

# QuAcc: Using Quantification to Predict Classifier Accuracy Under Prior Probability Shift

Intelligenza Artificiale

1–17

© The Author(s) 2025

Article reuse guidelines:

sagepub.com/journals-permissions

DOI: 10.1177/17248035251338347

journals.sagepub.com/home/ina

| IOS Press

Lorenzo Volpi<sup>1</sup> , Alejandro Moreo<sup>1</sup> and Fabrizio Sebastiani<sup>1</sup>

## Abstract

Using cross-validation to predict the accuracy of a classifier on unseen data can be done reliably only in the absence of dataset shift, i.e., when the training data and the unseen data are IID. In this work we deal instead with the problem of predicting classifier accuracy on unseen data affected by prior probability shift (PPS), an important type of dataset shift. We propose QuAcc, a method built on top of “quantification” algorithms robust to PPS, i.e., algorithms devised for estimating the prevalence values of the classes in unseen data affected by PPS. QuAcc is based on the idea of viewing the cells of the contingency table (on which classifier accuracy is computed) as classes, and of estimating, via a quantification algorithm, their prevalence values on the unseen data labelled by the classifier. We perform systematic experiments in which we compare the prediction error incurred by QuAcc with that of state-of-the-art classifier accuracy prediction (CAP) methods.

## Keywords

classifier accuracy prediction, quantification, prior probability shift

Received: February 14, 2025; accepted: April 14, 2025

## Introduction

The standard way of predicting the accuracy of a classifier on unseen data is using cross-validation (CV).<sup>1</sup> However, the accuracy estimates that CV returns are accurate only when the training data  $T$  and the unseen data  $U$  are IID, i.e., when no *dataset shift* (Quiñero-Candela et al., 2009) is present in the data. Unfortunately, dataset shift is ubiquitous in real-world applications, for a variety of reasons. One such reason is the possible non-stationarity of the environment across time and/or space and/or other variables, in which case the deployment conditions are irreproducible at training time. Another reason is the possible presence of sample selection bias in the training data, as when the process of labelling these data has introduced bias in them intentionally (e.g., when oversampling the minority class) or unintentionally (e.g., when using active learning). In all these cases, predicting the accuracy of a classifier on unseen data is problematic.

In this paper we tackle the problem of predicting the accuracy of a classifier on unseen data affected by *prior probability shift* (PPS), an important type of dataset shift in which  $P(X|Y)$ , the class-conditional distribution of the covariates, is invariant across the training data and the test data, but  $P(Y)$ , the distribution of the class labels, is not. We propose a novel classifier accuracy prediction (CAP) method based on two key ideas.

The first idea is that of viewing the cells of the contingency table (on which classifier accuracy is computed) as *classes*, so that estimating the accuracy of a classifier can be performed by counting the number of test datapoints assigned to each such class by a (different) classifier. In other words, this involves a *relabelling* of the datapoints, i.e., from the original set  $\mathcal{Y}$  of classes to a new set  $\mathcal{C}$  of classes representing contingency table cells.

The second idea starts from the observation that, when there is PPS between training data and test data labelled according to  $\mathcal{Y}$ , there is (as we prove) PPS also when the same training data and test data are relabelled according to  $\mathcal{C}$ . This observation allows us to estimate the counts of the contingency table cells via *quantification* algorithms (Esuli et al., 2023) robust to PPS, i.e., algorithms whose task is predicting the prevalence values of the classes (i.e., the class priors) in samples of unseen data affected by

<sup>1</sup>Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, 56124 Italy

## Corresponding Author:

Lorenzo Volpi, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124 Pisa, Italy.

Email: lorenzo.volpi@isti.cnr.it

PPS. We thus name our novel CAP method QuAcc (“Quantification for Accuracy Prediction”).

We present experiments in which we test QuAcc against state-of-the-art CAP methods on data generated by a robust experimental protocol, i.e., on settings characterised by different amounts of training data imbalance, test data imbalance, and PPS. In order to show how well QuAcc deals with the binary case and with the multiclass case, we present separate experiments for these two settings. In order to ensure the reproducibility of our experiments we make available the code and the data on which they are based.<sup>2</sup>

The rest of the paper is structured as follows. After discussing the problem setting (Section “Preliminaries”) and reviewing related work (Section “Related Work”), in Section “QuAcc” we illustrate our method for classifier accuracy prediction. Section “Experiments” presents our experimental results, while Section “Conclusion” wraps up, pointing at avenues for future research.

## Preliminaries

In this paper we adopt the following notation. By  $\mathbf{x}$  we indicate a datapoint drawn from a domain  $\mathcal{X}$ . By  $y$  we indicate a class drawn from a set of classes  $\mathcal{Y} = \{y_1, \dots, y_n\}$ , which we call the *codeframe*; when  $n = 2$  we write  $\mathcal{Y} = \{\oplus, \ominus\}$  to indicate that  $\oplus$  is the “positive” class, which is usually the minority class and represents the concept we aim to characterise, and  $\ominus$  is the “negative” class. We write  $(\mathbf{x}, y)$  to indicate that  $y$  is the class label of  $\mathbf{x}$ . By  $T$  (the training set),  $V$  (the validation set), and  $U$  (the unlabelled set) we denote three sets of labelled datapoints, where the labels of  $U$  are assumed unknown.

We use symbol  $\sigma$  to denote a *sample*, i.e., a non-empty set of labelled datapoints  $(\mathbf{x}, y)$ . We use  $p_\sigma(y)$  to denote the (true) prevalence of class  $y$  in sample  $\sigma$  (i.e., the fraction of items in  $\sigma$  that belong to  $y$ ); note that  $p_\sigma(y)$  is just a shorthand of  $\Pr(Y = y \mid \mathbf{x} \in \sigma)$ , where  $\Pr$  indicates probability and  $Y$  is a random variable that ranges on  $\mathcal{Y}$ . Note that  $(p_\sigma(y_1), \dots, p_\sigma(y_n)) \in \Delta^{(n-1)}$  is a *probability distribution* over  $\mathcal{Y}$ , where  $\Delta^{(n-1)}$  denotes the probability simplex, i.e., the set of all probability distributions over  $n$  classes.

We define a *quantifier* as a function  $q: 2^{\mathcal{X}} \rightarrow \Delta^{(n-1)}$ , i.e., a function that maps a sample  $\sigma$  into a probability distribution  $(\hat{p}_\sigma^q(y_1), \dots, \hat{p}_\sigma^q(y_n)) \in \Delta^{(n-1)}$ , where  $\hat{p}_\sigma^q(y_i)$  denotes the estimate of  $p_\sigma(y_i)$  returned by  $q$ . A quantifier is thus an estimator of class prevalence values, and is trained by an inductive learning algorithm on a set of labelled datapoints. Since in the binary case it holds that  $p_\sigma(\ominus) = 1 - p_\sigma(\oplus)$ , binary quantification reduces to estimating the prevalence of class  $\oplus$  only.

By  $h: \mathcal{X} \rightarrow \mathcal{Y}$  we denote a classifier trained on  $T$ . We assume  $h$  to return labels in  $\mathcal{Y}$  by the usual rule

$$h(\mathbf{x}) = \arg \max_{y_i \in \mathcal{Y}} \Pr(y_i \mid \mathbf{x})$$

where the  $\Pr(y_i \mid \mathbf{x})$  (the *posterior probabilities*, which we assume to be well-calibrated) each represent the probability that the classifier subjectively attributes to the fact that  $\mathbf{x}$  belongs to class  $y_i$ . This assumption brings about no loss of generality, since a classifier that returns generic confidence scores (instead of posterior probabilities), or posterior probabilities that are not well-calibrated, can be made to return well-calibrated posterior probabilities by means of a calibration function.

Given a set of classes  $\mathcal{Y} = \{y_1, \dots, y_n\}$ , we indicate by  $\mathcal{C} = \{C_{11}, \dots, C_{nn}\}$  the contingency table used for measuring the accuracy of (single-label) classifiers that map datapoints into  $\mathcal{Y}$ . We denote by  $c_{ij}^U$  the value of cell  $C_{ij}$  that results from the application of a classifier  $h: \mathcal{X} \rightarrow \mathcal{Y}$  to an unlabelled set  $U$ , i.e., the number of datapoints in  $U$  that belong to class  $y_i$  and have been assigned class  $y_j$  by  $h$ . By  $A(h, U)$  we denote a function that measures the accuracy of classifier  $h$  on an unlabelled set  $U$ , while by  $E(A(h, U), \hat{A}(h, U))$  we denote a function that measures the CAP error incurred by estimating  $A(h, U)$  as  $\hat{A}(h, U)$ .

## Problem Setting

Our goal is to estimate the accuracy  $A(h, U)$  of a classifier  $h$  on an unlabelled set  $U$ . In this paper we make three assumptions that are typically made in the CAP literature (see Section “Related Work”), i.e., (i) that the set  $T$  on which  $h$  has been trained is no longer available, (ii) that a validation set  $V$  is instead available, and that (iii)  $T$  and  $V$  are drawn IID from the same distribution  $P(X, Y)$  (where  $X$  is a random variable that ranges on  $\mathcal{X}$ ) while  $U$  is drawn from a different distribution  $Q(X, Y)$ .<sup>3</sup> In this paper we assume the presence of PPS between  $P(X, Y)$  and  $Q(X, Y)$ , i.e., we assume that

$$P(X|Y) = Q(X|Y) \quad (1)$$

$$P(Y) \neq Q(Y) \quad (2)$$

Hereafter, we call Equations 1 and 2 the *PPS assumptions*. Note that the goal (of QuAcc and of its competing methods) is not to *adapt*  $h$  to  $U$  (as in domain adaptation), but merely to estimate its accuracy on  $U$ .

## Related Work

A number of methods for CAP under dataset shift have appeared in the last few years.

Bhaskaruni et al. (2018) propose a “reverse testing” framework where (i) the classifier  $h$  generates labels for the datapoints in  $U$ , (ii) these (pseudo-)labels are used to train another classifier  $h'$  on  $U$ , and (iii)  $h'$  labels the datapoints in  $T$ . Any accuracy measure can be computed on the predictions of  $h'$  (since the true labels of  $T$  are known) and be used as a proxy of the accuracy of  $h$  on  $U$ .

This idea is later refined in the *Reverse Classification Accuracy* (RCA) method by Elsahar and Gallé (2019). RCA carries out a similar “back-and-forth” process in which a classifier  $h'$  is trained on the (pseudo-)labels generated by  $h$  on  $U$ . The RCA score is defined in terms of the discrepancy between the labels assigned by the two classifiers  $h$  and  $h'$  to the datapoints of a held-out validation set  $V$ .

A variant (RCA\*) is also proposed that aims at factoring out the discrepancy between the classifiers that can be explained simply by the accumulation of errors in the two-way training process, using additional validation samples. This approach requires several validation sets  $V_i$  on which the RCA (or RCA\*) scores are computed. These values, together with the true accuracy values of  $h$  on each  $V_i$ , are used to train a regressor, which, given as input the RCA (or RCA\*) scores obtained on  $U$ , directly predicts the accuracy of  $h$  on  $U$ .

The idea of using pseudo-labels for learning from  $U$  is further investigated by Chen et al. (2021), who propose an iterative self-training ensemble  $h'$  (the “check model”) that estimates the error of  $h$  as the fraction of test datapoints on which classifiers  $h$  and  $h'$  disagree. Following this idea, Jiang et al. (2022) propose Generalization Disagreement Equality (GDE), a CAP method based on the disagreement of two datapoints of the same model. In a similar vein, You et al. (2022) combine the two worlds, training a regressor on top of the outputs generated by an ensemble model.

The idea of training a regressor is also put forth by Redyuk et al. (2019), who use a sampling generation protocol on a validation sample  $V$  to generate many subsamples  $V_i$  that hopefully mimic a type of shift similar to the one present in the test sample. A regressor is then trained on a representation of the entire sample (the authors propose to use fixed percentiles of the distribution of the class-specific posteriors) to predict classifier accuracy on every such validation subsample. Somehow similarly, Deng and Zheng (2021) train a linear regressor on the Frechét distances between the validation samples  $V_i$  and  $T$ .

Guillory et al. (2021) propose a method that also relies on a regressor for predicting the accuracy of  $h$  in  $U$ . In addition to the validation sample  $V$ , the method assumes the existence of different validation samples  $V_i$  exhibiting some type of dataset shift with respect to  $V$ . For every validation set  $V_i$ , the average max confidence of the classifier ( $C_{V_i}$ ) is computed. The scores used to train the regressor are computed as the *Difference of Confidences* (DoC – which is also the name for the proposed method) between  $V$  and each  $V_i$ , i.e., as  $\text{DoC}(V, V_i) = C_V - C_{V_i}$ . Such scores are used to train a regressor to predict the difference in accuracy that  $h$  exhibits on  $V$  and  $V_i$ . Using this regressor, the accuracy of  $h$  in  $U$  can be computed straightforwardly from the score  $\text{DoC}(V, U)$  and the (known) accuracy of  $h$  in  $V$ .

Methods relying on *Importance Weighting* (IW) try instead to estimate classifier accuracy on  $U$  using estimates  $\hat{r}(\mathbf{x})$  of the true *density ratio*  $r(\mathbf{x}) = \frac{q(\mathbf{x})}{p(\mathbf{x})}$ , with  $p$  and  $q$  the

density functions of the train and test probability distributions  $P$  and  $Q$ , respectively (Sugiyama et al., 2007). Given that  $\mathbb{E}_Q[f(\mathbf{x})] = \mathbb{E}_P[\frac{q(\mathbf{x})}{p(\mathbf{x})}f(\mathbf{x})]$  for any  $f$ , knowing the density ratio would allow for direct estimation of  $h$ 's vanilla accuracy in the test set  $U$ , by simply considering  $f$  be the 0-1 loss  $\mathcal{L}_{0-1}(y, \hat{y}) = \mathbb{1}[y \neq \hat{y}]$ , i.e.,

$$\begin{aligned} A(h, U) &= \frac{1}{|U|} \sum_{(\mathbf{x}, y) \in U} \mathcal{L}_{0-1}(y, h(\mathbf{x})) \\ &\approx \frac{1}{|T|} \sum_{(\mathbf{x}, y) \in T} \mathcal{L}_{0-1}(y, h(\mathbf{x})) \cdot \hat{r}(\mathbf{x}) \end{aligned}$$

One of the best-known IW approaches in the literature is the *Kullback-Leibler Importance Estimation Procedure* (KLIEP) (Sugiyama et al., 2007). A more recent approach, called *Mandoline* (Chen et al., 2021), builds on top of a variant of KLIEP that uses more elaborated representations crafted from the original covariates by means of some special type of user-defined transformations called *slicing functions*. These representations may additionally incorporate information from the classifier itself (e.g., the posterior probabilities). One limitation of this approach is that non-linear evaluation measures, such as  $F_1$ , cannot be estimated.

The *Average Threshold Confidence* (ATC) method (Garg et al., 2022) learns instead a threshold  $t$  on (a function of) the posterior probabilities of the classifier, such that datapoints that obtain a score higher than  $t$  are assumed to be correctly classified. The authors investigate two such functions  $m$ , one based on maximum confidence and one based on negative entropy. Threshold  $t$  is learned on the validation set  $V$  as the value above which the proportion of datapoints scoring higher than  $t$  equates the proportion of correctly classified datapoints (i.e., coincide with the value of vanilla accuracy). Assuming that  $\text{Pr}(y_i|\mathbf{x})$  are the posterior probabilities for  $\mathbf{x}$  returned by classifier  $h$ , the accuracy that  $h$  will obtain on  $U$  is estimated as the fraction of datapoints in  $U$  that score higher than  $t$ , i.e.:

$$\text{ATC}_m(U) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} \mathbb{1}[m((\text{Pr}(y_1|\mathbf{x}), \dots, \text{Pr}(y_n|\mathbf{x}))) > t]$$

*Linear Equations for Accuracy Prediction* (LEAP) is instead a recently proposed method (Volpi et al., 2024) designed to predict the entries  $c_{ij}^U$  of a contingency table resulting from the application of classifier  $h$  to unlabelled dataset  $U$ . LEAP treats the values  $c_{ij}^U$  as  $n^2$  unknowns and constructs a system of  $n^2$  equations by leveraging (a) the values  $c_{ij}^V$  of the contingency table obtained by applying classifier  $h$  to validation set  $V$ , and (b) Equations 1 and 2, which derive from assuming PPS. The solution of this system provides an estimation of the values  $c_{ij}^U$  of the contingency table on  $U$ , which can then be used to assess the accuracy of  $h$  on  $U$  via the chosen evaluation measure.

Note that all the methods discussed in this section, with the exception of those proposed in Bhaskaruni et al. (2018)

and Volpi et al. (2024), have been tested by their authors only with vanilla accuracy as the classifier accuracy measure.

## QuAcc

QuAcc, the method we propose, comes in three different variants, that we describe in the next three subsections.

### The $1 \times n^2$ Method

In the general multiclass case, most classifier accuracy measures can be computed from the values  $c_{ij}^U$  of a contingency table resulting from the application of classifier  $h$  to the set  $U$  of unlabelled datapoints. Of course, in operational situations the values  $c_{ij}^U$  are unknown, since the true labels of the datapoints in  $U$  are unknown. Our method for estimating  $A(h, U)$  under PPS is based on the following idea:

1. View the cells of the contingency table  $\mathcal{C} = \{C_{11}, \dots, C_{nn}\}$  as a codeframe, i.e., consider each cell  $C_{ij} \in \mathcal{C}$  as a class. Note that  $\mathcal{C} \equiv \mathcal{Y} \times \mathcal{Y}$ .
2. Train, on a labelled set  $V$ , a model that estimates the values  $c_{ij}^U$ .
3. Use the estimates  $\hat{c}_{ij}^U$  to compute  $\hat{A}(h, U)$ .

Step 2 can obviously be recast as training on  $V$  a model that estimates the prevalence values  $p_U(C_{ij})$  of classes  $C_{ij}$  in  $U$ , since  $c_{ij}^U = p_U(C_{ij}) \cdot |U|$ . A key aspect of QuAcc is that it uses, for tackling this step, techniques from *quantification* (Esuli et al., 2023), since most such techniques are indeed concerned with training estimators of the class prevalence values under PPS.

In order to carry out Step 2, we represent the datapoints as pairs  $(\tilde{\mathbf{x}}, \tilde{y})$ . Here  $\tilde{\mathbf{x}}$  is a vector

$$\tilde{\mathbf{x}} = (\mathbf{x}, \Pr(y_1|\mathbf{x}), \dots, \Pr(y_n|\mathbf{x}))$$

which incorporates (i) the original representation  $\mathbf{x}$  that classifier  $h$  has used, and (ii) the posterior probabilities  $\Pr(y_i|\mathbf{x})$  that  $h$  has returned for  $\mathbf{x}$ . In other words, we train a quantifier  $q$  by providing it with all the information we have about  $\mathbf{x}$  that  $q$  might use to figure out which cell  $C_{ij}$  datapoint  $\mathbf{x}$  is likely to belong to. (Of course, the quantifier is not interested in individual datapoints *per se*, but is interested in them only insofar as they contribute to the distribution of  $U$  across the classes.)

In our pairs  $(\tilde{\mathbf{x}}, \tilde{y})$ ,  $\tilde{y}$  is instead a label that ranges not on  $\mathcal{Y}$  but on  $\mathcal{C}$ . When  $\tilde{y}$  is the label of a datapoint in  $V$ , it is its true label. When  $\tilde{y}$  is the label of a datapoint in  $U$ , instead,  $\tilde{y}$  is unknown. While it is not our goal to guess  $\tilde{y}$  for this datapoint in particular, it is our goal to estimate, for each  $y_i, y_j \in \mathcal{Y}$ , the prevalence  $p_U(C_{ij}) \equiv c_{ij}^U/|U|$  of datapoints  $(\mathbf{x}, y_i) \in U$  such that  $h(\mathbf{x}) = y_j$ . It is for reaching this goal that we use a quantifier. Once we have estimated the prevalence in  $U$  of all  $C_{ij} \in \mathcal{C}$ , by multiplying all these estimates

by  $|U|$  we obtain estimates of all counts  $c_{ij}^U$ , and we can thus estimate  $A(h, U)$ . Since our quantifier works on a codeframe comprised of  $n^2$  classes, we call this the  $1 \times n^2$  method.

An important aspect of this method is that it works for any classifier accuracy measure  $A$  defined in terms of a contingency table, since it does not estimate measure  $A$  directly but estimates the values  $c_{ij}^U$  of the cells of the contingency table  $\mathcal{C}$  on which measure  $A$  is based. Another important aspect of this method is that it is classifier-independent and quantifier-independent, i.e., it does not make any assumption on which method has been used for training  $h$  and on which method is to be used for training  $q$ .

In the binary case, in which  $\mathcal{Y} = \{\oplus, \ominus\}$ , the contingency table is  $\mathcal{C} = \{\text{TP}, \text{TN}, \text{FP}, \text{FN}\}$ , and we need to train a single multiclass quantifier  $q$  that operates on these four classes; we thus call this the  $1 \times 4$  method.

### The $n \times n$ Method

A variant of the  $1 \times n^2$  method can be obtained by observing that, once we have applied classifier  $h$  to  $U$ , we already know the value of  $c_j^U \equiv \sum_{i=1}^n c_{ij}^U$  for each class  $y_j \in \mathcal{Y}$ , since this is the number of datapoints in  $U$  which  $h$  has assigned to  $y_j$ . This means that we can leverage this information and solve our CAP problem by training, instead of one multiclass quantifier operating on  $n^2$  classes,  $n$  multiclass quantifiers operating on  $n$  classes each, where the  $j$ -th such quantifier is tasked with estimating how the datapoints in  $c_j^U$  are distributed across  $\{c_{1j}^U, \dots, c_{nj}^U\}$ . Since this method involves  $n$  multiclass quantifiers that operate on  $n$  classes each, we call this the  $n \times n$  method.

In the binary case, switching from the binary analogue of the  $1 \times n^2$  method to the binary analogue of the  $n \times n$  method means switching from a single quantifier that operates on 4 classes to two binary quantifiers, which can be trained by leveraging the value of  $|\text{TP} \cup \text{FP}|$  (resp., the value of  $|\text{TN} \cup \text{FN}|$ ), since this is the number of datapoints in  $U$  that  $h$  has assigned to class  $\oplus$  (resp.,  $\ominus$ ), and is thus known. We call this the  $2 \times 2$  method.

### The $1 \times (n + 1)$ Method

A variant of the  $1 \times n^2$  method can be derived by observing that all the mispredicted points can altogether be conceptualized as *background noise*. This suggests a method that models the correctly predicted proportions via  $n$  classes  $C_{11}, \dots, C_{nn}$ , and models background noise via an additional class  $C_{\neq}$ .<sup>4</sup> This solution thus involves a single multiclass quantifier with  $(n + 1)$  classes (we thus call this method the  $1 \times (n + 1)$  method), that needs to estimate how the datapoints are distributed across classes  $C_{11}, \dots, C_{nn}, C_{\neq}$ . Even though this method cannot exploit the pairwise information that captures the individual class-class correlations, it may be advantageous since it operates on a

reduced, coarser-grain (hence, hopefully simpler) codeframe than the original  $1 \times n^2$  method.

One potential disadvantage of the  $1 \times (n + 1)$  method with respect to the two previous ones is that it does *not* work for all classifier accuracy measures. One example measure for which this solution does not work is macroaveraged  $F_1$  (see Equation 9), since the fact that all the misclassified datapoints are merged into a single class  $C_{\neq}$  prevents the evaluation of  $F_1$  for the individual classes  $y_1, \dots, y_n$ . Other example measures for which this solution does not work are all the measures  $A$  to which different types of mispredicted datapoints contribute differently, such as cost-sensitive accuracy measures. However, the  $1 \times (n + 1)$  method does work for an important classifier accuracy measure such as vanilla accuracy, whose mathematical form (see Equation 8) is such that knowing the value of  $\sum_{i \neq j} c_{ij}$  suffices, and knowing the individual values of  $c_{ij}$  is not strictly required.

In the binary case, the  $1 \times (n + 1)$  method translates into a method employing a multiclass quantifier operating on three classes, i.e., the classes corresponding to TP, TN,  $FP \cup FN$ , the latter combining the false positives and the false negatives into a single class; we call this binary variant the  $1 \times 3$  method. In this case, along with vanilla accuracy, also  $F_1$  can be employed, since for its formula (see Equation 7) knowing the individual values of  $|FP|$  and  $|FN|$  is not necessary, since knowing  $|FP| + |FN|$  suffices. The  $1 \times 3$  method is particularly interesting, since we may conjecture that the FPs and the FNs mostly lie on a relatively compact region of space that flanks, from different sides, the separating surface, i.e., we may conjecture that most of the FPs and the FNs lie in a *contiguous* region of the space.

### Adding Covariates

We also explore the impact of enriching the vectorial representations  $\tilde{\mathbf{x}}$  with additional covariates that convey information potentially useful for the quantification process. Given a datapoint  $\mathbf{x}$  and the vector  $\mathbf{p} = (p_1, \dots, p_n)$  (where  $p_i$  is used as a shorthand for the posterior probability  $\Pr(y_i|\mathbf{x})$  returned by classifier  $h$ ), we explore three additional covariates that make explicit to the quantifier information only implicitly present in  $\mathbf{p}$ .

The first covariate we consider is *maximum confidence*, defined as

$$\text{MC}(\mathbf{p}) = \max_{i \in \{1, \dots, n\}} p_i$$

MC provides a measure of how confident  $h$  is in predicting the class label of  $\mathbf{x}$ : the higher the value, the greater the confidence.

A second covariate we consider is *negative entropy*, given by

$$\text{NE}(\mathbf{p}) = \sum_{i=1}^n p_i \log p_i$$

which also provides a measure (alternative to MC) of how confident  $h$  is in predicting the class label of  $\mathbf{x}$ . Both MC and NE (which are also used in the ATC method (Garg et al., 2022)) reach their maximum values when  $p_i = 1$  for some  $i$ , and attain their minimum values for the uniform distribution (i.e., when the  $p_i$ 's all equal  $1/n$ ).

A third covariate we explore is based on the softmax function  $s$ , that maps any real-valued vector  $\mathbf{z} = (z_1, \dots, z_n)$  into a probability distribution  $s(\mathbf{z}) \equiv \mathbf{p} = (p_1, \dots, p_n)$  where  $p_i = e^{z_i} \cdot (\sum_{j=1}^n e^{z_j})^{-1}$ .

We consider, as the basis for a new covariate, the values returned by the *inverse softmax*  $s^{-1}(\mathbf{p}) = (z'_1, \dots, z'_n)$ , where  $z'_i = \log p_i + c$  and  $c$  is the logarithm of the normalisation factor  $c = \log(\sum_{j=1}^n e^{z_j})$ . Since  $c$  is undetermined (the actual values  $z_j$  are unknown), we set  $c = -\frac{1}{n} \sum_{j=1}^n \log p_j$ , thus centring the resulting values  $z'_i$  around zero.

The rationale behind inverse softmax to  $\mathbf{p}$  is that of amplifying, in a non-linear way, the difference between low-confidence and high-confidence values. As with MC, we focus on the maximum value. The resulting *max inverse softmax* (MIS) covariate is thus given by

$$\text{MIS}(\mathbf{p}) = \max_{i \in \{1, \dots, n\}} \left( \log p_i - \frac{1}{n} \sum_{j=1}^n \log p_j \right)$$

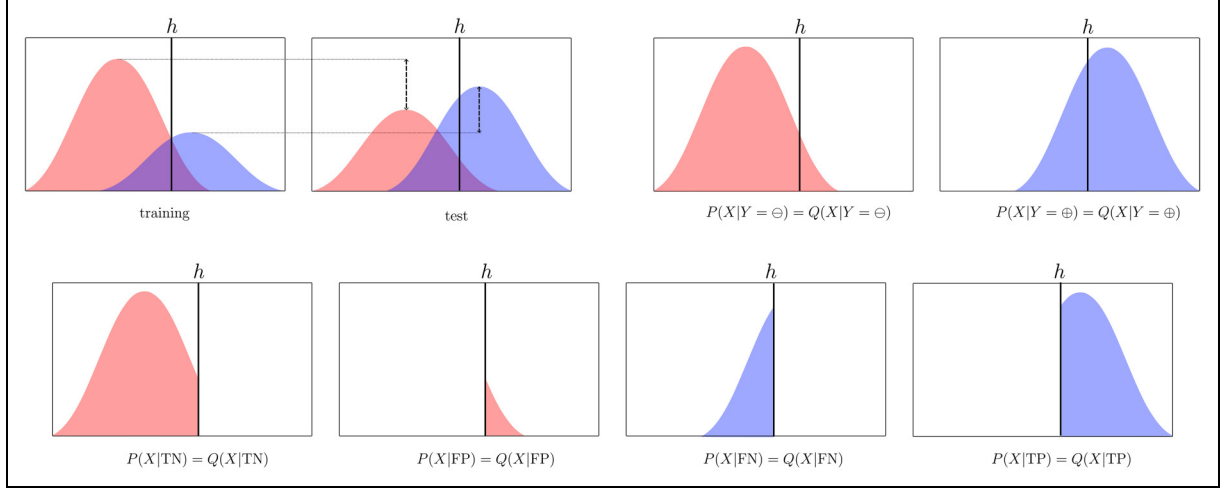
### A Look Back At the Relationships Between QuAcc and the PPS Assumptions

Before discussing our experiments, it is worthwhile to stop for a moment and look back at how and if QuAcc addresses the PPS assumptions. In essence, QuAcc may be seen as performing a *relabelling* of the datapoints, from a labelling based on codeframe  $\mathcal{Y}$  to one based on codeframe  $\mathcal{C}$ .

However, the PPS assumptions underlying our problem setting, and specifically the assumption that the class-conditional densities are stationary (Equation 1), cannot be taken for granted after this relabelling; in other words, the fact that  $P(X|Y) = Q(X|Y)$  does not mean that  $P(X|C) = Q(X|C)$ , where  $C$  is a random variable that ranges on  $\mathcal{C}$ .

Luckily, since the original class-conditional distributions are partitioned into two complementary distributions by a deterministic function (the classifier), and since the original class-conditional distributions are assumed to be stationary, the complementary parts remain stationary after the relabelling as well, as illustrated in Figure 1. This is more formally stated in the following proposition, a proof of which is given in Appendix “Proof of Proposition”.

**Proposition 1** If two distributions  $P(X, Y)$  and  $Q(X, Y)$  are such that  $P(X|Y) = Q(X|Y)$ , then a relabelling of the datapoints in  $\mathcal{X}$  from  $\mathcal{Y} = \{y_1, \dots, y_n\}$  to  $\mathcal{C} = \mathcal{Y} \times \mathcal{Y}$ , with  $\mathcal{C}$  the set of cells of the contingency



**Figure 1.** Graphical depiction of the effects of QuAcc’s “relabelling” strategy. A red area depicts a hypothetical distribution of the covariates in the examples of class  $\ominus$ , while a blue area does the same for the examples of class  $\oplus$ ; in Figure (a) these are the distributions  $P(X|Y = \ominus)P(Y = \ominus)$  and  $P(X|Y = \oplus)P(Y = \oplus)$  of the *training* data, while in Figure (b) these are the distributions  $Q(X|Y = \ominus)Q(Y = \ominus)$  and  $Q(X|Y = \oplus)Q(Y = \oplus)$  of the *test* data. The two figures show an example of PPS, since (i) the class-conditional distribution of the covariates is invariant (i.e.,  $P(X|Y) = Q(X|Y)$ ) as shown in Figure (c) and (d), as shown by the fact that the two red areas have identical *shape* (and the same goes for the two blue areas), and (ii) the distribution of the priors is not invariant (i.e.,  $P(Y) \neq Q(Y)$ ), as shown by the fact that the two red shapes have different *size* (and the same goes for the two blue curves). The bar labelled  $h$  represents the classifier, which partitions the examples into predicted positives (here assumed to be to the right of the  $h$  bar) and predicted negatives (to the left of the  $h$  bar). The regions of the true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP), are shown in Figures (e), (f), (g), (g), respectively. It is clear from Figures (a) and (b) that each of these four regions has the same *shape*, but different *size*, in the training and in the test data; this implies that, if we view TN, FP, FN, TP as *classes* and relabel our examples accordingly, the PPS assumptions are valid also after the relabelling.

table resulting from classifying the datapoints in  $\mathcal{X}$  via a classifier  $h$ , is such that  $P(X|C) = Q(X|C)$ , with  $C$  a random variable ranging on  $\mathcal{C}$ .  $\square$

However, while this reasoning holds for the  $I \times n^2$  and  $n \times n$  variants of QuAcc, it does not apply to the  $I \times (n+I)$  variant. In this case, the split merges different parts of the original distributions into the new “background noise” class  $C_{\neq}$ , which is a mixture of all the class-specific misclassification types with a mixture parameter imposed by the priors of the training distribution. In the binary case, and considering that  $C_{\neq} = \text{FP} \cup \text{FN}$  and  $\text{FP} \cap \text{FN} = \emptyset$ , it follows that

$$\begin{aligned} P(X|C_{\neq}) &= \frac{P(X, C_{\neq})}{P(C_{\neq})} = \frac{P(X, \text{FP}) + P(X, \text{FN})}{P(C_{\neq})} \\ &= \frac{P(X|\text{FP})P(\text{FP}) + P(X|\text{FN})P(\text{FN})}{P(C_{\neq})} \\ &= P(X|\text{FP})P(\text{FP}|C_{\neq}) + P(X|\text{FN})P(\text{FN}|C_{\neq}) \end{aligned}$$

As a result, this class cannot accommodate the new prior of the test distribution, i.e.,  $P(X|C_{\neq}) \neq Q(X|C_{\neq})$  since, by virtue of the PPS assumptions it holds that  $P(X|\text{FP}) = Q(X|\text{FP})$  and  $P(X|\text{FN}) = Q(X|\text{FN})$  but also that the priors change  $P(\text{FP}) \neq Q(\text{FP})$  and  $P(\text{FN}) \neq Q(\text{FN})$ , and so there is no guarantee that  $P(\text{FP}|C_{\neq}) = Q(\text{FP}|C_{\neq})$  nor  $P(\text{FN}|C_{\neq}) = Q(\text{FN}|C_{\neq})$ ; see Figure 2.

Despite this conceptual pitfall, this method is still interesting from the point of view of efficiency: it models fewer classes than the  $I \times n^2$  and  $n \times n$  counterparts (and this is especially true in the multiclass case), and this reduction in the complexity of the model might well pay off in terms of prediction accuracy. We analyse this trade-off empirically in Section “Experiments”.

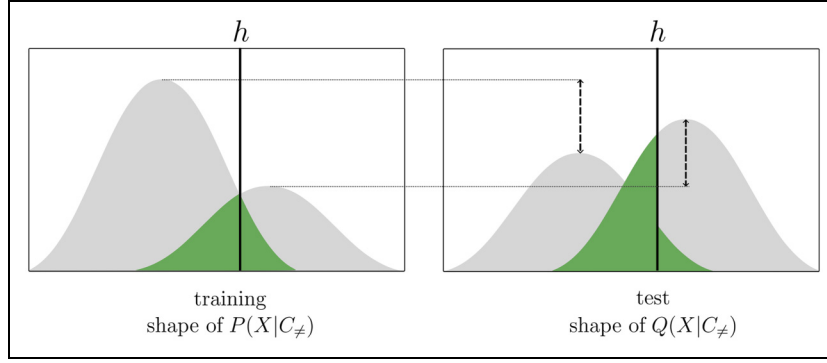
## Experiments

In this section we describe the experiments we have carried out and discuss the results we have obtained.

### Evaluation Protocol

A dataset for testing quantification systems is usually the result of the application of an *extraction protocol* (Esuli et al., 2023, § 3.4) to a dataset  $\Omega = (\mathcal{T}, \mathcal{V}, \mathcal{U})$  otherwise used to test *classification* systems, where  $\mathcal{T}$  is the training pool,  $\mathcal{V}$  is the validation pool, and  $\mathcal{U}$  is the test pool. (By “pool” we mean a – usually large – set from which various samples will be extracted according to the extraction protocol.)

In the binary case we here use the widely adopted *artificial prevalence protocol* (APP – Esuli et al. (2023, § 3.4.2)), which consists of randomly extracting from  $\mathcal{U}$  a number of test samples  $U_1, \dots, U_k$  characterised by values of  $p(\oplus)$



**Figure 2.** In the  $1 \times 3$  method, the region of the false negatives (green region to the left of the  $h$  bar) is merged with the region of the false positives (green region to the right of the  $h$  bar) and viewed as a single class  $C_+$ . The two figures show that, after the relabelling, the PPS assumptions do not hold any longer, since not only the green areas in the training data and in the test data have different size (which means that  $P(C) \neq Q(C)$ , consistently with the PPS assumptions), but they also have different shape (which means that  $P(X|C) \neq Q(X|C)$ , which conflicts with the PPS assumptions).

that lie on a predefined grid of values (in this case: the 21-point grid  $\{0.00, 0.05, \dots, 0.95, 1.00\}$ ). In our experiments, for each value on the grid we randomly extract (with replacement) 100 test samples consisting of 1000 datapoints each, for a total of  $k=2100$  test samples.

For each value that lies on the 9-point grid  $\{0.1, \dots, 0.9\}$  (i) we also randomly extract (with replacement) from  $\mathcal{T}$  a training sample  $T_i$  with  $p(\oplus)$  equal to that value, and (ii) we randomly extract (with replacement) from  $\mathcal{V}$  a validation sample  $V_i$  with  $p(\oplus)$  equal to that value. In both cases, these 9 samples are equally sized, and their size is

$$|T_i| = \frac{10}{9} \cdot \min \{p(\oplus), p(\ominus)\} \cdot |\mathcal{T}| \quad (3)$$

$$|V_i| = \frac{10}{9} \cdot \min \{p(\oplus), p(\ominus)\} \cdot |\mathcal{V}| \quad (4)$$

In both cases, this is the largest possible number that can result in 9 equally-sized without necessarily resorting to re-sampling the same datapoints. In all our experiments we keep  $p_{T_i}(\oplus) = p_{V_i}(\oplus)$ ; while using training samples  $T_i$  characterised by different  $p_{T_i}(\oplus)$  values is meant to test our methods under different conditions of class imbalance, stipulating that  $p_{T_i}(\oplus) = p_{V_i}(\oplus)$  is meant to simulate the fact that (as mentioned in Section “Problem Setting”)  $T_i$  and  $V_i$  are sampled from the same distribution  $P(X, Y)$ .

In the multiclass case, since the application of the APP would result in a number of valid combinations that grows combinatorially with the number of classes, we resort to a variant of the APP based on the Kraemer algorithm for sampling probability distributions uniformly at random (Esuli et al., 2023, § 3.4.3). In other words, in our application of this protocol we still randomly extract (with replacement) from  $\mathcal{U}$  a number of 2100 test samples  $U_i$ , each comprised of 1000 datapoints, but instead of

insisting that their  $p(y_i)$ ’s lie on a grid of predetermined class prevalence values, for each of the 2100 samples we impose that their  $p(y_i)$ ’s comply with a probability distribution drawn uniformly at random from the probability simplex according to the Kraemer algorithm.

We use a similar approach to randomly extract (with replacement) from  $\mathcal{T}$  nine training samples  $T_1, \dots, T_9$ , each using a randomly generated distribution; the same distributions are similarly employed to randomly extract (with replacement) from  $\mathcal{V}$  nine validation samples  $V_1, \dots, V_9$ . In both cases, the nine samples are again equally sized, and their sizes are  $|T_i| = \frac{|\mathcal{T}|}{9}$  and  $|V_i| = \frac{|\mathcal{V}|}{9}$ . This increase in size with respect to the binary case (where sample size is defined by Equations 3 and 4) is made necessary by the higher level of imbalance of the multiclass datasets; since, in Equations 3 and 4, sample size depends on the size of the least frequent class, adopting Equations 3 and 4 for the multiclass case too would excessively reduce the size of the samples. In this case too we keep  $p_{T_i}(y_j) = p_{V_i}(y_j)$  for all  $y_j \in \mathcal{Y}$ , so as to simulate the fact that  $T_i$  and  $V_i$  are sampled from the same distribution  $P(X, Y)$ .

Each result we report is thus the average value of  $E(A(h, U), \hat{A}(h, U))$  across all combinations of

- in the binary case: 9 values of  $p_{T_i}(\oplus)$  (and  $p_{V_i}(\oplus)$ ), 21 values of  $p_{U_j}(\oplus)$ , and 100 samples for each value of  $p_{U_j}(\oplus)$ ;
- in the multiclass case: 9 distributions  $p_{T_i}$  (and  $p_{V_i} = p_{T_i}$ ), and 2100 distributions  $p_{U_j}$ ;

i.e., across  $9 \times 2100 = 18,900$  combinations for both the binary case and the multiclass case. Therefore, the result of using this protocol is that of “stress-testing” the CAP methods, i.e., testing their ability to correctly estimate classifier accuracy in a wide range of situations, characterised



by different values of training class imbalance, test class imbalance, and PPS.

Concerning this latter, note that this experimental protocol clearly simulates PPS, since the distribution of the covariates  $X$  conditional on the distribution of the labels  $Y$  is the same for the  $T_i$ 's and the  $U_j$ 's, but the distribution of the labels  $Y$  is not the same for the  $T_i$ 's and the  $U_j$ 's (in the language of Section "Problem Setting", our protocol simulates a situation in which  $P(X|Y) = Q(X|Y)$  and  $P(Y) \neq Q(Y)$ , i.e., enforces the PPS assumptions).

## Datasets

As the datasets  $\Omega = (\mathcal{T}, \mathcal{V}, \mathcal{U})$  from which the data for testing the CAP methods are extracted, we use four binary datasets (see also Table 1) and four multiclass datasets (see also Table 2), all publicly available. Most of our datasets are extracted from Reuters Corpus Volume I - version 2 (RCV1-v2), a multi-label dataset of Reuters news-wire stories manually classified under 103 classes, which comes split into a training set of 23,149 stories and a test set of 781,265 stories (Lewis et al., 2004); the other dataset we employ is IMDB (Maas et al., 2011), described below.

The datasets we use for our binary experiments are:

- IMDB, a binary dataset of 50,000 movie reviews classified as positive ( $\oplus$ ) or negative ( $\ominus$ ), which comes split into a training set and a test set (25,000 reviews each, both perfectly balanced). We split the above training set into a training pool  $\mathcal{T}$  and a validation pool  $\mathcal{V}$  of 12,500 reviews each via stratified sampling. We use the TFIDF-vectorised version of IMDB provided by the QuaPy library (Moreo et al., 2021).
- The CCAT, GCAT, MCAT classes of the RCV1-v2 dataset. From the original set of 103 classes we pick the three largest classes, i.e., CCAT, GCAT, MCAT. For each such class, we split the original training set into a training pool  $\mathcal{T}$  and a validation pool  $\mathcal{V}$  of 11,575 stories each via stratified sampling. We use the TFIDF-vectorised version of this dataset provided by Scikit-Learn (Pedregosa et al., 2011).

The datasets we use for our multiclass experiments are:

- The Root, CCAT, M1, M14 nodes in the class hierarchy of RCV1-v2. The classes in the RCV1-v2 dataset are organised in a hierarchical structure, where each class corresponds to a node. Each node  $y_k$  is the root of a subtree and can be treated as an independent dataset, with its classes being the direct children of the node and its datapoints being those labelled as belonging to  $y_k$ . We select these four nodes as they generate the largest and best-balanced multiclass datasets of RCV1-v2. Before

**Table 1.** Main Features of the Binary Datasets used in this Paper;  $|\mathcal{T}|$  is the Size of the Training Pool (Which is Equal to  $|\mathcal{V}|$ , the Size of the Validation Pool),  $|\mathcal{U}|$  is the Size of the Test Pool,  $p_{\mathcal{T}}(\oplus)$  is the Prevalence Value of Class  $\oplus$  in the Training Pool  $\mathcal{T}$  (Which is the Same as the Prevalence Value  $p_{\mathcal{V}}(\oplus)$  of  $\oplus$  in the Validation Pool  $\mathcal{V}$ ), While  $|T_i|$  is the Size of the Samples used for Classifier Training (which is the Same as the Size  $|V_i|$  of the Samples used for Quantifier Training).

	$ \mathcal{T} $	$ \mathcal{U} $	$p_{\mathcal{T}}(\oplus)$	$ T_i $
IMDB	12,500	25,000	0.500	6,944
CCAT	11,575	781,265	0.474	5,992
GCAT	11,575	781,265	0.297	3,872
MCAT	11,575	781,265	0.255	3,267

**Table 2.** Main Features of the Multiclass Datasets used in this Paper;  $|\mathcal{T}|$ ,  $|\mathcal{U}|$ , and  $|T_i|$  are Defined as in table 1,  $n = |\mathcal{Y}|$  is the Number of Classes in  $\mathcal{Y}$ , while  $p_{\mathcal{T}}$  is the Distribution of the Datapoints in the Training Pool  $\mathcal{T}$  (Which is the Same as the Distribution  $p_{\mathcal{V}}$  of of the Datapoints in the Validation Pool  $\mathcal{V}$ ), Reported as an Element of the Probability Simplex  $\Delta^{(n-1)}$ .

	$ \mathcal{T} $	$ \mathcal{U} $	$n$	$p_{\mathcal{T}}$	$ T_i $
Root	9,903	665,265	4	[0.085, 0.25, 0.24]	4,951
CCAT	4,708	319,689	4	[0.109, 0.096, 0.055]	2,354
M1	2,816	190,208	4	[0.102, 0.247, 0.441]	1,408
M14	1,188	77,671	3	[0.124, 0.248]	594

using them, we remove all multi-labelled datapoints so as to generate a purely single-label classification setting. For each dataset, we split its RCV1-v2 training set into a training pool  $\mathcal{T}$  and a validation pool  $\mathcal{V}$  of the same size. The exact sizes of training, validation, and test pools, and the number of classes of the four datasets, are reported in Table 2. We use the vectorised version of RCV1-v2 provided by Scikit-Learn (Pedregosa et al., 2011).

The reason why we choose these datasets is that they are fairly large. Indeed, a large enough number of positive training examples is needed for generating large enough training samples (see Equations 3 and 4), and a large enough number of test examples is needed for generating test samples with only minimal overlap.

## Evaluation Measures

As the measure of the error a method incurs when predicting the accuracy  $A(h, U)$  of classifier  $h$  on a set  $U$  of unlabelled datapoints, we use the absolute difference between the true accuracy value and its estimate, i.e.,

$$Err(A(h, u), \hat{A}(h, U)) = |A(h, U) - \hat{A}(h, U)| \quad (5)$$

We run experiments with two different measures  $A$  of classifier accuracy, i.e., vanilla accuracy and  $F_1$ , defined,



for the binary case, as

$$Acc = \frac{|TP| + |TN|}{|TP| + |FP| + |FN| + |TN|} \quad (6)$$

$$F_1 = \frac{2 \cdot |TP|}{2 \cdot |TP| + |FP| + |FN|} \quad (7)$$

and, for the multiclass case, as

$$Acc = \frac{\sum_{i=1}^{|\mathcal{Y}|} c_{ii}}{\sum_{i=1}^{|\mathcal{Y}|} \sum_{j=1}^{|\mathcal{Y}|} c_{ij}} \quad (8)$$

$$F_1 = \frac{1}{|\mathcal{Y}|} \sum_{i=1}^{|\mathcal{Y}|} \frac{2 \cdot c_{ii}}{\sum_{j=1}^{|\mathcal{Y}|} c_{ij} + \sum_{j=1}^{|\mathcal{Y}|} c_{ji}} \quad (9)$$

with Equation 9 describing the average value of binary  $F_1$  (Equation 7) across the classes in  $\mathcal{Y}$  (a measure usually called *macroaveraged*  $F_1$ , and noted  $F_1^M$ ).

We do not discuss measures of quantification error since this error is not important in itself but only inasmuch as it leads to accurate estimation of  $A(h, U)$ . We thus do not show experimental results in which we measure the quantification error incurred by our quantifiers. However, the quality of the latter can be gauged by comparing the results (see last three rows of Table 3) obtained by using the trivial ‘‘Classify and Count’’ quantification method (that simply counts the number of times a class label has been attributed by the classifier), with those obtained by using quantification methods (described below) robust to PPS such as SLD (Saerens et al., 2002) or KDEy (Moreo et al., 2025).

## Baselines

As the baselines for our experiments we use the five CAP methods that, in comparative experiments reported in the literature, have proven the best-performing ones, i.e.,

- *Reverse Classification Accuracy* (RCA) (Elsahar & Gall , 2019), of which we use both variants (RCA and RCA\*), since in the authors’ experiments neither of the two was clearly superior to the other;<sup>5</sup>
- *Mandoline* (Chen et al., 2021);<sup>6</sup>
- *Difference of Confidences* (DoC) (Guillory et al., 2021);<sup>7</sup>
- *Average Threshold Confidence* (ATC) (Garg et al., 2022), of which we use the ‘‘maxconf’’ variant since Garg et al. (2022) found it to be the best-performing variant;<sup>8</sup>
- *Linear Equations for Accuracy Prediction* (LEAP) (Volpi et al., 2024), of which we use the ‘‘KDEy’’ variant, reported to be the best performing one by its authors.<sup>9</sup>

See Section ‘‘Related Work’’ for a description of these five methods; none of the methods from (a) to (d) has

hyperparameters to optimise; for LEAP we could fine-tune the hyperparameters of the underlying quantifier (KDEy), but we decided to use the default ones, mirroring the experimental setup of the original paper (Volpi et al., 2024). We also test a na ve baseline consisting of applying classifier  $h$  to  $V$  and using the value of  $A(h, V)$  as the estimate  $\hat{A}(h, U)$  of the accuracy of  $h$  on  $U$ .

## Configuring QuAcc

Within our CAP methods, as the quantification algorithm for estimating the prevalence values  $p_U(C_{ij})$  of classes (i.e., contingency table cells)  $C_{ij}$  in  $U$ , we experiment with two alternatives, i.e., (a) the well-known *Saerens-Latinne-Decaestecker* method (SLD) (Saerens et al., 2002), thus called from the names of its proposers, and (b) the recently proposed KDEy method (Moreo et al., 2025); both methods are natively multiclass, which is important since the QuAcc variants of Sections ‘‘The  $1 \times n^2$  Method’’ and ‘‘The  $1 \times (n + 1)$  Method’’ require estimating the prevalence values of more than 2 classes in both the binary and the multiclass case, and the QuAcc variant of Section ‘‘The  $n \times n$  Method’’ requires the same in the multiclass setting.

SLD was the best performer in a recent data challenge devoted to tackling quantification on data affected by PPS (Esuli et al., 2022), and works by training a (calibrated) classifier and then using expectation maximisation (Dempster et al., 1977) (i) to tune the posterior probabilities  $\Pr(C_{ij}|\mathbf{x})$  that the classifier returns, and (ii) to re-estimate the class prevalence values  $p_U(C_{ij})$ . SLD carries out Steps (i) and (ii) in an iterative, mutually recursive way, until convergence, which occurs when the estimates of the class prevalence values  $p_U(C_{ij})$  have stabilised. Alexandari et al. (2020) propose a variant of SLD in which, before applying expectation maximisation, the classifier is recalibrated by means of *bias-corrected temperature scaling* (BCTS – the best performer among the recalibration techniques tested in their paper); we leave the decision whether to apply BCTS or not to the model selection phase.

KDEy (Moreo et al., 2025) is a recent distribution-matching method (Esuli et al., 2023, § 4.2.8) for quantification that models samples by means of Gaussian mixture models over the probability simplex containing the posterior probabilities of the datapoints in the sample. For this purpose, the method relies on kernel density estimation (hence its name). In contrast to previously proposed distribution-matching methods that create class-specific histograms of posterior probabilities, KDEy models multivariate densities, thus accounting for inter-class dependencies. In particular, we use the KDEy-ML variant since, in the experiments of Moreo et al. (2025), it was the best performer and, in particular, proved superior to SLD in multiclass problems.

**Table 3.** Results of our Binary Experiments (Expressed in Terms of  $Err(A(h, U), \hat{A}(h, U))$  on the IMDB, CCAT, GCAT, MCAT Datasets with Vanilla Accuracy (Acc) or  $F_1$  as the Classifier Accuracy Measure A.

	IMDB			CCAT			GCAT			MCAT		
	Acc	$F_1$		Acc	$F_1$		Acc	$F_1$		Acc	$F_1$	
Baselines												
Naïve	.1799 ± .208	.1512 ± .226		.1274 ± .165	.1327 ± .229		.1071 ± .150	.1288 ± .225		.1183 ± .164	.1300 ± .223	
RCA (Elsahar & Gallé, 2019)	.1085 ± .116	.1204 ± .148		.0493 ± .056	.1008 ± .134		.0519 ± .051	.1058 ± .147		.0618 ± .076	.1147 ± .151	
RCA* (Elsahar & Gallé, 2019)	.1099 ± .118	.2085 ± .237		.0564 ± .073	.2593 ± .284		.0504 ± .049	.2347 ± .238		.0610 ± .076	.2486 ± .250	
Mandoline (Chen et al., 2021)	.3616 ± .280	—		.1920 ± .176	—		.1581 ± .161	—		.3599 ± .346	—	
ATC (Garg et al., 2022)	.0613 ± .078	.1015 ± .133		.0292 ± .036	.0798 ± .116		.0151 ± .017	.0918 ± .141		.0230 ± .030	.0765 ± .124	
DoC (Guillory et al., 2021)	.0226 ± .020	.0809 ± .096		.0145 ± .013	.0951 ± .109		.0112 ± .010	.0947 ± .126		.0199 ± .022	.0911 ± .123	
LEAP (Volpi et al., 2024)	.0258±0.027	.0291±0.045		.0157±0.016	.0216±0.035		.0111±0.011	<b>.0174±0.036</b>		.0183±0.017	<b>.0213±0.034</b>	
Ours	.0474 ± .038	.0640 ± .091		.0297 ± .024	.0499 ± .082		.0201 ± .015	.0470 ± .094		.0313 ± .042	.0492 ± .092	
QuAcc(CC)	<b>.0162 ± .013</b>	<b>.0259 ± .038</b>		.0151 ± .014	<b>.0199 ± .032</b>		.0097 ± .007	.0221 ± .051		<b>.0136 ± .010</b>	.0227 ± .055	
QuAcc(SLD)	<b>.0167<sup>†</sup> ± .018</b>	.0292 ± .066		<b>.0137 ± .012</b>	<b>.0201<sup>†</sup> ± .051</b>		<b>.0090 ± .008</b>	.0240 ± .067		.0143 ± .013	.0302 ± .080	
QuAcc(KDEy)	28.32%	11.00%		5.52%	7.87%		18.92%	-27.01%		25.68%	-6.57%	
Error reduction												

Note: **Boldface** indicates the best method, while underlining indicates the best baseline. The last row indicates the reduction in error obtained by the best method over the best baseline. No results are available for Mandoline with  $F_1$  since (a) its proposers use vanilla accuracy only as the classifier accuracy measure, and since (b) it is not obvious how to modify the Mandoline code for use with a measure different from Acc. (Also the other baselines were tested by their proposers with vanilla accuracy only, but in their case it is immediate to understand how to modify the code for tackling  $F_1$ ).

In our experiments we use the implementations of SLD and KDEy provided by the QuaPy library (Moreo et al., 2021).<sup>10</sup>

In this work we always use logistic regression as the method for training (i) the classifiers whose accuracy we want to estimate, and (ii) the classifiers that underlie both SLD and KDEy. Note that the classifiers used in (i) and (ii) are independent of each other, i.e., are different classifiers (the former are trained on the  $T_i$ 's while the latter are trained on the  $V_i$ 's) even if they are both trained via logistic regression.

## Results

Table 3 summarises all our experimental results for the binary case, and Table 4 does the same for the multiclass case. In both cases, we provide separate results for 8 combinations of a dataset  $\Omega$  (4 datasets) and a classifier accuracy measure  $A$  (2 measures). Each figure in Tables 3 and 4 is the average value of the  $Err(A(h, U), \hat{A}(h, U))$  measure across the  $9 \times 2100 = 18,900$  combinations of a training sample  $T_i$  (paired with a validation sample  $V_i$  drawn from the same distribution) and a test sample  $U_j$ . All results refer to systems whose hyperparameters have been optimised on a held-out portion of the validation sample using  $E$  as the loss function; in particular, model selection (a) chooses the best value for the regularisation parameter  $C$  of the logistic regression classifier that the quantifier uses, in the range  $\{10^i\}_{i=-3}^{i=3}$ , (b) for the same logistic regression classifier, decides whether to rebalance (by setting the “ClassWeight” hyperparameter to “Balanced”) the datapoints in order to counter the possible class imbalance of the training sample, (c) decides whether to recalibrate (via BCTS) the classifier underlying SLD, (d) chooses the best “bandwidth” for KDEy in the range  $(0.01, 0.02, \dots, 0.20)$ , (e) checks whether the additional covariates of Section “Adding Covariates” are worth including, and (f) decides which of the three variants of Sections “The  $1 \times n^2$  Method” to “The  $1 \times (n + 1)$  Method” should be used. Note that

- model selection here concerns the CAP methods, and not the training of  $h$ , which in all cases we have trained via logistic regression with default parameters;
- the variant of QuAcc  $1 \times (n + 1)$  is excluded from Step (f) of model selection in the multiclass case when the considered evaluation measure is  $F_1$  since, as discussed in Section “The  $1 \times (n + 1)$  Method”, this measure cannot be applied in this case.

Table 5 reports how often, in the binary case, the model selection phase has answered positively for the choices of (b), (c), and (e), while Table 6 reports, for both the binary case and the multiclass case, how often each variant of QuAcc is chosen in Step (f) of model selection.

The main observations we can derive from Tables 3 to 6 are the following:

1. In the binary case, the best performance, on 6 of the 8 combinations of a dataset and a classifier accuracy measure, is obtained by either QuAcc(SLD) or QuAcc(KDEy), while in the multiclass case this happens for 5 of the 8 combinations; this confirms the validity of the intuitions underlying QuAcc. In the 5 cases (2 binary, 3 multiclass) in which neither QuAcc(SLD) nor QuAcc(KDEy) are the best method, the best method is LEAP.
2. While the best performer is in most cases (11 out of 16 combinations) one of QuAcc(SLD) and QuAcc(KDEy), it is not always the same. (This is true also for the baselines, since the best baseline is not always the same: it is LEAP in 14 out of 16 cases, and it is DoC in the other 2 cases.) QuAcc(SLD) is the best on 7 combinations (4 in the binary case and 3 in the multiclass case) and QuAcc(KDEy) on 4 (2 in the binary case and 2 in the multiclass case). It is noteworthy, though, that in 7 out of the 11 combinations in which the best method is one of QuAcc(SLD) and QuAcc(KDEy), *both* methods beat the best baseline.
3. In terms of the amounts of error reduction that our methods obtain with respect to the best baseline when they turn out to be the best method overall (see last row of Tables 3 and 4), these amounts are noteworthy, with important peaks of +28.32% in the binary case (IMDB/Acc) and +42.96% in the multiclass case (M14/Acc). However, it must also be noted that when the best method overall is LEAP, also the latter displays noteworthy amounts of error reduction with respect to QuAcc, with peaks of 27.01% in the binary case (GCAT/ $F_1$ ) and of 37.30% in the multiclass case (Root/Acc).
4. Table 6 shows that all the three methods discussed in Sections “The  $1 \times n^2$  Method” to “The  $1 \times (n + 1)$  Method” are chosen in the model selection phase a substantive number of times. Out of 288 invocations of model selection (8 datasets  $\times$  2 quantification methods  $\times$  9 ( $T_i, V_i$ ) pairs  $\times$  2 accuracy measures), Method  $1 \times n^2$  is selected 98 times (42 in the binary case and 56 in the multiclass case), Method  $n \times n$  is selected 116 times (62 in the binary case and 54 in the multiclass case), and Method  $1 \times (n + 1)$  is selected 74 times (40 in the binary case and 34 in the multiclass case). This shows that none of them is a strawman, while also confirming the intuition that Method  $n \times n$  is superior to Method  $1 \times n^2$ , given that it additionally exploits the knowledge of each  $c_j^U \in \{c_1^U, \dots, c_n^U\}$ .

**Table 4.** Results of our Multiclass Experiments (Expressed in Terms of  $Err(A(h, U), \hat{A}(h, U))$  on the Root, CCAT, MI, M14 Datasets with Vanilla Accuracy (Acc) or  $F_1$  as the Classifier Accuracy Measure A.

		Root		CCAT		MI		M14	
		Acc	$F_1$	Acc	$F_1$	Acc	$F_1$	Acc	$F_1$
Baselines	Naïve	.1693±0.144	.1261±0.094	.2537±0.180	.1581±0.089	.1976±0.168	.1434±0.100	.1818±0.218	.1075±0.105
	ATC (Garg et al., 2022)	.0300±0.039	.1100±0.085	.0856±0.081	.1944±0.083	.0552±0.069	.1291±0.094	.0934±0.171	.1762±0.174
	DoC (Guillory et al., 2021)	.0314±0.030	.0464±0.037	.0582±0.050	.0500±0.043	.0503±0.054	.0431±0.039	.0616±0.080	.0519±0.049
	LEAP (Volpi et al., 2024)	<b>.0126±0.013</b>	.0232 <sup>†</sup> ±0.033	<b>.0380±0.036</b>	<b>.0362±0.042</b>	.0160±0.016	.0362 <sup>†</sup> ±0.051	.0270±0.032	.0262±0.048
Ours	QuAcc(CC)	.0250±0.019	.0346±0.031	.0433±0.035	.0601±0.055	.0328±0.037	.0392±0.040	.0133±0.015	.0328±0.037
	QuAcc(SLD)	.0173±0.016	<b>.0223±0.023</b>	.0579±0.035	.0467±0.040	.0169±0.012	<b>.0349±0.032</b>	.0194±0.020	<b>.0177±0.024</b>
	QuAcc(KDEy)	.0198±0.030	.0283±0.030	.0474±0.045	.0454±0.049	<b>.0153±0.012</b>	.0361 <sup>†</sup> ±0.039	<b>.0116±0.012</b>	.0207±0.024
Error reduction		-37.30%	3.88%	-13.95%	-25.41%	4.38%	3.59%	42.96%	32.44%

Note: **Boldface** indicates the best method, while underlining indicates the best baseline. We do not consider the RCA and RCA\* (Elsahar & Gallé, 2019) and Mandoline (Chen et al., 2021) baselines in our multiclass experimental setup because of their poor performance in the binary setting and high run time.

**Table 5.** Frequencies with which the Model Selection Phase Chooses, in our Binary Experiments, Options Additional Covariates (see Section “Adding Covariates”), BCTS Recalibration (for the SLD quantifier), and Class Balancing (for logistic regression).

Dataset	Quantifier	Additional Covariates	Recalibration	Class Balancing
IMDB	SLD	81.49%	77.78%	59.26%
	KDEy	71.43%	—	88.18%
CCAT	SLD	62.97%	70.38%	77.78%
	KDEy	78.06%	—	84.82%
GCAT	SLD	76.93%	61.54%	65.39%
	KDEy	71.36%	—	78.65%
MCAT	SLD	62.97%	48.15%	74.08%
	KDEy	71.03%	—	68.70%

**Table 6.** Frequencies with which the Model Selection Phase Chooses the three Variants of our Method ( $1 \times n^2$ ,  $n \times n$ ,  $1 \times (n + 1)$ ).

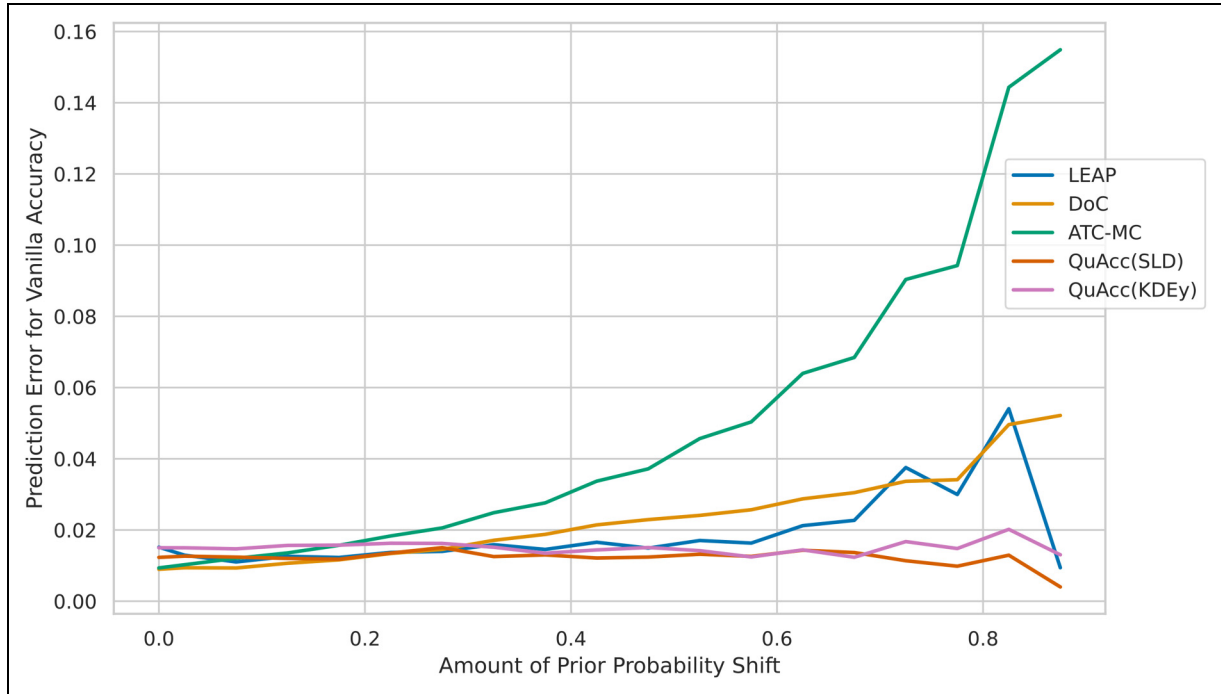
		QuAcc(SLD)			QuAcc(KDEy)		
		$1 \times n^2$	$n \times n$	$1 \times (n + 1)$	$1 \times n^2$	$n \times n$	$1 \times (n + 1)$
binary	IMDB	66.67%	11.11%	22.22%	55.56%	11.11%	33.33%
	CCAT	22.22%	66.67%	11.11%	22.22%	55.56%	22.22%
	GCAT	0.00%	55.56%	44.44%	11.11%	44.44%	44.44%
	MCAT	33.33%	66.67%	0.00%	22.22%	33.33%	44.44%
multiclass	Root	38.89%	44.44%	16.67%	38.89%	38.89%	22.22%
	CCAT	44.44%	16.67%	38.89%	50.00%	27.78%	22.22%
	MI	44.44%	27.78%	27.78%	38.89%	38.89%	22.22%
	MI4	11.11%	72.22%	16.67%	44.44%	33.33%	22.22%

Note: Concerning the values of this table, note that they are all multiples of 5.55% (i.e.,  $\frac{1}{18}$ ); this is because, for each combination of a dataset and quantifier (SLD or KDEy), 9 model selections (i.e., one for each pair  $(T_i, V_i)$ ) are performed for each of 2 accuracy measures (vanilla accuracy and  $F_1$ ).

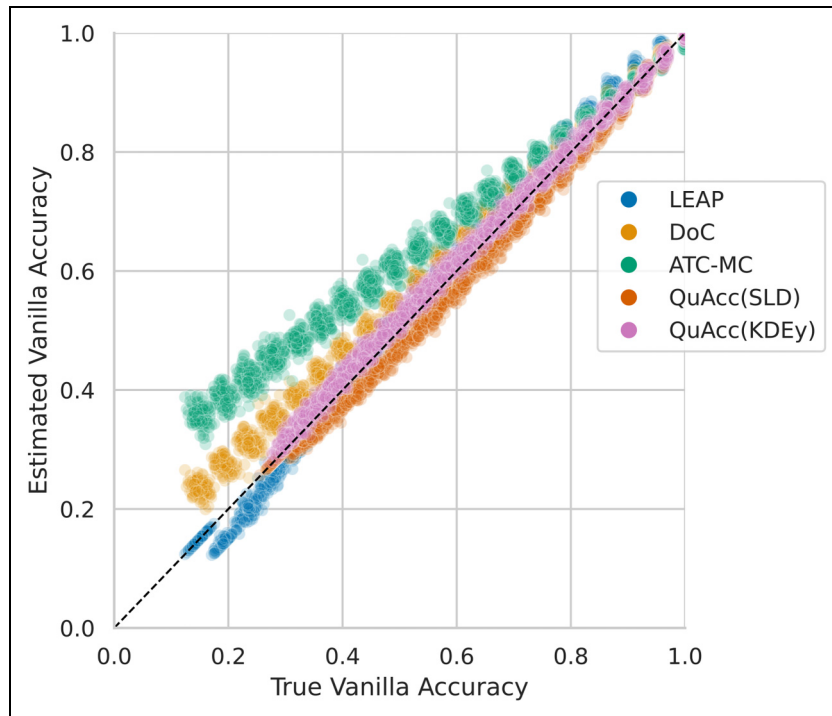
5. QuAcc(CC), a variant of QuAcc that uses the trivial “Classify and Count” (CC) quantifier, is (see 3rd-to-last row of Tables 3 and 4) either the best or the second best baseline in 10 out of 16 combinations. Since CC is not robust to PPS (Esuli et al., 2023, § 1.2), this result shows that the idea of estimating the prevalence values of the cells of the contingency table is a good idea in itself. However, QuAcc(CC) always dramatically underperforms QuAcc(SLD) and QuAcc(KDEy), which shows that using “true” quantification algorithms robust to PPS for performing this estimation is also a good idea. This observation is further reinforced by the fact that the best baseline method LEAP is also one that builds on top of quantification techniques.
6. The best-performing baseline is always either LEAP (Volpi et al., 2024) or DoC (Guillory et al., 2021). ATC (Garg et al., 2022) is always worse than LEAP, while RCA and RCA\* (Elsahar & Gallé, 2019) are always worse than both LEAP and DoC; Mandoline (Chen et al., 2021) always follows the others at a distance.
7. Finally, as can be seen in Table 5, (a) BCTS recalibration and (b) class rebalancing of the logistic

regression classifier that underlies the SLD quantification method, and (c) the use of the additional covariates when training the quantification algorithms, are all fruitful ideas, since model selection more frequently adopts them than not.

We have performed statistical significance tests at different confidence levels so as to check the differences in performance between the best method and all other methods. In Table 3 and 4, all methods whose scores are *not* statistically significantly different from the best one, according to a Student’s *t*-test on paired samples, are marked with † if  $0.001 < p\text{-value} < 0.05$  or with ‡ if  $0.05 \leq p\text{-value}$ ; the absence of any such symbol thus indicates that  $p\text{-value} \leq 0.001$ . What emerges, for the binary results, is that the performance of QuAcc(SLD) is not significantly different from that of the best method LEAP in one of the two combinations in which LEAP is the best method (MCAT/ $F_1$ ); for all the combinations in which one of QuAcc(SLD) and QuAcc(KDEy) is the best method, the performance of all the baselines is significantly different from the performance of these two methods. In the multiclass case, in 2 of the 5 combinations in which one of QuAcc(SLD) and QuAcc(KDEy) beats all the baselines, the performance of the best baseline LEAP is not significantly different to



**Figure 3.** Classifier accuracy prediction error as a function of the amount of PPS, averaged over the training samples extracted for the binary dataset CCAT; the classification accuracy measure is vanilla accuracy. For ease of display, only the five best-performing CAP methods are shown.



**Figure 4.** Estimated vanilla accuracy (y-axis) as a function of the true vanilla accuracy (x-axis) for dataset CCAT and training sample drawn from distribution  $(p_{T_i}(\ominus), p_{T_i}(\oplus)) = (0.1, 0.9)$ .

that of the best method; in all 3 combinations in which LEAP is the best method, instead, the performance of LEAP is significantly different from that of both QuAcc(SLD) and QuAcc(KDEy). It is also noteworthy that, on 3 of the 7 combinations (both binary and multiclass) in which both QuAcc(SLD) and QuAcc(KDEy) beat the best baseline, the error values of the two methods are not significantly different.

Figure 3 displays the CAP error (measured in terms of the  $Err(A(h, U, \hat{A}(h, U)) = |A(h, U) - \hat{A}(h, U)|$  function of Equation 5) that different methods produce when predicting the accuracy (here: vanilla accuracy) of a classifier. CAP error (on the y-axis) is plotted as a function of the amount of PPS (along the x-axis) between the training sample and the test sample; the plotted values are averages across all tested pairs consisting of a training sample (here: of the CCAT binary dataset) and a test sample that display the indicated amount of PPS. The plot reveals how QuAcc(SLD) and QuAcc(KDEy) tend to perform steadily at different amounts of PPS, predicting classifier accuracy with a slightly higher error than the baselines when the amount of PPS is low, but behaving substantially better when the amount of PPS grows. Even though this plot refers to the results for the binary dataset CCAT only, we have observed this trend in all our experiments.

Figure 4 displays the estimated vanilla accuracy (indicated on the y-axis) as a function of the true vanilla accuracy (indicated on the x-axis) for dataset CCAT; the example concerns the case in which the training prevalence is  $(p_{T_i}(\ominus), p_{T_i}(\oplus)) = (0.1, 0.9)$ . This plot shows that, when the positive datapoints outnumber the negative datapoints, ATC and DoC tend to overestimate the classifier's accuracy, especially in cases in which the classifier does not perform well. LEAP tends instead to underestimate the classifier's accuracy when the classifier does not perform well, while it tends to overestimate this accuracy in cases when the classifier performs well. QuAcc(SLD) and QuAcc(KDEy) instead closely follow the diagonal, which represents the ideal behaviour for a CAP method.

## Conclusion

We have presented QuAcc, a new method for predicting classifier accuracy under PPS, an important type of dataset shift. QuAcc is built on top of “quantification” methods robust to PPS, i.e., methods devised for estimating the class prevalence values in samples of unlabelled datapoints affected by PPS. QuAcc is based on the key intuition of viewing the cells of the contingency table, which is used for computing classifier accuracy, as classes, and of training a quantifier that estimates the values of these cells. The experiments we run on eight large datasets simulate a wide variety (a) of amounts of training and/or test data imbalance, and (b) of amounts of PPS. The results of these experiments show that QuAcc systematically


outperforms four of the five state-of-the-art baselines we have employed, while managing to outperform the fifth (LEAP (Volpi et al., 2024)) in 11 experimental configurations out of 16, and exhibiting, in these configurations, average levels of error reduction (with respect to the best such baseline) ranging from 3.59% to 42.96%. QuAcc is independent of the algorithm used for training the classifier, of the algorithm used for training the quantifier, and of the metric used for measuring classifier accuracy.


We plan to extend this research by testing our algorithms on the more challenging scenario in which the data  $V$  on which the quantifier is trained and the data  $T$  on which the classifier was trained are related by PPS. Additional future research we intend to carry out concerns the problem of classifier accuracy prediction under types of dataset shift different from PPS, such as covariate shift. This might not necessarily mean devising methods alternative to QuAcc, but might mean instantiating QuAcc with quantification methods which prove robust to types of dataset shift different from PPS. This may prove non-trivial, though, since most of the quantification literature has focused on PPS (González et al., 2024; Pérez-Mon et al., 2025), somehow neglecting other types of dataset shift.


## Acknowledgments

Lorenzo Volpi's work was supported by project “Italian Strengthening of ESFRI RI RESILIENCE” (ITSERR), funded by the European Union under the NextGenerationEU funding scheme (CUP B53C22001770006). Alejandro Moreo's and Fabrizio Sebastiani's work was partially supported by project “Future Artificial Intelligence Research” (FAIR), project “Quantification under Dataset Shift” (QuaDaSh), and project “Strengthening the Italian RI for Social Mining and Big Data Analytics” (SoBigData.it), all funded by the European Union under the NextGenerationEU funding scheme (CUP B53D22000980006, CUP B53D23026250001, CUP B53C22001760006, respectively).

## ORCID iDs

Lorenzo Volpi  <https://orcid.org/0009-0006-0851-8041>

Alejandro Moreo  <https://orcid.org/0000-0002-0377-1025>

Fabrizio Sebastiani  <https://orcid.org/0000-0003-4221-6427>

## Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The author(s) received financial support as stated in Section “Acknowledgments” for the research, authorship and publication of this article.

## Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.



## Notes

1. This paper reports and extends the results of the first author's MSc thesis (Volpi, 2024), which received, from the Italian Association for Artificial Intelligence (AIXIA), the 2024 Leonardo Lesmo Award for the Best MSc Thesis discussed in 2023/24 at an Italian university in the field of artificial intelligence.
2. <https://github.com/lorenzovolpi/CAP>.
3. For QuAcc and for all the methods we compare it against, assumptions (i) and (ii) are not binding, since all these methods still work if a validation set  $V$  is not available and the original training set  $T$  is instead available; in this case, all these methods rely on  $k$ -fold cross-validation to make up for the absence of  $V$ .
4. This is reminiscent of other multiclass classification-related endeavours in which  $n$  classes represent  $n$  well-defined concepts and an  $(n+1)$ -th class, often called Other, acts as a grab bag for all datapoints that are not deemed to belong to the first  $n$  classes.
5. <https://github.com/naver/domainshift-prediction>.
6. <https://github.com/HazyResearch/mandoline>.
7. <https://github.com/deving09/PerformancePred>.
8. [https://github.com/saurabhgarg1996/ATC\\_code](https://github.com/saurabhgarg1996/ATC_code).
9. <https://github.com/lorenzovolpi/LEAP>.
10. <https://github.com/HLT-ISTI/QuaPy>.

## References

- Alexandari, A., Kundaje, A., & Shrikumar, A. (2020). 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.
- Bhaskaruni, D., Moss, F. P., & Lan, C. (2018). Estimating prediction qualities without ground truth: A revisit of the reverse testing framework. In *Proceedings of the 24th International Conference on Pattern Recognition (ICPR 2018)* (pp. 49–54). Beijing, CN. DOI: <https://doi.org/10.1109/ICPR.2018.8545706>.
- Chen, M., Goel, K., Sohoni, N. S., Poms, F., Fatahalian, K., & Ré, C. (2021). Mandoline: Model evaluation under distribution shift. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)* (pp. 1617–1629). Virtual Event.
- Chen, J., Liu, F., Avci, B., Wu, X., Liang, Y., & Jha, S. (2021). Detecting errors and estimating accuracy on unlabeled data with self-training ensembles. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021)* (pp. 14980–14992). Virtual Event.
- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39(1), 1–38.
- Deng, W., & Zheng, L. (2021). AutoEval: Are labels always necessary for classifier accuracy evaluation?. In *Proceedings of the 2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2021)* (pp. 15069–15078). Virtual Event. DOI: <https://doi.org/10.1109/TPAMI.2021.3136244>.
- Elsahar, H., & Gallé, M. (2019). To annotate or not? Predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)* (pp. 2163–2173). Hong Kong, CN. DOI: <https://doi.org/10.18653/v1/D19-1222>.
- Esuli, A., Fabris, A., Moreo, A., & Sebastiani, F. (2023). *Learning to quantify*. Springer Nature, Cham, CH.
- Esuli, A., Moreo, A., Sebastiani, F., & Sperduti, G. (2022). 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.
- Garg, S., Balakrishnan, S., Lipton, Z. C., Neyshabur, B., & Sedghi, H. (2022). Leveraging unlabeled data to predict out-of-distribution performance. In *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)* Virtual Event.
- González, P., Moreo, A., & Sebastiani, F. (2024). Binary quantification and dataset shift: An experimental investigation. *Data Mining and Knowledge Discovery*, 38(4), 1670–1712. <https://doi.org/10.1007/s10618-024-01014-1>
- Guillory, D., Shankar, V., Ebrahimi, S., Darrell, T., & Schmidt, L. (2021). Predicting with confidence on unseen distributions. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV 2021)* (pp. 1114–1124). Montreal, CA. DOI: <https://doi.org/10.1109/ICCV48922.2021.00117>.
- Jiang, Y., Nagarajan, V., Baek, C., & Kolter, J. Z. (2022). Assessing generalization of SGD via disagreement. In *Proceedings of the International Conference on Learning Representations (ICLR 2022)* Virtual Event.
- Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5, 361–397.
- Lipton, Z. C., Wang, Y., & Smola, A. J. (2018). Detecting and correcting for label shift with black box predictors. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)* (pp. 3128–3136). Stockholm, SE.
- Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)* (pp. 142–150). Portland, US.
- Moreo, A., del Coz, P., & González, J. J. (2025). Kernel density estimation for multiclass quantification. *Machine Learning*, 114(4), 92. <https://doi.org/10.1007/s10994-024-06726-5>
- Moreo, A., Esuli, A., & Sebastiani, F. (2021). 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. DOI: <https://doi.org/10.1145/3459637.3482015>.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Pérez-Mon, O., del Coz, A., Moreo, J. J., & González, P. (2025). Quantification using permutation-invariant networks based on histograms. *Neural Computing and Applications*, 37, 3505–3520. <https://doi.org/10.1007/s00521-024-10721-1>
- Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2009). *Dataset shift in machine learning*. The MIT Press, Cambridge, US.
- Redyuk, S., Schelter, S., Rukat, T., Markl, V., & Bießmann, F. (2019). Learning to validate the predictions of black box machine learning models on unseen data. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA@SIGMOD 2019)* (pp. 4:1–4:4). Amsterdam, NL. <https://doi.org/10.1145/3328519.3329126>.
- Saerens, M., Latinne, P., & Decaestecker, C. (2002). Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1), 21–41. <https://doi.org/10.1162/089976602753284446>
- Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P., & Kawanabe, M. (2007). Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the 21st Conference on Advances in Neural Information Processing Systems (NIPS 2007)* (pp. 1433–1440). Vancouver, CA.
- Volpi, L. (2024). *Estimating classifier accuracy under dataset shift*. Master's Thesis, University of Pisa, Pisa, Italy.
- Volpi, L., Moreo, A., & Sebastiani, F. (2024). A simple method for classifier accuracy prediction under prior probability shift. In *Proceedings of the 27th International Conference on Discovery Science (DS 2024)* (pp. 267–283). Pisa, IT. [https://doi.org/10.1007/978-3-031-78980-9\\_17](https://doi.org/10.1007/978-3-031-78980-9_17).
- You, S. D., Liu, H., & Liu, C. (2022). Predicting classification accuracy of unlabeled datasets using multiple deep neural networks. *IEEE access*, 10, 44627–44637. <https://doi.org/10.1109/ACCESS.2022.3169279>

**Proof of Proposition 1** In this section, we offer a formal proof of Proposition 1.

**Proof.** We need to prove that, if  $P(X|Y) = Q(X|Y)$  with  $Y$  ranging on  $\mathcal{Y}$ , then  $P(X|C) = Q(X|C)$  with  $C$  ranging on  $\mathcal{C}$ . Note that  $C_{ij}$  corresponds to the random event  $Y = y_i \wedge \hat{Y} = y_j$ , so that

$$\begin{aligned} P(X|C = C_{ij}) &= \frac{P(X, C = C_{ij})}{P(C = C_{ij})} \\ &= \frac{P(X, Y = y_i, \hat{Y} = y_j)}{P(Y = y_i, \hat{Y} = y_j)} \\ &= \frac{P(X, \hat{Y} = y_j|Y = y_i)P(Y = y_i)}{P(\hat{Y} = y_j|Y = y_i)P(Y = y_i)} \end{aligned} \quad (10)$$

Since  $\hat{Y} = h(X)$  is a random variable that depends on  $X$  only through a deterministic measurable function (the classifier), and since  $P(X|Y) = Q(X|Y)$ , it then holds (see Lipton et al., 2018, Lemma 1) that  $P(\hat{Y}|Y) = Q(\hat{Y}|Y)$ . For analogous reasons, since the outcome of the deterministic function  $h$  does not alter the stochastic relationship between  $Y$  and  $X$  across the two distributions, and since the output of  $h(x)$  does not depend on the underlying distribution, it also holds that  $P(X, \hat{Y}|Y) = Q(X, \hat{Y}|Y)$ , i.e.,

$$\begin{aligned} P(X = x, \hat{Y} = y_j|Y = y_i) &= \begin{cases} P(X = x|Y = y_i) & \text{if } h(x) = y_j \\ 0 & \text{if } h(x) \neq y_j \end{cases} \\ Q(X = x, \hat{Y} = y_j|Y = y_i) &= \begin{cases} Q(X = x|Y = y_i) & \text{if } h(x) = y_j \\ 0 & \text{if } h(x) \neq y_j \end{cases} \end{aligned}$$

From Equation 10 it thus follows that

$$\begin{aligned} P(X|C = C_{ij}) &= \frac{P(X, \hat{Y} = y_j|Y = y_i)}{P(\hat{Y} = y_j|Y = y_i)} \\ &= \frac{Q(X, \hat{Y} = y_j|Y = y_i)}{Q(\hat{Y} = y_j|Y = y_i)} \\ &= \frac{Q(X, \hat{Y} = y_j, Y = y_i)Q(Y = y_i)}{Q(\hat{Y} = y_j, Y = y_i)Q(Y = y_i)} \\ &= Q(X|\hat{Y} = y_j, Y = y_i) \\ &= Q(X|C = C_{ij}) \end{aligned}$$

□