropy loss, Random Forest (RandomForest), and XGBoost (XGBoost). Results are shown in Table 3. The empirical coverage trend at the variation of the target coverage c appears uniform over the base classifiers, except for a consistently large violation for ResNet in the GiveMe and in the CSDS2 datasets compared to the other base classifiers. The simple Logistic model shows the best empirical coverage (25 out of 42 times). As expected, empirical selective accuracy shows an opposite trend, with the more complex base classifiers ResNet and XGBoost showing the best performances. *In summary*, SCROSS performs consistently w.r.t. the base classifier, extending its predictive accuracy to the case of the reject option. Deviations from the target coverage are lower for simpler models.

Conclusions

We proposed a simple-to-implement, effective, model-agnostic heuristics to lift a probabilistic binary classifier to a selective classifier with a minimum target coverage. The approach is supported by a theoretical result on quantile estimation, which shows that using a linear combination of quantile estimates over the full sample and estimates on subsamples provides a better second-order estimator. Our approach does not require a separate validation set, as the PLUGIN method. Experiments show that SCROSS (with LightGBM as a base classifier) outperforms state-of-the-art DNN-specific methods, such as SELNET and SAT, on tabular data.

A few issues remain open for future research. First, we will investigate non-additive loss functions, such as AUC and Gini (see, e.g., (Shen, Yang, and Gao 2020)), which are widely used in evaluating performance in imbalanced scenarios. Second, (Jones et al. 2021) point out that selective classification may amplify unfair decisions, for which fair-selective algorithms are required (Lee et al. 2021). A recent work explicitly deals with unfairness in selective classification by regularising the loss function (Schreuder and Chzhen 2021). We will consider how to lift such a regularization approach to SCROSS. Third, our method assumes that the higher the confidence, the lower the loss. If this assumption fails, using a single bound on the softmax confidence function may not be appropriate. In such a case, we will investigate the performance of a piecewise selection function. Finally, we intend to explore the intersection between selective classification and eXplainable AI (XAI) (Minh et al. 2022; Guidotti et al. 2019) in order to provide intelligible explanations of the reasons for abstaining. This could be valuable information, e.g., for informing the human decision-maker taking over the decision.

Acknowledgements

This paper is part of the project FINDHR that has received funding from the European Union's Horizon Europe research and innovation program under g.a. No 101070212. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the EU. Neither the EU nor the granting authority can be held responsible for them.

References

Angelopoulos, A. N.; and Bates, S. 2021. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. *CoRR*, abs/2107.07511.

Barddal, J. P.; Loezer, L.; Enembreck, F.; and Lanzuolo, R. 2020. Lessons learned from data stream classification applied to credit scoring. *Expert Syst. Appl.*, 162: 113899.

Bartlett, P. L.; and Wegkamp, M. H. 2008. Classification with a Reject Option using a Hinge Loss. *J. Mach. Learn. Res.*, 9: 1823–1840.

Chernozhukov, V.; Chetverikov, D.; Demirer, M.; Duflo, E.; Hansen, C.; Newey, W.; and Robins, J. 2018. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1): C1–C68.

Chow, C. K. 1970. On optimum recognition error and reject tradeoff. *IEEE Trans. Inf. Theory*, 16(1): 41–46.

Cortes, C.; DeSalvo, G.; and Mohri, M. 2016. Boosting with Abstention. In *NIPS*, 1660–1668.

Dastile, X.; Çelik, T.; and Potsane, M. 2020. Statistical and machine learning models in credit scoring: A systematic literature survey. *Appl. Soft Comput.*, 91: 106263.

Demsar, J. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.*, 7: 1–30.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. *Public Repository*.

El-Yaniv, R.; and Wiener, Y. 2010. On the Foundations of Noise-free Selective Classification. *J. Mach. Learn. Res.*, 11: 1605–1641.

European Parliament and the Council. 2021. Regulation of the European Parliament and of the Council laying down harmonised rules on Artificial Intelligence (Artificial Intelligence Act) and amending certain Union legislative acts. *2021/0106(COD)*.

Franc, V.; and Průša, D. 2019. On discriminative learning of prediction uncertainty. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, 1963–1971. PMLR.

Fumera, G.; and Roli, F. 2002. Support Vector Machines with Embedded Reject Option. In *SVM*, volume 2388 of *Lecture Notes in Computer Science*, 68–82. Springer.

Geifman, Y.; and El-Yaniv, R. 2017. Selective Classification for Deep Neural Networks. In *NIPS*, 4878–4887.

Geifman, Y.; and El-Yaniv, R. 2019. SelectiveNet: A Deep Neural Network with an Integrated Reject Option. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, 2151–2159. PMLR.

Gorishniy, Y.; Rubachev, I.; Khrulkov, V.; and Babenko, A. 2021. Revisiting Deep Learning Models for Tabular Data. In *NeurIPS*, 18932–18943.

Grinsztajn, L.; Oyallon, E.; and Varoquaux, G. 2022. Why do tree-based models still outperform deep learning on tabular data? *CoRR*, abs/2207.08815.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2019. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.*, 51(5): 93:1–93:42.

Hendrickx, K.; Perini, L.; der Plas, D. V.; Meert, W.; and Davis, J. 2021. Machine Learning with a Reject Option: A survey. *CoRR*, abs/2107.11277.

Herbei, R.; and Wegkamp, M. H. 2006. Classification with reject option. *Can. J. Stat.*, 34(4): 709—-721.

Herbold, S. 2020. Autorank: A Python package for automated ranking of classifiers. *J. of Open Source Software*, 5(48): 2173.

Huang, L.; Zhang, C.; and Zhang, H. 2020. Self-Adaptive Training: beyond Empirical Risk Minimization. In *NeurIPS*.

Jones, E.; Sagawa, S.; Koh, P. W.; Kumar, A.; and Liang, P. 2021. Selective Classification Can Magnify Disparities Across Groups. In *ICLR*. OpenReview.net.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *NIPS*, 3146–3154.

Knight, K.; and Bassett, G. W. 2003. Second order improvements of sample quantiles using subsamples. *Unpublished*.

Lee, J. K.; Bu, Y.; Rajan, D.; Sattigeri, P.; Panda, R.; Das, S.; and Wornell, G. W. 2021. Fair Selective Classification Via Sufficiency. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, 6076–6086. PMLR.

Liu, Z.; Wang, Z.; Liang, P. P.; Salakhutdinov, R.; Morency, L.; and Ueda, M. 2019. Deep Gamblers: Learning to Abstain with Portfolio Theory. In *NeurIPS*, 10622–10632.

Minh, D.; Wang, H. X.; Li, Y. F.; and Nguyen, T. N. 2022. Explainable artificial intelligence: a comprehensive review. *Artif. Intell. Rev.*, 55(5): 3503–3568.

Okati, N.; De, A.; and Gomez-Rodriguez, M. 2021. Differentiable Learning Under Triage. In *NeurIPS*, 9140–9151.

Pietraszek, T. 2005. Optimizing abstaining classifiers using ROC analysis. In *ICML*, volume 119 of *ACM International Conference Proceeding Series*, 665–672. ACM.

Rajkomar, A.; Oren, E.; Chen, K.; Dai, A. M.; Hajaj, N.; Hardt, M.; Liu, P. J.; Liu, X.; Marcus, J.; Sun, M.; et al. 2018. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1): 1–10.

Schreuder, N.; and Chzhen, E. 2021. Classification with abstention but without disparities. In *UAI*, volume 161 of *Proceedings of Machine Learning Research*, 1227–1236. AUAI Press.

Shafer, G.; and Vovk, V. 2008. A Tutorial on Conformal Prediction. *J. Mach. Learn. Res.*, 9: 371–421.

Shen, S.; Yang, B.; and Gao, W. 2020. AUC Optimization with a Reject Option. In *AAAI*, 5684–5691. AAAI Press.

Shwartz-Ziv, R.; and Armon, A. 2022. Tabular data: Deep learning is not all you need. *Inf. Fusion*, 81: 84–90.

Tortorella, F. 2005. A ROC-based reject rule for dichotomizers. *Pattern Recognit. Lett.*, 26(2): 167–180.

Wing, J. M. 2021. Trustworthy AI. *Commun. ACM*, 64(10): 64–71.

Yeh, I.; and Lien, C. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Syst. Appl.*, 36(2): 2473–2480.