Proceedings of the 4th International Workshop on Learning to Quantify (LQ 2024)

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

Preface

The 4th International Workshop on Learning to Quantify (LQ 2024 – `https://lq-2024.github.io/`) was held in Vilnius, LT, on September 13, 2024, as a satellite workshop of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2024). While the 1st edition of the workshop (LQ 2021 – `https://cikmlq2021.github.io/`) had to be an entirely online event, LQ 2024 (like the 2nd edition LQ 2022 – `https://lq-2022.github.io/` and 3rd edition LQ 2023 – `https://lq-2023.github.io/`) was a hybrid event, with presentations given in-presence, and both in-presence attendees and remote attendees.

The workshop was the second part (Sep 13 afternoon) of a full-day event, whose first part (Sep 13 morning) consisted of a tutorial on Learning to Quantify presented by Mirko Bunse and Alejandro Moreo. The LQ 2024 workshop consisted of the presentations of three contributed papers, plu a number of invited contributions about the LeQua 2024 challenge, i.e., an overview of the challenge presented by the organisers, plus five brief presentations by LeQua 2024 participants. The program ended with a final collective discussion on LeQua 2024, on the open problems of learning to quantify, and on future initiatives. The present volume contains the text of these nine contributions.

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

Regular Papers

LeQua 2024: The 2nd International Data Challenge on Learning to Quantify

LQ 2024 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)

Rocío Alaíz-Rodríguez, University of León, ES
Gustavo Batista, University of New South Wales, AU
Juan José del Coz, University of Oviedo, ES
Andrea Esuli, Consiglio Nazionale delle Ricerche, IT
Alessandro Fabris, Max Planck Institute for Security and Privacy, DE
Cèsar Ferri, Universitat Politècnica de València, ES
George Forman, Amazon Research, US
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

Acknowledgments

The work of Alejandro Moreo and Fabrizio Sebastiani has been supported by the SoBigdata++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1, by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020, and by the FAIR, SoBigdata.it, and QuaDaSh projects, funded by the European Union under the NextGenerationEU funding scheme. The organizers' opinions do not necessarily reflect those of the European Commission.

Comments on Friedman's Method for Class Distribution Estimation

Dirk Tasche[0000−0002−2750−2970]

Unit for Data Science and Computing, North-West University, South Africa
55801447@nwu.ac.za

Abstract. The purpose of class distribution estimation (also known as quantification) is to determine the values of the prior class probabilities in a test dataset without class label observations. A variety of methods to achieve this have been proposed in the literature, most of them based on the assumption that the distributions of the training and test data are related through prior probability shift (also known as label shift). Among these methods, Friedman's method has recently been found to perform relatively well both for binary and multi-class quantification. We discuss the properties of Friedman's method and another approach mentioned by Friedman (called DeBias method in the literature) in the context of a general framework for designing linear equation systems for class distribution estimation.

Keywords: Prior probability shift · Label shift · Class prevalence · Quantification · Asymptotic variance

1 Introduction

The purpose of class distribution estimation (also known as quantification) is to determine the values of the prior class probabilities in a test dataset without class label observations. A variety of methods to achieve this have been proposed in the literature, most of them based on the assumption that the distributions of the training and test data are related through prior probability shift (also known as label shift). See González et al. [10] and Esuli et al. [6] for recent surveys of applications of and methods for quantification.

Friedman's [9] method has recently been found to perform relatively well both for binary and multi-class quantification (Schuhmacher et al. [18], Donyavi et al. [5]). On many real-world datasets, the performance of Friedman's method seems to exceed the performance of the EM algorithm (Saerens et al. [16]) which is an implementation of the maximum likelihood estimator for the test prior class probabilities (also called class prevalences). This observation is somewhat surprising because both Friedman's estimator and the EM algorithm involve estimates of the training posterior class probabilities which are notoriously hard to estimate. Hence one might expect that the performances of Friedman's method and the EM algorithm are at a more comparable level.

In order to find an explanation for the relatively good performance of Friedman's method, we study its properties and the properties of another approach mentioned by Friedman (called DeBias method by Castaño et al. [4]) in the context of a general framework for designing linear equation systems for class distribution estimation.

The outline of this paper and its contributions to the literature are as follows:

- Section 2 sets out the general assumptions and notation for the rest of the paper.
- In Section 3, we discuss the general framework for designing linear equation systems for class distribution estimation. (3b) and (3c) of Theorem 1 below appear to be novel, covariance-based versions of the basic equation (3a).
- In Section 4, we describe Friedman's method in detail and propose an alternative implementation that avoids direct estimation of the posterior class probabilities (Remark 1 below).
- In Section 5, we investigate conditions for the uniqueness of the solutions to linear equation systems for class distribution estimation. In Remark 3, we show that DeBias, the second method proposed for the binary case by Friedman [9] which involves the variance of one of the posterior class probabilities, is a special case of a covariance matrix-based approach to the multi-class case considered in Corollary 2. This provides an answer to the open research question "How to generalise the inequality of Corollary 6 of Tasche [21] to the multi-class case?" raised in Section 4.12 of Krempl et al. [13]. In addition, we show that the population versions of DeBias and 'Probabilistic adjusted count (PAC)' by Bella et al. [1] are identical (Remark 5 below).
- In Section 6, we compare the asymptotic variances of DeBias, Friedman's method and the maximum likelihood estimator in the binary case by means of a numerical example. The setting of the example is semi-asymptotic with an infinite training dataset and a finite large test dataset.
- Section 7 concludes the paper with a summary of the findings.

2 Setting

For this paper, we assume the following setting which is quite common in the study of dataset shift (see, for instance, Moreno-Torres et al. [15]):

- A class variable Y with values in $\mathcal{Y} = \{1, \ldots, \ell\}$ with $\ell \geq 2$ (multi-class case). A features vector X with values in $\mathcal{X}$.
- Each example (or instance) 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) of which the training dataset has been sampled.
- In the test dataset, only the features X of an example can immediately be observed. Its class label Y becomes known only with delay or not at all. Q denotes the test (joint) distribution, also called target distribution, of (X, Y) of which the test dataset has been sampled.

- We assume $0 < P[Y = y] < 1$, $0 < Q[Y = y] < 1$ for all $y \in \mathcal{Y}$.
- For the sake of a more concise notation, we define $p_y = P[Y = y]$ and $q_y = Q[Y = y]$ for $y \in \mathcal{Y}$.

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* or *distribution shift* in the literature if training and test distribution are not the same, i.e. $P \neq Q$. In the rest of the paper, we consider the following more specific type of dataset shift.

Definition 1. *The training distribution P and the test distribution Q are related through* prior probability shift *if for all $y \in \mathcal{Y}$ and all measurable sets $M \subset \mathcal{X}$ it holds that*[1]

$$P[X \in M | Y = y] \;=\; Q[X \in M | Y = y].$$

The term 'prior probability shift' appears to have been coined by Storkey [20]. In the literature, prior probability shift is also called *target shift* (Zhang et al. [27]), *label shift* (Lipton et al. [14]), or *global drift* (Hofer and Krempl [12]).

Prior probability shift implies dataset shift, i.e. $P \neq Q$, if $P[Y = y] \neq Q[Y = y]$ for at least one $y \in \mathcal{Y}$. Hence, as the class labels Y are not observed in the test dataset, the test prior probabilities $q_y = Q[Y = y]$ must be estimated from feature observations in the test dataset as well as feature and class label observations in the training dataset. Such an estimation procedure is called *quantification* or *class distribution estimation*.

3 Linear equations for class distribution estimation

In the following, we treat class distribution estimation under prior probability shift as a parametric estimation problem in a family of mixture distributions:

- We consider the distributions Q_X on $\mathcal{X}$ that can be represented as

$$Q_X[M] \;=\; \sum\nolimits_{y=1}^{\ell} q_y \, P[X \in M | Y = y] \tag{1}$$

  for all measurable sets $M \subset \mathcal{X}$. The family of these distributions is parametrised through the test prior class probabilities $(q_1, \ldots, q_\ell) \in (0, 1)^\ell$ with the additional constraint

$$\sum\nolimits_{y=1}^{\ell} q_y \;=\; 1. \tag{2}$$

- Unless stated otherwise, for the purposes of this paper we assume that the conditional feature distributions $P[X \in M | Y = y]$, $M \subset \mathcal{X}$, under the training distribution are known and do not contribute to the estimation uncertainty.

[1] Recall the notion of conditional probability for events A and B: $P[A|B] = \frac{P[A \cap B]}{P[B]}$ if $P[B] > 0$ and $P[A|B] = 0$ otherwise.

– The parametrised distribution family defined in (1) is identifiable in the sense of Definition 11.2.2 of Casella and Berger [3], i.e. Q_X and Q_X^* differ whenever the corresponding parametrisations $(q_1, \ldots, q_\ell)$ and $(q_1^*, \ldots, q_\ell^*)$ differ.

According to San Martín and Quintana [17], identifiability is necessary for the existence of both asymptotically unbiased estimates and consistent estimates. This observation leaves open the question of how to find such estimates. In the following, we strive to design estimators of the class prior probabilities q_y as unique solutions to systems of linear equations[2].

Calling the following result a theorem is an exaggeration as its proof is very short and basic. But it is fundamental for the study and estimation of prior probability shift and in that sense deserves being called a theorem. Of course, Theorem 1 is not novel. In particular (3a) was mentioned by Saerens et al. [16] (Eq. (2.5), with Z chosen as a hard classifier) and quite likely also in earlier works. Even so, linking the notion of prior probability shift to the training dataset covariances of functions of the features and the indicators of the classes or the posterior class probabilities might have some degree of novelty, at least in the multi-class case.

Theorem 1. *Let $p_y = P[Y = y]$ and $q_y = Q[Y = y]$ for $y \in \mathcal{Y}$. Suppose that P and Q are related through prior probability shift in the sense of Definition 1 and that the random variable Z is integrable both under P and Q. Then it holds that*[3]

$$E_Q[Z] = \sum\nolimits_{y=1}^{\ell} q_y \, E_P[Z|Y = y] \tag{3a}$$

$$= \sum\nolimits_{y=1}^{\ell} \frac{q_y}{p_y} \operatorname{cov}_P\big(Z, \, \mathbf{1}_{\{Y=y\}}\big) + E_P[Z]. \tag{3b}$$

If Z is X-measurable, i.e. if there is a function $f : \mathcal{X} \to \mathbb{R}$ such that $Z = f(X)$, then it also follows that[4]

$$E_Q[Z] = \sum\nolimits_{y=1}^{\ell} \frac{q_y}{p_y} \operatorname{cov}_P\big(Z, \, P[Y = y|X]\big) + E_P[Z]. \tag{3c}$$

Proof. The theorem follows from the law of total probability combined with the definitions of conditional expectation and covariance respectively. □

(3a) provides the theoretical basis for Firat's ([7], Section 3.2) constrained regression approach for quantification under prior probability shift. Firat's K

[2] Other popular approaches to designing estimators include distribution matching (Gonzàlez et al. [10] and the references therein), ensemble methods (Serapião et al. [19] and the references therein) and expectation maximisation as implementation of maximum likelihood estimation (Saerens et al. [16]).

[3] For sets S, define the indicator function $\mathbf{1}_S$ by $\mathbf{1}_S(s) = 1$ if $s \in S$ and $\mathbf{1}_S(s) = 0$ if $s \notin S$.

[4] $P[Y = y|X]$ denotes the posterior probability of $Y = y$ given X in the sense of general conditional probability as defined, for instance, in Section 33 of Billingsley [2].

classes correspond to the ℓ classes of this paper. The L rows of Firat's matrix $\mathbf{X}$ emerge when (3a) is applied to L different variables $Z_1, \ldots, Z_L$.

As noted by Firat, (3a), (3b) or (3c) can be the starting point for setting up a system of linear equations for estimating the class prior probabilities q_y under prior probability shift. For instance, the choice $f_y(X) = \mathbf{1}_{C_y}(X)$ as crisp (or hard) 'one vs. all' classifier for class y, learned on the training dataset only, leads to the 'Adjusted Count' estimation approach used in the popular paper by Lipton et al. [14] who described it as 'method of moments'. Observe that for this version of 'one vs. all', there is no problem with changing the type of dataset shift, in contrast to the issue for combined 'one vs. all' quantifiers noted by Friedman [9] and Donyavi et al. [5].

Some questions should be considered when designing a concrete instance of such a linear equation system for quantification.

How many equations should be used? If the number of classes in the model is $\ell = |\mathcal{Y}|$ one might conclude that at least ℓ equations are needed in order to obtain a unique solution. However, as another consequence of the law of total probability, the q_y must additionally fulfil the linear equation (2). Hence, in order to achieve uniqueness of the solution, at least ℓ equations must be set up if (2) is considered a constraint that is checked once a solution has been found. Alternatively, if (2) is to be taken into account at the same time as the other equations, for uniqueness as a minimum it suffices to set up $\ell - 1$ additional equations on the basis of Theorem 1. Sticking with $\ell - 1$ equations has the advantage of reducing the number of random variables Z that must be chosen for the equations in Theorem 1.

If more then ℓ equations are set up the resulting linear equation system for the q_y is overdetermined such that in its sample-based versions there might be no exact solution at all. Nonetheless, the overdetermined case is naturally encountered when distribution-matching algorithms are implemented via binning of the feature space $\mathcal{X}$ (DF_x methods) or of the range of a continuous scoring classifier (DF_y methods), see Firat [7], Castaño et al. [4] and the references in the latter paper. To work around the lack of exact solutions, typically approximate solutions are determined by jointly minimising the differences between the left-hand and right-hand sides of the equations with respect to some specific metric like the Euclidean norm or the Hellinger divergence (see for instance Castaño et al. [4]).

In the following, we focus on the cases of systems of ℓ and $\ell - 1$ equations, in the latter case together with constraint (2).

How should the random variables Z appearing in the equations of Theorem 1 be chosen? A very basic criterion for choosing the variables Z is that it must be possible to compute them from observations of the features X only. This follows from the fact that on the left-hand sides of the equations in Theorem 1, the variables Z are integrated under the test distribution Q but the class labels Y are not observed under Q. Hence one must make sure that $Z = f(X)$ for some function f.

Among others, the following criteria for selecting such functions f have been considered in the literature:

- Reducing the variances of the estimated q_y. See Friedman [9] and Vaz et al. [25] for approaches to the direct minimisation of the variance. Findings by Vaz et al. [24] and Tasche [22] suggest that deploying variables Z that are able to separate the classes with high accuracy also reduces the variances of the class prior estimates.
- Speed of computation. See for instance Hassan et al. [11].

With the exception of Hassan et al. [11], in the literature primarily the choices $Z = \mathbf{1}_{C_y}$ (hard classifier for one of the classes y in $\mathcal{Y}$) and $Z = P[Y = y|X]$ (posterior probability under P for class y) have been considered. Below, we consider Friedman's [9] choices of hard classifiers and $Z = P[Y = y|X]$ in more detail.

4 Friedman's method

Friedman [9] proposed two class distribution estimation methods:

- He discussed in detail one method (later called 'Friedman's method' by Schuhmacher et al. [18]) based on a specific choice of hard classifiers both for the binary and multi-class cases. We revisit Friedman's method in this section.
- Another method, specified only for the binary case, is based on the variance of the posterior positive class probability under the training distribution (later called 'DeBias' method by Castaño et al. [4]). This method, without being named, had been mentioned before by Tasche [21] as Corollary 6. We discuss this approach in Remark 3 below.

First, we consider Friedman's method in the binary case $\ell = 2$. As Friedman himself wrote he was not the first researcher to think about this method.

Method Max (Forman [8], Section 2.2). Forman wrote on page 173: "Considering the earlier discussion of small denominators, another likely policy is where the denominator *(tpr-fpr)* is maximized: *method Max*." Here, Forman referred to crisp binary classifiers (not necessary most accurate) which were derived from a 'raw classifier' (i.e. a real-valued scoring classifier).

Accordingly, Friedman's method in the binary case is the special case of Forman's method Max when the underlying scoring classifier is chosen as the Bayes classifier, i.e. the posterior probability of the positive class.

Derivation of Friedman's method. Firat [7] describes on p. 2 the rationale for Friedman's method as follows: "Friedman uses the optimum threshold that minimizes the variance of proportion estimates (Friedman, 2014)." This statement is somewhat misleading, as Friedman [9] actually does not maximise the variance of the estimator but only the denominator on the right-hand side of the following equation (in the notation of this paper)

$$q_1 \;=\; \frac{E_Q[Z] - E_P[Z|y=2]}{E_P[Z|y=1] - E_P[Z|y=2]}, \tag{4}$$

over all random variables $0 \leq Z = f(X) \leq 1$. Note that (4) is a special case of (3a) for $\ell = 2$.

It turns out that

$$\arg \max_{f:\mathcal{X}\to[0,1]} E_P[f(X)|y=1] - E_P[f(X)|y=2] \;=\; f^* \tag{5}$$

with $f^*(x) = 1$ if $P[Y=1|X=x] > p_1$, $f^*(x) = 0$ if $P[Y=1|X=x] < p_1$ and $f^*(x)$ arbitrary if $P[Y=1|X=x] = p_1$.

The solution to the problem of minimising the sample variance of the estimator defined by (4) is less obvious. It has been tackled numerically by Vaz et al. ([25], Section 2.3), and by Tian et al. [23] by involving influence functions.

Remark 1. Friedman [9] and subsequent users of his method appear to have implemented it by means of plugging-in an estimate of the posterior probability $P[Y=1|X]$ into the function f^* as defined in (5). However, as $P[Y=1|X]$ could be difficult to estimate with satisfactory accuracy, such an implementation might be suboptimal.

Note that (5) is equivalent to

$$\arg \min_{f:\mathcal{X}\to[0,1]} (1-p_1)\, E_P[f(X)\, \mathbf{1}_{\{Y=1\}}] + p_1\, E_P[(1-f(X))\, \mathbf{1}_{\{Y=2\}}] \;=\; 1-f^*, \tag{6}$$

with f^* as in (5). (6) can be read as the problem to minimise the expected cost-sensitive error for a binary classification problem. This problem can be dealt with directly through a variety of approaches, resulting in approximations of the optimal classifier which do not require the estimation of $P[Y=1|X]$. The cost-sensitive minimisation problem can also be translated into a standard classification problem by appropriate re-weighting (Zadrozny et al. [26]). □

Friedman's method for more than two classes. Friedman [9] suggested defining $Z_y = f_y^*(X)$ for $y \in \mathcal{Y}$ with $f_y^*(x) = 1$ if $P[Y=y|X=x] > p_y$, $f_y^*(x) = 0$ if $P[Y=y|X=x] \leq p_y$, and then using (3a) with Z_y, $y = 1, \ldots, \ell$, to obtain a system of ℓ linear equations for the test prior probabilities of the classes $y \in \mathcal{Y}$.

According to Schuhmacher et al. [18], Friedman's method works well in binary quantification problems and still achieves good performance in multi-class settings.

5 Uniqueness of solutions and covariance matrix-based approaches

As discussed in Section 3, uniqueness of the solutions is an important design criterion for setting up a system of linear equations for class distribution estimation under prior probability shift. In this section, we provide more detail regarding the number of equations needed and look closer at designs based on covariance matrices estimated in the training dataset.

5.1 How many equations are needed?

(3c) of Theorem 1 is interesting because the choice $Z = P[Y = y|X]$ for fixed $y = 1, \ldots, \ell$, implies the matrix identity

$$\begin{pmatrix} E_Q\big[P[Y=1|X]\big] - p_1 \\ \vdots \\ E_Q\big[P[Y=\ell|X]\big] - p_\ell \end{pmatrix} = \Sigma_P \times \begin{pmatrix} \frac{q_1}{p_1} \\ \vdots \\ \frac{q_\ell}{p_\ell} \end{pmatrix}, \tag{7}$$

$$\Sigma_P = \begin{pmatrix} \mathrm{cov}_P\big(P[Y=1|X],\, P[Y=1|X]\big) & \ldots & \mathrm{cov}_P\big(P[Y=1|X],\, P[Y=\ell|X]\big) \\ \vdots & \ddots & \vdots \\ \mathrm{cov}_P\big(P[Y=\ell|X],\, P[Y=1|X]\big) & \ldots & \mathrm{cov}_P\big(P[Y=\ell|X],\, P[Y=\ell|X]\big) \end{pmatrix}.$$

(7) connects the prior class probabilities p_y under the training distribution, the prior class probabilities q_y under the test distribution, and the averages under the test distribution $E_Q\big[P[Y = y|X]\big]$ of the training posterior class probabilities through the covariance matrix Σ_P of the training posterior probabilities under the training distribution. All quantities in (7) but the test class prior probabilities q_y can be estimated from the training dataset and the features in the test dataset in principle. Hence, if the square matrix Σ_P were invertible, (7) could be solved for the q_y by matrix inversion.

Unfortunately, as follows from the following proposition, the covariance matrix Σ_P is never invertible.

Proposition 1. *Let $Z_1, \ldots, Z_r$ be integrable random variables under the distribution P. Suppose that Y is a discrete random variable with values in $\mathcal{Y} = \{1, \ldots, \ell\}$ with $\ell \geq 2$ and X is a random vector with values in $\mathcal{X}$. Define the matrices $M = (m_{ij})_{\substack{i=1,\ldots,r \\ j=1,\ldots,\ell}}$ and $M^* = (m^*_{ij})_{\substack{i=1,\ldots,r \\ j=1,\ldots,\ell}}$ by*

$$m_{ij} = \mathrm{cov}(Z_i,\, \mathbf{1}_{\{Y=j\}}) \quad \textit{and} \quad m^*_{ij} = \mathrm{cov}\big(Z_i,\, P[Y=j|X]\big).$$

Then it follows that

$$\mathrm{rank}(M) \leq \ell - 1 \quad \textit{and} \quad \mathrm{rank}(M^*) \leq \ell - 1.$$

Proof. Due to the fact that $1 = \sum_{j=1}^{\ell} \mathbf{1}_{\{Y=j\}}$ and $1 = \sum_{j=1}^{\ell} P[Y = j|X]$, the vector $v = (1, 1, \ldots, 1)^T \in \mathbb{R}^\ell$ is an element of the kernels of M and M^*, i.e. it holds that $M \times v = 0 = M^* \times v$. This implies the assertion. □

As a consequence of Proposition 1, there is no possible choice of random variables $Z_1, \ldots, Z_\ell$ that could serve on the basis of (3b) or (3c) to create a system of ℓ linear equations with a unique solution for the ℓ unknowns $q_1, \ldots, q_\ell$. However, Proposition 1 leaves open the question if such an equation system can be constructed on the basis of (3a).

Remark 2. For integrable random variables $Z_1, \ldots, Z_r$, define the matrix $\widetilde{M} = (\widetilde{m}_{ij})_{\substack{i=1,\ldots,r \\ j=1,\ldots,\ell}}$ by $\widetilde{m}_{ij} = E_P[Z_i|Y = j]$.

$\widetilde{M}$ can be rewritten as

$$\widetilde{M} \;=\; L \times D, \tag{8a}$$

where

$$L \;=\; \begin{pmatrix} E_P[Z_1\,\mathbf{1}_{\{Y=1\}}] & \dots & E_P[Z_1\,\mathbf{1}_{\{Y=\ell\}}] \\ \vdots & \ddots & \vdots \\ E_P[Z_r\,\mathbf{1}_{\{Y=1\}}] & \dots & E_P[Z_r\,\mathbf{1}_{\{Y=\ell\}}] \end{pmatrix} \tag{8b}$$

and $D = (d_{ij})_{i,j=1,\dots,\ell}$ is the diagonal matrix with $d_{ij} = \frac{1}{p_i}$ if $i = j$ and $d_{ij} = 0$ if $i \neq j$. In particular, we have $\operatorname{rank}(D) = \ell$.

Define the vector $v = (1, 1, \dots, 1)^T$ as in the proof of Proposition 1. Then it follows that $L \times v = (E_P[Z_1], \dots, E_P[Z_r])^T$. If $Z_1, \dots, Z_r$ are chosen such that $(E_P[Z_1], \dots, E_P[Z_r]) \neq 0$, as a consequence $L \times v \neq 0$ results. Hence there is no obvious reason as in the case of Proposition 1 for the rank of L (and by (8a) also of $\widetilde{M}$) to be less than maximal, i.e. being equal to $\min(r, \ell)$. This observation suggests that (3a) can be used to obtain a system of ℓ linear equations with a unique solution for the test class prior probabilities $q_1, \dots, q_\ell$. □

5.2 Invertible covariance matrices

The fact that the covariance Σ_P of the posterior class probabilities $P[Y = y|X]$, $y \in \mathcal{Y}$ in (7) cannot be inverted is caused by the linear dependence between the posterior probabilities since $\sum_{y=1}^{\ell} P[Y = y|X] = 1$. This issue can be avoided by disregarding one of probabilities, say $P[Y = \ell|X]$. Indeed, making use of the identity $\mathbf{1}_{\{Y=\ell\}} = 1 - \sum_{y=1}^{\ell-1} \mathbf{1}_{\{Y=y\}}$ in (3b) produces the following corollary to Theorem 1.

Corollary 1. *Let $p_y = P[Y = y]$ and $q_y = Q[Y = y]$ for $y \in \mathcal{Y}$. Suppose that P and Q are related through prior probability shift in the sense of Definition 1 and that the random variable Z is integrable both under P and Q. Then it holds that*

$$E_Q[Z] = \sum\nolimits_{y=1}^{\ell-1} \left(\frac{q_y}{p_y} - \frac{q_\ell}{p_\ell}\right) \operatorname{cov}_P\big(Z,\, \mathbf{1}_{\{Y=y\}}\big) + E_P[Z]. \tag{9a}$$

If Z is X-measurable, i.e. if there is a function $f : \mathcal{X} \to \mathbb{R}$ such that $Z = f(X)$, then it also follows that

$$E_Q[Z] = \sum\nolimits_{y=1}^{\ell-1} \left(\frac{q_y}{p_y} - \frac{q_\ell}{p_\ell}\right) \operatorname{cov}_P\big(Z,\, P[Y = y|X]\big) + E_P[Z]. \tag{9b}$$

Corollary 1 suggests the following approach to estimating the test class prior probabilities $q_1, \dots, q_\ell$.

Corollary 2. *Assume that the functions $f_1, \dots, f_{\ell-1} : X \to \mathbb{R}$ are such that the matrix*

$$C \;=\; \begin{pmatrix} \operatorname{cov}(f_1(X),\, \mathbf{1}_{\{Y=1\}}) & \dots & \operatorname{cov}(f_1(X),\, \mathbf{1}_{\{Y=\ell-1\}}) \\ \vdots & \ddots & \vdots \\ \operatorname{cov}(f_{\ell-1}(X),\, \mathbf{1}_{\{Y=1\}}) & \dots & \operatorname{cov}(f_{\ell-1}(X),\, \mathbf{1}_{\{Y=\ell-1\}}) \end{pmatrix} \tag{10a}$$

has rank $\ell - 1$, i.e. it is invertible.

Let $\big(E_Q[f_1(X)] - E_P[f_1(X)], \ldots, E_Q[f_{\ell-1}(X)] - E_P[f_{\ell-1}(X)]\big)^T = z$ *and* $C^{-1} \times z = (s_1, \ldots, s_{\ell-1})^T$.

Then it follows that

$$q_y = p_y \left(s_y + 1 - \sum\nolimits_{i=1}^{\ell-1} p_i \, s_i \right), \; y = 1, \ldots, \ell - 1, \quad q_\ell = p_\ell \left(1 - \sum\nolimits_{i=1}^{\ell-1} p_i \, s_i \right). \tag{10b}$$

Observe that as a consequence of the general properties of conditional expectation[5] matrix C of (10a) can be represented as

$$C = \begin{pmatrix} \operatorname{cov}(f_1(X),\, P[Y = 1|X]) & \ldots & \operatorname{cov}(f_1(X),\, P[Y = \ell - 1|X]) \\ \vdots & \ddots & \vdots \\ \operatorname{cov}(f_{\ell-1}(X),\, P[Y = 1|X]) & \ldots & \operatorname{cov}(f_{\ell-1}(X),\, P[Y = \ell - 1|X]) \end{pmatrix}. \tag{11}$$

With the special choice $f_y(X) = P[Y = y|X]$ for $y = 1, \ldots, \ell - 1$ matrix C as represented in (11) becomes the covariance matrix of Σ_P^* of $P[Y = 1|X], \ldots, P[Y = \ell - 1|X]$.

Remark 3 (DeBias method). Suppose we are in the binary case $\ell = 2$ and apply Corollary 2 with C as given in (11) and $f_1(X) = P[Y = 1|X]$. This implies $C = \Sigma_P^* = \operatorname{var}\big[P[Y = 1|X]\big]$. We then obtain by means of (10b)

$$q_1 = \frac{p_1 \, (1 - p_1)}{\operatorname{var}_P\big[P[Y = 1|X]\big]} \left(E_Q\big[P[Y = 1|X]\big] - p_1 \right) + p_1, \tag{12a}$$

which is equivalent to

$$E_Q\big[P[Y = 1|X]\big] = q_1 \, \frac{\operatorname{var}_P\big[P[Y = 1|X]\big]}{p_1 \, (1 - p_1)} + p_1 \left(1 - \frac{\operatorname{var}_P\big[P[Y = 1|X]\big]}{p_1 \, (1 - p_1)} \right). \tag{12b}$$

(12b) appears to have been first published by Tasche [21] (Corollary 6) and then to have been presented at a conference by Friedman [9]. This approach to estimating q_1 has been called 'DeBias' method by Castaño et al. [4].

Hence, Corollary 2 with $C = \Sigma_P^*$ may be interpreted as multi-class extension of the DeBias approach. □

Remark 4 (Probabilistic Adjusted Count (PAC)). Suppose again we are in the binary case $\ell = 2$ and apply Corollary 2, this time with C as represented in (10a) and $f_1(X) = P[Y = 1|X]$. This implies $C = E_P[\big[P[Y = 1|X]\, \mathbf{1}_{\{Y=1\}}\big] - p_1^2$. We then obtain by means of (10b)

$$q_1 = p_1 \, (1 - p_1) \frac{E_Q\big[P[Y = 1|X]\big] - p_1}{E_P[\big[P[Y = 1|X]\, \mathbf{1}_{\{Y=1\}}\big] - p_1^2} + p_1 \tag{13a}$$

[5] See, for instance, Problem 34.6 of Billingsley [2]).

which is equivalent to

$$q_1 = \frac{E_Q\big[P[Y=1|X]\big] - E_P\big[P[Y=1|X] \;\big|\; Y=2\big]}{E_P\big[P[Y=1|X] \;\big|\; Y=1\big] - E_P\big[P[Y=1|X] \;\big|\; Y=2\big]}. \tag{13b}$$

(13b) was called 'probability estimation & average (P&A)' method by Bella et al. [1] but is now commonly referred to as 'probabilistic adjusted count (PAC)' (González et al. [10]). Its multi-class extension is sometimes called 'generalized probabilistic adjusted count (GPAC)' (see, for instance, Schuhmacher et al. [18]) and also covered by Corollary 2 with the choice $f_y(X) = P[Y=y|X]$ in (10a). □

Remark 5. Observe that in (13a) it holds that

$$E_P[\big[P[Y=1|X]\,\mathbf{1}_{\{Y=1\}}\big] - p_1^2 \;=\; \mathrm{var}_P\big[P[Y=1|X]\big].$$

By (12a), therefore in the binary case the DeBias and PAC methods for class distribution estimation are identical at population level, i.e. with infinite training and test datasets. This observation is not necessarily true in practice when DeBias and PAC estimates respectively are calculated based on sample versions of (12a) and (13a). □

6 Comparing asymptotic variances

As mentioned in Section 2, we consider class distribution estimation as a two-sample problem:

- A training sample for estimating certain quantities (e.g. the true positive and false negative rates of a classifier) under the training distribution because the quantities are needed for estimating the class prior probabilities under the test distribution.
- A test sample for estimating the class prior probabilities under the test distribution, based on the quantities estimated on the training sample.

Hence minimising the error of a method for class distribution estimation means minimising the estimation errors on the two samples.

In the following, we look at the semi-asymptotic binary case ($\ell = 2$) where

- the training distribution P is known (infinite sample) such that the prior class probabilities p_y and the posterior class probabilities $P[Y=y|X]$ can be exactly determined in the sense that the estimation error on the training sample vanishes.
- From the test distribution a finite but large sample of size n is given, and we focus upon unbiased estimators of the class prior probabilities.

For unbiased estimators the Cramér-Rao lower bound specifies a minimum value for the variance that cannot be undercut. Denote by $\widehat{q}_n^{\mathrm{ML}}$ the maximum-likelihood

(ML) estimator of the test prior probability q_1 of class 1 and by σ^2_{ML} its so-called asymptotic variance. Then $\frac{\sigma^2_{\mathrm{ML}}}{n}$ is the Cramér-Rao lower bound for the variances of the unbiased estimators of q_1 on test samples of size n when the training distribution is known (called here 'asymptotic setting'), see Section 5 of Tasche [22].

We compare σ^2_{ML} with the asymptotic variances in the sense of Definition 10.1.9 of Casella and Berger [3] of the Friedman estimator $\widehat{q}_n^{\mathrm{Fried}}$ and the DeBias estimator $\widehat{q}_n^{\mathrm{DeBias}}$ of the test prior probability q_1 of class 1.

We assume that both the conditional distribution of X given $Y = 1$ and the conditional distribution of X given $Y = 2$ have densities $g_1 > 0$ and $g_2 > 0$ with respect to some measure[6] μ. In particular, then the posterior probability $P[Y = 1|X = x]$ can be represented as

$$P[Y = 1|X = x] \;=\; \frac{p_1\, g_1(x)}{p_1\, g_1(x) + (1 - p_1)\, g_2(x)}, \tag{14}$$

and the density of the feature vector X under the test distribution Q is given by

$$g_Q \;=\; q_1\, g_1 + (1 - q_1)\, g_2. \tag{15}$$

Since the training distribution P is assumed to be known, in the following all expected values $E_P[Z]$ are deterministic values that need not be estimated. In particular, also the prior probabilities p_1 and $p_2 = 1 - p_1$ are known constants. In contrast, the test distribution Q is not known but an i.i.d. sample $X_1, \ldots, X_n$ of the feature vector X drawn from its distribution under Q is observed.

ML estimator. For a detailed description of the ML estimator $\widehat{q}_n^{\mathrm{ML}}(X_1, \ldots, X_n) = \widehat{q}_n^{\mathrm{ML}}$ we refer to Section 4 of Tasche [22], as there is no closed-form representation of the ML estimator. However, its asymptotic variance σ_{ML} under Q is known:

$$\sigma^2_{\mathrm{ML}} = E_Q\left[\left(\frac{g_1(X) - g_2(X)}{g_Q(X)}\right)^2\right]^{-1} \;=\; \frac{q_1^2\,(1 - q_1)^2}{\mathrm{var}_Q\big[E_Q[Y = 1|X]\big]}. \tag{16}$$

σ^2_{ML} is characterised through the property that $\sqrt{n}\,\big(\widehat{q}_n^{\mathrm{ML}} - q_1\big)$ converges in distribution toward the normal distribution with mean 0 and variance σ^2_{ML}. Observe that σ^2_{ML} is a function of q_1 but not of p_1.

Friedman estimator. In the binary case, under the assumption on semi-asymptotics made for this section, the Friedman estimator $\widehat{q}_n^{\mathrm{Fried}}(X_1, \ldots, X_n) = \widehat{q}_n^{\mathrm{Fried}}$ based on the homonymous method presented in Section 4 can be written as

$$\widehat{q}_n^{\mathrm{Fried}} \;=\; \frac{\frac{1}{n}\sum_{i=1}^n f^*(X_i) - E_P[f^*(X)|y = 2]}{E_P[f^*(X)|y = 1] - E_P[f^*(X)|y = 2]}, \tag{17a}$$

with f^* defined through (5). Friedman [9] observed that f^* can also be represented as

$$f^*(x) \;=\; \begin{cases} 1, & \text{if } g_1(x) > g_2(x), \\ 0, & \text{if } g_1(x) \le g_2(x). \end{cases} \tag{17b}$$

[6] In Example 1 below μ is the Lebesgue measure on $\mathbb{R}$.

As a consequence of (17b), the right-hand side of (17a) does not depend on p_1 or p_2 for $f^*(X)$ or any of the $f^*(X_i)$. Therefore, also $\widehat{q}_n^{\,\mathrm{Fried}}$ as defined in (17a) does not change if p_1 or p_2 are changed. From the central limit theorem, it follows that $\sqrt{n}\,(\widehat{q}_n^{\,\mathrm{Fried}} - q_1)$ under Q converges toward a normal distribution with mean 0 and variance $\sigma^2_{\mathrm{Fried}}$. More precisely, the asymptotic variance of $\widehat{q}_n^{\,\mathrm{Fried}}$ is

$$\sigma^2_{\mathrm{Fried}} \;=\; \frac{E_Q[f^*(X)]\,\big(1 - E_Q[f^*(X)]\big)}{(E_P[f^*(X)|y=1] - E_P[f^*(X)|y=2])^2}. \tag{17c}$$

DeBias estimator. In the binary case, under the assumption on semi-asymptotics made for this section, the DeBias estimator $\widehat{q}_n^{\,\mathrm{DeBias}} = \widehat{q}_n^{\,\mathrm{DeBias}}(X_1, \ldots, X_n)$ based on the method presented in Remark 3 can be written as

$$\widehat{q}_n^{\,\mathrm{DeBias}} \;=\; \frac{p_1\,(1-p_1)}{\mathrm{var}_P\big[P[Y=1|X]\big]} \left(\frac{1}{n} \sum\nolimits_{i=1}^{n} P[Y=1|X=X_i] - p_1 \right) + p_1. \tag{18a}$$

From the central limit theorem, it follows that $\sqrt{n}\,(\widehat{q}_n^{\,\mathrm{DeBias}} - q_1)$ under Q converges toward the normal distribution with mean 0 and variance $\sigma^2_{\mathrm{DeBias}}$, or more precisely, the asymptotic variance of $\widehat{q}_n^{\,\mathrm{DeBias}}$ is

$$\sigma^2_{\mathrm{DeBias}} \;=\; \left(\frac{p_1\,(1-p_1)}{\mathrm{var}_P\big[P[Y=1|X]\big]} \right)^2 \mathrm{var}_Q\big[P[Y=1|X]\big]. \tag{18b}$$

Note that it follows from (16) and (18b) that $\sigma^2_{\mathrm{ML}} = \sigma^2_{\mathrm{DeBias}}$ in the case of no shift, i.e. $p_1 = q_1$. As all quantities derived from P are assumed to be constant in the setting of this section, it follows as in Remark 5 that the asymptotic variance σ^2_{PAC} of the PAC estimator discussed in Remark 4 is identical with $\sigma^2_{\mathrm{DeBias}}$, i.e. $\sigma^2_{\mathrm{DeBias}} = \sigma^2_{\mathrm{PAC}}$. For this reason, PAC is omitted from the following numerical example.

Example 1. We consider the same univariate binormal model with equal variances of the class-conditional distributions as in Section 7 of Tasche [22]: The two normal class-conditional distributions of the feature variable X are given by

$$X \,|\, Y = i \;\sim\; \mathcal{N}(\mu_i, \sigma^2), \qquad i = 1, 2 \tag{19a}$$

for conditional means $\mu_2 < \mu_1$ and some $\sigma > 0$. We choose

$$\mu_1 = 1.5, \quad \mu_2 = 0, \quad \text{and} \quad \sigma = 1. \tag{19b}$$

The model is then completely specified by choosing $p_1 = 0.15$ for the training prior probability of class 1. The test prior probability q_1 of class 1 is not fixed as we calculate asymptotic variances of the three above-mentioned prior distribution estimators for the whole range $(0, 1)$ of q_1. The results are shown in Figure 1. □

The following observations can be made from Figure 1:

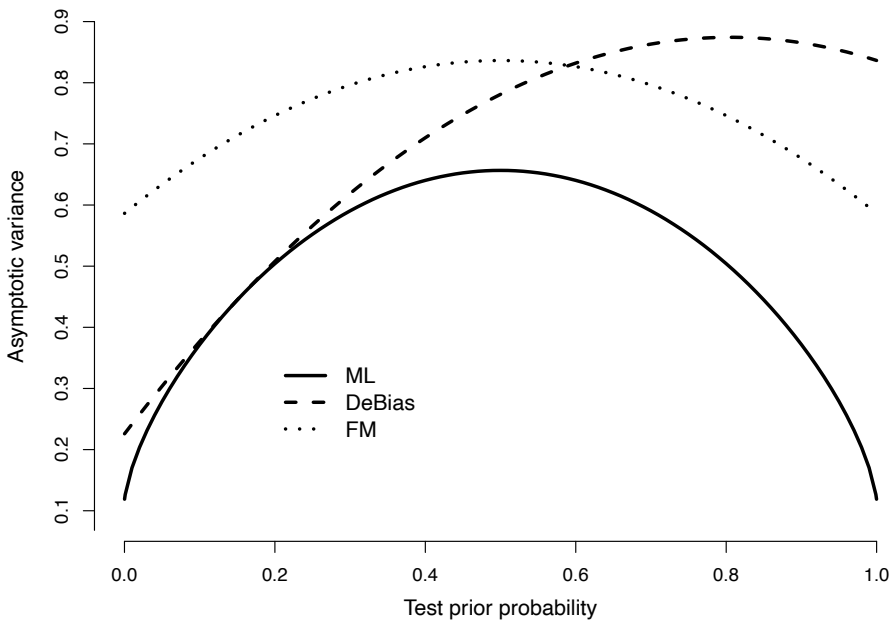

Fig. 1. Asymptotic variances of maximum likelihood estimator, DeBias estimator and Friedman estimator in a binormal model. See Example 1 for the specification of the underlying model.

- The asymptotic variance of the ML estimator is uniformly lower than the asymptotic variances of the other estimators for the whole possible range of the test prior probability of class 1 as is to be expected as a consequence of the Cramér-Rao inequality.
- The asymptotic variance of the Friedman estimator is not uniformly lower than the asymptotic variance of the DeBias estimator and vice versa.
- The DeBias estimator is almost optimal in the vicinity of the training prior probability ($p_1 = 0.15$) of class 1, as a consequence of (16) and (18b).
- In contrast, the asymptotic variance of the DeBias estimator is much larger than the asymptotic variance of the Friedman estimator in the $(0.8, 1)$ range of the test prior probability that is far away from the training prior probability 0.15.

7 Conclusions

We have considered Friedman's [9] method in the context of a general framework for designing linear equation systems for class distribution estimation and compared its binary version with DeBias which is another method proposed by Friedman, and the maximum likelihood estimator. The main findings of this paper are the following:

- The population versions of DeBias and Probability Adjusted Count (PAC, Bella et al. [1]) are identical and the binary special case of a new estimation approach based on inverting the covariance matrix of the training posterior class probabilities (see Section 5.2).
- Although the definition of Friedman's method appears to involve evaluations of the posterior probabilities under the training distribution, the method is potentially less sensitive to inaccuracies of the posterior estimates on smaller training datasets than the maximum likelihood estimator. This is a consequence of the fact that Friedman's methode can be implemented without a need to estimate the training posterior class probabilities (see Section 4).
- As shown in Example 1, Friedman's method may be locally outperformed in terms of asymptotic variance by DeBias. But thanks to its independence of the training prior class probabilities its performance is relatively uniform over the full range of possible values of the test prior probability of the positive class (class 1 in Example 1), in contrast to DeBias' poor performance for test prior probabilities which are very different to the corresponding training prior probability.

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

References

1. Bella, A., Ferri, C., Hernandez-Orallo, J., Ramírez-Quintana, M.: Quantification via probability estimators. In: Data Mining (ICDM), 2010 IEEE International Conference on Data Mining. pp. 737–742. IEEE (2010)
2. Billingsley, P.: Probability and measure. John Wiley & Sons, second edn. (1986)
3. Casella, G., Berger, R.: Statistical Inference. Duxbury Press, second edn. (2002)
4. Castaño, A., Alonso, J., González, P., Pérez, P., del Coz, J.: QuantificationLib: A Python library for quantification and prevalence estimation. SoftwareX **26** (2024). https://doi.org/10.1016/j.softx.2024.101728
5. Donyavi, Z., Serapião, A., Batista, G.: MC-SQ: A Highly Accurate Ensemble for Multi-class Quantification. In: Proceedings of the 2023 SIAM International Conference on Data Mining (SDM). pp. 622–630. SIAM (2023)
6. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to Quantify. Springer Cham (2023). https://doi.org/https://doi.org/10.1007/978-3-031-20467-8
7. Firat, A.: Unified framework for quantification. arXiv preprint arXiv:1606.00868 (2016)
8. Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowledge Discovery **17**(2), 164–206 (2008)
9. Friedman, J.: Class counts in future unlabeled samples. Presentation at MIT CSAIL Big Data Event (2014)
10. González, P., Castaño, A., Chawla, N., Coz, J.D.: A Review on Quantification Learning. ACM Comput. Surv. **50**(5), 74:1–74:40 (2017)
11. Hassan, W., Maletzke, A., Batista, G.: Accurately Quantifying a Billion Instances per Second. In: 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–10 (2020)

12. Hofer, V., Krempl, G.: Drift mining in data: A framework for addressing drift in classification. Computational Statistics & Data Analysis **57**(1), 377–391 (2013)
13. Krempl, G., Hofer, V., Webb, G., Hüllermeier, E.: Beyond Adaptation: Understanding Distributional Changes (Dagstuhl Seminar 20372). Dagstuhl Reports **10**(4), 1–36 (2021). https://doi.org/10.4230/DagRep.10.4.1
14. 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)
15. Moreno-Torres, J., Raeder, T., Alaiz-Rodriguez, R., Chawla, N., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (2012)
16. 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
17. San Martín, E., Quintana, F.: Consistency and identifiability revisited. Brazilian Journal of Probability and Statistics pp. 99–106 (2002)
18. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods. arXiv preprint arXiv:2103.03223 (2021)
19. Serapião, A., Donyavi, Z., Batista, G.: Ensembles of Classifiers and Quantifiers with Data Fusion for Quantification Learning. In: Bifet, A., Lorena, A., Ribeiro, R., Gama, J., Abreu, P. (eds.) Discovery Science. pp. 3–17. Springer Nature Switzerland, Cham (2023)
20. 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)
21. Tasche, D.: Exact fit of simple finite mixture models. Journal of Risk and Financial Management **7**(4), 150–164 (2014)
22. Tasche, D.: Minimising quantifier variance under prior probability shift. In: Cong, G., Ramanath, M. (eds.) Proceedings of the CIKM 2021 Workshops (2021), first International Workshop on Learning to Quantify: Methods and Applications (LQ 2021)
23. Tian, Q., Zhang, X., Zhao, J.: ELSA: Efficient Label Shift Adaptation through the Lens of Semiparametric Models. In: Proceedings of the 40th International Conference on Machine Learning. ICML'23 (2023), https://proceedings.mlr.press/v202/
24. Vaz, A., Izbicki, R., Stern, R.: Prior Shift Using the Ratio Estimator. In: Polpo, A., Stern, J., Louzada, F., Izbicki, R., Takada, H. (eds.) International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering. pp. 25–35. Springer (2017)
25. Vaz, A., Izbicki, R., Stern, R.: Quantification Under Prior Probability Shift: the Ratio Estimator and its Extensions. Journal of Machine Learning Research **20**(79), 1–33 (2019), http://jmlr.org/papers/v20/18-456.html
26. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: Third IEEE International Conference on Data Mining. pp. 435–442 (2003). https://doi.org/10.1109/ICDM.2003.1250950
27. Zhang, K., Schölkopf, B., Muandet, K., Wang, Z.: Domain Adaptation Under Target and Conditional Shift. In: Proceedings of the 30th International Conference on International Conference on Machine Learning – Volume 28. pp. III–819–III–827. ICML'13, JMLR.org (2013)

Quantification Over Time

Feiyu Li, Hassan H. Gharakheili, and Gustavo Batista (✉)

University of New South Wales, Sydney NSW 2052, Australia
{feiyu.li, h.habibi, gbatista}@unsw.edu.au

Abstract. Quantification is the supervised machine learning task that estimates the class distribution in a sample. Therefore, quantification applications typically involve predicting aggregated quantities, such as the prevalence of positive comments about a product, personality or company on a set of social media posts. However, quantification analysis is more informative when performed over time, such as when we are interested in tracking public opinion on social media and relating changes in opinion with relevant events. The vast majority of the literature considers quantification as a standalone task, assuming the output of quantifiers to be independent even when applied to temporal data. This paper proposes a new quantification task, Quantification over Time (QoT), that allies quantification with time series forecasting methods. We propose an approach based on the Kalman filter, which can help improve the performance of standalone quantifications and a general framework that includes both ours and SOTA methods. In an experimental comparison with several textual datasets and numeral datasets, we show that our method outperforms existing methods for QoT in the literature, such as a simple composition of the *classify and count* method with moving averages and *ReadMe2* as a standalone quantifier. We also show that our proposal can outperform several baselines, including recently proposed quantifiers used as standalone approaches. Codes are available at https://github.com/frieli11/quantification-over-time

Keywords: Machine learning · class distribution estimation · quantification · time series · Kalman filter.

1 Introduction

Quantification is the supervised Machine Learning task defined by Forman [9] as the induction of a system "*that takes an unlabeled test set as input and returns its best estimate of the number of cases in each class.*" This definition opposes quantification to classification that focuses on predicting individual instances' labels. Therefore, quantification finds application in areas where we are interested in understanding the behaviour of groups instead of predicting the class of individuals.

The simplest existing quantifier is *Classify and Count* (CC). It consists of the direct application of classifiers to quantification problems. This approach classifies each instance in the unlabeled set and counts the number of instances

predicted in each class. CC is a biased quantifier, as previous research [8,9] showed a systematic error that increases linearly as the unlabeled set class distribution differs from the classifier's training distribution. CC's limitations have led to the proposal of several recent quantifiers that can accurately estimate the class distribution for a wide range of class prevalence.

Quantification papers have accessed their proposals as standalone methods, in which the current decision relies only upon the current data and ignores previous quantifications. The main reason is that the quantification literature has adopted the Artificial Prevalence Protocol (APP) in their experimental setups. The APP uses subsampling on a classification dataset to create many test sets with varying class distributions. However, such experimental protocol does not match how quantifiers are often used in practice. As Forman noticed in his seminal work [9], quantifiers are useful "*to monitor for changes or trends in the class distribution over time.*"

For example, a standalone quantification can estimate a presidential candidate's approval rate based on posts on social media from a certain period, such as the last 24 hours [26]. However, a more fundamental question is how the public support of the candidate varies over the election period, leading to a series of quantifications that are likely to show time dependency. Fig. 1 shows how Biden and Trump supporters' online activity relates to events such as the candidate's debates [1]. Other examples of applications requiring quantification over time are disease-vector mosquito [6] and pollinator surveillance [20] and disease outbreak forecasting [15].

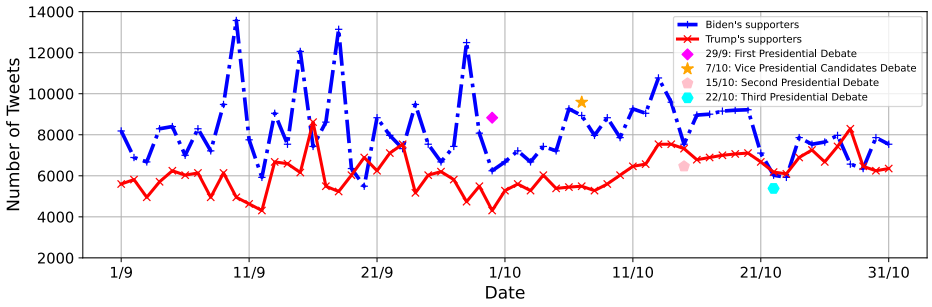

Fig. 1. Prevalence of polarized tweets on different presidential candidates from Sept. 1 to Oct. 31, 2020. Adapted from [1].

This paper introduces *Quantification over Time* (QoT), a quantification task that allies time series forecasting and quantification. In QoT, a series of quantifications give origin to a time series. The objective of QoT is to provide an accurate estimate of the class distribution for an unlabeled set S_t given the data in this set and previous estimates for unlabelled sets $S_1, \ldots, S_{t-1}$.

This paper also proposes KF-MA, which integrates the Kalman filter and moving average, aiming to adjust the result from a standalone quantification for QoT without increasing the requirement for extra labelled data. An algorithm framework is also introduced to conclude how selected time series forecasting methods conduct the adjustment in both KF-MA and state-of-the-art methods. We compare KF-MA with observed methods in literature on several textual and numeral datasets. We demonstrate in our experiments how KF-MA outperforms the state-of-the-art. We evaluate KF-MA in various conditions, including different classifiers and recently proposed quantifiers. The results show that our proposal provides evident improvements in the estimations of standalone quantifications.

This paper is organized into the following sections. Section 2 formalizes the quantification over time task and defines the notation used throughout this paper. Section 3 reviews the relevant literature, including the methods incorporated in our experimental comparison. Section 4 introduces our proposed approach. Section 5 describes the experiment settings and discusses the results. Section 6 provides our concluding remarks and suggests future developments of quantification over time.

2 Definitions, Notation and Background

This section formalizes the task of Quantification over Time (QoT). We start defining the quantification task to highlight the similarities and differences between these two tasks.

2.1 Quantification

Quantification is a supervised machine learning task that predicts the class prevalence (prior probability or relative frequency) in an unlabeled dataset. Like other Machine Learning tasks, quantification requires a training set $D_{train} = \{(\mathbf{x}_i,\ y_i)\}_{i=1}^{N}$ which is a collection with N examples, where each example $\mathbf{x}_i \in \mathcal{X}$ is a vector with M attributes, and $y_i \in \mathcal{L} = \{l_j\}_{j=1}^{L}$ is the class label associated with $\mathbf{x}_i$. The goal of quantification is to learn a function h from D_{train} such that:

$$h : 2^{\mathcal{X}} \rightarrow \Delta^{L} \tag{1}$$

where $2^{\mathcal{X}}$ is the power set of $\mathcal{X}$, *i.e.*, the set with all possible sets of samples with the representation $\mathcal{X}$. Δ^{L} is the L-probability simplex defined as:

$$\Delta^{L} = \{\{p_i\}_{i=1}^{L} | p_i \in [0,1], \sum_{i=1}^{L} p_i = 1\} \tag{2}$$

Given an unlabelled set $S \in 2^{\mathcal{X}}$, h returns an L-dimensional vector $\hat{\mathbf{p}} = [\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_L]^{\top}$, such that $\sum_{i=1}^{L} \hat{p}_i = 1$. The function h is the quantifier, and $\hat{\mathbf{p}}$ is the estimated class prevalence in set S.

2.2 Quantification over Time (QoT)

QoT requires a dataset with timestamps so instances can be grouped in periods of interest, such as hours, days or weeks. For some time t, there is an unlabelled sample S_t and a set of quantified class prevalences of previous samples, S_1 to S_{t-1}, represented as a vector $\hat{\boldsymbol{\pi}}_{t-1} = [\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \ldots, \hat{\mathbf{p}}_{t-1}]$. Each $\hat{\mathbf{p}}_i$ is a vector with the estimated class prevalences for the set S_i. The goal of QoT is to find a function g such that:

$$g : (\hat{\boldsymbol{\pi}}_{t-1}, S_t) \rightarrow \Delta^L \tag{3}$$

where g returns the class distribution estimate for S_t considering previous estimates in $\hat{\boldsymbol{\pi}}_{t-1}$.

2.3 QoT, quantification and time series forecasting

QoT relates to quantification and time series forecasting in the following ways:

- If g only relies on data from S_t, then $g = h$, *i.e.*, the QoT quantifier will ignore the time dependency of the quantifications.
- If g only uses $\hat{\boldsymbol{\pi}}_{t-1}$, then g as a time series forecasting model with the additional constraint that the estimates must be a valid probability distribution.

Therefore, we can understand QoT as a research task at the intersection of time series forecasting and quantification. However, we must emphasize that existing methods from these two areas are inappropriate solutions for the QoT problems since:

1. Time series forecasting models assume a *prediction horizon*, which is the number of time steps ahead that the models should estimate. QoT problems have an *infinite* prediction horizon. For most QoT applications, we should never expect to receive labels for the unlabelled sets S_t. Suppose we use time series forecasting models for QoT problems by inputting forecasts into the model. In that case, we can expect that the error accumulation will make the model forecasts excessively inaccurate after a long period.
2. Quantification methods ignore the time dependency of the estimates. The incorporation of time information provides the model with an expected rate of change. This work hypothesizes that incorporating historical information with the current quantification provides more accurate estimates.

To conclude this section, we provide additional details about how most QoT applications work in practice regarding label availability. We use the social media sentiment analysis from Section 1 as an example.

For most QoT applications, we should *not* expect to see class labels after the system deployment. In the case of social media, we should not expect humans to label any posts manually during the system's operation. However, we often have a labelled dataset to train and tune the model's hyperparameters. We

simulate this condition in our experiments by using the initial samples in the data stream as a training/validation set. This situation differs significantly from the typical operation of time series forecasting models. In regression applications, it is common for the target information to auto-reveal after the forecast horizon has passed. For instance, if we predict stock prices over 24 hours, we will know the actual value of the stocks after this period. This allows us to feed these actual prices to the model when we need to provide a forecast for the next horizon.

The lack of labels after model deployment gives the impression that the QoT task will accumulate errors, providing inaccurate predictions for longer forecast windows, as would happen if we fed time series forecasting methods with estimates instead of actual values. However, QoT inherits a pivotal assumption from quantification: $P_{te}(\mathbf{X}|Y) = P_{tr}(\mathbf{X}|Y)$, *i.e.*, the conditional distribution of the features given the class remains constant from training to test.

This assumption is reasonable for most applications. In the sentiment analysis example, $P(\mathbf{X}|Y)$ captures the relationship between words and sentiments and is mostly constant with small changes in the long term as the language evolves. Under this assumption, the quantifiers can provide relatively accurate prevalence predictions that will not allow the time series forecaster to drift away from reality. At the same time, the time series forecaster will provide temporal dependency to the quantifier, allowing it to improve its predictions.

3 Related Work

Due to the popularity of sentiment analysis for social media and the fact that social media posts are timestamped, plenty of research papers quantify sentiment over time. However, these papers fall into two categories: The first uses the classify and count quantifier without temporal information, and the second uses the same quantifier with a moving average to smooth the predicted prevalences.

Before reviewing the relevant literature, we should understand why the CC quantifier is a suboptimal choice despite its popularity in the sentiment analysis community [18]. Classify and count is a biased quantifier because the classifier assumes that the training and test samples come from the same underlying distribution. However, in quantification problems, the class distribution of the test samples can differ significantly from the test distribution.

Let us suppose that a binary-class classifier is trained with a balanced distribution. In this case, when the positive class prevalence increases, the CC quantifier tends to underestimate the prevalence of the positives. Essentially, the classifier expects the test samples to have the same class distribution as the training samples and thus pushes the class estimates towards a 50%/50% estimate. Similarly, when the prevalence of positives decreases, the same quantifier tends to overestimate its prevalence.

Forman [8,9] provides a formal analysis of the CC quantifier error and shows that it increases linearly as the test class distribution moves away from the training distribution. The classifier error, specifically the difference between the true positive and negative rate, defines the error slope.

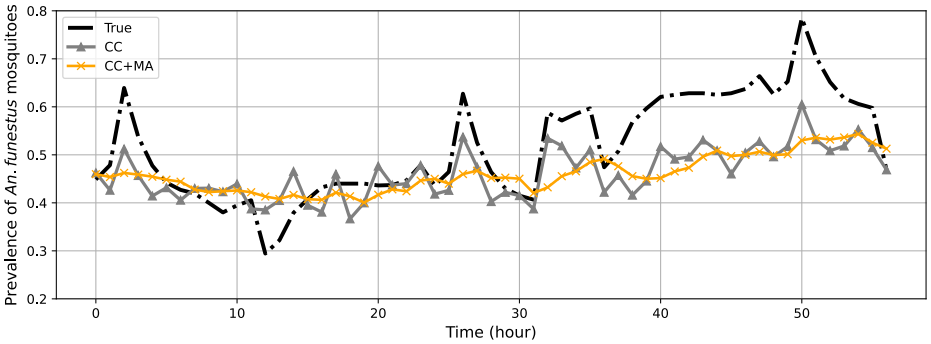

Fig. 2. A comparison of the performance of the classify and count (CC) quantifier with and without moving average (MA). The classifier was trained with a balanced training set with signals from female *Anopheles funestus* and *Culex quinquefasciatus* mosquitoes. CC underestimates high prevalences and overestimates low prevalences. CC+MA performs even worse as the moving average operates as a low-pass filter.

Fig. 2 illustrates this issue in a mosquito surveillance dataset. The data represents the number of *Anopheles funestus* mosquitoes captured by a mosquito trap that uses Machine Learning classifiers to recognize the mosquito species based on data collected from their wing movement. We trained a Machine Learning classifier with a balanced distribution of mosquitoes, which achieved a respectful 80% accuracy. When test class distribution increases, the CC quantifier underestimates the species prevalence. Such underestimation is proportional to the species prevalence and achieves its highest value for the peak at hour 50. The opposite occurs when class prevalence decreases, as we can observe at hour 12.

An even worse behavior occurs when we apply a moving average of window size of 4 hours to the classify and count output. As we will see later in this section, this is a popular approach in the sentiment analysis community. A moving average operates as a low-pass filter, providing a smoother sequence of predictions but worsening the under/over-estimation issue.

In the next two sections, we briefly summarize the relevant literature. Section 3.1 focuses on papers that approach the QoT problem using quantifiers only without incorporating time information. Section 3.2 summarizes the related work that combines the classify and count quantifier with a moving average.

3.1 QoT with quantification only

The literature contains various studies quantifying sentiments on social platforms over time and exploring the relations between public opinions online and real-world events. The vast majority of the relevant papers use the classify and count quantifier. We believe such a poor design decision can only be explained by a lack of understanding of the classification and count quantifier's limitations in the sentiment analysis community.

In what follows, we provide some examples of sentiment analysis papers that use the classify and count quantifier to solve QoT problems. Bollen et al. [2] investigate whether measurements of collective mood states derived from large-scale Twitter feeds correlate to the value of the Dow Jones Industrial Average (DJIA) over time. Borge-Holthoefer et al. [3] track the public opinion dynamics about political events in Egypt on Twitter and analyze the motivation of people switching political polarization. They believe that tracking the relevant change of proportions of the sentiment index instead of absolute values might diminish the quantification bias. Lamsal [15] analyzes tweets about COVID-19, aiming to understand the public opinion patterns related to the ongoing pandemic. Liu et al. [16] also study the aggregate sentiments on Twitter over time based on an algorithm related to CC to predict the presidential election. Notably, the authors claim their trained classifier with an evaluated accuracy of 57%, as it should be predictably biased when deploying classify and count.

Few papers use a quantifier other than classify and count in QoT applications. Hopkins and King [10] analyze the impact of political speech incidents on blog sentiments toward candidates. Aware of the bias of the classify and count quantifier, they propose *ReadMe* to quantify sentiments over time. Similarly, Ceron et al. [4] conducted sentiment analysis on tweets to estimate the distribution flow of citizens' political preferences. They also use ReadMe, applying this quantifier across time while disregarding eventual time dependency among forecasted prevalences.

3.2 QoT with classify and count and moving average

There is abundant literature in sentiment analysis that uses the classify and count quantifier with moving averages. The moving average can be seen as a basic time series analysis method that incorporates time dependency to smooth prevalence forecasts. Overall, the papers summarized in this section employ moving averages to improve the visualization of trends.

O'Connor et al. [19] argue social media sentiment can serve a role similar to that of traditional polling and surveys. The authors acknowledge the impact of misclassification on the estimates generated by the classify and count quantifier. However, they think the occurrence of false positives and negatives would balance each other out when aggregating the sentiment. Wen et al. [25] use a 3-day sliding window alongside the CC quantifier on sentiment polarity analysis to understand students' opinions towards the course and course tools. They also believe that false positives and negatives could potentially offset each other during the counting of classified instances.

Both O'Connor et al. [19] and Wen et al. [25] make the incorrect assumption that the classify and count is potentially an accurate quantifier as the errors nullifying each other. In fact, if a binary classifier makes the same number of false positive and negative errors, the quantification can be flawless despite the imperfection of the classifier [9]. However, as the test class distribution changes, one type of error may prevail. For instance, a classifier may make the equal number of false positive and negative misclassifications in a balanced training

set. However, if the prevalence of positive instances increases to 90% in a test sample, then the number of false negative misclassifications will surpass the false positives, causing the classifier to underestimate the positives, as shown in Fig. 2.

In addition, Lai [14] correlates the sentiment trend on Twitter with a traditional presidential performance poll. The author applies a moving average to the survey data, which the paper regards as a gold standard for the sentiment trend. Rani and Kumar [21] focus on sentiment analysis within teaching research. They propose a system that analyzes student feedback sourced from online platforms and course surveys. A method similar to moving average, called a mean emotion vector, is used to smooth the quantified results.

4 Methodology

This section presents our proposed method, Kalman Filter-Moving Average (KF-MA), for QoT problems. KF-MA is part of a more general framework that we name the *adjustment framework*. Such a framework also includes approaches based on the moving average, popularly employed in the sentiment analysis literature. We first introduce the framework in Section 4.1 and then introduce KF-MA in Section 4.2.

4.1 Adjustment Framework

Our framework adjusts class prevalence estimates for a sample S_t collected at a time point t by leveraging existing quantifiers and the temporal dependency of previous prevalence estimates.

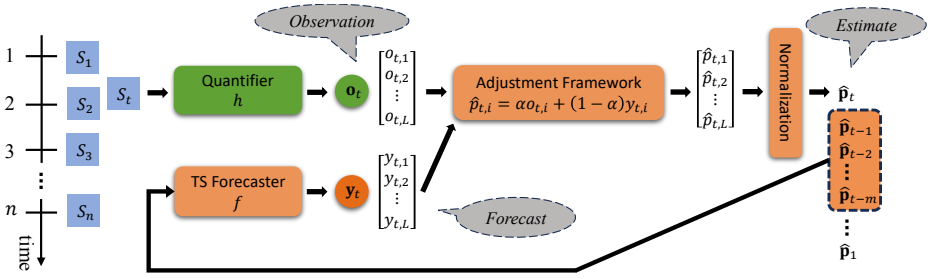

Fig. 3. A general view of the adjustment framework for QoT, which integrates predictions from a quantifier and a time series forecaster, having inputs from last m estimates.

We use the standard nomenclature of the Probabilistic Graphical Model literature and name the output of quantifiers as *observations*. At time point t, we have an observation $\mathbf{o}_t = [o_{t,1}, o_{t,2}, \cdots, o_{t,L}]^\top = h(S_t)$ from a quantifier h, being $o_{t,i}$ the observation corresponding to the estimate of class l_i's true prevalence

$p_{t,i}$. Besides, we have a *forecast* $\mathbf{y}_t = [y_{t,1}, y_{t,2}, \cdots, y_{t,L}]^\top$, where $y_{t,i} = f(\hat{\boldsymbol{\pi}}_{t-1})$ is the prevalence prediction for class l_i We integrate these two predictions into a single value

$$\hat{p}_{t,i} = \alpha\, o_{t,i} + (1-\alpha)\, f(\hat{\boldsymbol{\pi}}_{t-1}) \tag{4}$$

as the estimate of the true prevalence. For time series forecaster f, a model that does not require a large-scale training process is preferred. The reason is that deploying non-trivial time series analysis methods, such as recurrent neuron networks and autoregression models, requires high cost of data, while manually labelling enough data for training those methods is cumbersome in QoT tasks as time series in QoT is different from most that out of streaming data, each data point of time series in QoT is an aggregated mark over a sample instead of an instance.

$\hat{\boldsymbol{\pi}}_{t-1}$ is generallized as a representation of historical information. It can be set to storing either the previous integrated estimates $\hat{\boldsymbol{p}}$ or previous observation $\mathbf{o}$, which can be found in the literature and will be discussed in the following section. In addition, this historical information may encompasses only the last m observations, where m is a hyperparameter that limits how long the method can observe from the past.

The $\hat{p}_{t,i}$ are not necessarily normalized in the sense that $\sum_i^L \hat{p}_{t,i} \neq 1$. We thus normalize these predictions to turn them into a probability distribution. Fig. 3 illustrates this general approach that we named the *adjustment framework*.

In the remainder of this text, we simplify the notation by dropping the i index from symbols such as $p_{t,i}$, $o_{t,i}$, and $y_{t,i}$. The reason is that the methods in the adjustment framework often work with one class at a time. Therefore, for a multi-class problem, these methods make $L-1$ independent predictions integrated in the normalization step.

In the remainder of this section, we discuss how the moving average (MA) approach integrates into our framework. We will use it later as a baseline for comparison with our proposed methods. O'Connor et al. [19] was the first to apply MA to QoT problems. They started with daily sentiment ratios $[o_1, o_2, \ldots, o_n]$ of positive and negative tweets obtained with the Classify and Count method (CC). The authors adjusted each ratio using a m-size window as follows:

$$\begin{aligned}\hat{p}_t &= \frac{1}{m}(o_{t-m+1} + o_{t-m+2} + \ldots + o_t) \\ &= \frac{m-1}{m}\frac{(o_{t-m+1} + o_{t-m+2} + \ldots + o_{t-1})}{m-1} + \frac{1}{m}o_t\end{aligned} \tag{5}$$

Notice that, as they work with binary-class problems, they can track only the positive class prevalence and estimate the negative prevalence, as both need to sum to one.

According to Eq. (4), we have $y_t = f(\mathbf{o}_t^m) = \frac{(o_{t-m+1}+o_{t-m+2}+\ldots+o_{t-1})}{m-1}$ and $\alpha = \frac{1}{m}$. Note that, for MA, with a fixed window size m, the weights of prediction and observation are static, and it primarily presents a retrospective average without effectively capturing trends.

4.2 KF-MA

Our proposal, Kalman Filter-Moving Average (KF-MA), introduces a dynamic weighting approach for parameter α between observations and predictions. KF-MA uses moving average as a time series forecasting function f and Kalman filter [13] to the adjustment framework.

Kalman filter [13] is a recursive algorithm for estimating the state of a dynamic system. It assumes all uncertainties from the environment, observation and hidden states are Gaussian distributed.

When using a Kalman filter to estimate a dynamic system with no known external influence, where the scales of measurement and states are identical, for state s_t, there is an observation $O_t \sim \mathcal{N}(\mu_{o_t}, R)$, where R is from the i.i.d.[1] random error of each measurement. A Kalman filter also models a forecast as $X'_t \sim \mathcal{N}(\mu_{x'_t}, \Sigma_{x'_t})$, such that:

$$\begin{cases} \mu_{x'_t} = F\,\mu_{x_{t-1}} \\ \Sigma_{x'_t} = F\,\Sigma_{x_{t-1}}\,F^\top + Q_t \end{cases} \tag{6}$$

where $X_{t-1} \sim \mathcal{N}(\mu_{x_{t-1}}, \Sigma_{x_{t-1}})$ is an estimate of s_{t-1}, and F is the matrix encapsulating the internal transition mechanism of the dynamic system. The transition covariance Q_t represents the uncertain influence of the external environment.

The filter integrates X'_t with O_t to have an estimate $X_t \sim \mathcal{N}(\mu_{x_t}, \Sigma_{x_t})$ for s_t. It is computed as follows:

$$\begin{cases} \mu_{x_t} = \mu_{x'_t} + K_t\,(\mu_{o_t} - \mu_{x'_t}) \\ \Sigma_{x_t} = (1 - K_t)\,\Sigma_{x'_t} \\ K_t = \frac{\Sigma_{x'_t}}{\Sigma_{x'_t} + R} \end{cases} \tag{7}$$

where K_t is named Kalman gain.

For a later state s_{t+1}, the filter recursively derives a prediction using estimate X_t of s_t. Only the estimate of the initial state should be specified manually since no prior estimates are made. To implement the Kalman filter, specifications for R, F, Q_t, and an initial estimate $X_0 \sim \mathcal{N}(\mu_{x_0}, \Sigma_{x_0})$ are required.

Since the forecast of the current state depends solely on the previous state, it is necessary to reformulate the state to integrate the Kalman filter into the adjustment framework. In KF-MA, for each class at time t, the forecast y_t of true prevalence p_t is modeled as:

$$y_t = f(\hat{\mathbf{p}}_t^m) = \mathbf{a}\,\hat{\mathbf{p}}_t^m \tag{8}$$

where the vector $\mathbf{a} = [a_1, a_2, \cdots, a_m]$, $\hat{\mathbf{p}}_t^m = [\hat{p}_{t-m}, \hat{p}_{t-m+1}, \cdots, \hat{p}_{t-1}]^\top$. For time t, define the state as a $m \times 1$ vector $\mathbf{p}_t^* = [p_{t-m+1}, \cdots, p_{t-1}, p_t]^\top$, hence we have the estimate $X_t \sim \mathcal{N}(\hat{\mathbf{p}}_t^*, P_t)$, in which $\hat{\mathbf{p}}_t^* = [\hat{p}_{t-m+1}, \cdots, \hat{p}_{t-1}, \hat{p}_t]^\top$, P_t is the $m \times m$ covariance matrix and $\hat{\mathbf{p}}_t^m = \hat{\mathbf{p}}_{t-1}^*$. Correspondingly, we have the forecast $X'_t \sim \mathcal{N}(\mathbf{y}_t^*, P'_t)$ and the observation $O_t \sim \mathcal{N}(\mathbf{o}_t^*, R)$. Therefore, we set the $m \times m$ transition matrix F as:

[1] Independent and identically distributed.

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_1(m) & a_2(m) & a_3(m) & \cdots & a_m(m) \end{bmatrix} \tag{9}$$

s.t.

$$\mathbf{y}_t^* = F\,\hat{\mathbf{p}}_{t-1}^* \tag{10}$$

Now Eq. (8) is transformed into Eq. (10). KF-MA uses MA for time series forecasting f, hence $a_j(m) = \frac{1}{m}$ for each in F. R is modeled by the mean squared error r of the quantifier through validation multiplied by an $m \times m$ identity matrix I_m as $R = r \cdot I_m$. Since no former state can be used to update the initial state covariance, it is initiated by observations and R. Set the time index starting from 1, the initial state $\mathbf{p}_t^* = [p_1, \cdots, p_{m-1}, p_m]^\top$. The initial estimate is $\mathcal{N}(\mathbf{o}_m^*, R)$. Similar to other time series forecast algorithms, we cannot estimate the initial m class prevalence. Hence, the window size m is expected to be set small, as detailed in Section 5. Assuming a stable outside influence over time, we denote the transition covariance as Q, which is fine-tuned through the validation process. Recalling the Eq. 7 and Eq. (4), we can now obtain the estimate $X_t \sim \mathcal{N}(\hat{\mathbf{p}}_t^*, P_t)$ by

$$\begin{cases} \hat{\mathbf{p}}_t^* = K_t\,\mathbf{o}_t^* + (1 - K_t)\,\mathbf{y}_t^* \\ P_t = (1 - K_t)\,P_t' \\ K_t = \frac{P_t'}{P_t' + R} \end{cases} \tag{11}$$

in which $\alpha = K_t$. For each class at time t, value $\hat{p}_t = \hat{\mathbf{p}}_t^*[m]$ is the estimate of its true prevalence. In comparing KF-MA with the moving average, one notable advantage of KF-MA is its dynamic weights for observations and predictions. This advantage stems from the recursive updating of the Kalman gain as the states change over time.

5 Experimental Evaluation

This section outlines the experimental evaluation settings and discusses the results, focusing on how KF-MA improves quantification accuracy and whether it outperforms MA. Additionally, we compare KF-MA to the state-of-the-art QoT methods in the literature.

5.1 Experimental Setup

Implementing a QoT approach involves four main components: a classifier, a quantifier, a time series forecaster and an adjustment framework. Most existing quantifiers produce a class prevalence estimate using the scores generated by a

classifier. The time series forecaster and adjustment framework adjust the output of the quantifier, improving its performance in the presence of historical data. To effectively evaluate our method, we created a diverse experiment involving multiple datasets, classifiers, and quantifiers. The objective is to introduce variations that enable a comprehensive comparison and simulate scenarios in which users might apply the method across various applications.

Datasets The requirements for datasets include true or hand-coded labels, timestamp features, and a task that involves counting a time-related topic or entity. We selected seven datasets meeting the criteria, three of which are textual data for sentiment classification:

NpSenti Sentiment analysis of COVID-19-related Tweets in Nepali [24].
AppleSenti User sentiments towards Apple company on Twitter[2].
GlobalSenti A collection of worldwide Tweets related to COVID-19[3].

The rationale for choosing these datasets is they represent applications similar to those discussed in Section 3, specifically in monitoring opinions of particular topics. In addition to textual datasets, we collected four other datasets for various applications:

Mosquito A mosquito surveillance dataset to count mosquitoes by species.
Bike An UCI dataset for bike sharing prediction[4].
Energy An UCI dataset for predicting energy consumption[5].
News An UCI dataset for predicting online news popularity[6].

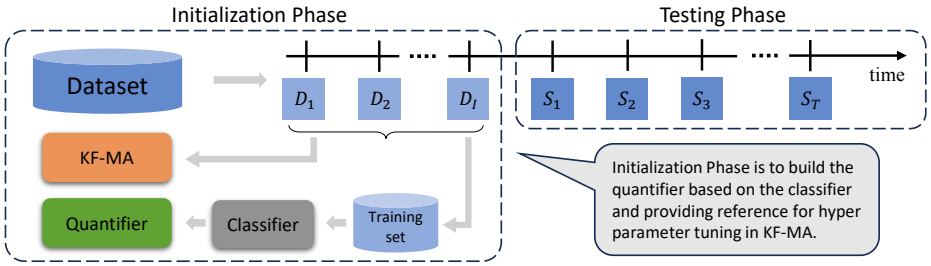

Fig. 4. Deployment of quantification over time with KF-MA

[2] https://data.world/crowdflower/apple-twitter-sentiment
[3] https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification/data
[4] https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset
[5] https://archive.ics.uci.edu/dataset/374/appliances+energy+prediction
[6] https://archive.ics.uci.edu/dataset/332/online+news+popularity

The datasets are preprocessed to align with our evaluation criteria. For each dataset, instances are assigned to time intervals, such as hours or days, and then grouped into subsamples based on these intervals.

An initialization process is necessary for training classifiers and quantifiers when not using pre-trained ones or searching for KF-MA hyperparameters. This phase uses the first subsamples, as illustrated in Fig. 4. This data serves two purposes:

1. It is used to train classifiers and quantifiers when not using pre-trained models and evaluate their performance.
2. It estimates the random error matrix, R, for the Kalman Filter used in KF-MA. Such a matrix estimation is obtained from the quantifier's mean squared error obtained in the initialization data.

The number of samples in the initialization phase, I, must follow the requirements found in practical applications. Large values of I allow us to train better classifiers and quantifiers and estimate hyperparameters better. However, large training sets are unavailable for many applications due to the high cost of labelling data. In our experiments, we used around 15% of the data for initialization of the non-textual datasets. We used the first 15 time units for the textual datasets, such as hours or days. After setting I, T samples remain for testing. The final models are trained in the concatenation of all $\{D_i\}_{i=1}^{I}$ as a single training set. Table 1 summarises the parameters of our experimental setup for each dataset.

Table 1. Summary of the experimental setup. #Classes is the number of classes. *Tr. Size* is the number of instances in the training set after concatenation. I is the number of time units in the training set. *Avg. Size/t* is the average number of instances per sample in the testing sets. T is the number of time units in the training set.

Dataset	#Classes	Tr. Size	I	Avg. Size/t	T	Time Unit
NpSenti	3	233	15	104	320	day
AppleSenti	3	1927	15	104	18	day
GlobalSenti	3	4454	15	1397	29	day
Mosquito	2	958	24	334	312	hour
Bike	2	2634	55	47	273	hour
Energy	3	2880	20	144	103	day
News	2	6943	36	155	185	day

Classifiers and quantifiers We chose to use two pre-trained sentiment classifiers on the three textual datasets: *VADER* [11] a lexicon-based sentiment analyzer, and AutoNLP from HuggingFace[7], referred as *Solanki*. We used Logistic

[7] https://huggingface.co/amansolanki/autonlp-Tweet-Sentiment-Extraction-20114061

Regression and Random Forest classifiers for the other four datasets with default hyperparameters.

For the quantifiers, we selected *Adjusted Classify and Count* (ACC) [9], *Distribution y-Similarity* (DyS) [17], *Generalized Probabilistic ACC* (GPACC) [7], and *Energy Distance-y* (EDy) [5]. These are popular quantifiers often ranked among the best performing according to recent empirical studies [22].

ACC adjusts the output of CC using the *true positive* (*tpr*) and *false positive rates* (*fpr*) through a validation process on the training set. DyS models the scores provided by a classifier using histograms. The model searches for a parameter that minimizes the distance between a mixture of positive and negative scores from the training set and the unlabelled scores from the test sample. GPACC is a generalization of ACC for multiclass problems with probabilistic classifiers. It uses a soft variation of the confusion matrix obtained from the training set using cross-validation. EDy interprets the dimensional density of the data as the posterior distribution. This approach allows EDy to extract more detailed information than GPACC in feature space.

CC, GPACC, and EDy are naturally capable of quantification on multi-class data, while ACC and DyS are designed for binary data. Therefore, ACC and DyS are deployed using *One versus All* (OVA) for multi-class datasets.

Hyperparameters One hyperparameter is the window size m for the time series forecasting model f. In our experiments, we set $m = 4$, taking into account the following factors:

(i) Larger window sizes are wasteful because the recommended model f does not capture the seasonality patterns; (ii) Covariance decreases as time points move further apart; and, (iii) Minimizing the number of data points that can not be adjusted in the initial phase.

In addition to m, the transition matrix Q in the Kalman filter is tuned during the initialization phase in Fig. 4. Matrix Q is assumed to be a scalar matrix, similar to the observation covariance R. In our experiments, we perform a ternary search on Q within the interval $\left[10^{-4}, 10^{-1}\right]$.

Evaluation Our evaluation uses the *Absolute Error* (AE) due its interpretability [23]:

$$\mathrm{AE}(\mathbf{p} - \hat{\mathbf{p}}) = \frac{1}{L} \sum_{i=1}^{L} |\mathbf{p}(i) - \hat{\mathbf{p}}(i)| \tag{12}$$

where $\mathbf{p}$ is the true and $\hat{\mathbf{p}}$ the predicted class prevalence. We compute each sample's AE and take the mean across all samples, denoted as *Mean Absolute Error* (MAE).

With seven datasets, four quantification methods, and two classifiers for each dataset, we create 56 experimental conditions. Each condition undergoes ten iterations with different seeds, and the results are averaged to ensure the stability and reliability of the findings.

Table 2. MAE results on textual data for different combinations of datasets, classifiers and quantifiers.

Quantifier	DyS		ACC		GPACC		EDy	
Classifier	VADER	Solanki	VADER	Solanki	VADER	Solanki	VADER	Solanki
Dataset	GlobalSenti							
QFY	0.0182	0.0590	0.0095	0.0640	0.0310	0.0730	0.0236	0.0620
MA	0.0300	**0.0465**	0.0280	0.0489	0.0359	0.0637	0.0325	0.0530
KF-MA	**0.0156**	0.0498	**0.0087**	**0.0483**	**0.0280**	**0.0609**	**0.0208**	**0.0502**
Best Method	KF-MA	MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA
Dataset	NpSenti							
QFY	0.1858	0.1712	0.1823	0.1828	0.2075	0.2529	0.1921	0.2575
MA	**0.1350**	**0.1617**	0.1505	**0.1546**	0.1672	**0.2434**	0.1549	**0.2486**
KF-MA	0.1354	0.1644	**0.1473**	0.1609	**0.1640**	0.2442	**0.1527**	0.2499
Best Method	MA	MA	KF-MA	MA	KF-MA	MA	KF-MA	MA
Dataset	AppleSenti							
QFY	0.1706	0.1289	0.1968	0.1187	0.1518	0.1134	0.1538	0.1146
MA	0.1215	0.0999	0.1688	0.0999	0.1334	0.1018	0.1340	0.1032
KF-MA	**0.1128**	**0.0929**	**0.1670**	**0.0935**	**0.1254**	**0.0975**	**0.1269**	**0.0986**
Best Method	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA

5.2 Comparison with QoT approaches in literature

We compare KF-MA to other approaches previously used in related work. From our knowledge, only three methods have been used for QoT: standalone CC, CC with moving average, and standalone ReadMe [10]. ReadMe is a quantification method specifically designed for textual data. It does not require a trained classifier as it operates directly on text features. Jerzak et al. [12] recently proposed an improved version of ReadMe, referred to as *ReadMe2*, which has shown competitive performance in sentiment quantification tasks. We assess ReadMe2 combined with our proposed method KF-MA, comparing it with CC, CC with moving average, and standalone Readme2 on the three textual datasets. Other datasets are not included in this experiment as they do not have the textual data expected by ReadMe2. *Solanki* classifier was used in the CC approach. This experiment evaluates if KF-MA improves upon the state-of-the-art approaches.

5.3 Results

We evaluate three QoT methods across 56 experimental conditions: standalone quantification (QFY), quantification with moving average (MA), and quantification with KF-MA (KF-MA). According to the results presented in Table 2 for textual data and Table 3 for numeral data, our proposed method KF-MA achieved the best performance in most conditions.

Table 3. MAE results on non-textual datasets for different combinations of datasets, classifiers and quantifiers. *LR* represents Logistic Regression classifier and *RF* represents Random Forest classifier.

Quantifier	DyS		ACC		GPACC		EDy	
Classifier	LR	RF	LR	RF	LR	RF	LR	RF
Dataset	Bike							
QFY	0.2113	0.2622	0.1780	0.2443	0.1982	0.2648	0.1827	0.2604
MA	**0.1916**	0.2054	0.1671	0.1886	**0.1858**	0.2164	0.1726	0.2111
KF-MA	0.1918	**0.1971**	**0.1667**	**0.1840**	0.1885	**0.2112**	**0.1725**	**0.2049**
Best Method	MA	KF-MA	KF-MA	KF-MA	MA	KF-MA	KF-MA	KF-MA
Dataset	Energy							
QFY	0.2813	0.2575	0.3294	0.3179	0.3442	0.4378	0.3281	0.4305
MA	0.2285	0.2259	0.3136	0.2531	0.2867	0.3906	0.2769	0.3788
KF-MA	**0.2089**	**0.1858**	**0.3041**	**0.1862**	**0.2611**	**0.3533**	**0.2514**	**0.3297**
Best Method	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA
Dataset	News							
QFY	0.2143	0.2282	0.3253	0.2248	0.2243	0.1981	0.2175	0.2039
MA	0.1694	0.1951	0.2208	0.2177	0.1803	0.1732	0.1735	0.1779
KF-MA	**0.1518**	**0.1772**	**0.0959**	**0.2162**	**0.1597**	**0.1638**	**0.1534**	**0.1680**
Best Method	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA
Dataset	Mosquito							
QFY	0.0399	0.0191	0.0468	0.0223	0.0403	0.0182	0.0404	0.0187
MA	0.0445	0.0411	0.0459	0.0420	0.0440	0.0404	0.0440	0.0406
KF-MA	**0.0327**	**0.0190**	**0.0348**	**0.0219**	**0.0315**	**0.0178**	**0.0316**	**0.0184**
Best Method	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA

MA won 8 out of 56, while QFY did not win any. Standalone quantification methods exhibit varying performances across the seven datasets. However, the MAE results demonstrate that KF-MA consistently improves the quantification accuracy of the standalone quantifiers.

Although not as effective as KF-MA, the moving average can improve the quantification accuracy over the standalone quantifiers in certain cases. However, for datasets such as *GlobalSenti* and *Mosquito*, in which standalone quantifiers underperform, applying a moving average tends to worsen the estimation due to its nature as a low-pass filter. In contrast, KF-MA provides adaptive filtering capabilities, which are particularly beneficial when observations are unbiased.

Table 4 shows that KF-MA improves the performance of the state-of-the-art approach ReadMe2. The quantification results of ReadMe2 with KF-MA on the three textual datasets show a consistent improvement over all methods previously applied in the literature.

Table 4. MAE results of different QoT methods on three textual datasets.

QoT Method	CC	CC+MA	ReadMe2	ReadMe2+KF-MA
NpSenti	0.2436	0.2465	0.1422	**0.1420**
GlobalSenti	0.2439	0.2264	0.0766	**0.0594**
AppleSenti	0.2000	0.1631	0.1182	**0.1148**
Mean	0.2292	0.2120	0.1123	**0.1054**

6 Conclusion

In this paper, we introduced the task of quantification over time (QoT), which is common in various fields, and proposed a method called the Kalman Filter-Moving Average (KF-MA) approach. Additionally, we propose a framework that accommodates both KF-MA and MA approaches. MA is a popular approach in the literature for adjusting the output of standalone quantifiers in QoT.

We evaluated our method through experiments under various combinations of datasets, quantifiers and classifiers, comparing them with state-of-the-art approaches. The results demonstrate that using time dependency enhances the performance of quantifiers in QoT problems. Our work provides practitioners with an accurate tool and offers fundamental guidelines and ideas to researchers interested in developing novel algorithms targeted at quantification over time.

In future work, we intend to develop and integrate more sophisticated approaches for time series forecasting that can learn from small quantities of data. One potential approach is the use of Gaussian Processes. We also intend to use multidimensional time series forecasters, using the dependencies among the class's prevalence to improve our results.

References

1. Belcastro, L., Branda, F., Cantini, R., Marozzo, F., Talia, D., Trunfio, P.: Analyzing voter behavior on social media during the 2020 us presidential election campaign. Social Network Analysis and Mining **12**(1), 83 (2022)
2. Bollen, J., Mao, H., Zeng, X.: Twitter mood predicts the stock market. Journal of computational science **2**(1), 1–8 (2011)
3. Borge-Holthoefer, J., Magdy, W., Darwish, K., Weber, I.: Content and network dynamics behind Egyptian political polarization on Twitter. In: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. pp. 700–711 (2015)
4. Ceron, A., Curini, L., Iacus, S.M., Porro, G.: Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. New media & society **16**(2), 340–358 (2014)
5. del Coz, J.J.: Unioviedo (team2) at Lequa 2022: comparison of traditional quantifiers and a new method based on energy distance. In: Working Notes of the 2022 Conference and Labs of the Evaluation Forum (CLEF 2022), Bologna, IT (2022)

6. De Nadai, B., Maletzke, A., Corbi, J., Batista, G., Reiskind, M.: The impact of body size on Aedes [stegomyia] aegypti wingbeat frequency: implications for mosquito identification. Medical and Veterinary Entomology **35**(4), 617–624 (2021)
7. Firat, A.: Unified framework for quantification. arXiv preprint arXiv:1606.00868 (2016)
8. Forman, G.: Counting positives accurately despite inaccurate classification. In: European conference on machine learning. pp. 564–575. Springer (2005)
9. Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowledge Discovery **17**, 164–206 (2008)
10. 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)
11. Hutto, C., Gilbert, E.: Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Proceedings of the international AAAI conference on web and social media. vol. 8, pp. 216–225 (2014)
12. Jerzak, C.T., King, G., Strezhnev, A.: An improved method of automated nonparametric content analysis for social science. Political Analysis **31**(1), 42–58 (2023)
13. Kalman, R.E.: A new approach to linear filtering and prediction problems (1960)
14. Lai, P.: Extracting strong sentiment trends from Twitter. https://nlp.stanford.edu/courses/cs224n/2011/reports/patlai.pdf (2010)
15. Lamsal, R.: Design and analysis of a large-scale COVID-19 tweets dataset. applied intelligence **51**, 2790–2804 (2021)
16. Liu, R., Yao, X., Guo, C., Wei, X.: Can we forecast presidential election using Twitter data? an integrative modelling approach. Annals of GIS **27**(1), 43–56 (2021)
17. Maletzke, A., dos Reis, D., Cherman, E., Batista, G.: DyS: A framework for mixture models in quantification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4552–4560 (2019)
18. Moreo, A., Sebastiani, F.: Tweet sentiment quantification: An experimental re-evaluation. Plos one **17**(9), e0263449 (2022)
19. O'Connor, B., Balasubramanyan, R., Routledge, B., Smith, N.: From tweets to polls: Linking text sentiment to public opinion time series. In: Proceedings of the international AAAI conference on web and social media. vol. 4, pp. 122–129 (2010)
20. Parmezan, A.R., Souza, V.M., Seth, A., Žliobaitė, I., Batista, G.E.: Hierarchical classification of pollinating flying insects under changing environments. Ecological Informatics **70**, 101751 (2022)
21. Rani, S., Kumar, P.: A sentiment analysis system to improve teaching and learning. Computer **50**(5), 36–43 (2017)
22. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods. arXiv preprint arXiv:2103.03223 (2021)
23. Sebastiani, F.: Evaluation measures for quantification: An axiomatic approach. Information Retrieval Journal **23**(3), 255–288 (2020)
24. Sitaula, C., Basnet, A., Mainali, A., Shahi, T.B., et al.: Deep learning-based methods for sentiment analysis on Nepali COVID-19-related tweets. Computational Intelligence and Neuroscience **2021** (2021)
25. Wen, M., Yang, D., Rose, C.: Sentiment analysis in MOOC discussion forums: What does it tell us? In: Educational data mining 2014. Citeseer (2014)
26. Yaqub, U., Chun, S.A., Atluri, V., Vaidya, J.: Analysis of political discourse on Twitter in the context of the 2016 us presidential elections. Government Information Quarterly **34**(4), 613–626 (2017)

Enhancing Quantification through Meta-Learning

Guilherme B. Gomes[1], Willian Zalewski[2], and André G. Maletzke[1]

[1] Western Paraná State University, Foz do Iguaçu, Paraná, BR
{guilherme.gomes,andre.maletzke}@unioeste.br
[2] Federal University for Latin American Integration, Foz do Iguaçu, Paraná, BR
willian.zalewski@unila.edu.br

Abstract. We advocate that no single quantifier consistently outperforms all others across every possible scenario. We also argue that experimental evaluation in quantification using the Artificial-Prevalence Protocol is significantly more costly than in classification. Although the community has made strides in reducing the number of algorithms to be tested in classification scenarios, this challenge in quantification remains unexplored. To address this issue, we introduce a method that recommends quantifiers for each dataset, leveraging the concept of meta-learning. By analyzing the intrinsic characteristics of datasets through meta-features, our method predicts the most suitable quantification algorithm likely to yield optimal results. Our proposal automates the selection process, providing data-driven recommendations that enhance the efficiency and effectiveness of quantification tasks. We achieved a recommendation accuracy of 83%, meaning that our system successfully identified the optimal quantifier for 83 out of 100 datasets. Furthermore, our architecture enables us to build an ensemble of quantifiers using, for instance, the recommended Top-k quantifiers. Our ensembles lead to superior quantification results compared to other state-of-the-art quantifiers, such as DyS, SORD, and MS.

Keywords: Meta-Features · Recommendation · Learning to quantify

1 Introduction

Meta-learning (MtL)-based recommendation systems can be a viable solution to automatically select data-driven algorithms using knowledge extracted from previous tasks [25]. Meta-learning systems indicate which algorithm should be utilized to achieve the best possible results for each task, according to its particularities [6]. However, for recommendations to be made, this system needs to acquire experience, for example, from (i) model evaluations, which involve recommending hyperparameter values, configuration search spaces, and optimization approaches for analogous tasks; (ii) previously successful models using transfer learning; and (iii) exploring task properties to recommend algorithms based on data characterization and learning performance [25].

Exploring meta-learning for building recommendation systems has been investigated over the years. For example, Ali & Smith [4] used accuracy and complexity measures to build a classifier recommendation system. In [1], the authors

also used meta-learning to recommend image segmentation algorithms, and in [24], meta-learning was used to recommend clustering algorithms. Meta-learning as a tool for selecting machine learning algorithms has shown great promise in various machine learning tasks [21,12,23].

Recently, a novel supervised task known as quantification has garnered significant interest from the machine learning community. The task aims to accurately determine the prevalence of each class within an unlabeled dataset. A key distinction between quantification and classification is that the class distribution in quantification is not fixed. Otherwise, it could easily predict the class proportions in the test set based on the training set. Several quantification algorithms have been proposed in the last decade. However, compared to the classification task, quantification has a significantly smaller number of methods. In contrast, quantification experiments are more expensive than classification, requiring several test sets with different class distributions. Consequently, determining the most suitable method for a new problem can be expensive and time consuming. Previous studies have dedicated efforts to evaluate the performance of several quantifiers under different conditions [20,16,27,14]

We advocate that no single quantifier consistently outperforms all others across every possible scenario. This variability in performance requires a tailored approach to selecting the most appropriate quantifier for each dataset. We also argue that experimental evaluation in quantification is massively more costly than classification. Although the community has delivered proposals to reduce the number of algorithms to be experimented within classification scenarios, this challenge in quantification remains unexplored.

Investigating a meta-learning strategy to develop recommendation systems for quantification algorithms is still uncharted territory. To address this challenge, we suggest quantifiers for each dataset using meta-learning principles. By examining the inherent properties of datasets through meta-features, our method forecasts the most appropriate quantification algorithm expected to deliver optimal outcomes. This approach automates the selection procedure, offering data-driven recommendations that improve the efficiency and effectiveness of quantification activities.

The structure of this article is as follows: Section 2 describes the basic concepts and differences between the classification and quantification tasks as well as the algorithm recommendation task formalization, including meta-learning concepts. Section 3 reviews the literature on related works. Sections 4 and 5 present our meta-learning architecture for recommending quantifiers and the experimental setup of this paper, respectively. Section 6 presents the empirical results and discussion. Finally, Section 7 concludes this work and present directions for future work.

2 Background

In machine learning, classification assigns one or more classes to a set of individual data items. To achieve this, a learner $\boldsymbol{h}$ is used to generalize a hy-

pothesis from a training set $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of length n, in which $x_i \in \mathcal{X} = \{a_{i1}, \ldots, a_{im}\}$ represents a vector with m attributes in the feature space $\mathcal{X}$ and $y_i \in \mathcal{Y} = \{c_1, \ldots, c_w\}$ represents a label in the w-label space $\mathcal{Y}$. The hypothesis aims to cover the available training examples and future unseen examples. In this paper, we focus on binary problems, *i.e.*, the label space $\mathcal{Y}$ is restricted to $\mathcal{Y} = \{\oplus, \ominus\}$. Therefore, a classifier is a function that maps an instance of $\mathcal{X}$ to a subset of $\mathcal{Y}$ as follows $\boldsymbol{h} : \mathcal{X} \to \{\oplus, \ominus\}$.

Many classification algorithms generate scores as an intermediate step in deciding which class is assigned to an instance [9]. In binary classification, it suffices to consider the score for only one of the classes, such as the score for the positive class. A scorer $\boldsymbol{s}(x_i)$ is a function that maps each instance x_i to a value correlating to $\mathrm{P}(y_i = \oplus | x_i)$ as the following equation $\boldsymbol{s} : x_i \to \mathbb{R}$ [17]. Therefore, a classifier is subsequently achieved by applying a threshold to the score values, categorizing them as positive or negative.

Over the years, algorithms to build data-driven models have been proposed, resulting in a large number of options. For instance, both $\boldsymbol{h}$ and $\boldsymbol{s}$ are models induced by some machine learning algorithm based on a dataset. Selecting the most appropriate method to fit $\boldsymbol{h}$ and $\boldsymbol{s}$ for a dataset can be laborious. This fact has inaugurated a new research field that aims to recommend algorithms based on data features. One approach is to develop an automated system for recommending algorithms, which relies on the relationship between algorithm performance and dataset characteristics to suggest suitable algorithms directly.

Let $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\mathcal{M}\}$ represents a set of $\mathcal{M}$ datasets across various domains and $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\mathcal{N}\}$ a set of $\mathcal{N}$ candidate algorithms for proper induction of $\boldsymbol{h}$ and $\boldsymbol{s}$ along with a model quality measure η. An algorithm recommendation for a given $\mathcal{S}_d$ can be defined as follows:

$$\text{Recommender}(\mathcal{S}_d) = \underset{\mathcal{A}_r \in \mathcal{A}}{\mathbf{arg\,max}}\, \eta(\mathcal{S}_d, \mathcal{A}_r)$$

where $\mathbf{arg\,max}_{\mathcal{A}_r \in \mathcal{A}}$ selects the algorithm $\mathcal{A}_r$ that maximizes the quality measure η for the dataset $\mathcal{S}_d$.

The selection of quality measures is intrinsically related to the specific nature of the problem and the task at hand. For instance, accuracy is typically employed as a quality metric in classification tasks, while Mean Absolute Error (MAE) is conventionally utilized in regression tasks.

2.1 Quantification

Classification aims at assigning one or more classes to each instance from a distribution. Nevertheless, in many cases, the primary objective is to estimate the proportion of each class in the test set rather than to label individual data points. In such applications, predictions of samples matter more than individual instances. This distinction highlights the difference between classification and quantification.

A quantifier is a predictive model $\boldsymbol{q}$ induced from a dataset to predict the class distribution of an unlabeled set, defined according to the following equation:

$$\boldsymbol{q} : 2^{\mathcal{X}} \rightarrow [0, 1]$$

where, $2^{\mathcal{X}}$ represents the power set of $\mathcal{X}$ and $\boldsymbol{q}$ outputs a single number in the interval $[0, 1]$ that correlates to the positive class prevalence.

An important distinction between quantification and classification lies in the non-stationarity of class distributions found in quantification problems. Otherwise, it would be trivial to predict the class proportions in the test set based on the distribution in the training set. Consequently, the most straightforward quantifier, called Classify and Count (CC), which involves classifying each instance and then counting how many of them belong to each class does not match in this context, suffering from the systemic error introduced when the class distribution varies. Numerous techniques have been suggested by the quantification community to address this issue. One of the first is the Adjusted Classify and Count (ACC), which corrects the bias in CC by adopting a correction factor based on the model's true positive rate (*tpr*) and false positive rate (*fpr*).

ACC provides perfect quantification results regardless of the classifier accuracy when precises *tpr* and *fpr* are provided. However, data scarcity and class imbalance make estimating these statistics challenging. The symbiosis between classification and quantification extends far beyond CC and ACC. The quantification community has proposed numerous methods, most of which rely on binary classifiers that produce a score representing the confidence in a positive classification as an intermediate step for quantification. González et al. (2017) [13] provide a comprehensive survey of quantification methods, structuring them into a taxonomy of three groups:

i. *Classify, Count, and Correct:* methods that classify each instance and subsequently count the examples that belong to each class. This group also includes methods that apply a correction factor to these counts, including techniques that account for classification error;
ii. *Adaptation of classification algorithms:* approaches that modify the mechanics of classification algorithms to transform them into quantifiers;
iii. *Distribution matching:* methods that model the training data distribution, typically represented by $P(x|y)$, varying $P(y)$, and then seek parameters that best match the test data distributions.

Table 1 briefly presents some of the most known and utilized quantifiers in the literature. The last column provides the corresponding reference for each method. All methods, except those in the group *ii*, require a preliminary step of learning a classifier capable of predicting a score for each unlabeled test instance. A score is a numerical value that correlates with the posterior probability of a particular class, *i.e.*, $P(\oplus|x)$ for binary problems. Various machine learning algorithms can be used to obtain a scorer, removing the decision threshold step. However, selecting the best scorer does not guarantee the best quantifier, as the premise $P_{training}(y) = P_{test}(y)$ cannot be held in quantification scenarios.

Selecting the best quantifier is highly dependent on the specific characteristics of each dataset, requiring performing extensive experiments using, for instance, the Artificial-Prevalence Protocol (APP) [10]. APP is the most commonly

Table 1: Quantifiers evaluated.

Taxonomy Group	Quantifier	Acronym	Reference
i	Classify and Count	CC	[10]
	Adjust Classify and Count	ACC	
	Probabilistic Classify	PCC	[5]
	Probabilistic Adjust Classify and Count	PACC	
	Threshold Selection Method		[11]
	Set the decision threshold where $(1 - tpr) = fpr$	X	
	Set the decision threshold where $tpr - fpr$ is maximized	MAX	
	Estimate tpr and fpr for several thresholds, returning the median of them	MS	
ii	Quantification Trees	QT	[22]
iii	Expectation Maximization Quantification	EMQ	[26]
	Distribution matching with Hellinger Distance	HDy	[15]
	Mixture Model Framework	DyS	[19]
	Sample Mean Matching	SMM	[16]
	Sample-ORD Method	SORD	[19]

employed experimental framework for evaluating and contrasting quantification techniques. For binary problems, APP generates several test sets, sub-sampling examples randomly from $\oplus$ or $\ominus$ with predetermined class distributions. Commonly, test sets are generated varying the class distribution across a wide range of possibilities, such as $p = P(\oplus) \in \{0, .01, .02, \ldots, .99, 1\}$. APP entails a probabilistic decision, making it susceptible to random variability. Consequently, many researchers opt to replicate this experiment to minimize variance. This procedure leads to an assessment over a large number of test sets [17].

Meta-learning emerges as a valuable tool, leveraging prior knowledge and experience from multiple learning tasks to guide quantifier selection. By analyzing the properties of datasets and their interactions with various quantifiers, meta-learning can identify patterns and correlations that are not immediately apparent. This enables the development of recommendation systems that predict the most suitable quantifiers or a group of quantifiers for a given dataset. According to Garcia et al. (2018) [12], meta-learning can help differentiate the performance of a set of machine learning methods, aiding in the selection of the best method for a given problem.

2.2 Meta-learning

Meta-learning aims to automate selecting, tuning, and combining machine learning algorithms to improve overall performance across a wide range of tasks [3]. To achieve this, past experiences are utilized to learn from the learning process itself, a concept known as learning to learn.

Meta-learning-based recommendation systems feature automatic technique selection driven by data, utilizing knowledge extracted from previous tasks [6]. According to Rivolli et al. (2022) [25], MtL incorporates the following components: the problem space ($\mathcal{S}$), the feature space ($\mathcal{F}$), the machine learning algorithm space ($\mathcal{A}$), the performance space ($\mathcal{E}$), and the machine learning algorithm used for MtL. Therefore, through a data-driven process, the performance ($\mathcal{E}$) of

a set of algorithms ($\mathcal{A}$) on various datasets ($\mathcal{S}$) is associated with characteristics of these datasets ($\mathcal{F}$), represented by meta-features. Thus, a machine learning model is induced from the meta-data, represented by $\mathcal{F}$, recommending the most suitable machine learning algorithm from $\mathcal{A}$ for a new dataset.

Meta-features must be tailored to the problem at hand, being able to characterize the problem aiming to induce an effective recommendation model. Existing an assorted strategies for constructing meta-features. Rivolli et al. (2022) [25] propose organizing these strategies into six groups as follows:

- **Simple:** measures that are easily determined and generally do not require high computational effort. Commonly referred to as general metrics, including the number of dataset instances and the number of attributes.
- **Statistical:** these measures capture statistical properties from a dataset, such as mean, standard deviation, correlation, skewness, and kurtosis.
- **Information Theory:** these measures explore information theory concepts to describe a dataset. These metrics rely on entropy, quantifying the information content and complexity within the data.
- **Model-Based:** these measures are extracted from a fitted model learned on the training data. Although these measures can be extracted from different sorts of models, typically, they are derived from decision tree models, including the count of leaves, nodes, and the structure of the decision tree.
- **Landmarking:** these measures are based on the performance of fast and simple learning systems and algorithms to characterize a dataset. These algorithms should exhibit different inductive biases and be able to capture relevant information at a low computational cost.
- **Others:** represent measures that do not fit into any of the previous groups, generally including domain-related concepts and time-related measures.

Constructing a recommender system with meta-features involves multiple stages. Firstly, meta-features are selected and extracted from a diverse set of datasets, creating a meta-dataset where each instance represents a dataset. After that, a performance metric for each recommendable algorithm is estimated and included in the meta-table. A meta-learner is then trained on this meta-dataset to predict the performance of different algorithms based on the meta-features. Finally, when a new dataset arrives, the meta-features are extracted and input into the trained meta-learner, predicting the best-performing algorithm.

3 Related Works

Meta-learning has been effectively used to develop recommender systems across various domains. Wang et al. [30] and Zhang et al. (2019) [31] created systems that recommend feature selection and imbalance learning methods, respectively, by associating dataset meta-features with algorithm performance in a meta-table. Similarly, das Dôres et al. (2016) [8] proposed a framework for recommending software fault prediction algorithms, while Garcia et al. (2018) [12] focused on recommending classifiers by predicting their accuracy using data complexity

measures. Zhu et al. (2018) [32] introduced a novel approach by applying link prediction in a network of datasets and classifiers to recommend appropriate classifiers. These studies demonstrate the broad applicability of meta-learning in building recommendation systems, with various frameworks proposed for different tasks [24,1,4].

In the last decade, quantification methods have been proposed to address various challenges in machine learning tasks. This rapid development has led to a diverse landscape of techniques, each with unique strengths and weaknesses tailored to specific types of data and problem scenarios. While this variety enriches the field, it also introduces a significant challenge: selecting the most suitable quantification method for a given problem has become increasingly complex.

To address this challenge, we propose a novel framework for recommending quantifiers based on meta-features. To the best of our knowledge, the task of recommending quantifiers has not been explored previously. Our approach leverages the wealth of existing quantification methods and automates the selection process by using meta-features to characterize datasets. These meta-features capture essential properties of the data, enabling the system to predict which quantifiers are likely to perform best.

4 Architecture for Recommending Quantifiers

In this section, we introduce a novel method that recommends quantifiers for each dataset by leveraging the concept of meta-learning. By analyzing the intrinsic characteristics of datasets through meta-features, our approach predicts the most suitable quantification algorithm likely to yield optimal results. This proposal automates the selection process, providing data-driven recommendations that enhance the efficiency and effectiveness of quantification tasks.

Initially, we extract meta-features from various datasets to capture their characteristics. Then, we estimate the performance of several quantifiers on these datasets using APP. After that, the extracted meta-features and performance metrics are unified to build the meta-table that serves, in the next step, to fit a recommender model. Finally, when a new and unseen dataset arrives, we extract its meta-features, providing them to the recommender that predicts the most appropriate quantifier. Figure 1 shows the architecture of our proposal.

Inspired by the literature on meta-learning recommendation, our proposal involves the following steps: (1) meta-features extraction, (2) meta-target estimation, (3) meta-learner induction, and (4) quantifier recommendation.

Step 1 - Meta-features extraction: for each dataset, meta-features are derived, resulting in a meta-instance within the meta-table. These meta-features encapsulate diverse attributes of the datasets, including statistical properties, information-theoretic metrics, and model-based characteristics. Consequently, each meta-instance in the meta-table offers a thorough summary of a dataset's principal features, aiding the recommender system in determining the most appropriate quantification algorithms for new datasets based

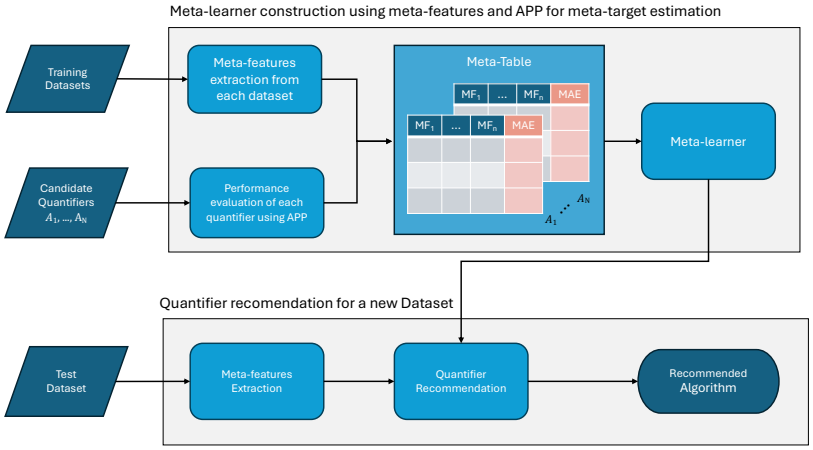

Fig. 1: Architecture for quantifiers recommendation.

on their meta-features. Figure 2 shows the intuition of the meta-feature extraction process.

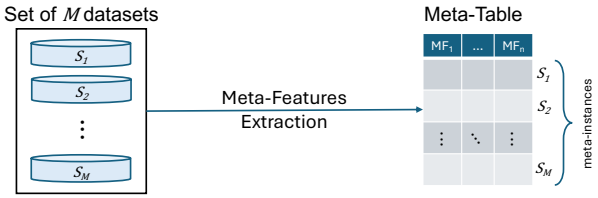

Fig. 2: Meta-features extraction process.

Both quantification and classification are supervised tasks that can explore diverse types of meta-features, *i.e.*, supervised and nonsupervised meta-features. In our architecture, we use the following groups of meta-features: simple, statistical, information-theoretic, model-based, and landmarking. The number of meta-features extracted varies across datasets. To standardize the meta-dataset, since instances cannot have different numbers of attributes, the features values are aggregated by the mean value for each feature, ensuring all feature sets have the same number of elements. Additionally, the range of the meta-features can vary significantly between datasets. To address this, the resulting meta-table can be normalized via min-max scaling.

Step 2 - Meta-target estimation: to build a recommendation model, the target attribute (or class) must be added to the meta-table, indicating, for example, which algorithm is most suitable for each dataset. For classification problems, the meta-target represents the performance of the classification

algorithm such as accuracy or F-score [25]. In the context of quantification, classification metrics are unsuitable. In contrast, in quantification, the performance is based on the ability of minimizing the difference between true ($P(c_i)$) and predicted ($\hat{P}(c_i)$) class distribution for a set of classes $\mathcal{Y}$. Various measures for quantifier evaluation have been adapted from other contexts, such as Mean Absolute Error and Kullback-Leibler Divergence [28]. We use MAE as the meta-target in our architecture. Figure 3 illustrates the process of meta-target estimation.

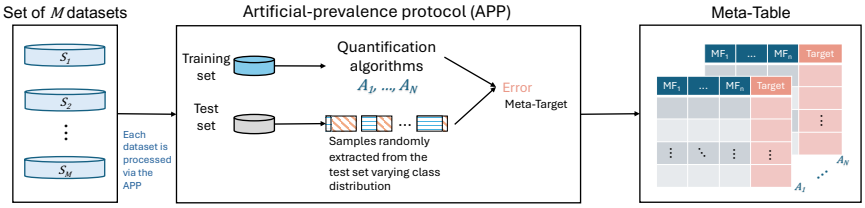

Fig. 3: Meta-target estimation using Artificial-Prevalence Protocol.

Evaluating a quantifier requires providing a sample of instances and varying the class distribution instead of a set of instances, as in the classification setup. We use the Artificial-Prevalence Procotol, which splits the dataset into training and test sets and then extracts several batches varying the class distribution from the test partition.

Step 3 - Meta-learner induction: from the meta-tables built in the previous step, we learn several regressors, one for each meta-table. These regressors serve as recommenders for an unknown dataset. Figure 4 illustrates the procedure that leads to the creation of a collection of meta-learners (regressors).

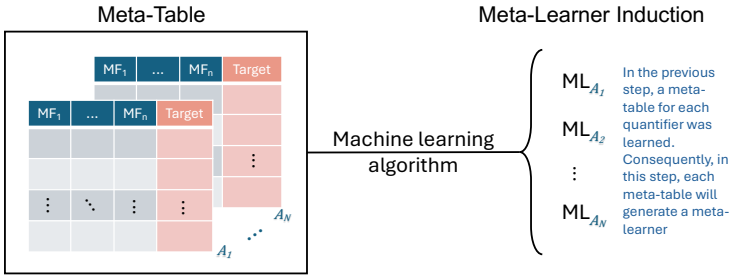

Fig. 4: Meta-learner induction step.

Step 4 - Quantifier recommendation: once the meta-learners are trained, the system can recommend suitable quantification algorithms for new datasets.

When a new dataset is presented, its meta-features are extracted and used as input to the trained meta-learners. Each meta-learner provides a prediction of the expected performance (in terms of MAE) for its corresponding quantifier. Figure 5 shows the recommendation process using meta-learners to predict the best quantifier for a new dataset. In order to enhance the

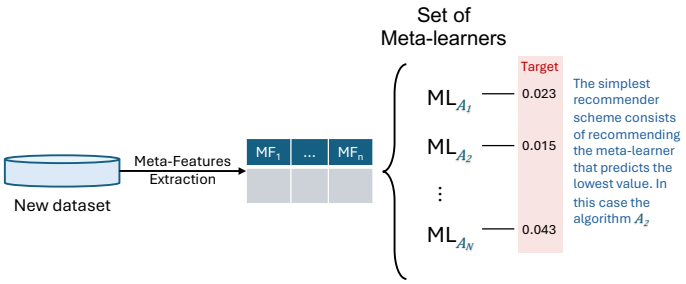

Fig. 5: Recommendation process based on a set of meta-learners.

recommendation's robustness, we propose forming an ensemble of quantifiers by selecting the Top-k recommended methods. The ensemble can be constructed using various strategies, such as averaging the predictions of the selected quantifiers or assigning weights to each quantifier based on their predicted errors[3], where the quantifier with the lowest error receives the highest weight. In our proposal, the following strategies are explored:

- **Top-1:** the recommender selects the best quantifier for a new dataset, choosing the method that produces the lower error. This quantifier is then employed to perform the quantification task, leveraging its suitability as determined by the meta-features of the dataset.
- **Top-k:** we adopt an ensemble strategy to select the k methods whose predicted errors were the lowest. Next, we merge the Top-k methods using the median of their predictions, with the goal of leveraging the advantages of various quantifiers.
- **Top-k+W:** we further refine the ensemble approach by weighting each of the k selected quantifiers based on their errors. Quantifiers with the lowest errors predicted are given higher weights. Let q_i represents the i-th quantifier, and e_i the predicted error by the recommender. The weight w_i for each quantifier is inversely proportional to its error. The weights w_i are calculated as follows: $w_i = \frac{1/e_i}{\sum_{j=1}^{k}(1/e_j)}$ The weighted ensemble quantifier is expressed as:

$$\text{Top-}k + \text{W} = \sum_{i=1}^{k} w_i \cdot \hat{p}_{q_i}(\oplus)$$

[3] Note that the meta-learner aims to predict the MAE for a new dataset. Consequently, meta-learners with the lowest MAE can be combined.

where $\hat{p}_{q_i}(\oplus)$ represents the predicted distribution of the positive class by q_i among the k quantifiers with the smallest errors. This equation guarantees that quantifiers with smaller errors have a greater influence on the final quantification outcome.

5 Experiments

This section details the experiments conducted with multiple established quantification algorithms and a wide range of meta-features. To reduce the presence of bias, we collected a total of 100 datasets of binary classification problems from different domains in public data repositories [18,29]. Each dataset was chosen not only for its diversity but also to ensure it contained enough instances to apply the APP with a batch size of 100 instances. We prepared each dataset by transforming incompatible attributes for machine learning through one-hot encoding for categorical attributes and removing attributes with missing values. Dataset descriptions and codes are available in the paper repository[4].

The meta-features were extracted using the Python package Meta-Feature Extractor (MFE) [2]. We extracted all the default meta-features from the MFE package, resulting in a total of 111 meta-features categorized into the following groups: Simple, Statistical, Information-Theoretic, Model-Based, and Landmarking. Mean and standard deviation were used as summary functions to aggregate the different numbers of meta-features. Min-max normalization was applied to the resulting meta-features.

Since we propose an ensemble method, we generated baseline ensembles for a fair comparison. For every dataset, we chose k quantifiers at random and used the median prediction of these k quantifiers as the ensemble output. We tested 1, 3, and 5 as values k. To reduce the influence of randomly selecting quantifiers, the baseline ensembles were generated 30 times for each dataset, and the mean result was reported for a fair comparison.

To build the meta-target we followed a structured process to accurately assess the performance of each quantification algorithm on the selected datasets. Using the APP, we split each dataset into training and test sets using stratified sampling without replacement with 70% and 30% proportions, respectively. Then, from the training set we learn a scorer (classifier) using Regularized Logistic Regression (LR), provided by the Scikit-Learn[5] library. For each dataset, we tune the following hyperparameters: **C** in the range $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$, and **class-weight** in *balanced* or *none*. Since some quantifiers require scores, *tpr*, and *fpr*, we estimate them using 10-fold stratified cross-validation on the training set. In our experiments, we include quantifiers that require a scorer/classifier in an intermediate step. The following quantifiers were included: CC, ACC, MAX, PCC, PACC, X, MS, HDy, SMM, SORD, and DyS. The hyperparameter settings for HDy and DyS were based on [15] and [19], respectively.

[4] `https://github.com/Bachega/lequa2024_workshop`
[5] `https://scikit-learn.org`

In the testing phase, from each test dataset, we create multiple samples composed of 100 instances with different class distributions in each sample. The positive class distribution in each test sample varies from 0% to 100%, in increments of 5%. We repeated each setting ten times to reduce the error variance and averaged the results.

To assess the performance of quantifiers, we use the MAE in two situations: estimating the meta-target of the meta-table in Step 2 and evaluating the final performance of quantifiers. We compare the methods according to the Friedman test with 95% confidence and the Nemenyi post-hoc test.

To evaluate the effectiveness of recommendations, we use the recommendation hit that evaluates the success of a recommendation system by checking if the recommended algorithm is among the top performing ones for a given dataset. It ensures that the recommended algorithm is either the best or performs similarly to the best algorithms, thereby validating the recommender's effectiveness [30]. We employ the Random Forest algorithm as the meta-learner, utilizing the default hyperparameters available in the Scikit-Learn library[5] library.

Suppose $\mathcal{A}_{\text{opt}}$ represents the optimal quantifier algorithm for a dataset $\mathcal{S}_d$, and $\mathcal{A}_{\text{Setopt}}$ denotes the set of quantifier algorithms in which each algorithm has no significant difference from $\mathcal{A}_{\text{opt}}$, including $\mathcal{A}_{\text{opt}}$ itself. The recommendation hit for a dataset $\mathcal{S}_d$ is defined as a binary measure that indicates whether the recommended algorithm $\mathcal{A}_{\text{rec}}$ is in the optimal algorithm set. Therefore, $\text{Hit}(\mathcal{A}_{\text{rec}}, S_d) = 1$ indicates that the recommendation is effective and the recommended quantifier algorithm $\mathcal{A}_{\text{rec}}$ is included in $\mathcal{A}_{\text{Setopt}}$ for $\mathcal{S}_d$. Conversely, $\text{Hit}(\mathcal{A}_{\text{rec}}, S_d) = 0$ means that the recommended quantifier algorithm $\mathcal{A}_{\text{rec}}$ does not belong to $\mathcal{A}_{\text{Setopt}}$. In other words, $\mathcal{A}_{\text{rec}}$ performs significantly worse than the optimal quantifier algorithm $\mathcal{A}_{\text{opt}}$ on $\mathcal{S}_d$, indicating a poor recommendation. The $\mathcal{A}_{\text{Setopt}}$ set for each dataset is defined using a Friedman test followed by the Conover post-hoc test with Holm procedure, as recommended by [30], performed at a significance level of 0.05. To evaluate the recommendation system, we calculate the average Hit Ratio using Leave-One-Out across all datasets $\mathcal{S}$:

$$\text{Hit Ratio}(\mathcal{A}_{\text{rec}}, \mathcal{S}) = \frac{1}{\mathcal{M}} \sum_{d=1}^{\mathcal{M}} \text{Hit}(\mathcal{A}_{\text{rec}}, \mathcal{S}_d)$$

6 Results

Our meta-learner achieves a Hit Ratio of 0.83 (83%) when it recommends the best quantifier ($k = 1$), which means that for 83 out of 100 datasets, the recommended quantifier is optimal, according to the Hit Ratio measure. As we consider the recommendation of the Top-3 ($k = 3$) and Top-5 ($k = 5$) scenarios, the Hit Ratios increase to 0.97 (97%) and 1.00 (100%), respectively. These results were expected given the criterion utilized by the Hit Ratio, which is based on a statistical test to define the k best quantifiers. This means that a Hit is recorded not only for the absolute best quantifier but also for those that do not have a statistically significant difference from the best one, ensuring a more comprehensive

evaluation of the recommended quantifiers' effectiveness. These results support our main hypothesis that an MtL architecture can effectively recommend high-quality quantifiers according to the characteristics of each dataset.

Inspired by our preliminary findings, our subsequent analysis seeks to demonstrate the improvement in quantification outcomes achieved by either suggesting the optimal quantifier ($k = 1$) or constructing an ensemble composed of the k recommended quantifiers, as outlined in Step 4 of our architecture. Figure 6 summarizes the results to illustrate the overall performance in terms of the average ranking of quantifiers across 100 datasets.

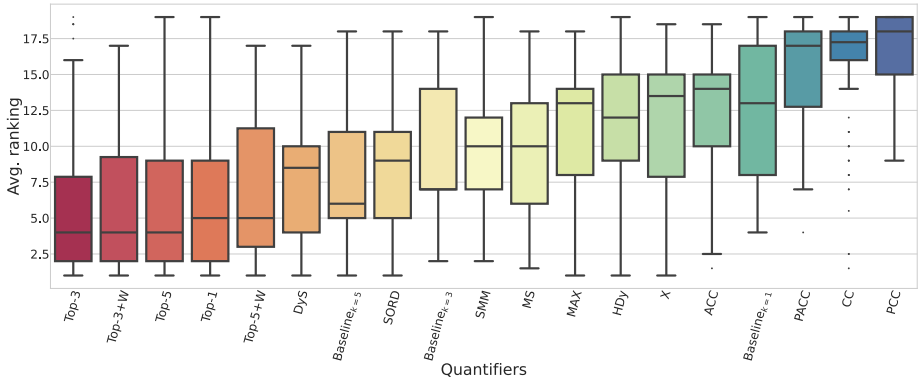

Fig. 6: Aggregation of several rank positions for all quantifiers, including those recommended by our MtL architecture.

To compute the average ranking for each dataset, we first assess the ranking of all quantifiers based on MAE for each dataset and subsequently calculate the mean ranks across all datasets. The quantifier that provides minimum MAE is ranked first. We highlight that the Leave-One-Out cross-validation was employed, ensuring that the dataset under testing is excluded from the training partition of the meta-leaner induction. Consequently, each dataset is tested individually while not contributing to the training process, providing an unbiased evaluation of the recommendation architecture's performance. This rigorous validation technique enhances the reliability of our results by ensuring that the model's predictions are not influenced by prior exposure to the test dataset.

Our MtL architecture improves on the best quantifiers and outperforms them in a large number of datasets from a variety of domains. This achievement corroborates our ancillary hypothesis, demonstrating the robustness and adaptability of our recommendation architecture. Additionally, our proposal helps to select the best k quantifiers using the Hit Ratio criterion, enabling the construction of an ensemble of quantifiers in a data-driven scheme. This approach further enhances the accuracy and reliability of quantification tasks by leveraging the combined strengths of multiple quantifiers tailored to the unique characteristics

of each dataset. Figure 7 shows the critical difference diagram for all quantification approaches, including those built by our architecture.

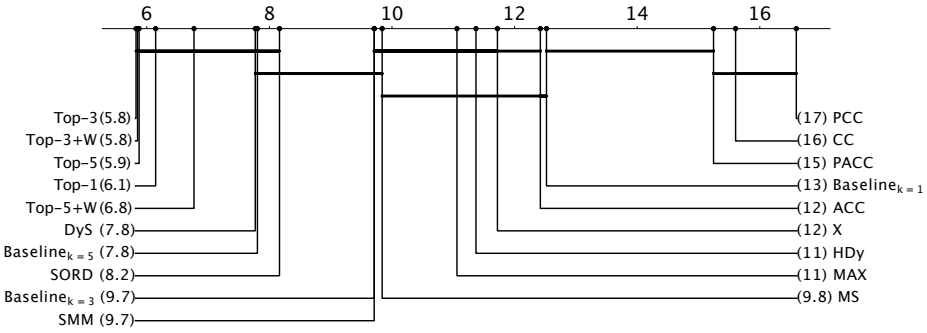

Fig. 7: Friedman's Nemenyi post-hoc test [7] for mean absolute quantification error. Groups of methods that are not significantly different at $p < 0.05$ are connected.

Our architecture identified the optimal quantifier for most datasets (Top-1), leading to enhanced performance compared to choosing a single quantifier (DyS). The results reveal that the recommended quantifier significantly outperform most quantifiers, except DyS and SORD. Using our architecture to select a quantifier for a given dataset, or applying our proposed ensemble methods, resulted in a statistical better performance than any other existing quantifier, except DyS and SORD. Although no statistical difference was observed between our proposals and state-of-the-art quantifiers, choosing a base quantifier tailored by data characteristics through our meta-learning approach proves to be highly effective and stable, being Top-1 ranked consistently better than any other existing methods.

The comparison between ensemble and nonensemble methods might be considered unfair. Thus, we include baseline ensembles that select k quantifiers randomly. The proposed ensembles consistently outperform the baselines. Whether weighted or not, the Top-3 and Top-5 methods demonstrate superior effectiveness compared to the baselines, which confirms that building ensembles through meta-learning leads to better quantification accuracy. Interestingly, the baselines also demonstrate competitive performance, particularly Baseline$_{k=3}$ and Baseline$_{k=5}$, surpassing most base quantifiers. The comparable performance of most base quantifiers can explain this result, as also noted by [27].

Finally, our ensembles consistently outperform the baseline methods and base methods, with the differences being statistically significant in most cases.

7 Conclusion

This paper presents the first architecture for quantifier recommendation using meta-learning, significantly improving the efficiency and effectiveness of quan-

tification tasks. By analyzing datasets through meta-features, our approach predicts the most suitable quantification algorithm for each dataset. Our method can identify the best quantifier for 83% of the studied datasets. Furthermore, our ensemble strategies outperform other state-of-the-art quantifiers, highlighting the robustness of our approach.

Our main contributions include introducing a meta-learning-based scheme for quantifier recommendation, validating its effectiveness through extensive experiments, and demonstrating the potential of ensemble strategies to surpass individual quantifiers. Our architecture is in its early stages, with potential for exploring additional ensemble-building strategies to enhance robustness and performance. Future research should map and analyze situations where recommendations fail, providing insights to refine the recommendation mechanism.

Acknowledgments. We thank the National Council for Scientific and Technological Development (CNPq) (001440855/2022-5) and the Ministry of Science, Technology and Innovation of Brazil (MCTI) for funding and support.

References

1. Aguiar, G.J., Mantovani, R.G., Mastelini, S.M., de Carvalho, A.C., Campos, G.F., Junior, S.B.: A meta-learning approach for selecting image segmentation algorithm. Pattern Recognit. Lett. **128**, 480–487 (2019)
2. Alcobaça, E., Siqueira, F., Rivolli, A., Garcia, L.P., Oliva, J.T., De Carvalho, A.C.: Mfe: Towards reproducible meta-feature extraction. J. Mach. Learn. Res. **21**(111), 1–5 (2020)
3. Alexandros, K., Melanie, H.: Model selection via meta-learning: a comparative study. Int. J. Artif. Intell. Tools **10**(04), 525–554 (2001)
4. Ali, S., Smith, K.A.: On learning algorithm selection for classification. Appl. Soft Comput. **6**(2), 119–138 (2006)
5. Bella, A., Ferri, C., Hernández-Orallo, J., Ramirez-Quintana, M.J.: Quantification via probability estimators. In: IEEE ICDM. pp. 737–742. IEEE (2010)
6. Brazdil, P., Carrier, C.G., Soares, C., Vilalta, R.: Metalearning: Applications to data mining. Springer Science & Business Media (2008)
7. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
8. das Dôres, S.N., Alves, L., Ruiz, D.D., Barros, R.C.: A meta-learning framework for algorithm recommendation in software fault prediction. In: SAC – ACM SIGAPP. pp. 1486–1491. ACM, Pisa, Italy (Apr 2016)
9. Flach, P.: Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press, USA (2012)
10. Forman, G.: Counting positives accurately despite inaccurate classification. In: ECML PKDD. pp. 564–575. Springer (2005)
11. Forman, G.: Quantifying trends accurately despite classifier error and class imbalance. In: KDD – ACM SIGKDD. pp. 157–166. ACM (2006)
12. Garcia, L., Lorena, A., Lehmann, J.: Ecol: Complexity measures for classification problems (2018)
13. González, P., Castaño, A., Chawla, N.V., Coz, J.J.D.: A review on quantification learning. ACM Comput. Surv. **50**(5), 74 (2017)

14. González, P., Moreo, A., Sebastiani, F.: Binary quantification and dataset shift: an experimental investigation. Data Min. Knowl. Discovery pp. 1–43 (2024)
15. González-Castro, V., Alaiz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the hellinger distance. Inf. Sci. **218**, 146 – 164 (2013)
16. Hassan, W., Maletzke, A., Batista, G.: Accurately quantifying a billion instances per second. In: IEEE DSAA. pp. 1–10. IEEE, Sydney, Australia (2020)
17. Hassan, W., Maletzke, A., Batista, G.: Pitfalls in quantification assessment. In: International Workshop on Learning to Quantify: Methods and Applications(LQ2021). vol. 3052. CIKM, GoldCoast, Australia (05 Nov 2021)
18. Kelly, M., Longjohn, R., Nottingham, K.: The uci machine learning repository. `https://archive.ics.uci.edu` (2023)
19. Maletzke, A., dos Reis, D., Cherman, E., Batista, G.: Dys: a framework for mixture models in quantification. In: AAAI. Honolulu, United States (2019)
20. Maletzke, A., Hassan, W., Reis, D.d., Batista, G.: The importance of the test set size in quantification assessment. In: IJCAI. pp. 2640–2646. Yokohama, Japan (July 2020)
21. Mantovani, R.G., Rossi, A.L., Alcobaça, E., Vanschoren, J., de Carvalho, A.C.: A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves svm classifiers. Inf. Sci. **501**, 193–221 (2019)
22. Milli, L., Monreale, A., Rossetti, G., Giannotti, F., Pedreschi, D., Sebastiani, F.: Quantification trees. In: IEEE ICDM. pp. 528–536. IEEE, Abu Dhabi, United Arab Emirates (2013)
23. Olmo, J.L., Romero, C., Gibaja, E., Ventura, S.: Improving meta-learning for algorithm selection by using multi-label classification: A case of study with educational data sets. Int. J. Comput. Intell. Syst. **8**(6), 1144–1164 (2015)
24. Pimentel, B.A., De Carvalho, A.C.: A new data characterization for selecting clustering algorithms using meta-learning. Inf. Sci. **477**, 203–219 (2019)
25. Rivolli, A., Garcia, L.P., Soares, C., Vanschoren, J., de Carvalho, A.C.: Meta-features for meta-learning. Knowledge-Based Syst. **240**, 108101 (2022)
26. Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. Neural Comput. **14**(1), 21–41 (2002)
27. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods. arXiv preprint arXiv:2103.03223 (2021)
28. Sebastiani, F.: Evaluation measures for quantification: An axiomatic approach. Inf. Retr. J. **23**(3), 255–288 (2020)
29. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: Openml: networked science in machine learning. SIGKDD Explorations **15**(2), 49–60 (2013). `https://doi.org/10.1145/2641190.2641198`
30. Wang, G., Song, Q., Sun, H., Zhang, X., Xu, B., Zhou, Y.: A feature subset selection algorithm automatic recommendation method. J. Artif. Intell. Res. **47**, 1–34 (2013)
31. Zhang, X., Li, R., Zhang, B., Yang, Y., Guo, J., Ji, X.: An instance-based learning recommendation algorithm of imbalance handling methods. Appl. Math. Comput. **351**, 204–218 (2019)
32. Zhu, X., Yang, X., Ying, C., Wang, G.: A new classification algorithm recommendation method based on link prediction. Knowledge-Based Syst. **159**, 171–185 (2018)

An Overview of LeQua 2024, the 2nd International Data Challenge on Learning to Quantify

Andrea Esuli, Alejandro Moreo, Fabrizio Sebastiani, and Gianluca Sperduti

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
{firstname.lastname}@isti.cnr.it

Abstract. LeQua 2024 is a data challenge about methods and systems for "learning to quantify" (a.k.a. "quantification", or "class prior estimation"), i.e., for training predictors of the relative frequencies of classes $\mathcal{Y} = \{y_1, ..., y_n\}$ in sets of unlabelled datapoints. While these predictions could be easily achieved by first classifying all datapoints via a classifier and then counting how many datapoints have been assigned to each class, a growing body of literature has shown this approach to be suboptimal, especially when the training data and the test data are affected by some form of dataset shift, and has proposed better methods. The goal of this data challenge is to provide a setting for the comparative evaluation of methods for learning to quantify. LeQua 2024 is the 2nd edition of the LeQua challenge, following the successful 1st edition of 2022.

In LeQua 2024, four tasks were offered. The first three tasks (T1, T2, T3) tackle learning to quantify under prior probability shift, while the fourth task (T4) tackles learning to quantify under covariate shift; T1 and T4 are about binary quantification, T2 is about single-label multiclass quantification, while T3 is about ordinal quantification. For all such tasks, data are provided to participants in ready-made vector form. In this overview article we describe in detail the structure of the data challenge and the results obtained by the participating teams.

1 Learning to Quantify

In a number of applications involving classification, the final goal is not determining which class (or classes) individual unlabelled datapoints (e.g., textual documents, images, or other) belong to, but estimating the *prevalence* (or "relative frequency", or "prior probability", or "prior") of each class $y \in \mathcal{Y} = \{y_1, ..., y_n\}$ in the unlabelled data. Training predictors of the class prevalence values in unlabelled data is known as *learning to quantify* (LQ – a.k.a. *quantification*, or *class prior estimation*) [14, 19, 22].

LQ has several applications in fields (such as the social sciences, political science, market research, epidemiology, and ecological modelling) which are inherently interested in characterising *aggregations* of individuals, rather than the

individuals themselves; disciplines like the ones above are usually *not* interested in finding the needle in the haystack, but in characterising the haystack. For instance, in most applications of tweet sentiment classification we are not concerned with estimating the true class (e.g., Positive, or Negative, or Neutral) of individual tweets. Rather, we are concerned with estimating the relative frequencies of these classes in the set of unlabelled tweets under study; or, put in another way, we are interested in estimating as accurately as possible the true distribution of tweets across the classes.

It has by now unequivocally been shown that performing quantification by classifying each unlabelled instance and then counting, for each class, the instances that have been attributed to the class (the "classify and count" method), usually leads to poor quantification accuracy (see e.g., [3, 7, 9, 12, 13, 23, 33, 34]), due to (a) classifier bias and the mismatch between classification loss and quantification loss, and (b) the presence of dataset shift (see below). This suboptimality of "classify and count" also evokes "Vapnik's principle" [49], which states

> If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem.

In our case, the problem to be solved directly is quantification, while the more general intermediate problem is classification.

One reason why "classify and count" is suboptimal is that many application scenarios suffer from *dataset shift* [31, 41], defined as the situation in which the distribution $P(X, Y)$ from which the labelled training data L have been drawn is different from the distribution $Q(X, Y)$ from which the unlabelled data U have been drawn. The presence of dataset shift means that the well-known IID assumption, on which most learning algorithms for training classifiers hinge, does not hold. In turn, this means that "classify and count" will perform suboptimally on sets of unlabelled datapoints that exhibit dataset shift with respect to the training set, and that the higher the amount of this shift, the worse we can expect "classify and count" to perform.

As a result of the suboptimality of the "classify and count" method, LQ has slowly evolved as a task in its own right, different (in goals, methods, techniques, and evaluation measures) from classification [14,22]. For the near future it is easy to foresee that the interest in LQ will increase, due (a) to the increased awareness that "classify and count" is a suboptimal solution when it comes to prevalence estimation, and (b) to the fact that, with larger and larger quantities of data becoming available and requiring interpretation, in more and more scenarios we will only be able to afford to analyse these data at the aggregate level rather than individually.

LeQua 2024 (`https://lequa2024.github.io/`) follows in the footsteps of LeQua 2022 [16, 17], the first edition of this data challenge. LeQua 2022 was the first data challenge ever to be entirely devoted to quantification; while this topic had surfaced in previous shared tasks, it had never been their real focus.

2 Setting up LeQua 2024

In quantification, a *datapoint* (usually represented as $\mathbf{x}$) is the individual unit of information; for instance, a textual document, an image, a video, are examples of datapoints. As in LeQua 2022, as datapoints we use textual objects (and, more specifically, product reviews); however, this choice causes no loss of generality, since these textual objects are provided to the participants already in vector form.

A datapoint $\mathbf{x}$ has a *class label*, i.e., it belongs to a certain class $y \in \mathcal{Y} = \{y_1, ..., y_n\}$; in this case we indicated by y the label of $\mathbf{x}$. In LeQua 2024, the classes are either the merchandise classes to which the products belong, or the sentiment scores that the authors have attached to the reviews they have written (see Section 2.2 for more). Some datapoints are such that their label is known to the quantification algorithm, and are thus called *labelled datapoints*; we typically use them as training examples for the quantifier-training algorithm. Some other datapoints are such that their label is unknown to the quantifier-training algorithm and to the trained quantifier, and are thus called *unlabelled datapoints*; for testing purposes we use datapoints whose label we hide to the quantifier-training algorithm and to the trained quantifier, and thus play the role of unlabelled datapoints.

Unlike a classifier, a quantifier must not predict labels for individual datapoints, but must predict *prevalence values* for *samples* (i.e., sets) of unlabelled datapoints. A prevalence value $p_\sigma(y_i)$ for a class $y_i \in \mathcal{Y}$ and a sample σ is a number in [0,1] such that the prevalence values $p_\sigma(y_1), ..., p_\sigma(y_n)$ for the classes in $\mathcal{Y}$ sum up to 1; in other words, $p_\sigma(y_1), ..., p_\sigma(y_n)$ are a *distribution* of the datapoints of σ over $\mathcal{Y}$. Note that when, in the following, we use the term "label", we always refer to the label of an individual datapoint (and not of a sample of datapoints; samples do not have labels, but prevalence values for classes).

2.1 The tasks

LeQua 2022 offered four tasks. In two of them, all (training, development, and test) datapoints were provided to participants in ready-made vector form, while in the other two the datapoints were provided in their original textual form. Each of these modalities included two variants: binary quantification and single-label multiclass quantification. In all four tasks, the data were characterised by prior probability shift (see below).

LeQua 2024 also offered four tasks (called T1, T2, T3, T4), but these tasks were (at least partially different) from those of LeQua 2024; in particular,

- In all LeQua 2024 tasks, the datapoints were provided to the participants in ready-made vector form; the goal was to allow the participants to concentrate on optimising their quantification methods, rather than spending time on optimising the process for producing vectorial representations of the datapoints.
- LeQua 2024 includes tasks characterised by different types of dataset shift.

Task	Codeframe structure	Type of dataset shift
T1	Binary	Prior probability shift
T2	Single-label multiclass	Prior probability shift
T3	Ordinal	Prior probability shift
T4	Binary	Covariate shift

Table 1: Main characteristics of the four tasks offered within LeQua 2024.

The main characteristics of the four tasks offered within LeQua 2024 are succinctly described in Table 1.

For each task, participant teams were required not to use any kind of (training / development / test) datapoints other than those provided for that task.

2.2 The data pool

The data we use are Amazon product reviews from a large crawl of such reviews (the same crawl we had used for LeQua 2022). From the result of this crawl we remove (a) all reviews shorter than 200 characters, and (b) all reviews that have not been recognised as "useful" by any user;[1] this yields the "pool" Ω of reviews that we use for our experimentation.

As for the set $\mathcal{Y}$ of class labels,

- for the two binary tasks (T1 and T4) we use two *sentiment* labels, i.e., Positive, which encompasses 4-stars and 5-stars reviews, and Negative, which encompasses 1-star and 2-stars reviews (we discard 3-stars reviews);
- for the multiclass task (T2) we use 28 *topic* labels, representing the merchandise class the product belongs to (e.g., Automotive, Baby, Beauty);[2]
- for the ordinal task (T3) we use the original sentiment label, which ranges on {1-Star, 2-Stars, 3-Stars, 4-Stars, 5-Stars}.

In order to generate the vectorial representations of the reviews we use the ELECTRA-Small model [10], giving each review in input to the generator and using the last hidden state of the model as the representation of the review. Each review is thus represented by a real-valued vector with 256 dimensions.

The data we use for LeQua 2024 are different from the ones we used for LeQua 2022. The main difference is that (a) different datapoints are chosen for the training samples, for the validation samples, and for the test samples, and (b) a different vectorisation is used (while we here use the ELECTRA-Small model, LeQua 2022 used a vectorisation method based on GloVe vectors [39]).

[1] This is meant to filter our "bogus" reviews (e.g., "I'm giving this 1 star because the package was damaged!") that would be difficult for any classifier to label correctly.

[2] The set of 28 topic classes is flat, i.e., there is no hierarchy defined upon it.

2.3 Types of dataset shift and types of data extraction protocols

Dataset shift is defined as the situation in which (i) the training (and development) data that are used for training a model are sampled from a joint distribution $P(X, Y)$, (ii) the unlabelled data on which the trained model is deployed are sampled from a joint distribution $Q(X, Y)$, and (iii) $P(X, Y) \neq Q(X, Y)$.

In LeQua 2024 we consider two types of dataset shift, i.e.,

1. *prior probability shift* (also known as *label shift*), defined as the case in which $P(Y) \neq Q(Y)$ and $P(X|Y) = Q(X|Y)$;
2. *covariate shift*, defined as the case in which $P(X) \neq Q(X)$ and $P(Y|X) = Q(Y|X)$.

The literature on quantification has mostly tackled prior probability shift, and only a few papers [5, 23, 48] have touched on the relationships between quantification and covariate shift.

2.4 The baseline systems

We made the participants aware of the availability of QuaPy [32], a Python-based open-source library[3] for quantification research and development that provides implementations of methods, evaluation measures, parameter optimisation routines, and evaluation protocols.

The implementations of the quantification methods we used as baselines can be accessed via GitHub.[4] These methods include:

- **Classify and Count** (CC): This is the trivial baseline, consisting in training a standard classifier h on the training set L, using this classifier to classify all the data items $\mathbf{x}$ in the sample σ, counting how many such items have been attributed to class y_i, doing this for all classes in $\mathcal{Y}$, and dividing the resulting counts by the cardinality $|\sigma|$ of the sample.
- **Probabilistic Classify and Count** (PCC) [2]: This is a probabilistic variant of CC where the "hard" classifier h is replaced by a "soft" (probabilistic) classifier s, and where counts are replaced by expected counts.
- **Adjusted Classify and Count** (ACC) [19]: This is an "adjusted" variant of CC in which the prevalence values predicted by CC are subsequently corrected by considering the misclassification rates of classifier h, as estimated on a held-out validation set. For our experiments, this held-out set consists of 40% of the training set.
- **Probabilistic Adjusted Classify and Count** (PACC) [2]: This is a probabilistic variant of ACC where the "hard" classifier h is replaced by a "soft" (probabilistic) classifier s, and where counts are replaced by expected counts. Equivalently, it is an "adjusted" variant of PCC in which the prevalence values predicted by PCC are corrected by considering the (probabilistic versions

[3] `https://github.com/HLT-ISTI/QuaPy`

[4] Check the branch `https://github.com/HLT-ISTI/QuaPy/tree/lequa2024/LeQua2024`

of the) misclassification rates of soft classifier s, as estimated on a held-out validation set. For our experiments, this held-out set consists of 40% of the training set.
- The **Saerens-Latinne-Decaestecker** algorithm (SLD) [43] (see also [15]): This is a method based on Expectation Maximization, whereby the posterior probabilities returned by a soft classifier s for data items in an unlabelled set U, and the class prevalence values for U, are iteratively updated in a mutually recursive fashion.
- **DM**, a multiclass implementation of the distribution-matching approach that adheres to the framework proposed by [4, 18], in which the divergence measure to minimise is the Hellinger Distance.
- The recently proposed **KDEy** algorithm [35], a distribution-matching method that models the distribution of the posterior probabilities using kernel density estimation. In particular, we adopt the variant that uses the Kullback-Leibler divergence as the target loss to minimise, which is akin to maximising the likelihood of the test data.

For all methods, we have trained the underlying classifiers via logistic regression, as implemented in the `scikit-learn` framework.[5] Note that all these methods are natively multiclass.

We optimize two hyperparameters of the logistic regression learner by exploring C (the inverse of the regularization strength) in the range $\{10^{-4}, 10^{-3}, \ldots, 10^{+4}\}$ and CLASS_WEIGHT (indicating the relative importance of each class) in {"balanced", "not-balanced"}. For DM, we also explore the number of bins in the range $\{2, 3, \ldots, 10, 12, \ldots, 32, 64\}$. For KDEy, we additionally explore the bandwidth in the range $\{0.01, 0.02, \ldots, 0.20\}$. For each quantification method, model selection is carried out by choosing the combination of hyperparameters that yields the lowest official evaluation error used for each task across all validation samples made available to it.

2.5 Task T1: Binary Quantification under Prior Probability Shift

Task T1 is essentially the same as task T1A in LeQua 2022, i.e., it is a binary quantification task on data affected by prior probability shift, and was re-proposed in LeQua 2024 in order to monitor the progress of the field on what can be considered the "mother" of all quantification tasks.

As mentioned in Section 2.2, the datapoints here have binary sentiment labels, obtained by considering as Positive all 4-stars and 5-stars reviews and by considering as Negative all 1-star and 2-stars reviews.

To obtain our data, first of all we removed from our pool Ω all reviews scored "3-stars". We then obtained the L_1 training set by randomly extracting 5000 reviews from Ω (different from those we used in LeQua 2022), after which we removed them from Ω; the prevalence values of the positives and of the negatives in the extracted set are 0.78 and 0.22, respectively. Subsequently, we simulated

[5] `https://scikit-learn.org/stable/index.html`

the presence of prior probability shift by extracting from Ω development samples and test samples according to the *artificial prevalence protocol* (APP), by now a standard protocol for artificially injecting prior probability shift in the data to be used in the evaluation of quantifiers.

In the general multiclass case, the APP consists of taking the set Ω of datapoints remaining after the extraction of the training set, and extracting from it a number of subsets (the *development samples* and the *test samples*), each characterised by a *predetermined* vector $(p_\sigma(y_1), ..., p_\sigma(y_n))$ of prevalence values, where $y_1, ..., y_n$ are the classes of interest. In other words, for extracting a sample σ, we generate a vector of prevalence values, and randomly select datapoints from Ω accordingly (i.e., by class-conditional random selection of datapoints, until the desired class prevalence values are obtained). Note that in the data released to the participants, for each development sample only the prevalence values that characterise the sample, and not the label of each individual datapoint, was disclosed. The goal of the APP is to generate samples characterised by widely different vectors of prevalence values; this is meant to test the robustness of a *quantifier* (i.e., of an estimator of class prevalence values) in confronting class prevalence values possibly different (or very different) from the ones of the set it has been trained on. For doing this we draw the vectors of class prevalence values uniformly at random from the set of all legitimate such vectors, i.e., from the *unit* $(n-1)$*-simplex* of all vectors $(p_\sigma(y_1), ..., p_\sigma(y_n))$ such that $p_\sigma(y_i) \in [0, 1]$ for all $y_i \in \mathcal{Y}$ and $\sum_{y_i \in \mathcal{Y}} p_\sigma(y_i) = 1$. For this we use the Kraemer algorithm [47], whose goal is that of sampling in such a way that all legitimate class distributions are picked with equal probability. For each vector thus picked we randomly generate a test sample.

Note that the APP indeed simulates prior probability shift, since

- the fact that the samples are randomly selected according to a pre-specified vector of probability values different (in general) from the one that characterises the training set, simulates condition $P(Y) \neq Q(Y)$;
- the fact that the samples are drawn from the same data source from which the training data are drawn simulates condition $P(X|Y) = Q(X|Y)$;
- these two conditions are (see Bullet 1 in Section 2.3) what altogether characterise prior probability shift.

In the binary case ($n = 2$), for generating the D_1 development set we used the APP to extract 1000 development samples consisting of 250 reviews each; for generating the U_1 test set we extracted, in the same way, 5000 test samples also consisting of 250 reviews each. The prevalence values for all samples in U_1 were disclosed to the participants after the end of the challenge.

The evaluation measure. In a theoretical study on the adequacy of evaluation measures for quantification [46], *relative absolute error* (RAE) and *absolute error* (AE) have been found to be, for binary and multiclass quantification, the most satisfactory, and are thus the only measures used in LeQua 2022. RAE and AE

are defined as

$$\mathrm{RAE}(p_\sigma, \hat{p}_\sigma) = \frac{1}{n} \sum_{y \in \mathcal{Y}} \frac{|\hat{p}_\sigma(y) - p_\sigma(y)|}{p_\sigma(y)} \tag{1}$$

$$\mathrm{AE}(p_\sigma, \hat{p}_\sigma) = \frac{1}{n} \sum_{y \in \mathcal{Y}} |\hat{p}_\sigma(y) - p_\sigma(y)| \tag{2}$$

where p_σ is the true distribution on sample σ, $\hat{p}_\sigma$ is the predicted distribution, $\mathcal{Y}$ is the set of classes of interest, and $n = |\mathcal{Y}|$. Note that RAE is undefined when at least one of the classes $y \in \mathcal{Y}$ is such that its prevalence in the sample σ of unlabelled datapoints is 0. To solve this problem, in computing RAE we smooth all $p_\sigma(y)$'s and $\hat{p}_\sigma(y)$'s via additive smoothing, i.e., we take $\underline{p}_\sigma(y) = (\epsilon + p_\sigma(y))/(\epsilon \cdot n + 1)$, where $\underline{p}_\sigma(y)$ denotes the smoothed version of $p_\sigma(y)$ and the denominator is just a normalising factor (same for the $\underline{\hat{p}}_\sigma(y)$'s); following [21], we use the quantity $\epsilon = 1/(2|\sigma|)$ as the smoothing factor. In Equation 1 we then use the smoothed versions of $p_\sigma(y)$ and $\hat{p}_\sigma(y)$ in place of their original non-smoothed versions; as a result, RAE is now always defined.

As the official measure according to which systems are ranked, we use RAE; we also compute AE results, but we do not use them for ranking the systems. The official score obtained by a given quantifier is the average value of the official evaluation measure (RAE) across all test samples; for each system we also compute and report the value of AE. For T1 (but we will do the same for T2, T3, T4 too) we use the Wilcoxon signed-rank test at different confidence levels ($\alpha = 0.05$ and $\alpha = 0.001$) to identify all participant runs that are *not* statistically significantly different from the best run, in terms of RAE and in terms of AE.

2.6 Task T2: Single-Label Multiclass Quantification under Prior Probability Shift

Similarly to Task T1, Task T2 is essentially the same as task T1B in LeQua 2022, i.e., it is a single-label multi-class quantification task on data affected by prior probability shift, and was reproposed in LeQua 2024. Aside from the fact that T1 is binary and T2 is multiclass, the two subtasks are very similar.

Task T2 uses 28 topic labels (the same as in T1B of LeQua 2022), representing the merchandise classes to which Amazon products belong to. We have randomly sampled 20,000 reviews from the pool Ω for use as the training set L_2. Following the same protocol adopted for Task T1, we remove the reviews of L_2 from Ω, and from this reduced pool we extract a development set D_2 composed of 1,000 development samples, each composed of 1,000 reviews. Any review appearing in a development sample is then removed from Ω, after which we proceed with the extraction of the U_2 test set, composed of 5,000 test samples consisting of of 1,000 reviews each. As the evaluation measures, here too we compute RAE (which is also used for ranking the systems) and AE. All other choices made in the design of this experimental setting are the same as for T1.

2.7 Task T3: Ordinal Quantification under Prior Probability Shift

Task T3 is about quantification using an ordinal scale, under prior probability shift; this task is new in LeQua 2024, since no ordinal scales were used in LeQua 2022. The quantification task is defined on a 1-star to 5-stars scale based on the scores assigned to the reviews by their authors. The total order relation among the five possible ratings makes this problem different from a single-label multi-class problem, since misassigning a probability mass to a class faraway from the correct class is a more serious mistake than misassigning it to a class closer in the total order.

For this first edition of the task we chose to follow the natural distribution of data in the pool. The training set is thus composed of samples, i.e., sets of reviews; each sample is composed of reviews of the same product. This is different from the training sets of the other tasks, which are just composed of reviews and leave the use of possible sampling strategies for training up to the participants. Obviously, nothing prevents participants to T3 from building different samples from the entire set of reviews contained in the training set of the task.

The training set L_3 is composed of 100 samples, each associated to a specific product; the 100 products were randomly selected from those with at least 200 reviews in Ω. Each sample in L_3 is composed of exactly 200 reviews. For products with more than 200 reviews we randomly sampled 200 reviews using a stratified random selection. This sampling protocol is known as the *natural prevalence protocol* (NPP). In the case of T3 the PPS among the samples is thus originated by the natural difference in the reviews that products of different quality receive. Our choice of the NPP in place of the APP is motivated by a shortage of data, that would prevent a good APP sampling: the selection of 6,100 products with at least 200 reviews ended up with a large portion of products having just a little more than 200 reviews[6], making the APP eventually produce many unrealistic samples mostly composed of duplicate documents.

Samples in L_3 were provided to participants with the star rating of each review. We then removed the 100 products selected for the training set from the pool and all of their reviews, and we identified a set of 1,000 products for the development set D_3. Each development sample is composed of 200 reviews. Being a development set, the participants were provided with the prevalence of star ratings for each sample, and not the star rating of the single reviews. After removing also the products in the development set from the pool, we sample the test set U_3. The test is composed of 5,000 test samples, related to 5,000 different products, each sample consisting of 200 reviews. Participants had no access to labels or prevalence values for samples in U_3, which have been released publicly after the end of the challenge.

The evaluation measure. The evaluation of quantification predictions for T3 requires taking into account the ordinal scale of the labels when comparing the

[6] This is due to a long-tailed distribution in the original pool, in which few products have many reviews, and many products have very few reviews.

true and predicted distributions. A measure derived from the Earth Mover's Distance, the *Normalised Match Distance* (NMD) [44, 50], takes into account the ordinal relations between classes. NMD is defined as

$$\mathrm{NMD}(p_\sigma, \hat{p}_\sigma) = \frac{1}{n-1} \sum_{j=1}^{n-1} d(y_j, y_{j+1}) \cdot |\sum_{i=1}^{j} \hat{p}_\sigma(y_i) - \sum_{i=1}^{j} p_\sigma(y_i)| \tag{3}$$

where $\frac{1}{n-1}$ is a normalisation factor that allows NMD to range between 0 (best) and 1 (worst), and $d(y_i, y_{i+1})$ is the distance in the ordinal scale among two consecutive labels, which we assume to be always 1. Given all the samples in the test set U_3, each with a true distribution and a predicted distribution, we evaluate NMD for each sample and compute the mean NMD value across all the samples.

In this first edition of T3 we sampled the distributions randomly from the pool, thus replicating in the validation and the test sets the natural unbalance of star-rating distributions towards a high number of stars. By doing a simple mean across all the samples, the more frequent distributions skewed towards a high number of stars give a bigger contribution to NMD. For this reason we evaluate also a macro version of NMD (Macro-NMD). We define $n-1$ bins, one of each interval from 1 to n, assigning each sample to the bin corresponding to the average of the ordinal labels in the sample. The NMD value is computed for each bin separately, and the Macro-NMD is the mean the resulting $n-1$ NMD values. In this way the Macro-NMD gives to the whole spectrum of mean ratings an equal relevance, regardless of how the samples are distributed.

2.8 Task T4: Binary Quantification under Covariate Shift

Another novel task for LeQua is the binary quantification under covariate shift. Any task presented so far in LeQua 2022, and the other tasks of LeQua 2024 are concerned only with PPS, i.e., $P(Y) \neq Q(Y)$, while this novel challenge adds also covariate shift, i.e., $P(X) \neq Q(X)$.

The quantification task of T4 is the same of T1, with the fundamental difference in the sampling process that generates the various sets. The training set L_4 is composed of 5,000 reviews. Reviews are sampled from the a restricted pool that includes only reviewer for products in the Books or in the Electronics categories. We use these two domains to simulate covariate shift because they are the biggest ones. We consider as negative the reviews with one or two stars, and as positive those with four or five stars, discarding the reviews with three stars. The sampling for L_4 is composed of 90% of Books reviews and 10% Electronics reviews. We chose this ratio to be able to simulate a broad range of covariate shift in the development and test samples by varying the ratio of sampling among the two categories. For the training set the sampling is stratified with respect to the sentiment labels, i.e., we replicate the natural distribution of labels from the sampling pool. The development set D_4 is composed of 1,000 development samples, each one composed of 250 reviews. The sampling pool of D_4 is the one of L_4 minus the reviews already included in L_4. The sampling

Table 2: The teams who participated in LeQua 2024 and the tasks for which they submitted runs.

	T1	T2	T3	T4
Lamarr	x	x	x	x
UniOeste	x			
TeamCUFE	x	x	x	x
UniOviedo(Team1)		x	x	x
UniOviedo(Team2)			x	
UniLeiden	x			
UNSW	x	x		

strategy of development samples uses APP on sentiment labels and also on the two categories. APP on sentiment labels generates PPS, while APP on the two categories generates covariate shift. For example, a sample can be created setting a 80%-20% distribution on sentiment labels and a 40%-60% distribution among categories. In this case, 32% of the reviews will be sampled from positive reviews in Books, 48% from positive reviews in Electronics, 8% of negative reviews in Books, and the remaining 12% from negative reviews in Electronics. The test set U_4 is generated in the same way of D_4. The test set samples are 5,000, each one composed of 250 reviews.

Since this is a binary quantification task, the evaluation measures we use are the same as in T1, i.e., RAE (our official evaluation measure for ranking the systems), and AE.

2.9 Preventing data reuse across tasks

All the tasks use the same pool of documents to sample from. The training data and the development data of each task differs from the one of the other tasks, but they are closely related. This holds specially for reviews with sentiment labels or star ratings in tasks T1, T2, and T4. In this scenario, a possible optimisation strategy could be merging all the training data in order to improve a sentiment classifier accuracy, which can give a sensible boost also to quantification accuracy. This and similar reuse of data across tasks does not add any useful contribution to the knowledge on the quantification problems and methods. We prevented this by using for each task a different random shuffle of the dimensions of the vectors produced by embedding model, so that the vectors are informative only within the task they belong.

3 The participating systems

Seven teams submitted runs to LeQua 2024. As shown in in Table 2, there is a quite balanced participation across all the tasks, with a minimum of three

participants in T4, and a maxium of five partipants in T1 and T3. This is a significant difference from LeQua 2022, when most of the participants focused on T1A (now T1). Two teams, Lamarr and TeamCUFE, participated to all four Tasks. We here list the teams in alphabetical order:

- **Lamarr** [28] submitted a run each for all four tasks. All their runs are based on solving an optimisation problem based on the negative log-likelihood loss proposed by [1], optimising a latent representation that is then passed through a softmax to convert it to a probability distribution. The loss has a regularisation component added that promotes distributions tending towards uniformity, with a specific version customised for T3 that promotes a smooth transition of distribution values across the ordinal scale. The contribution of the regularisation component is controlled by a parameter. The posteriors given as input to the optimisation process are obtained training a multi layer perceptron for T1, T2, and T3, and a Logistic Regressor for T4. All the hyperparameters of the classifiers and the one of the regularisation loss have been optimised running a grid search, evaluated on the validation data.
- **UniOeste** [29] submitted a run for T1. The idea is pretty straightforward. The system consists of an ensemble of several binary quantifiers. Different well-known quantification systems from the literature are used, including DyS, HDy, and SLD (among many others), and different classifiers are trained at the basis, including XGBoost, CatBoost, Random Forest, and SVMs. Each quantifier issues a prediction for the test bag and the predictions are then ranked based on the quantifiers' performance. Only the output of the top-performing quantifiers are used to produce the final estimation, which is obtained by averaging the class prevalence predictions of each member of the committee.
- **TeamCUFE** submitted a run each for all four tasks. This team did not give a description of their methods, and thus they cannot be included in the description of results.
- **UniOviedo(Team1)** [40] submitted a run each for T2, T3, and T4. They employed a deep learning method that relies on a novel (permutation-invariant) pooling layer, which models the distribution of bag instances in a latent space as a mixture of Gaussian distributions with learnable mean and covariance matrices. The network uses parallel pooling layers of this type and enhances their combined utility by regularizing them towards minimal Centered Kernel Alignment (CKA). This method follows the "symmetric" approach, where training instances are bags labelled by prevalence (rather than individual data items labelled by class), thus functioning as a bag-based regressor. As such, the network can be trained with specific error metrics in mind. The authors optimised the network for the official evaluation metrics used in each of the tasks they participated in (RAE for T2, T4, and NMD for T4). One of the key differences compared to most other participant teams is that UniOviedo(Team1) utilised part of the validation samples not only for model selection but also for training the model. Additionally, they

also applied some data augmentation heuristics for increasing the number of training bags.

- **UniOviedo(Team2)** submitted a run for T3. Although the participating team did not submit a notebook description of their method, the members (consisting of David Pérez Román and Juan José del Coz from the University of Oviedo) have informed us that the method they applied correspond to their implementation, as made available in the `QuantificationLib` package [6], of the method EDy [8]. EDy belongs to the distribution-matching family of methods and is a variant of the original Energy Distance method proposed by [25]. EDy relies on the Earth Mover Distance (EMD – also known as the Wasserstein loss) as the divergence measure, which is particularly well-suited for ordinal problems, as is the case of T3.
- **UniLeiden** [26] submitted a run for T1. This team used the Continuous Sweep method [27], using an optimised SVM with RBF kernel as the base classifier. They made a comparison on validation data against three other quantifiers, i.e., Median Sweep [20], SLD [42], and DyS [30]. They found that Sweep-based methods performed better than the other methods when the underlying classifier performed poorly, and vice-versa, indicating a link with the results of [45] in which Median Sweep method performed better than SLD and DyS on datasets with smaller training sets (and thus with a likely lower performance of the classifier) than those used in LeQua.
- **UNSW** [11] participated in T1 and T2. This team proposed two ensemble methods for these tasks. For T1, they proposed a Multiple Classifiers - Single Quantifier method (MC-SQ). This method uses an ensemble in which all the members are instances of the same aggregative quantifier (they used DyS [30]) equipped with different classifiers. The classification algorithms used to form the ensemble include Logistic Regression, Linear Discriminant Analysis, Support Vector Machines, Light Gradient Boosting Machines, Gradient Boosting, and CatBoost. The simple rationale of the method is to have many different classifier which are all known to be good overall performers and to exploit the strength of the ensemble to filter out the cases in which some of them may perform worse. Each classifier-quantifier pair in the ensemble was subjected to a joint hyperparameter optimisation. This means that each DyS instance in the ensemble has its own optimised number of bins that is dependent on the specific classifier the quantifier is paired with. Given a test sample, all the quantifiers in the ensemble make their predictions, and the median value is taken as the final prediction of the MC-SQ method. For T2, UNSW followed an approach that is the opposite of T1: a single classification algorithm paired with many different quantifiers, i.e., a Single-Classifier - Multiple Quantifiers (SC-MQ) method. In this case the authors identified Logistic Regression as the most stable and best performing classifier, and decide to evaluate the ensemble approach varying the quantification methods. They use four quantification methods: Energy Distance (EDy) [25], Kernel Density Estimation (KDEy) [36], Generalized Probabilistic Adjusted Classify & Count (GPACC) [18], and a newly proposed EMQ-ini method. The EMQ-ini method is a variant of EMQ [42] that uses the priors

from GPACC as the initial priors for the unlabelled set, instead of directly using the priors from the classifier. Also for T2 the optimisation of each classifier-quantifier pair in the ensemble is performed jointly.

4 Results

In this section we discuss the results obtained by the participant teams in the four subtasks we have proposed. The evaluation campaign started on February 15, 2024, with the release of the training sets ($L_1 \dots L_4$) and of the development sets ($D_1 \dots D_4$); alongside them, the participant teams were provided with a dummy submission, a format checker, and the official evaluation script.[7] The unlabelled test sets ($U_1 \dots U_4$) were released on May 1, 2024; and runs had to be submitted by June 15, 2024.

We used Codalab (`https://codalab.org/`) as the platform for the submission of runs by the teams; each team could submit up to three runs per subtask. The official competition can be accessed at.[8] In this edition we set up a second Codalab instance using the same validation set provided to them. The validation version is available at.[9] This second instance allowed teams to have an immediate evaluation of their methods on validation data, allowing them to check the correctness of their submissions and the consistence of their evaluations with the one performed by the official evaluation platform.

The true labels of the unlabelled test sets were released on May 3, 2024, after the submission period was over and the official results had been announced to the participants. In the rest of this section we discuss the results that the participants' systems and the baseline systems have obtained in the Binary Quantification task (T1, Section 4.1), the Single-Label Multi-Class Quantification task (T2, Section 4.2), the Ordinal Quantification task (T3, Section 4.3), and the Covariate Shift task (T4, Section 4.4),

In the sections to come, we use the following notational conventions for the tables displaying the results of the participant teams. The first column corresponds to the official measure used for ranking the participant systems (RAE in T1, T2, and T4; NMD in T3) while the second column displays the secondary evaluation measure (AE in T1, T2, and T4; Macro-NMD in T3). Results are averaged across the 5,000 test samples. **Boldface** indicates the best method for a given evaluation measure. Superscripts † and ‡ denote the methods (if any) whose scores are *not* statistically significantly different from the best one according to the Wilcoxon signed-rank test at different confidence levels: symbol † indicates $0.001 < p\text{-value} < 0.05$ while symbol ‡ indicates $0.05 \leq p\text{-value}$. The absence of any such symbol indicates $p\text{-value} \leq 0.001$ (i.e., that the difference in performance between the method and the best one is statistically significant at a high confidence level). Baseline methods are typeset in *italic*.

[7] `https://github.com/HLT-ISTI/LeQua2024_scripts/tree/main`
[8] `https://codalab.lisn.upsaclay.fr/competitions/18965`
[9] `https://codalab.lisn.upsaclay.fr/competitions/19100`

4.1 Task T1: Binary Quantification under Prior Probability Shift

Rank	Run	RAE	AE
1	UNSW	**0.09811 ± 0.27043**	0.02063‡± 0.01608
2	*KDEy*	0.10179‡± 0.30431	**0.02043 ± 0.01589**
3	Lamarr	0.10653†± 0.31885	0.02128 ± 0.01667
4	*DM*	0.10699 ± 0.28473	0.02175 ± 0.01669
5	UniOeste	0.10850‡± 0.35492	0.02096†± 0.01648
6	*SLD*	0.11103‡± 0.36698	0.02113 ± 0.01660
7	*PACC*	0.13390 ± 0.46333	0.02399 ± 0.01809
8	UniLeiden	0.13917 ± 0.53773	0.02379 ± 0.01818
9	*ACC*	0.16439 ± 0.60318	0.02644 ± 0.02037
10	*CC*	0.97742 ± 3.91905	0.07955 ± 0.04816
11	*PCC*	1.26562 ± 5.11243	0.10175 ± 0.05985
12	TeamCUFE	2.53730 ± 10.82087	0.22472 ± 0.15197

Table 3: Results of Task T1, binary quantification under prior probability shift.

Table 3 shows the results of the participating teams in T1. The team obtaining the best averaged result is UNSW. In this task, UNSW used a variant of an ensemble method called MC-SQ that combines the output of different classifiers with one aggregative quantifier (see Section 3). This method is not only the one scoring the lowest RAE, but also the one showcasing the smallest variance of the lot. Notwithstanding this, the differences in performance with respect to four other methods (KDEy, Lamarr, UniOeste, and SLD) are not statistically significant according to the statistical test.

In terms of AE, the best performing method is KDEy. This may come as a surprise, as KDEy was originally proposed with multiclass problems in mind, and is not expected to bring to bear any significant advantage in the binary case. Be it as it may, the differences in performance with respect to UNSW (MC-SL) and UniOeste are not significant.

The four CC-variants are relegated to the bottom half of the results table. The UniLeiden's system performance is positioned between that of the baseline methods PACC and ACC in the results table. This is significant, since UniLeiden proposes a variant of the Medium Sweep algorithm, which in turn is an improved variant of ACC. Despite the improvement brought to bear by UniLeiden, the original PACC still seems to perform slightly better in terms of RAE, while at the same time displaying a smaller variance. As a final remark, the (bad) performances of the "unadjusted" variants (CC and PCC) are not comparable with the rest of the methods in the table, yielding errors close to one order of magnitude higher. TeamCUFE's system produces even higher errors, both for RAE and AE, and with much higher variation.

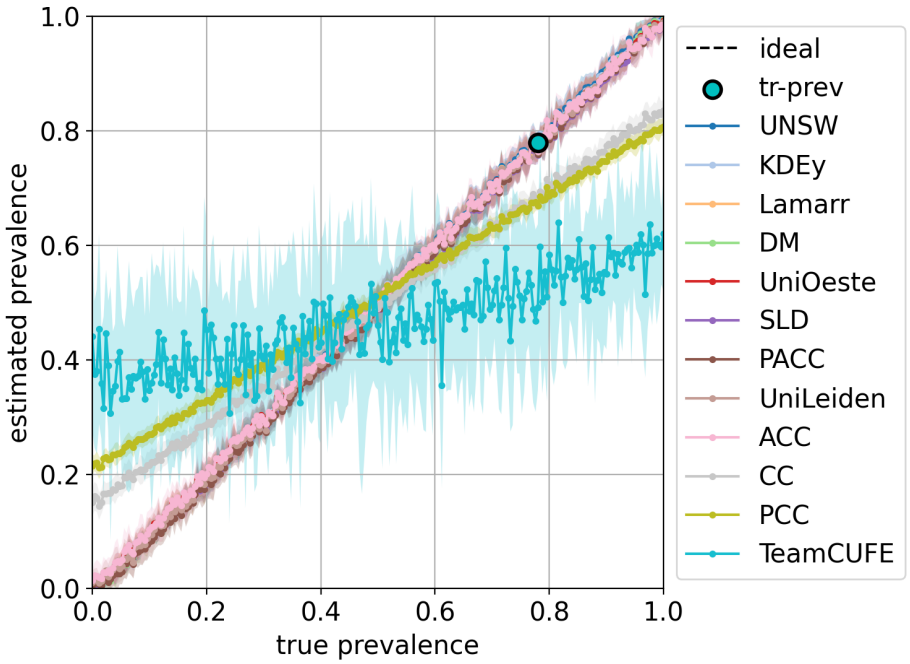

Fig. 1: Diagonal plot for T1.

Figure 1 shows a so-called "diagonal plot" which displays the estimated prevalence value for the positive class (y-axis) as a function of the true positive prevalence (x-axis). The plot is termed "diagonal" because the behaviour of an ideal quantifier is represented by the diagonal line from (0,0) to (1,1). This plot reveals that most of the methods perform fairly well at predicting the positive class prevalence. Exceptions are CC and PCC. The reason for this is that neither of these methods applies any correction to the raw counts obtained from a hard or soft classifier, respectively. Interestingly, CC and PCC do not intersect the diagonal at the point where the true prevalence equals the training prevalence (marked on the plot as a cyan dot), which would be expected since CC and PCC are known to be biased towards the training prevalence. The reason for this deviation is that the classifier's hyperparameters were optimized through model selection on the validation samples, which are designed to exhibit significant variations in prior probability shifts. As a result, the best hyperparameters are those that manage to shift the classifier's bias from the training prevalence to 0.5, which yields a lower averaged RAE across all validation samples. The most noisy output is attained by TeamCUFE. Unfortunately, we do not have details on how this method performs.

Figure 2 shows a different plot that we may dub the "error-by-shift" plot, in which the error (here displayed as log(RAE) to better highlight performance

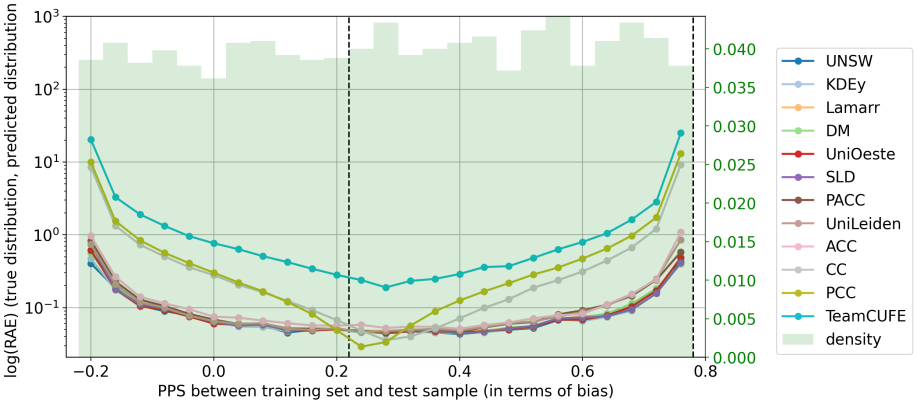

Fig. 2: Error-by-shift plot for T1.

differences) is shown as a function of the amount of PPS of the test samples with respect to the training sample. Here, we measure PPS in terms of the signed difference between the training positive prevalence and the test positive prevalence. The greened region represents the density of experiments carried out, which is close to uniform. Something which seems evident from the plot is that most of the methods yield very small errors across the majority of the shift spectrum, with higher errors concentrated at the extremes of the spectrum, i.e., where all instances are either positive or negative. CC, PCC, and TeamCUFE achieve the smallest error at around 0.25 of shift, rather than at 0 (no shift), as one might expect. This aligns with the previous observation that, through model selection, these methods have shifted their inherent biases from the training prevalence (0.78) to 0.5 in order to minimize the validation error. Consequently, these methods happen to be biased towards the point 0.78-0.5 $\approx$ 0.25, where they obtain the best results. For the rest of the methods, the differences in performance are very thin.

4.2 Task T2: Single-Label Multiclass Quantification under Prior Probability Shift

Table 4 shows the results obtained by the participating teams in T2. UniOviedo(Team1) obtained the best RAE score in the multiclass quantification problem under prior probability shift. This score is not only the smallest, but also the one displaying the smallest variation. The Wilcoxon test reveals this score is statistically significantly better than the rest of the methods with high confidence. In terms of AE, instead, the UNSW team obtained the best averaged score, which is statistically significantly better than the rest of the methods. In this case, the variant employed by the UNSW team corresponds to the a configuration SC-MQ, i.e., to one in which there is only one classifier generating predictions for an ensemble of aggregation methods (see Section 3).

Rank	Run	RAE	AE
1	UniOviedo(Team1)	**0.92173 ± 0.70476**	0.02097 ± 0.00566
2	Lamarr	1.03016 ± 0.84658	0.01412 ± 0.00322
3	UNSW	1.07856 ± 0.96585	**0.01274 ± 0.00353**
4	*SLD*	1.16158 ± 0.99066	0.01343 ± 0.00346
5	*PACC*	1.19418 ± 1.13479	0.01552 ± 0.00424
6	*KDEy*	1.20166 ± 1.05091	0.01367 ± 0.00367
7	*DM*	1.27189 ± 1.09683	0.01578 ± 0.00405
8	*ACC*	1.34787 ± 1.16063	0.01640 ± 0.00427
9	*CC*	2.30963 ± 1.38323	0.01660 ± 0.00310
10	*PCC*	2.67505 ± 1.60472	0.01931 ± 0.00337
11	TeamCUFE	4.02872 ± 2.12809	0.02587 ± 0.00334

Table 4: Results of Task T2, single-label multiclass quantification under prior probability shift.

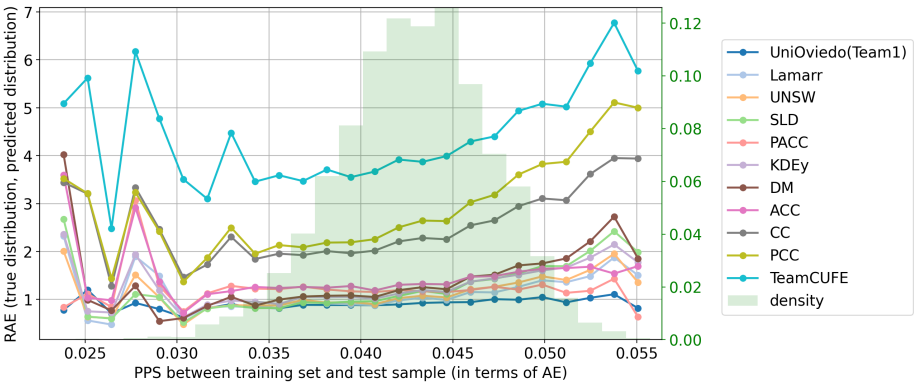

Fig. 3: Error-by-shift plot for T2.

Figure 3 displays the "error-by-shift" plot of T2. In this case, we measure the amount of shift in terms of absolute error (since the signed difference we used in Figure 2 is only defined in the binary case). This plot reveals some interesting facts. First, that most methods degrade their performance at increasing levels of PPS. UniOviedo(Team1) seems to be the most robust in this respect, though. Second, the performance of all methods seems erratic at very high and (specially) very smaller amounts of PPS. However, the density of experiments (greened background) tells us that the number of experiments involved in such cases is very small, which explains the higher variability. This plot also reveals that the distribution of the "amount of shift" generated via APP is close to normal; something that was already echoed in previous research [38].

Rank	Run	NMD	Macro-NMD
1	UniOviedo(Team1)	**0.06438 ± 0.04773**	0.08007‡± 0.02397
2	Lamarr	0.06585 ± 0.04706	**0.07878 ± 0.02038**
3	*PCC*	0.06680 ± 0.05069	0.09004‡± 0.03532
4	*KDEy*	0.06904 ± 0.04982	0.08297‡± 0.01993
5	UniOviedo(Team2)	0.07212 ± 0.05310	0.08771‡± 0.02564
6	*CC*	0.07974 ± 0.04760	0.08820‡± 0.01524
7	TeamCUFE	0.10726 ± 0.07755	0.18647‡± 0.11300
8	*DM*	0.10939 ± 0.06160	0.11445‡± 0.01200
9	*SLD*	0.11107 ± 0.06903	0.10947‡± 0.01220
10	*ACC*	0.11944 ± 0.06496	0.12473‡± 0.01498
11	*PACC*	0.12363 ± 0.06522	0.12830‡± 0.01189

Table 5: Results of Task T3, ordinal quantification under prior probability shift.

4.3 Task T3: Ordinal Quantification under Prior Probability Shift

Table 5 shows the results of the participating teams in T3. Also in this case, UniOviedo(Team1) achieved the best result for the official evaluation metric, which in this case is NMD since this is an ordinal problem. The score obtained by UniOviedo(Team1) is statistically significantly better than the rest of the participating systems. However, in terms of Macro-NMD, the Lamarr team achieved the best result. Nevertheless, according to the Wilcoxon test, no method appears to be statistically significantly different in terms of average rank performance.

It was expected that UniOviedo(Team1) and Lamarr would perform well in this task, as both teams implemented solutions that take into account the ordinal nature of the data. For example, UniOviedo(Team1) directly optimised for the evaluation loss, while Lamarr explicitly regularised their solutions towards a uniform distribution, which ultimately favours smooth solutions. What was unexpected, however, was the seemingly good performance of PCC, a method that not only disregards the ordinal nature of the data but also does not attempt to counter any shift in the priors. This is a clear indication that the samples provided may be characterized by a lower degree of prior probability shift compared to tasks T1 and T2. The reason why, is that the samples are "natural" (generated via NPP), i.e., are not generated via artificially (via APP) imposing widely varying degrees of prior probability shift, as was instead the case for T1 and T2.

Figure 4 displays the averaged performance of the top-5 methods as a function of different levels of "jaggedness" of the tested distributions. More precisely, we compute the jaggedness of a distribution p_σ as:

$$J(p_\sigma) = \frac{1}{2}\sum_{i=2}^{n-1}\left(-p_\sigma(y_{i-1}) + 2p_\sigma(y_i) - p_\sigma(y_{i+1})\right)^2 \tag{4}$$

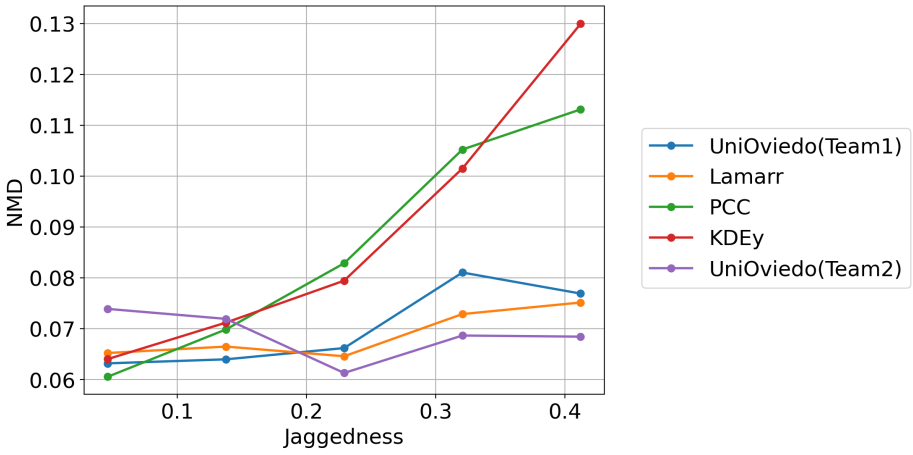

Fig. 4: Quantification performance as a function of the distribution "jaggedness".

The plot reveals that all methods degrade their performance as the distributions become "jaggy". However, UniOviedo(Team1), Lamarr, and UniOviedo(Team2) seem to behave more robustly across the entire spectrum. That Lamarr and UniOviedo(Team2) performed well in T3 was to be expected, as both methods have been designed with ordinal considerations in mind. Also, UniOviedo(Team1) performed well in T3, despite the fact that this method is not specifically suited for ordinal data. Notwithstanding this, note that UniOviedo(Team1) is trained using the validation data as well, which inherently showcase many plausible patterns of ordinal distributions, so one might expect that this method learns to handle the ordinality which resides in the distributions. In contrast, other methods such as PCC and KDEy, which are completely agnostic to the ordinal nature of the data, tend to perform worse in difficult scenarios. The fact that, notwithstanding this, PCC and KDEy rank second and third in the table is a consequence of the fact that most of the samples display low levels of jaggedness.

4.4 Task T4: Binary Quantification under Covariate Shift

Table 6 reports the results obtained for T4. In this case, the Lamarr team scored the best result in terms of the official evaluation measure (RAE), while SLD obtained the best AE. Somehow surprisingly, though, the RAE score obtained by UniOviedo(Team1) is not statistically significantly different from the Lamarr's score, even though UniOviedo(Team1) ranked 5th. This may be explained by the variance of their system, which is markedly higher than that of methods SLD, DM, and KDEy, which occupy the 2nd, 3rd, and 4th positions in the rank, respectively.

As recalled from Section 2.8, task T4 is not only characterized by covariate shift but also by a shift in the priors. It is thus interesting to disentangle the

Rank	Run	RAE	AE
1	Lamarr	**0.10930 ± 0.33394**	0.02191 ± 0.01765
2	*SLD*	0.11497 ± 0.39872	**0.02013 ± 0.01594**
3	*DM*	0.11559 ± 0.32287	0.02341 ± 0.01853
4	*KDEy*	0.11804 ± 0.32916	0.02380 ± 0.01865
5	UniOviedo(Team1)	0.12975‡± 0.42556	0.02122 ± 0.01617
6	*ACC*	0.26187 ± 1.09686	0.03054 ± 0.02454
7	*PACC*	0.28918 ± 1.33046	0.03078 ± 0.02480
8	*CC*	1.11975 ± 4.33704	0.08292 ± 0.04915
9	*PCC*	1.45553 ± 5.63934	0.10735 ± 0.06316
10	TeamCUFE	2.49902 ± 9.93390	0.23017 ± 0.16531

Table 6: Results of Task T4, binary quantification under covariate shift.

systems' performance in terms of both types of shifts, separately. Figure 5 displays the performance of the methods (on logarithmic scale) as a function of the amount of prior shift. The trends we observe are, by and large, in line with those of T1. Figure 6 instead displays the distribution of the errors for the top-5 methods as a function of the prevalence of the Books domain in the test samples, therefore effectively reflecting the amount of covariate shift with respect to the training distribution. This plot shows a weak tendency to improve as the prevalence of Books in the test samples increases, thereby approximating the training mixture (made of 90% Books and 10% Electronics) and reducing the amount of covariate shift. This tendency is more evident for Lamarr, the best performer system for this task. One possible reason why the covariate shift has a weak effect on the results may be that the embeddings were generated using the ELECTRA-Small model [10], which is specifically trained to capture sentiment polarity, rather than topical information, which may therefore be obscured.

Interestingly, PCC has obtained a poor score. This is relevant since this method is considered to behave robustly in the presence of covariate shift. However, it is also known that its performance degrades when some shift in the priors is also at play [24], as is the case for T4. Figure 7 compares the performance of PCC against Lamarr, the top performer in Table 6, proving that PCC's performance in terms of RAE is out of scale.

Figure 8 reports the diagonal plot for T4. Interestingly enough, most methods seem to perform very well notwithstanding the fact that the underlying distributions are affected by covariate shift as well. As witnessed for T1, CC, PCC, and TeamCUFE struggle to obtain good predictions for the entire spectrum, since these methods seem to be strongly biased toward the center of the distribution.

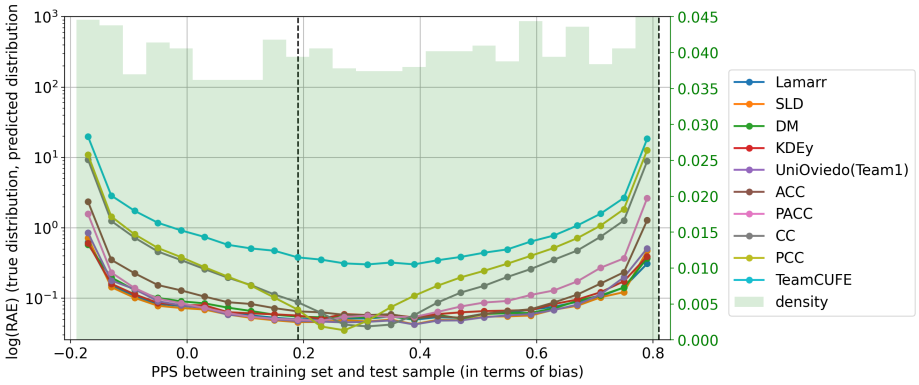

Fig. 5: Error-by-shift plot for T4.

5 Final remarks

In this edition of LeQua, we have observed several tendencies worth remarking. First, that the "symmetric approach" adopted by UniOviedo(Team1) shows promising performance across many different scenarios and involving differently characterised types of dataset shift. This symmetric approach regards the problem as a regression one, in which both the training instances and the test instances are composed by bags of individual datapoints. Such an approach thus requires many bags, each labelled with its class prevalence values. The team used (among other things) the validation samples otherwise provided for model selection. Given that such samples were representative of the type of shift at play in each task, the model was able to learn the particularities of each problem. UniOviedo(Team1) thus leveraged more information during the training phase than the competitors; however, this strategy is not unfair, as the validation samples were made available to all participation teams. Yet another differentiating aspect of this method has to do with its ability to optimise specific loss functions. This capability allowed UniOviedo(Team1) to tune their models for the very same evaluation measures used to rank the participating systems. Perhaps the most important differentiating aspect of this method is the fact that no classifier is involved. This appears promising from the point of view of the Vapnik's principle (outlined in Section 1), since the method is trained to solve the quantification problem *directly*.

One notable observation is that the binary quantification problem (exemplified by T1) is almost a "solved problem", with most methods performing remarkably well in this case. In contrast, the multiclass quantification problem remains significantly more challenging, leaving substantial room for improvement.

Compared to the previous edition, we observed many methods clearly outperforming SLD. This is noteworthy, since SLD is widely considered a very hard-to-beat system [1,37,38] and was one of the top-performing methods in the past LeQua 2022 edition.

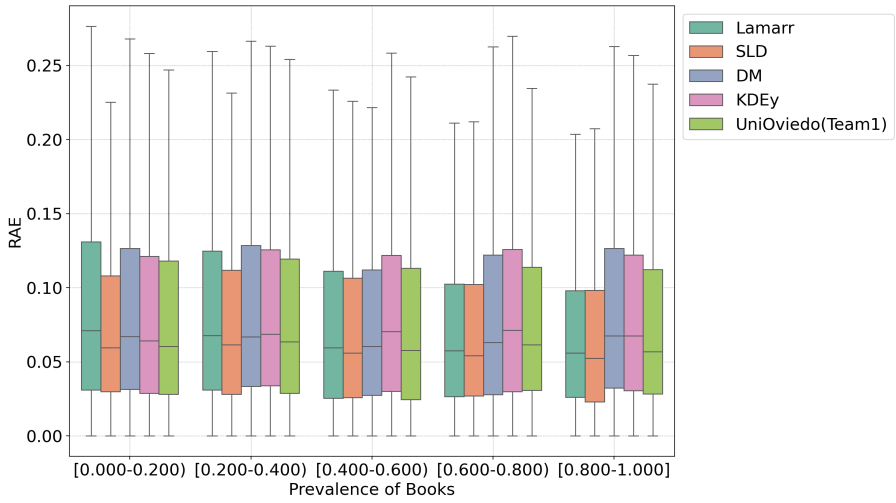

Fig. 6: Errors of the top-5 methods at different proportions of book and electronics domains.

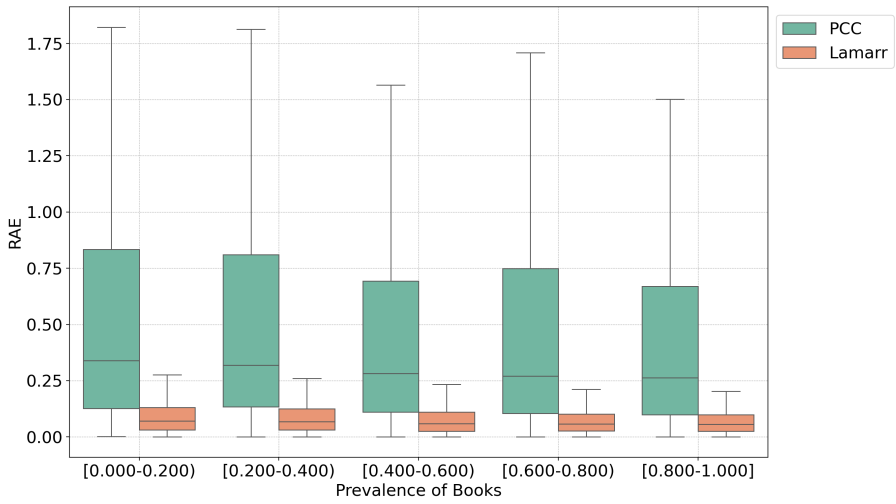

Fig. 7: A comparison of the errors produced by PCC and the top performer method Lamarr, at different proportions of book and electronics domains.

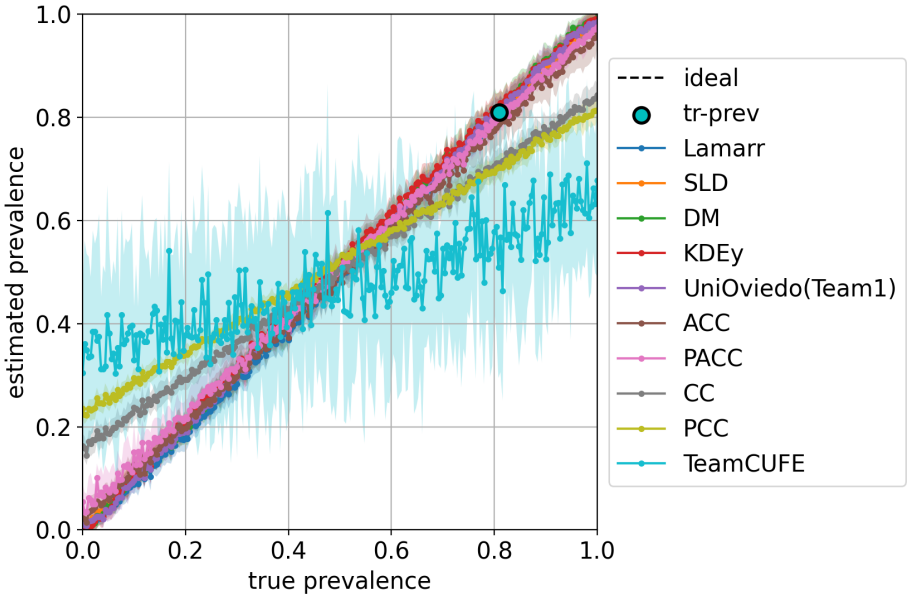

Fig. 8: Diagonal plot for T4.

Another important point concerns the difficulty in simulating meaningful levels of prior shift in ordinal problems. Plausible ordinal distributions impose certain constraints (e.g., smoothing requirements) that conflict with protocols designed to artificially alter them. In future editions, we plan to explore more sophisticated mechanisms to challenge participant systems with more abrupt shifting conditions in ordinal problems.

Looking back, we believe that using a sentiment-specific feature extractor reduced the impact of the covariate-shift effect introduced in task T4. For future editions, we plan to incorporate alternative representation mechanisms that better account for the controlled mixture of domains.

To conclude, we believe that LeQua 2024 has provided participating teams with the opportunity to stress-test their cutting-edge systems in a controlled setting, offering valuable insights to the community.

Acknowledgments

This work has been supported by the SoBigData++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1, and by the SoBigData.it and FAIR projects funded by the European Commission under the NextGenerationEU program. This work has also been funded by the QuaDaSh project (P2022TB5JF) "Finanziato dall'Unione europea —Next Generation EU, Missione 4 Componente 2 CUP B53D23026250001".

The authors' opinions do not necessarily reflect those of the European Commission.

References

1. Alexandari, A., Kundaje, A., Shrikumar, A.: Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In: Proceedings of the 37th International Conference on Machine Learning (ICML 2020). pp. 222–232. Virtual Event (2020)
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., Moreo, A., Sebastiani, F., Senz, M.: Regularization-based methods for ordinal quantification. arXiv:2310.09210 [cs.LG] (2023)
4. Bunse, M., Morik, K.: Unification of algorithms for quantification and unfolding. In: Proceedings of the 2nd International Workshop on Learning to Quantify (LQ 2022). pp. 1–10. Grenoble, IT (2022)
5. Card, D., Smith, N.A.: The importance of calibration for estimating proportions from annotations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2018). pp. 1636–1646. New Orleans, US (2018). https://doi.org/10.18653/v1/n18-1148
6. Castaño, A., Alonso, J., González, P., Pérez, P., del Coz, J.J.: QuantificationLib: A python library for quantification and prevalence estimation. SoftwareX **26**, 101728 (2024)
7. Castaño, A., Alonso, J., González, P., del Coz, J.J.: An equivalence analysis of binary quantification methods. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI-23). pp. 6944–6952. Washington, US (2023)
8. Castaño, A., González, P., González, J.A., del Coz, J.J.: Matching distributions algorithms based on the Earth Mover's Distance for ordinal quantification. IEEE Transactions On Neural Networks and Learning Systems (2022). https://doi.org/10.1109/TNNLS.2022.3179355, forthcoming
9. Castaño, A., González, P., González, J.A., del Coz, J.J.: Matching distributions algorithms based on the Earth mover's distance for ordinal quantification. IEEE Transactions on Neural Networks and Learning Systems **35**(1), 1050–1061 (2024). https://doi.org/10.1109/TNNLS.2022.3179355
10. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: ELECTRA: Pre-training text encoders as discriminators rather than generators. In: Proceedings of the 8th International Conference on Learning Representations (ICLR 2020). Addis Ababa, ET (2020), `https://openreview.net/pdf?id=r1xMH1BtvB`
11. Donyavi, Z., Li, F., Batista, G.: Ensemble Learning to Quantify: The CSE UNSW Team at LeQua 2024. In: Working Notes of the Learning to Quantify: Methods and Applications (LQ 2024) workshop, co-located at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2024), Vilnius, Lituania. Vilnius, LI (2024)
12. Donyavi, Z., Serapio, A., Batista, G.: MC-SQ: A highly accurate ensemble for multi-class quantification. In: Proceedings of the 23rd SIAM International Conference on Data Mining (SDM 2023). pp. 622–630. Minneapolis, US (2023). https://doi.org/10.1137/1.9781611977653.ch70

13. Dussap, B., Blanchard, G., Chérief-Abdellatif, B.: Label shift quantification with robustness guarantees via distribution feature matching. In: Proceedings of the 34th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML / PKDD 2023). pp. 69–85. Torino, IT (2023). https://doi.org/10.1007/978-3-031-43424-2_5
14. 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
15. 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), Article 19 (2021). https://doi.org/10.1145/3433164
16. Esuli, A., Moreo, A., Sebastiani, F., Sperduti, G.: A concise overview of LeQua 2022: Learning to quantify. In: Proceedings of the 13th International Conference of the CLEF Association (CLEF 2022). pp. 362–381. Bologna, IT (2022). https://doi.org/10.1007/978-3-031-13643-6_23
17. 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)
18. Firat, A.: Unified framework for quantification (2016), arXiv:1606.00868v1 [cs.LG] 2 Jun 2016
19. 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
20. Forman, G.: BNS feature scaling: An improved representation over TF.IDF for SVM text classification. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008). pp. 263–270. Napa Valley, US (2008)
21. Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowledge Discovery **17**(2), 164–206 (2008). https://doi.org/10.1007/s10618-008-0097-y
22. 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
23. González, P., Moreo, A., Sebastiani, F.: Binary quantification and dataset shift: An experimental investigation. Data Mining and Knowledge Discovery **38**(4), 1670–1712 (2024). https://doi.org/10.1007/s10618-024-01014-1
24. González, P., Moreo, A., Sebastiani, F.: Binary quantification and dataset shift: an experimental investigation. Data Mining and Knowledge Discovery pp. 1–43 (2024)
25. Kawakubo, H., Du Plessis, M.C., Sugiyama, M.: Computationally efficient class-prior estimation under class balance change using energy distance. IEICE Transactions on Information and Systems **99**, 176–186 (2016)
26. Kloos, K.: UniLeiden at LeQua2024: Evaluating Continuous Sweep and Comparison Using Underlying Classifiers. In: Working Notes of the Learning to Quantify: Methods and Applications (LQ 2024) workshop, co-located at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2024), Vilnius, Lituania. Vilnius, LI (2024)
27. Kloos, K., Karch, J.D., Meertens, Q.A., de Rooij, M.: Continuous Sweep: An improved, binary quantifier. arXiv:2308.08387 [stat.ML] (2023)

28. Lotz, T., Bunse, M.: Lamarr at LeQua2024: Regularized Soft-Max Likelihood Maximization. In: Working Notes of the Learning to Quantify: Methods and Applications (LQ 2024) workshop, co-located at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2024), Vilnius, Lituania. Vilnius, LI (2024)
29. Luth, L., Daniel, O., Gomes, G., Maletzke, A.: UniOeste at LeQua 2024: Combining the top-ranked quantifiers. In: Working Notes of the Learning to Quantify: Methods and Applications (LQ 2024) workshop, co-located at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2024), Vilnius, Lituania. Vilnius, LI (2024)
30. Maletzke, A., Moreira dos Reis, D., Cherman, E., Batista, G.: DyS: A framework for mixture models in quantification. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019). pp. 4552–4560. Honolulu, US (2019). https://doi.org/10.1609/aaai.v33i01.33014552
31. Moreno-Torres, J.G., Raeder, T., Alaíz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (2012). https://doi.org/10.1016/j.patcog.2011.06.019
32. 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
33. Moreo, A., Francisco, M., Sebastiani, F.: Multi-label quantification. ACM Transactions on Knowledge Discovery and Data **18**(1), Article 4 (2023). https://doi.org/10.1145/3606264
34. Moreo, A., González, P., del Coz, J.J.: Kernel density estimation for multiclass quantification. arXiv:2401.00490 [cs.LG] (2023). https://doi.org/10.48550/arXiv.2401.00490
35. Moreo, A., González, P., del Coz, J.J.: Kernel density estimation for multiclass quantification (2024), `https://arxiv.org/abs/2401.00490`
36. Moreo, A., González, P., del Coz, J.J.: Kernel density estimation for multiclass quantification. Machine Learning (2024), forthcoming
37. Moreo, A., Sebastiani, F.: Re-assessing the "classify and count" quantification method. In: Proceedings of the 43rd European Conference on Information Retrieval (ECIR 2021). vol. II, pp. 75–91. Lucca, IT (2021). https://doi.org/10.1007/978-3-030-72240-1_6
38. 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
39. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 12th Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). pp. 1532–1543. Doha, QA (2014)
40. Pérez-Mon, O., González, P.: UniOvi Team at LeQua 2024: Quantification via Gaussian Latent Space Representations. In: Working Notes of the Learning to Quantify: Methods and Applications (LQ 2024) workshop, co-located at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2024), Vilnius, Lituania. Vilnius, LI (2024)
41. Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N.D. (eds.): Dataset shift in machine learning. The MIT Press, Cambridge, US (2009). https://doi.org/10.7551/mitpress/9780262170055.001.0001
42. Saerens, M., Decaestecker, C.: Decision-making with unknown priors in supervised classification (2010), unpublished manuscript

43. 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
44. Sakai, T.: Comparing two binned probability distributions for information access evaluation. In: Proceedings of the 41st International ACM Conference on Research and Development in Information Retrieval (SIGIR 2018). pp. 1073–1076. Ann Arbor, US (2018). https://doi.org/10.1145/3209978.3210073
45. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods. arXiv preprint arXiv:2103.03223 (2021)
46. Sebastiani, F.: Evaluation measures for quantification: An axiomatic approach. Information Retrieval Journal **23**(3), 255–288 (2020). https://doi.org/10.1007/s10791-019-09363-y
47. Smith, N.A., Tromble, R.W.: Sampling uniformly from the unit simplex. Tech. rep., Johns Hopkins University (2004), `https://www.cs.cmu.edu/~nasmith/papers/smith+tromble.tr04.pdf`
48. Tasche, D.: Class prior estimation under covariate shift: No problem? In: Proceedings of the 2nd International Workshop on Learning to Quantify (LQ 2022). pp. 11–26. Grenoble, IT (2022)
49. Vapnik, V.: Statistical learning theory. Wiley, New York, US (1998)
50. Werman, M., Peleg, S., Rosenfeld, A.: A distance metric for multidimensional histograms. Computer Vision, Graphics, and Image Processing **32**, 328–336 (1985)

UniLeiden at LeQua2024: Evaluating Continuous Sweep and Comparison Using Underlying Classifiers

Kevin Kloos

University of Leiden, NL

Abstract. We compared the continuous sweep quantifier with median sweep, SLD, and DyS using Task T1 of the LeQua2024 competition data. We fitted 100 different underlying support vector machines and evaluated all quantifiers on all models. Continuous sweep is outperformed by SLD and DyS if a well-performing underlying classifier exists. Using worse quantifiers, continuous sweep and median sweep appeared to be more stable than DyS and SLD. We compared these findings with existing results in quantification literature.

1 Introduction

Quantification Learning is a task that is focused on predicting the prevalence of a data set rather than individually labelling each observation [1, 2]. In earlier years, quantification has been a side product of classification, but over the last decades, quantification learning has been developed to a standalone field with sophisticated tasks and methods. One of those tasks are the LeQua competitions which are designed to evaluate and compare quantifiers with each other. The LeQua competitions contains various tasks regarding quantification learning [3]. In this paper, we focus on task T1 of the LeQua2024 competition. This task concerns binary quantifiers from which the data is affected by prior-probability shift.

In this paper, we compare the continuous sweep quantifier with median sweep, SLD, and DyS. All quantifiers use a support vector machine with a radial basis function as an underlying classifier. We not only investigate the raw performance of the quantifiers, but also investigate the relationship with the underlying classifiers. We finalize the paper with a discussion about the competition and a generalization to other studies.

2 Methods

In this study, we compare our four quantifiers. These four quantifiers need an underlying classifier to compute probabilities for each observation. In this section, we explain how we computed the underlying classifier and elaborate on the

four quantifiers. Extra emphasis is put on the Continuous Sweep quantifier, because the author of this paper is one of the developers of Continuous Sweep and it is the most unknown quantifier out of the four. Moreover, we submitted the prevalences extracted from Continuous Sweep to the LeQua2024 competition.

Underlying classifier The underlying classifier of all quantifiers is a support vector machine using a radial basis function. Using the kernlab package supported by the Tidymodels library [4], we can tune this support vector machine with two hyperparameters: the cost (C) and the radial basis function sigma (σ). Moreover, the data is preprocessed by normalizing all predictor variables. A max entropy grid is used to find 100 pairs of hyperparameters that optimally cover the hyperparameterspace. We therefore fit 100 SVMs with different hyperparameters. Using the SVM, we can predict the probabilties of observations that belong to the positive class. With 5-fold cross validation, we also estimate the probabilties of all observations in the training data, which will be used to fit the quantifiers.

Median Sweep (MS) Median sweep is an ensemble quantifier in the group *Classify, Count, and Correct* [5]. First, we use the SVM to compute probabilities of all observations in the test data. Accordingly, we compute an adjusted count estimate for every probability that occurs in the test data. The adjusted count estimates are computed using true and false positive rate estimated by the cross validated probabilities from the training data. Consequently, we discard unreliable estimates, that is, every adjusted count estimate where the difference between the true and false positive rates are smaller than 0.25. Finally, we compute the median of the remaining adjusted count estimates as our median sweep prevalence estimate.

Continuous Sweep (CS) Continuous sweep is a quantifier that is similar to median sweep [6]. More details of Continuous Sweep can be found in [6], but we provide a short explanation of the method. Continuous sweep and median sweep are different in two characteristics. First, continuous sweep uses continuous functions to estimate the true and false positive rates. In this task, we used the kernel cumulative density function from R's *ks* package to compute the true and false positive rates with the cross-validated training data. The continuous functions of the true and false positive functions enabled us to integrate the adjusted count estimates with respect to the estimated probabilities. Therefore, the second change is that we compute the mean of all adjusted count estimates using integration, instead of computing the median. Moreover, we were able to improve the procedure of discarding unreliable adjusted count estimates. We derived derivations for the bias and variance of continuous sweep which enabled us to optimize the value of the minimal difference between true and false positive rates (i.e., p^{Δ}) that an adjusted count estimate is determined as "reliable". The reliable prevalence estimates are located between θ_l and θ_r. The final continuous

sweep estimate is the mean area under the curve of the adjusted count function between θ_l and θ_r.

SLD The SLD algorithm is a expectation maximization approach to the quantification task [7]. The prevalence of the training data is used as a starting point to update the test prevalence iteratively by an optimal Bayes classifier until convergence is reached. The stopping criterium is a prevalence difference of 0.0001 between two updates or a maximum of 1000 iterations.

DyS The DyS algorithm is algorithm that matches histograms to find the prevalence of a test set [8]. First, the cross validated probabilities of the training set are used to construct histograms of the probabilities in the positive and negative class. Moreover, the probabilities of the test set are used to make a histogram for the test data. Using a tenary search, we aim to find the optimal mixture between the positive and negative class probabilities to match the test histogram as good as possible. The measure we used to describe the difference between histogram is the hellinger distance. Moreover, default values of 8 bins are used to construct the histograms and a the stopping criterium is a prevalence difference of 0.0001 between two updates.

3 Results

We fitted 100 different SVM-RBF models with different hyperparameters using a max entropy grid. Out of these 100 models, 76 converged. We compared the performance of the four quantifiers with 1) their best MAE overall, and 2) using the model with the lowest ROC-AUC. From Table 1, we see that SLD has the lowest MAE out of the four models in the two scenarios. Moreover, the differences between the minimum MAE with the MAE of the model with the lowest ROC-AUC are small across the quantifiers.

We also compared the performance of the classifiers against their quantification performance, both within as between quantifiers. For all 76 converged models, we evaluated the ROC-AUC of the classifier against the cross-validated probabilities from the training data and we computed the MAE of all four quantifiers on each of the models. Two models had a ROC lower than 0.8, which means that those models perform very bad compared to the other models. Those two models are excluded from the illustrations. In Figure 1, we see that the best classifiers are embraced by all quantifiers. SLD and DyS perform better than CS and MS using the best classifiers. However, these two quantfiers are less stable if the classifiers perform worse. For a ROC-AUC around $[0.90 - 0.92]$, SLD had a remarkable drop in performance. Moreover, out of the 76 converged models, SLD had a MAE lower than 0.05 for 22 models, whereas DyS, MS, and CS had a MAE lower than 0.05 for 47, 45, and 44 respectively. For classifier with a low ROC-AUC, CS and MS outperformed SLD and DyS, but the MAE's respectively are fairly high compared to the good classifiers. Moreover, we extract from Figure 1 that DyS performed either good or very bad for worse classifiers.

Table 1. The MAE of four quantifiers evaluated on 1000 development sets. At the top, the lowest MAE of every quantifier across all models. At the bottom, the MAE of every quantifier based on the largest ROC-AUC. The columns of the table denote the quantifier, the model number, the cost hyperparameter (C), the rbf sigma hyperparameter (σ), the ROC-AUC metric of the model, and the MAE of the quantifier given the classifier.

Quantifier	Model no.	C	σ	ROC-AUC	MAE
Continuous Sweep	77	4.436	3.28×10^{-4}	0.9177	*0.0241*
Median Sweep	74	0.236	3.68×10^{-3}	0.9099	0.0267
SLD	77	4.436	3.28×10^{-4}	0.9177	**0.0205**
DyS	77	4.436	3.28×10^{-4}	0.9177	0.0223
Continuous Sweep	78	25.1	4.45×10^{-4}	0.9198	*0.0246*
Median Sweep	78	25.1	4.45×10^{-4}	0.9198	0.0291
SLD	78	25.1	4.45×10^{-4}	0.9198	**0.0209**
DyS	78	25.1	4.45×10^{-4}	0.9198	0.0224

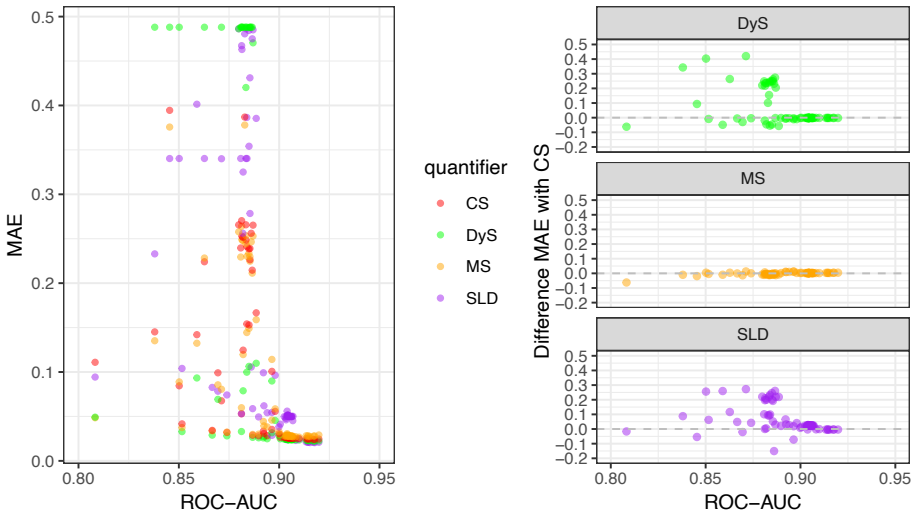

Fig. 1. Plot that compares the MAE of he four quantifiers against the ROC-AUC. On the left, we compare the MAE against the ROC-AUC in general for all models. On the left, we compare the difference between the MAE of the displayed quantifier against Continuous Sweep, where a value higher than zero indicates a higher MAE for the respective quantifier.

4 Discussion

Our quantifier, Continuous Sweep, underperforms against good baseline quantifiers like SLD and DyS in the LeQua2024 competition. If a good classifier can be obtained from the training data, SLD and DyS performed well. Similar results have been found in the previous LeQua2022 competition, where DyS and SLD were good baseline classifiers [3]. If the underlying classifier performs worse, the ensemble methods such as Continuous Sweep and Median Sweep had a lower performance drop than, and outperformed, SLD and DyS. These findings could be translated to the elaborative comparison of [9], where median sweep generally outperformed SLD and DyS for datasets with a binary target variable. These datasets used were usually much smaller than the datasets of the LeQua competitions. Whereas the LeQua competition contained a training set of 5000 observations with 1000 development sets containing 250 observations, the datasets used in [9] were smaller baseline UCI or Kaggle datasets only splitted into a training and a test set.

In future research, it could be interesting to investigate the performance of quantifiers when it is difficult to construct a good (probabilistic) classifier. Moreover, it could be interesting to investigate whether some decision rules could be developed to choose a suitable quantifier based on the characteristics of an underlying classifier. For example, if one has an excellent classifier, we could safely apply SLD, whereas a worse classifier might be better used by a quantifier like continuous sweep. Last, it might also be interesting to investigate the performance of direct learners when it is difficult to construct a good classifier.

References

1. González P, Castaño A, Chawla NV, del Coz JJ. A Review on Quantification Learning. ACM Computing Surveys. 2017;50(5):74:1-74:40.
2. Esuli A, Fabris A, Moreo A, Sebastiani F. Learning to Quantify. 1st ed. The Information Retrieval Series. Cham: Springer Nature; 2023.
3. Esuli A, Moreo A, Sebastiani F, Sperduti G. A Detailed Overview of LeQua@ CLEF 2022: Learning to Quantify; 2022. Available from: `https://api.semanticscholar.org/CorpusID:251471941`.
4. Kuhn M, Wickham H. Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles; 2020. Available from: `https://www.tidymodels.org`.
5. Forman G. Quantifying Counts and Costs via Classification. Data Mining and Knowledge Discovery. 2008;17(2):164-206.
6. Kloos K, Karch JD, Meertens QA, de Rooij M. Continuous Sweep: an improved, binary quantifier; 2023. Available from: `https://arxiv.org/abs/2308.08387`.
7. Saerens M, Latinne P, Decaestecker C. Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure. Neural Computation. 2002;14(1):21-41.
8. Maletzke A, dos Reis D, Cherman E, Batista G. DyS: A Framework for Mixture Models in Quantification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33; 2019. p. 4552-60.
9. Schumacher T, Strohmaier M, Lemmerich F. A Comparative Evaluation of Quantification Methods; 2023.

Ensemble Learning to Quantify: The CSE UNSW Team at LeQua 2024

Zahra Donyavi, Feiyu Li, and Gustavo Batista

University of New South Wales, Sydney, Australia
{z.donyavi, feiyu.li, g.batista}@unsw.edu.au

Abstract. LeQua 2024 is a data challenge to facilitate the comparative evaluation of quantification methods for class-prior estimation, also known as quantification or learning to quantify. The challenge focuses on training predictors, termed "quantifiers," to estimate the relative frequencies of classes within sets of unlabeled data points. Notably, the datasets are affected by class prevalence shifts, exhibiting prevalences in the test set that differ from the training set. We propose two ensemble methods, Multiple Classifiers - Single Quantifier (MC-SQ) and Single Classifier - Multiple Quantifiers (SC-MQ), for binary and multi-class quantification tasks. Additionally, we introduce EMQ-ini, a new variation of the Expectation–Maximization algorithm for Quantification (EMQ) method. This variation uses the predicted target prior from the quantifier Generalized Probabilistic Adjusted Classify & Count (GPACC) as the initial point of log-likelihood maximization. We use EMQ-ini as one of the base quantifiers of SC-MQ. Our MC-SQ method ranked first in Mean Relative Absolute Error (MRAE), the official competition performance measure, and second in Absolute Error (AE) on the binary quantification task. Our SC-MQ method ranked third in MRAE and first in AE for the multi-class quantification task.

Keywords: Prevalence estimation · Target prevalence shift · Quantification.

1 Introduction

Quantification learning is used in many real-world scenarios where the objective is to predict the behavior of groups. It is particularly useful in sentiment analysis, which tracks how overall opinions about products, people, or organizations change over time [1]. Instead of classifying individual behaviors, quantification focuses on estimating the distribution of opinions, providing insights into broader trends in public sentiment.

The simplest quantification approach, known as *Classify & Count* (CC), directly applies classification to quantification problems. However, this method suffers from systematic bias in which the error increases linearly as we approach the more skewed class distributions [2]. To address this, researchers have proposed novel quantification methods to provide more accurate estimates of class distributions in the presence of class prevalence shifts.

LeQua 2024 is a competition that challenges participants to evaluate various techniques for binary, multi-class, and ordinal quantification tasks using real-world Amazon product review datasets. The challenge encompasses four tasks:

Task T1 evaluates binary quantifiers on data affected by prior probability shift (label shift), akin to Task T1A of LeQua 2022.

Task T2 assesses single-label multi-class quantifiers operating on data points belonging to one of $L > 2$ classes, with data affected by prior probability shift, similar to Task T1B of LeQua 2022.

Task T3 new to LeQua 2024, evaluates ordinal quantifiers handling a set of $L > 2$ ordered classes, also involving data affected by prior probability shifts.

Task T4 another new addition, evaluates binary quantifiers on data affected by covariate shifts.

Our contributions focus on Tasks T1 and T2. For the binary quantification task T1, we employ *Multiple Classifiers - Single Quantifier* (MC-SQ) [3], an ensemble method that achieved the top rank in MRAE. MC-SQ leverages multiple classifiers combined with a single quantifier. For the multi-class quantification task T2, we introduce *Single Classifier - Multiple Quantifiers* (SC-MQ), an ensemble approach that secured the third rank in MRAE. SC-MQ combines a single classifier with multiple quantifiers, including our proposed *Expectation-Maximization Quantifier with Initialization Adaptation* (EMQ-ini) method.

The performance of our ensemble methods can be attributed to the combination of multiple classifiers and quantifiers, leveraging the strengths of diverse models while mitigating individual weaknesses. These ensemble approaches enhance overall quantification accuracy. Furthermore, extensive hyperparameter tuning optimized the performance of each component within the ensembles, contributing to the top-ranking results.

This paper is organized as follows: Section 2 describes the classification and quantification methods employed in our ensemble approaches, MC-SQ and SC-MQ. Section 3 details the evaluation process and comprehensive hyperparameter tuning strategy. Finally, Section 4 concludes our work and presents directions for future research.

2 Methods

This section describes our proposed ensemble approaches, MC-SQ and SC-MQ, employed for the binary quantification task (T1) and multi-class quantification task (T2).

2.1 MC-SQ Approach for T1

For T1, we employ MC-SQ, an ensemble approach that previously achieved the top rank in T1B for the LeQua 2022 competition, based on our post-competition experiments [3]. Figure 1 illustrates the MC-SQ architecture, which consists of

an ensemble of six pairs of classifiers and quantifiers. We introduce diversity by varying the base classifiers while keeping the base quantifier fixed. The chosen quantifier is *Distribution y-Similarity* (DyS) [4] as it has been recognized as one of the top-performing quantifiers for binary problems in the comparative study conducted by [5].

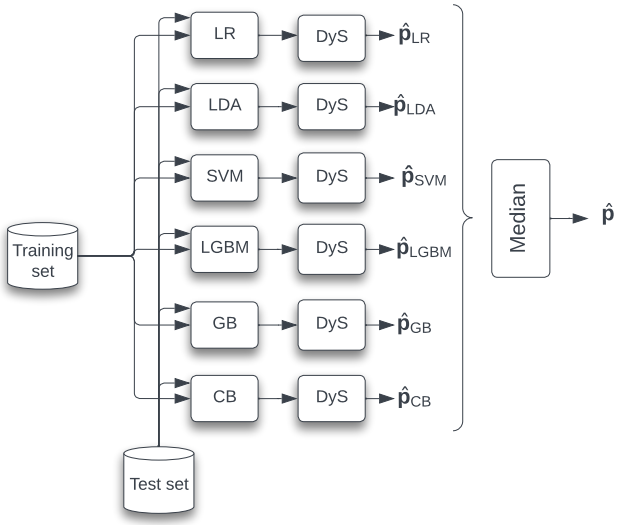

Fig. 1. Schematic of the proposed MC-SQ ensemble approach.

Our approach encompasses the following classifiers: Logistic Regression (LR), Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Light Gradient Boosting Machines (LGBM), Gradient Boosting (GB), and CatBoost (CB). The rationale behind selecting these algorithms stems from their diverse learning paradigms and their frequent success in various Machine Learning applications. As initially proposed, MC-SQ [3] also incorporates Random Forest and Naive Bayes as base classifiers. However, an extensive evaluation of the competition dataset showed that pairing these classifiers with DyS exhibited higher quantification error rates on the validation data. Consequently, we removed them from our ensemble and introduced CatBoost to maintain diversity among the base classifiers while leveraging its strength in quantification tasks.

We conduct comprehensive hyperparameter tuning for each classifier and quantifier combination to optimise our ensemble approach. This process systematically explores the parameters space to identify the configurations that yielded the best quantification performance on the validation set. Section 3 provides fur-

ther details on our methodology and experimental setup, including the comprehensive hyperparameter tuning process employed to optimize the performance of our ensemble approach.

2.2 SC-MQ Approach for T2

For the T2 task, we introduce SC-MQ, an ensemble approach that combines diverse quantification algorithms with a single base classifier. We evaluated various classifiers as potential base classifiers by measuring the quantification error over the validation set. Our analysis showed that logistic regression (LR), as the base classifier, coupled with the assessed quantifiers, yielded the lowest quantification error on the validation set. Consequently, we selected LR as the base classifier for our ensemble, aiming to optimize overall quantification performance. By fixing the *Single Classifier* as Logistic Regression, we aim to minimize the variability introduced by the classification component. This allows the quantification algorithms to be the primary source of diversity within the ensemble. Figure 2 shows the architecture of our proposed SC-MQ method.

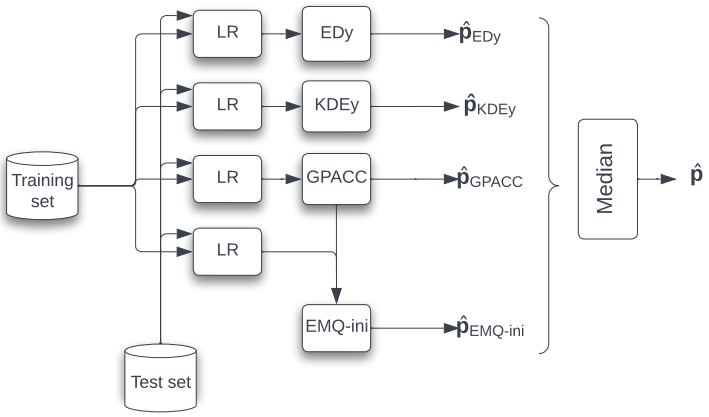

Fig. 2. Schematic of the proposed SC-MQ ensemble approach.

We employ four quantification methods in our experiments: *Energy Distance* (EDy) [6], *Kernel Density Estimation* (KDEy) [7], *Generalized Probabilistic Adjusted Classify & Count* (GPACC) [8], and our newly proposed EMQ-ini. We select these methods based on our prior knowledge of quantification techniques and their ability to handle multi-class problems. Each algorithm represents a distinct approach to quantification. For KDEy, we specifically utilize the Maximum

Likelihood (ML) variation, as it showed the best performance for multi-class datasets among all KDEy variants in the original study [7].

Complementing these established quantifiers, we introduce EMQ-ini, a novel variation of the EMQ method [9] we are currently working on. EMQ-ini uses the predicted target priors from GPACC as the initial point of log-likelihood maximization. This competition provides an opportunity to test our idea and evaluate the performance of EMQ-ini on the quantification tasks in a real-world setting. We discuss the EMQ-ini method in detail in the next section.

2.3 EMQ-ini

EMQ-ini is a new quantification method designed to enhance the performance of traditional EM-based quantifiers. It leverages an initial estimate from GPACC to provide a more informed starting point for the EM process, improving convergence speed and accuracy, especially in scenarios where class distribution shifts are significant between the training and target domains.

Alg. 1 provides a detailed description of EMQ-ini. Suppose we have $\mathbf{y} = \{y_i\}_{i=1}^{L}$ as the set of labels, and $\mathbf{x}$ is the unlabelled test set with N instances sampled from the target domain. EMQ-ini takes three inputs: $\hat{p}_{ini}(\mathbf{y})$, which is an estimate of the target prevalence from GPACC, used as an initiation for EM; $p_t(\mathbf{y})$, as the class prevalence from a training set; and $\hat{p}_t(\mathbf{y}|\mathbf{x})$ is the estimate posterior class probabilities (*scores*) from a classifier trained on training set sampled from the source domain.

EMQ-ini updates the scores before applying the EM iterations. This update is based on the Bayes' rule, similar to the *E step* in EMQ. It begins by computing the prevalence ratio $\mathbf{r}$, which is the element-wise division of prevalence estimate $\hat{p}_{ini}(\mathbf{y})$ by the training prevalence $\hat{p}_t(\mathbf{y})$. This ratio is then used to compute the scores $\hat{p}_{ini}(\mathbf{y}|\mathbf{x})$.

The main iterative process of the EMQ-ini is similar to EMQ and consists of the following steps:

- **E-step**: In this step, the algorithm computes the updated posterior probabilities $\hat{p}^{(s)}(\mathbf{y}|\mathbf{x})$ for the current iteration s. This is done by adjusting the initial posterior probabilities $\hat{p}_{ini}(\mathbf{y}|\mathbf{x})$ based on the ratio of the current prevalence estimate $\hat{p}^{(s)}(\mathbf{y})$ to the initial prevalence estimate $\hat{p}_{ini}(\mathbf{y})$, using Bayes' rule.
- **M-step**: In this step, the algorithm updates the prevalence estimate $\hat{p}^{(s+1)}(\mathbf{y})$ for the next iteration $s+1$. This is done by taking the average of the updated posterior probabilities $\hat{p}^{(s)}(\mathbf{y}|\mathbf{x})$ across all instances in the dataset.

The E-step and M-step are iteratively performed until a stopping condition is met (e.g., convergence or maximum iterations reached).

After the iterative process, the final prevalence estimate $\hat{p}(\mathbf{y})$ is set to the last computed prevalence estimate $\hat{p}^{(s)}(\mathbf{y})$.

Algorithm 1: EM Quantifier with Initialization Adaptation.

Input: $\hat{p}_{ini}(\mathbf{y})$, $p_t(\mathbf{y})$, $\hat{p}_t(\mathbf{y}|\mathbf{x})$
Output: $\hat{p}(\mathbf{y})$
Prevalence ratio: $\mathbf{r} \leftarrow \dfrac{\hat{p}_{ini}(\mathbf{y})}{p_t(\mathbf{y})}$;
Updated posterior: $\hat{p}_{ini}(\mathbf{y}|\mathbf{x}) \leftarrow \left[\dfrac{r_i \cdot \hat{p}_t(y_i|\mathbf{x})}{\sum_{j=1}^{L} r_j \cdot \hat{p}_t(y_j|\mathbf{x})} \textbf{ for } i \leftarrow 1 \textbf{ to } L\right]$;
State: $s \leftarrow 0$;
Target prevalence estimate in state s: $\hat{p}^{(s)}(\mathbf{y}) \leftarrow \hat{p}_{ini}(\mathbf{y})$;
while *stopping condition* = *false* **do**
  E step: $\hat{p}^{(s)}(\mathbf{y}|\mathbf{x}) \leftarrow \left[\dfrac{\frac{\hat{p}^{(s)}(y_i)}{\hat{p}_{ini}(y_i)} \cdot \hat{p}_{ini}(y_i|\mathbf{x})}{\sum_{j=1}^{L} \frac{\hat{p}^{(s)}(y_j)}{\hat{p}_{ini}(y_j)} \cdot \hat{p}_{ini}(y_j|\mathbf{x})} \textbf{ for } i \leftarrow 1 \textbf{ to } L\right]$;
  M step: $\hat{p}^{(s+1)}(\mathbf{y}) \leftarrow \dfrac{1}{N} \sum_{x \in \mathbf{x}} \hat{p}^{(s)}(\mathbf{y}|x)$;
  $s \leftarrow s + 1$;
$\hat{p}(\mathbf{y}) \leftarrow \hat{p}^{(s)}(\mathbf{y})$;
return $\hat{p}(\mathbf{y})$;

3 Evaluation

The performance of the proposed methods is evaluated on both the validation and test sets for Tasks T1 and T2. Tables 1 and 3 present the results in terms of Mean Absolute Error (MAE) and Mean Relative Absolute Error (MRAE) for the validation and test sets, respectively.

For the binary quantification task T1, Table 1 compares the performance of the individual classifiers coupled with the DyS quantifier and the proposed MC-SQ ensemble method. Among the individual methods, LR-DyS and the MC-SQ ensemble achieved the best MAE of 0.0206 on both the validation and test sets. However, MC-SQ outperformed LR-DyS regarding MRAE, achieving the lowest scores of 0.0869 and 0.0981 on the validation and test sets, respectively.

The parameters for the classifiers and quantifiers used in the T1 task were selected using grid search and optimization to minimize the MRAE metric. This process was facilitated by the QuaPy library [10]. Table 2 presents the selected parameters obtained through this optimization procedure. Notable parameters include the regularization and gamma parameters for Logistic Regression and Support Vector Machines and the number of bins (nbins) used by the DyS quantifier, which varied across the different methods. Any other parameters or hyperparameters not explicitly mentioned in the parameters table are set to their respective default values as defined by the implemented methods.

For the multi-class quantification task T2, Table 3 compares the performance of the proposed SC-MQ ensemble with its individual components: EDy, EMQ-ini,

Table 1. Performance comparison of methods on validation and test sets for T1.

Method	Validation set		Test set	
	MAE	**MRAE**	**MAE**	**MRAE**
LR-DyS	**0.0206**	0.0910	**0.0206**	0.1024
LDA-DyS	0.0218	0.0987	0.0218	0.1121
SVM-DyS	0.0213	0.1023	0.0217	0.1026
LGBM-DyS	0.0253	0.1034	0.0255	0.1220
GB-DyS	0.0258	0.1052	0.0258	0.1176
CB-DyS	0.0233	0.0954	0.0233	0.1145
MC-SQ	**0.0206**	**0.0869**	**0.0206**	**0.0981**

Table 2. Classifier and Quantifier selected parameters for T1.

Method	**Classifier**	**Quantifier**
LR-DyS	C = 10, class-weight = balanced	nbins = 40
LDA-DyS	None	nbins = 30
SVM-DyS	C = 24.1967, gamma = 0.0114	nbins = 30
LGBM-DyS	None	nbins = 30
GB-DyS	None	nbins = 16
CB-DyS	Depth = 2, learning_rate = 0.1, l2_leaf_reg = 7, iterations = 900	nbins = 18

KDEy, and GPACC. The SC-MQ ensemble achieved the best performance, with MAE scores of 0.0129 and 0.0127 on the validation and test sets, respectively. It also obtained the lowest MRAE scores of 1.1160 and 1.0786 on the validation and test sets.

Table 3. Performance comparison of methods on validation and test sets for T2.

Method	Validation set		Test set	
	MAE	**MRAE**	**MAE**	**MRAE**
EDy	0.0137	1.3053	0.0135	1.2390
EMQ-ini	0.0139	1.1351	0.0137	1.1038
KDEy	0.0179	1.4542	0.0176	1.4355
GPACC	0.0155	1.2021	0.0155	1.1950
SC-MQ	**0.0129**	**1.1160**	**0.0127**	**1.0786**

The performance of SC-MQ can be attributed to the combination of a robust base classifier (Logistic Regression) with a diverse set of quantification algorithms, including the novel EMQ-ini method proposed in this work. Among the individual quantifiers, EMQ-ini was the best-performing single quantifier, outperforming other methods like EDy, KDEy, and GPACC. By leveraging the strengths of multiple quantifiers, with EMQ-ini being the strongest contributor while maintaining a consistent base classifier, the ensemble could effectively capture the diverse characteristics of the multi-class quantification problem.

Table 4 presents the selected parameters using the grid search and minimizing the MRAE for the classifier and quantifiers used in the T2 task. Notable parameters include the regularization parameter for Logistic Regression, the bandwidth parameter for KDEy, and the solver used by GPACC. Additionally, EMQ-ini utilized the exact training prevalences during the initialization step.

Table 4. Classifier and Quantifier selected parameters for T2.

Method	**Classifier**	**Quantifier**
EDy	C = 1, class-weight = balanced	None
KDEy	C = 100, class-weight = None	bandwidth = 0.14
GPACC	C = 0.1, class-weight = balanced	solver = minimize
EMQ-ini	C = 1, class-weight = None	exact_train_prev = True

4 Conclusion

The results demonstrate the effectiveness of the proposed ensemble approaches, MC-SQ and SC-MQ, in addressing the binary and multi-class quantification tasks, respectively. By combining diverse classifiers and quantifiers, these methods could leverage the strengths of individual components while mitigating their weaknesses, leading to improved quantification performance on both tasks.

For future work, we will investigate EMQ-ini in more depth, focusing on the effect of the initial point of maximum likelihood optimization on this method. Additionally, we will evaluate EMQ-ini on various datasets to ensure its generality and robustness across different scenarios.

References

1. A. Moreo and F. Sebastiani, "Tweet sentiment quantification: An experimental re-evaluation," *PLoS One*, vol. 17(9), 2022.
2. G. Forman, "Quantifying counts and costs via classification," *Data Min Knowl Discov*, vol. 17, no. 2, pp. 164–206, 2008.
3. Z. Donyavi, A. Serapio, and G. Batista, "Mc-sq: A highly accurate ensemble for multi-class quantification," in *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM, 2023, pp. 622–630.
4. A. Maletzke, D. dos Reis, E. Cherman, and G. Batista, "Dys: a framework for mixture models in quantification," in *AAAI Conference*, vol. 33, no. 01, 2019, pp. 4552–4560.
5. T. Schumacher, M. Strohmaier, and F. Lemmerich, "A comparative evaluation of quantification methods," *arXiv preprint arXiv:2103.03223*, 2021.
6. J. J. del Coz, "Unioviedo (team2) at lequa 2022: Comparison of traditional quantifiers and a new method based on energy distance." in *CLEF (Working Notes)*, 2022, pp. 1869–1874.

7. A. Moreo, P. González, and J. J. del Coz, "Kernel density estimation for multiclass quantification," *arXiv preprint arXiv:2401.00490*, 2023.
8. A. Firat, "Unified framework for quantification," *arXiv preprint arXiv:1606.00868*, 2016.
9. M. Saerens, P. Latinne, and C. Decaestecker, "Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure," *Neural computation*, vol. 14, no. 1, pp. 21–41, 2002.
10. A. Moreo, A. Esuli, and F. Sebastiani, "Quapy: a python-based framework for quantification," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4534–4543.

Lamarr at LeQua2024: Regularized Soft-Max Likelihood Maximization

Tobias Lotz and Mirko Bunse

Lamarr Institute for Machine Learning and Artificial Intelligence,
44227 Dortmund, Germany
{tobias.lotz,mirko.bunse}@cs.tu-dortmund.de

Abstract. As members of the Lamarr Institute, we participated in the open LeQua2024 competition. The goal in this competition was to predict the prevalences of classes in unlabeled sets of data, given a labeled training set. Our submission builds on the regularized maximization of a likelihood function with constraints that are implemented through a soft-max operator. Ultimately, this method ranked in the top three across all four disciplines of LeQua2024; most notably, we achieved the first place in discipline T4, a binary quantification task with covariate shift. In this paper, we detail our approach to the competition.

Keywords: Quantification · prior probability shift · covariate shift.

1 Introduction

LeQua2024[1] was a competition hosted for the evaluation of quantification methods. These methods estimate class prevalences in unlabeled sets of data, i.e., they estimate how often each class appears in each data set [4]. To learn the correspondence between class labels and feature vectors, a labeled training set is provided. Unlike classification, quantification is not concerned with predicting the label of each individual data item; what matters are aggregate predictions for sets of data items [5]. These predictions are complicated by shifts between the training distribution and the target distributions.

LeQua2024 consists of four disciplines that are separately evaluated. Each discipline represents a different quantification setting. Our team from the Lamarr Institute ranked in the top three across all disciplines.

T1 Binary quantification with prior probability shift (3^{rd} place).
T2 Multi-class quantification with prior probability shift (2^{nd} place).
T3 Ordinal quantification with prior probability shift (2^{nd} place).
T4 Binary quantification with covariate shift (1^{st} place).

In Sec. 2, we introduce the quantification method that we used across all disciplines. Sec. 3 details our optimization of the method's hyper-parameters. We conclude with Sec. 4.

[1] See https://lequa2024.github.io

2 Method

Let C be the number of classes and let $\mathcal{P} = \{\mathbf{p} \geq 0 : 1 = \sum_{i=1}^{C} \mathbf{p}_i\}$ be the set of valid class prevalence vectors. Making a prediction, such that $\hat{\mathbf{p}}_i$ is an estimate of $\mathbb{P}(Y = i)$ in set unlabeled data items, can be realized through the constrained minimization of a loss function $\ell : \mathcal{P} \to \mathbb{R}$,

$$\hat{\mathbf{p}} \;=\; \underset{\mathbf{p} \in \mathcal{P}}{\arg\min}\; \ell(\mathbf{p}) \tag{1}$$

where ℓ is typically defined in terms of an unlabeled data set $\mathcal{D} = \{x \in \mathcal{X}\}$.

Our approach to all disciplines of LeQua2024 evolves around three aspects: the choice of the loss function, the implementation of the $\mathbf{p} \in \mathcal{P}$ constraint, and regularization. Regularization is the only aspect through which we adapt our method to the particularities of each discipline.

Loss Function We employ the negative log-likelihood loss proposed by Alexandari et al. [1]. This choice is motivated by an exceptional performance recently reported for this loss function in combination with kernel density estimates [6] and by the winning performance of a closely related method in the first edition of LeQua in 2022 [7]. We opted against kernel densities because our initial experiments did not verify a performance improvement due to them.

Constraints We ensure that $\hat{\mathbf{p}} \in \mathcal{P}$ by optimizing over a latent vector $\mathbf{l} \in \mathbb{R}^{C-1}$ that we feed through a soft-max operator $\sigma : \mathbb{R}^{C-1} \to \mathcal{P}$ [2].

Regularization In T3, we promote solutions that we deem ordinally plausible [3]. In all other disciplines, we promote uniform solutions, assuming that the testing protocol of the competition exhibits a slight preference towards these outcomes. The impact of regularization is controlled through a hyper-parameter $\tau \geq 0$.

Combining all of the above aspects, our prediction $\hat{\mathbf{p}} = \sigma(\hat{\mathbf{l}}) \in \mathcal{P}$ is obtained after optimizing

$$\hat{\mathbf{l}} \;=\; \underset{\mathbf{l} \in \mathbb{R}^{C-1}}{\arg\min}\; -\sum_{i=1}^{C} \log \sum_{x \in \mathcal{D}} \frac{\hat{\mathbb{P}}(Y = i \mid X = x)}{\hat{\mathbb{P}}(Y = i)} \cdot [\sigma(\mathbf{l})]_i + r(\mathbf{l}) \tag{2}$$

$$\text{where} \quad [\sigma(\mathbf{l})]_i \;=\; \begin{cases} \frac{1}{1+\sum_{j=1}^{C-1} \exp(\mathbf{l}_j)} & \text{if } i = 1 \\ \frac{\exp(\mathbf{l}_i)}{1+\sum_{j=1}^{C-1} \exp(\mathbf{l}_j)} & \text{else} \end{cases}$$

$$\text{and} \quad r(\mathbf{l}) \;=\; \begin{cases} \tau \cdot \sum_{i=1}^{C-1} \left([\sigma(\mathbf{l})]_i - [\sigma(\mathbf{l})]_{i+1} \right)^2 & \text{for T1, T2, T4} \\ \tau \cdot \sum_{i=1}^{C-2} \left([\sigma(\mathbf{l})]_i - 2[\sigma(\mathbf{l})]_{i+1} + [\sigma(\mathbf{l})]_{i+2} \right)^2 & \text{for T3} \end{cases}$$

We estimate the posteriors $\hat{\mathbb{P}}(Y = i \mid X = x)$ through a multi layer perceptron (MLP) classifier, except for discipline T4, where a logistic regression outperformed the MLP on the validation set. The prior $\hat{\mathbb{P}}(Y = i)$ is estimated on the training set, which is also used to train the classifier within each discipline.

The numerical optimization of the loss function in Eq. 2 is realized through an unconstrained Newton conjugate gradient trust-region method [9], as it is implemented in the SciPy package [8]. The full implementation of our approach is publicly available on GitHub[2].

3 Hyper-Parameter Optimization

Each discipline of LeQua2024 provides, in addition to a labeled training set, a validation set for hyper-parameter optimization. This validation set consists of multiple sets $\mathcal{D}$ of data items, each sampled with a discipline-specific type of distribution shift. Performance evaluations on the validation data act as an estimate of the final performance on the test data; for the latter, the ground-truth remained hidden throughout the competition. Participants were able to choose those hyper-parameters that perform best during validation.

In order to optimize our hyper-parameter selection, we employed a coarse-to-fine grid-search adaptation strategy, starting from heuristically chosen starting grids. For numeric parameters, if the optimal value is located at the smallest or largest value in the grid, we shift it in such a way, that the currently optimal value now lies at the center. Otherwise, we decrease the difference of the candidate values, in order to allow finer updates to further improve the performance.

The final hyper-parameters that we selected for the different classifiers can be seen in tables 1 and 2. We note that regularization only had a minor impact on the validation performance of our method.

Table 1. Final hyper-parameters used for MLP classifiers.

task	hidden_layer_size	activation	alpha	learning_rate	solver	τ
T1	512	tanh	1e−1	1e−3	sgd	0
T2	512	tanh	1e−1	1e−5	adam	0
T3	320	tanh	1e−6	1e−3	adam	1e−3

Table 2. Final hyper-parameters used for logistic regressions.

task	C	class_weight	τ
T4	0.43571	None	1e−5

[2] See https://github.com/tobiaslotz/lequa2024

4 Conclusion

Our participation in LeQua2024 evolves around a maximum likelihood estimate with constraints that are implemented through a soft-max operator. We employed this estimate across all four disciplines of LeQua2024, with a discipline-specific regularization that only played a minor role on quantification performance. Our method achieved top-ranking results throughout the competition.

The reasons for this outcome—and their implications on future research—remain yet to be discussed. Two essential prerequisites for this discussion are *i)* specific information about the submissions of the other teams and *ii)* specific information about the pre-processing of LeQua's data. Lacking both prerequisites at the moment, we are looking forward to the conclusions that are to be drawn by LeQua's organizers.

References

1. Alexandari, A., Kundaje, A., Shrikumar, A.: Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In: Int. Conf. on Mach. Learn. pp. 222–232 (2020), http://proceedings.mlr.press/v119/alexandari20a.html
2. Bunse, M.: On multi-class extensions of adjusted classify and count. In: Int. Worksh. on Learn. to Quantify: Meth. and Appl. pp. 43–50 (2022), https://lq-2022.github.io/proceedings/CompleteVolume.pdf
3. Bunse, M., Moreo, A., Sebastiani, F., Senz, M.: Ordinal quantification through regularization. In: Europ. Conf. on Mach. Learn. and Knowl. Discov. in Databases. pp. 36–52. Springer (2022). https://doi.org/10.1007/978-3-031-26419-1_3
4. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to Quantify. Inform. Retriev. Series, Springer (2023). https://doi.org/10.1007/978-3-031-20467-8
5. Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowl. Discov. pp. 164–206 (2008). https://doi.org/10.1007/s10618-008-0097-y
6. Moreo, A., González, P., del Coz, J.J.: Kernel density estimation for multiclass quantification (2024), https://arxiv.org/abs/2401.00490
7. Senz, M., Bunse, M.: DortmundAI at LeQua 2022: Regularized SLD. In: Conf. and Labs of the Eval. Forum. pp. 1911–1915 (2022), http://ceur-ws.org/Vol-3180/paper-152.pdf
8. Virtanen, P., et al.: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Meth. pp. 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
9. Wright, S.J., Nocedal, J.: Numerical optimization. Operations Res. and Financial Engin., Springer, 2nd edn. (2006). https://doi.org/10.1007/978-0-387-40065-5

UniOvi Team at LeQua 2024: Quantification via Gaussian Latent Space Representations

Olaya Pérez-Mon[0000−0002−4527−6698] and Pablo González[0000−0002−9250−0920]

Artificial Intelligence Center, University of Oviedo, Gijón, Asturias, Spain
UO257319@uniovi.es
gonzalezgpablo@uniovi.es

Abstract. Most traditional quantification methods often depend on prior probability shift assumptions to develop models that use classifiers for optimal prevalence estimation. In contrast, the idea of this team was to introduce an end-to-end neural network that utilizes Gaussian distributions to achieve invariant sample representations. This paper describes this deep learning architecture and its application to the LeQua 2024 quantification competition, specifically, addressing tasks T2, T3, and T4.

1 Motivation

Our main goal in this competition was to evaluate the performance of a novel deep-learning representation layer for quantification. This layer is based on modelling latent spaces using Gaussian distributions, to obtain an invariant representation of bags of examples. Using this layer in a deep neural network, with an architecture suited for quantification [1], we can undertake prevalence estimation tasks and compare the performance of our method against some of the most popular quantification algorithms, which served as baselines in the competition, as well as against the other participants.

We focused on tackling tasks T2 and T3, both complex multiclass problems, as well as T4, which, despite being a binary problem, presents the added challenge of covariate shift. We opted not to participate in task T1, a binary quantification problem where traditional quantification methods are highly effective and much simpler (using our approach for this task might have been seen as overkill). Our solution delivered consistently strong results in the multiclass tasks (T2 and T3), achieving a gold medal in both tasks. In the binary task with covariate shift, T4, we achieved fifth place. The setup and methodology employed to tackle the competition tasks will be detailed in Section 2.

2 Method

For this competition, we used a deep neural network designed to address quantification problems. Our approach builds on the same architecture as Deep

Quantification Network (DQN) [1], but instead of using pooling layers for representing the bags, we use Gaussian distributions. The architecture consists of three main components: i) a feature extraction module, which is adapted to the specific task, it projects features into latent spaces. ii) a bag representation module that summarizes the feature vectors into a numerical representation of each bag. In our case, Gaussian distributions are used to model the latent space. And, iii) a quantification module, a combination of dense linear layers, that learns the relationship between bag representations and bag prevalences, outputting prevalence estimates due to a softmax activation function.

This architecture is flexible, allowing the optimization of different loss functions, such as the Relative Absolute Error (RAE) for tasks T2 and T4, and the Normalized Match Distance (NMD) for T3. In addition, it can be trained using bags labelled by prevalence, with or without individual example labels.

2.1 Layer details

Our method introduces a novel approach to generating invariant bag features using learnable multivariate Gaussian distributions. This representation layer models the latent space, to which the bag examples are projected, by covering it with Gaussian distributions, each defined by a mean vector and a covariance matrix, which are learnable parameters of the network. Each example projected in the latent space is evaluated against the Gaussian distributions by calculating their likelihoods. The mean likelihood across all examples in a bag then shows how well each Gaussian represents the entire bag, resulting in a compact and invariant representation in the latent space.

In order to capture more useful information about the bags, we propose an extension of the network incorporating multiple latent space representations, each modelled with its own Gaussian distributions. To favor learning different latent spaces with different information, we use the Centered Kernel Alignment (CKA) score (for more information, please, see [2]). Since this metric measures the similarity between latent spaces, we incorporated it into the loss function we are minimizing to make them as distinct and informative as possible.

Another important aspect of our approach is that the network can be trained using only bags labelled by prevalence, but can also benefit when utilizing individual labelled examples, adding a classification layer to the feature extraction module that allows the network to learn representations, combining quantification and classification losses during training. It is also possible to train the net by combining these two techniques.

When labelled bags are limited, we use data augmentation techniques to generate new bags. There are two options: i) when only individual labels are available, we generate new bags using the APP protocol [3] that allows us to generate bags with a desired prevalence and, ii) when only bag prevalences are available, we can mix two real bags producing a new augmented bag whose prevalence is the average of the original bags. When possible, we can combine both strategies.

3 Experiments

The datasets used in this study correspond to the T2, T3 and T4 tasks from the LeQua 2024 competition. T2 is a multiclass problem with prior probability shift, T3 is a multiclass ordinal quantification problem, and T4 is a binary problem with covariate shift.

The official loss function for T2 and T4 is the Relative Absolute Error (RAE) and Normalized Mean Distance (NMD) for task T3 (see [4] for more information on quantification loss functions).

The experimental setup was tailored to optimize the network according to the official loss function for each task. Our quantification method's flexibility allows for various training data configurations, as explained in Section 2. Specifically, for T2 and T4, half of the training bags were generated using APP from example-labelled data, while the other half were labelled by prevalence (with 700 out of 1,000 bags used for training and 300 for validation). To prevent overfitting, the labelled data was augmented by randomly mixing real bags, as described in Section 2.

For T3, the network was trained exclusively on bags generated using APP from the example-labeled dataset, with the 1,000 training bags used for validation and early stopping. In this case, the label information was incorporated into the network helping its convergence.

The number of Gaussian distributions to cover the latent space was fixed to 100. For T2, the most challenging task, we used 18 different latent spaces, each in 5 dimensions. The number of latent spaces for T3 was set to 9, while the other parameters remained unchanged. The limitation in the number of latent spaces was given due to memory limitations in the GPU used for training (12Gb). For T4, we utilized 1 latent space with 17 dimensions and 156 Gaussians. For T4, these parameter values were found using Optuna [5]. For T2 and T3 parameters were chosen manually, without using any automated procedure.

3.1 Results

In this section, we present the performance of our proposed method on the three tasks: T2, T3, and T4.

For the T2 dataset, see Table 1, our method significantly outperforms all baseline methods as well as other participants' ones, in terms of the optimized loss function, Relative Absolute Error (RAE). The performance gap is significant, demonstrating the effectiveness of our approach in handling this complex multiclass problem with prior probability shift.

The T3 dataset presents a multiclass ordinal problem with five classes, where the objective loss function to optimize is the NMD, see Table 1. In this case, our method also demonstrates superior performance, although the margin of improvement is narrower compared to T2. Nevertheless, our approach proves to be more effective in dealing with the complexities of ordinal quantification.

Finally, for the T4 dataset, see Table 1, our method was outperformed by other teams' approaches. In this case, our thought was that the ability to train

Ranking	T2 (RAE)	T3 (NMD)	T4 (RAE)
1	**0.9217 (Ours)**	**0.0644 (Ours)**	**0.1093 (tobiaslotz)**
2	1.0302 (tobiaslotz)	0.0659 (tobiaslotz)	0.1150 (EMQ)
3	1.0786 (hustav)	0.0668 (PCC)	0.1156 (DistMatching-y)
4	1.1616 (EMQ)	0.0690 (KDEy)	0.1180 (KDEy)
5	1.1942 (PACC)	0.0721 (juanjodelcoz)	0.1298 (Ours)

Table 1: Top five competitors for each of the tasks in which we competed.

the network using bags with a particular type of shift might prove an advantage when this shift is reproduced in the test bags. In practice, quantification methods using an underlying classifier and designed for prior probability shift, performed substantially well in this task, with errors very similar to those obtained by teams in task T1.

4 Conclusions

We have reached several interesting conclusions. Firstly, our method has demonstrated strong adaptability across a variety of quantification scenarios, achieving robust results in the presence of prior probability shift, covariate shift and, even in ordinal quantification problems. It is particularly effective in handling multiclass problems, which are often more complex and demanding.

Additionally, our method can optimize different loss functions, such as RAE and NMD, delivering strong results in both cases. Furthermore, our approach is flexible regarding training data. It can handle situations where data is labelled by individual examples, where bags are labelled by prevalence, or when there is a combination of both cases.

References

1. Lei Qi, Mohammed Khaleel, Wallapak Tavanapong, Adisak Sukul, and David Peterson. A framework for deep quantification learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*, pages 232–248. Springer, 2021.
2. Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019.
3. Andrea Esuli, Alessandro Fabris, Alejandro Moreo, and Fabrizio Sebastiani. *Learning to Quantify*. Springer, Cham, CH, 2023.
4. Fabrizio Sebastiani. Evaluation measures for quantification: An axiomatic approach. *Information Retrieval Journal*, 23(3):255–288, 2020.
5. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

UNIOESTE at LeQua 2024: Combining the top-ranked quantifiers

Luiz Luth, Daniel Ojeda, Guilherme Gomes, and Andre Maletzke

Western Paraná State University, Foz do Iguaçu, Brazil
{luiz.junior90, daniel.ojeda, guilherme.gomes, andre.maletzke}@unioeste.br

Abstract. Keywords: Counting · Class prevalence shift · label shift

1 Introduction

In recent years, the problem of class prior estimation (also known as quantification or learning to quantify) has gained significant attention due to its importance in real-world applications where estimating the distribution of classes in unlabelled data is crucial. The challenge extends beyond classifying individual instances to accurately predicting the proportion of different classes in a dataset, a task complicated by prior probability shift (or label shift), where the class distribution in the test data differs from the training data, making standard supervised learning approaches less effective.

The LeQua 2024 competition addresses these challenges through several tasks, and our proposal specifically aims to tackle Task T1. This task focuses on evaluating binary quantifiers, which are tasked with predicting the relative frequencies of a class and its complement under conditions of prior probability shift.

Our proposal for Task T1 of the LeQua 2024 competition addresses the challenge of predicting class proportions under prior probability shift using a straightforward and effective strategy. Instead of relying on a single quantifier, we propose to run several quantifiers on the target dataset. Each quantifier generates a prediction for the class proportions, and we then rank these quantifiers based on their performance. Once ranked, we select the top-performing quantifiers and form the final prediction by averaging the predictions of the top-ranked models. This approach leverages the strengths of multiple quantification methods, increasing robustness and reducing the reliance on any one model. By combining the predictions of the best-performing quantifiers, we aim to produce more reliable and accurate estimates of the class proportions, effectively tackling the prior probability shift that characterizes Task T1.

2 Method

This section describes the method applied to obtain predictions for Task T1. First, we evaluated a set of base classifiers for Task T1, including CatBoost [1],

Random Forests [2], Support Vector Machines [3], and XGBClassifier [4]. We conducted a grid search for each classifier to determine the best hyperparameters, as detailed in Table 1.

Table 1. Parameter grids for different models

Model	Parameter	Values
XGBoost	`n_estimators`	{100, 200, 300}
	`gamma`	{0.01, 0.1}
	`max_depth`	{3, 6, 9, 12}
	`learning_rate`	{0.001, 0.01, 0.1, 1}
CatBoost	`depth`	{4, 5, 6, 7, 8, 9, 10}
	`learning_rate`	{0.01, 0.05, 0.2, 0.5, 0.8, 1.0}
	`iterations`	{10, 20, 50, 100}
RandomForest	`n_estimators`	{100, 200, 300, 400, 500, 600, 700, 800}
	`criterion`	{`gini`, `entropy`}
	`max_depth`	{-1, 10, 50, 100}
SVC	`kernel`	{`linear`, `poly`, `rbf`, `sigmoid`}
	`gamma`	{`scale`, `auto`}

Furthermore, to account for label shift, we applied the correction technique proposed by Lipton et al. (2018) [5], which adjusts the predictions to mitigate the impact of prior probability shift in the test data.

Next, we employed several well-established quantifiers to generate the class proportion estimates. We selected the best-performing quantifiers for binary problems as described by Schumacher et al. (2021) [6]. We used the following quantifiers: DyS, HDy, SORD, CC, ACC, MS, and EMQ, each with their respective default parameters.

Finally, to generate the predictions, we ranked all the quantifiers and measured the Mean Absolute Error (MAE) for each individual. We also evaluated the performance by combining the quantifiers in groups of three and five, calculating the average prediction for each group. We then compared the performance of the individual quantifiers and their combinations, selecting the best-performing setup based on these results to predict the test set. This approach ensured that we identified the most robust configuration for accurate quantification under the conditions of prior probability shift.

3 Results

The results of our experiments indicated that the best-performing setup for Task T1 was a combination of both an individual classifier and a group of quantifiers. Specifically, the SVC classifier with `gamma` = `auto` and `kernel` = `poly` yielded the most accurate results among the base classifiers. In terms of quantifiers, the combination of DyS, HDy, and EMQ proved to be the most effective, providing the lowest Mean Absolute Error when their predictions were averaged. This

combined approach outperformed other configurations, and thus it was selected to predict the test set, ensuring robust and accurate quantification under the conditions of a prior probability shift.

4 Conclusion

Our approach to Task T1 of the LeQua 2024 competition effectively addressed the challenge of prior probability shift by combining multiple quantification techniques and leveraging an optimized classifier. To further enhance our results, we applied a correction technique that adjusts predictions to mitigate the impact of prior probability shift in the test data, ensuring more reliable class proportion estimates. By averaging the predictions of the top quantifiers and applying the correction, we successfully selected the best-performing setup, which proved to be effective in handling distributional shifts. This strategy highlights the importance of combining classifier performance, quantification methods, and correction techniques to achieve high predictive accuracy under challenging conditions.

References

1. L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features, 2017," *arXiv preprint arXiv:1706.09516*, vol. 201, 2017.
2. L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.
3. C. Cortes, "Support-vector networks," *Machine Learning*, 1995.
4. T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
5. Z. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," in *International conference on machine learning*. PMLR, 2018, pp. 3122–3130.
6. T. Schumacher, M. Strohmaier, and F. Lemmerich, "A comparative evaluation of quantification methods," *arXiv preprint arXiv:2103.03223*, 2021.