

Decision Tree Leaf Binning with Expectation Maximization for Quantification

David Kaftan^{1,2} , George Corliss¹ , Anna Bershteyn² , and Richard Povinelli¹ 

¹ Marquette University, Milwaukee WI, USA

{david.kaftan, richard.povinelli}@marquette.edu

² NYU Grossman School of Medicine, New York NY, USA

Abstract. Recent large-scale evaluations have reported poor empirical performance of decision trees for quantification, the task of estimating class prevalence in an unlabelled bag of data. We hypothesize that this failure stems from the choice of aggregation method rather than from the tree structure itself. Drawing on unfolding work in experimental physics, we propose Leaf EM, a quantification method that uses decision-tree leaves as a discrete binning of the feature space and applies Iterative Bayesian Unfolding to estimate class prevalence directly from leaf-count distributions, without requiring assumptions about classifier calibration. We evaluate Leaf EM against seven aggregative quantifiers across three tree splitting criteria (including Gini and quantification criteria Classification Error Balancing and Classification-Quantification Balancing) on 40 binary and 17 multiclass datasets. On binary tasks, the Saerens-Latinne-Decaestecker algorithm applied to a Classification-Quantification Balancing tree achieved the best overall mean rank (4.47 of 21 tree-quantifier combinations), while Leaf EM attained the best within-tree mean rank on Gini trees (2.35 out of 7 quantifiers) and was significantly better than Adjusted Classify and Count, Probabilistic Adjusted Classify and Count, and the Saerens-Latinne-Decaestecker algorithm with temperature scaling on the same tree. Adjusted Classify and Count, previously evaluated on decision trees, performed among the worst quantifiers across tree types. These results indicate that with an appropriate aggregator, decision trees can be greatly improved for quantification.

Keywords: Quantification · Decision Trees · Expectation-Maximization · Maximum Likelihood Estimation · Prior Probability Shift · Prevalence Estimation

1 Introduction

Quantification is the supervised learning task of estimating the prevalence of each class within an unlabelled data sample, rather than predicting the class of individual instances. Quantification arises wherever population-level estimates matter more than individual predictions: tracking political sentiment on social

media [24], estimating disease prevalence from imperfect screening tests [25], monitoring e-commerce catalog defect rates [32], estimating species distributions in ecology [20], and assessing classifier fairness across demographic subpopulations [16].

Quantification typically occurs in two stages. First, a base classifier reports scores for an unlabelled bag. Second, these scores are aggregated into estimates of prevalence for each class. Most aggregative methods can, in principle, be applied to any base classifier. However, the characteristics of the base classifier can influence which aggregative quantifier is optimal. For example, the expectation maximization algorithm proposed by Saerens et al. is highly sensitive to the calibration of the base classifier [13]. It is therefore important to explore aggregative methods that perform well for specific base classifiers.

Classification trees are a simple but powerful classifier. Their ability to model non-linear relationships and their interpretability make classifier trees a relevant machine learning technique decades after they were first popularized. Unfortunately, aggregative quantification methods have shown poor empirical performance when applied to decision trees [26, 34]. Additionally, decision trees are commonly miscalibrated [37], which leads to poor performance of certain aggregative quantifiers [13]. Despite this, empirical evaluations of quantification methods on decision trees have been limited.

Our work builds on two prior lines of research. First, Börner et al. showed that decision-tree leaves can provide an informative discretization of the feature space for unfolding problems [4]. Second, Bunse showed that unfolding and quantification are closely related problems, and that Iterative Bayesian Unfolding (IBU), an expectation maximization algorithm, can be interpreted within a quantification framework [7]. We propose Leaf EM, a quantification method for decision trees that uses tree leaves as discrete bins and infers class prevalence directly from leaf-count distributions. We also provide a large-scale empirical evaluation showing that the weak quantification performance previously reported for decision trees is attributable in part to the aggregation method, and that with a more appropriate aggregator, decision trees can be substantially improved for quantification.

2 Background

2.1 Quantification under Prior Probability Shift

Quantification, or class prevalence estimation, is the task of estimating the distribution of class labels in an unlabelled data sample [12]. Given a feature space $\mathcal{X}$ and a label set $Y \in \mathcal{Y} = \{Y_1, \dots, Y_{n_Y}\}$, the goal is to estimate the prevalence vector $\mathbf{p} = (p_1, \dots, p_{n_Y})$ where $p_i = P(Y = Y_i)$ for an unlabelled test sample. When quantification is performed as a post-process to calibration, the feature space $\mathcal{X}$ typically represents the output of the classifier. The training data consists of labeled instances drawn from a distribution $P_{train}(X, Y)$, while the test data is drawn from a different distribution $P_{test}(X, Y)$. Under

the assumption of prior probability shift, we have $P_{train}(X|Y) = P_{test}(X|Y)$ but $P_{train}(Y) \neq P_{test}(Y)$ [12]. This means that while the relationship between features and labels remains stable, the overall class prevalences differ between training and test sets. This also implies that, under prior probability shift, simply counting the predicted labels from a classifier will yield biased prevalence estimates on the test bag. There are several methods for quantification that exploit the assumption of prior probability shift, including the method proposed in this manuscript.

One such method optimizes $\mathbf{p}$ to maximize the likelihood of observing an unlabelled bag of data (Eq. 1).

$$L(X) = \prod_{i=1}^N \sum_{j=1}^{n_Y} P(X_i|Y_j) \mathbf{p}_j \quad (1)$$

Saerens, Latinne, and Decaestecker developed an EM algorithm (SLD), which iteratively updates prevalence estimates using Bayes' theorem, converging to the maximum likelihood estimate of prevalence [33]. SLD avoids the difficult task of estimating $P(X|Y)$ by instead relying on the posterior probabilities $P(Y|X)$ reported by a base classifier, using the substitution in Eq. 2. This makes SLD highly sensitive to classifier calibration [13]. Despite this, SLD often ranks among the state of the art for quantification tasks [14, 15, 34], and has been shown to be Fisher consistent [36].

$$P(X_i | Y_j) = \frac{P^{(0)}(Y_j | X_i) P^{(0)}(X_i)}{P^{(0)}(Y_j)} \quad (2)$$

In addition to SLD, several other methods have been proposed and evaluated for quantification under prior probability shift. These methods have been broadly grouped into three categories: classify, count, and correct; machine learners adapted for quantification; and distribution matching methods [19]. Many of these methods can be applied as a post-process to decision trees, though evaluation and direct adaptation to decision trees have received relatively little attention. In the next sections, we review the few methods that have been directly proposed for quantification with decision trees, as well as the unfolding methods from physics that we adapt in this work.

2.2 Decision Trees for Quantification

Decision trees are one such machine learner adapted for quantification. Milli et al. (2013) propose quantification trees, a type of decision tree optimized not for individual classification accuracy, but directly for the quantification task of estimating class prevalences [26]. Milli proposes two splitting criteria that provide the best balance between false positives and false negatives across classes. These two criteria are Classification Error Balancing (CEB), where for each class i the quantification error is $E_i = |FP_i - FN_i|$, and Classification-Quantification Balancing (CQB), where $E_i = |FP_i^2 - FN_i^2|$, which additionally penalizes total misclassification. These per-class errors form a single vector QE_r , and the

goodness of a split is measured by the gain $\Delta = \|QE_r^{parent}\|_2 - \|QE_r^{child}\|_2$. However, in a recent large-scale evaluation by Schumacher et al. (2025), both quantification forests and their ACC-corrected variant performed poorly [34].

2.3 Unfolding Methods for Quantification

Unfolding is a family of techniques developed in experimental physics for the problem of recovering a “true” distribution of a physical quantity from observations that have been distorted by detector resolution, measurement noise, and limited efficiency [10]. Such distortions are ubiquitous in high-energy and astroparticle physics, where the quantity of interest can only be inferred indirectly through correlated, measurable observables [7]. The problem is typically discretized by binning the underlying quantity into a set of cause bins $\{C_i\}_{i=1}^{n_C}$ and the observable into a set of effect bins $\{E_j\}_{j=1}^{n_E}$, so that the measurement process is characterized by the conditional probabilities $P(E_j | C_i)$ of observing effect E_j given cause C_i . The goal of unfolding is to recover the distribution over causes from the observed distribution over effects, for which the field has developed several approaches, including Iterative Bayesian Unfolding (IBU) [9, 10], described in the forthcoming Methods section; Singular Value Decomposition-based unfolding [23]; and Blobel’s Regularized Unfolding [2, 3]. At its core, unfolding addresses the same inferential problem as quantification: recovering a distribution over classes (or bins) from imperfect observations [7].

There are several methods for defining the bins E_j . One such method is to use the leaf nodes of a decision tree as bins, as proposed by Börner et al. (2020) [4]. In this approach, a decision tree is trained on the data, and each leaf node corresponds to a bin. Börner did not perform unfolding, but demonstrated that decision tree leaf binning can partition the feature space into informative effect bins. Combining this approach for estimating $P(E_j | C_i)$, and Bunse’s unification of IBU into a quantification framework [7], we propose a method for quantification with decision trees that uses the tree leaves as bins and IBU as the solver.

3 Methods

3.1 Overview: Tree-Leaf Binning + EM Quantification

We adapt IBU to the context of decision tree-based quantification. The method consists of two stages: first, a decision tree is trained on the labeled training data, and the leaf nodes are used to both classify instances and define a discrete partition of the feature space. The class conditional distribution $P(E | C)$ characterizes how instances from each true cause are distributed across the effects. $P(E | C)$ in unfolding can be translated to $P(X | Y)$ in quantification. Second, given an unlabelled test sample, we count the number of instances that fall into each leaf node, yielding an observed distribution $\mathbf{n}$. We then apply IBU to solve for the prevalence vector $\mathbf{p}$ that best explains the observed leaf counts $\mathbf{n}$ under the class conditional distribution.

3.2 Tree Construction

Trees are constructed using standard classification tree algorithms (e.g., CART [5]) with various splitting criteria (Gini, CEB, CQB). Trees using Gini Impurity were implemented using scikit-learn v1.8.0 [30], while trees constructed using CEB and CQB splitting criteria were reimplemented in Python according to Milli et al. [26]. The tree is trained on a subset of the labeled training data, and the remaining labeled data is used as a held-out estimation set to estimate the class conditional distributions. Details of split proportions are given in Section 4. Trees are grown without a maximum depth limit, or a minimum number of samples per leaf constraint during fitting. Instead, the tree is pruned post-hoc using a minimum of 5 samples per leaf from the held-out estimation set to both reduce overfitting and ensure a stable estimate of the class conditional distributions.

3.3 EM Solver for Prevalence Estimation

Iterative Bayesian Unfolding (IBU), originally proposed by D’Agostini [9], is an expectation-maximization approach to solving the unfolding problem. Starting from a prior estimate $\mathbf{p}^{(0)}$ (e.g., a uniform distribution $\mathbf{p}^{(0)} = \frac{1}{n_Y}$), IBU iteratively updates the prevalence estimate according to Bayes’ theorem, converging to the maximum likelihood estimate of prevalence under the observed leaf counts $\mathbf{n}$ and the class conditional distribution $P(\text{Leaf} \mid Y)$ (see Algorithm 1). The IBU algorithm, as applied to quantification by Bunse et al., is nearly identical to the SLD algorithm. The only difference is that SLD estimates the class conditional probabilities $P(X \mid Y)$ using the posterior probabilities $P(Y \mid X)$ output by a base classifier according to Eq. 2, while IBU assumes that the class conditional probabilities are known, typically through a partitioning of X . It is for this reason that SLD is highly sensitive to classifier calibration. Decision trees are known to produce poorly calibrated probabilities [31, 35], motivating the use of IBU in this context.

Decision trees map input data into leaves. We therefore redefine class conditional distributions $P(X \mid Y) = P(\text{Leaf} \mid Y)$ [4]. We choose to model $P(\text{Leaf} \mid Y_i)$ as a Dirichlet distribution with Laplace smoothing and updated by counting the number of instances from each true class that fall into each leaf node on the held-out estimation set. We use the expected value as a well-defined, regularized set of class conditional distributions that can be used in the IBU algorithm without requiring any assumptions about classifier calibration. Because trees are pruned based on the held-out estimation data, we avoid using leaves that provide little information.

4 Experimental methodology

To evaluate the performance of our proposed quantification approach, we follow the experimental protocol of Schumacher et al. [34]. In total, we evaluate 40 binary datasets and 17 multiclass datasets across the remaining combinations of train-test splits, training class prevalences, and test bag prevalences. Across

Algorithm 1 Iterative Bayesian Unfolding (IBU) adapted for Leaf EM

Require: $n(Leaf_j)$, $P(Leaf_j | Y_i)$
Ensure: Estimated prevalence vector $\hat{\mathbf{p}} = (\hat{p}_1, \dots, \hat{p}_{n_Y})$

```

1:  $k \leftarrow 0$ 
2:  $P^{(0)}(Y_i) \leftarrow \frac{1}{n_Y}$  for  $i = 1, \dots, n_Y$ 
3: repeat
4:   for  $j = 1$  to  $n_{Leaf}$  do
5:     for  $i = 1$  to  $n_Y$  do
6:       
$$P^{(k)}(Y_i | Leaf_j) \leftarrow \frac{P(Leaf_j | Y_i) P^{(k)}(Y_i)}{\sum_{i'=1}^{n_Y} P(Leaf_j | Y_{i'}) P^{(k)}(Y_{i'})}$$

7:     end for
8:   end for
9:   for  $i = 1$  to  $n_Y$  do
10:    
$$\hat{n}^{(k+1)}(Y_i) \leftarrow \sum_{j=1}^{n_{Leaf}} n(Leaf_j) P^{(k)}(Y_i | Leaf_j)$$

11:    
$$P^{(k+1)}(Y_i) \leftarrow \frac{\hat{n}^{(k+1)}(Y_i)}{\sum_{i'=1}^{n_Y} \hat{n}^{(k+1)}(Y_{i'})}$$

12:   end for
13:    $k \leftarrow k + 1$ 
14: until convergence
15:  $\hat{p}_i \leftarrow P^{(k)}(Y_i)$  for  $i = 1, \dots, n_Y$ 
16: return  $\hat{\mathbf{p}} = (\hat{p}_1, \dots, \hat{p}_{n_Y})$ 

```

all variables, we run 12,540 experiments, each repeated ten times with different random seeds, for a total of 125,400 experiments.

Deviating from the original setup, we do not employ stratified 10-fold cross-validation when estimating the misclassification rates, classifier scores, or other quantities typically determined through cross-validation. Instead, we estimate these quantities from a single held-out estimation set (25% of training data). This demonstrates the performance of a single, interpretable tree, rather than an ensemble. While ensemble classifiers have been shown to improve performance [26, 29, 34], we consider ensemble methods to be orthogonal to the contributions of this work, and leave the question of how best to ensemble our proposed method to future work.

Details on training and testing splits, class prevalences, datasets, and seeding can be found in [34]. Quantification methods and evaluation metrics are described in the following.

4.1 Quantification Methods

We compare the following quantification methods.

Leaf EM: Our proposed method, which uses the leaf nodes of a decision tree as bins, and applies Iterative Bayesian Unfolding to estimate class prevalences. The held-out estimation set is used to estimate $P(Leaf | Y)$.

ACC: Adjusted Classify and Count, which corrects the prevalence estimates of a base classifier by adjusting for its true and false positive rates [17, 18]. ACC

is used in this experiment as it is the quantification technique applied in the original quantification tree paper by Milli et al. (2013) [26], and in Schumacher et al. (2025) [34]. The multiclass extension of ACC is implemented according to the unconstrained least squares method proposed by Bunse [6] to minimize the loss function proposed by Hopkins and King [24]. The held-out estimation set is used to estimate the true and false positive rates for each class.

PACC: Probabilistic Adjusted Classify and Count, which extends ACC to probabilistic classifiers by using the predicted probabilities instead of hard classifications [1]. The multiclass version of this method is implemented using the same methods as ACC. The held-out estimation set is used to estimate the average predicted probability for each class.

KDEyML: Kernel Density Estimation with Maximum Likelihood Estimation, which estimates the class conditional distributions of classifier scores using kernel density estimation, and then applies maximum likelihood estimation to recover the prevalence vector [28]. While this method is designed for continuous classifier scores, we apply it to the discrete scores output by decision trees because it bears a similarity to the unfolding approach we propose. Specifically, as the bandwidth of the kernel approaches zero, KDEyML approaches the same likelihood function as our proposed method (as long as no leaves produce identical classifier scores). By using KDEyML, we can compare the performance of our proposed method to a similar method that does not take advantage of the discrete nature of decision tree outputs. For this experiment, a bandwidth of 0.05 is used. The held-out estimation set is used to fit the kernel density estimates for each class.

SLD: The EM algorithm described in Section 2.1. Because calibration is important for this method, it is evaluated with and without temperature scaling, denoted SLD (TS). The held-out estimation set is used to perform temperature scaling [22].

HDy: The Hellinger Distance method proposed by Gonzalez-Castro et al. [21], which estimates class prevalences by minimizing the Hellinger distance between the predicted and true distributions of classifier scores. It performs well as a binary quantification method, and is included as a baseline for comparison [8, 34]. It was not implemented for multiclass quantification as it is not inherently designed for multiclass problems and has been shown to perform poorly [34].

For each of these methods, we evaluated three base classifiers: a standard decision tree using the Gini impurity criterion, a quantification tree using splitting criteria designed for quantification: the CEB criterion and the CQB criterion [26]. All comparative methods were implemented using the QuaPy library in Python 3.13 [27].

4.2 Evaluation Measures

Following [34], we evaluate quantification performance using Absolute Error (AE), summed across all classes (Eq. 3). This metric is used, rather than mean absolute error across classes, as it allows for consistent comparisons across datasets with different class cardinality, as described in [34].

$$e_{\text{AE}}(p, \hat{p}) = \sum_{i=1}^L |p_i - \hat{p}_i|. \quad (3)$$

Most methods run by Schumacher et al. [34] used logistic regression base classifiers, which are able to provide informative confidence scores when trained on few data. They evaluated several settings where the base classifier was trained on little data with high class imbalance. Under these settings, the CEB and CQB splitting criteria frequently reached their stopping conditions before making a single split, yielding degenerate single-leaf trees. Additionally, the pruning policy of requiring a minimum number of samples in each leaf also contributed to the formation of degenerate trees. While degenerate trees are clearly undesirable, their relative impact on performance across quantification methods is arbitrary as the trees will provide no information about the underlying class distribution. For this reason, we exclude all trials where at least one tree was degenerate from calculation of absolute error. We do, however, report the percent of trials by tree type that were degenerate to evaluate the relative impact of degenerate trees across tree types.

For each (algorithm, dataset) pair, we average errors across all sampled scenarios. To assess statistical significance of performance differences across algorithms, we follow the two-step procedure of Demšar [11] as adopted in [34]: a Friedman test on the null hypothesis of equal performance, followed by a Nemenyi post-hoc test at the 5% significance level when the Friedman test is rejected. These tests are conducted on average AE across datasets, yielding 40 samples for binary tests and 17 samples for multiclass tests.

5 Results

5.1 Binary Quantification: Overall Performance

The CQB tree with SLD outperformed all other methods for binary quantification, achieving the best (smallest) mean rank (4.47 out of 21 methods) and lowest absolute error in 16 out of 40 datasets (Table 1, Figure 2). The next best performing tree-quantifier combination was Leaf EM applied to the Gini tree (mean rank: 4.70), followed by SLD applied to the CEB tree (mean rank: 4.72). The top performing quantifier varied by tree type, with SLD performing the best for the CQB and CEB trees and Leaf EM for the Gini tree. The performance of ACC was moderate, with a mean rank of 6.10–10.60 across all trees. Temperature Scaling (TS) did not improve SLD performance, indicating temperature scaling is not an effective calibration method for decision trees on our binary quantification tasks.

Both overall and within-tree Friedman tests indicated statistically significant differences in performance across quantification methods ($p < 0.01$). Within each tree type, the Nemenyi critical difference was 1.4245, and across all tree types, the critical difference was 4.9518. Nemenyi post-hoc tests indicated that the Gini tree with Leaf EM was significantly better than the Gini tree with SLD (TS),

PACC, and ACC. SLD when applied to the CQB and CEB trees was significantly better than any quantifier applied to those same trees. In all cases, ACC was significantly worse than the best performing quantifier for that tree type.

Most tree-quantifier pairs exhibited little bias (Figure 1) when averaged across datasets. SLD exhibited a weak bias towards a uniform distribution, while most other methods exhibited weak biases towards the extremes of the prevalence range.

Leaf EM performed best on the Gini tree (mean overall rank: 3.45), followed by a sharp dropoff by the CQB tree (mean rank: 13.66), and then the CEB tree (mean rank: 14.76). Across all quantifiers, the Gini tree tallied the most datasets with the lowest AE, indicating the Gini splitting criterion yields trees that are particularly well-suited for quantification. That is not to say that the CEB and CQB trees do not have their strengths; they each had the best performance for multiple datasets.

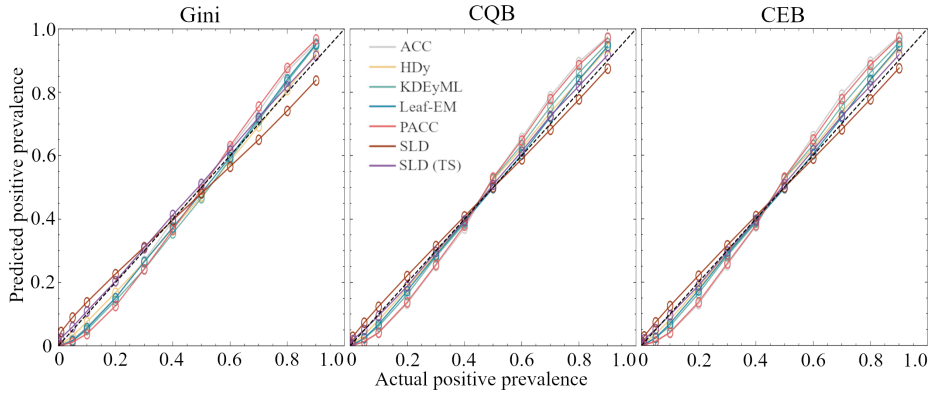


Fig. 1. Model Performance on Binary Quantification Tasks.

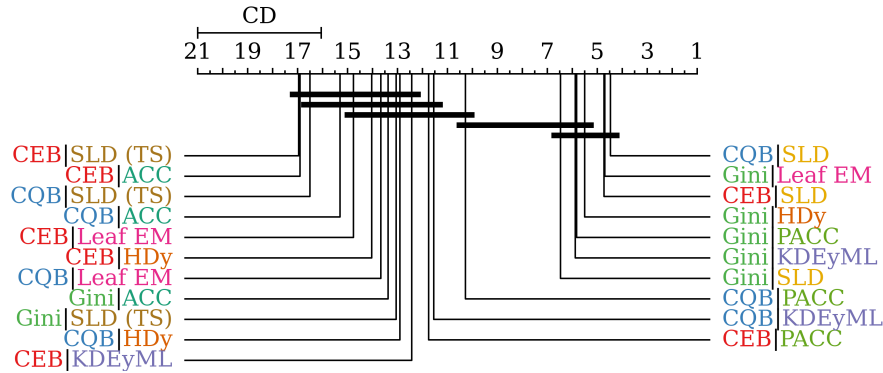


Fig. 2. Critical Difference Diagram for Binary Quantification Tasks.

Table 1. Absolute Error by Binary Dataset, Tree Type, and Algorithm. **Bold** indicates the lowest value in each row overall. *Italic* indicates the lowest value within each tree type per row. A Friedman test indicated statistical significance ($p < 0.05$) of performance differences. Summary rows report mean rank across datasets (Nemenyi critical difference: 4.9518) and mean rank within each tree type (Nemenyi critical difference: 1.4245).

Dataset	Gini					CQB					CEB							
	ACC	HDy	KDEyML	Leaf	EM	PACC	SLD	SLD	(TS)	ACC	HDy	KDEyML	Leaf	EM	PACC	SLD	SLD	(TS)
adult	0.058	0.033	0.038	0.030	0.045	0.031	0.041	0.075	0.074	0.074	0.076	0.072	<i>0.047</i>	0.080	0.077	0.074	0.074	0.079
avila	0.093	0.067	0.066	0.064	0.072	0.090	0.089	0.124	0.132	0.119	0.160	0.115	0.095	0.168	0.125	0.132	0.120	0.161
bike	0.104	0.074	0.069	0.070	0.076	0.071	0.088	0.112	0.104	0.101	0.111	0.103	<i>0.080</i>	0.125	0.112	0.106	0.102	0.115
blog	0.061	0.035	0.038	0.035	0.043	0.056	0.057	0.067	0.060	0.058	0.079	0.057	<i>0.038</i>	0.067	0.067	0.061	0.058	0.080
bc-cat	0.133	0.133	0.133	0.114	0.133	0.097	0.138	0.180	0.180	0.179	0.152	0.178	<i>0.100</i>	0.163	0.180	0.180	0.179	0.152
bc-cont	0.115	0.111	0.112	0.102	0.113	<i>0.073</i>	0.129	0.118	0.117	0.117	0.103	0.117	<i>0.070</i>	0.107	0.120	0.119	0.118	0.103
cars	0.211	0.121	0.124	0.149	0.137	0.087	0.128	0.195	0.178	0.179	0.181	0.187	<i>0.115</i>	0.192	0.222	0.204	0.205	0.202
conc	0.278	0.235	0.231	0.202	0.224	<i>0.183</i>	0.297	0.289	0.275	0.273	0.267	0.268	0.169	0.301	0.295	0.278	0.275	0.271
contra	0.429	0.383	0.379	0.361	0.368	<i>0.318</i>	0.458	0.368	0.369	0.359	0.381	0.353	0.271	0.405	0.380	0.374	0.370	0.389
cappl	0.204	0.134	0.136	0.134	0.145	0.101	0.182	0.287	0.269	0.267	0.255	0.263	<i>0.169</i>	0.300	0.302	0.280	0.277	0.267
ccard	0.214	0.090	0.091	0.089	0.095	0.141	0.204	0.223	0.238	0.222	0.283	0.206	<i>0.194</i>	0.293	0.225	0.239	0.222	0.286
diam	0.050	<i>0.021</i>	0.022	0.023	0.024	0.029	0.035	0.025	0.020	0.020	0.022	0.021	0.013	0.020	0.025	0.021	0.020	0.022
dota	0.527	0.363	0.408	0.489	0.249	0.492	0.573	0.399	0.430	0.441	0.500	0.320	<i>0.456</i>	0.365	0.412	0.422	0.438	0.502
drugs	0.244	0.146	0.147	0.125	0.139	0.110	0.183	0.222	0.220	0.213	0.216	0.214	<i>0.154</i>	0.253	0.240	0.235	0.225	0.230
ener	0.172	0.100	0.100	0.100	0.099	0.149	0.196	0.191	0.214	0.189	0.265	0.181	<i>0.170</i>	0.248	0.191	0.213	0.188	0.266
fifa	0.090	0.046	0.047	0.044	0.055	0.047	0.070	0.079	0.069	0.066	0.072	0.066	<i>0.045</i>	0.072	0.080	0.069	0.065	0.072
flare	0.473	0.430	0.424	0.414	0.417	<i>0.316</i>	0.487	0.428	0.422	0.419	0.438	0.421	<i>0.281</i>	0.474	0.428	0.418	0.416	0.433
grid	0.172	0.133	0.120	0.115	0.119	0.203	0.202	0.184	0.203	0.178	0.217	0.169	<i>0.157</i>	0.261	0.184	0.203	0.178	0.217
ads	0.147	0.132	0.133	0.135	0.133	0.088	0.155	0.177	0.173	0.172	0.162	0.172	<i>0.111</i>	0.182	0.177	0.174	0.172	0.162
magic	0.129	0.080	0.076	0.070	0.081	0.117	0.107	0.113	0.123	0.113	0.129	0.106	<i>0.084</i>	0.129	0.113	0.123	0.113	0.129
boone	0.022	0.015	0.022	0.016	0.019	0.050	0.035	0.029	0.026	0.027	0.037	0.026	<i>0.020</i>	0.027	0.029	0.026	0.027	0.037
mush	0.041	0.034	0.036	0.035	0.038	0.022	0.035	0.045	0.039	0.041	0.037	0.043	<i>0.023</i>	0.038	0.045	0.040	0.041	0.037
music	0.536	0.516	0.516	0.503	0.520	<i>0.461</i>	0.701	0.647	0.573	0.576	0.576	0.591	0.449	0.723	0.647	0.573	0.576	0.576
musk	0.164	0.150	0.148	0.143	0.149	0.152	0.187	0.199	0.197	0.185	0.193	0.183	<i>0.152</i>	0.204	0.203	0.201	0.188	0.196
news	0.295	0.175	0.180	0.196	0.139	0.313	0.468	0.248	0.282	0.277	0.395	<i>0.208</i>	0.286	0.449	0.395	0.208	0.286	0.449
nurse	0.000	0.000	0.001	0.013	0.000	0.006	0.000	0.000	0.000	0.001	0.013	0.000	0.006	0.000	0.013	0.000	0.006	0.000
occup	0.012	0.009	0.010	0.015	0.012	0.007	0.010	0.016	0.014	0.015	0.015	0.015	<i>0.009</i>	0.013	0.016	0.014	0.015	0.015
phish	0.056	0.032	0.033	0.046	0.040	0.025	0.034	0.063	0.062	0.062	0.060	0.062	<i>0.043</i>	0.065	0.064	0.063	0.063	0.061
craft	0.219	0.156	0.152	<i>0.137</i>	0.157	0.161	0.212	0.183	0.169	0.163	0.160	0.161	0.104	0.186	0.184	0.169	0.163	0.160
spam	0.107	0.094	0.095	0.087	0.097	<i>0.083</i>	0.119	0.119	0.112	0.108	0.101	0.108	0.070	0.120	0.121	0.114	0.109	0.103
alco	0.467	0.459	0.452	0.443	0.449	<i>0.372</i>	0.565	0.547	0.508	0.516	0.524	0.513	0.363	0.634	0.562	0.517	0.524	0.531
study	0.211	0.165	0.164	0.134	0.167	0.119	0.164	0.194	0.189	0.189	0.171	0.188	<i>0.121</i>	0.203	0.192	0.187	0.187	0.170
cond	0.050	0.036	0.037	0.031	0.038	0.045	0.041	0.076	0.072	0.066	0.070	0.068	<i>0.049</i>	0.079	0.076	0.072	0.066	0.070
telco	0.217	0.143	0.133	0.133	0.134	0.164	0.208	0.152	0.150	0.143	0.149	0.140	0.103	0.167	0.154	0.150	0.144	0.150
thrm	0.423	0.405	0.405	0.382	0.383	<i>0.348</i>	0.537	0.503	0.449	0.459	0.457	0.452	<i>0.320</i>	0.524	0.504	0.449	0.458	0.456
turk	0.507	0.408	0.418	0.391	0.393	0.398	0.535	0.379	0.378	0.350	0.370	0.350	<i>0.301</i>	0.344	0.399	0.383	0.360	0.370
vgame	0.252	0.158	0.154	0.154	0.155	0.181	0.245	0.204	0.205	0.195	0.214	0.187	<i>0.146</i>	0.258	0.206	0.202	0.196	0.215
voice	0.055	0.046	0.047	0.044	0.051	0.029	0.046	0.092	0.082	0.081	0.071	0.084	<i>0.045</i>	0.077	0.092	0.082	0.081	0.071
wine	0.255	0.177	0.175	0.160	0.172	0.201	0.279	0.232	0.218	0.205	0.240	0.201	<i>0.173</i>	0.257	0.236	0.223	0.208	0.245
yeast	0.382	0.307	0.308	0.298	0.307	0.257	0.408	0.445	0.404	0.406	0.412	0.409	<i>0.315</i>	0.481	0.454	0.414	0.418	0.418
Mean Rank	13.38	5.50	5.88	<i>4.70</i>	5.83	6.47	13.05	15.30	12.90	11.55	13.66	10.28	4.47	16.50	16.90	14.03	12.43	14.76
Mean Rank (per Tree)	6.30	3.20	3.38	2.35	3.80	3.12	5.85	5.53	4.22	3.55	4.50	3.08	1.35	5.78	5.55	4.20	3.45	4.42

5.2 Multiclass Quantification

Performance on multiclass quantification was quite different than on binary quantification (Table 2, Figure 3). The best performing method was SLD applied to the Gini tree, which achieved the lowest absolute error in 12 out of 17 datasets, and the lowest overall mean rank across all tree types. The best performing tree-quantifier combination was SLD applied to the Gini tree (mean rank: 1.65), followed by SLD applied to the CQB tree (mean rank: 2.88), and then SLD applied to the CEB tree (mean rank: 4.65). Across all trees, the ordinal ranking of the top three quantifiers was consistent: SLD performed best, followed by KDEyML, then PACC. The performance of Leaf EM was particularly poor, with a mean rank of 8.29–13.47 across all trees.

A Friedman test indicated statistically significant differences in performance across algorithms, both within each tree type ($p < 0.01$) and across all tree types ($p < 0.01$). The Nemenyi critical difference was 1.8288 within each tree type, and 6.3887 across all tree types. Nemenyi post-hoc tests indicated that SLD applied to the Gini tree was significantly better than Leaf EM, ACC, PACC, and SLD (TS) applied to the Gini tree.

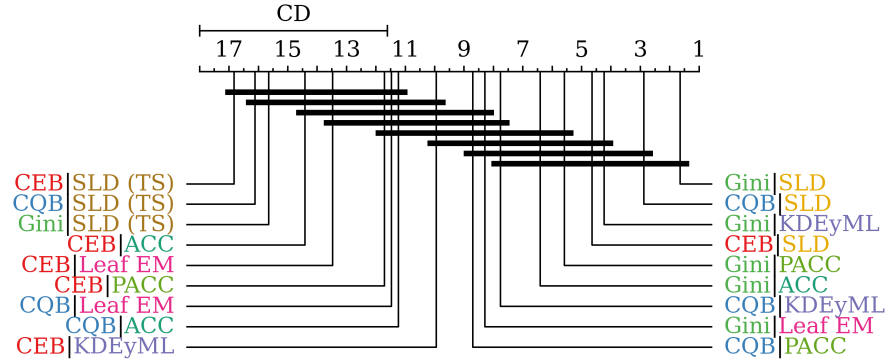


Fig. 3. Critical Difference Diagram for Multiclass Quantification Tasks.

5.3 Degenerate Trees

Overall, 30.4% of trials had at least one degenerate tree, with the Gini tree having the lowest percent of degenerate trials (14.1%) and the CQB and CEB trees having the highest (27.5% and 27.4%, respectively). A trial was considered degenerate if a tree had only a single node, which provides no information to the quantification methods. If these trials are included in the analysis, the ranks of Gini trees improve, with Leaf EM applied to Gini trees achieving the best overall mean rank (Table 3).

6 Discussion

The Gini tree with Leaf EM achieved the best mean rank across all quantifiers applied to a Gini tree for binary quantification (Table 1). The Nemenyi post-hoc

Table 2. Absolute Error by Multiclass Dataset, Tree Type, and Algorithm. **Bold** indicates the lowest value in each row overall. *Italic* indicates the lowest value within each tree type per row. A Friedman test indicated statistical significance ($p < 0.05$) of performance differences. Summary rows report mean rank across datasets (Nemenyi critical difference: 6.3887) and mean rank within each tree type (Nemenyi critical difference: 1.8288).

Dataset	Gini					CQB					CEB										
	ACC	KDEyML	Leaf	EM	PACC	SLD	SLD (TS)	ACC	KDEyML	Leaf	EM	PACC	SLD	SLD (TS)	ACC	KDEyML	Leaf	EM	PACC	SLD	SLD (TS)
bike	0.326	0.270	0.337	0.282	0.209	0.433	0.429	0.340	0.393	0.376	<i>0.275</i>	0.591	0.466	0.372	0.419	0.417	<i>0.285</i>	0.602			
blog	0.309	0.169	0.168	0.217	0.142	0.387	0.356	0.232	0.296	0.316	<i>0.195</i>	0.498	0.370	0.255	0.304	0.312	<i>0.204</i>	0.523			
conc	0.412	0.417	0.476	0.418	0.395	0.572	0.493	0.478	0.549	0.463	<i>0.410</i>	0.708	0.558	0.528	0.612	0.546	<i>0.448</i>	0.723			
contra	0.554	0.598	0.600	0.575	<i>0.475</i>	0.816	0.574	0.537	0.539	0.558	0.428	0.757	0.619	0.562	0.573	0.579	<i>0.442</i>	0.801			
diam	0.053	0.044	0.055	0.048	<i>0.037</i>	0.092	0.056	0.052	0.065	0.052	0.037	0.063	0.066	0.059	0.078	0.057	<i>0.048</i>	0.067			
drugs	0.323	0.329	0.324	0.319	0.267	0.447	0.366	0.364	0.390	0.347	<i>0.289</i>	0.455	0.461	0.444	0.482	0.432	<i>0.366</i>	0.620			
ener	0.292	0.266	0.339	0.243	0.311	0.597	0.320	0.341	0.472	0.281	0.333	0.632	0.344	0.352	0.518	<i>0.313</i>	0.314	0.625			
fifa	0.265	0.161	0.187	0.215	0.128	0.339	0.298	0.211	0.251	0.274	<i>0.158</i>	0.396	0.390	0.265	0.283	0.340	<i>0.191</i>	0.507			
news	0.593	0.426	0.438	0.514	0.383	1.018	0.644	0.548	0.585	0.579	<i>0.447</i>	0.995	0.692	0.605	0.625	0.618	<i>0.467</i>	0.959			
nurse	0.041	0.022	0.160	0.031	0.019	0.020	0.091	0.079	0.105	0.085	<i>0.054</i>	0.078	0.099	0.085	0.111	0.092	<i>0.063</i>	0.086			
craft	0.374	0.354	0.386	0.351	<i>0.341</i>	0.607	0.366	0.357	0.460	0.345	<i>0.305</i>	0.538	0.390	0.362	0.458	0.344	0.297	0.520			
cond	0.155	0.119	0.167	0.133	0.106	0.287	0.204	0.164	0.215	0.181	<i>0.133</i>	0.384	0.262	0.201	0.227	0.209	<i>0.140</i>	0.393			
thrm	0.702	0.713	0.778	0.707	0.590	1.007	0.779	0.751	0.852	0.770	<i>0.611</i>	0.974	0.788	0.762	0.858	0.778	<i>0.623</i>	0.930			
turk	0.621	0.576	0.597	0.582	<i>0.478</i>	0.980	0.690	0.639	0.667	0.658	0.453	1.158	0.702	0.636	0.664	0.667	<i>0.471</i>	1.173			
vgame	0.574	0.566	0.577	0.596	0.437	0.994	0.606	0.586	0.578	0.613	<i>0.465</i>	0.974	0.607	0.566	0.570	0.617	<i>0.468</i>	0.930			
wine	0.639	0.676	0.749	0.708	0.518	1.077	0.712	0.694	0.727	0.727	<i>0.545</i>	1.051	0.771	0.732	0.783	0.772	<i>0.622</i>	1.047			
yeast	0.443	0.439	0.462	0.464	0.278	0.707	0.525	0.553	0.588	0.558	<i>0.412</i>	0.811	0.599	0.589	0.628	0.599	<i>0.471</i>	0.836			
Mean Rank	6.41	4.24	8.29	5.59	1.65	15.65	11.24	7.76	11.47	8.71	2.88	16.12	14.41	9.94	13.47	11.71	4.65	16.82			
Mean Rank (per Tree)	3.59	2.71	4.41	3.35	1.18	5.76	4.12	2.53	4.41	3.12	1.12	5.71	4.35	2.29	4.35	3.18	1.06	5.76			

Table 3. Overall mean rank for binary (Nemenyi critical difference: 4.95) and multi-class (Nemenyi critical difference: 6.39) settings. Bold indicates the best overall method within a setting (comparing all tree-method combinations in that setting); italic indicates the best within each tree type.

Setting	Tree	ACC	HDy	KDEyML	Leaf EM	PACC	SLD	SLD (TS)
Binary	Gini	9.9	4.7	5.3	4.0	5.8	4.3	10.7
	CQB	16.2	13.5	12.8	14.2	11.5	<i>6.3</i>	17.1
	CEB	17.1	14.1	12.9	15.0	12.2	<i>6.2</i>	17.3
Multi	Gini	6.3	–	4.4	8.5	5.8	1.6	15.5
	CQB	11.1	–	7.7	11.5	8.7	<i>3.0</i>	16.1
	CEB	14.3	–	9.9	13.5	11.6	<i>4.7</i>	16.9

test indicated that the Gini–Leaf EM combination was significantly better than ACC, PACC, and SLD (TS) applied to the Gini tree. Within the Gini tree, Leaf EM was not significantly better than KDEyML, SLD, or HDy. SLD applied to CQB performed the best among all quantifier-tree combinations on binary quantification, but CQB and CEB trees had a higher percentage of degenerate trials than Gini trees, indicating it might not be an ideal choice for small, imbalanced datasets. Consistent with how certain quantifiers are best for certain base classifiers, certain splitting criteria are best for certain quantifiers. Leaf EM performed poorly in the multiclass setting. Ultimately, the best tree-quantifier combination will depend on the dataset and the quantification task.

SLD applied to Gini trees performed relatively poorly on binary quantification. This aligns with the expectation that SLD’s performance is dependent on the calibration of the base classifier. SLD maximizes the same likelihood function as Leaf EM, but SLD depends on the calibration of $P(Y | X)$ and is therefore subject to the miscalibration of the base classifier. Decision trees are known to produce poorly calibrated probabilities, and Leaf EM avoids this dependency. Poor calibration is often attributed to the purity driven splitting used to fit a tree [37]. CEB and CQB are splitting criteria that are not directly tied to increasing the purity of the resulting leaves. It follows that CEB and CQB trees are likely to produce better calibrated probabilities than Gini trees, and therefore produce better SLD estimates of prevalence.

The Gini tree outperformed the CEB or CQB trees when used as a base classifier for Leaf EM. The CEB and CQB splitting criteria were designed for the quantification task, while Gini was designed for classification. However, Gini impurity can be directly tied to the Leaf EM approach. Gini impurity at a $Leaf_i$ can be written in terms of class conditionals as

$$G(Leaf_i) = 1 - \sum_{i=1}^{n_Y} \left(\frac{P(Leaf_i | Y_i) P(Y_i)}{\sum_{i'=1}^{n_Y} P(Leaf_i | Y_{i'}) P(Y_{i'})} \right)^2, \quad (4)$$

which is minimized when one $P(Leaf_i | Y_i)$ dominates the others. Gini’s split criterion is therefore used to maximize the purity of the same class conditionals $P(Leaf | Y)$ that Leaf EM consumes.

Leaf EM and KDEyML perform similarly, particularly when applied to the Gini tree. KDEyML approaches the same likelihood function as Leaf EM as

the kernel bandwidth approaches zero, provided no two leaves produce identical classifier scores. The two methods were applied here without hyperparameter tuning: KDEyML used a fixed bandwidth of 0.05. Had the KDEyML bandwidth been tuned, it is plausible that the optimal value would have been close to zero, in which case KDEyML would have approximated Leaf EM directly. The competitive performance of KDEyML on the Gini tree indicates KDEyML may be learning a similar representation of class conditional distributions, but recovering it through a different mechanism.

Schumacher et al. reported that quantification forests and their ACC-corrected variant performed poorly relative to other quantification methods [34]. The results presented here suggest that this failure is attributable in part to the choice of aggregation method rather than to a fundamental limitation of quantification trees. When CEB and CQB trees are paired with most quantifiers rather than ACC, performance improves substantially.

This study was subject to several limitations. Neither Leaf EM nor any of the comparator methods were subjected to hyperparameter tuning. KDEyML used a fixed bandwidth of 0.05, and trees were pruned with a fixed minimum of 5 samples per leaf on the held-out estimation set. Moreo et al. found that tuning the KDEyML bandwidth can substantially improve performance [28]. Additionally, temperature scaling is used to calibrate SLD. While temperature scaling is commonly used in quantification, it is typically evaluated on continuous scores (e.g., from a neural network) rather than discrete tree-leaf predictions. Future work could explore alternative calibration methods that are designed for decision trees, such as kernel density smoothing proposed by Smyth et al. [35]. Finally, removing trials with degenerate trees from the analysis may have biased the results in favor of quantifiers that perform particularly poorly when fit to small, imbalanced datasets.

7 Conclusion

We introduced Leaf EM, a quantification method that pairs decision-tree leaves with Iterative Bayesian Unfolding and requires no assumption of classifier calibration. On binary tasks, SLD applied to a CQB tree achieved the best overall mean rank, while the Gini–Leaf EM combination attained the second best overall mean rank and the best within-tree mean rank on Gini trees. On multiclass tasks, SLD applied to a Gini tree was the strongest tree-quantifier combination. Together, these findings demonstrate that recent reports of poor decision-tree performance on quantification reflect the choice of aggregator, and not necessarily a limitation of decision trees themselves.

Acknowledgments. This study was funded by The National Institute of Allergy and Infectious Diseases (grant number R01AI174932).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bella, A., Ferri, C., Hernández-Orallo, J., Ramirez-Quintana, M.J.: Quantification via probability estimators. In: 2010 IEEE International conference on data mining. pp. 737–742. IEEE (2010)
2. Blobel, V.: Unfolding methods in high-energy physics experiments. Tech. rep., CERN (1984)
3. Blobel, V.: An unfolding method for high energy physics experiments. arXiv preprint hep-ex/0208022 (2002)
4. Börner, M., Hoinka, T., Meier, M., Menne, T., Rhode, W., Morik, K.: Measurement/simulation mismatches and multivariate data discretization in the machine learning era. *Astronomical Data Analysis Software and Systems XXVII* **522**, 431 (2020)
5. Breiman, L., Friedman, J., Olshen, R.A., Stone, C.J.: Classification and regression trees. Chapman and Hall/CRC (1984)
6. Bunse, M.: On multi-class extensions of adjusted classify and count. In: Proceedings of the 2nd International Workshop on Learning to Quantify (LQ 2022). pp. 43–50 (2022)
7. Bunse, M.: Unification of algorithms for quantification and unfolding. In: INFORMATIK 2022. pp. 459–468. Gesellschaft für Informatik, Bonn (2022)
8. Castaño, A., Alonso, J., González, P., del Coz, J.J.: An equivalence analysis of binary quantification methods. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 6944–6952 (2023)
9. D’Agostini, G.: A multidimensional unfolding method based on Bayes’ theorem. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **362**(2-3), 487–498 (1995)
10. D’Agostini, G.: Improved iterative Bayesian unfolding. arXiv preprint arXiv:1010.0632 (2010)
11. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* **7**(Jan), 1–30 (2006)
12. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F., et al.: Learning to quantify. *The information retrieval series* **47** (2023)
13. Esuli, A., Molinari, A., Sebastiani, F.: A critical reassessment of the saeren-slatinne-decaestecker algorithm for posterior probability adjustment. *ACM Transactions on Information Systems (TOIS)* **39**(2), 1–34 (2020)
14. Esuli, A., Moreo, A., Sebastiani, F., Sperduti, G.: A concise overview of LeQua@CLEF 2022: learning to quantify. In: International Conference of the Cross-Language Evaluation Forum for European Languages. pp. 362–381. Springer (2022)
15. Esuli, A., Moreo, A., Sebastiani, F., Sperduti, G.: An overview of LeQua 2024, the 2nd International Data Challenge on Learning to Quantify. In: 4th International Workshop on Learning to Quantify (LQ 2024). p. 51 (2024)
16. Fabris, A., Esuli, A., Moreo, A., Sebastiani, F.: Measuring fairness under unawareness of sensitive attributes: A quantification-based approach. *Journal of Artificial Intelligence Research* **76**, 1117–1180 (2023)
17. Forman, G.: Counting positives accurately despite inaccurate classification. In: European conference on machine learning. pp. 564–575. Springer (2005)
18. Gart, J.J., Buck, A.A.: Comparison of a screening test and a reference test in epidemiologic studies. II. A probabilistic model for the comparison of diagnostic tests. *American Journal of Epidemiology* **83**(3), 593–602 (May 1966). <https://doi.org/10.1093/oxfordjournals.aje.a120610>

19. González, P., Castaño, A., Chawla, N.V., Coz, J.J.D.: A review on quantification learning. *ACM Computing Surveys (CSUR)* **50**(5), 1–40 (2017)
20. González, P., Castaño, A., Peacock, E.E., Díez, J., Del Coz, J.J., Sosik, H.M.: Automatic plankton quantification using deep features. *Journal of Plankton Research* **41**(4), 449–463 (2019)
21. González-Castro, V., Alaiz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the hellinger distance. *Information Sciences* **218**, 146–164 (2013)
22. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: *International conference on machine learning*. pp. 1321–1330. PMLR (2017)
23. Hoecker, A., Kartvelishvili, V.: SVD approach to data unfolding. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **372**(3), 469–481 (1996)
24. 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)
25. King, G., Lu, Y.: Verbal autopsy methods with multiple causes of death (2008)
26. Milli, L., Monreale, A., Rossetti, G., Giannotti, F., Pedreschi, D., Sebastiani, F.: Quantification trees. In: *2013 IEEE 13th International Conference on Data Mining*. pp. 528–536. IEEE (2013)
27. Moreo, A., Esuli, A., Sebastiani, F.: QuaPy: a python-based framework for quantification. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. pp. 4534–4543 (2021)
28. Moreo, A., González, P., del Coz, J.J.: Kernel density estimation for multiclass quantification. *Machine Learning* **114**(4), 92 (2025)
29. Moreo, A., Salvati, N.: An efficient method for deriving confidence intervals in aggregative quantification. In: *5th International Workshop on Learning to Quantify (LQ 2025)*. p. 12 (2025)
30. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011)
31. Provost, F.J., Fawcett, T., Kohavi, R., et al.: The case against accuracy estimation for comparing induction algorithms. In: *ICML*. vol. 98, pp. 445–453 (1998)
32. Sadinle, M., Bouyarmane, K., Galloway, G., Sam, S., Yuan, C., Tutar, I.: Semi-automated estimation of weighted rates for e-commerce catalog quality monitoring (2022), <https://www.amazon.science/publications/semi-automated-estimation-of-weighted-rates-for-e-commerce-catalog-quality-monitoring>
33. 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)
34. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods. *Journal of Machine Learning Research* **26**(55), 1–54 (2025)
35. Smyth, P., Gray, A., Fayyad, U.M.: Retrofitting decision tree classifiers using kernel density estimation. In: *Machine Learning Proceedings 1995*, pp. 506–514. Elsevier (1995)
36. Tasche, D.: Fisher consistency for prior probability shift. *Journal of Machine Learning Research* **18**(95), 1–32 (2017)
37. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. p. 609–616. ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)