SCUOLA
NORMALE
SUPERIORE

Class of Science

Ph.D. in Data Science

XXXVI cycle

Topics in Selective Prediction

Scientific Disciplinary Area **INF/01**

Candidate

Andrea Pugnana

Supervisors

Prof. Salvatore Ruggieri

Prof.ssa Francesca Chiaromonte

Academic year 2023-2024

Thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy (Ph.D.) in Data Science (XXXVI cycle)

Abstract

With the increasing deployment of Machine Learning (ML) models in many socially-sensitive predictive tasks, there is a growing demand for reliable and trustworthy predictions. For example, the Artificial Intelligence (AI) Act of the European Union rules that "[h]igh-risk AI systems shall be designed and developed in such a way that they achieve, in the light of their intended purpose, an appropriate level of accuracy [and] robustness".

One way to accomplish these requirements is to allow ML models to abstain from making a prediction when there is a high risk of making an error. Among the different approaches that build abstaining systems, the *selective prediction* framework adds a selection mechanism to the ML model, which *selects* those instances for which the model will provide a prediction. Selective prediction aims at balancing the fraction of abstained instances versus the improvement in predictive performance on the selected ones.

In this thesis, we address three main challenges faced by current approaches in selective prediction. *First*, we tackle the limitations of model-specific approaches by proposing model-agnostic heuristics for both the selective classification and the selective regression tasks. *Second*, we depart from distributive metric losses (such as accuracy), which are not well-suited in the case of imbalanced classes, by devising an approach to optimize the AUC metrics. The approach is used in a deployed credit scoring system for labeling predicted scores with uncertainty ratings. *Third*, we clarify the relative strengths and weaknesses of the state-of-the-art approaches, also including those introduced in this thesis, by an extensive and throughout empirical evaluation.

The effectiveness of the methods proposed in the thesis is supported by theoretical foundations and experimental results.

Acknowledgments

“Chi non ha testa, ha gambe” (the one who has no brain must have legs) is one of the many great wisdom pills from my Grandma. She often uses it when someone, i.e. 99% of the time my Grandpa, forgets something in the garden and has to go back to take it. I have always thought that this would be a perfect motto to describe the PhD journey, as perseverance is the only way to go through the challenges of a PhD.
Despite this mandatory premise, the PhD is mainly about the people you meet. Hence, I would like to thank all the people who were part of the journey. First, I would like to deeply thank my supervisor prof. Salvatore Ruggieri. Salvatore has been inspiring and supportive since my first day of PhD and he is a role model for me. If I ever managed to become a professor one day, I would hope my students could feel as lucky as I am to have such a great mentor as him.
Second, I want to thank my other supervisor prof. Francesca Chiaromonte, who always managed to find some time to hear my worries and tell me the right words at the right time. Thanks a lot Francesca.
Then I would like to thank prof. Jesse Davis, who hosted me at KU Leuven for my visiting period abroad. Of course, I have to thank all the amazing people I met there - too many Italians to be fair - who made me feel at home: Eleonora, Gabriele, Laurens, Lorenzo (il giovane), Lorenzo (il Magnifico), Luca (il bimbo).
I also wish to thank all the KDD Lab people for their support, and I want to thank all the people I collaborated with, i.e. Rita, Laura, Dino, Joao, Franco, Carlos, Dan, and Riccardo: each of you taught me something, and made me a better researcher.
A special thanks to Prof. Anna Monreale, who made this journey possible by believing in me when I was still a working student, to Jose, with whom we spent hours exchanging precious feedback and discussing our research (we will manage to merge them eventually), to Mattia, who is always willing to help non-computer scientists (as me), to Martina, to make the burden of commuting even enjoyable and to the bouldering team (Isacco, Marta, Eleonora). To conclude, I would like to thank my family: my parents, who always let me [chose] whatever I want[ed] to do, even when I was younger" and supported me unconditionally; my sister, who is the best example of never giving up, and my grandparents, who have always found a way to make me feel better.
A final thanks to my other side of the world, Caroline, for dealing with me every day and walking by my side to the next Chapter.

Contents

1 Introduction **1**
1.1 Thesis Goals and Contributions 2
1.2 Outline of the Thesis 5
1.3 List of Publications 5

2 Background **7**
2.1 General background 7
2.2 Ambiguity Rejection 8
2.2.1 Selective Prediction 8
2.2.2 Learning to Reject 12
2.3 Novelty Rejection 14
2.4 Concerns from abstaining classifiers 15
2.5 Related Work 17
2.5.1 Uncertainty Quantification 17
2.5.2 Conformal Prediction 19
2.5.3 Learning to Defer 20

3 A Model-Agnostic Heuristics for Selective Classification **21**
3.1 Introduction 21
3.2 Model-Agnostic Heuristics 22
3.2.1 Preliminaries 22
3.2.2 The SCross algorithm 23
3.3 Experimental Evaluation 24
3.3.1 Research Questions 24
3.3.2 Experimental Settings 25
3.3.3 Experimental Results 27
3.4 Conclusions 36

4 Model-Agnostic Heuristics for Selective Regression **37**
4.1 Introduction 37
4.2 Current Limitations 38
4.3 Model-Agnostic Heuristics 38
4.3.1 Cross-fitting for Selective Regression 39
4.3.2 Bootstrap for Selective Regression 39
4.4 Experimental Evaluation 41
4.4.1 Research Questions 41
4.4.2 Experimental Settings 42

4.4.3 Experimental Results 45
4.5 Conclusions 48

5 Selective Classification for AUC 52
5.1 Introduction 52
5.2 Background 53
5.3 Methodology 54
5.4 Algorithms for AUC improvement 60
5.4.1 The PluginAUC algorithm 60
5.4.2 The AUCross algorithm 62
5.5 Experimental Evaluation 63
5.5.1 Research Questions 63
5.5.2 Experimental Setting 63
5.5.3 Experimental Results 65
5.6 Conclusions 69

6 A Case Study: the LSP system 70
6.1 Introduction 70
6.2 Related Work 72
6.3 Credit Scoring Background 73
6.4 The Leasing Industry 76
6.5 The LSP Framework 76
6.5.1 Architecture, workflow, inputs and outputs 77
6.5.2 The scoring model 81
6.5.3 Selective classification for self-assessment 82
6.5.4 Explanation methods 85
6.5.5 Other features 86
6.6 Experimental Evaluation 88
6.6.1 Experimental Setting 88
6.6.2 Experimental Results 91
6.7 Conclusions 93

7 Deep Neural Network Benchmarks for Selective Classification 95
7.1 Introduction 95
7.2 Baselines 96
7.2.1 Learn-to-Abstain Methods 96
7.2.2 Learn-to-Select Methods 98
7.2.3 Score-based Methods 99
7.3 Experimental Evaluation 101
7.3.1 Research Questions 101
7.3.2 Experimental Setting 102
7.3.3 Experimental Results 104
7.3.4 Discussion 111
7.4 Conclusions 112

8 Conclusions **113**
8.1 Final Remarks . . . 113
8.2 Future Work . . . 114

List of Figures **115**

List of Tables **118**

A Appendix to Model-Agnostic Heuristics for Selective Regression **140**
A.1 Data Details . . . 140
A.2 Results for Mean Absolute Error . . . 140

B Appendix to Selective Classification for AUC **146**
B.1 Proofs . . . 146
B.2 Experimental Evaluation . . . 147
B.2.1 Datasets description . . . 147
B.2.2 Baselines details . . . 147
B.2.3 Other experimental results . . . 148
B.2.4 Results for different classifiers . . . 149
B.2.5 Results varying number of folds . . . 149

C Appendix to Deep Neural Network Benchmarks for Selective Classification **163**
C.1 Experimental Details . . . 163
C.1.1 Datasets . . . 163
C.1.2 Hyperparameter Settings . . . 165
C.2 Additional Experimental Results . . . 176
C.2.1 Q1: Results by Dataset Type . . . 176
C.2.2 Q5: Additional Experiments . . . 176
C.2.3 Dataset-level Results . . . 180

D Other Publications **190**

Chapter 1

Introduction

In the last decade, the adoption of Machine Learning (ML) models has been increasing in a plethora of application domains. Together with the impressive achievements of ML, many concerns have been raised about the potential harms to individuals and society at large, especially when these systems are used for socially sensitive scenarios. E.g. ML models are often unintelligible to humans, or "black boxes" [1]. This may hide many forms of biases embedded in the models [2]. Making ML trustworthy means tackling multiple dimensions, including reliability, ethical adherence, transparency, and the capacity to build trust in users and stakeholders [3].

Trustworthiness is one of the main goals of the recent EU Regulatory framework on Artificial Intelligence (AI) [4], which identifies four different levels of risk associated with AI usage: minimal, limited, high, and unacceptable risk. AI systems categorized as unacceptable are banned: these include applications that harm the safety of people and human rights, e.g. social scoring by a government. The three remaining categories have to meet increasingly binding obligations before being deployed. In particular, "[h]igh-risk AI systems shall be designed and developed in such a way that they achieve, in the light of their intended purpose, an appropriate level of accuracy [and] robustness". High-risk AI systems pertain to socially sensitive domains, including: healthcare, where predictions might be used to determine treatments [5]; justice, where predictions can assess the risk of recidivism [6]; hiring, where predictions can determine rankings of candidates or explain their turnover intention [7, 8]; credit scoring, where predictions are routinely used to estimate the probability of repaying a debt [9].

In all such high-risk contexts, the goal is to reduce AI systems' error-proneness because AI errors can have severe consequences. For example, consider a bank that uses ML to score the credit risk of loan applications. In such a setting, an error in predictions could translate into money loss for the bank, or to an unjust denial of credit to applicants.

A long-standing approach to reduce errors in predictions consists of allowing ML models to refrain from making a prediction when not confident enough. This idea dates back to the 1970s, with the seminal paper by Chow [10], which introduced the *reject option*. Such a strategy is inherent in human reasoning when facing an unknown phenomenon. For example, human bankers who are

unsure about a specific loan application do not (have to) provide an answer as soon as they are asked. Indeed, they may require additional financial documents to verify the loan's feasibility or they may ask for expert consultation. This approach allows for minimizing the risk of a wrong evaluation.
Likewise, allowing ML models to predict only when confident enough helps mitigate the risks of wrong predictions. On the one hand, the reject option idea allows for achieving better predictive performances over the accepted instances. On the other hand, rejected instances can be dealt with in other ways. Human experts can be involved in the loop to oversee difficult instances, e.g. a banker can oversee the difficult-to-evaluate loan applications. Alternatively, the prediction task can be demanded to more complex ML models, possibly using additional and costly-to-compute features. Moreover, predictors that can abstain increase trust in AI systems [11], which is in the direction of satisfying the ethical and legal concerns we mentioned in the beginning.
To build abstaining predictors, two main approaches emerge from the literature: *ambiguity rejection* and *novelty rejection* [12]. The former focuses on rejecting the instances for which the ML model is uncertain. The latter avoids making predictions on instances unlikely to occur given the statistical distribution of training data. To make this distinction clearer, let us consider the example provided in Figure 1.1, where we have trained a Support Vector Machine (SVM) to distinguish between pink diamond-shaped points and blue circle-shaped ones. The SVM draws a hyperplane in the feature space that separates the two kinds of points, i.e. the black dashed line. This hyperplane determines which prediction will be provided for each point: if an instance falls above the black dashed line, it will be considered as a blue circle-shaped, while it will be considered a pink diamond-shaped otherwise. However, there are still some points that would be wrongly classified. Ambiguity rejection addresses this concern, by avoiding providing predictions for those instances that are close to the decision boundary, as highlighted in Figure 1.2. In this case, we would avoid predicting on the instances falling in the orange area delimited by the blue dotted lines. On the other hand, novelty rejection can be applied to instances for which the models has not been sufficiently trained about, as shown in Figure 1.3. Here abstention occurs for black-crossed instances, which are far away from the distribution of the training data.

1.1 Thesis Goals and Contributions

In this thesis, we will focus on *selective prediction* [13], which is the most popular framework of ambiguity rejection. Selective prediction extends ML models with a selection function (reject option/strategy) to determine whether or not a prediction should be accepted. The selection function *selects* the instances where the model is confident enough that it is not making mistakes. If a sufficiently high-level confidence is not reached, the selective predictor abstains from providing a forecast – e.g., a risk score in a credit evaluation application or a diagnosis in a healthcare application. A trade-off arises between abstention and model performance: the more we allow for a model to abstain, the

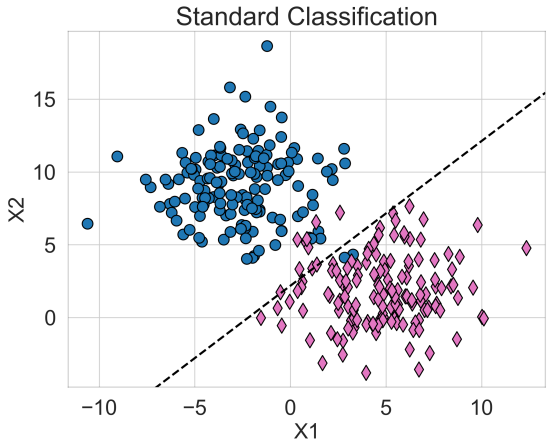


Figure 1.1: Canonical SVM Classifier. Instances above the black line are predicted as blue circle-shaped instances, while those below are predicted as pink diamond-shaped.

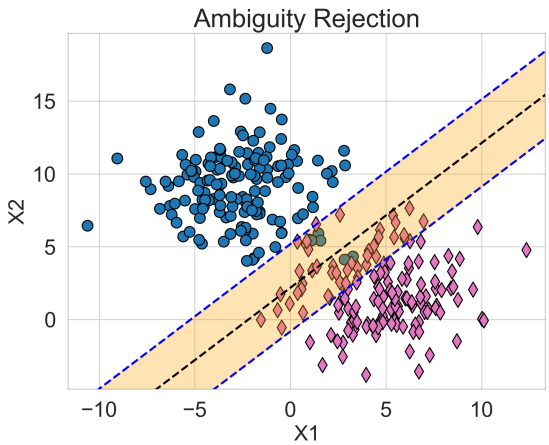


Figure 1.2: Ambiguity Rejection example. Abstention occurs for those instances within the orange area delimited by the two blue dotted lines, i.e. those close to the decision boundary.

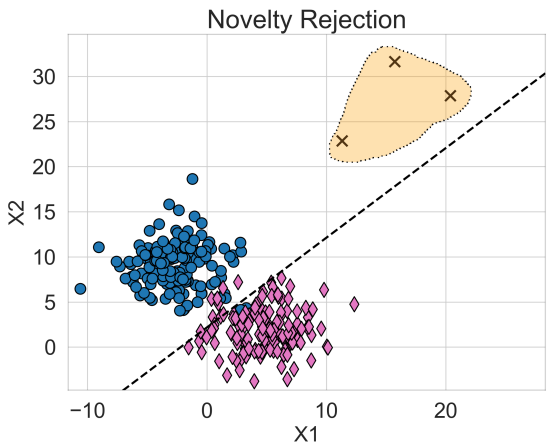


Figure 1.3: Novelty Rejection example. Abstention occurs for the cross-marked instances within the orange area, i.e. those far from the original data.

more we will expect correct predictions over selected instances [13].
Existing approaches in selective prediction suffer from a few shortcomings, some of which will be addressed in this thesis. More specifically, we envision the following research requirements:

Req. 1: several selective prediction methods are model-specific, e.g., they work only for neural networks [14, 15, 16, 17, 18, 19, 20]. This setting translates into selective predictors that share the strengths and weaknesses of the base models. Hence, model-agnostic approaches are highly sought, because, instead, they can be adapted to different base models and settings;

Req. 2: the canonical choice in evaluating selective predictors is through distribution-based loss functions, where the loss is defined on the distribution of labels. The typical scenario considers accuracy over accepted instances, or selective accuracy [13]. However, there are cases where ranking-based loss functions, such as the Area Under the ROC Curve (AUC) [21], are more informative, e.g., when the classes are imbalanced. This is a common scenario in several high-risk domains (e.g., finance and healthcare). Thus, novel approaches to selective predictions are required to explicitly optimize for the AUC;

Req. 3: despite the appeal of the selective prediction approach, little attention has been devoted to its usage in real-life applications [11]. For instance, practitioners might be interested in understanding how to equip ML models with a reject option to allow for self-assessing uncertainty of predictions;

Req. 4 the empirical evaluation of the majority of approaches in the literature is limited. For instance, many papers account in experiments only a small number of baselines and/or focus on specific kinds of data [14, 15, 22, 16, 17, 18, 19, 20, 23]. A better understanding of the actual strengths and limitations of existing approaches is a useful to both researchers and practitioners.

In this thesis, we address the requirements above with the following contributions:

- regarding *Req. 1*, we propose a few model-agnostic heuristics:
  - for the selective classification task, we develop an approach called SCross, which is based on cross-fitting and an improved second-order quantile estimator;
  - for selective regression, we devise methods based on different sample-splitting strategies, which prevent the selective regressors to overfit;
- concerning *Req. 2*, we introduce the problem of improving the discriminative power of binary probabilistic classifiers once abstention is allowed. We propose a theoretical approach specifc for the AUC, and we devise two algorithms to implement such a strategy;

- regarding *Req. 3*, we present LSP, a deployed credit scoring system that is specific to the leasing industry. LSP addresses multiple objectives defined by various stakeholders. In particular, we use selective classification methods to embed the prediction of a ML model with a self-awareness assessment;

- we address *Req. 4* by benchmarking deep neural network approaches for selective classification. We extensively compare 18 baselines over 44 datasets and evaluate their strengths and weaknesses under different settings.

1.2 Outline of the Thesis

The thesis starts by introducing the framework of selective prediction in Chapter 2. We provide a brief yet comprehensive formal background and overview closely related research areas, pointing out similarities and differences.
In Chapter 3, we present the SCross methodology to perform selective classification, while in Chapter 4 we present the model-agnostic heuristics for selective regression. Both Chapters address *Req. 1*.
In Chapter 5, we tackle *Req. 2* by presenting the AUC-based selective classification framework.
In Chapter 6, we present the Leasing Score Prediction (LSP) project and how to use selective classification to self-assess uncertainty, answering to *Req. 3*.
In Chapter 7, we describe the benchmarking of deep neural network selective classifiers, addressing *Req. 4*.
To conclude, Chapter 8 summarizes the thesis's contributions and highlights some research directions for future works.

1.3 List of Publications

A brief summary of the thesis project was accepted for the AAAI Doctoral Consortium 2023 and published as:

- Andrea Pugnana (2023). Topics in selective classification. In *AAAI*, pages 16129-16130. AAAI Press, 2023.

The work behind Chapter 3 was published as:

- Andrea Pugnana and Salvatore Ruggieri. A Model-agnostic Heuristics for Selective Classification. In *AAAI*, pages 94619469. AAAI Press, 2023.

The work behind Chapter 5 was published as:

- Andrea Pugnana and Salvatore Ruggieri. AUC-based selective classification. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pages 24942514. PMLR, 2023.

The work behind Chapter 6 was published as:

- Giuseppe Cianci, Roberto Goglia, Riccardo Guidotti, Matteo Kapllaj, Roberto Mosca, Andrea Pugnana, Franco Riccotti, and Salvatore Ruggieri. Applied data science for leasing score prediction. In *IEEE Big Data*, pages 16871696. IEEE, 2023.

A preliminary version of Chapter 7 was published as:

- Andrea Pugnana, Lorenzo Perini, Jesse Davis, and Salvatore Ruggieri. Deep Neural Network Benchmarks for Selective Classification. In *Verifying Learning AI Systems (Verilearn) Workshop, ECAI, 2023.*

and an extended version is currently under consideration for journal publication.

Chapter 2

Background

This chapter introduces the main concepts used throughout the thesis and positions the selective prediction task within the related literature. In Section 2.1, we will introduce a standard notation and present the two main ways to perform abstention, i.e. ambiguity rejection and novelty rejection. Section 2.2 summarises how abstention can be performed within the ambiguity rejection framework, focusing mainly on the theoretical aspects of the selective prediction task. Section 2.3 presents the main methods to perform novelty rejection. We explore ethical concerns deriving from the reject option in Section 2.4. Finally, in the last Section 2.5, we briefly overview other fields closely related to abstaining predictors.

2.1 General background

Let $\mathcal{X}$ be an d-dimensional input space, $\mathcal{Y}$ be the target space and $P(\mathbf{X}, Y)$ be the probability distribution over $\mathcal{X} \times \mathcal{Y}$. Generally speaking, we will denote random variables with capital letters (e.g. $\mathbf{X}$), while we refer to their realizations using lowercase letters (e.g. $\mathbf{x}$). If $\mathcal{Y}$ takes a finite number of values - i.e. $\mathcal{Y} = \{1, \ldots, m\}$ - we deal with a classification problem, otherwise if $\mathcal{Y} = \mathbb{R}$ we deal with a regression task. Given a hypothesis space $\mathcal{H}$ of functions that maps $\mathcal{X}$ to $\mathcal{Y}$, the goal of a machine learning model (a learner) is to find the hypothesis $h \in \mathcal{H}$ (i.e., a classifier or a regressor) that minimizes the *risk*:

$$R(h) = \int_{\mathcal{X} \times \mathcal{Y}} l(h(\mathbf{x}), y) d\mathcal{P}(\mathbf{x}, y) = \mathbb{E}[l(h(\mathbf{X}), Y)], \tag{2.1}$$

where $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a user-specified loss function and $\mathcal{P}$ is the probability measure linked to the joint distribution P over the space $\mathcal{X} \times \mathcal{Y}$. Because $P(\mathbf{X}, Y)$ is generally unknown, it is typically assumed that we have access to an i.i.d. sample $\mathcal{T}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. This allows us to estimate an empirical counterpart of risk $R(h)$, called empirical risk:

$$\hat{R}(h \mid \mathcal{T}_n) = \frac{1}{n} \sum_{i=1}^{n} l(h(\mathbf{x}_i), y_i) \tag{2.2}$$

It can be used to learn a predictor $\hat{h}$[1], such that

$$\hat{h} \in \arg\min_{h \in \mathcal{H}} \hat{R}(h \mid \mathcal{T}_n) \tag{2.3}$$

Under the classification framework, we define the Bayes Optimal Classifier as:

$$h_B \in \arg\min_{\hat{y} \in \mathcal{Y}} \sum_{y \in \mathcal{Y}} P(Y = y | \mathbf{X} = \mathbf{x}) l(\hat{y}, y) \tag{2.4}$$

For example, in a binary classification setting (i.e. when $\mathcal{Y} = \{0, 1\}$), the Bayes Optimal classifier can be further characterized as

$$h_B(\mathbf{x}) = \mathbb{1}\{\eta(\mathbf{x}) \geq 1/2\}, \tag{2.5}$$

where $\eta(\mathbf{x}) = P(Y = 1 | \mathbf{X} = \mathbf{x})$. More generally, we will refer to the score $s_y : \mathcal{X} \to [0, 1]$ as a function that aims to estimate the conditional probability function, i.e. $s_y(\mathbf{x}) \approx P(Y = y | \mathbf{X} = \mathbf{x})$. In this case, the score is also known as a probabilistic classifier.
Because the learned model is prone to making mistakes, one can extend the canonical setting to include a selection mechanism that allows the predictor to refrain from offering a forecast for those instances likely to be misclassified. If the predictor does not provide a forecast for an instance, a label can be assigned in another manner such as by asking a user to check it by hand or employing another, more expensive model (e.g., one which requires more features, or which takes longer to make a prediction).

2.2 Ambiguity Rejection

As shown in Figure 2.1, within the Ambiguity Rejection setup, we can devise two main ways to learn abstaining predictors, i.e. selective prediction (SP) and learning to reject (LtR). In the next subsections, we will provide an overview of both frameworks.

2.2.1 Selective Prediction

Formally, a *selective predictor* is a pair (h, g) where h is a standard predictor and $g : \mathcal{X} \to \{0, 1\}$ is a *selection function* that determines whether h's prediction is provided or the model abstains (or rejects):

$$(h, g)(\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if } g(\mathbf{x}) = 1 \\ \text{abstain} & \text{otherwise.} \end{cases} \tag{2.6}$$

We refer to *selective classification* (SC) whenever h is a classifier: in this case, we call the selective predictor a *selective classifier*. We refer to *selective regression* (SR) when h is a regressor. The selective predictor employed in such

[1] We will use the terms model and predictor interchangeably in the rest of the work.

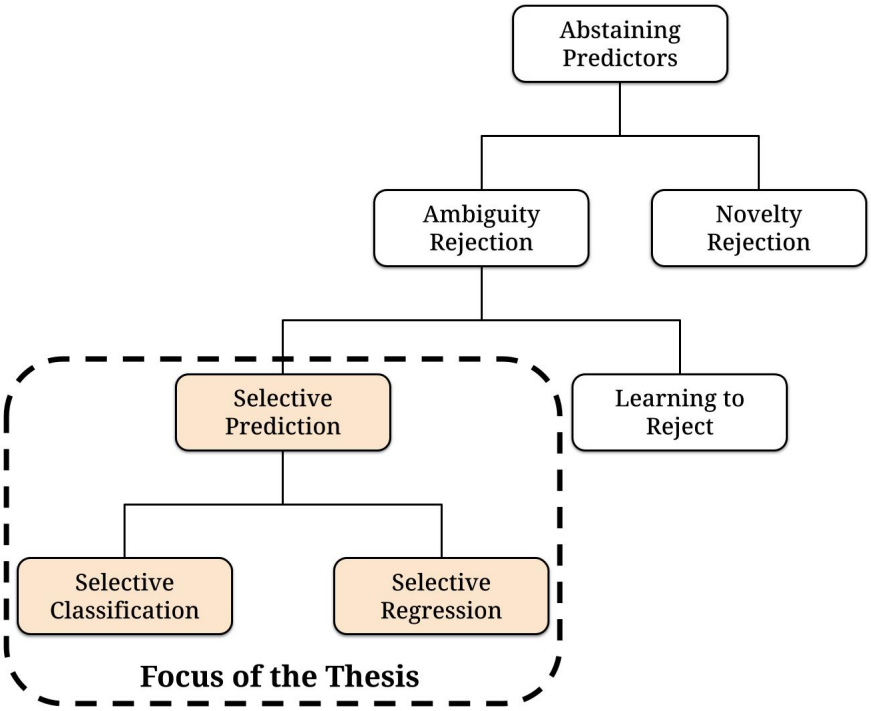


Figure 2.1: A hierarchy of methods to build abstaining predictors.

a task is called *selective regressor*. In practice, rather than directly learning the selection function in Eq. 2.6, one approximates it by (1) learning a *confidence function*[2] $\upsilon_h : \mathcal{X} \to [0, 1]$ (sometimes called soft selection [14]) that measures how likely it is that the predictor h is correct, and (2) setting a threshold $\tau \in [0, 1]$ that defines the minimum confidence for providing a prediction, yielding

$$g(\mathbf{x}) = \mathbb{1}\{\upsilon_h(\mathbf{x}) > \tau\} \tag{2.7}$$

To prevent the selective predictor from abstaining on too many (test) instances, SP methods also consider the *coverage* metric, which is defined as

$$\phi(g) = \int_{\mathcal{X}} g(\mathbf{x}) d\mathcal{P}_{\mathbf{X}}(\mathbf{x}) = \mathbb{E}[g(\mathbf{X})], \tag{2.8}$$

where $\mathcal{P}_{\mathbf{X}}$ is the probability measure associated to the marginal probability $P_{\mathbf{X}}$ over $\mathcal{X}$. In simpler words, the coverage is the expected probability mass of the non-rejected region. Non-rejected instances are commonly referred to as either *accepted* or *selected*, and we will use these terms interchangeably. Another core measure of the selective prediction framework is the risk over the accepted region, commonly called the *selective risk* and defined as:

$$R(h, g) = \frac{\mathbb{E}[l(h(\mathbf{X}), Y)g(\mathbf{X})]}{\phi(g)} = \mathbb{E}[l(h(\mathbf{X}), Y)|g(\mathbf{X}) = 1] \tag{2.9}$$

[2]A good confidence function υ_h should rank instances based on descending loss, i.e., if $\upsilon_h(\mathbf{x}_i) \leq \upsilon_h(\mathbf{x}_j)$ then $l(h(\mathbf{x}_i), y_i) \geq l(h(\mathbf{x}_j), y_j)$.

The inherent trade-off between coverage and risk can be summarized by a *risk-coverage curve* [13]. Moreover, such a trade-off allows framing the selective prediction task according to two different formulations: the bounded improvement model and the bounded abstention model [24].
In the bounded improvement model, the problem is formulated by fixing an upper bound ϵ - the *target risk* - to the selective risk and then looking for a selective predictor that maximizes coverage [14], i.e.:

Problem 1 (Bounded-improvement model)**.** *Given a target risk ϵ, an optimal selective predictor (h, g) parametrized by (θ^*, ψ^*) can be defined as:*

$$\arg\max_{\theta,\psi} \phi(g_\psi) \quad s.t. \quad R(h_\theta, g_\psi) \leq \epsilon \tag{2.10}$$

Conversely, in the bounded-abstention model, we fix a lower bound c for coverage (called *target coverage*) and then look for a selective predictor that minimizes selective risk [15], i.e.:

Problem 2 (Bounded-abstention model)**.** *Given a target coverage c, an optimal selective predictor (h, g) parameterized by θ^*, ψ^* can be defined as:*

$$\arg\min_{\theta,\psi} R(h_\theta, g_\psi) \quad s.t. \quad \phi(g_\psi) \geq c \tag{2.11}$$

The empirical counterpart of coverage and selective risk are respectively defined as the *empirical coverage*:

$$\hat{\phi}(g|\mathcal{T}_n) = \frac{\sum_{i=1}^n g(\mathbf{x}_i)}{n},$$

and the *empirical selective risk*:

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

These can be used to learn the selective predictor by plugging them in (2.10) or (2.11). Finally, we call *coverage-calibration* the post-training procedure of estimating the threshold τ in (2.7) for the target coverage c specified in Eq. 2.11. This is generally done by estimating the $(1-c) \cdot 100$-th percentile of the confidence function over a held-out calibration dataset.

Selective classification

Optimal selective classifiers have been mainly studied from a theoretical point of view. The main results for the bounded-improvement model come from the seminal paper by El-Yaniv and Wiener [13], who presented a comprehensive analysis to build an optimal selective classifier. Their analysis requires a noise-free environment, where they aim to maximize the coverage while ensuring that the selected risk is zero. The authors present a comprehensive analysis of the risk-coverage curves in the Probably Approximately Correct (PAC) framework, offering a detailed definition of both the lower and upper bounds. Geifman

and El-Yaniv [14] propose an algorithm that achieves optimal (in a PAC sense) results by thresholding the confidence function. Nevertheless, they do not consider how to obtain the confidence function, relying on commonly used approaches like the softmax response.
When moving to the bounded-abstention setting in a noisy binary setting, Denis and Hebiri [25] provide an optimal strategy to build a selective classifier using confidence sets[3] when considering the 0-1 loss namely $l(h(\mathbf{X}, Y) = \mathbb{1}\{h(\mathbf{X}) \neq Y\}$. In the rest of the work, we will refer to the risk obtained using the 0-1 loss as the error rate. The selective counterpart will be referred to as the selective error rate.
We denote with Γ_h a confidence set that outputs either $\{h(\mathbf{x})\}$ - i.e. a singleton with the prediction provided by the classifier h - or $\{0, 1\}$. The selective classifier $(h, g)_\Gamma$ associated with Γ_h can be set as:

$$(h, g)_\Gamma(\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if} \quad \texttt{card}(\Gamma_h(\mathbf{x})) = 1 \\ \text{abstain} & \text{otherwise,} \end{cases} \tag{2.12}$$

where `card` refers to the cardinality of confidence set Γ. In other words, the selection function accepts instances only when Γ_h outputs a singleton, namely $g_\Gamma(\mathbf{x}) = \mathbb{1}\{\texttt{card}(\Gamma_h(\mathbf{x})) = 1\}$. Let us also define c-confidence sets as:

Definition 1 (c-confidence set)**.** *Let $c \in [0, 1]$. Then the c-confidence set $\Gamma_c(\cdot)$ can be written as*

$$\Gamma_c(\mathbf{x}) = \begin{cases} \{h_B(\mathbf{x})\} & \textit{if} \quad \varphi(\mathbf{x}) \geq \vartheta_c \\ \{0, 1\} & \text{otherwise,} \end{cases} \tag{2.13}$$

where $\varphi(\mathbf{x}) = \max\{1 - \eta(\mathbf{x}), \eta(\mathbf{x})\}$, $\vartheta_c = F_m^{-1}(1 - c)$ and F_m^{-1} is the inverse of the cumulative function of φ. Under this framework, the authors prove that the selective classifier derived from a c-confidence set is optimal and can be written as:

$$(h_B, g_m)(\mathbf{x}) = \begin{cases} h_B(\mathbf{x}) & \text{if} \quad \varphi(\mathbf{x}) \geq \vartheta_c \\ \text{abstain} & \text{otherwise} \end{cases} \tag{2.14}$$

Moreover, Denis and Hebiri [25] show that a plug-in estimator of such a selective classifier is a reliable strategy if we have a consistent estimator for φ.
Franc et al. [24] generalize the result proposing optimal strategies to build selective classifiers also in the multiclass setting. Given the conditional expected risk function $\rho_h(\mathbf{x})$:

$$\rho_h(\mathbf{x}) = \sum_{y \in \mathcal{Y}} p(Y = y | \mathbf{X} = \mathbf{x}) l(h(\mathbf{x}), y), \tag{2.15}$$

we can obtain an optimal selective classifier by setting the classifier as the Bayes classifier $h_B(\mathbf{x})$, and consider the following (randomized) selection function:

$$g_R(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \rho_h(\mathbf{x}) < \iota \\ \chi & \text{if} \quad \rho_h(\mathbf{x}) = \iota \\ 0 & \text{if} \quad \rho_h(\mathbf{x}) > \iota, \end{cases} \tag{2.16}$$

[3]Confidence sets are measurable functions $\Gamma : \mathcal{X} \to 2^{\mathcal{Y}}$ mapping from $\mathcal{X}$ to the powerset of $\mathcal{Y}$.

where $\rho_h(\mathbf{x})$ is the conditional risk for classifier h, ι is a threshold that determines acceptance, and χ is the probability of providing a prediction when the optimal selection function randomizes at the boundary case, i.e. when $\rho_h(\mathbf{x}) = \iota$. Depending on whether we frame the problem as in Problem 1 or as in Problem 2, Franc et al. [24] show how to define and estimate the pair (χ, ι). For instance, under the bounded abstention framework, when considering a binary classification task and the 0-1 loss, the selective function g_R equals g_m in (2.14).
From a practical perspective, in recent years, several neural network architectures - e.g. [15, 18, 20, 17] - have been proposed to solve the bounded-abstention problem. For an extensive characterization of these existing methods, we refer to Chapter 7.2, where we provide a comprehensive analysis of DNN-based approaches, categorizing them depending on how their selection function is built.

Selective Regression

While several results were developed for selective classification, limited attention has been devoted to the problem of selective regression. The main contributions in this setting are due to Zaoui et al. [26], where the authors provide a series of theoretical results for selective regression when the loss function is the mean squared error (MSE). More precisely, let us define the regression function as $f(\mathbf{x}) = \mathbb{E}[Y \mid \mathbf{X} = \mathbf{x}]$ and the conditional variance function as $\sigma^2(\mathbf{x}) = \mathbb{E}[(Y - f(\mathbf{x}))^2 \mid \mathbf{X} = \mathbf{x}]$. Zaoui et al. [26] show that the optimal strategy can be obtained by directly thresholding the conditional variance $\sigma^2(\mathbf{x})$, i.e.

$$(f, g)(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \sigma^2(\mathbf{x}) < \tau_{reg} \\ \text{abstain} & \text{otherwise} \end{cases} \tag{2.17}$$

Furthermore, if we assume that the cumulative distribution function of $\sigma^2(\mathbf{x})$ (denoted as $F_{\sigma^2(\mathbf{x})}$) is continuous, we can estimate the optimal threshold τ_{reg} achieving coverage c. We denote the inverse of the cumulative function $F^{-1}_{\sigma^2(\mathbf{X})}$ defined on $u \in [0, 1]$ as $F^{-1}_{\sigma^2(\mathbf{X})}(u) = \inf\{t \in \mathbb{R} : F(t) \geq u\}$, then τ_{reg} is such that

$$\tau_{reg} = F^{-1}_{\sigma^2(\mathbf{x})}(c) \tag{2.18}$$

In simpler words, the optimal selective regressor associated with a target coverage c forecasts whenever the variance is below τ_{reg}, which equals the $c-$quantile of the $\sigma^2(\mathbf{x})$ function. Such a value can be estimated using an (unlabelled) calibration set, through the *coverage-calibration* procedure.

2.2.2 Learning to Reject

SC can be seen as one of the two main ways to achieve ambiguity rejection, as the focus is on trading off coverage and selective risk. The other main framework is commonly referred to as Learning to Reject (LtR) or cost-based abstention [24] and is based on the seminal work by Chow [10]. Similarly

to SC, LtR aims at learning a pair (classifier, rejector) such that the rejector determines when the classifier predicts, limiting the predictions to the region where the classifier is likely correct. LtR methods learn the trade-off between abstention and prediction through a parameter a, representing the cost of abstention. Given the cost a, one can define the expected risk $R_B(h, g, a)$ as:

$$R_B(h, g, a) = \int_{\mathcal{X}\times\mathcal{Y}} (l(h(\mathbf{x}), y)g(\mathbf{x})) + ((1 - g(\mathbf{x}))\, a)\, d\mathcal{P}(\mathbf{x}, y) \tag{2.19}$$

The goal then becomes to find the pair (h, g) such that the expected risk $R_B(h, g, a)$ is minimized. More formally,

Problem 3 (Cost-based model [22])**.** *Given the cost of rejection a, an optimal pair classifier-rejector (h_B, g_B) can be defined as:*

$$\arg\min_{h,g} R_B(h, g, a) \tag{2.20}$$

Multiple approaches have been proposed to solve this task. For instance, Chow [10] provides an optimal strategy in the case $P(\mathbf{X}, Y)$ is known, and the loss is the 0-1 loss. The extension to the case where $P(\mathbf{X}, Y)$ is not known is discussed by Herbei and Wegkamp [27], where the plug-in approach is considered to estimate the conditional probability $P(Y|\mathbf{X})$. Fumera et al. [28] study the usage of multiple thresholds for the binary setting, where there might be difficulties in estimating the conditional distribution of the minority class.
While the LtR methods presented so far are model-agnostic with respect to the choice of the algorithm employed for estimating h, many proposed approaches exploit the peculiarities of different algorithms. For instance, Hellman [29] presents how to extend k-Nearest Neighbour classifiers (kNN) to include a reject option. Given the cost for rejection a, the proposed method looks at the k nearest neighbours of a certain instance and abstention is prescribed if $k' < k/2$ nearest neighbours belong to a different class than the majority of the neighbours. The optimal k' depends on the value of a. Fumera and Roli [30] devise an approach to learn Support Vector Machines with an embedded reject option in the binary setting. By including the cost of rejection a in the 0-1 loss, they define the training task as a non-convex optimization problem and they propose an algorithm to solve it. Cordella et al. [31] propose the usage of a linear performance function to train neural networks that are able to reject. Such a function is based on a linear combination of the misclassification cost and the rejection cost a. Similarly, Sundararajan and Pal [32] propose the Conservative Perceptron Learning Rule algorithm to embed perceptrons with a reject option. Such an algorithm trains a hyperplane with a bandwidth for rejection and, by restricting the bandwidth within a certain range, it is guaranteed to converge to a solution. Cortes et al. [33] study how ensembling techniques could be used within LtR, by providing a theoretical analysis of bounds and a boosting-style algorithm to perform LtR in the binary setting. Fischer et al. [34] investigate optimal rejection methods for classifiers that partition the input space, such as prototype-based classifiers, support vector machines, or decision trees. More precisely, they consider (un)certainty

metrics, such as the distance to the closest decision boundary, and propose methods to determine optimal local thresholds on given data, linking the LtR problem to the Knapsack problem [35].
We conclude by highlighting that LtR deviates from SC in two major aspects. First, the rejection strategy does not rely on confidence functions, but it is based on the parameter cost a. Defining such a parameter is not straightforward, and it is heavily context-dependent [25]. Second, LtR methods are not meant to tackle the problem of minimizing a risk given a target coverage c. A more in-depth theoretical analysis for both LtR and SC can be found in [24], where the authors show that both frameworks share similar optimal strategies.

2.3 Novelty Rejection

While LtR methods and SC are typically used to perform *ambiguity rejection*, an alternative approach is to abstain from making predictions on instances that are unlikely to be seen in the distribution generating the training data. This approach is commonly referred to as *novelty rejection* [36, 37].
Such a rejection strategy is especially sought whenever the training and the test distribution differs [12]. For instance, in the healthcare context, der Pias et al. [38] use a LOF-based rejector [39] to avoid providing predictions for patients that are younger than the ones used to train the model.
Overall, we can distinguish different ways to build novelty rejectors. A popular approach is to use a rejector that is trained independently from the final classifier. For example, one can estimate the marginal density $P(\mathbf{X})$ and reject if the probability of a novel instance is below a certain threshold $\tau_{novelty}$. The intuition is that such an approach quantifies the likelihood of a specific instance belonging to a feature space that closely resembles the training set one. Several methods have been considered for such a purpose: Landgrebe et al. [40] propose the usage of Gaussian Mixtures Models (GMM) to estimate the marginal density and then perform rejection depending on these values. Resorting to deep neural networks, Nalisnick et al. [41] consider the usage of Normalizing Flows, while Wang and Yiu [42] employ Variational Autoencoders to estimate the marginal density $P(\mathbf{X})$.
Another option is to employ a one-class classification model that learns to encapsulate the training dataset and flag as novelty all the instances falling out of this region [43].
Alternatively, some external rejectors assign scores representing the novelty of an example and abstain when such a score is above a certain level: Liang et al. [44] show that employing temperature scaling [45] and adding small perturbations to the input help separate the neural network softmax score distributions for out-of-distribution (o.o.d.) instances from in-sample ones, allowing for more effective detection of novel examples. Kühne et al. [46] combine a deep learning model with an Autoencoder following these steps: first, the novel instance to evaluate is passed through a previously trained Autoencoder to obtain a reconstructed input. Intuitively, if the original instance is from the training set, then the autoencoder should be able to properly reconstruct the instance,

making the reconstructed instance similar to the original data. Then both the reconstructed input and the original instance are passed through the deep learning model, which provides two final predictions to compare. Thus, if the predictions are similar, then the example likely belongs to the training set.
Finally, another option is to use a confidence-based approach. Differently from the ambiguity rejection framework, here a popular choice is to approximate the class conditional density function $P(\mathbf{X} = \mathbf{x} \mid Y = y)$ [36]. We can then define a confidence function as:

$$\upsilon_{nov} = \max_{y \in \mathcal{Y}} P(\mathbf{X} = \mathbf{x} \mid Y = y) \tag{2.21}$$

The intuition is straightforward: the higher the value of υ_{nov} the more likely the sample is a common one [47]. In order to estimate the confidence function, one can rely on the usage of GMM, as done for example by Vailaya and Jain [48], or on heuristics, such as the distance from the closest prototype [49].

2.4 Concerns from abstaining classifiers

The usage of the reject option introduces a few concerns, such as fairness and explainability of the rejection strategy. For the sake of completeness, in this section, we review the main concerns raised by introducing a reject option and some of the current approaches to tackle them.

Fairness. The attention devoted to building fair AI systems has been increasingly growing in the last decade [50].
When introducing a reject option, one might be interested in evaluating if such a solution is unfair towards some protected groups. For example, one wants to avoid the selective predictor rejecting only members of a specific minority group.
For the selective prediction framework, only a few of works try to address these concerns. For instance, Jones et al. [51] were the first to show that even if accuracy increases overall for accepted instances, there might be some protected groups that are negatively affected by the introduction of an abstention mechanism. To address this issue, Lee et al. [52] propose a surrogate loss that computes the performance on different subgroups of instances. This loss is employed to train a neural network that enforces a sufficiency condition to avoid unfair results. Shah et al. [53] propose a similar strategy for selective regression under fairness criteria. A different approach is proposed by Schreuder and Chzhen [54], who provide a theoretical characterization of the selective classification framework when introducing a fairness constraint in Problem 2.

Explainability. Explainable AI methods [1] are increasingly used to overcome the difficulty of interpreting AI outputs. In particular, adding explanations to rejections allows for characterizing the areas where the predictor is not confident enough.

To the best of our knowledge, the current literature on explaining the selection mechanism is limited. Fischer et al. [34] propose a reject option for natively interpretable models such as prototype-based ones. This allows directly characterizing the reject option. We point out that such a strategy is not always viable, e.g. when the confidence is obtained through a complex AI system.
Alternatively, one can consider black-box explanation methods for explaining the abstention decision. For instance, Artelt et al. [55] consider counterfactual explanations to explain reject options of learning vector quantization classifiers. Moreover, Artelt and Hammer [56] propose semi-factual explanations for the reject option, which yields a model-agnostic approach at the expense of potentially high complexity. Finally, Artelt et al. [57] present a model-agnostic framework to improve abstaining classifiers with explanation mechanisms. In particular, they pair a conformal predictor[4] with local surrogates computed for the region of interest. This allows one to consider model-agnostic explainable methods such as counterfactual, semi-factual, and factual approaches.

Decision aids. The deployment of AI systems as a support for decision-making has been increasingly discussed in the literature. Multiple studies suggest that the AI system can induce some cognitive bias in the human decision-maker [58]. For instance, Green and Chen [59] study the effectiveness of using an ML-based risk assessment model as a decision aid in a controlled experiment. Their results suggest that the human participants (i) were incapable of accurately assessing the validity of their predictions or the risk assessment's predictions; (ii) did not adjust their level of trust in the risk assessment tool based on its performance; (iii) displayed prejudice in their interactions with the risk assessment model.
Although these concerns apply also to abstaining predictors, to the best of our knowledge, only Bondi et al. [11] examine the impact of using a selective predictor on human performance. Utilizing real-world data, the authors show that human evaluations are affected by how information is conveyed. Specifically, their attention is directed towards (i) whether humans have access to the AI system's prediction and (ii) whether they are provided the AI system's decision to defer. Through the manipulation of these communications components, they show that it is possible to have a substantial impact on human performance. In particular, their findings suggest that performance can be enhanced by alerting the human about the decision to defer while withholding the AI's prediction.

Robustness and adversarial attacks. Robustness refers to the extent to which an AI system operates reliably and accurately under harsh conditions. In particular, we are interested in the system's ability to maintain its integrity and operate effectively in challenging settings, e.g. adversarial attacks, disturbances, and data contamination [60].
A few works tried to enhance the robustness of an AI system to adversarial attacks via the reject option. For instance, Laidlaw and Feizi [61] present a

[4]See Section 2.5.2 for a brief overview.

technique known as Combined Abstention Robustness Learning (CARL) to simultaneously train a classifier and determine the specific region of the input space where it should abstain. They demonstrate that training with CARL can lead to a classifier that is both more accurate and more resilient.
Pang et al. [62] show it is possible to distinguish correctly classified inputs from misclassified ones. For such a purpose, they combine two interconnected measures, namely the prediction score s_y and the true confidence, i.e. $s_{y_{true}} \approx P(Y = y_{true}|\mathbf{X} = \mathbf{x})$, which refers to the score assigned to the ground-truth value y_{true}. In their work, they provide a way for estimating a rectified confidence that represents true confidence and using it to train a selected classifier in an adversarial manner.
Balcan et al. [63] introduce a random feature subspace threat model and show that classifiers lacking the capability to reject are susceptible to this adversary. However, by enabling the classifier to abstain, such as through a thresholded nearest-neighbor algorithm, these attacks can be effectively countered.
Chen et al. [64] theoretically analyze the stratified rejection setting and devise a novel defense approach, called Adversarial Training with Consistent Prediction-based Rejection (CPR) for building an abstaining classifier that mitigates the exposure to adversarial attacks.

2.5 Related Work

In this section, we briefly introduce some of the fields related to the Selective Prediction framework.

2.5.1 Uncertainty Quantification

Traditionally, uncertainty estimation has been closely associated with standard probability and probabilistic predictions in the statistical realm. However, in the field of machine learning, new challenges have arisen, such as trust, robustness, and safety, requiring the development of novel methodological approaches. For instance, when dealing with abstaining predictors, we would like to abstain over those instances where the model is more uncertain.
Typically, in statistics, one can distinguish between aleatoric uncertainty and epistemic uncertainty [65]. Aleatoric uncertainty, also known as statistical uncertainty, pertains to the concept of randomness. It encompasses the variability in the results of an experiment that arises from essentially random factors. For example, when one rolls a die, the data-generating procedure in such an experiment contains a random component that cannot be mitigated by any other source of information. In contrast, epistemic uncertainty, also known as systematic uncertainty, arises from a lack of knowledge (e.g. about the best model to consider). Unlike aleatoric uncertainty, epistemic uncertainty can be reduced (in principle) by acquiring more information.
There are multiple ways to categorize methods of estimating uncertainty. For an in-depth review of uncertainty estimation in the machine learning field, refer to Hüllermeier and Waegeman [66]. A common distinction is between set-

valued prediction and uncertainty quantification methods. The former methods provide a set of predictions that comprise the true value with a specified (desired) level of confidence. Conversely, uncertainty quantification methods generally provide a single prediction and equip it with additional information about how that prediction is certain.

We will discuss in more detail one of the most popular set-valued methods, i.e. conformal prediction, in the next subsection, while for the remainder of this paragraph, we will briefly overview some uncertainty quantification methods.

A popular approach for uncertainty quantification is to estimate directly a probabilistic classifier $s : \mathcal{X} \to [0; 1]$: in the case of classification this translates into estimating the conditional probability for each class $y \in \mathcal{Y}$; while for the regression task, it means estimating a density function on $\mathbb{R}$.

To train a probabilistic classifier, one can typically minimize suitable losses, such as proper scoring rules [67]. However, one often obtains (at best) pseudo-probabilities that do not accurately reflect the underlying true probability. Hence, several calibration methods have been proposed to scale these learned scores into proper probabilities. Typically, these techniques work as follows: first, they learn a mapping from the learned scores to the unit interval. Then, the learned mapping can be applied ex-post to the scores to obtain proper probabilities. Popular methods include binning [68, 69], isotonic regression [70], logistic scaling [71], temperature scaling [45] and parametric forms of scaling [72, 73]. We refer to de Menezes e Silva Filho et al. [74] for a complete overview of different calibration techniques.

Another widely adopted method for uncertainty quantification is *ensembling* [75]. The core idea is to train multiple models, which will provide multiple predictions. Averaging these predictions enables one to improve predictive performance. Moreover, one can exploit the multiple models to construct measures of uncertainty. This approach has proven successful in various applications, including selective classification [23], monitoring model degradation [76], detecting misclassifications [77], and addressing adversarial attacks [78], among others.

Alternatively, one can rely on Bayesian approaches. For instance, Seeger [79] introduces Gaussian Processes that generalize the Bayesian inference for multivariate (but finite-dimensional) random variables to inference for (infinite-dimensional) functions. Other popular Bayesian approaches rely on neural networks, including Bayesian Neural Networks [80, 81, 82], approximate variational techniques (e.g. Dropout Variational Inference [83]) and methods that try to distinguish between aleatoric and epistemic uncertainty [84, 85].

To conclude, a connection between the two kinds of uncertainty - i.e., aleatoric and epistemic - and the rejection types - i.e., ambiguity and novelty- can be drawn. As highlighted by Hendrickx et al. [12], ambiguity rejection resembles aleatoric uncertainty since the difficulty of providing a prediction for a specific instance can arise due to the intrinsic probabilistic nature of the relationship between $\mathbf{X}$ and Y. On the other hand, we can pair novelty rejection to epistemic uncertainty because the novelty of a certain instance comes from the fact that we did not see (enough) similar examples in the training set. Hence, we

can reduce the uncertainty around novel instances by enlarging our training data.

2.5.2 Conformal Prediction

Conformal Prediction (CP) [86] is one of the most popular set-valued uncertainty prediction methods.
More formally, given a training set $\mathcal{T}_n$ with n samples $\{(\mathbf{X}_i, Y_i)\}_{i=1}^n$ drawn from an unknown distribution P on $\mathcal{X} \times \mathcal{Y}$, a nominal error level α and a new i.i.d. instance $(\mathbf{X}_{n+1}, Y_{n+1})$, a conformal predictor is a confidence set $\hat{\Gamma}_n : \mathcal{X} \to 2^{\mathcal{Y}}$ such that:

$$P\left(Y_{n+1} \in \hat{\Gamma}_n(\mathbf{X}_{n+1})\right) \geq 1 - \alpha \tag{2.22}$$

The most straightforward way to build such sets relies on thresholding the so-called *conformal score*, a function $v : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that increases whenever a prediction is uncertain.
For example, given a probabilistic classifier $s : \mathcal{X} \to [0,1]^{\mathcal{Y}}$, one can set the conformal score as $v(\mathbf{x}, y) = 1 - s_{y_{true}}(\mathbf{x})$, where $s_{y_{true}}(\mathbf{x}) \approx P(Y = y_{true}|\mathbf{X} = \mathbf{x})$ is the estimated probability for the true class conditional distribution. We emphasize that estimating this quantity requires access to the true class value y_{true}, which is only available at training time[5]. Afterward, one can use a calibration dataset to compute a quantile $\hat{q}$ such that it covers $100(1-\alpha)\%$ of the scores $s_{y_{true}}$. The prediction set is then built including all the softmax values s_y that overcome the threshold $\hat{q}$. Vovk [87] advocates the usage of a cross-fitting strategy to avoid splitting available data into a training set and a calibration dataset. Theoretical guarantees for such a strategy can be found in Kim et al. [88]. Other choices for conformal scores are possible: for instance, Martinez et al. [89] propose the usage of influence functions as valid conformal scores. In this way, one is guaranteed that at least a $100(1-\alpha)\%$ of the instances are above the level $1 - \hat{q}$. This method (called inductive conformal prediction) was proposed by Papadopoulos et al. [90], who provided the theoretical guarantees for such a strategy.
There are some similarities with selective prediction, as the conformal score is closely related to the concept of confidence function. However, the main difference with SC is that CP focuses on quantifying the uncertainty associated with predictions rather than aiming to minimize selective risk (maximize coverage) given a target coverage (error rate) [19]. Moreover, while the confidence function maps from the feature space $\mathcal{X}$ to $\mathbb{R}$, the conformal score is generally defined as a function mapping from $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$, as it requires ground truth values.
Some attempts have tried to merge abstaining predictors and the conformal prediction framework: for instance, in [91], CP is used to give guarantees over the selective error rate in an SC scenario by: (i) training a conformal predictor (e.g., SVC [92]), (ii) calibrating its confidence levels, (iii) setting a selection threshold over the confidence or p-values generated by the conformal predictor.

[5]For example, if one faces a binary classification task, s will output scores for two classes, i.e. s_0 and s_1. If $y_{true} = 1$, then $s_{y_{true}} = s_1$, otherwise $s_{y_{true}} = s_0$.

Another example is the work by Feng et al. [93], where the authors build models that can either provide prediction sets or abstain.
To conclude, for a more in-depth overview of conformal prediction, we refer to Angelopoulos and Bates [94].

2.5.3 Learning to Defer

Learning to Defer (LtD) [95] is a generalization of LtR, where rather than incurring a rejection cost, the AI system can defer instances to human expert(s). Compared with LtR and SC, one of the main differences is that the expert's predictions might be wrong under the LtD framework.
This is generally modeled using a cost function[96]: by denoting with l_M and l_H the loss functions connected with the ML model and the human(s), the loss function in (2.19) can be generalized to

$$R_B(h_M, h_H, g) = \int_{\mathcal{X}\times\mathcal{Y}} (l_M(h_M(\mathbf{x}), y)g(\mathbf{x}))+((1-g(\mathbf{x}))\, l_H\,(h_H(\mathbf{x}), y))\, d\mathcal{P}(\mathbf{x}, y), \tag{2.23}$$

where h_M and h_H represent the predictions made by the ML model and the human(s) respectively.
Thus, by including human experts in the loop, one aims to find an optimal selection strategy for the whole human-AI system. Roughly speaking, such a strategy decides whether to make the model predict, which results in a cost equal to the model loss, or defer the prediction to the user, which incurs the user cost. From a theoretical perspective, De et al. [97] show that the problem of learning under human assistance when choosing a ridge regression as a base predictor is NP-hard. By reformulating the problem using submodular functions, they devise a greedy algorithm with some theoretical guarantees. Similar results also hold for the classification setting when considering margin-based classifiers, as shown in De et al. [98]. Okati et al. [99] provide a formal characterization of the specific scenarios where a predictive model can take advantage of including humans in the loop. They show that standard ML models that are trained to predict over all the instances may be suboptimal when it comes to LtD, proposing a deterministic threshold rule to determine when the AI system or the human has to predict.
Due to the difficulties in directly optimizing 2.23, several attempts provide surrogate losses to learn predictors that are able to defer to experts: Mozannar and Sontag [96] propose a method that jointly learns both the rejector and the ML predictor with some generalization bounds; Verma et al. [100] consider the problem of deferring to a multitude of experts; Mozannar et al. [101] provide a Mixed Integer Linear Programming formulation to solve the problem in the linear setting and a novel surrogate loss that is realizable and consistent.

Chapter 3

A Model-Agnostic Heuristics for Selective Classification

The goal of this chapter is to introduce an alternative way to perform selective classification, answering to *Req. 1*. We propose a model-agnostic method that builds the selective classifier by exploiting a cross-fitting strategy. We show the goodness of this approach through extensive experiments. This chapter is based on the work [102] presented at AAAI 2023.

3.1 Introduction

Given the pressing demand for trustworthy AI systems in various sensitive contexts, building selective classifiers that achieve state-of-the-art results is highly sought. As mentioned in the previous chapter, many methods rely on architecture-specific solutions to build selective classifiers. In particular, starting from Geifman and El-Yaniv [14], researchers have been proposing novel approaches that are tailored to DNNs, e.g. in the case of SelectiveNet (SelNet) [15] and Self-Adapting Training (SAT) [18].
Although DNNs have enabled unprecedented advances in several domains, such as computer vision and natural language processing, they are not a panacea. For instance, tree-based algorithms still outperform them when considering tabular data [103], which are used in several sensitive settings, e.g. banking [104].
Due to this, there is an uprising need for selective classifiers that can adapt to various domains, without relying on peculiar model characteristics. In this chapter, we present a simple yet effective heuristics addressing these concerns. Our contribution consists of a model-agnostic approach that transforms any base probabilistic binary classifier into a selective (probabilistic binary) classifier.
Inspired by popular techniques in causal inference [105] and conformal prediction [87, 88], we propose an algorithm, called SCross, exploiting a cross-fitting strategy. First, SCross splits the available data into K folds. Second, it trains a classifier over $K-1$ folds and predicts the confidence values over the remaining K-th fold. Finally, it stacks (all) the predicted confidence values altogether

to approximate the confidence over the full dataset. These can be used to estimate the bounds on prediction confidence for which the base classifier should abstain.
The estimation of the bounds relies on theoretical results on quantile estimation [106], which guarantee a second-order improvement over sample quantile estimation. Moreover, the cross-fitting approach allows one to exploit the whole training sample to directly calibrate the selection function, without requiring an external validation set.
We evaluate our approach through 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 SelNet and SAT.
The rest of the chapter is organized as follows: we present our approach in Section 3.2, while Section 3.3.1 reports experimental results. We finally summarize our contribution in Section 3.4.

3.2 Model-Agnostic Heuristics

3.2.1 Preliminaries

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 $P(Y = 1|\mathbf{X} = \mathbf{x})$. 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 selection function $g(\mathbf{x}) = \mathbb{1}\{\upsilon_h(\mathbf{x}) > \tau\}$, with the canonical choice of the maximum of scores as confidence function:

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

Our approach aims at determining a threshold τ for the selective classification problem (2.11) that is a better estimate of the $100(1 - c)$ percentile of φ_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 [106], reported next for completeness.

Theorem 1 ([106], 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.*

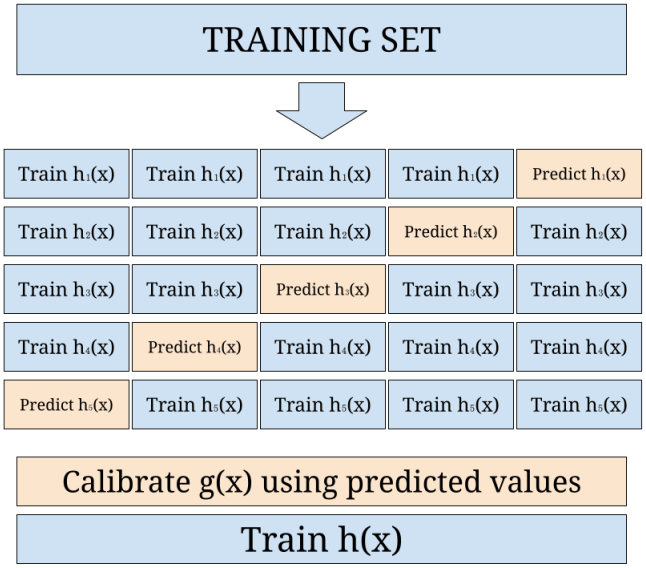


Figure 3.1: The cross-fitting strategy

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 [105].

3.2.2 The SCross algorithm

Let us describe our approach.

Given a training set $\mathcal{T}$, we partition $\mathcal{T}$ into K non overlapping stratified folds $\mathcal{T}_1, \ldots, \mathcal{T}_K$, and we define $\mathcal{T}_{-k} = \mathcal{T} \setminus \mathcal{T}_k$ for $k = 1, \ldots, K$. For each k, we train a base probabilistic classifier s_k over $\mathcal{T}_{-k}$. We then compute the scores $\mathbf{s}_k(\mathbf{x})$ for $\mathbf{x} \in \mathcal{T}_k$, and the confidence values $\varphi_{s_k}(\mathbf{x}) = \max\{\mathbf{s}_k(\mathbf{x}), 1 - \mathbf{s}_k(\mathbf{x})\}$. Figure 3.1 briefly summarizes this cross-fitting strategy. Let us define $\varphi_{s_k}(\mathcal{T}_k) = \{\varphi_{s_k}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{T}_k\}$, and $\varphi(\mathcal{T}) = \cup_{k=1}^{K} \varphi_{s_k}(\mathcal{T}_k)$. Notice that $\varphi(\mathcal{T})$ is an approximation of the (unknown) population of confidence values of the classifier s, since each s_k is built on $K-1$ stratified folds of the entire training set $\mathcal{T}$ which, in turn, is a set of i.i.d. realizations from $\mathcal{D}$. For a target coverage c, we set $\hat{\tau}$ as as the $100(1-c)$ percentile of the empirical distribution of $\varphi(\mathcal{T})$. According to Theorem 1, to minimize the variance of the quantile estimator, we then randomly partition $\varphi(\mathcal{T})$ in two subsets of equal size and we compute the empirical $100(1-c)$ percentiles over each of them, denoting such percentiles

Algorithm 1: SCross.fit()

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

Output: (h, g) - selective classifier

1 $\mathcal{T}_1, \ldots, \mathcal{T}_K \leftarrow \text{StratifiedKFold}(\mathcal{T}, K)$ `// partitioning`
2 **for** $k \in 1, \ldots, K$ **do** `// for each fold`
3 $\quad \mathcal{T}_{-k} = \mathcal{T} \setminus \mathcal{T}_k$ `// training data`
4 $\quad s_k \leftarrow S.fit(\mathcal{T}_{-k})$ `// train` k^{th} `classifier`
5 $\quad \mathbf{s}_k = s_k.score(\mathbf{X}_k)$ `// score data`
6 $\quad \varphi_k = \max\{\mathbf{s}_k, 1 - \mathbf{s}_k\}$ `// compute confidence`
7 $\varphi \leftarrow \cup_{k=1}^{K} \varphi_k$ `// store all the scores`
8 $\hat{\tau} \leftarrow \text{perc}(\varphi, 100(1-c))$ `// calculate` $\hat{\tau}$
9 $\varphi_1, \varphi_2 \leftarrow \text{KFold}(\varphi, 2)$ `// partitioning`
10 $\bar{\tau}_1 \leftarrow \text{perc}(\varphi_1, 100(1-c))$ `// calculate` $\bar{\tau}_1$,
11 $\bar{\tau}_2 \leftarrow \text{perc}(\varphi_2, 100(1-c))$ `// calculate` $\bar{\tau}_2$
12 $\tilde{\tau} = \frac{1}{\sqrt{2}}\hat{\tau} + (1 - \frac{1}{\sqrt{2}})(\frac{1}{2}\bar{\tau}_1 + \frac{1}{2}\bar{\tau}_2)$ `// final` $\bar{\tau}$ `estimate`
13 $s \leftarrow S.fit(\mathcal{T})$ `// classifier`
14 $h \leftarrow$ **lambda** $\mathbf{x} : \mathbb{1}\{s.score(\mathbf{x}) \geq \frac{1}{2}\}$ `// final classifier`
15 $g \leftarrow$ **lambda** $\mathbf{x} : \mathbb{1}\{\max\{1 - s.score(\mathbf{x}), s.score(\mathbf{x})\} \geq \bar{\tau}\}$ `// selection function`
16 **return** (h, g)

as $\bar{\tau}_1$ and $\bar{\tau}_2$ respectively. The final estimator $\tilde{\tau}$ is:

$$\tilde{\tau} = \frac{1}{\sqrt{2}}\hat{\tau} + (1 - \frac{1}{\sqrt{2}})(\frac{1}{2}\bar{\tau}_1 + \frac{1}{2}\bar{\tau}_2) \tag{3.1}$$

We report the details of our approach in Algorithm 1, which we name SCross for Selective Cross-Classification. Notice that the final classifier h is built at lines 13-15 on the full training set $\mathcal{T}$. This is not strictly required, e.g., in the case of an already trained classifier.

3.3 Experimental Evaluation

3.3.1 Research Questions

To evaluate the effectiveness of SCross we run several experiments, trying to address the following questions:

Q1: Are the results comparable to state-of-the-art methods?

Q2: Are the results affected by the choice of the base classifier?

Q3: Are the results affected by the choice of the parameter K?

The code to reproduce the experiments is available at `https://github.com/andrepugni/SCross`.

3.3.2 Experimental Settings

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 pre-processing details.

`Adult.` We dropped from the raw census data [107] 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[1] 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 [108] 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[2] 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 [107] concerns whether or not a credit card holder will default in the next six months [109]. 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[3] containing a collection of cats and dogs images.

[1] `https://www.kaggle.com/wordsforthewise/lending-club`
[2] `https://www.kaggle.com/c/GiveMeSomeCredit`
[3] `https://www.kaggle.com/competitions/dogs-vs-cats`

The task here is to distinguish between the two species. The training and test sets were split as described in [16]. 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.

Baselines. We compare our approach with a bounded-abstention version of the plug-in rule [27] (PLUG-IN), the SELNET method [15], and the SAT method [18]. PLUG-IN uses the softmax function as the confidence function, as in our approach, but it computes the τ 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.

Architectures and Hyperparameters. For tabular data, we train both SELNET and SAT using a ResNet structure [104], while we use LightGBM [110] as a base classifier for both SCROSS and PLUGIN. We also evaluate how the performances of SCROSS are affected by the base classifier choice, using a Logistic Regression, a Random Forest, a ResNet and an XGBoost classifier. For the image data (the CatsVsDogs dataset), we compare all the methods on the same VGG architecture provided by [16] 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 [18]; for SELNET we use its original loss function[4] setting parameters α and λ to .50 and 32, as in [15]. For SELNET and SAT approach, we set 300 epochs in training, Stochastic Gradient Descent as an optimizer, a learning rate of .10 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 consists of 10% of the training set as in their original implementations.

Evaluation metrics. The metrics used for performance evaluation include the empirical coverage, the empirical accuracy (1 minus the selective error rate), 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 sets 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 [111]. We follow the approach in [112] to test whether differences across multiple classifiers are statistically significant at .001 significance level.[5]

Hardware. 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,

[4]See Chapter 7.2 for a detailed description.

[5]We rely on the Python package `autorank` [113] for statistical analysis.

OS Ubuntu 20.04.4, programming language Python 3.8.12.

Comparing approaches. We compare 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.

Limitations. Before discussing the experimental results, a clarification should be made 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 the same trained base classifier for all of the four methods, as SelNet and SAT would 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 does 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, [103, 114]. However, the selective accuracy of all four methods for $c = .99$ is very close (see Table 3.2). This means that the performances of the LightGBM base classifier and of the ResNet trained by SelNet and SAT are very close 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 with, and results will be described later on.

3.3.3 Experimental Results

Q1: SCross performance. Let us start considering results in terms of empirical coverage, as reported in Table 3.1. Regarding empirical coverage, SCross is the closest to target coverage c 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 violations from target coverages.
Regarding selective accuracy, we can see from Table 3.2 that 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 perfor-

Table 3.1: Empirical coverage for SCross and baselines (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Empirical Coverage			
	c	**SCross**	**PlugIn**	**SelNet**	**SAT**
Adult	.99	.993 ± (.001)	.992 ± (.001)	.991 ± (.001)	**.990 ± (.001)**
	.95	**.950 ± (.002)**	.947 ± (.002)	.946 ± (.002)	.941 ± (.002)
	.90	.904 ± (.003)	**.900 ± (.003)**	**.899 ± (.003)**	.893 ± (.003)
	.85	**.850 ± (.003)**	.846 ± (.003)	.840 ± (.004)	**.850 ± (.003)**
	.80	**.802 ± (.004)**	.794 ± (.004)	.786 ± (.004)	.828 ± (.004)
	.75	**.746 ± (.004)**	.743 ± (.004)	.741 ± (.004)	.798 ± (.004)
Lending	.99	.994 ± (.001)	.995 ± (.001)	**.992 ± (.001)**	1.00 ± (.000)
	.95	**.969 ± (.001)**	.970 ± (.001)	.979 ± (.001)	1.00 ± (.000)
	.90	.938 ± (.001)	.939 ± (.001)	**.918 ± (.001)**	1.00 ± (.000)
	.85	**.908 ± (.001)**	.909 ± (.001)	.931 ± (.001)	1.00 ± (.000)
	.80	**.879 ± (.001)**	.880 ± (.001)	.897 ± (.001)	1.00 ± (.000)
	.75	.849 ± (.001)	.850 ± (.001)	**.821 ± (.001)**	1.00 ± (.000)
GiveMe	.99	**.990 ± (.001)**	**.990 ± (.001)**	**.991 ± (.001)**	.992 ± (.001)
	.95	**.950 ± (.002)**	.949 ± (.002)	.948 ± (.002)	.988 ± (.001)
	.90	**.902 ± (.002)**	.903 ± (.002)	**.901 ± (.002)**	.984 ± (.001)
	.85	**.852 ± (.002)**	.856 ± (.002)	.853 ± (.002)	.961 ± (.001)
	.80	**.800 ± (.003)**	.807 ± (.003)	.806 ± (.003)	.955 ± (.002)
	.75	.748 ± (.003)	.756 ± (.003)	**.750 ± (.003)**	.950 ± (.002)
UCICredit	.99	**.993 ± (.001)**	**.993 ± (.002)**	**.988 ± (.002)**	1.00 ± (.000)
	.95	**.951 ± (.003)**	**.948 ± (.003)**	**.949 ± (.003)**	1.00 ± (.000)
	.90	.906 ± (.004)	**.897 ± (.004)**	**.904 ± (.004)**	.992 ± (.002)
	.85	.848 ± (.005)	**.851 ± (.005)**	.846 ± (.005)	.992 ± (.002)
	.80	.797 ± (.005)	**.803 ± (.005)**	.806 ± (.005)	.992 ± (.002)
	.75	**.750 ± (.005)**	**.751 ± (.006)**	.770 ± (.005)	.992 ± (.002)
CSDS1	.99	.980 ± (.001)	.980 ± (.001)	.980 ± (.001)	**.986 ± (.001)**
	.95	.917 ± (.001)	.918 ± (.001)	.924 ± (.001)	**.934 ± (.001)**
	.90	.849 ± (.002)	.853 ± (.002)	.859 ± (.002)	**.869 ± (.002)**
	.85	.790 ± (.002)	.792 ± (.002)	.795 ± (.002)	**.805 ± (.002)**
	.80	.736 ± (.002)	.737 ± (.002)	**.741 ± (.002)**	.740 ± (.002)
	.75	.682 ± (.002)	.681 ± (.002)	.680 ± (.002)	**.690 ± (.002)**
CSDS2	.99	.983 ± (.002)	**.984 ± (.002)**	.933 ± (.003)	1.00 ± (.000)
	.95	.916 ± (.003)	.927 ± (.003)	.905 ± (.003)	**.934 ± (.003)**
	.90	.834 ± (.004)	.851 ± (.004)	.814 ± (.004)	**.875 ± (.003)**
	.85	.761 ± (.004)	.773 ± (.004)	.747 ± (.004)	**.812 ± (.004)**
	.80	.687 ± (.005)	.715 ± (.005)	.682 ± (.005)	**.761 ± (.004)**
	.75	.620 ± (.005)	.651 ± (.005)	.616 ± (.005)	**.703 ± (.005)**
CSDS3	.99	**.991 ± (.001)**	.992 ± (.001)	.992 ± (.001)	.995 ± (.001)
	.95	.955 ± (.002)	.957 ± (.002)	**.949 ± (.002)**	.954 ± (.002)
	.90	.911 ± (.002)	.909 ± (.002)	**.906 ± (.002)**	.904 ± (.002)
	.85	.865 ± (.003)	.859 ± (.003)	**.850 ± (.003)**	.862 ± (.003)
	.80	.818 ± (.003)	**.808 ± (.003)**	**.808 ± (.003)**	.824 ± (.003)
	.75	.769 ± (.003)	**.758 ± (.003)**	.758 ± (.003)	.777 ± (.003)
CatsVsDogs	.99	**.992 ± (.002)**	.997 ± (.001)	.923 ± (.004)	.913 ± (.004)
	.95	**.960 ± (.003)**	.979 ± (.003)	.855 ± (.006)	.867 ± (.005)
	.90	**.922 ± (.004)**	.948 ± (.004)	.809 ± (.006)	.819 ± (.006)
	.85	**.889 ± (.005)**	.912 ± (.004)	.795 ± (.006)	.777 ± (.006)
	.80	.846 ± (.005)	.869 ± (.005)	**.758 ± (.007)**	.729 ± (.007)
	.75	.794 ± (.006)	.822 ± (.006)	**.720 ± (.007)**	.682 ± (.007)
	#	20/48	11/48	17/48	12/48

Table 3.2: Selective Accuracy for SCross and baselines (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Empirical Selective Accuracy			
	c	**SCross**	**PlugIn**	**SelNet**	**SAT**
Adult	.99	.871 ± (.003)	**.872 ± (.003)**	.847 ± (.003)	.845 ± (.003)
	.95	**.888 ± (.003)**	**.888 ± (.003)**	.856 ± (.003)	.845 ± (.003)
	.90	.902 ± (.003)	**.903 ± (.003)**	.875 ± (.003)	.845 ± (.004)
	.85	.919 ± (.003)	**.920 ± (.003)**	.894 ± (.003)	.845 ± (.004)
	.80	.934 ± (.003)	**.936 ± (.003)**	.902 ± (.003)	.846 ± (.004)
	.75	**.950 ± (.003)**	**.950 ± (.003)**	.923 ± (.003)	.845 ± (.004)
Lending	.99	**.899 ± (.001)**	**.899 ± (.001)**	.873 ± (.001)	.876 ± (.001)
	.95	**.910 ± (.001)**	.909 ± (.001)	.880 ± (.001)	.876 ± (.001)
	.90	**.924 ± (.001)**	.923 ± (.001)	.891 ± (.001)	.876 ± (.001)
	.85	**.938 ± (.001)**	.937 ± (.001)	.900 ± (.001)	.876 ± (.001)
	.80	**.951 ± (.001)**	**.951 ± (.001)**	.918 ± (.001)	.876 ± (.001)
	.75	**.963 ± (.001)**	**.963 ± (.001)**	.940 ± (.001)	.876 ± (.001)
GiveMe	.99	**.942 ± (.002)**	**.942 ± (.002)**	.939 ± (.002)	.937 ± (.002)
	.95	**.956 ± (.002)**	**.956 ± (.002)**	.953 ± (.002)	.939 ± (.002)
	.90	**.967 ± (.001)**	**.967 ± (.001)**	.964 ± (.001)	.940 ± (.002)
	.85	**.973 ± (.001)**	**.973 ± (.001)**	.969 ± (.001)	.945 ± (.002)
	.80	**.978 ± (.001)**	.977 ± (.001)	.971 ± (.001)	.946 ± (.002)
	.75	**.981 ± (.001)**	.980 ± (.001)	.975 ± (.001)	.946 ± (.002)
UCICredit	.99	**.814 ± (.005)**	**.814 ± (.005)**	.813 ± (.005)	.813 ± (.005)
	.95	**.827 ± (.005)**	.826 ± (.005)	.818 ± (.005)	.813 ± (.005)
	.90	**.839 ± (.005)**	.838 ± (.005)	.829 ± (.005)	.814 ± (.005)
	.85	**.855 ± (.005)**	.849 ± (.005)	.848 ± (.005)	.814 ± (.005)
	.80	**.867 ± (.005)**	.863 ± (.005)	.859 ± (.005)	.814 ± (.005)
	.75	**.875 ± (.005)**	.872 ± (.005)	.867 ± (.005)	.814 ± (.005)
CSDS1	.99	**.863 ± (.002)**	**.863 ± (.002)**	.862 ± (.002)	.861 ± (.002)
	.95	**.875 ± (.002)**	**.875 ± (.002)**	.873 ± (.002)	.872 ± (.002)
	.90	**.885 ± (.002)**	**.885 ± (.002)**	.882 ± (.002)	.882 ± (.002)
	.85	**.892 ± (.002)**	**.892 ± (.002)**	.890 ± (.002)	.889 ± (.002)
	.80	**.899 ± (.002)**	.898 ± (.002)	.897 ± (.002)	.896 ± (.002)
	.75	**.904 ± (.002)**	**.904 ± (.002)**	.903 ± (.002)	.902 ± (.002)
CSDS2	.99	.982 ± (.002)	.982 ± (.002)	**.984 ± (.002)**	.982 ± (.002)
	.95	**.985 ± (.002)**	.983 ± (.002)	**.985 ± (.002)**	.984 ± (.002)
	.90	**.985 ± (.002)**	.984 ± (.002)	**.986 ± (.002)**	.984 ± (.002)
	.85	**.986 ± (.002)**	.985 ± (.002)	**.986 ± (.002)**	**.986 ± (.002)**
	.80	**.987 ± (.002)**	.986 ± (.002)	**.987 ± (.002)**	.986 ± (.002)
	.75	**.987 ± (.002)**	.986 ± (.002)	**.987 ± (.002)**	**.987 ± (.002)**
CSDS3	.99	**.816 ± (.003)**	**.816 ± (.003)**	.809 ± (.003)	.810 ± (.003)
	.95	**.827 ± (.003)**	.826 ± (.003)	.819 ± (.003)	.821 ± (.003)
	.90	**.841 ± (.003)**	**.841 ± (.003)**	.835 ± (.003)	.836 ± (.003)
	.85	.855 ± (.003)	**.856 ± (.003)**	.849 ± (.003)	.849 ± (.003)
	.80	.869 ± (.003)	**.871 ± (.003)**	.862 ± (.003)	.859 ± (.003)
	.75	.883 ± (.003)	**.884 ± (.003)**	.874 ± (.003)	.872 ± (.003)
CatsVsDogs	.99	.972 ± (.003)	.964 ± (.003)	.980 ± (.003)	**.988 ± (.002)**
	.95	.982 ± (.002)	.963 ± (.003)	.988 ± (.002)	**.990 ± (.002)**
	.90	.989 ± (.002)	.962 ± (.003)	**.993 ± (.002)**	.990 ± (.002)
	.85	**.993 ± (.002)**	.962 ± (.003)	.992 ± (.002)	.990 ± (.002)
	.80	.994 ± (.002)	.960 ± (.003)	**.995 ± (.002)**	.990 ± (.002)
	.75	.994 ± (.002)	.959 ± (.004)	**.997 ± (.002)**	.990 ± (.002)
	#	36/48	24/48	9/48	4/48

Table 3.3: Elapsed Training Time for SCross and baselines.

		Elapsed Training Time			
	c	**SCross**	**PlugIn**	**SelNet**	**SAT**
Adult	.99	1.84	**0.21**	2059.5	1087.9
	.95	1.84	**0.21**	2064.9	1087.9
	.90	1.84	**0.21**	2066.6	1087.9
	.85	1.84	**0.21**	2065.5	1087.9
	.80	1.84	**0.21**	2059.2	1087.9
	.75	1.84	**0.21**	2072.1	1087.9
Lending	.99	13.21	**2.44**	8932.8	7668.3
	.95	13.21	**2.44**	8935.0	7668.3
	.90	13.21	**2.44**	8916.4	7668.3
	.85	13.21	**2.44**	8919.7	7668.3
	.80	13.21	**2.44**	8911.0	7668.3
	.75	13.21	**2.44**	8903.2	7668.3
GiveMe	.99	1.64	**0.27**	2419.8	1364.4
	.95	1.64	**0.27**	2628.4	1364.4
	.90	1.64	**0.27**	2282.7	1364.4
	.85	1.64	**0.27**	2281.2	1364.4
	.80	1.64	**0.27**	2277.6	1364.4
	.75	1.64	**0.27**	2276.4	1364.4
UCICredit	.99	1.31	**0.19**	1719.4	871.9
	.95	1.31	**0.19**	1677.1	871.9
	.90	1.31	**0.19**	1684.3	871.9
	.85	1.31	**0.19**	1675.1	871.9
	.80	1.31	**0.19**	1691.2	871.9
	.75	1.31	**0.19**	1681.0	871.9
CSDS1	.99	6.13	**1.06**	4933.4	2888.3
	.95	6.13	**1.06**	5158.1	2888.3
	.90	6.13	**1.06**	5145.6	2888.3
	.85	6.13	**1.06**	5151.3	2888.3
	.80	6.13	**1.06**	5154.9	2888.3
	.75	6.13	**1.06**	5152.4	2888.3
CSDS2	.99	1.33	**0.23**	668.5	351.4
	.95	1.33	**0.23**	669.4	351.4
	.90	1.33	**0.23**	668.7	351.4
	.85	1.33	**0.23**	668.6	351.4
	.80	1.33	**0.23**	671.0	351.4
	.75	1.33	**0.23**	657.9	351.4
CSDS3	.99	5.25	**0.80**	1316.5	696.1
	.95	5.25	**0.80**	1318.0	696.1
	.90	5.25	**0.80**	1315.2	696.1
	.85	5.25	**0.80**	1306.1	696.1
	.80	5.25	**0.80**	1319.5	696.1
	.75	5.25	**0.80**	1316.4	696.1
CatsVsDogs	.99	18638.2	**3239.8**	3534.7	3767.1
	.95	18638.2	**3239.8**	3580.7	3767.1
	.90	18638.2	**3239.8**	3580.8	3767.1
	.85	18638.2	**3239.8**	3573.2	3767.1
	.80	18638.2	**3239.8**	3586.9	3767.1
	.75	18638.2	**3239.8**	3580.2	3767.1
	#	0/48	**48/48**	0/48	0/48

mances on the image dataset (in 5 out of 6 target coverages): this is partly due to the fact that SelNet and SAT cover considerably 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, 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.

Q2: 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 Tables 3.4 and 3.5. 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.

Q3: Impact of K on 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 3.6 shows the empirical coverage, while Table 3.7 reports 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

Table 3.4: Empirical Coverage for SCross with different base classifiers over tabular datasets $k = 5$, and 1,000 bootstrap runs over the test set, results as mean ± stdev.

		Empirical Coverage			
	c	**Logistic**	**ResNet**	**RandomForest**	**XGBoost**
Adult	.99	.989 ± (.001)	**.991 ± (.001)**	.989 ± (.001)	.992 ± (.001)
	.95	**.950 ± (.002)**	.949 ± (.002)	**.950 ± (.002)**	.951 ± (.002)
	.90	**.901 ± (.003)**	.895 ± (.003)	.899 ± (.003)	.903 ± (.003)
	.85	**.851 ± (.003)**	.847 ± (.003)	.857 ± (.003)	.857 ± (.003)
	.80	**.800 ± (.004)**	**.800 ± (.004)**	.808 ± (.004)	.801 ± (.004)
	.75	.752 ± (.004)	.753 ± (.004)	.755 ± (.004)	**.749 ± (.004)**
Lending	.99	**.992 ± (.001)**	.994 ± (.001)	.996 ± (.001)	.994 ± (.001)
	.95	**.958 ± (.001)**	.969 ± (.001)	.970 ± (.001)	.970 ± (.001)
	.90	**.917 ± (.001)**	.937 ± (.001)	.927 ± (.001)	.939 ± (.001)
	.85	**.878 ± (.001)**	.906 ± (.001)	.893 ± (.001)	.909 ± (.001)
	.80	**.842 ± (.001)**	.875 ± (.001)	.861 ± (.001)	.880 ± (.001)
	.75	**.809 ± (.001)**	.845 ± (.001)	.824 ± (.001)	.853 ± (.001)
GiveMe	.99	.991 ± (.001)	.999 ± (.001)	.991 ± (.001)	**.990 ± (.001)**
	.95	**.949 ± (.002)**	.752 ± (.003)	.952 ± (.002)	**.949 ± (.002)**
	.90	**.901 ± (.002)**	.710 ± (.003)	.905 ± (.002)	.902 ± (.002)
	.85	**.849 ± (.002)**	.653 ± (.003)	.858 ± (.002)	**.849 ± (.002)**
	.80	**.799 ± (.003)**	.586 ± (.003)	**.801 ± (.003)**	**.799 ± (.003)**
	.75	**.750 ± (.003)**	.525 ± (.003)	.767 ± (.003)	.749 ± (.003)
UCICredit	.99	**.992 ± (.002)**	.994 ± (.001)	.995 ± (.001)	.989 ± (.002)
	.95	**.952 ± (.003)**	.957 ± (.003)	.957 ± (.003)	.955 ± (.003)
	.90	.898 ± (.004)	.915 ± (.004)	.906 ± (.004)	**.903 ± (.004)**
	.85	.852 ± (.005)	.907 ± (.004)	**.850 ± (.005)**	.851 ± (.005)
	.80	.798 ± (.005)	.795 ± (.005)	**.800 ± (.005)**	.797 ± (.005)
	.75	.740 ± (.005)	**.748 ± (.005)**	.759 ± (.006)	.745 ± (.006)
CSDS1	.99	**.986 ± (.001)**	.984 ± (.001)	.985 ± (.001)	.980 ± (.001)
	.95	.923 ± (.001)	**.932 ± (.001)**	**.932 ± (.001)**	.921 ± (.001)
	.90	.855 ± (.002)	**.878 ± (.002)**	.871 ± (.002)	.851 ± (.002)
	.85	.793 ± (.002)	.809 ± (.002)	**.820 ± (.002)**	.792 ± (.002)
	.80	.736 ± (.002)	.749 ± (.002)	**.765 ± (.002)**	.735 ± (.002)
	.75	.684 ± (.002)	.691 ± (.002)	**.700 ± (.002)**	.681 ± (.002)
CSDS2	.99	.978 ± (.002)	.968 ± (.002)	.980 ± (.002)	**.984 ± (.002)**
	.95	.902 ± (.003)	.865 ± (.003)	.926 ± (.003)	**.928 ± (.003)**
	.90	.819 ± (.004)	.760 ± (.004)	.853 ± (.004)	**.860 ± (.004)**
	.85	.744 ± (.004)	.664 ± (.005)	**.798 ± (.004)**	.794 ± (.004)
	.80	.672 ± (.005)	.584 ± (.005)	.724 ± (.004)	**.729 ± (.005)**
	.75	.605 ± (.005)	.508 ± (.005)	.619 ± (.005)	**.669 ± (.005)**
CSDS3	.99	**.990 ± (.001)**	**.990 ± (.001)**	.992 ± (.001)	.991 ± (.001)
	.95	**.951 ± (.002)**	**.949 ± (.002)**	.963 ± (.002)	.953 ± (.002)
	.90	**.901 ± (.002)**	.896 ± (.003)	.918 ± (.002)	.903 ± (.002)
	.85	.847 ± (.003)	.845 ± (.003)	.860 ± (.003)	**.852 ± (.003)**
	.80	**.795 ± (.003)**	**.795 ± (.003)**	.816 ± (.003)	**.805 ± (.003)**
	.75	.744 ± (.003)	**.751 ± (.003)**	.772 ± (.003)	.756 ± (.003)
	#	22/42	9/42	9/42	13/42

Table 3.5: Selective Accuracy for SCross with different base classifiers over tabular datasets $k = 5$, and 1,000 bootstrap runs over the test set, results as mean ± stdev.

		Empirical Selective Accuracy			
	c	**Logistic**	**ResNet**	**RandomForest**	**XGBoost**
Adult	.99	.850 ± (.003)	.847 ± (.003)	.841 ± (.004)	**.872 ± (.003)**
	.95	.863 ± (.003)	.861 ± (.003)	.854 ± (.004)	**.886 ± (.003)**
	.90	.879 ± (.003)	.877 ± (.003)	.871 ± (.003)	**.901 ± (.003)**
	.85	.894 ± (.003)	.893 ± (.003)	.885 ± (.003)	**.916 ± (.003)**
	.80	.910 ± (.003)	.906 ± (.003)	.899 ± (.003)	**.933 ± (.003)**
	.75	.923 ± (.003)	.921 ± (.003)	.915 ± (.003)	**.949 ± (.003)**
Lending	.99	.849 ± (.001)	.885 ± (.001)	.877 ± (.001)	**.903 ± (.001)**
	.95	.862 ± (.001)	.894 ± (.001)	.887 ± (.001)	**.913 ± (.001)**
	.90	.878 ± (.001)	.907 ± (.001)	.904 ± (.001)	**.927 ± (.001)**
	.85	.894 ± (.001)	.919 ± (.001)	.918 ± (.001)	**.941 ± (.001)**
	.80	.911 ± (.001)	.931 ± (.001)	.933 ± (.001)	**.955 ± (.001)**
	.75	.927 ± (.001)	.944 ± (.001)	.949 ± (.001)	**.966 ± (.001)**
GiveMe	.99	.936 ± (.002)	.934 ± (.002)	.939 ± (.002)	**.941 ± (.002)**
	.95	.943 ± (.002)	.949 ± (.002)	.953 ± (.002)	**.955 ± (.002)**
	.90	.946 ± (.002)	.952 ± (.002)	.962 ± (.001)	**.966 ± (.001)**
	.85	.949 ± (.002)	.956 ± (.002)	.969 ± (.001)	**.973 ± (.001)**
	.80	.952 ± (.002)	.958 ± (.002)	.974 ± (.001)	**.978 ± (.001)**
	.75	.955 ± (.002)	.959 ± (.002)	.976 ± (.001)	**.981 ± (.001)**
UCICredit	.99	.787 ± (.005)	**.817 ± (.005)**	.811 ± (.005)	.808 ± (.005)
	.95	.801 ± (.005)	**.826 ± (.005)**	.823 ± (.005)	.818 ± (.005)
	.90	.823 ± (.005)	**.837 ± (.005)**	.838 ± (.005)	.836 ± (.005)
	.85	.841 ± (.005)	.839 ± (.005)	**.852 ± (.005)**	.850 ± (.005)
	.80	.852 ± (.005)	**.861 ± (.005)**	.859 ± (.005)	**.861 ± (.005)**
	.75	.858 ± (.005)	.871 ± (.005)	.865 ± (.005)	**.872 ± (.005)**
CSDS1	.99	.861 ± (.002)	.861 ± (.002)	.854 ± (.002)	**.862 ± (.002)**
	.95	.873 ± (.002)	.872 ± (.002)	.865 ± (.002)	**.874 ± (.002)**
	.90	.882 ± (.002)	.880 ± (.002)	.873 ± (.002)	**.884 ± (.002)**
	.85	**.891 ± (.002)**	.889 ± (.002)	.879 ± (.002)	**.891 ± (.002)**
	.80	.896 ± (.002)	.896 ± (.002)	.885 ± (.002)	**.897 ± (.002)**
	.75	.902 ± (.002)	.901 ± (.002)	.889 ± (.002)	**.903 ± (.002)**
CSDS2	.99	**.983 ± (.002)**	**.983 ± (.002)**	**.983 ± (.002)**	.982 ± (.002)
	.95	.984 ± (.002)	**.985 ± (.002)**	.984 ± (.002)	.983 ± (.002)
	.90	.985 ± (.002)	**.986 ± (.002)**	.985 ± (.002)	.984 ± (.002)
	.85	.986 ± (.002)	**.987 ± (.002)**	.986 ± (.002)	.984 ± (.002)
	.80	.986 ± (.002)	**.988 ± (.002)**	.986 ± (.002)	.984 ± (.002)
	.75	.986 ± (.002)	**.988 ± (.002)**	.987 ± (.002)	.985 ± (.002)
CSDS3	.99	.811 ± (.003)	.808 ± (.003)	.810 ± (.003)	**.812 ± (.003)**
	.95	**.823 ± (.003)**	.820 ± (.003)	.818 ± (.003)	**.823 ± (.003)**
	.90	**.838 ± (.003)**	.835 ± (.003)	.831 ± (.003)	**.838 ± (.003)**
	.85	.854 ± (.003)	.851 ± (.003)	.847 ± (.003)	**.855 ± (.003)**
	.80	**.869 ± (.003)**	.865 ± (.003)	.859 ± (.003)	.868 ± (.003)
	.75	**.882 ± (.003)**	.876 ± (.003)	.873 ± (.003)	**.882 ± (.003)**
	#	6/42	10/42	2/42	31/42

Table 3.6: Empirical coverage results for SCross with different K, and 1,000 bootstrap runs over the test set, results as mean ± stdev.

		Empirical Coverage				
	c	$\boldsymbol{K=2}$	$\boldsymbol{K=3}$	$\boldsymbol{K=5}$	$\boldsymbol{K=7}$	$\boldsymbol{K=10}$
Adult	.99	.992 ± (.001)	**.991 ± (.001)**	.993 ± (.001)	.992 ± (.001)	.992 ± (.001)
	.95	**.950 ± (.002)**	.949 ± (.002)	**.950 ± (.002)**	**.951 ± (.002)**	**.951 ± (.002)**
	.90	.906 ± (.003)	**.904 ± (.003)**	**.904 ± (.003)**	.904 ± (.003)	.904 ± (.003)
	.85	.849 ± (.003)	.850 ± (.003)	.850 ± (.003)	**.850 ± (.003)**	**.850 ± (.003)**
	.80	.798 ± (.004)	**.801 ± (.004)**	.802 ± (.004)	.802 ± (.004)	**.801 ± (.004)**
	.75	.745 ± (.004)	**.747 ± (.004)**	.746 ± (.004)	.746 ± (.004)	.747 ± (.004)
Lending	.99	.995 ± (.001)	.995 ± (.001)	**.994 ± (.001)**	.995 ± (.001)	.994 ± (.001)
	.95	.970 ± (.001)	.969 ± (.001)	**.969 ± (.001)**	.970 ± (.001)	.970 ± (.001)
	.90	.939 ± (.001)	.938 ± (.001)	**.938 ± (.001)**	.938 ± (.001)	.938 ± (.001)
	.85	.909 ± (.001)	.908 ± (.001)	**.908 ± (.001)**	.908 ± (.001)	.908 ± (.001)
	.80	.880 ± (.001)	.879 ± (.001)	.879 ± (.001)	.879 ± (.001)	**.878 ± (.001)**
	.75	.850 ± (.001)	.849 ± (.001)	.849 ± (.001)	.849 ± (.001)	**.849 ± (.001)**
GiveMe	.99	**.990 ± (.001)**	**.990 ± (.001)**	.990 ± (.001)	.990 ± (.001)	**.990 ± (.001)**
	.95	.948 ± (.002)	.949 ± (.002)	**.950 ± (.002)**	.950 ± (.002)	**.950 ± (.002)**
	.90	**.901 ± (.002)**	**.901 ± (.002)**	.902 ± (.002)	**.901 ± (.002)**	.902 ± (.002)
	.85	.853 ± (.002)	.852 ± (.002)	**.852 ± (.002)**	.852 ± (.002)	.853 ± (.002)
	.80	.797 ± (.003)	.799 ± (.003)	**.800 ± (.003)**	**.800 ± (.003)**	**.801 ± (.003)**
	.75	.744 ± (.003)	.746 ± (.003)	.748 ± (.003)	.747 ± (.003)	**.749 ± (.003)**
UCICredit	.99	**.992 ± (.002)**	.993 ± (.001)	.993 ± (.001)	.993 ± (.001)	.993 ± (.001)
	.95	**.950 ± (.003)**	**.951 ± (.003)**	**.951 ± (.003)**	.953 ± (.003)	.953 ± (.003)
	.90	**.903 ± (.004)**	.906 ± (.004)	.906 ± (.004)	.907 ± (.004)	**.903 ± (.004)**
	.85	.849 ± (.005)	**.851 ± (.005)**	.848 ± (.005)	.853 ± (.005)	.849 ± (.005)
	.80	.795 ± (.005)	.795 ± (.005)	.797 ± (.005)	**.801 ± (.005)**	.797 ± (.005)
	.75	.745 ± (.005)	.745 ± (.005)	**.750 ± (.005)**	.754 ± (.005)	**.751 ± (.005)**
CSDS1	.99	.979 ± (.001)	.979 ± (.001)	.980 ± (.001)	**.981 ± (.001)**	.981 ± (.001)
	.95	.917 ± (.001)	**.918 ± (.001)**	.917 ± (.001)	.917 ± (.001)	.917 ± (.001)
	.90	**.850 ± (.002)**	.849 ± (.002)	.849 ± (.002)	.848 ± (.002)	.849 ± (.002)
	.85	**.792 ± (.002)**	.791 ± (.002)	.790 ± (.002)	.790 ± (.002)	.790 ± (.002)
	.80	**.739 ± (.002)**	.737 ± (.002)	.736 ± (.002)	.736 ± (.002)	.736 ± (.002)
	.75	**.684 ± (.002)**	.682 ± (.002)	.682 ± (.002)	.682 ± (.002)	.682 ± (.002)
CSDS2	.99	.983 ± (.002)	.983 ± (.002)	.983 ± (.002)	.984 ± (.002)	**.984 ± (.002)**
	.95	.902 ± (.003)	.915 ± (.003)	.916 ± (.003)	**.919 ± (.003)**	.918 ± (.003)
	.90	.791 ± (.004)	.824 ± (.004)	.834 ± (.004)	**.836 ± (.004)**	**.836 ± (.004)**
	.85	.686 ± (.005)	.736 ± (.004)	.761 ± (.004)	**.765 ± (.004)**	.764 ± (.004)
	.80	.597 ± (.005)	.657 ± (.005)	.687 ± (.005)	**.692 ± (.005)**	.691 ± (.005)
	.75	.521 ± (.005)	.586 ± (.005)	.620 ± (.005)	.623 ± (.005)	**.624 ± (.005)**
CSDS3	.99	**.990 ± (.001)**	**.990 ± (.001)**	**.991 ± (.001)**	.991 ± (.001)	**.991 ± (.001)**
	.95	.953 ± (.002)	**.953 ± (.002)**	.955 ± (.002)	.955 ± (.002)	.955 ± (.002)
	.90	**.907 ± (.002)**	.907 ± (.002)	.911 ± (.002)	.910 ± (.002)	.910 ± (.002)
	.85	**.861 ± (.003)**	.862 ± (.003)	.865 ± (.003)	.865 ± (.003)	.863 ± (.003)
	.80	**.813 ± (.003)**	.815 ± (.003)	.818 ± (.003)	.817 ± (.003)	.817 ± (.003)
	.75	**.766 ± (.003)**	.768 ± (.003)	.769 ± (.003)	.769 ± (.003)	.768 ± (.003)
CatsVsDogs	.99	.997 ± (.001)	.992 ± (.002)	.992 ± (.002)	**.991 ± (.002)**	**.990 ± (.002)**
	.95	.976 ± (.003)	.963 ± (.003)	.960 ± (.003)	**.950 ± (.004)**	.948 ± (.004)
	.90	.956 ± (.003)	.926 ± (.004)	.922 ± (.004)	**.901 ± (.005)**	.889 ± (.005)
	.85	.938 ± (.004)	.884 ± (.005)	.889 ± (.005)	**.855 ± (.005)**	.793 ± (.006)
	.80	.923 ± (.004)	.833 ± (.006)	.846 ± (.005)	**.774 ± (.006)**	.718 ± (.007)
	.75	.909 ± (.005)	**.775 ± (.006)**	.794 ± (.006)	.691 ± (.007)	.658 ± (.007)
	#	15/48	12/48	12/48	15/48	16/48

Table 3.7: Selective Accuracy results for SCross with different K, and 1,000 bootstrap runs over the test set, results as mean ± stdev.

		Empirical Selective Accuracy				
	c	$K = 2$	$K = 3$	$K = 5$	$K = 7$	$K = 10$
Adult	.99	.872 ± (.003)	**.872 ± (.003)**	.871 ± (.003)	**.872 ± (.003)**	**.872 ± (.003)**
	.95	**.888 ± (.003)**	**.888 ± (.003)**	**.888 ± (.003)**	.887 ± (.003)	.887 ± (.003)
	.90	.901 ± (.003)	**.902 ± (.003)**	**.902 ± (.003)**	**.902 ± (.003)**	**.902 ± (.003)**
	.85	**.920 ± (.003)**	.920 ± (.003)	.919 ± (.003)	.919 ± (.003)	.919 ± (.003)
	.80	**.935 ± (.003)**	.934 ± (.003)	.934 ± (.003)	.934 ± (.003)	.934 ± (.003)
	.75	**.950 ± (.003)**	.949 ± (.003)	.950 ± (.003)	.949 ± (.003)	.949 ± (.003)
Lending	.99	**.899 ± (.001)**	**.899 ± (.001)**	**.899 ± (.001)**	**.899 ± (.001)**	**.899 ± (.001)**
	.95	**.910 ± (.001)**	.910 ± (.001)	**.910 ± (.001)**	.910 ± (.001)	.910 ± (.001)
	.90	.923 ± (.001)	.924 ± (.001)	**.924 ± (.001)**	.924 ± (.001)	.924 ± (.001)
	.85	.937 ± (.001)	.937 ± (.001)	**.938 ± (.001)**	.937 ± (.001)	**.938 ± (.001)**
	.80	.950 ± (.001)	.951 ± (.001)	.951 ± (.001)	.951 ± (.001)	**.951 ± (.001)**
	.75	.962 ± (.001)	.963 ± (.001)	.963 ± (.001)	.963 ± (.001)	**.963 ± (.001)**
GiveMe	.99	.942 ± (.002)	.942 ± (.002)	**.942 ± (.002)**	**.942 ± (.002)**	**.942 ± (.002)**
	.95	**.957 ± (.002)**	.957 ± (.002)	.956 ± (.002)	.956 ± (.002)	.956 ± (.002)
	.90	**.967 ± (.001)**	.967 ± (.001)	.967 ± (.001)	.967 ± (.001)	**.967 ± (.001)**
	.85	**.973 ± (.001)**	.973 ± (.001)	**.973 ± (.001)**	.973 ± (.001)	.973 ± (.001)
	.80	**.978 ± (.001)**	**.978 ± (.001)**	.978 ± (.001)	.978 ± (.001)	.978 ± (.001)
	.75	**.981 ± (.001)**	**.981 ± (.001)**	**.981 ± (.001)**	**.981 ± (.001)**	**.981 ± (.001)**
UCICredit	.99	**.814 ± (.005)**	.814 ± (.005)	.814 ± (.005)	.814 ± (.005)	.814 ± (.005)
	.95	**.827 ± (.005)**	**.827 ± (.005)**	**.827 ± (.005)**	.826 ± (.005)	.826 ± (.005)
	.90	**.840 ± (.005)**	.839 ± (.005)	.839 ± (.005)	.839 ± (.005)	**.840 ± (.005)**
	.85	**.855 ± (.005)**	.854 ± (.005)	**.855 ± (.005)**	.854 ± (.005)	**.855 ± (.005)**
	.80	**.867 ± (.005)**	**.867 ± (.005)**	**.867 ± (.005)**	.866 ± (.005)	**.867 ± (.005)**
	.75	.875 ± (.005)	**.876 ± (.005)**	.875 ± (.005)	.875 ± (.005)	.875 ± (.005)
CSDS1	.99	**.863 ± (.002)**	**.863 ± (.002)**	**.863 ± (.002)**	.862 ± (.002)	.863 ± (.002)
	.95	**.875 ± (.002)**	**.875 ± (.002)**	.875 ± (.002)	.875 ± (.002)	.875 ± (.002)
	.90	**.885 ± (.002)**	.885 ± (.002)	.885 ± (.002)	.885 ± (.002)	.885 ± (.002)
	.85	.892 ± (.002)	.892 ± (.002)	**.892 ± (.002)**	**.892 ± (.002)**	**.892 ± (.002)**
	.80	.899 ± (.002)	**.899 ± (.002)**	**.899 ± (.002)**	**.899 ± (.002)**	**.899 ± (.002)**
	.75	**.904 ± (.002)**	**.904 ± (.002)**	**.904 ± (.002)**	**.904 ± (.002)**	**.904 ± (.002)**
CSDS2	.99	**.983 ± (.002)**	**.983 ± (.002)**	**.982 ± (.002)**	**.983 ± (.002)**	**.983 ± (.002)**
	.95	**.985 ± (.002)**	.985 ± (.002)	.985 ± (.002)	.984 ± (.002)	.985 ± (.002)
	.90	**.986 ± (.002)**	.986 ± (.002)	.985 ± (.002)	.985 ± (.002)	.985 ± (.002)
	.85	**.987 ± (.002)**	.986 ± (.002)	.986 ± (.002)	.986 ± (.002)	.986 ± (.002)
	.80	**.988 ± (.002)**	.987 ± (.002)	.987 ± (.002)	.987 ± (.002)	.987 ± (.002)
	.75	**.987 ± (.002)**	**.988 ± (.002)**	**.987 ± (.002)**	**.988 ± (.002)**	**.988 ± (.002)**
CSDS3	.99	**.817 ± (.003)**	**.817 ± (.003)**	.816 ± (.003)	.816 ± (.003)	**.817 ± (.003)**
	.95	**.828 ± (.003)**	**.828 ± (.003)**	.827 ± (.003)	.827 ± (.003)	.827 ± (.003)
	.90	**.842 ± (.003)**	**.842 ± (.003)**	.841 ± (.003)	.841 ± (.003)	.841 ± (.003)
	.85	**.856 ± (.003)**	**.856 ± (.003)**	.855 ± (.003)	.855 ± (.003)	.856 ± (.003)
	.80	**.870 ± (.003)**	.870 ± (.003)	.869 ± (.003)	.869 ± (.003)	.869 ± (.003)
	.75	**.884 ± (.003)**	.883 ± (.003)	.883 ± (.003)	.883 ± (.003)	.883 ± (.003)
CatsVsDogs	.99	.971 ± (.003)	.971 ± (.003)	.972 ± (.003)	.972 ± (.003)	**.972 ± (.003)**
	.95	.978 ± (.003)	.980 ± (.003)	**.982 ± (.002)**	.982 ± (.002)	.982 ± (.002)
	.90	.983 ± (.002)	.987 ± (.002)	.989 ± (.002)	.991 ± (.002)	**.992 ± (.002)**
	.85	.988 ± (.002)	.992 ± (.002)	.993 ± (.002)	.994 ± (.002)	**.994 ± (.002)**
	.80	.990 ± (.002)	.994 ± (.002)	.994 ± (.002)	.994 ± (.002)	**.995 ± (.002)**
	.75	.991 ± (.002)	.994 ± (.002)	.994 ± (.002)	.995 ± (.002)	**.996 ± (.002)**
	#	32/48	19/48	19/48	10/48	23/48

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 is a good option to trade off the contrasting metrics of empirical coverage, selective accuracy, and elapsed training time.

3.4 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 the analyzed tabular data.

Chapter 4

Model-Agnostic Heuristics for Selective Regression

In this chapter, we address *Req. 1*, by devising novel approaches to perform selective regression in a model-agnostic way. We propose a few algorithms to achieve such a goal, by building on state-of-the-art uncertainty quantification techniques. We extensively evaluate the proposed methods over a large and diverse set of data, showing the effectiveness of our approach. The chapter is based on a working paper, which is currently under review.

4.1 Introduction

Selective prediction allows machine learning systems to refrain from forecasting, adding the option to abstain whenever the risk of mispredicting is too high.
While selective classification has received considerable attention in the past decade, the problem of selective regression remains relatively understudied. Existing approaches for selective regression primarily rely on either deep-learning-based methods [15, 115] or on the ability to estimate the conditional variance function [26]. As a result, a few concerns might arise: for instance, deep neural network methods are outperformed by tree-based methods when one has to deal with tabular data, which are the *de facto* standard in several industry domains. As a consequence, methods relying on deep neural network architectures could be suboptimal. Moreover, the model-agnostic state-of-the-art method proposed by Zaoui et al. [26] estimates both the regression function and the conditional variance on the same training data. This double usage of training data is known as "double-dipping" and it is prone to overfitting concerns [116].
Furthermore, due to the limited scope of empirical evaluations in current state-of-the-art methods, there is a lack of practical insights into the effectiveness and applicability of existing techniques, hindering the development of robust and reliable selective regression models.
This chapter aims to address these concerns by presenting a few model-agnostic methods that rely on uncertainty quantification techniques. These methods

are tested over a large and diverse set of 93 datasets. Our extensive experimental evaluation shows that the proposed methods outperform existing model-agnostic baselines.

4.2 Current Limitations

Let us recall that $f(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$ is the regression function and $\sigma^2(\mathbf{x}) = \mathbb{E}[(Y - f(\mathbf{x}))^2 \mid \mathbf{X} = \mathbf{x}]$ is the conditional variance function.
As recalled in Section 2.2.1, Zaoui et al. [26] show that an optimal selective regressor[1] can be built by thresholding the conditional variance function $\sigma^2(\mathbf{x})$, which measures the uncertainty around the prediction.
Given a training set $\mathcal{T}_{train}$, their approach (PlugIn-REG) estimates a regressor $\hat{f}$ on $\mathcal{T}_{train}$ by minimizing:

$$\frac{1}{n_{train}} \sum_{i \in \mathcal{T}_{train}} (y_i - \hat{f}(\mathbf{x}_i))^2, \tag{4.1}$$

where n_{train} refers to the number of instances in the training set $\mathcal{T}_{train}$. Then, it estimates the conditional variance function $\hat{\sigma}^2(\mathbf{x}) \approx \mathbb{E}\left[(Y - f(\mathbf{X}))^2 \mid \mathbf{X} = \mathbf{x}\right]$ by fitting another regression using the training set squared residuals $e_i^2 = \left(y_i - \hat{f}(\mathbf{x}_i)\right)^2$ as the target variable.
This procedure might introduce biases in the estimation, as shown for instance by [117]. This is because the true residuals $e_i = y_i - f(\mathbf{x}_i)$ become correlated with $f(\mathbf{x}_i) - \hat{f}(\mathbf{x}_i)$. As a result, the observed residuals $y - \hat{f}(\mathbf{x}_i)$ have an artificially low variance. More precisely, the bias introduced by equation 4.1 is

$$\begin{aligned}\mathbb{E}\left[(Y - \hat{f})^2\right] - \mathbb{E}\left[(Y - f)^2\right] &= \mathbb{E}\left[(Y - f + f - \hat{f})^2\right] - \mathbb{E}\left[(Y - f)^2\right] = \\ &= -2\mathbb{E}\left[(Y - f)(\hat{f} - f)\right] + \mathbb{E}\left[(f - \hat{f})^2\right]\end{aligned} \tag{4.2}$$

The first term represents the bias deriving from estimating both $\hat{f}$ and $\hat{\sigma}^2$ on the same training data. Such a concern is especially relevant whenever we employ regularized estimators, e.g. boosting or penalized neural networks, because the regularization prevents the variance from exploding, but it introduces biases in the estimators [105]. A possible solution to this challenge is to use a different sample to estimate the two functions. In this case, the bias term goes to zero [117, 116]. This observation motivates the choices of our proposed algorithms to perform selective regression.

4.3 Model-Agnostic Heuristics

Since the usage of the same training data for estimating both the regressor and the conditional variance function is prone to overfitting concerns, we propose to

[1] This is the optimal strategy when considering the Mean Squared Error (MSE) loss function, defined as $MSE = \mathbb{E}\left[(Y - f(\mathbf{X}))^2\right]$

build selective predictors exploiting widely known sample splitting strategies, i.e. cross-fitting and non-parametric bootstrap.

4.3.1 Cross-fitting for Selective Regression

To mitigate the bias shown in equation 4.2, we first consider cross-fitting. This strategy is widely used in causal inference [105, 116] and conformal prediction [87] to handle overfitting issues. Moreover, the approach presented in Chapter 3 was shown to be successful in building model-agnostic selective classifiers. We proposed two variants of cross-fitting, as illustrated in Alg. 2 and 3, respectively. We call the two variants SCross-ENS and SCross-POOL. Both approaches require a labeled training set $\mathcal{T}_{train}$ to train the selective regressor and an unlabeled validation set $\mathcal{T}_{val}$ to calibrate the selection function to target coverage c. SCross-ENS divides the training set in two folds: one is used to estimating a regression function, while the other one for the conditional variance function. Then the role of the folds is swapped to regain full efficiency [105]. One can finally obtain the final estimators by averaging the trained models (lines 7-8). As for the PlugIn-REG approach, one can calibrate the selection function using a held-out dataset (lines 10-11). SCross-POOL is a modification of the same principle. Rather than considering two folds, it splits the available data into K folds. Second, it trains a regression function over $K-1$ folds and predicts the squared residuals over the remaining K-th fold (lines 2-6). Third, it stacks these predicted residual values altogether and uses them to train the conditional variance function (lines 8-9). Finally, a calibration set is used to estimate the final selection function (lines 10-12). For both strategies, the residuals are estimated using a different set, which is independent from the one used to train the regressor. This mitigates the risks of overfitting deriving from double dipping the training data.

4.3.2 Bootstrap for Selective Regression

Another strategy relies on a different sample splitting approach, namely non-parametric bootstrap. We will consider a variation of the state-of-the-art Doubt uncertainty estimation method by Mougan and Nielsen [76], which is sketched by Algorithm 4 and briefly described here for completeness. First, the authors assume the following relationship between the true values and the model predictions:

$$Y = f(\mathbf{X}) + e(\mathbf{X}) \tag{4.3}$$

Here $e(\mathbf{X})$ is assumed to have zero mean and with a finite variance. Second, they split $e(\mathbf{X})$ into *model variance noise*, *observation noise* and *model bias*. Third, they estimate these different parts using bootstrap samples (lines 4-21, Alg. 4), avoiding the double usage of the training data. Finally, they combine these estimates in a prediction set $C(\mathbf{X})$. Under mild assumptions on f, Mougan and Nielsen [76] show that the obtained $C(\mathbf{X})$ converges in distribution to $e(\mathbf{X})$ as the number of training samples and bootstraps tend to infinity. This is because Doubt's prediction set shares similar asymptotic

Algorithm 2: SCross-ENS.fit() (ensemble)

Input : $\mathcal{T}_{train} = \{\mathbf{X}_{train}, \mathbf{y}_{train}\}$ - training set,
$\mathcal{T}_{val} = \{\mathbf{X}_{val}\}$ - unlabelled validation set,
$\mathcal{S}$ - base regression algorithm,
$\mathcal{V}$ - base regression algorithm for variance
c - target coverage

Output: (f, g) - selective regressor

1 $\mathcal{T}_1, \ldots, \mathcal{T}_K \leftarrow \text{KFold}(\mathcal{T}_{train}, 2)$ // partitioning in two folds
2 **for** $k \in 1, 2$ **do** // for each fold
3 $\quad \mathcal{T}_{-k} \leftarrow \mathcal{T} \setminus \mathcal{T}_k$ // training data
4 $\quad f_k \leftarrow \mathcal{S}.fit(\mathcal{T}_{-k})$ // train k^{th} regressor
5 $\quad e_k \leftarrow (y_k - f_k.score(\mathbf{X}_k))^2$ // residual over k−th fold
6 $\quad \sigma_k^2 \leftarrow \mathcal{V}.fit(\mathbf{X}_k, e_k)$ // fit conditional variance σ_k^2
7 $f.score \leftarrow$ **lambda** $\mathbf{x} : \frac{1}{2}\sum_{k=1}^{2} f_k.score(\mathbf{x})$ // regression function
8 $\sigma^2.score \leftarrow$ **lambda** $\mathbf{x} : \frac{1}{2}\sum_{k=1}^{2} \sigma_k^2.score(\mathbf{x})$ // conditional variance
9 $\boldsymbol{\sigma^2}_{val} \leftarrow \sigma^2.score(\mathbf{X}_{val})$ // compute σ^2 values on validation
10 $\hat{\tau} \leftarrow \text{perc}(\boldsymbol{\sigma^2}_{val}, 100(c))$ // calculate $\hat{\tau}$
11 $g \leftarrow$ **lambda** $\mathbf{x} : \mathbb{1}\{\mathbf{x} : \sigma^2.score(\mathbf{x}) \leq \hat{\tau}\}$ // compute g
12 **return** (f, g)

properties of the prediction interval introduced in Kumar and Srivastava [118, Section 4.1.3].

Hence, as for cross-fitting methods, we devise two variants relying on Doubt.

First, we notice that since convergence in distribution implies convergence of quantiles (see, e.g., Theorem 2A in Parzen [119]), the above implies that the resulting prediction intervals built using the quantiles of $C(\mathbf{X})$ are asymptotically correct. Consequently, we consider the width of the prediction interval obtained from $C(\mathbf{X})$ as a confidence measure to build the final selection function. This is because, the larger the interval, the more uncertainty around the prediction. Thus, abstention should be preferable. We call this approach Doubt-Int, which is detailed in Algorithm 5. The inputs are a training set $\mathcal{T}_{train}$, a validation set $\mathcal{T}_{val}$, a base regression algorithm $\mathcal{S}$, the number of bootstrap samples B, the (l, u) quantiles for the interval width, and a target coverage c. Doubt-Int fits the Doubt algorithm over the training set (line 1). After Doubt is fitted, Doubt-Int computes the interval width of the prediction set $C(\mathbf{X})$ (lines 2-4) for each instance in the validation set. The obtained interval widths are used to estimate a threshold $\hat{\tau}$ (line 5). Finally, Doubt-Int builds the selection function as follows (line 6): if an instance has an interval width larger than the estimated $\hat{\tau}$, it is rejected, otherwise the prediction is provided.

As a second option, we consider a slight modification to Doubt-Int, called Doubt-Var, which is detailed in Algorithm 6. The algorithm follows the same steps as Doubt-Int, but rather than considering the interval width as a measure of uncertainty, it computes the variance of the prediction set $C(\mathbf{X})$

Algorithm 3: SCross-POOL.fit() (pooled)

Input : $\mathcal{T}_{train} = \{\mathbf{X}_{train}, \mathbf{y}_{train}\}$ - training set,
$\mathcal{T}_{val} = \{\mathbf{X}_{val}\}$ - unlabelled validation set,
$\mathcal{S}$ - base regression algorithm,
$\mathcal{V}$ - base regression algorithm for variance
c - target coverage
K - number of folds

Output: (f, g) - selective regressor

1 $\mathcal{T}_1, \ldots, \mathcal{T}_K \leftarrow \text{StratifiedKFold}((\mathbf{X}, \mathbf{y}), K)$ // partitioning
2 **for** $k \in 1, \ldots, K$ **do** // for each fold
3 $\quad \mathcal{T}_{-k} \leftarrow \mathcal{T} \setminus \mathcal{T}_k$ // training data
4 $\quad f_k \leftarrow \mathcal{S}.fit(\mathcal{T}_{-k})$ // train k^{th} regressor
5 $\quad e_k \leftarrow (y_k - f_k.score(\mathcal{T}_k))^2$ // residual over k−th fold
6 $\quad \sigma_k^2 \leftarrow \mathcal{V}.fit(\mathbf{X}_k, e_k)$ // fit conditional variance σ_k^2
7 $e \leftarrow \cup_{k=1}^{K} e_k$ // stack all residuals
8 $f \leftarrow \mathcal{S}.fit(\mathcal{T}_{train})$ // estimate f
9 $\sigma^2 \leftarrow \mathcal{V}.fit(\mathbf{X}_{train}, e)$ // estimate σ^2
10 $m \leftarrow \sigma^2.score(\mathbf{X}_{val})$ // compute σ^2 values on validation
11 $\hat{\tau} \leftarrow \text{perc}(m, 100(c))$ // calculate $\hat{\tau}$
12 $g \leftarrow$ **lambda** $\mathbf{x} : \mathbb{1}\{\mathbf{x} : \sigma^2.score(\mathbf{x}) \leq \hat{\tau}\}$ // compute g
13 **return** (f, g)

(lines 2-4). The intuition is similar: the larger the variance of $C(\mathbf{X})$, the more uncertainty around the prediction, thus abstention is preferable. In our empirical evaluation, we show that this heuristics is a valid alternative to the PlugIn-REG approach proposed by Zaoui et al. [26].

4.4 Experimental Evaluation

4.4.1 Research Questions

We present in this section the experiments to validate our proposed methods, addressing the following questions:

Q1 Do the proposed approaches achieve state-of-the-art performance for selective regression in terms of Mean Squared Error?

Q2 Do the proposed approaches achieve state-of-the-art performance for selective regression in terms of Coverage Violations?

Q3 Are the results to **Q1** affected by the regression algorithm used for prediction?

Algorithm 4: DOUBT.fit() [76]

Input : $\mathcal{T}_{train} = \{\mathbf{X}_{train}, \mathbf{y}_{train}\}$ - training set,
$\mathcal{S}$ - base regression algorithm,
B - number of bootstraps
Output: f - fitted regressor
C - errors prediction set

1 $f \leftarrow \mathcal{S}.fit(\mathcal{T}_{train})$ // train the regressor on $\mathcal{T}_{train}$
2 $\mathbf{e}_{train} \leftarrow \mathbf{y}_{train} - f.score(\mathbf{X}_{train})$ // compute training residuals
3 $\mathcal{T}_1, \ldots, \mathcal{T}_B \leftarrow \text{Bootstrap}(\mathcal{T}_{train}, B)$ // bootstrap samples
4 **for** $b \in 1, \ldots, B$ **do** // for each fold
5 $\quad \mathcal{T}_{-b} \leftarrow \mathcal{T} \setminus \mathcal{T}_b$ // validation data
6 $\quad f_b \leftarrow \mathcal{S}.fit(\mathcal{T}_b)$ // train b^{th} regressor
7 $\quad \mathbf{e}_{-b} \leftarrow (\mathbf{y}_{-b} - f_b.score(\mathbf{X}_{-b}))$ // residuals for $\mathcal{T}_{-b}$
8 $\mathbf{e}_{boot} \leftarrow [\mathbf{e}_{-1}, \ldots, \mathbf{e}_{-B}]$ // stack all bootstrap residuals
9 **for** $i \in 1, \ldots, 99$ **do** // for each percentile
10 $\quad q_{i,boot} \leftarrow \text{perc}(\mathbf{e}_{boot}, i)$ // compute quantile i for $\mathbf{e}_{boot}$
11 $\quad q_{i,train} \leftarrow \text{perc}(\mathbf{e}_{train}, i)$ // compute quantile i for $\mathbf{e}_{train}$
12 $\mathbf{q}_{train} \leftarrow [q_{1,train}, \ldots, q_{99,train}]$ // store train quantiles
13 $\mathbf{q}_{boot} \leftarrow [q_{1,boot}, \ldots, q_{99,boot}]$ // store bootstraps quantiles
14 $\texttt{NoInfo} \leftarrow {}^{1}/_{n_{train}^2} \sum_{i=1}^{n_{train}} \sum_{j=1}^{n_{train}} (y_i - f.score(\mathbf{x}_i))^2$ // no info coefficient
15 $\texttt{num} \leftarrow |mean(\mathbf{q}_{train}) - mean(\mathbf{q}_{boot})|$ // numerator overfitting rate
16 $\texttt{den} \leftarrow |\texttt{NoInfo} - mean(\mathbf{q}_{train})|$ // denominator overfitting rate
17 $\texttt{OverFitRate} \leftarrow \frac{\texttt{num}}{\texttt{den}}$ // compute overfitting rate
18 $w \leftarrow \frac{.632}{1-0.368*\texttt{OverFitRate}}$ // compute adjusted .632 bootstrap weight
19 $\mathbf{res} \leftarrow (1-w) * \mathbf{q}_{train} + w * \mathbf{q}_{boot}$ // estimate bias and observation noise
20 $\bar{f}_B \leftarrow \textbf{lambda } \mathbf{x} : {}^{1}/_{B} \sum_{b=1}^{B} f_b.score(\mathbf{x})$ // mean of f_b regressors
21 $\tilde{e}_b \leftarrow f_b.score(\mathbf{x}) - \bar{f}_B.score(\mathbf{x})$ // estimate model variance noise
22 $\tilde{\mathbf{e}} \leftarrow [\tilde{e}_1, \ldots \tilde{e}_B]$ // store all model variance noises predictors
23 **return** $f, \tilde{\mathbf{e}}, \mathbf{res}$

Algorithm 5: DOUBT-INT.fit()

Input : $\mathcal{T}_{train} = \{\mathbf{X}_{train}, \mathbf{y}_{train}\}$ - training set,
$\mathcal{S}$ - base regression algorithm,
B - number of bootstraps,
(l, u) - quantiles for prediction intervals,
$\mathcal{T}_{val} = \{\mathbf{X}_{val}\}$ - unlabelled validation set,
c - target coverage
Output: (f, g) - selective regressor

1 $f, \tilde{\mathbf{e}}, \mathbf{res} \leftarrow \text{DOUBT}.fit(\mathcal{T}_{train}, \mathcal{S}, B)$ // train DOUBT on $\mathcal{T}_{train}$
2 $C \leftarrow \textbf{lambda } \mathbf{x} : \tilde{\mathbf{e}}.score(\mathbf{x}) + \mathbf{res}$ // return the prediction error set
3 $C.interval(\mathbf{x}, l, u) \leftarrow \textbf{lambda } \mathbf{x} : \text{perc}(C(\mathbf{x}), u) - \text{perc}(C(\mathbf{x}), l)$
// return prediction interval width for x
4 $C_{val} \leftarrow C.interval(\mathbf{X}_{val}, l, u)$ // compute widths for all $\mathbf{X}_{val}$
5 $\hat{\tau} \leftarrow \text{perc}(C_{val}, 100(c))$ // calculate $\hat{\tau}$
6 $g \leftarrow \textbf{lambda } \mathbf{x} : \mathbb{1}\{\mathbf{x} : C.interval(\mathbf{x}, l, u) \leq \hat{\tau}\}$ // compute g
7 **return** (f, g)

4.4.2 Experimental Settings

Datasets. We consider 93 tabular regression datasets, comprehending real data from existing regression benchmarks [120, 103] and the 51 US states

Algorithm 6: DOUBT-VAR.fit()

Input : $\mathcal{T}_{train} = \{\mathbf{X}_{train}, \mathbf{y}_{train}\}$ - training set,
$\mathcal{S}$ - base regression algorithm,
B - number of bootstraps,
$\mathcal{T}_{val} = \{\mathbf{X}_{val}\}$ - unlabelled validation set,
c - target coverage

Output: (f, g) - selective regressor

1. $f, \tilde{\mathbf{e}}, \mathbf{res} \leftarrow$ DOUBT.$fit(\mathcal{T}_{train}, \mathcal{S}, B)$ // train DOUBT on $\mathcal{T}_{train}$
2. $C \leftarrow$ **lambda** $\mathbf{x} : \tilde{\mathbf{e}}.score(\mathbf{x}) + \mathbf{res}$ // return the prediction error set
3. $C.variance(\mathbf{x}) \leftarrow$ **lambda** $\mathbf{x} : \text{variance}(C(\mathbf{x}))$ // return the variance of $C(\mathbf{x})$
4. $C_{val} \leftarrow C.variance(\mathbf{X}_{val})$ // compute variance for all instances in $\mathbf{X}_{val}$
5. $\hat{\tau} \leftarrow \text{perc}(C_{val}, 100(c))$ // calculate $\hat{\tau}$
6. $g \leftarrow$ **lambda** $\mathbf{x} : \mathbb{1}\{\mathbf{x} : C.variance(\mathbf{x}) \leq \hat{\tau}\}$ // compute g
7. **return** (f, g)

data in `folktables` [121]. We include applications from different domains, such as finance, healthcare, and natural sciences. To limit the computational effort, we follow the approach of [103] by limiting the training sample size to $50,000$ instances. For larger datasets, we randomly sample instances without replacement. We additionally exclude datasets with a size $n \leq 1000$, to have enough data to test different target coverages. We further detail the datasets' characteristics in the Appendix A. We apply target encoding to categorical variables and employ feature Min-Max normalization to account for the varying ranges in the feature distributions. We run experiments using five different seeds and consider the average results over these runs[2].

Baselines. As we employ tabular datasets, in which tree-based methods such as gradient boosting have been shown to achieve better performance [104, 103], we compare our approach to other non-deep learning model-agnostic methods from the literature. The baselines we compare with are:

- PLUGIN-REG, a model-agnostic method introduced by Zaoui et al. [26]. We recall PLUGIN-REG works as follows: given a target coverage c, training samples ($\mathcal{T}_{train}$) and *unlabelled* calibration samples $\mathcal{T}_{cal} \sim P_X$, it estimates the selective regressor by fitting a regression function $\hat{f}$ on $\mathcal{T}_{train}$. Next, the training residuals $e^2_{train} := \left(Y_{train} - \hat{f}(\mathbf{X}_{train})\right)^2$ are computed. It then fits another regression function $\hat{\sigma}^2$ on $(\mathbf{X}_{train}, e^2_{train})$ and compute the predicted conditional variance values $\hat{\sigma}^2(\mathbf{X}_{train})$ of the calibration set $\mathcal{T}_{cal}$. Lastly, it uses these values to calibrate the selective regressor;

- MAPIE, a method that estimates the uncertainty around predictions using the conformal prediction technique from Kim et al. [88], Romano et al. [122], Xu and Xie [123]. It first fits the regressor over the training set. It then uses the CV$^+$ technique [124] with $K = 5$ to produce prediction sets at 95%, and we consider the width of the provided prediction interval as

[2]Since we do not use large neural networks and we limit the sample sizes, this procedure is feasible, conversely to the experimental approaches of other chapters.

a proxy to build selection function g (the larger the interval, the more likely the rejection). It finally calibrates the selective regressor over an unlabeled calibration dataset $\mathcal{T}_{cal}$;

- GoldCase, an Oracle implementation with access to Y labels. This method rejects instances whose residual value is above the c-th percentile. GoldCase provides an upper bound to the performance of all the other baselines.

Hyperparameters. For **Q1** and **Q2**, we consider `XGBoost` [125] as the base algorithm since it achieves state-of-the-art performance in regression task [103, 126].
For **Q3**, we also consider another gradient boosting algorithm, i.e. LightGBM, and `scikit-learn` implementations of random forests (RandomForest), multi-layer perceptrons (MLP) and linear regression (LinearRegression). Hyperparameters are set to defaults. Concerning MLP, we consider also a 4-layer deep architecture as used in [103]. We refer to this baseline as MLP-Large. We did not include deeper architectures due to the computational time required to train neural networks over large data. For Doubt-Var and Doubt-Int, we set the number of bootstraps to the default value $\sqrt{n_{train}}$, with n_{train} representing the training set size. For SCross-POOL and Mapie, we set the number of folds to their default value $K = 5$.

Metrics. Concerning predictive performance, we consider the percentage decrease of Mean Squared Error, i.e.

$$\Delta MSE(h, g, c) := \frac{MSE(h, g, c)}{MSE(h, g, 1)} - 1,$$

where $MSE(h, g, c)$ is the Mean Squared Error computed over accepted instances at desired coverage c and $MSE(h, g, 1)$ is the Mean Squared Error on the full sample. The lower ΔMSE, the more the model can improve its performance once we allow abstention. This allows for a relative comparison of performances across multiple datasets and coverages. Moreover, since the lower the coverage, the lower the expected MSE, we need to account for coverage violations to guarantee a fair comparison across methods. As for Section 7.3.3, we consider a user-defined tolerance ε to account for different levels of coverage violations. We recall that the *ε-coverage violation* is:

$$CovViol_{\varepsilon}(g \mid \mathcal{T}_{test}) = \min(0,\ \hat{\phi}(g \mid \mathcal{T}_{test}) - c + \varepsilon),$$

while the satisfaction of the coverage constraint is defined as:

$$ConSat(\varepsilon) = \mathbb{1}(CovViol_{\varepsilon} = 0)$$

Experimental setup. We randomly split each dataset into a 70/10/20% training/calibration/test split and train the selective regressor on the training set. We then choose different target coverages $c \in \{.99, .95, .90, .85, .80, .75,$

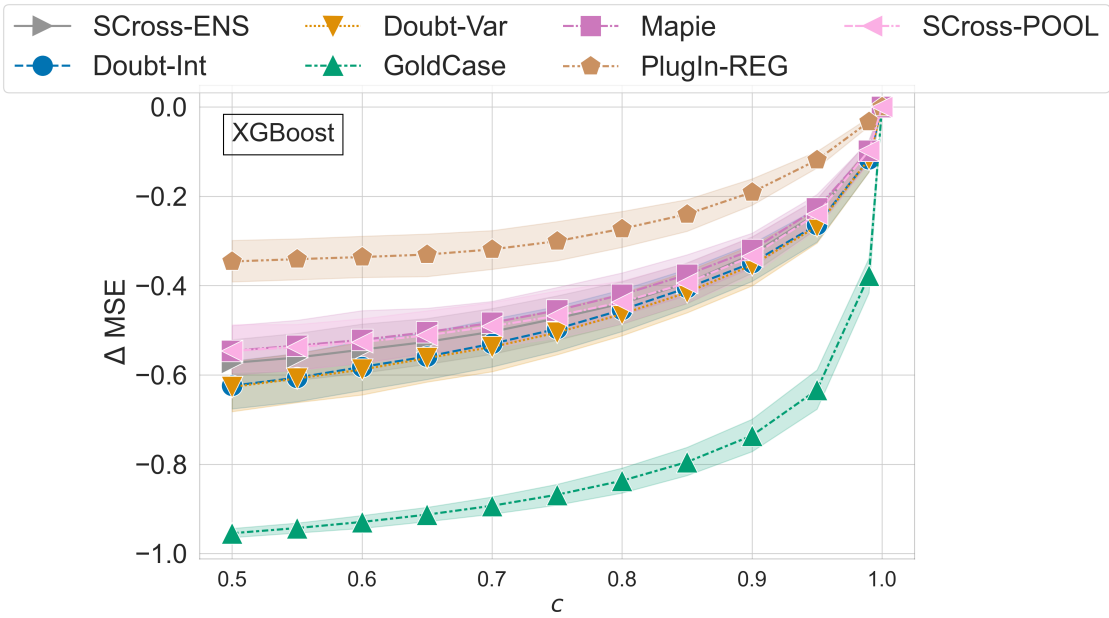


Figure 4.1: ΔMSE as a function of tolerance c when using `XGBoost` as a base regressor.

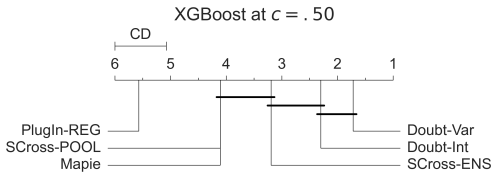
.70, .65, .60..55, .50}, and calibrate a selection function over the calibration set for each of these. Lastly, we compute $ConSat$ and ΔMSE over accepted instances on the test set. For $ConSat$, we consider multiple ε values, namely $\{0, .01, .02, .05, .10\}$.

Hardware. We divide the workload over two machines: (1) a 25 nodes cluster equipped with 2×16-core @ 2.7 GHz (3.3 GHz Turbo) POWER9 Processor and 4 NVIDIA Tesla V100 each, OS RedHatEnterprise Linux release 8.4; (2) a 96 cores machine with Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz and two NVIDIA RTX A6000, OS Ubuntu 20.04.4 The experiments took roughly eight days on the servers, with an estimated CO_2 consumption of $\sim 30.5kg$ according to `codecarbon` [127, 128].

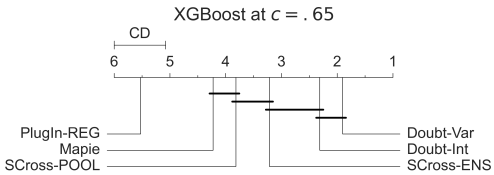
Limitations. Our experimental evaluation does not consider large deep-learning methods, due to their computational burden. As a fact, the extension to deeper architectures would increase the required training time, but it has also a non-negligible environmental cost. Nevertheless, we stress that for tabular data, tree-based methods are still the preferable option [104, 103].

4.4.3 Experimental Results

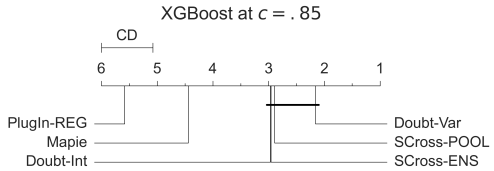
Q1: predictive performance. Figure 4.1 shows the mean ΔMSE values for the different baseline methods and coverages over the analyzed datasets. For lower coverages, Doubt-Int and Doubt-Var reach the best performance, with an average drop in MSE of $\approx 62.4\%$ and $\approx 62.7\%$, respectively, at $c = .5$. PlugIn-REG achieves the worst performance, with an average drop



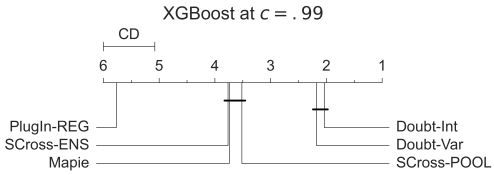
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$



(c) CD plot at $c = .85$



(d) CD plot at $c = .99$

Figure 4.2: Q1: CD plots of ΔMSE for different target coverages. The closer to one, the better the method.

at $c = .5$ of $\approx 34.9\%$, which is almost half than than the drop achieved by Doubt-Var and Doubt-Int.
To evaluate the statistical significance of our results, we use the Nemenyi post-hoc test at a 95% significance level[3] [112]. Figure 4.2 provides the Critical Difference (CD) plots resulting from the tests at $c = .50$, $c = .65$, $c = .85$ and $c = .99$ (the minimum, the first quartile, the third quartile, and the maximum values of target coverages).
At $c = .99$ we observe that Doubt-Int is ranked first, even though it is not statistically different from Doubt-Var. When compared with other methods, the differences become significant, with bootstrap-based approaches outperforming the other methods. Interestingly, also cross-fitting-based methods outperform PlugIn-REG in a statistically significant sense.
At $c = .85$, Doubt-Var becomes the best method, even though there are no statistically significant differences compared with SCross-POOL, SCross-ENS and Doubt-Int. However, we can see that this group of methods has a statistically significant difference w.r.t. Mapie and PlugIn-REG.
At $c = .65$ and $c = .50$, we see similar patterns, as the difference between the bootstrap-based strategies and all the other baselines is statistically significant. On the other hand, the performance of Doubt-Var and Doubt-Int is not statistically distinguishable.
Hence, experimental evidence supports the usage of bootstrap uncertainty estimation as a reliable way to perform selective regression, answering positively **Q1**.

Q2: coverage violations. The constraint on the target coverage c in (2.11) is essential in many scenarios. Nevertheless, most papers do not consider the actual coverage achieved by the baselines. We assess the coverage violations

[3]We apply a Bonferroni correction for accounting for the different base regressors used.

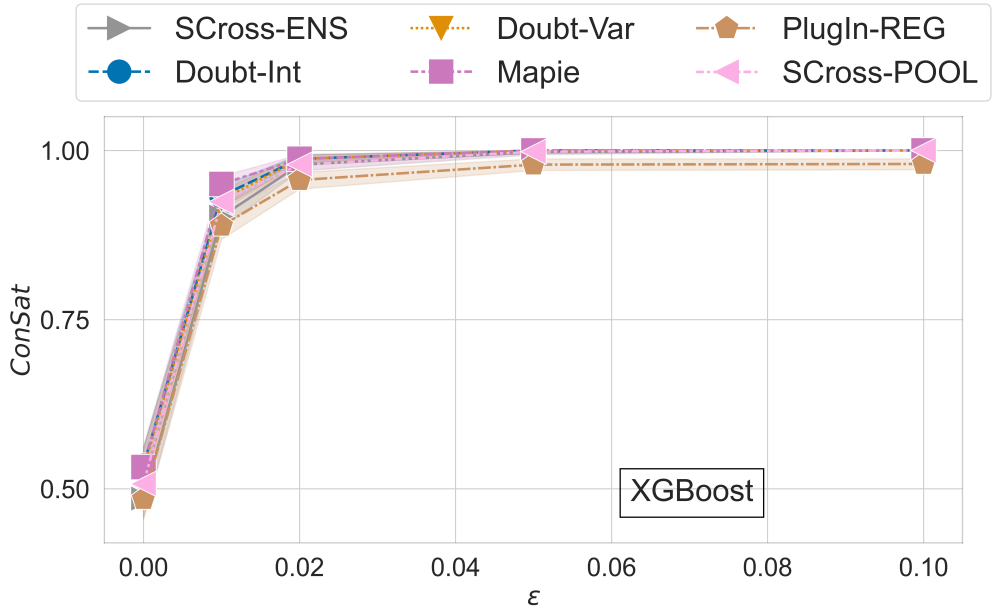
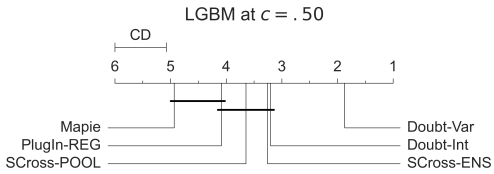


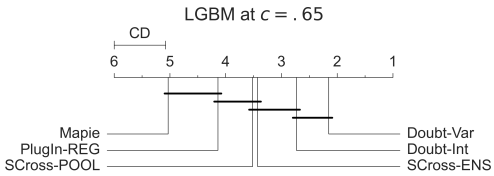
Figure 4.3: *ConSat* as a function of tolerance ε when using `XGBoost` as a base regressor.

concerning the target coverage c by looking at the achieved coverage over the test set.
Figure 4.3 reports the mean and standard deviation of *ConSat* for ε in $\{0, .01, .02, .05, .10\}$. As expected, the percentage of satisfied constraints increases for all baselines when increasing ε, and the gap among the baselines decreases.
More precisely, when we do not allow any violation, the best method is Doubt-Int, with the constraint satisfied in $\approx 53.4\%$ of the cases. The worst method is instead PlugIn-REG, with no violations in $\approx 48.5\%$ of the times. When we allow for small violations (i.e. $\varepsilon \leq .02$) we see large improvements, with the constraint being satisfied in more than 95% of the cases by all the methods. When increasing the tolerance further to $\varepsilon = .05$, Doubt-Int, Doubt-Var and Mapie satisfy the constraint all the times. Finally, when considering $\varepsilon = .10$, only PlugIn-REG does not always satisfy the constraint, with a percentage of $ConSat = 1$ of $\approx 98\%$.
Overall, these findings suggest that the coverage-calibration technique works adequately for all the methods, with only a few rare exceptions of coverage violated when we increase the tolerance ε.

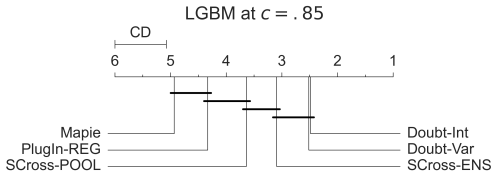
Q3: changing base regressor. We evaluate the ability of our methods to adapt to different base regressors. Figures 4.4-4.8 provide CD plots for the other base regressors we consider in the analysis.
When using LightGBM, bootstrap-based methods are once again the top performers, with Doubt-Var becoming the best method at $c = .50$ with statistically significant effects. Cross-fitting approaches are always ranked better than PlugIn-REG, even though they are sometimes statistically indistinguishable from it. For all the analyzed target coverages c, the worst method



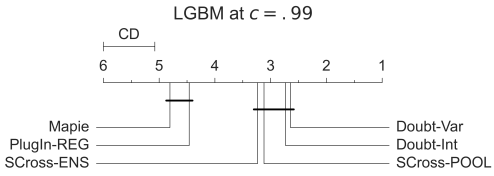
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$



(c) CD plot at $c = .85$



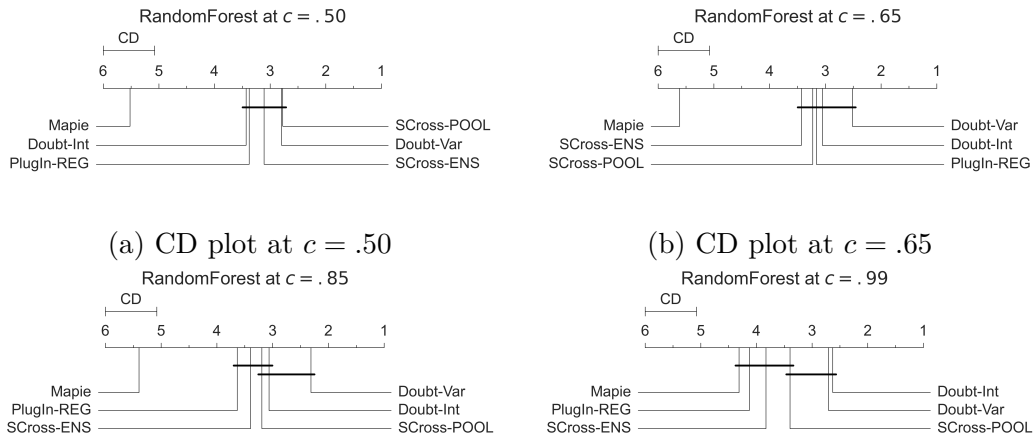
(d) CD plot at $c = .99$

Figure 4.4: Q1: CD plots of ΔMSE for different target coverages, using LightGBM. The closer to one, the better the method.

is Mapie.
When looking at RandomForest, we observe that a group of five methods consistently outperforms Mapie. However, we are not able to distinguish these five methods in a statistically significant sense for $c = .50$ and $c = .65$, while when looking at higher values $c = .85$ and $c = .99$, Doubt-Var outperform also PlugIn-REG and SCross-ENS.
When we look at MLP, we observe that Doubt-Var is once again the best methodology, being the top-ranked method at $c = .85$, $c = .65$ and $c = .50$. In the last two cases, the difference with the other methods is statistically significant.
When considering MLP-Large, we see similar results, with Mapie always ranked last in the comparison. Moreover, Doubt-Var ranks first for all coverages below $c = .85$, with statistically significant differences when compared to the state-of-the-art method PlugIn-REG.
Finally, we check results for LinearRegression, which should be less prone to overfitting concerns. We observe that in this setting, bootstrap-based approaches are not the top performers, while the best methods are cross-fitting based, i.e. SCross-ENS and SCross-POOL, and PlugIn-REG.
Overall, these results suggest that Doubt-Var is always as good or better than other approaches, when employing machine learning algorithms that are prone to overfitting. These include a large set of methods, such as boosting and neural networks. The superiority of bootstrap-based methods gradually vanishes when considering simpler algorithms.

4.5 Conclusions

This chapter addressed the understudied problem of selective regression, introducing four novel methods: two were based on cross-fitting and two on

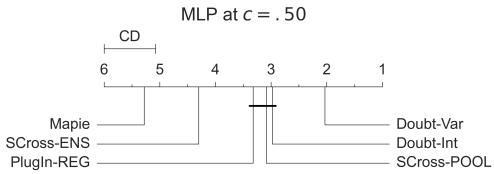


(a) CD plot at $c = .50$ (b) CD plot at $c = .65$

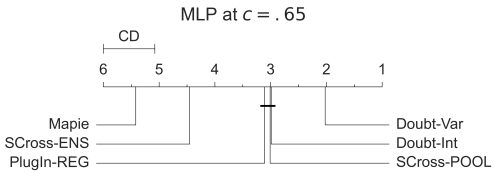
(c) CD plot at $c = .85$ (d) CD plot at $c = .99$

Figure 4.5: Q1: CD plots of ΔMSE for different target coverages, using RandomForest. The closer to one, the better the method.

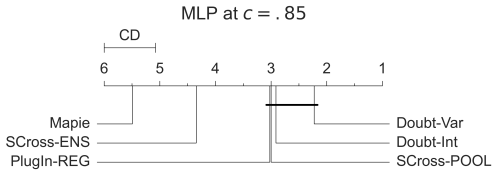
bootstrap sampling. These methods are model-agnostic, extending any regressor to abstain from making predictions when facing uncertainty. To validate our methods, we run an extensive empirical evaluation over 93 datasets and different regression algorithms. Results show that the proposed approaches are as good or better than existing selective regressors. In particular, the improvement is more relevant when considering regression algorithms that are more prone to overfitting concerns.



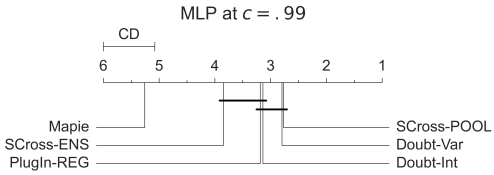
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$

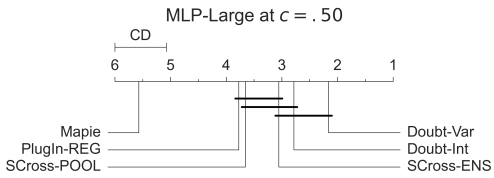


(c) CD plot at $c = .85$

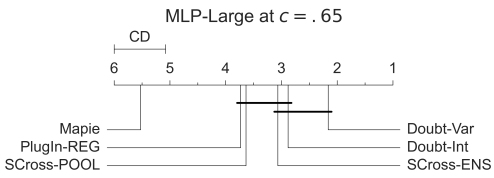


(d) CD plot at $c = .99$

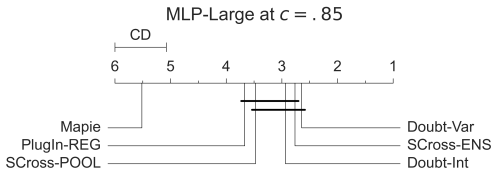
Figure 4.6: Q1: CD plots of ΔMSE for different target coverages, using MLP. The closer to one, the better the method.



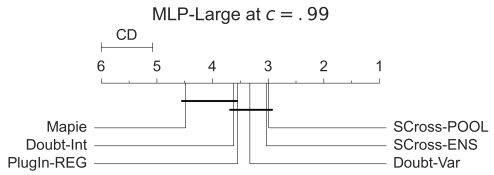
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$

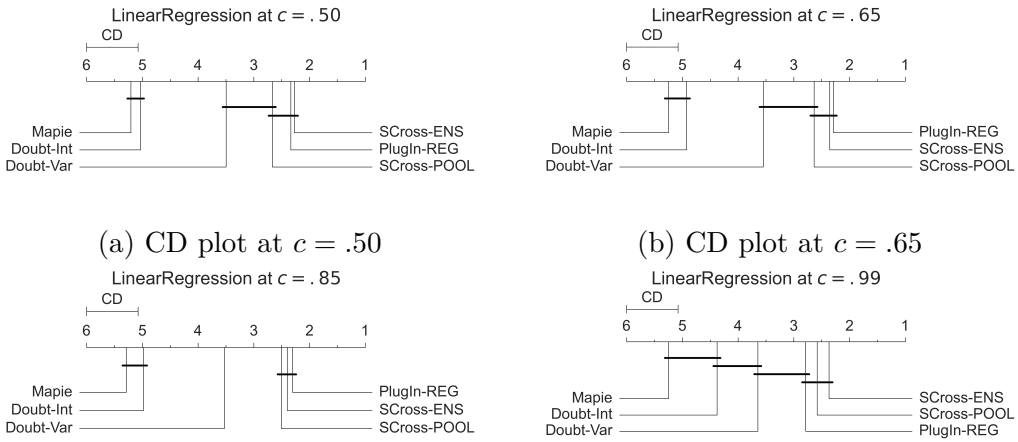


(c) CD plot at $c = .85$



(d) CD plot at $c = .99$

Figure 4.7: Q1: CD plots of ΔMSE for different target coverages, using MLP-Large. The closer to one, the better the method.



(a) CD plot at $c = .50$

(b) CD plot at $c = .65$

(c) CD plot at $c = .85$

(d) CD plot at $c = .99$

Figure 4.8: Q1: CD plots of ΔMSE for different target coverages, using LinearRegression. The closer to one, the better the method.

Chapter 5

Selective Classification for AUC

The goal of this chapter is to address *Req. 2*, by presenting a novel approach to perform selective classification when one is interested in trading off coverage for improved discriminative power. We provide a theoretical analysis to achieve such a goal. Based on that, we devise two algorithms specific to the AUC quality metric. The chapter is based on the work [129], presented at AISTATS 2023.

5.1 Introduction

In many real-world scenarios, such as credit scoring [130], the quality of a classifier is concerned with the discriminative power of the ranking induced by its scores [131]. This ability can be measured by the Area Under the ROC Curve (AUC) [132] or, equivalently, by the Gini coefficient [133]. Cortes and Mohri [21] point out how algorithms designed to minimize, e.g., the error rate may not lead to the best possible AUC values. Since all these scenarios where the discriminative power of the ranking are generally considered high-risk settings, one could be interested in building a selective classifier that improves the ability to rank instances properly once abstention is allowed. However, the literature on selective classification has mainly focused on distributive losses, such as the error rate, not considering these kinds of concerns.

In this chapter, we introduce the problem of improving the ranking ability of a classifier once it can abstain and call it *AUC-based selective classification*. This consists of inducing a selective classifier that optimizes the AUC over the sub-population of accepted predictions while guaranteeing a minimum probability mass (i.e. the *coverage*) of such sub-population.

We then focus on an instance of the AUC-selective classification problem, where we assume that: *(i)* the classification algorithm is given, including its hyper-parameters, and *(ii)* the selection function abstains on a range of the scores (*score-based selective functions*). Assumption *(i)* allows lifting an existing classifier to a selective classifier. In this regard, our approach is model-agnostic, as we do not make assumptions about the classification algorithm. Assumption *(ii)* is widespread in selective classification, supported by theoretical results, e.g. [13]. Such an assumption allows for a theoretical analysis of

the AUC-based selective classification problem. We devise two model-agnostic algorithms, i.e. PluginAUC and AUCross, for estimating the bounds of score-based selective functions while maximizing the AUC for target coverage. Both methods build their selection strategy based on theoretical analysis of (sufficient) conditions for abstaining that increase AUC. Moreover, AUCross exploits a cross-fitting strategy over the training set similar to SCross strategy presented in Chapter 3.2.2. Experiments show that both approaches outperform other existing selective classification methods targeted at optimizing accuracy. Moreover, we show the achieved performance is very close to an oracle score-based selective function, which has access to the true class of instances.
The rest of the chapter is organized as follows. Section 5.2 reports some additional notation and introduces the AUC-based selective classification problem. A theoretical analysis is presented in Section 5.3, while we introduce AUCross algorithm in Section 5.4. Section 5.5.3 reports experimental results. Finally, we summarize the contribution of the Chapter in Section 5.6.

5.2 Background

We consider binary classes, i.e. $\mathcal{Y} = \{0, 1\}$. We call instances whose $y = 0$ the negatives, while we refer to those with $y = 1$ as the positives. We focus on (binary) probabilistic classifiers $s : \mathcal{X} \to [0, 1]$, where $s(\mathbf{x}) \in [0, 1]$ is an estimate of the probability that $\mathbf{x}$ is positive (i.e. $\eta(\mathbf{x})$)[1]. Let $\mathcal{D}_1 = P(\mathbf{X}|Y = 1)$ and $\mathcal{D}_0 = P(\mathbf{X}|Y = 0)$ be the conditional distributions of positives and negatives, respectively. In formal terms, the AUC can be defined as follows:

$$AUC(s) = \mathbb{E}_{\mathbf{X}_0 \sim \mathcal{D}_0, \mathbf{X}_1 \sim \mathcal{D}_1}[\mathbb{1}\{s(\mathbf{X}_1) > s(\mathbf{X}_0)\}]. \tag{5.1}$$

We can extend AUC to **selective AUC** by considering the probability that a randomly drawn positive receives a higher score than a randomly drawn negative, conditioned to the fact that both are selected according to g, i.e.

$$AUC(s, g) = \mathbb{E}_{\mathbf{X}_0 \sim \mathcal{D}_0, \mathbf{X}_1 \sim \mathcal{D}_1}[\mathbb{1}\{s(\mathbf{X}_1) > s(\mathbf{X}_0)\}| \tag{5.2}$$

$$g(\mathbf{X}_0) = 1, g(\mathbf{X}_1) = 1] \tag{5.3}$$

The *AUC-selective classification problem* can then be stated as follows:

Problem 4 (AUC-based bounded-abstention model)**.** *Given a target coverage c, an optimal selective (probabilistic) classifier (s, g) parameterized by θ^*, ψ^* can be defined as:*

$$\arg\max_{\theta,\psi} AUC(s_\theta, g_\psi) \quad s.t. \quad \phi(g_\psi) \geq c \tag{5.4}$$

In this setting, a good confidence function should order instances based on ascending contribution to the AUC while allowing for controlling the target

[1]Note that in the binary setting it suffices to estimate the probability of being a positive instance, since $P(Y = 0|\mathbf{X} = \mathbf{x}) = 1 - \eta(\mathbf{x}) \approx 1 - s(\mathbf{x})$.

coverage. Empirical AUC given $\mathcal{T}_n$ can be defined by restricting to the set $\mathcal{T}_m^g = \{(\mathbf{x}_i, y_i) \in \mathcal{T}_n \mid g(\mathbf{x}_i) = 1\}$ of accepted instances directly from the definition above by resorting to the Mann-Whitney U-statistic:

$$\widehat{AUC}(s, g|\mathcal{T}_n) = \widehat{AUC}(s|\mathcal{T}_m^g) = \frac{1}{m^+}\frac{1}{m^-} \sum_{(\mathbf{x}_i,1)\in\mathcal{T}_m^g} \sum_{(\mathbf{x}_i,0)\in\mathcal{T}_m^g} \mathbb{1}\left(s(\mathbf{x}_i) > s(\mathbf{x}_j)\right), \tag{5.5}$$

where m^+ is the number of positives in $\mathcal{T}_m^g$, and m^- is the number of negatives in S_m^g. An alternative calculation of $\widehat{AUC}$ is the Area Under the Receiver Operating Characteristic (ROC) Curve [132].
We further assume that the parameters of s are fixed, i.e., we only need to optimize ψ, i.e. the parameters of the selection function g in (5.4).

5.3 Methodology

Let us denote by S a binary probabilistic classifier induction algorithm whose hyper-parameters are fixed. The algorithm induces the classifier h[2] starting from a training set. We propose to lift h to a selective classifier by calculating a selection function g from the following hypothesis space (*score-bounded selection functions*):

$$g(\mathbf{x}) = \begin{cases} 0 & \text{if } \tau_l \leq s(\mathbf{x}) \leq \tau_u \\ 1 & \text{otherwise} \end{cases} \tag{5.6}$$

Intuitively, the probabilistic selective classifier abstains if the score is within the bounds τ_l and τ_u.
To identify the bounds of g, we consider the following scenario, where we have a sample $\mathcal{T}_n = \{\mathbf{x}_i, y_i\}_{i=1}^n$ and the (binary probabilistic) classifier s. Our objective is to estimate a score-bounded selection function g such that: *(i)* $\hat{\phi}(g|\mathcal{T}_n) \geq c$; and *(ii)*: $\widehat{AUC}(s, g|\mathcal{T}_n)$ is as large as possible. Let us define $n = n^+ + n^-$, with n^+ (resp., n^-) the number of positive (resp., negative) instances in $\mathcal{T}_n$. Let $p^+ = n^+/n$ be the fraction of positives and let $p^- = 1 - p^+$ be the fraction of negatives. Without loss of generality, we can assume that $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is ordered by non-ascending scores, i.e., $s(\mathbf{x}_i) \geq s(\mathbf{x}_{i+1})$ for $i \in [1, n-1]$. The rank of an instance is $r(\mathbf{x}_i) = i$, and its relative rank is $r(\mathbf{x}_i)/n$. We write $t(\mathbf{x}_i)$ to indicate the number of positive instances occurring in ranks from 1 to i. The true positive rate (TPR) at $\mathbf{x}_i$ is the ratio $tpr(\mathbf{x}_i) = t(\mathbf{x}_i)/n^+$. We depart from reasoning directly on the definition of the AUC. Instead, we consider the linearly related definition of the Gini coefficient [130]:

$$G = 2 \cdot AUC - 1 \tag{5.7}$$

for which the maximization problem is equivalent. The Gini coefficient (also known as the Accuracy Ratio) can be defined from the Cumulative Accuracy

[2]Note that we can always lift the probabilistic classifier s to a classifier by considering $h(\mathbf{x}) = \mathbb{1}\{s(\mathbf{x}) > 1/2\}$

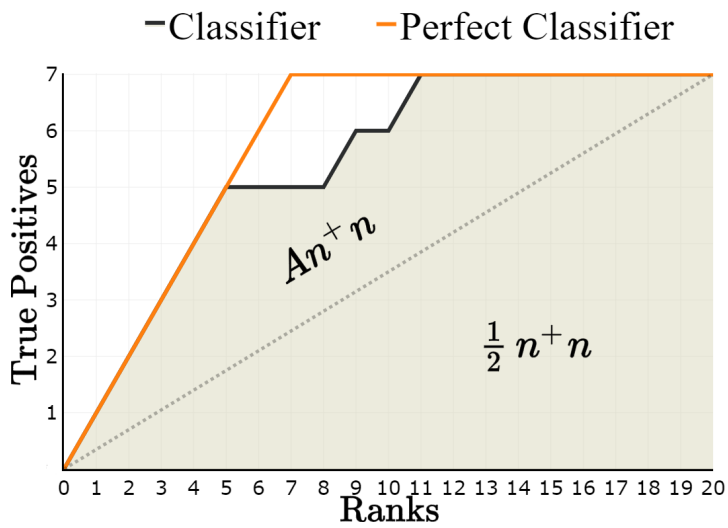


Figure 5.1: Example of a CAP plot with $n = 20$ and $n^+ = 7$.

Profile (CAP), which maps relative rank to TPR. Hence, the Gini coefficient can be written as $G = A/(A + B)$ where A is the area of the CAP between the diagonal and the line of the classifier s, and B is the area between the line of h and the line of a perfect classifier $s^\star$. A perfect classifier assigns a score of 1 to positives and a score of 0 to negatives; thus, it ranks all positives first, then all negatives afterward. The diagonal line represents the performance of a random classifier. In summary, the Gini coefficient measures the discriminative power of a probabilistic classifier as a fraction of the difference in power between a perfect classifier and a random classifier. Using the CAP plot and the Gini coefficient will be crucial in proving our results. Figure 5.1 shows a sample CAP plot where the axes are ranks and true positives instead of their relative counterparts. The Gini coefficient can equivalently be computed as ratio $An^+n/(An^+n + Bn^+n)$. We observe that $(A + B)n^+n$ is $n^+n^+/2$ (the area of $s^\star$ for positives) plus n^-n^+ (the area of $s^\star$ for negatives) minus $n^+n/2$ (the area under the diagonal). Hence $A + B = n^-/(2n) = p^-/2$.
Let us define $r'(\mathbf{x}_i) = n - r(\mathbf{x}_i) + 1$, i.e. $r'(\mathbf{x}_i)$ is the rank over the non-descending scores. We next show a (sufficient) condition to improve $\widehat{AUC}(s, g|\mathcal{T}_n)$ by abstaining on positive instances.

Proposition 1. *For any (number of) positive instance* $\mathbf{x}_i$ *such that* $g(\mathbf{x}_i) = 1$ *and:*

$$\frac{r'(\mathbf{x}_i)}{n} \leq \widehat{AUC}(s, g|\mathcal{T}_n) \cdot p^- \tag{5.8}$$

we have: $\widehat{AUC}(s, g|\mathcal{T}_n) \leq \widehat{AUC}(s, g'|\mathcal{T}_n)$*, where* $g'(\mathbf{x}_i) = 0$ *and* $g'(\mathbf{x}) = g(\mathbf{x})$ *otherwise. The inequality is strict if at least one such* $\mathbf{x}_i$ *exists in* $\mathcal{T}_n$*.*

Proof. By (5.7), we can equivalently show the result for the Gini coefficient. Let $G = A/(A+B)$, and let $\bar{G} = \bar{A}/(\bar{A}+\bar{B})$ be the Gini coefficient after removing (a.k.a., abstaining on) one positive instance $\mathbf{x}_i$. We have $A + B = n^-/(2n)$

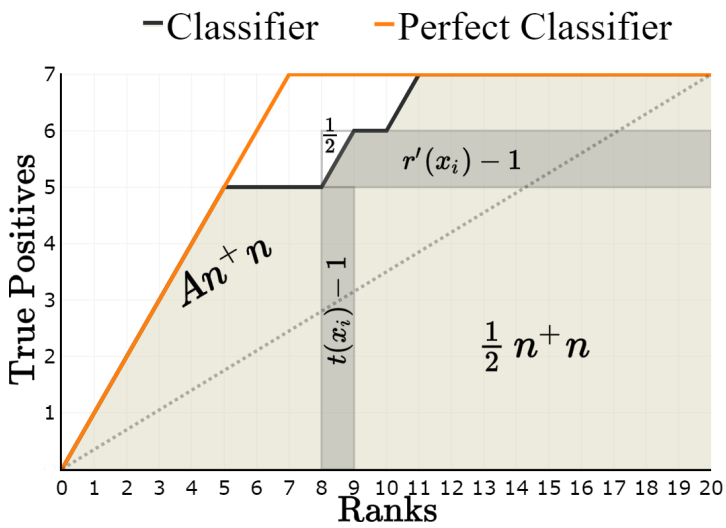


Figure 5.2: CAP plot if we were to remove a positive instance.

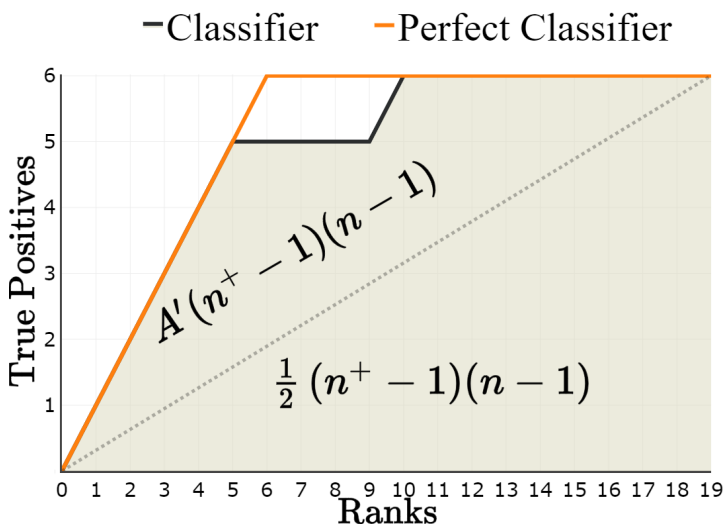


Figure 5.3: CAP plot after removing a positive instance.

and $\bar{A} + \bar{B} = n^-/(2(n-1))$. Then $\bar{G} > G$ iff:

$$\bar{A}(n-1) > An \tag{5.9}$$

As $r'(\mathbf{x}_i) = n - r(\mathbf{x}_i) + 1$, we have that $\bar{A}$ is related to A as follows:

$$\bar{A} + \tfrac{1}{2} = (A + \tfrac{1}{2} - \left[\tfrac{(r'(\mathbf{x}_i)-1)+(t(\mathbf{x}_i)-1)+1/2}{n^+n}\right])\tfrac{n^+n}{(n^+-1)(n-1)} \tag{5.10}$$

Using such equality, the condition in (5.9) can be simplified to (see Appendix B.1 for full derivation):

$$\frac{r'(\mathbf{x}_i)}{n} \leq A + \frac{p^-}{2} + \frac{(n^+ - t(\mathbf{x}_i))}{n}$$

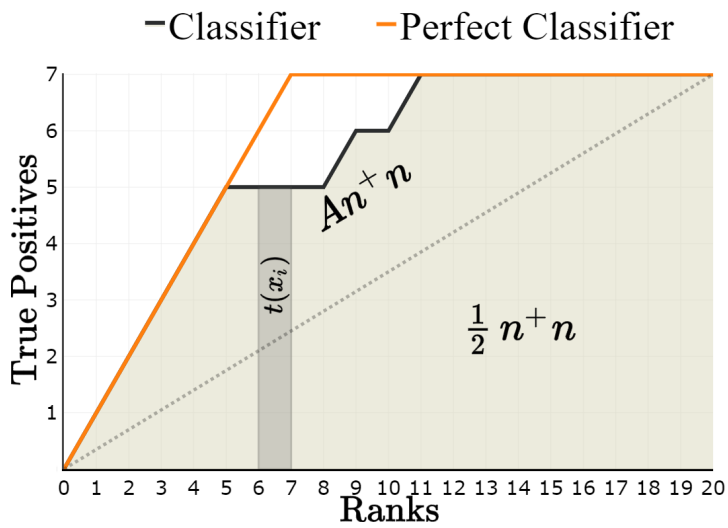


Figure 5.4: CAP plot if we were to remove a negative instance.

Since $n^+ \geq t(\mathbf{x}_i)$, the inequality above is satisfied if:

$$\frac{r'(\mathbf{x}_i)}{n} \leq A + \frac{p^-}{2} \tag{5.11}$$

By (5.7) and $G = A/(A+B) = 2A/p^-$, we have $A + {}^{p^-}/_2 = \widehat{AUC}(s, g|\mathcal{T}_n) \cdot p^-$, and thus the conclusion (5.8) holds after removing *one* positive instance. Let us then consider the case when we remove $(1-\alpha)n^+$ positive instances, with $\alpha \in [0, 1[$. Inequality (5.11) becomes:

$$\frac{r'(\mathbf{x}_i)}{n} \leq A_\alpha \frac{(n^- + \alpha n^+)}{n} + \frac{p^-}{2} \tag{5.12}$$

where A_α is the numerator of the Gini fraction after removing $(1-\alpha)n^+$ positives. By repeatedly removing one positive instance at a time, (5.9) implies that $A_\alpha(n^- + \alpha n^+) > An$. Therefore, if (5.8) holds, then (5.12) holds for any α. Therefore, by removing as many as possible positive instances $\mathbf{x}_i$ such that (5.8) holds, we increase the Gini coefficient, and, a fortiori, the $\widehat{AUC}(s, g'|\mathcal{T}_n)$. □

A key step in the proof is equation (5.10). Let us give some intuitions. Abstaining on a positive instance means removing some areas in the CAP plot, as shown in grey in Figure 5.2. The vertical grey band consists of $t(\mathbf{x}_i) - 1$ cells in the nn^+ grid. The horizontal grey band consists of $r'(\mathbf{x}_i) - 1$ cells. In addition, an area of 1/2 is removed from the cell with the increase of the classifier line. Finally, after removing those areas, the grid is rescaled from nn^+ to $(n^+ - 1)(n - 1)$, which provides the rescaling factor in (5.10). Figure 5.3 shows the CAP plot after removing a positive instance.
A similar result holds when removing negative examples. The condition is based on the TPR of the removed instance.

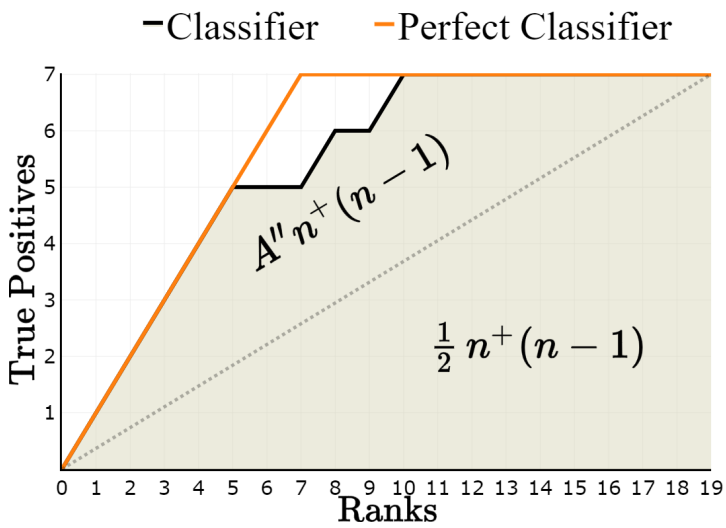


Figure 5.5: CAP plot after removing a negative instance.

Proposition 2. *For any (number of) negative instance* $\mathbf{x}_i$ *such that* $g(\mathbf{x}_i) = 1$ *and:*

$$\frac{t(\mathbf{x}_i)}{n^+} \leq \widehat{AUC}(s, g|\mathcal{T}_n) - \frac{1}{n^+} \tag{5.13}$$

we have: $\widehat{AUC}(s, g|\mathcal{T}_n) \leq \widehat{AUC}(s, g'|\mathcal{T}_n)$*, where* $g'(\mathbf{x}_i) = 0$ *and* $g'(\mathbf{x}) = g(\mathbf{x})$ *otherwise. The inequality is strict if at least one such* $\mathbf{x}_i$ *exists in* $\mathcal{T}_n$*.*

Proof. We can equivalently show the result for the Gini coefficient. Let $G = A/(A+B)$, and let $\bar{G} = \bar{A}/(\bar{A}+\bar{B})$ be the Gini coefficient after removing (a.k.a., abstaining on) one negative instance $\mathbf{x}_i$. We have $\bar{A}+\bar{B} = (n^- - 1)/(2(n-1))$. Then $\bar{G} > G$ iff

$$\bar{A}(n-1)n^- > An(n^- - 1) \tag{5.14}$$

As in Proposition 1, we can link the area $\bar{A}$ after the removal of the negative instance $\mathbf{x}_i$ to the original area A as follows:

$$\bar{A} + \frac{1}{2} = (A + \frac{1}{2} - \frac{t(\mathbf{x}_i)}{n^+ n})\frac{n}{n-1}$$

As for the positive case, this can be intuitively understood by looking at Figure 5.4, where the grey area highlights the loss from the original CAP plot after we remove the negative instance. Then we rescale to account for the new number

of instances, as in Figure 5.5. Then:

$$\begin{aligned}
\bar{G} > G \quad \text{iff} \quad & ((A + \frac{1}{2} - \frac{t(\mathbf{x}_i)}{n^+ n})\frac{n}{n-1} - \frac{1}{2})(n-1)n^- > An(n^- - 1) \\
\text{iff} \quad & (A + \frac{1}{2} - \frac{t(\mathbf{x}_i)}{n^+ n})nn^- - \frac{(n-1)n^-}{2} > An(n^- - 1) \\
\text{iff} \quad & Ann^- + \frac{nn^-}{2} - \frac{t(\mathbf{x}_i)}{n^+}n^- - \frac{(n-1)n^-}{2} > Ann^- - An \\
\text{iff} \quad & An + \frac{nn^-}{2} - \frac{(n-1)n^-}{2} > \frac{t(\mathbf{x}_i)}{n^+}n^- \\
\text{iff} \quad & A\frac{n}{n^-} + \frac{1}{2} > \frac{t(\mathbf{x}_i)}{n^+} \\
\text{iff} \quad & A\frac{n}{n^-} + \frac{1}{2} - \frac{1}{n^+} \geq \frac{t(\mathbf{x}_i)}{n^+}
\end{aligned}$$

Since $\widehat{AUC}(h, g'|\mathcal{T}_n) = (G+1)/2 = (2A/p^- + 1)/2 = A^n/n^- + 1/2$, and

$$\frac{t(\mathbf{x}_i)}{n^+} \leq \widehat{AUC}(h, g|\mathcal{T}_n) - \frac{1}{n^+} \tag{5.15}$$

is assumed to hold, we have the $\bar{G} > G$ (a.k.a., the conclusion of Proposition 2) after removing *one* negative instance. Moreover, if (5.15) holds for a second negative instance $\mathbf{x}_i$, then:

$$\frac{t(\mathbf{x}_i)}{n^+} \leq \widehat{AUC}(h, g|\mathcal{T}_n) - \frac{1}{n^+} \leq \widehat{AUC}(h, g'|\mathcal{T}_n) - \frac{1}{n^+}$$

where g' abstains on the first negative instance. In fact, $\bar{G} > G$ implies that $\widehat{AUC}(h, g|\mathcal{T}_n) < \widehat{AUC}(h, g'|\mathcal{T}_n)$. Therefore, we can iterate the conclusion that the Gini coefficient (or, equivalently, empirical AUC) increases by abstaining on any number of negative instances that satisfy the assumption of Proposition 2. □

In general, the selection functions g' in Prop. 1 and Prop. 2 are not score-bounded. Let us assume that $g(\mathbf{x}) = 1$ for all $\mathbf{x}$, hence $\widehat{AUC}(s, g|\mathcal{T}_n) = \widehat{AUC}(s|\mathcal{T}_n)$. We can easily lift condition (5.8) to scores as follows:

$$s(\mathbf{x}_i) \leq \hat{q}_n(\widehat{AUC}(s|\mathcal{T}_n) \cdot p^-) = \tau_u$$

where $\hat{q}_n$ is the empirical quantile function over the scores $\{s(\mathbf{x}_i) \mid (\mathbf{x}_i, y_i) \in \mathcal{T}_n\}$. As a consequence, g' is score bounded as it boils down to $g'(\mathbf{x}) = 0$ iff $0 \leq s(\mathbf{x}) \leq \tau_u$. Regarding (5.13), since TPR is anti-monotonic with the scores, we can restate the condition as follows:

$$\tau_l = \hat{q}_{n^+}(\widehat{AUC}(s|\mathcal{T}_n) - 1/n^+) \leq s(\mathbf{x}_i)$$

where $\hat{q}_{n^+}$ is the empirical quantile function over the scores of the positives $\{s(\mathbf{x}_i) \mid (\mathbf{x}_i, 1) \in \mathcal{T}_n\}$. In this case, g' is score bounded as $g'(\mathbf{x}) = 0$ iff $\tau_l \leq s(\mathbf{x}) \leq 1$.

In summary, if $s(\mathbf{x}_i)$ is within the bounds τ_l and τ_u, then, irrespective whether $\mathbf{x}_i$ is positive or negative, by abstaining on $\mathbf{x}_i$ we obtain an increase in $\widehat{AUC}$. In summary, we have the following result.

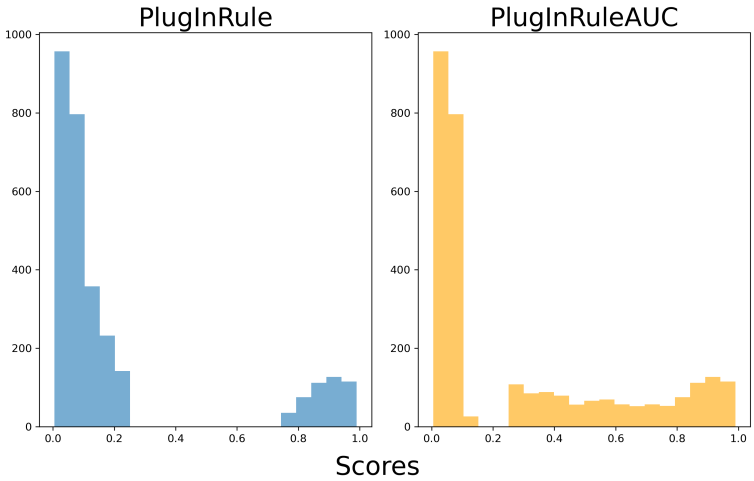


Figure 5.6: An example of the rejection area using PLUGINAUC vs the standard PLUGIN approach.

Proposition 3. *Called* $\tau_l = \hat{q}_{n^+}(\widehat{AUC}(s|\mathcal{T}_n) - 1/n^+)$ *and* $\tau_u = \hat{q}_n(\widehat{AUC}(s|\mathcal{T}_n) \cdot p^-)$*, we define* $g(x)$ *as:*

$$g(\mathbf{x}) = \begin{cases} 0 & \text{if } \tau_l \leq s(\mathbf{x}) \leq \tau_u \\ 1 & \text{otherwise.} \end{cases}$$

Then we have: $\widehat{AUC}(s|\mathcal{T}_n) \leq \widehat{AUC}(s, g|\mathcal{T}_n)$*. The inequality is strict if at least one* $\mathbf{x}_i$ *exists in* $\mathcal{T}_n$ *such that* $g(\mathbf{x}_i) = 0$*.*

5.4 Algorithms for AUC improvement

Given the theoretical results of the previous section, we can devise ways to improve AUC when abstaining. We start by considering Alg. 7: given scores and the true class labels, the algorithm computes the bounds according to the formulas in Prop. 3. We emphasize once more that the thresholds τ_l and τ_u are specifically designed for the AUC selective classification problem (5.4), and differ from the standard PLUGIN approach presented in (2.14) as the rejection area is data-dependent and not necessarily centered around $1/2$ (as shown in Figure 5.6).

5.4.1 The PluginAUC algorithm

The first way we present to build an AUC-based selective classifier relies on the plug-in principle by [134]. The process, referred to as PLUGINAUC, is described in Algorithm 8.
Given a training set $\mathcal{T}_{train}$, a labeled validation set $\mathcal{T}_{val}$, a base classifier S and a target coverage c, PLUGINAUC fits the classifier over the training set (line 1). Then, it computes the bounds over the validation set, using the fitted classifier's predicted scores (lines 2-3). Once again, we adopt a quantile

Algorithm 7: EstimateThetasAUC()

Input : $(\mathbf{s}, \mathbf{y})$ - scores and true class labels
Output: (τ_l, τ_u) - bounds for AUC-based selective classification

$n, n^+, p^- \leftarrow |\mathbf{y}|, |\mathbf{y} == 1|, 1 - n^+/n$
$\widehat{AUC} \leftarrow AUC.ROC(\mathbf{s}, \mathbf{y})$ // compute empirical AUC
$u \leftarrow \lfloor \widehat{AUC} \cdot p^- \cdot n \rfloor$ // τ_u position
$\mathbf{tpr} \leftarrow 1 - cum.sum(\mathbf{y}[order(\mathbf{s})])/n^+$ // compute true positive rates
$l \leftarrow search.sorted(\mathbf{tpr}, \widehat{AUC} - 1/n^+)$ // τ_l position
$\mathbf{ss} \leftarrow sort(\mathbf{s})$ // sort scores ascending
$\tau_l, \tau_u \leftarrow \mathbf{ss}[l], \mathbf{ss}[u]$ // bounds of g
return (τ_l, τ_u)

Algorithm 8: PluginAUC.fit()

Input : $\mathcal{T}_{train} = \{\mathbf{X}_{train}, \mathbf{y}_{train}\}$ - training set,
$\mathcal{T}_{val} = \{\mathbf{X}_{val}, \mathbf{y}_{val}\}$ - labelled validation set,
S - binary probabilistic classifier,
c - target coverage
Output: (h, g) - selective classifier

1 $s \leftarrow S.fit(\mathcal{T}_{train})$ // fit the classifier
2 $\mathbf{s}_{val} \leftarrow s.score(\mathbf{X}_{val})$ // score validation data
3 $\tau_l, \tau_u \leftarrow EstimateThetasAUC(\mathbf{s}_{val}, \mathbf{y}_{val})$ // bounds for $\mathbf{s}_{val}$
4 $J \leftarrow KFold((\mathbf{s}_{val}, \mathbf{y}_{val}), 2)$ // split $\mathbf{s}_{val}$ and y_{val} in two sub-samples
5 **for** $(\mathbf{s}_j, \mathbf{y}_j) \in J$ **do** // for each sub-sample
6 $\quad \tau_{l_j}, \tau_{u_j} \leftarrow EstimateThetasAUC(\mathbf{s}_j, \mathbf{y}_j)$ // compute τ_{l_j}, τ_{u_j} on subsamples
7 $\tau_{u^*} \leftarrow \frac{1}{\sqrt{2}}\tau_u + (1 - \frac{1}{\sqrt{2}}) \sum_{i=1}^{2} \frac{\tau_{u_i}}{2}$ // final estimate of τ_u
8 $\tau_{l^*} \leftarrow \frac{1}{\sqrt{2}}\tau_l + (1 - \frac{1}{\sqrt{2}}) \sum_{i=1}^{2} \frac{\tau_{l_i}}{2}$ // final estimate of τ_l
9 $mid \leftarrow \lfloor n \cdot (\tau_{u^*} + \tau_{l^*})/2 \rfloor$ // compute mid point position
10 $\delta \leftarrow n \cdot (1 - c)/2$ // half-width of rejection area
11 $u' \leftarrow \min\{\lfloor mid + \delta \rfloor, n\}$ // upper bound position
12 $l' \leftarrow \max\{1, \lfloor mid - \delta \rfloor\}$ // lower bound position
13 $\mathbf{ss} \leftarrow sort(\mathbf{s})$ // sort scores ascending
14 $\tau_{l'}, \tau_{u'} \leftarrow \mathbf{ss}[l'], \mathbf{ss}[u']$ // bounds of g
15 $g \leftarrow$ **lambda** $\mathbf{x} : 1 - \mathbb{1}(\tau_{l'} \leq s.score(\mathbf{x}) \leq \tau_{u'})$ // selection function
16 $h \leftarrow$ **lambda** $\mathbf{x} : \mathbb{1}\{s.score(\mathbf{x}) \geq 1/2\}$ // final classifier
17 **return** (h, g)

estimator based on subsamples (lines 5-8), which improves over the variance of a full-sample quantile estimator [106] (see Theorem 1). We notice that the considered quantile estimator, and in particular τ_l and τ_u in Prop. 3, are intended to maximize the AUC independently of the desired coverage c. To address the minimum coverage constraint c, we centre the rejection area at the midpoint between the instances with score τ_{l^*} and τ_{u^*} (line 9). Such a mid point mid is the median of the distribution of scores from τ_{l^*} up to τ_{u^*}. We consider a rejection area $[l', u']$ centered in mid and with width $n \cdot (1 - c)$, also

checking not to exceed the range $[1, n]$ (lines 10-14). Finally, these estimates are used to determine the final selection function g (line 15).

5.4.2 The AUCross algorithm

The main drawback of PluginAUC is that it requires a labeled validation set. This might require splitting the training set, losing some efficiency in the estimation [105]. Hence, we prevent setting apart a validation set from the training set (as done in state-of-the-art methods) by using a cross-fitting strategy, as in Chapter 3. This allows us to be able to fit the classifier h on the whole available training set. We call this induction algorithm AUCross. We report the pseudo-code of AUCross in Alg. 9.

Algorithm 9: AUCross.fit()

Input : $\mathcal{T} = \{\mathbf{X}, \mathbf{y}\}$ - training set,
S - binary probabilistic classifier,
c - target coverage,
K - number of folds

Output: (h, g) - selective classifier

1 $Lbs, Ubs \leftarrow [], []$ // empty lists of bounds
2 $\mathcal{T}_1, \ldots, \mathcal{T}_K \leftarrow \text{StratifiedKFold}(\mathcal{T}, K)$ // partitioning
3 **for** $k \in 1, \ldots, K$ **do** // for each fold
4 $\quad \mathcal{T}_{-k} \leftarrow \mathcal{T} \setminus \mathcal{T}_k$ // training data
5 $\quad s_k \leftarrow S.fit(\mathcal{T}_{-k})$ // train k^{th} classifier
6 $\quad \mathbf{s}_k \leftarrow s_k.score(\mathbf{X}_k)$ // score test data
7 $\mathbf{s} \leftarrow \cup_{k=1}^{K} \mathbf{s}_k$ // store all the scores
8 $n \leftarrow |\mathbf{y}|$ // number of instances
9 $\tau_l, \tau_u \leftarrow EstimateThetasAUC(\mathbf{s}, \mathbf{y})$
// bounds for scores over whole training set
10 $J \leftarrow KFold((\mathbf{s}_{val}, \mathbf{y}_{val}), 2)$
// split $\mathbf{s}_{val}$ and y_{val} in two sub-samples
11 **for** $(\mathbf{s}_j, \mathbf{y}_j) \in J$ **do** // for each sub-sample
12 $\quad \tau_{l_j}, \tau_{u_j} \leftarrow EstimateThetasAUC(\mathbf{s}_j, \mathbf{y}_j)$
// estimate τ_{l_j}, τ_{u_j} over subsamples
13 $\tau_{u^*} \leftarrow \frac{1}{\sqrt{2}}\tau_u + (1 - \frac{1}{\sqrt{2}}) \sum_{i=1}^{2} \frac{\tau_{u_i}}{2}$ // final estimate of τ_u
14 $\tau_{l^*} \leftarrow \frac{1}{\sqrt{2}}\tau_l + (1 - \frac{1}{\sqrt{2}}) \sum_{i=1}^{2} \frac{\tau_{l_i}}{2}$ // final estimate of τ_l
15 $mid \leftarrow \lfloor n \cdot (\tau_{u^*} + \tau_{l^*})/2 \rfloor$ // compute mid point position
16 $\delta \leftarrow n \cdot (1 - c)/2$ // half-width of rejection area
17 $u' \leftarrow \min\{\lfloor mid + \delta \rfloor, n\}$ // upper bound position
18 $l' \leftarrow \max\{1, \lfloor mid - \delta \rfloor\}$ // lower bound position
19 $\mathbf{ss} \leftarrow sort(\mathbf{s})$ // sort scores ascending
20 $\tau_{l'}, \tau_{u'} \leftarrow \mathbf{ss}[l'], \mathbf{ss}[u']$ // bounds of g
21 $s \leftarrow S.fit(\mathcal{T})$ // train classifier
22 $g \leftarrow$ **lambda** $\mathbf{x} : 1 - \mathbb{1}(\tau_{l'} \leq s.score(\mathbf{x}) \leq \tau_{u'})$ // selection function
23 $h \leftarrow$ **lambda** $\mathbf{x} : \mathbb{1}\{s.score(\mathbf{x}) \geq 1/2\}$ // final classifier
24 **return** (h, g)

AUCross iterates over a stratified K-fold partitioning of the training set (lines 3-6 of Alg. 9), where, for each of the folds k, we use $K-1$ folds to train a classifier s_k (using S) and to predict scores $s_k(\mathbf{X}_k)$ over the fold k (lines 4-5). We store all these predictions in $\mathbf{s}$ (line 7), and we use them to compute quantiles according to Prop. 3 (line 9 and Alg. 7). Since τ_l and τ_u are quantiles estimates, we can improve their second-order behaviour by randomly splitting the obtained scores and the target variable $(\mathbf{s}, \mathbf{y})$ into two parts (line 10) and repeat the estimation of quantiles over subsamples (lines 11-12). We combine the estimates according to Thm. 1 to obtain the final quantiles τ_{l^*} and τ_{u^*} (lines 13-14).
To meet the minimum coverage constraint, once again we consider the rejection area starting from the midpoint between the instances with score τ_{l^*} and τ_{u^*} (line 15). As for PluginAUC we also check not to exceed the range $[1, n]$ (lines 16-18). The final selective classifier (s, g) is then obtained by: *(i)* fitting S on the whole sample to get s (line 21); *(ii)* setting the bounds in the selection function g to the scores $\tau_{l'}$ and $\tau_{u'}$ at the boundaries l' and u' of the rejection area (lines 20 and 22).
We point out that the bounds computed by PluginAUC and AUCross may be sup-optimal, either because they do not achieve the minimum target coverage c or do not maximize the AUC. Even when restricting to the class of score-bounded selection functions, the reached AUC may be sub-optimal: for instance, we used (5.11) as a sufficient but not necessary condition during the proof of Prop. 1. We investigate the experimental performance of AUCross in the next section, also concerning such theoretical caveats.

5.5 Experimental Evaluation

5.5.1 Research Questions

To evaluate our proposed approach, we aim to answer the following questions:

Q1: Are the results comparable to other state-of-the-art approaches in terms of empirical coverage?

Q2: Are the results comparable to other state-of-the-art approaches in terms of selective AUC?

Q3: Is our approach comparable to the optimal approach in terms of selective AUC?

The code to reproduce all the experiments is available at `https://github.com/andrepugni/AUCbasedSelectiveClassification`.

5.5.2 Experimental Setting

Datasets We run experiments on nine real-world datasets (seven tabular datasets and two image datasets). We split available data instances into 75%

Table 5.1: Experimental datasets.

Dataset	# Features	Training Size	Test Size	Positive Rate
`Adult`	55	30,162	15,060	.246
`LendingClub`	65	1,364,697	445,912	.225
`GiveMe`	12	112,500	37,500	.067
`UCICredit`	23	22,500	7,500	.221
`CSDS1` [108]	155	230,409	76,939	.144
`CSDS2` [108]	35	37,100	12,533	.018
`CSDS3` [108]	144	71,177	23,288	.253
`CatsVsDogs`	64x64	20,000	5,000	.500
`CIFAR-10-cat`	32x32	50,000	10,000	.100

for training selective classifiers and 25% for performance testing. When possible, the split was based on timestamps to allow for out-of-time validation. Otherwise, a stratified random split is performed. A summary of the experimental datasets is reported in Table 5.1, including the number of features, size for training and test, and the rate of positives in the test set. Since we are interested in a binary classification task, we transform the `CIFAR-10` dataset labeling all the cat images of `CIFAR-10-cat` as the positive class, while images of other classes are considered the negatives.

Baselines. For **Q1** and **Q2**, we compare the performance of PluginAUC and AUCross to a few baselines. The first one is a bounded-improvement version of the plug-in rule [27] (PlugIn). PlugIn uses softmax as confidence function, i.e., in the binary case, $g(\mathbf{x}) = \mathbb{1}(\max\{s(\mathbf{x}), 1 - s(\mathbf{x})\} > \tau)$, and it estimates the τ parameter as the $(1 - c)$ quantile on a validation set. Such an approach requires splitting the training set into a dataset for building the classifier and a validation set for estimating τ (10% of the original dataset in our experiments). In contrast, AUCross allows for building the classifier using the whole training set. Second, we consider also the cross-fitting version of PlugIn (i.e. SCross presented in Chapter 3.2.2), where the confidence function is the softmax and the estimation of the bounds exploits the second-order improvement of of Thm. 1.
We also consider two state-of-the-art methods for selective classification, namely SelectiveNet [15] (SelNet), and the Self Adaptive Training [18] (SAT) methods. We also emphasize that the approach from [135] is not included in the baselines since it rejects the comparison of *pairs* of instances, while we focus on the rejection of predictions on instances in isolation.

Architectures and Hyperparameters. For tabular data, SelNet and SAT are built using a ResNet structure [104], while AUCross, PlugIn, PlugInAUC and SCross are based on a LightGBM classifier. We report in the Appendix results for other base classifiers, including the same base neural network architecture of SAT and SelNet. For `CatsVsDogs` and `CIFAR-10-cat`, a VGG architecture [136] is used for all the methods. For all the DNN ap-

proaches, 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.

Hardware. 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. For **Q1** we consider the empirical coverage, using six target coverages, namely $.75, .80, .85, .90, .95, .99$. For **Q2** we analyze the empirical selective AUC on the same six target coverages. We report in the Appendix B.2.3 results for the selective accuracy (1 minus the selective error rate), the positive rate of accepted instances and the training time. All the measures are computed on 1,000 bootstrap samples of the test set (25% of available data), as proposed by Rajkomar et al. [111]. Performances are shown in the format "mean ± stdev". For **Q3**, we consider the difference between the empirical AUC ($\widehat{AUC}$) of an Oracle with access to true labels and the one of AUCross[3], considering target coverage c at various levels, i.e. $.75, .80, .85, .90, .95, .99$. The difference is computed on the test sets of the experimental datasets.

5.5.3 Experimental Results

Q1: evaluating Empirical Coverage Table 5.2 shows that both PlugInAUC and AUCross achieve an empirical coverage close to the target c in most of the cases for tabular data. The most significant violation occurs for `LendingClub`, where all the analyzed methods have a large difference compared to the target coverage. In particular, SAT performs poorly on such a dataset. However, we emphasize that in this case, the empirical coverage is larger than the target one, implying that the coverage constraint is still satisfied. The other largest violation occurs for `CIFAR-10-cat`. Interestingly, the best method w.r.t. coverage is PlugInAUC, which has the smallest average violation. A striking case is the `CSDS2` dataset, which is the most imbalanced one with a positive rate of 1.8%. Here, both AUCross and PlugInAUC have small violations, while all other approaches have large ones. We argue that this is due to the bounds determined by Prop. 1 and Prop. 2, which are specific for positives and negatives, respectively, while the other methods calibrate the selection function independently of the class labels.

Q2: evaluating Selective AUC Regarding the empirical AUC, AUCross and PlugInAUC outperform the other methods in most cases. Interestingly, we see a clear pattern. When focusing on more balanced data, e.g. `CatsDogs`, all the methods improve AUC when abstention is allowed. Instead, for unbalanced datasets - e.g. `GiveMe`, `CSDS1` and `CSDS2` - we notice that traditional

[3]Similar results hold for PluginAUC.

Table 5.2: Empirical coverage (1,000 bootstrap runs over the test set, results as mean ± stdev). V (for violation) is the absolute difference between the empirical coverage and the target coverage c.

		Empirical Coverage					
	c	**AUCross**	**PlugIn**	**PlugInAUC**	**SCross**	**SelNet**	**SAT**
Adult	.99	**.989 ± .001**	.992 ± .001	.991 ± .001	.993 ± .001	.985 ± .001	.998 ± .001
	.95	**.950 ± .002**	.947 ± .002	.950 ± .002	**.950 ± .002**	.954 ± .002	.984 ± .002
	.90	.896 ± .003	**.900 ± .003**	.898 ± .003	.904 ± .003	.897 ± .003	.966 ± .002
	.85	.849 ± .003	.846 ± .003	.850 ± .003	**.850 ± .003**	.772 ± .004	.952 ± .002
	.80	.797 ± .004	.794 ± .004	.793 ± .004	**.802 ± .004**	.793 ± .004	.934 ± .003
	.75	**.750 ± .004**	.743 ± .004	.744 ± .004	.746 ± .004	.736 ± .004	.919 ± .003
Lending	.99	.996 ± .001	.995 ± .001	.997 ± .001	**.994 ± .001**	.997 ± .001	.073 ± .001
	.95	.980 ± .001	.970 ± .001	.980 ± .001	**.969 ± .001**	.985 ± .001	.073 ± .001
	.90	.959 ± .001	**.939 ± .001**	.958 ± .001	**.938 ± .001**	.928 ± .001	.073 ± .001
	.85	.936 ± .001	.909 ± .001	.935 ± .001	**.908 ± .001**	.866 ± .001	.073 ± .001
	.80	.912 ± .001	.880 ± .001	.911 ± .001	.879 ± .001	**.850 ± .001**	.073 ± .001
	.75	.886 ± .001	.850 ± .001	.884 ± .001	**.849 ± .001**	.880 ± .001	.073 ± .001
GiveMe	.99	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	**.944 ± .002**	.746 ± .003
	.95	**.950 ± .002**	.949 ± .002	.947 ± .002	**.950 ± .002**	.944 ± .002	.727 ± .003
	.90	.898 ± .002	.903 ± .002	.897 ± .002	.902 ± .002	**.900 ± .002**	.705 ± .003
	.85	.848 ± .002	.856 ± .002	.849 ± .002	.852 ± .002	**.848 ± .002**	.699 ± .003
	.80	.795 ± .003	.807 ± .003	.801 ± .003	**.800 ± .003**	.802 ± .003	.694 ± .003
	.75	.748 ± .003	.756 ± .003	.750 ± .003	.748 ± .003	**.754 ± .003**	.689 ± .003
UCICredit	.99	.988 ± .002	.993 ± .002	.993 ± .001	.993 ± .001	**.989 ± .002**	1.000± .000
	.95	.944 ± .003	.948 ± .003	.946 ± .003	**.951 ± .003**	.909 ± .004	1.000± .000
	.90	.896 ± .004	**.897 ± .004**	.900 ± .004	.906 ± .004	.904 ± .004	.994 ± .001
	.85	.840 ± .005	**.851 ± .005**	.846 ± .005	.848 ± .005	.855 ± .005	.994 ± .001
	.80	.789 ± .005	**.803 ± .005**	**.797 ± .005**	**.797 ± .005**	.810 ± .005	.994 ± .001
	.75	.737 ± .006	.751 ± .006	.739 ± .006	**.750 ± .005**	.759 ± .005	.994 ± .001
CSDS1	.99	**.990 ± .001**	.980 ± .001	.991 ± .001	.980 ± .001	.985 ± .001	.984 ± .001
	.95	**.950 ± .001**	.918 ± .001	.949 ± .001	.917 ± .001	.924 ± .001	.937 ± .001
	.90	**.898 ± .002**	.853 ± .002	.898 ± .002	.849 ± .002	.863 ± .002	.861 ± .002
	.85	**.849 ± .002**	.792 ± .002	.845 ± .002	.790 ± .002	.800 ± .002	.803 ± .002
	.80	**.799 ± .002**	.737 ± .002	.795 ± .002	.736 ± .002	.740 ± .002	.739 ± .002
	.75	**.748 ± .002**	.681 ± .002	.747 ± .002	.682 ± .002	.686 ± .002	.687 ± .002
CSDS2	.99	**.990 ± .001**	.984 ± .002	.991 ± .001	.983 ± .002	.980 ± .002	1.000± .000
	.95	**.952 ± .002**	.927 ± .003	.954 ± .002	.916 ± .003	.894 ± .003	1.000± .000
	.90	**.904 ± .003**	.851 ± .004	.907 ± .003	.834 ± .004	.805 ± .004	.880 ± .003
	.85	**.860 ± .004**	.773 ± .004	.861 ± .004	.761 ± .004	.712 ± .005	.817 ± .004
	.80	**.811 ± .004**	.715 ± .005	.818 ± .004	.687 ± .005	.663 ± .005	.756 ± .004
	.75	**.760 ± .004**	.651 ± .005	.776 ± .004	.620 ± .005	.564 ± .005	.697 ± .005
CSDS3	.99	**.991 ± .001**	.992 ± .001	.992 ± .001	**.991 ± .001**	.988 ± .001	.992 ± .001
	.95	**.948 ± .002**	.957 ± .002	.960 ± .002	.955 ± .002	.942 ± .002	.951 ± .002
	.90	**.901 ± .002**	.909 ± .002	.909 ± .002	.911 ± .002	.913 ± .002	.902 ± .002
	.85	**.849 ± .003**	.859 ± .003	.856 ± .003	.865 ± .003	.867 ± .003	.856 ± .003
	.80	**.805 ± .003**	.808 ± .003	.811 ± .003	.818 ± .003	.796 ± .003	.819 ± .003
	.75	.759 ± .003	**.758 ± .003**	.765 ± .003	.769 ± .003	.765 ± .003	.776 ± .003
CatsVsDogs	.99	.992 ± .002	**.990 ± .002**	.991 ± .002	.992 ± .002	.991 ± .002	.993 ± .002
	.95	.962 ± .003	.965 ± .003	**.954 ± .004**	.959 ± .003	**.954 ± .003**	.959 ± .003
	.90	.920 ± .004	.910 ± .004	**.907 ± .005**	.917 ± .004	.891 ± .005	.910 ± .005
	.85	.883 ± .005	.854 ± .005	.858 ± .005	.871 ± .005	**.852 ± .006**	.862 ± .005
	.80	.841 ± .006	**.797 ± .006**	.805 ± .006	.807 ± .006	.809 ± .006	.818 ± .006
	.75	.781 ± .006	.740 ± .007	**.749 ± .007**	.744 ± .007	.765 ± .007	.768 ± .007
CIFAR10-CAT	.99	1.000± .001	.991 ± .001	.989 ± .002	.992 ± .001	.992 ± .001	**.990 ± .002**
	.95	.998 ± .001	**.950 ± .003**	.948 ± .003	.953 ± .003	.956 ± .003	.952 ± .003
	.90	.994 ± .001	**.899 ± .004**	.898 ± .003	.903 ± .003	.904 ± .003	.913 ± .003
	.85	.987 ± .002	.849 ± .004	**.851 ± .004**	.886 ± .004	.863 ± .004	.854 ± .004
	.80	.969 ± .002	.796 ± .005	**.799 ± .004**	.882 ± .004	.806 ± .005	.803 ± .005
	.75	.945 ± .003	**.749 ± .005**	**.749 ± .005**	.879 ± .004	.763 ± .005	.759 ± .005
	#	21/54	12/54	8/54	15/54	8/54	1/54
	V	.026 ± .046	.021 ± .029	.013 ± .026	.028 ± .036	.029 ± .041	.171 ± .251

Table 5.3: Selective AUC (1,000 bootstrap runs over the test set, results as mean ± stdev). V (for violation) is the absolute difference between the empirical coverage and the target coverage c.

		Selective AUC					
	c	**AUCross**	**PlugIn**	**PlugInAUC**	**SCross**	**SelNet**	**SAT**
Adult	.99	**.929 ± .003**	.928 ± .003	.928 ± .003	.928 ± .003	.901 ± .003	.902 ± .003
	.95	**.935 ± .003**	**.935 ± .003**	.934 ± .003	.935 ± .003	.907 ± .003	.903 ± .003
	.90	**.943 ± .002**	.941 ± .003	.942 ± .003	.941 ± .003	.917 ± .003	.904 ± .003
	.85	**.950 ± .002**	.947 ± .003	.949 ± .002	.948 ± .003	.848 ± .005	.904 ± .003
	.80	**.958 ± .002**	.952 ± .003	.957 ± .002	.953 ± .003	.924 ± .004	.904 ± .003
	.75	**.963 ± .002**	.958 ± .003	.962 ± .002	.958 ± .003	.917 ± .004	.904 ± .003
Lending	.99	.983 ± .001	**.984 ± .001**	.984 ± .001	.983 ± .001	.973 ± .001	.664 ± .004
	.95	**.985 ± .001**	**.985 ± .001**	**.985 ± .001**	.984 ± .001	.973 ± .001	.664 ± .004
	.90	**.987 ± .001**	.986 ± .001	**.987 ± .001**	.986 ± .001	.981 ± .001	.664 ± .004
	.85	**.989 ± .001**	.988 ± .001	**.989 ± .001**	.987 ± .001	.977 ± .001	.664 ± .004
	.80	.990 ± .001	.989 ± .001	**.991 ± .001**	.989 ± .001	.991 ± .001	.664 ± .004
	.75	**.992 ± .001**	.990 ± .001	**.992 ± .001**	.990 ± .001	.976 ± .001	.664 ± .004
GiveMe	.99	.868 ± .004	.861 ± .004	**.869 ± .004**	.859 ± .004	.747 ± .007	.730 ± .006
	.95	.874 ± .004	.824 ± .006	**.875 ± .004**	.826 ± .006	.748 ± .007	.729 ± .006
	.90	**.883 ± .004**	.784 ± .007	**.883 ± .004**	.784 ± .007	.708 ± .008	.728 ± .007
	.85	**.890 ± .004**	.759 ± .008	**.890 ± .004**	.757 ± .008	.658 ± .009	.728 ± .007
	.80	**.898 ± .004**	.744 ± .009	.897 ± .004	.743 ± .009	.677 ± .009	.727 ± .007
	.75	**.904 ± .004**	.738 ± .010	**.904 ± .004**	.734 ± .010	.676 ± .010	.727 ± .007
UCICredit	.99	.772 ± .008	.772 ± .007	**.774 ± .007**	.770 ± .008	.765 ± .007	.768 ± .007
	.95	.778 ± .008	.769 ± .008	**.780 ± .007**	.765 ± .008	.718 ± .009	.768 ± .007
	.90	.785 ± .008	.758 ± .008	**.787 ± .008**	.754 ± .009	.707 ± .009	.769 ± .007
	.85	.792 ± .008	.746 ± .009	**.795 ± .008**	.734 ± .009	.668 ± .009	.769 ± .007
	.80	.800 ± .008	.722 ± .010	**.802 ± .008**	.710 ± .010	.644 ± .010	.769 ± .007
	.75	.809 ± .008	.695 ± .011	**.810 ± .008**	.679 ± .011	.620 ± .011	.769 ± .007
CSDS1	.99	**.686 ± .003**	.675 ± .003	**.686 ± .003**	.675 ± .003	.671 ± .003	.669 ± .003
	.95	**.689 ± .003**	.654 ± .004	**.689 ± .003**	.655 ± .004	.652 ± .004	.654 ± .004
	.90	**.693 ± .003**	.640 ± .004	**.694 ± .003**	.641 ± .004	.638 ± .004	.635 ± .004
	.85	.697 ± .003	.631 ± .004	**.698 ± .003**	.631 ± .004	.630 ± .004	.625 ± .004
	.80	.702 ± .004	.624 ± .004	**.703 ± .003**	.620 ± .004	.618 ± .004	.616 ± .004
	.75	.706 ± .004	.613 ± .005	**.707 ± .004**	.616 ± .005	.609 ± .005	.608 ± .005
CSDS2	.99	.616 ± .020	.585 ± .020	.587 ± .020	.616 ± .020	.607 ± .020	**.623 ± .019**
	.95	**.619 ± .020**	.572 ± .021	.589 ± .020	.581 ± .022	.587 ± .022	.623 ± .019
	.90	**.620 ± .020**	.566 ± .023	.590 ± .021	.570 ± .023	.560 ± .024	.588 ± .022
	.85	**.631 ± .021**	.552 ± .024	.590 ± .021	.572 ± .025	.562 ± .026	.589 ± .023
	.80	**.631 ± .021**	.532 ± .025	.592 ± .021	.555 ± .026	.547 ± .027	.582 ± .024
	.75	**.635 ± .021**	.536 ± .027	.592 ± .021	.541 ± .028	.579 ± .029	.575 ± .026
CSDS3	.99	**.851 ± .003**	.850 ± .003	.850 ± .003	.850 ± .003	.844 ± .003	.844 ± .003
	.95	**.858 ± .003**	.852 ± .003	.855 ± .003	.853 ± .003	.851 ± .003	.849 ± .003
	.90	**.865 ± .003**	.855 ± .003	.863 ± .003	.856 ± .004	.850 ± .004	.855 ± .004
	.85	**.873 ± .003**	.857 ± .004	.871 ± .003	.858 ± .004	.851 ± .004	.858 ± .004
	.80	**.880 ± .003**	.859 ± .004	.878 ± .003	.859 ± .004	.859 ± .004	.858 ± .004
	.75	**.887 ± .003**	.858 ± .004	.884 ± .003	.859 ± .004	.843 ± .004	.853 ± .004
CatsVsDogs	.99	.984 ± .003	**.985 ± .002**	**.985 ± .002**	.985 ± .002	.979 ± .003	**.985 ± .002**
	.95	.986 ± .003	.985 ± .002	**.987 ± .002**	.987 ± .003	**.987 ± .002**	.987 ± .002
	.90	.987 ± .003	.984 ± .002	.988 ± .002	.988 ± .003	**.990 ± .002**	.989 ± .002
	.85	.987 ± .003	.984 ± .002	.989 ± .002	.988 ± .003	**.990 ± .002**	.990 ± .002
	.80	.987 ± .003	.985 ± .002	.990 ± .002	.989 ± .003	**.991 ± .002**	.990 ± .002
	.75	.987 ± .003	.983 ± .002	.990 ± .002	.988 ± .003	**.992 ± .002**	.990 ± .002
CIFAR10-CAT	.99	.890 ± .009	.915 ± .007	.909 ± .008	**.922 ± .007**	.896 ± .008	.893 ± .008
	.95	.889 ± .009	.895 ± .009	**.911 ± .007**	.906 ± .009	.902 ± .008	.863 ± .011
	.90	.889 ± .009	.896 ± .009	**.913 ± .007**	.851 ± .013	.899 ± .010	.760 ± .017
	.85	.888 ± .009	.898 ± .009	**.916 ± .007**	.815 ± .016	.676 ± .022	.574 ± .024
	.80	.885 ± .009	.906 ± .008	**.919 ± .007**	.804 ± .017	.359 ± .026	.572 ± .024
	.75	.870 ± .011	.905 ± .008	**.921 ± .007**	.797 ± .018	.408 ± .028	.575 ± .025
	#	28/54	4/54	29/54	6/54	5/54	2/54

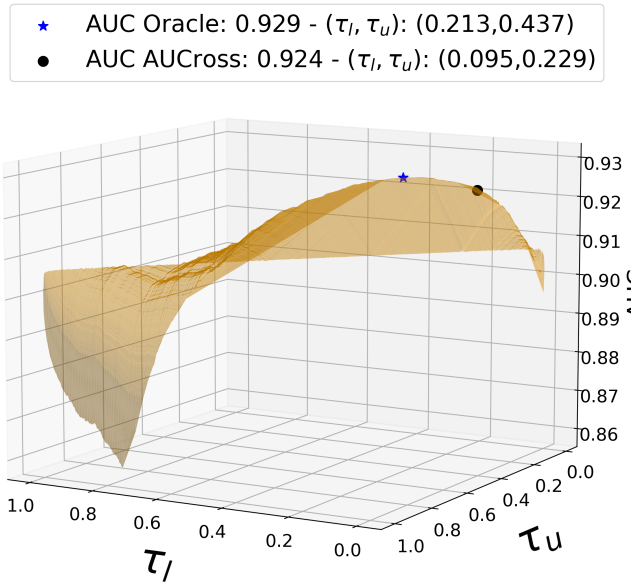


Figure 5.7: AUC by varying bounds: Adult and $c = .90$.

methods fail to improve the empirical AUC when decreasing the coverage. Conversely, the AUC-specific methods, i.e. AUCross and PlugInAUC, are able to improve their performance, increasing the selective AUC once abstention is allowed. This confirms the goodness of the approach in detecting instances that are not correctly ranked by the classifier. Nevertheless, a trade-off exists in terms of selective accuracy, because both AUCross and PlugInAUC are outperformed in terms of accuracy over the accepted region. We report these results in the Appendix Table B.1. This is not surprising since our algorithms are aiming to improve the AUC metric, and it is well-known that accuracy and AUC cannot be jointly optimized [21]. The main reason for this trend is a by-product effect of our rejection strategy: while traditional methods end up rejecting more instances belonging to the positives, PluginAUC and AUCross keep the positive rate stable over accepted instances. We also report these results in the Appendix Table B.2.

Q3: AUCross vs Oracle As our theoretical results only provide sufficient conditions to improve AUC when abstaining, we investigate how well our approach approximates the optimal solution to the problem of maximizing $\widehat{AUC}(s, g|\mathcal{T}_n)$ s.t. $\hat{\phi}(g|\mathcal{T}_n) \geq c$. We consider the AUCross algorithm and we limit our attention to the family of score-bounded selection functions and compute the results over the test set $\mathcal{T}_n$. We enumerate all the score-bounded selection functions $g(\mathbf{x}) = 0$ iff $\tau_l \leq s(\mathbf{x}) \leq \tau_u$ by varying the bounds $\tau_l, \tau_u \in \{s(\mathbf{x}_i) \mid (\mathbf{x}_i, y) \in \mathcal{T}_n\}$ such that $\hat{\phi}(g|\mathcal{T}_n) \geq c$. The Oracle approach chooses the function g which maximizes $\widehat{AUC}(s, g|\mathcal{T}_n)$. This strategy requires knowing the true class of instances in $\mathcal{T}_n$; hence it is not feasible in practice. Table 5.4 shows the difference between the empirical AUC ($\widehat{AUC}$) achieved by

Table 5.4: Absolute deviation in empirical AUC of AUCross w.r.t. an Oracle.

c	Adult	Lending	GiveMe	UCICredit	CSDS1	CSDS2	CSDS3	CatsVsDogs	CIFAR-10-cat
.99	.0003	.0000	.0004	.0008	.0003	.0057	.0001	.0000	.0001
.95	.0011	.0000	.0009	.0008	.0012	.0050	.0005	.0000	.0002
.90	.0022	.0000	.0004	.0007	.0024	.0080	.0011	.0000	.0006
.85	.0033	.0000	.0003	.0013	.0033	.0027	.0017	.0004	.0016
.80	.0033	.0000	.0006	.0007	.0039	.0006	.0020	.0004	.0049
.75	.0037	.0001	.0005	.0001	.0053	.0027	.0023	.0000	.0193

the Oracle and the one of AUCross on the test sets of the experimental datasets. We notice that AUCross reaches the Oracle performance for the `LendingClub` and `CatsVsDogs` datasets. For the datasets `Adult`, `CSDS1`, `CSDS3` and `CIFAR-10-cat`, there are larger violations for smaller c's, while we do not see such a trend for `GiveMe`, `UCICredit` and `CSDS2`. In our opinion, this might suggest that the strategy of centering the rejection area at a midpoint (line 15 of Alg. 9) might be sub-optimal in terms of AUC improvements. Figure 5.7 shows the performance over the space of bounds in a specific case, highlighting the bounds of Oracle and AUCross.

5.6 Conclusions

Selective classification can help prevent poor or even harmful decisions by abstaining from making an output, possibly deferring the decision to a human. This is especially relevant in all the socially sensitive contexts, which are often characterized by imbalanced classes. In these scenarios, the traditional accuracy measure fails to capture the best classifier. Hence, we have extended the selective classification framework to a widely used classifier evaluation metric, the Area Under the ROC Curve. Through an analytic characterization, we devised two methods for computing an estimator of the best score-bounded selection function. These methods are effective and outperform existing approaches designed for optimizing accuracy.

Chapter 6

A Case Study: the LSP system

The goal of this chapter is to describe the design, architecture, and evaluation of the Leasing Score Prediction (LSP) system – a credit scoring and credit rating system for the leasing sector deployed at the Italian Association of Leasing Companies (ASSILEA). Due to its challenging objectives, the design and complexity of the LSP system represent a unique contribution to the best practices in the field of credit scoring.
We cover requirements by managers, users, and regulations about rigorous backtesting, statistical validation, calibration, explainability, robustness, and, most importantly for the thesis contribution, *uncertainty self-assessment via selective classification* addressing *Req. 3*.
The Chapter is based on [137], a work presented at IEEE-Big Data 2023.

6.1 Introduction

Credit scoring is both a well-studied and an extremely relevant application of data science and advanced analytics [138]. Lease scoring, particularly in the application scenario tackled in this Chapter, brings several specific objectives that deserve the development of a tailored system, which we call Leasing Score Prediction (LSP). In particular, Lease financing is highly regulated and monitored, with applicable laws and guidelines about credit scoring (see, e.g., the European Banking Authority[1]), financial accounting (see, e.g. IFRS9[2] and Basel III IRB[3]), anti-money laundering and counter-terrorist financing[4], consumer credit and data protection [139]. Moreover, the assessment of the lessee creditworthiness is classified as a high-risk application by the draft European Union (EU) regulation on AI (AI Act) [4].
The LSP project started in January 2020 to develop a risk scoring and risk rating system based on supervised Machine Learning (ML), able to predict defaults of payments by the lessee at the time of contract definition. The

[1] https://www.eba.europa.eu/regulation-and-policy/credit-risk
[2] https://www.ifrs.org/issued-standards/list-of-standards/ifrs-9-financial-instruments/
[3] https://www.bis.org/basel_framework/chapter/CRE/30.htm
[4] https://www.leaseurope.org/policy/compliance/fraud

available data, collected in a centralized credit information system, are contributed by more than 40 leasing companies associated with the Italian leasing association (ASSILEA). Data distributions are then a mixture resulting from different target markets, types of lessees/assets, contract conditions, and risk appetites.
The goal of LSP is to address the following six objectives, namely:

O1: LSP is expected to outperform a scoring system trained on the single associate data;

O2: LSP should handle missing data with external sources;

O3: LSP should provide an uncertainty self-assessment for both the input features and the output scores;

O4: LSP should be able to explain its own scores;

O5: LSP should provide real-time answers;

O6: LSP is expected to adapt to prior distribution shifts

The rationale behind *Objective 1* is building a system that can leverage the whole data available and not only those belonging to a single associate. As a result, one should obtain a more general AI system, with tangible improvements in terms of predictive performance.
Regarding *Objective 2*, the available data may not represent the complete information used for decision-making by the single associate: for example, leasing companies do not contribute external credit bureau information or corporate data sources. Hence, LSP has to properly deal with this asymmetry in the conferred information.
For *Objective 3*, risk score model performances - such as AUC and calibration error - can broadly vary over (subsets of) the input data distributions, depending on the quality of the input. This requires the model to be able to (self) identify those instances where it is more prone to make mistakes.
Concerning *Objective 4*, multiple stakeholders and regulations mandate a clear understanding of the reasons behind an ML model output, including the explainability of decision drivers and counterfactual analyses.
With *Objective 5* we address the need of final users to query the system interactively in evaluating what-if scenarios.
Finally, *Objective 6* answers the urge to deal with distribution shifts that can occur over time, e.g., due to the economic cycle or disruptive events such as floods, earthquakes, or pandemics. In particular, while working on this project, Italy was the first Western country to face the COVID-19 outbreak, and to implement social counter-measures, e.g., lock-downs, and fiscal counter-measures, e.g., suspension of credit payments[5]
The LSP system entered production in April 2022 for a pilot subset of associate leasing companies, and from July 2022, it became gradually subscribable

[5]See `https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Italy`.

to all of them. We describe the design, the architecture, and the evaluation of LSP, regarding the above specific objectives. The rest of the chapter is structured as follows. First, we discuss some related work in Section 6.2. Section 6.3 summarizes the background on credit scoring, and Section 6.4 introduces the scenario of lease financing. Section 6.5 discusses the LSP framework. In particular, Section 6.5.3 describes how we can use selective classification methods to (self-)asses uncertainty around the predictions. Section 6.6 focuses on experimental validation. Finally, we summarize lessons learned and possible extensions.

6.2 Related Work

As LSP aims to address different objectives, here we briefly overview the main covered aspects. Due to the large body of literature, we prioritize survey papers, where applicable, and recent works.

Credit Scoring Credit scoring models assess the creditworthiness of a lender [138], either w.r.t. an individual loan or w.r.t. a portfolio of loans. The latter case is typically considered when monitoring the status of a portfolio of active (i.e., already started) contracts, e.g., for early warnings. In this work, we tackle the former case: LSP predicts the probability of default w.r.t. a leasing contract when it is about to start. Problems related to credit scoring include business failure/bankruptcy prediction [140], where similar predictive features and models are adopted, and loan survival analysis [141], which estimates the time to default.
The survey by Markov et al. [138] provides a systemic review of the most recent works in credit scoring, also concerning the common problems tackled.

Model selection Statistical and machine learning models have been used for credit scoring in econometrics, banking, finance, and data science [142, 9]. The former methods focus on summarization and (causal) inference, while the latter focus mainly on prediction [143]. Ensemble methods are the state-of-the-art machine learning classification models for tabular data [144, 104, 103], in particular for imbalanced data [145], and for credit scoring [146]. They exhibit the best predictive performances in benchmark results [147, 148], also when restricting to simple base classifiers [149]. The superior predictive performance of ensembles was confirmed on our leasing contract data by preliminary experiments, for which LSP core model has been set to an ensemble of gradient-boosting models.

Model Calibration Calibration of scoring models aims at producing risk scores that can be interpreted as the probability of default [150]. Calibration techniques can be model-agnostic - e.g., sigmoid, isotonic, beta calibration - or model specific, as the one for decision tree models [151] used in a bank credit scoring systems [152]. With this regard, we empirically notice that the

cross-fitting approach used by LSP helps improve the calibration of the base classifier without the need for post-processing techniques.

Explainability. Complex ML scoring models lack the interpretability of their internal logic and the explainability of their outputs. Explainability of AI (XAI) methods [1] are being increasingly used in the context of credit scoring [153].
Common approaches consider post-hoc interpretability of models [154]. For instance, the explainability of the model's output has been explored by feature attribution (such as the Shapley values [155]) in [156, 157]. We also resort to Shapley values for factual explanations of scores. Factual explanations answer the "*why?*" of a model's output.
Alternatively, counterfactual explanations [158] answer the "*why not?*" question by providing instances that are similar to the one under analysis but that receive a lower risk score. Different methods for searching counterfactuals have been proposed in the context of credit scoring [159, 160] and contrasted to expert-generated explanations [161]. All these proposed methods do not account for a critical issue in a deployment scenario: computational efficiency, an enabling factor for interactive usage of the credit scoring system (our Objective O5) in what-if analysis. FasterRisk [162] (fast and accurate interpretable risk scores), markedly faster than its competitors, "produces a collection of high-quality risk scores within minutes". Concerning this aspect, the LSP system produces risk scores, factual and counterfactual explanations in 1.1 seconds on average.

Data shifts. Probabilistic predictive models, including credit scoring models, suffer from shifts in the probability distribution from training to test data [163, 164]. Several approaches have been considered to mitigate the impact of data shifts for credit scoring [108]. Prior shifts, namely change in the distribution of defaults, directly impact on calibration of predictions. LSP deals with predictable prior shifts, for which an estimate of the new prior is available. Such an estimate is provided by official statistics forecasts.

Related systems. Table 6.1 compares LSP to a few complex credit scoring systems that are built and evaluated on real large-scale data. LSP is the only system covering the whole spectrum of features.

6.3 Credit Scoring Background

Credit scoring and credit rating aim to estimate the lender's probability of default (PD) to meet the contractual obligations. Credit risk models, particularly in the banking sector, cover other metrics of interest: the expected percentage of loss given default (LGD), the expected exposure amount at default (EAD), and the expected loss ($\text{EL} = \text{PD} \cdot \text{LGD} \cdot \text{EAD}$). These are not part of the LSP project, mainly due to their minor relevance in the context of leasing: EAD

Table 6.1: Related credit scoring systems. OOT $d \times n$ is Out of Time with d = test period duration, n = number of periods.

	[165]	[152]	[166]	LSP
Domain	Loans	Bankruptcy	Mortgages	Leasing
Scoring	XGBoost	LightGBM	XGBoost	Ensemble
Rating		✓		✓
Calibration		✓		✓
Uncertainty				✓
Factual expl.	✓	✓	✓	✓
Counterfactuals				✓
Prior shift				✓
Self-learning				✓
Backtesting	Holdout	OOT 1y×1	Holdout	OOT 3m×23
Deployed	✓			✓

can be easily calculated, and LGD is typically a small fraction – due to the ownership of the asset by the lessor. Similarly, we skip profit-based [167, 168] versions of the evaluation metrics introduced next.

Credit Scoring A credit scoring model is a function $\xi_t : \mathcal{X}_t \to [0, 1]$ mapping from a feature space $\mathcal{X}_t$ to a risk score value between 0 and 1. The feature space distribution and the model can change over time: the subscript t specifies the point of time t at which the model is used. We omit the subscript if no ambiguity arises. The risk score $\xi(\mathbf{x})$, or simply the score, of a (credit application) instance $\mathbf{x}$ estimates the likelihood that a risk event occurs, such as, for instance, the insolvency within a certain time frame Δt from t. We denote by y the true outcome, namely $y = 1$ if the event occurs (positive instance) and $y = 0$ otherwise (negative instance). Notice that the realization y can be known only if the credit application is approved and the credit contract starts. We assume that at time t we have access to a *training set* $\mathcal{T}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_{train}}$ drawn from the unknown distribution $P_t(\mathbf{X}, Y)$. This can be used to learn a credit scoring model according to 2.3. We say the credit scoring model $\xi(\mathbf{X})$ is *calibrated* [74] if the following holds:

$$P(Y = 1|\xi(\mathbf{X}) = p) = p, \tag{6.1}$$

i.e. the score can be interpreted as the probability that the event will occur.

Metrics Quality metrics in credit scoring models can be classified based on the purpose [169, 130]. Metrics that evaluate the discriminative power to separate positives and negatives include AUC, Gini, and KS statistics. As introduced in 5.1, we recall that the Area Under the ROC Curve (AUC) [170] is the probability (the higher, the better) that a randomly drawn positive instance receives a higher (credit) score than a randomly drawn negative instance, namely

$$AUC(\xi) = \mathbb{E}_{\mathbf{X}_0 \sim \mathcal{D}_{t,0}, \mathbf{X}_1 \sim \mathcal{D}_{t,1}}[\mathbb{1}\{\xi(\mathbf{X}_1) > \xi(\mathbf{X}_0)\}].$$

where $\mathcal{D}_{t,1}$ (resp., $\mathcal{D}_{t,0}$) is the conditional distribution $\mathcal{D}_t$ of positives (resp., negatives) at time t. We also recall that the Gini coefficient is linearly related to the AUC as follows $Gini = 2 \cdot AUC - 1$ [133].
Metrics that evaluate the accuracy of the scores include BLS and Log-loss scores. The Brier Loss Score (BLS) [171] is the expected quadratic loss of scores:

$$BLS = \mathbb{E}[(\xi(\mathbf{X}) - Y)^2]$$

A metric that evaluates the degree of calibration over a binning $b_1, \ldots, b_R$ of the score interval $[0, 1]$ is the Binary Expected Calibration Error (BIN-ECE) [74]:

$$BIN\text{-}ECE = \mathbb{E}[|P\,(Y = 1|\xi(\mathbf{X}) \in b_i) - \mathbb{E}[\xi(\mathbf{X})|\xi(\mathbf{X}) \in b_i]|]$$

Finally, variants of those metrics have been proposed to take into account unbalanced distributions of positives and negatives. In particular, the Brier Skill Score [171] is defined as the error reduction (the higher, the better): $BSS = 1 - {}^{BLS}/_{BLS_{ref}}$ relatively to BLS_{ref}, which is the Brier Skill Score of a baseline classifier scoring any instance as the fraction of positives observed in the training set $\mathcal{T}_{train}$.
For evaluation, we estimate these metrics over a hold-out dataset $\mathcal{T}_{test} = \{(\mathbf{x}, y_{\mathbf{x}})\}$, called the *test set*, using the empirical counterparts of their definition at the population level. In particular, *BIN-ECE* is estimated over the test set as the weighted absolute difference between the observed default rate OB_i in the bin b_i (which estimates $P(Y = 1|\xi(\mathbf{X}) \in b_i)$) and the mean score in the bin b_i (which estimates $\mathbb{E}[\xi(\mathbf{X})|\xi(\mathbf{X}) \in b_i]$).

Credit Rating. A credit rating model is a function $\texttt{rating}_t : \mathcal{X}_t \to \{1, \ldots, R\}$, mapping instances to an ordered set of R rating classes. Typically R is in the range 5-10, with class 1 denoting low risk and class R denoting high risk. Each rating class i is assigned an apriori probability of default PD_i, increasing with i, for which the model is expected to be calibrated, namely $P(Y = 1|\texttt{rating}_t(\mathbf{x}) = i) = PD_i$. A variant of BIN-ECE quantifies the calibration error of rating models:

$$BIN\text{-}ECE\text{-}\texttt{rating} = \mathbb{E}[|P(Y = 1|\texttt{rating}(\mathbf{X}) = i) - PD_i|]$$

where the observed default rate is compared to the expected one, rather than the average score. BIN-ECE-`rating` is estimated over the test set as the weighted mean value of $|OB_i - PD_i|$, where OB_i is the observed default rate for instances in the test set rated in class i. Moreover, the observed rate OB_i is expected to be lower or equal than the predicted rate PD_i – otherwise, the model underestimates the actual risk. A binomial test of the hypothesis $H_0 : P(Y = 1|\texttt{rating}(\mathbf{x}) = i) \leq PD_i$ is adopted at some confidence level for such a purpose. Another statistical test based on the multinomial distribution is the Extended Traffic Light [172], which results in an immediately grasped colour (green, yellow, orange, red).

6.4 The Leasing Industry

A lease contract conveys the right to use an asset for a period of time in exchange for payment [173]. The leasing company, called the *lessor*, owns the asset. The asset user, called the *lessee*, can be a company or a natural person. The asset is bought by the lessor from a *vendor* based on the lessee's requirements. The payments of the contract include an initial payment, periodic installments, and an optional redemption price. Lease financing has many advantages. The asset is immediately available for the lessee, while its cost is split over the contract duration. The direct acquisition of expensive equipment, plant and machinery requires, instead, capital outlay or securities to access credit financing. For the lessor, the asset can be repossessed when the lessee defaults on payments; hence the lessor's interest is fully secured. These and other advantages have been driving a continuous growth of the leasing market in the last decade, with an estimated volume of ≈$1,675B worldwide in 2023 (from ≈$1,520B in 2022). The European market is estimated ≈€415B in 2022.
Assilea (`https://www.assilea.it`) is the Italian leasing association, with more than 70 associated companies, representing more than one half of the Italian market in 2021 (≈$18B in value). Each associate runs its own business, possibly in competition with other associates, adhering to its own credit risk policies and procedures regarding target markets, type of assets, region of operation, appetite for risk, etc. More than 40 of the associated companies contribute monthly to Assilea raw data on newly signed contracts and on the payment status of active contracts. These data are integrated into a data warehouse of past and active lease contracts. Assilea provides back to the contributing associates a credit information service called BDCR ("Banca Dati Centrale Rischi del Leasing" in Italian). Information provided aggregates data from the various associates to summarize the number and status of a prospective lessee's past and active lease contracts. The benefit for the associates in contributing to the BDCR consists of more in-depth information on the past history and the current financial commitments of prospect lessees leading to a better evaluation of their creditworthiness. The contribution to the BDCR and its usage is ruled by a self-regulation code signed by the associates. The code is compliant to the European General Data Protection Regulation [139]. Prospect lessees can submit complaints about data processing by the BDCR to a national authority which monitors credit information systems.

6.5 The LSP Framework

The LSP project has followed the de-facto industry standard of the CRISP-DM process model [174]. Business objectives have been determined by involving a few representative associate companies. Users with different roles (risk managers, risk analysts, credit analysts, data analysts, head of organization and control, head of legal) have been interviewed several times at different project stages, both in person and through a questionnaire. Various definitions of de-

fault payments have been discussed with the users. The one adopted considers the occurrence of severe insolvency or legal default within a lookahead period of 12 months from the signing date of the contract. Severe insolvency is triggered in case of overdue payments for a total of 5% of the contract or a total of 35% of the payments due in the next semester. A legal default, causing contract termination, is triggered in case of overdue payments for at least six monthly installments for real estate assets or four monthly installments for other assets, even if not consecutive, or equivalent amount [175].

6.5.1 Architecture, workflow, inputs and outputs

We describe the architecture and workflow of the LSP framework and detail the design rationale of the critical and novel components of the LSP system, also w.r.t. the objectives stated in the introduction. The components of the framework have been implemented in Python (LSP), Sadas Engine[6] (ETL, data warehouse), and Java (web service).

Architecture and workflow. Figure 6.1 summarizes the architecture of the LSP system and the workflow of data and information from and towards the leasing associate companies. Monthly, each associate contributes raw data to the BDCR credit information system, which stores in a centralized data warehouse the data on new contracts and the status of payments of active contracts. Data preprocessing outputs a training set at the granularity of one instance per contract. Financial indicator features are collected through an external credit bureau service both in bulk (offline) and on the fly at the time of predictions. This is intended to answer the *Objective 2*. The credit scoring model is trained quarterly from the training set. Users at a leasing associate submit queries to the LSP system through a web interface. A query includes the prospect contractual information, including the (id of the) lessee, the vendor(s), the guarantor (s), the amount of the contract, its duration, the initial payment amount, and the redemption price. These data are joined with financial indicators and other pre-computed features, mainly behavioral indicators, not only about the lessee, but also about the vendors, and the guarantors – since good vendors tend to bring good lessees. The joined feature values are passed in input to the scoring model, which produces a creditworthiness score, a rating class on an 8-scale, and two quality labels: one about the input feature values, and one about the output score. The quality labels are intended to answer the *Objective O3*. An eXplainability in AI (XAI) module adds factual and counterfactual explanations of the predicted score. The (counter)factual explanations are intended to answer the *Objective O4*. The outputs are arranged and sent back to the user by the web service. An example output is shown in Figure 6.2. The average time for serving a user query is 1.1 seconds. The most time-consuming task is the invocation of the external credit bureau service (1.68 seconds on average), which, however, takes place only if the user query regards a lessee that is a company with some specific legal form, and such that

[6] A big data columnar database, see `https://www.sadasdb.com/`.

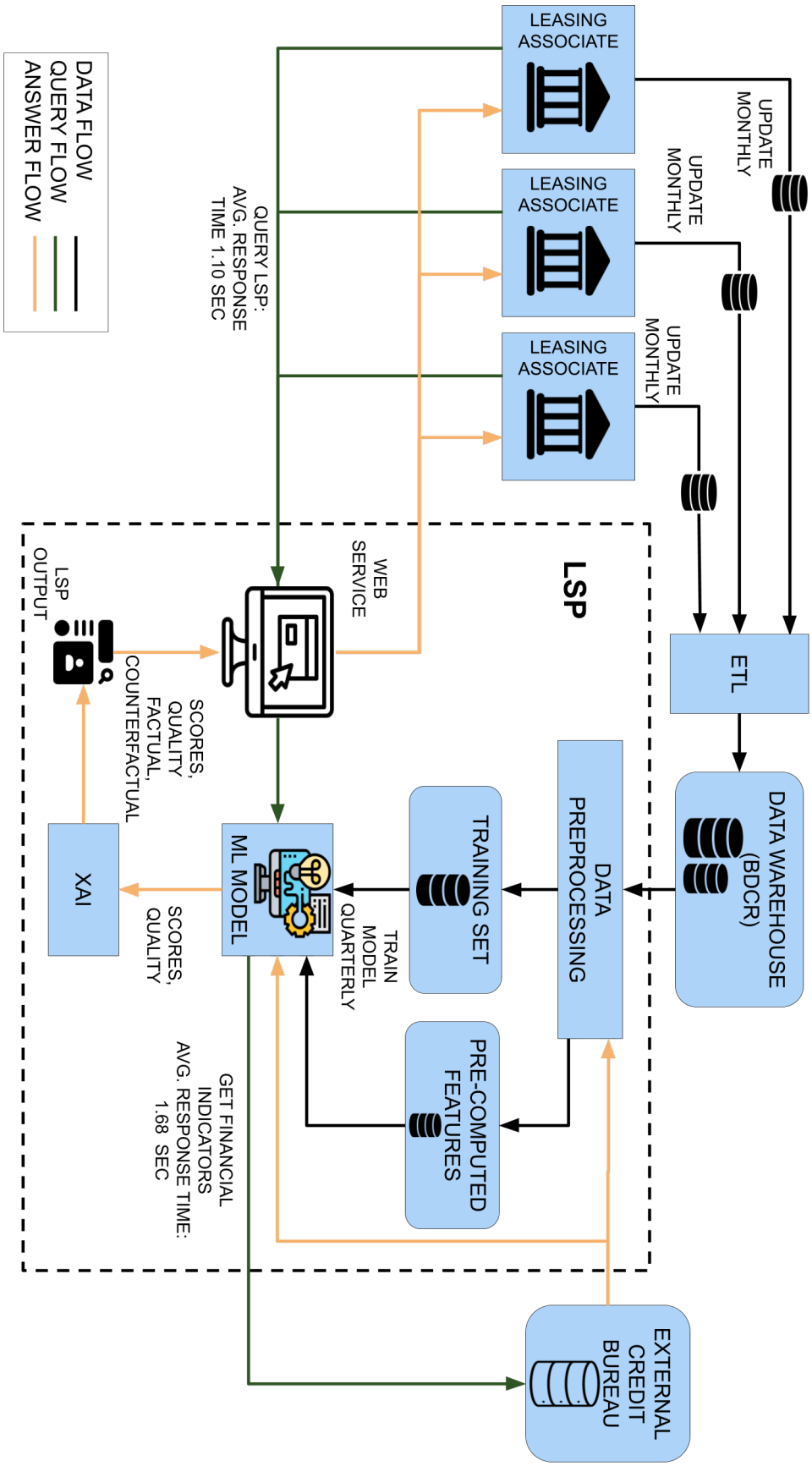


Figure 6.1: The LSP workflow.

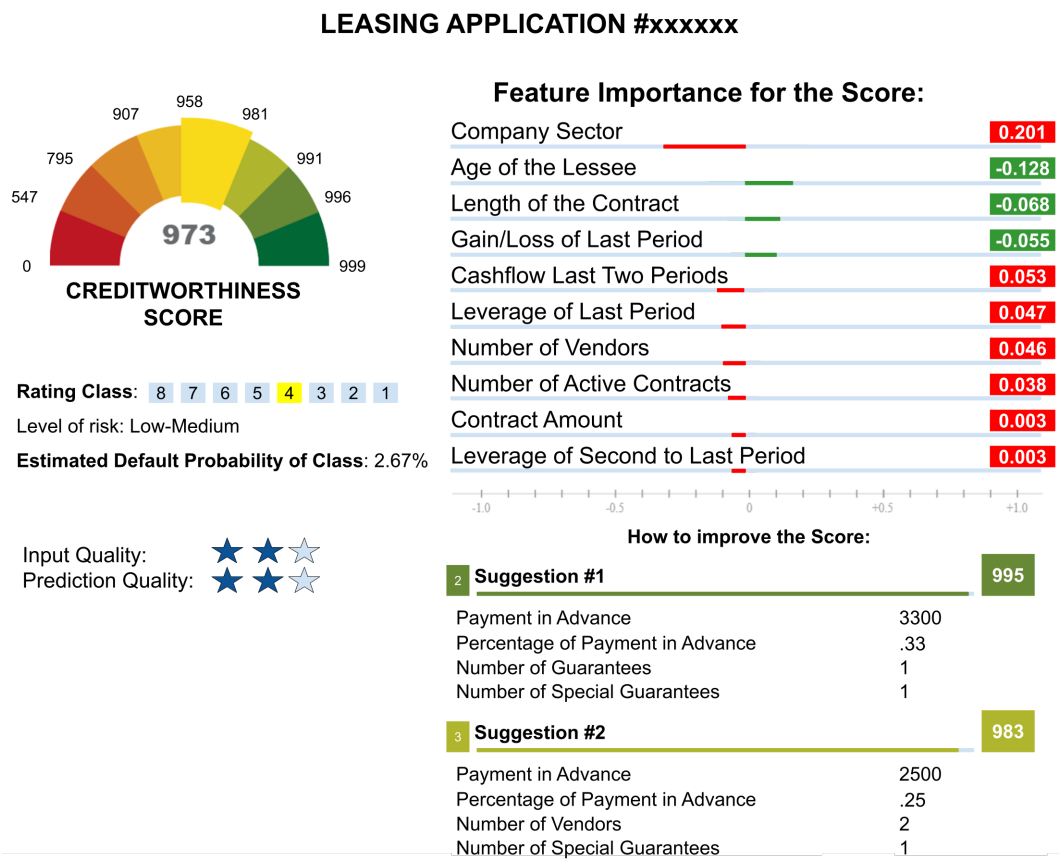


Figure 6.2: An example of LSP output.

there are no updated financial indicators cached about. These computational performances allow for interactively querying the LSP, e.g., by changing some value of the contractual data to re-evaluate the score and its explanation in a what-if scenario *(Objective O5)*. Also, the performances are optimized for scaling to a potential number of 600K calls per year, which is currently the number of calls to the BDCR credit information system from all associates.

Input features. Data preprocessing is run monthly after updating the BDCR data warehouse. Predictive features and a target feature are computed for each contract, with information available at the time of the contract starting date. These features are about:

- *the contract* including duration, total, initial payment, redemption price, number of installments, fraction of initial payment, fraction of redemption price, type of asset, state of the asset (e.g., used, new, to build);

- *the lessee,* including age of business (or age of natural persons), region/province of company headquarters (residence for natural persons), industry sector classification, legal form, financial indicators about the last and the second to last accounting closing year available;

- *the vendor(s),* including region/province of company headquarters, industry sector classification, legal form;

- *the guarantor(s),* including type of guarantee (e.g., surety or obligation to buy), amount guaranteed, legal form;

- *the current commitments,* including number/volume of active contracts by the lessee, by the vendor(s), by the guarantor(s);

- *the behavioural indicators,* including number/volume of contracts concluded by the lessee, by the vendor(s), by the guarantor(s), and number/volume of defaults for those contracts in the previous six months, 12 months, 24 months, ten years;

- *the target feature,* which is 1 (positives) if the default of the contract has been observed during the lookahead period, 0 (negatives) if no default has been observed in the lookahead period, and '*unknown*' if the lookahead period has not passed yet, but no default has been observed so far.

LSP outputs. Figure 6.2 shows an example of the output returned to the user. The following items are included:

- *a creditworthiness score*, predicting the good outcome of the contract obligations w.r.t. the default notion. The creditworthiness score is in a three digit range 0-999, and it is calculated as $(1 - r) \cdot 999$ where r is the risk score of the LSP model;

- *a rating class*, in the range 1 to 8, where 1 is very low risk and 8 very high risk. Class boundaries are shown in the rainbow around the creditworthiness score, and the rating class assigned is larger than the others. Also, the rating class is shown as a number and with a textual description. On mouse-over the rainbow, the textual description of any class can be viewed. The estimated PD of the assigned rating class is also reported;

- *two quality labels*, on a 1 to 3-stars scale, referring to the quality of input and to a self-assessed quality of the predicted score.

- *a factual explanation*, of the score, using a feature importance plot, where for the top 10 features, the positive or negative contribution to the creditworthiness score is reported both as a bar and a value, in colour (green for positive, red for negative);

- *up to three counterfactual explanations*, consisting of changes in the features of the contract that lead to better creditworthiness score and, possibly, to a better rating class.

Algorithm 10: LSP.fit()

Input : $\mathcal{T} = \{\mathbf{X}, \mathbf{y}\}$ - training set,
S - LightGBM classifier,
K - number of folds
p_t - estimated positive rate in the test set
Output: s - probabilistic classifier,
π - prediction quality evaluator

1 $\mathcal{T}_1, \ldots, \mathcal{T}_K \leftarrow \text{StratifiedKFold}(\mathcal{T}, K)$ `// partitioning`
2 **for** $k \in 1, \ldots, K$ **do** `// for each fold`
3 $\quad \mathcal{T}_{-k} = \mathcal{T} \setminus \mathcal{T}_k$ `// training data`
4 $\quad s_k \leftarrow S.fit(\mathcal{T}_{-k})$ `// train` k^{th} `classifier`
5 $\quad \gamma_k \leftarrow mean(\mathbf{y}_{-k})/(1 - mean(\mathbf{y}_{-k}))/p_t * (1 - p_t)$ `// odds` k^{th} `fold`
6 $\quad \mathbf{s}_k \leftarrow s_k.score(\mathbf{X}_k)$ `// score test data`
7 $\quad \mathbf{s}_k \leftarrow \frac{\mathbf{s}_k}{\mathbf{s}_k + \gamma_k(1-\mathbf{s}_k)}$ `// Bayesian correction`
8 $\mathbf{s} \leftarrow \cup_{k=1}^{K} \mathbf{s}_k$ `// store all the scores`
9 $\tau_{l,.9}, \tau_{u,.9} \leftarrow EstimateThetaAUC(\mathbf{s}, .90)$ `// bounds for` $c = .90$
10 $\tau_{l,.5}, \tau_{u,.5} \leftarrow EstimateThetaAUC(\mathbf{s}, .50)$ `// bounds for` $c = .50$
11 $s.score \leftarrow$ **lambda x** : $\frac{1}{K}\sum_{k=1}^{K} s_k.score(\mathbf{x})$ `// score function`
12 $\gamma = mean(\mathbf{y})/(1 - mean(\mathbf{y}))/p_t * (1 - p_t)$ `// odds factor`
13 $s.score \leftarrow \frac{s.score}{s.score + \gamma(1 - s.score)}$ `// Bayesian correction`
14 $\pi \leftarrow$ **lambda x** : $\begin{cases} * & \text{if } \tau_{l,.9} \leq s.score(\mathbf{x}) \leq \tau_{u,.9} \\ ** & \text{if } \tau_{l,.5} \leq s.score(\mathbf{x}) \leq \tau_{u,.5} \\ *** & \text{otherwise} \end{cases}$
`// prediction quality assessment`
15 **return** (s, π)

6.5.2 The scoring model

The core of the LSP system, shown in Algorithm 10, is based on a selective classifier with Bayesian correction of the prediction score. Our approach is a variant of the model-agnostic AUCross approach presented in Algorithm 9. It takes as input a training set $\mathcal{T}$, a base probabilistic LightGBM[7] [110] classifier S, the number of folds K, and the predicted positive rate in the test set p_t.

The training set consists of the instances where the target value is known, i.e., $y = 0$ or $y = 1$. Instances with '*unknown*' target value will be considered later on in Section 6.5.5. Algorithm 10 splits the training set into K stratified folds (line 1) – we set $K = 5$ in our implementation. For each fold k, a base classifier (s_k) is trained on the data of the other $K - 1$ folds (line 4). The

[7]LightGBM adopts a gradient-boosting approach with state-of-the-art performances over tabular datasets, a fast parallel implementation, API's for a few programming languages, management of categorical attributes and missing values without encoding/imputation, and the calculation of TreeSHAP values used by LSP for factual explanations. We have experimented with other base classifiers, including `scikit-learn` models, XGBoost [125] and CatBoost [176]. CatBoost was the only classifier offering features comparable to LightGBM without statistically significant performance differences. We finally chose LightGBM as it was the fastest method.

scores of s_k on the data of the K-th fold (line 6) are scaled through a Bayesian correction (line 7) and then accumulated in $\mathbf{s}$ (line 8). We will detail the Bayesian correction procedure in the next paragraph. The scoring function of Algorithm 10 is obtained by averaging the scores of the K classifiers (line 11), exploiting an ensembling strategy. An alternative would have been to train a final classifier on the whole training set, as in the AUCross approach. However, the model's score averaging was beneficial w.r.t. calibration of the scores, also in comparison with using state-of-the-art calibration techniques.

Bayesian correction for prior data shifts *Objective 6* requires to address prior data shifts. A Bayesian correction of the scores is performed (line 7 and 13) for coping with a change in the proportion of positives from training to test data [163]. This correction takes into input a factor γ, which is the odds ratio of positive rate between the training and test population, i.e., $\gamma \approx (P_S(Y=1)/P_S(Y=0))/(P_T(Y=1)/P_T(Y=0))$ where P_S and P_T are the distributions over the training/source and test/target populations. $P_S(Y=1)$ is estimated on the training set. $P_T(Y=1)$ is estimated by p_t, which is an input parameter. The correction is suitable for coping with events potentially predictable in advance, such as those occurring after a policy change. The correction scales the output scores. Hence, it does not affect ranking metrics such as AUC, but it does affect accuracy scores, such as BLS and BSS, and the calibration of scores and ratings. We used the correction twice. First, the financial crisis of 2008 led to a long period of recession and stagnation in Italy, with an increasing fraction of non-performing loans (NPL). The peak was reached in 2014. Afterwards, the NPL fraction decreased over time for all businesses, including leasing companies. Yearly forecasts of the Bank of Italy[8], and historical data from the BDCR data warehouse (see Figure 6.5 top), anticipated decreasing positive rates p_t. This information has been used in the backtesting of the LSP system to correct the prediction scores, setting $p_t = 0.02$ up to the first quarter of 2020. Second, in the first quarter of 2020, the government counter-measures against the COVID-19 outbreak included a suspension of loan repayments. Consequently, the positive rate dropped substantially (see Figure 6.5 top). As a forward-looking strategy for LSP, we set $p_t = 0.01$ for the suspension period.

6.5.3 Selective classification for self-assessment

Objective 3 requires the uncertainty self-assessment of the risk score.
We adopt a novel approach that relies on selective classification. Because a selective classifier uses the selection function to determine whether a prediction should be accepted or the classifier should abstain, we can exploit it to assess the trustworthiness/uncertainty of a specific prediction.
The core idea of the method is that, given a target coverage c, the selection function[9] splits the feature space into two parts, i.e. the accepted region and

[8] `https://www.abi.it/studi/outlook-crediti-deteriorati/`
[9] We assume that selection functions are built thresholding good confidence functions (see

the rejected region. Now let us consider the case where we have access to two target coverages c_1 and c_2, and assume that $c_1 \leq c_2$. We denote with g_c a selection function g that satisfies the coverage constraint at c, i.e. $\phi(g) \geq c$. In this setting, starting from a selection function g, one can calibrate two different selection functions g_{c_1}, g_{c_2} to meet the coverage constraint for c_1 and c_2 respectively. We then define the monotonic rejection property as follows:

Definition 2. *Given two target coverages c_1, c_2, such that $c_1 \leq c_2$, we say two selection functions g_{c_1}, g_{c_2} satisfy the monotonic rejection property if*

$$\forall \mathbf{x} \in \mathcal{X}, \qquad g_{c_1}(\mathbf{x}) = 1 \implies g_{c_2}(\mathbf{x}) = 1 \tag{6.2}$$

Intuitively, this requires that all the instances accepted by g_{c_1}, which can abstain on more instances, are also accepted by a selection function g_{c_2} that can abstain on fewer instances. In practice, it is not difficult to obtain selection functions that satisfy (6.2). For example, one can use the *coverage-calibration* approach presented in Section 2.1 or the approach considered in Chapter 5.4.2 for AUC, depending on the metrics of interest. When considering the *coverage-calibration* procedure, notice that abstension occurs whenever υ is below the $(1-c_1)\dot{1}00$-th percentile ($\hat{\tau}_{c_1}$) and the $(1-c_2)\dot{1}00$-th ($\hat{\tau}_{c_2}$), respectively. More formally, we define the selection functions as $g_{c_1}(\mathbf{x}) = \mathbb{1}\{\upsilon(\mathbf{x}) \geq \hat{\tau}_{c_1}\}$ and $g_{c_2}(\mathbf{x}) = \mathbb{1}\{\upsilon(\mathbf{x}) \geq \hat{\tau}_{c_2}\}$. Clearly, since $c_1 \leq c_2$, $\hat{\tau}_{c_1} \geq \hat{\tau}_{c_2}$ and, as a result, $g_{c_1}(\mathbf{x}) = 1 \implies g_{c_2}(\mathbf{x}) = 1$.

Alternatively, for the AUC-based method, given two coverages $c_1 \leq c_2$, we recall that we build the rejection area starting at the midpoint between the instances with score τ_{l^*} and τ_{u^*} (line 9, Alg. 8-9). Then, the rejection area $[l', u']$ is centered in *mid* and we build symmetric intervals with width $n \cdot (1-c_1)$ and $n \cdot (1-c_2)$, which are concentric. Since $c_2 \geq c_1$, the selection functions built according to the AUC-based method satisfy the monotonic rejection property.

Let us now assume we have access to two selection functions g_{c_1} and g_{c_2} satisfying the monotonic rejection property. First, we observe that g_{c_1} and g_{c_2} can partition the feature space into three parts:

$$\mathcal{X} = \begin{cases} \mathcal{X}_{<c_1} = \{\mathbf{x} \in \mathcal{X} : g_{c_1}(\mathbf{x}) = 1 \wedge g_{c_2}(\mathbf{X}) = 1\} \\ \mathcal{X}_{c_1<c_2} = \{\mathbf{x} \in \mathcal{X} : g_{c_1}(\mathbf{x}) = 0 \wedge g_{c_2}(\mathbf{x}) = 1\} \\ \mathcal{X}_{>c_2} = \{\mathbf{x} \in \mathcal{X} : g_{c_1}(\mathbf{x}) = 0 \wedge g_{c_2}(\mathbf{x}) = 0\} \end{cases} \tag{6.3}$$

As a consequence, it holds that:

$$\mathbb{E}[l\,(Y, h(\mathbf{X}))\, g_{c_2}(\mathbf{X})] \geq \mathbb{E}[l\,(Y, h(\mathbf{X}))\, g_{c_1}(\mathbf{X})], \tag{6.4}$$

Chapter 2.1).

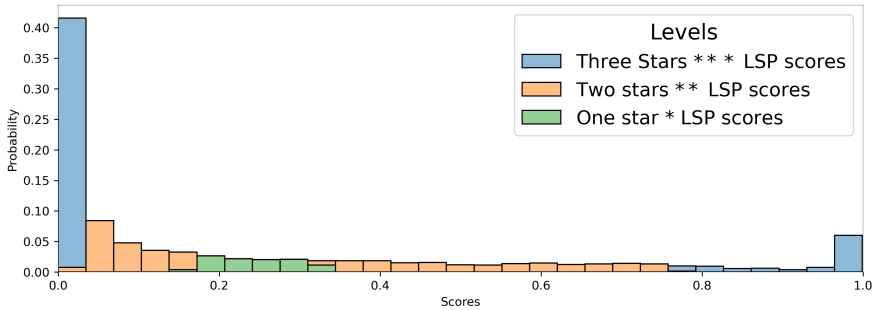


Figure 6.3: Example of LSP quality prediction distribution on simulated data.

since by construction, we have that:

$$\begin{aligned}
&\mathbb{E}[l\left(Y, h(\mathbf{X})\right) g_{c_2}(\mathbf{X})] - \mathbb{E}[l\left(Y, h(\mathbf{X})\right) g_{c_1}(\mathbf{X})] = \\
&\int_{\mathcal{X}\times\mathcal{Y}} l(h(\mathbf{x}), y) g_{c_2}(\mathbf{x}) d\mathcal{P}(\mathbf{x}, y) - \int_{\mathcal{X}\times\mathcal{Y}} l(h(\mathbf{x}), y) g_{c_1}(\mathbf{x}) d\mathcal{P}(\mathbf{x}, y) = \\
&\int_{\mathcal{X}_{<c1}\times\mathcal{Y}} l(h(\mathbf{x}), y) g_{c_2}(\mathbf{x}) d\mathcal{P}(\mathbf{x}, y) + \int_{\mathcal{X}_{c_1<c_2}\times\mathcal{Y}} l(h(\mathbf{x}), y) g_{c_2}(\mathbf{x}) d\mathcal{P}(\mathbf{x}, y) - \\
&\int_{\mathcal{X}_{<c_1}\times\mathcal{Y}} l(h(\mathbf{x}), y) g_{c_1}(\mathbf{x}) d\mathcal{P}(\mathbf{x}, y) = \int_{\mathcal{X}_{c_1<c_2}\times\mathcal{Y}} l(h(\mathbf{x}), y) g_{c_2}(\mathbf{x}) d\mathcal{P}(\mathbf{x}, y) \geq 0
\end{aligned} \tag{6.5}$$

Therefore, the selection functions (satisfying 6.2) induce partitions of $\mathcal{X}$ - those in (6.3) - that are "ranked" in terms of expected risk. We highlight that this approach can be extended also to more than two target coverages, as long as we consider selection functions that satisfy the monotonic rejection property (in a pairwise manner)[10].

More generally, this reasoning allows us to build a self-assessment of uncertainty and achieve *Objective 3*. In particular, resorting to LSP, we consider two target coverages, i.e. $c = .50$ and $c = .90$. We then build the selection functions through the procedure described in Chapter 5.4.2: first,(corrected) predicted scores $\mathbf{s}$ are used to estimate two pair of bounds, namely $[\tau_{l,.9}, \tau_{u,.9}]$ at $c = .90$ and $[\tau_{l,.5}, \tau_{u,.5}]$ at $c = .50$ (lines 8-10, Alg 10). We recall that these bounds are built such that scores outside $[\tau_{l,*}, \tau_{l,*}]$ amount at a specified coverage $c = *$. Moreover, they maximize the AUC of the predictions (see Algorithm 7).

We then build the prediction quality function π (line 14) to determine the quality/uncertainty of the output scores. π ranks them into three levels: three stars for the top 50% of the predictions, i.e. those instances whose score falls outside of $[\tau_{l,.5}, \tau_{u,.5}]$; two stars for the next 40% of cases, i.e. those instances whose score is outside of $[\tau_{l,.9}, \tau_{u,.9}]$, but within $[\tau_{l,.5}, \tau_{u,.5}]$; one star to the remaining 10% (bottom) predictions w.r.t. AUC maximization (i.e. those whose score is between $[\tau_{l,.9}, \tau_{u,.9}]$).

[10]Notice that when considering $M \geq 1$ target coverages, the feature space can be divided into $M + 1$ partitions determined by the selection functions.

Other techniques. Other approaches could be considered for uncertainty estimation. For hard-labeled predictions, one of the most popular frameworks is conformal prediction [86] (see Section 2.5.2). Recall that, given a nominal error level α and a conformal score $v(\mathbf{x}, y)$, the conformal prediction framework builds predictive sets including all the classes in $\mathcal{Y}$ whose score v is above a threshold level $\hat{q}$. In a binary classification task, if both 0 and 1 are included in the prediction set, it means the classifier is uncertain about its prediction, as it can't distinguish between the two classes. However, in our setting, we aim to evaluate uncertainty around the risk score. Extending the conformal prediction framework for such a task is not straightforward. To the best of our knowledge, only the recent work by Stutz et al. [177] tried to build predictive intervals for the score, assuming uncertainty around the Y.
Alternatively, Bayesian methods (see Section 2.5.1) could be used to quantify uncertainty around prediction scores. However, they entail a non-negligible computational burden [178]. Since LSP deals with a large amount of data, we did not opt for these kinds of methods.
To conclude, we highlight that various stakeholders demanded to measure performance in terms of discriminative power. Because our approach to self-assess uncertainty is based on the methodology presented in Chapter 5, it guarantees to correctly identify feature space areas where the discriminative power of the classifier improves.

6.5.4 Explanation methods

LSP provides post-hoc explanations of the scores, both in factual and in counterfactual terms, to address *Objective 4*.

Factual explanations We provide factual explanations in terms of relative feature relevance w.r.t. the model's score. We rely on Shapley values [155], which are computed very fastly by the LightGBM implementation of the exact TreeSHAP algorithm [179]. Shapley values are a coalition game theory concept that aims to allocate the score generated by a coalition of features to each of the features. Shapley values are additive, hence the Shapley values of the LSP model, which averages the scores of k classifiers (line 9 in Algorithm 10) is the mean of the Shapley values of the k classifiers. TreeSHAP returns the Shapley values of the logit transformation of the risk score, i.e., the sum of Shapley values is the log odd of the risk score. By going back to the risk score (a.k.a., the probability space), we lose the additivity property, namely the sum of inverse log odds of Shapley values does not equal to the risk score probability. This may generate confusion in the reading by the user. We then choose to present to the user the *relative contribution* of each feature to the log odd of the risk score, calculated as the L1 normalized Shapley values. Normalized negative (resp., positive) values correspond to negative (resp., positive) contribution in risk, compared to the mean score. They are shown in red (resp., green) in Figure 6.2).

Counterfactual explanations. They consist of up to three examples in which changes in the contract features would result in a lower risk score (or higher creditworthiness score), possibly improving the rating class. Changes should be made only on certain actionable features, such as the initial payment amount, the duration of the contract, and the type of guarantee. Also, changes may be subject to some constraints, such as the range of possible values or the minimum amount of increase/decrease. These and several other requirements on counterfactual generation (minimal number of changes, plausible instances, diversity among counterfactuals, stability of results) are rarely dealt with by a single tool. We designed a novel ensemble of base counterfactual explainers, each one contributing several counterfactual instances. We filter (or modify) those not satisfying requirements not directly implemented by the tools, and from the collection of all remaining counterfactual instances, we finally select three instances (or less, if not possible) to return by following the diversity-maximization approach of [180]. Two types of base counterfactual explainers have been considered – the choice being driven by computational efficiency requirements. The first type is a brute force approach, generating all possible changes in actionable features (restricted to changing max two features, and considering a 10 equal-width binning of continuous features). The first type takes into account similarity and minimality. The second type projects instances into a lower-dimensionality space using PCA. Then it applies a random perturbation at that space, and converts back the perturbed instances to the original space, filtering only instances that result to be counterfactuals (lower risk score by the LSP model). Hence, it takes into account diversity. The counterfactual ensemble explainer runs four base classifiers of type one, three of type two using PCA for explaining 99% of variance, and three of type two using PCA for explaining 75% of variance. A total of 256 counterfactuals are produced, then filtered, and then 3 of them are finally selected. Compared to [180], the elapsed time for generating three counterfactuals lowered from 8 seconds to 0.5 seconds on average.

6.5.5 Other features

In this subsection, we briefly outline some other tasks and topics tackled by the LSP project.

The credit rating model. Credit rating models output an ordered class instead of a probability, with each class associated an apriori estimated probability of default (PD). A standard approach to obtain a rating model from a credit scoring model is binning the $[0, 1]$ range of scores and then determining a PD for each bin. For example, Provenzano et al. [152] consider 9 rating classes and adopt a genetic algorithm for optimizing the choice of the class PDs w.r.t. BLS. LSP adopts a logarithmic binning of the $[0, 1]$ interval to adapt on the unbalanced distribution of defaults. The advantage of our choice is that the binning is data-independent. Hence it is not affected by random/small perturbations of the data distribution. The drawback is that, for the same

reason, we need to closely monitor the binning quality. We rely on both binomial tests and the extended traffic light approach (see Section 2.1) as well as on monitoring the stability of the model [181, 182].

Feature selection. The most relevant feature selection task was concerned with financial indicators [183]. The associated companies do not contribute such indicators, even if they might be available for decision-making in some cases. Together with behavioural indicators, they are highly relevant in credit scoring and related problems [184]. However, they are expensive to obtain, as the fees paid to the credit bureau service are proportional to the number of indicators requested. For this, we selected the many financial indicators available from balance sheets, taking care that feature selection may affect the prediction model performance [183]. We collected more than 20 financial indicators for a sample of companies subject to mandatory submission of balance sheets. We selected six indicators by a variance inflation factor (VIF) stepwise variable elimination [185]. The selected indicators cover business performance (profit), efficiency in revenue generation (return on assets, and total assets turnover), indebtedness (current liabilities/total assets, and leverage), and liquidity (cashflow). Experiments showed a very narrow loss in performance when using the selected set of indicators w.r.t. the entire set. Another set of external features was considered, regarding market conditions (interest rates, inflation, GDP, public debt, etc.) and company demography (new/closed companies per region/type of business). However, these did not improve the performances of the scoring model.

Feature encodings. While LightGBM natively deals with categorical features, to compare with other baselines, we considered some encodings, including one-hot encoding and target encoding [186]. LightGBM with one-hot encoding showed slightly better performance than with categorical features. For this reason and compatibility with other tools (such as external auditing), we one-hot encoded the training set, reaching a final number of ≈ 500 features (≈ 90 without one-hot encoding). No explicit normalization of features is done since decision trees rely on the ranks of values.

Missing values and quality of input. The legal nature of the data contributed to the data warehouse (contracts, payments, etc.) ensures a certain quality of the training set, with a few issues already solved by ETL (repeated entries, outliers, impossible values). Other issues may persist, however, e.g., due to missing contributions for technical reasons, for which the status of a contract could be missing in certain months. Financial indicators may also be missing since only certain company types have the legal obligation to submit balance sheets. Finally, a user query may not include all prospect contract features since only a few are mandatory. LightGBM natively deals with missing values in the training set and the input instance. No imputation method was selected, yet a few were experimented with. This allows LSP to evaluate the

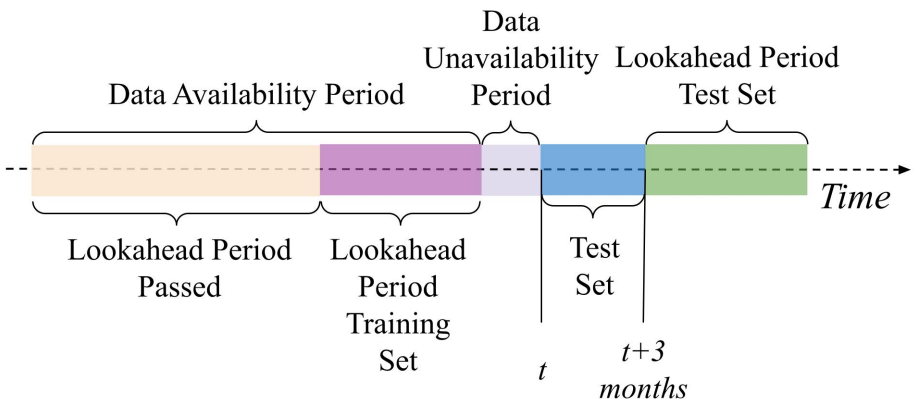


Figure 6.4: Backtesting framework.

input quality through a 1-star to 3-star indicator (see Figure 6.2), which summarizes the impact of missing values in the input instance. A missing value is weighted by the feature importance, calculated by the mean decrease in AUC when testing the base classifier s_i (line 6 in Algorithm 10) with such a feature set to missing (and all features calculated from that one, e.g., ratios, also set to missing).

Self-learning. On top of Algorithm 10, we experimented with a self-training approach close to the one described in [187], exploiting the instances with target feature value '*unknown*'. Such instances regard contracts started since less than 12 months, for which the default event has not been observed. We iteratively re-train LSP by adding the '*unknown*' instances that are predicted with high confidence and assigning to them the predicted label (0 or 1). Such a procedure is computationally expensive. This resulted in a better calibration, but comparable with the impact of the Bayesian correction. Intuitively, the '*unknown*' instances were mostly predicted to be negatives, thus reducing the positive rate in the augmented training set.

6.6 Experimental Evaluation

We evaluate LSP according to the following criteria:

Q1: does LSP (Algorithm 10) outperform existing approaches in predicting default?

Q2: is LSP able to self-assess its own performance?

Q3: does LSP improves compared to single-associates scoring models?]

6.6.1 Experimental Setting

Experimental setup. Given the need for deploying LSP in the real world, we performed an extensive backtesting of the system, intending to simulate

Figure 6.5: Positive rate, training size and time.

the performances of LSP by resampling historical data. We adopt an out-of-time testing procedure with rolling-origin [188], also known as walk-forward, forward-chaining, or nested cross-validation. In fact, standard cross-validation exhibits a positive bias error for time-related data [189].

Figure 6.4 shows the situation pretending to be at a time point t at which a new credit score model ξ_t is available (see Section 2.1 for notation). Such a model will be used to score instances (the test set $\mathcal{T}_{test}$) for a specific test period, which, for LSP, is set to 3 months. After such a period, a new model will be available, and the time point t will roll forward by three months. As shown in Figure 6.1, LSP re-trains a new model every quarter. For an instance $\mathbf{x}$, the true default outcome y is defined w.r.t. a lookahead period of 12 months. To evaluate the model performances, we need data from the lookahead period following $t+3$ months. Moreover, since the BDCR data warehouse is updated monthly, and updates refer to the status of contracts at the previous month, we consider that data is actually available with a delay of two months. Therefore, the test periods in our experiments stop at $t =$ 2020-12. The last test period ranges from 2020-12 to 2021-02, the lookahead period for its third month (2021-02) ends on 2022-02, and the data regarding payments on 2022-02 are available in the data warehouse on 2022-04, which is our last month of raw data. The tests were performed on a machine with 18 cores, 36 threads, equipped with Intel(R) Core (TM) i9-10980XE CPU @ 3.00GHz, OS Ubuntu 20.04.3, programming language Python 3.8.12.

Dataset. The dataset contain 2M leasing contracts with about 500 features (about 90 before one-hot encoding) ranging from 2011-01 (Jan 2011) to 2022-

04 (Apr 2022). Given the temporal structure of the data, in our empirical evaluation, we considered different today dates t, obtaining different splits of the available data, as detailed in the Experimental Setup paragraph. For each temporal split, we build a different training set as follows.
Data on the two months before t should be considered unavailable at the time t. This is shown as *Data Unavailability Period* in Figure 6.4. Those data were not considered in the training set. We then consider data in the availability period. Such instances $\mathbf{x}$ may have the target feature y known (1 for default, 0 for not default), or set to '*unknown*' (see Section 6.5.1). The former instances are included in the training set $\mathcal{T}_{train}$. The latter are included in the dataset for self-learning (see Section 6.5.5). All instances for which the lookahead period has passed (shown as *Lookahead Period Passed* in Figure 6.4) are in the training set, which also includes instances for which the lookahead period has not passed yet, but the default event already occurred.
Figure 6.5 (bottom) shows the size of training sets over rolling time t, and the elapsed training times of LSP and of the LSP with the time-consuming self-learning option (LSP`+Self`). Both training set sizes and elapsed times grow linearly over time. Training set reaches more than 1.5M instances in the last period. The training time of LSP reaches a maximum of about 300 seconds, while the one of LSP`+Self` is an order of magnitude larger.

Baselines. In the experiments, we compare LSP to an ablation version without Bayes correction (LSP`-Cor`), to a version without Bayes correction but with Beta calibration [72] of scores (LSP`-Cor+Beta`), and to a baseline consisting of a single LightGBM classifier with no Bayes correction nor calibration (`LGBM`). Such a baseline approximates the related credit scoring system in Table 6.1 (but with the same hyperparameters as LSP).

Metrics. For each period t, we assess the performance of the different baselines by computing different metrics over the test set for that period. For **Q1**, we focused on AUC, BSS and BIN-ECE. For **Q2**, we compute the coverage and the AUC for the different levels of prediction qualities (1-3 stars). For **Q3**, we compute the difference between the AUC obtained by training LSP on the whole data and the AUC obtained by training LSP only on single associates data.

Hyperparameters. An extensive experimental phase of the project considered the tuning of the hyper-parameters of LSP and its base classifier LightGBM. In Algorithm 10, we fixed $K = 5$ as a trade-off between predictive performance and elapsed running time (see Chapter 5). The quality bounds (lines 9-10) were set by expert guidance. The hyper-parameters of LightGBM were initially tuned at each model re-training by relying on the Optuna framework[11] [190]. During backtesting, however, we observed stability in the range of hyper-parameters chosen, for which we fixed these hyper-parameters for

[11] https://optuna.readthedocs.io/

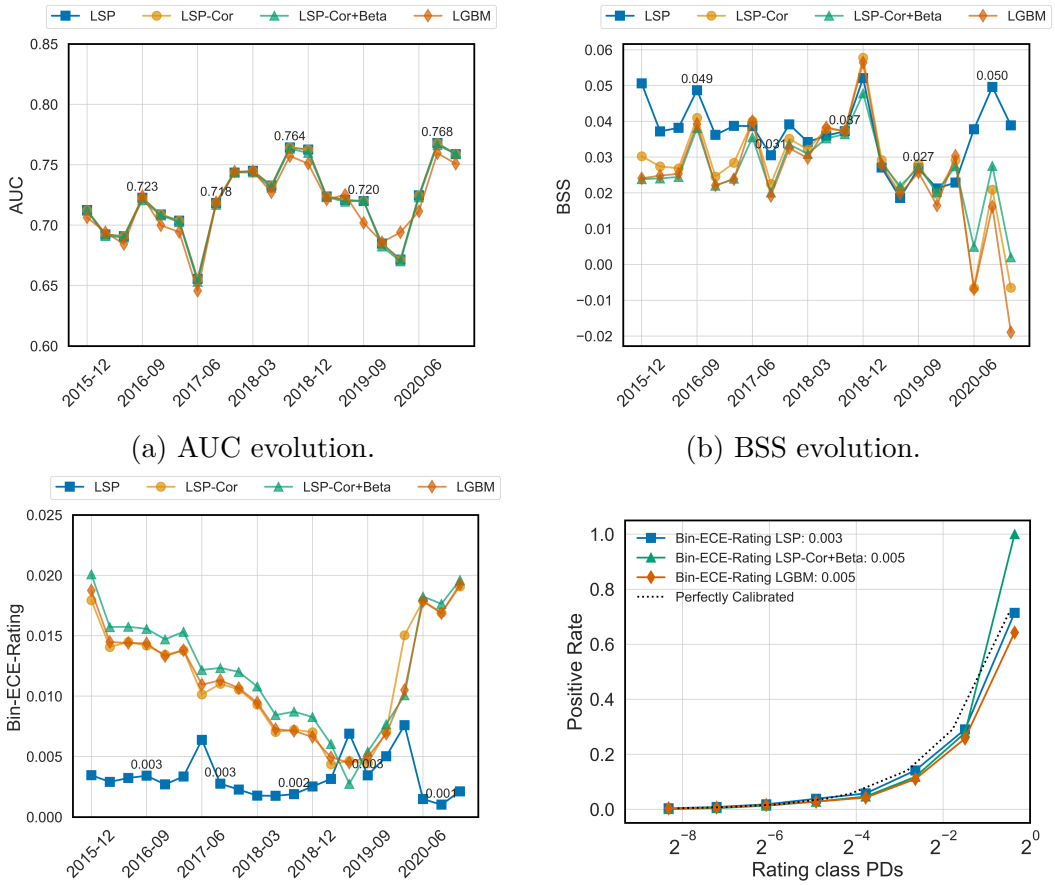
models trained after 2016. This aligns with results from Delgado et al. [144]. Model monitoring activities re-evaluate periodically such a choice.

6.6.2 Experimental Results

Q1: evaluating performance. We start our analysis by evaluating the contribution of the Bayes correction and the contribution of the "ensemble of ensembles" strategy of Algorithm 10. We follow the approach in [112] to test whether differences across classifiers are statistically significant at .01 significance level. Figure 6.6 displays the results for the AUC, BSS and Bin-Ece-Rating metrics. Regarding AUC, LSP performs slightly better than `LGBM` (mean $\pm$ stdev of $.72 \pm .03$ vs $.716 \pm .029$, not statistically significant), and ties with LSP-`Cor`. The latter is expected, as Bayesian correction is a monotonic transformation of the scores, which does not affect ranking metrics. The plots for BSS and Bin-Ece-Rating reveal, instead, the statistically significant advantage of using Bayesian correction over the other approaches – especially for calibration during the COVID-19 period starting in 2020-06. Figure 6.6d contrasts the calibration of LSP, `LGBM`, and LSP-`Cor+Beta` on a single test period t – the one with the lowest Bin-ECE-Rating gap between the models. It shows that LSP achieves calibration without the need for post-processing methods.

Q2: assessing objective O3. Figure 6.7 shows that the novel approach based on selective classification can separate instances with the most accurate predictions (3 stars quality prediction, shown as LSP***), from instances with medium accuracy (2 stars, shown as LSP**), and from instances with low accuracy (1 star, shown as LSP*). All these differences are statistically significant. In particular, for those instances that receive three stars, we observe an average improvement of $\approx 6.5\%$ and $\approx 102\%$ in terms of AUC and BSS, respectively. Alternatively, when we focus on instances that would be rejected at $c = .90$, the metrics deteriorate. AUC and BSS decrease by an average of $\approx 26\%$ and $\approx 89\%$ when considering one-star instances. This suggests that LSP can detect those instances where its own performance is worse. Such a feature is extremely important in the leasing context, as it can signal instances where more attention should be needed by the final user evaluating contracts. To conclude, our findings show also that the strategy of Algorithm 10 provides small violations in terms of coverage, as pointed out by the right plot in Figure 6.7.

Q3: assessing Objective O1. We test the improvement in performance for an associate to use the LSP model trained on all data against using the LSP model trained on its own data only *(Objective O1)*. The performances are evaluated over the subset of the test set, which consists of instances/contracts of that specific associate. Only associates with data in at least 10 test periods are considered. Figure 6.8a shows the mean improvement ($\pm$ stdev) over the rolling test periods. The improvement is negatively correlated with the size



(a) AUC evolution.

(b) BSS evolution.

(c) Bin-ECE-Rating evolution.

(d) Reliability plot.

Figure 6.6: Performance comparison: LSP vs LSP-`Cor` vs LSP-`Cor+Beta` vs `LGBM`.

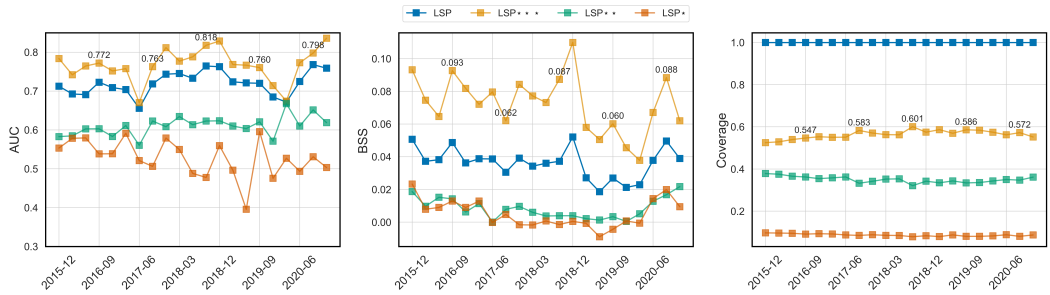
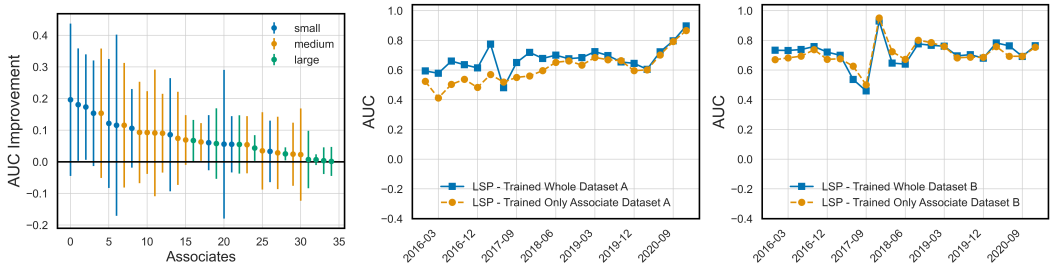


Figure 6.7: Performance over subsets of test data by prediction quality.

of the associate. We categorize small (l<10K contracts in total), medium (10K to 50K) and large (>50K contracts) associates. Such a correlation is not surprising, as large associates operate across the whole country; hence their contracts are representative of the market. Small associates, instead, have a regional scope, and the model built on their instances only is less general. On average, small associates benefit from an absolute improvement in AUC of $\approx 11\%$, which reduces at $\approx 7\%$ for medium and $\approx 3\%$ for large associates. Figures 6.8b-6.8c show the detailed comparison over rolling test periods for two associates. Associate A is a large one, and the plot shows a continuous,



(a) AUC improvement (mean ± stdev). (b) Associate company A. (c) Associate company B.

Figure 6.8: LSP trained on all data w.r.t. LSP trained a single associate data.

yet decreasing, improvement over time. The decrease is due to accumulating a sufficiently large number of instances over time. Associate B is also large, but the improvement is instead fluctuating.

6.7 Conclusions

The LSP project has faced several challenging objectives. *Objective O1* was achieved as a by-product of sharing data from several associate leasing companies. The increased performances w.r.t. using only their own data was marked for small and medium companies, and moderate for large companies. *Objective O2* required a complex workflow to collect in real-time (and to cache) the missing financial indicators from an external credit bureau, as well as a feature selection of those to be collected. *Objective O3* led to a novel usage of selective classification for uncertainty estimation of risk score. Our approach considers multiple coverages to partition the feature space in different areas, which allows the classifier to self-assess where it is more confident. In particular, LSP distinguishes across top 50%, next 40%, and bottom 10% of the predictions w.r.t. AUC maximization. *Objective O4* was achieved by standard Shapley values, and a novel counterfactual ensemble algorithm aimed at producing results very quickly, also in light of *Objective O5* requiring interaction and what-if analysis. The average elapsed time for serving a user query is 1.1 seconds. Finally, regarding *Objective O6*, the prior data shift was tackled by a Bayesian correction. The critical choice of the correction parameter is supported by default forecasts provided by the Bank of Italy.

The LSP system has been designed to be integrated into associate leasing companies' information and decision processes. Input data are collected from the existing BDCR data warehouse. Interaction with the user occurs at the same web interface already offering statistics and analytical reports based on the BDCR. Such a design choice is intended to reduce the information gap and the users' resistance to adopting the system in daily operations. Explanations of the output scores and uncertainty self-assessment aim at establishing trust in the system beyond the backtesting evidence. In particular, our Selective Classification approach allows for detecting those instances where the

AI system is more prone to errors. This warns users to avoid over-relying on predictions with a low prediction quality score, minimizing the risk of (costly) errors.

Finally, we mention two issues open for future work. *First,* data contributed to the BDCR pertain approved applications whose contract has actually started. Rejected applications are not contributed. If they were, we could have approached reject inference methods [191] in a similar way as we dealt with self-learning. An example approach in this line is [192]. *Second,* a pressing urge is to control for bias and unfairness in data-driven AI [2] to prevent discriminatory decisions against protected-by-law social groups. Leasing contracts in the BDCR regard mostly companies, with only 3% of the contracts having a natural person lessees (in our data, they were also indistinguishable from sole proprietorship/individual businesses). While EU anti-discrimination laws apply to natural persons only [139], other biases and forms of ethically unacceptable decisions may indirectly be generated through proxy features that relate social groups to e.g., type of business or region of operation.

Chapter 7

Deep Neural Network Benchmarks for Selective Classification

This chapter provides an extensive empirical analysis of existing selective classifiers that are based on neural network architectures, answering to *Req. 4*. We benchmark existing approaches considering a large collection of datasets, including tabular and image data. The results indicate that there is not a single clear winner among the surveyed baselines, and the best method depends on the users' objectives. A preliminary version of this Chapter was presented at the Verilearn Workshop at ECAI 2023 [193]. An extended version is currently under review.

7.1 Introduction

A wide range of methods has been proposed recently for the selective classification task [14, 15, 16, 18, 19, 20]. The primary emphasis is on implementing selective classification in the context of DNN models.
Unfortunately, we lack insights into the relative merits of existing SC approaches for DNNs because existing empirical evaluations in the literature suffer from several shortcomings. First, they always involve ≤ 10 datasets, and primarily consider only image data. Second, only a handful of approaches (never more than seven) are compared. Third, most studies mainly focus on comparing approaches based on a single metric: their predictive accuracy on the selected examples. However, there are other relevant performance characteristics of SC methods such as whether their coverage constraint holds, whether they disproportionately reject instances from one class, or how they behave on unseen data.
Our goal is to fill this gap by performing the first comprehensive benchmarking of SC methods for DNN architectures. Specifically, our evaluation goes substantially beyond existing studies by:

1. Including 18 SC methods;

2. Evaluating the considered methods on 44 datasets that include both image and tabular data; and

3. Considering five different aspects of SC models' performance.

Our results suggest that the choice of the baseline depends on the performance criterion to be prioritized. In fact, most methods perform with no statistically significant difference across the different tasks. To summarize, the main contributions of this paper are that we:

(i) briefly survey the state-of-the-art methods in SC;

(ii) provide the widest experimental evaluation of SC methods in terms of baselines, datasets and tasks;

(iii) point out the limitations of compared methods, which highlights potential avenues for future research directions; and

(iv) release a public repository with all software code and datasets for reproducing the baseline algorithms and the experiments.[1]

7.2 Baselines

There are multiple ways to devise abstaining classifiers. We restrict our attention to DNN approaches aiming to solve the bounded-abstention problem (2.11). We present and categorize a few baselines according to their definition of the confidence function, extending the work of Feng et al. [20]. We distinguish among three categories of methods: **Learn-to-Abstain**, **Learn-to-Select**, and **Score-based**.

7.2.1 Learn-to-Abstain Methods

Learn-to-Abstain methods tackle the selective classification task by adding a new class label $(m+1)$ representing abstention to the classification problem. While there are no actual instances belonging to this class, these approaches design loss functions to enable the classifier to assign a positive score $s_{m+1}(\mathbf{x})$ to ambiguous instances. This score serves as a confidence function, i.e., $v_h(\mathbf{x}) = 1 - s_{m+1}(\mathbf{x})$ [20]. In Figure 7.1, we provide an example of a canonical Learn-to-Abstain architecture.
The first method to take this approach was **DeepGamblers** [16]. It uses a reward hyperparameter o for the class $m+1$ to set how often the classifier should abstain. Formally, DeepGamblers trains a neural network minimizing the following loss:

$$\mathcal{L}_{\text{DeepGamblers}} = \mathbb{E}_{P(\mathbf{X},Y)}\left[\log(s_y(\mathbf{X}) + \frac{1}{o}s_{m+1}(\mathbf{X}))\right], \tag{7.1}$$

[1]The code is available at `github.com/andrepugni/ESC/`.

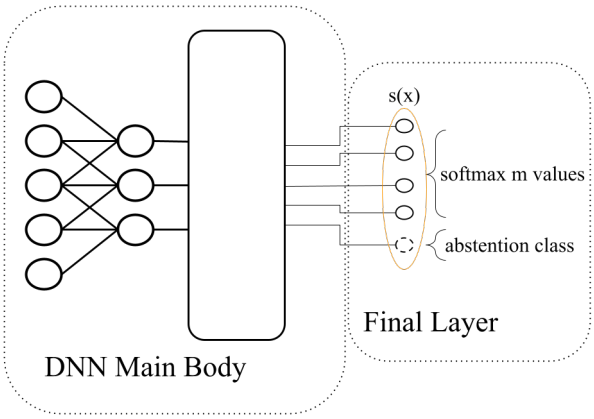


Figure 7.1: A generic Learn-to-Abstain architecture

where $s_y(\mathbf{X})$ and $s_{m+1}(\mathbf{X})$ are the neural network softmax values, respectively, over the true class $Y = y$ and $m + 1$ (abstention). Intuitively, a higher o encourages the network to be confident in its prediction, and a low o makes it less confident and more likely to abstain. However, DeepGamblers does not have any explicit way to guide abstention towards more difficult examples during training, as the reward o remains fixed for the whole training procedure. To overcome this limitation, Self-Adaptive Training (**SAT**) [18] trains the selective classifier through a convex combination of predictions and true labels. This combination is dynamically adapted during the training process to identify those instances that are more difficult to correctly classify and, hence are good candidates for abstention. More precisely, for the first E_s (user-defined) epochs, the training target - $\mathbf{t} \in [0, 1]^m$ - is equal to the one-hot encoded true label vector $\mathbf{y}$. Afterwards, it becomes the convex combination of (probabilistic) predictions and true labels, namely $\mathbf{t} = \gamma\mathbf{t} + (1 - \gamma)\mathbf{s}(\mathbf{x})$, with $\mathbf{s}(\mathbf{x})$ representing the neural network softmax values and γ the weight of the convex combination. The final selective classifier is then optimized by minimizing the loss function:

$$\mathcal{L}_{\text{SAT}} = -\mathbb{E}_{P(\mathbf{X},Y)} \left[\mathbf{t}' \log(\mathbf{s}(\mathbf{X})) + (1 - t_y) \log s_{m+1}(\mathbf{X}) \right], \tag{7.2}$$

where t_y is the value of vector $\mathbf{t}$ corresponding to the index of true value y and $s_{m+1}(\mathbf{x})$ represents the softmax value for the abstention class.[2] Both DeepGamblers and SAT add an extra softmax value to the neural network output to identify difficult-to-predict instances. However, Feng et al. [20] argue that incorporating this extra class in the training loss could result in overfitting on examples that are easier to classify. To mitigate this, **SAT+EM** [20] adds

[2]For instance, if $y = 1$ and $m = 2$, then $\mathbf{t}' = [0, 1]$ and $t_y = 1$ when epoch is below E_s. Intuitively, the first term is the cross-entropy loss between the classifier and the adaptive training target, which allows the learning of a good multi-class classifier. The second term serves as a confidence function, identifying uncertain samples in the dataset. The balance between these terms is controlled by the value of t_y, which determines whether the classifier learns to abstain or make accurate predictions.

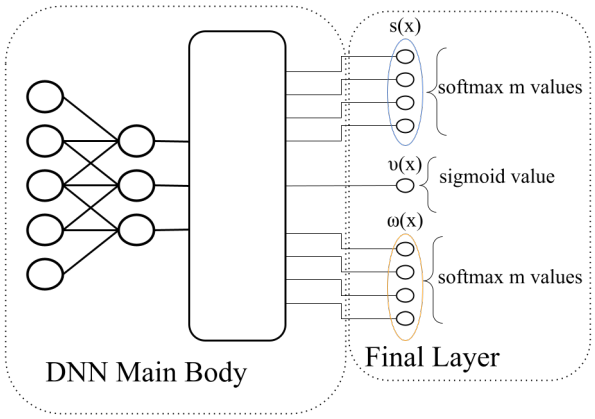


Figure 7.2: An example of SelNet, a Learn-to-Select architecture.

an average entropy term $\mathcal{E}(\mathbf{s}(\mathbf{x}))$ to SAT's loss :

$$\mathcal{L}_{\text{SAT+EM}} = \mathcal{L}_{\text{SAT}} + \beta\mathcal{E}(\mathbf{s}(\mathbf{X})) \tag{7.3}$$

where $\mathbf{s}(\mathbf{X})$ represents the neural network of m softmax values, and β is a hyperparameter that measures the impact of the entropy term. All the learn-to-abstain methods are calibrated for the target coverage c using a calibration set (as discussed in Section 2.1).

7.2.2 Learn-to-Select Methods

Like Learn-to-Abstain methods, Learn-to-Select methods simultaneously learn the classifier and its specific confidence function. However, in this setting, the confidence function does not rely on an additional abstention class but aims at achieving a specific target coverage c. This procedure ensures that the classifier's parameters are optimized to correctly predict instances less likely to be rejected.

The main architecture belonging to this class is SelectiveNet (**SelNet**) [15]. Given a target coverage c, SelNet jointly trains the final classifier and the confidence function to maximize the performance over the $100 \cdot c\%$ most confident instances. SelNet's architecture has four main components, each with a different purpose, as depicted in Figure 7.2: the main body, the predictive head s, the selective head υ, and the auxiliary head ω. The main body consists of deep layers shared by all three heads: any deep-learning architecture can be used in this part (e.g., convolutional layers, linear layers, recurrent layers etc.). The predictive head $s(\mathbf{x})$, consisting of a final linear layer with softmax, is used to make the classifier prediction. The selective head $\upsilon(\mathbf{x})$ outputs a confidence function using a linear layer with a final sigmoid activation. The auxiliary head $\omega(\mathbf{x})$ replicates the structure of the predictive head and mitigates the risk of overfitting on the accepted instances. Given the target coverage c, SelNet is

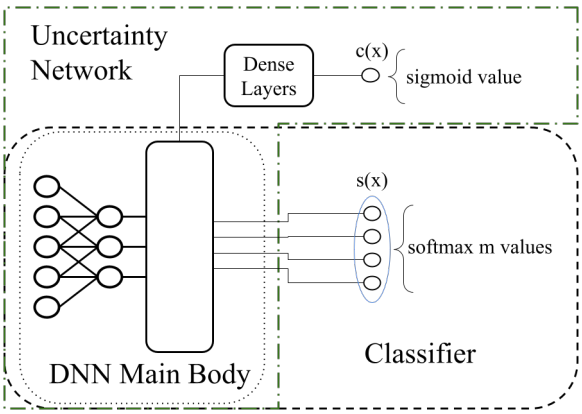


Figure 7.3: An example of ConfidNet, a score-based architecture.

trained using the following loss function:

$$\mathcal{L}_{\text{SelNet}} = \alpha \frac{\mathbb{E}_{P(\mathbf{X},Y)}\left[l(s(\mathbf{X}),Y)\upsilon(\mathbf{X})\right]}{\phi(k)} + \lambda(\max(0, c-\phi(k)))^2 + (1-\alpha)\mathbb{E}_{P(\mathbf{X},Y)}\left[l(\omega(\mathbf{X}),Y)\right] \quad (7.4)$$

where $l(s(\mathbf{X}),Y)$ is the cross-entropy loss for the predictive head $s(\mathbf{X})$; $\phi(\upsilon)$ is the coverage obtained by selective head υ; $l(\omega(\mathbf{X}),Y)$ is the cross entropy loss for auxiliary head prediction $\omega(\mathbf{X})$; α is a hyperparameter to control the relative importance between the losses for the predictive and the auxiliary head; and λ is a penalization term for coverage violations.
For the sake of completeness, following the same reasoning as for SAT+EM [20], we include **SelNet+EM** in the comparison. This approach adapts the SelNet objective function to contain an additional entropy term.

7.2.3 Score-based Methods

Score-based methods compute and set a threshold on a confidence function - as formalized in Eq. 2.7 - that is based on the classifier's output. Conceptually, this means that predictions are only made for the test examples for which the model is most confident. Because this can be viewed as a post-hoc approach, this confers the advantage of being applicable to already trained models.
The most popular technique is the Softmax Response **SmR** [14], which defines the confidence function as the maximum value of a final softmax layer, i.e., $\upsilon_{\text{SmR}}(\mathbf{x}) = \max_{y\in\mathcal{Y}} s_y(\mathbf{x})$. Given a coverage c, SmR sets the selection threshold τ using the calibration procedure explained in Section 2.1.
Since the SmR principle is very general, it can be applied to any method that provides scores for the classes [24]. For example, Feng et al. [20] propose to improve learn-to-abstain and learn-to-select methods by replacing their confidence function with the SmR confidence. In particular, three novel methods are presented, i.e., **SAT+SmR**, **SAT+EM+SmR**, **SelNet+SmR**, which are trained using $\mathcal{L}_{\text{SAT}}$, $\mathcal{L}_{\text{SAT+EM}}$ and $\mathcal{L}_{\text{SelNet}}$ respectively. For the sake of

completeness, we include also **SelNet+EM+SmR** in the comparison, i.e. a network trained with the SelNet+EM's loss and using the SmR selection strategy.
Another score-based popular option is using ensembles of neural networks. For example, Lakshminarayanan et al. [75] train multiple networks with different initialization and build a selective classifier by computing the entropy of the (multiple) network outputs (**Ensemble**). The intuition is more disagreement among the outputs vary indicates that ensemble is uncertain about its prediction, and hence rejection is appropriate. However, despite the advantages of using ensembles in terms of performance, relating a dispersion measure to the correctness of predictions is not straightforward. Hence, the authors also propose using the average softmax response (i.e., $\upsilon_{\text{Ensemble}}(\mathbf{x}) = {}^1\!/_J \sum_{j=1}^{J} \upsilon_{\text{SmR},j}(\mathbf{x})$, where J is the number of networks in the ensemble) as a confidence measure. We will refer to this baseline as **EnsembleSR**. A theoretical analysis of the advantages of using EnsembleSR can be found in Ding et al. [23].
The main concern with using $\upsilon_{\text{SmR}}(\mathbf{x})$ as a confidence measure is that may provide high values both for mistakes and correct predictions, making them indistinguishable. On the other hand, when the model misclassifies an example, the score $s_{y_{true}}(\mathbf{x})$ associated with the true class probability $P(Y = y_{true}|\mathbf{X} = \mathbf{x})$ should be low, making it a viable option to perform selective classification. However, one cannot access true labels at test time, making it impossible to use $s_{y_{true}}(\mathbf{x})$ directly. Corbière et al. [17] address this concern by estimating $s_{y_{true}}(\mathbf{x})$ with a two-step procedure called **ConfidNet** (as depicted in Figure 7.3). First, they estimate $s_{y_{true}}(\mathbf{x})$ by training a neural network classifier. Next, they build a second (uncertainty) network on top of the classifier: the main body is kept unchanged, while the final part of the original classifier is replaced with a series of dense layers. This uncertainty network is then trained considering the following loss function:

$$\mathcal{L}_{\text{ConfidNet}} = \mathbb{E}_{P(\mathbf{X},Y)}[(\mathsf{c}(\mathbf{X}) - s_{y_{true}}(\mathbf{X}))^2] \tag{7.5}$$

with $\mathsf{c}(\mathbf{X})$ referring to the final output of the uncertainty network. Intuitively, $\mathsf{c}(\mathbf{x})$ should mimic $s_{y_{true}}(\mathbf{x})$ and can be used as a confidence function: the higher $\mathsf{c}(\mathbf{x})$, the higher the chance the classifier is right on instance $\mathbf{x}$.
Franc et al. [24] also use a classifier and an uncertainty estimator. They propose two different approaches, named **Reg** and **Sele**. Both of them learn the classifier on half of the training data and use the other half to directly estimate where the classifier is more likely to make mistakes. In particular, these two methods focus on learning an uncertainty score u, which mirrors the confidence function υ: the higher u is, the higher the likelihood of making mistakes (thus, abstention is preferable in the latter). Neither Sele nor Reg are tied to specific neural network architectures, i.e., they are model-agnostic and can be adapted to other learning models. Reg poses the problem of learning the uncertainty score as a regression problem, where given a set of hypotheses $\mathcal{U}$, the uncertainty score $u \in \mathcal{U}$ minimizes the following:

$$\mathcal{L}_{\text{Reg}} = \mathbb{E}_{P(\mathbf{X},Y)}[(l(h(\mathbf{X}, Y) - u(\mathbf{X}))^2] \tag{7.6}$$

Intuitively, the higher the value of u, the higher the loss. Hence, abstention should be preferred. On the other hand, given a hypothesis space $\mathcal{U}$, SELE considers a surrogate loss of the risk coverage curve, i.e.,

$$\mathcal{L}_{\text{SELE}} = \mathbb{E}_{(\mathbf{X}_1,Y_1),(\mathbf{X}_2,Y_2)\sim P(X,Y)}[l(h(\mathbf{X}_1,Y_1))\log(1+\exp(u(\mathbf{X}_2)-u(\mathbf{X}_1)))] \quad (7.7)$$

and then learns the uncertainty score $u \in \mathcal{U}$ by minimizing $\mathcal{L}_{\text{SELE}}$.
The approaches presented so far require a held-out calibration dataset. Unfortunately, for problems where only little data is available, reducing the amount of training data may deteriorate the classifier's performance. Moreover, splitting the data into a dataset for training and a dataset for calibration may introduce randomness effects on both the classifier and the selection function. The SCROSS algorithm, presented in Chapter 3, tackles these concerns. Moreover, for imbalanced scenarios, one can consider the AUC-based methods, i.e. PLUGINAUC and AUCROSS, presented in Chapter 5.

7.3 Experimental Evaluation

In this section, we present our empirical evaluation of the analyzed baselines.

7.3.1 Research Questions

We aim to evaluate the relative strength of the baselines introduced in Section 7.2 with respect to the following research questions:

Q1: Are there significant differences across baselines and scenarios regarding selective error rate?

Q2: Are there significant differences across baselines and scenarios regarding violations of the expected target coverage?

Q3: How are the methods' rejection rates distributed among the classes?

Q4: How do the methods behave when flipping the learning task to maximise the coverage under constraints on the error rate?

Q5: How do the methods react to out-of-distribution test examples?

We differentiate from previous works in several respects:

- Regarding **Q1**, our study goes beyond existing ones in two important ways. First, prior evaluations involving SC methods were performed using less than seven baselines and less than ten datasets whereas we consider 18 methods and 44 datasets. Second, prior benchmarks largely focused on image data whereas our benchmark also include tabular data.

- Concerning **Q2**, only a few works investigate coverage violations, i.e., Geifman and El-Yaniv [15] and the works presented in Chapter 3 and Chapter5. As in **Q1**, this was done on a much smaller scale: for example,

Geifman and El-Yaniv [15] considered only a single image dataset, while in the previous chapters we limited our analysis to eight and nine binary datasets respectively;

- Only our empirical evaluation in Chapter 5 addresses **Q3** and highlights that the rejection rate is biased against the minority class. However, we considered nine binary datasets and only six baselines;
- We are the first to empirically evaluate **Q4** and assess performances when switching from minimizing selective risk to maximizing coverage on a large and diverse set of data and settings;
- We are the first to evaluate **Q5** and evaluate how selective classification methods perform when dealing with shifts in the feature space.

7.3.2 Experimental Setting

Datasets and Baselines. We run experiments on 44 benchmark datasets from real-life scenarios, such as finance and healthcare [194]. Among these, 20 are image data and 24 are tabular data. Moreover, 13 of these datasets were previously used in testing (at least one) the baselines in their original paper. Details are provided in Tables C.1-C.2 of the Appendix C.1. We compare a total of 18 baseline methods (presented in Section 7.2) representing the state-of-the-art SC methods: DeepGamblers, SAT, SAT+EM (*learn-to-abstain*); SelNet, SelNet+EM (*learn-to-select*); SmR, SAT+SmR, SAT+EM+SmR, SelNet+SmR, SelNet+EM+SmR, Ensemble, EnsembleSR, ConfidNet, Reg, Sele, SCross, PluginAUC, AUCross (*score-based*).

Hyperparameters. The baselines share the same neural-network architecture. For image data, we use either a Resnet34 architecture [195] or the one specified in the original paper. For tabular data, since neural networks are not state-of-the-art methods, we use the architectures proposed by [104, 103], which revised DNN models for tabular data. Overall, we consider two sets of hyperparameters: network-specific (e.g., hidden layers, learning rate), and loss-specific (e.g., β for SAT+EM). All networks are trained for 300 epochs. We optimize the hyperparameters using `optuna` [190], a framework for multi-objective Bayesian optimization, with the following inputs: coverage violation and cross-entropy loss as target metrics, `BoTorch` as sampler [196], 10 initial independent trials out of 20 total trials. Among the 20 trials, we select the configuration that (1) has the highest accuracy on the validation set and (2) reaches the target coverage (± 0.05). Moreover, some baselines require the target coverage c to be known at training time (e.g., SelNet). For the sake of reducing the computational cost[3], we optimize their hyperparameters using only three values $c \in \{.99, .85, .70\}$ and fixing the best-performing architecture for all target coverages. Moreover, SCross, AUCross, Ensemble, EnsembleSR and

[3]Tuning the networks is computationally expensive, requiring more than 15 days on some large datasets, such as `food101`.

PluginAUC use the same optimal hyperparameters found for SmR as they share the same training loss. Similarly, SelNet+SmR, SelNet+EM+SmR, SAT+SmR and SAT+EM+SmR employ the same optimal configuration as, respectively, SelNet, SelNet+EM, SAT and SAT+EM. We detail the parameter choices in the Appendix C.1.2.

Experimental setup. For each combination of datasets and baselines, we run the following experiment: (i) we randomly split the available data into training, calibration, validation, and test sets using the 60/10/10/20% of the original data, (ii) we consider the following seven target coverages $c \in \{.70, .75, .80, .85, .90, .95, .99\}$, (iii) we tune the baseline's hyperparameters using training, calibration, and validation sets as described in the previous paragraph, (iv) we use such optimal hyperparameters to train the baseline on the training set and calibrate the confidence function on the calibration set, (v) we draw 100 bootstraps datasets from the test set (see [111]) with the same size at the test set, and, finally, (vi) we compute the selective error rate[4] $\widehat{Err}(h, g \mid \mathcal{T}_{test})$, the empirical coverage $\hat{\phi}(g \mid \mathcal{T}_{test})$, and, for binary datasets, the class distribution over the accepted instances for each of the 100 bootstrapped datasets $\mathcal{T}_{test}$. For each evaluation metric, we compute its mean and standard deviation over the 100 bootstrap runs. In reporting results, we distinguish between binary and multi-class (i.e., > 2 classes) problems because PluginAUC and AUCross are specific for binary classification.

Hardware. Regarding computational resources, we split the workload over three machines: (1) a 25 nodes cluster equipped with 2×16-core @ 2.7 GHz (3.3 GHz Turbo) POWER9 Processor and 4 NVIDIA Tesla V100 each, OS RedHatEnterprise Linux release 8.4; (2) a 96 cores machine with Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz and two NVIDIA RTX A6000, OS Ubuntu 20.04.4; (3) a 128 cores machine with AMD EPYC 7502 32-Core Processor and four NVIDIA RTX A5000, OS Ubuntu 20.04.6.

Limitations. For the sake of a fair comparison, our study focuses on neural network classifiers, as some of the methods assume a deep learning architecture for the classifier. We acknowledge that a few studies, e.g. Gorishniy et al. [104], Grinsztajn et al. [103], point out that for tabular datasets the usage of tree-based models is the current state of the art. In this sense, model-agnostic methods could benefit from the usage of other base classifiers, as shown in Chapter 3 and Chapter 5.
Moreover, our bootstrap procedure quantifies variability only in the test set. According to several works, such as Kohavi [197], the best resampling method is stratified k-fold cross-validation with $K = 10$. Unfortunately, these strategies are not computationally feasible when employing large neural networks as in

[4]The selective error rate is the empirical risk (2.9) w.r.t. the 0-1 loss. Almost all of the baselines are optimized for such a metric, except PluginAUC and AUCross that are designed for increasing AUC.

our study. Hence, we had to opt for a single train-test-split and bootstrap only the test set, as done for instance by Rajkomar et al. [111].
Finally, our study does not report on the running times of the baselines, since, due to load balancing issues, we had to distribute the experiments over several machines with different hardware settings. This made it impossible to compare the running times of runs over different machines. However, we point out that the running times are proportional to the number of training tasks required by each method. E.g., Ensemble requires training ten neural networks (see Appendix C.1.2), leading to a running time of about ten times the one of PluginAUC, which requires training a single neural network.

7.3.3 Experimental Results

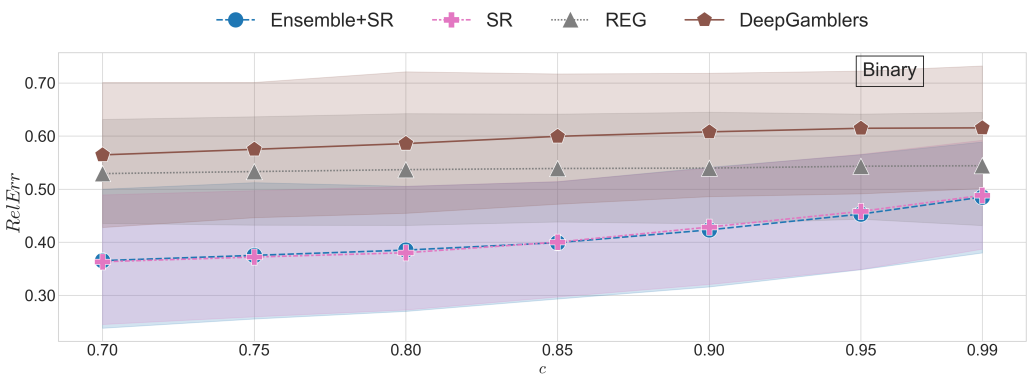
We report here the main experimental results w.r.t. the research questions Q1–Q5. Additional results are reported in the Appendix C.2.

Q1. Comparing the error rates. We introduce a normalized version of the selective error rate, called *relative error* rate:

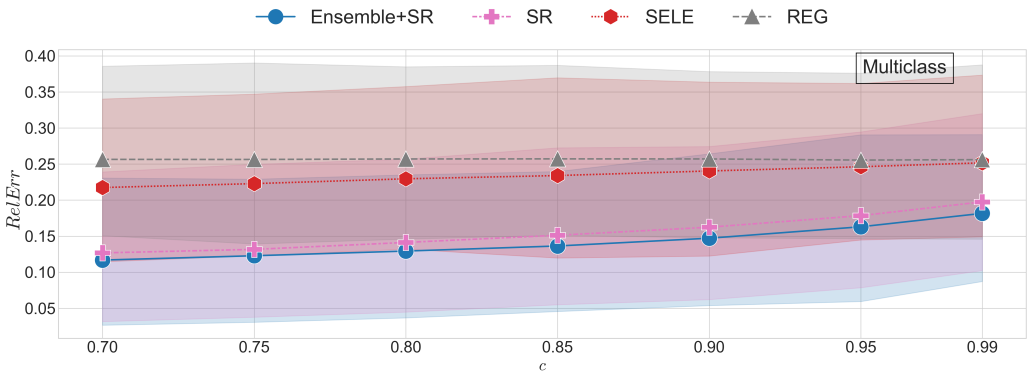
$$RelErr(h, g \mid \mathcal{T}_{test}) = \frac{\widehat{Err}(h, g \mid \mathcal{T}_{test})}{\widehat{Err}(h_{maj}, g \mid \mathcal{T}_{test})}, \tag{7.8}$$

where $\widehat{Err}(h, g \mid \mathcal{T}_{test})$ is the selective error rate considering the classifier h and $\widehat{Err}(h_{maj}, g \mid \mathcal{T}_{test})$ is the selective error rate obtained by always predicting the majority class in the training set. This normalization accounts for variability in task prediction difficulty. Intuitively, the closer the relative error rate to 0 the better. Values close to 1 denote selective error rates similar to the ones of a majority classifier.
Figure 7.4 reports the mean relative error rates for the top two and the worst two[5] baselines. We limit the number of reported baselines for clarity. Tables with detailed results at the dataset level are reported in the Appendix C.2.3.
For binary data, the best-performing methods are EnsembleSR and SmR. EnsembleSR's relative error rate is $\approx .485$ at $c = .99$, decreasing to $\approx .365$ at $c = .70$. SmR ranges from $\approx .488$ at $c = .99$ to $\approx .363$ at $c = .70$. The worst baselines are DeepGamblers and Reg, with relative error rates of $\approx .615$ and .544 at $c = .99$ respectively, up to $\approx .564$ and $\approx .529$ at $c = .70$.
Also for multiclass data, EnsembleSR and SmR achieve the best results. The relative error rate EnsembleSR ranges from $\approx .182$ at $c = .99$ to $\approx .117$ at $c = .70$, while SmR starts from $\approx ..197$ at $c = .99$, and decreases down to $\approx .127$ for $c = .70$. Sele and Reg are the worst methods. The former passes from $\approx .252$ at $c = .99$ to $\approx .217$ at $c = .70$. The latter achieves $\approx .256$ at $c = .99$ and $c = .70$, with no improvement for small target coverages.
Next, we check the statistical significance of these results. For every target coverage and bootstrapped dataset, we rank the methods from 1 (the best) to

[5] We rank baselines based on the mean value of relative error rate over all the target coverages.



(a) Binary datasets.



(b) Multiclass datasets.

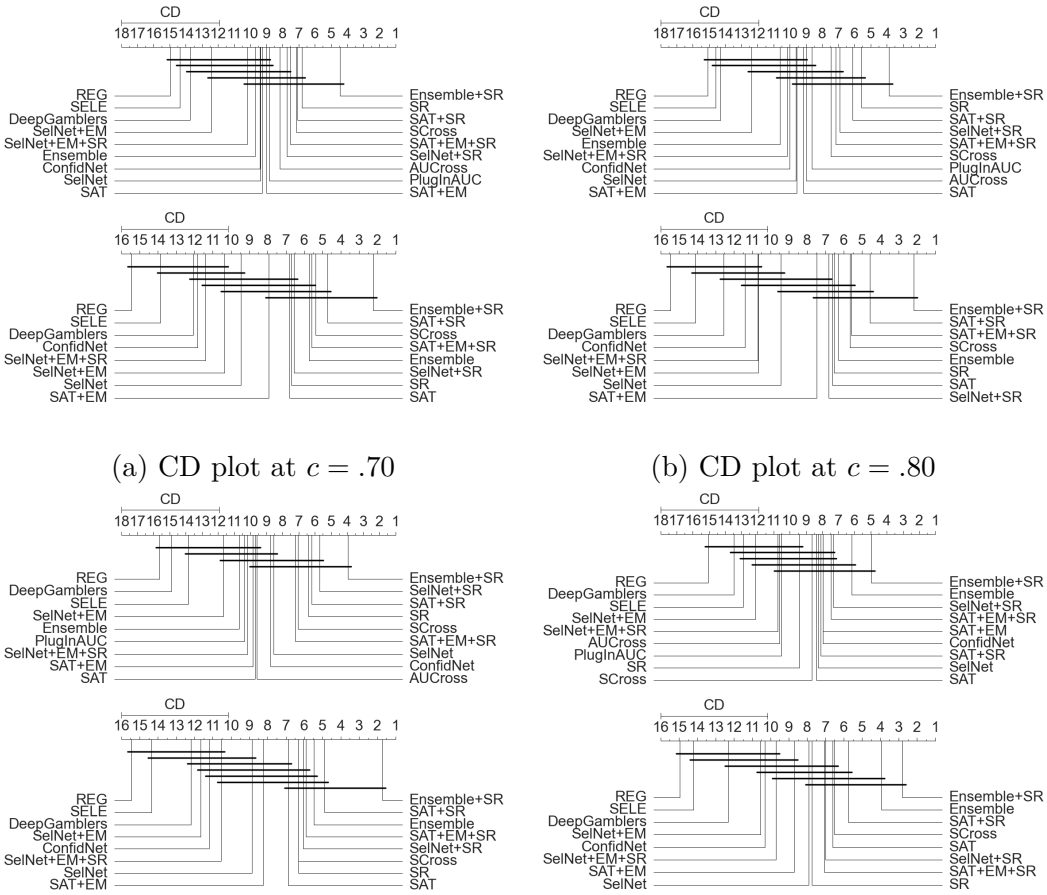
Figure 7.4: Q1: *RelErr* as a function of target coverage c for two best and worst approaches on (a) binary and (b) multiclass problems. On each subplot, only the two best and worst approaches are shown for readability.

18 (the worst), obtaining a mean rank for every dataset. These rankings are then used in Friedman's test of equality of means and post-hoc Nemenyi tests, following the steps in Demsar [112]. The Critical Difference (CD) plots in Figure 7.5 show that there is no clear winner regardless of the coverage and of the binary/multiclass classification task. The group of not-statistically-different top methods contains between 8 and 14 baselines. However, EnsembleSR is always the top ranked baseline, which makes it a good choice in general.

Q2. Comparing the empirical coverages. The constraint on the target coverage c in (2.11) is essential in many scenarios. Nevertheless, most papers do not sufficiently investigate the actual coverage achieved by the baselines. We assess how much the empirical coverage deviates from the target coverage c on the bootstrap dataset $\mathcal{T}_{test}$. To account for small coverage violations, we introduce a user-defined tolerance ε, and define the *ε-coverage violation*:

$$CovViol_{\varepsilon}(g \mid \mathcal{T}_{test}) = \min(0,\ \hat{\phi}(g \mid \mathcal{T}_{test}) - c + \varepsilon), \tag{7.9}$$

where $\hat{\phi}(g \mid \mathcal{T}_{test})$ is the empirical coverage on $\mathcal{T}_{test}$. Intuitively, $CovViol_{\varepsilon}$ is zero when the empirical coverage is greater or equal than the target coverage



(a) CD plot at $c = .70$ (b) CD plot at $c = .80$

(c) CD plot at $c = .90$ (d) CD plot at $c = .99$

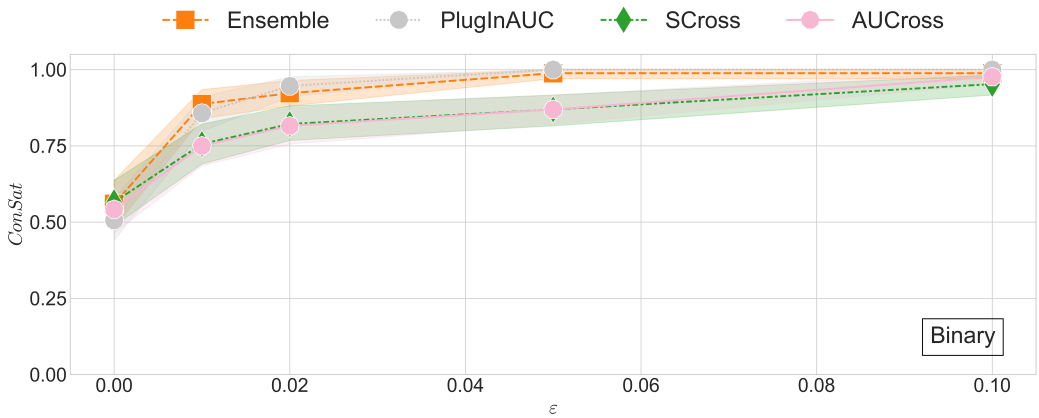
Figure 7.5: Q1: CD plots of relative error rate *RelErr* for different target coverages. Top plots for binary datasets. Bottom plots for multiclass datasets.

minus the tolerance; and it is greater than zero when the empirical coverage is smaller than $c - \varepsilon$. By looking at different tolerances ε, one can evaluate how the baselines perform w.r.t. small, medium or large coverage violations. We define the satisfaction of the constraint as:

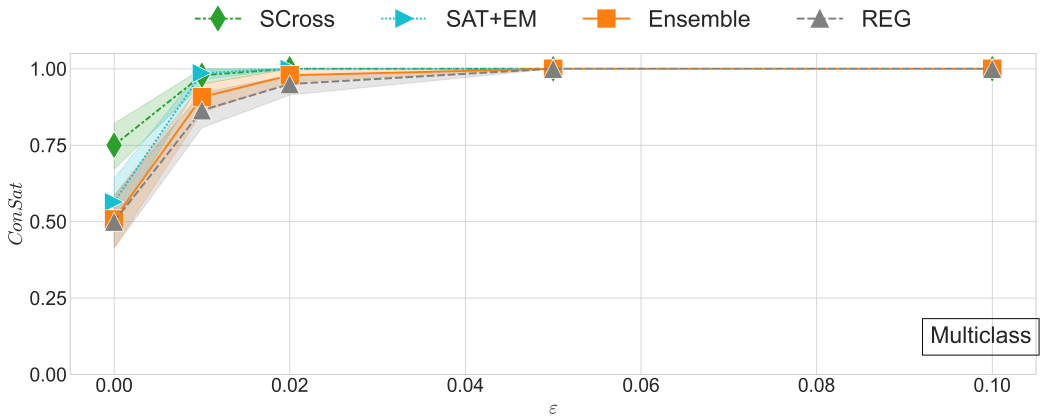
$$ConSat(\varepsilon) = \mathbb{1}(CovViol_{\varepsilon} = 0)$$

and report in Figure 7.6 the mean and standard deviation of *ConSat* for ε in $\{0, .01, .02, .05, .10\}$. As for Figure 7.4, we limit the number of baselines to the two best and worst ones w.r.t. *ConSat*.

As one would expect, the overall performances gradually improve for all baselines when increasing ε, and the gap among the baselines decreases. For binary data, the best methods are Ensemble and PluginAUC, and the worst methods are AUCross and SCross. When considering no violation allowed, i.e. $\varepsilon = 0$, the baselines satisfy the constraint between $\approx 39.9\%$ (ConfidNet) and $\approx 56.5\%$ (SCross) of the times. For $\varepsilon = .01$ Ensemble has the highest value of *ConSat* ($\approx .887$); for $\varepsilon = .02$ SAT is the best method (≈ 0.976); for $\varepsilon = .05$ both PluginAUC and SAT satisfy the constraint all the times.



(a) Binary datasets.



(b) Multiclass datasets.

Figure 7.6: Q2: *ConSat* as a function of tolerance ε for two best and worst approaches on (a) binary and (b) multiclass problems. On each subplot, only the two best and worst approaches are shown for readability.

For multiclass data, the top performers are SCross and SAT+EM, while the worst methods are Reg and DeepGamblers. At $\varepsilon = 0$, SCross has no coverage violations $\approx 75\%$ of the times, which is 25 percentage points more than the worst performing methods (Sele and Reg). Interestingly, already at $\varepsilon = .02$, four methods (i.e. SCross, SAT+EM, SmR, SAT+SmR) always reach zero violations. For $\varepsilon = .05$, only ConfidNet and SelNet+SmR do not reach zero violations. In summary, coverage violations are generally limited, and noticeable differences among the baselines only occur at very small tolerances.

Q3. Rejection rate over classes. In Chapter 5, we observed that, in imbalanced classification tasks, selective classification methods reject proportionally more instances from the minority class. In this paragraph, we analyze this behavior on 7 binary class datasets of our collection with a minority class prior estimate $p \leq 0.25$ [198]. Detailed results for the other binary datasets are

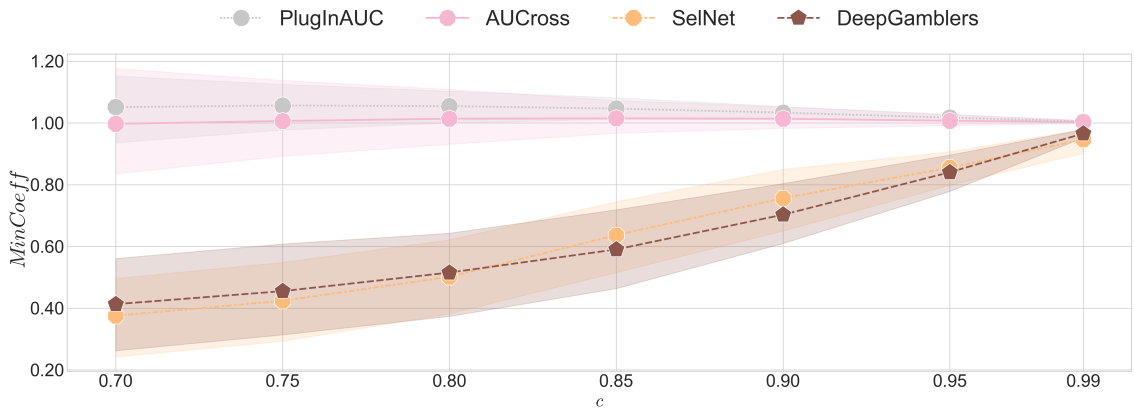


Figure 7.7: Q3: *MinCoeff* as a function of target coverage c for two best and worst approaches. Only the two best and worst approaches are shown for readability.

reported in the Appendix C.2.3. First, let us introduce the *minority coefficient*:

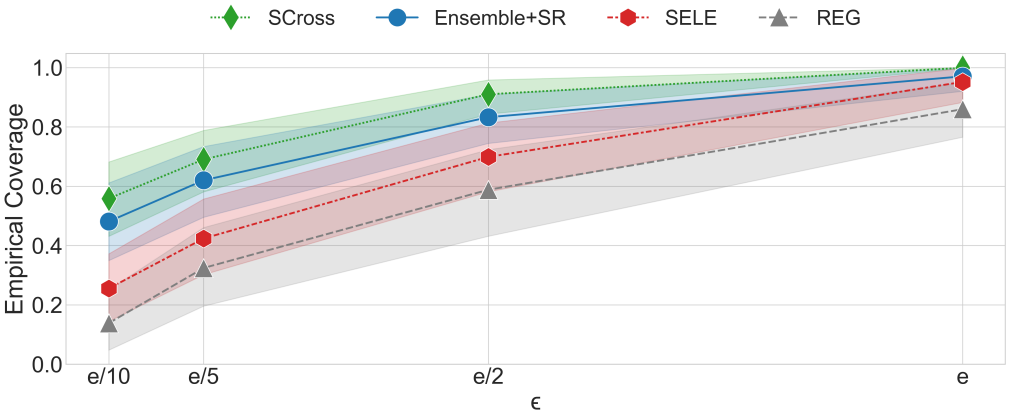
$$MinCoeff = \frac{p_a}{p}, \tag{7.10}$$

defined as the ratio of the minority class proportion p_a in the accepted instances over the minority class prior p. Ideally, the minority coefficient should be ≈ 1. Lower values indicate that the selective function introduces bias against the minority class.

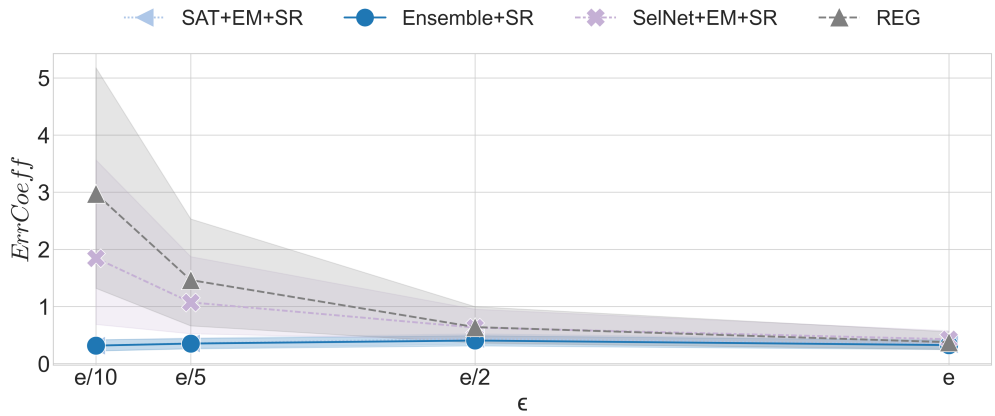
Figure 7.7 shows the mean minority coefficient for the best two and worst two baselines at the variation of the target coverage c. The best methods are AUCross and PluginAUC. Their minority coefficient is ≈ 1.00 and ≈ 1.01 respectively at $c = .99$, and it remains steady for lower coverages. At $c = .70$, PluginAUC reaches a mean *MinCoeff* ≈ 1.05, and AUCross achieves *MinCoeff* ≈ 0.997. For all the other baselines, there is a clear trend: the smaller the target coverage, the smaller the minority coefficient. For 11 out of 18 baselines, *MinCoeff* drops below .50 at $c = .70$. The worst methods are SelNet and DeepGamblers. For the former, the mean *MinCoeff* ranges from $\approx .946$ at $c = .99$ to $\approx .375$ at $c = .70$. For the latter, the mean *MinCoeff* ranges from $\approx .966$ at $c = .99$ to $\approx .413$ for $c = .70$.

These results support our findings in Chapter 5, highlighting that the current approaches to SC, with the exception of AUCross and PluginAUC, do not take into account the issue of class balancing in the selected instances.

Q4. Flipping the learning task to maximize the model coverage under error constraints. The vast majority of methods focus on the bounded-abstention model of Problem 2. To the best of our knowledge, the only method explicitly addressing the bounded-improvement model of Problem 1 is due to Gangrade et al. [19], whose code has not been fully released. However, tackling the bounded-improvement model is useful in some application scenarios. For example, consider a bank using a complex and expensive-to-train ML classifier to predict the default of borrowers. SC is appealing because, due to limited



(a) W.r.t. the empirical coverage $\hat{\phi}$.



(b) W.r.t. the error ratio *ErrCoeff*.

Figure 7.8: Q4: SGR performance as a function of target error rate ϵ for the two best and worst approaches in terms of (a) coverage $\hat{\phi}$, and (b) *ErrCoeff*. Only the two best and worst approaches are shown for readability.

resources, the bank can set a target coverage and use a selective classifier for easy-to-decide cases (where the classifier performs accurately), while demanding human experts the decision for the most difficult cases. The bank can also be interested in measuring the developed model's coverage without incurring too many (costly) mistakes and, a fortiori, planning the amount of human effort needed for the difficult cases.

In this subsection, we evaluate the performances of the bounded-abstention baselines when flipping the task to the bounded-abstention problem through the *Selection with Guaranteed Risk* (SGR) algorithm proposed by Geifman and El-Yaniv [14]. SGR is a [classifier h, confidence υ_h]-agnostic approach that optimizes the selection threshold τ (see (2.7) such that the selective error rate at test time is guaranteed to be bounded ($\leq \epsilon$) with probability $> 1 - \delta$ and the coverage is maximized. We apply SGR on all the baselines but AUCross and PluginAUC, as their hard selection function is not compatible with SGR. Moreover, since SelNet needs specific coverage for training, we use all c's one at a time and compute the average results after applying SGR. We

run experiments for four target error rates $\epsilon \in \{\hat{\epsilon}_{maj}/10, \hat{\epsilon}_{maj}/5, \hat{\epsilon}_{maj}/2, \hat{\epsilon}_{maj}\}$, where $\hat{\epsilon}_{maj}$ is the dataset-specific error of the majority-class classifier h_{maj} on the whole test set, and set $\delta = 0.001$.
Figure 7.8 reports the results for the best two and the worst two baselines. The top plot shows the mean empirical coverage over the test sets of all baseline datasets (the higher, the better). The bottom plot shows the mean *error ratio* (the smaller, the better):

$$ErrCoeff \;=\; \widehat{Err}(h, g \mid \mathcal{T}_{test})/\epsilon,$$

between the empirical selective error rate $\widehat{Err}(h, g \mid \mathcal{T}_{test})$ and the target error rate ϵ. When looking at the empirical coverage, the best-performing baseline is SCross, with coverage ranging from .999 for $\epsilon = \hat{\epsilon}_{maj}$ to .558 for $\epsilon = \hat{\epsilon}_{maj}/10$. This is 40 percentage points higher than the worst method, namely Reg. The second-best method is EnsembleSR, with a mean coverage of $\approx .970$ at $\epsilon = \hat{\epsilon}_{maj}$ and $\approx .481$ at $\epsilon = \hat{\epsilon}_{maj}/10$.
Concerning *ErrCoeff*, we observe that for less strict target errors (i.e. $\hat{\epsilon}_{maj}$ and $\hat{\epsilon}_{maj}/2$), all the baselines have error ratios close to 0. For more restrictive target errors, there is a gradual increase in the mean value of *ErrCoeff*. The methods with the smallest error ratios are EnsembleSR and SAT+EM+SmR, reaching *ErrCoeff* $\approx .316$ and *ErrCoeff* $\approx .317$ respectively at $\epsilon = \hat{\epsilon}_{maj}/10$. The worst methods are Reg and SelNet+EM+SmR, with a mean error ratio of ≈ 2.97 and ≈ 1.84 at $\epsilon = \hat{\epsilon}_{maj}/10$.

Q5. Testing the methods on out-of-distribution examples. Although SC methods are not necessarily designed for working in o.o.d. settings[6], the robustness of selective classifiers w.r.t. data shifts is highly sought. We investigate this property here by generating o.o.d. instances at test time for image datasets. We take an extreme approach by creating a test set with images made of uniformly random pixel values.[7] An ideal selection function should reject the whole test set since the images have close to zero probability of being drawn from the same distribution generating the training set.
Figure 7.9 shows the empirical coverage obtained over the o.o.d. test set for the 20 image datasets and the 16 baselines that tackle multi-class classification.
First, we observe that there is no single method that manages to reject all the test instances across all datasets. For example, most of the baselines obtain empirical coverage close to 0 (best) for the 5 lowest coverage values on `SVHN`. On the other hand, on `tissuemnist`, the majority of baselines have always nearly maximum empirical coverage, with the sole exception of Reg and Ensemble that manage to reject all the o.o.d. images.
Ensemble methods are generally better than other methods: Ensemble reaches the lowest mean empirical coverage on all datasets of $\approx .221$, ranging from $\approx .462$ at $c = .99$ to $\approx .148$ at $c = .70$, and EnsembleSR is the second best method with a mean coverage $\approx .353$, ranging from $\approx .701$ at $c = .99$ to $\approx .217$ at $c = .70$.

[6]See the novelty rejection approaches in Section 2.3.
[7]We provide additional results with less extreme shifts in the Appendix.

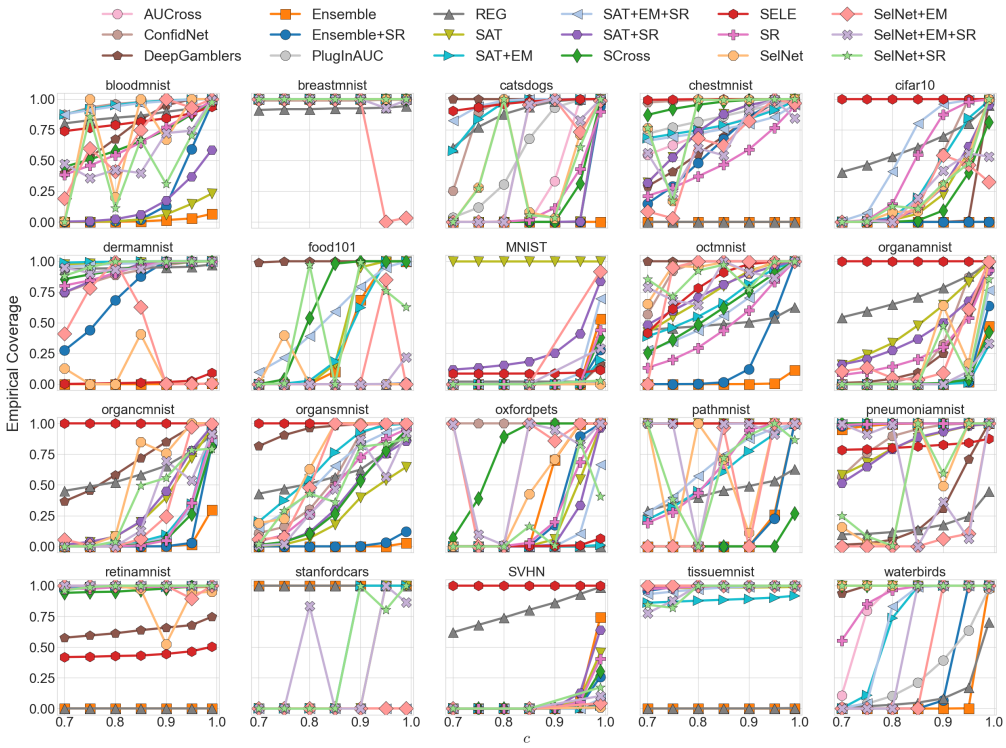


Figure 7.9: Q5: Empirical coverage $\hat{\phi}$ for out-of-distribution test sets on 20 image datasets for different target coverages c.

7.3.4 Discussion

We conclude the experimental section by briefly summarizing the main findings.

Regarding **Q1**, our results do not disclaim the folk wisdom that an ensemble strategy (paired with the softmax response) overcome other baseline methods: EnsembleSR always ranks first in terms of selective error rate. However, we stress that, depending on the target coverage, there are always at least nine baselines whose performance can't be distinguished from EnsembleSR's one in a statistically significant sense. Conversely, some methods, i.e. Reg, Sele, DeepGamblers, SelNetSelNet+EM,SelNet+EM+SmR, ConfidNet, Ensemble, and PluginAUC, perform at least once worse (in a statistically significant sense) than top-performing baselines. Hence, our findings suggest that the claimed "superiority of the considered state-of-the-art methods should be treated with caution: when we increase the number of experimental datasets, most methods perform equally well.

Our results on **Q2** show that coverage violations are generally small with a few exceptions of coverage violations above 10%. This confirms that the employed calibration strategies are well suited to achieving an empirical coverage that is fairly close to the target one.

For **Q3**, a significant differences arise among the methods. Only PluginAUC and AUCross reject equally across classes, while all the other methods abstain more relatively more frequently on the minority class. This behavior can have

unforeseen consequences such as inducing cognitive bias in the human decision-maker that must make the decision on the rejected instances [58]. For example, by abstaining more often on bad loan applicants, humans could be prone to associate the model's rejections with bad applicants, even if this might not be necessarily true [11].
The experiments for **Q4** show that SGR can effectively switch from the bounded abstention to the bounded improvement model assuming that target error rate is not too strict. In highly sensitive scenarios, where stronger guarantees are required, SGR often fails, thus suggesting a potential direction for future research towards methods specifically designed for the bounded improvement model.
For **Q5**, the results indicate that the current state-of-the-art baselines fail to reject consistently under distribution shifts. Consequently, practitioners should be cautious about applying SC techniques in the wild without considering potential issues deriving from data shifts. From a research perspective, this opens an intriguing future direction for shift-aware selective classification methods.
We point out that the methods which require to train several neural networks might not be a feasible option for very large datasets, due to the huge computational power required. Such methods include the baselines Ensemble, EnsembleSR, SCross, AUCross, ConfidNet as well as the learn-to-select methods that require training a separate model for every target coverage.

7.4 Conclusions

We extensively evaluated 18 SC baselines over 44 datasets, taking into account both images and tabular data as well as both binary and multiclass classification tasks. Regarding previously investigated tasks, our extended analysis shows that: *(i)* there are no statistically significant differences among most of the methods in terms of selective error rate, even though EnsembleSR always ranks first across all the baselines; *(ii)* large coverage violations are rare for all the methods with no significant difference among baselines for our data; *(iii)* on binary classification tasks, we observed different patterns between imbalanced and balanced domains regarding rejection rates across classes: in the former case, only AUCross and PluginAUC succeeded in not primarily rejecting the minority class. Moreover, we also emphasize novel findings: *(iv)* we tested empirically the effectiveness of SGR to switch from the bounded-abstention setting to the bounded-improvement one, noticing room for improvement when a very small target error rate is required; *(v)* we show how current methods fail in correctly rejecting instances when extreme feature shifts occur, pointing to a highly relevant open problem in the area.

Chapter 8

Conclusions

8.1 Final Remarks

This thesis's work has contributed to the advancement on the state-of-the-art of selective prediction along four different research requirements: *Req. 1* about overcoming the limits of model-specific selective predictors; *Req. 2* about extending selective prediction methods beyond distribution-based metrics; *Req. 3* concerns the applications of selective classification, e.g., to uncertainty self-assessment; *Req. 4* refers to the limited scope of current selective classifiers' empirical evaluation.

First, we addressed *Req. 1* by devising model-agnostic heuristics for selective prediction. In Chapter 3, we proposed a model-agnostic selective classifier, called SCross, which exploits a cross-fitting strategy and uses an improved second-order quantile estimation technique. Moreover, it does not require an additional hold-out set to calibrate the selection function. This is especially useful for small datasets. The empirical evaluation showed that SCross improves over existing selective classifiers on tabular data.

In Chapter 4 we devised model-agnostic heuristics for selective regression by proposing approaches based on cross-fitting and bootstrap resampling. We designed two variants for each sampling scheme: SCross-ENS and SCross-POOL rely on cross-fitting, while Doubt-Int and Doubt-Var are based on non-parametric bootstrap. We evaluated these methods over 93 tabular datasets, showing how our approaches perform on the par or better than model-agnostic baselines.

Second, we tackled *Req. 2* in Chapter 5 by presenting an AUC-based selective classification framework. Since the metrics is non-distributive, standard approaches cannot be easily lifted. We provided theoretical results to guarantee AUC improvement under abstention. Based on these results, we devised two algorithms, i.e. PluginAUC and AUCross. The former relies on a labeled calibration set to build the selection function, while AUCross exploits a cross-fitting strategy similar to SCross. The empirical evaluation showed that the proposed approaches are the only ones that guarantee AUC improvement and to balance rejection over the class values.

Next, we focused on *Req. 3* in Chapter 6 by presenting LSP, a credit-scoring

system deployed at the Italian Leasing Association. Several objectives are considered for the design of LSP, including the improvement over models trained on the single associate data, the adaptation to prior distribution shifts, and the uncertainty self-assessment of the scores. With regard to the last objective, we built on selective classification to design a self-assessment method rating predictions into three levels of uncertainty.
Finally, we addressed *Req. 4* in Chapter 7. We extensively evaluated 18 neural-network-based selective classifiers over 44 datasets, taking into account both images and tabular data as well as binary and multiclass classification tasks. Our analysis showed that: *(i)* no clear winner emerges in terms of selective error rate, as multiple methods achieve statistically indistinguishable performance; *(ii)* there are no large coverage violations among baselines; *(iii)* on binary classification tasks, only AUCross and PluginAUC succeed in not primarily rejecting the minority class when considering imbalanced settings; *(iv)* when switching from the bounded-abstention setting to the bounded-improvement one, there is room for improvement whenever a very small target error rate is required; *(v)* current methods fail in rejecting instances when extreme feature shifts occur.

8.2 Future Work

Several potential directions are open for future research.
Existing selective classification methods are not able to properly reject instances drawn from distributions different from the one of training. Novelty rejection methods are designed for such a task, even though they might fail in detecting instances where the classifier makes mistakes. How to combine ambiguity rejection and novelty rejection is still an open question. A potential way to improve current methodologies can be to consider selective predictors that are based on neuro-symbolic approaches. These approaches extend traditional learning techniques with reasoning strategies (e.g., based on logic or causality), improving their generalization ability. For example, a recent work by Misino et al. [199] proposes integrating a probabilistic logic module within a variational autoencoder to improve the neural network's ability to solve unseen generation tasks. Adapting such an approach to selective prediction might be used to build more robust abstaining predictors. Alternatively, depending on the context, (causal) domain adaptation techniques [200] might be considered to improve current methods.
An open issue is how to properly evaluate selective predictors. The majority of works (including ours) consider metrics computed over the accepted instances, without taking into account the rejected ones. Only a few works have tried to tackle such an aspect, but they either rely on strong assumptions [201] or they only consider distribution-based losses [202]. Evaluation on rejected instances becomes especially relevant when assessing ethical aspects, such as fairness. E.g., a selective predictor can be fair on the set of accepted instances, while unfairly rejecting instances belonging to protected social groups.
AI systems have mainly been evaluated in a static setting, while only a few

works focused on dynamic scenarios [203]. Defining the long-term effects of the reject option is an open question. Recently, the performative prediction framework [204] has been proposed to study how ML models affect the distribution of future data. Adding the reject option within the performative prediction setting could help us model what are the long-term consequences of introducing the reject option.

Another research topic regards the effectiveness of the reject option in improving human-AI collaboration. XAI tools have succeeded in improving the user's understanding of how AI systems work in many domains [205], thus fostering trust in AI systems. However, Bondi et al. [11] highlight how the perception of the reject option is affected by different levels of human expertise. Therefore, a potential direction is to study how XAI and the rejection option can merge towards the direction of trust in human-AI collaboration [57].

Abstaining methods have been developed mainly in the setting of classification and regression. How to build abstaining predictors for other tasks, e.g., quantification [206] or quantile regression [207], remains an unaddressed problem. Another relatively understudied setting is the one of multi-target prediction [208]. A few approaches have been proposed to embed multi-target predictors with a reject option, considering cost-based approaches and relying on thresholding the score of predicted targets [209, 210]. Extending our AUC-based approach to (binary) multitarget settings could help account for the imbalance in the various targets.

Finally, other ethical concerns in the context of selective prediction remain unaddressed. For example, privacy issues have been overlooked, even though adding the reject option might expose AI systems to many privacy threats. For instance, abstention could provide information to malignant agents in performing new kinds of membership attacks [211].

List of Figures

1.1 Canonical SVM Classifier. Instances above the black line are predicted as blue circle-shaped instances, while those below are predicted as pink diamond-shaped. 3
1.2 Ambiguity Rejection example. Abstention occurs for those instances within the orange area delimited by the two blue dotted lines, i.e. those close to the decision boundary. 3
1.3 Novelty Rejection example. Abstention occurs for the cross-marked instances within the orange area, i.e. those far from the original data. 3

2.1 A hierarchy of methods to build abstaining predictors. 9

3.1 The cross-fitting strategy 23

4.1 ΔMSE as a function of tolerance c when using `XGBoost` as a base regressor. 45
4.2 Q1: CD plots of ΔMSE for different target coverages. The closer to one, the better the method. 46
4.3 *ConSat* as a function of tolerance ε when using `XGBoost` as a base regressor. 47
4.4 Q1: CD plots of ΔMSE for different target coverages, using LightGBM. The closer to one, the better the method. 48
4.5 Q1: CD plots of ΔMSE for different target coverages, using RandomForest. The closer to one, the better the method. . . 49
4.6 Q1: CD plots of ΔMSE for different target coverages, using MLP. The closer to one, the better the method. 50
4.7 Q1: CD plots of ΔMSE for different target coverages, using MLP-Large. The closer to one, the better the method. 50
4.8 Q1: CD plots of ΔMSE for different target coverages, using LinearRegression. The closer to one, the better the method. 51

5.1 Example of a CAP plot with $n = 20$ and $n^+ = 7$. 55
5.2 CAP plot if we were to remove a positive instance. 56
5.3 CAP plot after removing a positive instance. 56
5.4 CAP plot if we were to remove a negative instance. 57
5.5 CAP plot after removing a negative instance. 58
5.6 An example of the rejection area using PluginAUC vs the standard PlugIn approach. 60

5.7 AUC by varying bounds: Adult and $c = .90$. 68

6.1 The LSP workflow. 78
6.2 An example of LSP output. 79
6.3 Example of LSP quality prediction distribution on simulated data. 84
6.4 Backtesting framework. 88
6.5 Positive rate, training size and time. 89
6.6 Performance comparison: LSP vs LSP-`Cor` vs LSP-`Cor+Beta` vs `LGBM`. 92
6.7 Performance over subsets of test data by prediction quality. . . . 92
6.8 LSP trained on all data w.r.t. LSP trained a single associate data. 93

7.1 A generic Learn-to-Abstain architecture 97
7.2 An example of SelNet, a Learn-to-Select architecture. 98
7.3 An example of ConfidNet, a score-based architecture. 99
7.4 Q1: *RelErr* as a function of target coverage c for two best and worst approaches on (a) binary and (b) multiclass problems. On each subplot, only the two best and worst approaches are shown for readability. 105
7.5 Q1: CD plots of relative error rate *RelErr* for different target coverages. Top plots for binary datasets. Bottom plots for multiclass datasets. 106
7.6 Q2: *ConSat* as a function of tolerance ε for two best and worst approaches on (a) binary and (b) multiclass problems. On each subplot, only the two best and worst approaches are shown for readability. 107
7.7 Q3: *MinCoeff* as a function of target coverage c for two best and worst approaches. Only the two best and worst approaches are shown for readability. 108
7.8 Q4: SGR performance as a function of target error rate ϵ for the two best and worst approaches in terms of (a) coverage $\hat{\phi}$, and (b) *ErrCoeff*. Only the two best and worst approaches are shown for readability. 109
7.9 Q5: Empirical coverage $\hat{\phi}$ for out-of-distribution test sets on 20 image datasets for different target coverages c. 111

A.1 Q1: CD plots of ΔMAE for different target coverages. The closer to one, the better the method. 143
A.2 Q1: CD plots of ΔMAE for different target coverages, using LightGBM. The closer to one, the better the method. 143
A.3 Q1: CD plots of ΔMAE for different target coverages, using RandomForest. The closer to one, the better the method. . . 144
A.4 Q1: CD plots of ΔMAE for different target coverages, using MLP. The closer to one, the better the method. 144

A.5 Q1: CD plots of ΔMAE for different target coverages, using MLP-Large. The closer to one, the better the method. 145
A.6 Q1: CD plots of ΔMAE for different target coverages, using LinearRegression. The closer to one, the better the method. 145

C.1 Q1: *RelErr* as a function of target coverage c for the two best and worst approaches on (a) binary tabular data, (b) binary image data, (c) multiclass tabular data and (d) multiclass image data. For readability, only the two best and two worst approaches are shown in each subplot. 177
C.2 Q1: CD plots of relative error rate *RelErr* for different target coverages on tabular datasets. Top plots for binary datasets. Bottom plots for multiclass datasets. 178
C.3 Q1: CD plots of relative error rate *RelErr* for different target coverages on image datasets. Top plots for binary datasets. Bottom plots for multiclass datasets. 179
C.4 Q5: Empirical coverage $\hat{\phi}$ for out-of-distribution test sets on 3 image datasets for different target coverages c. 180

List of Tables

3.1 Empirical coverage for SCross and baselines (1,000 bootstrap runs over the test set, results as mean ± stdev). 28
3.2 Selective Accuracy for SCross and baselines (1,000 bootstrap runs over the test set, results as mean ± stdev). 29
3.3 Elapsed Training Time for SCross and baselines. 30
3.4 Empirical Coverage for SCross with different base classifiers over tabular datasets $k = 5$, and 1,000 bootstrap runs over the test set, results as mean ± stdev. 32
3.5 Selective Accuracy for SCross with different base classifiers over tabular datasets $k = 5$, and 1,000 bootstrap runs over the test set, results as mean ± stdev. 33
3.6 Empirical coverage results for SCross with different K, and 1,000 bootstrap runs over the test set, results as mean ± stdev. 34
3.7 Selective Accuracy results for SCross with different K, and 1,000 bootstrap runs over the test set, results as mean ± stdev. 35

5.1 Experimental datasets. 64
5.2 Empirical coverage (1,000 bootstrap runs over the test set, results as mean ± stdev). V (for violation) is the absolute difference between the empirical coverage and the target coverage c. 66
5.3 Selective AUC (1,000 bootstrap runs over the test set, results as mean ± stdev). V (for violation) is the absolute difference between the empirical coverage and the target coverage c. 67
5.4 Absolute deviation in empirical AUC of AUCross w.r.t. an Oracle. 69

6.1 Related credit scoring systems. OOT $d \times n$ is Out of Time with d = test period duration, n = number of periods. 74

A.1 Data from `PMLB` benchmark [120] 140
A.2 Data from Grinsztajn et al. benchmark [103] 141
A.3 Data from `folktables` [121] 142

B.1 Selective accuracy (1,000 bootstrap runs over the test set, results as mean ± stdev). 150
B.2 Positive rate (1,000 bootstrap runs over the test set, results as mean ± stdev). 151
B.3 Time required for running different methods (in seconds). 152

B.4 Empirical coverage for AUCross using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev). . . . 153
B.5 Empirical coverage for PlugInAUC using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).154
B.6 Selective AUC for AUCross using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev). . . . 155
B.7 Selective AUC for PlugInAUC using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev). . . . 156
B.8 Selective accuracy for AUCross using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev). . . . 157
B.9 Selective accuracy for PlugInAUC using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).158
B.10 Positive rate for AUCross using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev). 159
B.11 Positive rate for PlugInAUC using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev). . . . 160
B.12 Empirical coverage for AUCross using different number of folds K over tabular datasets (1,000 bootstrap runs over the test set, results as mean ± stdev). 161
B.13 Selective AUC for AUCross using different number of folds K over tabular datasets (1,000 bootstrap runs over the test set, results as mean ± stdev). 162

C.1 Dataset sources. 164
C.2 Dataset details. 165
C.3 Hyperparameter spaces. 166
C.4 Best configurations for DeepGamblers, divided by architectures. 167
C.5 Best configurations for SAT, divided by architectures. 168
C.6 Best configurations for SAT+EM, divided by architectures. . . 169
C.7 Best configurations for SelNet, divided by architectures. 170
C.8 Best configurations for SelNet+EM, divided by architectures. 171
C.9 Best configurations for ConfidNet, divided by architectures. . 172
C.10 Best configurations for Reg, divided by architectures. 173
C.11 Best configurations for Sele, divided by architectures. 174
C.12 Best configurations for SmR, divided by architectures. 175
C.13 Results for `adult`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 181
C.14 Results for `aloi`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 181
C.15 Results for `bank`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 181
C.16 Results for `bloodmnist`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 181
C.17 Results for `breastmnist`: *mean*±*std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 182
C.18 Results for `catsdogs`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 182

C.19 Results for `chestmnist`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 182
C.20 Results for `cifar10`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . . . 182
C.21 Results for `compass`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 182
C.22 Results for `covtype`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . . . 183
C.23 Results for `dermamnist`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . . . 183
C.24 Results for `electricity`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 183
C.25 Results for `eye`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 183
C.26 Results for `food101`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . . . 183
C.27 Results for `giveme`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 184
C.28 Results for `helena`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 184
C.29 Results for `heloc`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 184
C.30 Results for `higgs`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 184
C.31 Results for `house`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 184
C.32 Results for `indian`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 185
C.33 Results for `jannis`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 185
C.34 Results for `kddipums97`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 185
C.35 Results for `letter`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 185
C.36 Results for `magic`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 185
C.37 Results for `miniboone`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 186
C.38 Results for `MNIST`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 186
C.39 Results for `octmnist`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . . . 186
C.40 Results for `online`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* . . . 186
C.41 Results for `organamnist`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . . . 186
C.42 Results for `organcmnist`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . . . 187
C.43 Results for `organsmnist`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . . . 187

C.44 Results for `oxfordpets`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 187
C.45 Results for `pathmnist`: *mean* $\pm$ *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 187
C.46 Results for `phoneme`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 187
C.47 Results for `pneumoniamnist`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 188
C.48 Results for `pol`: *mean* $\pm$ *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. . 188
C.49 Results for `retinamnist`: *mean* $\pm$ *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 188
C.50 Results for `rl`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 188
C.51 Results for `stanfordcars`: *mean* $\pm$ *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 188
C.52 Results for `SVHN`: *mean* $\pm$ *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 189
C.53 Results for `tissuemnist`: *mean* $\pm$ *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$. 189
C.54 Results for `ucicredit`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 189
C.55 Results for `upselling`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 189
C.56 Results for `waterbirds`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff* 189

Bibliography

[1] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019.

[2] Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, Ioannis Kompatsiaris, Katharina Kinder-Kurlanda, Claudia Wagner, Fariba Karimi, Miriam Fernández, Harith Alani, Bettina Berendt, Tina Kruegel, Christian Heinze, Klaus Broelemann, Gjergji Kasneci, Thanassis Tiropanis, and Steffen Staab. Bias in data-driven artificial intelligence systems - an introductory survey. *WIREs Data Mining Knowl. Discov.*, 10(3), 2020.

[3] Davinder Kaur, Suleyman Uslu, Kaley J. Rittichier, and Arjan Durresi. Trustworthy Artificial Intelligence: A review. *ACM Comput. Surv.*, 55 (2):39:1–39:38, 2023.

[4] European Commission. Proposal for a Regulation of the European Parliament and of the Council Laying down harmonised rules on Artificial Intelligence (AI Act) and amending certain Union legislative acts, 2021. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206`.

[5] Sarah JC Craig, Ana M Kenney, Junli Lin, Ian M Paul, Leann L Birch, Jennifer S Savage, Michele E Marini, Francesca Chiaromonte, Matthew L Reimherr, and Kateryna D Makova. Constructing a polygenic risk score for childhood obesity using functional data analysis. *Econometrics and statistics*, 25:66–86, 2023.

[6] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica, May*, 23(2016):139–159, May 2016.

[7] Alessandro Fabris, Nina Baranowska, Matthew J. Dennis, Philipp Hacker, Jorge Saldivar, Frederik J. Zuiderveen Borgesius, and Asia J. Biega. Fairness and bias in algorithmic hiring. *CoRR*, abs/2309.13933, 2023.

[8] Matilde Lazzari, José M. Álvarez, and Salvatore Ruggieri. Predicting and explaining employee turnover intention. *Int. J. Data Sci. Anal.*, 14 (3):279–292, 2022.

[9] Xolani Dastile, Turgay Çelik, and Moshe Potsane. Statistical and machine learning models in credit scoring: A systematic literature survey. *Appl. Soft Comput.*, 91:106263, 2020.

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

[11] Elizabeth Bondi, Raphael Koster, Hannah Sheahan, Martin J. Chadwick, Yoram Bachrach, A. Taylan Cemgil, Ulrich Paquet, and Krishnamurthy Dvijotham. Role of human-ai interaction in selective prediction. In *AAAI*, pages 5286–5294. AAAI Press, 2022.

[12] Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis. Machine learning with a reject option: a survey. *Mach. Learn.*, 113(5):3073–3110, 2024.

[13] Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *J. Mach. Learn. Res.*, 11:1605–1641, 2010.

[14] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *NIPS*, pages 4878–4887, 2017.

[15] Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2151–2159. PMLR, 2019.

[16] Ziyin Liu, Zhikang Wang, Paul Pu Liang, Ruslan Salakhutdinov, Louis-Philippe Morency, and Masahito Ueda. Deep gamblers: Learning to abstain with portfolio theory. In *NeurIPS*, pages 10622–10632, 2019.

[17] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. In *NeurIPS*, pages 2898–2909, 2019.

[18] Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. In *NeurIPS*, 2020.

[19] Aditya Gangrade, Anil Kag, and Venkatesh Saligrama. Selective classification via one-sided prediction. In *AISTATS*, volume 130 of *Proceedings of Machine Learning Research*, pages 2179–2187. PMLR, 2021.

[20] Leo Feng, Mohamed Osama Ahmed, Hossein Hajimirsadeghi, and Amir H. Abdi. Towards better selective classification. In *ICLR*. OpenReview.net, 2023.

[21] Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In *NIPS*, pages 313–320. MIT Press, 2003.

[22] Vojtech Franc and Daniel Průša. On discriminative learning of prediction uncertainty. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 1963–1971. PMLR, 2019.

[23] Qiang Ding, Yixuan Cao, and Ping Luo. Top-ambiguity samples matter: Understanding why deep ensemble works in selective classification. In *NeurIPS*, 2023.

[24] Vojtech Franc, Daniel Průša, and Václav Vorácek. Optimal strategies for reject option classifiers. *J. Mach. Learn. Res.*, 24:11:1–11:49, 2023.

[25] Christophe Denis and Mohamed Hebiri. Consistency of plug-in confidence sets for classification in semi-supervised learning. *J. of Nonpar. Statistics*, 32(1):42–72, 2020.

[26] Ahmed Zaoui, Christophe Denis, and Mohamed Hebiri. Regression with reject option and application to knn. In *NeurIPS*, 2020.

[27] Radu Herbei and Maten H. Wegkamp. Classification with reject option. *Can. J. Stat.*, 34(4):709–721, 2006.

[28] Giorgio Fumera, Fabio Roli, and Giorgio Giacinto. Reject option with multiple thresholds. *Pattern Recognit.*, 33(12):2099–2101, 2000.

[29] Martin E. Hellman. The nearest neighbor classification rule with a reject option. *IEEE Trans. Syst. Sci. Cybern.*, 6(3):179–185, 1970.

[30] Giorgio Fumera and Fabio Roli. Support vector machines with embedded reject option. In *SVM*, volume 2388 of *Lecture Notes in Computer Science*, pages 68–82. Springer, 2002.

[31] Luigi P. Cordella, Claudio De Stefano, Francesco Tortorella, and Mario Vento. A method for improving classification reliability of multilayer perceptrons. *IEEE Trans. Neural Networks*, 6(5):1140–1147, 1995.

[32] Ramasubramanian Sundararajan and Asim K Pal. A conservative approach to perceptron learning. In *NNA*, 2004.

[33] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Boosting with abstention. In *NIPS*, pages 1660–1668, 2016.

[34] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Optimal local rejection for classifiers. *Neurocomputing*, 214:445–457, 2016.

[35] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[36] Bernard Dubuisson and Mylène Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognit.*, 26(1):155–165, 1993.

[37] Luigi P. Cordella, Claudio De Stefano, Carlo Sansone, and Mario Vento. An adaptive reject option for LVQ classifiers. In *ICIAP*, volume 974 of *Lecture Notes in Computer Science*, pages 68–73. Springer, 1995.

[38] Dries Van der Pias, Wannes Meert, Johan Verbraecken, and Jesse Davis. A novel reject option applied to sleep stage scoring. In *SDM*, pages 820–828. SIAM, 2023.

[39] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. In *SIGMOD Conference*, pages 93–104. ACM, 2000.

[40] Thomas C.W . Landgrebe, David M. J. Tax, Pavel Paclík, Robert P. W. Duin, and Andrew Colin. A combining strategy for ill-defined problems. In *Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa*, pages 57–62, 2004.

[41] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 4723–4732. PMLR, 2019.

[42] Xin Wang and Siu-Ming Yiu. Classification with rejection: Scaling generative classifiers with supervised deep infomax. In *IJCAI*, pages 2980–2986. ijcai.org, 2020.

[43] Lize Coenen, Ahmed K. A. Abdullah, and Tias Guns. Probability of default estimation, with a reject option. In *DSAA*, pages 439–448. IEEE, 2020.

[44] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR (Poster)*. OpenReview.net, 2018.

[45] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.

[46] Joana Kühne, Christian März, et al. Securing deep learning models with autoencoder based anomaly detection. In *PHM Society European Conference*, volume 6, pages 13–13, 2021.

[47] Filipe Condessa, José M. Bioucas-Dias, and Jelena Kovacevic. Robust hyperspectral image classification with rejection fields. In *WHISPERS*, pages 1–4. IEEE, 2015.

[48] Aditya Vailaya and Anil K. Jain. Reject option for vq-based bayesian classification. In *ICPR*, pages 2048–2051. IEEE Computer Society, 2000.

[49] Donatello Conte, Pasquale Foggia, Gennaro Percannella, Alessia Saggese, and Mario Vento. An ensemble of rejecting classifiers for anomaly detection of audio events. In *AVSS*, pages 76–81. IEEE Computer Society, 2012.

[50] Salvatore Ruggieri, José M. Álvarez, Andrea Pugnana, Laura State, and Franco Turini. Can we trust fair-ai? In *AAAI*, pages 15421–15430. AAAI Press, 2023.

[51] Erik Jones, Shiori Sagawa, Pang Wei Koh, Ananya Kumar, and Percy Liang. Selective classification can magnify disparities across groups. In *ICLR*. OpenReview.net, 2021.

[52] Joshua K. Lee, Yuheng Bu, Deepta Rajan, Prasanna Sattigeri, Rameswar Panda, Subhro Das, and Gregory W. Wornell. Fair selective classification via sufficiency. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 6076–6086. PMLR, 2021.

[53] Abhin Shah, Yuheng Bu, Joshua K. Lee, Subhro Das, Rameswar Panda, Prasanna Sattigeri, and Gregory W. Wornell. Selective regression under fairness criteria. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 19598–19615. PMLR, 2022.

[54] Nicolas Schreuder and Evgenii Chzhen. Classification with abstention but without disparities. In *UAI*, volume 161 of *Proceedings of Machine Learning Research*, pages 1227–1236. AUAI Press, 2021.

[55] André Artelt, Johannes Brinkrolf, Roel Visser, and Barbara Hammer. Explaining reject options of learning vector quantization classifiers. In *IJCCI*, pages 249–261. SCITEPRESS, 2022.

[56] André Artelt and Barbara Hammer. "even if ..." - diverse semifactual explanations of reject. In *SSCI*, pages 854–859. IEEE, 2022.

[57] André Artelt, Roel Visser, and Barbara Hammer. "i do not know! but why?" - local model-agnostic example-based explanations of reject. *Neurocomputing*, 558:126722, 2023.

[58] Charvi Rastogi, Yunfeng Zhang, Dennis Wei, Kush R. Varshney, Amit Dhurandhar, and Richard Tomsett. Deciding fast and slow: The role of cognitive biases in ai-assisted decision-making. *Proc. ACM Hum. Comput. Interact.*, 6(CSCW1):83:1–83:22, 2022.

[59] Ben Green and Yiling Chen. The principles and limits of algorithm-in-the-loop decision making. *Proc. ACM Hum. Comput. Interact.*, 3 (CSCW):50:1–50:24, 2019.

[60] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[61] Cassidy Laidlaw and Soheil Feizi. Playing it safe: Adversarial robustness with an abstain option. *CoRR*, abs/1911.11253, 2019.

[62] Tianyu Pang, Huishuai Zhang, Di He, Yinpeng Dong, Hang Su, Wei Chen, Jun Zhu, and Tie-Yan Liu. Two coupled rejection metrics can tell adversarial examples apart. In *CVPR*, pages 15202–15212. IEEE, 2022.

[63] Maria-Florina Balcan, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. An analysis of robustness of non-lipschitz networks. *J. Mach. Learn. Res.*, 24:98:1–98:43, 2023.

[64] Jiefeng Chen, Jayaram Raghuram, Jihye Choi, Xi Wu, Yingyu Liang, and Somesh Jha. Stratified adversarial robustness with rejection. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 4867–4894. PMLR, 2023.

[65] Robin Senge, Stefan Bösner, Krzysztof Dembczynski, Jörg Haasenritter, Oliver Hirsch, Norbert Donner-Banzhoff, and Eyke Hüllermeier. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Inf. Sci.*, 255:16–29, 2014.

[66] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.*, 110(3):457–506, 2021.

[67] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[68] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, pages 609–616. Morgan Kaufmann, 2001.

[69] Urvashi Ramdasani, Gunjan Vinzuda, Sudeep Tanwar, Rajesh Gupta, and Mohsen Guizani. Dubloq: Blockchain and q-learning based drug discovery in healthcare 4.0. In *IWCMC*, pages 284–289. IEEE, 2022.

[70] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*, pages 694–699. ACM, 2002.

[71] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[72] Meelis Kull, Telmo de Menezes e Silva Filho, and Peter A. Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 623–631. PMLR, 2017.

[73] Meelis Kull, Miquel Perelló-Nieto, Markus Kängsepp, Telmo de Menezes e Silva Filho, Hao Song, and Peter A. Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *NeurIPS*, pages 12295–12305, 2019.

[74] Telmo de Menezes e Silva Filho, Hao Song, Miquel Perelló-Nieto, Raúl Santos-Rodríguez, Meelis Kull, and Peter A. Flach. Classifier calibration: a survey on how to assess and improve predicted class probabilities. *Mach. Learn.*, 112(9):3211–3260, 2023.

[75] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, pages 6402–6413, 2017.

[76] Carlos Mougan and Dan Saattrup Nielsen. Monitoring model deterioration with explainable uncertainty estimation via non-parametric bootstrap. In *AAAI*, pages 15037–15045. AAAI Press, 2023.

[77] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *NeurIPS*, pages 14680–14691, 2019.

[78] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, pages 15262–15271. Computer Vision Foundation / IEEE, 2021.

[79] Matthias W. Seeger. Gaussian processes for machine learning. *Int. J. Neural Syst.*, 14(2):69–106, 2004.

[80] John S. Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In *NIPS*, pages 853–859. Morgan Kaufmann, 1990.

[81] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, 1992.

[82] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[83] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org, 2016.

[84] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*, pages 5574–5584, 2017.

[85] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1192–1201. PMLR, 2018.

[86] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *J. Mach. Learn. Res.*, 9:371–421, 2008.

[87] Vladimir Vovk. Cross-conformal predictors. *CoRR*, abs/1208.0806, 2012.

[88] Byol Kim, Chen Xu, and Rina Foygel Barber. Predictive inference is free with the jackknife+-after-bootstrap. In *NeurIPS*, 2020.

[89] Javier Abad Martinez, Umang Bhatt, Adrian Weller, and Giovanni Cherubin. Approximating full conformal prediction at scale via influence functions. In *AAAI*, pages 6631–6639. AAAI Press, 2023.

[90] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alexander Gammerman. Inductive confidence machines for regression. In *ECML*, volume 2430 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2002.

[91] Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *CoRR*, abs/2110.01052, 2021.

[92] Yaniv Romano, Matteo Sesia, and Emmanuel J. Candès. Classification with valid and adaptive coverage. In *NeurIPS*, 2020.

[93] Jean Feng, Arjun Sondhi, Jessica Perry, and Noah Simon. Selective prediction-set models with coverage rate guarantees. *Biometrics*, 79(2): 811–825, 2023.

[94] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *CoRR*, abs/2107.07511, 2021.

[95] David Madras, Toniann Pitassi, and Richard S. Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *NeurIPS*, pages 6150–6160, 2018.

[96] Hussein Mozannar and David A. Sontag. Consistent estimators for learning to defer to an expert. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 7076–7087. PMLR, 2020.

[97] Abir De, Paramita Koley, Niloy Ganguly, and Manuel Gomez-Rodriguez. Regression under human assistance. In *AAAI*, pages 2611–2620. AAAI Press, 2020.

[98] Abir De, Nastaran Okati, Ali Zarezade, and Manuel Gomez Rodriguez. Classification under human assistance. In *AAAI*, pages 5905–5913. AAAI Press, 2021.

[99] Nastaran Okati, Abir De, and Manuel Gomez-Rodriguez. Differentiable learning under triage. In *NeurIPS*, pages 9140–9151, 2021.

[100] Rajeev Verma, Daniel Barrejón, and Eric T. Nalisnick. Learning to defer to multiple experts: Consistent surrogate losses, confidence calibration, and conformal ensembles. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pages 11415–11434. PMLR, 2023.

[101] Hussein Mozannar, Hunter Lang, Dennis Wei, Prasanna Sattigeri, Subhro Das, and David A. Sontag. Who should predict? exact algorithms for learning to defer to humans. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pages 10520–10545. PMLR, 2023.

[102] Andrea Pugnana and Salvatore Ruggieri. A model-agnostic heuristics for selective classification. In *AAAI*, pages 9461–9469. AAAI Press, 2023.

[103] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS*, 2022.

[104] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *NeurIPS*, pages 18932–18943, 2021.

[105] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018.

[106] Keith Knight and Gilbert W. Bassett. Second order improvements of sample quantiles using subsamples, 2003. *Unpublished*.

[107] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. *Public Repository*.

[108] Jean Paul Barddal, Lucas Loezer, Fabrício Enembreck, and Riccardo Lanzuolo. Lessons learned from data stream classification applied to credit scoring. *Expert Syst. Appl.*, 162:113899, 2020.

[109] I-Cheng Yeh and Che-hui Lien. 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, 2009.

[110] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, pages 3146–3154, 2017.

[111] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):1–10, 2018.

[112] Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[113] Steffen Herbold. Autorank: A python package for automated ranking of classifiers. *J. Open Source Softw.*, 5(48):2173, 2020.

[114] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Inf. Fusion*, 81:84–90, 2022.

[115] Wenming Jiang, Ying Zhao, and Zehan Wang. Risk-controlled selective prediction for regression deep neural network models. In *IJCNN*, pages 1–8. IEEE, 2020.

[116] Edward H Kennedy. Semiparametric doubly robust targeted double machine learning: a review. *arXiv preprint arXiv:2203.06469*, 2022.

[117] Aaron Fisher and Virginia Fisher. Three-way cross-fitting and pseudo-outcome regression for estimation of conditional effects and other linear functionals. *arXiv preprint arXiv:2306.07230*, 2023.

[118] Sricharan Kumar and Ashok Srivastava. Bootstrap prediction intervals in non-parametric regression with applications to anomaly detection. In *Proc. 18th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2012.

[119] Emanuel Parzen. Quantile functions, convergence in quantile, and extreme value distribution theory. *Texas A & M University Technical Report No. B-3*, 1980.

[120] Randal S. Olson, William G. La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Min.*, 10(1):36:1–36:13, 2017.

[121] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. In *NeurIPS*, pages 6478–6490, 2021.

[122] Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. Conformalized quantile regression. In *NeurIPS*, pages 3538–3548, 2019.

[123] Chen Xu and Yao Xie. Conformal prediction interval for dynamic time-series. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 11559–11569. PMLR, 2021.

[124] Rina Foygel Barber, Emmanuel J. Candes, Aaditya Ramdas, and Ryan J. Tibshirani. Predictive inference with the jackknife+, 2020.

[125] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM, 2016.

[126] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *CoRR*, abs/2110.01889, 2021.

[127] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *CoRR*, abs/1910.09700, 2019.

[128] Benoit Courty, Victor Schmidt, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, SabAmine, Mathilde Léval, Alexis Cruveiller, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, LiamConnell, Amine Saboni, Douglas Blank, Ziyao Wang, Armin Catovic, alencon, JPW, MinervaBooks, SangamSwadiK, brotherwolf, and Martin Pollard. mlco2/codecarbon: v2.2.7, 2023. URL `https://doi.org/10.5281/zenodo.8181237`.

[129] Andrea Pugnana and Salvatore Ruggieri. AUC-based selective classification. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pages 2494–2514. PMLR, 2023.

[130] Bernd Engelmann. Measures of a ratings discriminative power: Applications and limitations. In B. Engelmann and R. Rauhmeier, editors, *The Basel II Risk Parameters*, pages 269–291. Springer, 2 edition, 2011.

[131] Jin Huang and Charles X. Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Trans. Knowl. Data Eng.*, 17(3):299–310, 2005.

[132] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognit. Lett.*, 27(8):861–874, 2006.

[133] Yun-Chun Wu and Wen-Chung Lee. Alternative performance measures for prediction models. *PLoS ONE*, 9(3):e91249, 2014.

[134] Radu Herbei and Marten H Wegkamp. Classification with reject option. *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, pages 709–721, 2006.

[135] Song-Qing Shen, Bin-Bin Yang, and Wei Gao. AUC optimization with a reject option. In *AAAI*, pages 5684–5691. AAAI Press, 2020.

[136] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[137] Giuseppe Cianci, Roberto Goglia, Riccardo Guidotti, Matteo Kapllaj, Roberto Mosca, Andrea Pugnana, Franco Riccotti, and Salvatore Ruggieri. Applied data science for leasing score prediction. In *IEEE Big Data*, pages 1687–1696. IEEE, 2023.

[138] Anton Markov, Zinaida Seleznyova, and Victor Lapshin. Credit scoring methods: Latest trends and points to consider. *The Journal of Finance and Data Science*, 8:180–201, 2022.

[139] European Parliament and the Council. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016.

[140] Sofie Balcaen and Hubert Ooghe. 35 years of studies on business failure: an overview of the classic statistical methodologies and their related problems. *The British Accounting Review*, pages 63–93, 2006.

[141] Lore Dirick, Gerda Claeskens, and Bart Baesens. Time to default in credit scoring using survival analysis: a benchmark study. *J. Oper. Res. Soc.*, 68(6):652–665, 2017.

[142] Joseph Breeden. A survey of machine learning in credit risk. *Journal of Credit Risk*, 17(3), 2021.

[143] Hal R. Varian. Big Data: New Tricks for Econometrics. *Journal of Economic Perspectives*, 28(2):3–28, 2014.

[144] Manuel Fernández Delgado, Eva Cernadas, Senén Barro, and Dinani Gomes Amorim. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133–3181, 2014.

[145] Paula Branco, Luís Torgo, and Rita P. Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Comput. Surv.*, 49(2):31:1–31:50, 2016.

[146] Diwakar Tripathi, Alok Kumar Shukla, B. Ramachandra Reddy, Ghanshyam S. Bopche, and D. Chandramohan. Credit scoring models using ensemble learning and classification approaches: A comprehensive survey. *Wirel. Pers. Commun.*, 123(1):785–812, 2022.

[147] Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow, and Lyn C. Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *Eur. J. Oper. Res.*, 247(1):124–136, 2015.

[148] Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperlì. A benchmark of machine learning approaches for credit score prediction. *Expert Syst. Appl.*, 165:113986, 2021.

[149] Joaquín Abellán and Javier G. Castellano. A comparative study on base classifiers in ensemble methods for credit scoring. *Expert Syst. Appl.*, 73: 1–10, 2017.

[150] Dirk Tasche. The art of probability-of-default curve calibration. *Journal of Credit Risk*, 9(4):63–103, 2013.

[151] Xinran He et al. Practical lessons from predicting clicks on ads at Facebook. In *ADKDD@KDD*, pages 5:1–5:9. ACM, 2014.

[152] A. R. Provenzano, D. Trifirò, A. Datteo, L. Giada, N. Jean, A. Riciputi, G. Le Pera, M. Spadaccino, L. Massaron, and C. Nordio. Machine learning approach for credit scoring. *arXiv:2008.01687*, 2020.

[153] Michael Bücker, Gero Szepannek, Alicja Gosiewska, and Przemyslaw Biecek. Transparency, auditability, and explainability of machine learning models in credit scoring. *J. Oper. Res. Soc.*, 73(1):70–90, 2022.

[154] Philippe Bracke, Anupam Datta, Carsten Jung, and Shayak Sen. Machine learning explainability in finance: an application to default risk analysis, 2019. Bank of England Working Paper.

[155] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017.

[156] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. Explainable machine learning in credit risk management. *Computational Economics*, 57(1):203–216, 2021.

[157] Neus Llop Torrent, Giorgio Visani, and Enrico Bagli. PSD2 explainable AI model for credit scoring. *CoRR*, abs/2011.10367, 2020.

[158] Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55, 2022.

[159] Rory McGrath, Luca Costabello, Chan Le Van, Paul Sweeney, Farbod Kamiab, Zhao Shen, and Freddy Lécué. Interpretable credit application predictions with counterfactual explanations. *CoRR*, abs/1811.05245, 2018.

[160] Andreas C. Bueff, Mateusz Cytrynski, Raffaella Calabrese, Matthew Jones, John Roberts, Jonathon Moore, and Iain Brown. Machine learning interpretability for a stress scenario generation in credit scoring based on counterfactuals. *Expert Syst. Appl.*, 202:117271, 2022.

[161] Xolani Dastile, Turgay Çelik, and Hans Vandierendonck. Model-agnostic counterfactual explanations in credit scoring. *IEEE Access*, 10:69543–69554, 2022.

[162] Jiachang Liu, Chudi Zhong, Boxuan Li, Margo I. Seltzer, and Cynthia Rudin. FasterRisk: Fast and accurate interpretable risk scores. In *NeurIPS*, 2022.

[163] Joaquin Quiñonero-Candela, Masashi Sugiyama, Neil D Lawrence, and Anton Schwaighofer, editors. *Dataset shift in machine learning*. The MIT Press, 2009.

[164] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai-Long Nguyen, and François Petitjean. Characterizing concept drift. *Data Min. Knowl. Discov.*, 30 (4):964–994, 2016.

[165] Wei Wang, Christopher Lesner, Alexander Ran, Marko Rukonic, Jason Xue, and Eric Shiu. Using small business banking data for explainable credit risk scoring. In *AAAI*, pages 13396–13401. AAAI Press, 2020.

[166] Luca Barbaglia, Sebastiano Manzan, and Elisa Tosetti. Forecasting loan default in Europe with Machine Learning. *Journal of Financial Econometrics*, 21(2):569–596, 2021.

[167] Thomas Verbraken, Cristián Bravo, Richard Weber, and Bart Baesens. Development and application of consumer credit scoring models using profit-based classification measures. *Eur. J. Oper. Res.*, 238(2):505–513, 2014.

[168] Wouter Verbeke, Bart Baesens, and Cristián Bravo. *Profit driven business analytics: A practitioner's guide to transforming big data into added value*. John Wiley & Sons, 2017.

[169] Martin Rezac and Frantisek Rezac. How to Measure the Quality of Credit Scoring Models. *Czech Journal of Economics and Finance*, 61(5): 486–507, 2011.

[170] Tianbao Yang and Yiming Ying. AUC maximization in the era of big data and AI: A survey. *ACM Comput. Surv.*, 55(8):172:1–172:37, 2023.

[171] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

[172] Dirk Tasche. A traffic lights approach to PD validation. *arXiv preprint cond-mat/0305038*, 2003.

[173] International Financial Reporting Standard. IFRS 16 Leases, 2016. `https://www.ifrs.org/issued-standards/list-of-standards/ifrs-16-leases/`.

[174] Christoph Schröer, Felix Kruse, and Jorge Marx Gómez. A systematic literature review on applying CRISP-DM process model. *Procedia Computer Science*, 181:526–534, 2021.

[175] Lorenzo Bertino. The standardisation of the leasing contract under Italian law and the Unidroit principles. *European Business Law Review*, 30 (5):841–852, 2019.

[176] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *NeurIPS*, pages 6639–6649, 2018.

[177] David Stutz, Abhijit Guha Roy, Tatiana Matejovicova, Patricia Strachan, Ali Taylan Cemgil, and Arnaud Doucet. Conformal prediction under ambiguous ground truth. *CoRR*, abs/2307.09302, 2023.

[178] Matthias Nagl, Maximilian Nagl, and Daniel Rösch. Quantifying uncertainty of machine learning methods for loss given default. *Frontiers in Applied Mathematics and Statistics*, 8:1076083, 2022.

[179] Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex J. DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.*, 2(1):56–67, 2020.

[180] Riccardo Guidotti and Salvatore Ruggieri. Ensemble of counterfactual explainers. In *DS*, volume 12986 of *Lecture Notes in Computer Science*, pages 358–368. Springer, 2021.

[181] Amal Ben Soussia, Chahrazed Labba, Azim Roussanaly, and Anne Boyer. Time-dependent metrics to assess performance prediction systems. *The Int. J. of Inf. and Learning Tech.*, 39(5):451–465, 2022.

[182] G. Castermans, David Martens, Tony Van Gestel, Bart Hamers, and Bart Baesens. An overview and framework for PD backtesting and benchmarking. *J. Oper. Res. Soc.*, 61(3):359–373, 2010.

[183] Philippe du Jardin. The influence of variable selection methods on the accuracy of bankruptcy prediction models. *Bankers, Markets & Investors*, 116:20–39, 2012.

[184] Mirko Moscatelli, Fabio Parlapiano, Simone Narizzano, and Gianluca Viggiano. Corporate default forecasting with machine learning. *Expert Syst. Appl.*, 161:113567, 2020.

[185] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 8th edition, 2017.

[186] Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *SIGKDD Explor.*, 3(1):27–32, 2001.

[187] Jingxian Liao, Wei Wang, Jason Xue, Anthony Lei, Xue Han, and Kun Lu. Combating sampling bias: A self-training method in credit risk models. In *AAAI*, pages 12566–12572. AAAI Press, 2022.

[188] Roger M. Stein. Benchmarking default prediction models: pitfalls and remedies in model validation. *Journal of Risk Model Validation*, 1:77113, 2007.

[189] Sudhir Varma and Richard Simon. Bias in error estimation when using cross-validation for model selection. *BMC Bioinform.*, 7:91, 2006.

[190] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *KDD*, pages 2623–2631. ACM, 2019.

[191] Adrien Ehrhardt, Christophe Biernacki, Vincent Vandewalle, Philippe Heinrich, and Sébastien Beben. Reject inference methods in credit scoring. *Journal of Applied Statistics*, 48(13-15):2734–2754, 2021.

[192] Nikita Kozodoi, Panagiotis Katsas, Stefan Lessmann, Luís Moreira-Matias, and Konstantinos Papakonstantinou. Shallow self-learning for reject inference in credit scoring. In *ECML/PKDD*, volume 11908 of *LNCS*, pages 516–532. Springer, 2019.

[193] Andrea Pugnana, Lorenzo Perini, Jesse Davis, and Salvatore Ruggieri. Deep neural network benchmarks for selective classification. In *Verifying Learning AI Systems (Verilearn) Workshop, ECAI*, 2023.

[194] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *CoRR*, abs/2110.14795, 2021.

[195] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

[196] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. In *NeurIPS*, 2020.

[197] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145. Morgan Kaufmann, 1995.

[198] Lorenzo Perini, Vincent Vercruyssen, and Jesse Davis. Class prior estimation in active positive and unlabeled learning. In *IJCAI*, pages 2915–2921. ijcai.org, 2020.

[199] Eleonora Misino, Giuseppe Marra, and Emanuele Sansone. VAEL: bridging variational autoencoders and probabilistic logic programming. In *NeurIPS*, 2022.

[200] Sara Magliacane, Thijs van Ommen, Tom Claassen, Stephan Bongers, Philip Versteeg, and Joris M. Mooij. Domain adaptation by using causal inference to predict invariant conditional distributions. In *NeurIPS*, pages 10869–10879, 2018.

[201] Yo Joong Choe, Aditya Gangrade, and Aaditya Ramdas. Counterfactually comparing abstaining classifiers. In *NeurIPS*, 2023.

[202] Filipe Condessa, José M. Bioucas-Dias, and Jelena Kovacevic. Performance measures for classification systems with rejection. *Pattern Recognit.*, 63:437–450, 2017.

[203] Aditya Gangrade, Anil Kag, Ashok Cutkosky, and Venkatesh Saligrama. Online selective classification with limited feedback. In *NeurIPS*, pages 14529–14541, 2021.

[204] Juan C. Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 7599–7609. PMLR, 2020.

[205] Cecilia Panigutti, Andrea Beretta, Daniele Fadda, Fosca Giannotti, Dino Pedreschi, Alan Perotti, and Salvatore Rinzivillo. Co-design of human-centered, explainable AI for clinical decision support. *ACM Trans. Interact. Intell. Syst.*, 13(4):21:1–21:35, 2023.

[206] Andrea Esuli, Alessandro Fabris, Alejandro Moreo, and Fabrizio Sebastiani. *Learning to Quantify*, volume 47 of *The Information Retrieval Series*. Springer, 2023.

[207] Roger Koenker. *Quantile regression*, volume 38. Cambridge University Press, 2005.

[208] Willem Waegeman, Krzysztof Dembczynski, and Eyke Hüllermeier. Multi-target prediction: a unifying view on problems and methods. *Data Min. Knowl. Discov.*, 33(2):293–324, 2019.

[209] Ignazio Pillai, Giorgio Fumera, and Fabio Roli. Multi-label classification with a reject option. *Pattern Recognit.*, 46(8):2256–2266, 2013.

[210] Vu-Linh Nguyen and Eyke Hüllermeier. Multilabel classification with partial abstention: Bayes-optimal prediction under label independence. *J. Artif. Intell. Res.*, 72:613–665, 2021.

[211] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Comput. Surv.*, 54(11s):235:1–235:37, 2022.

[212] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[213] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Dan Hendrycks, Yixuan Li, and Ziwei Liu. Openood: Benchmarking generalized out-of-distribution detection. In *NeurIPS*, 2022.

Appendix A

Appendix to Model-Agnostic Heuristics for Selective Regression

A.1 Data Details

We provide in Tables A.1-A.3 the set of data employed in our analysis. We divide them depending on the original benchmark. We report the name of the dataset, the link to the original repository, the number of training instances, and the number of features employed in the regression task.

Table A.1: Data from `PMLB` benchmark [120]

Dataset	Link	Training Size	Feature Space Dim.
1028_SWD	https://epistasislab.github.io/pmlb/profile/1028_SWD.html	700	10
1029_LEV	https://epistasislab.github.io/pmlb/profile/1029_LEV.html	700	4
1030_ERA	https://epistasislab.github.io/pmlb/profile/1030_ERA.html	700	4
1193_BNG_lowbwt	https://epistasislab.github.io/pmlb/profile/1193_BNG_lowbwt.html	21,772	9
1199_BNG_echoMonths	https://epistasislab.github.io/pmlb/profile/1199_BNG_echoMonths.html	12,246	9
197_cpu_act	https://epistasislab.github.io/pmlb/profile/197_cpu_act.html	5,733	21
215_2dplanes	https://epistasislab.github.io/pmlb/profile/215_2dplanes.html	28,537	10
225_puma8NH	https://epistasislab.github.io/pmlb/profile/225_puma8NH.html	5,733	8
227_cpu_small	https://epistasislab.github.io/pmlb/profile/227_cpu_small.html	5,733	12
294_satellite_image	https://epistasislab.github.io/pmlb/profile/294_satellite_image.html	4,504	36
344_mv	https://epistasislab.github.io/pmlb/profile/344_mv.html	28,537	10
4544_GeographicalOriginalofMusic	https://epistasislab.github.io/pmlb/profile/4544_GeographicalOriginalofMusic.html	741	117
503_wind	https://epistasislab.github.io/pmlb/profile/503_wind.html	4,601	14
529_pollen	https://epistasislab.github.io/pmlb/profile/529_pollen.html	2,693	4
564_fried	https://epistasislab.github.io/pmlb/profile/564_fried.html	28,537	10

A.2 Results for Mean Absolute Error

We define Mean Absolute Error (MAE) as:

$$MAE = \mathbb{E}\left[\mid Y - f(\mathbf{X}) \mid\right].$$

We consider the percentage decrease of Mean Absolute Error as the performance metric, namely

$$\Delta MAE(h, g, c) := \frac{MAE(h, g, c)}{MAE(h, g, 1)} - 1,$$

Table A.2: Data from Grinsztajn et al. benchmark [103]

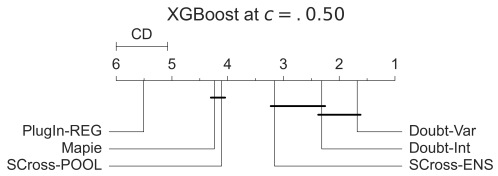
Dataset	Link	Training Size	Feature Space Dim.
Ailerons	https://www.openml.org/search?type=data&status=active&id=44137	9,625	33
Airlines_DepDelay_1M	https://www.openml.org/search?type=data&status=active&id=45047	50,000	5
Allstate_Claims_Severity	https://www.openml.org/search?type=data&status=active&id=45046	50,000	124
Bike_Sharing_Demand	https://www.openml.org/search?type=data&status=active&id=44063	12,165	6
Brazilian_houses	https://www.openml.org/search?type=data&status=active&id=44062	7,483	8
Mercedes_Benz_Greener_Manufacturing	https://www.openml.org/search?type=data&status=active&id=44061	2,946	359
MiamiHousing2016	https://www.openml.org/search?type=data&status=active&id=44147	9,751	13
SGEMM_GPU_kernel_performance	https://www.openml.org/search?type=data&status=active&id=44069	50,000	9
abalone	https://www.openml.org/search?type=data&status=active&id=45042	2,923	7
analcatdata_supreme	https://www.openml.org/search?type=data&status=active&id=44055	2,835	7
delays_zurich_transport	https://www.openml.org/search?type=data&status=active&id=45045	50,000	8
delays_zurich_transport	https://www.openml.org/search?type=data&status=active&id=45045	50,000	11
diamonds	https://www.openml.org/search?type=data&status=active&id=44059	37,758	6
elevators	https://www.openml.org/search?type=data&status=active&id=44134	11,619	16
house_16H	https://www.openml.org/search?type=data&status=active&id=44139	15,948	16
house_sales	https://www.openml.org/search?type=data&status=active&id=44066	15,128	15
houses	https://www.openml.org/search?type=data&status=active&id=44138	14,448	8
medical_charges	https://www.openml.org/search?type=data&status=active&id=45048	50,000	3
nyc-taxi-green-dec-2016	https://www.openml.org/search?type=data&status=active&id=44065	50,000	9
particulate-matter-ukair-2017	https://www.openml.org/search?type=data&status=active&id=44068	50,000	6
pol	https://www.openml.org/search?type=data&status=active&id=44133	10,500	26
seattlecrime6	https://www.openml.org/search?type=data&status=active&id=45043	36,421	4
sulfur	https://www.openml.org/search?type=data&status=active&id=44145	7,056	6
superconduct	https://www.openml.org/search?type=data&status=active&id=44148	14,883	79
topo_2_1	https://www.openml.org/search?type=data&status=active&id=45041	6,219	255
visualizing_soil	https://www.openml.org/search?type=data&status=active&id=44056	6,048	4
wine_quality	https://www.openml.org/search?type=data&status=active&id=44136	4,547	11
yprop_4_1	https://www.openml.org/search?type=data&status=active&id=45032	6,219	42

where $MAE(h, g, c)$ is the Mean Absolute Error computed over accepted instances at desired coverage c and $MAE(h, g, 1)$ is the Mean Absolute Error on the full sample. Similarly to ΔMSE, the lower ΔMAE, the more the model can improve its performance once we allow abstention. Looking at ΔMAE, provides a measure robust to heteroskedasticity in the data.

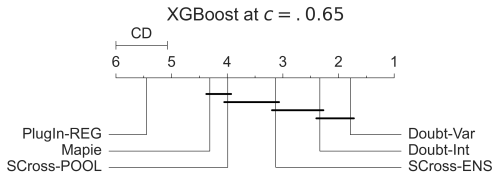
Results are provided in Figures A.1-A.6 Overall, we observe trends similar to those presented in the main section, with bootstrap-based methods outperforming other baselines for most regressors.

Table A.3: Data from `folktables` [121]

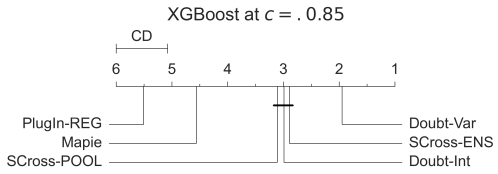
Dataset	Link	Training Size	Feature Space Dim.
FOLKTABLES_US_2018_INCOME_AK	https://github.com/socialfoundations/folktables	2,481	10
FOLKTABLES_US_2018_INCOME_AL	https://github.com/socialfoundations/folktables	15,587	10
FOLKTABLES_US_2018_INCOME_AR	https://github.com/socialfoundations/folktables	9,750	10
FOLKTABLES_US_2018_INCOME_AZ	https://github.com/socialfoundations/folktables	23,293	10
FOLKTABLES_US_2018_INCOME_CA	https://github.com/socialfoundations/folktables	50,000	10
FOLKTABLES_US_2018_INCOME_CO	https://github.com/socialfoundations/folktables	21,913	10
FOLKTABLES_US_2018_INCOME_CT	https://github.com/socialfoundations/folktables	13,849	10
FOLKTABLES_US_2018_INCOME_DE	https://github.com/socialfoundations/folktables	3,298	10
FOLKTABLES_US_2018_INCOME_FL	https://github.com/socialfoundations/folktables	50,000	10
FOLKTABLES_US_2018_INCOME_GA	https://github.com/socialfoundations/folktables	35,640	10
FOLKTABLES_US_2018_INCOME_HI	https://github.com/socialfoundations/folktables	5,411	10
FOLKTABLES_US_2018_INCOME_IA	https://github.com/socialfoundations/folktables	12,421	10
FOLKTABLES_US_2018_INCOME_ID	https://github.com/socialfoundations/folktables	5,785	10
FOLKTABLES_US_2018_INCOME_IL	https://github.com/socialfoundations/folktables	46,910	10
FOLKTABLES_US_2018_INCOME_IN	https://github.com/socialfoundations/folktables	24,514	10
FOLKTABLES_US_2018_INCOME_KS	https://github.com/socialfoundations/folktables	11,064	10
FOLKTABLES_US_2018_INCOME_KY	https://github.com/socialfoundations/folktables	15,403	10
FOLKTABLES_US_2018_INCOME_LA	https://github.com/socialfoundations/folktables	14,466	10
FOLKTABLES_US_2018_INCOME_MA	https://github.com/socialfoundations/folktables	28,079	10
FOLKTABLES_US_2018_INCOME_MD	https://github.com/socialfoundations/folktables	23,128	10
FOLKTABLES_US_2018_INCOME_ME	https://github.com/socialfoundations/folktables	4,900	10
FOLKTABLES_US_2018_INCOME_MI	https://github.com/socialfoundations/folktables	35,005	10
FOLKTABLES_US_2018_INCOME_MN	https://github.com/socialfoundations/folktables	21,714	10
FOLKTABLES_US_2018_INCOME_MO	https://github.com/socialfoundations/folktables	22,164	10
FOLKTABLES_US_2018_INCOME_MS	https://github.com/socialfoundations/folktables	9,232	10
FOLKTABLES_US_2018_INCOME_MT	https://github.com/socialfoundations/folktables	3,823	10
FOLKTABLES_US_2018_INCOME_NC	https://github.com/socialfoundations/folktables	36,446	10
FOLKTABLES_US_2018_INCOME_ND	https://github.com/socialfoundations/folktables	3,118	10
FOLKTABLES_US_2018_INCOME_NE	https://github.com/socialfoundations/folktables	7,549	10
FOLKTABLES_US_2018_INCOME_NH	https://github.com/socialfoundations/folktables	5,575	10
FOLKTABLES_US_2018_INCOME_NJ	https://github.com/socialfoundations/folktables	33,446	10
FOLKTABLES_US_2018_INCOME_NM	https://github.com/socialfoundations/folktables	6,097	10
FOLKTABLES_US_2018_INCOME_NV	https://github.com/socialfoundations/folktables	10,364	10
FOLKTABLES_US_2018_INCOME_NY	https://github.com/socialfoundations/folktables	50,000	10
FOLKTABLES_US_2018_INCOME_OH	https://github.com/socialfoundations/folktables	43,494	10
FOLKTABLES_US_2018_INCOME_OK	https://github.com/socialfoundations/folktables	12,541	10
FOLKTABLES_US_2018_INCOME_OR	https://github.com/socialfoundations/folktables	15,343	10
FOLKTABLES_US_2018_INCOME_PA	https://github.com/socialfoundations/folktables	47,815	10
FOLKTABLES_US_2018_INCOME_PR	https://github.com/socialfoundations/folktables	6,349	10
FOLKTABLES_US_2018_INCOME_RI	https://github.com/socialfoundations/folktables	3,997	10
FOLKTABLES_US_2018_INCOME_SC	https://github.com/socialfoundations/folktables	17,415	10
FOLKTABLES_US_2018_INCOME_SD	https://github.com/socialfoundations/folktables	3,429	10
FOLKTABLES_US_2018_INCOME_TN	https://github.com/socialfoundations/folktables	23,801	10
FOLKTABLES_US_2018_INCOME_TX	https://github.com/socialfoundations/folktables	50,000	10
FOLKTABLES_US_2018_INCOME_UT	https://github.com/socialfoundations/folktables	11,435	10
FOLKTABLES_US_2018_INCOME_VA	https://github.com/socialfoundations/folktables	32,300	10
FOLKTABLES_US_2018_INCOME_VT	https://github.com/socialfoundations/folktables	2,636	10
FOLKTABLES_US_2018_INCOME_WA	https://github.com/socialfoundations/folktables	27,960	10
FOLKTABLES_US_2018_INCOME_WI	https://github.com/socialfoundations/folktables	22,883	10
FOLKTABLES_US_2018_INCOME_WV	https://github.com/socialfoundations/folktables	5,671	10
FOLKTABLES_US_2018_INCOME_WY	https://github.com/socialfoundations/folktables	2,144	10



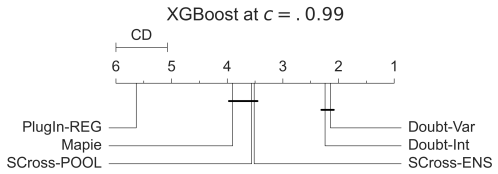
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$

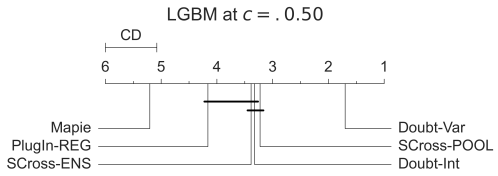


(c) CD plot at $c = .85$

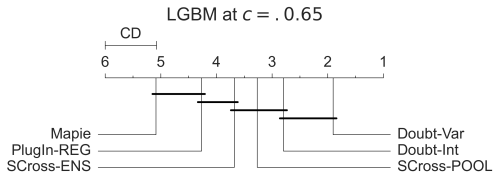


(d) CD plot at $c = .99$

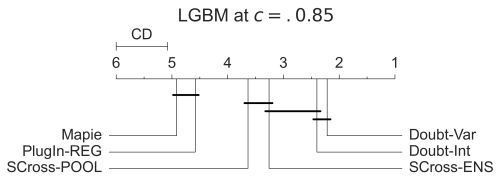
Figure A.1: Q1: CD plots of ΔMAE for different target coverages. The closer to one, the better the method.



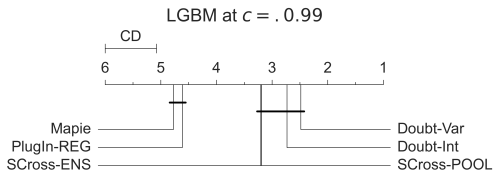
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$

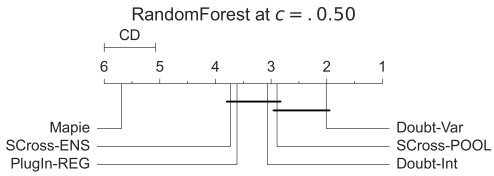


(c) CD plot at $c = .85$

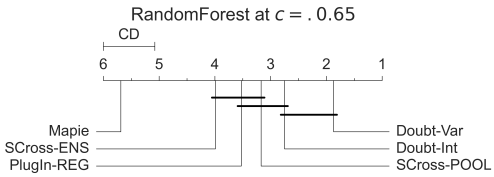


(d) CD plot at $c = .99$

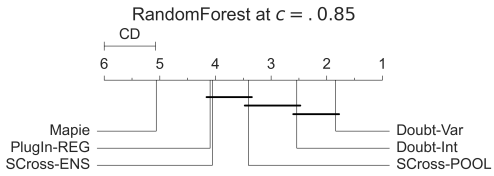
Figure A.2: Q1: CD plots of ΔMAE for different target coverages, using LightGBM. The closer to one, the better the method.



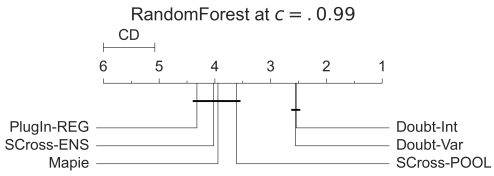
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$

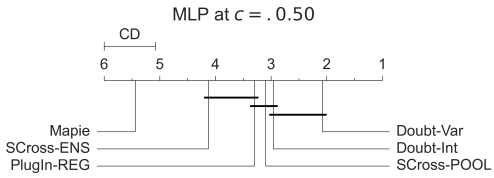


(c) CD plot at $c = .85$

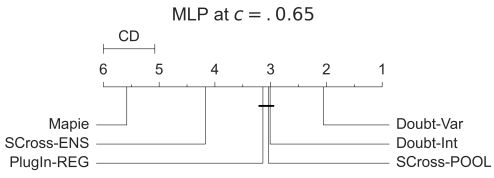


(d) CD plot at $c = .99$

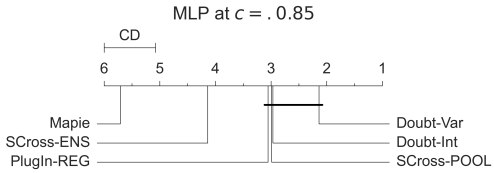
Figure A.3: Q1: CD plots of ΔMAE for different target coverages, using RandomForest. The closer to one, the better the method.



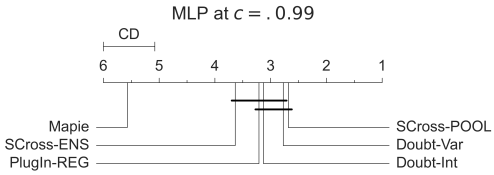
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$

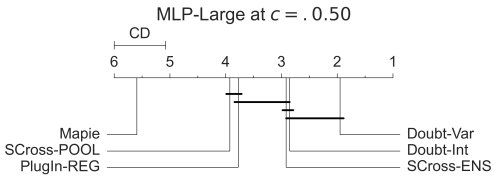


(c) CD plot at $c = .85$

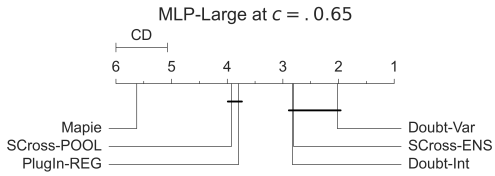


(d) CD plot at $c = .99$

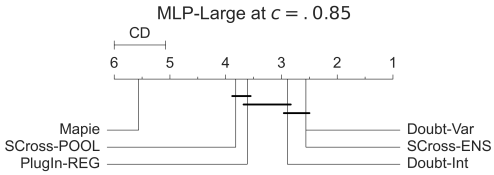
Figure A.4: Q1: CD plots of ΔMAE for different target coverages, using MLP. The closer to one, the better the method.



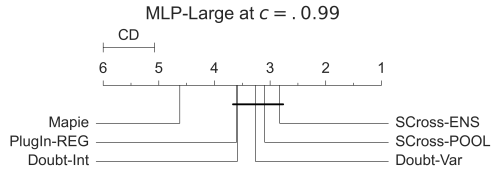
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$

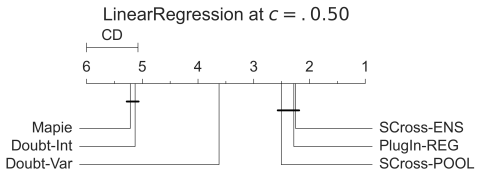


(c) CD plot at $c = .85$

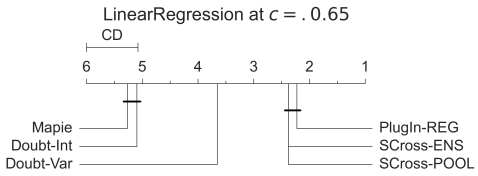


(d) CD plot at $c = .99$

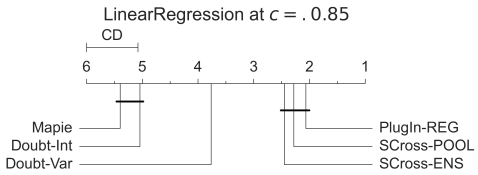
Figure A.5: Q1: CD plots of ΔMAE for different target coverages, using MLP-Large. The closer to one, the better the method.



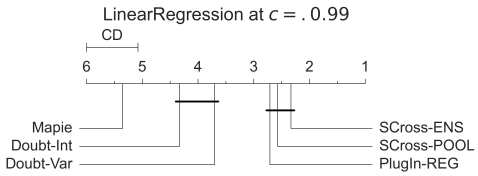
(a) CD plot at $c = .50$



(b) CD plot at $c = .65$



(c) CD plot at $c = .85$



(d) CD plot at $c = .99$

Figure A.6: Q1: CD plots of ΔMAE for different target coverages, using LinearRegression. The closer to one, the better the method.

Appendix B

Appendix to Selective Classification for AUC

B.1 Proofs

We detail here the missing part of the proof for **Proposition 1**.

Proof. Consider the following equivalence:

$$
\begin{aligned}
\bar{G} > G \quad & \text{iff} \quad \bar{A}(n-1) > An \\
& \text{iff} \quad \{ \text{ by (5.10) } \} \\
& \text{iff} \quad ((A + \frac{1}{2} - \left[\frac{(r'(\mathbf{x}_i) - 1) + (t(\mathbf{x}_i) - 1) + 1/2}{n^+ n}\right]) \\
& \qquad \frac{n^+ n}{(n^+ - 1)(n-1)} - \frac{1}{2})(n-1) > An \\
& \text{iff} \quad (A + \frac{1}{2} - \frac{(r'(\mathbf{x}_i) + t(\mathbf{x}_i) - 1.5)}{n^+ n})\frac{n^+ n}{(n^+ - 1)} - \frac{(n-1)}{2} > An \\
& \text{iff} \quad A\frac{n^+ n}{(n^+ - 1)} + \frac{n^+ n}{2(n^+ - 1)} - \frac{(r'(\mathbf{x}_i) + t(\mathbf{x}_i) - 1.5)}{(n^+ - 1)} - \frac{(n-1)}{2} > An \\
& \text{iff} \quad A\frac{n}{(n^+ - 1)} + \frac{n^+ n}{2(n^+ - 1)} - \frac{(r'(\mathbf{x}_i) + t(\mathbf{x}_i) - 1.5)}{(n^+ - 1)} - \frac{(n-1)}{2} > 0 \\
& \text{iff} \quad A\frac{n}{(n^+ - 1)} + \frac{n^+ n - (n-1)(n^+ - 1)}{2(n^+ - 1)} - \frac{(r'(\mathbf{x}_i) + t(\mathbf{x}_i) - 1.5)}{(n^+ - 1)} > 0 \\
& \text{iff} \quad An + \frac{n + n^+ - 1}{2} - (r'(\mathbf{x}_i) + t(\mathbf{x}_i) - 1.5) > 0 \\
& \text{iff} \quad An + \frac{n + n^+}{2} - t(\mathbf{x}_i) > r'(\mathbf{x}_i) - 1 \\
& \text{iff} \quad An + \frac{n + n^+}{2} - t(\mathbf{x}_i) \geq r'(\mathbf{x}_i) \\
& \text{iff} \quad An + \frac{n - n^+}{2} + (n^+ - t(\mathbf{x}_i)) \geq r'(\mathbf{x}_i) \\
& \text{iff} \quad \frac{r'(\mathbf{x}_i)}{n} \leq A + \frac{p^-}{2} + \frac{(n^+ - t(\mathbf{x}_i))}{n}
\end{aligned}
$$

□

B.2 Experimental Evaluation

B.2.1 Datasets description

`Adult` is an extract from the 1994 US Census, with class the binarization of income into $\leq 50K$ and $> 50K$. The final training set contains 30,162 instances and 55 features after one-hot encoding. The test set size is 15,060.
`LendingClub` regards repaying a loan obtained by an online platform. We used a temporal split to build the final training set (1,364,697 instances) and the test set (445,912 instances). The dataset has 65 features.
The `GiveMe` dataset aims at predicting the financial distress of a borrower within two years. The training and the test set were obtained by stratified random sampling, and they contain 12 features, and 112,500 and 37,500 instances respectively.
`UCICredit` regards credit card defaults in Taiwan. This dataset from [107] concerns whether or not a credit card holder will default in the next six months [109]. 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.
`CSDS1`, `CSDS2` and `CSDS3` - from [108] - regard predicting 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). Training set and test set were divided through a timestamp variable. 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).
The `CatsVsDogs` dataset is a collection of cats and dogs images. The task here is to distinguish between the two species. The training and test sets were obtained as described in [16]. The training set contains 20,000 images, each one of 64x64 pixels. The test set consists of 5,000 images.
Finally we considered the image dataset `CIFAR-10-cat` from [212]. For each of the 10 class labels, we have 5,000 32x32 images in the training set and 1,000 in the test set. We transformed it into a binary classification task by using the *cat* label as the positive class.

B.2.2 Baselines details

AUCross and PlugInAUC. In Chapter 5.5.3, we used as base classifier a LightGBM classifier with default parameters. In Tables B.4-B.11 we provide also results for a Logistic Regression and a Random Forest Classifier from `sklearn` package with default parameters, a ResNet implementation with default parameters from `rtdl` [104] and an XGBoost classifier from `xgboost` package with default parameters.

PlugIn and SCross. As for AUCross, we considered a LightGBM classifier with default parameters.

SelNet. We use `pytorch` to model SelNet. For tabular datasets, we build the main body part using ResNet with default parameters provided by `rtdl` [104], as authors claim that ResNet is a valid baseline on tabular data. For images, we use the VGG16 architecture [136] as the main body as done in the original paper [15]. We then add the classification head, the selection head, and the auxiliary head following SelNet paper. For CSDS1, Lending and GiveMe the prediction head and the auxiliary head are made by a first linear layer with 512 nodes followed by a batch normalization layer and ReLu activation; a second layer with 256 nodes, batch normalization and ReLu activation; a final dense layer with 128 nodes ending with two nodes and softmax activation. For the other datasets, the classification and auxiliary heads are made by a single linear layer with 128 nodes and a final softmax activation. We built selective heads using a linear layer with 128 nodes, batch normalization, relu activation, and another 64-node linear layer ending with a single node and Sigmoid activation. All the models are available in the code here. We point out that the lack of a clear design methodology of the SelectiveNet structure for a given dataset is a major drawback of SelNet compared to the flexibility of our model-agnostic method. Models are trained, for every target coverage c, using the same loss function as in [15]. Both parameters λ and α are set as in [15] to $\lambda = 32$ and $\alpha = .5$. The batch size was 512 for Lending; 128 for GiveMe, CSDS1, CSDS2, CSDS3 and image data; 32 for Adult and UCICredit. The learning procedure was run for 300 epochs with 7.4 as loss and it was used as an optimizer Stochastic Gradient Descent setting *learning-rate=.10*, *momentum* = .90, *Nesterov* = `True` and a decay of .5 every 25 epochs as in the original paper. The training was performed over 90% of training set instances while we used the remaining 10% to calibrate the selective head.

SAT. The code was based on the `pytorch` implementation available here. We employed the same batch sizes as for SelNet and we set up training details as in the original paper of SAT: we used the loss in 7.2 for 300 epochs and as an optimizer Stochastic Gradient Descent, setting *learning-rate=.10*, *momentum* = .90,*Nesterov* = `True` and a decay of .5 every 25 epochs. The training was performed over 90% of training set instances while we used the remaining 10% to calibrate the selection function.

B.2.3 Other experimental results

We report results for selective accuracy (Table B.1) and positive rate (Table B.2). As discussed in the main paper, AUCross and PlugInAUC do not guarantee improvements in terms of accuracy whenever the target coverage decreases. Interestingly, they are able to maintain the positive rate more stable than the compared approaches. Finally, we report training times for all the methods in Table B.3. PlugInAUC and PlugIn are clear winners over tab-

ular datases as they can exploit fast classifiers. Both AUCross and SCross pay a factor proportional to the number of folds K used in the cross-fitting part of the algorithm. This extra cost can be potentially mitigated on tabular datasets by parallelizing the cross-fitting procedure. Finally, notice that SelNet is the only approach which require a separate run for each target coverage c.

B.2.4 Results for different classifiers

We report in Tables B.4-B.11 the results for AUCross and PlugInAUC using different classifiers over tabular datasets. Regarding coverage, we see the harshest violations for both AUCross-RandForest and PlugInAUC-RandForest over *CSDS2* and for AUCross-ResNet over *GiveMe*. Coverage violations occur also for *Lending* dataset, independently of the classifier. Regarding the AUC, both AUCross and PlugInAUC succeed in increasing it while target coverage drops, regardless of the considered base classifier. At the same time, we notice that for all the base classifiers lowering coverage does not guarantee selective accuracy to increase, highlighting once more the trade-off between these two metrics. Finally, we see similar results across all the classifiers for positive rate.

B.2.5 Results varying number of folds

We report empirical coverage and selective AUC for different choices of the parameter K in Table B.12 and in Table B.13, respectively. The default value $K = 5$ shows a slightly better trade-off between empirical coverage and selective AUC.

Table B.1: Selective accuracy (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Selective Accuracy					
	c	**AUCross**	**PlugIn**	**PlugInAUC**	**SCross**	**SelNet**	**SAT**
Adult	.99	.870 ± .003	**.872 ± .003**	.870 ± .003	.871 ± .003	.848 ± .003	.845 ± .003
	.95	.875 ± .003	**.888 ± .003**	.874 ± .003	**.888 ± .003**	.858 ± .003	.845 ± .003
	.90	.882 ± .003	**.903 ± .003**	.880 ± .003	.902 ± .003	.872 ± .003	.845 ± .003
	.85	.888 ± .003	**.920 ± .003**	.887 ± .003	.919 ± .003	.886 ± .003	.846 ± .003
	.80	.899 ± .003	**.936 ± .003**	.897 ± .003	.934 ± .003	.901 ± .003	.846 ± .003
	.75	.907 ± .003	**.950 ± .003**	.905 ± .003	**.950 ± .003**	.908 ± .003	.845 ± .004
Lending	.99	**.899 ± .001**	**.899 ± .001**	**.899 ± .001**	**.899 ± .001**	.870 ± .001	.754 ± .003
	.95	.908 ± .001	.909 ± .001	.909 ± .001	**.910 ± .001**	.873 ± .001	.754 ± .003
	.90	.921 ± .001	.923 ± .001	.921 ± .001	**.924 ± .001**	.912 ± .001	.754 ± .003
	.85	.934 ± .001	.937 ± .001	.934 ± .001	**.938 ± .001**	.885 ± .001	.754 ± .003
	.80	.948 ± .001	**.951 ± .001**	.949 ± .001	**.951 ± .001**	.926 ± .001	.754 ± .003
	.75	.964 ± .001	.963 ± .001	**.965 ± .001**	.963 ± .001	.924 ± .001	.754 ± .003
GiveMe	.99	.938 ± .002	.942 ± .002	.938 ± .002	.942 ± .002	**.954 ± .002**	.916 ± .002
	.95	.937 ± .002	**.956 ± .002**	.937 ± .002	**.956 ± .002**	.954 ± .002	.915 ± .002
	.90	.938 ± .002	**.967 ± .001**	.937 ± .002	**.967 ± .001**	.961 ± .001	.913 ± .002
	.85	.939 ± .002	**.973 ± .001**	.938 ± .002	**.973 ± .001**	.967 ± .001	.913 ± .002
	.80	.939 ± .002	.977 ± .001	.938 ± .002	**.978 ± .001**	.969 ± .001	.912 ± .002
	.75	.939 ± .002	.980 ± .001	.939 ± .002	**.981 ± .001**	.971 ± .001	.912 ± .002
UCICredit	.99	.811 ± .005	.814 ± .005	.812 ± .005	.814 ± .005	**.815 ± .005**	.812 ± .005
	.95	.810 ± .005	.826 ± .005	.811 ± .005	**.827 ± .005**	.824 ± .005	.812 ± .005
	.90	.809 ± .005	.838 ± .005	.811 ± .005	**.839 ± .005**	.830 ± .005	.813 ± .005
	.85	.806 ± .005	.849 ± .005	.811 ± .005	**.855 ± .005**	.845 ± .005	.813 ± .005
	.80	.806 ± .006	.863 ± .005	.810 ± .006	**.867 ± .005**	.858 ± .005	.813 ± .005
	.75	.806 ± .006	.872 ± .005	.808 ± .006	**.875 ± .005**	.871 ± .005	.813 ± .005
CSDS1	.99	.857 ± .002	**.863 ± .002**	.857 ± .002	**.863 ± .002**	.861 ± .002	.862 ± .002
	.95	.856 ± .002	**.875 ± .002**	.856 ± .002	**.875 ± .002**	.873 ± .002	.871 ± .002
	.90	.854 ± .002	**.885 ± .002**	.855 ± .002	**.885 ± .002**	.882 ± .002	.883 ± .002
	.85	.853 ± .002	**.892 ± .002**	.853 ± .002	**.892 ± .002**	.890 ± .002	.890 ± .002
	.80	.852 ± .002	.898 ± .002	.852 ± .002	**.899 ± .002**	.897 ± .002	.897 ± .002
	.75	.850 ± .002	**.904 ± .002**	.850 ± .002	**.904 ± .002**	.902 ± .002	.902 ± .002
CSDS2	.99	.982 ± .002	.982 ± .002	.982 ± .002	.982 ± .002	**.983 ± .002**	.982 ± .002
	.95	.982 ± .002	.983 ± .002	.982 ± .002	.985 ± .002	**.984 ± .002**	.982 ± .002
	.90	.982 ± .002	.984 ± .002	.982 ± .002	.985 ± .002	**.986 ± .002**	.985 ± .002
	.85	.982 ± .002	.985 ± .002	.981 ± .002	.986 ± .002	**.986 ± .002**	.985 ± .002
	.80	.982 ± .002	.986 ± .002	.981 ± .002	.987 ± .002	**.987 ± .002**	.986 ± .002
	.75	.981 ± .002	.986 ± .002	.981 ± .002	.987 ± .002	**.988 ± .002**	.987 ± .002
CSDS3	.99	.814 ± .003	**.816 ± .003**	.813 ± .003	**.816 ± .003**	.810 ± .003	.809 ± .003
	.95	.817 ± .003	.826 ± .003	.816 ± .003	**.827 ± .003**	.822 ± .003	.820 ± .003
	.90	.820 ± .003	**.841 ± .003**	.819 ± .003	**.841 ± .003**	.830 ± .003	.835 ± .003
	.85	.825 ± .003	**.856 ± .003**	.823 ± .003	.855 ± .003	.844 ± .003	.850 ± .003
	.80	.829 ± .003	**.871 ± .003**	.826 ± .003	.869 ± .003	.860 ± .003	.861 ± .003
	.75	.835 ± .003	**.884 ± .003**	.830 ± .003	.883 ± .003	.871 ± .003	.872 ± .003
CatsVsDogs	.99	**.954 ± .004**	.950 ± .004	.944 ± .004	.956 ± .003	.941 ± .004	.950 ± .004
	.95	**.968 ± .003**	.949 ± .004	.953 ± .004	.969 ± .003	.964 ± .003	.964 ± .003
	.90	.977 ± .003	.947 ± .004	.971 ± .003	.979 ± .003	**.978 ± .003**	.977 ± .003
	.85	**.984 ± .002**	.944 ± .004	.982 ± .003	.985 ± .002	.983 ± .002	**.984 ± .002**
	.80	.987 ± .002	.942 ± .004	.986 ± .002	.989 ± .002	**.990 ± .002**	.985 ± .002
	.75	.988 ± .002	.938 ± .004	.988 ± .002	.990 ± .002	**.990 ± .002**	.987 ± .002
CIFAR10-CAT	.99	.952 ± .003	.952 ± .003	.953 ± .003	.956 ± .002	.952 ± .003	**.958 ± .002**
	.95	.952 ± .003	.952 ± .003	.951 ± .003	.969 ± .002	.967 ± .002	**.969 ± .002**
	.90	.953 ± .003	.950 ± .003	.950 ± .003	**.977 ± .002**	.977 ± .002	.973 ± .002
	.85	.954 ± .003	.947 ± .003	.948 ± .003	**.979 ± .002**	.977 ± .002	.976 ± .002
	.80	.959 ± .003	.945 ± .003	.946 ± .003	**.980 ± .002**	.984 ± .002	.975 ± .002
	.75	.971 ± .002	.942 ± .003	.943 ± .003	.980 ± .002	**.986 ± .002**	.975 ± .002
	#	4/54	21/54	2/54	29/54	12/54	3/54

Table B.2: Positive rate (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Positive Rate					
	c	**AUCross**	**PlugIn**	**PlugInAUC**	**SCross**	**SelNet**	**SAT**
Adult	.99	**.246 ± .004**	.245 ± .004	**.246 ± .004**	.245 ± .004	.242 ± .004	**.246 ± .004**
	.95	**.246 ± .004**	.234 ± .004	.247 ± .004	.234 ± .004	.235 ± .004	.247 ± .004
	.90	**.246 ± .004**	.221 ± .004	.248 ± .004	.220 ± .004	.225 ± .004	.249 ± .004
	.85	**.247 ± .004**	.203 ± .004	.249 ± .004	.206 ± .004	.115 ± .003	.250 ± .004
	.80	**.245 ± .004**	.185 ± .004	.249 ± .004	.190 ± .004	.180 ± .004	.251 ± .004
	.75	**.245 ± .005**	.169 ± .004	.250 ± .005	.170 ± .004	.149 ± .004	.252 ± .004
Lending	.99	**.224 ± .001**	.222 ± .001	.224 ± .001	**.222 ± .001**	.223 ± .001	.249 ± .003
	.95	**.217 ± .001**	.206 ± .001	**.217 ± .001**	.205 ± .001	.215 ± .001	.249 ± .003
	.90	**.207 ± .001**	.184 ± .001	**.207 ± .001**	.183 ± .001	.191 ± .001	.249 ± .003
	.85	.197 ± .001	.162 ± .001	.197 ± .001	.161 ± .001	.134 ± .001	**.249 ± .003**
	.80	.186 ± .001	.140 ± .001	.186 ± .001	.139 ± .001	.157 ± .001	**.249 ± .003**
	.75	.174 ± .001	.118 ± .001	.174 ± .001	.116 ± .001	.143 ± .001	**.249 ± .003**
GiveMe	.99	.068 ± .002	.063 ± .002	**.067 ± .002**	.063 ± .002	.047 ± .002	.085 ± .002
	.95	**.068 ± .002**	.046 ± .002	**.068 ± .002**	.047 ± .002	.047 ± .002	.086 ± .002
	.90	**.067 ± .002**	.034 ± .001	**.068 ± .002**	.034 ± .001	.040 ± .001	.088 ± .002
	.85	**.067 ± .002**	.028 ± .001	**.068 ± .002**	.028 ± .001	.034 ± .001	.088 ± .002
	.80	**.067 ± .002**	.024 ± .001	**.067 ± .002**	.023 ± .001	.032 ± .001	.089 ± .002
	.75	**.067 ± .002**	.021 ± .001	**.068 ± .002**	.020 ± .001	.030 ± .001	.089 ± .002
UCICredit	.99	**.222 ± .005**	.220 ± .005	**.222 ± .005**	.220 ± .005	.220 ± .005	**.222 ± .005**
	.95	.225 ± .006	.210 ± .005	.224 ± .005	.209 ± .005	.180 ± .005	**.222 ± .005**
	.90	.228 ± .006	.194 ± .005	.226 ± .006	.195 ± .005	.176 ± .005	**.221 ± .005**
	.85	.233 ± .006	.179 ± .005	.229 ± .006	.176 ± .005	.156 ± .005	**.221 ± .005**
	.80	.236 ± .006	.161 ± .005	.232 ± .006	.158 ± .005	.143 ± .005	**.221 ± .005**
	.75	.239 ± .006	.143 ± .005	.237 ± .006	.141 ± .005	.130 ± .005	**.221 ± .005**
CSDS1	.99	.145 ± .002	.139 ± .002	**.144 ± .002**	.138 ± .002	.140 ± .002	.139 ± .002
	.95	**.146 ± .002**	.126 ± .002	**.146 ± .002**	.126 ± .002	.128 ± .002	.130 ± .002
	.90	**.147 ± .002**	.116 ± .002	**.147 ± .002**	.116 ± .002	.119 ± .002	.118 ± .002
	.85	**.149 ± .002**	.109 ± .002	**.149 ± .002**	.109 ± .002	.111 ± .002	.111 ± .002
	.80	**.150 ± .002**	.103 ± .002	**.150 ± .002**	.102 ± .002	.104 ± .002	.104 ± .002
	.75	**.152 ± .002**	.097 ± .002	**.152 ± .002**	.097 ± .002	.099 ± .002	.099 ± .002
CSDS2	.99	**.019 ± .002**	**.019 ± .002**	**.019 ± .002**	**.019 ± .002**	.018 ± .002	**.019 ± .002**
	.95	.019 ± .002	**.018 ± .002**	.019 ± .002	.016 ± .002	.017 ± .002	.019 ± .002
	.90	**.019 ± .002**	.017 ± .002	**.019 ± .002**	.016 ± .002	.015 ± .002	.016 ± .002
	.85	**.019 ± .002**	.016 ± .002	**.019 ± .002**	.015 ± .002	.015 ± .002	.016 ± .002
	.80	**.019 ± .002**	.015 ± .002	**.019 ± .002**	.014 ± .002	.014 ± .002	.015 ± .002
	.75	**.019 ± .002**	.015 ± .002	.020 ± .002	.014 ± .002	.013 ± .002	.014 ± .002
CSDS3	.99	**.253 ± .003**	.252 ± .003	.254 ± .003	.251 ± .003	.251 ± .003	.252 ± .003
	.95	**.254 ± .003**	.242 ± .003	**.254 ± .003**	.242 ± .003	.244 ± .003	.243 ± .003
	.90	**.254 ± .003**	.228 ± .003	.255 ± .003	.229 ± .003	.233 ± .003	.232 ± .003
	.85	**.254 ± .003**	.214 ± .003	.255 ± .003	.215 ± .003	.219 ± .003	.219 ± .003
	.80	**.254 ± .004**	.199 ± .003	.256 ± .004	.200 ± .003	.206 ± .003	.207 ± .003
	.75	**.254 ± .004**	.182 ± .003	.257 ± .004	.185 ± .003	.178 ± .003	.188 ± .003
CatsVsDogs	.99	**.500 ± .008**	**.505 ± .008**	.503 ± .008	**.500 ± .008**	**.500 ± .008**	**.500 ± .008**
	.95	**.498 ± .008**	.514 ± .008	.512 ± .008	.495 ± .008	.494 ± .008	.497 ± .008
	.90	.494 ± .008	.531 ± .008	.518 ± .008	.490 ± .008	.482 ± .008	**.496 ± .008**
	.85	.488 ± .008	.546 ± .008	.524 ± .008	.488 ± .008	.485 ± .008	**.494 ± .008**
	.80	.473 ± .008	.556 ± .008	.528 ± .008	**.493 ± .008**	.476 ± .008	.482 ± .008
	.75	.445 ± .008	.561 ± .009	.529 ± .009	**.500 ± .009**	.472 ± .008	.464 ± .008
CIFAR10-CAT	.99	**.101 ± .004**	.092 ± .003	.102 ± .004	.097 ± .004	.098 ± .004	.096 ± .004
	.95	**.100 ± .004**	.072 ± .003	.106 ± .004	.078 ± .003	.084 ± .003	.075 ± .003
	.90	**.100 ± .004**	.074 ± .003	.111 ± .004	.049 ± .003	.064 ± .003	.044 ± .003
	.85	**.099 ± .004**	.077 ± .003	.115 ± .004	.040 ± .003	.024 ± .002	.025 ± .002
	.80	**.095 ± .004**	.080 ± .004	.122 ± .004	.037 ± .003	.017 ± .002	.026 ± .002
	.75	**.084 ± .003**	**.084 ± .004**	.129 ± .004	.036 ± .003	.015 ± .002	.026 ± .002
	#	39/54	4/54	20/54	5/54	1/54	14/54

Table B.3: Time required for running different methods (in seconds).

		Training Time (seconds)					
	c	**AUCross**	**PlugIn**	**PlugInAUC**	**SCross**	**SelNet**	**SAT**
Adult	.99	1.43	**0.22**	**0.22**	1.15	1787.93	2319.66
	.95	1.43	**0.22**	**0.22**	1.15	1779.74	2319.66
	.90	1.43	**0.22**	**0.22**	1.15	1786.99	2319.66
	.85	1.43	**0.22**	**0.22**	1.15	1781.64	2319.66
	.80	1.43	**0.22**	**0.22**	1.15	1789.44	2319.66
	.75	1.43	**0.22**	**0.22**	1.15	1788.53	2319.66
Lending	.99	13.16	**2.55**	**2.55**	12.10	9168.31	14182.26
	.95	13.16	**2.55**	**2.55**	12.10	9198.68	14182.26
	.90	13.16	**2.55**	**2.55**	12.10	9187.33	14182.26
	.85	13.16	**2.55**	**2.55**	12.10	9204.72	14182.26
	.80	13.16	**2.55**	**2.55**	12.10	9165.67	14182.26
	.75	13.16	**2.55**	**2.55**	12.10	9199.28	14182.26
GiveMe	.99	1.60	**0.27**	**0.27**	1.39	2341.00	3104.83
	.95	1.60	**0.27**	**0.27**	1.39	2323.07	3104.83
	.90	1.60	**0.27**	**0.27**	1.39	2332.69	3104.83
	.85	1.60	**0.27**	**0.27**	1.39	2332.97	3104.83
	.80	1.60	**0.27**	**0.27**	1.39	2326.58	3104.83
	.75	1.60	**0.27**	**0.27**	1.39	2333.06	3104.83
UCICredit	.99	1.26	**0.20**	**0.20**	1.11	1547.44	2070.54
	.95	1.26	**0.20**	**0.20**	1.11	1549.75	2070.54
	.90	1.26	**0.20**	**0.20**	1.11	1527.84	2070.54
	.85	1.26	**0.20**	**0.20**	1.11	1547.15	2070.54
	.80	1.26	**0.20**	**0.20**	1.11	1548.82	2070.54
	.75	1.26	**0.20**	**0.20**	1.11	1541.74	2070.54
CSDS1	.99	5.60	**1.02**	**1.02**	5.31	4815.42	5524.00
	.95	5.60	**1.02**	**1.02**	5.31	4812.96	5524.00
	.90	5.60	**1.02**	**1.02**	5.31	4813.66	5524.00
	.85	5.60	**1.02**	**1.02**	5.31	4817.26	5524.00
	.80	5.60	**1.02**	**1.02**	5.31	4818.25	5524.00
	.75	5.60	**1.02**	**1.02**	5.31	4812.89	5524.00
CSDS2	.99	1.23	**0.21**	**0.21**	1.41	609.99	357.92
	.95	1.23	**0.21**	**0.21**	1.41	609.32	357.92
	.90	1.23	**0.21**	**0.21**	1.41	606.45	357.92
	.85	1.23	**0.21**	**0.21**	1.41	606.47	357.92
	.80	1.23	**0.21**	**0.21**	1.41	610.34	357.92
	.75	1.23	**0.21**	**0.21**	1.41	605.20	357.92
CSDS3	.99	5.07	**0.80**	**0.80**	4.60	1188.31	748.88
	.95	5.07	**0.80**	**0.80**	4.60	1193.35	748.88
	.90	5.07	**0.80**	**0.80**	4.60	1192.50	748.88
	.85	5.07	**0.80**	**0.80**	4.60	1190.79	748.88
	.80	5.07	**0.80**	**0.80**	4.60	1191.03	748.88
	.75	5.07	**0.80**	**0.80**	4.60	1191.56	748.88
CatsVsDogs	.99	14668.55	**2624.68**	**2624.68**	14475.64	2700.46	2735.28
	.95	14668.55	**2624.68**	**2624.68**	14475.64	2699.13	2735.28
	.90	14668.55	**2624.68**	**2624.68**	14475.64	2733.35	2735.28
	.85	14668.55	**2624.68**	**2624.68**	14475.64	2769.44	2735.28
	.80	14668.55	**2624.68**	**2624.68**	14475.64	2766.80	2735.28
	.75	14668.55	**2624.68**	**2624.68**	14475.64	2762.34	2735.28
CIFAR10-CAT	.99	17401.61	**3026.98**	**3026.98**	16924.30	3317.14	3239.50
	.95	17401.61	**3026.98**	**3026.98**	16924.30	3296.70	3239.50
	.90	17401.61	**3026.98**	**3026.98**	16924.30	3337.28	3239.50
	.85	17401.61	**3026.98**	**3026.98**	16924.30	3358.18	3239.50
	.80	17401.61	**3026.98**	**3026.98**	16924.30	3262.57	3239.50
	.75	17401.61	**3026.98**	**3026.98**	16924.30	3294.08	3239.50
	#	0/54	54/54	54/54	0/54	0/54	0/54

Table B.4: Empirical coverage for AUCross using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Empirical Coverage			
	c	**Logistic**	**RandomForest**	**ResNet**	**XGBoost**
Adult	.99	.991 ± .001	.988 ± .001	.994 ± .001	**.990 ± .001**
	.95	**.950 ± .002**	.945 ± .002	.961 ± .002	.951 ± .002
	.90	**.901 ± .003**	.893 ± .003	.912 ± .003	.891 ± .003
	.85	**.850 ± .003**	.853 ± .003	.840 ± .004	.846 ± .003
	.80	**.800 ± .004**	.804 ± .004	.792 ± .004	.793 ± .004
	.75	.747 ± .004	**.752 ± .004**	.735 ± .004	.744 ± .004
Lending	.99	.996 ± .001	**.992 ± .001**	.995 ± .001	.996 ± .001
	.95	.976 ± .001	**.971 ± .001**	.976 ± .001	.980 ± .001
	.90	.951 ± .001	**.945 ± .001**	.951 ± .001	.958 ± .001
	.85	.926 ± .001	**.918 ± .001**	.925 ± .001	.936 ± .001
	.80	.900 ± .001	**.890 ± .001**	.898 ± .001	.912 ± .001
	.75	.871 ± .001	**.856 ± .001**	.868 ± .001	.886 ± .001
GiveMe	.99	**.990 ± .001**	.942 ± .002	.944 ± .002	**.990 ± .001**
	.95	.953 ± .002	.942 ± .002	.755 ± .003	**.951 ± .002**
	.90	.904 ± .002	.858 ± .002	.640 ± .003	**.898 ± .002**
	.85	**.855 ± .002**	.788 ± .003	.570 ± .003	.849 ± .002
	.80	.805 ± .003	.762 ± .003	.499 ± .003	**.799 ± .003**
	.75	.758 ± .003	.741 ± .003	.431 ± .003	**.750 ± .003**
UCICredit	.99	.991 ± .002	.949 ± .003	.987 ± .002	**.990 ± .002**
	.95	**.953 ± .003**	.949 ± .003	.939 ± .003	.947 ± .003
	.90	**.903 ± .004**	.878 ± .004	.873 ± .004	.894 ± .004
	.85	**.853 ± .005**	.827 ± .005	.814 ± .005	.840 ± .005
	.80	**.808 ± .005**	.777 ± .005	.759 ± .005	.784 ± .005
	.75	**.762 ± .005**	.712 ± .006	.706 ± .006	.729 ± .006
CSDS1	.99	**.991 ± .001**	.913 ± .002	**.991 ± .001**	**.991 ± .001**
	.95	**.951 ± .001**	.913 ± .002	.953 ± .001	**.951 ± .001**
	.90	.902 ± .002	.870 ± .002	.905 ± .002	**.901 ± .002**
	.85	.852 ± .002	.822 ± .002	.854 ± .002	**.851 ± .002**
	.80	.803 ± .002	.784 ± .002	.797 ± .002	**.799 ± .002**
	.75	.752 ± .002	.729 ± .002	.744 ± .002	**.749 ± .002**
CSDS2	.99	.993 ± .001	.653 ± .005	**.990 ± .001**	**.990 ± .001**
	.95	.953 ± .002	.653 ± .005	.942 ± .003	**.949 ± .002**
	.90	**.903 ± .003**	.653 ± .005	.881 ± .003	.890 ± .003
	.85	**.854 ± .004**	.653 ± .005	.824 ± .004	.834 ± .004
	.80	**.803 ± .004**	.653 ± .005	.761 ± .004	.781 ± .004
	.75	**.756 ± .004**	.653 ± .005	.703 ± .005	.732 ± .004
CSDS3	.99	**.990 ± .001**	.980 ± .001	.992 ± .001	.989 ± .001
	.95	.948 ± .002	.933 ± .002	.954 ± .002	**.949 ± .002**
	.90	**.899 ± .002**	.887 ± .003	.907 ± .002	.903 ± .002
	.85	**.851 ± .003**	.831 ± .003	.860 ± .003	.854 ± .003
	.80	**.799 ± .003**	.795 ± .003	.789 ± .003	.814 ± .003
	.75	**.744 ± .003**	.738 ± .003	**.744 ± .003**	.760 ± .003
	#	22/42	7/42	3/42	16/42
	V	.015 ± .034	.064 ± .087	.064 ± .108	.019 ± .038

Table B.5: Empirical coverage for PlugInAUC using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Empirical Coverage			
	c	**Logistic**	**RandomForest**	**ResNet**	**XGBoost**
Adult	.99	.988 ± .001	**.989 ± .001**	.988 ± .001	.992 ± .001
	.95	.947 ± .002	.949 ± .002	.946 ± .002	**.951 ± .002**
	.90	.896 ± .003	.894 ± .003	.903 ± .003	**.899 ± .003**
	.85	.848 ± .003	.836 ± .004	.849 ± .003	**.850 ± .003**
	.80	.799 ± .004	.783 ± .004	.801 ± .004	**.800 ± .004**
	.75	.744 ± .004	.743 ± .004	**.749 ± .004**	.748 ± .004
Lending	.99	.996 ± .001	**.992 ± .001**	.995 ± .001	.996 ± .001
	.95	.976 ± .001	**.971 ± .001**	.975 ± .001	.980 ± .001
	.90	.951 ± .001	**.945 ± .001**	.949 ± .001	.958 ± .001
	.85	.926 ± .001	**.917 ± .001**	.922 ± .001	.936 ± .001
	.80	.899 ± .001	**.884 ± .001**	.894 ± .001	.912 ± .001
	.75	.871 ± .001	**.855 ± .001**	.865 ± .001	.886 ± .001
GiveMe	.99	**.991 ± .001**	.944 ± .002	.977 ± .001	**.991 ± .001**
	.95	**.950 ± .002**	.944 ± .002	.930 ± .002	**.948 ± .002**
	.90	**.898 ± .002**	.863 ± .002	.884 ± .002	**.898 ± .002**
	.85	.844 ± .002	.794 ± .003	.841 ± .002	**.849 ± .002**
	.80	.796 ± .003	.780 ± .003	**.797 ± .003**	**.797 ± .003**
	.75	.746 ± .003	.667 ± .003	.748 ± .003	**.750 ± .003**
UCICredit	.99	**.990 ± .002**	.954 ± .003	**.991 ± .002**	**.990 ± .002**
	.95	.962 ± .003	.904 ± .004	.946 ± .003	**.948 ± .003**
	.90	.913 ± .004	.857 ± .005	**.901 ± .004**	.894 ± .004
	.85	.864 ± .004	.838 ± .005	**.849 ± .005**	.848 ± .005
	.80	.816 ± .005	.775 ± .005	**.796 ± .005**	.806 ± .005
	.75	.770 ± .005	.733 ± .006	.744 ± .006	**.752 ± .005**
CSDS1	.99	.991 ± .001	.961 ± .001	**.990 ± .001**	.991 ± .001
	.95	**.950 ± .001**	.914 ± .001	.949 ± .001	**.950 ± .001**
	.90	**.900 ± .002**	.872 ± .002	.899 ± .002	.899 ± .002
	.85	.851 ± .002	.820 ± .002	.851 ± .002	**.850 ± .002**
	.80	**.800 ± .002**	.786 ± .002	.798 ± .002	**.800 ± .002**
	.75	**.749 ± .002**	.698 ± .002	.746 ± .002	.752 ± .002
CSDS2	.99	**.990 ± .001**	.829 ± .004	.989 ± .001	.991 ± .001
	.95	**.950 ± .003**	.829 ± .004	.947 ± .003	.954 ± .002
	.90	.894 ± .003	.829 ± .004	**.902 ± .003**	.907 ± .003
	.85	.841 ± .004	.650 ± .005	.856 ± .004	**.851 ± .004**
	.80	**.798 ± .004**	.650 ± .005	.806 ± .004	.806 ± .004
	.75	**.749 ± .004**	.516 ± .005	.756 ± .004	.754 ± .004
CSDS3	.99	.992 ± .001	.989 ± .001	**.991 ± .001**	**.991 ± .001**
	.95	.955 ± .002	.941 ± .002	**.952 ± .002**	.955 ± .002
	.90	.905 ± .002	.896 ± .003	**.901 ± .003**	.906 ± .002
	.85	.857 ± .003	.838 ± .003	**.850 ± .003**	.860 ± .003
	.80	.799 ± .003	**.803 ± .003**	**.803 ± .003**	.814 ± .003
	.75	.746 ± .003	**.748 ± .003**	.753 ± .003	.762 ± .003
	#	12/42	9/42	13/42	18/42
	V	.017 ± .034	.056 ± .069	.016 ± .032	.017 ± .038

Table B.6: Selective AUC for AUCross using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Selective AUC			
	c	**Logistic**	**RandomForest**	**ResNet**	**XGBoost**
Adult	.99	.903 ± .003	.888 ± .003	.901 ± .003	**.928 ± .003**
	.95	.910 ± .003	.895 ± .003	.907 ± .003	**.934 ± .003**
	.90	.918 ± .003	.903 ± .003	.914 ± .003	**.944 ± .002**
	.85	.926 ± .003	.910 ± .003	.925 ± .003	**.950 ± .002**
	.80	.934 ± .003	.918 ± .003	.933 ± .003	**.959 ± .002**
	.75	.941 ± .003	.925 ± .003	.942 ± .003	**.965 ± .002**
Lending	.99	.956 ± .001	.977 ± .001	.981 ± .001	**.987 ± .001**
	.95	.958 ± .001	.979 ± .001	.983 ± .001	**.988 ± .001**
	.90	.962 ± .001	.982 ± .001	.986 ± .001	**.990 ± .001**
	.85	.965 ± .001	.984 ± .001	.988 ± .001	**.992 ± .001**
	.80	.968 ± .001	.987 ± .001	.991 ± .001	**.993 ± .001**
	.75	.971 ± .001	.989 ± .001	.993 ± .001	**.994 ± .001**
GiveMe	.99	.699 ± .006	.836 ± .005	.670 ± .006	**.865 ± .004**
	.95	.703 ± .006	.836 ± .005	.688 ± .007	**.871 ± .004**
	.90	.708 ± .006	.846 ± .005	.705 ± .008	**.878 ± .004**
	.85	.713 ± .006	.854 ± .005	.720 ± .008	**.885 ± .004**
	.80	.720 ± .006	.858 ± .005	.732 ± .008	**.892 ± .004**
	.75	.726 ± .006	.861 ± .005	.749 ± .009	**.899 ± .004**
UCICredit	.99	.701 ± .009	.760 ± .008	**.768 ± .007**	.767 ± .008
	.95	.704 ± .009	.760 ± .008	**.774 ± .007**	.774 ± .008
	.90	.709 ± .009	.769 ± .008	**.783 ± .007**	.780 ± .008
	.85	.714 ± .009	.776 ± .008	**.792 ± .007**	.787 ± .008
	.80	.719 ± .009	.784 ± .008	**.799 ± .007**	.794 ± .008
	.75	.724 ± .009	.794 ± .008	**.807 ± .007**	.801 ± .008
CSDS1	.99	.676 ± .003	.636 ± .004	.679 ± .003	**.680 ± .003**
	.95	.679 ± .003	.636 ± .004	.682 ± .003	**.683 ± .003**
	.90	.682 ± .003	.639 ± .004	.685 ± .003	**.686 ± .003**
	.85	.685 ± .003	.640 ± .004	**.690 ± .004**	**.690 ± .003**
	.80	.689 ± .004	.643 ± .004	**.694 ± .004**	**.694 ± .004**
	.75	.693 ± .004	.644 ± .004	**.698 ± .004**	.697 ± .004
CSDS2	.99	**.615 ± .019**	.604 ± .023	.615 ± .019	.575 ± .021
	.95	.615 ± .019	.604 ± .023	**.622 ± .020**	.574 ± .021
	.90	.619 ± .020	.604 ± .023	**.623 ± .020**	.574 ± .021
	.85	.622 ± .020	.604 ± .023	**.628 ± .020**	.576 ± .022
	.80	**.626 ± .021**	.604 ± .023	.619 ± .020	.571 ± .022
	.75	**.633 ± .022**	.604 ± .023	.616 ± .020	.568 ± .022
CSDS3	.99	.840 ± .003	.838 ± .003	.840 ± .003	**.845 ± .003**
	.95	.846 ± .003	.845 ± .003	.845 ± .003	**.851 ± .003**
	.90	.852 ± .003	.851 ± .003	.851 ± .003	**.858 ± .003**
	.85	.859 ± .003	.859 ± .003	.858 ± .003	**.865 ± .003**
	.80	.866 ± .003	.865 ± .004	.869 ± .003	**.871 ± .003**
	.75	.873 ± .003	.873 ± .004	.875 ± .003	**.879 ± .003**
	#	3/42	0/42	12/42	29/42

Table B.7: Selective AUC for PlugInAUC using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Selective AUC			
	c	**Logistic**	**RandomForest**	**ResNet**	**XGBoost**
Adult	.99	.904 ± .003	.886 ± .003	.906 ± .003	**.927 ± .003**
	.95	.911 ± .003	.892 ± .003	.912 ± .003	**.934 ± .003**
	.90	.918 ± .003	.900 ± .003	.919 ± .003	**.942 ± .003**
	.85	.925 ± .003	.908 ± .003	.927 ± .003	**.949 ± .002**
	.80	.932 ± .003	.916 ± .003	.934 ± .003	**.956 ± .002**
	.75	.939 ± .003	.922 ± .003	.942 ± .003	**.962 ± .002**
Lending	.99	.956 ± .001	.976 ± .001	.975 ± .001	**.987 ± .001**
	.95	.958 ± .001	.979 ± .001	.978 ± .001	**.989 ± .001**
	.90	.962 ± .001	.982 ± .001	.981 ± .001	**.990 ± .001**
	.85	.965 ± .001	.984 ± .001	.985 ± .001	**.992 ± .001**
	.80	.968 ± .001	.987 ± .001	.988 ± .001	**.993 ± .001**
	.75	.971 ± .001	.989 ± .001	.991 ± .001	**.994 ± .001**
GiveMe	.99	.698 ± .006	.840 ± .005	.834 ± .005	**.862 ± .004**
	.95	.702 ± .006	.840 ± .005	.840 ± .005	**.869 ± .004**
	.90	.708 ± .006	.850 ± .005	.847 ± .005	**.876 ± .004**
	.85	.714 ± .006	.857 ± .005	.853 ± .005	**.883 ± .004**
	.80	.720 ± .006	.859 ± .005	.859 ± .005	**.891 ± .004**
	.75	.727 ± .007	.870 ± .005	.866 ± .005	**.897 ± .004**
UCICredit	.99	.702 ± .009	.757 ± .008	**.768 ± .007**	.760 ± .008
	.95	.704 ± .009	.764 ± .008	**.774 ± .007**	.765 ± .008
	.90	.708 ± .009	.770 ± .008	**.779 ± .007**	.772 ± .008
	.85	.713 ± .009	.772 ± .008	**.785 ± .007**	.778 ± .008
	.80	.718 ± .009	.780 ± .008	**.793 ± .008**	.783 ± .008
	.75	.723 ± .009	.786 ± .008	**.802 ± .008**	.790 ± .008
CSDS1	.99	.676 ± .003	.634 ± .003	.679 ± .003	**.680 ± .003**
	.95	.679 ± .003	.636 ± .004	**.683 ± .003**	**.683 ± .003**
	.90	.682 ± .003	.639 ± .004	.686 ± .003	**.687 ± .003**
	.85	.685 ± .004	.641 ± .004	.690 ± .004	**.691 ± .003**
	.80	.689 ± .004	.644 ± .004	.693 ± .004	**.695 ± .003**
	.75	.693 ± .004	.647 ± .004	.697 ± .004	**.698 ± .004**
CSDS2	.99	.616 ± .019	.587 ± .021	**.624 ± .020**	.577 ± .019
	.95	.621 ± .020	.587 ± .021	**.626 ± .020**	.578 ± .019
	.90	.627 ± .020	.587 ± .021	**.628 ± .020**	.580 ± .019
	.85	.631 ± .021	.580 ± .023	**.633 ± .021**	.583 ± .020
	.80	.636 ± .021	.580 ± .023	**.639 ± .022**	.585 ± .020
	.75	**.644 ± .022**	.591 ± .025	.639 ± .022	.588 ± .021
CSDS3	.99	.840 ± .003	.835 ± .003	**.845 ± .003**	.843 ± .003
	.95	.844 ± .003	.841 ± .003	**.850 ± .003**	.848 ± .003
	.90	.851 ± .003	.847 ± .003	**.857 ± .003**	.856 ± .003
	.85	.856 ± .003	.855 ± .004	**.864 ± .003**	.863 ± .003
	.80	.865 ± .003	.861 ± .004	**.870 ± .003**	.869 ± .003
	.75	.872 ± .003	.868 ± .004	**.878 ± .003**	.877 ± .003
	#	1/42	0/42	18/42	24/42

Table B.8: Selective accuracy for AUCross using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Selective Accuracy			
	c	**Logistic**	**RandomForest**	**ResNet**	**XGBoost**
Adult	.99	.846 ± .003	.838 ± .004	.844 ± .003	**.870 ± .003**
	.95	.851 ± .003	.842 ± .004	.847 ± .004	**.874 ± .003**
	.90	.856 ± .003	.847 ± .004	.853 ± .004	**.883 ± .003**
	.85	.863 ± .004	.853 ± .004	.861 ± .004	**.890 ± .003**
	.80	.870 ± .004	.860 ± .004	.870 ± .004	**.901 ± .003**
	.75	.879 ± .004	.868 ± .004	.883 ± .003	**.911 ± .003**
Lending	.99	.848 ± .001	.878 ± .001	.883 ± .001	**.903 ± .001**
	.95	.857 ± .001	.890 ± .001	.894 ± .001	**.913 ± .001**
	.90	.869 ± .001	.903 ± .001	.909 ± .001	**.927 ± .001**
	.85	.882 ± .001	.918 ± .001	.924 ± .001	**.941 ± .001**
	.80	.896 ± .001	.933 ± .001	.940 ± .001	**.957 ± .001**
	.75	.911 ± .001	.952 ± .001	.957 ± .001	**.975 ± .001**
GiveMe	.99	.934 ± .002	.935 ± .002	.933 ± .002	**.937 ± .002**
	.95	.934 ± .002	.935 ± .002	.929 ± .002	**.937 ± .002**
	.90	.934 ± .002	.934 ± .002	.928 ± .002	**.937 ± .002**
	.85	.933 ± .002	.932 ± .002	.928 ± .002	**.937 ± .002**
	.80	.934 ± .002	.933 ± .002	.926 ± .002	**.938 ± .002**
	.75	.934 ± .002	.933 ± .002	.923 ± .003	**.939 ± .002**
UCICredit	.99	.784 ± .005	.806 ± .005	**.814 ± .005**	.805 ± .005
	.95	.780 ± .005	.806 ± .005	**.812 ± .005**	.805 ± .005
	.90	.777 ± .006	.803 ± .005	**.809 ± .005**	.803 ± .005
	.85	.773 ± .006	.801 ± .006	**.808 ± .006**	.800 ± .006
	.80	.769 ± .006	.800 ± .006	**.806 ± .006**	.796 ± .006
	.75	.765 ± .006	.799 ± .006	**.802 ± .006**	.794 ± .006
CSDS1	.99	**.857 ± .002**	.846 ± .002	**.857 ± .002**	.856 ± .002
	.95	**.856 ± .002**	.846 ± .002	**.856 ± .002**	.855 ± .002
	.90	**.854 ± .002**	.844 ± .002	**.854 ± .002**	.854 ± .002
	.85	**.852 ± .002**	.842 ± .002	**.853 ± .002**	.852 ± .002
	.80	**.851 ± .002**	.840 ± .002	**.851 ± .002**	.850 ± .002
	.75	**.850 ± .002**	.836 ± .002	.849 ± .002	.848 ± .002
CSDS2	.99	**.982 ± .002**	.980 ± .002	**.982 ± .002**	**.982 ± .002**
	.95	**.982 ± .002**	.980 ± .002	**.982 ± .002**	**.982 ± .002**
	.90	**.982 ± .002**	.980 ± .002	**.982 ± .002**	.981 ± .002
	.85	**.982 ± .002**	.980 ± .002	.981 ± .002	.981 ± .002
	.80	**.982 ± .002**	.980 ± .002	.980 ± .002	.980 ± .002
	.75	**.982 ± .002**	.980 ± .002	.980 ± .002	.980 ± .002
CSDS3	.99	.808 ± .003	.808 ± .003	.806 ± .003	**.811 ± .003**
	.95	.810 ± .003	.810 ± .003	.807 ± .003	**.813 ± .003**
	.90	.811 ± .003	.813 ± .003	.808 ± .003	**.815 ± .003**
	.85	.812 ± .003	.817 ± .003	.810 ± .003	**.818 ± .003**
	.80	.814 ± .003	.820 ± .003	.817 ± .003	**.822 ± .003**
	.75	.817 ± .003	.826 ± .003	.820 ± .003	**.828 ± .003**
	#	12/42	0/42	14/42	26/42

Table B.9: Selective accuracy for PlugInAUC using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Selective Accuracy			
	c	**Logistic**	**RandomForest**	**ResNet**	**XGBoost**
Adult	.99	.847 ± .003	.839 ± .004	.844 ± .003	**.869 ± .003**
	.95	.851 ± .003	.842 ± .004	.847 ± .003	**.874 ± .003**
	.90	.855 ± .003	.847 ± .004	.853 ± .003	**.880 ± .003**
	.85	.861 ± .004	.851 ± .004	.860 ± .004	**.888 ± .003**
	.80	.868 ± .004	.858 ± .004	.866 ± .004	**.896 ± .003**
	.75	.875 ± .004	.864 ± .004	.874 ± .004	**.905 ± .003**
Lending	.99	.848 ± .001	.877 ± .001	.874 ± .001	**.904 ± .001**
	.95	.857 ± .001	.888 ± .001	.884 ± .001	**.914 ± .001**
	.90	.869 ± .001	.902 ± .001	.896 ± .001	**.928 ± .001**
	.85	.882 ± .001	.917 ± .001	.910 ± .001	**.942 ± .001**
	.80	.895 ± .001	.936 ± .001	.925 ± .001	**.957 ± .001**
	.75	.911 ± .001	.951 ± .001	.941 ± .001	**.974 ± .001**
GiveMe	.99	.934 ± .002	.935 ± .002	.933 ± .002	**.936 ± .002**
	.95	.934 ± .002	.935 ± .002	.932 ± .002	**.936 ± .002**
	.90	.934 ± .002	.934 ± .002	.931 ± .002	**.936 ± .002**
	.85	.934 ± .002	.932 ± .002	.930 ± .002	**.936 ± .002**
	.80	.935 ± .002	.932 ± .002	.929 ± .002	**.937 ± .002**
	.75	.935 ± .002	.929 ± .002	.929 ± .002	**.937 ± .002**
UCICredit	.99	.785 ± .005	.805 ± .005	**.814 ± .005**	.805 ± .005
	.95	.783 ± .005	.804 ± .005	**.812 ± .005**	.804 ± .005
	.90	.777 ± .006	.801 ± .006	**.810 ± .005**	.803 ± .005
	.85	.774 ± .006	.799 ± .006	**.806 ± .006**	.802 ± .005
	.80	.770 ± .006	.795 ± .006	**.804 ± .006**	.800 ± .006
	.75	.765 ± .006	.793 ± .006	**.806 ± .006**	.798 ± .006
CSDS1	.99	**.857 ± .002**	.847 ± .002	**.857 ± .002**	.856 ± .002
	.95	**.856 ± .002**	.845 ± .002	**.856 ± .002**	.854 ± .002
	.90	**.855 ± .002**	.844 ± .002	**.854 ± .002**	.853 ± .002
	.85	**.852 ± .002**	.841 ± .002	**.853 ± .002**	.851 ± .002
	.80	**.851 ± .002**	.840 ± .002	**.851 ± .002**	.850 ± .002
	.75	**.850 ± .002**	.834 ± .002	.849 ± .002	.848 ± .002
CSDS2	.99	**.982 ± .002**	.981 ± .002	**.982 ± .002**	**.982 ± .002**
	.95	**.982 ± .002**	.981 ± .002	**.982 ± .002**	**.982 ± .002**
	.90	**.982 ± .002**	.981 ± .002	**.982 ± .002**	**.982 ± .002**
	.85	**.982 ± .002**	.980 ± .002	**.982 ± .002**	**.982 ± .002**
	.80	**.982 ± .002**	.980 ± .002	**.982 ± .002**	**.982 ± .002**
	.75	**.982 ± .002**	.979 ± .002	**.982 ± .002**	**.982 ± .002**
CSDS3	.99	**.809 ± .003**	.804 ± .003	.808 ± .003	.808 ± .003
	.95	.809 ± .003	.805 ± .003	.809 ± .003	**.810 ± .003**
	.90	.809 ± .003	.807 ± .003	.811 ± .003	**.814 ± .003**
	.85	.810 ± .003	.811 ± .003	.813 ± .003	**.817 ± .003**
	.80	.812 ± .003	.815 ± .003	.816 ± .003	**.821 ± .003**
	.75	.815 ± .003	.819 ± .003	.821 ± .003	**.825 ± .003**
	#	13/42	0/42	17/42	29/42

Table B.10: Positive rate for AUCross using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Positive Rate			
	c	**Logistic**	**RandomForest**	**ResNet**	**XGBoost**
Adult	.99	**.246 ± .004**	**.246 ± .004**	**.246 ± .004**	**.246 ± .004**
	.95	.245 ± .004	**.246 ± .004**	**.246 ± .004**	**.246 ± .004**
	.90	.245 ± .004	**.246 ± .004**	.245 ± .004	**.246 ± .004**
	.85	.245 ± .004	.245 ± .004	.246 ± .004	**.246 ± .004**
	.80	**.244 ± .004**	**.244 ± .004**	.242 ± .004	**.244 ± .004**
	.75	**.243 ± .004**	**.243 ± .005**	.238 ± .005	**.243 ± .005**
Lending	.99	**.224 ± .001**	.222 ± .001	.223 ± .001	**.224 ± .001**
	.95	**.216 ± .001**	.213 ± .001	.214 ± .001	**.216 ± .001**
	.90	**.206 ± .001**	.202 ± .001	.202 ± .001	.205 ± .001
	.85	**.196 ± .001**	.190 ± .001	.190 ± .001	.193 ± .001
	.80	**.184 ± .001**	.179 ± .001	.177 ± .001	.181 ± .001
	.75	**.171 ± .001**	.165 ± .001	.164 ± .001	.168 ± .001
GiveMe	.99	.067 ± .002	**.068 ± .002**	**.068 ± .002**	.067 ± .002
	.95	.067 ± .002	.068 ± .002	**.072 ± .002**	.068 ± .002
	.90	.068 ± .002	.070 ± .002	**.072 ± .002**	.068 ± .002
	.85	.068 ± .002	.072 ± .002	**.073 ± .002**	.068 ± .002
	.80	.067 ± .002	.071 ± .002	**.075 ± .002**	.068 ± .002
	.75	.067 ± .002	.070 ± .002	**.077 ± .003**	.067 ± .002
UCICredit	.99	.222 ± .005	**.226 ± .006**	.223 ± .005	.222 ± .005
	.95	**.227 ± .006**	.226 ± .006	.226 ± .006	.223 ± .005
	.90	.231 ± .006	**.232 ± .006**	**.232 ± .006**	.227 ± .006
	.85	.235 ± .006	.235 ± .006	**.237 ± .006**	.232 ± .006
	.80	.239 ± .006	.239 ± .006	**.242 ± .006**	.238 ± .006
	.75	.243 ± .006	.243 ± .007	**.249 ± .007**	.242 ± .006
CSDS1	.99	.144 ± .002	**.147 ± .002**	.145 ± .002	.145 ± .002
	.95	.145 ± .002	**.147 ± .002**	.146 ± .002	.146 ± .002
	.90	.147 ± .002	**.148 ± .002**	.147 ± .002	.147 ± .002
	.85	.149 ± .002	**.150 ± .002**	.148 ± .002	.149 ± .002
	.80	.150 ± .002	**.151 ± .002**	.150 ± .002	**.151 ± .002**
	.75	.152 ± .002	**.154 ± .002**	.152 ± .002	.153 ± .002
CSDS2	.99	.019 ± .002	**.021 ± .002**	.019 ± .002	.019 ± .002
	.95	.019 ± .002	**.021 ± .002**	.019 ± .002	.019 ± .002
	.90	.019 ± .002	**.021 ± .002**	.019 ± .002	.019 ± .002
	.85	.019 ± .002	**.021 ± .002**	.020 ± .002	.019 ± .002
	.80	.019 ± .002	**.021 ± .002**	**.021 ± .002**	.020 ± .002
	.75	.019 ± .002	**.021 ± .002**	**.021 ± .002**	**.021 ± .002**
CSDS3	.99	**.254 ± .003**	**.254 ± .003**	**.254 ± .003**	**.254 ± .003**
	.95	**.255 ± .003**	**.255 ± .003**	**.255 ± .003**	.254 ± .003
	.90	**.258 ± .003**	.256 ± .003	.257 ± .003	.255 ± .003
	.85	**.261 ± .003**	.257 ± .004	.259 ± .003	.256 ± .003
	.80	**.263 ± .004**	.257 ± .004	.258 ± .004	.256 ± .004
	.75	**.266 ± .004**	.257 ± .004	.259 ± .004	.255 ± .004
	#	16/42	22/42	16/42	11/42

Table B.11: Positive rate for PlugInAUC using different classifiers (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Positive Rate			
	c	**Logistic**	**RandomForest**	**ResNet**	**XGBoost**
Adult	.99	**.246 ± .004**	**.246 ± .004**	**.246 ± .004**	**.246 ± .004**
	.95	.246 ± .004	**.247 ± .004**	**.247 ± .004**	.246 ± .004
	.90	.247 ± .004	**.248 ± .004**	.246 ± .004	.248 ± .004
	.85	.248 ± .004	**.250 ± .004**	.246 ± .004	.246 ± .004
	.80	.247 ± .004	**.250 ± .004**	.246 ± .004	.247 ± .004
	.75	.248 ± .004	**.250 ± .005**	.245 ± .005	.248 ± .005
Lending	.99	**.224 ± .001**	.222 ± .001	**.224 ± .001**	**.224 ± .001**
	.95	**.217 ± .001**	.213 ± .001	.216 ± .001	.216 ± .001
	.90	**.207 ± .001**	.202 ± .001	**.207 ± .001**	.205 ± .001
	.85	**.196 ± .001**	.190 ± .001	**.196 ± .001**	.195 ± .001
	.80	**.185 ± .001**	.175 ± .001	.184 ± .001	.182 ± .001
	.75	**.172 ± .001**	.163 ± .001	.171 ± .001	.170 ± .001
GiveMe	.99	**.068 ± .002**	**.068 ± .002**	**.068 ± .002**	**.068 ± .002**
	.95	.067 ± .002	.068 ± .002	**.069 ± .002**	.068 ± .002
	.90	.067 ± .002	.069 ± .002	**.070 ± .002**	.068 ± .002
	.85	.067 ± .002	**.071 ± .002**	**.071 ± .002**	.068 ± .002
	.80	.067 ± .002	.071 ± .002	**.072 ± .002**	.068 ± .002
	.75	.066 ± .002	**.075 ± .002**	.073 ± .002	.067 ± .002
UCICredit	.99	.222 ± .005	**.224 ± .005**	.222 ± .005	.222 ± .005
	.95	.225 ± .005	**.227 ± .006**	.226 ± .006	.224 ± .005
	.90	.230 ± .006	**.232 ± .006**	.229 ± .006	.227 ± .006
	.85	.234 ± .006	.234 ± .006	**.236 ± .006**	.230 ± .006
	.80	.238 ± .006	**.241 ± .006**	.240 ± .006	.233 ± .006
	.75	.244 ± .006	**.245 ± .006**	.241 ± .006	.238 ± .006
CSDS1	.99	.144 ± .002	**.146 ± .002**	.144 ± .002	.144 ± .002
	.95	.145 ± .002	**.147 ± .002**	.145 ± .002	.146 ± .002
	.90	.147 ± .002	**.148 ± .002**	.147 ± .002	.147 ± .002
	.85	.149 ± .002	**.150 ± .002**	.148 ± .002	.149 ± .002
	.80	.150 ± .002	**.151 ± .002**	.150 ± .002	.150 ± .002
	.75	.151 ± .002	**.155 ± .002**	.152 ± .002	.153 ± .002
CSDS2	.99	.019 ± .002	**.020 ± .002**	.019 ± .002	.019 ± .002
	.95	.019 ± .002	**.020 ± .002**	.019 ± .002	.019 ± .002
	.90	.019 ± .002	**.020 ± .002**	.019 ± .002	.019 ± .002
	.85	.019 ± .002	**.021 ± .002**	.019 ± .002	.018 ± .002
	.80	.019 ± .002	**.021 ± .002**	.019 ± .002	.019 ± .002
	.75	.019 ± .002	**.022 ± .002**	.019 ± .002	.019 ± .002
CSDS3	.99	**.254 ± .003**	**.254 ± .003**	**.254 ± .003**	**.254 ± .003**
	.95	**.256 ± .003**	.255 ± .003	**.256 ± .003**	.255 ± .003
	.90	**.259 ± .003**	.256 ± .003	.258 ± .003	.254 ± .003
	.85	**.263 ± .003**	.257 ± .004	.260 ± .003	.255 ± .004
	.80	**.265 ± .004**	.256 ± .004	.261 ± .004	.255 ± .004
	.75	**.268 ± .004**	.257 ± .004	.261 ± .004	.256 ± .004
	#	14/42	27/42	13/42	4/42

Table B.12: Empirical coverage for AUCross using different number of folds K over tabular datasets (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Empirical Coverage				
	c	$K = 2$	$K = 3$	$K = 5$	$K = 7$	$K = 10$
Adult	.99	**.990 ± .001**	.991 ± .001	.989 ± .001	.991 ± .001	.991 ± .001
	.95	.952 ± .002	.951 ± .002	**.950 ± .002**	.949 ± .002	.947 ± .002
	.90	.896 ± .003	.897 ± .003	.896 ± .003	.896 ± .003	**.900 ± .003**
	.85	.848 ± .003	**.850 ± .003**	.849 ± .003	.848 ± .003	.849 ± .003
	.80	.796 ± .004	.797 ± .004	.797 ± .004	.799 ± .004	**.800 ± .004**
	.75	**.750 ± .004**	.752 ± .004	**.750 ± .004**	.753 ± .004	.751 ± .004
Lending	.99	**.996 ± .001**	**.996 ± .001**	**.996 ± .001**	**.996 ± .001**	**.996 ± .001**
	.95	**.980 ± .001**	**.980 ± .001**	**.980 ± .001**	**.980 ± .001**	**.980 ± .001**
	.90	.958 ± .001	**.959 ± .001**	**.959 ± .001**	**.959 ± .001**	**.959 ± .001**
	.85	.935 ± .001	**.936 ± .001**	**.936 ± .001**	**.936 ± .001**	**.936 ± .001**
	.80	**.912 ± .001**	**.912 ± .001**	**.912 ± .001**	**.912 ± .001**	**.912 ± .001**
	.75	**.886 ± .001**	**.886 ± .001**	**.886 ± .001**	**.886 ± .001**	**.886 ± .001**
GiveMe	.99	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	.991 ± .001
	.95	.951 ± .002	**.950 ± .002**	**.950 ± .002**	**.950 ± .002**	**.950 ± .002**
	.90	**.900 ± .002**	.899 ± .002	.898 ± .002	.899 ± .002	.898 ± .002
	.85	**.850 ± .002**	.849 ± .002	.848 ± .002	.849 ± .002	.849 ± .002
	.80	.794 ± .003	.795 ± .003	.795 ± .003	**.796 ± .003**	**.796 ± .003**
	.75	.747 ± .003	.749 ± .003	.748 ± .003	.749 ± .003	**.750 ± .003**
UCICredit	.99	.989 ± .002	.989 ± .002	.988 ± .002	.988 ± .002	**.990 ± .002**
	.95	.944 ± .003	**.946 ± .003**	.944 ± .003	.945 ± .003	**.946 ± .003**
	.90	.886 ± .004	.894 ± .004	.896 ± .004	.895 ± .004	**.900 ± .004**
	.85	.830 ± .005	.838 ± .005	.840 ± .005	.840 ± .005	**.843 ± .005**
	.80	.775 ± .005	.789 ± .005	.789 ± .005	.788 ± .005	**.795 ± .005**
	.75	.720 ± .006	.737 ± .006	.737 ± .006	.735 ± .006	**.745 ± .006**
CSDS1	.99	.991 ± .001	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**
	.95	**.950 ± .001**	**.950 ± .001**	**.950 ± .001**	.949 ± .001	**.950 ± .001**
	.90	.898 ± .002	**.899 ± .002**	.898 ± .002	.898 ± .002	.898 ± .002
	.85	.848 ± .002	**.849 ± .002**	**.849 ± .002**	.848 ± .002	.848 ± .002
	.80	.796 ± .002	**.799 ± .002**	**.799 ± .002**	.797 ± .002	.797 ± .002
	.75	.746 ± .002	**.748 ± .002**	**.748 ± .002**	.747 ± .002	.747 ± .002
CSDS2	.99	**.990 ± .001**	.992 ± .001	**.990 ± .001**	**.990 ± .001**	.992 ± .001
	.95	.952 ± .002	**.951 ± .002**	.952 ± .002	.956 ± .002	.956 ± .002
	.90	**.904 ± .003**	**.904 ± .003**	**.904 ± .003**	.909 ± .003	.911 ± .003
	.85	**.854 ± .004**	**.854 ± .004**	.860 ± .004	.861 ± .003	.866 ± .003
	.80	.808 ± .004	**.804 ± .004**	.811 ± .004	.814 ± .004	.819 ± .004
	.75	.760 ± .004	**.754 ± .004**	.760 ± .004	.765 ± .004	.770 ± .004
CSDS3	.99	**.991 ± .001**	.992 ± .001	**.991 ± .001**	**.991 ± .001**	**.991 ± .001**
	.95	.948 ± .002	**.950 ± .002**	.948 ± .002	**.950 ± .002**	.951 ± .002
	.90	.902 ± .002	**.901 ± .002**	**.901 ± .002**	.902 ± .002	**.901 ± .002**
	.85	**.850 ± .003**	.853 ± .003	.849 ± .003	**.850 ± .003**	**.850 ± .003**
	.80	.809 ± .003	.810 ± .003	**.805 ± .003**	**.805 ± .003**	**.805 ± .003**
	.75	.760 ± .003	.759 ± .003	.759 ± .003	**.758 ± .003**	**.758 ± .003**
	#	15/42	23/42	**20/42**	**16/42**	24/42
	V	0.019 ± 0.038	0.017 ± 0.038	0.018 ± 0.038	0.018 ± 0.038	0.018 ± 0.038

Table B.13: Selective AUC for AUCross using different number of folds K over tabular datasets (1,000 bootstrap runs over the test set, results as mean ± stdev).

		Selective AUC				
	c	$K = 2$	$K = 3$	$K = 5$	$K = 7$	$K = 10$
Adult	.99	**.929 ± .003**	.928 ± .003	**.929 ± .003**	.928 ± .003	.928 ± .003
	.95	.935 ± .003	.935 ± .003	.935 ± .003	.935 ± .003	**.936 ± .003**
	.90	**.943 ± .002**	**.943 ± .002**	**.943 ± .002**	**.943 ± .002**	**.943 ± .002**
	.85	**.950 ± .002**	**.950 ± .002**	**.950 ± .002**	**.950 ± .002**	**.950 ± .002**
	.80	**.958 ± .002**	**.958 ± .002**	**.958 ± .002**	**.957 ± .002**	**.957 ± .002**
	.75	.963 ± .002	.963 ± .002	.963 ± .002	**.964 ± .002**	**.964 ± .002**
Lending	.99	.983 ± .001	.983 ± .001	.983 ± .001	.983 ± .001	.983 ± .001
	.95	.985 ± .001	.985 ± .001	.985 ± .001	.985 ± .001	.985 ± .001
	.90	.987 ± .001	.987 ± .001	.987 ± .001	.987 ± .001	.987 ± .001
	.85	.989 ± .001	.989 ± .001	.989 ± .001	.989 ± .001	.989 ± .001
	.80	.990 ± .001	.990 ± .001	.990 ± .001	.990 ± .001	.990 ± .001
	.75	.992 ± .001	.992 ± .001	.992 ± .001	.992 ± .001	.992 ± .001
GiveMe	.99	**.868 ± .004**	**.868 ± .004**	**.868 ± .004**	**.868 ± .004**	.867 ± .004
	.95	**.874 ± .004**	**.874 ± .004**	**.874 ± .004**	**.874 ± .004**	**.874 ± .004**
	.90	.882 ± .004	.882 ± .004	**.883 ± .004**	**.883 ± .004**	**.883 ± .004**
	.85	**.890 ± .004**	**.890 ± .004**	**.890 ± .004**	**.890 ± .004**	**.890 ± .004**
	.80	**.898 ± .004**	.897 ± .004	**.898 ± .004**	**.898 ± .004**	.897 ± .004
	.75	**.904 ± .004**	.903 ± .004	**.904 ± .004**	**.904 ± .004**	.903 ± .004
UCICredit	.99	.772 ± .008	.772 ± .007	.772 ± .008	.772 ± .008	.772 ± .008
	.95	.778 ± .008	.778 ± .008	.778 ± .008	.778 ± .008	.778 ± .008
	.90	**.786 ± .008**	.785 ± .008	.785 ± .008	.784 ± .008	.784 ± .008
	.85	**.794 ± .008**	.793 ± .007	.792 ± .008	.792 ± .008	.792 ± .008
	.80	**.802 ± .008**	.800 ± .008	.800 ± .008	.800 ± .008	.800 ± .008
	.75	**.811 ± .008**	.808 ± .008	.809 ± .008	.809 ± .008	.807 ± .008
CSDS1	.99	.686 ± .003	.686 ± .003	.686 ± .003	.686 ± .003	.686 ± .003
	.95	.689 ± .003	.689 ± .003	.689 ± .003	.689 ± .003	.689 ± .003
	.90	.693 ± .003	.693 ± .003	.693 ± .003	.693 ± .003	.693 ± .003
	.85	.697 ± .003	.697 ± .003	.697 ± .003	.697 ± .003	.697 ± .003
	.80	.702 ± .004	.702 ± .004	.702 ± .004	.702 ± .004	.702 ± .004
	.75	.706 ± .004	.706 ± .004	.706 ± .004	.706 ± .004	.706 ± .004
CSDS2	.99	.615 ± .020	**.618 ± .020**	.616 ± .020	**.618 ± .020**	.617 ± .020
	.95	**.622 ± .020**	.619 ± .020	.619 ± .020	.619 ± .020	.621 ± .020
	.90	.620 ± .020	**.626 ± .020**	.620 ± .020	.625 ± .020	.624 ± .020
	.85	.622 ± .020	.629 ± .021	**.631 ± .021**	.626 ± .021	.626 ± .021
	.80	.628 ± .021	**.636 ± .021**	.631 ± .021	.632 ± .021	.630 ± .021
	.75	.628 ± .021	**.637 ± .021**	.635 ± .021	.634 ± .021	.630 ± .021
CSDS3	.99	.851 ± .003	.851 ± .003	.851 ± .003	.851 ± .003	.851 ± .003
	.95	**.858 ± .003**	.857 ± .003	**.858 ± .003**	.857 ± .003	.857 ± .003
	.90	.865 ± .003	.865 ± .003	.865 ± .003	.865 ± .003	.865 ± .003
	.85	.873 ± .003	.873 ± .003	.873 ± .003	.873 ± .003	.873 ± .003
	.80	.879 ± .003	.879 ± .003	**.880 ± .003**	**.880 ± .003**	**.880 ± .003**
	.75	.886 ± .003	**.887 ± .003**	**.887 ± .003**	**.887 ± .003**	**.887 ± .003**
	#	29/42	28/42	31/42	30/42	27/42

Appendix C

Appendix to Deep Neural Network Benchmarks for Selective Classification

C.1 Experimental Details

We provide all the additional information on datasets, settings, and code for replicating the experiments of the Chapter 7.

C.1.1 Datasets

Table C.1 reports the datasets used in our benchmarking and a link to retrieve the original data. We also include whether the dataset was considered in a previous SC evaluation.

Table C.2 reports some experimental details, including training size; batch size used for training; feature space in terms of features for tabular data and image size for image data; target space number of classes m; the percentage of the minority class in each dataset. We also report the Deep Neural Network (DNN) architectures we employed for each dataset. Such a choice was made according to the following criteria:

(i) a former paper in the literature used this dataset and employed a specific architecture;

(ii) if point *(i)* does not apply, we did the following:

- for image data, we employed a ResNet34 architecture;
- for tabular data, if a dataset was tested in [104], we applied the best-performing architecture on that specific dataset. Otherwise, we employed the FTTransformer architecture following the suggestion by [103].

All the data were re-shuffled, normalized and split into training, test, calibration and validation sets, according to a 60%, 20%, 10%, and 10% proportion

Table C.1: Dataset sources.

Dataset	Data Type	Link	Previous SC paper
adult	Tabular	uci/adult	Pugnana and Ruggieri [102, 129]
aloi	Tabular	openml/id=1592	–
bank	Tabular	uci/bank+marketing	Franc et al. [24]
bloodmnist	Image	zenodo/6496656/files/bloodmnist	–
breastmnist	Image	zenodo/6496656/files/breastmnist	–
catsdogs	Image	kaggle/dogs-vs-cats	Geifman and El-Yaniv [15], Liu et al. [16], Huang et al. [18], Pugnana and Ruggieri [102, 129]
chestmnist	Image	zenodo/6496656/files/chestmnist	–
cifar10	Image	pytorch/vision/CIFAR10	Geifman and El-Yaniv [15], Corbière et al. [17], Huang et al. [18], Feng et al. [20], Pugnana and Ruggieri [129]
compass	Tabular	openml/id=44162	–
covtype	Tabular	openml/id=1596	Franc et al. [24]
dermamnist	Image	zenodo/6496656/files/dermamnist	–
electricity	Tabular	openml/id=44120	–
eye	Tabular	openml/id=44157	–
food101	Image	pytorch/vision/Food101	Feng et al. [20]
giveme	Tabular	kaggle/GiveMeSomeCredit	Pugnana and Ruggieri [102, 129]
helena	Tabular	openml/id=41169	–
heloc	Tabular	openml/id=45023	–
higgs	Tabular	openml/id=23512	–
house	Tabular	openml/id=43957	–
indian	Tabular	openml/id=41972	–
jannis	Tabular	openml/id=44079	–
kddipums97	Tabular	openml/id=44124	–
letter	Tabular	openml/id=6	Franc et al. [24]
magic	Tabular	openml/id=44125	–
miniboone	Tabular	openml/id=44119	–
MNIST	Image	pytorch/vision/MNIST	Lakshminarayanan et al. [75], Liu et al. [16], Corbière et al. [17]
octmnist	Image	zenodo/6496656/files/octmnist	–
online	Tabular	openml/id=45060	–
organamnist	Image	zenodo/6496656/files/organamnist	–
organcmnist	Image	zenodo/6496656/files/organcmnist	–
organsmnist	Image	zenodo/6496656/files/organsmnist	–
oxfordpets	Image	pytorch/vision/oxfordpets	–
pathmnist	Image	zenodo/6496656/files/pathmnist	–
phoneme	Tabular	openml/id=44127	–
pneumoniamnist	Image	zenodo/6496656/files/pneumoniamnist	–
pol	Tabular	openml/id=43991	–
retinamnist	Image	zenodo/6496656/files/retinamnist	–
rl	Tabular	openml/id=44160	–
stanfordcars	Image	pytorch/vision/StanfordCars	Feng et al. [20]
SVHN	Image	pytorch/vision/SVHN	Geifman and El-Yaniv [14, 15], Liu et al. [16], Corbière et al. [17]
tissuemnist	Image	zenodo/6496656/files/tissuemnist	–
ucicredit	Tabular	openml/id=42477	Pugnana and Ruggieri [102, 129]
upselling	Tabular	openml/id=44158	–
waterbirds	Image	stanford.edu/dro/waterbird	Jones et al. [51]

(respectively). In the code repository, we provide the `Python` scripts to recreate the data employed in this analysis.

Table C.2: Dataset details.

Dataset	Training Size	Batch Size	# Features	# Classes	Minority Ratio	DNN Architecture
adult	$29,303$	256	13	2	23.9%	FTTransformer
aloi	$64,800$	512	128	1000	0.1%	TabResnet
bank	$27,126$	128	16	2	11.7%	TabResnet
bloodmnist	$10,253$	128	28×28	8	7.1%	Resnet18
breastmnist	468	64	28×28	2	26.9%	Resnet18
catsdogs	$15,000$	128	64×64	2	50.0%	VGG
chestmnist	$67,272$	512	28×28	2	10.3%	Resnet18
cifar10	$36,000$	128	32×32	1	100.0%	VGG
compass	$9,985$	128	17	2	50.0%	FTTransformer
covtype	$348,605$	1024	54	7	0.5%	FTTransformer
dermamnist	$6,008$	128	28×28	7	1.1%	Resnet18
electricity	$23,083$	128	7	2	50.0%	FTTransformer
eye	$4,564$	128	23	2	50.0%	FTTransformer
food101	$60,600$	256	224×224	101	1.0%	Resnet34
giveme	$90,000$	512	8	2	6.7%	TabResnet
helena	$39,116$	512	27	100	0.2%	TabResnet
heloc	$6,000$	128	22	2	50.0%	FTTransformer
higgs	$58,829$	512	28	2	47.1%	FTTransformer
house	$8,092$	128	16	2	50.0%	FTTransformer
indian	$5,485$	128	220	8	0.2%	TabResnet
jannis	$34,548$	512	54	2	50.0%	FTTransformer
kddipums97	$3,112$	128	20	2	50.0%	FTTransformer
letter	$12,000$	128	16	26	3.7%	FTTransformer
magic	$8,024$	128	10	2	50.0%	FTTransformer
miniboone	$43,798$	256	50	2	50.0%	TabResnet
MNIST	$42,000$	128	28×28	10	9.0%	Resnet34
octmnist	$65,585$	512	28×28	4	8.1%	Resnet18
online	$7,398$	128	17	2	15.5%	FTTransformer
organamnist	$35,310$	256	28×28	11	4.0%	Resnet18
organcmnist	$14,196$	128	28×28	11	4.8%	Resnet18
organsmnist	$15,132$	128	28×28	11	4.6%	Resnet18
oxfordpets	$4,409$	128	224×224	2	32.3%	Resnet34
pathmnist	$64,308$	512	28×28	9	8.9%	Resnet18
phoneme	$1,901$	128	5	2	50.0%	FTTransformer
pneumoniamnist	$3,512$	128	28×28	2	27.1%	Resnet18
pol	$9,000$	128	26	11	1.7%	FTTransformer
retinamnist	960	128	28×28	5	5.8%	Resnet18
rl	$2,982$	128	12	2	50.0%	FTTransformer
stanfordcars	$9,710$	128	224×224	196	0.3%	Resnet34
SVHN	$59,573$	128	32×32	10	6.3%	VGG
tissuemnist	$141,830$	1024	28×28	8	3.5%	Resnet18
ucicredit	$18,000$	128	23	2	22.1%	TabResnet
upselling	$3,017$	128	45	2	50.0%	FTTransformer
waterbirds	$7,072$	128	224×224	2	22.6%	Resnet50

C.1.2 Hyperparameter Settings

We optimize the hyperparameters using `Optuna` [190], a framework for multi-objective Bayesian optimization, with the following inputs: coverage violation and cross-entropy loss as target metrics, `BoTorch` as sampler [196], 10 initial independent trials out of 20 total trials. We report in Table C.3 the parameter space we used during the tuning procedure. Some hyperparameters are loss-specific, as they refer to a specific baseline loss, while others are network-specific, as they refer to a specific deep neural network architecture. We report the search space in the last column, where the notation $[a; b]_X$ stands for all values within $[a; b]$ that are linearly spaced with a gap equal to X. For example, $[0, 60]_{15}$ indicates the set of values $\{0, 15, 30, 45, 60\}$. For small sets of values, we directly indicate all the possibilities using the notation $\{a_1, a_2, \dots\}$. We discretize the search space of most hyperparameters because Bayesian Optimization suffers from high-dimensional spaces. We specify that if `time decay`

Table C.3: Hyperparameter spaces.

Parameter	Loss-Specific	Network-Specific	Search Space
o	DeepGamblers	No	$[1; m]_{.02}$
γ	SAT, SAT+EM	No	$[0.9; 0.99]_{.01}$
E_s	SAT, SAT+EM	No	$[0; 60]_{15}$
β	SAT+EM, SelNet+EM	No	$\{1e{-}4, 1e{-}3, , 1e{-}2, 1e{-}1\}$
α	SelNet, SelNet+EM	No	$\{.25; .75\}_{.05}$
λ	SelNet, SelNet+EM	No	$\{8, 16, 32, 64\}$
`optimizer`	No	No	$\{$`SGD`, `Adam`, `AdamW`$\}$
`learning rate`	No	No	$\{1e-5, 1e-4, 1e-3, 1e-2\}$
`optimizer unc.`	ConfidNet, Sele, Reg	Yes	$\{$`SGD`, `Adam`, `AdamW`$\}$
`learning rate unc.`	ConfidNet, Sele, Reg	Yes	$\{1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2\}$
`time decay`	No	No	$\{$`True`, `False`$\}$
`nesterov`	No	No	$\{$`True`, `False`$\}$
`nesterov unc.`	ConfidNet, Sele, Reg	Yes	$\{$`True`, `False`$\}$
`weight decay`	No	No	$\{1e{-}6, 1e{-}5, 1e{-}4, 1e{-}3\}$
`d_token`	No	FTTransformer, TabResNet	$[64, 512]_{64}$
`n_blocks`	No	FTTransformer, TabResNet	$\{1, 2, 3, 4\}$
`d_hidden_factor`	No	FTTransformer, TabResNet	$[2/3; 8/3]_{1/3}$
`attention_dropout`	No	FTTransformer	$\{0; .5\}_{.05}$
`residual_dropout`	No	FTTransformer	$\{0; .2\}_{.05}$
`ffn_dropout`	No	FTTransformer	$\{0; .5\}_{.05}$
`d_main`	No	TabResNet	$[64, 512]_{64}$
`d_dropout_first`	No	TabResNet	$\{0; .5\}_{.05}$
`d_dropout_second`	No	TabResNet	$\{0; .5\}_{.05}$
`batch_norm`	No	VGG	$\{$`True`, `False`$\}$
`zero_init_residual`	No	ResNet34, ResNet50	$\{$`True`, `False`$\}$

is set to `True`, we halve the learning rate every 25 epochs as done in [15]. Moreover, for image data, which we found to be more unstable during the training, we start the `Optuna` optimization procedure using default values if these were suggested in some SC paper. Due to the huge computational cost of the tuning procedure, SelNet+SmR, SelNet+EM+SmR, SAT+SmR and SAT+EM+SmR use the same optimal hyperparameters as, respectively, SelNet, SelNet+EM, SAT and SAT+EM. Similarly, SCross, Ensemble, EnsembleSR, AUCross, and PluginAUC employ the best configuration found for SmR as they share the same training loss, i.e. cross-entropy. For both SCross and AUCross we set $K = 5$, following the suggestions in Pugnana and Ruggieri [102, 129]. For both Ensemble and EnsembleSR we used the default value of $K = 10$, following the suggestions in Lakshminarayanan et al. [75]. For the uncertainty network of ConfidNet, we employed the same choice architecture detailed in the original paper [17], i.e. the same main body as the network classifier followed by 4 dense layers in a single node with sigmoid activation. We used such a structure also for building Sele and Reg uncertainty networks. Following the empirical evaluation in [24], we split the training data in half to train Sele and Reg networks: on the one half we train the classifier, on the other half, the uncertainty network. We provide the best configurations we employed in the final analysis in Tables C.4-C.12.

Table C.4: Best configurations for DEEPGAMBLERS, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	o	n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	SGD	$1e-03$	$1e-04$	False	.92	False	1.0	1	320	.30		2.67		FTTransformer
compass	Adam	$1e-04$	$1e-04$	False			2.0	1	64	.20		1.33	.45	
covtype	Adam	$1e-04$	$1e-04$	False			4.2	2	512	.15	.20	1.67	.10	
electricity	Adam	$1e-05$	$1e-03$	False			2.0	1	192	.05		1.00	.50	
eye	Adam	$1e-05$	$1e-04$	False			2.0	1	64	.15		1.33	.45	
heloc	AdamW	$1e-05$	$1e-05$	True			1.8	3	384	.25	.10	1.33	.45	
higgs	Adam	$1e-05$	$1e-04$	False			2.0	1	64	.15		1.33	.45	
house	Adam	$1e-05$	$1e-04$	False			2.0	1	64	.15		1.33	.45	
jannis	Adam	$1e-05$	$1e-04$	False			2.0	1	64	.15		1.33	.45	
kddipums97	Adam	$1e-03$	$1e-04$	False			2.0	1	64	.15		1.33	.45	
letter	Adam	$1e-05$	$1e-03$	False			18.4	4	64	.05		1.00	.40	
magic	Adam	$1e-05$	$1e-04$	False			2.0	1	64	.15		1.33	.45	
online	Adam	$1e-03$	$1e-04$	False			2.0	1	64	.20		1.67	.50	
phoneme	Adam	$1e-05$	$1e-04$	False			2.0	1	64	.15		1.33	.45	
pol	AdamW	$1e-03$	$1e-05$	True			8.4	4	256	.05	.10	1.67	.40	
rl	Adam	$1e-04$	$1e-03$	True			1.4	2	320	.30		1.67	.15	
upselling	Adam	$1e-05$	$1e-04$	False			2.0	1	64	.15		1.33	.45	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	o	n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	SGD	$1e-02$	$1e-04$	False	.96	True	748.4	4	128	192	2.00	.20	.40	TabResnet
bank	SGD	$1e-01$	$1e-03$	True	.9	False	1.2	4	64	192	1.00	.45	.30	
giveme	AdamW	$1e-05$	$1e-04$	True			2.0	1	384	512	4.00	.40	.30	
helena	AdamW	$1e-04$	$1e-04$	True			89.0	1	384	448	3.00	.40	.25	
indian	SGD	$1e-01$	$1e-03$	True	.9	False	3.2	4	64	192	1.00	.40	.30	
miniboone	AdamW	$1e-04$	$1e-04$	True			2.0	1	384	448	3.00	.40	.25	
ucicredit	AdamW	$1e-05$	$1e-04$	True			2.0	1	448	448	3.00	.30	.25	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	o	zero_init_resid	Arch.
MNIST	Adam	$1e-04$	$1e-04$	True			2.8	True	Resnet18-50
bloodmnist	Adam	$1e-04$	$1e-04$	True			4.0	False	
breastmnist	SGD	$1e-01$	$1e-04$	True	.9	False	5.0	False	
chestmnist	SGD	$1e-01$	$1e-04$	True	.9	False	5.0	False	
dermamnist	AdamW	$1e-04$	$1e-03$	True			3.4	True	
food101	Adam	$1e-03$	$1e-06$	False			31.4	True	
octmnist	SGD	$1e-01$	$1e-04$	True	.9	False	5.0	False	
organamnist	SGD	$1e-02$	$1e-03$	False	.93	True	7.2	True	
organcmnist	SGD	$1e-02$	$1e-03$	False	.94	True	6.8	True	
organsmnist	AdamW	$1e-04$	$1e-03$	True			5.8	True	
oxfordpets	SGD	$1e-01$	$1e-03$	True	.9	False	1.2	True	
pathmnist	Adam	$1e-03$	$1e-05$	False			3.4	True	
pneumoniamnist	SGD	$1e-01$	$1e-04$	True	.9	False	5.0	False	
retinamnist	AdamW	$1e-04$	$1e-03$	True			2.8	True	
stanfordcars	Adam	$1e-04$	$1e-06$	False			161.0	False	
tissuemnist	SGD	$1e-03$	$1e-03$	False	.96	True	8.0	True	
waterbirds	AdamW	$1e-04$	$1e-06$	False			2.0	False	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	o	b_norm	Arch.
SVHN	SGD	$1e-04$	$1e-05$	False	.99	True	7.8	True	VGG
catsdogs	SGD	$1e-02$	$1e-03$	False	.91	True	1.4	True	
cifar10	SGD	$1e-02$	$1e-03$	False	.91	True	5.2	True	

Table C.5: Best configurations for SAT, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	E_s	γ	n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	AdamW	$1e-05$	$1e-05$	True			45	.97	3	128	.40		1.67		FTTransformer
compass	AdamW	$1e-05$	$1e-05$	False			45	.92	2	128	.50		2.33	.45	
covtype	Adam	$1e-04$	$1e-06$	True			15	.95	4	320	.15		.67	.05	
electricity	Adam	$1e-04$	$1e-06$	False			15	.98	3	128	.45		1.00	.10	
eye	AdamW	$1e-02$	$1e-06$	True			45	.97	3	256	.20		2.33		
heloc	AdamW	$1e-05$	$1e-05$	False			45	.92	2	192	.45		2.00	.45	
higgs	Adam	$1e-04$	$1e-05$	True			60	.99	2	192	.15	.10	1.00		
house	AdamW	$1e-03$	$1e-06$	False			15	.97	1	384	.25	.20	1.00	.35	
jannis	Adam	$1e-04$	$1e-06$	True			15	.94	3	128	.25		1.00	.30	
kddipums97	Adam	$1e-04$	$1e-06$	True			15	.93	4	128	.25		1.00	.30	
letter	Adam	$1e-04$	$1e-06$	True			15	.92	4	256	.25		1.00	.30	
magic	Adam	$1e-03$	$1e-06$	True			15	.94	4	128	.25		1.00	.30	
online	Adam	$1e-04$	$1e-05$	True			60	.99	2	192	.15	.10	1.00		
phoneme	Adam	$1e-03$	$1e-06$	True			15	.93	4	64	.30		1.00	.30	
pol	Adam	$1e-03$	$1e-06$	True			15	.93	4	192	.15		1.00	.30	
rl	Adam	$1e-04$	$1e-06$	False			0	.98	3	64	.45		1.00	.10	
upselling	Adam	$1e-04$	$1e-06$	True			15	.93	4	128	.25		1.00	.30	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	E_s	γ	n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	Adam	$1e-04$	$1e-05$	False			0	.93	2	320	384	3.00	.50	.50	TabResnet
bank	Adam	$1e-02$	$1e-04$	True			15	.93	2	320	320	2.00	.45	.15	
giveme	Adam	$1e-04$	$1e-04$	False			30	.91	1	512	192	3.00	.15	.25	
helena	Adam	$1e-04$	$1e-04$	False			30	.9	4	192	128	2.00	.45	.15	
indian	Adam	$1e-03$	$1e-03$	True			0	.98	4	512	192	1.00	.10	.20	
miniboone	Adam	$1e-03$	$1e-03$	True			0	.98	4	512	192	1.00	.10	.20	
ucicredit	Adam	$1e-03$	$1e-03$	True			0	.98	4	512	192	1.00	.10	.20	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	E_s	γ	zero_init_resid	Arch.
MNIST	SGD	$1e-02$	$1e-05$	False	.92	False	0	.99	True	Resnet18-50
bloodmnist	Adam	$1e-02$	$1e-06$	True			60	.9	True	
breastmnist	AdamW	$1e-05$	$1e-03$	False			0	.91	False	
chestmnist	Adam	$1e-02$	$1e-06$	True			60	.9	True	
dermamnist	AdamW	$1e-04$	$1e-05$	True			30	.94	True	
food101	Adam	$1e-02$	$1e-06$	True			45	.9	False	
octmnist	AdamW	$1e-03$	$1e-05$	True			60	.92	True	
organamnist	Adam	$1e-03$	$1e-06$	False			0	.98	True	
organcmnist	Adam	$1e-03$	$1e-06$	True			0	.91	True	
organsmnist	Adam	$1e-03$	$1e-06$	True			45	.93	True	
oxfordpets	AdamW	$1e-03$	$1e-06$	False			45	.91	False	
pathmnist	Adam	$1e-02$	$1e-06$	True			15	.91	True	
pneumoniamnist	SGD	$1e-01$	$1e-04$	True	.9	False	60	.9	False	
retinamnist	AdamW	$1e-04$	$1e-06$	True			0	.95	True	
stanfordcars	Adam	$1e-03$	$1e-05$	False			15	.93	True	
tissuemnist	Adam	$1e-02$	$1e-06$	True			15	.91	True	
waterbirds	SGD	$1e-02$	$1e-03$	False	.9	True	30	.92	False	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	E_s	γ	b_norm	Arch.
SVHN	Adam	$1e-02$	$1e-06$	True			15	.9	True	VGG
catsdogs	Adam	$1e-02$	$1e-06$	True			45	.9	True	
cifar10	Adam	$1e-02$	$1e-06$	True			45	.93	True	

Table C.6: Best configurations for SAT+EM, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	E_s	γ	β	n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	SGD	$1e-04$	$1e-06$	False	.95	True	45	.9	.01	3	256	.50		2.67	.50	FTTransformer
compass	Adam	$1e-03$	$1e-06$	False			60	.96	.0001	1	512	.45		1.33	.10	
covtype	AdamW	$1e-05$	$1e-05$	False			30	.98	.001	3	320			2.33	.10	
electricity	AdamW	$1e-05$	$1e-05$	False			30	.96	.001	3	320	.05		2.33	.10	
eye	AdamW	$1e-05$	$1e-04$	True			30	.91	.01	2	448	.30		1.33	.15	
heloc	AdamW	$1e-03$	$1e-05$	True			30	.92	.0001	2	384	.10	.15	2.33	.50	
higgs	AdamW	$1e-03$	$1e-05$	True			30	.98	.001	2	448	.10		1.67	.30	
house	Adam	$1e-03$	$1e-06$	False			60	.96	.0001	1	512	.45		1.33	.10	
jannis	AdamW	$1e-03$	$1e-06$	True			30	.97	.0001	3	448	.10		2.00	.25	
kddipums97	Adam	$1e-03$	$1e-06$	False			60	.95	.0001	1	512	.45		1.33	.10	
letter	AdamW	$1e-05$	$1e-05$	False			30	.96	.001	3	384			2.67		
magic	Adam	$1e-03$	$1e-06$	False			60	.94	.0001	1	512	.45		1.33	.10	
online	Adam	$1e-04$	$1e-06$	False			60	.96	.0001	1	512	.45		1.33	.10	
phoneme	AdamW	$1e-04$	$1e-06$	True			15	.99	.01	3	320	.20		2.67	.40	
pol	AdamW	$1e-04$	$1e-06$	True			30	.98	.0001	3	448	.15		2.00	.30	
rl	Adam	$1e-04$	$1e-06$	False			60	.97	.0001	1	512	.50		1.33	.15	
upselling	AdamW	$1e-05$	$1e-06$	True			30	.99	.0001	3	448	.10		2.00	.30	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	E_s	γ	β	n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	SGD	$1e-01$	$1e-04$	False	.93	False	15	.91	.001	3	192	512	4.00	.10	.25	TabResnet
bank	Adam	$1e-02$	$1e-06$	True			15	.94	.001	1	384	256	1.00	.30	.05	
giveme	Adam	$1e-04$	$1e-06$	True			0	.93	.0001	2	384	256	2.00	.25	.05	
helena	Adam	$1e-04$	$1e-06$	True			15	.95	.0001	2	384	256	2.00	.25	.05	
indian	Adam	$1e-03$	$1e-06$	True			0	.94	.001	2	384	256	1.00	.30	.05	
miniboone	Adam	$1e-02$	$1e-06$	True			15	.94	.001	1	384	256	1.00	.30	.05	
ucicredit	AdamW	$1e-05$	$1e-06$	False			60	.92	.001	2	512	128	3.00	.40	.10	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	E_s	γ	β	zero_init_resid	Arch.
MNIST	SGD	$1e-01$	$1e-05$	False	.96	True	60	.98	.0001	True	Resnet18-50
bloodmnist	Adam	$1e-03$	$1e-06$	False			60	.96	.0001	False	
breastmnist	SGD	$1e-02$	$1e-04$	True	.9	False	45	.9	.01	False	
chestmnist	Adam	$1e-03$	$1e-06$	True			0	.93	.0001	False	
dermamnist	SGD	$1e-04$	$1e-05$	False	.99	True	60	.98	.0001	True	
food101	Adam	$1e-02$	$1e-06$	True			0	.9	.01	True	
octmnist	Adam	$1e-03$	$1e-06$	True			0	.93	.0001	False	
organamnist	Adam	$1e-04$	$1e-06$	False			0	.96	.0001	False	
organcmnist	Adam	$1e-03$	$1e-06$	False			60	.96	.0001	False	
organsmnist	SGD	$1e-02$	$1e-05$	False	.95	True	60	.98	.0001	True	
oxfordpets	SGD	$1e-02$	$1e-04$	True	.93	True	0	.98	.0001	False	
pathmnist	Adam	$1e-03$	$1e-06$	False			0	.95	.0001	False	
pneumoniamnist	Adam	$1e-05$	$1e-06$	False			0	.94	.0001	False	
retinamnist	Adam	$1e-04$	$1e-06$	True			0	.93	.0001	False	
stanfordcars	Adam	$1e-03$	$1e-04$	False			15	.98	.001	False	
tissuemnist	Adam	$1e-02$	$1e-06$	True			0	.93	.0001	False	
waterbirds	Adam	$1e-03$	$1e-06$	False			0	.96	.0001	False	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	E_s	γ	β	b_norm	Arch.
SVHN	Adam	$1e-04$	$1e-06$	True			0	.93	.0001	False	VGG
catsdogs	SGD	$1e-01$	$1e-05$	False	.96	True	60	.98	.0001	True	
cifar10	SGD	$1e-01$	$1e-04$	False	.93	True	60	.98	.0001	True	

Table C.7: Best configurations for SelNet, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	λ	α	n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	AdamW	$1e-05$	$1e-06$	True			32	.65	3	192	.40		2.00		FTTransformer
compass	Adam	$1e-04$	$1e-05$	True			64	.75	2	192	.15	.10	1.00		
covtype	Adam	$1e-04$	$1e-06$	True			16	.25	3	384	.10		.67		
electricity	Adam	$1e-04$	$1e-04$	True			64	.75	2	192	.05	.05	1.00		
eye	AdamW	$1e-05$	$1e-06$	True			32	.70	3	320	.35		2.67	.05	
heloc	Adam	$1e-05$	$1e-06$	False			8	.70	2	64	.50		1.33	.10	
higgs	AdamW	$1e-04$	$1e-06$	True			64	.65	3	192	.25		2.67		
house	Adam	$1e-04$	$1e-05$	True			64	.75	2	192	.15	.10	1.00		
jannis	AdamW	$1e-04$	$1e-05$	False			64	.35	2	256	.50		2.33	.50	
kddipums97	Adam	$1e-04$	$1e-06$	False			8	.70	3	64	.50		1.33	.10	
letter	Adam	$1e-04$	$1e-06$	True			16	.40	4	320	.10		.67	.20	
magic	Adam	$1e-04$	$1e-05$	True			32	.55	3	256	.15		.67	.10	
online	Adam	$1e-03$	$1e-05$	True			64	.70	2	192	.15	.20	1.00		
phoneme	Adam	$1e-03$	$1e-06$	True			8	.35	3	320	.10		1.33	.30	
pol	Adam	$1e-04$	$1e-05$	True			64	.75	2	192	.15	.10	1.00		
rl	Adam	$1e-03$	$1e-06$	False			8	.70	3	64	.50		1.00	.10	
upselling	Adam	$1e-04$	$1e-06$	True			16	.45	4	128	.25		1.00	.30	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	λ	α	n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	Adam	$1e-04$	$1e-05$	False			32	.25	4	128	128	2.00	.50	.05	TabResnet
bank	Adam	$1e-03$	$1e-03$	True			8	.70	4	512	192	1.00	.10	.20	
giveme	Adam	$1e-03$	$1e-03$	True			8	.70	4	512	192	1.00	.10	.20	
helena	Adam	$1e-04$	$1e-04$	False			32	.25	4	128	128	2.00	.50	.10	
indian	AdamW	$1e-02$	$1e-04$	True			32	.65	2	256	320	2.00	.40	.15	
miniboone	Adam	$1e-03$	$1e-03$	True			8	.70	4	512	192	1.00	.10	.20	
ucicredit	Adam	$1e-04$	$1e-04$	False			32	.25	4	192	64	2.00	.45	.10	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	λ	α	zero_init_resid	Arch.
MNIST	SGD	$1e-01$	$1e-04$	True	.94	True	32	.55	False	Resnet18-50
bloodmnist	SGD	$1e-01$	$1e-04$	True	.99	True	32	.45	False	
breastmnist	AdamW	$1e-03$	$1e-05$	True			32	.45	True	
chestmnist	Adam	$1e-02$	$1e-06$	True			64	.25	True	
dermamnist	AdamW	$1e-05$	$1e-03$	False			16	.30	False	
food101	SGD	$1e-01$	$1e-04$	True	.9	False	32	.50	False	
octmnist	Adam	$1e-02$	$1e-06$	True			16	.30	True	
organamnist	SGD	$1e-01$	$1e-04$	True	.94	True	32	.50	False	
organcmnist	Adam	$1e-03$	$1e-06$	True			32	.30	True	
organsmnist	Adam	$1e-03$	$1e-06$	True			16	.25	True	
oxfordpets	AdamW	$1e-03$	$1e-06$	False			16	.25	False	
pathmnist	SGD	$1e-01$	$1e-04$	True	.93	True	64	.60	False	
pneumoniamnist	Adam	$1e-02$	$1e-06$	True			16	.30	True	
retinamnist	Adam	$1e-02$	$1e-06$	True			16	.30	True	
stanfordcars	Adam	$1e-03$	$1e-06$	True			64	.25	True	
tissuemnist	Adam	$1e-02$	$1e-06$	True			16	.30	True	
waterbirds	Adam	$1e-03$	$1e-06$	False			8	.45	True	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	λ	α	b_norm	Arch.
SVHN	Adam	$1e-02$	$1e-06$	True			64	.25	True	VGG
catsdogs	Adam	$1e-02$	$1e-06$	True			32	.25	True	
cifar10	Adam	$1e-03$	$1e-06$	False			8	.70	True	

Table C.8: Best configurations for SelNet+EM, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	λ	α	β	n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	AdamW	$1e-02$	$1e-05$	True			16	.25	.0001	2	384	.15		2.33	.50	FTTransformer
compass	AdamW	$1e-03$	$1e-06$	True			16	.75	.0001	3	448	.15		2.00	.30	
covtype	AdamW	$1e-05$	$1e-05$	False			16	.60	.001	3	320	.05		2.33	.10	
electricity	AdamW	$1e-05$	$1e-06$	False			16	.45	.0001	4	256	.10		1.00	.35	
eye	AdamW	$1e-04$	$1e-04$	True			64	.25	.01	1	448	.25		1.67	.20	
heloc	AdamW	$1e-03$	$1e-04$	True			64	.25	.01	1	448	.35		1.33	.15	
higgs	AdamW	$1e-03$	$1e-04$	True			64	.35	.01	1	384	.20		1.33	.10	
house	AdamW	$1e-05$	$1e-05$	True			8	.50	.0001	3	384	.20		1.33		
jannis	Adam	$1e-04$	$1e-06$	False			64	.60	.0001	1	512	.50		1.67	.15	
kddipums97	AdamW	$1e-03$	$1e-06$	True			16	.70	.0001	3	448	.10		2.00	.30	
letter	AdamW	$1e-04$	$1e-06$	True			16	.70	.0001	3	448	.10		2.00	.35	
magic	AdamW	$1e-03$	$1e-06$	True			16	.75	.0001	3	448	.15		2.00	.25	
online	AdamW	$1e-03$	$1e-06$	True			16	.75	.0001	3	448	.15		2.00	.30	
phoneme	AdamW	$1e-05$	$1e-06$	False			16	.50	.0001	4	192	.15		1.00	.35	
pol	AdamW	$1e-05$	$1e-05$	False			16	.65	.001	3	320			2.67		
rl	Adam	$1e-04$	$1e-06$	False			64	.55	.0001	1	512	.45		1.33	.10	
upselling	AdamW	$1e-05$	$1e-06$	True			16	.70	.0001	3	448	.10		2.00	.30	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	λ	α	β	n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	SGD	$1e-01$	$1e-05$	False	.99	False	64	.65	.0001	4	320	384	3.00	.15	.05	TabResnet
bank	Adam	$1e-03$	$1e-06$	True			16	.55	.0001	1	320	320	1.00	.30	.05	
giveme	Adam	$1e-03$	$1e-06$	True			16	.45	.0001	1	384	256	1.00	.25	.05	
helena	Adam	$1e-03$	$1e-06$	True			8	.50	.0001	2	448	192	1.00	.25	.05	
indian	AdamW	$1e-05$	$1e-06$	False			64	.35	.001	2	512	128	3.00	.40	.05	
miniboone	Adam	$1e-04$	$1e-06$	True			8	.45	.0001	2	384	256	2.00	.25	.05	
ucicredit	Adam	$1e-03$	$1e-06$	True			16	.45	.001	1	384	256	1.00	.30	.05	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	λ	α	β	zero_init_resid	Arch.
MNIST	SGD	$1e-03$	$1e-05$	False	.93	False	64	.35	.0001	False	Resnet18-50
bloodmnist	Adam	$1e-03$	$1e-06$	False			64	.55	.0001	False	
breastmnist	Adam	$1e-04$	$1e-06$	True			8	.40	.0001	False	
chestmnist	SGD	$1e-02$	$1e-05$	False	.93	True	64	.30	.0001	False	
dermamnist	Adam	$1e-05$	$1e-06$	False			16	.60	.0001	False	
food101	Adam	$1e-04$	$1e-06$	True			8	.40	.0001	False	
octmnist	Adam	$1e-04$	$1e-06$	False			16	.60	.0001	False	
organamnist	Adam	$1e-03$	$1e-06$	True			8	.55	.0001	False	
organcmnist	Adam	$1e-03$	$1e-06$	False			64	.55	.0001	False	
organsmnist	Adam	$1e-04$	$1e-06$	False			64	.60	.0001	False	
oxfordpets	SGD	$1e-02$	$1e-04$	True	.95	False	32	.50	.0001	False	
pathmnist	Adam	$1e-03$	$1e-06$	False			64	.55	.0001	False	
pneumoniamnist	Adam	$1e-03$	$1e-06$	False			64	.55	.0001	False	
retinamnist	Adam	$1e-02$	$1e-06$	False			8	.60	.0001	False	
stanfordcars	Adam	$1e-04$	$1e-06$	True			8	.40	.0001	False	
tissuemnist	Adam	$1e-04$	$1e-06$	True			8	.40	.0001	False	
waterbirds	Adam	$1e-04$	$1e-06$	False			16	.60	.0001	False	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	λ	α	β	b_norm	Arch.
SVHN	Adam	$1e-04$	$1e-06$	True			8	.40	.0001	False	VGG
catsdogs	Adam	$1e-03$	$1e-06$	False			16	.65	.0001	False	
cifar10	SGD	$1e-02$	$1e-03$	True	.99	True	32	.50	.001	True	

Table C.9: Best configurations for ConfidNet, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc. n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	Adam	$1e-05$	$1e-05$	True			AdamW	$1e-06$			4	64	.50		2.33		FTTransformer
compass	Adam	$1e-03$	$1e-05$	True			AdamW	$1e-06$			2	64	.40		2.00	.15	
covtype	Adam	$1e-04$	$1e-05$	False			Adam	$1e-05$			2	448	.10	.15	2.00	.35	
electricity	Adam	$1e-05$	$1e-06$	False			Adam	$1e-05$			4	448	.10	.05	1.67	.25	
eye	SGD	$1e-04$	$1e-03$	False	.96	True	AdamW	$1e-06$			3	128	.35		.67	.15	
heloc	AdamW	$1e-05$	$1e-04$	False			SGD	$1e-06$	.92	True	2	128	.50		2.33	.45	
higgs	AdamW	$1e-04$	$1e-03$	False			Adam	$1e-06$			1	512	.40		2.67	.30	
house	AdamW	$1e-03$	$1e-03$	False			Adam	$1e-06$			1	512	.40		2.67	.30	
jannis	AdamW	$1e-03$	$1e-03$	False			Adam	$1e-06$			1	512	.40		2.67	.30	
kddipums97	AdamW	$1e-04$	$1e-03$	False			Adam	$1e-06$			1	512	.40		2.67	.25	
letter	SGD	$1e-03$	$1e-03$	False	.97	True	AdamW	$1e-05$			3	128	.35		.67	.15	
magic	AdamW	$1e-03$	$1e-03$	False			Adam	$1e-06$			1	512	.40		2.67	.30	
online	AdamW	$1e-03$	$1e-03$	False			Adam	$1e-06$			1	512	.40		2.67	.30	
phoneme	AdamW	$1e-03$	$1e-03$	False			Adam	$1e-06$			1	512	.40		2.67	.30	
pol	Adam	$1e-03$	$1e-05$	False			Adam	$1e-05$			4	384	.05	.20	2.00	.25	
rl	AdamW	$1e-04$	$1e-03$	False			Adam	$1e-06$			1	448	.40		2.67	.25	
upselling	SGD	$1e-04$	$1e-03$	False	.96	True	AdamW	$1e-06$			3	128	.35		.67	.15	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc. n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	Adam	$1e-04$	$1e-05$	False			AdamW	$1e-05$			4	512	192	2.00	.15	.15	TabResnet
bank	Adam	$1e-03$	$1e-03$	True			Adam	$1e-04$			3	64	128	2.00	.40	.45	
giveme	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-04$			4	448	448	1.00	.30	.40	
helena	Adam	$1e-04$	$1e-06$	True			AdamW	$1e-07$			3	128	512	3.00	.25	.40	
indian	AdamW	$1e-04$	$1e-04$	True			Adam	$1e-05$			1	128	320	1.00	.35	.25	
miniboone	Adam	$1e-02$	$1e-03$	True			Adam	$1e-04$			2	64	128	2.00	.40	.45	
ucicredit	Adam	$1e-02$	$1e-03$	True			Adam	$1e-04$			2	64	128	2.00	.40	.45	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc. zero_init_resid	Arch.
MNIST	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-07$			True	Resnet18-50
bloodmnist	SGD	$1e-01$	$1e-04$	True	.9	False	Adam	$1e-06$			False	
breastmnist	Adam	$1e-05$	$1e-04$	False			AdamW	$1e-07$			True	
chestmnist	SGD	$1e-01$	$1e-03$	True	.98	True	Adam	$1e-06$			False	
dermamnist	AdamW	$1e-05$	$1e-05$	False			AdamW	$1e-06$			True	
food101	SGD	$1e-03$	$1e-03$	True	.99	True	Adam	$1e-06$			False	
octmnist	SGD	$1e-02$	$1e-04$	True	.95	True	Adam	$1e-05$			False	
organamnist	Adam	$1e-03$	$1e-04$	False			SGD	$1e-04$	.99	True	False	
organcmnist	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-07$			True	
organsmnist	SGD	$1e-02$	$1e-03$	True	.96	True	Adam	$1e-06$			False	
oxfordpets	Adam	$1e-03$	$1e-04$	False			SGD	$1e-05$	.97	False	False	
pathmnist	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-06$			True	
pneumoniamnist	Adam	$1e-04$	$1e-03$	False			SGD	$1e-06$	.95	True	False	
retinamnist	AdamW	$1e-05$	$1e-04$	False			Adam	$1e-07$			False	
stanfordcars	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-07$			True	
tissuemnist	Adam	$1e-03$	$1e-03$	True			Adam	$1e-08$			False	
waterbirds	Adam	$1e-04$	$1e-04$	False			AdamW	$1e-08$			True	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc. b_norm	Arch.
SVHN	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-07$			True	VGG
catsdogs	AdamW	$1e-03$	$1e-05$	False			Adam	$1e-05$			False	
cifar10	Adam	$1e-03$	$1e-04$	False			AdamW	$1e-07$			True	

Table C.10: Best configurations for REG, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc. n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	AdamW	$1e-05$	$1e-04$	False			SGD	$1e-05$	.92	True	2	128	.50		2.33	.45	FTTransformer
compass	SGD	$1e-01$	$1e-06$	True	.9	False	AdamW	$1e-07$			4	256	.10		1.67	.35	
covtype	Adam	$1e-03$	$1e-05$	True			AdamW	$1e-06$			2	320	.10	.05	.67	.15	
electricity	SGD	$1e-03$	$1e-04$	True	.99	True	AdamW	$1e-04$			2	128	.35		1.33		
eye	SGD	$1e-04$	$1e-03$	False	.96	True	AdamW	$1e-05$			3	128	.35		.67	.15	
heloc	Adam	$1e-03$	$1e-05$	True			SGD	$1e-04$	.9	False	1	64	.15		2.00	.25	
higgs	AdamW	$1e-04$	$1e-04$	False			SGD	$1e-04$	.9	True	2	64	.50		2.33	.50	
house	SGD	$1e-03$	$1e-04$	False	.95	True	AdamW	$1e-06$			3	256	.35	.05	.67	.15	
jannis	Adam	$1e-04$	$1e-05$	True			AdamW	$1e-04$			4	64	.25		2.67	.05	
kddipums97	AdamW	$1e-03$	$1e-03$	False			Adam	$1e-07$			1	512	.40		2.67	.30	
letter	SGD	$1e-02$	$1e-04$	False	.9	False	AdamW	$1e-06$			3	320	.20	.05	1.67	.25	
magic	Adam	$1e-04$	$1e-05$	True			AdamW	$1e-04$			3	128	.05		1.33	.25	
online	Adam	$1e-04$	$1e-06$	False			Adam	$1e-04$			1	384	.30		2.67	.20	
phoneme	AdamW	$1e-03$	$1e-03$	False			Adam	$1e-05$			1	512	.40		2.67	.30	
pol	Adam	$1e-04$	$1e-03$	True			AdamW	$1e-06$			4	320	.25		2.00	.05	
rl	AdamW	$1e-03$	$1e-03$	False			AdamW	$1e-06$			2	128	.10		.67	.30	
upselling	SGD	$1e-04$	$1e-03$	False	.96	True	AdamW	$1e-05$			3	128	.35		.67	.15	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc. n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	Adam	$1e-03$	$1e-06$	True			AdamW	$1e-06$			3	128	512	4.00	.50	.45	TabResnet
bank	SGD	$1e-01$	$1e-06$	True	.99	False	AdamW	$1e-07$			4	256	64	3.00	.45	.30	
giveme	Adam	$1e-04$	$1e-05$	False			AdamW	$1e-04$			4	512	192	2.00	.15	.15	
helena	Adam	$1e-04$	$1e-03$	True			SGD	$1e-06$	.99	False	3	192	320	3.00	.25	.05	
indian	AdamW	$1e-04$	$1e-04$	True			Adam	$1e-03$			3	256	512	1.00	.10	.05	
miniboone	Adam	$1e-04$	$1e-03$	True			SGD	$1e-05$	.97	False	3	256	384	3.00	.25		
ucicredit	AdamW	$1e-05$	$1e-04$	True			Adam	$1e-07$			3	192	512	1.00	.15		

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc. zero_init_resid	Arch.
MNIST	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-07$			True	Resnet18-50
bloodmnist	Adam	$1e-02$	$1e-06$	False			SGD	$1e-06$	.9	False	True	
breastmnist	Adam	$1e-04$	$1e-04$	True			AdamW	$1e-03$			True	
chestmnist	AdamW	$1e-02$	$1e-04$	True			Adam	$1e-07$			False	
dermamnist	SGD	$1e-01$	$1e-04$	True	.99	False	Adam	$1e-04$			True	
food101	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-06$			True	
octmnist	SGD	$1e-02$	$1e-05$	True	.99	True	Adam	$1e-06$			True	
organamnist	Adam	$1e-03$	$1e-06$	False			SGD	$1e-05$	.98	True	True	
organcmnist	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-07$			True	
organsmnist	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-07$			True	
oxfordpets	Adam	$1e-04$	$1e-06$	True			AdamW	$1e-04$			True	
pathmnist	SGD	$1e-01$	$1e-05$	True	.97	True	Adam	$1e-07$			True	
pneumoniamnist	Adam	$1e-03$	$1e-06$	False			SGD	$1e-03$	.9	False	True	
retinamnist	SGD	$1e-01$	$1e-04$	True	.99	False	Adam	$1e-06$			True	
stanfordcars	Adam	$1e-03$	$1e-04$	False			AdamW	$1e-03$			False	
tissuemnist	SGD	$1e-01$	$1e-04$	True	.97	False	Adam	$1e-06$			True	
waterbirds	Adam	$1e-03$	$1e-06$	False			SGD	$1e-03$	.97	True	True	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc. b_norm	Arch.
SVHN	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-06$			True	VGG
catsdogs	Adam	$1e-02$	$1e-05$	False			AdamW	$1e-04$			True	
cifar10	AdamW	$1e-03$	$1e-05$	False			SGD	$1e-03$	.99	False	True	

Table C.11: Best configurations for SELE, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc.	n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	SGD	$1e-03$	$1e-06$	True	.93	True	AdamW	$1e-07$				4	256	.05	.10	1.67	.40	FTTransformer
compass	SGD	$1e-04$	$1e-03$	False	.96	True	AdamW	$1e-05$				3	128	.35		.67	.15	
covtype	SGD	$1e-01$	$1e-06$	True	.9	False	AdamW	$1e-07$				4	256	.05		1.33	.35	
electricity	Adam	$1e-05$	$1e-06$	False			Adam	$1e-06$				4	448	.10	.15	1.67	.25	
eye	SGD	$1e-04$	$1e-03$	False	.96	True	AdamW	$1e-05$				3	128	.35		.67	.15	
heloc	Adam	$1e-03$	$1e-04$	True			SGD	$1e-04$	.94	True		1	128	.50	.05	1.33	.15	
higgs	AdamW	$1e-04$	$1e-05$	True			AdamW	$1e-04$				3	192	.15	.10	2.00	.40	
house	SGD	$1e-04$	$1e-03$	False	.96	True	AdamW	$1e-05$				3	128	.35		.67	.15	
jannis	SGD	$1e-03$	$1e-03$	False	.94	True	AdamW	$1e-07$				3	128	.35		.67	.20	
kddipums97	SGD	$1e-04$	$1e-03$	False	.96	True	AdamW	$1e-05$				3	128	.35		.67	.15	
letter	SGD	$1e-02$	$1e-05$	False	.95	True	SGD	$1e-04$	.99	True		2	320	.25	.05	2.33	.40	
magic	Adam	$1e-03$	$1e-05$	True			AdamW	$1e-04$				3	128	.15	.20	.67	.10	
online	AdamW	$1e-03$	$1e-04$	False			Adam	$1e-04$				1	512	.50		2.67	.25	
phoneme	Adam	$1e-05$	$1e-06$	False			Adam	$1e-04$				4	448	.10	.05	1.67	.25	
pol	Adam	$1e-03$	$1e-05$	True			SGD	$1e-04$	.93	True		2	384	.05	.05	1.67	.30	
rl	Adam	$1e-05$	$1e-06$	False			Adam	$1e-04$				4	448	.10	.05	1.67	.25	
upselling	Adam	$1e-04$	$1e-05$	True			AdamW	$1e-04$				3	128	.35		2.67	.05	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc.	n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	AdamW	$1e-04$	$1e-05$	False			SGD	$1e-03$	.92	True		1	64	512	3.00	.40	.15	TabResnet
bank	Adam	$1e-02$	$1e-03$	True			Adam	$1e-03$				2	64	128	2.00	.40	.45	
giveme	AdamW	$1e-04$	$1e-04$	True			Adam	$1e-03$				3	256	512	1.00	.20	.05	
helena	Adam	$1e-04$	$1e-06$	True			AdamW	$1e-05$				3	128	512	4.00	.50	.25	
indian	Adam	$1e-02$	$1e-03$	True			Adam	$1e-03$				2	64	128	2.00	.40	.45	
miniboone	AdamW	$1e-04$	$1e-04$	True			Adam	$1e-06$				3	320	512	1.00	.20	.15	
ucicredit	Adam	$1e-02$	$1e-03$	True			Adam	$1e-03$				2	64	128	2.00	.40	.45	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc.	zero_init_resid	Arch.
MNIST	SGD	$1e-01$	$1e-03$	True	.95	False	Adam	$1e-06$				True	Resnet18-50
bloodmnist	Adam	$1e-02$	$1e-06$	False			SGD	$1e-04$	.94	True		True	
breastmnist	Adam	$1e-04$	$1e-06$	True			AdamW	$1e-04$				True	
chestmnist	SGD	$1e-01$	$1e-05$	True	.95	False	Adam	$1e-03$				True	
dermamnist	SGD	$1e-01$	$1e-04$	True	.93	False	Adam	$1e-04$				True	
food101	Adam	$1e-03$	$1e-04$	False			AdamW	$1e-04$				False	
octmnist	SGD	$1e-02$	$1e-04$	True	.96	True	Adam	$1e-04$				True	
organamnist	SGD	$1e-01$	$1e-04$	True	.94	True	Adam	$1e-06$				True	
organcmnist	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-07$				True	
organsmnist	Adam	$1e-04$	$1e-03$	False			AdamW	$1e-06$				True	
oxfordpets	Adam	$1e-04$	$1e-06$	True			AdamW	$1e-07$				True	
pathmnist	SGD	$1e-01$	$1e-04$	True	.93	True	Adam	$1e-06$				True	
pneumoniamnist	AdamW	$1e-03$	$1e-04$	True			Adam	$1e-06$				False	
retinamnist	SGD	$1e-01$	$1e-04$	True	.99	False	Adam	$1e-06$				True	
stanfordcars	Adam	$1e-03$	$1e-04$	False			AdamW	$1e-03$				False	
tissuemnist	SGD	$1e-02$	$1e-04$	True	.94	True	Adam	$1e-03$				True	
waterbirds	Adam	$1e-04$	$1e-06$	False			SGD	$1e-04$	.93	True		True	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	optim.	unc. l. rate	unc. mom.	SGD unc. nest.	unc.	b_norm	Arch.
SVHN	AdamW	$1e-04$	$1e-04$	False			Adam	$1e-06$				False	VGG
catsdogs	AdamW	$1e-03$	$1e-04$	False			SGD	$1e-03$	.95	True		True	
cifar10	AdamW	$1e-03$	$1e-05$	False			SGD	$1e-03$	.99	True		True	

Table C.12: Best configurations for SmR, divided by architectures.

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	n_blocks	d_token	att._drop.	res._drop.	d_ffn_factor	ffn_drop.	Arch.
adult	SGD	$1e-03$	$1e-04$	False	.91	False	4	128	.25		1.00	.30	FTTransformer
compass	Adam	$1e-03$	$1e-06$	False			1	320	.10	.10	2.00	.25	
covtype	Adam	$1e-04$	$1e-05$	False			3	384	.45		2.33	.35	
electricity	Adam	$1e-04$	$1e-05$	False			3	384	.45		2.33	.35	
eye	Adam	$1e-05$	$1e-06$	False			1	384	.15	.05	2.33	.20	
heloc	SGD	$1e-04$	$1e-04$	False	.93	True	4	128	.30		1.00	.30	
higgs	Adam	$1e-05$	$1e-06$	False			2	384	.20		2.00	.15	
house	Adam	$1e-05$	$1e-06$	False			2	320	.20	.10	2.00	.15	
jannis	Adam	$1e-05$	$1e-06$	False			2	384	.25	.15	2.00	.20	
kddipums97	Adam	$1e-05$	$1e-06$	False			4	320	.35	.20	1.00	.10	
letter	Adam	$1e-04$	$1e-05$	False			3	384	.45		2.33	.35	
magic	Adam	$1e-04$	$1e-06$	False			1	384	.10		2.33	.30	
online	Adam	$1e-04$	$1e-05$	False			1	384	.45		2.33	.35	
phoneme	Adam	$1e-02$	$1e-04$	False			1	128		.05	1.33	.45	
pol	Adam	$1e-03$	$1e-04$	False			1	192	.05	.10	1.67	.40	
rl	Adam	$1e-04$	$1e-03$	False			3	512	.50		1.67	.30	
upselling	SGD	$1e-02$	$1e-04$	False	.99	True	4	64	.25		1.00	.20	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	n_blocks	d_token	d_main	d_hidden_f	d_drop_first	d_drop_sec.	Arch.
aloi	AdamW	$1e-05$	$1e-06$	False			4	192	320	4.00	.10	.20	TabResnet
bank	AdamW	$1e-03$	$1e-04$	True			4	256	64	3.00	.40	.30	
giveme	AdamW	$1e-04$	$1e-05$	True			3	64	256	2.00	.30	.40	
helena	AdamW	$1e-05$	$1e-06$	False			3	192	320	4.00	.10	.20	
indian	AdamW	$1e-03$	$1e-04$	True			4	256	64	3.00	.40	.30	
miniboone	AdamW	$1e-03$	$1e-04$	True			4	256	64	3.00	.40	.30	
ucicredit	AdamW	$1e-04$	$1e-04$	True			3	192	128	2.00	.25	.35	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	zero_init_resid	Arch.
MNIST	Adam	$1e-04$	$1e-04$	False			False	Resnet18-50
bloodmnist	AdamW	$1e-03$	$1e-04$	True			True	
breastmnist	AdamW	$1e-03$	$1e-04$	True			True	
chestmnist	AdamW	$1e-03$	$1e-03$	True			True	
dermamnist	AdamW	$1e-04$	$1e-04$	True			True	
food101	Adam	$1e-03$	$1e-04$	True			False	
octmnist	AdamW	$1e-03$	$1e-04$	True			True	
organamnist	SGD	$1e-01$	$1e-04$	True	.93	True	False	
organcmnist	AdamW	$1e-03$	$1e-03$	True			True	
organsmnist	Adam	$1e-04$	$1e-04$	False			False	
oxfordpets	Adam	$1e-02$	$1e-05$	True			True	
pathmnist	SGD	$1e-02$	$1e-03$	True	.97	True	False	
pneumoniamnist	Adam	$1e-04$	$1e-06$	False			True	
retinamnist	Adam	$1e-05$	$1e-06$	False			True	
stanfordcars	Adam	$1e-03$	$1e-04$	False			False	
tissuemnist	AdamW	$1e-03$	$1e-04$	True			True	
waterbirds	Adam	$1e-04$	$1e-04$	False			False	

Dataset	optim.	l. rate	w. decay	t. decay	mom.	SGD nest.	b_norm	Arch.
SVHN	SGD	$1e-03$	$1e-06$	False	.91	True	False	VGG
catsdogs	Adam	$1e-03$	$1e-04$	False			False	
cifar10	Adam	$1e-03$	$1e-04$	False			True	

C.2 Additional Experimental Results

C.2.1 Q1: Results by Dataset Type

Figure C.1 plots the best two and the worst two baselines mean *RelErr* by data type.

For binary tabular datasets (Figure C.1a), SAT+EM+SmR and SmR are the best two performing methods. The former's relative error rate ranges from $\approx .508$ at $c = .99$ to $\approx .405$ at $c = .70$, while the latter achieves $\approx .511$ at $c = .99$ and $\approx .393$ at $c = .70$. The worst two methods are DeepGamblers, with *RelErr* of $\approx .632$ at $c = .99$ and $\approx .559$ at .70, and Reg with *RelErr* of $\approx .547$ at .99 and $\approx .527$ at .70.

For multiclass tabular datasets (Figure C.1c), the best two methods are EnsembleSR and SAT+SmR, with a mean relative error rate of $\approx .164$ and $\approx .158$ at $c = .99$ respectively, up to $\approx .094$ and $\approx .096$ at $c = .70$. The worst methods are Reg, which reaches a mean relative error rate of $\approx .211$ at $c = .99$ and of $\approx .203$ at $c = .70$, and SelNet+EM+SmR, with a relative error rate ranging from $\approx .195$ at $c = .99$ to $\approx .218$ at $c = .70$.

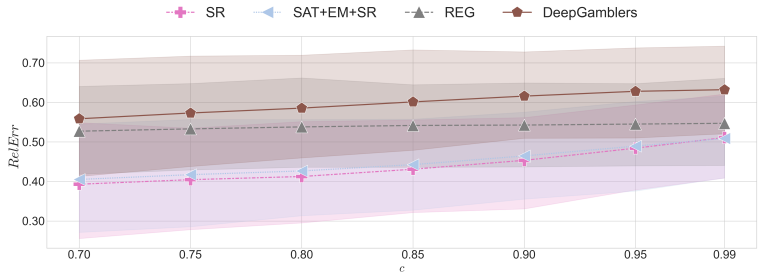
For image datasets, methods based on ensembles, i.e. Ensemble and EnsembleSR, achieve the lowest relative error rate. For binary image datasets (Figure C.1b), EnsembleSR reaches $\approx .378$ at $c = .99$ and $\approx .228$ at $c = .70$, while Ensemble ranges from $\approx .386$ at $c = .99$ to $\approx .234$ at $c = .70$. In this setting, the worst baselines are Sele and DeepGamblers, with a mean relative error rate of $\approx .529$ and $\approx .565$ at $c = .99$ respectively, up to $\approx .564$ and $\approx .582$ at $c = .70$ respectively.

For multiclass image datasets (Figure C.1d), EnsembleSR passes from a mean relative error rate of $\approx .189$ at $c = .99$ to $\approx .126$ at $c = .70$, while Ensemble achieves $\approx .191$ at $c = .99$ up to $\approx .151$ at $c = .70$. The worst methods here are Reg and Sele. The former's relative error rate ranges from $\approx .276$ at $c = .99$ to $\approx .279$ at $c = .70$, while the latter achieves $\approx .272$ at $c = .99$ and $\approx .238$ at $c = .70$.

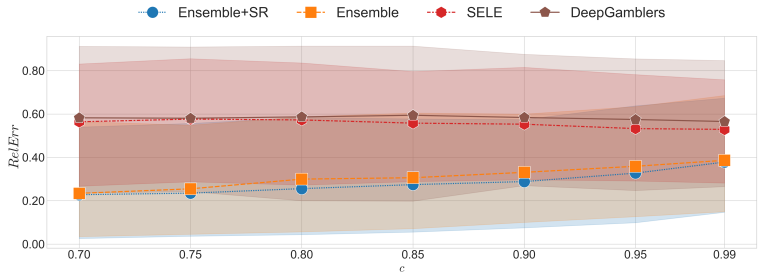
Then, we perform the Nemenyi post hoc test to check for statistically significant differences. Figures C.2 and C.3 provide CD plots when considering tabular and image data respectively at $c = .99$, $c = .90$, $c = .80$ and $c = .70$. As for aggregated results, all the best performing methods are not distinguishable in a statistically significant sense.

C.2.2 Q5: Additional Experiments

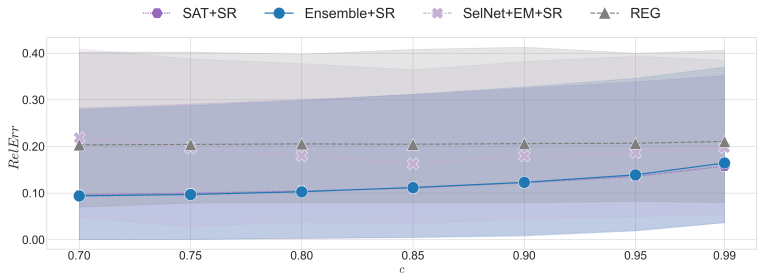
We provide additional results w.r.t. distribution shifts. We perform the same experiment as for Q5, but now considering datasets in the `OpenOOD` benchmark [213], which is specific for out-of-distribution detection, rather than randomly generated pictures. For `cifar10` we use as test set a random sample from `cifar100`, for `MNIST` a random sample from `FashionMNIST` and for `SVHN` a random sample from `cifar10`. Figure C.4 reports the results and the overall mean over the 3 datasets for the 16 baselines considered.



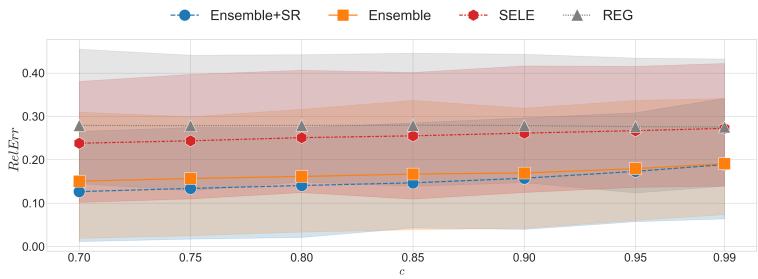
(a) Binary Tabular Data.



(b) Binary Image Data.

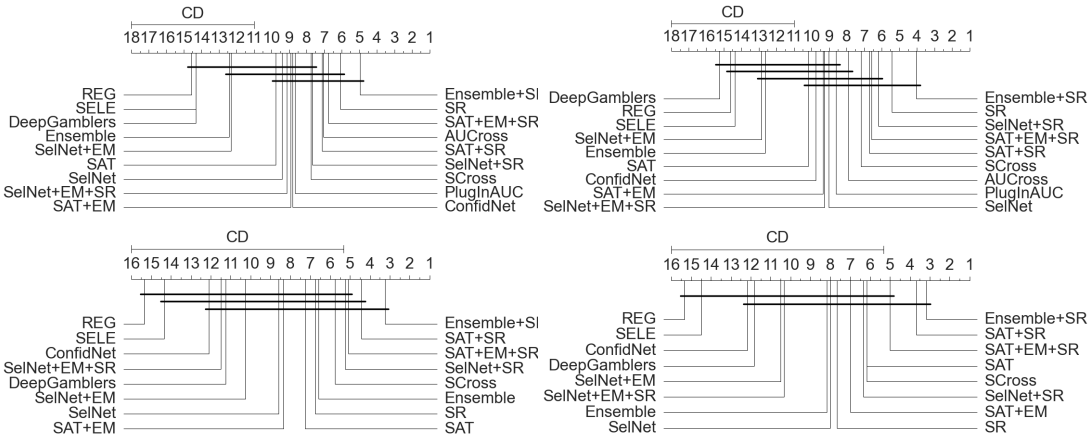


(c) Multiclass Tabular Data.

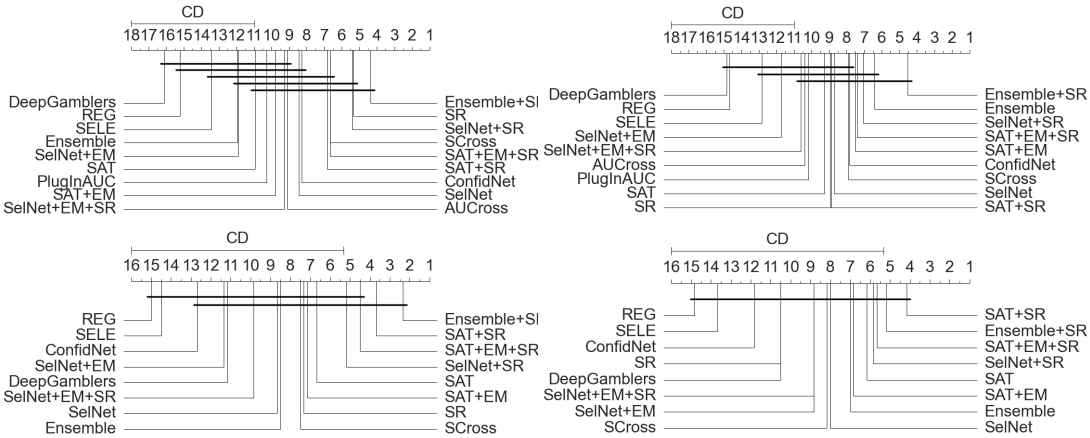


(d) Multiclass Image Data.

Figure C.1: Q1: *RelErr* as a function of target coverage c for the two best and worst approaches on (a) binary tabular data, (b) binary image data, (c) multiclass tabular data and (d) multiclass image data. For readability, only the two best and two worst approaches are shown in each subplot.

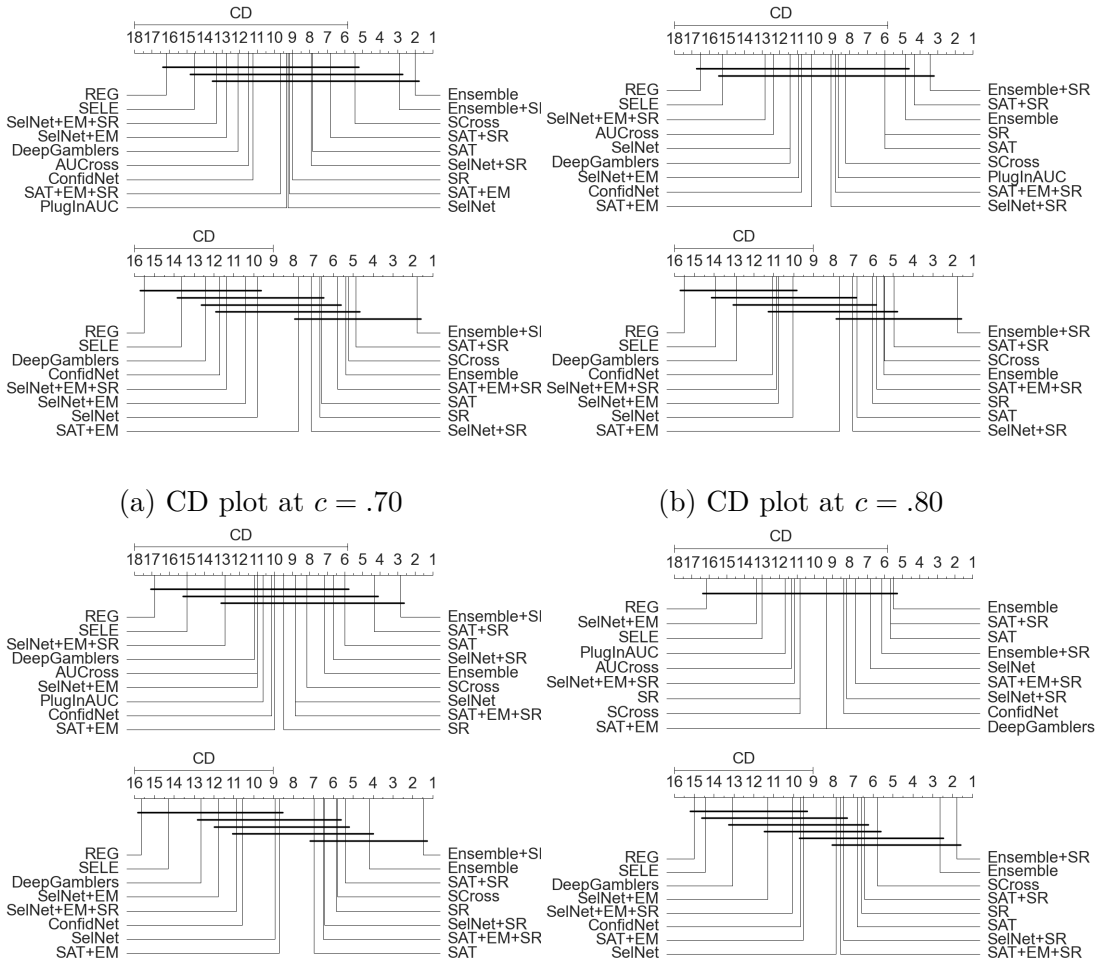


(a) CD plot at $c = .70$ (b) CD plot at $c = .80$



(c) CD plot at $c = .90$ (d) CD plot at $c = .99$

Figure C.2: Q1: CD plots of relative error rate *RelErr* for different target coverages on tabular datasets. Top plots for binary datasets. Bottom plots for multiclass datasets.



(a) CD plot at $c = .70$

(b) CD plot at $c = .80$

(c) CD plot at $c = .90$

(d) CD plot at $c = .99$

Figure C.3: Q1: CD plots of relative error rate *RelErr* for different target coverages on image datasets. Top plots for binary datasets. Bottom plots for multiclass datasets.

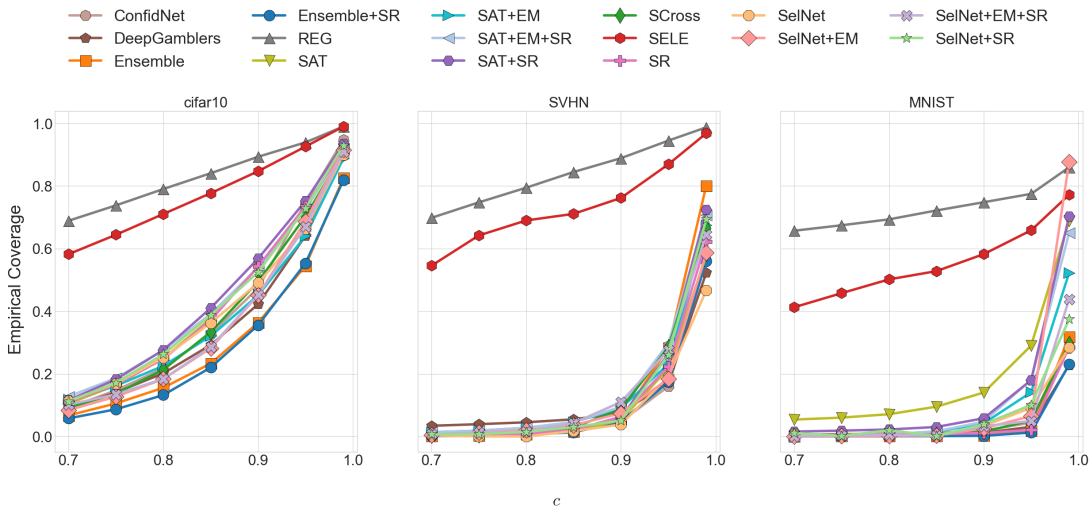


Figure C.4: Q5: Empirical coverage $\hat{\phi}$ for out-of-distribution test sets on 3 image datasets for different target coverages c.

Similarly to the experiments in Section 7.3.3, we observe that under this milder data shift, no baseline drops all the instances at $c = .99$. We can also see that for lower target coverages, we have higher rejection rates, as expected. Moreover, there is a clear worst-performing method, namely Reg.
For `cifar10`, the method dropping more instances is EnsembleSR, reaching an actual coverage of $\approx 5.7\%$ at $c = .70$. The runner-up is Ensemble, accepting only $\approx 6\%$ of instances. All the remaining baselines have an empirical coverage above 8% at $c = .70$.
For `MNIST` and `SVHN` we observe similar patterns: eleven out of sixteen baselines reach a coverage below 1% at $c = .70$. We also highlight that SelNet reaches zero coverage for $c = .75$ and .70 on the `SVHN` dataset.
To conclude, the experiments show the difficulty for current SC methods to properly perform rejection under distribution shifts. For test data close to the (learned) decision boundary, the baselines correctly reject the instances, since all the methods are built to perform ambiguity rejection. For test data far from the decision boundary, the selection function become confident that the shifted instances are very likely to belong to a certain training class, ending up not rejecting the instances. We think that a potential way to mitigate these problems consists of mixing ambiguity rejection with novelty rejection methods, highlighting the need for further research in this direction.

C.2.3 Dataset-level Results

We provide detailed results for all the datasets in Tables C.13-C.56. Each table reports $mean \pm std$ over the 100 bootstrap samples of $\hat{Err}$ and empirical coverage $\hat{\phi}$, for every baseline and every target coverage. For binary datasets, we also include the *MinCoeff* metric introduced in the main text. For error rate, we highlight the baseline column with the lowest average (and standard error in case of ties) in bold case. For *Coverage* and *MinCoeff*, we highlight

in bold the values closest to target coverage c and 1, respectively.

Table C.13: Results for `adult`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.153 ± .003	.136 ± .004	.125 ± .004	**.123 ± .003**	.160 ± .003	**.123 ± .003**	.132 ± .004	.132 ± .004	.132 ± .004	.158 ± .003	.130 ± .003	.125 ± .003	.138 ± .004	.133 ± .004	.133 ± .003	.130 ± .004	.131 ± .004	.131 ± .004
	.95	.154 ± .003	.126 ± .004	.115 ± .004	.108 ± .003	.137 ± .004	**.101 ± .003**	.118 ± .004	.116 ± .004	.114 ± .004	.135 ± .004	.130 ± .003	.103 ± .003	.131 ± .004	.128 ± .004	.123 ± .003	.116 ± .003	.125 ± .004	.133 ± .004
	.90	.142 ± .003	.113 ± .004	.102 ± .004	.098 ± .003	.117 ± .003	**.080 ± .003**	.097 ± .003	.101 ± .003	.096 ± .003	.120 ± .003	.130 ± .003	**.080 ± .003**	.121 ± .004	.119 ± .004	.119 ± .003	.101 ± .003	.114 ± .003	.134 ± .004
	.85	.119 ± .003	.097 ± .003	.088 ± .003	.080 ± .003	.106 ± .003	**.061 ± .003**	.080 ± .003	.083 ± .003	.078 ± .003	.106 ± .003	.128 ± .003	**.061 ± .003**	.106 ± .004	.104 ± .004	.113 ± .003	.086 ± .003	.103 ± .003	.134 ± .004
	.80	.097 ± .003	.079 ± .003	.071 ± .003	.067 ± .003	.082 ± .003	**.046 ± .002**	.067 ± .003	.071 ± .003	.067 ± .003	.082 ± .003	.126 ± .003	.048 ± .003	.091 ± .003	.092 ± .004	.107 ± .004	.075 ± .003	.089 ± .003	.134 ± .004
	.75	.077 ± .003	.062 ± .003	.058 ± .003	.056 ± .003	.076 ± .003	**.030 ± .002**	.056 ± .003	.057 ± .003	.055 ± .003	.064 ± .003	.123 ± .003	.035 ± .002	.074 ± .003	.082 ± .004	.101 ± .004	.064 ± .003	.073 ± .003	.131 ± .004
	.70	.063 ± .003	.055 ± .003	.046 ± .003	.045 ± .003	.066 ± .003	**.020 ± .002**	.044 ± .003	.044 ± .003	.045 ± .003	.065 ± .003	.118 ± .003	.024 ± .002	.056 ± .003	.076 ± .003	.099 ± .004	.055 ± .003	.057 ± .003	.128 ± .004
$\hat{\phi}$	.99	.983 ± .001	.994 ± .001	.967 ± .002	.924 ± .003	.998 ± .000	.980 ± .001	.985 ± .001	**.991 ± .001**	**.991 ± .001**	.994 ± .001	.995 ± .001	.986 ± .001	.980 ± .001	.984 ± .001	.983 ± .001	.992 ± .001	**.991 ± .001**	**.990 ± .001**
	.95	.927 ± .003	.958 ± .002	.925 ± .002	.913 ± .003	.951 ± .002	.912 ± .003	.943 ± .002	.945 ± .002	.939 ± .002	.942 ± .002	.982 ± .001	.917 ± .002	.936 ± .003	.955 ± .002	.923 ± .003	.955 ± .002	.948 ± .002	**.949 ± .002**
	.90	.855 ± .004	.914 ± .003	.875 ± .003	.886 ± .003	.915 ± .003	.842 ± .003	.883 ± .003	.891 ± .003	.883 ± .003	.924 ± .002	.969 ± .002	.840 ± .003	.888 ± .003	.906 ± .003	.878 ± .004	.907 ± .003	**.896 ± .003**	**.896 ± .003**
	.85	.793 ± .004	.855 ± .003	.825 ± .004	.818 ± .003	**.854 ± .003**	.771 ± .004	.824 ± .004	.836 ± .003	.820 ± .003	.855 ± .003	.960 ± .002	.767 ± .004	.839 ± .004	.813 ± .004	.825 ± .004	.859 ± .003	.844 ± .003	.836 ± .004
	.80	.742 ± .004	**.798 ± .004**	.772 ± .004	.775 ± .004	.796 ± .004	.689 ± .004	.771 ± .004	.789 ± .004	.772 ± .004	.795 ± .004	.948 ± .002	.701 ± .004	.790 ± .004	.741 ± .004	.768 ± .004	.815 ± .004	.790 ± .004	.780 ± .004
	.75	.686 ± .004	.732 ± .004	.722 ± .004	.725 ± .004	.731 ± .004	.616 ± .005	.723 ± .004	.734 ± .004	.722 ± .004	.675 ± .004	.931 ± .002	.632 ± .005	**.740 ± .004**	.699 ± .005	.723 ± .004	.768 ± .004	.738 ± .005	.713 ± .005
	.70	.632 ± .005	.679 ± .004	.671 ± .004	.672 ± .004	.683 ± .004	.563 ± .005	.669 ± .004	.679 ± .004	.674 ± .004	.677 ± .004	.909 ± .002	.579 ± .005	**.687 ± .005**	.670 ± .005	.686 ± .005	.728 ± .004	**.687 ± .005**	.659 ± .004
MinCoeff	.99	.948 ± .018	.994 ± .019	.949 ± .020	.835 ± .017	.999 ± .019	.980 ± .019	.983 ± .019	.993 ± .018	.992 ± .019	.995 ± .019	.999 ± .019	.988 ± .019	.999 ± .019	**.997 ± .019**	.994 ± .019	.988 ± .019	.995 ± .019	1.007 ± .019
	.95	.788 ± .018	.946 ± .019	.872 ± .019	.897 ± .019	.941 ± .020	.907 ± .019	.935 ± .019	.942 ± .019	.934 ± .018	.920 ± .019	.977 ± .019	.921 ± .018	.988 ± .020	.981 ± .019	.960 ± .020	.936 ± .019	**.988 ± .019**	1.035 ± .020
	.90	.612 ± .017	.878 ± .019	.770 ± .019	.872 ± .019	.902 ± .019	.810 ± .019	.870 ± .019	.886 ± .019	.871 ± .019	.919 ± .020	.964 ± .019	.827 ± .019	.969 ± .020	.948 ± .019	.944 ± .020	.871 ± .019	**.973 ± .019**	1.068 ± .021
	.85	.490 ± .016	.778 ± .017	.690 ± .016	.789 ± .018	.812 ± .018	.712 ± .018	.794 ± .018	.817 ± .018	.799 ± .018	.813 ± .018	**.955 ± .019**	.726 ± .018	.926 ± .019	.836 ± .019	.904 ± .020	.798 ± .019	.953 ± .020	1.104 ± .022
	.80	.393 ± .014	.675 ± .016	.604 ± .015	.729 ± .018	.801 ± .020	.606 ± .016	.730 ± .018	.762 ± .018	.739 ± .018	.800 ± .019	**.943 ± .019**	.637 ± .017	.872 ± .019	.720 ± .018	.854 ± .020	.731 ± .018	.933 ± .020	1.143 ± .022
	.75	.320 ± .013	.569 ± .015	.533 ± .015	.650 ± .017	.552 ± .016	.533 ± .017	.659 ± .018	.695 ± .018	.666 ± .018	.531 ± .016	**.927 ± .020**	.550 ± .017	.799 ± .019	.633 ± .018	.799 ± .020	.655 ± .017	.907 ± .021	1.184 ± .024
	.70	.261 ± .013	.460 ± .015	.479 ± .016	.550 ± .016	.537 ± .015	.511 ± .018	.582 ± .017	.608 ± .016	.577 ± .017	.541 ± .015	**.903 ± .019**	.504 ± .017	.674 ± .018	.557 ± .017	.771 ± .020	.597 ± .017	.872 ± .021	1.222 ± .025

Table C.14: Results for `aloi`: *mean* $\pm$ *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.035 ± .001	.034 ± .001	.029 ± .001	.039 ± .001	.031 ± .001	.036 ± .001	.033 ± .001	**.027 ± .001**	.037 ± .001	.030 ± .001	.034 ± .001	.032 ± .001	.035 ± .001	.063 ± .002	.063 ± .002	.031 ± .001
	.95	.026 ± .001	.022 ± .001	.017 ± .001	.032 ± .001	.047 ± .001	.018 ± .001	.017 ± .001	**.013 ± .001**	.029 ± .001	.035 ± .001	.022 ± .001	.016 ± .001	.032 ± .001	.056 ± .002	.063 ± .002	.016 ± .001
	.90	.022 ± .001	.013 ± .001	.009 ± .001	.026 ± .001	.044 ± .002	.007 ± .001	.007 ± .001	.006 ± .001	.018 ± .001	.024 ± .001	.014 ± .001	**.005 ± .001**	.030 ± .001	.049 ± .002	.062 ± .002	.006 ± .001
	.85	.019 ± .001	.008 ± .001	.005 ± .000	.023 ± .001	.018 ± .001	**.002 ± .000**	**.002 ± .000**	.004 ± .000	.008 ± .001	.012 ± .001	.008 ± .001	**.002 ± .000**	.030 ± .001	.042 ± .002	.060 ± .002	**.002 ± .000**
	.80	.016 ± .001	.006 ± .001	.003 ± .000	.022 ± .001	.035 ± .001	**.001 ± .000**	**.001 ± .000**	.002 ± .000	.009 ± .001	.011 ± .001	.006 ± .001	**.001 ± .000**	.030 ± .001	.036 ± .001	.059 ± .002	**.001 ± .000**
	.75	.014 ± .001	.004 ± .000	.002 ± .000	.019 ± .001	.032 ± .001	.001 ± .000	.001 ± .000	.001 ± .000	.006 ± .001	.011 ± .001	.002 ± .000	**.000 ± .000**	.029 ± .001	.031 ± .001	.056 ± .002	.001 ± .000
	.70	.012 ± .001	.003 ± .000	.002 ± .000	.019 ± .001	.022 ± .001	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.005 ± .001	.012 ± .001	.001 ± .000	**.000 ± .000**	.028 ± .001	.027 ± .001	.054 ± .002	**.000 ± .000**
$\hat{\phi}$	.99	.991 ± .001	.989 ± .001	.991 ± .001	**.991 ± .001**	.989 ± .001	.991 ± .001	.991 ± .001	**.990 ± .001**	.991 ± .001	.992 ± .001	.991 ± .001	**.990 ± .001**	**.991 ± .001**	.991 ± .001	.991 ± .001	.994 ± .001
	.95	.951 ± .001	**.950 ± .001**	.954 ± .001	**.949 ± .001**	**.951 ± .001**	**.951 ± .001**	.953 ± .001	.956 ± .001	.954 ± .001	.949 ± .001	**.950 ± .002**	.952 ± .002	**.950 ± .001**	.952 ± .002	.948 ± .001	.959 ± .001
	.90	.904 ± .002	**.900 ± .002**	.902 ± .002	**.901 ± .002**	**.901 ± .002**	.902 ± .002	.903 ± .002	.905 ± .002	**.899 ± .002**	.897 ± .002	.899 ± .002	**.901 ± .002**	**.901 ± .002**	.897 ± .002	**.900 ± .002**	.916 ± .002
	.85	.854 ± .003	.852 ± .003	.850 ± .002	.848 ± .003	.848 ± .002	.849 ± .003	.847 ± .002	.852 ± .003	**.850 ± .002**	.855 ± .002	.848 ± .002	.851 ± .002	.857 ± .002	.848 ± .003	**.849 ± .002**	.871 ± .002
	.80	.803 ± .003	.806 ± .003	.798 ± .003	**.800 ± .003**	**.801 ± .003**	**.800 ± .003**	**.801 ± .003**	**.799 ± .003**	.803 ± .003	**.800 ± .003**	**.800 ± .003**	**.800 ± .003**	.806 ± .003	**.800 ± .003**	**.800 ± .003**	.828 ± .002
	.75	.755 ± .003	.754 ± .003	.746 ± .003	**.750 ± .003**	.751 ± .003	.755 ± .003	.751 ± .003	.745 ± .003	.752 ± .003	.752 ± .003	.752 ± .003	.753 ± .003	.753 ± .003	.751 ± .003	.744 ± .003	.781 ± .003
	.70	.706 ± .003	**.702 ± .003**	.697 ± .003	**.701 ± .004**	.704 ± .003	.704 ± .003	**.703 ± .003**	.694 ± .003	**.702 ± .003**	**.702 ± .003**	**.699 ± .003**	**.702 ± .003**	**.701 ± .003**	**.700 ± .004**	.695 ± .003	.738 ± .003

Table C.15: Results for `bank`: *mean* $\pm$ *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.112 ± .003	.089 ± .002	.087 ± .002	.092 ± .002	.092 ± .002	.092 ± .002	.087 ± .002	.087 ± .002	.090 ± .002	.092 ± .002	.092 ± .003	.090 ± .002	**.084 ± .002**	.091 ± .002	.103 ± .003	.090 ± .002	.092 ± .002	.095 ± .003
	.95	.090 ± .003	.073 ± .002	.073 ± .002	.076 ± .002	.075 ± .002	.072 ± .002	**.068 ± .002**	.072 ± .002	.072 ± .002	.073 ± .002	.082 ± .003	.072 ± .002	.071 ± .002	.081 ± .002	.103 ± .003	.073 ± .002	.089 ± .003	.093 ± .003
	.90	.063 ± .002	.054 ± .002	.055 ± .002	.056 ± .002	.056 ± .002	.054 ± .002	**.051 ± .002**	.053 ± .002	.052 ± .002	.055 ± .002	.067 ± .002	.054 ± .002	.055 ± .002	.063 ± .002	.103 ± .003	.055 ± .002	.084 ± .003	.091 ± .003
	.85	.041 ± .002	.038 ± .002	.039 ± .002	.042 ± .002	.041 ± .002	.039 ± .002	**.036 ± .002**	.037 ± .002	.039 ± .002	.042 ± .002	.053 ± .002	.037 ± .002	.041 ± .002	.050 ± .002	.103 ± .003	**.036 ± .002**	.078 ± .003	.087 ± .003
	.80	.028 ± .002	.028 ± .002	.026 ± .002	.027 ± .002	.028 ± .002	.027 ± .002	.026 ± .002	**.025 ± .001**	.027 ± .002	.028 ± .002	.043 ± .002	.027 ± .002	.026 ± .002	.034 ± .002	.101 ± .003	.025 ± .002	.069 ± .003	.081 ± .003
	.75	.019 ± .001	**.016 ± .001**	.018 ± .001	.019 ± .001	.018 ± .001	.019 ± .001	.017 ± .001	.017 ± .001	.019 ± .001	.022 ± .001	.033 ± .002	.018 ± .001	.019 ± .002	.021 ± .002	.099 ± .003	.017 ± .001	.052 ± .003	.071 ± .003
	.70	.013 ± .001	.013 ± .001	**.011 ± .001**	.013 ± .001	.014 ± .001	.012 ± .001	.012 ± .001	**.011 ± .001**	.013 ± .001	.015 ± .001	.023 ± .002	.012 ± .001	.012 ± .001	.015 ± .001	.099 ± .003	.012 ± .001	.037 ± .002	.056 ± .003
$\hat{\phi}$	.99	**.990 ± .001**	**.990 ± .001**	.988 ± .001	.992 ± .001	.988 ± .001	**.991 ± .001**	.987 ± .001	.986 ± .001	.988 ± .001	**.991 ± .001**	**.989 ± .001**	**.989 ± .001**	.986 ± .001	**.990 ± .001**	**.990 ± .001**	**.991 ± .001**	.988 ± .001	**.991 ± .001**
	.95	.952 ± .002	.947 ± .002	**.950 ± .002**	**.949 ± .002**	.947 ± .002	.942 ± .002	.941 ± .003	.946 ± .002	.946 ± .003	.948 ± .002	**.951 ± .002**	.946 ± .002	**.949 ± .002**	**.949 ± .002**	**.950 ± .002**	**.951 ± .002**	.948 ± .002	.946 ± .002
	.90	**.898 ± .003**	.895 ± .003	.897 ± .003	**.899 ± .003**	.897 ± .003	.896 ± .003	.892 ± .003	.892 ± .003	.890 ± .003	.892 ± .003	.904 ± .003	.896 ± .003	**.901 ± .003**	**.898 ± .003**	**.900 ± .003**	**.899 ± .003**	.897 ± .003	.897 ± .003
	.85	.845 ± .004	.846 ± .004	**.850 ± .004**	**.851 ± .004**	**.849 ± .004**	.848 ± .004	.846 ± .004	.847 ± .004	.848 ± .004	.848 ± .004	**.849 ± .004**	.848 ± .004	.856 ± .003	.854 ± .004	.856 ± .004	.842 ± .004	.848 ± .004	.855 ± .004
	.80	.798 ± .004	.803 ± .004	.799 ± .004	**.801 ± .004**	.802 ± .004	**.801 ± .004**	**.800 ± .004**	.800 ± .004	.798 ± .004	**.800 ± .004**	.799 ± .004	**.799 ± .004**	.802 ± .004	.799 ± .004	.810 ± .004	.790 ± .004	.793 ± .005	.807 ± .004
	.75	**.751 ± .004**	.753 ± .004	.755 ± .004	.753 ± .004	.755 ± .004	.754 ± .004	.756 ± .004	.755 ± .004	**.750 ± .004**	.754 ± .004	.749 ± .004	.755 ± .004	.750 ± .004	.751 ± .004	.755 ± .005	.743 ± .004	.746 ± .005	.754 ± .005
	.70	.700 ± .005	.706 ± .004	.704 ± .005	.702 ± .004	.704 ± .004	.701 ± .004	.706 ± .004	.706 ± .005	.702 ± .005	.704 ± .004	**.701 ± .004**	.709 ± .004	.702 ± .005	.702 ± .004	.710 ± .005	.695 ± .004	.690 ± .005	.699 ± .005
MinCoeff	.99	.957 ± .027	.962 ± .028	.959 ± .029	.969 ± .028	.953 ± .028	.986 ± .029	.959 ± .028	.964 ± .028	.958 ± .027	.977 ± .028	.974 ± .028	.974 ± .030	.948 ± .028	.964 ± .028	**.995 ± .029**	.980 ± .029	**.997 ± .029**	1.001 ± .029
	.95	.772 ± .025	.797 ± .027	.815 ± .028	.798 ± .026	.797 ± .026	.829 ± .028	.789 ± .026	.821 ± .026	.824 ± .025	.820 ± .026	.902 ± .028	.855 ± .028	.812 ± .027	.783 ± .026	**.998 ± .030**	.868 ± .026	.976 ± .030	.993 ± .030
	.90	.539 ± .019	.573 ± .024	.605 ± .024	.594 ± .022	.603 ± .022	.690 ± .025	.591 ± .024	.629 ± .024	.585 ± .022	.621 ± .023	.814 ± .029	.684 ± .024	.612 ± .023	.572 ± .022	**1.000 ± .030**	.690 ± .024	.950 ± .032	.985 ± .032
	.85	.354 ± .017	.377 ± .017	.412 ± .019	.355 ± .015	.349 ± .015	.491 ± .021	.420 ± .019	.452 ± .021	.405 ± .018	.381 ± .016	.704 ± .029	.511 ± .023	.465 ± .021	.420 ± .018	**.988 ± .032**	.474 ± .022	.913 ± .032	.963 ± .032
	.80	.236 ± .014	.240 ± .014	.229 ± .013	.231 ± .014	.237 ± .014	.315 ± .017	.269 ± .015	.269 ± .016	.276 ± .016	.277 ± .017	.579 ± .027	.313 ± .019	.223 ± .014	.280 ± .018	**.986 ± .033**	.284 ± .017	.848 ± .032	.925 ± .031
	.75	.161 ± .012	.138 ± .011	.157 ± .010	.161 ± .011	.156 ± .011	.170 ± .014	.148 ± .011	.162 ± .010	.161 ± .011	.238 ± .015	.433 ± .023	.165 ± .014	.162 ± .013	.177 ± .014	**.961 ± .034**	.150 ± .012	.759 ± .031	.856 ± .032
	.70	.108 ± .011	.108 ± .011	.092 ± .010	.109 ± .011	.116 ± .011	.106 ± .010	.102 ± .011	.096 ± .010	.112 ± .011	.131 ± .012	.328 ± .020	.104 ± .011	.104 ± .011	.124 ± .012	**.964 ± .036**	.103 ± .011	.626 ± .028	.770 ± .033

Table C.16: Results for `bloodmnist`: *mean*$\pm$*std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.066 ± .004	**.033 ± .003**	.039 ± .003	.036 ± .003	.045 ± .003	.043 ± .003	.036 ± .003	.039 ± .003	.038 ± .003	.043 ± .003	.037 ± .003	.036 ± .003	.047 ± .004	.064 ± .004	.064 ± .004	.041 ± .003
	.95	.056 ± .004	.023 ± .003	.026 ± .003	.025 ± .003	.034 ± .003	.024 ± .003	.022 ± .003	.026 ± .003	.021 ± .002	.030 ± .003	.022 ± .002	**.019 ± .002**	.035 ± .003	.063 ± .004	.065 ± .004	.026 ± .003
	.90	.043 ± .003	.016 ± .002	.014 ± .002	.015 ± .002	.021 ± .003	.015 ± .002	.014 ± .002	.013 ± .002	.013 ± .002	.018 ± .002	.013 ± .002	**.012 ± .002**	.027 ± .004	.062 ± .004	.065 ± .004	.015 ± .002
	.85	.037 ± .003	.010 ± .002	.008 ± .002	.012 ± .002	.009 ± .002	.010 ± .002	.008 ± .002	**.007 ± .002**	.008 ± .002	.008 ± .002	.008 ± .002	.008 ± .002	.020 ± .004	.063 ± .004	.063 ± .004	.009 ± .002
	.80	.029 ± .003	**.005 ± .001**	.005 ± .002	.007 ± .002	.011 ± .002	**.005 ± .001**	**.005 ± .001**	**.005 ± .001**	**.005 ± .001**	.007 ± .002	**.005 ± .001**	**.005 ± .001**	.016 ± .004	.064 ± .004	.063 ± .004	.005 ± .002
	.75	.025 ± .003	.005 ± .001	.004 ± .001	.006 ± .002	.006 ± .002	.004 ± .001	.005 ± .001	.004 ± .001	.004 ± .001	.005 ± .001	.004 ± .001	**.002 ± .001**	.011 ± .004	.065 ± .004	.066 ± .004	.003 ± .001
	.70	.018 ± .003	.004 ± .001	.002 ± .001	.003 ± .001	.005 ± .001	.004 ± .001	.003 ± .001	.002 ± .001	.002 ± .001	.004 ± .001	.002 ± .001	**.001 ± .001**	.009 ± .003	.066 ± .004	.067 ± .005	.002 ± .001
$\hat{\phi}$	.99	**.990 ± .002**	.984 ± .002	.991 ± .002	**.990 ± .002**	.989 ± .002	.992 ± .002	.991 ± .002	.991 ± .002	.992 ± .002	.988 ± .002	.984 ± .002	**.989 ± .002**	.989 ± .004	.991 ± .002	.993 ± .002	.991 ± .001
	.95	.959 ± .003	.946 ± .004	.946 ± .004	**.948 ± .004**	.947 ± .004	.948 ± .004	.946 ± .004	.955 ± .003	.948 ± .004	**.949 ± .004**	.941 ± .004	.942 ± .004	.947 ± .004	**.950 ± .004**	.955 ± .004	**.951 ± .004**
	.90	.909 ± .005	.896 ± .005	.894 ± .005	**.900 ± .005**	.908 ± .004	**.901 ± .006**	.891 ± .005	.893 ± .005	**.901 ± .005**	.904 ± .005	.894 ± .005	.896 ± .005	.901 ± .006	.893 ± .006	**.901 ± .005**	**.897 ± .005**
	.85	.861 ± .006	.843 ± .006	**.846 ± .005**	.841 ± .006	.842 ± .006	.844 ± .007	.836 ± .006	.846 ± .005	**.847 ± .006**	.840 ± .006	.843 ± .006	.845 ± .007	.850 ± .007	.844 ± .007	.843 ± .006	.844 ± .006
	.80	.814 ± .007	.795 ± .007	**.801 ± .006**	.798 ± .007	.811 ± .006	.795 ± .007	.787 ± .007	**.799 ± .006**	.800 ± .007	.800 ± .007	.802 ± .007	.799 ± .007	.805 ± .007	.788 ± .008	.795 ± .008	.802 ± .007
	.75	.767 ± .007	.751 ± .008	**.747 ± .006**	**.747 ± .007**	.754 ± .007	.754 ± .007	.746 ± .008	**.753 ± .006**	.747 ± .007	.754 ± .007	.749 ± .007	**.751 ± .007**	.757 ± .007	.737 ± .008	.742 ± .008	.759 ± .007
	.70	.705 ± .008	.703 ± .008	.698 ± .007	**.699 ± .007**	.709 ± .008	.705 ± .008	.712 ± .008	**.698 ± .007**	.689 ± .007	**.703 ± .007**	.707 ± .008	.702 ± .008	.702 ± .009	.680 ± .009	.682 ± .009	.712 ± .008

Table C.17: Results for `breastmnist`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	**.137 ± .029**	.161 ± .031	.159 ± .030	.143 ± .031	.177 ± .033	.176 ± .034	.158 ± .031	.145 ± .030	.141 ± .031	.163 ± .033	.163 ± .032	.151 ± .032	.189 ± .035	.189 ± .033	.205 ± .035	.149 ± .031	.147 ± .031	.188 ± .034
	.95	.145 ± .031	.160 ± .031	.158 ± .031	.155 ± .033	.192 ± .034	.164 ± .033	.154 ± .032	.123 ± .027	**.119 ± .029**	.170 ± .034	.145 ± .031	.122 ± .031	.173 ± .037	.186 ± .034	.206 ± .036	.138 ± .030	.146 ± .032	.188 ± .034
	.90	.150 ± .031	.145 ± .031	.135 ± .031	.157 ± .035	.155 ± .033	.161 ± .034	.116 ± .027	.126 ± .028	.108 ± .030	.141 ± .033	.123 ± .029	**.091 ± .028**	.177 ± .038	.182 ± .035	.192 ± .033	.137 ± .031	.132 ± .030	.165 ± .033
	.85	.139 ± .030	.135 ± .031	.129 ± .029	.122 ± .032	.139 ± .033	.109 ± .032	.107 ± .030	.104 ± .028	**.082 ± .028**	.140 ± .033	.112 ± .028	.092 ± .028	.173 ± .039	.155 ± .036	.205 ± .035	.131 ± .032	.133 ± .031	.143 ± .034
	.80	.133 ± .032	.140 ± .032	.119 ± .030	.146 ± .033	.078 ± .025	.094 ± .031	.107 ± .030	.100 ± .029	.095 ± .031	.110 ± .031	.115 ± .029	**.069 ± .024**	.159 ± .039	.162 ± .037	.195 ± .034	.104 ± .029	.117 ± .030	.107 ± .032
	.75	.135 ± .032	.141 ± .032	.108 ± .029	.109 ± .033	.113 ± .030	.099 ± .032	.101 ± .030	.089 ± .027	.088 ± .034	.131 ± .035	.052 ± .022	**.051 ± .022**	.165 ± .040	.167 ± .038	.192 ± .035	.078 ± .026	.084 ± .026	.098 ± .032
	.70	.144 ± .034	.115 ± .030	.111 ± .030	.076 ± .029	.092 ± .029	.105 ± .034	.103 ± .030	.081 ± .026	.064 ± .027	.082 ± .031	**.035 ± .020**	.045 ± .022	.169 ± .041	.142 ± .037	.200 ± .036	.072 ± .025	.072 ± .025	.103 ± .034
$\hat{\phi}$	.99	1.000 ± .000	**994 ± .006**	.962 ± .017	.987 ± .009	.973 ± .015	.961 ± .017	.971 ± .012	.961 ± .015	.969 ± .013	.982 ± .011	**993 ± .006**	.988 ± .008	.974 ± .014	.981 ± .011	.972 ± .015	.975 ± .013	.983 ± .008	1.000 ± .000
	.95	.949 ± .018	.957 ± .017	.931 ± .021	.938 ± .019	.976 ± .012	.947 ± .020	**959 ± .014**	.930 ± .020	.926 ± .021	.918 ± .023	.972 ± .012	.906 ± .023	.857 ± .028	.961 ± .013	.903 ± .025	.947 ± .018	**958 ± .015**	1.000 ± .000
	.90	.917 ± .024	.925 ± .019	.848 ± .031	.840 ± .028	.789 ± .035	.923 ± .024	.889 ± .023	**903 ± .024**	.850 ± .033	.854 ± .032	.879 ± .024	.835 ± .028	.795 ± .035	.879 ± .025	.840 ± .031	.918 ± .020	.905 ± .026	.942 ± .021
	.85	.890 ± .029	.906 ± .020	.842 ± .031	.806 ± .034	.872 ± .022	.824 ± .034	.787 ± .029	**.854 ± .028**	.766 ± .036	.879 ± .023	.816 ± .032	.828 ± .029	.747 ± .038	.818 ± .030	.788 ± .036	.861 ± .029	.898 ± .026	.856 ± .031
	.80	.786 ± .034	.876 ± .024	.805 ± .033	.765 ± .035	.745 ± .034	.749 ± .042	**787 ± .029**	.821 ± .031	.739 ± .038	.815 ± .032	.792 ± .033	.755 ± .036	.698 ± .039	.780 ± .033	.714 ± .039	.828 ± .031	.839 ± .031	.780 ± .040
	.75	.779 ± .035	.826 ± .030	.788 ± .035	.652 ± .041	**.748 ± .032**	.711 ± .044	.767 ± .031	.791 ± .032	.613 ± .040	.746 ± .036	.633 ± .038	.654 ± .042	.672 ± .040	.760 ± .034	.696 ± .039	.764 ± .035	.769 ± .036	.717 ± .043
	.70	.727 ± .038	.727 ± .034	.761 ± .037	.620 ± .043	.706 ± .039	.673 ± .046	.748 ± .033	.784 ± .032	.618 ± .046	.641 ± .043	.598 ± .040	.610 ± .043	.659 ± .040	**.716 ± .037**	.669 ± .039	.746 ± .036	.739 ± .036	.686 ± .044
MinCoeff	.99	1.003 ± .047	1.000 ± .047	.997 ± .048	1.006 ± .047	**1.001 ± .046**	.995 ± .048	1.010 ± .048	.997 ± .049	1.000 ± .048	1.013 ± .046	1.010 ± .046	.998 ± .048	1.009 ± .047	1.022 ± .048	.992 ± .049	1.001 ± .047	1.008 ± .046	1.003 ± .047
	.95	.983 ± .048	.986 ± .049	1.013 ± .048	1.030 ± .048	1.002 ± .047	.990 ± .048	1.005 ± .048	.985 ± .051	.991 ± .050	.986 ± .049	1.002 ± .047	1.000 ± .048	1.058 ± .050	1.034 ± .048	1.001 ± .051	1.011 ± .046	**1.004 ± .046**	1.003 ± .047
	.90	.969 ± .049	1.010 ± .049	1.009 ± .049	1.075 ± .046	1.015 ± .057	.980 ± .049	.995 ± .051	.973 ± .052	1.008 ± .048	1.002 ± .051	1.027 ± .047	1.017 ± .049	1.045 ± .052	1.073 ± .048	1.024 ± .050	1.017 ± .046	**1.012 ± .045**	.988 ± .048
	.85	.967 ± .051	**1.002 ± .050**	1.016 ± .048	1.086 ± .048	1.029 ± .046	.976 ± .049	1.023 ± .053	.970 ± .052	.996 ± .054	1.025 ± .048	1.032 ± .050	1.014 ± .049	1.066 ± .052	1.117 ± .052	1.001 ± .052	1.053 ± .047	**1.019 ± .046**	.980 ± .049
	.80	.947 ± .053	.990 ± .050	**1.000 ± .050**	1.044 ± .050	1.032 ± .055	.960 ± .050	1.023 ± .053	.975 ± .053	1.041 ± .051	.984 ± .053	1.043 ± .049	1.045 ± .050	1.088 ± .051	1.126 ± .054	1.030 ± .051	1.051 ± .047	1.060 ± .047	.954 ± .049
	.75	.943 ± .053	.977 ± .051	**1.011 ± .049**	1.020 ± .053	1.008 ± .056	.949 ± .051	1.014 ± .054	.982 ± .055	1.012 ± .057	.994 ± .051	1.097 ± .052	1.037 ± .056	1.078 ± .053	1.131 ± .052	1.032 ± .053	1.063 ± .048	1.065 ± .048	.942 ± .051
	.70	.926 ± .056	.999 ± .056	**999 ± .050**	1.058 ± .052	1.010 ± .057	.975 ± .051	1.005 ± .054	.991 ± .055	1.002 ± .056	.963 ± .060	1.110 ± .053	1.084 ± .053	1.071 ± .054	1.149 ± .053	1.019 ± .054	1.066 ± .047	1.076 ± .047	.957 ± .051

Table C.18: Results for `catsdogs`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	**.040 ± .003**	**.040 ± .003**	.042 ± .003	.066 ± .003	.060 ± .003	.068 ± .003	**.040 ± .003**	.042 ± .003	.067 ± .003	.060 ± .003	.042 ± .003	.045 ± .003	.052 ± .003	.060 ± .003	.063 ± .004	.082 ± .004	.079 ± .004	.070 ± .004
	.95	**.026 ± .002**	.027 ± .002	.030 ± .003	.051 ± .003	.047 ± .003	.052 ± .003	.027 ± .002	.030 ± .002	.050 ± .003	.046 ± .003	.032 ± .003	.030 ± .002	.044 ± .003	.059 ± .003	.062 ± .004	.071 ± .004	.056 ± .003	.059 ± .003
	.90	**.014 ± .002**	.017 ± .002	.019 ± .002	.023 ± .002	.079 ± .004	.039 ± .003	.018 ± .002	.021 ± .002	.023 ± .002	.083 ± .004	.023 ± .002	.019 ± .002	.034 ± .003	.060 ± .003	.062 ± .004	.057 ± .004	.050 ± .003	.049 ± .003
	.85	**.008 ± .002**	.010 ± .001	.013 ± .002	.023 ± .003	.049 ± .004	.027 ± .003	.010 ± .002	.014 ± .002	.024 ± .003	.047 ± .004	.015 ± .002	.011 ± .002	.026 ± .002	.060 ± .003	.061 ± .004	.049 ± .003	.042 ± .003	.035 ± .003
	.80	**.004 ± .001**	.008 ± .001	.009 ± .002	.029 ± .003	.019 ± .002	.017 ± .002	.007 ± .001	.008 ± .001	.029 ± .003	.019 ± .002	.009 ± .001	.008 ± .001	.018 ± .002	.058 ± .003	.062 ± .004	.044 ± .003	.037 ± .003	.014 ± .002
	.75	**.002 ± .001**	.005 ± .001	.007 ± .001	.088 ± .005	.017 ± .002	.010 ± .002	.005 ± .001	.007 ± .001	.092 ± .005	.016 ± .002	.006 ± .001	.005 ± .001	.014 ± .002	.058 ± .003	.062 ± .004	.039 ± .003	.033 ± .003	.009 ± .001
	.70	**.002 ± .001**	.005 ± .001	.007 ± .001	.008 ± .001	.011 ± .002	.005 ± .001	.005 ± .001	.006 ± .001	.008 ± .001	.011 ± .002	.004 ± .001	.004 ± .001	.011 ± .002	.058 ± .004	.061 ± .004	.035 ± .003	.032 ± .003	.007 ± .001
$\hat{\phi}$	.99	**.991 ± .001**	.986 ± .001	.988 ± .001	.989 ± .001	.994 ± .001	.992 ± .001	.985 ± .002	.988 ± .001	**.989 ± .001**	.993 ± .001	.988 ± .002	.992 ± .001	.992 ± .001	**990 ± .001**	**991 ± .001**	.995 ± .001	.995 ± .001	**989 ± .001**
	.95	.956 ± .003	.946 ± .003	**952 ± .003**	**947 ± .003**	.955 ± .003	**952 ± .003**	.946 ± .003	**951 ± .003**	.945 ± .003	.957 ± .003	.944 ± .003	**951 ± .003**	.954 ± .003	**950 ± .003**	.955 ± .003	.969 ± .003	.966 ± .003	.942 ± .003
	.90	.899 ± .005	.897 ± .004	.904 ± .004	.891 ± .005	.901 ± .004	.912 ± .004	.899 ± .004	.908 ± .004	.890 ± .005	**901 ± .004**	.899 ± .004	.898 ± .004	.907 ± .004	.904 ± .004	.901 ± .005	.936 ± .003	.943 ± .004	.893 ± .005
	.85	.848 ± .006	**850 ± .004**	.852 ± .005	.847 ± .004	.853 ± .004	.861 ± .005	.851 ± .005	.854 ± .005	.850 ± .005	.851 ± .004	.856 ± .005	.852 ± .005	.850 ± .005	.854 ± .005	.844 ± .005	.910 ± .005	.917 ± .004	.849 ± .005
	.80	.800 ± .006	.797 ± .005	**802 ± .005**	.788 ± .005	.796 ± .006	.804 ± .006	.791 ± .005	.806 ± .005	.791 ± .005	.796 ± .006	.811 ± .006	.813 ± .006	.785 ± .006	.801 ± .006	.788 ± .006	.880 ± .006	.893 ± .005	**.800 ± .005**
	.75	.744 ± .006	.742 ± .006	**752 ± .005**	.736 ± .006	.735 ± .005	.750 ± .006	.735 ± .006	.752 ± .005	.739 ± .006	.732 ± .005	.763 ± .006	.761 ± .006	.741 ± .006	.748 ± .007	.733 ± .007	.855 ± .006	.867 ± .006	.757 ± .006
	.70	.693 ± .006	.685 ± .006	.709 ± .006	.712 ± .007	.684 ± .006	.691 ± .007	.688 ± .006	**702 ± .006**	.708 ± .007	.682 ± .006	.712 ± .007	.711 ± .007	.690 ± .006	.699 ± .007	.690 ± .007	.820 ± .006	.848 ± .006	.713 ± .007
MinCoeff	.99	**1.004 ± .017**	**999 ± .017**	1.006 ± .017	**998 ± .017**	**1.002 ± .017**	**997 ± .017**	**999 ± .017**	1.007 ± .017	**999 ± .017**	**1.001 ± .017**	.994 ± .017	**997 ± .017**	1.005 ± .017	1.007 ± .017	**1.002 ± .017**	**999 ± .017**	**999 ± .017**	1.010 ± .017
	.95	**1.001 ± .017**	.989 ± .017	1.012 ± .017	.982 ± .017	**997 ± .017**	.971 ± .017	.989 ± .017	1.012 ± .017	.980 ± .017	1.006 ± .017	.972 ± .017	.968 ± .017	1.016 ± .017	1.021 ± .017	**1.000 ± .017**	.977 ± .017	.978 ± .017	1.031 ± .018
	.90	.989 ± .017	.964 ± .017	1.020 ± .017	**1.002 ± .017**	.929 ± .017	.945 ± .017	.963 ± .017	1.022 ± .017	**1.004 ± .017**	.925 ± .017	.941 ± .017	.932 ± .018	1.029 ± .017	1.027 ± .017	.998 ± .017	.949 ± .017	.959 ± .017	1.061 ± .017
	.85	.977 ± .017	.943 ± .017	1.019 ± .017	.960 ± .018	.890 ± .018	.913 ± .017	.943 ± .017	1.026 ± .018	.963 ± .017	.888 ± .018	.914 ± .018	.901 ± .018	1.030 ± .017	1.031 ± .018	**998 ± .018**	.926 ± .017	.940 ± .017	1.082 ± .018
	.80	.964 ± .017	.917 ± .018	1.020 ± .018	.932 ± .017	.913 ± .018	.879 ± .017	.906 ± .018	1.034 ± .018	.929 ± .017	.914 ± .018	.886 ± .018	.878 ± .018	1.103 ± .018	1.029 ± .018	**1.004 ± .019**	.895 ± .017	.920 ± .017	1.098 ± .018
	.75	.944 ± .017	.910 ± .018	1.013 ± .018	.906 ± .018	.903 ± .019	.864 ± .018	.891 ± .018	1.035 ± .018	.902 ± .018	.915 ± .019	.865 ± .018	.851 ± .018	1.134 ± .018	1.034 ± .018	**1.009 ± .019**	.873 ± .017	.895 ± .017	1.109 ± .018
	.70	.929 ± .018	.933 ± .019	**1.008 ± .018**	.919 ± .019	.907 ± .019	.875 ± .019	.902 ± .018	1.038 ± .019	.922 ± .019	.905 ± .019	.836 ± .018	.816 ± .018	1.165 ± .019	1.039 ± .019	1.014 ± .019	.832 ± .017	.875 ± .017	1.123 ± .018

Table C.19: Results for `chestmnist`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.101 ± .002	.100 ± .002	.100 ± .002	**.099 ± .002**	**.099 ± .002**	**.099 ± .002**	.100 ± .002	**.099 ± .002**	.100 ± .002	.100 ± .002	.102 ± .002	**.099 ± .002**	.101 ± .002	.101 ± .002	.103 ± .002	.100 ± .002	.104 ± .002	.103 ± .002
	.95	.093 ± .002	.090 ± .002	**.089 ± .002**	.091 ± .002	.090 ± .002	.091 ± .002	.090 ± .002	**.089 ± .002**	.090 ± .002	.091 ± .002	.099 ± .002	.090 ± .002	.091 ± .002	.092 ± .002	.102 ± .002	.090 ± .002	.104 ± .002	.103 ± .002
	.90	.086 ± .002	**.080 ± .002**	.081 ± .002	**.080 ± .002**	.082 ± .002	.081 ± .002	**.080 ± .002**	.081 ± .002	.081 ± .002	.084 ± .002	.096 ± .002	.081 ± .002	.082 ± .002	.084 ± .002	.100 ± .002	.081 ± .002	.103 ± .002	.104 ± .002
	.85	.081 ± .002	.074 ± .002	**.073 ± .002**	.074 ± .002	**.073 ± .002**	**.073 ± .002**	.074 ± .002	**.073 ± .002**	.074 ± .002	.097 ± .002	.090 ± .002	**.073 ± .002**	.075 ± .002	.078 ± .002	.099 ± .002	**.073 ± .002**	.103 ± .002	.103 ± .002
	.80	.076 ± .002	**.066 ± .002**	**.066 ± .002**	**.066 ± .002**	.067 ± .002	.067 ± .002	**.066 ± .002**	**.066 ± .002**	**.066 ± .002**	.129 ± .002	.086 ± .002	**.066 ± .002**	.069 ± .002	.071 ± .002	.097 ± .002	**.066 ± .002**	.101 ± .002	.103 ± .002
	.75	.072 ± .002	**.059 ± .002**	.060 ± .002	.061 ± .002	.061 ± .002	**.059 ± .002**	**.059 ± .002**	.060 ± .002	.061 ± .002	.137 ± .003	.083 ± .002	**.059 ± .002**	.064 ± .002	.065 ± .002	.096 ± .002	.061 ± .002	.100 ± .002	.103 ± .002
	.70	.068 ± .002	.055 ± .002	**.054 ± .002**	.055 ± .002	.057 ± .002	.055 ± .002	.055 ± .002	.055 ± .002	.055 ± .002	.204 ± .003	.079 ± .002	**.054 ± .002**	.058 ± .002	.059 ± .002	.095 ± .002	.055 ± .002	.100 ± .003	.100 ± .003
$\hat{\phi}$	.99	.991 ± .001	.989 ± .001	.989 ± .001	.989 ± .001	.989 ± .001	.989 ± .001	.989 ± .001	.988 ± .001	.990 ± .001	.990 ± .001	.987 ± .001	.989 ± .001	.990 ± .001	.990 ± .001	.992 ± .001	.991 ± .001	**.990 ± .001**	.990 ± .001
	.95	.953 ± .001	.952 ± .001	**.949 ± .001**	.952 ± .001	**.949 ± .001**	**.950 ± .001**	.952 ± .001	**.949 ± .001**	.952 ± .001	**.950 ± .002**	.945 ± .002	**.950 ± .001**	**.950 ± .001**	**.950 ± .002**	.953 ± .001	.954 ± .001	**.950 ± .001**	.949 ± .002
	.90	.899 ± .002	.899 ± .002	.903 ± .002	.901 ± .002	.899 ± .002	.898 ± .002	.899 ± .002	.902 ± .002	.901 ± .002	.899 ± .002	.900 ± .002	.902 ± .002	.899 ± .002	.901 ± .002	**.900 ± .002**	.906 ± .002	.903 ± .002	.897 ± .002
	.85	.846 ± .002	**.850 ± .002**	**.850 ± .002**	.854 ± .002	**.849 ± .002**	.852 ± .002	**.850 ± .002**	**.850 ± .002**	.853 ± .002	.857 ± .002	.847 ± .003	**.851 ± .002**	**.850 ± .002**	.852 ± .002	.853 ± .002	.855 ± .002	.853 ± .003	.846 ± .003
	.80	.799 ± .003	.798 ± .003	.802 ± .002	**.799 ± .002**	.804 ± .003	.804 ± .003	.798 ± .003	.802 ± .002	.799 ± .002	.807 ± .002	.798 ± .003	.800 ± .003	**.801 ± .002**	.805 ± .002	.800 ± .002	.805 ± .002	.803 ± .003	.798 ± .003
	.75	**.748 ± .003**	.748 ± .003	.754 ± .002	.748 ± .003	**.751 ± .003**	**.750 ± .003**	.748 ± .003	.754 ± .003	.748 ± .003	.757 ± .003	.748 ± .003	**.751 ± .003**	.752 ± .003	.752 ± .003	.748 ± .003	.756 ± .003	**.751 ± .003**	.750 ± .003
	.70	**.701 ± .003**	**.701 ± .003**	.698 ± .003	.700 ± .003	.702 ± .002	**.702 ± .003**	**.701 ± .003**	**.699 ± .003**	.699 ± .003	.708 ± .003	.701 ± .003	.698 ± .003	.702 ± .003	.698 ± .003	.696 ± .003	.707 ± .003	**.700 ± .003**	.699 ± .003
MinCoeff	.99	.978 ± .020	.971 ± .021	.969 ± .020	.967 ± .020	.965 ± .020	.962 ± .021	.973 ± .021	.969 ± .020	.970 ± .021	.974 ± .020	.993 ± .021	.962 ± .021	.979 ± .020	.981 ± .021	**999 ± .021**	.972 ± .021	1.003 ± .021	1.001 ± .021
	.95	.908 ± .019	.875 ± .019	.864 ± .018	.878 ± .020	.876 ± .019	.879 ± .019	.876 ± .019	.864 ± .018	.879 ± .019	.880 ± .019	.963 ± .022	.871 ± .020	.887 ± .020	.892 ± .020	.984 ± .021	.879 ± .020	1.003 ± .022	**1.001 ± .022**
	.90	.837 ± .018	.782 ± .018	.785 ± .018	.782 ± .018	.801 ± .019	.783 ± .019	.782 ± .018	.785 ± .018	.785 ± .018	.811 ± .019	.928 ± .023	.791 ± .018	.795 ± .019	.819 ± .019	.961 ± .021	.788 ± .018	.997 ± .023	**1.002 ± .022**
	.85	.783 ± .018	.716 ± .018	.706 ± .017	.720 ± .018	.710 ± .017	.711 ± .018	.716 ± .018	.707 ± .017	.719 ± .018	.803 ± .019	.876 ± .023	.705 ± .017	.731 ± .018	.753 ± .018	.956 ± .021	.711 ± .017	.993 ± .023	**997 ± .022**
	.80	.741 ± .019	.645 ± .018	.642 ± .016	.643 ± .016	.655 ± .017	.651 ± .017	.647 ± .018	.642 ± .016	.644 ± .016	.822 ± .021	.839 ± .022	.642 ± .016	.666 ± .018	.687 ± .018	.936 ± .022	.645 ± .018	.980 ± .024	**999 ± .023**
	.75	.698 ± .019	.578 ± .017	.587 ± .017	.589 ± .018	.593 ± .018	.577 ± .017	.580 ± .016	.586 ± .017	.589 ± .018	.822 ± .021	.801 ± .023	.577 ± .018	.618 ± .018	.632 ± .018	.923 ± .022	.597 ± .017	.967 ± .023	**994 ± .023**
	.70	.665 ± .019	.533 ± .016	.527 ± .018	.538 ± .017	.550 ± .016	.533 ± .017	.534 ± .016	.531 ± .018	.537 ± .017	.905 ± .023	.770 ± .023	.521 ± .017	.567 ± .017	.572 ± .018	.911 ± .021	.535 ± .017	.967 ± .024	**969 ± .024**

Table C.20: Results for `cifar10`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.093 ± .002	.088 ± .003	.099 ± .003	.078 ± .002	.064 ± .002	.090 ± .003	.088 ± .003	.098 ± .003	.079 ± .003	.064 ± .002	.059 ± .002	**.058 ± .002**	.091 ± .003	.119 ± .003	.119 ± .003	.088 ± .002
	.95	.074 ± .003	.069 ± .002	.078 ± .002	.073 ± .002	.049 ± .002	.071 ± .003	.067 ± .002	.075 ± .002	.074 ± .002	.046 ± .002	.045 ± .002	**.039 ± .002**	.074 ± .003	.115 ± .003	.119 ± .003	.069 ± .002
	.90	.055 ± .002	.047 ± .002	.056 ± .002	.045 ± .002	.031 ± .002	.051 ± .002	.047 ± .002	.053 ± .002	.045 ± .002	.030 ± .002	.031 ± .002	**.025 ± .002**	.053 ± .002	.112 ± .003	.119 ± .003	.049 ± .002
	.85	.040 ± .002	.033 ± .002	.039 ± .002	.037 ± .002	.019 ± .001	.037 ± .002	.032 ± .002	.037 ± .002	.037 ± .002	.019 ± .001	.021 ± .001	**.014 ± .001**	.039 ± .002	.108 ± .003	.120 ± .003	.032 ± .002
	.80	.029 ± .002	.020 ± .001	.026 ± .002	.028 ± .002	.011 ± .001	.024 ± .002	.022 ± .001	.026 ± .002	.026 ± .002	.013 ± .001	.013 ± .001	**.009 ± .001**	.025 ± .002	.105 ± .003	.120 ± .003	.021 ± .001
	.75	.020 ± .002	.015 ± .001	.019 ± .001	.019 ± .001	.007 ± .001	.015 ± .002	.014 ± .001	.017 ± .001	.018 ± .001	.008 ± .001	.008 ± .001	**.006 ± .001**	.018 ± .001	.101 ± .003	.121 ± .003	.015 ± .001
	.70	.012 ± .001	.009 ± .001	.014 ± .001	.011 ± .001	.006 ± .001	.011 ± .001	.010 ± .001	.011 ± .001	.011 ± .001	.008 ± .001	.005 ± .001	**.003 ± .001**	.013 ± .001	.098 ± .003	.121 ± .003	.010 ± .001
$\hat{\phi}$	.99	.992 ± .001	**989 ± .001**	**989 ± .001**	**990 ± .001**	**990 ± .001**	.992 ± .001	**990 ± .001**	**990 ± .001**	**991 ± .001**	**990 ± .001**	.987 ± .001	**989 ± .001**	.992 ± .001	.995 ± .001	**990 ± .001**	.988 ± .001
	.95	**.951 ± .002**	**950 ± .002**	.947 ± .002	.945 ± .002	**951 ± .002**	.952 ± .002	.946 ± .002	.942 ± .002	**949 ± .002**	.947 ± .002	.946 ± .002	.948 ± .002	.956 ± .002	.954 ± .002	.942 ± .002	.948 ± .002
	.90	.895 ± .002	.901 ± .003	.893 ± .003	.894 ± .003	.900 ± .003	.902 ± .002	.901 ± .003	.892 ± .003	.896 ± .003	**.900 ± .003**	.896 ± .003	.899 ± .003	.902 ± .002	.908 ± .003	.894 ± .003	.897 ± .003
	.85	**.848 ± .003**	.854 ± .003	.842 ± .003	**.852 ± .003**	.847 ± .003	.854 ± .003	.855 ± .003	.845 ± .003	.855 ± .003	.846 ± .003	.841 ± .004	.844 ± .003	.858 ± .003	.857 ± .003	**.848 ± .003**	.844 ± .003
	.80	**.799 ± .003**	**800 ± .003**	.790 ± .003	**.799 ± .004**	**.801 ± .003**	**.799 ± .003**	.805 ± .003	**.799 ± .003**	.796 ± .004	**.800 ± .004**	.791 ± .004	.796 ± .004	.803 ± .003	.809 ± .003	.804 ± .004	.791 ± .004
	.75	**.751 ± .003**	**751 ± .003**	.745 ± .003	.756 ± .004	.756 ± .004	.752 ± .003	**.752 ± .003**	.746 ± .003	.754 ± .004	.756 ± .004	**.750 ± .004**	**.751 ± .004**	.756 ± .004	.758 ± .004	**.750 ± .004**	.742 ± .004
	.70	.694 ± .003	.695 ± .004	**.701 ± .004**	.695 ± .004	.710 ± .004	.703 ± .004	.697 ± .004	.696 ± .004	.696 ± .004	.708 ± .004	**.701 ± .004**	**.702 ± .004**	.704 ± .004	.711 ± .004	.698 ± .004	.689 ± .004

Table C.21: Results for `compass`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.310 ± .008	.298 ± .008	.301 ± .008	.293 ± .007	.287 ± .008	.292 ± .007	.297 ± .008	.302 ± .008	.293 ± .007	.288 ± .008	.289 ± .007	**.285 ± .007**	.297 ± .007	.296 ± .008	.307 ± .008	.300 ± .008	.300 ± .008	.292 ± .007
	.95	.303 ± .009	.293 ± .008	.296 ± .009	.290 ± .008	.290 ± .007	.284 ± .007	.293 ± .008	.295 ± .008	.284 ± .007	.287 ± .007	.289 ± .007	**.275 ± .008**	.282 ± .008	.292 ± .008	.302 ± .008	.291 ± .008	.285 ± .008	.284 ± .008
	.90	.294 ± .009	.287 ± .008	.288 ± .009	.276 ± .008	.280 ± .008	.272 ± .007	.282 ± .008	.286 ± .008	.272 ± .007	.282 ± .008	.282 ± .008	**.264 ± .008**	.276 ± .008	.283 ± .008	.300 ± .008	.273 ± .008	.269 ± .008	.272 ± .007
	.85	.290 ± .009	.275 ± .009	.279 ± .009	.273 ± .008	.267 ± .008	.260 ± .007	.271 ± .008	.276 ± .008	.262 ± .008	.266 ± .008	.282 ± .008	.257 ± .008	.266 ± .008	.276 ± .009	.298 ± .008	.260 ± .008	**.255 ± .008**	.265 ± .007
	.80	.282 ± .009	.266 ± .008	.274 ± .009	.262 ± .008	.257 ± .009	.252 ± .008	.262 ± .008	.271 ± .008	.255 ± .008	.257 ± .009	.279 ± .008	**.245 ± .008**	.256 ± .008	.274 ± .009	.295 ± .009	.247 ± .008	.245 ± .009	.255 ± .008
	.75	.280 ± .009	.255 ± .009	.262 ± .009	.250 ± .009	.246 ± .008	.244 ± .008	.250 ± .009	.262 ± .009	.243 ± .008	.247 ± .009	.273 ± .008	**.234 ± .008**	.248 ± .008	.269 ± .009	.290 ± .009	.237 ± .009	.236 ± .009	.248 ± .008
	.70	.274 ± .009	.243 ± .009	.253 ± .009	.233 ± .009	.237 ± .009	.231 ± .008	.234 ± .009	.252 ± .009	.229 ± .009	.234 ± .009	.271 ± .009	**.221 ± .008**	.235 ± .009	.257 ± .010	.283 ± .009	.231 ± .009	.229 ± .009	.243 ± .008
$\hat{\phi}$	.99	**990 ± .002**	.988 ± .002	.987 ± .002	.986 ± .002	.988 ± .002	.995 ± .001	.991 ± .002	**990 ± .002**	.987 ± .002	**990 ± .002**	.989 ± .002	**991 ± .002**	.985 ± .002	.993 ± .001	**990 ± .002**	.986 ± .002	.989 ± .002	.989 ± .002
	.95	.954 ± .001	.962 ± .003	.945 ± .004	.953 ± .004	.955 ± .004	.953 ± .004	.953 ± .003	.951 ± .004	.941 ± .004	.948 ± .004	.947 ± .004	**.951 ± .004**	.932 ± .005	.946 ±

Table C.22: Results for `covtype`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.036 ± .001	.028 ± .000	.029 ± .000	.027 ± .000	.031 ± .000	.041 ± .001	.027 ± .000	.028 ± .000	**.025 ± .000**	.029 ± .000	.029 ± .000	.027 ± .000	.035 ± .000	.070 ± .001	.056 ± .001	.037 ± .000
	.95	.029 ± .001	.016 ± .000	.017 ± .000	.015 ± .000	.020 ± .000	.027 ± .000	.013 ± .000	.014 ± .000	**.012 ± .000**	.017 ± .000	.020 ± .000	.015 ± .000	.025 ± .000	.069 ± .001	.057 ± .001	.024 ± .000
	.90	.023 ± .000	.007 ± .000	.009 ± .000	.009 ± .000	.011 ± .000	.016 ± .000	.006 ± .000	.006 ± .000	**.005 ± .000**	.008 ± .000	.011 ± .000	.007 ± .000	.018 ± .000	.067 ± .001	.058 ± .001	.015 ± .000
	.85	.018 ± .000	.004 ± .000	.005 ± .000	.005 ± .000	.007 ± .000	.010 ± .000	.003 ± .000	.003 ± .000	**.002 ± .000**	.005 ± .000	.006 ± .000	.004 ± .000	.014 ± .000	.065 ± .001	.059 ± .001	.009 ± .000
	.80	.014 ± .000	.002 ± .000	.003 ± .000	.002 ± .000	.004 ± .000	.007 ± .000	.002 ± .000	.002 ± .000	**.001 ± .000**	.003 ± .000	.003 ± .000	.002 ± .000	.011 ± .000	.064 ± .001	.059 ± .001	.006 ± .000
	.75	.011 ± .000	.002 ± .000	.002 ± .000	**.001 ± .000**	.002 ± .000	.005 ± .000	**.001 ± .000**	**.001 ± .000**	**.001 ± .000**	.007 ± .000	.002 ± .000	**.001 ± .000**	.009 ± .000	.062 ± .001	.059 ± .001	.004 ± .000
	.70	.009 ± .000	**.001 ± .000**	**.001 ± .000**	**.001 ± .000**	**.001 ± .000**	.003 ± .000	**.001 ± .000**	**.001 ± .000**	**.001 ± .000**	.030 ± .001	**.001 ± .000**	**.001 ± .000**	.007 ± .000	.060 ± .001	.059 ± .001	.002 ± .000
$\hat{\phi}$	.99	.989 ± .000	.989 ± .000	**.990 ± .000**	**.990 ± .000**	**.990 ± .000**	.989 ± .000	.991 ± .000	**.990 ± .000**	**.990 ± .000**	**.990 ± .000**	.991 ± .000	**.990 ± .000**	**.990 ± .000**	.989 ± .000	.989 ± .000	.992 ± .000
	.95	.950 ± .001	.948 ± .001	.950 ± .001	**.950 ± .001**	.950 ± .001	.949 ± .001	.948 ± .001	.951 ± .001	**.950 ± .001**	.951 ± .001	.950 ± .001	.950 ± .001	.948 ± .001	.948 ± .001	.949 ± .001	.955 ± .001
	.90	.901 ± .001	.898 ± .001	.899 ± .001	**.900 ± .001**	.898 ± .001	.899 ± .001	.899 ± .001	.899 ± .001	.899 ± .001	.898 ± .001	.900 ± .001	.901 ± .001	.898 ± .001	.899 ± .001	.900 ± .001	.911 ± .001
	.85	**.850 ± .001**	**.849 ± .001**	**.849 ± .001**	**.851 ± .001**	**.849 ± .001**	**.849 ± .001**	**.849 ± .001**	**.849 ± .001**	**.850 ± .001**	**.849 ± .001**	**.851 ± .001**	**.850 ± .001**	.847 ± .001	.848 ± .001	**.849 ± .001**	.866 ± .001
	.80	.797 ± .001	.799 ± .001	**.799 ± .001**	**.799 ± .001**	.798 ± .001	**.801 ± .001**	**.799 ± .001**	.798 ± .001	**.800 ± .001**	.797 ± .001	**.801 ± .001**	**.801 ± .001**	.798 ± .001	.796 ± .001	**.800 ± .001**	.820 ± .001
	.75	.748 ± .001	**.749 ± .001**	**.749 ± .001**	.749 ± .001	.748 ± .001	**.750 ± .001**	.749 ± .001	**.750 ± .001**	**.750 ± .001**	.748 ± .001	**.751 ± .001**	**.750 ± .001**	**.750 ± .001**	.746 ± .001	.752 ± .002	.775 ± .001
	.70	.698 ± .001	**.699 ± .001**	**.700 ± .001**	**.700 ± .002**	.697 ± .001	**.699 ± .001**	**.700 ± .001**	**.701 ± .001**	**.699 ± .001**	**.700 ± .001**	**.701 ± .001**	**.700 ± .001**	**.700 ± .001**	.698 ± .001	.699 ± .002	.728 ± .001

Table C.23: Results for `dermamnist`: *mean*±*std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.269 ± .010	.229 ± .009	.239 ± .009	.243 ± .010	.241 ± .010	.228 ± .009	.232 ± .009	.236 ± .009	.239 ± .011	.240 ± .010	.226 ± .009	**.223 ± .009**	.245 ± .010	.270 ± .009	.273 ± .009	.231 ± .009
	.95	.250 ± .010	.217 ± .009	.230 ± .009	.227 ± .010	.227 ± .010	.207 ± .009	.214 ± .009	.222 ± .009	.224 ± .011	.221 ± .009	.213 ± .008	**.206 ± .009**	.238 ± .010	.263 ± .010	.267 ± .010	.214 ± .008
	.90	.213 ± .010	.202 ± .009	.212 ± .010	.207 ± .010	.210 ± .010	.191 ± .009	.194 ± .009	.196 ± .009	.195 ± .011	.206 ± .010	.194 ± .009	**.182 ± .009**	.218 ± .010	.259 ± .010	.263 ± .011	.193 ± .008
	.85	.194 ± .010	.180 ± .009	.201 ± .009	.198 ± .010	.198 ± .010	.177 ± .009	.173 ± .009	.174 ± .009	.182 ± .010	.180 ± .010	.175 ± .009	**.161 ± .009**	.200 ± .010	.252 ± .010	.263 ± .011	.166 ± .009
	.80	.180 ± .010	.159 ± .009	.179 ± .009	.192 ± .010	.183 ± .011	.152 ± .009	.151 ± .009	.162 ± .009	.202 ± .010	.199 ± .010	.159 ± .010	.147 ± .008	.179 ± .010	.242 ± .011	.261 ± .011	**.146 ± .008**
	.75	.163 ± .010	.144 ± .009	.162 ± .010	.164 ± .011	.165 ± .010	.129 ± .009	.129 ± .009	.140 ± .009	.182 ± .010	.185 ± .010	.146 ± .010	**.123 ± .008**	.161 ± .011	.224 ± .011	.258 ± .012	.126 ± .008
	.70	.143 ± .011	.120 ± .009	.143 ± .009	.146 ± .010	.148 ± .010	.114 ± .008	.110 ± .009	.119 ± .009	.172 ± .010	.165 ± .010	.130 ± .010	**.108 ± .008**	.147 ± .010	.203 ± .010	.254 ± .011	.109 ± .009
$\hat{\phi}$	.99	.994 ± .002	.983 ± .003	.993 ± .002	.993 ± .002	.992 ± .002	**.991 ± .002**	.989 ± .002	**.989 ± .002**	**.990 ± .002**	.992 ± .002	.994 ± .002	.994 ± .002	**.991 ± .002**	.989 ± .002	.993 ± .002	**.989 ± .002**
	.95	.962 ± .004	.946 ± .004	.965 ± .004	.955 ± .005	.964 ± .004	**.947 ± .005**	**.949 ± .005**	**.950 ± .005**	.954 ± .004	.959 ± .004	.956 ± .005	.960 ± .004	.967 ± .004	.933 ± .007	.957 ± .005	**.952 ± .004**
	.90	.898 ± .008	**.900 ± .006**	.906 ± .007	.912 ± .006	.920 ± .006	.905 ± .006	.896 ± .007	.892 ± .008	.898 ± .007	.907 ± .007	.896 ± .006	.910 ± .006	.913 ± .007	.892 ± .008	.905 ± .006	.905 ± .006
	.85	.862 ± .008	.849 ± .008	.868 ± .007	.866 ± .008	.882 ± .007	.865 ± .007	.851 ± .008	.845 ± .008	.852 ± .008	.857 ± .007	.837 ± .008	.852 ± .008	.871 ± .008	.850 ± .008	.870 ± .008	**.849 ± .007**
	.80	.820 ± .009	.785 ± .009	.818 ± .008	.820 ± .009	.823 ± .009	.797 ± .009	.796 ± .009	.816 ± .009	.800 ± .010	.807 ± .010	.789 ± .008	.809 ± .008	.818 ± .009	.803 ± .008	.818 ± .009	**.800 ± .008**
	.75	.760 ± .009	.746 ± .010	.757 ± .008	.757 ± .010	.764 ± .009	.739 ± .010	.749 ± .009	.760 ± .011	.760 ± .011	.763 ± .010	.730 ± .009	**.750 ± .009**	.766 ± .010	**.748 ± .009**	.777 ± .009	.749 ± .009
	.70	.709 ± .010	.687 ± .010	.705 ± .010	.711 ± .011	.705 ± .010	**.702 ± .010**	.692 ± .010	.698 ± .011	.720 ± .010	.703 ± .010	.684 ± .009	**.702 ± .010**	.711 ± .010	.694 ± .010	.736 ± .010	.681 ± .010

Table C.24: Results for `electricity`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.248 ± .004	.176 ± .005	.163 ± .005	.179 ± .004	.187 ± .005	.174 ± .004	.176 ± .005	.162 ± .005	.179 ± .004	.185 ± .005	.162 ± .005	**.161 ± .005**	.167 ± .004	.206 ± .005	.200 ± .004	.169 ± .005	.170 ± .005	.174 ± .004
	.95	.245 ± .004	.164 ± .004	.152 ± .004	.172 ± .005	.174 ± .004	.162 ± .004	.162 ± .004	**.149 ± .005**	.166 ± .004	.170 ± .004	.159 ± .005	.150 ± .004	.155 ± .004	.198 ± .005	.196 ± .004	.158 ± .004	.161 ± .005	.166 ± .004
	.90	.238 ± .005	.151 ± .004	.138 ± .004	.161 ± .004	.162 ± .005	.147 ± .004	.145 ± .004	.135 ± .004	.155 ± .005	.160 ± .004	.155 ± .005	**.134 ± .004**	.139 ± .004	.190 ± .005	.193 ± .004	.144 ± .004	.148 ± .005	.158 ± .005
	.85	.227 ± .005	.135 ± .004	.125 ± .004	.149 ± .005	.143 ± .005	.135 ± .004	.133 ± .004	**.120 ± .004**	.139 ± .005	.139 ± .004	.148 ± .005	**.120 ± .004**	.129 ± .004	.182 ± .005	.190 ± .005	.130 ± .004	.136 ± .004	.143 ± .004
	.80	.217 ± .005	.121 ± .004	.112 ± .004	.136 ± .004	.138 ± .005	.119 ± .004	.117 ± .004	**.107 ± .004**	.125 ± .004	.138 ± .005	.140 ± .005	**.107 ± .004**	.117 ± .004	.176 ± .005	.187 ± .005	.117 ± .004	.121 ± .004	.130 ± .005
	.75	.203 ± .005	.110 ± .004	.102 ± .004	.125 ± .005	.125 ± .005	.107 ± .004	.101 ± .004	.096 ± .004	.113 ± .004	.125 ± .005	.131 ± .004	**.093 ± .004**	.109 ± .004	.169 ± .005	.184 ± .005	.104 ± .004	.106 ± .004	.118 ± .004
	.70	.186 ± .005	.096 ± .004	.088 ± .004	.124 ± .004	.116 ± .005	.097 ± .004	.089 ± .004	**.084 ± .004**	.112 ± .005	.117 ± .005	.127 ± .005	.085 ± .004	.098 ± .004	.160 ± .005	.178 ± .005	.095 ± .004	.092 ± .004	.105 ± .004
$\hat{\phi}$	.99	**.990 ± .001**	**.991 ± .001**	.993 ± .001	.992 ± .001	.995 ± .001	**.990 ± .001**	**.989 ± .001**	**.990 ± .001**	**.990 ± .001**	.992 ± .001	**.990 ± .001**	.992 ± .001	.988 ± .001	**.990 ± .001**	**.989 ± .001**	.992 ± .001	**.990 ± .001**	**.989 ± .001**
	.95	.949 ± .003	.949 ± .003	.951 ± .002	.961 ± .002	.958 ± .002	.950 ± .003	.949 ± .003	.954 ± .002	**.951 ± .002**	.953 ± .003	.955 ± .002	.951 ± .003	.950 ± .003	.953 ± .002	.948 ± .003	.954 ± .002	.949 ± .002	.951 ± .002
	.90	.898 ± .004	**.901 ± .004**	.902 ± .004	.911 ± .003	.907 ± .004	**.899 ± .003**	.896 ± .004	.905 ± .004	.908 ± .003	**.902 ± .004**	.909 ± .003	**.900 ± .004**	**.901 ± .004**	.908 ± .004	**.900 ± .003**	.909 ± .004	**.900 ± .003**	.903 ± .004
	.85	**.849 ± .004**	.850 ± .004	.859 ± .004	.859 ± .004	.852 ± .004	.856 ± .004	.851 ± .004	.852 ± .004	.854 ± .004	.846 ± .004	.860 ± .004	.852 ± .004	.853 ± .004	.853 ± .005	.844 ± .004	.863 ± .004	.856 ± .004	.856 ± .005
	.80	**.801 ± .004**	.803 ± .005	.817 ± .005	.817 ± .004	.807 ± .005	.807 ± .005	**.798 ± .005**	.805 ± .005	.806 ± .004	**.801 ± .005**	.807 ± .004	.803 ± .005	**.802 ± .005**	.796 ± .005	.795 ± .005	.813 ± .005	.808 ± .005	**.802 ± .005**
	.75	**.752 ± .005**	.763 ± .005	.769 ± .005	.765 ± .005	.754 ± .005	.760 ± .005	**.751 ± .005**	.761 ± .005	**.749 ± .005**	**.750 ± .005**	.756 ± .004	.752 ± .005	.757 ± .005	**.749 ± .005**	**.750 ± .005**	.767 ± .005	.761 ± .006	.758 ± .004
	.70	.703 ± .005	.716 ± .005	.719 ± .006	.710 ± .005	.721 ± .005	.717 ± .005	.706 ± .005	.722 ± .006	.712 ± .005	.711 ± .005	.712 ± .005	.712 ± .006	.715 ± .005	**.700 ± .005**	.692 ± .005	.719 ± .006	.709 ± .006	.712 ± .005
MinCoeff	.99	**1.003 ± .011**	**1.000 ± .011**	**1.001 ± .011**	**1.000 ± .011**	**1.000 ± .011**	.998 ± .011	**1.002 ± .011**	**1.002 ± .011**	.998 ± .011	**1.001 ± .011**	1.003 ± .011	**1.001 ± .011**	.998 ± .011	**1.000 ± .011**	1.004 ± .011	**.999 ± .011**	**1.002 ± .011**	**1.002 ± .011**
	.95	1.021 ± .011	1.003 ± .011	1.004 ± .011	.992 ± .011	**1.002 ± .011**	**1.000 ± .011**	1.010 ± .011	1.005 ± .011	.997 ± .011	1.004 ± .011	1.008 ± .011	1.002 ± .011	.999 ± .011	**1.001 ± .011**	1.006 ± .011	.997 ± .011	1.013 ± .011	1.012 ± .011
	.90	1.037 ± .011	1.006 ± .012	1.007 ± .012	.991 ± .011	1.009 ± .012	1.006 ± .011	1.009 ± .012	1.014 ± .011	.993 ± .011	1.009 ± .012	1.017 ± .011	1.005 ± .012	1.003 ± .012	**1.000 ± .011**	1.007 ± .012	1.000 ± .011	1.024 ± .011	1.031 ± .012
	.85	1.049 ± .012	1.011 ± .012	1.013 ± .012	**.999 ± .012**	1.009 ± .012	1.008 ± .012	1.018 ± .012	1.019 ± .012	1.005 ± .012	1.009 ± .013	1.027 ± .011	1.003 ± .012	1.006 ± .012	1.012 ± .012	1.004 ± .012	**1.004 ± .011**	1.031 ± .011	1.042 ± .012
	.80	1.060 ± .012	1.013 ± .013	1.017 ± .012	**.997 ± .012**	1.026 ± .012	1.012 ± .012	1.021 ± .012	1.024 ± .012	1.006 ± .011	1.030 ± .012	1.035 ± .011	1.008 ± .013	1.007 ± .013	1.025 ± .013	1.007 ± .013	1.006 ± .012	1.045 ± .012	1.057 ± .012
	.75	1.066 ± .013	1.025 ± .013	1.024 ± .012	**1.002 ± .012**	1.036 ± .012	1.016 ± .012	1.029 ± .013	1.030 ± .012	1.013 ± .012	1.035 ± .013	1.036 ± .012	1.011 ± .013	1.010 ± .013	1.039 ± .014	1.005 ± .013	1.003 ± .013	1.062 ± .012	1.074 ± .012
	.70	1.067 ± .014	1.033 ± .014	1.028 ± .013	1.002 ± .014	1.050 ± .013	1.020 ± .013	1.034 ± .013	1.033 ± .012	1.008 ± .013	1.052 ± .013	1.042 ± .013	1.014 ± .013	1.013 ± .013	1.057 ± .014	.988 ± .014	**1.002 ± .013**	1.073 ± .013	1.097 ± .012

Table C.25: Results for `eye`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.433 ± .013	.441 ± .014	.415 ± .012	.419 ± .012	.413 ± .013	.425 ± .013	.444 ± .014	.414 ± .012	.415 ± .012	.414 ± .013	.427 ± .013	.421 ± .013	.414 ± .012	**.411 ± .013**	.412 ± .013	.426 ± .013	.425 ± .013	.427 ± .013
	.95	.435 ± .013	.436 ± .015	.417 ± .012	.414 ± .012	.418 ± .012	.420 ± .014	.436 ± .015	.412 ± .012	.415 ± .012	.411 ± .013	.439 ± .014	.419 ± .013	.414 ± .012	**.405 ± .013**	.413 ± .013	.421 ± .013	.421 ± .013	.421 ± .014
	.90	.435 ± .014	.433 ± .016	.415 ± .012	.422 ± .012	.413 ± .013	.413 ± .014	.430 ± .015	.411 ± .013	.412 ± .012	.406 ± .013	.451 ± .014	.412 ± .013	.408 ± .012	**.398 ± .013**	.412 ± .013	.417 ± .012	.418 ± .014	.412 ± .013
	.85	.435 ± .014	.429 ± .016	.410 ± .013	.413 ± .013	.416 ± .014	.407 ± .014	.431 ± .015	.397 ± .013	**.393 ± .012**	.406 ± .013	.457 ± .015	.407 ± .013	.405 ± .012	.400 ± .013	.415 ± .013	.410 ± .014	.411 ± .015	.409 ± .014
	.80	.437 ± .014	.429 ± .017	.412 ± .013	.406 ± .013	.424 ± .013	.401 ± .014	.424 ± .016	.394 ± .013	.398 ± .013	**.390 ± .013**	.452 ± .015	.403 ± .014	.400 ± .013	.399 ± .014	.412 ± .014	.402 ± .014	.407 ± .015	.402 ± .014
	.75	.437 ± .014	.431 ± .017	.401 ± .014	.403 ± .013	.419 ± .013	.394 ± .014	.421 ± .016	**.385 ± .013**	.391 ± .013	.387 ± .014	.443 ± .016	.396 ± .014	.391 ± .014	.395 ± .015	.410 ± .015	.394 ± .014	.400 ± .015	.399 ± .014
	.70	.433 ± .014	.424 ± .018	.399 ± .014	.399 ± .014	.416 ± .014	.389 ± .015	.412 ± .017	**.377 ± .014**	.384 ± .014	.389 ± .014	.443 ± .017	.393 ± .014	.382 ± .014	.386 ± .016	.407 ± .015	.392 ± .014	.397 ± .015	.387 ± .014
$\hat{\phi}$	.99	.994 ± .002	.983 ± .004	.993 ± .002	.985 ± .003	**.989 ± .002**	.993 ± .002	.992 ± .002	.987 ± .003	**.990 ± .003**	.985 ± .003	.981 ± .004	.992 ± .002	**.989 ± .003**	.995 ± .002	**.989 ± .002**	.988 ± .003	**.991 ± .002**	.985 ± .003
	.95	.947 ± .006	.943 ± .006	.947 ± .005	.950 ± .006	.947 ± .005	.968 ± .005	.959 ± .005	.938 ± .006	.954 ± .006	.949 ± .006	.928 ± .007	.951 ± .006	.953 ± .005	.941 ± .007	.945 ± .006	.954 ± .006	**.950 ± .005**	.941 ± .006
	.90	.889 ± .008	.889 ± .008	.886 ± .008	.904 ± .008	**.902 ± .007**	.898 ± .008	.915 ± .007	.895 ± .009	.896 ± .008	**.897 ± .007**	.872 ± .008	.893 ± .008	.895 ± .007	.883 ± .009	.908 ± .007	.907 ± .008	.911 ± .007	.894 ± .008
	.85	.847 ± .009	.846 ± .010	.853 ± .008	.855 ± .009	.862 ± .030	.860 ± .009	.848 ± .009	.833 ± .010	.855 ± .009	.856 ± .009	.837 ± .009	.855 ± .009	.861 ± .009	.825 ± .010	**.851 ± .008**	.862 ± .009	.855 ± .008	.858 ± .009
	.80	.811 ± .010	.787 ± .011	**.796 ± .009**	.796 ± .010	.815 ± .010	.809 ± .010	.799 ± .011	.784 ± .010	**.795 ± .009**	.792 ± .011	.779 ± .010	.806 ± .010	.790 ± .010	.775 ± .011	.787 ± .010	.816 ± .010	.808 ± .010	.814 ± .010
	.75	.758 ± .012	.731 ± .012	.746 ± .010	.753 ± .011	.760 ± .012	.762 ± .011	.757 ± .010	**.747 ± .011**	.737 ± .010	.748 ± .011	.727 ± .011	.759 ± .010	.741 ± .011	.738 ± .011	.725 ± .011	.772 ± .011	.751 ± .011	.762 ± .011
	.70	.716 ± .013	.687 ± .012	.707 ± .011	.715 ± .011	.712 ± .012	.694 ± .011	.706 ± .010	.697 ± .012	.692 ± .012	.696 ± .013	.679 ± .011	.717 ± .011	.682 ± .013	.686 ± .013	.680 ± .012	.717 ± .011	.715 ± .011	**.702 ± .011**
MinCoeff	.99	.999 ± .029	1.007 ± .030	.999 ± .029	.997 ± .029	1.001 ± .029	**1.005 ± .029**	1.003 ± .029	1.001 ± .030	.996 ± .030	1.003 ± .030	1.012 ± .030	1.007 ± .029	1.006 ± .029	1.003 ± .030	1.001 ± .029	1.004 ± .030	1.001 ± .030	1.000 ± .030
	.95	.989 ± .030	**.996 ± .029**	**.996 ± .029**	.993 ± .031	.985 ± .030	1.004 ± .030	.991 ± .030	.996 ± .030	.999 ± .030	.997 ± .030	1.045 ± .031	1.009 ± .029	1.007 ± .030	.997 ± .030	.995 ± .029	1.009 ± .030	.999 ± .030	1.006 ± .030
	.90	.981 ± .031	.999 ± .031	.979 ± .031	.977 ± .032	.981 ± .029	.998 ± .031	.995 ± .030	1.001 ± .031	.991 ± .032	**1.009 ± .030**	1.080 ± .031	1.011 ± .030	.996 ± .031	.993 ± .031	.995 ± .030	1.010 ± .030	.999 ± .031	1.015 ± .030
	.85	.972 ± .031	1.001 ± .031	.980 ± .031	.979 ± .031	.970 ± .029	1.009 ± .031	.993 ± .033	.999 ± .032	1.001 ± .030	1.007 ± .031	1.096 ± .031	1.019 ± .031	.986 ± .031	1.008 ± .033	1.005 ± .031	**1.006 ± .030**	.995 ± .032	1.008 ± .030
	.80	.962 ± .031	1.004 ± .031	.968 ± .031	.971 ± .034	.946 ± .030	1.011 ± .031	.989 ± .034	.987 ± .033	.979 ± .032	**1.003 ± .030**	1.101 ± .034	1.008 ± .032	.967 ± .034	1.018 ± .034	1.017 ± .032	1.002 ± .031	.998 ± .032	1.009 ± .031
	.75	.949 ± .031	1.020 ± .032	.969 ± .033	.972 ± .035	.938 ± .032	**1.007 ± .031**	.981 ± .035	.989 ± .033	.972 ± .033	.999 ± .032	1.098 ± .035	1.005 ± .033	.970 ± .036	1.008 ± .034	1.022 ± .033	1.001 ± .032	1.011 ± .033	1.006 ± .033
	.70	.947 ± .033	1.026 ± .034	.962 ± .034	.945 ± .037	.929 ± .033	1.001 ± .034	.976 ± .037	.977 ± .033	.950 ± .034	**1.005 ± .032**	1.100 ± .037	.996 ± .033	.969 ± .038	1.024 ± .035	1.031 ± .032	.997 ± .034	1.021 ± .035	1.004 ± .034

Table C.26: Results for `food101`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.464 ± .004	.269 ± .003	.306 ± .004	.322 ± .003	.341 ± .004	.256 ± .003	.268 ± .003	.304 ± .004	.318 ± .003	.339 ± .004	.215 ± .003	**.211 ± .003**	.271 ± .003	.388 ± .003	.384 ± .004	.242 ± .003
	.95	.442 ± .004	.250 ± .003	.286 ± .004	.287 ± .004	.328 ± .003	.233 ± .003	.244 ± .003	.280 ± .004	.266 ± .004	.317 ± .003	.196 ± .003	**.190 ± .003**	.256 ± .003	.380 ± .004	.383 ± .004	.220 ± .003
	.90	.416 ± .004	.226 ± .003	.260 ± .004	.264 ± .003	.315 ± .004	.205 ± .003	.219 ± .003	.250 ± .004	.232 ± .003	.292 ± .003	.172 ± .003	**.165 ± .003**	.237 ± .003	.370 ± .004	.382 ± .004	.198 ± .003
	.85	.392 ± .004	.203 ± .003	.237 ± .003	.283 ± .003	.299 ± .004	.174 ± .003	.191 ± .003	.225 ± .004	.243 ± .003	.264 ± .003	.150 ± .003	**.134 ± .002**	.219 ± .003	.361 ± .003	.383 ± .004	.174 ± .003
	.80	.366 ± .004	.179 ± .003	.212 ± .003	.224 ± .004	.278 ± .004	.149 ± .003	.162 ± .003	.199 ± .004	.175 ± .003	.237 ± .003	.128 ± .003	**.108 ± .002**	.200 ± .003	.353 ± .003	.382 ± .004	.151 ± .003
	.75	.341 ± .004	.156 ± .003	.190 ± .003	.210 ± .003	.266 ± .003	.126 ± .003	.141 ± .003	.172 ± .004	.148 ± .003	.218 ± .003	.106 ± .002	**.086 ± .002**	.185 ± .003	.344 ± .004	.383 ± .004	.128 ± .003
	.70	.318 ± .004	.137 ± .003	.169 ± .003	.219 ± .004	.246 ± .003	.106 ± .003	.117 ± .003	.145 ± .003	.157 ± .003	.189 ± .003	.088 ± .002	**.069 ± .002**	.170 ± .003	.334 ± .004	.386 ± .004	.107 ± .002
$\hat{\phi}$	.99	.989 ± .001	.988 ± .001	.993 ± .001	.991 ± .001	**.990 ± .001**	**.990 ± .001**	.987 ± .001	.991 ± .001	**.990 ± .001**	.988 ± .001	.990 ± .001	.987 ± .001	.991 ± .001	.991 ± .001	.992 ± .001	.991 ± .001
	.95	.948 ± .001	.946 ± .001	.951 ± .002	.952 ± .002	**.951 ± .001**	.951 ± .002	.946 ± .002	.948 ± .002	**.951 ± .002**	.948 ± .002	**.951 ± .001**	.951 ± .002	.951 ± .002	.952 ± .001	.955 ± .001	.955 ± .001
	.90	**.899 ± .002**	.897 ± .002	.896 ± .002	.897 ± .002	.903 ± .002	**.899 ± .002**	**.900 ± .002**	.891 ± .002	.903 ± .002	.902 ± .002	.898 ± .002	.904 ± .002	**.901 ± .002**	.902 ± .002	.909 ± .002	.915 ± .002
	.85	.851 ± .002	.848 ± .003	.846 ± .003	.850 ± .003	.852 ± .003	.846 ± .003	.848 ± .003	.844 ± .003	**.850 ± .002**	.849 ± .003	.847 ± .002	.847 ± .003	.849 ± .003	.856 ± .003	.860 ± .003	.872 ± .003
	.80	.801 ± .003	.796 ± .003	.793 ± .003	.792 ± .003	.801 ± .003	.795 ± .003	.792 ± .003	.797 ± .003	.795 ± .003	**.799 ± .003**	.794 ± .003	.795 ± .003	.798 ± .003	.809 ± .003	.810 ± .003	.826 ± .003
	.75	**.752 ± .003**	.744 ± .004	.745 ± .003	**.750 ± .003**	**.751 ± .003**	.746 ± .003	.747 ± .003	.745 ± .003	.753 ± .003	**.752 ± .003**	.744 ± .004	.745 ± .004	**.750 ± .003**	.760 ± .003	.758 ± .003	.777 ± .003
	.70	**.700 ± .003**	.696 ± .004	.695 ± .004	.705 ± .003	**.700 ± .003**	**.700 ± .003**	.696 ± .003	.695 ± .004	.706 ± .004	**.701 ± .004**	**.698 ± .004**	.697 ± .004	**.700 ± .004**	.709 ± .003	.704 ± .003	.730 ± .003

Table C.27: Results for `giveme`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and $MinCoeff$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.061 ± .001	.060 ± .001	.060 ± .001	.060 ± .001	.060 ± .001	.061 ± .001	.061 ± .001	**.059 ± .001**	**.059 ± .001**	.062 ± .001	.061 ± .001	.060 ± .001	**.059 ± .001**	.060 ± .001	.065 ± .001	.060 ± .001	.064 ± .001	.065 ± .001
	.95	.047 ± .001	**.046 ± .001**	**.046 ± .001**	**.046 ± .002**	**.046 ± .002**	**.046 ± .001**	**.046 ± .001**	**.046 ± .001**	**.046 ± .001**	**.046 ± .001**	.050 ± .001	.047 ± .001	**.046 ± .001**	.047 ± .001	.065 ± .001	.047 ± .001	.064 ± .001	.066 ± .001
	.90	.036 ± .001	**.035 ± .001**	**.035 ± .001**	**.035 ± .001**	**.035 ± .001**	**.035 ± .001**	**.035 ± .001**	**.035 ± .001**	**.035 ± .001**	.037 ± .001	.046 ± .001	**.035 ± .001**	**.035 ± .001**	.036 ± .001	.065 ± .001	.036 ± .001	.065 ± .001	.068 ± .001
	.85	.031 ± .001	**.029 ± .001**	**.029 ± .001**	**.029 ± .001**	**.029 ± .001**	.030 ± .001	**.029 ± .001**	**.029 ± .001**	**.029 ± .001**	.040 ± .001	.045 ± .001	**.029 ± .001**	**.029 ± .001**	**.029 ± .001**	.059 ± .001	.030 ± .001	.065 ± .002	.069 ± .001
	.80	.028 ± .001	.026 ± .001	**.025 ± .001**	**.025 ± .001**	**.025 ± .001**	.026 ± .001	**.025 ± .001**	**.025 ± .001**	**.025 ± .001**	.034 ± .001	.043 ± .001	.026 ± .001	**.025 ± .001**	.026 ± .001	.057 ± .001	.026 ± .001	.065 ± .002	.070 ± .002
	.75	.025 ± .001	**.023 ± .001**	**.023 ± .001**	**.023 ± .001**	**.023 ± .001**	**.023 ± .001**	**.023 ± .001**	**.023 ± .001**	**.023 ± .001**	.028 ± .001	.041 ± .001	**.023 ± .001**	**.023 ± .001**	**.023 ± .001**	.053 ± .001	.024 ± .001	.065 ± .002	.071 ± .002
	.70	.023 ± .001	.022 ± .001	**.021 ± .001**	**.021 ± .001**	.022 ± .001	.022 ± .001	.022 ± .001	.022 ± .001	**.021 ± .001**	.026 ± .001	.038 ± .001	**.021 ± .001**	**.021 ± .001**	**.021 ± .001**	.051 ± .001	.023 ± .001	.063 ± .002	.072 ± .002
$\hat{\phi}$	.99	**.990 ± .001**	.989 ± .001	.989 ± .001	.989 ± .001	.989 ± .001	.991 ± .001	.991 ± .001	.988 ± .001	.988 ± .001	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	.989 ± .001	**.990 ± .001**	.991 ± .000	.991 ± .000	.994 ± .000	.989 ± .001
	.95	.948 ± .001	.947 ± .001	**.949 ± .001**	**.950 ± .001**	**.949 ± .001**	**.949 ± .001**	**.949 ± .001**	**.949 ± .001**	**.950 ± .001**	**.949 ± .001**	.948 ± .001	**.950 ± .001**	**.950 ± .001**	.949 ± .001	.967 ± .001	.952 ± .001	.964 ± .001	.947 ± .001
	.90	.896 ± .002	.897 ± .002	.896 ± .002	**.898 ± .002**	**.899 ± .002**	**.898 ± .002**	.898 ± .002	.897 ± .002	**.899 ± .002**	.894 ± .002	.896 ± .002	**.898 ± .002**	**.898 ± .002**	.897 ± .002	.967 ± .001	.904 ± .002	.927 ± .002	.895 ± .002
	.85	.848 ± .002	.847 ± .002	.847 ± .002	.846 ± .002	.846 ± .002	.849 ± .002	.847 ± .002	.847 ± .002	.847 ± .002	.843 ± .002	.845 ± .002	.847 ± .002	.847 ± .002	.847 ± .002	.854 ± .002	.854 ± .002	.891 ± .002	**.850 ± .002**
	.80	**.799 ± .002**	**.799 ± .002**	.794 ± .003	.797 ± .002	.796 ± .002	.796 ± .002	**.799 ± .002**	.795 ± .003	**.798 ± .002**	.794 ± .002	**.798 ± .002**	.797 ± .002	.797 ± .002	**.798 ± .002**	.807 ± .002	.803 ± .002	.846 ± .002	.803 ± .002
	.75	.748 ± .002	**.750 ± .003**	.747 ± .003	**.750 ± .003**	.748 ± .003	.747 ± .003	**.749 ± .003**	.748 ± .003	.747 ± .003	.748 ± .003	.748 ± .003	.748 ± .003	.748 ± .003	.749 ± .003	.759 ± .003	.767 ± .003	.806 ± .002	.754 ± .003
	.70	.696 ± .002	**.701 ± .003**	.697 ± .003	.697 ± .003	**.699 ± .003**	.698 ± .003	**.701 ± .003**	.698 ± .003	.698 ± .003	.696 ± .003	.696 ± .003	.697 ± .003	**.701 ± .003**	.696 ± .003	.707 ± .003	.730 ± .003	.756 ± .003	.706 ± .003
$MinCoeff$	.99	.936 ± .020	.936 ± .021	.925 ± .021	.901 ± .020	.905 ± .020	.942 ± .021	.939 ± .020	.918 ± .020	.908 ± .021	.926 ± .020	.945 ± .020	.933 ± .020	.932 ± .021	.933 ± .020	.994 ± .022	.941 ± .020	**1.000 ± .021**	**1.001 ± .021**
	.95	.718 ± .019	.684 ± .017	.691 ± .017	.691 ± .017	.685 ± .017	.705 ± .018	.722 ± .018	.697 ± .017	.689 ± .017	.685 ± .017	.781 ± .019	.706 ± .018	.690 ± .017	.715 ± .017	.990 ± .022	.702 ± .018	**1.003 ± .022**	1.018 ± .022
	.90	.545 ± .017	.520 ± .015	.519 ± .016	.524 ± .015	.523 ± .016	.526 ± .016	.528 ± .016	.523 ± .016	.525 ± .015	.548 ± .016	.716 ± .019	.530 ± .016	.524 ± .015	.536 ± .016	.990 ± .022	.537 ± .016	**1.009 ± .022**	1.044 ± .023
	.85	.463 ± .016	.437 ± .015	.433 ± .015	.440 ± .015	.437 ± .016	.443 ± .015	.440 ± .015	.437 ± .015	.437 ± .016	.600 ± .019	.694 ± .019	.439 ± .015	.436 ± .016	.437 ± .015	.890 ± .020	.449 ± .015	**1.012 ± .024**	1.061 ± .024
	.80	.419 ± .016	.384 ± .015	.380 ± .015	.381 ± .015	.378 ± .015	.387 ± .015	.382 ± .015	.381 ± .015	.379 ± .015	.511 ± .019	.672 ± .020	.387 ± .015	.380 ± .015	.390 ± .015	.859 ± .020	.396 ± .014	**1.014 ± .025**	1.074 ± .025
	.75	.375 ± .016	.347 ± .015	.345 ± .015	.347 ± .015	.348 ± .016	.344 ± .015	.344 ± .015	.346 ± .015	.342 ± .015	.414 ± .017	.636 ± .020	.349 ± .015	.343 ± .015	.345 ± .015	.804 ± .019	.365 ± .016	**1.017 ± .026**	1.094 ± .026
	.70	.344 ± .016	.323 ± .015	.319 ± .016	.319 ± .016	.330 ± .016	.325 ± .016	.324 ± .015	.322 ± .016	.316 ± .017	.400 ± .017	.592 ± .020	.320 ± .015	.318 ± .017	.317 ± .016	.768 ± .021	.339 ± .016	**995 ± .028**	1.110 ± .029

Table C.28: Results for `helena`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.618 ± .004	.614 ± .004	.619 ± .004	.613 ± .004	.613 ± .004	.615 ± .004	.612 ± .004	.618 ± .004	.611 ± .004	.614 ± .004	.612 ± .004	**.610 ± .004**	.615 ± .004	.633 ± .004	.632 ± .004	.611 ± .004
	.95	.608 ± .004	.605 ± .004	.609 ± .004	.600 ± .004	.601 ± .004	.603 ± .004	.599 ± .004	.605 ± .004	.599 ± .005	.599 ± .004	.605 ± .004	.598 ± .004	.607 ± .004	.623 ± .004	.631 ± .004	**.597 ± .004**
	.90	.594 ± .004	.592 ± .005	.597 ± .004	.583 ± .004	.585 ± .005	.585 ± .004	.585 ± .004	.590 ± .004	.582 ± .004	.584 ± .005	.594 ± .004	.582 ± .004	.595 ± .004	.609 ± .004	.631 ± .004	**.581 ± .004**
	.85	.581 ± .004	.581 ± .005	.584 ± .005	.570 ± .005	.570 ± .005	.569 ± .004	.566 ± .005	.572 ± .004	.564 ± .005	.566 ± .005	.582 ± .004	**.563 ± .004**	.586 ± .004	.597 ± .005	.628 ± .004	.563 ± .005
	.80	.567 ± .005	.567 ± .005	.570 ± .005	.552 ± .005	.557 ± .005	.547 ± .005	.548 ± .005	.553 ± .005	.547 ± .005	.552 ± .005	.568 ± .005	**.545 ± .005**	.577 ± .004	.583 ± .005	.628 ± .004	.547 ± .005
	.75	.555 ± .005	.551 ± .005	.557 ± .005	.540 ± .005	.541 ± .005	.529 ± .005	.528 ± .005	.536 ± .005	.529 ± .005	.534 ± .005	.555 ± .005	**.526 ± .005**	.566 ± .005	.570 ± .005	.624 ± .004	.529 ± .005
	.70	.541 ± .005	.535 ± .006	.540 ± .006	.530 ± .005	.525 ± .005	.510 ± .005	.510 ± .005	.517 ± .005	.515 ± .005	.520 ± .005	.540 ± .005	**.509 ± .005**	.556 ± .005	.555 ± .005	.621 ± .004	**.509 ± .005**
$\hat{\phi}$	.99	**.989 ± .001**	**.991 ± .001**	**.990 ± .001**	**.989 ± .001**	.989 ± .001	.987 ± .001	**.989 ± .001**	**.990 ± .001**	.984 ± .001	**.990 ± .001**	.988 ± .001	.988 ± .001	**.991 ± .001**	.988 ± .001	.987 ± .001	.988 ± .001
	.95	.947 ± .002	.953 ± .002	.944 ± .002	.945 ± .002	.946 ± .002	.946 ± .002	**.949 ± .002**	.948 ± .002	.948 ± .002	**.948 ± .002**	.948 ± .002	.953 ± .002	**.950 ± .002**	**.949 ± .002**	.939 ± .002	.944 ± .002
	.90	.898 ± .003	**.900 ± .003**	.892 ± .003	.890 ± .003	.892 ± .002	.895 ± .003	**.900 ± .003**	**.901 ± .003**	**.899 ± .003**	.898 ± .003	.898 ± .002	**.900 ± .003**	.894 ± .002	.889 ± .003	.894 ± .003	.897 ± .003
	.85	.846 ± .003	**.852 ± .003**	.845 ± .004	.838 ± .003	.840 ± .003	**.849 ± .003**	.846 ± .003	**.848 ± .003**	.842 ± .003	.846 ± .003	.846 ± .003	.843 ± .003	.844 ± .003	.838 ± .003	.842 ± .003	.846 ± .003
	.80	.792 ± .003	**.801 ± .003**	.796 ± .004	.788 ± .003	.790 ± .004	.789 ± .003	.792 ± .003	.797 ± .003	**.798 ± .003**	.793 ± .003	**.799 ± .003**	.794 ± .004	**.798 ± .003**	.788 ± .004	.788 ± .003	.795 ± .004
	.75	.742 ± .004	**.752 ± .004**	.751 ± .005	.745 ± .004	.744 ± .004	.739 ± .004	.743 ± .004	.748 ± .004	.747 ± .004	.747 ± .003	.752 ± .004	.743 ± .004	.745 ± .004	.743 ± .004	.740 ± .003	.746 ± .004
	.70	.695 ± .004	**.701 ± .004**	.699 ± .004	**.700 ± .004**	.688 ± .004	.693 ± .004	.697 ± .004	.697 ± .004	.697 ± .004	.698 ± .004	.702 ± .004	.699 ± .004	.697 ± .004	.690 ± .004	.691 ± .003	**.699 ± .004**

Table C.29: Results for `heloc`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and $MinCoeff$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.309 ± .010	.279 ± .009	.277 ± .010	.279 ± .011	.277 ± .010	.285 ± .009	.277 ± .009	.280 ± .010	.277 ± .011	.277 ± .010	.275 ± .010	**.273 ± .010**	.283 ± .010	.283 ± .010	.289 ± .010	.274 ± .010	.276 ± .010	.285 ± .009
	.95	.307 ± .011	.272 ± .010	.273 ± .010	.281 ± .011	.279 ± .010	.281 ± .010	.276 ± .009	.280 ± .010	.275 ± .011	.271 ± .010	.273 ± .010	.263 ± .010	.283 ± .010	.283 ± .010	.292 ± .009	.268 ± .010	**.260 ± .010**	.277 ± .010
	.90	.301 ± .011	.259 ± .010	.263 ± .010	.265 ± .010	.260 ± .011	.261 ± .010	.259 ± .010	.262 ± .010	.261 ± .011	.260 ± .010	.273 ± .010	.258 ± .011	.275 ± .010	.275 ± .010	.290 ± .010	.259 ± .010	**.245 ± .010**	.270 ± .010
	.85	.296 ± .011	.253 ± .010	.248 ± .011	.253 ± .011	.247 ± .011	.248 ± .011	.245 ± .010	.249 ± .011	.253 ± .011	.248 ± .010	.266 ± .011	.242 ± .010	.281 ± .010	.274 ± .011	.291 ± .010	.252 ± .010	**.237 ± .010**	.242 ± .011
	.80	.290 ± .011	.253 ± .010	.243 ± .011	.242 ± .011	.256 ± .011	.232 ± .010	.231 ± .010	.233 ± .011	.240 ± .010	.238 ± .011	.258 ± .011	.225 ± .011	.288 ± .010	.276 ± .011	.287 ± .010	.236 ± .011	**.224 ± .011**	.239 ± .012
	.75	.283 ± .012	.218 ± .011	.226 ± .011	.228 ± .011	.321 ± .013	.216 ± .011	.218 ± .011	.223 ± .011	.225 ± .011	.226 ± .011	.255 ± .011	.213 ± .011	.291 ± .011	.278 ± .012	.285 ± .010	.228 ± .011	**.207 ± .011**	.238 ± .012
	.70	.283 ± .012	.202 ± .011	.213 ± .011	.209 ± .011	.210 ± .011	.202 ± .011	.201 ± .011	.208 ± .011	.207 ± .011	.212 ± .011	.242 ± .012	.202 ± .011	.294 ± .011	.285 ± .012	.284 ± .011	.210 ± .011	**.193 ± .011**	.231 ± .012
$\hat{\phi}$	.99	.994 ± .001	.994 ± .002	.987 ± .003	.993 ± .002	.993 ± .002	.997 ± .001	.986 ± .003	.993 ± .002	**.990 ± .002**	.989 ± .003	.996 ± .001	.985 ± .003	**.990 ± .002**	.999 ± .001	**.990 ± .002**	**.990 ± .002**	.986 ± .003	.989 ± .002
	.95	.974 ± .003	**.947 ± .005**	.968 ± .004	1.000 ± .000	.999 ± .001	.986 ± .002	.984 ± .003	.990 ± .002	.974 ± .004	.963 ± .004	.946 ± .005	.947 ± .005	.988 ± .002	.999 ± .001	**.951 ± .005**	.974 ± .003	.907 ± .006	**.950 ± .004**
	.90	.951 ± .005	.893 ± .006	**.905 ± .006**	.916 ± .006	.908 ± .006	.916 ± .006	.913 ± .006	.913 ± .006	.907 ± .006	.913 ± .006	.904 ± .006	.933 ± .006	.911 ± .006	.905 ± .007	.901 ± .008	.927 ± .007	.861 ± .007	.908 ± .006
	.85	.911 ± .006	.864 ± .008	.846 ± .009	.867 ± .008	.847 ± .009	.860 ± .007	.866 ± .007	.858 ± .008	.866 ± .008	.853 ± .008	**.852 ± .007**	.860 ± .007	.878 ± .008	.846 ± .008	.862 ± .009	.894 ± .008	.830 ± .007	.814 ± .008
	.80	.854 ± .007	.864 ± .008	.801 ± .008	.823 ± .009	.818 ± .008	.821 ± .007	.820 ± .008	.798 ± .009	.816 ± .009	.811 ± .009	**.800 ± .008**	.806 ± .009	.821 ± .009	.795 ± .009	.806 ± .009	.807 ± .009	.781 ± .009	.783 ± .009
	.75	.777 ± .009	.762 ± .010	.754 ± .009	.761 ± .009	.747 ± .011	.764 ± .009	.768 ± .009	.758 ± .009	.759 ± .009	.763 ± .010	**.750 ± .009**	.759 ± .010	.769 ± .009	.735 ± .010	.761 ± .011	.754 ± .009	.731 ± .010	.766 ± .009
	.70	.777 ± .009	.711 ± .011	.714 ± .010	.716 ± .010	.715 ± .010	.716 ± .010	.720 ± .010	.704 ± .010	.711 ± .011	.704 ± .010	.688 ± .009	**.702 ± .010**	.713 ± .009	.692 ± .010	.724 ± .011	.690 ± .010	.690 ± .010	.722 ± .010
$MinCoeff$	.99	.997 ± .023	1.001 ± .023	1.003 ± .024	1.001 ± .023	.999 ± .023	.999 ± .023	1.002 ± .023	1.001 ± .023	1.004 ± .023	1.002 ± .023	**1.001 ± .023**	1.002 ± .024	1.009 ± .023	1.002 ± .023	.994 ± .024	1.001 ± .023	1.008 ± .023	1.003 ± .023
	.95	.991 ± .024	1.005 ± .023	1.004 ± .024	1.000 ± .023	1.000 ± .023	**.999 ± .023**	1.011 ± .023	1.001 ± .023	1.007 ± .023	1.001 ± .024	.979 ± .024	1.011 ± .023	1.010 ± .023	1.002 ± .023	.989 ± .024	1.000 ± .024	1.014 ± .024	1.005 ± .024
	.90	.990 ± .024	1.016 ± .024	1.017 ± .025	1.006 ± .025	1.000 ± .025	1.007 ± .023	1.005 ± .024	1.009 ± .024	1.016 ± .024	1.008 ± .025	.958 ± .025	**1.006 ± .023**	1.033 ± .025	**1.000 ± .025**	.986 ± .024	.995 ± .025	1.011 ± .024	1.011 ± .025
	.85	.980 ± .026	1.009 ± .024	1.020 ± .024	1.004 ± .026	**1.010 ± .024**	1.013 ± .024	1.013 ± .024	1.010 ± .025	1.015 ± .025	1.012 ± .025	.952 ± .026	1.014 ± .024	1.063 ± .026	.977 ± .025	.983 ± .025	.988 ± .025	1.016 ± .025	1.026 ± .026
	.80	.975 ± .026	**1.009 ± .024**	1.019 ± .025	1.026 ± .025	.965 ± .026	1.022 ± .024	1.012 ± .025	1.018 ± .025	1.019 ± .026	1.015 ± .025	.934 ± .026	1.013 ± .025	1.114 ± .026	.963 ± .024	.983 ± .027	.984 ± .027	1.024 ± .025	1.041 ± .026
	.75	.977 ± .027	**1.001 ± .026**	1.025 ± .025	1.018 ± .026	.767 ± .023	1.025 ± .026	1.014 ± .026	1.024 ± .025	1.023 ± .027	1.020 ± .025	.922 ± .028	1.012 ± .026	1.157 ± .026	.958 ± .025	.995 ± .029	.975 ± .027	1.030 ± .026	1.053 ± .027
	.70	.977 ± .027	**1.006 ± .027**	1.026 ± .026	1.011 ± .028	1.028 ± .027	1.031 ± .027	1.012 ± .027	1.022 ± .026	1.024 ± .027	1.015 ± .026	.894 ± .028	1.005 ± .028	1.207 ± .027	.941 ± .027	1.002 ± .029	.950 ± .028	1.043 ± .027	1.079 ± .027

Table C.30: Results for `higgs`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and $MinCoeff$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.385 ± .005	.274 ± .003	.286 ± .004	.271 ± .003	.279 ± .003	.273 ± .003	.273 ± .003	.286 ± .004	.271 ± .003	.279 ± .003	.269 ± .003	**.268 ± .003**	.280 ± .003	.287 ± .003	.296 ± .003	.278 ± .004	.279 ± .003	.275 ± .004
	.95	.388 ± .010	.269 ± .004	.279 ± .004	.264 ± .003	.269 ± .003	.264 ± .004	.265 ± .003	.277 ± .004	.263 ± .003	.270 ± .003	.264 ± .003	**.260 ± .003**	.273 ± .003	.282 ± .003	.296 ± .003	.268 ± .004	.273 ± .003	.268 ± .004
	.90	.388 ± .016	.261 ± .003	.268 ± .004	.255 ± .003	.263 ± .003	.253 ± .004	.255 ± .003	.268 ± .004	.249 ± .003	.262 ± .003	.258 ± .003	**.248 ± .004**	.266 ± .003	.275 ± .003	.296 ± .003	.255 ± .004	.264 ± .004	.258 ± .004
	.85	.387 ± .021	.253 ± .004	.256 ± .004	.247 ± .003	.251 ± .003	.242 ± .004	.243 ± .004	.258 ± .004	.245 ± .003	.249 ± .003	.252 ± .003	**.236 ± .004**	.257 ± .003	.269 ± .003	.296 ± .003	.242 ± .004	.256 ± .004	.246 ± .004
	.80	.382 ± .022	.247 ± .004	.246 ± .004	.230 ± .003	.306 ± .004	.234 ± .004	.231 ± .004	.245 ± .004	.229 ± .003	.237 ± .003	.246 ± .004	**.225 ± .004**	.248 ± .004	.263 ± .003	.294 ± .004	.231 ± .004	.244 ± .004	.237 ± .004
	.75	.377 ± .023	.239 ± .004	.233 ± .004	.217 ± .004	.301 ± .004	.219 ± .004	.220 ± .004	.233 ± .004	.215 ± .003	.228 ± .004	.240 ± .004	**.213 ± .004**	.240 ± .004	.259 ± .003	.295 ± .004	.221 ± .004	.231 ± .004	.225 ± .004
	.70	.370 ± .025	.230 ± .004	.219 ± .004	.207 ± .004	.306 ± .004	.207 ± .004	.209 ± .004	.221 ± .004	.209 ± .004	.217 ± .004	.234 ± .004	**.201 ± .004**	.231 ± .004	.253 ± .003	.294 ± .004	.208 ± .004	.219 ± .004	.213 ± .004
$\hat{\phi}$	.99	.990 ± .001	.991 ± .001	.992 ± .001	.989 ± .001	**.990 ± .001**	.988 ± .001	**.990 ± .001**	.991 ± .001	.989 ± .001	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	.990 ± .001	**.990 ± .001**	.991 ± .001	.989 ± .001	**.990 ± .001**	.991 ± .001
	.95	.949 ± .002	.948 ± .002	.953 ± .001	.951 ± .002	.951 ± .001	.947 ± .002	.952 ± .002	**.951 ± .001**	.953 ± .002	.948 ± .002	.945 ± .002	.952 ± .001	**.950 ± .001**	**.949 ± .002**	.948 ± .002	.948 ± .002	.951 ± .002	.951 ± .002
	.90	.897 ± .004	.899 ± .002	.903 ± .002	.903 ± .002	**.900 ± .002**	.903 ± .002	.902 ± .002	**.901 ± .002**	.898 ± .002	**.900 ± .002**	.896 ± .002	.898 ± .002	.902 ± .002	.897 ± .002	.899 ± .002	.893 ± .002	.896 ± .002	.902 ± .002
	.85	.847 ± .005	.843 ± .002	.849 ± .003	.855 ± .002	**.851 ± .002**	.852 ± .003	.852 ± .003	.853 ± .002	.853 ± .002	.848 ± .002	.848 ± .003	**.850 ± .002**	.849 ± .003	.844 ± .003	.847 ± .002	.842 ± .003	.849 ± .003	.849 ± .002
	.80	.796 ± .004	.792 ± .003	.803 ± .003	.798 ± .002	.798 ± .003	.804 ± .003	**.800 ± .003**	**.800 ± .003**	**.800 ± .003**	.803 ± .003	.799 ± .003	.802 ± .003	.796 ± .003	.792 ± .003	.797 ± .003	.794 ± .003	.796 ± .003	**.800 ± .003**
	.75	.749 ± .005	.743 ± .003	.752 ± .003	.750 ± .003	.741 ± .003	.755 ± .003	.748 ± .003	**.751 ± .003**	.747 ± .003	.754 ± .003	.753 ± .003	.753 ± .003	**.749 ± .003**	.750 ± .003	**.749 ± .003**	.742 ± .003	.745 ± .003	.752 ± .003
	.70	.696 ± .007	.689 ± .003	.698 ± .003	.703 ± .003	.698 ± .003	.702 ± .004	**.701 ± .003**	.699 ± .003	.703 ± .003	.705 ± .003	.705 ± .003	.707 ± .003	**.699 ± .003**	.702 ± .003	.696 ± .003	.692 ± .003	.697 ± .003	.703 ± .003
$MinCoeff$	.99	1.005 ± .008	**1.000 ± .007**	**1.002 ± .006**	**1.001 ± .006**	**1.000 ± .006**	**1.001 ± .006**	**1.000 ± .006**	**1.001 ± .006**	**1.001 ± .006**	**1.002 ± .006**	**1.000 ± .006**	**1.001 ± .006**	**1.002 ± .006**	**1.000 ± .007**	**1.000 ± .006**	**1.002 ± .006**	1.003 ± .006	1.002 ± .007
	.95	1.022 ± .018	1.002 ± .007	1.006 ± .006	1.006 ± .006	1.002 ± .007	1.003 ± .006	**1.001 ± .006**	1.009 ± .006	1.003 ± .006	1.003 ± .007	1.004 ± .007	1.004 ± .007	1.006 ± .007	1.002 ± .007	**1.000 ± .007**	1.007 ± .007	1.011 ± .007	1.010 ± .007
	.90	1.038 ± .031	1.006 ± .007	1.015 ± .007	1.009 ± .006	**998 ± .007**	1.007 ± .007	1.007 ± .007	1.019 ± .007	1.007 ± .007	1.009 ± .007	1.007 ± .007	1.009 ± .007	1.010 ± .007	1.005 ± .007	.996 ± .007	1.013 ± .007	1.024 ± .007	1.018 ± .007
	.85	1.052 ± .041	1.010 ± .007	1.029 ± .007	1.013 ± .007	1.026 ± .007	1.012 ± .007	1.011 ± .007	1.031 ± .007	1.011 ± .007	1.012 ± .007	1.008 ± .007	1.012 ± .007	1.018 ± .007	1.010 ± .007	**996 ± .007**	1.022 ± .007	1.038 ± .007	1.030 ± .007
	.80	1.061 ± .045	1.014 ± .007	1.039 ± .007	1.012 ± .007	1.016 ± .007	1.016 ± .007	1.016 ± .007	1.043 ± .007	1.008 ± .008	1.015 ± .007	1.010 ± .007	1.019 ± .008	1.024 ± .008	1.016 ± .008	**997 ± .007**	1.032 ± .007	1.055 ± .008	1.042 ± .008
	.75	1.069 ± .051	1.020 ± .008	1.053 ± .008	1.018 ± .007	1.200 ± .007	1.025 ± .007	1.020 ± .007	1.060 ± .007	1.024 ± .007	1.027 ± .008	1.011 ± .008	1.024 ± .008	1.033 ± .008	1.017 ± .008	**997 ± .008**	1.041 ± .008	1.075 ± .008	1.058 ± .008
	.70	1.079 ± .058	1.026 ± .008	1.070 ± .008	1.029 ± .008	1.247 ± .007	1.031 ± .008	1.024 ± .007	1.075 ± .008	1.022 ± .008	1.032 ± .008	1.013 ± .008	1.028 ± .008	1.045 ± .008	1.023 ± .008	**997 ± .008**	1.055 ± .008	1.093 ± .009	1.072 ± .008

Table C.31: Results for `house`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and $MinCoeff$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.217 ± .007	.126 ± .007	.130 ± .007	.131 ± .006	.133 ± .006	.128 ± .006	.124 ± .007	.129 ± .007	.128 ± .006	.132 ± .006	.124 ± .006	.123 ± .006	.128 ± .007	.140 ± .006	.143 ± .006	**.120 ± .006**	.124 ± .006	.127 ± .006
	.95	.208 ± .007	.114 ± .006	.120 ± .006	.122 ± .006	.123 ± .006	.117 ± .006	.113 ± .006	.116 ± .006	.114 ± .006	.117 ± .006	.116 ± .006	.106 ± .006	.116 ± .006	.131 ± .006	.146 ± .006	**.105 ± .006**	.117 ± .006	.120 ± .006
	.90	.190 ± .007	.098 ± .006	.106 ± .006	.111 ± .006	.112 ± .006	.101 ± .006	.099 ± .006	.103 ± .005	.097 ± .006	.100 ± .006	.109 ± .006	.092 ± .006	.099 ± .006	.123 ± .006	.147 ± .007	**.091 ± .006**	.110 ± .006	.112 ± .007
	.85	.175 ± .007	.081 ± .005	.089 ± .005	.093 ± .005	.093 ± .005	.084 ± .006	**.078 ± .005**	.085 ± .005	.090 ± .006	.088 ± .005	.099 ± .006	**.078 ± .005**	.096 ± .006	.113 ± .006	.147 ± .007	.079 ± .006	.097 ± .006	.101 ± .006
	.80	.158 ± .008	.068 ± .005	.078 ± .006	.077 ± .006	.080 ± .006	**.064 ± .005**	.066 ± .005	.077 ± .006	.073 ± .006	.077 ± .006	.093 ± .006	**.064 ± .005**	.068 ± .005	.102 ± .006	.149 ± .008	.065 ± .006	.088 ± .005	.088 ± .006
	.75	.144 ± .008	.057 ± .005	.069 ± .006	.062 ± .006	.066 ± .006	.057 ± .005	.057 ± .005	.071 ± .006	.062 ± .006	.064 ± .005	.086 ± .006	**.055 ± .005**	.060 ± .005	.095 ± .006	.148 ± .008	.056 ± .006	.075 ± .006	.078 ± .006
	.70	.141 ± .008	.050 ± .005	.060 ± .006	.055 ± .006	.052 ± .005	**.049 ± .005**	.050 ± .005	.061 ± .006	.052 ± .005	.054 ± .005	.072 ± .006	**.049 ± .005**	.052 ± .005	.085 ± .006	.149 ± .008	.053 ± .005	.063 ± .006	.067 ± .006
$\hat{\phi}$	.99	.984 ± .002	.988 ± .002	.994 ± .001	.990 ± .002	.992 ± .002	.989 ± .003

Table C.32: Results for `indian`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.050 ± .005	.038 ± .004	.041 ± .005	.046 ± .005	.131 ± .008	**.036 ± .004**	.038 ± .004	.039 ± .005	.041 ± .005	.146 ± .008	.040 ± .005	.040 ± .005	.055 ± .005	.065 ± .006	.084 ± .006	.037 ± .005
	.95	.038 ± .005	.029 ± .004	.023 ± .004	.031 ± .005	.107 ± .008	.023 ± .004	.027 ± .004	.023 ± .004	.029 ± .004	.123 ± .008	.024 ± .004	**.017 ± .003**	.042 ± .005	.062 ± .006	.079 ± .006	.027 ± .004
	.90	.032 ± .005	.029 ± .004	.012 ± .003	.022 ± .004	.099 ± .008	.011 ± .003	.017 ± .003	.012 ± .003	.016 ± .003	.115 ± .008	.012 ± .003	**.010 ± .003**	.034 ± .005	.059 ± .006	.079 ± .006	.019 ± .004
	.85	.028 ± .005	.012 ± .003	.007 ± .002	.014 ± .003	.113 ± .008	.007 ± .002	.012 ± .003	.008 ± .002	.013 ± .003	.093 ± .007	**.006 ± .002**	**.006 ± .002**	.029 ± .005	.056 ± .006	.074 ± .006	.013 ± .003
	.80	.024 ± .004	.012 ± .003	.005 ± .002	.012 ± .003	.094 ± .007	.007 ± .002	.012 ± .003	.005 ± .002	.031 ± .004	.139 ± .009	**.003 ± .001**	.005 ± .002	.022 ± .004	.051 ± .006	.072 ± .007	.007 ± .002
	.75	.020 ± .004	.011 ± .003	.004 ± .001	.007 ± .002	.087 ± .008	**.003 ± .001**	.010 ± .003	.004 ± .001	.006 ± .002	.252 ± .011	**.003 ± .001**	**.003 ± .001**	.018 ± .004	.044 ± .006	.066 ± .006	.006 ± .002
	.70	.016 ± .004	.011 ± .003	**.002 ± .001**	.005 ± .002	.068 ± .007	.003 ± .002	.010 ± .003	.003 ± .001	.005 ± .002	.237 ± .011	.003 ± .001	**.002 ± .001**	.016 ± .004	.043 ± .006	.061 ± .006	.004 ± .002
$\hat{\phi}$	.99	.992 ± .002	**.990 ± .002**	.992 ± .002	.994 ± .002	.980 ± .003	.982 ± .003	.988 ± .003	**.991 ± .002**	.986 ± .003	.992 ± .002	.985 ± .003	.992 ± .002	.979 ± .003	.994 ± .002	**.991 ± .002**	.992 ± .002
	.95	.939 ± .006	.962 ± .004	.944 ± .005	.953 ± .005	**.952 ± .004**	.941 ± .005	.950 ± .006	.941 ± .005	.952 ± .006	.944 ± .006	.935 ± .006	.931 ± .006	.933 ± .006	**.950 ± .005**	**.949 ± .005**	.957 ± .005
	.90	.877 ± .008	.907 ± .007	.884 ± .008	.896 ± .007	**.899 ± .006**	.885 ± .008	.900 ± .007	.891 ± .008	.890 ± .008	.888 ± .008	.889 ± .008	.882 ± .008	.892 ± .007	.908 ± .007	**.901 ± .007**	.919 ± .006
	.85	.828 ± .010	.842 ± .008	.848 ± .008	.832 ± .008	.858 ± .008	.831 ± .010	.836 ± .009	.838 ± .009	.838 ± .009	**.849 ± .008**	.826 ± .010	.833 ± .010	.843 ± .009	.852 ± .009	.825 ± .009	.869 ± .008
	.80	.782 ± .010	.796 ± .010	.792 ± .010	.805 ± .009	**.802 ± .009**	.784 ± .011	.784 ± .010	.785 ± .010	.809 ± .010	.791 ± .010	.777 ± .010	.787 ± .010	.785 ± .010	.789 ± .010	.759 ± .011	.823 ± .010
	.75	.735 ± .011	.739 ± .011	.741 ± .010	.741 ± .010	**.751 ± .010**	.730 ± .011	.744 ± .011	.743 ± .011	.739 ± .010	.747 ± .012	.728 ± .010	.735 ± .011	.719 ± .011	.736 ± .011	.705 ± .012	.795 ± .011
	.70	.674 ± .012	.705 ± .011	.683 ± .012	.692 ± .011	.683 ± .011	.689 ± .012	.688 ± .011	.687 ± .012	**.700 ± .011**	.693 ± .012	.688 ± .011	.671 ± .011	.673 ± .011	.689 ± .011	.669 ± .013	.759 ± .012

Table C.33: Results for `jannis`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and $MinCoeff$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.291 ± .004	.207 ± .004	.204 ± .004	.205 ± .004	.203 ± .003	.208 ± .004	.207 ± .004	.204 ± .004	.205 ± .004	.202 ± .004	.203 ± .004	**.200 ± .003**	.210 ± .004	.221 ± .004	.225 ± .003	.202 ± .003	.203 ± .003	.210 ± .004
	.95	.284 ± .004	.198 ± .004	.196 ± .004	.191 ± .004	.195 ± .003	.197 ± .003	.195 ± .004	.191 ± .004	**.188 ± .004**	.194 ± .003	.196 ± .004	.189 ± .004	.200 ± .004	.218 ± .004	.223 ± .004	.189 ± .003	.192 ± .003	.199 ± .004
	.90	.277 ± .004	.186 ± .004	.181 ± .004	.180 ± .003	.185 ± .004	.184 ± .004	.180 ± .004	.178 ± .004	.178 ± .003	.182 ± .004	.191 ± .004	.177 ± .004	.189 ± .004	.214 ± .004	.222 ± .004	**.176 ± .003**	**.176 ± .003**	.186 ± .003
	.85	.267 ± .004	.172 ± .004	.169 ± .004	.166 ± .004	.168 ± .004	.167 ± .004	.167 ± .004	.165 ± .004	.166 ± .004	.172 ± .004	.181 ± .004	.165 ± .004	.177 ± .003	.210 ± .004	.221 ± .004	**.163 ± .003**	.164 ± .003	.168 ± .004
	.80	.261 ± .004	.159 ± .004	.157 ± .004	.151 ± .003	.159 ± .004	.153 ± .004	.155 ± .004	.152 ± .004	**.150 ± .004**	.159 ± .003	.174 ± .004	**.150 ± .004**	.166 ± .003	.210 ± .004	.221 ± .004	.151 ± .003	.153 ± .003	.155 ± .004
	.75	.253 ± .004	.145 ± .004	.141 ± .004	.142 ± .003	.151 ± .004	.142 ± .004	.141 ± .004	.139 ± .004	.139 ± .003	.150 ± .004	.164 ± .004	**.136 ± .004**	.154 ± .004	.207 ± .004	.220 ± .004	.138 ± .004	.142 ± .004	.144 ± .004
	.70	.244 ± .004	.134 ± .004	.128 ± .003	.128 ± .004	.144 ± .004	.130 ± .004	.133 ± .004	.127 ± .004	.127 ± .004	.143 ± .004	.156 ± .004	**.124 ± .004**	.143 ± .004	.203 ± .004	.218 ± .004	.128 ± .004	.134 ± .004	.136 ± .004
$\hat{\phi}$	.99	.986 ± .001	.993 ± .001	**.991 ± .001**	**.989 ± .001**	.988 ± .001	.988 ± .001	.988 ± .001	**.990 ± .001**	.993 ± .001	.988 ± .001	.988 ± .001	**.991 ± .001**	.986 ± .001	.992 ± .001	**.989 ± .001**	**.989 ± .001**	**.990 ± .001**	**.990 ± .001**	.992 ± .001
	.95	.943 ± .002	.953 ± .002	.954 ± .002	.948 ± .002	.946 ± .002	.952 ± .002	.952 ± .002	.952 ± .002	.948 ± .002	.948 ± .002	.947 ± .002	.947 ± .002	**.950 ± .002**	.947 ± .002	.945 ± .002	.948 ± .002	.950 ± .002	.955 ± .002
	.90	.893 ± .003	.908 ± .003	.903 ± .003	.903 ± .003	.907 ± .003	.907 ± .003	**.900 ± .003**	.908 ± .003	.905 ± .003	.903 ± .003	.896 ± .003	**.901 ± .003**	**.900 ± .003**	.899 ± .003	.893 ± .003	**.899 ± .003**	**.900 ± .003**	.907 ± .003
	.85	.845 ± .003	.857 ± .003	.855 ± .004	.857 ± .003	.850 ± .003	.851 ± .004	**.851 ± .003**	.860 ± .004	.853 ± .003	.853 ± .004	.839 ± .003	.856 ± .003	.851 ± .003	.848 ± .004	.846 ± .003	.851 ± .004	.854 ± .003	.854 ± .003
	.80	.792 ± .004	.802 ± .004	.807 ± .004	.807 ± .003	.808 ± .004	.800 ± .004	.803 ± .004	.805 ± .004	.805 ± .003	.803 ± .004	.795 ± .004	.803 ± .004	.807 ± .003	**.800 ± .004**	.797 ± .004	**.799 ± .004**	.803 ± .004	.797 ± .004
	.75	.744 ± .005	.752 ± .005	.754 ± .004	.755 ± .004	.747 ± .004	.753 ± .004	.751 ± .004	**.751 ± .004**	**.751 ± .004**	.748 ± .005	.738 ± .004	.748 ± .004	.755 ± .004	.751 ± .004	.748 ± .004	.748 ± .004	.750 ± .004	.748 ± .004
	.70	.691 ± .005	.701 ± .005	.701 ± .004	.702 ± .004	.702 ± .005	.702 ± .004	.703 ± .005	**.700 ± .004**	.701 ± .004	.701 ± .005	**.700 ± .004**	.698 ± .004	.703 ± .004	.703 ± .004	.696 ± .004	.695 ± .004	.701 ± .004	.703 ± .004
$MinCoeff$	.99	**.997 ± .010**	**.999 ± .010**	**.997 ± .010**	**1.000 ± .010**	**1.001 ± .010**	**1.001 ± .010**	**.999 ± .010**	**.998 ± .010**	**.999 ± .010**	**.999 ± .010**	**.999 ± .010**	**.998 ± .010**	**.997 ± .010**	**.999 ± .010**	**1.002 ± .010**	**1.001 ± .010**	**1.001 ± .010**	**1.000 ± .010**
	.95	.991 ± .010	.996 ± .010	.996 ± .010	**1.001 ± .011**	**1.002 ± .010**	1.004 ± .010	.996 ± .010	.996 ± .010	**.999 ± .010**	**1.002 ± .010**	**1.002 ± .010**	**1.000 ± .010**	**.998 ± .010**	.994 ± .010	1.005 ± .010	**1.001 ± .010**	**1.003 ± .010**	1.005 ± .010
	.90	.981 ± .010	.996 ± .011	.995 ± .011	**1.002 ± .011**	.991 ± .011	1.007 ± .011	.995 ± .011	.992 ± .010	1.002 ± .011	.998 ± .011	1.006 ± .010	1.003 ± .010	.994 ± .010	.986 ± .010	1.018 ± .011	**.999 ± .010**	1.011 ± .010	1.009 ± .011
	.85	.973 ± .011	.991 ± .011	.991 ± .011	**.997 ± .011**	1.009 ± .010	1.010 ± .011	.993 ± .011	.988 ± .011	**1.002 ± .011**	1.024 ± .010	1.007 ± .011	1.005 ± .011	.994 ± .010	.975 ± .011	1.028 ± .011	.994 ± .011	1.016 ± .011	1.016 ± .011
	.80	.955 ± .011	.987 ± .012	.985 ± .011	.979 ± .011	**.998 ± .010**	1.010 ± .011	.987 ± .011	.980 ± .011	.986 ± .011	.992 ± .010	1.008 ± .011	1.002 ± .011	.987 ± .011	.957 ± .011	1.040 ± .011	.984 ± .011	1.025 ± .011	1.029 ± .011
	.75	.937 ± .011	.984 ± .011	.972 ± .011	**.999 ± .011**	.972 ± .011	1.012 ± .011	.982 ± .011	.968 ± .011	1.002 ± .011	.987 ± .011	1.007 ± .012	.997 ± .011	.984 ± .011	.948 ± .011	1.057 ± .011	.973 ± .011	1.034 ± .011	1.040 ± .011
	.70	.914 ± .011	.977 ± .012	.959 ± .011	.979 ± .011	1.011 ± .012	**1.005 ± .011**	.974 ± .012	.953 ± .011	.988 ± .011	1.010 ± .012	1.007 ± .012	.987 ± .011	.975 ± .012	.937 ± .011	1.073 ± .012	.963 ± .011	1.044 ± .011	1.054 ± .012

Table C.34: Results for `kddipums97`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and $MinCoeff$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.123 ± .010	.154 ± .011	.130 ± .011	.136 ± .011	.148 ± .011	.137 ± .010	.150 ± .011	.130 ± .011	.130 ± .010	.145 ± .011	.137 ± .010	.138 ± .010	**.119 ± .010**	.151 ± .011	.143 ± .011	.137 ± .011	.145 ± .011	.136 ± .010
	.95	.121 ± .010	.138 ± .011	.120 ± .011	.121 ± .010	.137 ± .010	.119 ± .010	.129 ± .010	.111 ± .010	.115 ± .010	.136 ± .011	.129 ± .010	.130 ± .010	**.109 ± .010**	.146 ± .011	.147 ± .012	.115 ± .011	.118 ± .011	.124 ± .010
	.90	.110 ± .009	.113 ± .011	.101 ± .010	.093 ± .010	.114 ± .011	.104 ± .009	.111 ± .010	.096 ± .010	.091 ± .010	.117 ± .011	.120 ± .010	.117 ± .010	.103 ± .010	.136 ± .010	.147 ± .012	.090 ± .010	**.089 ± .010**	.106 ± .010
	.85	.085 ± .009	.096 ± .010	.086 ± .010	.080 ± .010	.102 ± .011	.087 ± .010	.097 ± .010	.090 ± .010	.075 ± .009	.092 ± .010	.107 ± .010	.098 ± .010	.083 ± .009	.131 ± .010	.148 ± .012	.063 ± .009	**.061 ± .009**	.093 ± .009
	.80	.058 ± .008	.081 ± .009	.058 ± .008	.067 ± .010	.090 ± .009	.078 ± .009	.085 ± .010	.060 ± .008	.068 ± .010	.095 ± .010	.105 ± .010	.082 ± .010	.061 ± .008	.118 ± .011	.149 ± .012	.054 ± .009	**.040 ± .007**	.075 ± .009
	.75	.048 ± .007	.073 ± .009	.051 ± .008	.077 ± .009	.077 ± .009	.061 ± .009	.078 ± .010	.052 ± .008	.065 ± .009	.077 ± .009	.098 ± .010	.068 ± .010	.050 ± .008	.111 ± .011	.144 ± .012	.045 ± .008	**.040 ± .007**	.067 ± .009
	.70	.041 ± .007	.056 ± .008	.043 ± .007	.059 ± .009	.071 ± .010	.045 ± .008	.064 ± .009	.043 ± .007	.046 ± .008	.074 ± .010	.088 ± .010	.052 ± .009	.041 ± .008	.108 ± .011	.140 ± .012	**.031 ± .007**	.038 ± .007	.057 ± .008
$\hat{\phi}$	.99	.992 ± .003	.993 ± .002	.997 ± .002	.992 ± .003	.994 ± .002	.992 ± .003	**.991 ± .003**	.989 ± .003	.983 ± .004	.988 ± .003	.994 ± .003	.991 ± .003	.984 ± .004	.980 ± .004	.986 ± .003	.967 ± .005	.978 ± .004	**.989 ± .003**
	.95	.961 ± .006	.946 ± .007	.955 ± .006	.958 ± .007	.947 ± .006	.938 ± .007	**.949 ± .006**	.934 ± .008	.942 ± .007	.952 ± .007	.955 ± .007	.959 ± .006	.944 ± .007	.958 ± .006	**.948 ± .007**	.894 ± .010	.895 ± .010	.961 ± .006
	.90	.926 ± .008	.881 ± .010	.901 ± .010	.881 ± .011	.895 ± .009	.893 ± .008	.885 ± .010	.895 ± .009	.895 ± .010	.912 ± .008	.902 ± .010	.912 ± .008	.914 ± .009	.907 ± .010	.894 ± .009	.817 ± .012	.816 ± .012	**.898 ± .008**
	.85	.852 ± .011	.833 ± .012	.850 ± .011	.830 ± .011	.849 ± .011	**.849 ± .010**	.847 ± .011	.863 ± .011	.822 ± .012	.849 ± .011	.867 ± .012	.858 ± .009	.847 ± .012	.853 ± .012	.834 ± .012	.750 ± .012	.748 ± .012	.858 ± .009
	.80	.788 ± .013	.798 ± .013	.782 ± .013	.778 ± .013	.793 ± .012	.812 ± .012	.800 ± .012	.786 ± .013	.776 ± .012	.807 ± .012	.832 ± .013	.805 ± .011	**.801 ± .013**	.812 ± .014	.793 ± .013	.708 ± .013	.694 ± .013	.813 ± .011
	.75	.740 ± .013	.766 ± .013	.737 ± .013	.773 ± .013	**.750 ± .012**	.757 ± .013	.773 ± .013	.740 ± .013	.754 ± .014	.743 ± .012	.777 ± .014	.769 ± .012	.746 ± .014	.762 ± .014	.748 ± .015	.667 ± .014	.675 ± .013	.767 ± .013
	.70	.691 ± .014	.717 ± .013	.701 ± .013	**.700 ± .013**	.692 ± .014	.701 ± .013	.731 ± .013	.699 ± .013	.706 ± .013	.715 ± .014	.731 ± .014	.725 ± .013	.693 ± .014	.713 ± .015	.707 ± .016	.631 ± .014	.657 ± .014	.705 ± .014
$MinCoeff$	.99	1.012 ± .030	1.010 ± .030	1.007 ± .030	1.004 ± .030	1.010 ± .030	1.006 ± .030	1.010 ± .029	1.007 ± .030	1.004 ± .030	1.002 ± .030	1.006 ± .030	**1.007 ± .030**	1.015 ± .029	**1.000 ± .030**	.992 ± .030	.997 ± .031	1.007 ± .031	**1.005 ± .030**
	.95	1.030 ± .030	1.007 ± .030	.992 ± .030	**1.002 ± .029**	1.003 ± .030	.997 ± .031	1.015 ± .030	1.017 ± .030	**1.001 ± .030**	1.001 ± .030	.993 ± .031	1.018 ± .030	1.028 ± .030	.997 ± .030	.957 ± .031	.983 ± .032	.975 ± .031	1.004 ± .030
	.90	1.032 ± .031	1.006 ± .032	1.011 ± .031	1.010 ± .031	1.003 ± .032	**.998 ± .031**	.996 ± .031	1.019 ± .031	1.007 ± .031	1.011 ± .031	.965 ± .032	1.016 ± .031	1.045 ± .031	.980 ± .031	.912 ± .031	.947 ± .032	.953 ± .032	**1.005 ± .030**
	.85	1.029 ± .031	.990 ± .033	1.015 ± .032	1.019 ± .032	.978 ± .033	1.001 ± .032	.988 ± .033	1.026 ± .031	1.022 ± .031	.991 ± .033	.962 ± .033	**1.006 ± .031**	1.028 ± .031	.943 ± .033	.868 ± .032	.923 ± .034	.945 ± .034	1.007 ± .031
	.80	1.019 ± .032	.983 ± .034	1.013 ± .032	.991 ± .034	.952 ± .034	.987 ± .032	.970 ± .033	1.013 ± .032	.970 ± .034	.953 ± .034	.933 ± .035	.978 ± .033	1.028 ± .031	.927 ± .034	.839 ± .034	.895 ± .036	.938 ± .035	**1.011 ± .032**
	.75	.982 ± .033	.963 ± .034	.974 ± .034	.964 ± .034	.937 ± .035	.966 ± .034	.957 ± .035	.975 ± .034	.966 ± .034	.928 ± .036	.900 ± .036	.961 ± .034	.987 ± .034	.884 ± .034	.800 ± .034	.865 ± .036	.911 ± .036	**1.009 ± .033**
	.70	.932 ± .035	.938 ± .035	.942 ± .033	.908 ± .036	.873 ± .038	.932 ± .035	.933 ± .035	.939 ± .034	.943 ± .034	.899 ± .035	.869 ± .037	**.944 ± .035**	.936 ± .035	.828 ± .035	.772 ± .033	.845 ± .037	.887 ± .037	1.065 ± .033

Table C.35: Results for `letter`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.122 ± .005	.016 ± .002	.014 ± .002	.017 ± .002	.017 ± .002	.026 ± .002	.015 ± .002	**.012 ± .002**	.014 ± .002	.014 ± .002	**.012 ± .002**	**.012 ± .002**	.033 ± .003	.061 ± .004	.048 ± .003	.020 ± .002
	.95	.105 ± .005	.006 ± .001	**.005 ± .001**	.013 ± .002	.012 ± .002	.014 ± .002	.006 ± .001	**.005 ± .001**	.006 ± .001	.006 ± .001	**.005 ± .001**	**.005 ± .001**	.028 ± .003	.058 ± .004	.047 ± .003	.009 ± .002
	.90	.084 ± .005	**.002 ± .001**	**.002 ± .001**	.007 ± .001	.009 ± .002	.006 ± .001	**.002 ± .001**	**.002 ± .001**	**.002 ± .001**	**.002 ± .001**	**.002 ± .001**	**.002 ± .001**	.022 ± .003	.048 ± .004	.047 ± .004	.005 ± .001
	.85	.070 ± .005	.001 ± .000	.002 ± .001	.007 ± .001	.007 ± .001	.003 ± .001	**.000 ± .000**	**.000 ± .000**	.001 ± .001	**.000 ± .000**	.001 ± .001	.001 ± .001	.019 ± .002	.041 ± .003	.046 ± .004	.002 ± .001
	.80	.053 ± .004	**.000 ± .000**	.002 ± .001	.004 ± .001	.006 ± .001	.002 ± .001	**.000 ± .000**	.001 ± .001	.001 ± .001	.001 ± .001	.001 ± .001	**.000 ± .000**	.015 ± .002	.039 ± .003	.044 ± .004	.001 ± .001
	.75	.042 ± .004	**.000 ± .000**	.001 ± .001	.003 ± .001	.003 ± .001	.001 ± .001	**.000 ± .000**	.001 ± .001	**.000 ± .000**	.001 ± .001	**.000 ± .000**	**.000 ± .000**	.014 ± .002	.039 ± .003	.044 ± .004	.001 ± .000
	.70	.037 ± .004	**.000 ± .000**	.001 ± .001	.003 ± .001	.003 ± .001	.001 ± .001	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.012 ± .002	.038 ± .003	.042 ± .004	.001 ± .000
$\hat{\phi}$	.99	.991 ± .001	.994 ± .001	.995 ± .001	**.990 ± .002**	.993 ± .001	.994 ± .001	.992 ± .002	.993 ± .001	.986 ± .002	**.990 ± .002**	.993 ± .001	.993 ± .001	.993 ± .001	**.990 ± .002**	.992 ± .002	.991 ± .001
	.95	.955 ± .003	.957 ± .003	.953 ± .004	.956 ± .003	**.950 ± .003**	.957 ± .003	.954 ± .003	.954 ± .003	.949 ± .004	.949 ± .003	.951 ± .003	.953 ± .003	.949 ± .004	.954 ± .003	.939 ± .004	.959 ± .003
	.90	.904 ± .004	**.903 ± .005**	**.903 ± .005**	.911 ± .004	.908 ± .004	.907 ± .005	.903 ± .005	.905 ± .005	**.901 ± .005**	**.902 ± .005**	**.902 ± .005**	**.903 ± .005**	**.901 ± .005**	.907 ± .005	.890 ± .005	.923 ± .005
	.85	.860 ± .006	.857 ± .006	.848 ± .006	.861 ± .005	.853 ± .006	.855 ± .006	.860 ± .006	.854 ± .006	.851 ± .007	.855 ± .006	.854 ± .006	.854 ± .006	.863 ± .005	**.849 ± .005**	.842 ± .005	.882 ± .005
	.80	.807 ± .006	.811 ± .006	**.801 ± .006**	.806 ± .006	.810 ± .006	**.799 ± .007**	.810 ± .006	.805 ± .006	.798 ± .007	.797 ± .007	.811 ± .006	.808 ± .006	**.803 ± .006**	.807 ± .006	.789 ± .006	.838 ± .006
	.75	.756 ± .007	.757 ± .007	.751 ± .007	.745 ± .007	.751 ± .007	.755 ± .007	.764 ± .007	**.749 ± .007**	.759 ± .007	.753 ± .007	.763 ± .007	.759 ± .007	**.751 ± .007**	.766 ± .007	.736 ± .006	.793 ± .006
	.70	.707 ± .008	.698 ± .007	.703 ± .008	.701 ± .007	.711 ± .008	.698 ± .008	.702 ± .007	.697 ± .007	.708 ± .008	.712 ± .008	.716 ± .008	.715 ± .007	**.701 ± .007**	.710 ± .007	.690 ± .006	.747 ± .007

Table C.36: Results for `magic`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and $MinCoeff$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.225 ± .008	.142 ± .007	.155 ± .007	.143 ± .007	.148 ± .007	.155 ± .008	.143 ± .007	.153 ± .007	**.141 ± .007**	.147 ± .008	.153 ± .008	.149 ± .007	.147 ± .008	.152 ± .008	.157 ± .007	.158 ± .008	.159 ± .008	.155 ± .008
	.95	.217 ± .008	**.127 ± .007**	.141 ± .007	.129 ± .007	.144 ± .007	.136 ± .007	.128 ± .007	.142 ± .007	**.127 ± .007**	.142 ± .007	.147 ± .007	.134 ± .007	.147 ± .008	.147 ± .008	.153 ± .007	.147 ± .007	.153 ± .008	.153 ± .008
	.90	.203 ± .008	.109 ± .007	.123 ± .007	.112 ± .007	.125 ± .007	.118 ± .007	.110 ± .007	.123 ± .007	**.106 ± .007**	.125 ± .007	.140 ± .007	.118 ± .007	.120 ± .007	.142 ± .008	.144 ± .007	.125 ± .007	.147 ± .008	.146 ± .008
	.85	.186 ± .008	.099 ± .007	.108 ± .006	.099 ± .007	.121 ± .007	.112 ± .007	**.097 ± .007**	.106 ± .006	.098 ± .007	.121 ± .007	.134 ± .007	.105 ± .007	.107 ± .006	.134 ± .008	.139 ± .007	.114 ± .007	.137 ± .008	.142 ± .008
	.80	.170 ± .008	.087 ± .006	.092 ± .006	**.085 ± .006**	.100 ± .006	.098 ± .006	.088 ± .006	.094 ± .006	.086 ± .006	.099 ± .006	.129 ± .007	.095 ± .006	.099 ± .006	.120 ± .008	.130 ± .007	.101 ± .007	.124 ± .008	.133 ± .008
	.75	.156 ± .007	.076 ± .006	.084 ± .006	**.071 ± .006**	.090 ± .006	.088 ± .006	.075 ± .006	.083 ± .006	.075 ± .006	.090 ± .006	.119 ± .007	.086 ± .006	.086 ± .006	.112 ± .007	.124 ± .007	.089 ± .006	.108 ± .008	.116 ± .008
	.70	.144 ± .008	.072 ± .006	.072 ± .006	**.069 ± .006**	.086 ± .005	.075 ± .006	**.069 ± .006**	.070 ± .006	.073 ± .006	.087 ± .006	.105 ± .007	.072 ± .005	.072 ± .006	.105 ± .008	.119 ± .007	.078 ± .006	.093 ± .007	.103 ± .008
$\hat{\phi}$	.99	.986 ± .003	.981 ± .002	.995 ± .001	**.990 ± .002**	.988 ± .002	.997 ± .001	.987 ± .002	.992 ± .002	.989 ± .002	.990 ± .002	.992 ± .002	.984 ± .002	.985 ± .002	.990 ± .002	.986 ± .002	.992 ± .002	.994 ± .002	.991 ± .002
	.95	.954 ± .004	.936 ± .005	.955 ± .005	.945 ± .004	.959 ± .004	.945 ± .004	.942 ± .005	.959 ± .004	.951 ± .005	.952 ± .004	.953 ± .005	.941 ± .005	.936 ± .005	.957 ± .004	.956 ± .004	**.951 ± .004**	.958 ± .004	.949 ± .005
	.90	.904 ± .006	.880 ± .006	.891 ± .007	**.886 ± .006**	**.897 ± .006**	.883 ± .006	.890 ± .006	.892 ± .006	**.887 ± .006**	**.898 ± .006**	.915 ± .006	.885 ± .006	.889 ± .006	.909 ± .006	.887 ± .005	.904 ± .006	.907 ± .006	.911 ± .006
	.85	**.849 ± .006**	.843 ± .007	.841 ± .008	.845 ± .007	.855 ± .007	.839 ± .007	.839 ± .007	.836 ± .007	.846 ± .007	.851 ± .007	.870 ± .007	.837 ± .007	.841 ± .007	.857 ± .007	.845 ± .007	.860 ± .007	.864 ± .007	.870 ± .007
	.80	.798 ± .007	.783 ± .008	.791 ± .008	.788 ± .008	.808 ± .007	.793 ± .008	.778 ± .008	.797 ± .008	.790 ± .009	.809 ± .008	.819 ± .009	.793 ± .008	**.799 ± .007**	**.800 ± .007**	.805 ± .007	.805 ± .007	.817 ± .008	.822 ± .007
	.75	**.749 ± .008**	.724 ± .009	.744 ± .008	.731 ± .008	.762 ± .009	.736 ± .009	.734 ± .009	.744 ± .008	.734 ± .009	.760 ± .009	.758 ± .009	.745 ± .009	.747 ± .008	.752 ± .008	.760 ± .008	.762 ± .008	.764 ± .009	.768 ± .008
	.70	**.698 ± .008**	.687 ± .010	**.698 ± .009**	.688 ± .009	.700 ± .009	.693 ± .010	.686 ± .009	.697 ± .009	.708 ± .008	.705 ± .009	.700 ± .009	.694 ± .009	.704 ± .009	.701 ± .009	.719 ± .009	.718 ± .009	.714 ± .010	.719 ± .009
$MinCoeff$	.99	1.001 ± .019	.996 ± .019	**.997 ± .019**	**1.001 ± .019**	.996 ± .019	.997 ± .019	**.999 ± .019**	.999 ± .019	.999 ± .019	.998 ± .019	.999 ± .019	.994 ± .019	**1.001 ± .019**	.999 ± .019	.995 ± .019	**.998 ± .019**	**1.002 ± .019**	1.003 ± .019
	.95	1.007 ± .020	.989 ± .019	**1.000 ± .019**	**1.002 ± .019**	.996 ± .019	.999 ± .019	.996 ± .019	**.998 ± .019**	.999 ± .020	.997 ± .019	.990 ± .020	.998 ± .019	.995 ± .019	1.013 ± .020	1.000 ± .019	.991 ± .020	1.014 ± .020	1.029 ± .020
	.90	1.017 ± .020	.994 ± .020	**.998 ± .019**	**.999 ± .019**	**.998 ± .020**	1.009 ± .020	.990 ± .020	**.998 ± .019**	.994 ± .020	**1.001 ± .019**	.984 ± .020	**1.003 ± .020**	.994 ± .020	1.034 ± .021	1.017 ± .020	.989 ± .020	1.042 ± .021	1.046 ± .021
	.85	1.023 ± .021	.997 ± .022	**.995 ± .021**	1.010

Table C.37: Results for `miniboone`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.065 ± .002	.061 ± .002	**.057 ± .002**	.059 ± .002	.066 ± .002	.061 ± .002	.060 ± .002	.058 ± .002	.060 ± .002	.067 ± .002	.062 ± .002	.060 ± .002	.067 ± .002	.075 ± .002	.073 ± .002	.064 ± .002	.068 ± .002	.062 ± .002
	.95	.055 ± .002	.049 ± .002	.047 ± .002	.047 ± .002	.052 ± .002	.046 ± .002	**.044 ± .002**	**.044 ± .002**	.046 ± .002	.050 ± .002	.051 ± .002	.046 ± .002	.051 ± .002	.071 ± .002	.074 ± .002	.050 ± .002	.066 ± .002	.048 ± .002
	.90	.047 ± .002	.036 ± .002	.033 ± .002	.033 ± .002	.038 ± .002	.033 ± .002	**.032 ± .002**	**.032 ± .002**	.033 ± .002	.035 ± .002	.040 ± .002	.034 ± .002	.038 ± .002	.067 ± .002	.073 ± .002	.036 ± .002	.064 ± .002	.036 ± .002
	.85	.041 ± .002	.024 ± .001	.023 ± .001	.024 ± .001	.027 ± .001	.022 ± .001	.023 ± .001	**.021 ± .001**	.024 ± .001	.029 ± .001	.031 ± .002	.023 ± .001	.027 ± .001	.065 ± .002	.071 ± .002	.025 ± .001	.058 ± .002	.029 ± .002
	.80	.036 ± .002	.017 ± .001	.017 ± .001	.018 ± .001	.020 ± .001	.018 ± .001	**.016 ± .001**	**.016 ± .001**	.018 ± .001	.020 ± .001	.023 ± .001	.017 ± .001	.021 ± .001	.062 ± .002	.070 ± .002	.019 ± .001	.049 ± .002	.022 ± .001
	.75	.030 ± .002	.013 ± .001	**.012 ± .001**	.014 ± .001	.017 ± .001	**.012 ± .001**	**.012 ± .001**	**.012 ± .001**	.013 ± .001	.016 ± .001	.018 ± .001	.013 ± .001	.015 ± .001	.060 ± .002	.069 ± .002	.013 ± .001	.037 ± .002	.017 ± .001
	.70	.027 ± .001	.010 ± .001	**.009 ± .001**	.010 ± .001	.012 ± .001	.010 ± .001	**.009 ± .001**	**.009 ± .001**	.010 ± .001	.013 ± .001	.014 ± .001	.010 ± .001	.012 ± .001	.059 ± .002	.069 ± .003	.010 ± .001	.029 ± .002	.015 ± .001
$\hat{\phi}$	.99	.989 ± .001	.987 ± .001	.988 ± .001	.988 ± .001	.988 ± .001	.991 ± .001	.989 ± .001	.990 ± .001	.991 ± .002	.992 ± .001	.990 ± .001	.990 ± .001	.986 ± .001	.986 ± .001	.991 ± .001	.989 ± .001	.994 ± .001	**.990 ± .001**
	.95	.946 ± .002	.947 ± .002	.952 ± .002	**.950 ± .002**	.946 ± .002	.947 ± .002	.947 ± .002	**.950 ± .002**	.951 ± .002	.947 ± .002	.946 ± .002	.950 ± .002	.947 ± .002	.945 ± .002	.949 ± .002	.946 ± .002	.971 ± .001	.946 ± .002
	.90	.895 ± .003	.897 ± .003	.895 ± .003	.894 ± .003	.895 ± .003	.896 ± .003	.897 ± .003	.896 ± .003	.897 ± .003	.894 ± .003	.897 ± .003	**.900 ± .003**	**.899 ± .003**	.898 ± .003	**.899 ± .003**	.892 ± .003	.939 ± .002	.894 ± .003
	.85	.847 ± .003	.843 ± .003	.845 ± .003	.843 ± .003	.839 ± .003	.840 ± .003	.845 ± .003	.846 ± .003	.845 ± .003	.840 ± .003	.845 ± .003	**.850 ± .003**	.845 ± .003	.850 ± .003	.849 ± .003	.837 ± .003	.904 ± .002	.843 ± .003
	.80	**.798 ± .003**	.790 ± .003	**.798 ± .003**	.790 ± .003	.790 ± .003	.792 ± .004	.790 ± .004	.795 ± .003	.791 ± .003	.791 ± .003	.793 ± .003	**.800 ± .003**	.794 ± .004	**.800 ± .003**	.798 ± .003	.786 ± .003	.864 ± .003	.790 ± .004
	.75	.740 ± .004	.741 ± .004	.743 ± .004	.735 ± .004	.740 ± .004	.741 ± .004	.738 ± .004	.746 ± .004	.734 ± .004	.738 ± .004	.742 ± .004	**.750 ± .004**	.740 ± .004	**.748 ± .004**	**.750 ± .004**	.729 ± .004	.820 ± .003	.734 ± .004
	.70	.692 ± .004	.692 ± .004	.693 ± .004	.686 ± .004	.689 ± .004	.692 ± .004	.689 ± .004	.691 ± .004	.683 ± .004	.686 ± .004	.693 ± .004	**.700 ± .004**	.691 ± .004	.697 ± .004	.704 ± .003	.670 ± .004	.771 ± .004	.685 ± .004
MinCoeff	.99	**1.000 ± .008**	**1.000 ± .008**	**.999 ± .008**	**1.000 ± .008**	**.999 ± .008**	**.999 ± .008**	**.999 ± .008**	**.999 ± .008**	**.999 ± .008**	**1.000 ± .008**	**1.001 ± .008**	**.999 ± .008**	**1.000 ± .008**	.996 ± .008	.995 ± .008	**.999 ± .008**	1.005 ± .008	**1.004 ± .008**
	.95	.990 ± .008	.993 ± .008	.990 ± .008	.994 ± .008	.996 ± .008	.990 ± .009	.993 ± .008	.991 ± .008	.996 ± .008	.994 ± .008	1.005 ± .008	.996 ± .009	**.997 ± .008**	.992 ± .009	.964 ± .008	.988 ± .009	1.022 ± .009	1.016 ± .008
	.90	.978 ± .009	.982 ± .009	.972 ± .008	.983 ± .008	.984 ± .009	.981 ± .009	.982 ± .009	.980 ± .009	.986 ± .008	.984 ± .009	1.006 ± .009	.985 ± .009	.991 ± .009	**.997 ± .009**	.956 ± .009	.972 ± .009	1.048 ± .009	1.032 ± .008
	.85	.954 ± .009	.968 ± .009	.955 ± .009	.973 ± .009	.976 ± .008	.966 ± .009	.968 ± .009	.961 ± .009	.978 ± .009	.987 ± .009	**1.002 ± .009**	.973 ± .009	.980 ± .009	.998 ± .009	.967 ± .009	.946 ± .009	1.072 ± .009	1.044 ± .008
	.80	.923 ± .009	.948 ± .009	.936 ± .009	.952 ± .009	.943 ± .009	.948 ± .009	.944 ± .009	.938 ± .009	.952 ± .009	.954 ± .009	**.995 ± .009**	.957 ± .009	.961 ± .009	.991 ± .010	.983 ± .009	.916 ± .009	1.097 ± .009	1.060 ± .009
	.75	.879 ± .009	.925 ± .009	.904 ± .010	.917 ± .009	.916 ± .009	.922 ± .009	.912 ± .009	.907 ± .010	.918 ± .009	.924 ± .009	.988 ± .010	.931 ± .009	.935 ± .009	.993 ± .010	**1.000 ± .009**	.877 ± .010	1.118 ± .009	1.077 ± .009
	.70	.830 ± .009	.892 ± .009	.869 ± .010	.879 ± .010	.891 ± .010	.890 ± .009	.883 ± .009	.873 ± .010	.874 ± .010	.894 ± .010	**.989 ± .010**	.905 ± .010	.904 ± .010	.985 ± .011	1.014 ± .009	.824 ± .010	1.133 ± .009	1.089 ± .010

Table C.38: Results for `MNIST`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.005 ± .001	.009 ± .001	.010 ± .001	**.002 ± .000**	.016 ± .001	.003 ± .000	.007 ± .001	.008 ± .001	.003 ± .000	.012 ± .001	.003 ± .000	**.002 ± .000**	.007 ± .001	.010 ± .001	.013 ± .001	.004 ± .000
	.95	**.001 ± .000**	.004 ± .001	.003 ± .000	**.001 ± .000**	.002 ± .000	**.001 ± .000**	.002 ± .000	.002 ± .000	**.001 ± .000**	.002 ± .000	**.001 ± .000**	**.001 ± .000**	.002 ± .000	.010 ± .001	.013 ± .001	**.001 ± .000**
	.90	**.000 ± .000**	.003 ± .000	.001 ± .000	.001 ± .000	.001 ± .000	**.000 ± .000**	.001 ± .000	.001 ± .000	.001 ± .000	.001 ± .000	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.010 ± .001	.013 ± .001	**.000 ± .000**
	.85	**.000 ± .000**	.002 ± .000	.001 ± .000	**.000 ± .000**	.003 ± .001	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.001 ± .000	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.009 ± .001	.013 ± .001	**.000 ± .000**
	.80	**.000 ± .000**	.001 ± .000	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.009 ± .001	.013 ± .001	**.000 ± .000**
	.75	**.000 ± .000**	.001 ± .000	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.008 ± .001	.013 ± .001	**.000 ± .000**
	.70	**.000 ± .000**	.001 ± .000	**.000 ± .000**	**.000 ± .000**	.001 ± .000	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.001 ± .000	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.008 ± .001	.012 ± .001	**.000 ± .000**
$\hat{\phi}$	.99	.989 ± .001	.991 ± .001	.988 ± .001	.976 ± .001	.988 ± .001	.991 ± .001	.991 ± .001	.989 ± .001	.991 ± .001	.989 ± .001	.989 ± .001	.988 ± .001	.992 ± .001	.991 ± .001	**.990 ± .001**	.990 ± .001
	.95	.949 ± .002	.952 ± .002	.948 ± .002	.933 ± .002	.948 ± .002	.948 ± .002	.949 ± .002	.950 ± .002	.948 ± .002	**.950 ± .002**	.950 ± .002	.949 ± .002	.948 ± .002	.949 ± .002	.949 ± .002	.954 ± .002
	.90	**.900 ± .003**	.898 ± .003	.898 ± .003	.882 ± .003	.897 ± .003	**.901 ± .003**	.893 ± .003	**.899 ± .003**	.903 ± .003	**.900 ± .003**	.894 ± .002	.895 ± .002	.895 ± .002	.903 ± .002	.894 ± .003	.906 ± .002
	.85	.848 ± .003	.842 ± .003	.845 ± .003	.842 ± .003	.853 ± .003	.847 ± .003	.839 ± .003	.848 ± .003	.848 ± .003	.852 ± .003	.846 ± .003	.844 ± .003	.840 ± .003	**.849 ± .003**	.845 ± .003	.862 ± .003
	.80	.792 ± .003	.788 ± .004	**.802 ± .003**	.792 ± .004	**.802 ± .003**	**.799 ± .003**	.792 ± .004	.797 ± .003	.798 ± .004	.811 ± .003	.795 ± .003	.794 ± .003	.791 ± .003	**.800 ± .003**	.789 ± .004	.818 ± .003
	.75	.745 ± .004	.741 ± .004	.754 ± .003	.738 ± .004	.747 ± .004	.754 ± .004	.743 ± .004	.753 ± .004	.745 ± .004	.793 ± .003	.745 ± .003	**.750 ± .004**	.740 ± .004	.747 ± .004	.744 ± .004	.772 ± .004
	.70	.695 ± .004	.694 ± .004	**.701 ± .004**	**.699 ± .004**	**.700 ± .004**	.708 ± .004	.693 ± .004	.709 ± .004	.699 ± .004	.694 ± .004	.694 ± .004	.706 ± .004	.690 ± .004	.697 ± .004	.690 ± .004	.721 ± .004

Table C.39: Results for `octmnist`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.099 ± .002	.083 ± .002	.079 ± .002	.077 ± .002	.088 ± .002	.082 ± .002	.083 ± .002	.079 ± .002	.077 ± .002	.088 ± .002	.075 ± .002	**.072 ± .002**	.089 ± .002	.111 ± .002	.108 ± .002	.079 ± .002
	.95	.088 ± .002	.067 ± .002	.063 ± .002	.063 ± .002	.076 ± .002	.067 ± .002	.066 ± .002	.063 ± .002	.061 ± .002	.073 ± .002	.065 ± .002	**.056 ± .002**	.073 ± .002	.103 ± .002	.108 ± .002	.065 ± .002
	.90	.076 ± .002	.052 ± .002	.046 ± .001	.051 ± .002	.059 ± .002	.049 ± .002	.050 ± .001	.045 ± .001	.047 ± .002	.058 ± .002	.052 ± .002	**.042 ± .001**	.055 ± .002	.093 ± .002	.108 ± .002	.049 ± .001
	.85	.066 ± .002	.039 ± .001	.035 ± .001	.039 ± .002	.046 ± .002	.040 ± .001	.039 ± .001	.034 ± .001	.037 ± .001	.044 ± .001	.042 ± .002	**.031 ± .001**	.043 ± .002	.083 ± .002	.108 ± .002	.037 ± .001
	.80	.059 ± .002	.030 ± .001	.028 ± .001	.030 ± .001	.054 ± .002	.030 ± .001	.030 ± .001	.028 ± .001	.029 ± .001	.053 ± .002	.033 ± .001	**.023 ± .001**	.032 ± .001	.073 ± .002	.108 ± .002	.029 ± .001
	.75	.053 ± .002	.024 ± .001	.022 ± .001	.024 ± .001	.037 ± .001	.024 ± .001	.024 ± .001	.022 ± .001	.023 ± .001	.038 ± .002	.026 ± .001	**.018 ± .001**	.026 ± .001	.066 ± .002	.108 ± .002	.024 ± .001
	.70	.048 ± .001	.019 ± .001	.018 ± .001	.018 ± .001	.024 ± .001	.019 ± .001	.018 ± .001	.018 ± .001	.018 ± .001	.023 ± .001	.021 ± .001	**.014 ± .001**	.021 ± .001	.057 ± .002	.108 ± .003	.019 ± .001
$\hat{\phi}$	.99	**.990 ± .001**	**.990 ± .001**	.991 ± .001	**.990 ± .001**	.992 ± .001	.989 ± .001	**.990 ± .001**	.991 ± .001	**.990 ± .001**	.992 ± .001	**.990 ± .001**	.989 ± .001	**.990 ± .001**	.989 ± .001	.989 ± .001	.991 ± .001
	.95	.952 ± .002	.949 ± .001	**.950 ± .002**	.947 ± .002	.952 ± .001	.952 ± .002	.947 ± .001	.952 ± .002	.946 ± .002	**.951 ± .002**	**.950 ± .001**	.946 ± .001	**.950 ± .001**	.951 ± .002	.948 ± .002	.954 ± .002
	.90	.902 ± .002	**.899 ± .002**	**.899 ± .002**	.897 ± .002	**.900 ± .002**	**.899 ± .002**	**.899 ± .002**	.898 ± .002	.897 ± .002	**.899 ± .002**	.896 ± .002	.898 ± .002	.897 ± .002	.898 ± .002	**.900 ± .002**	.906 ± .002
	.85	.849 ± .002	.848 ± .002	.849 ± .002	.848 ± .002	**.850 ± .002**	.852 ± .002	.851 ± .002	.849 ± .002	.851 ± .002	.847 ± .002	.843 ± .002	.846 ± .002	**.850 ± .003**	.844 ± .002	**.849 ± .003**	.857 ± .002
	.80	.804 ± .003	.798 ± .003	**.800 ± .003**	.798 ± .002	.796 ± .003	.799 ± .003	.799 ± .003	.800 ± .003	.798 ± .002	.794 ± .002	.796 ± .002	.795 ± .003	.793 ± .003	.794 ± .003	.795 ± .003	.809 ± .003
	.75	.754 ± .003	**.751 ± .003**	**.750 ± .003**	.747 ± .003	.754 ± .003	.747 ± .003	**.749 ± .003**	.751 ± .003	.748 ± .003	.749 ± .003	.748 ± .003	.746 ± .003	.747 ± .003	.748 ± .003	.745 ± .003	.762 ± .003
	.70	.705 ± .003	.695 ± .003	.699 ± .003	.697 ± .003	.697 ± .003	.696 ± .003	.697 ± .003	**.699 ± .003**	**.699 ± .003**	.698 ± .003	.703 ± .003	.697 ± .003	.692 ± .003	.699 ± .003	.692 ± .003	.718 ± .003

Table C.40: Results for `online`: $mean \pm std$ for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	**.091 ± .006**	.095 ± .006	.093 ± .006	.100 ± .006	.099 ± .006	.094 ± .006	.094 ± .006	.092 ± .006	.102 ± .006	.101 ± .006	.093 ± .006	.093 ± .006	.092 ± .006	.094 ± .006	.103 ± .006	.092 ± .006	.096 ± .006	.099 ± .006
	.95	.079 ± .006	.087 ± .006	**.078 ± .006**	.083 ± .006	.089 ± .006	.081 ± .006	.081 ± .006	.082 ± .006	.084 ± .006	.088 ± .006	.091 ± .006	**.078 ± .006**	.081 ± .006	.093 ± .006	.106 ± .007	.085 ± .006	.095 ± .006	.099 ± .006
	.90	.066 ± .005	.073 ± .005	.067 ± .005	.071 ± .005	.068 ± .006	.064 ± .005	.067 ± .005	.065 ± .005	.068 ± .005	.069 ± .006	.084 ± .006	**.063 ± .005**	.069 ± .005	.087 ± .006	.108 ± .007	.070 ± .005	.095 ± .006	.098 ± .006
	.85	.056 ± .005	.061 ± .005	.054 ± .005	.059 ± .006	.062 ± .006	.051 ± .005	.053 ± .004	.055 ± .005	.060 ± .006	.062 ± .006	.075 ± .006	.049 ± .005	.051 ± .005	.082 ± .006	.111 ± .008	**.047 ± .005**	.088 ± .006	.095 ± .007
	.80	.048 ± .005	.047 ± .005	.040 ± .004	.046 ± .005	.051 ± .005	.040 ± .005	.042 ± .004	.042 ± .004	.047 ± .005	.049 ± .005	.068 ± .005	**.038 ± .004**	.043 ± .005	.079 ± .006	.113 ± .008	.039 ± .004	.081 ± .006	.090 ± .007
	.75	.039 ± .005	.038 ± .004	**.031 ± .004**	.041 ± .005	.041 ± .005	.032 ± .004	.035 ± .004	**.031 ± .004**	.041 ± .005	.042 ± .004	.060 ± .005	.033 ± .004	.036 ± .004	.071 ± .006	.115 ± .008	.032 ± .004	.067 ± .006	.084 ± .007
	.70	.032 ± .004	.031 ± .004	**.026 ± .004**	.030 ± .004	.039 ± .005	.028 ± .004	.027 ± .004	.028 ± .004	.033 ± .004	.035 ± .004	.052 ± .005	.028 ± .004	.033 ± .005	.057 ± .005	.120 ± .008	.028 ± .004	.065 ± .006	.071 ± .007
$\hat{\phi}$	.99	.988 ± .002	.988 ± .002	**.991 ± .002**	**.991 ± .002**	.988 ± .002	.988 ± .002	**.990 ± .002**	.987 ± .002	**.992 ± .002**	**.990 ± .002**	**.990 ± .002**	.994 ± .002	.987 ± .002	.986 ± .002	**.989 ± .002**	.993 ± .002	.993 ± .002	.988 ± .002
	.95	.948 ± .005	.957 ± .004	.955 ± .004	.952 ± .004	.954 ± .004	.958 ± .004	.955 ± .004	.958 ± .004	.960 ± .004	.960 ± .003	.954 ± .004	.957 ± .004	.954 ± .004	**.951 ± .004**	.944 ± .005	.971 ± .003	.962 ± .004	.953 ± .004
	.90	.902 ± .006	.907 ± .006	.925 ± .006	.908 ± .006	.920 ± .006	.914 ± .006	.915 ± .005	.920 ± .006	.901 ± .007	.923 ± .006	.896 ± .006	.913 ± .006	.916 ± .006	.895 ± .006	.887 ± .007	.934 ± .005	.916 ± .006	**.902 ± .005**
	.85	.869 ± .008	.868 ± .007	.869 ± .008	.869 ± .007	.862 ± .008	.862 ± .008	.868 ± .007	.868 ± .007	.870 ± .007	.871 ± .007	**.851 ± .007**	.864 ± .008	.857 ± .008	.834 ± .007	.829 ± .008	.854 ± .008	.858 ± .007	.844 ± .007
	.80	.829 ± .008	.821 ± .009	.820 ± .009	.820 ± .008	.822 ± .009	.819 ± .009	.819 ± .008	.822 ± .009	.826 ± .008	.824 ± .009	.816 ± .008	.820 ± .009	.821 ± .008	.790 ± .008	.795 ± .009	.806 ± .008	**.795 ± .008**	.790 ± .008
	.75	.779 ± .009	.777 ± .009	.767 ± .009	.775 ± .010	.776 ± .009	.770 ± .009	.776 ± .009	.769 ± .009	.780 ± .009	.779 ± .009	.768 ± .009	.777 ± .009	.778 ± .008	.733 ± .009	**.742 ± .009**	.766 ± .010	.740 ± .009	.731 ± .008
	.70	.708 ± .009	.732 ± .009	.712 ± .009	.721 ± .011	.729 ± .010	.732 ± .009	.716 ± .010	.719 ± .009	.718 ± .010	.718 ± .010	.707 ± .011	.729 ± .009	.710 ± .009	.680 ± .009	.690 ± .009	.720 ± .010	**.698 ± .010**	.675 ± .009
MinCoeff	.99	.983 ± .046	**991 ± .047**	.990 ± .049	.978 ± .047	.976 ± .048	.972 ± .047	.982 ± .048	.968 ± .048	.972 ± .047	.983 ± .047	**994 ± .047**	.994 ± .048	.981 ± .048	.991 ± .048	1.015 ± .048	.993 ± .048	1.006 ± .048	1.008 ± .047
	.95	.913 ± .047	.893 ± .045	.887 ± .048	.866 ± .046	.890 ± .048	.911 ± .050	.907 ± .047	.901 ± .047	.911 ± .046	.900 ± .046	.933 ± .046	.901 ± .047	.900 ± .048	.986 ± .050	1.060 ± .049	.938 ± .047	**1.010 ± .046**	1.018 ± .047
	.90	.811 ± .046	.759 ± .042	.822 ± .047	.751 ± .047	.782 ± .046	.819 ± .048	.818 ± .046	.809 ± .046	.734 ± .044	.803 ± .045	.821 ± .048	.780 ± .046	.793 ± .044	.972 ± .051	1.045 ± .051	.867 ± .046	**1.031 ± .046**	1.034 ± .049
	.85	.735 ± .045	.638 ± .039	.675 ± .045	.727 ± .042	.699 ± .043	.630 ± .042	.686 ± .042	.667 ± .044	.721 ± .041	.700 ± .043	.750 ± .045	.625 ± .042	.657 ± .042	.945 ± .052	1.046 ± .052	.630 ± .044	**1.012 ± .052**	1.046 ± .052
	.80	.684 ± .045	.502 ± .038	.585 ± .045	.472 ± .038	.459 ± .038	.554 ± .043	.567 ± .041	.590 ± .044	.536 ± .038	.489 ± .039	.685 ± .043	.533 ± .040	.548 ± .041	.900 ± .053	1.063 ± .051	.539 ± .041	**.997 ± .055**	1.042 ± .056
	.75	.614 ± .047	.374 ± .033	.484 ± .040	.264 ± .031	.267 ± .030	.467 ± .040	.492 ± .039	.479 ± .040	.304 ± .031	.270 ± .028	.605 ± .043	.462 ± .039	.411 ± .039	.806 ± .050	1.068 ± .055	.475 ± .039	**.965 ± .058**	1.040 ± .057
	.70	.580 ± .046	.205 ± .027	.416 ± .039	.193 ± .028	.250 ± .030	.418 ± .040	.430 ± .038	.426 ± .038	.213 ± .028	.229 ± .029	.489 ± .043	.372 ± .039	.363 ± .040	.730 ± .053	1.094 ± .055	.433 ± .037	.964 ± .060	**.994 ± .058**

Table C.41: Results for `organamnist`: $mean \pm std$ for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.005 ± .001	.002 ± .001	.003 ± .000	.003 ± .000	**.001 ± .000**	.002 ± .000	.002 ± .000	.002 ± .000	**.001 ± .000**	**.001 ± .000**	**.001 ± .000**	**.001 ± .000**	.007 ± .001	.009 ± .001	.009 ± .001	.002 ± .000
	.95	.002 ± .000	**.000 ± .000**	.001 ± .000	.001 ± .000	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.002 ± .001	.008 ± .001	.009 ± .001	**.000 ± .000**
	.90	.001 ± .000	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.008 ± .001	.009 ± .001	**.000 ± .000**
	.85	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.045 ± .002	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.008 ± .001	.009 ± .001	**.000 ± .000**
	.80	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.048 ± .002	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.008 ± .001	.009 ± .001	**.000 ± .000**
	.75	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.002 ± .000	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.007 ± .001	.010 ± .001	**.000 ± .000**
	.70	.001 ± .000	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.006 ± .001	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.007 ± .001	.010 ± .001	**.000 ± .000**
$\hat{\phi}$	.99	.987 ± .001	**.990 ± .001**	**.989 ± .001**	.987 ± .001	**.989 ± .001**	**.991 ± .001**	**.990 ± .001**	.987 ± .001	.992 ± .001	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	.988 ± .001	**.991 ± .001**	**.991 ± .001**	**.990 ± .001**
	.95	.952 ± .002	.954 ± .002	**.950 ± .002**	**.951 ± .002**	.955 ± .002	.954 ± .002	**.950 ± .002**	**.949 ± .002**	.954 ± .002	.954 ± .002	.956 ± .002	.955 ± .002	**.952 ± .002**	.952 ± .002	**.952 ± .002**	.956 ± .002
	.90	.905 ± .002	.904 ± .003	**.900 ± .003**	.906 ± .003	.904 ± .003	.909 ± .002	**.901 ± .003**	**.900 ± .003**	.902 ± .003	.910 ± .003	.904 ± .003	.903 ± .003	.905 ± .003	**.900 ± .003**	.906 ± .002	.910 ± .003
	.85	**.848 ± .003**	.853 ± .003	**.849 ± .003**	**.852 ± .003**	**.853 ± .003**	.859 ± .003	**.852 ± .003**	.846 ± .003	.858 ± .003	.922 ± .002	.853 ± .003	.855 ± .003	.858 ± .003	**.851 ± .003**	.855 ± .003	.869 ± .003
	.80	**.800 ± .004**	.805 ± .003	**.802 ± .003**	.807 ± .004	.806 ± .004	.806 ± .004	.808 ± .003	.802 ± .004	.808 ± .004	.953 ± .002	.805 ± .003	.805 ± .003	.807 ± .003	.803 ± .004	.805 ± .004	.829 ± .003
	.75	**.750 ± .003**	.755 ± .004	.755 ± .004	.768 ± .004	.760 ± .004	.758 ± .004	.757 ± .004	.758 ± .004	.758 ± .004	.961 ± .002	.753 ± .004	.756 ± .004	.762 ± .003	.753 ± .004	.753 ± .004	.787 ± .003
	.70	.705 ± .004	.698 ± .004	**.700 ± .004**	.710 ± .004	.709 ± .004	.706 ± .004	.720 ± .004	.703 ± .004	.715 ± .004	.951 ± .002	.712 ± .004	.710 ± .004	.709 ± .004	.705 ± .004	**.700 ± .004**	.745 ± .003

Table C.42: Results for `organcmnist`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.034 ± .003	.023 ± .002	.023 ± .002	.021 ± .002	.025 ± .002	.020 ± .002	.023 ± .002	.021 ± .002	.020 ± .002	.024 ± .002	.019 ± .002	.018 ± .002	.033 ± .002	.043 ± .003	.046 ± .003	**.017 ± .002**
	.95	.023 ± .002	.012 ± .002	.014 ± .002	.013 ± .002	.016 ± .002	.010 ± .001	.010 ± .001	.012 ± .002	.009 ± .001	.012 ± .002	.008 ± .001	**.005 ± .001**	.027 ± .002	.039 ± .003	.045 ± .003	.007 ± .001
	.90	.015 ± .002	.004 ± .001	.007 ± .001	.007 ± .001	.005 ± .001	.005 ± .001	.003 ± .001	.007 ± .001	.003 ± .001	.004 ± .001	.004 ± .001	**.001 ± .001**	.020 ± .002	.036 ± .003	.046 ± .003	.003 ± .001
	.85	.011 ± .002	.002 ± .001	.004 ± .001	.006 ± .001	.003 ± .001	.002 ± .001	**.001 ± .000**	.004 ± .001	.003 ± .001	.002 ± .001	.003 ± .001	**.001 ± .000**	.016 ± .002	.032 ± .003	.046 ± .003	.001 ± .001
	.80	.008 ± .001	**.001 ± .000**	.002 ± .001	.004 ± .001	.010 ± .001	.001 ± .001	**.001 ± .000**	.002 ± .001	.001 ± .001	.001 ± .001	**.001 ± .000**	**.001 ± .000**	.012 ± .002	.030 ± .003	.044 ± .003	**.001 ± .000**
	.75	.006 ± .001	.001 ± .000	.001 ± .001	.001 ± .001	.002 ± .001	**.000 ± .000**	.001 ± .000	.001 ± .000	.001 ± .000	.040 ± .003	**.000 ± .000**	**.000 ± .000**	.008 ± .002	.028 ± .003	.040 ± .003	**.000 ± .000**
	.70	.005 ± .001	.001 ± .000	.001 ± .001	.002 ± .001	.001 ± .001	**.000 ± .000**	.001 ± .000	.001 ± .000	**.000 ± .000**	.111 ± .006	**.000 ± .000**	**.000 ± .000**	.007 ± .002	.026 ± .003	.041 ± .004	**.000 ± .000**
$\hat{\phi}$	.99	.992 ± .001	.991 ± .001	.992 ± .001	**.990 ± .001**	**.991 ± .001**	**.991 ± .001**	**.991 ± .001**	**.989 ± .001**	**.991 ± .001**	**.990 ± .001**	.994 ± .001	.993 ± .001	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**	**.990 ± .001**
	.95	.958 ± .003	.952 ± .003	.953 ± .003	.958 ± .003	.950 ± .003	.958 ± .003	.954 ± .003	.955 ± .003	.960 ± .003	**.950 ± .003**	.950 ± .003	.953 ± .003	**.950 ± .003**	.956 ± .003	.949 ± .003	.961 ± .003
	.90	.911 ± .004	.897 ± .005	**.902 ± .004**	.891 ± .005	.897 ± .005	.903 ± .005	**.900 ± .005**	.902 ± .005	.903 ± .004	.897 ± .004	.905 ± .005	.904 ± .005	.897 ± .005	.910 ± .004	.900 ± .005	.919 ± .004
	.85	.862 ± .005	**.850 ± .005**	.842 ± .006	.853 ± .005	.855 ± .005	.848 ± .006	**.850 ± .005**	**.851 ± .005**	.856 ± .005	.854 ± .005	**.850 ± .005**	.853 ± .005	.854 ± .006	.858 ± .005	.859 ± .005	.869 ± .005
	.80	.809 ± .006	.796 ± .006	.794 ± .006	.809 ± .006	.792 ± .005	.801 ± .006	.800 ± .006	.796 ± .006	.801 ± .006	.804 ± .005	.799 ± .006	.798 ± .006	.801 ± .006	.811 ± .005	**.801 ± .005**	.825 ± .006
	.75	.754 ± .006	.742 ± .006	.752 ± .007	.761 ± .006	.753 ± .006	.745 ± .006	.750 ± .007	.750 ± .007	.766 ± .006	.757 ± .006	.757 ± .006	.757 ± .007	.754 ± .006	.767 ± .006	**.748 ± .006**	.783 ± .006
	.70	.710 ± .006	.698 ± .007	.705 ± .007	.704 ± .007	.694 ± .007	.703 ± .007	**.701 ± .007**	.706 ± .007	.708 ± .007	.715 ± .007	**.701 ± .007**	.703 ± .007	.706 ± .006	.718 ± .006	**.702 ± .006**	.725 ± .007

Table C.43: Results for `organsmnist`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.162 ± .005	.072 ± .004	.066 ± .004	.077 ± .004	.079 ± .004	.066 ± .003	.071 ± .004	.066 ± .004	.077 ± .004	.076 ± .004	.055 ± .003	**.053 ± .003**	.065 ± .004	.114 ± .005	.116 ± .005	.064 ± .004
	.95	.146 ± .004	.057 ± .004	.052 ± .003	.063 ± .004	.063 ± .003	.051 ± .003	.054 ± .003	.050 ± .003	.055 ± .004	.066 ± .004	.043 ± .003	**.036 ± .003**	.051 ± .003	.107 ± .005	.116 ± .005	.049 ± .003
	.90	.134 ± .004	.039 ± .003	.032 ± .002	.041 ± .003	.038 ± .003	.031 ± .003	.035 ± .003	.029 ± .002	.038 ± .003	.044 ± .003	.037 ± .003	**.016 ± .002**	.036 ± .003	.098 ± .005	.117 ± .005	.032 ± .002
	.85	.121 ± .004	.022 ± .002	.017 ± .002	.026 ± .002	.033 ± .003	.017 ± .002	.018 ± .002	.015 ± .002	.018 ± .002	.041 ± .003	.033 ± .002	**.008 ± .001**	.031 ± .003	.089 ± .005	.116 ± .005	.017 ± .002
	.80	.106 ± .004	.013 ± .002	.010 ± .002	.017 ± .002	.015 ± .002	.008 ± .001	.010 ± .002	.006 ± .001	.011 ± .002	.037 ± .003	.026 ± .003	**.004 ± .001**	.028 ± .003	.082 ± .005	.114 ± .005	.008 ± .001
	.75	.098 ± .004	.006 ± .002	.006 ± .001	.010 ± .002	.007 ± .001	.004 ± .001	.004 ± .001	**.002 ± .001**	.006 ± .001	.042 ± .004	.017 ± .002	**.002 ± .001**	.025 ± .003	.077 ± .005	.110 ± .005	.005 ± .001
	.70	.093 ± .004	.004 ± .001	.004 ± .001	.007 ± .002	.003 ± .001	.003 ± .001	.003 ± .001	.001 ± .001	.003 ± .001	.028 ± .003	.008 ± .002	**.001 ± .000**	.022 ± .003	.072 ± .005	.105 ± .005	.002 ± .001
$\hat{\phi}$	.99	**.989 ± .002**	.993 ± .001	**.989 ± .002**	**.990 ± .001**	**.990 ± .001**	.989 ± .002	.988 ± .002	.988 ± .002	.994 ± .001	.986 ± .002	.994 ± .001	**.990 ± .002**	**.990 ± .001**	.988 ± .002	**.991 ± .001**	.987 ± .002
	.95	.941 ± .003	.955 ± .003	.952 ± .003	.955 ± .003	.956 ± .003	.955 ± .003	**.950 ± .003**	.952 ± .003	.946 ± .003	.948 ± .003	.942 ± .003	**.950 ± .003**	**.950 ± .003**	.941 ± .003	.953 ± .003	.957 ± .002
	.90	.890 ± .004	.903 ± .004	**.902 ± .004**	.903 ± .004	.897 ± .004	.902 ± .004	.902 ± .004	.906 ± .004	.899 ± .004	.894 ± .004	.892 ± .004	.899 ± .004	.907 ± .004	**.901 ± .004**	.902 ± .004	.914 ± .004
	.85	.845 ± .005	**.852 ± .005**	**.852 ± .005**	.856 ± .005	.845 ± .005	**.852 ± .005**	**.849 ± .005**	.858 ± .005	.855 ± .005	.834 ± .006	.855 ± .005	**.849 ± .005**	.860 ± .005	.847 ± .005	.857 ± .005	.866 ± .005
	.80	.798 ± .006	.805 ± .006	.810 ± .005	.803 ± .006	.803 ± .006	**.800 ± .006**	.799 ± .006	.809 ± .005	.798 ± .006	.793 ± .006	.809 ± .006	.808 ± .006	.812 ± .006	.792 ± .006	.809 ± .006	.817 ± .006
	.75	.754 ± .006	.743 ± .006	.774 ± .006	.761 ± .006	.753 ± .007	.738 ± .006	.744 ± .007	.762 ± .006	**.750 ± .006**	.757 ± .007	.753 ± .007	.759 ± .007	.756 ± .007	.747 ± .007	.756 ± .006	.769 ± .006
	.70	.708 ± .007	.702 ± .007	.719 ± .006	.709 ± .006	.701 ± .007	.695 ± .007	**.699 ± .007**	.720 ± .006	.705 ± .007	.699 ± .007	.703 ± .007	.705 ± .007	.710 ± .007	**.701 ± .007**	.704 ± .006	.724 ± .007

Table C.44: Results for `oxfordpets`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.317 ± .012	.053 ± .006	.087 ± .007	**.039 ± .005**	.166 ± .009	.059 ± .006	.050 ± .006	.090 ± .007	.042 ± .006	.166 ± .010	**.039 ± .005**	.041 ± .005	.050 ± .006	.169 ± .009	.170 ± .009	.041 ± .005	.040 ± .005	.057 ± .006
	.95	.315 ± .012	.035 ± .006	.072 ± .007	.055 ± .006	.120 ± .008	.039 ± .006	.036 ± .006	.075 ± .006	.047 ± .005	.123 ± .009	.029 ± .005	.029 ± .005	.050 ± .006	.165 ± .009	.172 ± .009	**.029 ± .004**	.030 ± .005	.038 ± .005
	.90	.315 ± .013	.022 ± .004	.058 ± .006	.024 ± .005	.131 ± .009	.025 ± .005	.025 ± .004	.057 ± .006	.021 ± .004	.125 ± .008	.018 ± .004	**.014 ± .003**	.046 ± .005	.163 ± .009	.166 ± .009	.017 ± .004	.020 ± .004	.027 ± .005
	.85	.314 ± .013	.016 ± .003	.044 ± .005	**.006 ± .002**	.180 ± .010	.020 ± .004	.016 ± .003	.042 ± .005	.007 ± .002	.178 ± .010	.008 ± .003	.008 ± .003	.036 ± .005	.161 ± .009	.168 ± .010	.013 ± .003	.013 ± .003	.016 ± .004
	.80	.315 ± .013	.013 ± .003	.032 ± .005	.018 ± .004	.107 ± .007	.012 ± .003	.012 ± .003	.033 ± .005	.020 ± .004	.090 ± .007	**.004 ± .002**	**.004 ± .002**	.030 ± .005	.156 ± .009	.165 ± .010	.007 ± .002	.006 ± .002	.013 ± .003
	.75	.316 ± .014	.005 ± .002	.029 ± .005	.004 ± .002	.038 ± .006	.007 ± .002	.006 ± .002	.024 ± .004	.005 ± .002	.035 ± .005	**.003 ± .002**	**.003 ± .002**	.021 ± .005	.155 ± .010	.167 ± .011	.005 ± .002	.005 ± .002	.010 ± .003
	.70	.313 ± .014	.003 ± .001	.016 ± .004	.006 ± .002	.150 ± .010	.006 ± .002	.003 ± .001	.020 ± .004	.005 ± .002	.159 ± .011	**.001 ± .001**	**.001 ± .001**	.015 ± .004	.155 ± .010	.172 ± .011	**.001 ± .001**	.002 ± .002	.007 ± .003
$\hat{\phi}$	.99	**.992 ± .002**	.992 ± .002	.980 ± .003	.985 ± .003	**.989 ± .003**	.992 ± .002	**.989 ± .003**	.982 ± .004	**.990 ± .003**	.982 ± .004	**.989 ± .003**	**.989 ± .003**	.985 ± .003	.987 ± .003	.985 ± .003	**.991 ± .002**	**.992 ± .002**	**.990 ± .003**
	.95	.950 ± .006	**.949 ± .006**	.936 ± .006	.951 ± .006	.929 ± .007	.943 ± .006	.953 ± .005	.947 ± .006	.958 ± .005	.949 ± .006	.952 ± .005	.951 ± .006	.943 ± .006	.935 ± .006	.945 ± .006	.953 ± .006	.957 ± .005	.943 ± .006
	.90	.912 ± .008	.904 ± .007	.890 ± .007	.901 ± .008	.898 ± .008	.896 ± .008	.911 ± .007	.888 ± .008	.892 ± .008	**.901 ± .007**	.907 ± .007	.903 ± .007	.903 ± .008	.888 ± .008	.883 ± .008	.902 ± .008	.908 ± .008	.898 ± .008
	.85	.860 ± .010	**.849 ± .009**	.835 ± .009	.827 ± .009	.819 ± .010	.855 ± .010	.848 ± .009	.835 ± .009	.834 ± .010	.864 ± .008	.827 ± .009	.828 ± .009	.843 ± .009	.854 ± .009	.831 ± .009	.858 ± .010	.864 ± .010	.844 ± .010
	.80	.819 ± .011	.803 ± .010	.770 ± .010	**.798 ± .009**	.831 ± .008	.800 ± .012	.802 ± .010	.776 ± .010	.780 ± .010	.794 ± .011	.771 ± .010	.771 ± .010	.788 ± .010	.811 ± .009	.768 ± .011	.807 ± .010	.809 ± .010	.800 ± .011
	.75	.758 ± .012	**.751 ± .010**	.734 ± .010	.741 ± .012	.737 ± .012	.750 ± .012	.753 ± .010	.725 ± .011	.751 ± .011	.730 ± .012	.729 ± .012	.725 ± .012	.739 ± .010	.768 ± .010	.707 ± .013	.755 ± .011	.755 ± .011	.755 ± .013
	.70	.717 ± .013	.693 ± .011	.676 ± .011	.676 ± .012	.695 ± .013	.691 ± .013	.696 ± .011	.677 ± .011	.676 ± .012	.703 ± .013	.674 ± .011	.681 ± .011	**.698 ± .010**	.708 ± .011	.669 ± .013	.703 ± .011	.714 ± .011	.697 ± .013
MinCoeff	.99	1.008 ± .018	1.005 ± .018	1.011 ± .018	1.013 ± .018	1.005 ± .018	1.007 ± .018	1.008 ± .018	1.005 ± .018	1.008 ± .018	1.005 ± .018	1.006 ± .018	1.007 ± .018	.996 ± .019	1.004 ± .018	**1.001 ± .018**	1.007 ± .018	1.005 ± .018	1.005 ± .018
	.95	1.012 ± .018	1.014 ± .019	1.023 ± .019	1.019 ± .019	**1.004 ± .019**	1.021 ± .018	1.012 ± .019	1.016 ± .019	1.036 ± .018	1.012 ± .019	1.022 ± .018	1.025 ± .018	.977 ± .019	1.008 ± .020	.997 ± .019	1.014 ± .018	1.016 ± .018	1.024 ± .018
	.90	1.011 ± .019	1.021 ± .019	1.036 ± .019	1.034 ± .019	1.015 ± .019	1.044 ± .018	1.021 ± .018	1.033 ± .020	1.041 ± .019	1.020 ± .020	1.041 ± .018	1.043 ± .018	.965 ± .019	1.021 ± .020	**994 ± .019**	1.032 ± .019	1.036 ± .019	1.044 ± .018
	.85	1.013 ± .019	1.026 ± .019	1.061 ± .020	1.063 ± .019	1.035 ± .019	1.054 ± .019	1.030 ± .019	1.055 ± .020	1.069 ± .019	1.028 ± .019	1.066 ± .018	1.067 ± .019	.951 ± .019	1.025 ± .020	**992 ± .020**	1.044 ± .019	1.053 ± .019	1.078 ± .018
	.80	**1.011 ± .019**	1.028 ± .020	1.093 ± .021	1.043 ± .019	1.046 ± .019	1.077 ± .019	1.028 ± .020	1.086 ± .020	1.015 ± .019	1.033 ± .020	1.082 ± .020	1.085 ± .020	.936 ± .021	1.030 ± .020	.997 ± .022	1.052 ± .020	1.067 ± .020	1.100 ± .019
	.75	1.010 ± .020	1.042 ± .019	1.119 ± .021	1.035 ± .020	1.070 ± .021	1.100 ± .020	1.043 ± .020	1.115 ± .021	**1.011 ± .019**	1.035 ± .022	1.100 ± .020	1.102 ± .020	.921 ± .022	1.031 ± .021	.987 ± .022	1.072 ± .020	1.097 ± .019	1.133 ± .020
	.70	**1.014 ± .021**	1.047 ± .021	1.159 ± .020	1.173 ± .018	1.086 ± .020	1.109 ± .021	1.048 ± .021	1.147 ± .020	1.134 ± .019	1.026 ± .022	1.120 ± .019	1.118 ± .020	.899 ± .022	1.033 ± .022	.980 ± .024	1.089 ± .021	1.116 ± .020	1.167 ± .020

Table C.45: Results for `pathmnist`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.027 ± .001	.011 ± .001	.020 ± .001	.014 ± .001	.022 ± .001	.010 ± .001	.011 ± .001	.019 ± .001	.012 ± .001	.019 ± .001	.007 ± .001	**.006 ± .001**	.020 ± .001	.028 ± .001	.033 ± .001	.008 ± .001
	.95	.019 ± .001	.004 ± .000	.010 ± .001	.005 ± .001	.016 ± .001	.003 ± .000	.003 ± .000	.008 ± .001	.005 ± .001	.013 ± .001	.002 ± .000	**.001 ± .000**	.006 ± .001	.028 ± .001	.034 ± .001	.003 ± .000
	.90	.013 ± .001	.002 ± .000	.004 ± .001	.003 ± .000	.023 ± .001	.001 ± .000	.002 ± .000	.004 ± .000	.002 ± .000	.022 ± .001	.001 ± .000	**.000 ± .000**	.002 ± .000	.027 ± .001	.035 ± .001	.001 ± .000
	.85	.010 ± .001	.001 ± .000	.002 ± .000	.001 ± .000	.004 ± .000	.001 ± .000	.001 ± .000	.002 ± .000	.001 ± .000	.004 ± .000	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.025 ± .001	.036 ± .001	.001 ± .000
	.80	.007 ± .001	**.000 ± .000**	.001 ± .000	.001 ± .000	.002 ± .000	.001 ± .000	**.000 ± .000**	.001 ± .000	.001 ± .000	.001 ± .000	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.024 ± .001	.037 ± .001	.001 ± .000
	.75	.006 ± .001	**.000 ± .000**	.001 ± .000	**.000 ± .000**	.012 ± .001	.001 ± .000	**.000 ± .000**	.001 ± .000	**.000 ± .000**	.010 ± .001	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.023 ± .001	.038 ± .001	**.000 ± .000**
	.70	.005 ± .001	**.000 ± .000**	.001 ± .000	.001 ± .000	.002 ± .000	.001 ± .000	**.000 ± .000**	.001 ± .000	**.000 ± .000**	.002 ± .000	**.000 ± .000**	**.000 ± .000**	.001 ± .000	.022 ± .001	.038 ± .001	**.000 ± .000**
$\hat{\phi}$	.99	.989 ± .001	.989 ± .001	**.989 ± .001**	**.990 ± .001**	.989 ± .001	.991 ± .001	**.990 ± .001**	.989 ± .001	**.990 ± .001**	.988 ± .001	.989 ± .001	.989 ± .001	.989 ± .001	.992 ± .001	**.990 ± .001**	.992 ± .001
	.95	.948 ± .002	**.951 ± .002**	**.949 ± .001**	.951 ± .002	.951 ± .002	**.950 ± .001**	**.950 ± .001**	**.950 ± .001**	.948 ± .002	.951 ± .002	**.951 ± .001**	**.950 ± .001**	.947 ± .001	.955 ± .002	.944 ± .002	.958 ± .001
	.90	.898 ± .002	**.901 ± .002**	**.899 ± .002**	**.899 ± .002**	.899 ± .002	**.900 ± .002**	.902 ± .002	**.901 ± .002**	**.901 ± .002**	.900 ± .002	.903 ± .002	**.902 ± .002**	.895 ± .002	.902 ± .002	.895 ± .002	.914 ± .002
	.85	.849 ± .002	.848 ± .002	.852 ± .003	.852 ± .002	.853 ± .002	**.850 ± .002**	.848 ± .002	.852 ± .002	.854 ± .002	.856 ± .002	**.851 ± .002**	.852 ± .002	.845 ± .002	.851 ± .002	**.849 ± .002**	.863 ± .002
	.80	.797 ± .003	.798 ± .002	.801 ± .003	.807 ± .002	.805 ± .003	**.800 ± .003**	.798 ± .003	.804 ± .003	.803 ± .002	.803 ± .003	.799 ± .002	.796 ± .002	.733 ± .003	.799 ± .003	**.800 ± .002**	.807 ± .003
	.75	.749 ± .003	**.751 ± .003**	.754 ± .003	.754 ± .003	.745 ± .003	.744 ± .003	.752 ± .003	.752 ± .003	**.750 ± .003**	.752 ± .003	**.751 ± .003**	.744 ± .003	.742 ± .003	**.749 ± .003**	.755 ± .003	.748 ± .003
	.70	.697 ± .003	.703 ± .003	.702 ± .003	.706 ± .003	**.700 ± .003**	.693 ± .003	.704 ± .003	.706 ± .003	.701 ± .003	.705 ± .003	.706 ± .003	.695 ± .003	.690 ± .003	**.701 ± .003**	.701 ± .003	.695 ± .003

Table C.46: Results for `phoneme`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.255 ± .017	.133 ± .013	.141 ± .015	.134 ± .014	.161 ± .015	.137 ± .015	.136 ± .014	.142 ± .015	**.127 ± .013**	.159 ± .015	.145 ± .014	.140 ± .015	.141 ± .015	.165 ± .014	.166 ± .014	.180 ± .015	.189 ± .015	.136 ± .014
	.95	.228 ± .017	.124 ± .014	.138 ± .015	.125 ± .015	.142 ± .015	.117 ± .014	.125 ± .014	**.116 ± .014**	.124 ± .015	.141 ± .015	.151 ± .015	.134 ± .015	.129 ± .014	.165 ± .015	.168 ± .014	.171 ± .016	.184 ± .015	.128 ± .014
	.90	.220 ± .017	.118 ± .014	.124 ± .014	.114 ± .013	.123 ± .015	**.103 ± .014**	.121 ± .014	.108 ± .014	.104 ± .013	.115 ± .014	.156 ± .016	.112 ± .014	.110 ± .013	.163 ± .015	.171 ± .014	.157 ± .016	.176 ± .015	.124 ± .015
	.85	.217 ± .017	.101 ± .013	.121 ± .014	**.081 ± .014**	.119 ± .014	.092 ± .014	.109 ± .014	.095 ± .013	.085 ± .013	.107 ± .014	.165 ± .017	.084 ± .013	.108 ± .013	.161 ± .015	.170 ± .015	.148 ± .015	.164 ± .016	.107 ± .015
	.80	.214 ± .018	.093 ± .013	.111 ± .014	.094 ± .014	.109 ± .014	.089 ± .014	.094 ± .013	.084 ± .013	.087 ± .013	.094 ± .013	.169 ± .018	**.083 ± .013**	.098 ± .013	.161 ± .015	.168 ± .015	.139 ± .015	.158 ± .016	.096 ± .014
	.75	.224 ± .018	.079 ± .013	.099 ± .014	.080 ± .012	.101 ± .014	.078 ± .013	.085 ± .014	.079 ± .012	.084 ± .013	.080 ± .012	.174 ± .019	**.067 ± .013**	.082 ± .012	.158 ± .015	.161 ± .015	.136 ± .015	.151 ± .016	.095 ± .014
	.70	.218 ± .018	.073 ± .012	.097 ± .015	.081 ± .014	.094 ± .013	.070 ± .012	.070 ± .012	.072 ± .012	.076 ± .014	.070 ± .012	.178 ± .019	.063 ± .012	**.063 ± .011**	.158 ± .016	.156 ± .016	.133 ± .016	.147 ± .016	.093 ± .015
$\hat{\phi}$	.99	.978 ± .005	.984 ± .005	.992 ± .004	.989 ± .004	1.000 ± .000	.997 ± .002	.984 ± .005	**.990 ± .003**	.970 ± .007	.990 ± .004	.988 ± .004	.994 ± .003	.989 ± .004	.995 ± .003	.988 ± .004	.980 ± .005	.994 ± .003	**.992 ± .003**
	.95	.921 ± .011	.948 ± .010	.961 ± .008	.931 ± .009	.960 ± .008	.938 ± .009	.945 ± .009	.929 ± .010	.949 ± .009	.936 ± .010	.947 ± .009	.975 ± .007	.954 ± .008	.946 ± .009	.966 ± .007	**.950 ± .008**	.961 ± .008	.939 ± .009
	.90	.882 ± .012	.907 ± .012	.927 ± .009	.894 ± .012	.943 ± .011	.895 ± .011	.909 ± .012	.893 ± .012	.892 ± .012	.858 ± .012	.898 ± .012	.916 ± .011	.899 ± .011	.922 ± .010	**.902 ± .010**	.914 ± .011	.912 ± .011	.907 ± .012
	.85	.837 ± .014	.857 ± .013	.874 ± .012	.832 ± .016	.866 ± .014	.840 ± .014	.865 ± .014	.827 ± .014	.870 ± .014	.832 ± .014	.851 ± .016	.851 ± .014	.861 ± .012	.890 ± .012	**.847 ± .012**	.882 ± .013	.867 ± .014	.839 ± .015
	.80	.791 ± .015	.819 ± .014	.813 ± .016	.786 ± .015	.799 ± .017	.812 ± .016	.820 ± .015	.785 ± .016	.787 ± .015	.797 ± .016	.787 ± .017	.804 ± .016	.828 ± .014	.841 ± .015	**.791 ± .013**	.844 ± .014	.828 ± .015	.811 ± .016
	.75	.736 ± .016	.760 ± .017	.752 ± .016	.767 ± .017	.730 ± .018	.764 ± .018	.762 ± .017	.734 ± .016	.767 ± .016	.740 ± .018	.764 ± .018	.754 ± .017	.769 ± .015	.804 ± .014	**.755 ± .014**	.805 ± .017	.818 ± .016	.755 ± .018
	.70	.693 ± .018	.704 ± .018	.691 ± .019	.735 ± .019	.706 ± .018	.729 ± .019	.685 ± .018	.699 ± .018	.733 ± .018	.678 ± .019	.739 ± .019	.704 ± .019	**.705 ± .016**	.765 ± .017	**.707 ± .015**	.765 ± .018	.778 ± .018	.712 ± .019
MinCoeff	.99	1.011 ± .038	1.006 ± .037	1.008 ± .037	.993 ± .037	**1.000 ± .037**	**1.000 ± .037**	1.008 ± .037	1.007 ± .037	1.006 ± .038	.993 ± .037	**1.001 ± .037**	1.003 ± .038	1.002 ± .037	1.005 ± .037	.988 ± .037	1.008 ± .037	1.003 ± .037	**1.001 ± .037**
	.95	1.037 ± .039	1.008 ± .038	1.009 ± .038	.984 ± .039	.988 ± .037	1.017 ± .038	1.003 ± .037	1.019 ± .038	.992 ± .039	.990 ± .038	1.029 ± .038	**1.004 ± .037**	1.005 ± .037	1.050 ± .040	.974 ± .037	1.020 ± .038	1.015 ± .039	1.026 ± .037
	.90	1.032 ± .039	.998 ± .039	1.018 ± .038	1.012 ± .038	**993 ± .037**	1.023 ± .039	.998 ± .038	1.011 ± .038	1.029 ± .038	.975 ± .039	1.086 ± .038	1.000 ± .038	1.016 ± .037	1.063 ± .040	.975 ± .037	1.029 ± .039	1.052 ± .039	1.045 ± .037
	.85	1.010 ± .040	.987 ± .040	1.017 ± .039	**995 ± .038**	.974 ± .039	1.013 ± .039	.998 ± .039	1.016 ± .039	1.019 ± .038	.970 ± .038	1.084 ± .040	1.020 ± .038	.999 ± .039	1.090 ± .042	.967 ± .036	1.049 ± .040	1.085 ± .040	1.075 ± .039
	.80	1.002 ± .041	.992 ± .041	1.019 ± .041	.989 ± .040	.987 ± .039	1.007 ± .040	.964 ± .041	1.022 ± .042	.988 ± .040	.971 ± .039	1.110 ± .042	**1.006 ± .040**	.998 ± .041	1.128 ± .045	.950 ± .037	1.065 ± .041	1.113 ± .042	1.104 ± .039
	.75	.961 ± .042	.995 ± .042	.987 ± .042	**995 ± .038**	.996 ± .040	1.005 ± .042	.967 ± .042	1.012 ± .042	.990 ± .038	.967 ± .039	1.124 ± .042	1.008 ± .040	.988 ± .042	1.155 ± .044	.939 ± .039	1.073 ± .041	1.119 ± .042	1.147 ± .042
	.70	.938 ± .043	.995 ± .044	.965 ± .045	1.006 ± .042	**1.004 ± .040**	1.007 ± .042	.958 ± .043	.993 ± .041	1.035 ± .042	.951 ± .041	1.118 ± .044	1.019 ± .043	.984 ± .044	1.259 ± .047	.910 ± .041	1.069 ± .043	1.143 ± .042	1.165 ± .043

Table C.47: Results for `pneumoniamnist`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.038 ± .006	.033 ± .005	.041 ± .006	.036 ± .006	.042 ± .006	.033 ± .005	.032 ± .005	.041 ± .006	.037 ± .006	.045 ± .006	.031 ± .005	.030 ± .005	**.028 ± .005**	.040 ± .006	.047 ± .006	.045 ± .006	.048 ± .006	.033 ± .005
	.95	.035 ± .006	.020 ± .005	.026 ± .006	.025 ± .005	.015 ± .004	.017 ± .004	.021 ± .005	.024 ± .005	.022 ± .004	.015 ± .004	.022 ± .005	**.012 ± .003**	.019 ± .004	.040 ± .006	.047 ± .006	.032 ± .006	.031 ± .005	.019 ± .004
	.90	.025 ± .005	.013 ± .004	.018 ± .005	**.007 ± .002**	**.007 ± .002**	.010 ± .003	.013 ± .004	.015 ± .004	.008 ± .003	.011 ± .003	.013 ± .003	.010 ± .003	.014 ± .004	.040 ± .006	.045 ± .007	.019 ± .005	.018 ± .004	.009 ± .003
	.85	.019 ± .005	.008 ± .003	.011 ± .004	.006 ± .003	.009 ± .003	.006 ± .002	.008 ± .003	.012 ± .004	.006 ± .003	.010 ± .003	.005 ± .002	**.004 ± .002**	.010 ± .004	.037 ± .007	.045 ± .007	.013 ± .004	.009 ± .003	.005 ± .002
	.80	.015 ± .004	.005 ± .002	.009 ± .004	.006 ± .002	.007 ± .003	.005 ± .002	.005 ± .002	.009 ± .004	.004 ± .002	.062 ± .008	.004 ± .002	**.003 ± .002**	.010 ± .004	.037 ± .007	.045 ± .007	.007 ± .003	.008 ± .003	.004 ± .002
	.75	.009 ± .003	.002 ± .001	.006 ± .003	.002 ± .001	.008 ± .003	.004 ± .002	.003 ± .002	.006 ± .003	.002 ± .001	.054 ± .008	**.001 ± .001**	.002 ± .002	.010 ± .004	.035 ± .007	.046 ± .008	**.001 ± .001**	.006 ± .003	.002 ± .001
	.70	.008 ± .003	.002 ± .002	.004 ± .002	.002 ± .002	.005 ± .002	.002 ± .002	.001 ± .001	.004 ± .002	.003 ± .002	.083 ± .009	**.000 ± .000**	**.000 ± .000**	.008 ± .004	.033 ± .007	.047 ± .008	**.000 ± .000**	.004 ± .002	.002 ± .001
$\hat{\phi}$	.99	.985 ± .003	.986 ± .003	.993 ± .002	.994 ± .002	**.989 ± .003**	.984 ± .003	.985 ± .003	.994 ± .002	.996 ± .002	.993 ± .003	**.991 ± .003**	**.989 ± .003**	.985 ± .003	.987 ± .003	.992 ± .002	.986 ± .003	.985 ± .004	.989 ± .003
	.95	**.947 ± .006**	.949 ± .007	.934 ± .007	.964 ± .005	.957 ± .006	.945 ± .006	.952 ± .007	.943 ± .006	.963 ± .005	.954 ± .006	.961 ± .006	.945 ± .007	.940 ± .007	.939 ± .008	**.952 ± .006**	.946 ± .007	.927 ± .007	.954 ± .006
	.90	.902 ± .009	.906 ± .008	.892 ± .008	.895 ± .008	**.902 ± .008**	.906 ± .008	.904 ± .009	.889 ± .009	.900 ± .008	.917 ± .007	.912 ± .008	.924 ± .008	.875 ± .010	.895 ± .009	.904 ± .008	.869 ± .009	.845 ± .009	.904 ± .009
	.85	.863 ± .010	.859 ± .010	**.848 ± .009**	.867 ± .009	.840 ± .010	.857 ± .009	.862 ± .011	.843 ± .010	.864 ± .009	.848 ± .010	.847 ± .010	.860 ± .009	.822 ± .011	.852 ± .011	.837 ± .009	.784 ± .011	.778 ± .012	.864 ± .010
	.80	.806 ± .011	**.805 ± .011**	.793 ± .011	.807 ± .011	.790 ± .011	.805 ± .011	.806 ± .011	.789 ± .011	.795 ± .011	.801 ± .012	.810 ± .010	**.800 ± .011**	.779 ± .011	.795 ± .013	.784 ± .013	.706 ± .012	.702 ± .013	.809 ± .011
	.75	.745 ± .013	.741 ± .011	.761 ± .013	.758 ± .012	.731 ± .013	.746 ± .012	.744 ± .011	.751 ± .012	.749 ± .013	.721 ± .013	.742 ± .012	.739 ± .013	.732 ± .012	.728 ± .015	.727 ± .014	.633 ± .012	.638 ± .013	**.749 ± .012**
	.70	.711 ± .013	**.700 ± .013**	.716 ± .014	.696 ± .013	.673 ± .015	.686 ± .013	.687 ± .012	.694 ± .013	**.701 ± .013**	.686 ± .015	.688 ± .013	.682 ± .013	.672 ± .013	.682 ± .015	.694 ± .015	.557 ± .014	.571 ± .014	.722 ± .012
MinCoeff	.99	.995 ± .017	1.000 ± .017	**1.000 ± .017**	**1.004 ± .016**	**.999 ± .016**	1.006 ± .017	**1.001 ± .016**	**1.002 ± .017**	**1.002 ± .016**	**1.000 ± .016**	.999 ± .017	**1.002 ± .017**	.998 ± .017	1.006 ± .017	**1.001 ± .017**	**1.002 ± .017**	1.012 ± .017	1.007 ± .017
	.95	.990 ± .017	1.005 ± .017	1.023 ± .017	1.005 ± .017	1.021 ± .017	1.027 ± .018	1.008 ± .017	1.016 ± .017	1.005 ± .017	1.023 ± .017	1.016 ± .017	1.015 ± .018	.992 ± .018	1.014 ± .016	**.998 ± .017**	1.007 ± .017	1.048 ± .017	1.019 ± .017
	.90	.991 ± .018	1.020 ± .018	1.038 ± .018	1.017 ± .018	1.052 ± .018	1.051 ± .018	1.016 ± .018	1.037 ± .018	1.015 ± .017	1.055 ± .017	1.017 ± .018	1.019 ± .018	.978 ± .019	1.017 ± .016	**.994 ± .017**	1.012 ± .019	1.108 ± .017	1.033 ± .018
	.85	.988 ± .019	1.026 ± .018	1.061 ± .019	1.030 ± .017	**.999 ± .019**	1.081 ± .017	1.020 ± .018	1.063 ± .018	1.032 ± .018	.991 ± .020	1.039 ± .018	1.039 ± .017	.963 ± .020	1.029 ± .017	.983 ± .018	1.029 ± .020	1.153 ± .017	1.048 ± .018
	.80	.986 ± .019	1.032 ± .019	1.091 ± .018	1.048 ± .019	1.058 ± .019	1.116 ± .017	1.029 ± .019	1.081 ± .018	1.046 ± .019	**.998 ± .019**	1.056 ± .019	1.052 ± .019	.948 ± .021	1.033 ± .017	.973 ± .019	1.058 ± .021	1.200 ± .015	1.068 ± .017
	.75	**.993 ± .020**	1.053 ± .019	1.107 ± .017	1.054 ± .020	1.224 ± .014	1.159 ± .016	1.042 ± .019	1.108 ± .018	1.024 ± .021	1.137 ± .017	1.071 ± .019	1.078 ± .019	.929 ± .022	1.026 ± .019	.982 ± .020	1.076 ± .023	1.250 ± .012	1.103 ± .018
	.70	**.994 ± .020**	1.055 ± .020	1.126 ± .017	1.090 ± .017	1.146 ± .017	1.205 ± .017	1.058 ± .019	1.145 ± .017	1.049 ± .018	1.106 ± .017	1.088 ± .019	1.100 ± .019	.900 ± .022	1.032 ± .020	.986 ± .020	1.085 ± .023	1.314 ± .010	1.127 ± .018

Table C.48: Results for `pol`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.047 ± .004	.049 ± .004	.060 ± .004	.052 ± .004	.061 ± .005	.070 ± .005	**.046 ± .004**	.061 ± .005	.051 ± .004	.061 ± .005	.061 ± .005	.062 ± .005	.075 ± .005	.063 ± .005	.081 ± .005	.105 ± .006
	.95	.040 ± .004	.038 ± .004	.050 ± .004	.043 ± .004	.058 ± .004	.045 ± .004	**.026 ± .004**	.047 ± .004	.038 ± .004	.064 ± .005	.049 ± .004	.039 ± .004	.063 ± .005	.062 ± .005	.075 ± .006	.084 ± .005
	.90	.028 ± .003	.024 ± .003	.035 ± .004	.034 ± .003	.042 ± .004	.030 ± .004	**.017 ± .003**	.029 ± .004	.025 ± .003	.063 ± .005	.037 ± .004	.023 ± .003	.045 ± .004	.063 ± .005	.070 ± .006	.059 ± .005
	.85	.014 ± .002	.010 ± .002	.017 ± .003	.017 ± .003	.035 ± .003	.014 ± .003	**.006 ± .002**	.017 ± .003	.018 ± .003	.049 ± .005	.023 ± .003	.012 ± .002	.029 ± .003	.063 ± .005	.067 ± .005	.032 ± .004
	.80	.005 ± .002	**.002 ± .001**	.003 ± .001	.003 ± .001	.003 ± .001	.007 ± .002	**.002 ± .001**	.004 ± .002	.004 ± .001	.047 ± .004	.007 ± .002	.003 ± .001	.006 ± .001	.064 ± .006	.065 ± .005	.008 ± .002
	.75	**.000 ± .000**	**.000 ± .000**	.001 ± .001	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.001 ± .001	**.000 ± .000**	.027 ± .003	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.066 ± .006	.064 ± .006	.001 ± .001
	.70	**.000 ± .000**	**.000 ± .000**	.000 ± .001	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.063 ± .005	**.000 ± .000**	**.000 ± .000**	**.000 ± .000**	.063 ± .006	.061 ± .005	**.000 ± .000**
$\hat{\phi}$	.99	.994 ± .002	.987 ± .002	.987 ± .002	.988 ± .002	.985 ± .002	.992 ± .002	.981 ± .002	.989 ± .002	.986 ± .002	.986 ± .002	.987 ± .002	.991 ± .002	**.990 ± .002**	.981 ± .003	.987 ± .002	.988 ± .002
	.95	.949 ± .004	**.950 ± .004**	.958 ± .003	.953 ± .003	.948 ± .004	.938 ± .004	.940 ± .004	.957 ± .004	.943 ± .004	.948 ± .004	.948 ± .004	.941 ± .004	.957 ± .004	.934 ± .004	.938 ± .004	.945 ± .004
	.90	.898 ± .006	.903 ± .006	.907 ± .005	.899 ± .006	.893 ± .006	.898 ± .006	.898 ± .006	.908 ± .006	.891 ± .006	**.899 ± .005**	.908 ± .005	.898 ± .006	.912 ± .006	.892 ± .005	.893 ± .006	.900 ± .006
	.85	.846 ± .007	.857 ± .007	.856 ± .007	.850 ± .007	.850 ± .007	.850 ± .007	.848 ± .008	.862 ± .007	.852 ± .007	.857 ± .007	.858 ± .007	.850 ± .008	.864 ± .007	**.847 ± .005**	.845 ± .007	.844 ± .007
	.80	.802 ± .008	.804 ± .008	.808 ± .008	.802 ± .008	.805 ± .008	.814 ± .008	.802 ± .008	.809 ± .008	.804 ± .008	.798 ± .008	.809 ± .008	.803 ± .008	.802 ± .008	**.798 ± .006**	.795 ± .008	.796 ± .008
	.75	.767 ± .008	.757 ± .008	**.752 ± .008**	.757 ± .009	.760 ± .008	.756 ± .008	.756 ± .008	.754 ± .008	.761 ± .009	.756 ± .008	.755 ± .009	.753 ± .009	.753 ± .009	.762 ± .007	**.749 ± .008**	.747 ± .009
	.70	.713 ± .009	.708 ± .009	.707 ± .009	.710 ± .009	.708 ± .009	.712 ± .009	.706 ± .009	.705 ± .009	.717 ± .009	.830 ± .006	.708 ± .009	.712 ± .009	.701 ± .009	.714 ± .007	**.700 ± .009**	.682 ± .009

Table C.49: Results for `retinamnist`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.502 ± .029	.463 ± .028	.492 ± .030	.498 ± .026	.453 ± .030	.451 ± .027	.467 ± .028	.488 ± .030	.507 ± .027	.459 ± .031	.449 ± .028	**.448 ± .027**	.485 ± .026	.490 ± .027	.487 ± .027	.471 ± .027
	.95	.494 ± .028	.445 ± .029	.483 ± .030	.498 ± .026	.527 ± .029	.444 ± .027	.468 ± .028	.475 ± .031	.498 ± .025	.532 ± .029	.443 ± .027	**.440 ± .028**	.476 ± .027	.478 ± .028	.492 ± .027	.459 ± .027
	.90	.474 ± .028	.435 ± .031	.461 ± .031	.479 ± .026	.503 ± .029	.431 ± .028	.443 ± .030	.456 ± .031	.447 ± .028	.500 ± .029	.431 ± .028	**.419 ± .029**	.459 ± .028	.469 ± .028	.495 ± .028	.455 ± .027
	.85	.463 ± .029	.424 ± .032	.458 ± .031	.457 ± .028	.489 ± .032	.427 ± .029	.427 ± .030	.438 ± .030	.436 ± .028	.491 ± .030	.445 ± .029	**.414 ± .029**	.440 ± .031	.459 ± .029	.495 ± .030	.435 ± .028
	.80	.442 ± .031	.425 ± .032	.444 ± .032	.456 ± .029	.456 ± .029	.415 ± .031	.415 ± .030	.430 ± .031	.444 ± .033	.429 ± .032	.452 ± .029	**.394 ± .030**	.408 ± .031	.459 ± .029	.494 ± .030	.419 ± .028
	.75	.416 ± .032	.404 ± .032	.427 ± .033	.467 ± .032	.466 ± .033	**.379 ± .033**	.404 ± .030	.415 ± .031	.412 ± .031	.503 ± .031	.448 ± .030	.385 ± .031	.405 ± .031	.445 ± .030	.494 ± .032	.412 ± .029
	.70	.388 ± .034	.380 ± .034	.406 ± .035	.449 ± .031	.416 ± .032	**.358 ± .034**	.385 ± .031	.385 ± .035	.414 ± .035	.415 ± .029	.449 ± .031	.361 ± .032	.401 ± .033	.431 ± .034	.490 ± .035	.393 ± .031
$\hat{\phi}$	.99	.981 ± .008	.974 ± .009	**.992 ± .005**	.959 ± .012	.959 ± .011	.988 ± .006	.993 ± .005	.994 ± .004	.984 ± .008	.975 ± .009	.978 ± .009	.987 ± .007	.994 ± .005	.986 ± .007	.975 ± .010	1.000 ± .000
	.95	.921 ± .015	.921 ± .014	**.953 ± .011**	.950 ± .013	.966 ± .010	.953 ± .013	.983 ± .007	**.951 ± .011**	.955 ± .013	.977 ± .009	.944 ± .014	.972 ± .009	.958 ± .012	.951 ± .012	.921 ± .015	.972 ± .010
	.90	.862 ± .020	.877 ± .019	**.898 ± .018**	.934 ± .014	.931 ± .017	.919 ± .018	.910 ± .015	.912 ± .015	.849 ± .018	.889 ± .016	.913 ± .016	.937 ± .013	.903 ± .018	.936 ± .014	.860 ± .019	.940 ± .015
	.85	.835 ± .022	.821 ± .022	.888 ± .019	.871 ± .020	.818 ± .022	.863 ± .019	**.857 ± .020**	.832 ± .022	**.859 ± .019**	.844 ± .022	.833 ± .022	.904 ± .016	.819 ± .023	.860 ± .019	.825 ± .020	.907 ± .018
	.80	.780 ± .023	.800 ± .022	.856 ± .021	.833 ± .021	.779 ± .024	.836 ± .021	.822 ± .020	.794 ± .021	.773 ± .023	.753 ± .027	.787 ± .025	.817 ± .021	.745 ± .026	.840 ± .020	**.804 ± .020**	.866 ± .021
	.75	.724 ± .027	.773 ± .023	.815 ± .022	.775 ± .024	.732 ± .024	.762 ± .024	.767 ± .022	**.749 ± .023**	.682 ± .025	.755 ± .025	.758 ± .027	.780 ± .024	.726 ± .026	.803 ± .023	.736 ± .023	.812 ± .023
	.70	.685 ± .028	.706 ± .027	.717 ± .022	.718 ± .024	.730 ± .026	.702 ± .026	.742 ± .022	.669 ± .025	.669 ± .025	.683 ± .023	.717 ± .027	.751 ± .025	.696 ± .027	.720 ± .024	**.690 ± .024**	.731 ± .029

Table C.50: Results for `rl`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.499 ± .016	.289 ± .015	.271 ± .014	.260 ± .014	.295 ± .015	.261 ± .015	.292 ± .015	.273 ± .014	.294 ± .015	.244 ± .016	**.236 ± .015**	.266 ± .014	.331 ± .016	.334 ± .015	.276 ± .016	.274 ± .016	.263 ± .015
	.95	.497 ± .016	.287 ± .015	.264 ± .015	.236 ± .015	.283 ± .015	.254 ± .015	.285 ± .016	.266 ± .015	.245 ± .016	.285 ± .016	.241 ± .016	**.225 ± .016**	.261 ± .014	.329 ± .015	.335 ± .015	.263 ± .016	.266 ± .016	.256 ± .015
	.90	.499 ± .018	.274 ± .015	.259 ± .015	.264 ± .016	.276 ± .016	.246 ± .016	.276 ± .017	.260 ± .015	.257 ± .017	.272 ± .017	.229 ± .016	**.216 ± .016**	.259 ± .015	.325 ± .015	.330 ± .015	.245 ± .016	.252 ± .017	.244 ± .016
	.85	.499 ± .019	.270 ± .016	.248 ± .016	.269 ± .015	.280 ± .018	.229 ± .016	.267 ± .017	.248 ± .015	.250 ± .015	.265 ± .016	.234 ± .016	**.209 ± .016**	.246 ± .015	.325 ± .016	.324 ± .016	.236 ± .016	.229 ± .017	.233 ± .016
	.80	.506 ± .020	.266 ± .016	.237 ± .017	.231 ± .016	.253 ± .017	.217 ± .016	.258 ± .017	.235 ± .016	.229 ± .017	.249 ± .017	.238 ± .017	**.204 ± .017**	.233 ± .015	.323 ± .016	.321 ± .016	.215 ± .016	.221 ± .018	.230 ± .017
	.75	.513 ± .020	.262 ± .016	.213 ± .016	.245 ± .016	.252 ± .018	.198 ± .016	.249 ± .018	.224 ± .016	.252 ± .017	.236 ± .017	**.192 ± .016**	.222 ± .016	.319 ± .016	.302 ± .016	.302 ± .016	.200 ± .017	.211 ± .018	.223 ± .017
	.70	.510 ± .021	.256 ± .017	.207 ± .017	.224 ± .018	.241 ± .018	.195 ± .017	.231 ± .019	.202 ± .017	.219 ± .017	.233 ± .017	.227 ± .017	**.175 ± .016**	.215 ± .015	.319 ± .017	.293 ± .016	.188 ± .018	.193 ± .017	.205 ± .017
$\hat{\phi}$	.99	.992 ± .003	.991 ± .003	.986 ± .003	.992 ± .003	.982 ± .004	.984 ± .003	.998 ± .002	.992 ± .003	.994 ± .003	.981 ± .004	**.990 ± .003**	.971 ± .005	.980 ± .005	.986 ± .004	.989 ± .004	.979 ± .005	.987 ± .003	.994 ± .002
	.95	.974 ± .005	.942 ± .007	.939 ± .008	.942 ± .008	.950 ± .007	.943 ± .007	.967 ± .006	**.950 ± .007**	.973 ± .005	.943 ± .007	.967 ± .005	.927 ± .008	.941 ± .008	.955 ± .007	.968 ± .005	.905 ± .008	.919 ± .009	.943 ± .008
	.90	.909 ± .009	**.901 ± .009**	.896 ± .010	.898 ± .010	.902 ± .010	.902 ± .008	.897 ± .010	.897 ± .010	.879 ± .010	.881 ± .010	.914 ± .009	.885 ± .009	.878 ± .010	.926 ± .009	.928 ± .007	.826 ± .012	.841 ± .012	.878 ± .011
	.85	.871 ± .010	.843 ± .011	.842 ± .012	.858 ± .014	.845 ± .011	.846 ± .011	.853 ± .012	.839 ± .011	.831 ± .013	**.847 ± .010**	.854 ± .012	.848 ± .011	.840 ± .012	.880 ± .011	.869 ± .010	.765 ± .013	.764 ± .013	.821 ± .012
	.80	.822 ± .012	.812 ± .012	.787 ± .014	.814 ± .011	.802 ± .014	.794 ± .013	**.803 ± .012**	.777 ± .014	.815 ± .012	.801 ± .013	.804 ± .014	.786 ± .013	.788 ± .014	.834 ± .011	.830 ± .011	.703 ± .014	.704 ± .014	**.800 ± .013**
	.75	.785 ± .013	.779 ± .014	.716 ± .015	.781 ± .012	.753 ± .014	.739 ± .014	**.752 ± .013**	.730 ± .015	.744 ± .015	.739 ± .014	.770 ± .013	.739 ± .014	.746 ± .013	.753 ± .013	.757 ± .013	.647 ± .015	.667 ± .016	.759 ± .013
	.70	.728 ± .015	.723 ± .015	.670 ± .016	.678 ± .016	.718 ± .015	**.701 ± .014**	.704 ± .014	.687 ± .016	.700 ± .015	.705 ± .015	.724 ± .014	.691 ± .015	.696 ± .015	.727 ± .013	.713 ± .014	.590 ± .017	.608 ± .017	.713 ± .015
MinCoeff	.99	**.999 ± .032**	1.004 ± .032	.999 ± .033	.997 ± .033	.999 ± .033	1.002 ± .033	1.003 ± .032	1.001 ± .033	1.001 ± .032	.998 ± .032	1.009 ± .033	1.005 ± .034	1.007 ± .033	.999 ± .033	1.004 ± .032	**.999 ± .032**	1.004 ± .033	1.005 ± .033
	.95	.994 ± .033	.998 ± .033	.996 ± .033	1.008 ± .034	.992 ± .034	1.003 ± .032	1.003 ± .033	1.007 ± .034	1.013 ± .033	1.002 ± .033	1.018 ± .034	1.008 ± .034	.999 ± .034	.997 ± .034	**.995 ± .032**	.980 ± .035	1.022 ± .034	1.017 ± .034
	.90	.998 ± .036	.991 ± .033	.992 ± .033	1.014 ± .033	.992 ± .035	**1.000 ± .033**	1.008 ± .035	1.000 ± .035	1.019 ± .034	1.021 ± .034	1.030 ± .036	1.020 ± .034	.995 ± .036	.998 ± .035	**.999 ± .033**	.982 ± .037	1.040 ± .036	1.031 ± .035
	.85	.998 ± .038	**.993 ± .033**	.990 ± .034	1.020 ± .036	.987 ± .036	1.009 ± .036	.992 ± .036	.994 ± .036	1.021 ± .035	1.044 ± .034	1.031 ± .035	1.019 ± .035	.993 ± .037	1.006 ± .035	.994 ± .035	1.004 ± .038	1.048 ± .039	1.040 ± .035
	.80	1.012 ± .039	**.999 ± .033**	.984 ± .038	1.014 ± .036	.990 ± .038	1.016 ± .037	.982 ± .037	1.002 ± .038	.993 ± .035	.989 ± .035	1.053 ± .036	1.040 ± .035	.975 ± .038	1.021 ± .035	.983 ± .037	1.001 ± .040	1.023 ± .040	1.038 ± .035
	.75	1.026 ± .040	**1.003 ± .034**	.952 ± .038	**1.005 ± .035**	.998 ± .038	1.028 ± .039	1.000 ± .038	.983 ± .038	1.002 ± .036	1.016 ± .037	1.062 ± .037	1.045 ± .035	.981 ± .038	1.012 ± .037	.974 ± .038	.997 ± .040	1.036 ± .040	1.040 ± .036
	.70	1.019 ± .042	**1.008 ± .036**	.979 ± .039	1.033 ± .039	.990 ± .039	1.027 ± .039	.986 ± .038	.985 ± .037	.985 ± .038	.971 ± .036	1.073 ± .038	1.045 ± .037	.992 ± .040	1.007 ± .037	.971 ± .040	.985 ± .042	1.062 ± .040	1.062 ± .038

Table C.51: Results for `stanfordcars`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.236 ± .008	.178 ± .007	.170 ± .006	.318 ± .009	.469 ± .008	.167 ± .006	.173 ± .007	.165 ± .006	.317 ± .008	.469 ± .008	**.107 ± .005**	.109 ± .006	.230 ± .008	.429 ± .009	.431 ± .009	.162 ± .007
	.95	.223 ± .008	.165 ± .007	.150 ± .006	.285 ± .007	.442 ± .009	.145 ± .006	.160 ± .007	.143 ± .006	.263 ± .008	.430 ± .009	.091 ± .005	**.086 ± .005**	.224 ± .008	.416 ± .009	.433 ± .009	.141 ± .007
	.90	.204 ± .008	.146 ± .007	.123 ± .006	.281 ± .008	.433 ± .008	.116 ± .005	.131 ± .006	.115 ± .006	.245 ± .007	.411 ± .008	.075 ± .005	**.064 ± .004**	.211 ± .008	.405 ± .009	.437 ± .010	.123 ± .006
	.85	.190 ± .008	.127 ± .006	.109 ± .006	.276 ± .009	.378 ± .010	.096 ± .005	.114 ± .006	.097 ± .005	.224 ± .008	.349 ± .010	.067 ± .004	**.048 ± .004**	.199 ± .008	.395 ± .010	.433 ± .010	.100 ± .006
	.80	.176 ± .008	.111 ± .006	.098 ± .006	.274 ± .008	.393 ± .010	.075 ± .005	.099 ± .006	.078 ± .005	.208 ± .008	.372 ± .010	.060 ± .005	**.039 ± .004**	.182 ± .008	.382 ± .010	.437 ± .010	.085 ± .006
	.75	.157 ± .008	.099 ± .006	.089 ± .006	.268 ± .009	.377 ± .008	.058 ± .004	.081 ± .006	.061 ± .004	.174 ± .008	.340 ± .009	.056 ± .005	**.028 ± .004**	.173 ± .008	.373 ± .010	.435 ± .010	.067 ± .005
	.70	.148 ± .008	.081 ± .006	.072 ± .005	.256 ± .009	.316 ± .009	.047 ± .004	.068 ± .005	.046 ± .004	.166 ± .008	.277 ± .010	.050 ± .004	**.022 ± .003**	.164 ± .008	.362 ± .011	.438 ± .011	.056 ± .005
$\hat{\phi}$	.99	.988 ± .002	.994 ± .001	.994 ± .001	.984 ± .002	.989 ± .002	**.989 ± .002**	.987 ± .002	.988 ± .002	.993 ± .001	.991 ± .002	**.990 ± .002**	.991 ± .002	.986 ± .002	.992 ± .002	.987 ± .002	.993 ± .001
	.95	.958 ± .003	.966 ± .003	.955 ± .004	.958 ± .004	**.948 ± .003**	.954 ± .003	.966 ± .003	**.952 ± .003**	.954 ± .003	**.949 ± .003**	.958 ± .004	.954 ± .004	.956 ± .004	.954 ± .003	.952 ± .004	.960 ± .003
	.90	.910 ± .005	.925 ± .004	.907 ± .005	.886 ± .006	**.900 ± .005**	.896 ± .005	.917 ± .004	.904 ± .004	.911 ± .004	**.899 ± .005**	.903 ± .005	**.901 ± .005**	.919 ± .005	.911 ± .005	.906 ± .005	.922 ± .004
	.85	.868 ± .006	.875 ± .005	.861 ± .006	.840 ± .007	.853 ± .006	.844 ± .005	.866 ± .006	.857 ± .005	.855 ± .006	.843 ± .006	.855 ± .006	**.852 ± .006**	.866 ± .005	.862 ± .006	.867 ± .006	.875 ± .005
	.80	.815 ± .006	.833 ± .006	.814 ± .006	.804 ± .007	**.802 ± .007**	.792 ± .006	.817 ± .007	**.804 ± .006**	.815 ± .006	.805 ± .007	**.804 ± .006**	.806 ± .007	.813 ± .006	.818 ± .006	.810 ± .007	.833 ± .006
	.75	.761 ± .007	.789 ± .007	.775 ± .007	.761 ± .008	.757 ± .007	**.748 ± .007**	.776 ± .008	.755 ± .007	.764 ± .008	.761 ± .008	.756 ± .007	**.751 ± .007**	.768 ± .007	.766 ± .006	.756 ± .008	.790 ± .007
	.70	.715 ± .007	.733 ± .008	.716 ± .007	.712 ± .007	.713 ± .007	.708 ± .007	.730 ± .008	.707 ± .007	.716 ± .008	.715 ± .008	**.706 ± .007**	**.706 ± .007**	.727 ± .007	.719 ± .007	.715 ± .008	.748 ± .007

Table C.52: Results for `SVHN`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.038 ± .002	**.030 ± .001**	.037 ± .001	**.030 ± .001**	.043 ± .002	.039 ± .001	**.030 ± .001**	.036 ± .001	.031 ± .001	.043 ± .002	.036 ± .001	.034 ± .001	.036 ± .001	.050 ± .002	.050 ± .002	.038 ± .002
	.95	.022 ± .001	.017 ± .001	.020 ± .001	**.016 ± .001**	.026 ± .001	.021 ± .001	.017 ± .001	.020 ± .001	**.016 ± .001**	.024 ± .001	.021 ± .001	.018 ± .001	.019 ± .001	.049 ± .002	.050 ± .002	.021 ± .001
	.90	.013 ± .001	**.008 ± .001**	.011 ± .001	.009 ± .001	.014 ± .001	.011 ± .001	**.008 ± .001**	.010 ± .001	.009 ± .001	.013 ± .001	.011 ± .001	.010 ± .001	.009 ± .001	.048 ± .002	.050 ± .002	.011 ± .001
	.85	.012 ± .001	**.006 ± .001**	.007 ± .001	**.006 ± .001**	.009 ± .001	.007 ± .001	**.006 ± .001**	.008 ± .001	**.006 ± .001**	.009 ± .001	.007 ± .001	.007 ± .001	.007 ± .001	.045 ± .002	.050 ± .002	.008 ± .001
	.80	.011 ± .001	**.005 ± .001**	.006 ± .001	**.005 ± .001**	.008 ± .001	.006 ± .001	**.005 ± .001**	.006 ± .001	**.005 ± .001**	.007 ± .001	.006 ± .001	**.005 ± .001**	.006 ± .001	.042 ± .002	.050 ± .002	.006 ± .001
	.75	.011 ± .001	**.004 ± .001**	.005 ± .001	.005 ± .001	.005 ± .001	.005 ± .001	.005 ± .001	.005 ± .001	.005 ± .001	.005 ± .001	.005 ± .001	.005 ± .001	.006 ± .001	.037 ± .002	.051 ± .002	.005 ± .001
	.70	.010 ± .001	**.004 ± .001**	.005 ± .001	.005 ± .001	**.004 ± .001**	**.004 ± .001**	**.004 ± .001**	.005 ± .001	.005 ± .001	**.004 ± .001**	**.004 ± .001**	**.004 ± .001**	.005 ± .001	.035 ± .002	.051 ± .002	**.004 ± .001**
$\hat{\phi}$	.99	.991 ± .001	**.990 ± .001**	.991 ± .001	**.990 ± .001**	.992 ± .001	.991 ± .001	.991 ± .001	.992 ± .001	.991 ± .001	.992 ± .001	.990 ± .001	.991 ± .001	.993 ± .001	**.990 ± .001**	.991 ± .001	.991 ± .001
	.95	.953 ± .002	.959 ± .001	.952 ± .002	.954 ± .002	.955 ± .001	.952 ± .002	.960 ± .001	.953 ± .002	.954 ± .001	.952 ± .001	.953 ± .002	.953 ± .002	.955 ± .001	.947 ± .002	**.951 ± .002**	.952 ± .001
	.90	.903 ± .002	.905 ± .002	.906 ± .002	.904 ± .002	.905 ± .002	.905 ± .002	.906 ± .002	.905 ± .002	.906 ± .002	.906 ± .002	.904 ± .002	.904 ± .002	.904 ± .002	**.900 ± .002**	.897 ± .002	.909 ± .002
	.85	.855 ± .003	.855 ± .003	.855 ± .002	.858 ± .003	.858 ± .002	.855 ± .003	.858 ± .003	**.853 ± .002**	.855 ± .003	.857 ± .002	.856 ± .002	.856 ± .002	.856 ± .002	.847 ± .003	**.852 ± .003**	.865 ± .003
	.80	**.799 ± .003**	.807 ± .003	.805 ± .003	.832 ± .003	.808 ± .003	.807 ± .003	.809 ± .003	.802 ± .003	.808 ± .003	.814 ± .003	.808 ± .003	.807 ± .003	.802 ± .003	**.800 ± .003**	.804 ± .003	.821 ± .003
	.75	**.753 ± .003**	.758 ± .003	.755 ± .003	.807 ± .003	**.753 ± .003**	.758 ± .003	.755 ± .003	.756 ± .003	.759 ± .003	.755 ± .003	.760 ± .003	.761 ± .003	**.753 ± .003**	.746 ± .003	.754 ± .003	.776 ± .003
	.70	.703 ± .003	.705 ± .003	.709 ± .003	.712 ± .003	.713 ± .003	.709 ± .003	.707 ± .003	.711 ± .003	.712 ± .003	.710 ± .003	.707 ± .003	.708 ± .003	.706 ± .003	.695 ± .003	**.702 ± .003**	.738 ± .003

Table C.53: Results for `tissuemnist`: *mean* ± *std* for $\widehat{Err}$ and empirical coverage $\hat{\phi}$.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss
$\widehat{Err}$	.99	.331 ± .002	.308 ± .002	.310 ± .002	.310 ± .002	.357 ± .002	.319 ± .002	.309 ± .002	.309 ± .002	.309 ± .002	.356 ± .002	.297 ± .002	**.295 ± .002**	.303 ± .002	.370 ± .002	.386 ± .002	.314 ± .002
	.95	.319 ± .002	.295 ± .002	.296 ± .002	.296 ± .002	.347 ± .002	.302 ± .002	.293 ± .002	.294 ± .002	.295 ± .002	.344 ± .002	.286 ± .002	**.278 ± .002**	.296 ± .002	.359 ± .002	.386 ± .002	.298 ± .002
	.90	.305 ± .002	.278 ± .002	.278 ± .002	.278 ± .002	.334 ± .002	.283 ± .002	.273 ± .002	.274 ± .002	.273 ± .002	.340 ± .002	.274 ± .002	**.258 ± .002**	.287 ± .002	.347 ± .002	.387 ± .002	.280 ± .002
	.85	.292 ± .002	.262 ± .002	.261 ± .002	.261 ± .002	.321 ± .002	.265 ± .002	.256 ± .002	.257 ± .002	.255 ± .002	.343 ± .002	.260 ± .002	**.239 ± .002**	.277 ± .002	.333 ± .002	.388 ± .002	.262 ± .002
	.80	.279 ± .003	.245 ± .002	.243 ± .002	.239 ± .002	.302 ± .002	.249 ± .002	.238 ± .002	.239 ± .002	.235 ± .002	.335 ± .003	.247 ± .002	**.221 ± .002**	.265 ± .002	.320 ± .002	.388 ± .002	.246 ± .002
	.75	.268 ± .003	.228 ± .002	.226 ± .002	.223 ± .002	.288 ± .002	.230 ± .002	.222 ± .002	.223 ± .002	.217 ± .002	.327 ± .003	.232 ± .002	**.204 ± .002**	.256 ± .002	.306 ± .002	.391 ± .002	.227 ± .002
	.70	.256 ± .003	.211 ± .002	.208 ± .002	.210 ± .002	.270 ± .003	.212 ± .002	.203 ± .002	.204 ± .002	.202 ± .002	.317 ± .003	.218 ± .002	**.187 ± .002**	.244 ± .002	.291 ± .002	.391 ± .002	.208 ± .002
$\hat{\phi}$	.99	.991 ± .000	.988 ± .000	.991 ± .000	.990 ± .000	**.990 ± .000**	.989 ± .000	**.990 ± .000**	.991 ± .000	.991 ± .000	.989 ± .000	**.990 ± .000**	.988 ± .000	**.990 ± .000**	**.990 ± .000**	**.990 ± .000**	**.990 ± .000**
	.95	.952 ± .001	**.950 ± .001**	**.951 ± .001**	**.950 ± .001**	**.950 ± .001**	.946 ± .001	**.950 ± .001**	**.951 ± .001**	**.951 ± .001**	**.949 ± .001**	.948 ± .001	.945 ± .001	**.950 ± .001**	.948 ± .001	.952 ± .001	**.949 ± .001**
	.90	**.899 ± .001**	**.899 ± .001**	.899 ± .001	.903 ± .001	**.901 ± .001**	.895 ± .002	.899 ± .001	.898 ± .001	.898 ± .002	**.901 ± .001**	.894 ± .002	.893 ± .001	**.899 ± .001**	.896 ± .001	**.901 ± .002**	**.899 ± .001**
	.85	.848 ± .002	.849 ± .002	**.850 ± .002**	.853 ± .002	.848 ± .002	.846 ± .002	.848 ± .002	**.850 ± .002**	**.850 ± .002**	.847 ± .002	.843 ± .002	.842 ± .002	.851 ± .002	.845 ± .002	.855 ± .002	**.850 ± .002**
	.80	.795 ± .002	.797 ± .002	**.800 ± .002**	.798 ± .002	.796 ± .002	.797 ± .002	**.801 ± .002**	.802 ± .002	.798 ± .002	**.800 ± .002**	.796 ± .002	.793 ± .002	.802 ± .002	.797 ± .002	.805 ± .002	.801 ± .002
	.75	.747 ± .002	.748 ± .002	**.750 ± .002**	**.751 ± .002**	**.749 ± .002**	.745 ± .002	.753 ± .002	.753 ± .002	.754 ± .002	**.750 ± .002**	.746 ± .002	.747 ± .002	.752 ± .002	.748 ± .002	.753 ± .002	.753 ± .002
	.70	.698 ± .002	**.698 ± .002**	**.700 ± .002**	**.702 ± .002**	**.698 ± .002**	**.699 ± .002**	**.700 ± .002**	.705 ± .002	.704 ± .002	.697 ± .002	.698 ± .002	**.700 ± .002**	.702 ± .002	.695 ± .002	.703 ± .002	.702 ± .002

Table C.54: Results for `ucicredit`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.181 ± .004	.182 ± .005	.178 ± .004	.177 ± .004	.179 ± .004	.179 ± .005	.181 ± .004	.177 ± .004	**.176 ± .004**	.179 ± .004	.181 ± .004	.178 ± .005	.180 ± .004	.181 ± .005	.188 ± .004	.180 ± .005	.184 ± .005	.184 ± .005
	.95	.170 ± .004	.172 ± .005	**.166 ± .004**	**.166 ± .004**	.170 ± .004	.168 ± .005	.168 ± .004	.168 ± .005	.167 ± .004	.169 ± .004	.177 ± .004	.166 ± .005	.171 ± .004	.171 ± .005	.186 ± .005	.170 ± .004	.184 ± .005	.185 ± .005
	.90	.160 ± .004	.159 ± .004	.154 ± .004	**.153 ± .004**	.156 ± .004	**.153 ± .004**	.156 ± .004	.155 ± .004	**.153 ± .004**	.158 ± .004	.174 ± .005	.154 ± .005	.155 ± .005	.161 ± .004	.184 ± .005	.155 ± .004	.186 ± .005	.186 ± .005
	.85	.151 ± .004	.148 ± .005	.146 ± .004	.144 ± .004	**.143 ± .004**	.144 ± .004	.146 ± .004	.144 ± .004	.145 ± .004	.151 ± .004	.171 ± .005	.146 ± .004	.146 ± .004	.151 ± .005	.180 ± .005	.146 ± .004	.187 ± .005	.189 ± .005
	.80	.144 ± .004	.140 ± .005	.139 ± .004	.137 ± .004	.139 ± .004	.137 ± .004	.139 ± .004	.140 ± .004	.143 ± .005	.143 ± .005	.167 ± .005	.138 ± .004	**.134 ± .004**	.143 ± .005	.177 ± .005	.138 ± .004	.186 ± .005	.190 ± .005
	.75	.139 ± .005	**.128 ± .005**	.132 ± .005	.131 ± .004	.130 ± .005	.131 ± .004	.134 ± .004	.132 ± .005	.139 ± .005	.133 ± .004	.163 ± .005	.130 ± .005	.129 ± .005	.135 ± .005	.175 ± .006	.130 ± .004	.188 ± .005	.190 ± .005
	.70	.136 ± .005	.123 ± .004	.126 ± .005	.126 ± .005	.125 ± .005	.126 ± .005	**.122 ± .004**	.126 ± .005	.130 ± .005	.146 ± .005	.159 ± .005	.125 ± .005	.125 ± .005	.128 ± .005	.171 ± .005	.124 ± .005	.190 ± .006	.191 ± .006
$\hat{\phi}$	.99	.989 ± .001	**.991 ± .001**	**.990 ± .001**	**.989 ± .001**	.987 ± .001	.988 ± .001	**.991 ± .001**	.988 ± .001	.987 ± .001	**.989 ± .001**	**.990 ± .001**	.986 ± .002	.989 ± .001	.986 ± .002	**.989 ± .001**	.992 ± .001	**.990 ± .001**	**.989 ± .001**
	.95	**.951 ± .003**	.957 ± .002	.947 ± .003	**.951 ± .003**	.946 ± .003	.952 ± .003	.952 ± .003	.945 ± .003	**.950 ± .003**	.952 ± .003	.942 ± .003	.950 ± .003	.953 ± .003	.948 ± .003	.944 ± .003	.958 ± .003	.953 ± .003	.948 ± .003
	.90	.902 ± .004	**.898 ± .004**	.899 ± .004	.896 ± .004	.896 ± .004	.897 ± .004	**.899 ± .004**	.900 ± .004	.898 ± .004	.904 ± .004	**.901 ± .004**	.901 ± .004	.903 ± .004	.892 ± .004	.897 ± .004	.908 ± .004	.902 ± .004	.897 ± .004
	.85	.851 ± .005	.843 ± .005	.848 ± .005	.846 ± .005	.845 ± .004	.849 ± .005	.860 ± .005	.849 ± .005	.858 ± .005	.857 ± .005	**.849 ± .004**	.857 ± .005	.849 ± .005	.840 ± .005	.844 ± .005	.863 ± .005	.852 ± .004	.849 ± .005
	.80	.803 ± .005	.798 ± .005	.794 ± .006	**.800 ± .005**	**.799 ± .005**	.804 ± .005	.805 ± .005	.801 ± .005	.811 ± .005	.801 ± .005	.803 ± .005	.806 ± .005	.795 ± .006	.796 ± .005	.790 ± .005	.813 ± .005	.806 ± .005	.794 ± .006
	.75	.743 ± .006	.741 ± .006	.745 ± .006	.756 ± .006	**.752 ± .006**	.755 ± .006	.756 ± .006	.758 ± .006	.766 ± .006	**.750 ± .006**	.760 ± .005	.751 ± .006	.751 ± .006	**.752 ± .006**	.746 ± .006	.762 ± .006	.750 ± .006	.744 ± .006
	.70	**.701 ± .006**	.691 ± .007	**.699 ± .007**	.710 ± .006	**.698 ± .006**	.703 ± .007	.705 ± .006	.709 ± .007	.714 ± .006	**.701 ± .006**	.715 ± .006	.706 ± .007	.710 ± .007	.705 ± .006	.692 ± .006	**.701 ± .006**	.696 ± .007	.693 ± .007
MinCoeff	.99	.991 ± .021	.987 ± .021	.994 ± .021	.986 ± .021	.979 ± .021	.990 ± .021	.988 ± .021	.993 ± .021	.983 ± .022	.987 ± .021	**1.000 ± .021**	.993 ± .021	.987 ± .021	.979 ± .021	.989 ± .021	.993 ± .021	1.006 ± .021	1.006 ± .021
	.95	.943 ± .022	.948 ± .022	.941 ± .021	.940 ± .022	.922 ± .021	.953 ± .022	.951 ± .022	.937 ± .022	.942 ± .021	.942 ± .021	**.988 ± .022**	.953 ± .023	.947 ± .021	.916 ± .020	.980 ± .022	.958 ± .022	1.015 ± .021	1.021 ± .022
	.90	.893 ± .022	.881 ± .022	.871 ± .022	.871 ± .023	.843 ± .022	.891 ± .021	.899 ± .022	.893 ± .021	.872 ± .022	.891 ± .022	**.980 ± .022**	.897 ± .022	.898 ± .022	.816 ± .020	.966 ± .023	.908 ± .021	1.030 ± .022	1.034 ± .022
	.85	.813 ± .022	.805 ± .024	.773 ± .022	.763 ± .020	.754 ± .021	.839 ± .022	.856 ± .023	.834 ± .022	.819 ± .022	.847 ± .023	.950 ± .023	.859 ± .022	.811 ± .022	.748 ± .021	.946 ± .023	.859 ± .022	**1.046 ± .023**	1.058 ± .022
	.80	.748 ± .020	.743 ± .023	.693 ± .022	.620 ± .019	.624 ± .019	.771 ± .021	.786 ± .024	.774 ± .022	.842 ± .024	.802 ± .024	.924 ± .023	.781 ± .021	.706 ± .021	.702 ± .021	.926 ± .025	.796 ± .023	**1.057 ± .024**	1.079 ± .024
	.75	.701 ± .021	.667 ± .023	.631 ± .021	.592 ± .020	.589 ± .021	.696 ± .022	.736 ± .023	.709 ± .023	.824 ± .026	.661 ± .022	.909 ± .023	.688 ± .022	.618 ± .021	.644 ± .021	**.924 ± .026**	.719 ± .022	1.081 ± .025	1.090 ± .025
	.70	.682 ± .022	.617 ± .022	.588 ± .021	.568 ± .021	.564 ± .021	.629 ± .022	.664 ± .024	.637 ± .021	.787 ± .025	.843 ± .027	.873 ± .024	.636 ± .021	.581 ± .022	.600 ± .021	**.904 ± .027**	.645 ± .021	1.109 ± .026	1.113 ± .025

Table C.55: Results for `upselling`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.277 ± .014	.168 ± .013	.171 ± .014	.164 ± .013	.178 ± .014	.186 ± .015	.169 ± .014	.171 ± .014	**.161 ± .013**	.179 ± .013	.189 ± .013	.186 ± .013	.177 ± .012	.185 ± .014	.185 ± .013	.177 ± .012	.185 ± .012	.191 ± .015
	.95	.278 ± .014	.160 ± .013	.163 ± .013	.162 ± .013	.171 ± .014	.174 ± .015	.163 ± .013	.159 ± .013	**.152 ± .013**	.171 ± .013	.195 ± .014	.168 ± .013	.166 ± .012	.179 ± .014	.178 ± .013	.177 ± .012	.185 ± .012	.183 ± .015
	.90	.276 ± .015	.150 ± .014	.150 ± .013	.145 ± .012	.155 ± .013	.163 ± .014	**.142 ± .013**	**.142 ± .013**	**.142 ± .013**	.156 ± .013	.208 ± .014	.156 ± .013	.148 ± .012	.164 ± .014	.172 ± .014	.177 ± .012	.185 ± .012	.164 ± .015
	.85	.275 ± .015	.136 ± .013	.134 ± .013	**.122 ± .012**	.139 ± .013	.145 ± .014	.126 ± .013	.129 ± .012	.128 ± .013	.137 ± .013	.221 ± .015	.142 ± .013	.138 ± .012	.162 ± .014	.168 ± .014	.177 ± .012	.185 ± .012	.157 ± .015
	.80	.291 ± .016	**.104 ± .012**	.121 ± .013	.115 ± .012	.121 ± .012	.130 ± .013	.118 ± .012	.116 ± .012	.110 ± .013	.124 ± .012	.230 ± .016	.118 ± .011	.114 ± .012	.144 ± .014	.168 ± .015	.176 ± .012	.185 ± .012	.147 ± .014
	.75	.297 ± .016	.094 ± .011	.110 ± .012	.104 ± .012	.102 ± .011	.106 ± .012	**.092 ± .012**	.103 ± .012	.101 ± .012	.105 ± .012	.244 ± .016	.095 ± .011	.098 ± .012	.128 ± .014	.164 ± .014	.171 ± .012	.185 ± .012	.123 ± .013
	.70	.304 ± .017	.078 ± .011	.081 ± .010	.075 ± .010	.080 ± .010	.080 ± .011	.080 ± .011	.086 ± .011	.077 ± .011	.077 ± .010	.257 ± .017	.076 ± .010	**.074 ± .010**	.120 ± .013	.158 ± .014	.171 ± .012	.185 ± .012	.108 ± .013
$\hat{\phi}$	.99	.994 ± .002	.988 ± .004	.985 ± .004	.993 ± .003	.984 ± .004	.989 ± .004	.988 ± .004	.995 ± .002	.984 ± .004	.994 ± .002	.992 ± .003	**.990 ± .003**	.993 ± .003	.990 ± .003	**.991 ± .003**	.978 ± .005	.998 ± .001	.996 ± .002
	.95	.948 ± .007	.950 ± .007	**.951 ± .007**	.964 ± .006	.950 ± .008	.951 ± .008	.965 ± .006	.953 ± .007	.940 ± .009	.944 ± .007	.961 ± .006	.941 ± .007	.953 ± .006	.962 ± .006	.939 ± .007	.978 ± .005	.998 ± .001	.955 ± .006
	.90	.888 ± .010	.906 ± .010	.883 ± .011	.915 ± .010	.910 ± .009	.923 ± .010	.884 ± .012	.887 ± .011	.902 ± .011	.912 ± .010	**.900 ± .010**	.895 ± .010	.913 ± .010	.907 ± .008	.881 ± .009	.978 ± .005	.998 ± .001	.890 ± .010
	.85	.833 ± .012	.860 ± .012	.842 ± .012	.841 ± .013	.863 ± .012	.879 ± .011	.841 ± .013	.849 ± .013	.851 ± .013	.848 ± .012	.847 ± .011	.858 ± .012	.867 ± .012	.881 ± .009	**.852 ± .010**	.978 ± .005	.998 ± .001	.872 ± .011
	.80	.759 ± .014	.808 ± .014	.800 ± .013	.805 ± .013	.801 ± .014	.833 ± .012	.803 ± .014	.807 ± .014	.795 ± .015	.816 ± .013	.810 ± .012	.809 ± .013	.822 ± .014	.823 ± .011	**.798 ± .012**	.977 ± .005	.998 ± .001	.825 ± .012
	.75	.718 ± .015	.764 ± .014	.758 ± .014	.772 ± .014	.758 ± .014	.786 ± .014	.758 ± .015	.768 ± .014	.769 ± .015	.760 ± .014	.760 ± .012	.768 ± .015	.772 ± .015	.762 ± .014	**.754 ± .013**	.964 ± .006	.998 ± .001	.772 ± .013
	.70	.677 ± .015	.724 ± .015	**.702 ± .015**	.722 ± .015	.706 ± .016	.725 ± .015	.721 ± .016	.739 ± .015	.722 ± .016	.724 ± .015	.722 ± .014	.709 ± .015	.723 ± .015	.730 ± .014	.717 ± .012	.964 ± .006	.998 ± .001	.721 ± .014
MinCoeff	.99	1.000 ± .031	1.003 ± .031	1.004 ± .031	1.002 ± .031	1.001 ± .031	1.001 ± .030	1.005 ± .030	1.002 ± .030	**1.000 ± .031**	.999 ± .030	1.009 ± .031	1.001 ± .031	1.003 ± .030	1.001 ± .031	.999 ± .031	1.003 ± .031	1.001 ± .031	1.001 ± .031
	.95	.977 ± .032	**1.000 ± .031**	1.004 ± .033	**1.001 ± .031**	**1.001 ± .031**	.995 ± .031	1.005 ± .031	1.016 ± .030	1.004 ± .031	1.000 ± .031	.985 ± .032	1.011 ± .031	**.997 ± .031**	**1.005 ± .031**	1.011 ± .032	1.003 ± .031	**1.001 ± .031**	1.013 ± .031
	.90	.950 ± .033	.998 ± .031	1.017 ± .034	.994 ± .032	1.004 ± .031	.997 ± .032	.993 ± .032	1.020 ± .031	**1.001 ± .032**	1.007 ± .031	.937 ± .034	1.012 ± .031	.996 ± .031	1.027 ± .032	1.007 ± .032	1.003 ± .031	1.001 ± .031	1.021 ± .032
	.85	.921 ± .034	.998 ± .031	1.018 ± .035	.991 ± .033	1.005 ± .033	1.004 ± .033	**.996 ± .032**	1.024 ± .031	.992 ± .032	1.018 ± .032	.885 ± .035	1.006 ± .031	1.005 ± .030	1.035 ± .032	.998 ± .033	1.003 ± .031	1.001 ± .031	1.022 ± .032
	.80	.840 ± .034	1.008 ± .032	1.022 ± .034	1.007 ± .032	1.011 ± .032	1.007 ± .033	.997 ± .032	1.026 ± .032	1.009 ± .033	1.004 ± .033	.839 ± .035	1.012 ± .032	.997 ± .032	1.048 ± .034	.984 ± .033	1.002 ± .031	**1.001 ± .031**	1.041 ± .033
	.75	.799 ± .034	1.018 ± .034	1.021 ± .036	1.019 ± .033	1.013 ± .033	1.019 ± .034	1.013 ± .033	1.015 ± .033	1.016 ± .034	1.016 ± .033	.781 ± .035	1.012 ± .033	1.015 ± .033	1.077 ± .034	.976 ± .034	1.000 ± .032	**1.001 ± .031**	1.050 ± .035
	.70	.746 ± .036	1.031 ± .035	1.039 ± .037	1.026 ± .034	1.042 ± .035	1.029 ± .035	1.020 ± .035	1.025 ± .034	1.017 ± .035	1.022 ± .033	.781 ± .036	1.022 ± .036	1.024 ± .034	1.095 ± .036	.965 ± .035	1.000 ± .032	**1.001 ± .031**	1.066 ± .037

Table C.56: Results for `waterbirds`: *mean* ± *std* for $\widehat{Err}$, empirical coverage $\hat{\phi}$, and *MinCoeff*.

Metric	c	DeepGamblers	SAT	SAT+EM	SelNet	SelNet+EM	SR	SAT+SR	SAT+EM+SR	SelNet+SR	SelNet+EM+SR	Ensemble	Ensemble+SR	ConfidNet	SELE	REG	SCRoss	AUCross	PlugInAUC
$\widehat{Err}$	.99	.143 ± .008	.093 ± .006	.109 ± .007	.114 ± .007	.120 ± .007	.094 ± .006	.094 ± .006	.110 ± .007	.115 ± .007	.117 ± .007	**.083 ± .006**	**.083 ± .006**	.101 ± .006	.139 ± .007	.144 ± .007	.102 ± .006	.108 ± .006	.099 ± .007
	.95	.142 ± .008	.078 ± .006	.094 ± .007	.102 ± .006	.113 ± .007	.078 ± .006	.080 ± .006	.098 ± .007	.102 ± .007	.119 ± .007	.075 ± .006	**.064 ± .006**	.093 ± .006	.137 ± .007	.141 ± .007	.084 ± .005	.110 ± .006	.097 ± .007
	.90	.134 ± .008	.064 ± .005	.085 ± .007	.101 ± .007	.087 ± .007	.068 ± .006	.062 ± .005	.073 ± .006	.095 ± .007	.091 ± .007	.063 ± .005	**.049 ± .005**	.081 ± .006	.133 ± .007	.141 ± .007	.065 ± .005	.115 ± .006	.098 ± .007
	.85	.112 ± .008	.051 ± .005	.073 ± .006	.086 ± .006	.083 ± .006	.056 ± .005	.049 ± .005	.064 ± .006	.090 ± .006	.076 ± .006	.052 ± .005	**.039 ± .005**	.064 ± .005	.127 ± .007	.139 ± .007	.048 ± .005	.119 ± .007	.096 ± .007
	.80	.089 ± .007	.043 ± .005	.056 ± .005	.098 ± .007	.059 ± .007	.047 ± .006	.040 ± .005	.053 ± .006	.069 ± .006	.063 ± .006	.041 ± .005	**.031 ± .004**	.055 ± .005	.126 ± .008	.139 ± .008	.035 ± .004	.127 ± .007	.094 ± .007
	.75	.072 ± .007	.033 ± .004	.044 ± .005	.082 ± .006	.080 ± .007	.040 ± .005	.033 ± .004	.041 ± .005	.065 ± .005	.055 ± .006	.034 ± .005	**.023 ± .004**	.049 ± .005	.124 ± .008	.141 ± .008	.029 ± .004	.130 ± .008	.088 ± .007
	.70	.058 ± .006	.027 ± .004	.042 ± .005	.079 ± .007	.063 ± .006	.034 ± .005	.027 ± .004	.040 ± .005	.075 ± .006	.049 ± .005	.028 ± .004	.018 ± .003	.038 ± .004	.119 ± .008	.141 ± .008	**.017 ± .004**	.135 ± .008	.083 ± .006
$\hat{\phi}$	.99	**.989 ± .002**	**.991 ± .002**	.986 ± .003	.981 ± .003	**.990 ± .002**	**.991 ± .002**	.994 ± .001	.987 ± .003	**.989 ± .002**	.985 ± .002	**.992 ± .002**	**.990 ± .002**	**.990 ± .002**	.973 ± .004	.995 ± .001	.988 ± .002	.987 ± .002	.994 ± .002
	.95	**.951 ± .005**	.946 ± .005	.936 ± .006	.938 ± .005	.943 ± .005	.943 ± .005	.959 ± .005	.936 ± .006	.927 ± .006	.943 ± .005	**.949 ± .005**	**.950 ± .004**	.945 ± .005	.937 ± .005	.956 ± .005	.932 ± .005	.939 ± .005	.940 ± .006
	.90	.904 ± .006	.897 ± .007	.884 ± .008	.899 ± .007	.913 ± .006	.906 ± .006	.906 ± .006	.879 ± .007	.899 ± .006	.912 ± .007	**.905 ± .006**	**.900 ± .007**	.893 ± .007	.878 ± .007	.896 ± .005	.857 ± .007	.886 ± .007	.894 ± .007
	.85	**.849 ± .007**	.841 ± .009	.845 ± .009	.842 ± .009	.835 ± .009	.855 ± .008	.844 ± .008	.826 ± .008	.842 ± .009	.831 ± .009	.856 ± .007	.849 ± .009	.845 ± .008	.812 ± .008	.854 ± .007	.775 ± .008	.821 ± .007	.847 ± .008
	.80	**.801 ± .009**	.785 ± .010	.776 ± .009	**.799 ± .009**	.803 ± .009	.804 ± .009	.775 ± .010	.779 ± .009	.790 ± .009	**.801 ± .009**	.796 ± .009	**.800 ± .009**	.808 ± .008	.761 ± .009	.807 ± .008	.695 ± .009	.752 ± .009	.807 ± .008
	.75	**.749 ± .010**	.716 ± .010	.723 ± .009	.756 ± .010	.757 ± .009	.754 ± .010	.717 ± .010	.729 ± .010	.745 ± .010	.733 ± .010	**.746 ± .009**	.750 ± .010	.756 ± .010	.710 ± .010	.759 ± .009	.640 ± .009	.685 ± .009	.760 ± .009
	.70	.705 ± .010	.677 ± .011	.684 ± .010	.693 ± .011	.710 ± .010	.699 ± .011	.671 ± .011	.689 ± .010	.697 ± .009	.717 ± .010	**.698 ± .009**	.699 ± .011	.690 ± .010	.664 ± .010	.707 ± .009	.573 ± .009	.626 ± .010	.701 ± .010
MinCoeff	.99	.968 ± .037	.987 ± .039	.987 ± .039	.988 ± .037	.980 ± .037	.991 ± .038	.993 ± .038	.982 ± .037	**.997 ± .038**	.980 ± .037	**.995 ± .037**	.988 ± .038	1.003 ± .038	.978 ± .038	1.009 ± .038	1.004 ± .038	1.017 ± .038	1.012 ± .038
	.95	.840 ± .037	.919 ± .039	.932 ± .039	.918 ± .037	.913 ± .037	.925 ± .039	.938 ± .039	.913 ± .039	.901 ± .038	.886 ± .036	.926 ± .037	.925 ± .039	.952 ± .039	.944 ± .038	1.001 ± .039	**.994 ± .038**	1.055 ± .040	1.034 ± .040
	.90	.683 ± .036	.835 ± .036	.874 ± .040	.900 ± .038	.871 ± .035	.878 ± .039	.854 ± .040	.832 ± .038	.873 ± .037	.871 ± .037	.846 ± .038	.853 ± .037	.876 ± .038	.870 ± .037	**1.006 ± .040**	1.009 ± .041	1.108 ± .041	1.068 ± .042
	.85	.496 ± .034	.763 ± .036	.829 ± .039	.664 ± .039	.

Appendix D

Other Publications

During the PhD studies, I was involved in some projects whose outputs were out of the scope of this thesis. They resulted in the the following publications:

- Ana Rita Nogueira., Andrea Pugnana, Salvatore Ruggieri, Dino Pedreschi, Joao Gama. Methods and tools for causal discovery and causal inference. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2023.
- Salvatore Ruggieri, José M. Álvarez, Andrea Pugnana, Laura State, and Franco Turini. Can we trust Fair-AI? In *AAAI*, pages 1542115430. AAAI Press, 2023.
- Salvatore Lusito, Andrea Pugnana, Riccardo Guidotti. Solving imbalanced learning with outlier detection and feature reduction. *Machine Learning*, to appear, 2024.