# Proceedings of the 5th International Workshop on Learning to Quantify (LQ 2025)

Mirko Bunse, Pablo González,
Alejandro Moreo, and Fabrizio Sebastiani (eds.)

# Preface

The 5th International Workshop on Learning to Quantify (LQ 2025 – `https://lq-2025.github.io/`) has been held in Porto, PT, on September 15, 2025, as a satellite workshop of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2025). While the 1st edition of the workshop (LQ 2021 – `https://cikmlq2021.github.io/`) had to be an entirely online event due to the COVID-19 pandemic, the 2nd edition (LQ 2022 – `https://lq-2022.github.io/`), 3rd edition (LQ 2023 – `https://lq-2023.github.io/`), 4th edition (LQ 2024 – `https://lq-2024.github.io/`), and this 5th edition, have been hybrid events, with presentations given in-presence, and both in-presence attendees and remote attendees.

The LQ 2025 workshop consisted of the presentations of seven contributed papers, that had each gone through a rigorous peer-reviewing process by three reviewers each, and a final collective discussion on the open problems of learning to quantify and on future initiatives. The present volume contains the text of five of the seven presentations given at the workshop (for the other two presentations the authors asked for their papers not to be in the proceedings). We hope that the availability of the present volume will increase the interest in the subject of quantification on the part of researchers and practitioners alike, and will contribute to making quantification better known to potential users of this technology and to researchers interested in advancing the field.

Mirko Bunse
Pablo González
Alejandro Moreo
Fabrizio Sebastiani

# Table Of Contents

# LQ 2025 Program Committee

Mirko Bunse, University of Dortmund, DE (co-Chair)
Pablo González, University of Oviedo, ES (co-Chair)
Alejandro Moreo, Consiglio Nazionale delle Ricerche, IT (co-Chair)
Fabrizio Sebastiani, Consiglio Nazionale delle Ricerche, IT (co-Chair)

Gustavo Batista, University of New South Wales, AU
Juan José del Coz, University of Oviedo, ES
Andrea Esuli, Consiglio Nazionale delle Ricerche, IT
Cèsar Ferri, Universitat Politècnica de València, ES
Wei Gao, Singapore Management University, SG
Rafael Izbicki, Federal University of São Carlos, BR
André G. Maletzke, Universidade Estadual do Oeste do Paraná, BR
Tobias Schumacher, University of Mannheim, DE
Marco Saerens, Catholic University of Louvain, BE
Dirk Tasche, Swiss Financial Market Supervisory Authority, CH
Pawel Piotr Czyż, ETH AI Center, CH
Zahra Donyavi, University of New South Wales, AU

# Acknowledgments

Pablo Gonzalez's work has been funded by Agencia de Ciencia, Competitividad Empresarial e Innovación Asturiana (Sekuens) within the program "Grants for research groups from organizations in the Principado de Asturias, 2024" (GRU-GIC-24-018). Alejandro Moreo's and Fabrizio Sebastiani's work was partially supported by project "Future Artificial Intelligence Research" (FAIR), project "Quantification under Dataset Shift" (QuaDaSh), and project "Strengthening the Italian RI for Social Mining and Big Data Analytics" (SoBigData.it), all funded by the European Union under the NextGenerationEU funding scheme (CUP B53D22000980006, CUP B53D23026250001, CUP B53C22001760006, respectively). The organizers' opinions do not necessarily reflect those of the European Commission.

# Classify-and-Count with Prediction Sets: A Conformal Approach to Label Distribution Estimation

Evgueni Smirnov and Filip Schlembach

Department of Advanced Computing Sciences,
Maastricht University,
Maastricht, The Netherlands
smirnov@maastrichtuniversity.nl, filip.schlembach@maastrichtuniversity.nl

**Abstract.** We introduce *Mondrian Inductive Conformal Quantification* (MICQ), a set-based approach to learning to quantify that combines Mondrian conformal prediction with *probability class counting* (PCC). For each unlabeled instance $X$ from test bag $B$, a Mondrian conformal predictor first constructs a prediction set $\Gamma^{\alpha}(X)$ at miscoverage level $\alpha$, and then PCC distributes the instance's weights to prevalences of class labels in $\Gamma^{\alpha}(X)$ proportionally to their posteriors. The final class prevalence estimates are obtained by averaging these weights across all instances in test bag $B$. Due to the per-class validity provided by the Mondrian conformal predictor MICQ remains reliable under prior shift. Experiments on standard benchmarks with controlled prior shifts demonstrate that MICQ indeed improves over PCC.

**Keywords:** Conformal prediction, learning to quantify, prior shift, Mondrian inductive conformal classification, probability class counting.

## 1 Introduction

The problem of *learning to quantify* (L2Q) is to estimate the distribution of class labels in a given (test) bag $B$ of unlabeled instances [1]. Standard quantification methods, such as the *probability class counting* (PCC) approach, first use a trained classifier to estimate the posterior probabilities of class labels for the instances from $B$, and then compute their average across all instances. While simple, PCC can be sensitive to *prior shift* when the class distribution changes between training and testing while the class-conditional feature distributions remain stable.

To address this limitation, we propose a quantification method based on *conformal prediction*, a framework for set-valued classification that offers rigorous finite-sample guarantees [6]. Our approach, called *Mondrian Inductive Conformal Quantification* (MICQ), employs the class-conditional guarantees of Mondrian conformal predictors to provide prevalence estimates that remain reliable under prior shift.

MICQ can be viewed as a *conformal extension of PCC*. For each instance $X \in B$ and a probabilistic classifier used in PCC, the approach first constructs a conformal prediction set $\Gamma^{\alpha}(X)$ of plausible labels at a given miscoverage level $\alpha$. The contribution (weights) of $X$ to class prevalences is then distributed among the class labels in $\Gamma^{\alpha}(X)$ in proportion to their posterior probabilities, modulated by a *temperature* parameter $\tau$ that controls the weight's sharpness. The final class prevalence estimates are obtained by averaging these weights across all instances in the test bag $B$. By adjusting both the size of the prediction sets $\Gamma^{\alpha}(X)$ and the concentration of weights, MICQ allows regularization which has a potential to improve robustness and stability of quantification under prior shift.

We evaluate MICQ against the standard PCC baseline on three datasets under controlled prior shift using Dirichlet sampling. Our experiments show that MICQ consistently achieves lower absolute error than PCC across a range of conditions.

The rest of the paper is organized as follows. The problem of L2Q and (Mondrian) conformal classification are discussed in Section 2. Section 3 presents MICQ in detail. The experiments are provided and discussed in Section 4. Section 5 concludes the paper.

## 2 Background

This section provides the necessary background to introduce MICQ. It first considers formally the problem of learning to quantify and then (Mondrian) conformal classification.

### 2.1 Problem of Learning to Quantify

We formalize the problem of learning to quantify (L2Q) as follows. Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ a finite class-label set. At training time the phenomenon under study is given by an unknown distribution $P$ on $\mathcal{X} \times \mathcal{Y}$ with $(X, Y) \sim P$. We observe $M$ i.i.d. realizations $(x_m, y_m)$ of $(X, Y)$ in $\mathcal{X} \times \mathcal{Y}$ that form the training data $D := \{(x_m, y_m)\}_{m=1}^{M}$. At test time, we face another unknown distribution $P^*$ on $\mathcal{X} \times \mathcal{Y}$ and we observe $N$ i.i.d. realizations $x_n$ of $X$ in $\mathcal{X}$ that form a test bag $B := \{x_n\}_{n=1}^{N}$. For each class label $y \in \mathcal{Y}$ we denote the class priors by $\pi_y := P(Y = y)$ and $\pi_y^* := P^*(Y = y)$ and combine them in vectors $\boldsymbol{\pi} = (\pi_y)_{y \in \mathcal{Y}}$ and $\boldsymbol{\pi}^* = (\pi_y^*)_{y \in \mathcal{Y}}$, respectively. In this context, the problem of L2Q is to estimate the class priors in $\boldsymbol{\pi}^*$, given the training set $D$ and the unlabeled test bag $B$ [1]. Thus, a L2Q estimator is supposed to output vector $\hat{\boldsymbol{\pi}}_B$ for bag $B$ as close as possible to $\boldsymbol{\pi}_B$.

In this paper we consider the problem of L2Q for two cases:

– No distribution shift: $P = P^*$, and

[1] We note that the individual test class labels are never observed and the goal is *not to classify*, but to estimate label proportions.

– Prior distribution shift: $P(Y) \neq P^*(Y)$ (i.e. $\pi_y \neq \pi^*_y$ for any $y \in \mathcal{Y}$) and $P(X \mid Y = y) = P^*(X \mid Y = y)$ for any class label $y \in \mathcal{Y}$.

## 2.2 Conformal Classification

Conformal prediction is a framework for producing set-valued predictions with rigorous, statistical finite-sample guarantees [6]. Conformal predictors construct valid prediction sets $\Gamma^\alpha(X)$ for any new test point $X$, meaning that the true class label $Y$ lies inside the prediction set with probability

$$\Pr_{(X,Y)\sim P}(Y \in \Gamma^\alpha(X)) \geq 1 - \alpha \tag{1}$$

for any user-specified miscoverage rate $\alpha \in (0, 1)$. This validity guarantee holds when there is no distribution shift ($P = P^*$) which implies that the training and test instances are exchangeable [6]. We later show how to maintain validity in the case of prior distribution shift.

The validity guarantee above is achieved by comparing the nonconformity score of a test example to those of the instances from the training set $D$. A nonconformity score function $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ measures how unusual a labeled instance $x \in \mathcal{X}$ appears w.r.t to other instances in $D$. Function $s$ has to be defined *label-symmetric* s.t. for any fixed $x \in \mathcal{X}$ and for any permutation $\tau : \mathcal{Y} \to \mathcal{Y}$, the set of values $\{s(x, y) : y \in \mathcal{Y}\}$ equals the set $\{s(x, \tau(y)) : y \in \mathcal{Y}\}$; i.e., permuting the class labels does not change the distribution of non-conformity scores across the class-label space. One common choice for the function $s$ is $s(x, y) = 1 - \widehat{p}(y \mid x)$, where $\widehat{p}(\cdot \mid x)$ is the posterior probability output by a probabilistic classifier $h$.

The most computationally efficient setting for conformal prediction is the *inductive setting* [2,3]. In this setting, the training set $D$ is partitioned into two disjoint subsets, a *proper training set* $D_t = \{(X_m, Y_m)\}_{m=1}^{M_t}$ and a *calibration set* $D_c = \{(X_m, Y_m)\}_{m=M_t+1}^{M}$ with sizes $M_t$ and $M_c = M - M_t$, respectively. A probabilistic classifier $h$ is first trained on the proper training set $D_t$ as a predictive function that can provide posterior probability estimates $\widehat{p}(y \mid X)$ for any class label $y \in \mathcal{Y}$ and input instance $X$; i.e. $h(X) = \{\widehat{p}(y \mid X)\}_{y\in\mathcal{Y}}$. The classifier $h$ is first applied to all the calibration instances $(X_m, Y_m) \in D_c$ to obtain set $S$ of their nonconformity scores $s_m := s(X_m, Y_m) = 1 - \widehat{p}(Y_m \mid X_m)$. Then, whenever we have a new test instance $X \sim P^*(X)$ the probabilistic classifier $h$ provides a posterior class-label distribution $\{\widehat{p}(y \mid X)\}_{y\in\mathcal{Y}}$. These class-label probability estimates allow computing their associated nonconformity scores $s(X, y) = 1 - \widehat{p}(y \mid X)$ for $X$ and every possible class label $y \in \mathcal{Y}$.

The p-values

$$p_y(X) = \frac{\#\{s_m \in S : s_m \geq s(X, y)\} + 1}{M_c + 1},$$

of test instance $X$ for each class label $y \in \mathcal{Y}$ determine if the class label $y$ is included in prediction set. If $p_y(X)$ is greater than a miscoverage rate $\alpha$, $y$ is included in prediction set $\varGamma^{\alpha}(X)$. Thus, the final prediction set for $X$ at $\alpha$ equals

$$\varGamma^{\alpha}(X) = \{\, y \in \mathcal{Y} : p_y(X) > \alpha \}.$$

As stated above, when there is no distribution shift, $P = P^*$, and exchangeability holds, so does the validity guarantee expressed in Equation (1). However, when there exists prior distribution shift, $P(Y) \neq P^*(Y)$, and $P(X \mid Y) = P^*(X \mid Y)$, validity is no longer guaranteed. To address this problem we describe below the Mondrian conformal predictor (MCP) in the inductive setting.

**Mondrian Inductive Conformal Classification** Mondrian inductive conformal predictor (MICP) is a class-conditional conformal predictor [6]. To handle distribution prior shift, it divides the nonconformity scores of the calibration instances by class labels $y$ [6]. For each class label $y$ we receive set $\mathcal{S}_y = \{s(X_m, Y_m) : (X_m, Y_m) \in D_c, Y_m = y\}$ with size $M_y$. This allows us to compute the Mondrian class-conditional p-value $p_y(X)$ of test instance $X$ for candidate class label $y$

$$p_y(X) = \frac{\#\{s_m \in S_y : s_m \geq s(X,y)\} + 1}{M_y + 1}, \tag{2}$$

and, subsequently, the class-conditional prediction set

$$\varGamma^{\alpha}(X) = \{y : p_y(X) > \alpha\}. \tag{3}$$

We note that under the prior-shift assumption, test instance $X \mid (Y = y)$ is drawn from the same distribution as each calibration instance for class label $y$, so the set $S_y \cup \{s(X, y)\}$ is exchangeable, i.e. every permutation of these $M_y + 1$ scores has the same joint distribution. Exchangeability implies that the rank $R_y$ of $s(X, y)$ among the $M_y + 1$ scores is uniformly distributed:

$$\Pr(R_y = k \mid Y = y) = \frac{1}{M_y + 1}, \qquad k = 1, \ldots, M_y + 1.$$

In this context, following Equation 2, the Mondrian class-conditional p-value can be redefined as

$$p_y(X) = \frac{R_y}{M_y + 1}.$$

Therefore,

$$\Pr\left(p_y(X) \leq \alpha \mid Y = y\right) = \frac{\lfloor \alpha(M_y + 1) \rfloor}{M_y + 1} \leq \alpha.$$

Because the event $Y \notin \varGamma^{\alpha}(X)$ is equivalent to $p_y(X) \leq \alpha$,

$$\Pr\left(Y \notin \varGamma^{\alpha}(X) \mid Y = y\right) = \Pr\left(p_y(X) \leq \alpha \mid Y = y\right) \leq \alpha. \tag{4}$$

Hence,

$$\Pr(Y \in \varGamma^{\alpha}(X) \mid Y = y) \geq 1 - \alpha$$

for every class $y$, showing that Mondrian inductive conformal predictors are valid for every class $y$.

## 3 Mondrian Inductive Conformal Quantification (MICQ)

Mondrian inductive conformal quantification (MICQ) is an approach to estimating class proportions in an unlabeled dataset. It offers robust estimations in cases of no distribution shift and distribution prior shift.

### 3.1 MICQ Description

The pseudocode of MICQ for the training and calibration phase is given in Algorithm 1. The input required for this phase consists of: a proper training set ($D_t$) and a calibration set ($D_c$). MICQ starts by training a probabilistic classifier $h$ on the training set $D_t$. Then it iterates over the calibration instances in $D_c$ to compute for each instance $(X_m, Y_m) \in D_c$ its probability estimate $\hat{p}(Y_m \mid X_m) = h(X_m)(Y_m)$ and its nonconformity score $s(X_m, Y_m) = 1 - \hat{p}(Y_m \mid X_m)$. Once the scores are known, MICQ forms the sets $\mathcal{S}_y$ of nonconformity scores for each class. At the end of the training and calibration phase MICQ outputs the probabilistic classifier $h$ and the set of the calibration nonconformity score sets $\{\mathcal{S}_y\}_{y \in \mathcal{Y}}$.

**Algorithm 1** MICQ – Training and Calibration Phase

```
Require: Proper training set $D_t = \{(X_m, Y_m)\}_{m=1}^{M_t}$, calibration set $D_c = \{(X_m, Y_m)\}_{m=M_t+1}^{M}$
Ensure: Trained classifier $h$, label-conditional score sets $\{\mathcal{S}_y\}$
1: Train a probabilistic classifier $h$ on $D_t$
2: for each calibration instance $(X_m, Y_m) \in D_c$ do
3:     $\hat{p}(Y_m \mid X_m) = h(X_m)(Y_m)$
4:     $s(X_m, Y_m) = 1 - \hat{p}(Y_m \mid X_m)$
5: end for
6: for each class $y \in \mathcal{Y}$ do
7:     $\mathcal{S}_y = \{s(X_m, y) : (X_m, Y_m) \in D_c, Y_m = y\}$
8: end for
9: return $h$ and $\{\mathcal{S}_y\}_{y \in \mathcal{Y}}$
```

The pseudocode of MICQ for the test and quantification phase is given in Algorithm 2. The input required for this phase consists of an unlabeled test bag $B$ which class proportions need to be estimated, a user-defined miscoverage rate ($\alpha$), a probabilistic classifier $h$, a set of the calibration nonconformity score sets $\{\mathcal{S}_y\}_{y \in \mathcal{Y}}$, a miscoverage rate $\alpha$, and a temperature parameter $\tau$. MICQ starts by initializing the prevalence value $\hat{\pi}_B^*(y)$ equal to 0 for all class labels $y \in \mathcal{Y}$. Then it iterates over instances $X$ in the test bag $B$. For each test instance $X$ and each possible class $y \in \mathcal{Y}$ MICQ computes its probability estimate $\hat{p}(y \mid X) = h(X)(y)$ using the probabilistic classifier $h$, and its nonconformity score $s(X, y) = 1 - \hat{p}(y \mid X)$. Then the p-value for the labeled instance $(X, y)$ is computed following the MICP procedure (see Equation (2)). Once the p-values $p_y$ are available for test instance $X$ and all class labels $y \in \mathcal{Y}$, MICQ computes the prediction set $\mathit{\Gamma}^{\alpha}(X)$ for $X$ according to Equation (3). The resulting prediction set $\mathit{\Gamma}^{\alpha}(X)$ is composed of all class labels with p-values greater than $\alpha$. Once the prediction set $\mathit{\Gamma}^{\alpha}(X)$ is available for test instance $X$, the weight $w_y(X)$ of $X$ for $y$ is computed according to:

$$w_y(X) = \begin{cases} \dfrac{\widehat{p}(y \mid X)^{\tau}}{\sum_{z \in \mathit{\Gamma}^{\alpha}(X)} \widehat{p}(z \mid X)^{\tau}}, & y \in \mathit{\Gamma}^{\alpha}(X), \\ 0, & \text{otherwise.} \end{cases}$$

where $\tau$ is a temperature parameter that controls the weight's sharpness ($\tau \geq 0$).

MICQ then updates the prevalence value $\hat{\pi}^*_B(y)$ for each class label $y$ by adding the weight $w_y(X)$ of $X$ to $\hat{\pi}^*_B(y)$. Then this entire procedure is repeated for all instances $X$ in the test bag $B$. Upon completion, the final vector $\hat{\boldsymbol{\pi}}^*_B$ that is output by MICQ gives the estimated prevalence for each class $y$:

$$\hat{\pi}^*_B(y) = \frac{1}{n} \sum_{X \in B} w_y(X).$$

**Algorithm 2** MICQ – Test and Quantification Phase

```
Require: Test bag $B = \{X_n\}_{n=1}^{N}$, set of the calibration nonconformity score sets
    $\{\mathcal{S}_y\}_{y \in \mathcal{Y}}$, miscoverage rate $\alpha$, and temperature parameter $\tau$
Ensure: Estimated prevalence vector $\hat{\boldsymbol{\pi}}^*_B$
 1: Initialize $\hat{\pi}^*_B(y) = 0$ for all $y \in \mathcal{Y}$
 2: for each test instance $X \in B$ do
 3:     for each class $y \in \mathcal{Y}$ do
 4:         $\hat{p}(y \mid X) = h(X)(y)$
 5:         $s(X, y) = 1 - \hat{p}(y \mid X)$
 6:         $p_y(X) = \frac{\#\{s_m \in \mathcal{S}_y : s_m \geq s(X,y)\} + 1}{|\mathcal{S}_y| + 1}$
 7:     end for
 8:     $\mathit{\Gamma}^{\alpha}(X) = \{y \in \mathcal{Y} : p_y(X) > \alpha\}$
 9:     for each class $y \in \mathit{\Gamma}^{\alpha}(X)$ do
10:         $w_y(X) = \dfrac{\widehat{p}(y \mid X)^{\tau}}{\sum_{z \in \mathit{\Gamma}^{\alpha}(X)} \widehat{p}(z \mid X)^{\tau}}$
11:         $\hat{\pi}^*_B(y) \leftarrow \hat{\pi}^*_B(y) + w_y(X)$
12:     end for
13: end for
14: for each class $y \in \mathcal{Y}$ do
15:     $\hat{\pi}^*_B(y) \leftarrow \frac{1}{N} \cdot \hat{\pi}^*_B(y)$
16: end for
17: return $\hat{\boldsymbol{\pi}}^*_B = (\hat{\pi}^*_B(y))_{y \in \mathcal{Y}}$
```

### 3.2 Robustness to Prior Shift

MICQ is robust to *prior shift*; i.e. when there are changes in class priors $\pi_y \neq \pi_y^*$ while the class-conditional distribution $P(X \mid Y) = P^*(X \mid Y)$ remain fixed. This property is mainly due to class-conditional calibration of the Mondrian conformal predictor employed in MICQ. This predictor splits the calibration scores by class label ($\mathcal{S}_y$ for each $y$) ensuring that the feature distribution *within each class* is the same in training and test. Thus, any multiset $\mathcal{S}_y \cup \{s(X, y)\}$ stays exchangeable for every class $y$ and the class-conditional $p$-value $p_y(X) = \frac{\#\{s \in \mathcal{S}_y : s \geq s(X,y)\}+1}{|\mathcal{S}_y|+1}$ satisfies the finite-sample coverage guarantee $\Pr\big(Y \in \Gamma^\alpha(X) \mid Y = y\big) \geq 1 - \alpha$ *even when class priors change.*

### 3.3 Regularization in MICQ

MICQ has two parameters that jointly control its bias and variance:

- **Miscoverage rate $\alpha$ (set size).** Smaller $\alpha$ yields *larger* prediction sets $\Gamma^\alpha$ which spreads each instance's contribution over more class labels and thus decreases variance and increases bias (toward more uniform class prevalences). Larger $\alpha$ yields *smaller* sets $\Gamma^\alpha$, and, thus, sharpens contributions and reduces bias at the cost of higher variance.
- **Temperature parameter $\tau$ (within-set concentration).** For a fixed set $\Gamma^\alpha$, when $\tau$ approaches $0^+$, the weight $w_y$ for any class label $y \in \Gamma^\alpha$ becomes closer to $1/|\Gamma^\alpha|$ which *increases bias* and *decreases variance.* When $\tau$ approaches $+\infty$, the weight $w_y$ becomes closer to 1 for class label $y$ with maximal posterior probability ($y = \arg\max_{z \in \Gamma^\alpha}$) and 0 for the remaining class labels. This *reduces bias* when posterior probabilities are informative and can *increase variance.*

The two parameters interact: the influence of $\tau$ is most pronounced when sets $|\Gamma^\alpha| > 1$ (moderate $\alpha$). In this case we need to choose $(\alpha, \tau)$ so that the sets are informative sets (through $\alpha$) and appropriately concentrated within-set weights (through $\tau$). In this context, we note that when sets $\Gamma^\alpha$ are singletons, $\tau$ has no effect.

### 3.4 Remark on Fisher Consistency

Fisher consistency requires that, when applied to the true test distribution, the expected estimate equals the true prevalence for all $y$. When $\alpha = 0$ and $\tau = 1$, the MICQ reduces to PCC, which *is* Fisher consistent.

For any $\alpha > 0$, however, Fisher consistency no longer holds. By construction, the conformal set $\Gamma^\alpha(X)$ excludes the true label with probability at most $\alpha$ (exactly $\alpha$ under continuous scores or randomized $p$-values), so the correct class receives weight 0 in a nonzero fraction of cases. In the remaining cases, the correct label is included but its contribution can be diminished due to the within-set normalization.

## 4 Experiments

This section evaluates and compares the performances of the MICQ quantifier and the standard probability class counting (PCC) quantifier. It provides experimental setup, experiment results, and discussion.

### 4.1 Experimental Setup

The L2Q quantifiers used in the experiments are the MICQ quantifier and PCC quantifier. To ensure model comparability both quantifiers employ the same base classifier the *Gaussian Naïve Bayes*. The Mondrian inductive conformal predictor in MICQ employs 67%/33% split, meaning that 67% of the data is used as training data $D_t$ of the base classifier (Gaussian Naïve Bayes) and 33% of the data is used for the calibration set $D_c$.

### 4.2 Datasets

We conduct experiments on three standard benchmarking datasets from `scikit-learn` [4]. The datasets are described in Table 1.

**Table 1.** Datasets used in experiments

| **Dataset** | **# Classes** | **# Features** | **# Instances** |
|---|---:|---:|---:|
| Iris | 3 | 4 | 150 |
| Breast Cancer | 2 | 30 | 569 |
| Digits | 10 | 64 | 1797 |

### 4.3 Validation Protocol

We follow a *10-fold stratified cross-validation* protocol. To simulate realistic quantification tasks under prior shift, we generate *bootstrap bags* as follows:

- For each test fold, we generate $T = 20$ test bags of the same size as the test set.
- Bags are sampled with replacement from the test fold.
- Each bag's class proportions are drawn from a Dirichlet distribution with concentration parameter $\alpha_{\text{Dir}} \in \{10, 100, 1000\}$ to simulate various prior shift conditions. We note that bigger (smaller) values for $\alpha_{\text{Dir}}$ imply smaller (bigger) prior shifts respectively.

### 4.4 Quantification and Evaluation

For each test bag, we compute the true class prevalences $\boldsymbol{\pi}^*$, the PCC estimates, and the MICQ estimates. Performance is evaluated using *Mean Absolute Error (MAE)*:

$$\mathrm{MAE}(\hat{\boldsymbol{p}}_B, \boldsymbol{\pi}^*) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \left|\hat{\pi}_B(y) - \pi_y^*\right|,$$

We study MAE as a function of miscoverage level $\alpha \in [0.005, 0.500]$ (step size $= 0.005$) to observe how the conformal miscoverage rate $\alpha$ acts as a *regularization parameter* in MICQ.

**Table 2.** Summary of experimental settings

| **Parameter** | **Value / Description** |
|---|---|
| Base Classifier | Gaussian Naïve Bayes |
| Cross-Validation | 10-fold Stratified CV |
| Calibration Ratio | 0.33 |
| Bags per Fold | $T = 20$ |
| Bag Size | Equal to test fold size |
| Bag Sampling | With replacement (bootstrap) |
| Dirichlet Parameters | $\alpha_{\mathrm{Dir}} \in \{10, 100, 1000\}$ |
| Miscoverage Levels | $\alpha \in [0.005, 0.500]$ |
| Temperature | $\tau = 2$ |
| Evaluation Metric | Mean Absolute Error (MAE) |

### 4.5 Results and Discussion

Figure 1 presents the results for the Iris, Breast Cancer, Digits datasets under varying Dirichlet prior strengths. For the Iris and Digits datasets, MICQ consistently outperforms the PCC baseline, with its MAE curve lying below the horizontal PCC baseline for all miscoverage levels $\alpha$. In contrast, on the Breast Cancer dataset the picture is different: for small miscoverage levels ($\alpha < 0.12$) the MICQ's MAE is below the PCC baseline with exception of very low $\alpha$ values. For $\alpha > 0.12$ MICQ is worse than PCC. The Breast-Cancer curve also exhibits a spike near $\alpha \approx 0.07$ due to discrete p-values and small per-class calibration counts. Thus, we may conclude that MICQ can outperform PCC across a spectrum of prior shift scenarios, from no shift to substantial shift.

The results presented in Figure 1 show a clear bias-variance trade-off in Mondrian-weighted MICQ. For very small miscoverage rate $\alpha$, prediction sets are large, leading to low variance but high bias and thus higher MAE. For moderate $\alpha$, prediction sets shrink, bias drops, and MAE reaches its minimum, often outperforming the PCC baseline. For large $\alpha$, MAE rises again due to high variance from prediction sets containing mostly single class labels. The trade-off pattern holds across varying degrees of prior shift ($\alpha_{\text{Dir}} = 10, 100, 1000$) and it confirms that Mondrian-weighted MICQ balances bias and variance and adapts well to prior shift.

## 5 Conclusion

We introduced *Mondrian Inductive Conformal Quantification* (MICQ), a simple and effective set-based extension of the *probability class counting* (PCC) approach to learning to quantify. By leveraging prediction sets from Mondrian conformal prediction, MICQ assigns soft fractional weights to classes, enabling accurate prevalence estimation under prior shift. We showed empirically that MICQ is capable of outperforming PCC. These results demonstrate that conformal prediction offers a way to regularize quantification and ensure robustness to distributional changes.

Future work may continue in two possible directions. The first one is to apply conformal prediction to more advanced L2Q techniques. The second direction is to apply conformal predictors specifically designed for covariate shifts [5].

## References

1. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to Quantify, The Information Retrieval Series, vol. 47. Springer International Publishing, Cham (2023). https://doi.org/10.1007/978-3-031-20467-8, https://link.springer.com/10.1007/978-3-031-20467-8
2. Papadopoulos, H., Proedrou, K., Vovk, V., Gammerman, A.: Inductive Confidence Machines for Regression. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) Machine Learning: ECML 2002. pp. 345–356. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
3. Papadopoulos, H., Vovk, V., Gammerman, A.: Qualified predictions for large data sets in the case of pattern recognition. In: Wani, M., Arabnia, H., Cios, K., Hafeez, K., Kendall, G. (eds.) Proceedings of the International Conference on Machine Learning and Applications. pp. 159–163. CSREA Press (2002)
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
5. Tibshirani, R.J., Foygel Barber, R., Candes, E., Ramdas, A.: Conformal Prediction Under Covariate Shift. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F.d., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf
6. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-06649-8, https://link.springer.com/10.1007/978-3-031-06649-8

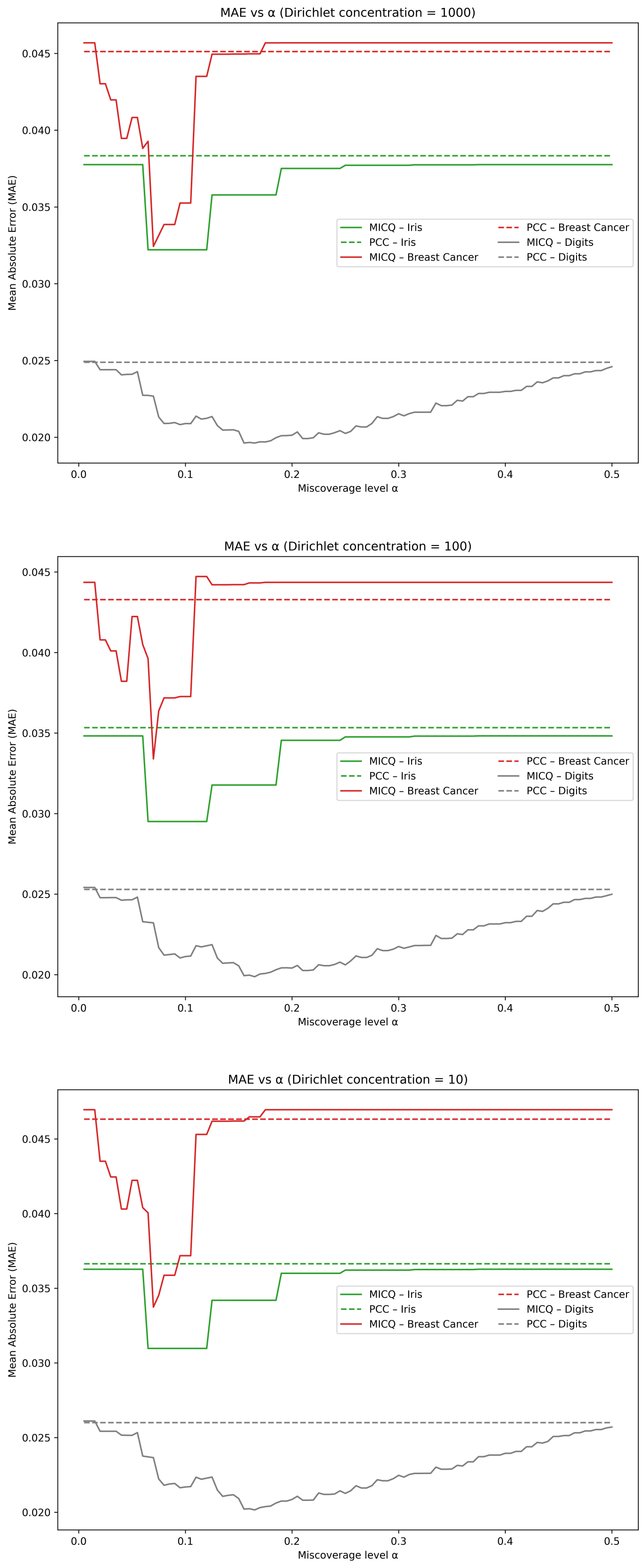

**Fig. 1.** Mean Absolute Error of MICQ vs. PCC across miscoverage levels for different Dirichlet strengths. From top to bottom: $\alpha_{\text{Dir}} = 1000$, $\alpha_{\text{Dir}} = 100$, $\alpha_{\text{Dir}} = 10$.

# An Efficient Method for Deriving Confidence Intervals in Aggregative Quantification

Alejandro Moreo[1] (✉) and Nicola Salvati[2]

[1] Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy alejandro.moreo@isti.cnr.it
[2] Dipartimento di Economia e Management, Università di Pisa, Pisa, Italy nicola.salvati@unipi.it

**Abstract.** This paper explores efficient methods for deriving confidence intervals in quantification, the area of machine learning concerned with estimating class prevalence values. By focusing on computationally efficient strategies, we propose a robust framework for quantifying uncertainty. The key idea is to disentangle the two main phases of current aggregative quantifiers (classification followed by aggregation) and apply bootstrap only to the second phase. We investigate different methods for constructing confidence regions, including confidence intervals, confidence ellipses in the simplex, and confidence regions in the transformed Centered Log-Ratio space. Additionally, we examine various bootstrap strategies, including model-based, population-based, and a combined approach. Our results demonstrate the effectiveness of combining model-based and population-based bootstrap approaches, particularly when used with traditional confidence intervals, while also achieving significant efficiency gains compared to a naive application of bootstrap.

**Keywords:** Confidence intervals · Class prevalence estimation · Quantification.

## 1 Introduction

Many disciplines, like the social sciences [16], epidemiology [18], ecological modelling [1], and sentiment analysis [12], are interested in knowing the class prevalence distribution of the population under study. Quantification is a key branch of supervised machine learning that focuses on estimating the prevalence of class labels in a population [13,7]. Such methods are called "quantifiers" and, just like with any other point estimators, their predictions are inevitably affected by error. It would therefore be desirable to accompany the outputs of a quantifier with a measure of *uncertainty*, in the form of confidence regions [24].

A standard method used in statistics for deriving confidence regions is *bootstrap* [6]. Bootstrap is a resampling method used to estimate the distribution of a statistic by repeatedly sampling with replacement from the observed data. It allows for the construction of confidence regions by calculating the statistic of interest (in our case, the sample prevalence) across multiple resampled bags and determining the region that captures the desired percentage of the resampled estimates (e.g., 95% of these).

In this report, we analyse how to apply bootstrap to quantification. In particular, we focus our attention to one class of quantifiers called *aggregative quantifiers* (see §4.2 in [7]), which rely, in order to predict class prevalence values, on the outputs of a surrogate classifier. This family of methods seems appealing for our scope since the procedure follows a pipeline of two steps: the learning phase comes down to (i) training a classifier which operates at the individual level, and (ii) fitting an *adjustment* method which operates at the aggregate level; while the inference phase can be subdivided into (i') generating classifier predictions for each test instance, and (ii') applying the adjustment function to the sample to produce prevalence estimates. Steps (i') and especially (i) are computational heavier than steps (ii') and (ii). Note thus that this two-step process is convenient for deriving confidence intervals via bootstrap since it offers us the possibility to carry out steps (i) and (i') only once, and apply bootstrap only in steps (ii) and (ii'), thus substantially speeding up the otherwise unaffordable computational cost.

The rest of this report is structured as follows. In Section 2 we overview past work on confidence intervals in quantification. Section 3 is devoted to explaining our application of bootstrap to aggregative quantifiers. Section 4 reports the experiments we have carried out while Section 5 wraps up and offers pointers for future research.

## 2 Related Work

The use of bootstrap in quantification is not novel. Probably, the first published record discussing the application of bootstrap for deriving confidence intervals in quantification is by [16]. However, the authors only mention the application of “standard bootstrapping procedures” without providing specific clues on how they apply it.

[17] proposes a generative probabilistic model for text quantification which assumes a document's text is generated conditional on the document label. As the generative model the authors explore multinomial Naïve Bayes and additive log-linear methods. Inference is carried out via marginal log-likelihood over the priors. While this paper uses real-world data in the experiments, these are restricted to binary quantification and textual data.

[4] propose a method called “Error-Adjusted Bootstrapping” which also relies on bootstrap resampling. The idea is to use a (crisp) classifier to issue label predictions for all the instances of the bootstrap samples. However, these labels are not directly used to obtain bootstrap prevalence predictions by simply averaging. Instead, the authors propose to randomly sample the predicted instance label based on the misclassification-rates distribution $P(Y|\hat{Y})$ that characterizes the classifier, which is modelled on training data. However, as noted by [24], this method is not appropriate for constructing confidence intervals in the presence of prior probability shift (which is the focus of our work –Section 3.1). The

reason is that the conditional distribution $P(Y|\hat{Y})$ is not stationary under such conditions. Also [4] limits the experimental section to binary quantification.

[24] carries out a simulation study in which several quantifiers are tested in their ability to derive confidence intervals and prediction intervals. This paper also clarifies that there is a distinction between confidence intervals and prediction intervals [19] which is often neglected in the literature. Be it as it may, the author himself raises questions on the necessity of pushing forth this distinction in practical applications of quantification. However, this study is limited to binary quantification only, and to the idealizations typical of simulated conditions that may obscure real-world complexities of the data.

Other works exist which tries to derive confidence intervals analytically. One example is by [9], which propose the "ratio estimator" method for quantification. The authors derive confidence intervals by analysing the asymptotic properties of the method, thus avoiding iterative resampling procedures. As a downside, the method is specific for the ratio estimator quantifier. In contrast, the method we propose is generic and applicable to any of the "aggregative" methods described in the quantification literature (by far, the largest class of methods in the field).

[5] shows that the Probabilistic Classify and Count method (PCC), a method that predicts prevalence values by averaging across the posterior probabilities on each data point that a probabilistic classifier generates [2], can be endowed with confidence intervals. The idea resides in treating this point estimate as the mean of a Poisson binomial distribution of the posterior probabilities, for which confidence intervals can be computed easily. However, PCC is known to be a weak quantifier under prior probability shift conditions [14].

Recently, a (tractable) Bayesian approach for deriving confidence intervals around prevalence point estimators based on black-box classifiers is presented in [25]. While this method caters for multiclass problems, it also suffers from a high computationally cost since the method relies on Hamiltonian Markov Chain Monte Carlo sampling.

## 3 Method

In this section, we turn to describe a methodology for applying bootstrap to aggregative quantification methods. First, we set down the notation we use.

### 3.1 Notation

Let $q : \mathbb{N}^{\mathcal{X}} \rightarrow \Delta^{n-1}$ be a function (a *quantifier*) that maps bags of elements from the input space $\mathcal{X}$ into class label distributions, i.e., into elements of the probability simplex $\Delta^{n-1} \equiv \{(\pi_1, \ldots, \pi_n) \mid \pi_i \geq 0, \sum_{i=1}^{n} \pi_i = 1\}$ where $n$ is the number of classes and $\pi_i$ represents the proportion of elements from the bag that belong to class $i$, with $\mathcal{Y} = \{1, \ldots, n\}$ the classes of interest.

Given an $\underline{u}$nlabelled test set $U$ with (unknown) prevalence $\boldsymbol{\pi}^* \in \Delta^{n-1}$, we are interested in training a quantifier $q$ such that $q(U) = \hat{\boldsymbol{\pi}}$ is a good approximation of $\boldsymbol{\pi}^*$ in terms of any given divergence metric.

To this aim, we assume to have access to a labelled training set $L = \{(x_i, y_i)\}_{i=1}^{l}$ of $l$ labelled datapoints $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$. We also consider a learner device $\Psi : (\mathcal{X} \times \mathcal{Y})^l \to \mathcal{Q}$ that takes as input a training set and generates (i.e., learns) a quantifier function $q \in \mathcal{Q}$ from a class of quantification functions $\mathcal{Q}$.

We assume to be in the presence of prior probability shift (PPS), a type of dataset shift in which the training distribution $P$ and the test distribution $Q$ are different, but in which the following assumptions are made:

$$P(Y) \neq Q(Y)$$
$$P(X|Y) = Q(X|Y)$$

where $Y$ is the random variable taking values on the output space $\mathcal{Y}$ and $X$ is the random variable taking values in the input space $\mathcal{X}$.

## 3.2 Bootstrap

This work aims to provide not only a point estimate but also a confidence region around it. *Bootstrap* is a standard and versatile method for constructing confidence regions by resampling data [6]. The idea is to uniformly draw a series of samples $U_1, \ldots, U_m$ from $U$, with replacement and with $|U_i| = |U|$, and to use $q$ to generate the corresponding predictions $\hat{\boldsymbol{\pi}}_1 = q(U_1), \ldots, \hat{\boldsymbol{\pi}}_m = q(U_m)$, which are then used to define the confidence region around the final point estimate which is finally computed as $\underline{\hat{\boldsymbol{\pi}}} = \frac{1}{m}\sum_{i=1}^{m} \hat{\boldsymbol{\pi}}_i$. This way of deriving confidence intervals based on resampling the test set is called the *population-based* approach.

A different way for deriving confidence regions is the so-called *model-based* approach, in which the resampling is instead applied to the training set $L$. This way, a series of training samples $L_1, \ldots, L_{m'}$ is generated from $L$, with replacement and with $|L_i| = |L|$. The series of point estimates are then computed as $\hat{\boldsymbol{\pi}}_1 = q_1(U), \ldots, \hat{\boldsymbol{\pi}}_{m'} = q_{m'}(U)$, where $q_i = \Psi(L_i)$. The final estimate is computed in the usual way: $\underline{\hat{\boldsymbol{\pi}}} = \frac{1}{m'}\sum_{i=1}^{m'} \hat{\boldsymbol{\pi}}_i$, and the series of estimates $\hat{\boldsymbol{\pi}}_1, \ldots, \hat{\boldsymbol{\pi}}_{m'}$ is used to derive confidence regions around $\underline{\hat{\boldsymbol{\pi}}}$.

Yet a third method for deriving confidence regions comes down to combining the population-based approach with the model-based approach. In this way, we generate a grid of $m \times m'$ point estimates $G = [g_{ij}] = q_j(U_i)$ where $q_j = \Psi(L_j)$. The point estimate is thus computed as $\underline{\hat{\boldsymbol{\pi}}} = \frac{1}{mm'}\sum_{i=1}^{m}\sum_{j=1}^{m'} g_{ij}$, and all the estimates in $G$ concur in deriving a confidence region around $\underline{\hat{\boldsymbol{\pi}}}$.

**Bootstrap is Quantification Experiments** Due to repeated resampling, bootstrap methods incur a considerable computational cost, specially when a model-based approach is adopted. A typical value for $m$ or $m'$ is 500 repetitions. Generating 500 predictions for a test set is already costly (population-based approach), but the problem is even more exacerbated if the number of trainings rounds increases to 500 (model-based approach). When a combination of population-based and model-based approaches are adopted, the prohibitive number of combinations $500 \times 500$ is often reduced to $100 \times 100$. Still, the cost seems unaffordable from a computational point of view.

The latter is especially true in a discipline like quantification, in which the evaluation of a model often consists of confronting the quantifier against *many* test samples. The reason is that one test set counts as only one test *instance* for a quantifier; this is in contrast to other disciplines like classification, or regression, in which one test set represents a number of test instances that equals the cardinality of the set.

Quantification papers do often resort to sampling protocols to generate, out of a test sample, many samples each characterized by (possibly radically) different prevalence values. The number of test samples thus generated is often high in order to allow for statistically significant results. Typical values encountered in the quantification literature easily reach 100 samples [20], 1000 samples [21], or even more [8,22]. The aforementioned “resampling” cost that bootstrap incurs (500, or $100 \times 100$ repetitions, depending on the case) must therefore be undertaken *for each test sample* extracted from a dataset and *for each dataset* under consideration (modern papers involve dozens of datasets [22,23,21]). With this landscape, the adoption of bootstrap in quantification clearly seems impractical.

**Bootstrapping Aggregative Quantifiers** In this study, we propose a variant of bootstrap applied on a specific class of quantification methods called *aggregative* (see §4.2 in [7]). These methods are such that the training procedure can be disentangled as follows:

1. Learn a crisp (resp. soft) classifier $h \in \mathcal{H}$, i.e., a functional $h : \mathcal{X} \rightarrow \mathcal{Y}$ (resp. $h : \mathcal{X} \rightarrow \Delta^{n-1}$) using a learner device $\mathcal{C} : (\mathcal{X} \times \mathcal{Y})^l \rightarrow \mathcal{H}$ and a training set $L$; then use $h$ to generate classifier predictions for each element in the training set via cross-validation, thus obtaining $\tilde{L} = \{(h(x_i), y_i) : (x_i, y_i) \in L\}$.
2. Learn an *adjustment* function $A : (\mathcal{Y} \times \mathcal{Y})^l \rightarrow \Delta^{n-1}$ (resp. $A : (\Delta^{n-1} \times \mathcal{Y})^l \rightarrow \Delta^{n-1}$) using the classifier's outputs $L$. The adjustment function is responsible for generating, out of the classifier predictions, the final class prevalence estimates.

Similarly, the inference procedure can be decomposed as:

1. Use $h$ to generate classifier predictions for all instances in the test set $\tilde{U} = \{h(x_i) : x_i \in U\}$.
2. Return the prevalence values generated via $A(\tilde{U})$.

Note that the first step of the training phase, and the first step of the inference phase, are often much more expensive, from a computational point of view, than the corresponding steps number 2. We take advantage of this characteristic and propose to apply:

– Model-based bootstrap:
  - During training, carry out step 1 once and for all, and apply resampling only during step 2, i.e., generating many samples to $\tilde{L}_1, \ldots, \tilde{L}_{m'}$ out of $\tilde{L}$, thus obtaining a series of adjustment functions $A_1, \ldots, A_{m'}$.

  - During inference, generate $\tilde{U}$ (step 1) and generate a series of point estimates $A_1(\tilde{U}), \ldots, A_{m'}(\tilde{U})$ applying step 2 on every adjustment function.
- Population-based bootstrap:
  - During training, run steps 1 and 2 of the training procedure the usual way.
  - During inference, generate all classifier predictions (step 1) only once, and then apply resampling over $\tilde{U}$ for step 2, thus generating a series of bags $\tilde{U}_1, \ldots, \tilde{U}_m$ with which a series of point estimates $A(\tilde{U}_1), \ldots, A(\tilde{U}_m)$ is computed.
- Combined model-based and population-based approach: a trivial combination of the above methods.
  - During training, carry out step 1 once and for all, and apply resampling only during step 2, i.e., generating many samples to $\tilde{L}_1, \ldots, \tilde{L}_{m'}$ out of $\tilde{L}$, thus obtaining a series of adjustment functions $A_1, \ldots, A_{m'}$.
  - During inference, generate all classifier predictions (step 1) only once, and then apply resampling over $\tilde{U}$ for step 2, thus generating a series of bags $\tilde{U}_1, \ldots, \tilde{U}_m$. Generate a grid of point estimates pairing every $A_i$ $(1 \leq i \leq m')$ adjustment function with every $\tilde{U}_j$ $(1 \leq j \leq m)$ bag.

## 3.3 Methods for Constructing Confidence Regions

In Section 3.2 we have discussed three bootstrap-based techniques for obtaining a series of point estimates. Regardless of the method used to generate the series, we will henceforth denote these series as $\hat{\boldsymbol{\pi}}_1, \ldots, \hat{\boldsymbol{\pi}}_m$, where $m$ represents the total number of estimates, assuming an abuse of notation for simplicity. In this section, we turn to discuss methods for constructing confidence regions based on these estimates. All confidence regions are constructed around the final estimate $\underline{\hat{\boldsymbol{\pi}}} = \frac{1}{m}\sum_{i=1}^{m} \hat{\boldsymbol{\pi}}_i$ with a certain confidence value $\alpha$. We will denote $\mathrm{CR}_\alpha$ a generic confidence region constructed around the mean estimate at confidence level $\alpha$.

**Confidence Intervals** Perhaps the simplest way to derive confidence regions are the confidence intervals (CI) generated via bootstrap quantiles.

This method is simple and non-parametric. The idea is to consider each class prevalence as an independent random variable of unknown distribution, and derive their confidence intervals using the empirical distribution generated via bootstrap.

We will therefore consider the (independent) random variable $P_i$ which takes on values $[\hat{\boldsymbol{\pi}}_1]_i, \ldots, [\hat{\boldsymbol{\pi}}_m]_i$, where $[\cdot]_i$ is the operator that returns the prevalence of the $i$th class in the input vector.

For each such random variable we arrange the bootstrap estimates in ascending order to obtain their empirical distribution, and identify the lower ($\mathbf{L}_i$) and upper ($\mathbf{U}_i$) bounds at confidence $(1-\alpha)\%$ (e.g., at 95%) as the $\left(\frac{\alpha}{2}\right)$ percentile (e.g., 2.5%) and the $\left(100 - \frac{\alpha}{2}\right)$ percentile (e.g., 97.5%) of the sorted distribution. For instance, given a dataset with a prevalence estimate of 0.65 and a bootstrap-derived distribution of prevalence values ranging from 0.55 to 0.75, the 95% confidence interval would be [0.56, 0.74].

Given the bounds $\mathbf{L}_i, \mathbf{U}_i$ $(1 \leq i \leq n)$ for all classes, we can determine whether a prevalence vector $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_n\}$ lies within the confidence region by computing:

$$\mathrm{CI}_\alpha(\boldsymbol{\pi}) = \begin{cases} 1: & \mathbf{L}_i \leq \pi_i \leq \mathbf{U}_i \;, \forall i, 1 \leq i \leq n \\ 0: & \text{otherwise} \end{cases} \tag{1}$$

Despite its simplicity, this method presents some drawbacks. First, it assumes the random variables are independent; however, we know the random variables take on values drawn from a more complex structure, the probability simplex, which imposes certain contraints. Second, the confidence region forms a hyper-rectangle (with each dimension representing the confidence interval of one variable) which intersects but also extends outside the probability simplex (see Section 3.4).

**Confidence Ellipse in $\boldsymbol{\Delta^{n-1}}$** A typical procedure for deriving confidence regions in multivariate spaces is to define a confidence (hyper-)ellipse (CE) around the point estimate $\underline{\hat{\boldsymbol{\pi}}}$.

A typical procedure for checking whether a prevalence vector $\boldsymbol{\pi}$ belongs to the confidence ellipse around $\underline{\hat{\boldsymbol{\pi}}}$ comes down to assuming $\hat{\boldsymbol{\pi}}_i \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, with $\boldsymbol{\mu}$ the (unknown) population mean which is estimated as $\underline{\hat{\boldsymbol{\pi}}}$, and $\Sigma$ the (unknown) population covariance matrix which is estimated with the sample covariance matrix $S$, and then check whether:

$$\mathrm{CE}_\alpha(\boldsymbol{\pi}) = \begin{cases} 1: & (\boldsymbol{\pi} - \underline{\hat{\boldsymbol{\pi}}})^\top S^{-1} (\boldsymbol{\pi} - \underline{\hat{\boldsymbol{\pi}}}) \leq \chi^2_{(n-1)}(1-\alpha) \\ 0: & \text{otherwise} \end{cases} \tag{2}$$

where $\chi^2$ is the chi-squared distribution, $S^{-1}$ is the inverse of the covariance matrix $S$ (aka the *precision* matrix), and $\alpha$ is the desired confidence level (e.g., $\alpha = 0.05$ for 95%).

While this method guarantees that the ellipse lies on the probability simplex (more on this in Section 3.4), it rests upon the incorrect assumption that the prevalence estimates are normally distributed. We know all our estimates lie on the probability simplex, which clashes with this assumption.

**Confidence Ellipse in the CLR-space** Confidence intervals and confidence ellipses do not take into account the inner geometry of the probability simplex space (i.e., all components must be positive and sum up to one). A more sophisticated way to derive confidence regions comes down to mapping the prevalence estimates onto an unconstrained Euclidean space using the Centered Log-Ratio (CLR) transformation. We call CT this confidence region in the transformed space. The CLR transformation $T : \Delta^{n-1} \to \mathbb{R}^n$ is given by:

$$T(\boldsymbol{\pi}) = \left[\log \frac{\pi_1}{g(\boldsymbol{\pi})}, \ldots, \log \frac{\pi_n}{g(\boldsymbol{\pi})}\right]$$

where $g(\boldsymbol{\pi})$ is the the geometric mean of $\boldsymbol{\pi}$.

We can define the confidence ellipse around the new center $\underline{\boldsymbol{\pi}}' = \frac{1}{m}\sum_{i=1}^{m} T(\boldsymbol{\pi}_i)$ as the region containing all (CLR-transformed) points verifying:

$$\mathrm{CT}_\alpha(\boldsymbol{\pi}) = \begin{cases} 1: & (T(\boldsymbol{\pi}) - \underline{\boldsymbol{\pi}}')^\top (S')^{-1} (T(\boldsymbol{\pi}) - \underline{\boldsymbol{\pi}}') \leq \chi^2_{(n-1)}(1-\alpha) \\ 0: & \text{otherwise} \end{cases} \tag{3}$$

where $S'$ is the sample covariance matrix of the CLR-transformed estimates $T(\boldsymbol{\pi}_1), \ldots, T(\boldsymbol{\pi}_n)$, and $\chi^2$ is the chi-squared distribution, as before. Note that the degrees of freedom of the distribution is still $(n-1)$ since the original components are constrained.

## 3.4 Goodness of the Confidence Region

Confidence regions are typically evaluated in terms of the *coverage* ($\mathcal{C}$) and their *amplitude* ($\mathcal{A}$).

The coverage measures the fraction of experiments in which the true prevalence value was contained in the confidence region (the higher, the better), and is simply computed as:

$$\mathcal{C} = \frac{\sum_{i=1}^{E} \mathrm{CR}_\alpha(\boldsymbol{\pi}_i^*)}{E} \tag{4}$$

with $E$ the total number of experiments, and $\mathrm{CR}_\alpha$ any of the regions discussed in Section 3.3 (Equations 1, 2, and 3), independently generated for the training set $L$ and test set $U$ of each experiment.

The amplitude measures the portion of the probability simplex that is included in the confidence region (the smaller, the better), and is computed as:

$$\mathcal{A} = \frac{V(\mathrm{CR}_\alpha)}{V(\Delta^{n-1})} \tag{5}$$

with $V$ a function measuring the *volume* of a space.

While computing the volume of the simplex is trivial, and comes down to calculating $V(\Delta^{n-1}) = \frac{1}{(n-1)!}$, computing the volume of a confidence region is not that easy. For example, one might be tempted to compute $V(\mathrm{CI}_\alpha)$ simply as $\prod_{i=1}^{n}(\mathbf{U}_i - \mathbf{L}_i)$. However, this would be misleading, since the volume we are computing corresponds to a region that does not lie within the simplex, but in $[0,1]^n$ (the region intersects with the simplex, but covers also part of the space outside the simplex).

Similarly, one might be tempted to compute

$$V(\mathrm{CE}_\alpha) = \frac{\pi^{\frac{n-1}{2}}}{\Gamma(\frac{n-1}{2}+1)} \prod_{i=1}^{n-1} \sqrt{\lambda_i \cdot \chi^2_{n-1}(1-\alpha)} \tag{6}$$

with $\pi$ the pi constant (not to be confused with a prevalence value), $\lambda_i$ the $i$th eigenvalue of the covariance matrix.[3] Again, this would be incorrect. The reason is that a confidence ellipse generated from points on a simplex may not be entirely confined within the simplex. While the centre of the ellipse is guaranteed to lie within the simplex, the ellipse's edges could well extend beyond the simplex's boundaries; see Figure 1.

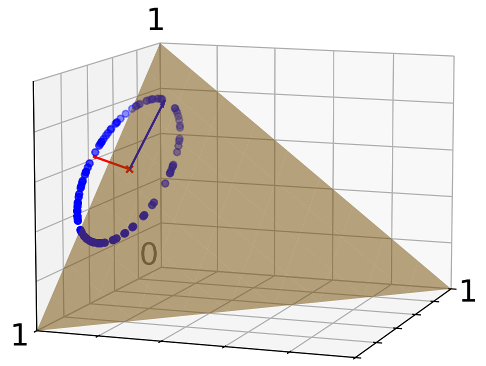

Fig. 1: Ellipse in a 2-dimensional simplex. The center is contained in the simplex, but part of the ellipse lies outside it.

Finally, note that $V(\mathrm{CT}_\alpha)$ would effectively account for the volume of the ellipse in the CLR-space, but there is no simple way to transform back such scalar to the original space.

We therefore propose a simple, yet effective method, to approximate the amplitude, based on Monte Carlo sampling. To this end, we generate $t = 10,000$ random prevalence points from the probability simplex, uniformly at random, and compute the fraction of these that belong to the region. That is:

$$\mathcal{A} \approx \frac{\sum_{i=1}^{t} \mathrm{CR}_\alpha(\tilde{\boldsymbol{\pi}}_i)}{t}\ , \tilde{\boldsymbol{\pi}}_i \sim \mathrm{Dir}(\mathbf{1}_n) \tag{7}$$

with $\mathbf{1}_n$ an $n$-dimensional vector of ones, the concentration parameter of the Dirichlet distribution. This method is generic, and is valid for $\mathrm{CI}_\alpha$, $\mathrm{CE}_\alpha$, and $\mathrm{CT}_\alpha$.

## 4 Experiments

In this section, we turn to describe the experiments we have carried out in order to evaluate our methods for deriving confidence regions.

[3] Only $(n-1)$ such eigenvalues are considered, since the last one is 0, given that the ellipse lies in an $(n-1)$-dimensional space, and thus has one null semiaxis corresponding to the $n$th dimension.

## 4.1 Datasets

The datasets we have used are taken from the publicly available UCI Machine Learning repository[4], and consists of 26 binary datasets and 24 multiclass datasets. The statistics of these datasets can be consulted online.[5]

For each dataset, we randomly extract 70% of the data for training our quantifiers, and use the remaining 30% for testing them. In the few cases in which the training set exceeds 25,000 instances, we randomly use 25,000 for training and the rest for testing.

Each value reported in the tables is the aggregation of 100 test samples of 500 instances each, drawn at random with the artificial prevalence protocol (APP) described in [22]. This protocol works by uniformly sampling prevalence values from the probability simplex, and then extracting, from the test pool, bags realizing these prevalence values. The APP protocol thus generates samples at widely varying prevalence values effectively representing cases of prior probability shift.

## 4.2 Evaluation metrics

In order to evaluate the quality of the quantifiers' predictions, we adopt the Absolute Error (AE) of the prevalence predictions with respect to the true prevalence values. AE is given by:

$$\mathrm{AE}(\boldsymbol{\pi}^*, \hat{\boldsymbol{\pi}}) = \frac{1}{n}\sum_{i=1}^{n} |\pi_i^* - \hat{\pi}_i|$$

we report the Mean Absolute Error (MAE) across all experiments.

We evaluate the quality of the confidence regions by means of the coverage ($\mathcal{C}$, see Equation 4), i.e., the proportion of experiments in which the true prevalence was contained in the confidence region, and the amplitude ($\mathcal{A}$, see Equation 7), a measure of the proportion of the simplex' volume that the region includes; see Section 3.4 for further details.

A good confidence region is one that displays high coverage and small amplitude. There is, to the best of our knowledge, no standard measure for combining both metrics into a single one. We propose a measure of goodness that we call Log-Ratio Goodness (LRG) and that we define as:

$$\mathrm{LRG}(\mathrm{CR}_\alpha) = \log\left(1 + \frac{\mathcal{C}}{\mathcal{A} + \epsilon}\right) \tag{8}$$

with $\epsilon$ a small value to prevent zero division in degenerated null regions.

In all cases, we consider confidence regions at a confidence level of $\alpha = 0.05$.

[4] `https://archive.ics.uci.edu`

[5] `https://hlt-isti.github.io/QuaPy/manuals/datasets.html#uci-machine-learning`

## 4.3 Quantification methods

We choose three representative aggregative quantifiers for testing the confidence regions (but any aggregative quantifier can be used in place):

- ACC: Adjusted Classify and Count [11] uses a crisp classifier $h$ to compute an estimate of the class priors (by classifying and counting), which is then corrected by taking into consideration the matrix of the misclassification rates of the classifier $M_h$. The adjustment procedure (in which model-based bootstrap is applied) thus amounts to constructing the misclassification rates matrix $M_h$, while the aggregation phase (in which the population-based bootstrap is applied) consists of correcting the estimate by inverting the misclassification rates matrix.
- PACC: Probabilistic Adjusted Classify and Count [2] is the probabilistic counterpart of ACC, in which the classifier $h$ is a soft one, and in which the counts and the misclassification rates matrix are computed on soft counts (posterior probabilities).
- DM: Distribution Matching is a multiclass extension, as proposed by [10,3], of the Hellinger Distance y (HDy) method proposed by [15]. In a nutshell, DM-based approaches try to reconstruct the distribution of the test datapoints by looking for the closest mixture of the class-conditional distributions of the training datapoints. Distributions are modelled as independent univariate histograms. The mixture parameter yielding the best such matching, in terms of the HD is an estimate of the sought class prevalence values.

In all cases we use Logistic Regression as the underlying classifier. Logistic Regression has become very popular in the quantification literature for its ability to produce well-calibrated posterior probabilities and due to the fact that is has shown good performance across a large variety of problems, a fundamental aspect for obtaining good confidence regions [24].

We took the implementations of these methods made available in the `QuaPy` software package [20]. Our implementations of the confidence regions and bootstrap methods as well as all the required scripts to reproduce our experiments can be accessed via GitHub.[6] The method has been integrated into QuaPy and will be made available in the upcoming v0.2.0; a pre-release is available at the `devel` branch [7].

## 4.4 Results

In this section we report on the results we have obtained. In the tables to come we use the following conventions. Each table reports the quantification errors (MAE, lower is better), accompanied by the coverage ($\mathcal{C}$, higher is better) and their amplitude ($\mathcal{A}$, lower is better); bold indicates the best value in each case. A summary of all the binary and multiclass results is reported in Tables 1 and

[6] `https://github.com/AlexMoreo/ConfidenceIntervalsQuantification`
[7] `https://github.com/HLT-ISTI/QuaPy/blob/devel/quapy/method/confidence.py`

Table 1: Summary of binary results grouped by bootstrap and confidence region. Symbol ↑ marks metrics for which the higher the better, while ↓ indicates the lower the better.

| Bootstrap | Conf. region | MAE ↓ | $\mathcal{C}$ ↑ | $\mathcal{A}$ ↓ | LRG ↑ |
|---|---|---|---|---|---|
| None | None | 0.057 | – | – | – |
| population-based | CI | 0.055 | 0.710 | **0.137** | 1.617 |
| | CE | 0.055 | 0.643 | 0.202 | 1.298 |
| | CT | 0.055 | 0.601 | 0.176 | 1.221 |
| model-based | CI | **0.054** | **0.871** | 0.210 | **1.767** |
| | CE | **0.054** | 0.744 | 0.255 | 1.363 |
| | CT | **0.054** | 0.662 | 0.263 | 1.147 |
| combined | CI | 0.055 | **0.871** | 0.211 | **1.767** |
| | CE | 0.055 | 0.747 | 0.258 | 1.357 |
| | CT | 0.055 | 0.627 | 0.245 | 1.096 |

Table 2: Summary of multiclass results grouped by bootstrap and confidence region. Symbol ↑ marks metrics for which the higher the better, while ↓ indicates the lower the better.

| Bootstrap | Conf. region | MAE ↓ | $\mathcal{C}$ ↑ | $\mathcal{A}$ ↓ | LRG ↑ |
|---|---|---|---|---|---|
| None | None | 0.054 | – | – | – |
| population-based | CI | 0.053 | 0.560 | **0.033** | 5.148 |
| | CE | 0.053 | 0.611 | 0.052 | 4.666 |
| | CT | 0.053 | 0.641 | 0.093 | 4.083 |
| model-based | CI | **0.048** | 0.723 | 0.076 | 5.765 |
| | CE | **0.048** | 0.772 | 0.100 | 5.141 |
| | CT | **0.048** | **0.807** | 0.209 | 4.145 |
| combined | CI | **0.048** | 0.726 | 0.076 | **5.769** |
| | CE | **0.048** | 0.753 | 0.093 | 4.833 |
| | CT | **0.048** | 0.789 | 0.205 | 3.809 |

2, in which all methods and datasets have been aggregated for the reader's convenience. The detailed tables displaying the individual performance of each quantification method in each dataset can be consulted in the additional material.

These results reveal some interesting findings. First, that in most cases, the point estimate generated by averaging across all bootstrap estimates often improves over the raw estimate obtained by the original method, which directly provides one single estimate. However, this is not always true if we take a closer look at the individual experiments (see the additional material), and there is one case (DM in the binary setting) where the original estimate is better, in terms of MAE, than those provided through bootstrap. In other cases, despite being worse, the results are statistically similar (3 cases out of 6).

Regarding the different ways for applying bootstrap (population-based, model-based, or combined), it seems that, in terms of MAE, there are no significant differences on the quality of the final estimates in the binary setup. However, the model-based and the combined approach seems to deliver slightly better results in the multiclass case, although the differences in performance are not statistically significant.

In terms of coverage ($\mathcal{C}$), the best bootstrap strategy seems to be the model-based one, while there is no clear winner within the various strategies for deriving confidence regions. What clearly emerges from the results is that there is a strict order CI $\succ$ CE $\succ$ CT in the binary setup (with A $\succ$ B meaning "A outperforms B"), while this order is reversed in the multiclass case.

While it may seem counter-intuitive that the coverage consistently falls below the nominal level of $1 - \alpha = 0.95$, this stems from a mismatch between the assumed distributional framework (prior probability shift) and the i.i.d. assumption underlying the standard bootstrap. As a result, resampling methods such as the bootstrap must be interpreted with caution. In practice, the bootstrap can approximate the variability of a quantifier with respect to the observed test sample, but it does not account for the additional uncertainty induced by distributional shift. In other words, it fails to fully reproduce the different sources of variability. Bootstrap intervals therefore reflect the internal stability of the quantifier given the available data, but they should not be interpreted as providing a formal coverage guarantee for the true prevalence. We are currently investigating alternative bootstrap schemes aimed at incorporating distributional shift into the variability estimation, with the goal of achieving more reliable coverage properties.

In terms of amplitude ($\mathcal{A}$) we observe a tendency towards CI $\succ$ CE $\succ$ CT. While there are few exceptions to this, CI always stands out in its ability to display low amplitude.

The combined evaluation of confidence regions LRG clearly indicates CI is the top performing method for deriving confidence regions with good coverage and small amplitude, simultaneously.

Overall, the experiments we have carried out seem to indicate the best strategy for generating confidence regions around the point estimate is the approach that combines both population-based and model-based bootstrap methods, paired with confidence intervals.

### 4.5 Execution times

A novel aspect of our approach resides in the fact that bootstrap is only applied to the pre-classified instances that concur in training the aggregation function (thus avoiding the cost of training different classifiers), and/or to pre-classified instances that are given as input to the aggregation function (thus avoiding the cost of classifying each bootstrap sample).

In this section, we report clocked times to showcase the improvements in efficiency that our proposal brings about with respect to a naive implementation of bootstrap in quantification, i.e., with respect to an implementation that, for

Table 3: Summary of clocked times in binary experiments grouped by bootstrap method.

| Bootstrap | Implem. | Train time (s) | (%Rel.Red) | Test time (s) | (%Rel.Red) |
|---|---|---|---|---|---|
| None | `QuaPy` | 0.023 | – | 0.003 | – |
| population-based | Naive | 0.023 | – | 1.681 | – |
| | Ours | 0.023 | (=) | 1.311 | (-22.02%) |
| model-based | Naive | 11.461 | – | 1.681 | – |
| | Ours | 0.287 | (-97.50%) | 1.360 | (-19.10%) |
| combined | Naive | 2.292 | – | 33.615 | – |
| | Ours | 0.073 | (-96.83%) | 27.404 | (-18.48%) |

Table 4: Summary of clocked times in multiclass experiments grouped by bootstrap method.

| Bootstrap | Implem. | Train time (s) | (%Rel.Red) | Test time (s) | (%Rel.Red) |
|---|---|---|---|---|---|
| None | `QuaPy` | 0.638 | – | 0.010 | – |
| population-based | Naive | 0.638 | – | 4.798 | – |
| | Ours | 0.638 | (=) | 4.666 | (-2.75%) |
| model-based | Naive | 318.760 | – | 4.798 | – |
| | Ours | 3.742 | (-98.83%) | 4.509 | (-6.03%) |
| combined | Naive | 63.752 | – | 95.962 | – |
| | Ours | 1.535 | (-97.59%) | 93.381 | (-2.69%) |

every bootstrap sample generated in a model-based approach, it undergoes the total cost associated with training a classifier and an aggregation function, and that for every bootstrap sample generated in a population-based approach, it undergoes the total cost of classifying all instances and aggregating them. We call this implementation “Naive”.

Tables 3 and 4 report the times we have clocked in on a desktop computer equipped with a 12th Gen Intel(R) i9-12900K processor and 64GB of RAM, running Ubuntu 22. Recall that, for these experiments, we are resampling 500 times during training with the model-based approach, 500 times during test with the population-based; for the combined approach we resample 100 times during training and 100 times during test (thus predicting $100\times100=10{,}000$ estimates). We also display the percentage of relative reduction with respect to the naive approach.

Overall, these tables clearly show that there are important gains in efficiency with respect to a naive implementation. These gains are more marked for the model-based and combined training times. The reason is that training the classifier is the most time-consuming task, which is carried out once and for all in our implementation. There are significant reductions also at inference time; however, these reductions appear less marked than the training ones, since the classification of the instances (that our methods undergoes only once) is far less costly than training a classifier.

## 5 Conclusions

In this paper, we have proposed a methodology for deriving confidence regions around class prevalence estimates. This methodology relies on bootstrap, and is aimed at circumventing the computational complexity this strategy entails. To do so, we have focused on a specific type of quantification methods called "aggregative". These methods carry out the training and inference phases in two stages, the first of which is considerably slower than the other. We exploit this characteristic to apply the bootstrap resampling only in the second phase of the training or inference phases, thus speeding up the otherwise unaffordable cost.

We have empirically evaluated three strategies for applying bootstrap: the model-based approach, which applies resampling to the training set; the population-based approach, which applies resampling to the test set; and a combined approach that applies bootstrap both to the training and test sets. We have also investigated three ways for deriving confidence regions, including traditional confidence intervals, confidence ellipses in the simplex, and confidence ellipses in a transformed space. Our findings suggest that the combined approach paired with confidence intervals is the most effective method for deriving confidence regions. Practitioners should consider this approach when computational efficiency is a priority and when robustness under prior probability shift is required.

By construction, standard bootstrap methods rely on the i.i.d. assumption and therefore cannot provide valid coverage guarantees under prior probability shift. In such cases, bootstrap intervals merely quantify the internal variability of the quantifier on the given sample, while the additional uncertainty induced by distributional shift is not captured. In the near future, we plan to investigate other resampling schemes that explicitly incorporate this shift as a promising direction for achieving more reliable coverage properties.

## Acknowledgments

This work has been funded by the QuaDaSh project "*Finanziato dall'Unione europea—Next Generation EU, Missione 4 Componente 2 CUP B53D23026250001*".

## References

1. Beijbom, O., Hoffman, J., Yao, E., Darrell, T., Rodriguez-Ramirez, A., Gonzalez-Rivero, M., Hoegh-Guldberg, O.: Quantification in-the-wild: Data-sets and baselines (2015), coRR abs/1510.04811 (2015). Presented at the NIPS 2015 Workshop on Transfer and Multi-Task Learning, Montreal, CA
2. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Quantification via probability estimators. In: Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2010). pp. 737–742. Sydney, AU (2010). `https://doi.org/10.1109/icdm.2010.75`

3. Bunse, M.: Unification of algorithms for quantification and unfolding. INFORMATIK 2022 (2022). https://doi.org/10.18420/inf2022_37
4. Daughton, A.R., Paul, M.J.: Constructing accurate confidence intervals when aggregating social media data for public health monitoring. In: Proceedings of the 3rd AAAI International Workshop on Health Intelligence (W3PHIAI 2019). pp. 9–17. Phoenix, US (2019)
5. Denham, B., Lai, E.M., Sinha, R., Naeem, M.A.: Gain-some-lose-some: Reliable quantification under general dataset shift. In: 2021 IEEE International Conference on Data Mining (ICDM). pp. 1048–1053. IEEE (2021)
6. Efron, B., Tibshirani, R.J.: An introduction to the bootstrap (1993)
7. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to quantify. Springer Nature, Cham, CH (2023). https://doi.org/10.1007/978-3-031-20467-8
8. Esuli, A., Moreo, A., Sebastiani, F., Sperduti, G.: A detailed overview of LeQua 2022: Learning to quantify. In: Working Notes of the 13th Conference and Labs of the Evaluation Forum (CLEF 2022). Bologna, IT (2022)
9. Fernandes Vaz, A., Izbicki, R., Bassi Stern, R.: Quantification under prior probability shift: The ratio estimator and its extensions. Journal of Machine Learning Research **20**, 79:1–79:33 (2019)
10. Firat, A.: Unified framework for quantification (2016)
11. Forman, G.: Counting positives accurately despite inaccurate classification. In: Proceedings of the 16th European Conference on Machine Learning (ECML 2005). pp. 564–575. Porto, PT (2005). https://doi.org/10.1007/11564096_55
12. Gao, W., Sebastiani, F.: From classification to quantification in tweet sentiment analysis. Social Network Analysis and Mining **6**(19), 1–22 (2016). https://doi.org/10.1007/s13278-016-0327-z
13. González, P., Castaño, A., Chawla, N.V., del Coz, J.J.: A review on quantification learning. ACM Computing Surveys **50**(5), 74:1–74:40 (2017). https://doi.org/10.1145/3117807
14. González, P., Moreo, A., Sebastiani, F.: Binary quantification and dataset shift: an experimental investigation. Data Mining and Knowledge Discovery pp. 1–43 (2024)
15. González-Castro, V., Alaiz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the Hellinger distance. Information Sciences **218**, 146–164 (2013). https://doi.org/10.1016/j.ins.2012.05.028
16. Hopkins, D.J., King, G.: A method of automated nonparametric content analysis for social science. American Journal of Political Science **54**(1), 229–247 (2010). https://doi.org/10.1111/j.1540-5907.2009.00428.x
17. Keith, K.A., O'Connor, B.: Uncertainty-aware generative models for inferring document class prevalence. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018). Brussels, BE (2018)
18. King, G., Lu, Y., Shibuya, K.: Designing verbal autopsy studies. Population Health Metrics **19**(8) (2010). https://doi.org/10.1186/1478-7954-8-19
19. Meeker, W.Q., Hahn, G.J., Escobar, L.A.: Statistical intervals: a guide for practitioners and researchers, vol. 541. John Wiley & Sons (2017)
20. Moreo, A., Esuli, A., Sebastiani, F.: QuaPy: A Python-based framework for quantification. In: Proceedings of the 30th ACM International Conference on Knowledge Management (CIKM 2021). pp. 4534–4543. Gold Coast, AU (2021). https://doi.org/10.1145/3459637.3482015
21. Moreo, A., González, P., del Coz, J.J.: Kernel density estimation for multiclass quantification. Machine Learning **114**(4) (2025). https://doi.org/10.1007/s10994-024-06726-5

22. Moreo, A., Sebastiani, F.: Tweet sentiment quantification: An experimental re-evaluation. PLOS ONE **17**(9), 1–23 (September 2022). `https://doi.org/10.1371/journal.pone.0263449`
23. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods (2023)
24. Tasche, D.: Confidence intervals for class prevalences under prior probability shift. Machine Learning and Knowledge Extraction **1**(3), 805–831 (2019). `https://doi.org/10.3390/make1030047`
25. Ziegler, A., Czyż, P.: Bayesian quantification with black-box estimators (2023), `https://arxiv.org/abs/2302.09159`

## A Additional Material

In this section, we report, in disaggregated form, the results summarized in Tables 1 and 2 of the paper "An Efficient Method for Deriving Confidence Intervals in Aggregative Quantification".

The notational conventions we used are as follows. For each dataset and each metric, we highlight in **boldface** the best results. For the sake of a rapid interpretation of the results, we highlight, independently for each evaluation metric, the best result in intense green and the worst result in intense red; the rest of the results are interpolated between both extremes. All results are confronted against the best one by means of a Wilcoxon signed-rank test. We use superscripts † and ‡ to indicate the methods (if any) whose scores are *not* statistically significantly different from the best one at different confidence levels: symbol † indicates $0.001 < p < 0.01$, while symbol ‡ indicates $p \geq 0.01$. Each table is devoted to one quantification method, and displays the results of the original quantification method (i.e., one that does not provide confidence regions), only in terms of MAE, and all combinations of bootstrap methods (population-based, model-based, and combined) with the tree methods for generating confidence regions (CI, CE, and CT), in terms of MAE, $\mathcal{C}$, $\mathcal{A}$.

Tables 5a, 5b, and 5c report the results we have obtained for ACC, PACC, and DM, in the binary datasets, respectively; while Tables 6a, 6b, and 6c report the results we have obtained for ACC, PACC, and DM, in the multiclass datasets, respectively. The goodness measures for ACC, PACC, and DM are reported in Tables 7a, 7b, and 7c for the binary datasets, and in Tables 8a, 8b, and 8c for the multiclass case.

| | ACC | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | Population-based | | | | | | | | | Model-based | | | | | | | | | Combined | | | | | | | | |
| | - | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | |
| | MAE | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ |
| balance.1 | .028 | **.023** | .82 | **.08** | .024$^\ddagger$ | .77 | .12 | **.023** | .81 | .10 | .029 | .84 | .09 | .029 | .76 | .13 | .029 | .72 | .12 | .029 | **.85** | .09 | .029 | .76 | .13 | .029 | .66 | .12 |
| balance.3 | .021 | .014$^\ddagger$ | .95 | .08 | **.014** | .91 | .11 | .014$^\ddagger$ | .79 | **.07** | .020 | .97 | .09 | .020 | .86 | .13 | .020 | .76 | .13 | .020 | **.98** | .10 | .020 | .82 | .13 | .020 | .72 | .10 |
| breast-cancer | .020 | .028 | .70 | **.07** | .028 | .71 | .12 | .028 | .71 | .10 | **.020** | .88 | .09 | .020$^\ddagger$ | .77 | .13 | **.020** | .81 | .17 | .020$^\ddagger$ | **.91** | .09 | .020$^\ddagger$ | .75 | .12 | .020$^\ddagger$ | .78 | .16 |
| cmc.1 | .110 | .122 | .53 | **.24** | .122 | .53 | .34 | .122 | .57 | .35 | .106$^\ddagger$ | **.80** | .30 | **.106** | .65 | .37 | .106$^\ddagger$ | .65 | .47 | .113 | .76 | .30 | .113 | .63 | .37 | .113 | .57 | .46 |
| cmc.2 | .171 | .133$^\ddagger$ | .84 | **.40** | **.132** | .81 | .46 | .133$^\ddagger$ | .63 | .52 | .158 | **.91** | .54 | .158 | .72 | .49 | .158 | .70 | .67 | .167 | .88 | .53 | .167 | .74 | .52 | .167 | .58 | .57 |
| cmc.3 | .187 | .243 | .37 | **.36** | .243 | .33 | .43 | .243 | .45 | .42 | **.168** | .85 | .54 | .168$^\ddagger$ | .65 | .48 | **.168** | .49 | .49 | .168$^\ddagger$ | **.88** | .54 | .169$^\ddagger$ | .65 | .49 | .168$^\ddagger$ | .56 | .54 |
| german | .080 | .082 | .66 | **.20** | .083 | .59 | .29 | .082 | .58 | .30 | **.077** | .83 | .28 | .078$^\ddagger$ | .74 | .35 | **.077** | .64 | .38 | .079 | **.84** | .28 | .079 | .66 | .35 | .079 | .65 | .40 |
| haberman | .149 | .102$^\ddagger$ | .85 | **.42** | .102$^\ddagger$ | .76 | .47 | .102$^\ddagger$ | .56 | .47 | .096$^\ddagger$ | **1.00** | 1.00 | **.095** | .72 | .64 | .096$^\ddagger$ | .59 | .59 | .111 | **1.00** | 1.00 | .111 | .91 | .78 | .111 | .53 | .53 |
| ionosphere | .083 | .072$^\ddagger$ | .16 | **.10** | **.072** | .21 | .14 | .072$^\ddagger$ | .20 | .11 | .084 | .50 | .17 | .084 | .46 | .22 | .084 | .40 | .23 | .082 | .45 | .17 | .082 | **.53** | .23 | .082 | .43 | .24 |
| iris.1 | **.000** | .001 | **1.00** | **.07** | .001 | .89 | .10 | .001 | .88 | .07 | .001 | **1.00** | **.07** | .001 | .89 | .10 | .001 | .88 | .07 | .000 | **1.00** | .07 | .000 | .95 | .10 | .000 | .92 | .07 |
| iris.2 | .064 | .049$^\ddagger$ | .95 | **.24** | **.049** | .86 | .32 | .049$^\ddagger$ | .76 | .33 | .065 | **1.00** | .57 | .065 | .87 | .62 | .065 | .57 | .56 | .068 | **1.00** | .54 | .068 | .96 | .61 | .068 | .55 | .53 |
| iris.3 | .085 | .091 | .00 | **.08** | .091 | .01 | .20 | .091 | .03 | .09 | .085$^\ddagger$ | .01 | .14 | **.084** | .21 | .19 | .085$^\ddagger$ | **.34** | .22 | .087 | .05 | .15 | .087 | .19 | .20 | .087 | .29 | .19 |
| mammographic | .044 | .049 | .63 | **.12** | .049 | .60 | .16 | .049 | .58 | .12 | .043 | .90 | .15 | .043 | .78 | .21 | .043 | .67 | .19 | .041$^\ddagger$ | **.91** | .15 | **.041** | .81 | .21 | .041$^\ddagger$ | .61 | .19 |
| pageblocks.5 | .115 | **.081** | .53 | .15 | .081$^\ddagger$ | .51 | .23 | **.081** | .47 | **.15** | .112 | **.85** | .26 | .112 | .54 | .37 | .112 | .58 | .27 | .114 | .76 | .24 | .114 | .54 | .35 | .114 | .54 | .25 |
| sonar | .126 | .104$^\ddagger$ | .34 | **.13** | **.103** | .33 | .49 | .104$^\ddagger$ | .46 | .23 | .117$^\ddagger$ | .53 | .27 | .117$^\ddagger$ | .50 | .41 | .117$^\ddagger$ | .48 | .40 | .119$^\ddagger$ | .54 | .28 | .119$^\ddagger$ | **.56** | .39 | .119$^\ddagger$ | .53 | .42 |
| spambase | **.015** | .016 | **1.00** | **.09** | .016 | .93 | .12 | .016 | .90 | .12 | .016 | **1.00** | .09 | .016 | .90 | .13 | .016 | .78 | .14 | .016 | **1.00** | .09 | .016 | .88 | .13 | .016 | .73 | .13 |
| spectf | .109 | .060$^\ddagger$ | .81 | **.21** | **.060** | .66 | .28 | .060$^\ddagger$ | .62 | .28 | .095 | **.96** | .43 | .095 | .74 | .51 | .095 | .64 | .57 | .095 | .93 | .41 | .095 | .86 | .51 | .095 | .54 | .46 |
| tictactoe | .008 | **.006** | **1.00** | **.07** | .006$^\ddagger$ | .91 | .10 | **.006** | .87 | .09 | .007 | **1.00** | .08 | .008 | .83 | .11 | .007 | .78 | .11 | .008 | **1.00** | .08 | .008 | .88 | .11 | .008 | .77 | .10 |
| transfusion | .132 | .120$^\ddagger$ | .86 | **.48** | .120$^\ddagger$ | .74 | .52 | .120$^\ddagger$ | .59 | .55 | .117 | **1.00** | .64 | .117 | .82 | .60 | .117 | .61 | .58 | **.112** | **1.00** | .70 | .112$^\ddagger$ | .94 | .69 | **.112** | .68 | .66 |
| wdbc | .018 | .017$^\ddagger$ | .84 | .08 | **.017** | .77 | .11 | .017$^\ddagger$ | .71 | .08 | .018$^\ddagger$ | **.87** | .08 | .018$^\ddagger$ | .79 | .11 | .018$^\ddagger$ | .72 | .09 | .018$^\ddagger$ | .86 | .08 | .018$^\ddagger$ | .68 | .11 | .018$^\ddagger$ | .62 | **.07** |
| wine.1 | .006 | .019 | .80 | .07 | .019 | .79 | .13 | .019 | .69 | **.06** | **.006** | **1.00** | .07 | .006$^\ddagger$ | .93 | .10 | **.006** | .83 | .11 | .007 | **1.00** | .08 | .007 | .93 | .11 | .007 | .88 | .09 |
| wine.2 | .019 | .011$^\ddagger$ | .92 | **.07** | **.011** | .89 | .10 | .011$^\ddagger$ | .86 | .07 | .019 | **1.00** | .09 | .019 | .88 | .12 | .019 | .78 | .12 | .020 | **1.00** | .09 | .020 | .88 | .13 | .020 | .81 | .14 |
| wine.3 | .023 | .020$^\ddagger$ | .88 | **.07** | .020$^\ddagger$ | .84 | .11 | .020$^\ddagger$ | .85 | .08 | .022 | **1.00** | .10 | .022 | .87 | .14 | .022 | .74 | .14 | **.020** | **1.00** | .10 | .020$^\ddagger$ | .91 | .14 | **.020** | .77 | .15 |
| wine-q-red | .042 | .032$^\ddagger$ | **.97** | **.16** | **.032** | .92 | .22 | .032$^\ddagger$ | .78 | .26 | .039 | **.97** | .20 | .039 | .83 | .25 | .039 | .70 | .35 | .040 | **.97** | .20 | .040 | .87 | .25 | .040 | .67 | .30 |
| wine-q-white | .040 | .052 | .86 | **.20** | .053 | .85 | .27 | .052 | .71 | .28 | **.039** | **.98** | .21 | .040$^\ddagger$ | .80 | .28 | **.039** | .75 | .36 | .040$^\ddagger$ | **.98** | .21 | .040$^\ddagger$ | .88 | .28 | .040$^\ddagger$ | .73 | .33 |
| yeast | .158 | .130$^\ddagger$ | .41 | **.21** | **.130** | .35 | .29 | .130$^\ddagger$ | .47 | .31 | .156 | .49 | .29 | .155 | .43 | .36 | .156 | **.56** | .45 | .146 | **.56** | .30 | .146 | .45 | .38 | .146 | .49 | .39 |
| Mean | .071$^\ddagger$ | .065$^\ddagger$ | .72 | **.17** | **.065** | .67 | .24 | .065$^\ddagger$ | .64 | .22 | .066$^\ddagger$ | **.85** | .26 | .066$^\ddagger$ | .73 | .29 | .066$^\ddagger$ | .66 | .31 | .067$^\ddagger$ | .85 | .26 | .067$^\ddagger$ | .75 | .30 | .067$^\ddagger$ | .64 | .29 |

(a) Binary experiments for ACC.

| | PACC | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | Population-based | | | | | | | | | Model-based | | | | | | | | | Combined | | | | | | | | |
| | - | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | |
| | MAE | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ |
| balance.1 | **.024** | .033 | .67 | **.08** | .033 | .57 | .12 | .033 | .58 | .10 | .025 | **.94** | .09 | .025 | .81 | .13 | .025 | .73 | .13 | .025 | .91 | .09 | .025 | .78 | .13 | .025 | .70 | .10 |
| balance.3 | .026 | .023$^\ddagger$ | .90 | **.08** | **.022** | .76 | .11 | .023$^\ddagger$ | .74 | .08 | .026 | **1.00** | .09 | .025 | .84 | .13 | .026 | .74 | .11 | .028 | **1.00** | .09 | .028 | .76 | .13 | .028 | .75 | .13 |
| breast-cancer | .016 | **.015** | .84 | **.07** | .015$^\ddagger$ | .74 | .11 | **.015** | .77 | .11 | .016$^\ddagger$ | **.92** | .09 | .016$^\ddagger$ | .83 | .12 | .016$^\ddagger$ | .73 | .14 | .016$^\ddagger$ | .91 | .09 | .016$^\ddagger$ | .74 | .12 | .016$^\ddagger$ | .71 | .11 |
| cmc.1 | .075 | .079 | .70 | **.20** | .079 | .57 | .29 | .079 | .61 | .27 | .074$^\ddagger$ | **.86** | .26 | **.074** | .84 | .34 | .074$^\ddagger$ | .56 | .30 | .080 | .85 | .26 | .080 | .70 | .32 | .080 | .66 | .39 |
| cmc.2 | .047 | .054 | .88 | **.21** | .054 | .81 | .28 | .054 | .59 | .25 | **.045** | **.98** | .28 | .045$^\ddagger$ | .81 | .36 | **.045** | .67 | .39 | .045$^\ddagger$ | **.98** | .29 | .045$^\ddagger$ | .79 | .36 | .045$^\ddagger$ | .65 | .40 |
| cmc.3 | .125 | .104$^\ddagger$ | .67 | **.23** | **.104** | .53 | .31 | .104$^\ddagger$ | .52 | .31 | .121 | .68 | .30 | .121 | .54 | .38 | .121 | .50 | .37 | .120 | **.70** | .30 | .120 | .60 | .36 | .120 | .51 | .39 |
| german | .037 | .037$^\ddagger$ | .85 | **.15** | .036$^\ddagger$ | .71 | .21 | .037$^\ddagger$ | .65 | .23 | **.034** | .97 | .22 | .034$^\ddagger$ | .88 | .30 | **.034** | .67 | .35 | .035$^\ddagger$ | **.98** | .22 | .034$^\ddagger$ | .86 | .28 | .035$^\ddagger$ | .51 | .26 |
| haberman | .117 | .113 | .51 | **.23** | .113 | .39 | .41 | .113 | .47 | .33 | .098 | .94 | .47 | .098 | .73 | .41 | .098 | .49 | .46 | **.094** | **.97** | .50 | .094$^\ddagger$ | .84 | .44 | **.094** | .44 | .44 |
| ionosphere | .097 | .086$^\ddagger$ | .01 | **.09** | **.086** | .04 | .16 | .086$^\ddagger$ | .10 | .09 | .097 | .18 | .16 | .097 | .25 | .21 | .097 | **.29** | .19 | .098 | .13 | .16 | .098 | .28 | .21 | .098 | .26 | .19 |
| iris.1 | .005 | .013 | .90 | **.07** | .013 | .85 | .11 | .013 | .77 | .08 | .005$^\ddagger$ | **1.00** | .07 | **.005** | .85 | .10 | .005$^\ddagger$ | .78 | .11 | .005 | **1.00** | .07 | .005 | .79 | .10 | .005 | .80 | .09 |
| iris.2 | .062 | .047$^\ddagger$ | .74 | **.15** | **.047** | .63 | .22 | .047$^\ddagger$ | .58 | .21 | .054 | **1.00** | .38 | .054 | .85 | .42 | .054 | .65 | .56 | .058 | **1.00** | .37 | .058 | .88 | .44 | .058 | .55 | .47 |
| iris.3 | .011 | .008$^\ddagger$ | .98 | **.08** | **.008** | .88 | .11 | .008$^\ddagger$ | .80 | .10 | .011$^\ddagger$ | **1.00** | .13 | .011$^\dagger$ | .84 | .17 | .011$^\ddagger$ | .73 | .16 | .011$^\ddagger$ | **1.00** | .13 | .011$^\ddagger$ | .87 | .17 | .011$^\ddagger$ | .66 | .14 |
| mammographic | .037 | .044 | .62 | **.11** | .045 | .61 | .15 | .044 | .52 | .12 | .036 | **.93** | .14 | .037 | .82 | .20 | .036 | .73 | .20 | .035$^\ddagger$ | .92 | .14 | **.035** | .79 | .19 | .035$^\ddagger$ | .66 | .17 |
| pageblocks.5 | .029 | **.021** | .96 | **.12** | .022$^\ddagger$ | .83 | .16 | **.021** | .74 | .15 | .031 | **1.00** | .18 | .031 | .84 | .25 | .031 | .78 | .22 | .033 | **1.00** | .18 | .033 | .85 | .24 | .033 | .66 | .19 |
| sonar | .164 | .095$^\ddagger$ | .39 | **.12** | **.095** | .36 | .36 | .095$^\ddagger$ | .42 | .18 | .157 | .36 | .24 | .157 | .30 | .33 | .157 | .45 | .35 | .157 | .35 | .24 | .157 | .33 | .36 | .157 | **.46** | .36 |
| spambase | **.012** | .013 | .99 | **.08** | .014 | .86 | .12 | .013 | .70 | .11 | .012 | **1.00** | .09 | .012 | .90 | .12 | .012 | .76 | .15 | .012 | **1.00** | .09 | .012 | .91 | .12 | .012 | .73 | .12 |
| spectf | .085 | **.070** | .54 | **.15** | .070$^\ddagger$ | .51 | .21 | **.070** | .54 | .25 | .074 | **.96** | .33 | .074 | .73 | .40 | .074 | .70 | .54 | .073 | .94 | .32 | .073 | .74 | .39 | .073 | .56 | .43 |
| tictactoe | .014 | .013$^\ddagger$ | .94 | **.08** | **.013** | .79 | .12 | .013$^\ddagger$ | .78 | .10 | .014$^\ddagger$ | .94 | .08 | .014$^\ddagger$ | .76 | .12 | .014$^\ddagger$ | .73 | .10 | .014$^\dagger$ | **.96** | .08 | .014$^\dagger$ | .78 | .13 | .014$^\dagger$ | .67 | .10 |
| transfusion | .061 | .065 | .71 | **.18** | .065 | .62 | .27 | .065 | .53 | .26 | **.058** | **.99** | .28 | .058$^\ddagger$ | .74 | .37 | **.058** | .62 | .36 | .059 | **.99** | .28 | .059 | .86 | .36 | .059 | .61 | .30 |
| wdbc | .017 | .013$^\ddagger$ | **.94** | .07 | **.013** | .85 | .11 | .013$^\ddagger$ | .71 | **.07** | .017 | .87 | .08 | .017 | .75 | .11 | .017 | .68 | .11 | .017 | .86 | .08 | .017 | .72 | .11 | .017 | .62 | .09 |
| wine.1 | .005 | .015 | .81 | **.07** | .015 | .73 | .13 | .015 | .75 | .08 | .004$^\ddagger$ | **1.00** | .09 | .004$^\ddagger$ | .89 | .12 | .004$^\ddagger$ | .78 | .13 | .004$^\ddagger$ | **1.00** | .09 | **.004** | .86 | .13 | .004$^\ddagger$ | .77 | .12 |
| wine.2 | .017 | **.009** | .95 | **.07** | .009$^\ddagger$ | .80 | .11 | **.009** | .77 | .08 | .017 | **1.00** | .09 | .017 | .83 | .14 | .017 | .82 | .15 | .016 | **1.00** | .10 | .016 | .91 | .13 | .016 | .67 | .13 |
| wine.3 | .026 | .026$^\ddagger$ | .81 | **.07** | .026$^\ddagger$ | .67 | .12 | .026$^\ddagger$ | .72 | .08 | **.026** | **.99** | .09 | .026$^\ddagger$ | .84 | .13 | **.026** | .71 | .12 | .026$^\ddagger$ | **.99** | .09 | .026$^\ddagger$ | .75 | .13 | .026$^\ddagger$ | .73 | .13 |
| wine-q-red | .030 | .032$^\ddagger$ | .90 | **.14** | .032$^\ddagger$ | .80 | .19 | .032$^\ddagger$ | .68 | .22 | .029$^\ddagger$ | **.98** | .17 | **.029** | .87 | .23 | .029$^\ddagger$ | .72 | .30 | .029$^\ddagger$ | **.98** | .18 | .029$^\ddagger$ | .81 | .23 | .029$^\ddagger$ | .68 | .31 |
| wine-q-white | .037 | .062 | .70 | **.15** | .063 | .63 | .21 | .062 | .58 | .20 | **.036** | **.93** | .17 | .037$^\ddagger$ | .84 | .23 | **.036** | .61 | .21 | .037$^\ddagger$ | **.93** | .17 | .036$^\ddagger$ | .76 | .24 | .037$^\ddagger$ | .62 | .21 |
| yeast | .128 | .131 | .07 | **.16** | .131 | .06 | .25 | .131 | .19 | .19 | .128 | .33 | .21 | .128 | .31 | .29 | .128 | .35 | .26 | .122$^\ddagger$ | .37 | .21 | **.122** | **.39** | .29 | .122$^\ddagger$ | .37 | .22 |
| Mean | .050$^\ddagger$ | **.047** | .73 | **.12** | .047$^\ddagger$ | .64 | .19 | **.047** | .61 | .16 | .048$^\ddagger$ | **.88** | .18 | .048$^\ddagger$ | .75 | .23 | .048$^\ddagger$ | .65 | .25 | .048$^\ddagger$ | .87 | .19 | .048$^\ddagger$ | .74 | .24 | .048$^\ddagger$ | .62 | .23 |

(b) Binary experiments for PACC.

| | DM | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | Population-based | | | | | | | | | Model-based | | | | | | | | | Combined | | | | | | | | |
| | - | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | |
| | MAE | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ |
| balance.1 | .012 | .029 | .72 | **.08** | .029 | .67 | .13 | .029 | .66 | .11 | .011$^\ddagger$ | **.95** | .09 | **.011** | .83 | .13 | .011$^\ddagger$ | .75 | .11 | .011 | **.95** | .09 | .011 | .77 | .12 | .011 | .74 | .12 |
| balance.3 | .024 | .020$^\ddagger$ | .94 | .08 | **.019** | .82 | .12 | .020$^\ddagger$ | .69 | .08 | .024 | **.95** | .08 | .024 | .73 | .13 | .024 | .74 | .08 | .025 | .93 | .08 | .025 | .76 | .12 | .025 | .69 | **.07** |
| breast-cancer | .018 | **.015** | **.96** | **.07** | .015$^\ddagger$ | .84 | .10 | **.015** | .73 | .08 | .019 | .93 | .08 | .019 | .83 | .12 | .019 | .82 | .10 | .019 | .93 | .08 | .019 | .79 | .11 | .019 | .71 | .10 |
| cmc.1 | .069 | .079 | .72 | **.21** | .079 | .62 | .27 | .079 | .51 | .27 | .062$^\ddagger$ | **.93** | .26 | **.061** | .77 | .33 | .062$^\ddagger$ | .65 | .39 | .067 | .92 | .25 | .067 | .77 | .33 | .067 | .62 | .31 |
| cmc.2 | .053 | .053$^\ddagger$ | .87 | **.20** | **.053** | .81 | .28 | .053$^\ddagger$ | .68 | .31 | .054$^\ddagger$ | .96 | .27 | .054$^\ddagger$ | .84 | .35 | .054$^\ddagger$ | .58 | .38 | .055$^\ddagger$ | **.97** | .28 | .055$^\ddagger$ | .82 | .35 | .055$^\ddagger$ | .65 | .37 |
| cmc.3 | .105 | **.082** | .76 | **.23** | .082$^\ddagger$ | .65 | .30 | **.082** | .58 | .34 | .099 | .78 | .30 | .098 | .59 | .36 | .099 | .51 | .38 | .098 | **.79** | .31 | .098 | .63 | .37 | .098 | .46 | .36 |
| german | .063 | .063 | .62 | **.14** | .063 | .53 | .27 | .063 | .54 | .22 | .053$^\ddagger$ | .88 | .23 | **.052** | .66 | .30 | .053$^\ddagger$ | .61 | .33 | .053$^\ddagger$ | **.90** | .23 | .053$^\ddagger$ | .65 | .29 | .053$^\ddagger$ | .54 | .31 |
| haberman | .072 | **.049** | .90 | **.22** | .049$^\ddagger$ | .75 | .30 | **.049** | .61 | .23 | .074 | **1.00** | .54 | .074 | .80 | .60 | .074 | .69 | .64 | .066 | **1.00** | .51 | .066 | .88 | .62 | .066 | .47 | .42 |
| ionosphere | .084 | .065$^\ddagger$ | .09 | **.09** | **.065** | .21 | .15 | .065$^\ddagger$ | .22 | .10 | .075 | .53 | .16 | .075 | **.61** | .22 | .075 | .49 | .22 | .074 | .55 | .16 | .074 | .54 | .21 | .074 | .45 | .20 |
| iris.1 | .002 | .016 | **1.00** | .07 | .016 | .91 | .10 | .016 | .85 | **.07** | **.002** | **1.00** | .07 | .002$^\ddagger$ | .93 | .10 | **.002** | .84 | .08 | .003 | **1.00** | .07 | .003 | .87 | .10 | .003 | .77 | .07 |
| iris.2 | **.034** | .051 | .62 | **.12** | .051 | .57 | .21 | .051 | .60 | .18 | .038 | **1.00** | .34 | .038 | .80 | .43 | .038 | .70 | .40 | .040 | **1.00** | .34 | .040 | .80 | .44 | .040 | .55 | .33 |
| iris.3 | .051 | **.031** | .65 | **.08** | .031$^\ddagger$ | .64 | .13 | **.031** | .59 | .08 | .042 | **.99** | .13 | .042 | .81 | .18 | .042 | .74 | .14 | .042 | .98 | .12 | .042 | .79 | .17 | .042 | .64 | .13 |
| mammographic | **.037** | .043 | .63 | **.11** | .043 | .62 | .15 | .043 | .48 | .11 | .039 | .91 | .14 | .039 | .74 | .19 | .039 | .69 | .16 | .038 | **.92** | .14 | .038 | .72 | .19 | .038 | .61 | .13 |
| pageblocks.5 | .048 | **.042** | .71 | **.10** | .042$^\ddagger$ | .62 | .14 | **.042** | .62 | .11 | .060 | .83 | .14 | .060 | .65 | .20 | .060 | .60 | .17 | .061 | **.89** | .15 | .061 | .62 | .21 | .061 | .67 | .20 |
| sonar | .220 | **.135** | .09 | **.11** | .135$^\ddagger$ | .07 | .16 | **.135** | .10 | .11 | .205 | .01 | .25 | .205 | .08 | .33 | .205 | .30 | .30 | .208 | .01 | .26 | .208 | .05 | .35 | .208 | **.34** | .34 |
| spambase | .011 | .012$^\ddagger$ | **.99** | **.08** | .012$^\ddagger$ | .86 | .11 | .012$^\ddagger$ | .81 | .10 | **.011** | **.99** | .08 | .011$^\ddagger$ | .83 | .12 | **.011** | .78 | .10 | .011$^\ddagger$ | **.99** | .08 | .011$^\ddagger$ | .91 | .12 | .011$^\ddagger$ | .77 | .10 |
| spectf | **.090** | .208 | .01 | **.13** | .208 | .04 | .28 | .208 | .09 | .15 | .121 | **.97** | .37 | .121 | .77 | .47 | .121 | .65 | .49 | .126 | .93 | .36 | .126 | .74 | .44 | .126 | .69 | .49 |
| tictactoe | .014 | **.010** | **.99** | **.07** | .010$^\ddagger$ | .92 | .11 | **.010** | .87 | .09 | .013 | **.99** | .08 | .013 | .92 | .12 | .013 | .80 | .11 | .014 | **.99** | .08 | .014 | .89 | .13 | .014 | .73 | .14 |
| transfusion | **.074** | .107 | .33 | **.17** | .107 | .37 | .24 | .107 | .38 | .25 | .090 | **.93** | .28 | .091 | .76 | .37 | .090 | .65 | .41 | .093 | .86 | .27 | .093 | .70 | .35 | .093 | .55 | .37 |
| wdbc | **.012** | .017 | .89 | .07 | .017 | .78 | .11 | .017 | .67 | **.07** | .012 | **.92** | .08 | .012 | .81 | .12 | .012 | .72 | .09 | .012 | **.92** | .08 | .012 | .81 | .11 | .012 | .72 | .07 |
| wine.1 | **.027** | .065 | .10 | .07 | .065 | .11 | .11 | .065 | .10 | **.06** | .031 | **.99** | .09 | .031 | .82 | .13 | .031 | .76 | .08 | .031 | **.99** | .09 | .031 | .83 | .13 | .031 | .77 | .09 |
| wine.2 | **.009** | .012 | .98 | .07 | .012 | .84 | .12 | .012 | .68 | **.07** | .012 | **.99** | .09 | .012 | .83 | .13 | .012 | .71 | .10 | .011 | **.99** | .09 | .011 | .82 | .13 | .011 | .68 | .08 |
| wine.3 | .006 | .006 | **.99** | **.07** | .006 | .87 | .11 | .006 | .83 | .09 | .005 | **.99** | .08 | .005 | .90 | .11 | .005 | .77 | .11 | .004$^\ddagger$ | **.99** | .08 | **.004** | .86 | .11 | .004$^\ddagger$ | .67 | .07 |
| wine-q-red | .028 | .030$^\ddagger$ | .92 | **.14** | .030$^\ddagger$ | .83 | .19 | .030$^\ddagger$ | .67 | .19 | .026$^\ddagger$ | **1.00** | .17 | **.026** | .83 | .23 | .026$^\ddagger$ | .74 | .23 | .026$^\ddagger$ | **1.00** | .17 | .026$^\ddagger$ | .83 | .24 | .026$^\ddagger$ | .70 | .25 |
| wine-q-white | **.037** | .056 | .74 | **.16** | .057 | .62 | .22 | .056 | .59 | .22 | .038 | **.94** | .17 | .039 | .83 | .23 | .038 | .64 | .22 | .038 | **.94** | .17 | .038 | .80 | .23 | .038 | .60 | .19 |
| yeast | .075 | .070$^\ddagger$ | .52 | **.14** | .070$^\ddagger$ | .49 | .20 | .070$^\ddagger$ | .42 | .15 | .069 | .72 | .18 | .069 | .66 | .25 | .069 | .57 | .24 | **.066** | **.77** | .18 | .066$^\ddagger$ | .65 | .25 | **.066** | .49 | .21 |
| Mean | **.049** | .052$^\ddagger$ | .68 | **.12** | .053$^\ddagger$ | .62 | .18 | .052$^\ddagger$ | .56 | .15 | .049$^\ddagger$ | .89 | .18 | .049$^\ddagger$ | .76 | .24 | .049$^\ddagger$ | .67 | .23 | .050$^\ddagger$ | **.89** | .18 | .050$^\ddagger$ | .74 | .24 | .050$^\ddagger$ | .63 | .21 |

(c) Binary experiments for DM.

Table 5: Binary experiments for ACC, PACC, and DM.

| | ACC | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | Population-based | | | | | | | | | Model-based | | | | | | | | | Combined | | | | | | | | |
| | - | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | |
| | MAE | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ |
| hcv | .297 | .227$^\ddagger$ | .34 | .45 | .227$^\ddagger$ | .35 | .48 | .227$^\ddagger$ | .36 | **.36** | .219$^\ddagger$ | .69 | .83 | **.218** | .70 | .85 | .219$^\ddagger$ | **.80** | .85 | .219$^\ddagger$ | .68 | .81 | .220$^\ddagger$ | .66 | .78 | .219$^\ddagger$ | .79 | .85 |
| image_seg | .006 | .008 | .67 | **.00** | .008 | .82 | .00 | .008 | .85 | .01 | .006 | .92 | .00 | .006 | .92 | .02 | .006 | **.98** | .01 | **.006** | .90 | .00 | .006$^\ddagger$ | .93 | .01 | **.006** | .94 | .02 |
| phishing | .073 | .077 | .51 | **.05** | .077 | .46 | .12 | .077 | .61 | .16 | **.066** | **.84** | .13 | .066$^\ddagger$ | .82 | .15 | **.066** | .82 | .52 | .070 | .83 | .14 | .070 | .80 | .13 | .070 | .79 | .50 |
| page_block | .054 | .068 | .19 | **.01** | .068 | .31 | .03 | .068 | .28 | .03 | .049$^\ddagger$ | .77 | .03 | **.049** | **.85** | .06 | .049$^\ddagger$ | .84 | .07 | .051 | .73 | .03 | .051 | .77 | .06 | .051 | .75 | .07 |
| chess | .045 | .037$^\dagger$ | .09 | **.00** | .037$^\dagger$ | .26 | .00 | .037$^\dagger$ | .13 | .05 | .036$^\ddagger$ | .19 | .00 | .036$^\ddagger$ | .42 | .00 | .036$^\ddagger$ | .16 | .07 | **.036** | .22 | .00 | **.036** | **.47** | .01 | **.036** | .21 | .08 |
| mhr | .126 | .124 | .31 | **.05** | .124 | .34 | .09 | .124 | .56 | .18 | .111$^\ddagger$ | .60 | .11 | .111$^\ddagger$ | .62 | .11 | .111$^\ddagger$ | .72 | .39 | **.111** | .59 | .11 | .111$^\ddagger$ | .59 | .10 | **.111** | **.76** | .41 |
| connect-4 | .134 | .111$^\ddagger$ | .82 | .21 | **.111** | .80 | .20 | .111$^\ddagger$ | **.83** | .71 | .116$^\ddagger$ | .81 | .23 | .116$^\ddagger$ | .78 | .20 | .116$^\ddagger$ | .80 | .67 | .117$^\ddagger$ | .78 | .23 | .117$^\ddagger$ | .79 | **.18** | .117$^\ddagger$ | .79 | .66 |
| poker_hand | .139 | .139 | .00 | **.00** | .139 | .00 | **.00** | .139 | .00 | **.00** | .121$^\ddagger$ | .00 | **.00** | .121$^\ddagger$ | .41 | .43 | .121$^\ddagger$ | **.65** | .65 | **.121** | .00 | **.00** | **.121** | .10 | .07 | **.121** | .26 | .33 |
| molecular | .019 | .018$^\ddagger$ | .77 | **.01** | **.018** | .79 | .03 | .018$^\ddagger$ | .77 | .04 | .019$^\ddagger$ | .78 | .01 | .019$^\ddagger$ | .79 | .03 | .019$^\ddagger$ | **.82** | .03 | .019$^\dagger$ | .76 | .01 | .019$^\dagger$ | .77 | .02 | .019$^\dagger$ | .80 | .04 |
| waveform-v1 | .012 | .013$^\ddagger$ | .97 | **.01** | .012$^\ddagger$ | .94 | .03 | .013$^\ddagger$ | .94 | .05 | .012$^\ddagger$ | .99 | .01 | **.012** | .94 | .03 | .012$^\ddagger$ | .91 | .05 | .012$^\ddagger$ | **1.00** | .01 | .012$^\ddagger$ | .93 | .04 | .012$^\ddagger$ | .97 | .06 |
| isolet | .002 | .002$^\ddagger$ | .66 | **.00** | .002$^\ddagger$ | **.97** | .00 | .002$^\ddagger$ | .92 | .04 | .002 | .69 | **.00** | .002$^\ddagger$ | .96 | .02 | .002 | **.97** | .04 | **.002** | .73 | **.00** | .002$^\ddagger$ | **.97** | .02 | **.002** | .95 | .07 |
| cmc | .110 | .121$^\dagger$ | .62 | **.16** | .121$^\dagger$ | .61 | .16 | .121$^\dagger$ | .65 | .43 | **.103** | **.89** | .24 | .103$^\ddagger$ | .82 | .25 | **.103** | .83 | .66 | .103$^\ddagger$ | .88 | .24 | .103$^\ddagger$ | .71 | .24 | .103$^\ddagger$ | .81 | .62 |
| shuttle | **.066** | .068 | .12 | **.00** | .068 | .11 | .00 | .068 | .12 | .00 | .067 | .19 | .00 | .067 | .22 | .01 | .067 | **.23** | .02 | .067 | .20 | .00 | .067 | .21 | .02 | .067 | **.23** | .04 |
| satellite | .015 | .015$^\ddagger$ | .80 | **.00** | .015$^\ddagger$ | .87 | .01 | .015$^\ddagger$ | .97 | .01 | .014$^\ddagger$ | .88 | .00 | .014$^\ddagger$ | .94 | .01 | .014$^\ddagger$ | **.99** | .02 | **.014** | .90 | .00 | .014$^\ddagger$ | .92 | .01 | **.014** | .95 | .02 |
| hand_digits | .004 | .004 | .96 | **.00** | .004 | .98 | .01 | .004 | **.99** | .00 | .004$^\dagger$ | .97 | **.00** | .004$^\ddagger$ | **.99** | .01 | .004$^\dagger$ | .96 | .01 | .004$^\ddagger$ | .98 | **.00** | **.004** | .95 | .01 | .004$^\ddagger$ | .98 | .02 |
| yeast | .158 | .130$^\ddagger$ | .41 | **.21** | **.130** | .35 | .29 | .130$^\ddagger$ | .47 | .31 | .156 | .49 | .29 | .155 | .43 | .36 | .156 | **.56** | .45 | .146 | **.56** | .30 | .146 | .45 | .38 | .146 | .49 | .39 |
| nursery | .011 | .011$^\ddagger$ | .95 | **.00** | **.010** | .92 | .01 | .011$^\ddagger$ | .96 | .02 | .011$^\ddagger$ | **.98** | .00 | .011$^\ddagger$ | .97 | .01 | .011$^\ddagger$ | .97 | .02 | .010$^\ddagger$ | **.98** | .00 | .010$^\ddagger$ | .95 | .01 | .010$^\ddagger$ | **.98** | .02 |
| obesity | .012 | .010$^\ddagger$ | .86 | **.00** | **.010** | .87 | .02 | .010$^\ddagger$ | .91 | .02 | .011 | .94 | .00 | .011 | .91 | .01 | .011 | **.97** | .02 | .011 | .95 | .00 | .011 | .91 | .02 | .011 | **.97** | .01 |
| abalone | .097 | .081 | .05 | **.01** | .081 | .05 | .04 | .081 | .07 | .08 | .068$^\ddagger$ | .34 | .10 | .068$^\dagger$ | .56 | .07 | .068$^\ddagger$ | .24 | .28 | .068$^\ddagger$ | .39 | .11 | **.068** | **.60** | .07 | .068$^\ddagger$ | .23 | .24 |
| letter | .006 | .007 | .49 | **.00** | .007 | .76 | .01 | .007 | .72 | .08 | **.006** | .53 | **.00** | .006$^\ddagger$ | .79 | .02 | **.006** | .75 | .08 | .006$^\ddagger$ | .58 | **.00** | .006$^\ddagger$ | **.83** | .03 | .006$^\ddagger$ | .80 | .11 |
| digits | .003 | .003$^\ddagger$ | .93 | **.00** | .004$^\ddagger$ | **.98** | .01 | .003$^\ddagger$ | .96 | .02 | .003$^\dagger$ | .95 | **.00** | .003$^\dagger$ | .96 | .01 | .003$^\dagger$ | **.98** | .00 | **.003** | .95 | **.00** | .003$^\ddagger$ | .95 | .01 | **.003** | **.98** | .01 |
| academic-success | .037 | .037$^\ddagger$ | .89 | **.04** | .037$^\ddagger$ | .90 | .08 | .037$^\ddagger$ | .87 | .15 | **.036** | .93 | .05 | .036$^\ddagger$ | **.94** | .10 | **.036** | .92 | .18 | .036$^\ddagger$ | **.94** | .05 | .036$^\ddagger$ | .92 | .11 | .036$^\ddagger$ | .88 | .18 |
| wine-quality | .171 | .166 | .01 | .00 | .166 | .01 | .00 | .166 | .00 | **.00** | .147$^\ddagger$ | .28 | .15 | .147$^\ddagger$ | .34 | .12 | .147$^\ddagger$ | .70 | .56 | .146$^\ddagger$ | .39 | .17 | **.146** | .36 | .13 | .146$^\ddagger$ | **.77** | .60 |
| dry-bean | .005 | .005 | .97 | **.00** | .005 | .92 | .00 | .005 | **.98** | .01 | .005 | .97 | .00 | .005$^\dagger$ | .97 | .01 | .005 | .96 | .00 | **.005** | .97 | .00 | .005$^\ddagger$ | .97 | .01 | **.005** | .94 | .02 |
| Mean | .067 | .062$^\dagger$ | .56 | **.05** | .062$^\dagger$ | .60 | .07 | .062$^\dagger$ | .62 | .12 | .058$^\ddagger$ | .69 | .09 | .058$^\ddagger$ | .75 | .12 | .058$^\ddagger$ | **.77** | .23 | **.058** | .70 | .09 | .058$^\ddagger$ | .73 | .10 | **.058** | .75 | .22 |

(a) Multiclass experiments for ACC.

| | PACC | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | Population-based | | | | | | | | | Model-based | | | | | | | | | Combined | | | | | | | | |
| | - | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | |
| | MAE | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ |
| hcv | .305 | .274 | .14 | **.17** | .273 | .14 | .19 | .274 | .23 | .30 | **.235** | .52 | .66 | .236$^\ddagger$ | .49 | .67 | **.235** | .72 | .78 | .237$^\ddagger$ | .51 | .65 | .237$^\ddagger$ | .50 | .66 | .237$^\ddagger$ | **.73** | .77 |
| image_seg | .006 | .007 | .83 | **.00** | .007 | .88 | .01 | .007 | .90 | .00 | .005$^\ddagger$ | .95 | .00 | .006 | **.96** | .01 | .005$^\ddagger$ | .95 | .01 | .005$^\ddagger$ | **.96** | .00 | **.005** | .93 | .01 | .005$^\ddagger$ | .94 | .01 |
| phishing | .035 | .037 | .71 | **.02** | .037 | .68 | .06 | .037 | .82 | .09 | .033 | **.92** | .05 | .033$^\dagger$ | .89 | .09 | .033 | **.92** | .24 | **.032** | .91 | .05 | **.032** | .87 | .10 | **.032** | .84 | .23 |
| page_block | .042 | .042 | .40 | **.01** | .042 | .56 | .03 | .042 | .46 | .04 | .039$^\ddagger$ | **.82** | .02 | **.039** | .80 | .04 | .039$^\ddagger$ | .81 | .06 | .042 | .69 | .02 | .042 | .69 | .04 | .042 | .80 | .04 |
| chess | .041 | .034$^\ddagger$ | .18 | **.00** | **.034** | **.38** | .01 | .034$^\ddagger$ | .21 | .07 | .034$^\ddagger$ | .22 | .00 | .034$^\ddagger$ | **.38** | .01 | .034$^\ddagger$ | .23 | .08 | .034$^\ddagger$ | .22 | .00 | .034$^\ddagger$ | **.38** | .01 | .034$^\ddagger$ | .25 | .07 |
| mhr | .056 | **.043** | .79 | **.05** | .043$^\ddagger$ | .76 | .07 | **.043** | .85 | .17 | .051 | .86 | .08 | .052 | .86 | .10 | .051 | .85 | .32 | .049 | **.90** | .08 | .049 | .77 | .09 | .049 | .86 | .33 |
| connect-4 | .033 | .032$^\ddagger$ | .88 | **.04** | **.032** | .89 | .08 | .032$^\ddagger$ | .94 | .13 | .032$^\ddagger$ | .88 | .04 | .032$^\ddagger$ | .88 | .08 | .032$^\ddagger$ | **.98** | .15 | .032$^\ddagger$ | .90 | .04 | .032$^\ddagger$ | .86 | .09 | .032$^\ddagger$ | .88 | .14 |
| poker_hand | .098 | .099 | .00 | .00 | .099 | .00 | **.00** | .099 | .14 | .14 | .091$^\ddagger$ | .12 | .10 | **.091** | .09 | .09 | .091$^\ddagger$ | **.85** | .86 | .091$^\ddagger$ | .10 | .09 | .091$^\ddagger$ | .08 | .08 | .091$^\ddagger$ | .83 | .86 |
| molecular | .017 | .017$^\ddagger$ | .77 | **.01** | **.017** | .78 | .02 | .017$^\ddagger$ | .79 | .03 | .017$^\ddagger$ | .80 | .01 | .017$^\ddagger$ | **.85** | .02 | .017$^\ddagger$ | .81 | .03 | .018$^\dagger$ | .80 | .01 | .018$^\dagger$ | .74 | .02 | .018$^\dagger$ | .77 | .04 |
| waveform-v1 | .010 | .010$^\dagger$ | **.99** | **.01** | .010$^\ddagger$ | .95 | .02 | .010$^\dagger$ | .91 | .04 | .010$^\ddagger$ | .98 | .01 | **.010** | .93 | .03 | .010$^\ddagger$ | .94 | .04 | .010$^\ddagger$ | .98 | .01 | .010$^\ddagger$ | .93 | .03 | .010$^\ddagger$ | .94 | .05 |
| isolet | .002 | .002$^\ddagger$ | .66 | **.00** | .002$^\ddagger$ | .97 | .01 | .002$^\ddagger$ | .96 | .04 | .002 | .68 | **.00** | .002$^\dagger$ | .97 | .01 | .002 | **.98** | .06 | **.002** | .71 | **.00** | **.002** | **.98** | .03 | **.002** | .97 | .03 |
| cmc | .103 | .146 | .06 | **.05** | .146 | .13 | .13 | .146 | .27 | .19 | .099 | .53 | .11 | .099 | .59 | .15 | .099 | **.73** | .41 | .097$^\ddagger$ | .56 | .11 | **.097** | .64 | .16 | .097$^\ddagger$ | .72 | .41 |
| shuttle | .049 | .061 | .33 | **.00** | .061 | .43 | .01 | .061 | .57 | .02 | **.043** | **.97** | .02 | .043$^\ddagger$ | .94 | .04 | **.043** | **.97** | .18 | .043 | **.97** | .02 | .043 | .93 | .03 | .043 | .94 | .20 |
| satellite | .010 | .011 | .92 | **.00** | .011$^\dagger$ | .92 | .01 | .011 | .94 | .01 | .010$^\ddagger$ | .94 | .00 | .010$^\ddagger$ | .93 | .01 | .010$^\ddagger$ | **.98** | .01 | **.010** | .95 | .00 | .010$^\ddagger$ | .92 | .01 | **.010** | .97 | .02 |
| hand_digits | .003 | .004 | .95 | **.00** | .004 | .96 | .00 | .004 | .97 | .00 | .003$^\ddagger$ | .97 | **.00** | .003$^\dagger$ | .96 | .00 | .003$^\ddagger$ | **.99** | .02 | .003$^\ddagger$ | .97 | **.00** | **.003** | .93 | .01 | .003$^\ddagger$ | .97 | .02 |
| yeast | .128 | .131 | .07 | **.16** | .131 | .06 | .25 | .131 | .19 | .19 | .128 | .33 | .21 | .128 | .31 | .29 | .128 | .35 | .26 | .122$^\ddagger$ | .37 | .21 | **.122** | **.39** | .29 | .122$^\ddagger$ | .37 | .22 |
| nursery | .010 | .008$^\ddagger$ | **.98** | **.00** | **.008** | .97 | .01 | .008$^\ddagger$ | .94 | .02 | .010 | .96 | .00 | .010 | .97 | .01 | .010 | .94 | .02 | .010 | .97 | .00 | .010 | .91 | .02 | .010 | .90 | .02 |
| obesity | .015 | .013$^\ddagger$ | .60 | **.00** | **.013** | .73 | .03 | .013$^\ddagger$ | .81 | .02 | .015$^\dagger$ | .66 | .00 | .015 | .78 | .02 | .015$^\dagger$ | **.86** | .00 | .015 | .66 | .00 | .015 | .81 | .03 | .015 | .82 | .02 |
| abalone | .079 | .072 | .14 | **.01** | .072 | .16 | .01 | .072 | .18 | .14 | **.057** | .55 | .07 | .057$^\ddagger$ | **.59** | .05 | **.057** | .52 | .40 | .057$^\ddagger$ | .51 | .07 | .057$^\ddagger$ | .58 | .05 | .057$^\ddagger$ | .49 | .37 |
| letter | .005 | .006 | .49 | **.00** | .006 | .77 | .02 | .006 | .72 | .07 | .005$^\ddagger$ | .55 | **.00** | **.005** | .81 | .01 | .005$^\ddagger$ | .86 | .07 | .005$^\ddagger$ | .56 | **.00** | .005$^\ddagger$ | **.88** | .03 | .005$^\ddagger$ | .86 | .06 |
| digits | **.003** | .003 | .91 | **.00** | .003 | .94 | .01 | .003 | .96 | .01 | .003 | .93 | **.00** | .003 | .94 | .02 | .003 | **.97** | .02 | .003 | .93 | **.00** | .003 | .95 | .01 | .003 | .93 | .02 |
| academic-success | .035 | .038 | .69 | **.02** | .038 | .73 | .06 | .038 | .80 | .10 | .034 | .88 | .03 | .034 | .84 | .06 | .034 | .89 | .11 | **.032** | .89 | .03 | .032$^\ddagger$ | .83 | .07 | **.032** | **.90** | .14 |
| wine-quality | .123 | .106$^\ddagger$ | .27 | **.02** | .106$^\ddagger$ | .35 | .03 | .106$^\ddagger$ | .34 | .16 | .106 | .43 | .06 | .106 | .51 | .06 | .106 | **.59** | .43 | .104$^\ddagger$ | .46 | .07 | **.104** | .51 | .07 | .104$^\ddagger$ | .58 | .42 |
| dry-bean | .004 | .005$^\dagger$ | **.98** | **.00** | .005$^\ddagger$ | .96 | .00 | .005$^\dagger$ | .95 | .02 | .005$^\ddagger$ | **.98** | **.00** | .005$^\ddagger$ | .94 | .02 | .005$^\ddagger$ | .95 | .01 | **.004** | **.98** | .00 | .004$^\ddagger$ | .89 | .01 | **.004** | .93 | .02 |
| Mean | .051 | .050$^\dagger$ | .57 | **.02** | .050$^\ddagger$ | .63 | .04 | .050$^\dagger$ | .66 | .08 | .045$^\ddagger$ | .73 | .06 | .045$^\ddagger$ | .76 | .08 | .045$^\ddagger$ | **.82** | .19 | .044$^\ddagger$ | .73 | .06 | **.044** | .75 | .08 | .044$^\ddagger$ | .80 | .19 |

(b) Multiclass experiments for PACC.

| | DM | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | Population-based | | | | | | | | | Model-based | | | | | | | | | Combined | | | | | | | | |
| | - | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | | CI | | | CE | | | CT | | |
| | MAE | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ | MAE | $\mathcal{C}$ | $\mathcal{A}$ |
| hcv | .223 | .202 | .17 | **.17** | .202 | .20 | .23 | .202 | .48 | .51 | .183$^\ddagger$ | .68 | .75 | .183$^\ddagger$ | .75 | .79 | .183$^\ddagger$ | **.89** | .91 | **.182** | .62 | .72 | .182$^\ddagger$ | .72 | .76 | **.182** | .85 | .86 |
| image_seg | .011 | .013 | .39 | **.00** | .013 | .77 | .00 | .013 | .82 | .02 | .011 | .84 | .00 | .011 | .96 | .00 | .011 | .95 | .02 | .011$^\ddagger$ | .89 | .00 | **.011** | **.97** | .00 | .011$^\ddagger$ | **.97** | .03 |
| phishing | **.050** | .075 | .29 | **.02** | .075 | .35 | .05 | .075 | .42 | .04 | .064 | .56 | .04 | .064 | .64 | .09 | .064 | .66 | .11 | .063 | .58 | .04 | .063 | .62 | .09 | .063 | **.68** | .11 |
| page_block | **.038** | .057 | .16 | **.01** | .057 | .22 | .02 | .057 | .31 | .02 | .043 | .56 | .02 | .043 | **.69** | .04 | .043 | **.69** | .05 | .045 | .46 | .02 | .045 | .54 | .04 | .045 | .62 | .06 |
| chess | .038 | .033 | .19 | **.00** | .033 | .34 | .00 | .033 | .24 | .08 | **.031** | .40 | .00 | .031$^\ddagger$ | .58 | .00 | **.031** | .41 | .14 | .031$^\ddagger$ | .43 | .00 | .031$^\ddagger$ | **.64** | .00 | .031$^\ddagger$ | .46 | .19 |
| mhr | .060 | .041$^\ddagger$ | .85 | **.04** | **.041** | .81 | .08 | .041$^\ddagger$ | .87 | .14 | .056 | .84 | .07 | .056 | .90 | .11 | .056 | .90 | .24 | .055 | .83 | .06 | .055 | .81 | .11 | .055 | **.91** | .24 |
| connect-4 | .033 | .032$^\ddagger$ | .95 | **.04** | **.031** | .93 | .07 | .032$^\ddagger$ | **.96** | .13 | .032$^\ddagger$ | .94 | .04 | .032$^\ddagger$ | .94 | .08 | .032$^\ddagger$ | .93 | .14 | .032$^\ddagger$ | .94 | .04 | .032$^\ddagger$ | .89 | .08 | .032$^\ddagger$ | .93 | .14 |
| poker_hand | .118 | .118 | .05 | .01 | .118 | .00 | **.00** | .118 | .03 | .02 | .099$^\ddagger$ | .35 | .32 | .099$^\ddagger$ | .49 | .52 | .099$^\ddagger$ | **.89** | .92 | .099$^\ddagger$ | .34 | .33 | **.099** | .46 | .47 | .099$^\ddagger$ | .87 | .94 |
| molecular | .018 | .016$^\ddagger$ | .83 | **.01** | **.016** | .82 | .02 | .016$^\ddagger$ | .81 | .02 | .019 | .81 | .01 | .019 | **.85** | .02 | .019 | .83 | .03 | .019 | .82 | .01 | .019 | .84 | .02 | .019 | .81 | .03 |
| waveform-v1 | .009 | .010$^\ddagger$ | **.98** | **.01** | .010$^\ddagger$ | .93 | .02 | .010$^\ddagger$ | .94 | .03 | .009$^\ddagger$ | **.98** | .01 | .009$^\ddagger$ | .94 | .02 | .009$^\ddagger$ | .95 | .04 | .009$^\ddagger$ | **.98** | .01 | **.009** | .91 | .03 | .009$^\ddagger$ | .97 | .04 |
| isolet | .004 | .004$^\dagger$ | .78 | **.00** | .004 | .98 | .01 | .004$^\dagger$ | .93 | .03 | .004$^\ddagger$ | .82 | **.00** | .004$^\ddagger$ | **1.00** | .02 | .004$^\ddagger$ | .98 | .07 | **.004** | .83 | **.00** | .004$^\ddagger$ | **1.00** | .00 | **.004** | .98 | .07 |
| cmc | .080 | .121 | .12 | **.06** | .120 | .22 | .12 | .121 | .30 | .20 | .068$^\ddagger$ | .87 | .12 | **.068** | .86 | .17 | .068$^\ddagger$ | **.90** | .41 | .068$^\ddagger$ | .89 | .12 | .068$^\ddagger$ | **.90** | .18 | .068$^\ddagger$ | .86 | .43 |
| shuttle | **.040** | .050 | .52 | **.01** | .050 | .69 | .01 | .050 | .66 | .04 | .050 | .83 | .01 | .050 | **.93** | .02 | .050 | .80 | .12 | .050 | .81 | .01 | .050 | .89 | .02 | .050 | .79 | .12 |
| satellite | .011 | .011 | .87 | **.00** | .011 | .90 | .00 | .011 | .90 | .02 | .011$^\dagger$ | .91 | .00 | .011$^\ddagger$ | .93 | .02 | .011$^\dagger$ | **.95** | .01 | **.011** | .90 | .00 | .011$^\ddagger$ | .91 | .01 | **.011** | **.95** | .01 |
| hand_digits | **.005** | .005 | .93 | **.00** | .005 | .95 | .01 | .005 | .95 | .01 | .005 | .93 | **.00** | .005 | **.96** | .02 | .005 | **.96** | .01 | .005 | .95 | **.00** | .005 | .93 | .01 | .005 | .94 | .03 |
| yeast | .075 | .070$^\ddagger$ | .52 | **.14** | .070$^\ddagger$ | .49 | .20 | .070$^\ddagger$ | .42 | .15 | .069 | .72 | .18 | .069 | .66 | .25 | .069 | .57 | .24 | **.066** | **.77** | .18 | .066$^\ddagger$ | .65 | .25 | **.066** | .49 | .21 |
| nursery | **.007** | .009 | .96 | **.00** | .009 | .94 | .01 | .009 | .95 | .02 | .009 | **.99** | .00 | .009 | .96 | .01 | .009 | .96 | .02 | .009 | **.99** | .00 | .009 | .95 | .01 | .009 | .97 | .02 |
| obesity | .015 | .015$^\dagger$ | .51 | **.00** | .015$^\ddagger$ | .72 | .01 | .015$^\dagger$ | .83 | .02 | .014$^\ddagger$ | .78 | .00 | **.014** | .86 | .01 | .014$^\ddagger$ | **.96** | .01 | .014$^\ddagger$ | .77 | .00 | .014$^\ddagger$ | .79 | .01 | .014$^\ddagger$ | .95 | .04 |
| abalone | .079 | .067 | .08 | **.01** | .067 | .12 | .02 | .067 | .14 | .09 | .055$^\ddagger$ | .75 | .13 | **.054** | **.82** | .07 | .055$^\ddagger$ | .78 | .60 | .055$^\ddagger$ | .71 | .13 | .055$^\ddagger$ | **.82** | .07 | .055$^\ddagger$ | .73 | .56 |
| letter | .007 | .007 | .17 | **.00** | .007 | .32 | .00 | .007 | .36 | .06 | **.007** | .23 | **.00** | .007$^\ddagger$ | .37 | .03 | **.007** | .39 | .04 | .007$^\ddagger$ | .23 | **.00** | .007$^\ddagger$ | **.45** | .03 | .007$^\ddagger$ | .42 | .08 |
| digits | .005 | .005$^\ddagger$ | .93 | **.00** | .005$^\ddagger$ | .87 | .02 | .005$^\ddagger$ | .95 | .01 | .004$^\ddagger$ | .93 | **.00** | .005 | **.97** | .01 | .004$^\ddagger$ | .95 | .02 | **.004** | .93 | **.00** | .004$^\ddagger$ | .95 | .02 | **.004** | .91 | .03 |
| academic-success | .026 | .025$^\dagger$ | .87 | **.02** | .025$^\dagger$ | .87 | .08 | .025$^\dagger$ | .84 | .08 | .024 | .92 | .03 | .024 | .92 | .06 | .024 | .92 | .10 | .023$^\ddagger$ | **.95** | .03 | **.023** | .89 | .07 | .023$^\ddagger$ | .89 | .09 |
| wine-quality | .141 | .134 | .11 | **.02** | .134 | .11 | .03 | .134 | .33 | .21 | **.124** | .40 | .09 | .124$^\ddagger$ | .39 | .06 | **.124** | **.72** | .57 | .125$^\ddagger$ | .36 | .09 | .125$^\ddagger$ | .36 | .05 | .125$^\ddagger$ | .69 | .53 |
| dry-bean | **.005** | .005 | .95 | **.00** | .005 | .95 | .01 | .005 | .93 | .01 | .005 | **.97** | .00 | .005 | .93 | .01 | .005 | **.97** | .01 | .005 | **.97** | .00 | .005 | .94 | .01 | .005 | .93 | .01 |
| Mean | .046$^\ddagger$ | .047$^\dagger$ | .55 | **.02** | .047$^\dagger$ | .60 | .04 | .047$^\dagger$ | .64 | .08 | .042$^\ddagger$ | .75 | .08 | .042$^\ddagger$ | .80 | .10 | .042$^\ddagger$ | **.83** | .20 | .041$^\ddagger$ | .75 | .08 | **.041** | .78 | .10 | .041$^\ddagger$ | .82 | .20 |

(c) Multiclass experiments for DM.

Table 6: Multiclass experiments for ACC, PACC, and DM.

| | Population-based | | | Model-based | | | Combined | | |
|---|---|---|---|---|---|---|---|---|---|
| | CI | CE | CT | CI | CE | CT | CI | CE | CT |
| balance.1 | **2.127** | 1.763 | 1.979$^\dagger$ | 2.050 | 1.643 | 1.586 | 2.097 | 1.647 | 1.477 |
| balance.3 | **2.516** | 2.140 | 2.013 | 2.432 | 1.900 | 1.695 | 2.434 | 1.781 | 1.670 |
| breast-cancer | 1.836$^\ddagger$ | 1.668 | 1.716$^\ddagger$ | 2.159$^\ddagger$ | 1.658 | 1.689 | **2.262** | 1.628 | 1.618 |
| cmc.1 | 0.840$^\ddagger$ | 0.702 | 0.642 | **1.201** | 0.862 | 0.649 | 1.113$^\ddagger$ | 0.850 | 0.506 |
| cmc.2 | **1.093** | 0.997$^\dagger$ | 0.552 | 1.021 | 0.847 | 0.571 | 0.990 | 0.849 | 0.426 |
| cmc.3 | 0.444 | 0.386 | 0.495 | 0.980$^\ddagger$ | 0.740$^\ddagger$ | 0.346 | **1.023** | 0.754$^\ddagger$ | 0.427 |
| german | 1.120$^\ddagger$ | 0.924$^\dagger$ | 0.756 | 1.250$^\ddagger$ | 1.006 | 0.712 | **1.259** | 0.907 | 0.770 |
| haberman | **1.050** | 0.947 | 0.469 | 0.693 | 0.551 | 0.409 | 0.693 | 0.714 | 0.367 |
| ionosphere | 0.392 | 0.461 | 0.406 | **0.959** | 0.803$^\dagger$ | 0.573 | 0.868$^\ddagger$ | 0.910$^\ddagger$ | 0.593 |
| iris.1 | **2.818** | 2.232 | 2.420 | **2.818** | 2.232 | 2.420 | 2.814$^\ddagger$ | 2.384 | 2.497 |
| iris.2 | **1.580** | 1.266 | 1.011 | 1.040 | 0.888 | 0.399 | 1.073 | 1.006 | 0.393 |
| iris.3 | 0.000 | 0.021 | 0.037 | 0.023 | 0.369$^\ddagger$ | **0.438** | 0.116$^\dagger$ | 0.305$^\ddagger$ | 0.410$^\ddagger$ |
| mammographic | 1.413$^\ddagger$ | 1.168 | 1.267$^\ddagger$ | 1.813$^\ddagger$ | 1.351 | 1.195 | **1.840** | 1.414 | 1.078 |
| pageblocks.5 | 1.196$^\ddagger$ | 0.994 | 0.957 | **1.486** | 0.879 | 0.888 | 1.376$^\dagger$ | 0.880 | 0.901 |
| sonar | 0.566$^\ddagger$ | 0.459$^\ddagger$ | 0.597$^\ddagger$ | 0.683$^\ddagger$ | 0.673 | 0.549 | **0.694** | 0.655 | 0.453 |
| spambase | **2.548** | 2.102 | 2.102 | 2.505 | 1.997 | 1.714 | 2.496 | 1.966 | 1.594 |
| spectf | **1.398** | 1.019 | 0.799 | 1.268 | 0.923 | 0.509 | 1.251 | 1.105 | 0.454 |
| tictactoe | **2.750** | 2.235 | 2.255 | 2.660 | 1.993 | 1.868 | 2.637 | 2.079 | 1.892 |
| transfusion | 1.010 | 0.861$^\ddagger$ | 0.445 | **1.023** | 0.851$^\dagger$ | 0.465 | 0.952 | 0.915 | 0.497 |
| wdbc | 2.204 | 1.780 | 1.812$^\ddagger$ | **2.224** | 1.770 | 1.755 | 2.203$^\ddagger$ | 1.526 | 1.551 |
| wine.1 | 2.148$^\ddagger$ | 1.864 | 1.837 | **2.736** | 2.264 | 2.082 | 2.719$^\ddagger$ | 2.255 | 2.263 |
| wine.2 | 2.516 | 2.162 | 2.294$^\ddagger$ | **2.591** | 1.976 | 1.831 | 2.573$^\ddagger$ | 1.969 | 1.871 |
| wine.3 | 2.349 | 1.981 | 2.216 | 2.473$^\ddagger$ | 1.925 | 1.581 | **2.479** | 1.969 | 1.664 |
| wine-q-red | **1.947** | 1.632 | 1.280 | 1.785 | 1.410 | 0.916 | 1.782 | 1.457 | 0.942 |
| wine-q-white | 1.547$^\ddagger$ | 1.362 | 1.054 | 1.729$^\ddagger$ | 1.233 | 0.998 | **1.734** | 1.368 | 1.006 |
| yeast | 0.715$^\ddagger$ | 0.540$^\dagger$ | 0.732$^\ddagger$ | 0.750$^\ddagger$ | 0.581 | 0.524 | **0.862** | 0.670 | 0.452 |
| *Average* | 1.543$^\ddagger$ | 1.295 | 1.236 | **1.629** | 1.282 | 1.091 | 1.629$^\ddagger$ | 1.306 | 1.068 |

(a) Goodness results for binary experiments using ACC.

| | Population-based | | | Model-based | | | Combined | | |
|---|---|---|---|---|---|---|---|---|---|
| | CI | CE | CT | CI | CE | CT | CI | CE | CT |
| balance.1 | 1.710$^\ddagger$ | 1.281 | 1.364 | **2.305** | 1.733 | 1.592 | 2.221$^\ddagger$ | 1.682 | 1.596 |
| balance.3 | 2.373 | 1.776 | 1.823$^\ddagger$ | 2.484$^\ddagger$ | 1.825 | 1.654 | **2.503** | 1.648 | 1.653 |
| breast-cancer | 2.217 | 1.702 | 1.811$^\ddagger$ | **2.315** | 1.799 | 1.575 | 2.283$^\ddagger$ | 1.634 | 1.604 |
| cmc.1 | 1.293$^\dagger$ | 0.896 | 0.858 | **1.395** | 1.215 | 0.706 | 1.381$^\ddagger$ | 1.023 | 0.746 |
| cmc.2 | **1.542** | 1.253 | 0.842 | 1.537 | 1.143 | 0.751 | 1.509 | 1.112 | 0.701 |
| cmc.3 | **1.129** | 0.777 | 0.623 | 0.968 | 0.701 | 0.460 | 0.992 | 0.795 | 0.486 |
| german | 1.708 | 1.279$^\ddagger$ | 1.029 | 1.693$^\ddagger$ | 1.372 | 0.835 | **1.729** | 1.388 | 0.641 |
| haberman | 0.698 | 0.452 | 0.459 | 1.187 | 0.957$^\ddagger$ | 0.389 | **1.224** | 1.129$^\ddagger$ | 0.331 |
| ionosphere | 0.023 | 0.080 | 0.323$^\ddagger$ | 0.385$^\ddagger$ | **0.462** | 0.383$^\ddagger$ | 0.297$^\ddagger$ | 0.433$^\ddagger$ | 0.330$^\ddagger$ |
| iris.1 | 2.449$^\ddagger$ | 2.024 | 1.999 | **2.753** | 2.077 | 1.905 | 2.749$^\ddagger$ | 1.966 | 2.019 |
| iris.2 | **1.472** | 1.102 | 0.921 | 1.357$^\ddagger$ | 1.128$^\dagger$ | 0.527 | 1.378$^\ddagger$ | 1.220$^\ddagger$ | 0.479 |
| iris.3 | **2.603** | 2.074 | 1.964 | 2.201 | 1.628 | 1.431 | 2.211 | 1.682 | 1.324 |
| mammographic | 1.434$^\ddagger$ | 1.226 | 1.139 | **1.952** | 1.497 | 1.334 | 1.937$^\ddagger$ | 1.445 | 1.255 |
| pageblocks.5 | **2.235** | 1.703 | 1.527 | 1.994 | 1.504 | 1.433 | 1.997 | 1.494 | 1.200 |
| sonar | **0.796** | 0.629 | 0.666$^\ddagger$ | 0.490 | 0.365 | 0.466$^\dagger$ | 0.481 | 0.387 | 0.430$^\dagger$ |
| spambase | **2.552** | 1.963 | 1.608 | 2.540 | 2.036 | 1.651 | 2.537 | 2.072 | 1.620 |
| spectf | 1.035$^\dagger$ | 0.889 | 0.789 | **1.444** | 1.002 | 0.626 | 1.438 | 1.038 | 0.506 |
| tictactoe | **2.483** | 1.842 | 1.935 | 2.410 | 1.748 | 1.735 | 2.469 | 1.790 | 1.583 |
| transfusion | 1.310$^\ddagger$ | 1.015$^\dagger$ | 0.738 | 1.541$^\ddagger$ | 1.078 | 0.701 | **1.549** | 1.224 | 0.762 |
| wdbc | **2.506** | 2.001 | 1.824 | 2.219 | 1.696 | 1.574 | 2.199 | 1.625 | 1.465 |
| wine.1 | 2.102$^\dagger$ | 1.689 | 1.872$^\ddagger$ | **2.552** | 2.032 | 1.779 | 2.520 | 1.940 | 1.762 |
| wine.2 | **2.563** | 1.897 | 1.936 | 2.517 | 1.808 | 1.808 | 2.496 | 2.034 | 1.431 |
| wine.3 | 2.158$^\dagger$ | 1.585 | 1.785$^\ddagger$ | **2.510** | 1.866 | 1.581 | 2.501$^\ddagger$ | 1.626 | 1.657 |
| wine-q-red | 1.891 | 1.495 | 1.132 | **1.896** | 1.532 | 1.052 | 1.877 | 1.411 | 0.939 |
| wine-q-white | 1.435$^\ddagger$ | 1.144 | 1.026 | 1.805$^\ddagger$ | 1.423 | 0.989 | **1.812** | 1.285 | 0.985 |
| yeast | 0.136 | 0.101 | 0.281 | 0.568$^\ddagger$ | 0.454 | 0.418$^\ddagger$ | **0.634** | 0.569$^\dagger$ | 0.471$^\ddagger$ |
| *Average* | 1.687$^\ddagger$ | 1.303 | 1.241 | **1.808** | 1.388 | 1.129 | 1.805$^\ddagger$ | 1.371 | 1.076 |

(b) Goodness results for binary experiments using PACC.

| | Population-based | | | Model-based | | | Combined | | |
|---|---|---|---|---|---|---|---|---|---|
| | CI | CE | CT | CI | CE | CT | CI | CE | CT |
| balance.1 | 1.865$^\ddagger$ | 1.541 | 1.591$^\dagger$ | 2.390$^\ddagger$ | 1.844 | 1.710 | **2.399** | 1.678 | 1.671 |
| balance.3 | **2.510** | 1.924 | 1.721 | 2.465 | 1.663 | 1.816 | 2.405 | 1.726 | 1.706 |
| breast-cancer | **2.627** | 2.063 | 1.867 | 2.424 | 1.869 | 1.981 | 2.387 | 1.810 | 1.686 |
| cmc.1 | 1.282$^\ddagger$ | 0.954 | 0.631 | **1.543** | 1.199 | 0.739 | 1.539$^\ddagger$ | 1.182 | 0.784 |
| cmc.2 | **1.558** | 1.302 | 0.992 | 1.524 | 1.207 | 0.596 | 1.536 | 1.191 | 0.759 |
| cmc.3 | **1.314** | 1.075 | 0.669 | 1.172 | 0.796 | 0.492 | 1.172 | 0.863 | 0.402 |
| german | 1.165$^\ddagger$ | 0.842 | 0.819 | 1.494$^\ddagger$ | 1.017 | 0.783 | **1.540** | 0.993 | 0.623 |
| haberman | **1.556** | 1.130 | 0.927 | 1.075 | 0.808 | 0.511 | 1.104 | 0.909 | 0.358 |
| ionosphere | 0.219 | 0.450 | 0.454 | 1.049$^\ddagger$ | **1.120** | 0.762$^\dagger$ | 1.107$^\ddagger$ | 0.993$^\ddagger$ | 0.686$^\dagger$ |
| iris.1 | **2.811** | 2.289 | 2.296 | 2.762 | 2.293 | 2.169 | 2.748 | 2.121 | 1.988 |
| iris.2 | 1.329$^\ddagger$ | 1.024$^\dagger$ | 1.059 | 1.410$^\ddagger$ | 0.983 | 0.797 | **1.415** | 0.994 | 0.600 |
| iris.3 | 1.718$^\ddagger$ | 1.486 | 1.492 | **2.224** | 1.611 | 1.491 | 2.216$^\ddagger$ | 1.612 | 1.311 |
| mammographic | 1.447$^\ddagger$ | 1.242 | 1.040 | 1.922$^\ddagger$ | 1.344 | 1.298 | **1.942** | 1.304 | 1.181 |
| pageblocks.5 | 1.769$^\ddagger$ | 1.355 | 1.402$^\ddagger$ | 1.780 | 1.237 | 1.129 | **1.867** | 1.155 | 1.215 |
| sonar | 0.287$^\ddagger$ | 0.158$^\ddagger$ | 0.227$^\ddagger$ | 0.016 | 0.094$^\dagger$ | **0.305** | 0.016 | 0.056 | 0.289$^\ddagger$ |
| spambase | **2.603** | 1.996 | 1.997 | 2.565 | 1.906 | 1.852 | 2.560 | 2.085 | 1.821 |
| spectf | 0.017 | 0.071 | 0.093 | **1.356** | 1.048 | 0.589 | 1.323$^\dagger$ | 0.994 | 0.659 |
| tictactoe | **2.684** | 2.208 | 2.219 | 2.586 | 2.154 | 1.890 | 2.577 | 2.061 | 1.664 |
| transfusion | 0.675 | 0.677 | 0.504 | **1.460** | 1.072 | 0.681 | 1.399 | 1.043 | 0.587 |
| wdbc | **2.386** | 1.831 | 1.748 | 2.379 | 1.860 | 1.734 | 2.376 | 1.840 | 1.786 |
| wine.1 | 0.333 | 0.321 | 0.282 | **2.500** | 1.819 | 1.842 | 2.494$^\ddagger$ | 1.817 | 1.836 |
| wine.2 | **2.690** | 2.055 | 1.747 | 2.516 | 1.814 | 1.654 | 2.492 | 1.800 | 1.616 |
| wine.3 | **2.741** | 2.109 | 2.106 | 2.647 | 2.114 | 1.843 | 2.651 | 2.033 | 1.648 |
| wine-q-red | 1.966 | 1.575 | 1.186 | **1.982** | 1.452 | 1.195 | 1.952 | 1.437 | 1.078 |
| wine-q-white | 1.491$^\ddagger$ | 1.064 | 0.951 | 1.843$^\ddagger$ | 1.439 | 1.021 | **1.849** | 1.389 | 1.006 |
| yeast | 1.130$^\ddagger$ | 0.929 | 0.771 | 1.382$^\ddagger$ | 1.128 | 0.859 | **1.475** | 1.117 | 0.761 |
| *Average* | 1.622$^\ddagger$ | 1.295 | 1.184 | 1.864$^\ddagger$ | 1.419 | 1.221 | **1.867** | 1.392 | 1.143 |

(c) Goodness results for binary experiments using DM.

Table 7: Goodness results for binary experiments using ACC, PACC, and DM in terms of Log-Ratio Goodness (LRG).

| | Population-based | | | Model-based | | | Combined | | |
|---|---|---|---|---|---|---|---|---|---|
| | CI | CE | CT | CI | CE | CT | CI | CE | CT |
| hcv | 0.332 | 0.323 | 0.292 | 0.523$^{\dagger}$ | 0.527$^{\dagger}$ | **0.574** | 0.524$^{\ddagger}$ | 0.501$^{\ddagger}$ | 0.571$^{\dagger}$ |
| image_seg | 10.378 | 9.596 | 9.843 | **14.200** | 10.099 | 10.219 | 13.318$^{\ddagger}$ | 9.981 | 8.156 |
| phishing | 1.555$^{\ddagger}$ | 1.245 | 1.114 | **2.012** | 1.951$^{\ddagger}$ | 0.950 | 1.989$^{\ddagger}$ | 1.925$^{\ddagger}$ | 0.949 |
| page_block | 0.893 | 1.128 | 1.028 | **2.911** | 2.598 | 2.419 | 2.796$^{\ddagger}$ | 2.319 | 2.152 |
| chess | 0.777 | 2.163$^{\dagger}$ | 0.485 | 1.361 | 3.321$^{\ddagger}$ | 0.571 | 1.553 | **3.648** | 0.676 |
| mhr | 0.883 | 0.947$^{\dagger}$ | 1.017 | 1.364$^{\dagger}$ | **1.461** | 0.946$^{\dagger}$ | 1.338 | 1.385$^{\ddagger}$ | 0.960 |
| connect-4 | 1.401 | 1.523$^{\ddagger}$ | 0.701 | 1.346 | 1.487$^{\ddagger}$ | 0.733 | 1.305 | **1.554** | 0.716 |
| poker_hand | 0.000 | 0.000 | 0.000 | 0.000 | 0.490$^{\ddagger}$ | **0.608** | 0.000 | 0.271 | 0.364$^{\ddagger}$ |
| molecular | **3.605** | 3.001 | 2.930 | 3.593 | 2.947 | 3.152 | 3.466 | 2.897 | 2.946 |
| waveform-v1 | 4.407 | 3.512 | 3.359 | 4.401$^{\ddagger}$ | 3.406 | 3.054 | **4.416** | 3.355 | 3.208 |
| isolet | 10.638$^{\dagger}$ | **12.750** | 10.137$^{\dagger}$ | 11.121$^{\ddagger}$ | 12.487$^{\ddagger}$ | 11.713$^{\ddagger}$ | 11.766$^{\ddagger}$ | 10.986$^{\dagger}$ | 10.089$^{\dagger}$ |
| cmc | 1.198$^{\ddagger}$ | 1.178$^{\ddagger}$ | 0.730 | **1.533** | 1.332 | 0.764 | 1.483 | 1.119 | 0.811 |
| shuttle | 0.839$^{\ddagger}$ | 0.721$^{\ddagger}$ | 0.680$^{\dagger}$ | 1.204$^{\ddagger}$ | 1.170$^{\ddagger}$ | 1.011$^{\ddagger}$ | **1.258** | 1.064$^{\ddagger}$ | 0.953$^{\ddagger}$ |
| satellite | 8.635$^{\ddagger}$ | 7.180 | 6.292 | 8.963$^{\ddagger}$ | 7.638 | 5.850 | **9.372** | 7.167 | 5.259 |
| hand_digits | 15.473$^{\ddagger}$ | 13.883 | 13.865 | 15.635$^{\ddagger}$ | 12.882 | 13.450 | **15.796** | 11.787 | 12.641 |
| yeast | 0.715$^{\ddagger}$ | 0.540$^{\dagger}$ | 0.732$^{\ddagger}$ | 0.750$^{\ddagger}$ | 0.581 | 0.524 | **0.862** | 0.670 | 0.452 |
| nursery | 6.235$^{\dagger}$ | 4.973 | 4.796 | **6.339** | 5.017 | 4.836 | 6.287$^{\ddagger}$ | 4.890 | 4.724 |
| obesity | 13.160$^{\ddagger}$ | 9.604 | 7.371 | **13.675** | 10.479 | 6.941 | 13.450$^{\ddagger}$ | 8.479 | 6.363 |
| abalone | 0.186 | 0.104 | 0.054 | 0.856 | 1.712$^{\ddagger}$ | 0.191 | 0.999 | **1.848** | 0.205 |
| letter | 7.898$^{\dagger}$ | 10.001$^{\ddagger}$ | 5.925 | 8.543$^{\ddagger}$ | **10.393** | 6.227 | 9.348$^{\ddagger}$ | 9.944$^{\ddagger}$ | 6.055 |
| digits | 14.990$^{\ddagger}$ | 13.035 | 13.723$^{\ddagger}$ | **15.312** | 13.103 | 13.952$^{\ddagger}$ | **15.312** | 11.625 | 12.532 |
| academic-success | **2.869** | 2.518 | 1.876 | 2.852 | 2.447 | 1.901 | 2.866 | 2.368 | 1.809 |
| wine-quality | 0.069 | 0.074 | 0.000 | 0.579$^{\ddagger}$ | 0.746$^{\ddagger}$ | 0.690$^{\ddagger}$ | **0.767** | 0.764$^{\ddagger}$ | 0.749$^{\ddagger}$ |
| dry-bean | **15.220** | 10.253 | 10.183 | 15.013$^{\ddagger}$ | 10.792 | 10.605 | 15.075$^{\ddagger}$ | 10.749 | 8.558 |
| *Average* | 5.098 | 4.594$^{\dagger}$ | 4.047 | 5.587$^{\ddagger}$ | 4.961$^{\ddagger}$ | 4.245 | **5.639** | 4.637$^{\ddagger}$ | 3.829 |

(a) Goodness results for multiclass experiments using ACC.

| | Population-based | | | Model-based | | | Combined | | |
|---|---|---|---|---|---|---|---|---|---|
| | CI | CE | CT | CI | CE | CT | CI | CE | CT |
| hcv | 0.170 | 0.149 | 0.183 | 0.434$^{\ddagger}$ | 0.412$^{\ddagger}$ | 0.518$^{\ddagger}$ | 0.428$^{\ddagger}$ | 0.428$^{\ddagger}$ | **0.522** |
| image_seg | 12.887$^{\dagger}$ | 10.099 | 10.465 | **14.753** | 11.370 | 9.164 | 14.499$^{\ddagger}$ | 9.835 | 9.121 |
| phishing | 2.633$^{\ddagger}$ | 2.087$^{\dagger}$ | 2.112 | **2.814** | 2.394 | 1.725 | 2.795$^{\ddagger}$ | 2.279 | 1.528 |
| page_block | 1.896 | 2.105 | 1.686 | **3.372** | 2.687 | 2.496 | 2.905$^{\ddagger}$ | 2.394 | 2.676 |
| chess | 1.648 | **3.335** | 0.571 | 1.893 | 3.085$^{\ddagger}$ | 0.697 | 1.870 | 3.250$^{\ddagger}$ | 0.845 |
| mhr | **2.491** | 2.189 | 1.700 | 2.281 | 2.162 | 1.296 | 2.419 | 1.911 | 1.274 |
| connect-4 | 2.977 | 2.458 | 2.140 | 2.928$^{\ddagger}$ | 2.418 | 2.137 | **2.984** | 2.298 | 1.936 |
| poker_hand | 0.000 | 0.000 | 0.215 | 0.279 | 0.230 | **0.634** | 0.232 | 0.227 | 0.610$^{\ddagger}$ |
| molecular | 3.629 | 2.987 | 3.088 | **3.750** | 3.226 | 3.156 | 3.710$^{\ddagger}$ | 2.847 | 2.956 |
| waveform-v1 | **4.617** | 3.672 | 3.341 | 4.438 | 3.456 | 3.442 | 4.398 | 3.423 | 3.207 |
| isolet | 10.638$^{\dagger}$ | **13.062** | 11.172$^{\dagger}$ | 10.960$^{\dagger}$ | 12.473$^{\ddagger}$ | 11.055$^{\ddagger}$ | 11.444$^{\dagger}$ | 10.927$^{\dagger}$ | 9.799 |
| cmc | 0.146 | 0.266 | 0.401 | 1.301$^{\dagger}$ | 1.214$^{\ddagger}$ | 0.926$^{\dagger}$ | **1.358** | 1.348$^{\ddagger}$ | 0.861$^{\dagger}$ |
| shuttle | 2.003 | 2.219 | 2.389 | **4.387** | 3.908 | 2.360 | 4.362$^{\ddagger}$ | 3.832 | 2.084 |
| satellite | **11.742** | 8.340 | 6.448 | 11.430$^{\ddagger}$ | 8.562 | 6.520 | 11.405$^{\ddagger}$ | 7.471 | 5.853 |
| hand_digits | 15.312$^{\ddagger}$ | 13.652 | 13.696$^{\dagger}$ | **15.635** | 13.335 | 13.608$^{\dagger}$ | **15.635** | 11.934 | 12.252 |
| yeast | 0.136 | 0.101 | 0.281 | 0.568$^{\ddagger}$ | 0.454 | 0.418$^{\ddagger}$ | **0.634** | 0.569$^{\dagger}$ | 0.471$^{\ddagger}$ |
| nursery | **6.652** | 5.346 | 4.795 | 6.448$^{\dagger}$ | 5.302 | 4.737 | 6.509$^{\ddagger}$ | 4.802 | 4.525 |
| obesity | 8.897$^{\ddagger}$ | 7.651$^{\ddagger}$ | 6.462 | 9.519$^{\ddagger}$ | 8.496$^{\ddagger}$ | 6.202 | **9.556** | 7.408$^{\dagger}$ | 5.835 |
| abalone | 0.532 | 0.709 | 0.179 | 1.741 | **2.179** | 0.533 | 1.550 | 2.092$^{\ddagger}$ | 0.560 |
| letter | 7.898 | 9.737$^{\ddagger}$ | 6.591 | 8.865$^{\dagger}$ | 10.622$^{\ddagger}$ | 7.162 | 9.026$^{\dagger}$ | **10.927** | 6.988 |
| digits | 14.667$^{\ddagger}$ | 12.600 | 13.519$^{\ddagger}$ | **14.990** | 12.801$^{\dagger}$ | 13.002$^{\dagger}$ | **14.990** | 11.093 | 12.190 |
| academic-success | 2.618$^{\ddagger}$ | 2.225 | 2.026 | 3.245$^{\ddagger}$ | 2.525 | 2.258 | **3.248** | 2.485 | 2.153 |
| wine-quality | 1.003$^{\dagger}$ | 1.335$^{\ddagger}$ | 0.586 | 1.191 | **1.555** | 0.620 | 1.232 | 1.459$^{\ddagger}$ | 0.645 |
| dry-bean | 15.381$^{\ddagger}$ | 11.433 | 10.507 | **15.443** | 10.450 | 10.729 | 15.305$^{\ddagger}$ | 9.536 | 9.055 |
| *Average* | 5.441 | 4.907$^{\dagger}$ | 4.356 | **5.944** | 5.222$^{\ddagger}$ | 4.391 | 5.937$^{\ddagger}$ | 4.782$^{\dagger}$ | 4.081 |

(b) Goodness results for multiclass experiments using PACC.

| | Population-based | | | Model-based | | | Combined | | |
|---|---|---|---|---|---|---|---|---|---|
| | CI | CE | CT | CI | CE | CT | CI | CE | CT |
| hcv | 0.269 | 0.270 | 0.413$^{\dagger}$ | 0.553$^{\ddagger}$ | 0.592$^{\ddagger}$ | **0.623** | 0.521$^{\ddagger}$ | 0.577$^{\ddagger}$ | 0.602$^{\ddagger}$ |
| image_seg | 5.789 | 8.620 | 6.977 | 12.047$^{\ddagger}$ | 11.290$^{\ddagger}$ | 6.720 | **12.176** | 11.175$^{\ddagger}$ | 6.328 |
| phishing | 1.207$^{\dagger}$ | 1.246 | 1.312 | 2.010$^{\dagger}$ | 1.880$^{\dagger}$ | 1.590$^{\dagger}$ | **2.094** | 1.818 | 1.651$^{\dagger}$ |
| page_block | 0.794 | 0.872 | 1.239 | 2.428$^{\ddagger}$ | **2.430** | 2.305$^{\ddagger}$ | 2.070$^{\ddagger}$ | 1.924$^{\ddagger}$ | 2.059$^{\ddagger}$ |
| chess | 1.459 | 2.533 | 0.608 | 2.884 | 4.716$^{\ddagger}$ | 0.915 | 2.890 | **5.059** | 1.017 |
| mhr | **2.783** | 2.286 | 1.907 | 2.412 | 2.289 | 1.599 | 2.368 | 2.063 | 1.575 |
| connect-4 | **3.258** | 2.618 | 2.253 | 3.173 | 2.674 | 2.046 | 3.158 | 2.478 | 2.030 |
| poker_hand | 0.209 | 0.000 | 0.081 | 0.496$^{\ddagger}$ | 0.528$^{\dagger}$ | **0.636** | 0.473$^{\ddagger}$ | 0.527 | 0.615$^{\ddagger}$ |
| molecular | **3.955** | 3.204 | 3.284 | 3.839 | 3.233 | 3.310 | 3.843 | 3.248 | 3.149 |
| waveform-v1 | **4.607** | 3.640 | 3.691 | 4.531 | 3.585 | 3.464 | 4.500 | 3.451 | 3.534 |
| isolet | 12.572 | 14.429$^{\ddagger}$ | 10.281 | 13.217 | 14.628$^{\ddagger}$ | 9.229 | 13.378 | **15.647** | 8.036 |
| cmc | 0.336 | 0.506 | 0.446 | 2.011$^{\ddagger}$ | 1.776 | 1.175 | **2.039** | 1.828 | 1.069 |
| shuttle | 2.707$^{\dagger}$ | 3.388$^{\ddagger}$ | 2.376 | 3.920$^{\ddagger}$ | **4.226** | 2.027 | 3.922$^{\ddagger}$ | 4.042$^{\ddagger}$ | 1.928 |
| satellite | **11.398** | 7.900 | 6.295 | 11.298$^{\ddagger}$ | 7.606 | 6.186 | 10.745$^{\ddagger}$ | 7.825 | 6.034 |
| hand_digits | 14.990$^{\ddagger}$ | 12.887 | 11.921 | 14.990$^{\ddagger}$ | 12.745 | 11.212 | **15.312** | 11.964 | 10.099 |
| yeast | 1.130$^{\ddagger}$ | 0.929 | 0.771 | 1.382$^{\ddagger}$ | 1.128 | 0.859 | **1.475** | 1.117 | 0.761 |
| nursery | 6.370$^{\ddagger}$ | 5.077 | 4.709 | 6.585$^{\ddagger}$ | 5.247 | 4.581 | **6.630** | 5.152 | 4.600 |
| obesity | 7.177 | 7.373 | 5.901 | **10.495** | 8.746$^{\ddagger}$ | 5.909 | 10.475$^{\ddagger}$ | 7.937 | 5.718 |
| abalone | 0.484 | 0.658 | 0.342 | 1.800 | 2.586$^{\ddagger}$ | 0.826 | 1.661 | **2.641** | 0.790 |
| letter | 2.740 | 4.770$^{\ddagger}$ | 3.128$^{\dagger}$ | 3.707 | **5.271** | 3.687$^{\ddagger}$ | 3.707 | 4.804$^{\ddagger}$ | 2.895$^{\dagger}$ |
| digits | **14.990** | 11.491 | 11.114 | **14.990** | 13.771$^{\ddagger}$ | 9.554 | **14.990** | 12.914$^{\dagger}$ | 8.142 |
| academic-success | 3.318 | 2.626 | 2.301 | 3.394 | 2.817 | 2.407 | **3.435** | 2.641 | 2.300 |
| wine-quality | 0.394 | 0.376 | 0.482 | 1.062$^{\ddagger}$ | **1.140** | 0.702$^{\dagger}$ | 0.963 | 1.117$^{\ddagger}$ | 0.710$^{\dagger}$ |
| dry-bean | 14.822$^{\ddagger}$ | 10.276 | 10.484 | **15.075** | 10.881 | 9.579 | 14.730$^{\ddagger}$ | 9.984 | 8.756 |
| *Average* | 4.907 | 4.499 | 3.846 | **5.762** | 5.241$^{\ddagger}$ | 3.798 | 5.731$^{\ddagger}$ | 5.080$^{\ddagger}$ | 3.517 |

(c) Goodness results for multiclass experiments using DM.

Table 8: Goodness results for multiclass ACC, PACC, and DM in terms of Log-Ratio Goodness (LRG).

# An Evaluation of Methods for Unlabeled Class Prevalence Hypothesis Testing

David Kaftan[1,2], Anna Bershteyn[2], and Richard Povinelli[1]

[1] Marquette University, Milwaukee WI, USA
`{david.kaftan,richard.povinelli}@marquette.edu`
[2] NYU Grossman School of Medicine, New York NY, USA

**Abstract.** The machine learning task of quantification aims to estimate the proportion of classes in an unlabeled dataset. This task is made difficult by label shift, where the class distribution of the target data differs from the training data. While many methods focus on precise prevalence estimation, a common practical goal is simply to detect whether a class's prevalence has shifted from a nominal value. This study systematically compares four methods for detecting such deviation of class prevalence in binary classification tasks: a one-sample t-test, a one-sample KS-test, an Expectation-Maximization Likelihood Ratio Test (EM LRT), and a bootstrapped confidence interval approach (EM Bootstrap CI). Using an Artificial Prevalence Protocol – a procedure for varying prevalence in sampled test sets – on two real-world and one simulated dataset, we evaluated the methods based on their calibration and discriminatory power. Our findings indicate that calibration of the underlying classifier is critical for the EM-based methods to achieve a target 5% false positive rate (FPR). The EM LRT, when paired with a calibrated classifier, consistently maintained an FPR near the target and had the highest discriminatory power as measured by the area under the receiver operating characteristic curve. In contrast, the t-test proved to be robust to calibration status of the underlying classifier, though it had discriminatory power lower than the EM LRT and EM Bootstrap CI. This empirical evaluation provides practical guidance on the performance characteristics of different unlabeled class prevalence hypothesis tests, highlighting the EM LRT discriminatory power and the importance of calibration for likelihood-based approaches.

**Keywords:** Hypothesis Testing · Quantification · Detection.

## 1 Introduction

The machine learning field of quantification, also known as class prevalence estimation or learning to quantify, focuses on accurately estimating the proportion of each class within a given unlabeled dataset. While precise point estimates of class prevalence are often the direct goal of quantification, there are numerous practical scenarios where a slightly different objective comes to the forefront: detecting whether a class's prevalence differs significantly from a predefined, nominal value. For instance, in public health, a health authority might not need to know the exact percentage of a population affected by a certain disease, but rather whether its prevalence has crossed an epidemic threshold [13]. Similarly, in quality control, a manufacturer might simply need to know if the defect rate in a batch of products has changed from a historical norm. We refer to this task of detecting changes in class prevalence as unlabeled class prevalence hypothesis testing.

In this paper, we investigate unlabeled class prevalence hypothesis testing as a one-sample, two-tailed hypothesis testing problem. We systematically compare methods for detecting when the prevalence of a binary class significantly deviates from a nominal value in simulated and benchmark datasets.

## 2 Background

The background section is split into two subsections. First, we will discuss prior work in dataset shift detection. Second, we introduce the task of quantification, including likelihood ratio testing methods for label shift detection and methods for constructing confidence intervals.

### 2.1 Dataset Shift Detection in Machine Learning

Dataset shift detection is a very similar task to unlabeled prevalence hypothesis testing. Dataset shift detection seeks to identify if a test set is sampled from a different distribution from the train dataset. Dataset shift between train and test sets can cause machine learning model performance to degrade and has therefore become a focus of machine learning researchers. Machine learning models can be designed to be robust to dataset shift, or dataset shift can be detected and models retrained. Rabanser et al evaluate several methods for detecting various types of dataset shift between train and test sets [20]. Rabanser took a two stage approach. First, the dimensionality of the train and test sets were reduced, then statistical tests comparing the two sets were applied. Several types of dataset shift were evaluated, including label shift, where prevalence of classes shifts between training and test sets. They found that Black Box Shift Detection [17] paired with a two-sample Kolmogorov-Smirnoff test (KS-test) performs consistently well at detecting various types of dataset shift, including label shift.

Unlabeled class prevalence hypothesis testing and label shift detection are related, but different tasks. Label shift detection seeks to detect if training and testing sets were sampled from populations with different class prevalence, while unlabeled class prevalence hypothesis testing detects if the prevalence in an unlabeled set deviates from a specific, nominal value.

### 2.2 Quantification

The goal of quantification is to estimate the prevalence of different classes in an unlabeled dataset, rather than predicting the class of individual samples.

Work on quantification has a long history in several fields, including epidemiology where researchers have long sought to estimate the prevalence of diseases given imperfect assays [14]. One of the key motivations behind quantification is to improve the accuracy of a classification model [9]. In these cases, classifier predictions are treated as posterior probabilities with respect to covariates and, therefore, a function of class priors. When a prior probability change - known as label shift - is detected, these posterior probabilities are adjusted to reflect the newly estimated prior. A common way to estimate a prior is to determine the prior probability that maximizes the likelihood of observing the unlabeled dataset (*eq 1*), where X is a sequence of unlabeled data (often the output of a classifier), and $\omega_i$ represents class $i$.

$$\begin{aligned} Likelihood\,(X) &= \prod P\,(x_i) \\ &= \prod_{i=1}^{N} P\,(x_i|\omega_0)\,P(\omega_0) + P\,(x_i|\omega_1)\,P(\omega_1) \end{aligned} \tag{1}$$

An expectation maximization (EM) algorithm to maximize likelihood with respect to prior probabilities was formalized by Saerens [21] (*eq 2*).

$$\begin{aligned} \hat{p}^{(0)}(\omega_i) &= \hat{p}_t(\omega_i) \\ \hat{p}^{(s)}(\omega_i|x_k) &= \frac{\frac{\hat{p}^{(s)}(\omega_i)}{\hat{p}_t(\omega_i)}\hat{p}_t(\omega_i|x_k)}{\frac{\hat{p}^{(s)}(\omega_0)}{\hat{p}_t(\omega_0)}\hat{p}_t(\omega_0|x_k) + \frac{\hat{p}^{(s)}(\omega_1)}{\hat{p}_t(\omega_1)}\hat{p}_t(\omega_1|x_k)} \\ \hat{p}^{(s+1)}(\omega_i) &= \frac{1}{N}\sum_{k=1}^{N}\hat{p}^{(s)}(\omega_i|x_k) \end{aligned} \tag{2}$$

where $\hat{p}^{(s)}(\omega_i)$ is the prior estimate at the $s^{th}$ iteration of the expectation maximization algorithm, $\hat{p}^{(s)}(\omega_i|x_k)$ are the classifier outputs adjusted for the new prior estimate, and $p_t$ denotes that under the training set. Expectation maximization is an algorithm which maximizes the likelihood of observing our dataset [8]. Though several methods for point estimates of binary class prevalence have since emerged [6], the EM approach has been shown to be difficult to beat, particularly when the underlying classifier is well calibrated [3, 10, 11].

**Likelihood Ratio Tests for Detecting Label Shift** Saerens et al. suggest using a likelihood ratio test to detect label shift [21]. They note that two times the log likelihood ratio of the unlabeled data under the true prevalence vs the maximum likelihood prevalence is distributed according to a chi-squared distribution. Label shift is then detected if the cumulative density function of the chi-squared distribution evaluated at two times the log likelihood ratio exceeds one minus the specified false positive rate (i.e., 0.05).

**Confidence Interval Methods** A fairly straightforward approach to unlabeled class prevalence hypothesis testing is to estimate confidence intervals around class prevalence estimates. The null hypothesis is then rejected when the prior associated with the null hypothesis falls outside of the confidence intervals. Confidence intervals are often constructed in quantification using standard bootstrapping techniques [16, 22]. Daughton and Paul developed the Error-Adjusted Bootstrap method to account for the errors associated with imperfect classifiers [7]. Tasche evaluated the coverage and confidence interval width for confidence intervals generated by bootstrapping several quantification methods [22]. Tasche found that the underlying classifier performance greatly impacted the confidence interval coverage and width.

## 3 Experiments

In this section, we describe the experimental protocols, datasets, and detection methods used.

### 3.1 Experimental Protocol

We run our experiments in an Artificial Prevalence Protocol (APP) [9] in the presence of label shift. An APP creates a variety of prevalences in the test set through random sub-sampling. We assumed that the class-conditional densities of classifier outputs are stationary between the train and the test set. We first split our training and testing datasets with a 50%-50% train-test stratified split. We train our classifiers on the entire training set. We construct our testing set by first selecting the number of positive instances to sample from a binomial distribution with positive class probability set to our artificially selected prevalence. To determine how performance scales, we use a test sample size of of 50, 75, and 100. We then sample, without replacement, the corresponding number of positive and negative samples from the test dataset.

We construct artificial prevalences under which the null hypotheses are true, and under which the true prevalence is 10% higher than that of the null hypothesis. We consider 3 null hypotheses: prevalence equal to 25%, 50%, and 75%. We calculate false positive rates (FPR) as the percent of simulations generated according to the null hypothesis that resulted in rejecting the null hypothesis. Similarly, we calculate true positive rates (TPR) as the percent of simulations generated with prevalence 10% higher than the null hypothesis that resulted in rejecting the null hypothesis. We simulate each scenario 2000 times to get stable estimates of performance. A summary of our experimental protocol can be found in Table 1.

### 3.2 Metrics

We measure the calibration and discriminatory power of each hypothesis test. A well-calibrated test has a false positive rate consistent with the significance level of the test. In our experiment, we report the false positive rate when each test is evaluated at a significance level of 0.05, so tests with false positive rates close to 5% should be considered calibrated. We report discriminatory power of each test as the area under the curve of the receiver operating characteristic (AUC).

### 3.3 Datasets

We use three datasets in our experiments, including the Wisconsin Breast Cancer dataset [1], the Pima Indians Diabetes dataset [2], and a simulated dataset. All three datasets have binary classification targets. Datasets were chosen to have a breadth of input feature space. The Wisconsin Breast Cancer dataset has 30 features, the Pima Indians Diabetes dataset has eight features, and the simulated dataset has a single feature. The simulated dataset was generated by sampling a single feature from normal distributions ($\sigma = 1$) with zero mean for negative classes, and 0.85 mean for positive classes.

### 3.4 Detection Methods

Logistic regression classifiers are trained. We evaluate performance with and without calibration. We calibrate our regression model using Platt's method [19], which fits a logistic regression model of our classifier outputs to the labels across cross-validation folds. Our classifier outputs are then fed into four detection methods: t-test, KS-test, expectation maximization likelihood ratio tests, and bootstrapped confidence intervals around expectation maximization estimates.

**T-Test** T-tests are commonly used to detect differences in population parameters. We perform a two-sided, one-sample t-test, comparing the mean classifier probability output on the unlabeled test set to the expected value under the null hypothesis (population prevalence equals $p_0$). The expected value is computed according to *eq 3*.

$$E[X] = E[X|\omega_1]p_0 + E[X|\omega_0](1 - p_0) \tag{3}$$

**KS-test** Rabanser demonstrated that a two-sample KS-test applied to the output of a classifier can detect label shift from train to test distribution [20]. One-sample KS-tests can be used to test if a univariate sample comes from a specified known distribution by comparing the empirical cumulative density function (CDF) of the sample to the CDF of the known distribution. In this application, the sample of classifier outputs are compared to a mixture CDF of the class conditional distributions under the null hypothesis, defined by *eq 4*.

$$P_{H_0}(X \leq x) = P(X \leq x|\omega_1)p_0 + P(X \leq x|\omega_0)(1 - p_0) \tag{4}$$

**Expectation Maximization Likelihood Ratio Test** We implement the expectation maximization (EM) algorithm proposed by Saerens et al., [21] initializing the iterative algorithm according to our null hypothesis such that $\hat{p}^{(0)} = p_0$. Saerens proposes a log likelihood ratio comparing the likelihood of the maximum likelihood prior $\hat{p}^{(s)}(\omega_i)$ to the prior in the training dataset $\hat{p}_t(\omega_i)$. We modify the likelihood ratio to compare any prior $p_0$ to the maximum likelihood estimate by adjusting $\hat{p}_0(\omega_i|x_k)$ from classifier outputs $\hat{p}_t(\omega_i|x_k)$ according to *eq 5*.

$$\hat{p}_0(\omega_i|x_k) = \frac{\frac{\hat{p}_0(\omega_i)}{\hat{p}_t(\omega_i)}\hat{p}_t(\omega_i|x_k)}{\frac{\hat{p}_0(\omega_0)}{\hat{p}_t(\omega_0)}\hat{p}_t(\omega_0|x_k) + \frac{\hat{p}_0(\omega_1)}{\hat{p}_t(\omega_1)}\hat{p}_t(\omega_1|x_k)} \tag{5}$$

The $\chi^2$ statistic is then calculated according to *eq 6*.

$$2log\frac{\prod_{k=1}^{N}\frac{\hat{p}^{(s)}(\omega_i)}{\hat{p}^{(s)}(\omega_i|x_k)}}{\prod_{k=1}^{N}\frac{\hat{p}_0(\omega_i)}{\hat{p}_0(\omega_i|x_k)}} \tag{6}$$

We then evaluate the $\chi^2$ test with 1 degree of freedom. We refer to this test as the Expectation Maximization Likelihood Ratio Test (EM LRT).

**Bootstrapped Confidence Intervals** Finally, we create bootstrapped confidence intervals. We resample classifier outputs of the unlabeled test set with replacement, then run Saerens' expectation maximization algorithm to calculate our bootstrap prevalence estimate. The $2.5^{th}$ and $97.5^{th}$ quantiles of the bootstrapped prevalence estimates are taken as the confidence interval. If $p_0$ falls outside of the confidence interval, we reject the null hypothesis. We refer to this method as the EM Bootstrap CI.

**Table 1.** Experimental Design Summary

| Parameter | Value(s) |
|---|---|
| Null Hypothesis Prevalence ($p_0$) | 25%, 50%, 75% |
| Test Sample Sizes | 50, 75, 100 |
| Number of Simulations | 2000 |
| Tests | EM LRT, EM Bootstrap CI, t-test, KS-test |
| Metrics | AUC, FPR |

## 4 Results

This section presents the results of the experiments comparing different hypothesis testing methods across various datasets and calibration settings. We first present the classification performance and training set prevalence on the datasets to provide context for the hypothesis testing results. We then report the AUC and FPR by dataset, test sample size, calibration, and unlabled prevalence hypothesis testing method.

**Table 2.** Classification performance and training set class prevalence

| Dataset | AUC | Train Prevalence | Train Set Size (N) |
|---|---|---|---|
| Diabetes | 0.827 | 34.8% | 385 |
| Cancer | 0.993 | 37.2% | 285 |
| Simulated | 0.726 | 50.0% | 1000 |

Table 2 demonstrates that the underlying classifier has a breadth of performance across datasets. Note that, despite evaluating both a calibrated and uncalibrated logistic regression models, we only report a single AUC. This is because Platt's calibration method does not change the AUC of the underlying classifier. The Breast Cancer Wisconsin (Cancer) dataset can be almost perfectly classified using a logistic regression model. Classifier performance was worse for the Pima Indians Diabetes (Diabetes) dataset, and worse still for the simulated dataset. Prevalence of the positive class was similar in the two real-world datasets ( 35%), while the simulated dataset had a balanced training set. There is a range of training set sizes across the datasets, with the simulated dataset being over three times the size of the Diabetes dataset.

**Table 3.** Area under the receiver operating curve and false positive rate for the Pima Indian Diabetes dataset. Top performing methods are in **bold.**

| | | AUC | | | | False Positive Rate | | | |
|---|---|---|---|---|---|---|---|---|---|
| Calibration | test-size | EM LRT | Bootstrap CI | t-test | KS -test | EM LRT | Bootstrap CI | t-test | KS -test |
| Platt's | 50 | **0.6034** | 0.6019 | 0.5830 | 0.5770 | **6.25%** | 7.38% | 6.48% | 8.45% |
| | 75 | **0.6368** | 0.6351 | 0.6105 | 0.6039 | **7.63%** | 8.28% | 7.83% | 10.73% |
| | 100 | **0.6623** | 0.6604 | 0.6359 | 0.6195 | **7.75%** | 8.48% | 8.23% | 11.63% |
| None | 50 | **0.6283** | 0.6238 | 0.5869 | 0.5756 | 9.00% | 11.55% | **6.43%** | 8.50% |
| | 75 | **0.6577** | 0.6528 | 0.6143 | 0.5997 | 11.23% | 12.90% | **7.70%** | 11.25% |
| | 100 | **0.6847** | 0.6790 | 0.6410 | 0.6152 | 13.65% | 14.93% | **8.07%** | 12.28% |

Table 3 displays the AUC and FPR for the Pima Indian Diabetes dataset. This dataset had a moderate number of samples (385 in the training set), and the underlying classifier performance was moderate (AUC: 0.827). For this dataset, EM LRT was the most skilled (AUC: 0.6034 - 0.6847), followed by the EM Bootstrap CI (AUC: 0.6019 - 0.6790) across different test sizes. The t-test was marginally inferior (AUC: 0.5830 - 0.6410), and the KS-test was substantially

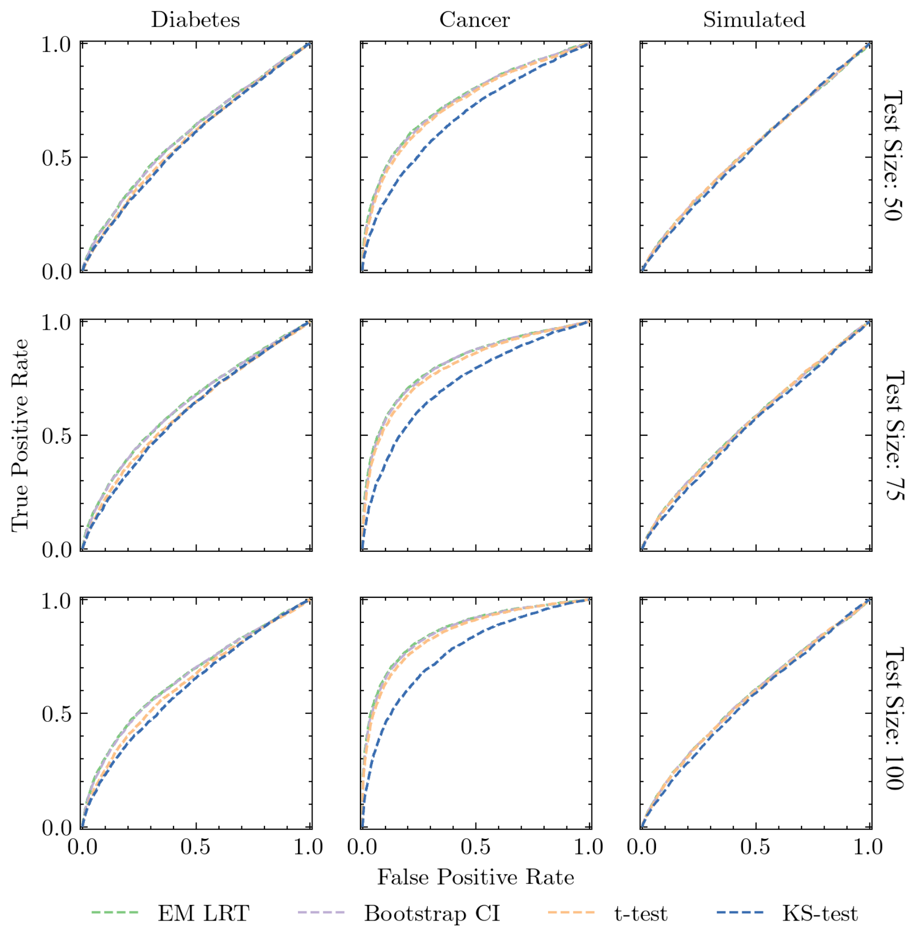

**Fig. 1.** Receiver operating characteristic curves show that across all tests, skill improves as sample size increases and performance of the underlying classifier improves.

worse (0.5770 - 0.6152). Notably, the FPRs for the EM LRT, EM Bootstrap CI, and t-test with calibration were close to or slightly above the target 5%, ranging from 6.25% to 8.48%, while the KS-test was poorly calibrated. Without calibration, EM LRT and EM Bootstrap CI achieved much higher FPRs (9.00% - 14.93%).

**Table 4.** Area under the receiver operating characteristic curve and false positive rate for the Breast Cancer Wisconsin dataset. Top performing methods are in **bold.**

| | | AUC | | | | False Positive Rate | | | |
|---|---|---|---|---|---|---|---|---|---|
| Calibration | test-size | EM LRT | Bootstrap CI | t-test | KS -test | EM LRT | Bootstrap CI | t-test | KS -test |
| Platt's | 50 | **0.7525** | 0.7491 | 0.7387 | 0.6781 | **5.20%** | 6.07% | 5.67% | 8.47% |
| | 75 | **0.8193** | 0.8165 | 0.8036 | 0.7314 | **5.17%** | 6.08% | 5.30% | 10.90% |
| | 100 | **0.8667** | 0.8633 | 0.8527 | 0.7727 | **5.65%** | 6.43% | 5.88% | 13.20% |
| None | 50 | **0.7555** | 0.7462 | 0.7284 | 0.6795 | 7.23% | 9.28% | **5.78%** | 8.33% |
| | 75 | **0.8129** | 0.8040 | 0.7899 | 0.7316 | 7.95% | 9.80% | **5.67%** | 10.85% |
| | 100 | **0.8478** | 0.8382 | 0.8411 | 0.7751 | 9.65% | 11.23% | **6.47%** | 13.00% |

Table 4 shows the AUC and FPR for the Breast Cancer Wisconsin dataset. This dataset had a moderate number of samples (285 in the training set), and the underlying classifier performance was nearly perfect (AUC: 0.993). The EM LRT method achieved the highest AUC across all evaluated conditions, indicating superior discriminatory power. This superiority was maintained regardless of test size or the application of Platt's calibration. As the test size increased from 50 to 100, the AUC for all methods showed a corresponding improvement. For example, with Platt's calibration, the AUC for the EM LRT increased from 0.7525 to 0.8667. Analysis of the false positive rates (FPR) revealed that the optimal method was dependent on the calibration strategy. When Platt's scaling was applied, the EM LRT method consistently produced the lowest FPR, with rates as low as 5.17%. In contrast, without calibration, the t-test yielded the lowest FPR, maintaining a rate between 5.67% and 6.47%. In all scenarios, the KS-test exhibited the highest FPR.

Table 5 presents the AUC and FPR for the simulated dataset. This dataset had a large number of samples (1000 in the training set) and the underlying classifier performance was poor (AUC: 0.726). In contrast to the other datasets, no single method consistently achieved the highest AUC. Top performing methods in terms of discriminatory power varied with test size. For example, under Platt's calibration, the t-test was marginally superior at a test size of 50 (AUC = 0.5460), while the EM LRT and EM Bootstrap CI methods performed best at test sizes of 75 (AUC = 0.5651) and 100 (AUC = 0.5781), respectively. A consistent trend across all methods was the modest improvement in AUC with increasing test size. The KS-test consistently yielded the lowest AUC values.

**Table 5.** Area under the receiver operating characteristic curve and false positive rate for the simulated dataset. Top performing methods are in **bold.**

| | | AUC | | | | False Positive Rate | | | |
|---|---|---|---|---|---|---|---|---|---|
| Calibration | test-size | EM LRT | Bootstrap CI | t-test | KS -test | EM LRT | Bootstrap CI | t-test | KS -test |
| Platt's | 50 | 0.5445 | 0.5446 | **0.5460** | 0.5390 | 5.70% | 7.13% | 5.95% | 6.18% |
| | 75 | **0.5651** | 0.5648 | 0.5622 | 0.5484 | 5.95% | 6.62% | 5.87% | 6.75% |
| | 100 | 0.5779 | **0.5781** | 0.5753 | 0.5628 | 6.38% | 6.92% | 6.52% | 7.45% |
| None | 50 | 0.5449 | 0.5439 | **0.5457** | 0.5388 | **5.20%** | 8.08% | 5.95% | 6.37% |
| | 75 | **0.5636** | 0.5625 | 0.5627 | 0.5487 | 6.07% | 7.50% | **6.02%** | 7.07% |
| | 100 | 0.5742 | 0.5730 | **0.5744** | 0.5619 | 7.23% | 8.20% | **6.45%** | 7.58% |

The control of the FPR was also variable. Without calibration, the EM LRT was most effective at the smallest test size (FPR = 5.20%), whereas the t-test was most effective at larger test sizes. With Platt's scaling, no single method consistently produced the lowest FPR. Overall, on the simulated dataset, the EM LRT, Bootstrap CI, and t-test methods yielded comparable results, with the optimal choice for both AUC and FPR being contingent on the specific test size.

## 5 Discussion

When using the outputs of a well calibrated logistic regression model, the EM LRT method most consistently yielded a false positive rate close to the specified 5%, though it was consistently above 5%. The EM Bootstrap CI method consistently resulted in a false positive rate higher than the specified 5%. This is in disagreement with Tasche's findings, which generally found that bootstrapping the EM estimate had adequate coverage [22]. This disagreement might arise from the fact that we are evaluating the confidence interval at a larger width (95%) than Tasche (90%), leading to less stable estimates. Additionally, Tasche only simulated 100 runs compared to our 2000, and it's possible that the minor FPR excess (no more than 3% among calibrated models) did not appear in Tasche's study due to stochasticity.

Calibration proved to be important in achieving false positive rate close to the expected value for the methods involving EM, particularly among the real-world datasets with smaller sample sizes. This is in line with Esuli et al., who found performance of point estimates of prevalence using the EM algorithm were highly sensitive to calibration of underlying classifiers [10]. The t-test method performance was robust to calibration status, performing nearly identically when applied to the calibrated vs uncalibrated model outputs. In applications where reliable calibration of the underlying classifier is not tenable, a t-test could be utilized.

The EM LRT demonstrated the greatest skill in terms of area under the receiver operating characteristic curve across the two benchmark datasets and all sample sizes. Detection skill was correlated with the skill of the underlying classifier, and the sample size, consistent with previous work [18, 22]. While the EM LRT performed the best in our experiments, it can only be utilized in two-sided hypothesis testing. For one-sided hypothesis testing, the EM Bootstrap CI could be employed as it performed similarly to EM LRT.

There are several limitations to this study. Use of an artificial prevalence protocol may create unreasonable and highly unlikely prevalences that do not reflect reality [12]. Additionally, we have not considered types of dataset shift outside of prior shift. Covariate shift and concept shift would likely affect our results, as González et al. found the EM algorithm to degrade in performance in the presence of these shifts [15]. Finally, our analysis was limited to binary classes and neglected more complex topics in quantification including multiclass quantification (for example, see [4]) or ordinal quantification (for example see [5]).

## 6 Conclusion

This paper systematically compared various methods for unlabeled class prevalence hypothesis testing in binary classification tasks using both real-world and simulated datasets. The findings highlight the importance of calibration, particularly for EM based methods, in achieving false positive rates that align with the desired 5% target. EM LRT generally provided the greatest discriminative power with the best calibration. While the EM LRT consistently produced false positive rates close to 5% with a well-calibrated logistic regression model, the EM Bootstrap Confidence Interval (EM Bootstrap CI) method often resulted in higher FPRs, possibly due to the wider confidence interval evaluated and increased simulation runs compared to prior work. The t-test, in contrast, demonstrated robust performance independent of calibration status. This empirical evaluation provides valuable insights into the performance characteristics of different prior shift detection techniques, contributing to the limited existing literature on this crucial task.

## References

1. Breast Cancer Wisconsin (Diagnostic) Data Set, `https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data`
2. Pima Indians Diabetes Database, `https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database`
3. Alexandari, A., Kundaje, A., Shrikumar, A.: Maximum Likelihood with Bias-Corrected Calibration is Hard-To-Beat at Label Shift Adaptation (Jun 2020). `https://doi.org/10.48550/arXiv.1901.06852`, `http://arxiv.org/abs/1901.06852`, arXiv:1901.06852 [cs]

4. Bunse, M.: On multi-class extensions of adjusted classify and count. In: Proceedings of the 2nd International Workshop on Learning to Quantify (LQ 2022). pp. 43–50 (2022), https://lq-2022.github.io/proceedings/Bunse2022b.pdf
5. Bunse, M., Moreo, A., Sebastiani, F., Senz, M.: Ordinal Quantification Through Regularization. In: Amini, M.R., Canu, S., Fischer, A., Guns, T., Kralj Novak, P., Tsoumakas, G. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 36–52. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-26419-1_3
6. Castaño, A., Alonso, J., González, P., Coz, J.J.d.: An Equivalence Analysis of Binary Quantification Methods. Proceedings of the AAAI Conference on Artificial Intelligence **37**(6), 6944–6952 (Jun 2023). https://doi.org/10.1609/aaai.v37i6.25849, https://ojs.aaai.org/index.php/AAAI/article/view/25849
7. Daughton, A.R., Paul, M.J.: A bootstrapping approach to social media quantification. Social Network Analysis and Mining **11**(1), 73 (Aug 2021). https://doi.org/10.1007/s13278-021-00760-0, https://doi.org/10.1007/s13278-021-00760-0
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum Likelihood from Incomplete Data Via the *EM* Algorithm. Journal of the Royal Statistical Society Series B: Statistical Methodology **39**(1), 1–22 (Sep 1977). https://doi.org/10.1111/j.2517-6161.1977.tb01600.x, https://academic.oup.com/jrsssb/article/39/1/1/7027539
9. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to Quantify, The Information Retrieval Series, vol. 47. Springer International Publishing, Cham (2023). https://doi.org/10.1007/978-3-031-20467-8, https://link.springer.com/10.1007/978-3-031-20467-8
10. Esuli, A., Molinari, A., Sebastiani, F.: A Critical Reassessment of the Saerens-Latinne-Decaestecker Algorithm for Posterior Probability Adjustment. ACM Transactions on Information Systems **39**(2), 1–34 (Apr 2021). https://doi.org/10.1145/3433164, https://dl.acm.org/doi/10.1145/3433164
11. Esuli, A., Moreo, A., Sebastiani, F., Sperduti, G.: A detailed overview of LeQua 2022: learning to quantify. In: Working Notes of the 13th conference and labs of the evaluation forum (CLEF 2022). Bologna, Italy (2022)
12. Esuli, A., Sebastiani, F.: Optimizing Text Quantifiers for Multivariate Loss Functions. ACM Trans. Knowl. Discov. Data **9**(4), 27:1–27:27 (Jun 2015). https://doi.org/10.1145/2700406, https://doi.org/10.1145/2700406
13. Evans, C., G'Sell, M.: Sequential label shift detection in classification data: An application to dengue fever. PLOS ONE **19**(9), e0310194 (Sep 2024). https://doi.org/10.1371/journal.pone.0310194, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11404796/
14. Gart, J.J., Buck, A.A.: Comparison of a screening test and a reference test in epidemiologic studies. II. A probabilistic model for the comparison of diagnostic tests. American Journal of Epidemiology **83**(3), 593–602 (May 1966). https://doi.org/10.1093/oxfordjournals.aje.a120610
15. González, P., Moreo, A., Sebastiani, F.: Binary quantification and dataset shift: an experimental investigation. Data Mining and Knowledge Discovery **38**(4), 1670–1712 (Jul 2024). https://doi.org/10.1007/s10618-024-01014-1, https://doi.org/10.1007/s10618-024-01014-1
16. Hopkins, D.J., King, G.: A Method of Automated Nonparametric Content Analysis for Social Science. American Journal of Political Science **54**(1), 229–247 (2010). https://doi.org/10.1111/j.1540-5907.2009.00428.x,

https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5907.2009.00428.x, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-5907.2009.00428.x

17. Lipton, Z., Wang, Y.X., Smola, A.: Detecting and Correcting for Label Shift with Black Box Predictors. In: Proceedings of the 35th International Conference on Machine Learning. pp. 3122–3130. PMLR (Jul 2018), https://proceedings.mlr.press/v80/lipton18a.html, iSSN: 2640-3498
18. Maletzke, A., Hassan, W., Reis, D.d., Batista, G.: The Importance of the Test Set Size in Quantification Assessment. vol. 3, pp. 2640–2646 (Jul 2020). https://doi.org/10.24963/ijcai.2020/366, https://www.ijcai.org/proceedings/2020/366, iSSN: 1045-0823
19. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in Large Margin Classifiers pp. 61–74 (1999)
20. Rabanser, S., Günnemann, S., Lipton, Z.: Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://papers.nips.cc/paper_files/paper/2019/hash/846c260d715e5b854ffad5f70a516c88-Abstract.html
21. Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the Outputs of a Classifier to New a *Priori* Probabilities: A Simple Procedure. Neural Computation **14**(1), 21–41 (Jan 2002). https://doi.org/10.1162/089976602753284446, https://direct.mit.edu/neco/article/14/1/21-41/6577
22. Tasche, D.: Confidence Intervals for Class Prevalences under Prior Probability Shift. Machine Learning and Knowledge Extraction **1**(3), 805–831 (Sep 2019). https://doi.org/10.3390/make1030047, https://www.mdpi.com/2504-4990/1/3/47, number: 3 Publisher: Multidisciplinary Digital Publishing Institute

# Recalibrating binary probabilistic classifiers

Dirk Tasche

Centre for Business Mathematics and Informatics, North-West University, South Africa
55801447@nwu.ac.za

**Abstract.** Recalibration of binary probabilistic classifiers to a target prior probability is an important task in areas like credit risk management. We analyse methods for recalibration from a distribution shift perspective. Distribution shift assumptions linked to the area under the curve (AUC) of a probabilistic classifier are found to be useful for the design of meaningful recalibration methods. Two new methods called parametric covariate shift with posterior drift (CSPD) and ROC-based quasi moment matching (QMM) are proposed and tested together with some other methods in an example setting. The outcomes of the test suggest that the QMM methods discussed in the paper can provide appropriately conservative results in evaluations with concave functions like for instance risk weights functions for credit risk.

**Keywords:** Probabilistic classifier · posterior probability · prior probability · calibration · distribution shift · dataset shift · credit risk.

## 1 Introduction

Occasionally binary probabilistic classifiers are learned on a training dataset and then are applied to a test dataset which reflects a joint distribution of features and labels different than the distribution of the training dataset. Actually, quite often it is unknown how much the training and test distributions differ because for the instances in the test dataset only the features but not the labels can be observed. Hence, the feature training and test distributions might be different while the posterior probabilities are identical for the training and test datasets. This kind of dataset shift is called covariate shift and is rather benign in principle as it would not require any change of the probabilistic classifier (Storkey [21]). If however, another type of dataset shift other than covariate shift is incurred, applying the classifier learned on the training dataset to the instances in the test dataset without any changes risks to generate unreliable predictions of the labels.

In this paper, we study the situation where indeed no labels are observed in the test dataset but where there exists an estimate of the proportion of positive labels, i.e. an estimate of the test prior probability of the positive class. The problem is then to *recalibrate* the probabilistic classifier learned on the training dataset such that the mean of the recalibrated classifier on the test dataset matches the estimate of the positive labels proportion.

This is a common situation in credit risk management, see for instance Chapter 4 of Bohn and Stein [3]. So called probabilities of default (PDs) are estimated on a training sample with observations of solvent and defaulted borrowers and must be recalibrated before being evaluated for the borrowers in a live portfolio. The future solvency states of these borrowers in general are unknown but typically an estimate of the proportion of borrowers who are going to default is available. Sometimes, such estimates are conservative, i.e. they are likely to significantly overestimate the proportion of defaulters. Conservatism of the estimates can be a regulatory requirement or it could be part of stress testing exercises intended to assess the impact of unfavourable economic circumstances on the portfolio.

Recalibration of posterior probabilities learned on a training dataset to a target prior probability on a test dataset is not a well-defined problem because there is more than one way to transform the original posterior probabilities such that the target is matched. Therefore, we are going to study the impact of the recalibration method on the values of concave or nearly concave functions of the posterior probabilities as an additional criterion to identify meaningful solutions. The risk weight functions for the calculation of minimum required capital under the "internal ratings based (IRB) approach" of the Basel credit risk framework (BCBS [2]) are a primary example of such functions. The findings of this paper can inform the choice of the recalibration method in credit risk management and similar contexts.

This paper is organised as follows: Section 2 puts the paper into the context of related work. Section 3 describes the technical details of the setting of the paper and the recalibration problem. In addition, it introduces the evaluation of the solutions with a concave function as a criterion for assessing their appropriateness. Section 4 presents a number of methods for recalibration. Parametric CSPD (Section 4.4) and ROC-based QMM (Section 4.5) seem to be new. With an example in Section 5, we illustrate the dependence of the solutions to the recalibration problem on assumptions of distribution shift and identify some less reliable recalibration methods. Section 6 proposes a way forward to prudent recalibration and concludes the paper. Appendix A provides additional technical details needed for the implementation of some of the methods discussed in Section 4.

## 2 Related work

Calibration of probabilistic classifiers has often been treated in the literature, see the surveys by Ojeda et al. [14] and Silva Filho et al. [20]. In the typical calibration setting, a real-valued score is learned on a training dataset with joint observations of instances and labels. The score is then *calibrated* or *mapped* to become a probabilistic classifier on a test (or validation or calibration) dataset for which there are also joint observations of instances and labels. If the original score is already a probabilistic classifier then the term recalibration is sometimes used instead of calibration.

Cautious calibration (Allikivi et al. [1]) is a variant of binary calibration with the goal to avoid either overconfidence or underconfidence of probabilistic classifiers. Like for calibration, it is assumed that the labels of the instances in the calibration and test datasets are known. A common feature with recalibration as defined in this paper is conservatism of the posterior estimates. In the setting of this paper, conservatism can be achieved by choosing an appropriate value for the target class-1 prior probability.

Recalibration of probabilities of default (PDs) on a dataset without observation of labels but with knowledge of the prior probability of positive labels has been a topic of research for twenty or more years in credit risk (Bohn and Stein [3] and the references therein). Such recalibration may be considered an extreme case of learning with label proportions (Quadrianto et al. [16]) where there are no individual label observations but label proportions for groups of instances are available. In general, 'learning with label proportions' in the binary case requires that there are at least two groups of instances with different proportions of positive labels such that results from the related research are not applicable to the recalibration problem as studied in this paper.

Quantification (or class distribution estimation, CDE) is another related problem. See Esuli et al. [8] for a recent survey. The goal of CDE in the binary case is to estimate the proportion of positive labels in a test dataset without any information on the labels. The primary common feature of the CDE and recalibration problems is the dependence of the solutions upon assumptions on the type of distribution shift between the training and test datasets. However, Remark 1 in Section 4.4 shows that the one-parameter version of the recalibration method 'parametric CSPD' may also be used for CDE.

Covariate shift with posterior drift (CSPD) as introduced by Scott [19] turns out to be a useful assumption on the type of distribution shift for tackling the recalibration problem. See Sections 4.4 and 5 below.

## 3 Setting

In this paper, we consider binary, i.e. two-class classification problems for which we assume the following setting:

- There are a class variable $Y$ with values in $\mathcal{Y} = \{0, 1\}$ and a features (also called covariates) vector $X$ with values in $\mathcal{X}$. Each example (or instance) to be classified has a class label $Y$ and features $X$.
- In the training dataset, for all examples their features $X$ and labels $Y$ are observed. $P$ denotes the training joint distribution, also called source distribution, of $(X, Y)$ from which the training dataset has been sampled.
- In the test dataset, only the features $X$ of an example can immediately be observed. Its label $Y$ becomes known only with delay or not at all. $Q$ denotes the test joint distribution, also called target distribution, of $(X, Y)$ from which the test dataset has been sampled.
- For the sake of a more concise notation, we write for short $p = P[Y = 1]$ and $q = Q[Y = 1]$ and assume $0 < p < 1$ and $0 < q < 1$.

We also use the notation $E_P[Z] = \int Z\,dP$ and $E_Q[Z] = \int Z\,dQ$ for integrable real-valued random variables $Z$.

The setting described above is called *dataset shift* (Storkey [21]) or *distribution shift* (Lipton et al. [12]) if source and target distribution of $(X, Y)$ are not the same, i.e. in case of $P(X, Y) \neq Q(X, Y)$.

### 3.1 Recalibration

We assume that since the joint source distribution $P(X, Y)$ of features $X \in \mathcal{X}$ and class $Y \in \{0, 1\} = \mathcal{Y}$ is given, also the posterior probability $\eta_P(X) = P[Y = 1 \,|\, X]$ is known or can be estimated from the training dataset. $\eta_P(X)$ is a special case of probabilistic classifiers $\eta$ which are real-valued statistics with $0 \le \eta \le 1$ and typically intended to approximate $\eta_P(X)$. Probabilistic classifiers for their part are special cases of scores (or scoring classifiers) which are real-valued statistics intended to provide a ranking of the instances in the dataset in the sense that a high score suggests a high likelihood that the instance has class label 1 (positive label).

By assumption only the target marginal feature distribution $Q(X)$ is observed while the target joint distribution $Q(X, Y)$ is unknown. Nonetheless, for the recalibration problem, we also assume the target marginal label distribution, specified by $q = Q[Y = 1]$ to be known. This is not in contradiction to $Q(X, Y)$ being unknown because in general the ensemble of marginal distributions does not uniquely determine the joint distribution. We will encounter an example for this phenomenon in Section 5 below.

The goal is to fit a posterior probability $\eta_Q(x) = Q[Y = 1 \,|\, X = x]$ as some transformation $T$ of $\eta_P(x)$ such that in particular it holds that

$$q \;=\; E_Q[\eta_Q(X)] \;=\; E_Q\big[T(\eta_P(X))\big]. \tag{1a}$$

In the following, we call this problem *recalibration* of the posterior probability $\eta_P(X)$ to a new prior probability $q$ of class 1 under the target distribution.

Note that the assumption

$$\eta_Q(X) \;=\; T\big(\eta_P(X)\big) \tag{1b}$$

appears quite natural but actually is rather strong. Indeed, by Theorems 32.5 and 32.6 of Devroye et al. [7], (1b) is equivalent to $\eta_P(X)$ being sufficient for $X$ with respect to $Y$ under $Q$, i.e. $Q[Y = 1 \,|\, X] \;=\; Q[Y = 1 \,|\, \eta_P(X)]$. This sufficiency property may be interpreted as 'the information provided by $\eta_P(X)$ about $Y$ is as good as the information by the whole set of features $X$ under the target distribution $Q$'.

### 3.2 Non-uniqueness of recalibration

The recalibration problem is not well-posed in the sense that its solution is not unique. Therefore, we study it in a context where underestimating $E_Q\big[C(\eta_Q(X))\big]$ for some fixed concave function $C : [0, 1] \to \mathbb{R}$ ought to be avoided. In general, it holds that (by Jensen's inequality and Lemma 1.2 of Lalley [11])

$$(1 - q)\, C(0) + q\, C(1) \;\le\; E_Q\big[C(Z)\big] \;\le\; C(q) \tag{2}$$

for any random variable $0 \le Z \le 1$ with $E_Q[Z] = q$, and $Z = \eta_Q(X)$ in particular. The maximum value $C(q)$ is taken for constant $Z = q$, the minimum value $(1 - q)\, C(0) + q\, C(1)$ is realised for $Z$ with $Q[Z = 1] = q = 1 - Q[Z = 0]$.

However, assuming a distribution shift between source $P$ and target $Q$ which results in a constant posterior probability $\eta_Q(x) = q$ appears too restrictive in most real world environments. In Section 5 below, we demonstrate that assuming preservation of classification performance as measured by AUC (Area Under the Curve, see next section) between source $P$ and target $Q$ strikes a sensible note between too restrictive and too tolerant assumptions for the estimation of $E_Q\big[C(\eta_Q(X))\big]$ under the target features distribution.

## 3.3 Area Under the Curve (AUC)

AUC (Area Under the Curve[1]) is a popular measure of performance of a binary classifier, i.e. AUC is considered an appropriate measure of the classifier's ability to predict the true class label of an instance. See Chen et al. [5] for related comments. Since AUC plays an important role in some of the recalibration methods discussed in the following sections, we present here the population-level (in contrast to sample-based) representations of AUC that are used in this paper.

Let $S = h(X)$ be a score which is a function of the features with values in an ordered set $\boldsymbol{\mathcal{S}}$. Define the class-conditional distributions $P_y$, $y \in \mathcal{Y} = \{0, 1\}$, of $S$ by $P_y[S \in M] = P[S \in M \,|\, Y = y]$, for all measurable $M \subset \boldsymbol{\mathcal{S}}$.

If the score $S$ is assumed to be large for instances with high likelihood to have class 1 and small for instances with high likelihood to have class 0, then the AUC for $S$ is defined as

$$AUC_S \;=\; P^*[S_1 > S_0] + \frac{1}{2}\,P^*[S_1 = S_0], \tag{3a}$$

where $P^*$ denotes the product measure of $P_1$ and $P_0$, and $S_1$ and $S_0$ are the coordinate projections of the space $\boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{S}}$ on which $P^*$ is defined. As a consequence, $S_1$ and $S_0$ are independent and $P^*[S_y \le s] = P_y[S \le s]$ for $y \in \{0, 1\}$ and all $s \in \boldsymbol{\mathcal{S}}$.

By Definition (3a), on the one hand $AUC_S$ is "equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance" (p. 868 of Fawcett [9]) if both of the class-conditional distribution functions of the score $S$ are continuous. On the other hand, it holds that $AUC_S = \frac{1}{2}$ for any uninformative score $S$ – i.e. in the case $P_0 = P_1$ – even if both of the class-conditional score distribution functions have discontinuities. Note that computing $AUC_S$ by means of (3a) at first glance requires knowledge of the joint distribution $P(S, Y)$ of $S$ and $Y$ which would have to be inferred from a sample of paired $(S, Y)$ observations.

However, $AUC_S$ can also be determined if the distribution $P(S)$ of $S$ and the posterior probabilities $\eta_P(S) = P[Y = 1 \,|\, S]$ are known. Then with $p =$

[1] 'Curve' refers to *ROC* (Receiver Operating Characteristic). See Fawcett [9] and the references therein for the most common definitions of ROC and AUC.

$E_P[\eta_P(S)]$ it holds that[2]

$$\begin{aligned} P_1[S \in M] &= \frac{E_P\big[\eta_P(S)\,\mathbf{1}_M(S)\big]}{p} \qquad \text{and} \\ P_0[S \in M] &= \frac{E_P\big[(1-\eta_P(S))\,\mathbf{1}_M(S)\big]}{1-p} \end{aligned} \tag{3b}$$

for all measurable sets $M \subset \boldsymbol{\mathcal{S}}$. Plug $P_0$ and $P_1$ from (3b) in (3a) to compute $AUC_S$.

To indicate the way $AUC_S$ is computed, in this paper we refer simply to AUC if $AUC_S$ is assumed to be computed or inferred by means of (3a) irrespectively of the origin of $P_0$ and $P_1$. We refer to *implied* AUC if $AUC_S$ is assumed to be computed by means of (3b) in combination with (3a). See Appendix A.1 for a more detailed formula for implied AUC in the case of a discrete-valued score $S$.

## 4 Approaches to recalibration

Any solution as in (1b) to the recalibration problem together with the marginal feature distribution $Q(X)$ completely determines the target distribution $Q(X,Y)$. Since $Q(X,Y) \neq P(X,Y)$ in case $q \neq p$, any given solution specifies some distribution shift between the source and target distributions. Hence, in order to better understand the consequences of selecting a particular solution transformation $T$, it is natural to explore the solution approaches through the lens of distribution shift. In the following, we assume $0 < \eta_P(x) < 1$ for all $x \in \mathcal{X}$.

### 4.1 Recalibration under assumption of stretched or compressed covariate shift

At first glance recalibration of $\eta_P(X)$ to a fixed target prior probability $q$ might appear to be straightforward: Just define $\eta_Q(x) = \frac{q}{E_Q[\eta_P(X)]}\,\eta_P(x)$ for $x \in \mathcal{X}$, then $E_Q[\eta_Q(X)] = q$ immediately follows. This approach is called *scaling* (Section 3.1 of Ptak-Chmielewska and Kopciuszewski [15]).

Unfortunately, in some cases there is a problem with this approach: $1 < \eta_Q(x)$ may be incurred in the case $q > E_Q[\eta_P(X)]$. To avoid this issue, one can modify the approach to become

$$\eta_Q(x) = \min\big(t\,\eta_P(x), 1\big), \tag{4a}$$

with $t > 0$ being determined by

$$q \;=\; E_Q\big[\min\big(t\,\eta_P(x), 1\big)\big]. \tag{4b}$$

(4a) could be called *capped scaling* of the source posterior probabilities. Obviously, (4a) implies that (1b) is satisfied with $T(\eta) = \min(t\,\eta, 1)$.

Recall that source distribution $P(X,Y)$ and target distribution $Q(X,Y)$ are related through *covariate shift* (in the sense of Storkey [21]) if it holds that

[2] The *indicator function* $\mathbf{1}_A$ is defined by $\mathbf{1}_A(a) = 1$ if $a \in A$ and $\mathbf{1}_A(a) = 0$ if $a \notin A$.

$P[Y = y \,|\, X] = Q[Y = y \,|\, X]$ for all $y \in \mathcal{Y}$ with probability 1 both under $P$ and $Q$. Hence if $t > 1$ in (4a), one might call the implied distribution shift *stretched covariate shift*.

In case $t < 1$ the induced distribution shift could be considered *compressed covariate shift*. In case $q > E_Q[\eta_P(X)]$, (4a) and (4b) imply $t > 1$ and $\eta_Q(x) = 1$ for all $x$ with $t\,\eta_P(x) \geq 1$. This may have the consequence of an unjustified increase of $AUC_{\eta_Q(X)}$ compared to $AUC_{\eta_P(X)}$ for the area under the curve AUC defined as in Section 3.3 below.

## 4.2 Recalibration under assumption of label shift

Source distribution $P(X, Y)$ and target distribution $Q(X, Y)$ are related through *label shift* (Lipton et al. [12]), previously called *prior probability shift* in the literature (Storkey [21]), if $P[X \in M \,|\, Y = y] = Q[X \in M \,|\, Y = y]$ for all measurable sets $M \subset \mathcal{X}$ and $y \in \mathcal{Y}$.

Under the assumption of label shift, the target feature distribution $Q(X)$ can be represented as

$$Q[X \in M] \;=\; q\,P[X \in M \,|\, Y = 1] + (1-q)\,P[X \in M \,|\, Y = 0], \tag{5}$$

for all measurable sets $M \subset \mathcal{X}$. If a prior probability $q = Q[Y = 1]$ is given, then according to the *posterior correction formula* (Eq. (2.4) of Saerens et al. [18]), $\eta_Q$ is determined through (recall $p = P[Y = 1]$)

$$\eta_Q(x) \;=\; \frac{\frac{q}{p}\,\eta_P(x)}{\frac{q}{p}\,\eta_P(x) + \frac{1-q}{1-p}\,(1-\eta_P(x))}, \tag{6}$$

for all $x \in \mathcal{X}$ with probability 1 under $Q$. In the credit risk community, the use of (6) for recalibration is popular (Section "Calibrating to PDs" of Bohn and Stein [3], Section 3.1 of Ptak-Chmielewska and Kopciuszewski [15]) because it avoids the problem of $\eta_Q(X)$ potentially taking the value 1 for large $\eta_P(X)$ which may be encountered with capped scaling as in (4a). Cramer [6] (Sections 6.2 and 6.3) pointed out that in the context of logistic regression the related special case of (6) was known at least since 1979. Note that (6) implies (1b) with $T(\eta) = \frac{\frac{p}{q}\,\eta}{\frac{p}{q}\,\eta + \frac{1-p}{1-q}\,(1-\eta)}$ strictly increasing in $\eta$.

Under the label shift assumption, the following observation is well-known. Define AUC (area under the curve) as in Section 3.3.

**Proposition 1.** *Define the score $S$ by $S = \eta_P(X)$ and the score $S^*$ by $S^* = \eta_Q(X)$. If $P$ and $Q$ are related through label shift then it follows that $AUC_S = AUC_{S^*}$.*

Proposition 1 is a consequence of (3a) in Section 3.3 as well as (1b) and (6). According to Proposition 1, recalibration under the assumption of label shift leaves the implied performance under the target distribution – sometimes called *discriminatory power* in the credit risk management literature (e.g. Bohn and Stein [3]) – unchanged when compared to the implied performance under the source distribution. In particular, Proposition 1 implies a necessary criterion for distribution shift to be label shift. Accordingly, in case $AUC_S \neq AUC_{S^*}$ source distribution $P$ and target distribution $Q$ cannot be related through label shift.

### 4.3 Recalibration under assumption of factorizable joint shift (FJS)

According to He at al. [10] and Tasche [25], the source distribution $P(X, Y)$ and the target distribution $Q(X, Y)$ are related through *factorizable joint shift* (FJS) if there are functions $g : \mathcal{X} \to [0, \infty)$ and $b : \mathcal{Y} \to [0, \infty)$ such that $(x, y) \mapsto g(x)\, b(y)$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ is a density of $Q(X, Y)$ with respect to $P(X, Y)$, i.e. it holds that $Q[(X, Y) \in M] = E_P\big[\mathbf{1}_M(X, Y)\, g(X)\, b(Y)\big]$ for all measurable sets $M \subset \mathcal{X} \times \mathcal{Y}$. See footnote 2 for the definition of the indicator $\mathbf{1}_M$.

By Corollary 4 of Tasche [25] and subject to mild technical conditions, under FJS the joint target distribution $Q(X, Y)$ is given by the target feature distribtuion $Q(X)$ and the class 1 posterior probability

$$\eta_Q(X) \;=\; \frac{\frac{q}{p}\, \eta_P(X)}{\frac{q}{p}\, \eta_P(X) + \frac{1}{\varrho}\, \frac{1-q}{1-p}\, (1 - \eta_P(X))}, \tag{7a}$$

where $0 < \frac{p}{(1-p)\, E_Q\left[\frac{\eta_P(X)}{1-\eta_P(X)}\right]} \le \varrho \le \frac{p}{(1-p)}\, E_Q\left[\frac{1-\eta_P(X)}{\eta_P(X)}\right]$ and $\varrho$ is the unique solution to the equation

$$q \;=\; E_Q\left[\frac{\frac{q}{p}\, \eta_P(X)}{\frac{q}{p}\, \eta_P(X) + \frac{1}{\varrho}\, \frac{1-q}{1-p}\, (1 - \eta_P(X))}\right]. \tag{7b}$$

Note that recalibration of $\eta_P(X)$ to a target prior probability $q$ under the assumption of FJS works for arbitrary target feature distributions $Q(X)$. This is in stark contrast to recalibration under the assumption of label shift which only works when assuming that $Q(X)$ is given by (5). However, by Proposition 1 recalibration under the label shift assumption entails $AUC_{\eta_Q(X)} = AUC_{\eta_P(X)}$. The example in Section 5 below shows that AUC preservation is not in general true for recalibration under the FJS assumption.

(7a) implies (1b) with $T(\eta) = T_\varrho(\eta) = \frac{\frac{p}{q}\, \eta}{\frac{p}{q}\, \eta + \frac{1}{\varrho}\, \frac{1-p}{1-q}\, (1-\eta)}$ strictly increasing in $\eta$. If $\varrho$ happens to take the value 1, at first glance we are back in the context of label shift as in Section 4.2 above. However, as (5) need not hold true under the FJS assumption, in such cases there cannot be label shift. The type of shift modelled instead is called ‘invariant density ratio’ shift (Tasche [23]), defined as the special case of FJS with $\varrho = 1$.

### 4.4 Recalibration under assumption of covariate shift with posterior drift (CSPD)

In Sections 4.1, 4.2 and 4.3, we identified the transformation $T$ of (1b) after we had described a recalibration method designed under assumption of one of three types of distribution shift, namely slightly modified covariate shift, label shift, and FJS. In contrast, in this section, we define $T$ and use it to characterise the type of distribution shift implied by the recalibration method.

According to Scott [19], the source distribution $P(X, Y)$ and the target distribution $Q(X, Y)$ are related through *covariate shift with posterior drift* (CSPD) if there is a *strictly increasing transformation* $T$ such that (1b) holds true.

As mentioned in Section 3.1, (1b) implies that $\eta_P(X)$ is sufficient for $X$ with respect to $Y$ under the target distribution $Q$. Since under CSPD the transformation $T$ is strictly increasing, (1b) also implies that $\eta_Q(X)$ and $\eta_P(X)$ are strongly comonotonic (Tasche [24]). As a consequence, Kendall's $\tau$ and Spearman's rank correlation both take the maximum value 1 when applied to $(\eta_Q(X), \eta_P(X))$. This suggests that CSPD is a strong assumption which might be less often true than one would hope for. Nonetheless, comonotonicity of score and posterior probability is a common assumption in the literature. Chen et al. [5] call this assumption the *rationality assumption*.

If labels are available in the test dataset, under the CSPD assumption the transformation $T$ of (1b) can be approximately determined by means of isotonic regression. However, the general assumption for this paper is that there are no label observations for the instances in the test dataset. Therefore, we are going to apply a moment matching approach instead (quasi moment matching, QMM).

To be able to do so, we consider *parametric CSPD* where the transformation $T$ of (1b) is specified as follows through some strictly increasing and continuous distribution function $F$ on the real line and parameters $a, b \in \mathbb{R}$:

$$T_{a,b}(u) \;=\; F\big(a\, F^{-1}(u) + b\big), \qquad \text{for } 0 < u < 1. \tag{8a}$$

This is not a radically new approach but rather a variation of what was called regression-based calibration by Ojeda et al. [14]. Here are two examples for natural choices of $F$ in (8a):

- Logistic distribution function (inverse logit): $F(x) = \frac{1}{1+\exp(-x)}$, $x \in \mathbb{R}$. The parametric CSPD approach with inverse logit is often called 'Platt scaling' in the literature. However, as pointed out by Ojeda et al. [14], 'Platt scaling' sometimes also refers to the specification of $T$ as

  $$T_{a,b}(u) \;=\; \frac{1}{1+\exp(-(a\,u+b))}, \qquad \text{for } 0 < u < 1. \tag{8b}$$

  In the following, we use the term *Platt scaling* to refer to (8b) and *logistic CSPD* to refer to (8a) with inverse logit.
- Standard normal distribution function (inverse probit): $F(x) = \Phi(x)$, $x \in \mathbb{R}$. We refer to this choice of $F$ as *normal CSPD*.

The idea for *quasi-moment matching (QMM)* with parametric CSPD is to determine the parameters $a, b \in \mathbb{R}$ by solving the following equation system:

$$q = E_Q\big[T_{a,b}(\eta_P(X))\big] \quad \text{and} \quad AUC_{\eta_P(X)} = AUC_{T_{a,b}(\eta_P(X))}, \tag{8c}$$

with $T_{a,b}$ as in (8a) or (8b) and AUC defined in Section 3.3. More precisely, in (8c), $AUC_{\eta_P(X)}$ is computed with respect to $P(X, Y)$ while $AUC_{T_{a,b}(\eta_P(X))}$ is computed as implied AUC with respect to $Q(X)$. See Section 3.3 for the different ways to compute AUC.

In (8c), AUC works like a second moment of the posterior probabilities. But in contrast to its effect on the second moment or the variance, the prior probability of the positive class has no effect on AUC. This makes the assumption of AUC invariance between source and target distributions more plausible (Tasche [22]).

*Remark 1.* One-parameter CSPD as in (8a) with $b = 0$ can be used for class distribution estimation (also called quantification), i.e. to determine an unknown class prior probability $q = Q[Y = 1]$ under the target distribution. In this case, instead of solving (8c) for two parameters $a$ and $b$, the following equation is solved for parameter $a$ only:

$$AUC_{\eta_P(X)} \;=\; AUC_{T_{a,0}(\eta_P(X))}. \tag{9a}$$

Then the mean of the resulting posterior probability $\eta_Q(X) = T_{a,0}(\eta_P(X))$ under the target distribution feature distribution $Q(X)$ is computed to obtain an estimate $\widehat{q}$ of $q$:

$$\widehat{q} \;=\; E_Q\bigl[T_{a,0}(\eta_P(X))\bigr]. \tag{9b}$$

One-parameter CSPD may be interpreted as a modification of quantification under the assumption of covariate shift. In contrast to assuming covariate shift when AUC can differ between source distribution and target distribution, with one-parameter CSPD by construction source AUC and target AUC are equal.

### 4.5 Quasi moment matching based on parametrised receiver operating characteristics

There is no guarantee that the QMM approach presented in Section 4.4 is feasible in the sense that there exists a solution to equation system (8c) or that the solution is unique. This reservation motivates the following alternative approach where instead of beginning with representation (8a) of the posterior probabilities, the starting point is a parametrised representation of the receiver operating characteristic (ROC) curve associated with the target distribution $Q(X, Y)$.

Tasche ([22], Section 5.2) modified an idea of van der Burgt [4] by assuming the ROC curve of a real-valued score $S$ to be

$$ROC_S(u) \;=\; \Phi\bigl(c + \Phi^{-1}(u)\bigr), \qquad u \in (0, 1), \tag{10a}$$

for some fixed parameter $c \in \mathbb{R}$, with $\Phi$ denoting the standard normal distribution function. The ROC curve of (10a) emerges when the class-conditional score distributions in a binary classification problem are both univariate normal distributions with equal variances. But ROC curves like in (10a) may also be incurred in circumstances where the class-conditional distributions are not normal (Proposition 5.3 of Tasche [22]).

Tasche [22] showed that (10a) implies the following representation for the posterior probability given the score $S$ under the target distribution $Q$:

$$Q[Y = 1\,|\,S] \;=\; \frac{1}{1 + \frac{1-q}{q}\,\exp\bigl(c^2/2 - c\,\Phi^{-1}(F_0(S))\bigr)}, \tag{10b}$$

where $F_0(s) = Q[S \le s\,|\,Y = 0]$ stands for the class-conditional distribution function of $S$ given $Y = 0$.

For $AUC_S$ as defined in Section 3.3, (10a) implies $AUC_S = \Phi\bigl(\frac{c}{\sqrt{2}}\bigr)$. Hence, if $AUC_S$ is known the parameter $c$ of (10b) is determined by

$$c \;=\; \sqrt{2}\,\Phi^{-1}(AUC_S). \tag{10c}$$

The score $S$ in (10b) can be chosen as $\eta_P(X)$ or any probabilistic classifier which approximates $\eta_P(X)$. The target prior probability $q$ is known by general assumption for this paper. Similarly to Section 4.4, for QMM to work one has to make the assumption that a prudent choice of $AUC_S$ under the target distribution (defined as implied AUC by (3b) and (3a) above) is informed by $AUC_S$ observed in the training dataset such that by (10c) also parameter $c$ is known.

However, the distribution function $F_0$ of the score $S$ conditional on $Y = 0$ appearing in (10b) is assumed not to be known under the target distribution since by general assumption for this paper, only the features but not the labels can be observed in the test dataset. Making use of (10b) therefore requires an iterative approach where in each step a refined estimate of the negative class-conditional score distribution function $F_0$ is calculated.

Denote by $f$ an unconditional density of $S$ under $Q$ and by $f_0$ a density of $F_0$. Then the posterior probability $Q[Y = 1 \,|\, S = s]$ can be represented as

$$Q[Y = 1 \,|\, S = s] \;=\; 1 - \frac{(1-q)\, f_0(s)}{f(s)} \tag{11a}$$

for $s$ in the range $\mathcal{S}$ of $S$. By (10b), (11a) implies

$$f_0(s) \;=\; \frac{f(s)}{1-q}\left(1 - \frac{1}{1 + \frac{1-q}{q}\,\exp\bigl(c^2/2 - c\,\Phi^{-1}(F_0(s))\bigr)}\right), \quad s \in \mathcal{S}. \tag{11b}$$

In the case of scores $S$ under a discrete or empirical distribution, as described in Appendices A.1 and A.2, (11b) can be treated as a fixed point equation for the probabilities $Q[S = s \,|\, Y = 0] = f_0(s)$ and be solved by a straightforward fixed-point iteration with intial values $Q[S = s]$, $s \in \mathcal{S}$. The numerical example of Section 5 below suggests that such an iterative approach converges as long as the prior probability $q$ is small and, hence, $f$ and $f_0$ are close to each other.

A further issue when making (10b) operational may occur when the class-conditional distribution function $F_0$ or any function approximating it takes the value 1. In particular, this will happen if $F_0$ is approximated by an empirical distribution function. Then the term $\Phi^{-1}\bigl(F_0(s)\bigr)$ is ill-defined. See Appendix A.2 for a workaround to deal with this issue.

Deploying a discrete distribution $F_0$ in (10b) as for instance an empirical approximation, makes it unlikely if not impossible to exactly match a pre-defined $AUC_S$ when using (10b). To control for the unavoidable deviation from the $AUC_S$ objective in this case we take recourse to the original QMM approach as proposed by Tasche [22]. Define

$$T^*_{a,b}(S) \;=\; \frac{1}{1 + \exp\bigl(b + a\,\Phi^{-1}(F_0(S))\bigr)}, \tag{12a}$$

with $F_0$ as in (10b) and parameters $a, b \in \mathbb{R}$ to be determined by quasi-moment matching (QMM) like in (8c):

$$q = E_Q\bigl[T^*_{a,b}(S)\bigr] \quad \text{and} \quad AUC_S = AUC_{T^*_{a,b}(S)}, \tag{12b}$$

To distinguish the two recalibration approaches presented in this section, we call the approach based on (10b) and (10c) *ROC-based QMM* and refer to the approach based on (12a) and (12b) as *2-parameter QMM*.

## 5 Example

The example presented in this section illustrates the impact of the recalibration methods and assumptions discussed in Section 4 on the following characteristics of the target distribution:

– The class 1 posterior probabilities.
– The mean of the class 1 posterior probabilities (which ought to equal the class 1 prior probability).
– The AUC implied by the class 1 posterior probabilities.
– The mean of the square root of the class 1 posterior probabilities as an expample involving a concave function of the posterior probabilities.

For the example we chose discrete source and target distributions of a feature (called score in the following) with values in an ordered set with 17 elements. These distributions may be interpreted as empirical distributions of samples with many ties or as the genuine distributions of discrete scores or ratings. For instance, the major credit rating agencies Standard & Poor's, Moody's and Fitch use rating scales with 17 to 19 different grades.

The source distribution is specified as follows: (1) The conditional feature distribution for class 0 is a binomial distribution with success probability 0.4, the conditional feature distribution for class 1 is a binomial distribution with success probability 0.55. The number of trials for both binomial distributions is 16 such that the support of the distribution is the set $\{0, 1, \ldots, 16\}$. (2) The class 1 prior probability is $p = 0.01$.

The target distribution is incompletely specified as follows: (1) The unconditional feature distribution is a binomial distribution with number of trials 16 whose success probability is a Vasicek-distributed random variable (Section 2.2 of Meyer [13]) with mean 0.3 and correlation parameter 0.3. (2) The target class 1 prior probability is $q = 0.05$.

Figure 1 shows[3] the source and target unconditional score distributions. They were intentionally chosen to be quite different such that any distribution shift assumed for the recalibration must be significant.

Figure 2 presents the source posterior probabilities and eight different sets of target posterior probabilities that have been calculated with the methods and assumptions discussed in Section 4:

– Capped scaling: Section 4.1
– Label shift: Section 4.2
– FJS: Factorizable joint shift, Section 4.3
– Platt scaling: Section 4.4, Eq. (8b)
– ROC QMM: ROC-based QMM, Section 4.5, Eq. (10b) and Eq. (10c)
– 2-param QMM: 2-parameter QMM, Section 4.5, Eq. (12a) and Eq. (12b)
– Logistic CSPD: Section 4.4, Eq. (8a) with $F(x) = \frac{1}{1+\exp(-x)}$
– Normal CSPD: Section 4.4, Eq. (8a) with $F(x) = \Phi(x)$, the standard normal distribution function

[3] Calculations were performed with R (R Core Team [17]). Details of some more involved calculations are described in Appendices A.1 and A.2. The R-scripts can be downloaded from https://www.researchgate.net/profile/Dirk-Tasche.

**Fig. 1.** Source and target score distributions for the example in Section 5.

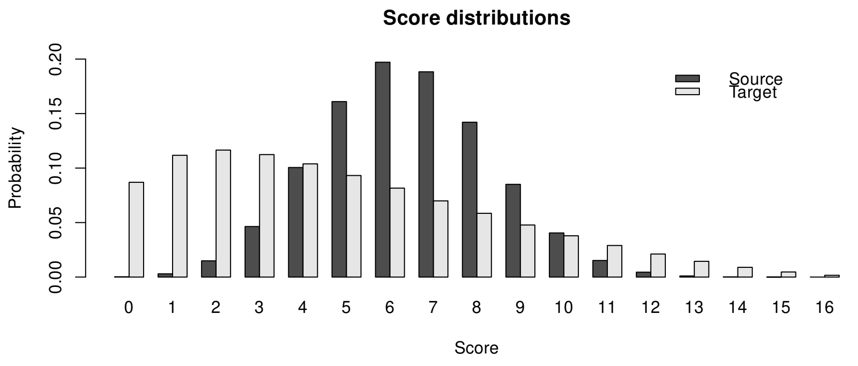

**Fig. 2.** Class 1 posterior probabilities for the example in Section 5. The scale of the vertical axis is logarithmic. The dots for the probabilities have been connected with straight lines for better readability. 'Source' refers to the posterior probabilities in the source distribution without recalibration.

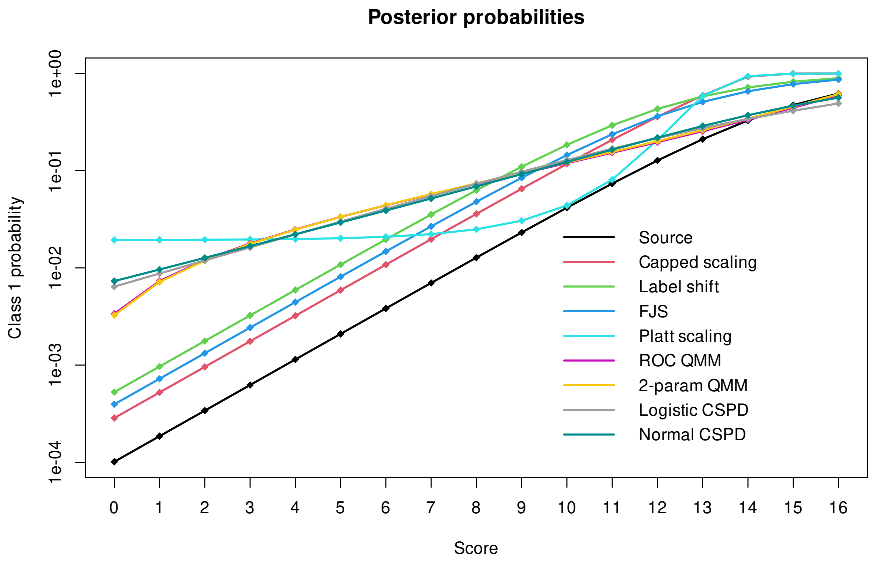

**Table 1.** Results for the recalibration methods presented in Section 4. The row 'Source' shows the values for the source distribution without recalibration. The numbers in all other rows refer to the target distribution. 'mean(probs)' is the class 1 prior probability calculated as mean of the recalibrated posterior probabilities. 'AUC' is the area under the ROC curve implied by the recalibrated posterior probabilities. 'mean(sqrt(probs))' is the mean of the square root of the recalibrated posterior probabilities.

| Method | mean(probs) | AUC | mean(sqrt(probs)) |
|---|---|---|---|
| Source | 0.010 | 0.802 | 0.084 |
| Capped scaling | 0.050 | 0.950 | 0.132 |
| Label shift | 0.060 | 0.930 | 0.160 |
| FJS | 0.050 | 0.932 | 0.142 |
| Platt scaling | 0.050 | 0.802 | 0.179 |
| ROC QMM | 0.049 | 0.799 | 0.191 |
| 2-param QMM | 0.050 | 0.802 | 0.191 |
| Logistic CSPD | 0.050 | 0.803 | 0.192 |
| Normal CSPD | 0.050 | 0.802 | 0.192 |

All target posterior probabilities are well above the source posterior probabilities. This does not come as a surprise given that the target class 1 prior probability of 5% is much higher than the source class 1 prior probability of 1%. Otherwise, there are two groups of target posterior probability curves: The curves based on capped scaling, label shift and FJS on the one hand, and the curves based on the five QMM methods described in Sections 4.4 and 4.5 on the other hand. The curves of the former group are rather steep compared to the curves of the latter group.

Table 1 displays three characteristics for the source distribution as well as for the eight different target distributions that result from the recalibration methods discussed in Section 4. Concluding from the entries in the column 'mean(probs)', only the recalibration method 'label shift' – introduced in Section 4.2 – is unreliable in so far as it does not achieve the required target class 1 prior probability $p = 0.05$. With all seven other recalibration methods the target prior probability is matched.

Column 'AUC' of Table 1 is more varied than column 'mean(probs)'. For the five QMM methods from Sections 4.4 and 4.5, AUC is essentially the same as the AUC of 0.802 under the source distribution. This is a consequence of the QMM design since one of the moment matching objectives is hitting the source AUC. Thanks to Proposition 1, one might expect also the 'label shift' AUC to match the source AUC. However, Proposition 1 is not applicable to the example of this section because by design the target feature distribution is not a mixture of the source class-conditional distributions. In any case, the high AUC values for methods 'capped scaling', 'label shift' and 'FJS' cause the greater slopes of their posterior probability curves in Figure 2 compared to the slopes of the QMM posterior probability curves.

As an example for the application of a concave function to the target posterior probabilities, we chose the function $C(u) = \sqrt{u}$, $u \in [0, 1]$, and the related expected value $E_Q\big[C(\eta_Q(X))\big]$ under the target distribution. Note that (2) implies $0.05 = q \le E_Q\big[\sqrt{\eta_Q(X)}\big] \le \sqrt{q} \approx 0.2236$.

It is clear from column 'mean(sqrt(probs))' of Table 1 that AUC is a driver of the mean of the concave function of the posterior probabilities. The lower the value of AUC, the higher is the mean of the concave function of the probabilities. Hence it makes sense to have AUC as a second objective to be matched, in addition to requiring that the target class 1 prior probability is reached. But even if both of these conditions are met there can still be variation in the mean of the concave function. This is demonstrated by the 'Platt scaling' posterior probabilities for which $E_Q\big[\sqrt{\eta_Q(X)}\big]$ takes a notedly lower value than for the other four QMM posterior probabilities with almost identical values.

## 6 Conclusions

Recalibration of binary probabilistic classifiers to a target prior probability is an important task in areas like credit risk management. This paper presents analyses and methods for recalibration from a distribution shift perspective. It has turned out that distribution shift assumptions linked to the performance in terms of area under the curve (AUC) of a probabilistic classifier under the source distribution are useful for the design of meaningful recalibration methods. Two new methods called parametric CSPD (covariate shift with posterior drift) and ROC-based QMM (quasi moment matching) have been proposed and have been tested together with some other methods in an example setting. The outcomes of this testing exercise suggest that the QMM methods discussed in the paper can provide appropriately conservative results in evaluations with concave functions like for instance risk weights functions for credit risk.

**Acknowledgments.** The author would like to thank two anonymous reviewers for their useful comments and suggestions.

## References

1. Allikivi, M.L., Järve, J., Kull, M.: Cautious Calibration in Binary Classification. Frontiers in Artificial Intelligence and Applications, vol. 392, pp. 1503–1510 (2024). https://doi.org/10.3233/FAIA240654
2. BCBS: CRE – Calculation of RWA for credit risk. Basel Committee on Banking Supervision, https://www.bis.org/basel_framework/standard/CRE.htm, Regulatory Standard
3. Bohn, J., Stein, R.: Active Credit Portfolio Management in Practice. John Wiley & Sons, Inc. (2009)
4. van der Burgt, M.: Calibrating low-default portfolios, using the cumulative accuracy profile. Journal of Risk Model Validation **1**(4), 17–33 (2008)
5. Chen, W., Sahiner, B., Samuelson, F., Pezeshk, A., Petrick, N.: Calibration of medical diagnostic classifier scores to the probability of disease. Statistical methods in medical research **27**(5) (2018). https://doi.org/10.1177/0962280216661371
6. Cramer, J.: Logit Models From Economics and Other Fields. Cambridge University Press (2003)

7. Devroye, L., Györfi, L., Lugosi, G.: A Probabilistic Theory of Pattern Recognition. Springer (1996)
8. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to Quantify. Springer Cham (2023). https://doi.org/10.1007/978-3-031-20467-8
9. Fawcett, T.: An introduction to ROC analysis. Pattern Recognit. Lett. **27**(8), 861–874 (2006). https://doi.org/10.1016/J.PATREC.2005.10.010
10. He, H., Yang, Y., Wang, H.: Domain Adaptation with Factorizable Joint Shift. Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning (2021). https://doi.org/10.48550/ARXIV.2203.02902
11. Lalley, S.: Concentration inequalities (2013), https://galton.uchicago.edu/~lalley/Courses/386/index.html, lecture notes
12. Lipton, Z., Wang, Y.X., Smola, A.: Detecting and Correcting for Label Shift with Black Box Predictors. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 3122–3130. PMLR (10–15 Jul 2018)
13. Meyer, C.: Estimation of intra-sector asset correlations. The Journal of Risk Model Validation **3**(3), 47–79 (2009)
14. Ojeda, F., Jansen, M., Thiéry, A., Blankenberg, S., Weimar, C., Schmid, M., Ziegler, A.: Calibrating machine learning approaches for probability estimation: A comprehensive comparison. Statistics in Medicine **42**(29), 5451–5478 (2023). https://doi.org/10.1002/sim.9921
15. Ptak-Chmielewska, A., Kopciuszewski, P.: New Definition of Default Recalibration of Credit Risk Models Using Bayesian Approach. Risks **10**(1) (2022), https://www.mdpi.com/2227-9091/10/1/16
16. Quadrianto, N., Smola, A., Caetano, T., Le, Q.: Estimating Labels from Label Proportions. Journal of Machine Learning Research **10**(82), 2349–2374 (2009), http://jmlr.org/papers/v10/quadrianto09a.html
17. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2024), https://www.R-project.org/
18. Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure. Neural Computation **14**(1), 21–41 (2002). https://doi.org/10.1162/089976602753284446
19. Scott, C.: A Generalized Neyman-Pearson Criterion for Optimal Domain Adaptation. In: Proceedings of Machine Learning Research, 30th International Conference on Algorithmic Learning Theory. vol. 98, pp. 1–24 (2019)
20. Silva Filho, T., Song, H., Perello-Nieto, M., Santos-Rodriguez, R., Kull, M., Flach, P.: Classifier calibration: a survey on how to assess and improve predicted class probabilities. Machine Learning **112**(9), 3211–3260 (2023). https://doi.org/10.1007/s10994-023-06336-7
21. Storkey, A.: When Training and Test Sets Are Different: Characterizing Learning Transfer. In: Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N. (eds.) Dataset Shift in Machine Learning, chap. 1, pp. 3–28. The MIT Press, Cambridge, Massachusetts (2009)
22. Tasche, D.: Estimating discriminatory power and PD curves when the number of defaults is small. Working paper (2009), http://arxiv.org/abs/0905.3928
23. Tasche, D.: Fisher Consistency for Prior Probability Shift. Journal of Machine Learning Research **18**(95), 1–32 (2017)
24. Tasche, D.: Calibrating sufficiently. Statistics **55**(6), 1356–1386 (2021). https://doi.org/10.1080/02331888.2021.2016767
25. Tasche, D.: Factorizable Joint Shift in Multinomial Classification. Machine Learning and Knowledge Extraction **4**(3), 779–802 (2022). https://doi.org/10.3390/make4030038

26. Tsyplakov, A.: Evaluation of Probabilistic Forecasts: Proper Scoring Rules and Moments (2013), https://mpra.ub.uni-muenchen.de/45186/
27. Vaicenavicius, J., Widmann, D., Andersson, C., Lindsten, F., Roll, J., Schön, T.: Evaluating model calibration in classification. In: The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. vol. 89, pp. 3459–3467. PMLR (2019)

## A Appendix

### A.1 The AUC of discrete auto-calibrated probabilitistic classifiers

How to calculate implied AUC for an auto-calibrated[4] probabilistic classifier $S$ under a joint distribution $\mu(S, Y)$ of $S$ and $Y \in \mathcal{Y} = \{0, 1\}$? We assume here that the distribution of $S$ under distribution $\mu$ is discrete, i.e. it is is given by pairs of score values $s_i$ and their probabilities $\mu[S = s_i] = \pi_i > 0$, $i = 1, \ldots, n$. Then it follows from the AUC-definition and properties in Section 3.3 that

$$AUC_S = \frac{\sum_{i=2}^{n} \pi_i \, s_i \left(\frac{1}{2}\, \pi_i (1 - s_i) + \sum_{k=1}^{i-1} \pi_k \,(1 - s_k)\right)}{\left(1 - \sum_{j=1}^{n} \pi_j \, s_j\right) \sum_{j=1}^{n} \pi_j \, s_j} \tag{13}$$

We use (13) to compute imlied AUC both under the source distribution for $S = \eta_P(X)$ with $\mu = P$ and under the target distribtion for $S = \eta_Q(X) = T\big(\eta_P(X)\big)$ with $\mu = Q$. Note that $\eta_P(X)$ under $P$ and $\eta_Q(X)$ under $Q$ are both auto-calibrated probabilistic classifiers (Proposition 1 of Vaicenavicius et al. [27]).

### A.2 Adapting discrete distributions for QMM

Consider for the real-valued score or probabilistic classifier $S$ with discrete distribution $\mu[S = s_i] = \pi_i$, $i = 1, \ldots, n$, its distribution function $G(s) = \mu[S \le s]$ for $s \in \mathbb{R}$. If the $s_i$ are increasingly ordered with $s_1 < \ldots < s_n$, then we obtain $G(s_i) = \sum_{j=1}^{i} \pi_j$, for $i = 1, \ldots, n$. In particular, it follows $G(s_n) = 1$ such that $\Phi^{-1}\big(G(s_n)\big) = \infty$ for the standard normal distribution function $\Phi$ and for the logistic distribution function.

For the purpose of this paper, to avoid this problem we apply the workaround proposed by van der Burgt [4]. He suggested replacing the distribution function $G$ with the mean of itself and its left-continuous version, i.e. with $G^*$ defined by $G^*(s) = \frac{1}{2}\big(\mu[S \le s] + \mu[S < s]\big)$, for $s \in \mathbb{R}$.

This implies for the $s_i$ which represent the support of $\mu$ that $G^*(s_i) = \left(\sum_{j=1}^{i} \pi_j\right) - \pi_i/2$, for $i = 1, \ldots, n$.

[4] The probabilistic classifier $S$ is *auto-calibrated* if for all $s$ in the range of $S$ it holds that $\mu[Y = 1 \,|\, S = s] = s$ (Tsyplakov [26], Section 2.2).

# QuaDapt: Drift-Resilient Quantification via Parameters Adaptation

João Pedro Ortega[1], Luiz Fernando Luth Junior[1], Willian Zalewski[1,2], and André Maletzke[1]

[1] Western Parana State University, Foz do Iguaçu, Brazil
Graduate Program on Electrical Engineering and Computing
{joao.ortega1,luiz.junior90,andre.maletzke}@unioeste.br
[2] Federal University for Latin American Integration, Foz do Iguaçu, Brazil
willian.zalewski@unila.edu.br

**Abstract.** Quantification is a supervised learning task focused on estimating the prevalence of classes in unlabeled test sets. Although existing quantification methods are effective under prior probability shift, they often fail under more general distribution shift scenarios. In this paper, we introduce `QuaDapt`, a generic and flexible framework that makes classifier-based quantifiers more resilient to distribution shifts. Rather than explicitly modeling specific types of concept drift, `QuaDapt` focuses on adapting the quantifier parameters based on the classifier score distribution observed in the test set. `QuaDapt` updates the quantifier's parameters without requiring labeled test data or retraining. We evaluate `QuaDapt` using artificially generated score distributions to simulate controlled variations in score complexity, as well as on real-world datasets where changes were induced to affect classifier outputs. In artificial experiments, standard quantifiers performed poorly, even as test score distributions became easier. In contrast, `QuaDapt` effectively reduced the quantification error in these scenarios. On 12 real-world datasets with concept drifts, `QuaDapt` consistently enhanced all quantifiers' performance, demonstrating its practical effectiveness and robustness.

**Keywords:** Dataset shift · Concept drift · Quantification.

## 1 Introduction

Real-world machine learning systems often face problems in non-stationary environments, where the data observed during deployment differs from the training data [9]. Machine learning methods rely on the assumption that training and test data are drawn from the same underlying distribution. This encompasses assumptions regarding the stability of class priors, class conditional distributions, and marginal distributions. However, these assumptions are often violated in practice, which can significantly degrade the performance and reliability of machine learning models when deployed in real-world environments.

Over the years, the machine learning community has extensively studied distribution shifts across various problem settings [23,26]. In particular, significant progress has been achieved in the context of classification, encompassing problem formalization as well as detection and adaptation methods for distribution shifts [12,27]. However, in less popular or newly formulated tasks, the understanding of how distribution shifts hinder the models remains poorly explored. One such task is quantification, a supervised learning task formalized by Forman (2005) [10] that aims to estimate the prevalence of each class in an unlabeled test set, rather than assigning labels to individual instances. A straightforward method involves classifying each instance and then counting the number of instances predicted for each class. This method, named Classify and Count (CC), suffers from the systemic error introduced when the class distribution drifts [5].

From Forman's seminal paper [10], a new machine learning community emerged, proposing a wide range of methods to tackle this task. Most of the existing approaches rely on a classifier in a preliminary step to produce either hard predictions or probabilistic scores on the test set. Then, different strategies are employed to infer the class distribution, including simple adjustments based on error rates, probabilistic modeling, and distribution matching [24,11,1,16,22,17].

Although there are contributions to make machine learning tasks more resilient against distribution shift scenarios, a gap remains in the literature regarding strategies that enhance the resilience of quantification under shift scenarios. The most recent contributions on this topic are the works of González et al. (2024) [15] and Maletzke et al. (2021) [20], which analyzed quantification methods under distribution shifts and proposed the first quantification method resilient to distribution shifts, respectively.

In this paper, we propose a generic framework that transforms any classifier-based quantifier into a drift-resilient version by updating its internal parameters using synthetic scores aligned with the estimated test set scores. Similarly to Maletzke et al. (2021), we do not assume any particular form of concept drift or shift. Instead, our framework updates the training-derived posterior probabilities by aligning them with the posterior probabilities observed at deployment through a mixture-model matching process. Based on this alignment, the framework re-estimates the quantifier parameters that depend on these probabilities, such as *tpr*, *fpr*, and the posterior probabilities themselves. Thereby, it adapts the quantifier to the current test conditions without requiring labeled data or retraining.

We evaluate the effectiveness of the proposal in both artificial and real-world scenarios. In the first, we systematically varied the complexity of classifier score distributions using synthetic data. This enabled us to isolate the effect of distribution shifts on quantifier performance. The results reveal a counterintuitive behavior of the current quantifiers, which performed poorly even when the test distributions became easier. In contrast, the `QuaDapt` quantifiers version demonstrated significantly improved robustness, effectively reducing quantification errors. In the second scenario, we show that the proposal consistently enhances the performance of all tested quantifiers. Notably, the `QuaDapt`-adapted quantifiers outperformed their standard counterparts in ten of the 12 datasets, including five datasets where they achieved improvements across all evaluated quantifiers.

The structure of this paper is as follows: Section 2 we present the paper background, including differences between classification and quantification tasks, types of distribution shifts, and a method to tackle quantification in shift scenarios. Section 3 presents the framework proposed for converting classifier-based quantifiers into resilient quantifiers in shift environments. Section 4 depicts the experimental setup and Section 5 presents the empirical results and discussion. Finally, Section 6 concludes this work and presents directions for future work.

## 2 Background

Quantification is a supervised learning task that aims to estimate the class distribution in an unlabeled test set. This task is closely related to the well-known classification task. For instance, both tasks operate over the same feature representation and assume a nominal output variable that defines the class label. Moreover, many quantification methods depend on a classifier in a preliminary step to produce either hard predictions, probabilistic scores, or error rates, which serve as the basis for estimating the class distribution in the test set [13].

Hence, defining the quantification task requires the classification definition. Let $h$ be a classifier induced from a train set $D = \{(x_1, y_1), ...(x_n, y_n)\}$, where $x_i \in \mathcal{X}$ is a $m$-dimensional vector in the feature space $\mathcal{X}$ with $m$ attributes and $y_i \in Y = \{c_1, ..., c_l\}$ the respective class label. Therefore, a classifier is defined as a model $h$ induced from $D$ such that [20]:

$$h : \mathcal{X} \to \{c_1, ...c_l\}$$

In binary tasks, the output space $Y$ comprises two classes, the *positive class* ($c_1 = \oplus$) and the *negative class* ($c_2 = \ominus$). A classifier $h$ aims to assign a class label to unseen instances by thresholding the values provided by a scoring function $h_s : \mathcal{X} \to \mathbb{R}$ that predicts a numerical value correlated to the posterior probability $P(y_i = \oplus \mid x_i)$. This value indicates the likelihood that $x_i$ belongs to the positive class [10,20].

On the other hand, quantification aims to estimate the prevalence of classes within a given test sample $S \in \mathbb{S}$, rather than assign labels to individual instances as in classification. Accordingly, a quantifier is defined as follows [20]:

$$q : \mathbb{S}^{\mathcal{X}} \to [0, 1]^l$$

where $\mathbb{S}^{\mathcal{X}}$ represents all the possible samples from $\mathcal{X}$. In a binary scenario, a quantifier estimates a vector $\hat{\mathbf{p}} = \{\hat{p}_\oplus, \hat{p}_\ominus\}$, where $\hat{p}_\oplus$ represents the posterior probability estimation for the positive class, such that $\hat{p}_\oplus + \hat{p}_\ominus = 1$ [20].

Classify and Count (CC) is one of the simplest quantification methods, estimating class prevalence by applying a classifier to the test set and counting the number of instances predicted for each class. However, despite its simplicity, CC suffers from flagrant shortcomings. The most notable is the systematic error introduced when the class distribution in the test set differs from that of the training data [14]. While CC can yield accurate estimates in the rare case where the false positive rate (*fpr*) equals the false negative rate (*fnr*), it generally performs poorly under prior shift, which is precisely the scenario where quantification is most relevant. In stationary settings, one could simply use the class proportions from the training set, making quantification unnecessary. Thus, CC's sensitivity to distribution shift severely limits its applicability in real-world quantification tasks.

Although quantification is an emergent task within the machine learning field, several methods have been proposed to overcome the known deficiencies of Classify and Count. In recent years, the community has introduced a variety of strategies aimed at improving quantification accuracy under distribution shift. A well-established taxonomy, proposed by González et al. (2017) [14], organizes these methods into three main categories based on how they leverage classifier outputs and adjust for distributional changes: (*i*) classify, count & correct, (*ii*) adaptation of classification algorithms, and (*iii*) distribution matching.

In this paper, we focus on quantification methods that rely on a classifier in previous steps, aiming to analyze their behavior under distribution shift scenarios and propose a framework to enhance their robustness. Therefore, we restrict our focus on methods from the group (*i*) and (*iii*) that are summarized in Table 1, along with their respective descriptions and references.

Table 1: Quantification taxonomy

| Group | Quantifier | Brief Description | Reference |
|---|---|---|---|
| I | CC | Classify & Count | [10,11] |
| | ACC | Adjusted Classify & Count | |
| | X | Classifier with decision threshold adjusted such that $fnr^1 = fpr^2$ | |
| | MAX | Classifier with decision threshold chosen such that $tpr^3 = fpr$ is maximum | |
| | T50 | Classifier with decision threshold adjusted such that $tpr = 50\%$ | |
| | MS | Median of Classify & Count computed along all decision threshold values | |
| III | HDy | Mixture model with Hellinger distance | [16] |
| | D$y$S | Framework for Mixture models | [22] |
| | SMM | Sample Mean Matching | [17] |

Although widely used and recognized in the quantification literature for their effectiveness in estimating class prevalence, most existing methods are explicitly designed to handle variations in the prior class distribution $P(Y)$. However, despite their widespread adoption in many studies [1,21,8,17,25,6,18], these methods often overlook other important forms of distribution shift, which are common in real-world applications and can substantially impair quantifier performance.

This oversight represents a significant limitation, as real-world data is seldom stationary and often undergoes complex, evolving shifts that can compromise the reliability of classifier-based quantifiers. González et al. (2024) [15] recently

[1] False Negative Rate (*fnr*): proportion of positives incorrectly predicted as negatives.
[2] False Positive Rate (*fpr*): proportion of negatives incorrectly predicted as positives.
[3] True Positive Rate (*tpr*): proportion of positives correctly predicted as positives.

highlighted these issues, emphasizing the need to better understand and address the vulnerabilities of quantification methods in the face of a broader range of distribution shifts. In the following section, we will present various types of distribution shifts that can severely impact the performance of current quantifiers.

### 2.1 Distribution Shifts

Supervised learning methods typically assume that training and test data are drawn from the same distribution. However, this assumption rarely holds in real-world scenarios, where changes in the underlying data are common. In the context of quantification, the primary focus has been on handling prior probability shift, where the class distribution $P(Y)$ changes between training and test sets. While many quantifiers have been designed under this assumption, they often neglect other types of shifts, such as changes in the feature distribution $P(X)$ or the class-conditional distribution $P(X|Y)$. These additional forms of shifts can severely degrade quantification accuracy, especially for methods that rely on classifier-derived information. As highlighted by González et al. (2024) [15], the performance of many state-of-the-art quantifiers deteriorates substantially under realistic drift scenarios, revealing a critical need for more resilient approaches.

According to González et al. (2024) [11], the following changes can affect quantification methods:

**Prior Probability Shift:** refers to changes in the class distribution between training and test data. Formally, this condition is characterized by $P^{\text{tr}}(Y) \neq P^{\text{ts}}(Y)$, where $P^{\text{tr}}(Y)$ and $P^{\text{ts}}(Y)$ denote the class distributions in the training and test sets, respectively.

**Covariate Shift:** refers to changes in the marginal distribution of features changes between training and test sets, formally $P^{\text{tr}}(X) \neq P^{\text{ts}}(X)$, while the posterior class probabilities remain stable, i.e., $P^{\text{tr}}(Y|X) = P^{\text{ts}}(Y|X)$. This implies that the predictive relationship between features and labels is preserved, even though the distribution of features itself varies.

**Concept Drift:** refers to changes in the relationship between features and classes, specifically $P(Y|X)$, which can also affect $P(X|Y)$. This drift can be particularly challenging to detect and manage, as it implies that the underlying patterns that link features to classes have evolved over time [15].

These data shifts can significantly impact the outcomes of quantification models, due to their effect on the scores assigned by a *scorer*. Maletzke et al. (2021) [20] analyze the impact of distributional changes on quantification by examining how such shifts affect the complexity of score distributions produced by binary classifiers. In their seminal work, they demonstrate that even when changes simplify the underlying decision boundary, effectively making the classification task easier, most quantification methods (with the exception of CC) still suffer from significant performance degradation. This counterintuitive behavior reveals a critical vulnerability and highlights the lack of robustness in existing quantifiers when faced with shifts that alter the complexity of score distribution.

In [20], the authors proposed D$y$Syn, a quantification method resilient to distribution shifts based on the quantifier D$y$S [22] and the method Model for Score Simulation (MoSS). In this paper, we extend the work of Maletzke et al. (2021) [20] by proposing a general framework that transforms any classifier-based quantifier into a distribution shift–resilient quantifier. MoSS and D$y$Syn are described in the next sections.

### 2.2 MoSS

Classification scores can be evaluated to detect mismatches between training and test data. In binary classification, if the classifier accurately models the concept of each class, then the scores generated by the classifier for each class will be far apart from each other. Maletzke et al. (2021) [20] noted in their paper that, in some circumstances, when a distribution shift occurs during the deployment phase, the predicted score distributions change, impacting the accuracy of quantifiers. They proposed modeling a new training score distribution based on the predicted score distribution from the deployment phase and then reapplying the quantifier. To model new training scores, Maletzke et al. (2021) [20] proposed the Model for Score Simulation (MoSS).

The MoSS model produces two synthetic distributions, simulating the scores of the positive and negative class labels, parameterizing the overlap between the scores of each class. Thus, beyond generating artificial scores, parameterization enables MoSS to produce scores that emulate different scenarios of complexity between the positive and negative classes.

MoSS relies on three parameters: the *merging* factor $\mathfrak{m}$, which ranges from 0 to 1, controlling the overlap degree between the positive and negative scores (0 representing the easiest scenario with well-separated scores, while 1 indicates the most challenging case with highly overlapping scores), the number of observations $n$, and $\alpha$ that defines the proportion of the positive class [20]. The MoSS model is presented in the following equation:

$$\text{MoSS}(n, \alpha, \mathfrak{m}) = \text{syn}(\oplus, \lfloor \alpha n \rfloor, \mathfrak{m}) \cup \text{syn}(\ominus, \lfloor (1-\alpha) n \rfloor, \mathfrak{m})$$

where,

$$\begin{aligned} \text{syn}(\oplus, n, \mathfrak{m}) &= \textstyle\bigcup_{i=1}^{n} \{X_i^{\mathfrak{m}}\}, \qquad && X_i \sim U(0,1) \\ \text{syn}(\ominus, n, \mathfrak{m}) &= \textstyle\bigcup_{i=1}^{n} \{1 - X_i^{\mathfrak{m}}\}, && X_i \sim U(0,1) \end{aligned}$$

A synthetic score in MoSS can be defined as a non-linear map of a uniformly distributed random variable ranging in $[0, 1]$, enabling flexible generation of different degrees of overlap between positive and negative scores. This synthetic generation process plays a central role in the D$y$Syn method, which we present in the following section.

### 2.3 D$y$Syn

In [20], the authors explored quantification to deal with distribution shifts by modifying methods based on the test set. The authors proposed the quantification algorithm named as Distribution $y$-Similarity (D$y$Syn) using Synthetic Scores. D$y$Syn combines the D$y$S framework and the MoSS model.

D$y$S aims to find the optimal mixture of positive and negative scores ($S^{\text{tr}}_{\oplus}$ and $S^{\text{tr}}_{\ominus}$, respectively), estimated from the training set, by searching for the $\alpha$ (proportion of the positive class) value that minimizes the distance function DS between the resulting mixed distribution and the test set distribution $S^{\odot}$. The D$y$S is defined as follows:

$$\text{D}y\text{S}(S^{\text{tr}}_{\oplus}, S^{\text{tr}}_{\ominus}, S^{\odot}) = \mathop{\mathbf{arg\,min}}_{0 \leq \alpha \leq 1} \left\{ \text{DS}\left(\alpha R[S^{\text{tr}}_{\oplus}] + (1-\alpha) R[S^{\text{tr}}_{\ominus}], R[S^{\odot}]\right) \right\}$$

where $S^{\odot}$, $R$, and DS are the scores of the test set, a representation function, and a distance function to operate over $R$ representation, respectively.

In summary, D$y$Syn runs an additional search across multiple values of the MoSS merging factor($\mathfrak{m}$), selecting the one that generates the mixed score distributions most similar to the test scores according to a distance function. This adaptive mechanism enables D$y$Syn to align with the underlying score distribution of the test data, making it more robust against distribution shifts. D$y$Syn is formally defined as follows:

$$\text{D}y\text{Syn}(S^{\odot}) = \mathop{\mathbf{arg\,min}}_{\substack{0 \leq \alpha \leq 1 \\ \text{s.t. } 0 \leq \mathfrak{m} \leq 1}} \left\{ \text{DS}\left(R[\text{MoSS}(n, \alpha, \mathfrak{m})], R[S^{\odot}]\right) \right\}$$

## 2.4 Related Works

Quantification methods are designed to address *prior probability shift*, which refers to changes in the class distribution $P(Y)$ between training and test data. However, *other types of distribution shift*, such as changes in the feature distribution $P(X)$ or in the class-conditional distribution $P(X|Y)$, can also affect machine learning models, including quantifiers. While these types of shifts have been extensively studied in the *classification* literature, they are overlooked in the context of quantification.

This gap remains poorly explored, with only a few recent efforts addressing the effects of broader distribution shifts. Maletzke et al. (2021) [20] discussed in their research the flaws of classic quantification methods. To address the limitations of standard quantifiers, Maletzke et al. (2021) [20] proposed an innovative method referred to as D$y$Syn, which modifies D$y$S utilizing MoSS to identify an optimal fit for the score distribution based on a new dataset from a test set, rather than relying on training scores. D$y$Syn was evaluated across 12 real-world datasets, outperforming 15 state-of-the-art quantifiers.

González et al. (2024) [15] assessed the performance of different quantification methods under three types of drift: (1) prior probability shift, (2) covariate shift, and (3) concept shift. The authors demonstrate that while literature quantifiers are resilient to prior probability shifts, they perform poorly under other types of drift, and this issue remains underexplored.

## 3 Proposal

In this paper, we extend the D*y*Syn method, showing that it is an instance of a more general and non-formalized framework, enabling any classifier-based quantifier to adapt its behavior without requiring access to labeled test data or retraining, thus ensuring robustness in real-world non-stationary environments.

To formalize the framework, consider $q$ a binary quantifier that predicts the class prevalence $\hat{p}$ in the test set as follows:

$$\hat{p} = q(h_S(S^{\odot}), \Theta)$$

where $h_S$, $S^{\odot}$, and $\Theta$ represent the scorer, the unlabeled test set, and the quantifier parameters, respectively. Quantifier parameters $\Theta$ vary according to the quantification method, ranging from classifier performance metrics to class-conditional scores in distribution matching methods. For the sake of clarity, we categorize the quantifier parameters based on the taxonomy of quantification, as follows:

– For **classify, count & correct** methods (e.g., ACC and MS), include classifier performance metrics such as true positive rate (*tpr* ) and false positive rate (*fpr*), which are typically derived from the training set using a decision threshold $\tau$.
– For **distribution matching** methods (e.g., D*y*S and HDy), $\Theta$ consists primarily of class-conditional scores. Additionally, score representations (e.g., histograms or kernel functions) and their respective hyperparameters (e.g., the number of bins and kernel type), along with the similarity function, are also parameters in this category.

In both groups, the **class-conditional scores** represent a fundamental parameter, i.e., their degree of separation determines the difficulty of the quantification problem. Well-separated class-conditional scores correspond to an easier scenario, which typically translates into higher *tpr* and lower *fpr* for correction-based methods, as well as into clearer and more discriminative score representations for distribution-matching methods.

As noted by Maletzke et al. (2020) [21] and Maletzke et al. (2018) [19], concept drift is often manifested as changes in the complexity of class-conditional scores, since shifts in the data distribution directly affect how the classifier separates positive and negative instances. For this reason, in this paper, we consider the class-conditional scores as the primary parameters for both groups of quantifiers, while keeping the remaining parameters, such as threshold, similarity functions, or representation formats, fixed.

For a given training set $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, the positive and negative class-conditional scores can be obtained through $k$-fold cross-validation. Therefore, a generic function to obtain the remaining quantifier parameters can be defined as follows:

$$\Theta = \Psi(S_{\oplus}^{\text{tr}},\ S_{\ominus}^{\text{tr}},\ \tau)$$

where $\tau$ represents the classification threshold used on the class-conditional scores for calculating performance measures, true positive rate (*tpr*), and the false positive rate (*fpr*). For example, given the training score distributions $S^{\text{tr}}_{\oplus}$ and $S^{\text{tr}}_{\ominus}$, different choices of $\tau$ (e.g., decision thresholds) will produce distinct values for *tpr* and *fpr*, impacting on the quantifiers accuracy.

In contrast to relying on fixed training-derived parameters, Maletzke et al. (2021) [20] noted that modifying the training scores of the D*y*S method based on the test set leads to more resilient results under drift scenarios. Based on this assumption, we propose **Drift-Resilient Quantification via Parameters Adaptation** (`QuaDapt`), a generic framework for quantifying in dynamic environments. Our framework, `QuaDapt`, concentrates on updating the **class-conditional scores** parameter, which represents the core parameter across both quantification methods groups. By realigning these scores with the characteristics of the test set, `QuaDapt` effectively influences both **classify, count & correct** methods, where performance measures such as *tpr* and *fpr* depend on score distributions, and **distribution matching** methods, where prevalence is inferred by comparing score distributions directly. Therefore, adapting class-conditional scores serves as a general mechanism that enhances the robustness of quantifiers across both categories under drift scenarios.

Figure 1 provides a conceptual overview of how the `QuaDapt` framework operates. Starting from a labeled training set, a scorer $h_S$ is fitted to produce posterior scores. During deployment, when an unlabeled test set arrives, the scorer outputs the test scores $S^{\odot}$. Instead of relying solely on the training-derived scores, `QuaDapt` generates multiple sets of synthetic scores using MoSS, varying the mixing factor $\mathfrak{m}$ to represent different levels of class separability. The framework then searches for the synthetic distribution that minimizes the dissimilarity (DS) with the observed test scores $S^{\odot}$. The selected synthetic scores $\hat{S}_{\oplus}$ and $\hat{S}_{\ominus}$ are finally used to re-estimate the quantifier parameters $\hat{\Theta}$, which include correction factors such as *tpr*, *fpr*, and D*y*S's histograms. By realigning parameters in this way, `QuaDapt` adapts the quantifier to the actual test conditions, ensuring robustness under distribution shifts.

`QuaDapt` is formally defined as follows:

$$\hat{p} = \mathcal{Q}_{\texttt{QuaDapt}}\Big(q[h_S(S^{\odot}), \hat{\Theta}]\Big)$$

where $q$ is a quantifier that requires a classifier or scorer as a previous step, and $\Theta$ represents the $q$ parameters updated by the same $\Psi$ procedure, as follows:

$$\hat{\Theta} = \Psi(\hat{S}_{\oplus},\ \hat{S}_{\ominus},\ \tau)$$

where:

- $\hat{S}_{\oplus}, \hat{S}_{\ominus}$: synthetic scores for the positive and negative classes, respectively, generated using MoSS with a mixing factor $\mathfrak{m}$. The mixing factor is selected minimizing the dissimilarity (DS) between the combined synthetic distribu-

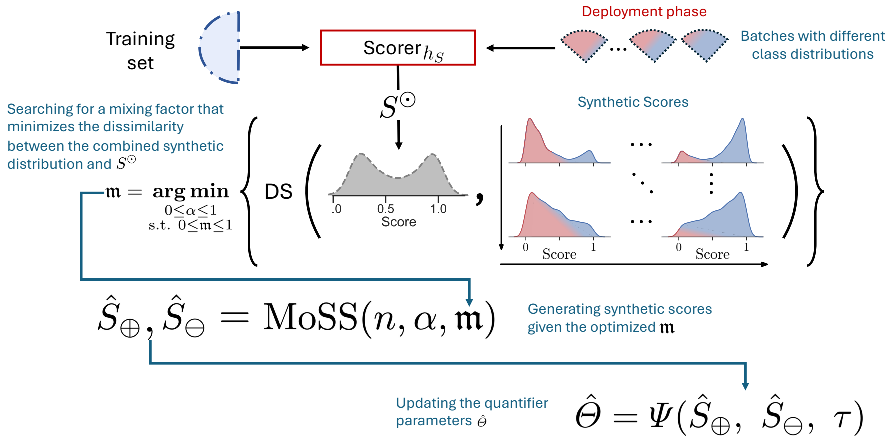
Training set
Scorer $h_S$
Deployment phase
Batches with different class distributions
$S^{\odot}$
Synthetic Scores
Searching for a mixing factor that minimizes the dissimilarity between the combined synthetic distribution and $S^{\odot}$
$\mathfrak{m} = \underset{\substack{0\leq\alpha\leq 1\\ \text{s.t. } 0\leq\mathfrak{m}\leq 1}}{\arg\min}$
DS
Score
$\hat{S}_{\oplus}, \hat{S}_{\ominus} = \text{MoSS}(n, \alpha, \mathfrak{m})$
Generating synthetic scores given the optimized $\mathfrak{m}$
Updating the quantifier parameters $\hat{\Theta}$
$\hat{\Theta} = \Psi(\hat{S}_{\oplus}, \hat{S}_{\ominus}, \tau)$

Fig. 1: Schematic representation of `QuaDapt`. Parts of this figure are adapted from [20].

tion and the distribution of test scores $S^{\odot}$, defined as follows:

$$\mathfrak{m} = \underset{\substack{0\leq\alpha\leq 1\\ \text{s.t. } 0\leq\mathfrak{m}\leq 1}}{\arg\min} \{\text{DS}\left(R[\text{MoSS}(n, \alpha, \mathfrak{m})], R[S^{\odot}]\right)\}$$

where $R$ and $n$ are the representation function (e.g., histogram or Cumulative Distribution Function) and number of synthetic scores generated by MoSS;

- $\tau$: same decision threshold used in training-based estimation, required to compute classifier metrics such as *tpr* and *fpr*.

## 4 Experiments

To evaluate our framework, we adopt the experimental setup introduced by Maletzke et al. (2021) [20]. Our evaluation is organized into two experiments, each designed to analyze the behavior of quantifiers under different conditions of distribution shift, reflected in the complexity of classifier scores.

In the **first experiment**, we leverage artificial score generation, using the MoSS model, to produce test scores with varying levels of complexity between training and test phases. A key advantage of using MoSS is that it allows us to abstract away from the specific source of data drift (e.g., covariate shift or concept drift). This setup enables a controlled evaluation of how well quantifiers can adapt to changes in score complexity, even in cases where the test-time classification task becomes easier. Thus, when the decision boundary becomes simpler, the expected result is improved quantifier performance, as observed in the classification task.

In the **second experiment**, we extend the analysis to a collection of real-world datasets. We explicitly enforce concept drift by modifying the test data so that the classification task becomes either simpler or more challenging relative to the training data. The goal is to assess how the quantification methods perform in realistic scenarios.

In both experiments, we evaluated the quantifiers listed in Table 1, ensuring consistency across experimental conditions. The `QuaDapt` framework adjusts classifier-based quantifiers by aligning artificial positive and negative score distributions with those observed in the test set, using a MoSS merging factor $\mathfrak{m}$ varied from 10% to 90% in 20% increments to generate diverse training score distributions. The performance of each method was assessed using the Mean Absolute Error (MAE), which quantifies the average absolute difference between the predicted and true prevalence of the positive class. Additionally, for the D$y$S method, the Topsøe distance was used as the divergence measure to guide its internal optimization process.

## 4.1 Experiment 1 - Artificial Scores

Our first experiment was designed to evaluate the existing quantification methods and the proposed framework under drift scenarios, without requiring manipulation of real datasets to induce distribution shifts.

Using MoSS, we generate synthetic training and test scores with varying levels of complexity, adjusting the merging factor $\mathfrak{m}$. Thus, we systematically control the separation between positive and negative score distributions, enabling the simulation of both easier and harder decision boundaries. This setup allows us to evaluate quantification performance across a range of drift scenarios where the score distributions differ between training and testing at controlled intensities, without being tied to a specific type of real-world shift.

In the training phase, we generated a total of 2,000 synthetic scores, 1,000 for the positive class ($\oplus$) and 1,000 for the negative class ($\ominus$), varying the MoSS merging factor for training ($\mathfrak{m}_{\text{tr}}$) from 5% to 95% in increments of 5%. For each training configuration defined by $\mathfrak{m}_{\text{tr}}$, we then evaluated multiple test scenarios by varying the following parameters in the testing phase:

- **Test size**: fixed at 100 instances, i.e., 100 score values per test set;
- **Class proportion** : from 0% to 100% in increments of 1%;
- **MoSS merging factor for test** ($\mathfrak{m}_{\text{ts}}$): varied from 5% to 95% in increments of 5%, simulating of test score distributions with different complexities;
- **Replicates**: each configuration was randomly repeated 10 times.

## 4.2 Experiment 2 - Real-world datasets

Our second experiment was designed to assess the performance of existing quantification methods and the proposed `QuaDapt` framework on real-world datasets. We adopt the experimental setup originally described in [20] to evaluate the standard quantifiers and quantifiers integrated into the `QuaDapt` framework.

Table 2 presents a brief description of the datasets. The datasets were obtained from UCI [4] and OpenML [28] repositories[3]. As performed by Maletzke

[3] Specific citations are requested for Avila [3] and Walking [2]. Jock A. Blackard and Colorado State University preserve copyright over Covertype.

et al. (2021) [20], we use multi-class datasets that are converted into binary problems by grouping the classes into positive and negative classes. The selection of class labels for the positive and negative classes was conducted strategically to create a subset of positive classes and two distinct subsets of negative classes, referred to as **easier** and **harder**. This setup was designed to simulate scenarios with varying complexity, thereby enabling the evaluation of quantification methods under different degrees of concept drift.

Table 2: Datasets description [20].

| Dataset | Classes | Positive | Negative Class | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Easier Subclass | | | Harder Subclass | | |
| | | | Classes Name | Overlap (%) | F1 Score | Classes Name | Overlap (%) | F1 Score |
| Avila | 12 | [A] | [I] | 0.43 | 0.99 | [E,F,G,H,X] | 6.83 | 0.86 |
| Chess game | 18 | [fourteen] | [zero,...,nine] | 3.05 | 0.98 | [thirteen] | 21.33 | 0.82 |
| Covertype | 7 | [Ponderosa_Pine] | [Krummholz] | 5.06 | 0.94 | [Doug.,Aspen,Cott.] | 23.69 | 0.81 |
| Dermatology | 6 | [6] | [3] | 0.00 | 1.00 | [1,4,5] | 0.30 | 0.97 |
| HAR | 6 | [5] | [6] | 0.00 | 1.00 | [4] | 7.45 | 0.93 |
| Land-use | 8 | [Corn] | [Hay] | 0.16 | 1.00 | [Soybeans] | 20.76 | 0.74 |
| MFeat | 10 | [3,5] | [2,9,10] | 2.54 | 0.99 | [4,7,8] | 19.31 | 0.81 |
| Mosquitoes | 3 | [*Ae.aegypti* ♀] | [*An.aquasalis* ♂] | 2.31 | 0.98 | [*Cx.quinquefasciatus* ♀] | 27.44 | 0.76 |
| Nursery | 5 | [priority] | [not_recom] | 0.00 | 1.00 | [very_recom,spec_prior] | 9.15 | 0.92 |
| Phishing URL | 5 | [Defacement] | [benign] | 1.38 | 0.99 | [malware,phishing] | 4.35 | 0.98 |
| Satimage | 6 | [4] | [2] | 0.46 | 0.99 | [3,5,7] | 19.74 | 0.80 |
| Walking | 22 | [6] | [10] | 1.22 | 0.99 | [9,12] | 43.39 | 0.79 |

To build the positive and negative classes (**easier** and **harder**), each dataset is split into training and test sets. A Random Forest classifier is then fit on the training set, and the predicted scores are obtained for the test set. We then computed the overlapped area between the positive and negative score distributions. A smaller overlap area indicates a simpler scenario, while a larger overlap reflects greater ambiguity between classes and thus a more complex setup. This procedure was systematically repeated to identify the easier and harder negative class groupings for each dataset. In addition, we computed the F1-Score to illustrate the difference between easier and harder scenarios, providing a complementary view of difficulty beyond overlap measurements.

Columns three, four, and seven in Table 2 represent the subclasses that compose the positive and negative classes, including the **easier** and **harder** negative subclasses for each dataset. Columns five, six, eight, and nine illustrate the intersection of the **easier** and **harder** negative subclasses with the positive class, respectively.

Therefore, after defining positive and negative classes, along with **easier** and **harder** subclasses, we sample each dataset according to the same distribution for every class. Then we split each dataset into two halves: training and test sets. We perform stratified sampling to preserve the distribution of the positive and negative subclasses. In the training portion, we apply 10-fold cross-validation to generate the score distributions required by the mixture models and to estimate the *tpr* and *fpr* rates employed by the quantifiers from the Classify, Count, and Correct group. We generate all classifier scores using Random Forests with 200 trees. In addition, we train a single Random Forest model, using the entire training set, to produce the test scores used in the evaluation phase.

To simulate realistic drift scenarios in our real-world experiment, we adhered to the Artificial Prevalence Protocol (APP) [7,24] by extracting multiple samples from the test set and systematically varying the prevalence of both positive and negative subclasses. We explored different combinations of **easier** and **harder** negative subclasses to create test distributions with varying levels of complexity. The configuration space for this experimental setup is detailed below:

– **Positive Class Proportion**: varied from 0% to 100% in increments of 1%;
– **Harder Negative Subclass Proportion**: from 0% to 100% in increments of 25%;
– **Test Set Size**: fixed at 100 instances per test sample;
– **Replicates**: each configuration was randomly repeated 10 times.

The quantifiers were assessed by conducting a statistical comparison between the original methods and their `QuaDapt` version. We utilized a paired two-tailed t-test to determine whether the observed differences in performance, measured by MAE, were statistically significant for each dataset.

## 5 Results

We open this section presenting the results of our **first experiment**, designed to assess quantifiers performance under controlled variations in score complexity. Although both $\mathfrak{m}_{\text{tr}}$ and $\mathfrak{m}_{\text{ts}}$ were independently varied across a wide range, we present plots of MAE as a function of $\mathfrak{m}_{\text{ts}}$ for only four selected fixed values of $\mathfrak{m}_{\text{tr}}$. This allows us to illustrate representative patterns while maintaining clarity in visual analysis. These results are presented in Figure 2.

Solid red lines indicate the performance of the standard quantifiers, while dashed green lines illustrate the results of their corresponding `QuaDapt`-adapted versions. The solid blue line represents the baseline CC method. The vertical dashed lines mark the point where $\mathfrak{m}_{\text{tr}} = \mathfrak{m}_{\text{ts}}$.

To the right of the vertical line ($\mathfrak{m}_{\text{ts}} > \mathfrak{m}_{\text{tr}}$), the problem becomes increasingly difficult, as the overlap between class score distributions grows. In these scenarios, MAE tends to rise for both standard and adapted methods, which is expected. However, to the left of the vertical line ($\mathfrak{m}_{\text{ts}} < \mathfrak{m}_{\text{tr}}$), the test score distributions become more separable, making the classification problem easier. In such cases, *standard quantifiers surprisingly continue to exhibit poor performance*, failing to exploit the increased separability.

This undesirable behavior of standard quantifiers under easier test conditions was also observed by Maletzke et al.(2021) [20]. However, the results in Figure 2 show that our proposal transforms classifier-based quantifiers into more resilient quantifiers when faced with these favorable changes. This represents a significant improvement, as the inability to take advantage of easier scenarios represents a serious limitation in practical applications of quantification.

Our **second experiment** aims to evaluate the effectiveness of our framework under realistic conditions. We adopted the experimental protocol described in [20]. Table 3 reports the MAE for each quantifier across 12 datasets. For each

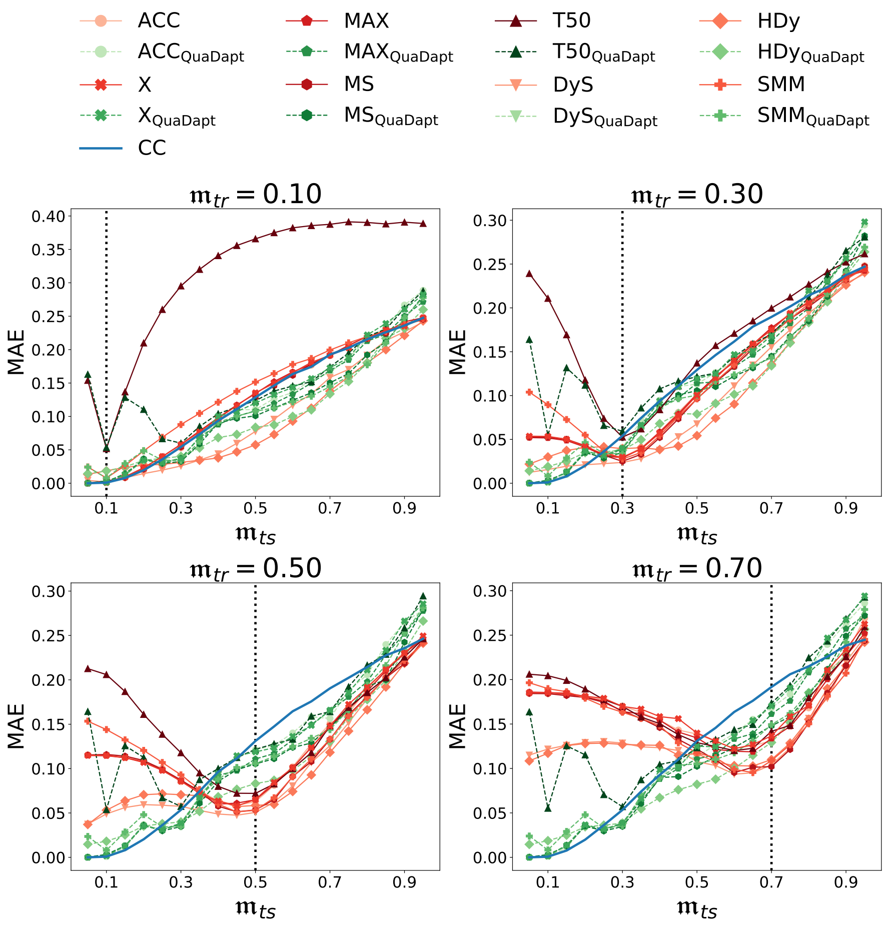

Fig. 2: MAE for fixed values of $\mathfrak{m}_{\mathrm{tr}} \in \{0.10, 0.30, 0.50, 0.70\}$, as $\mathfrak{m}_{\mathrm{ts}}$ varies. Vertical dashed lines indicate the point where $\mathfrak{m}_{\mathrm{tr}} = \mathfrak{m}_{\mathrm{ts}}$.

quantifier, we present results in pairs: the first line corresponds to the original (unadapted) quantifier, and the second line to its `QuaDapt` version. The best MAE values for each dataset are shown in **bold**, with the corresponding standard deviation reported in parentheses. Results that are statistically better ($p < 0.05$) are underlined.

The row **`QuaDapt` win rate** shows, for each dataset, the percentage and absolute number of cases where the `QuaDapt` quantifiers outperformed their standard counterparts in terms of MAE.

Notably, in five out of the twelve datasets, our proposal improved the performance of all evaluated quantifiers (win rate = 100%), demonstrating consistent effectiveness across diverse scenarios. Furthermore, in ten out of twelve datasets, the `QuaDapt` framework enhanced the performance of at least half of the evalu-

Table 3: Mean absolute error of standard quantifiers and their corresponding `QuaDapt` versions across all datasets.

| **Quantifiers** | **Avila** | **Chess game** | **Covertype** | **Dermatology** | **HAR** | **Land-use** |
|---|---|---|---|---|---|---|
| Baseline (CC) | 0.056 (0.039) | 0.045 (0.033) | 0.077 (0.053) | 0.002 (0.005) | 0.017 (0.015) | 0.061 (0.041) |
| ACC | 0.065 (0.044) | 0.052 (0.037) | 0.059 (0.045) | 0.002 (0.005) | 0.026 (0.020) | 0.084 (0.059) |
| ACC$_{\texttt{QuaDapt}}$ | **0.046 (0.054)** | **0.031 (0.026)** | **0.057 (0.055)** | 0.002 (0.005) | **0.017 (0.015)** | **0.048 (0.037)** |
| X | 0.060 (0.041) | 0.046 (0.033) | 0.070 (0.054) | 0.003 (0.006) | 0.020 (0.014) | 0.071 (0.046) |
| X$_{\texttt{QuaDapt}}$ | **0.046 (0.054)** | **0.030 (0.027)** | **0.056 (0.054)** | **0.002 (0.005)** | **0.017 (0.015)** | **0.048 (0.037)** |
| MAX | 0.070 (0.061) | 0.046 (0.038) | 0.071 (0.057) | 0.002 (0.005) | 0.020 (0.014) | 0.085 (0.075) |
| MAX$_{\texttt{QuaDapt}}$ | **0.050 (0.061)** | **0.034 (0.030)** | **0.062 (0.064)** | 0.002 (0.005) | **0.017 (0.015)** | **0.052 (0.044)** |
| T50 | 0.153 (0.110) | 0.127 (0.088) | **0.073 (0.059)** | **0.089 (0.067)** | 0.164 (0.102) | 0.160 (0.110) |
| T50$_{\texttt{QuaDapt}}$ | **0.094 (0.092)** | **0.088 (0.086)** | 0.121 (0.104) | 0.211 (0.127) | **0.144 (0.103)** | **0.091 (0.085)** |
| MS | 0.156 (0.118) | 0.109 (0.071) | **0.085 (0.063)** | **0.022 (0.019)** | 0.119 (0.088) | 0.143 (0.097) |
| MS$_{\texttt{QuaDapt}}$ | **0.074 (0.060)** | **0.060 (0.049)** | 0.096 (0.070) | 0.120 (0.106) | **0.073 (0.044)** | **0.063 (0.052)** |
| HDy | 0.079 (0.055) | 0.048 (0.036) | 0.072 (0.057) | **0.008 (0.008)** | 0.038 (0.030) | 0.102 (0.064) |
| HDy$_{\texttt{QuaDapt}}$ | **0.049 (0.047)** | **0.039 (0.031)** | **0.059 (0.050)** | 0.050 (0.044) | **0.026 (0.021)** | **0.038 (0.033)** |
| D$y$S | 0.065 (0.047) | 0.035 (0.029) | 0.062 (0.050) | **0.002 (0.005)** | 0.016 (0.013) | 0.065 (0.051) |
| D$y$S$_{\texttt{QuaDapt}}$ | **0.044 (0.051)** | **0.030 (0.027)** | **0.060 (0.054)** | 0.010 (0.010) | **0.013 (0.011)** | **0.038 (0.033)** |
| SMM | 0.102 (0.063) | 0.070 (0.044) | **0.079 (0.058)** | **0.006 (0.006)** | 0.053 (0.035) | 0.114 (0.070) |
| SMM$_{\texttt{QuaDapt}}$ | **0.090 (0.068)** | **0.065 (0.048)** | 0.091 (0.064) | 0.036 (0.023) | **0.048 (0.025)** | **0.065 (0.053)** |
| `QuaDapt` **win rate** | 100% (8/8) | 100% (8/8) | 63% (5/8) | 13% (1/8) | 100% (8/8) | 100% (8/8) |

| **Quantifiers** | **Mfeat** | **Mosquitoes** | **Nursery** | **Phishing URL** | **Satimage** | **Walking** |
|---|---|---|---|---|---|---|
| Baseline (CC) | 0.052 (0.036) | 0.062 (0.043) | 0.022 (0.018) | 0.018 (0.015) | 0.034 (0.023) | 0.098 (0.066) |
| ACC | 0.058 (0.039) | 0.060 (0.042) | 0.027 (0.020) | 0.026 (0.018) | 0.090 (0.064) | **0.062 (0.049)** |
| ACC$_{\texttt{QuaDapt}}$ | **0.042 (0.032)** | **0.045 (0.033)** | 0.027 (0.025) | **0.018 (0.015)** | **0.034 (0.023)** | 0.073 (0.056) |
| X | 0.058 (0.038) | 0.055 (0.039) | **0.026 (0.018)** | 0.019 (0.014) | 0.058 (0.036) | **0.070 (0.059)** |
| X$_{\texttt{QuaDapt}}$ | **0.042 (0.032)** | **0.045 (0.033)** | 0.027 (0.025) | **0.018 (0.015)** | **0.033 (0.023)** | 0.072 (0.056) |
| MAX | 0.048 (0.036) | 0.054 (0.042) | **0.025 (0.018)** | 0.018 (0.014) | 0.051 (0.035) | 0.078 (0.067) |
| MAX$_{\texttt{QuaDapt}}$ | **0.044 (0.033)** | **0.048 (0.036)** | 0.028 (0.027) | 0.018 (0.015) | **0.034 (0.023)** | **0.077 (0.065)** |
| T50 | 0.075 (0.056) | 0.111 (0.081) | **0.120 (0.082)** | **0.049 (0.042)** | 0.167 (0.115) | 0.066 (0.055) |
| T50$_{\texttt{QuaDapt}}$ | **0.069 (0.055)** | **0.106 (0.086)** | 0.161 (0.136) | 0.103 (0.074) | **0.053 (0.046)** | **0.063 (0.058)** |
| MS | 0.064 (0.043) | 0.097 (0.065) | 0.084 (0.054) | 0.036 (0.027) | 0.144 (0.100) | 0.074 (0.062) |
| MS$_{\texttt{QuaDapt}}$ | 0.064 (0.045) | **0.071 (0.055)** | **0.074 (0.058)** | **0.020 (0.017)** | **0.054 (0.038)** | **0.070 (0.052)** |
| HDy | 0.039 (0.031) | 0.055 (0.041) | **0.031 (0.024)** | **0.017 (0.014)** | 0.064 (0.043) | 0.073 (0.062) |
| HDy$_{\texttt{QuaDapt}}$ | **0.035 (0.029)** | **0.042 (0.031)** | 0.050 (0.036) | 0.025 (0.021) | **0.038 (0.039)** | **0.055 (0.047)** |
| D$y$S | 0.038 (0.029) | **0.040 (0.030)** | **0.020 (0.018)** | 0.014 (0.012) | 0.035 (0.025) | **0.066 (0.058)** |
| D$y$S$_{\texttt{QuaDapt}}$ | **0.037 (0.027)** | 0.048 (0.033) | 0.027 (0.023) | **0.013 (0.011)** | **0.017 (0.013)** | 0.067 (0.053) |
| SMM | **0.051 (0.035)** | 0.070 (0.046) | 0.051 (0.032) | 0.023 (0.017) | 0.090 (0.057) | **0.071 (0.056)** |
| SMM$_{\texttt{QuaDapt}}$ | 0.059 (0.038) | **0.067 (0.044)** | **0.049 (0.036)** | **0.019 (0.014)** | **0.049 (0.027)** | 0.083 (0.063) |
| `QuaDapt` **win rate** | 75% (6/8) | 88% (7/8) | 25% (2/8) | 63% (5/8) | 100% (8/8) | 50% (4/8) |

ated quantifiers, underscoring its robustness to provide broad and reliable performance gains across different data distributions and drift conditions.

Among the quantifiers, X$_{\texttt{QuaDapt}}$ achieved the highest win rate, improving its standard version in 83% of the datasets, followed by ACC$_{\texttt{QuaDapt}}$, MAX$_{\texttt{QuaDapt}}$, MS$_{\texttt{QuaDapt}}$, and HDy$_{\texttt{QuaDapt}}$, each with a 75.0% win rate. For T50$_{\texttt{QuaDapt}}$, D$y$S$_{\texttt{QuaDapt}}$, and SMM$_{\texttt{QuaDapt}}$, the framework improved their performance in 67%. On average, across all datasets and quantifiers, the framework provided an improvement of 73%, highlighting its consistent ability to enhance quantification performance under distribution shifts.

Dermatology and Nursery datasets emerged as special cases in our evaluation. Both datasets present a very high F1-scores for the harder subclasses (0.97 and 0.92), indicating that even their **harder** scenarios remain relatively simple, with well-separated classes and low ambiguity. As a result, standard quantifiers already achieve near-optimal accuracy, leaving limited space for `QuaDapt` to provide improvements. Nevertheless, the proposal achieved competitive results for some quantifiers, such as ACC$_{\texttt{QuaDapt}}$, MAX$_{\texttt{QuaDapt}}$, and X$_{\texttt{QuaDapt}}$, confirming that even in easy scenarios where performance margins are narrow, `QuaDapt` can still match standard methods. In summary, the poor win rates observed in these datasets (13% and 25%) reflect the absence of meaningful drift challenges rather than a shortcoming of the framework itself.

Due to the lack of space, Figure 3 presents the results[4] for only five quantifiers for Avila and Chess game datasets, where the contribution of our proposal was most evident. This figure shows the performance of five quantifiers as the proportion of the harder subclass increases. Larger proportions of the harder subclass lead to greater overlap between classes, thereby making the task progressively more difficult. The vertical dashed line at 0.5 indicates the scenario where training and test sets have the equivalent complexity. For both selected datasets, traditional quantification methods tend to exhibit optimal performance under conditions akin to those undergone during training. However, their effectiveness diminishes even in less challenging scenarios, surprisingly. In contrast, the `QuaDapt` methods show greater resilience in different conditions, maintaining more stable performance in both easier and harder settings.

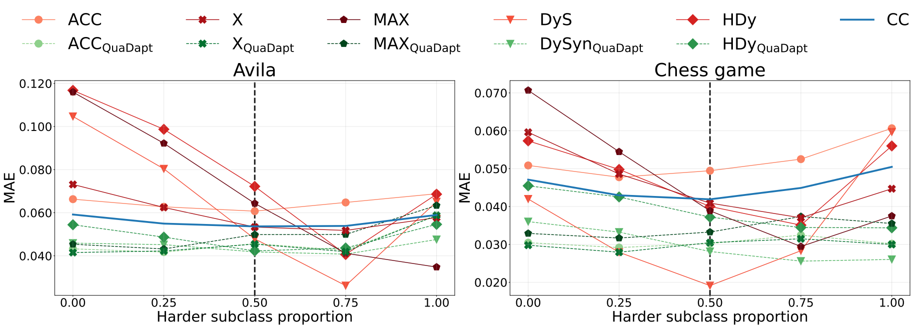

Fig. 3: Quantifiers MAE by harder subclass proportion.

In general, our results on real-world datasets are consistent with the findings from our first experimental setup using artificial scores. Specifically, methods that rely strictly on score distributions and correction rates derived from the training data tend to perform poorly when there is a mismatch between the training and testing distributions, a condition commonly observed under distribution shifts. This degradation occurs even when the test distribution becomes more separable, which makes the task easier.

We highlight a fundamental contrast between classification and quantification under changing data conditions. In classification, when the underlying task becomes easier, such as when the decision boundary becomes more distinct or class separation increases, classification accuracy typically improves, reflecting the reduced complexity of the decision space. However, as observed in both artificial and real-world experiments, this expected behavior does not hold for existing quantifiers. Many quantification methods remain vulnerable, even when the test distribution becomes more separable. This counterintuitive behavior further exposes a critical weakness in current quantification methods. The improvements

[4] All results are available in our supplemental material repository (`https://github.com/JPOrtegaa/QuaDapt-Lequa25`)

observed with `QuaDapt` quantifiers directly address this issue, demonstrating that quantifiers can and should respond positively to easier test conditions.

## 6 Conclusion

In this paper, we present `QuaDapt`, a generic and adaptable framework that enables classifier-based quantifiers to be more resilient to a wide range of distribution shifts, including concept drift. While traditional quantifiers have primarily focused on addressing prior probability drift, `QuaDapt` enhances this capability by dynamically adjusting quantifier parameters based on the observed characteristics of test-time score distributions. This adjustment enables quantifiers to remain effective even when changes in data distribution impact classifier outputs.

Among our main contributions, we formalized the `QuaDapt` framework and demonstrated its effectiveness through extensive experiments on both synthetic and real-world datasets. Our findings provide the first empirical evidence that quantifiers can and should adapt to favorable distributional changes, such as increased class separability, where existing methods frequently fail.

Future research involves integrating drift detection mechanisms and systematically investigating how types of distribution shifts impact quantifier behavior, improving our understanding of quantification vulnerabilities in real-world, nonstationary settings.

**Acknowledgments.** We thank the Araucária Foundation (14/2024), the Innovation Agency for Sustainable Regional Development (AGEUNI - Conv. 41/2024), and the State Secretariat of Science, Technology and Higher Education of Paraná (SETI) for funding and support.

## References

1. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Quantification via probability estimators. In: ICDM. pp. 737–742 (2010)
2. Casale, P., Pujol, O., Radeva, P.: Personalization and user verification in wearable systems using biometric walking patterns. Pers. Ubiquitous Comput. **16**(5), 563–580 (2012)
3. De Stefano, C., Maniaci, M., Fontanella, F., di Freca, A.S.: Reliable writer identification in medieval manuscripts through page layout features: The "avila" bible case. Engineering Applications of Artificial Intelligence **72**, 99–110 (2018)
4. Dheeru, D., Karra Taniskidou, E.: UCI machine learning repository (2017), `http://archive.ics.uci.edu/ml`
5. Dietterich, T.G., Kong, E.B.: Machine learning bias, statistical bias, and statistical variance of decision tree algorithms (1995)
6. Donyavi, Z., Serapiao, A.B., Batista, G.: Mc-sq and mc-mq: Ensembles for multi-class quantification. IEEE Trans. Knowl. Data Eng. (2024)
7. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to Quantify. Springer Nature (2023)

8. Firat, A.: Unified framework for quantification (6 2016), `http://arxiv.org/abs/1606.00868`
9. Flovik, V.: Quantifying distribution shifts and uncertainties for enhanced model robustness in machine learning applications. arXiv preprint arXiv:2405.01978 (2024)
10. Forman, G.: Lnai 3720 - counting positives accurately despite inaccurate classification. Tech. rep. (2005)
11. Forman, G.: Quantifying counts and costs via classification. Data Min. Knowl. Discovery **17**, 164–206 (10 2008)
12. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: SBIA. pp. 286–295. Sao Luis, Maranhao, Brazil (2004)
13. González, P., Díez, J., Chawla, N., del Coz, J.J.: Why is quantification an interesting learning problem? Prog. Artif. Intell. **6**, 53–58 (2017)
14. González, P., Castaño, A., Chawla, N.V., Coz, J.J.D.: A review onquantification learning. ACM Comput. Surv. **50** (9 2017)
15. González, P., Moreo, A., Sebastiani, F.: Binary quantification and dataset shift: an experimental investigation. Data Min. Knowl. Discovery **38**, 1670–1712 (7 2024)
16. González-Castro, V., Alaiz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the hellinger distance. Inf. Sci. **218**, 146–164 (1 2013)
17. Hassan, W., Maletzke, A., Batista, G.: Accurately quantifying a billion instances per second. In: DSAA. pp. 1–10 (10 2020)
18. Li, F., Gharakheili, H.H., Batista, G.: Quantification over time. In: ECML PKDD. pp. 282–299. Springer, Vilnius, Lithuania (2024)
19. Maletzke, A., dos Reis, D., Cherman, E., Batista, G.: On the need of class ratio insensitive drift tests for data streams. In: Second International Workshop on Learning with Imbalanced Domains: Theory and Applications. vol. 94, pp. 110–124. PMLR (10 Sep 2018)
20. Maletzke, A., Reis, D.D., Hassan, W., Batista, G.: Accurately quantifying under score variability. In: ICDM. pp. 1228–1233. IEEE (12 2021)
21. Maletzke, A., Hassan, W., Reis, D., Batista, G.: The importance of the test set size in quantification assessment. In: IJCAI. pp. 2640–2646. Yokohama, Japan (2020)
22. Maletzke, A., Reis, D.D., Cherman, E., Batista, G.: Dys: A framework for mixture models in quantification. In: AAAI. vol. 33, pp. 4552–4560 (2019)
23. Moreno-Torres, J., Raeder, T., Alaiz-Rodríguez, R., Chawla, N., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognit. **45**, 521–530 (2012)
24. Saerens, M., Latinne, P., Decaestecker, C., Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. Tech. rep. (2001)
25. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods. arXiv preprint arXiv:2103.03223 (2021)
26. Storkey, A.: When Training and Test Sets Are Different: Characterizing Learning Transfer, pp. 3–28 (01 2009). `https://doi.org/10.7551/mitpress/9780262170055.003.0001`
27. Tsymbal, A.: The problem of concept drift: definitions and related work. Tech. rep., Department of Computer Science, Trinity College Dublin, Dublin, Ireland (2004)
28. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: Openml: Networked science in machine learning. ACM SIGKDD Explorations Newsletter **15**(2), 49–60 (2013)