2013 IEEE 13th International Conference on Data Mining

Quantification Trees

Letizia Milli*, Anna Monreale†, Giulio Rossetti*, Fosca Giannotti*, Dino Pedreschi† and Fabrizio Sebastiani*
* ISTI-CNR
Via G. Moruzzi, 1 - Pisa, Italy
Email: name.surname@isti.cnr.it
† University of Pisa
Largo Pontecorvo, 3 - Pisa, Italy
Email: {annam,pedre}@di.unipi.it

***Abstract*—In many applications there is a need to monitor how a population is distributed across different classes, and to track the changes in this distribution that derive from varying circumstances; an example such application is monitoring the percentage (or "prevalence") of unemployed people in a given region, or in a given age range, or at different time periods. When the membership of an individual in a class cannot be established deterministically, this monitoring activity requires classification. However, in the above applications the final goal is *not* determining which class each individual belongs to, but simply estimating the prevalence of each class in the unlabeled data. This task is called *quantification*. In a supervised learning framework we may estimate the distribution across the classes in a test set from a training set of labeled individuals. However, this may be suboptimal, since the distribution in the test set may be substantially different from that in the training set (a phenomenon called *distribution drift*). So far, quantification has mostly been addressed by learning a classifier optimized for individual classification and later adjusting the distribution it computes to compensate for its tendency to either under- or over-estimate the prevalence of the class. In this paper we propose instead to use a type of decision trees (*quantification trees*) optimized not for individual classification, but directly for quantification. Our experiments show that quantification trees are more accurate than existing state-of-the-art quantification methods, while retaining at the same time the simplicity and understandability of the decision tree framework.**

I. Introduction

In many real-world applications there is a need to estimate the distribution of a population across different classes, and to track the changes in this distribution that may derive from varying circumstances. Example such applications are estimating the percentage (or "prevalence") of unemployed people across different geographical regions, or age ranges, or genders, or across different time period; or estimating the prevalence of positive comments in a population of textual comments on a given product; or estimating the prevalence of calls related to a specific issue in a population of phone calls to tech support.

This problem is closely related to *density estimation* [22]; a classic, textbook example of density estimation is estimating the prevalence of white balls in a large urn containing white balls and black balls. However, the above-mentioned applications are different from the ones usually addressed in classic density estimation. For instance, the above "urn" example assumes that, when we pick a ball from the urn, we can deterministically assess whether the ball is black or white, by simple visual inspection. The case of textual comments on products is different, since assessing whether a given comment is positive or negative is not a deterministic operation, since it depends on subjective judgment. Another key difference is the fact that the density estimation problem arises from the fact that in many applications it is practically impossible to assess class membership for each single individual (e.g., we do *not* want to inspect every single ball in the urn); however, in the case of product reviews mentioned above it is feasible to analyze every single individual, since this is done automatically.

These differences clearly indicate the existence of a task different from density estimation, and characterized (a) by the need to assess class prevalence when class membership cannot be established deterministically, and (b) by the fact that all individuals that make up the population can be analyzed. Both facts indicate that our task is closely related to *classification*, a task in which facts (a) and (b) both hold. However, the goal of classification is different from the one we have set ourselves, since in classification we are interested in correctly estimating the true class of each single individual. We are instead interested in classifying our individuals with a different goal, that of estimating class prevalence; in the literature, this task has been called *quantification* [5], [8], [9], [24], [26].

Classification and quantification are different because, while a perfect classifier is also a perfect quantifier, not necessarily a good classifier is also a good quantifier. For instance, a binary classifier for which $FP = 20$ and $FN = 20$ is a worse classifier than one for which, on the same test set, $FP = 0$ and $FN = 10$, but is a better quantifier[1]; indeed, it is a perfect quantifier, since FP and FN are equal and thus compensate each other, so that the distribution of the test items across the class and its complement is estimated perfectly.

So far, quantification has mostly been addressed by learning a standard classifier (i.e., optimized for individual classification) followed by a post-processing phase, where the distribution resulting from the classifier is adjusted to compensate for the estimated tendency of the classifier to either under- or over-estimate the prevalence of the class. The basic idea of this paper is instead to take quantification into account not only at post-processing time, but also at learning time, by designing a specific learning methodology that is optimized

[1] As usual, TP, FP, FN, TN, indicate the numbers of true positives, false positives, false negatives, true negatives, respectively.

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.122

for quantification, instead of classification. To this purpose, we directly operate on the learning algorithm and propose a new type of decision tree, called a *quantification tree*, optimized not for individual classification, but directly for (collective) quantification.

Our experiments show that the use of quantification trees yields more accurate quantification than existing state-of-the-art quantification methods, while retaining at the same time the simplicity and understandability that are typical of the decision tree framework. Moreover, our quantifiers exhibit better resilience with respect to sharp changes of the class distribution in the test set, thus enabling more reliable estimates even in situations characterized by high "distribution drift".

We believe that the potential impact of robust quantification methods, such as those introduced in this paper, is high, especially in big data analytics and official statistics. One promising approach is the use of labelled data obtained from surveys to learn a reliable quantifier of a phenomenon, and then apply the quantifier to big data to monitor the phenomenon across different geographical areas or time ranges, even if the distribution varies substantially across space and time. An example of this approach is the socio-meter of urban population proposed in [11], for estimating the proportion of city users that fall into three categories: residents, commuters, visitors. In this study, a massive dataset of mobile phone call detail records (CDR) is used to characterize the call profiles of the people observed in an urban space along a few weeks. The call profile of a city user captures how often the person calls during weekdays or weekends across the entire observation period, as well as how often during early morning hours, working hours, or night hours. By means of a focused campaign, a small fragment of the available call profiles are labelled with one of the three classes; a classifier is then learned on the labelled fragment of dataset; finally, the classifier is applied to the entire population of mobile phone users in the dataset, in order to continuously estimate the proportion of residents, commuters and visitors in town, at any moment in time. Remarkably, the call profiles of the three classes of city users are stable in time: residents call essentially anytime, commuters tend to call only during weekdays and working hours, visitors call sporadically. Instead, the proportion among the three categories tend to vary significantly over time, and monitoring these variations is an important information for planning urban services, such as waste management, energy supply or public transport. Clearly, an accurate and robust quantification method is at the heart of this very promising approach to monitor social indicators by means of big data — especially in presence of high distribution drift with relatively stable profiles of class membership.

The remainder of the paper is organized as follows. Section II introduces background notions and defines the quantification problem. Section III describes our approach to customizing decision trees for quantification. In Section IV we propose the methods for the learning of decision trees for quantification. Section V shows the empirical evaluation of our methods. In Section VI we discuss the related works and finally, Section VII concludes the paper.

II. Quantification

Quantification is closely related to classification, but their final goals differs. Quantification aims at finding the class frequencies in a set of unlabeled data, while classification aims at determining the class of each specific item in the same dataset. In other words, a quantifier does not care about perfectly predicting the class of a single item, but to guess the global trend of the classes in a new set of data.

In this paper, we address the quantification problem, and propose a method based on decision trees that can be applied to binary and multiple class labels. In the following, we formally define the quantification problem and introduce the basic notions. Then, we review the few methods for quantification proposed in the literature.

A. Problem Definition

Given a collection of records $D = \{r_1, \ldots, r_m\}$, where each r_j has a class label $r_j.class$ belonging to a set of classes $C = \{c_1, c_2, \ldots, c_n\}$, a classifier f is a function $f : D \to C$ that assigns a class label $c_i \in C$ to each record $r_j \in D$.

The actual frequency of a class c_i with respect to a dataset D is $freq_D(c_i) = \frac{|\{r_j \in D | r_j.class = c_i\}|}{|D|}$. The estimated frequency using the learnt function f, namely the result of the classifier, is $\widehat{freq}_D(c_i) = \frac{|\{r_j \in D | f(r_j) = c_i\}|}{|D|}$

We denote by $\mathcal{T}_r$ the training set on which we train our model for quantification aims, while we use $\mathcal{T}_e$ to denote the test set on which we test the performance of our quantifier.

We use the standard notation to indicate the set of *true positives* (TP), *false positives* (FP), *true negatives* (TN) and *false negatives* (FN) of a binary classifier. We use $tpr = \frac{TP}{TP+FN}$ to denote the *true positive rate* and $fpr = \frac{FP}{TN+FP}$ to denote the *false positive rate*.

Now we are ready to formally define the quantification problem.

Definition 1 (Quantification Problem): Let $C = \{c_1, c_2, \ldots, c_n\}$ be a set of class labels. Given a partition of the labelled dataset D in training set $\mathcal{T}_r$ and test set $\mathcal{T}_e$ and a classifier f learnt from $\mathcal{T}_r$, the quantification problem consists in finding the best estimation of the class label distribution in $\mathcal{T}_e$, i.e., $\forall c_i \in C$ we want to minimize the difference between the actual frequency $freq_{\mathcal{T}_e}(c_i)$ and the estimated one $\widehat{freq}_{\mathcal{T}_e}(c_i)$.

The following example highlights the final goal of the quantification task.

Example 1: Consider the training set in Figure 1 (left) of 65 records with three nominal attributes ("sex", "marital status" and "race") and a class attribute with two possible values ("employed" and "unemployed"). The number (n) represents how many times the combination of values occurs.

Now, consider the test set in Figure 1 (right) composed of 145 instances, where 105 belongs to the class "unemployed" and 40 to the class "employed".

Standard classification uses the training set for learning a classifier capable to accurately guess the class of each record of the test set. As an example, a good classier f may yield the following confusion matrix, representing FP, FN, TP and TN in Table I.

Sex	Marital Status	Race	Class
f	yes	white	employed (10)
m	yes	white	unemployed (12)
f	yes	notwhite	unemployed (5)
m	no	white	unemployed (5)
m	no	white	employed (15)
m	no	white	unemployed (4)
m	yes	notwhite	unemployed (5)
m	no	notwhite	unemployed (5)
f	no	white	employed (4)

Sex	Marital Status	Race	Class
f	yes	notwhite	unemployed (30)
f	yes	notwhite	employed (20)
m	no	white	unemployed (15)
f	no	white	employed (20)
f	no	notwhite	unemployed (60)

Fig. 1. Training set (left) and Test set (right)

	unemployed	**employed**
unemployed	90	15
employed	40	0

TABLE I. CONFUSION MATRIX ON THE TEST SET: ROWS ARE ACTUAL CLASS LABELS, WHILE COLUMNS ARE PREDICTED CLASS LABELS

The quantification task tries to estimate the distribution of the two classes; an optimal quantifier in this case returns an estimated frequency of class "unemployed" of 72.41% and one of class "employed" of 27.59%. While the estimation that can be inferred from the confusion matrix in Table I is $\frac{90+40}{145} = 89.65\%$ for class "unemployed" and $\frac{15}{145} = 10.34\%$ for class 'employed".

B. Quantification methods via classification

The typical approach in the literature for addressing the quantification problem is based on standard *classification*. The idea presented in the existing works [7], [8], [9] is to use a standard classifier and then post-processing the results with specific methods to improve the quantification accuracy.

The proposed methods that solve quantification via classification address the case of binary class label, but they can be extended to multi classes. In [9] the following methods are introduced:

Classify and Count. This simple method generates a classifier from the training set $\mathcal{T}_r$, classifies the unlabeled records in $\mathcal{T}_e$, and estimates for each class c_i its frequency $freq_{\mathcal{T}_e}(c_i)$ by counting the fraction of records in T_e that belong to c_i. We indicate the estimation computed by this method by $freq^{CC}_{\mathcal{T}_e}(c_i)$.

Adjusted Classify and Count. This methods improves the result obtained by the previous by adjusting the quantification obtained by *Classify and Count* $freq^{CC}_{\mathcal{T}_e}(c_i)$ with the information about the true positive rate and false positive rate with respect to the training set:

$$freq^{AC}_{\mathcal{T}_e}(c_i) = \frac{freq^{CC}_{\mathcal{T}_e}(c_i) - fpr_{\mathcal{T}_r}}{tpr_{\mathcal{T}_r} - fpr_{\mathcal{T}_r}}. \quad (1)$$

However, standard classifiers, optimized for predicting the class of single records, are not optimal for quantification. Indeed, we observe that, a good classifier is one with a very small number of FN and FP. While the key point of a good quantifier is to balance the errors of classification, hence a classification model with a equal number of FP and FN is perfect because this leads to a correct distribution of the classes.

Example 2: Continuing with Example 1, a quantifier that uses a standard classifier with the confusion matrix in Table I will give the following result by using the Classify and Count approach: $freq^{CC}_{\mathcal{T}_e}(unemployed) = \frac{90+40}{145} = 89.65\%$. A

	unemployed	**employed**
unemployed	75	30
employed	30	10

TABLE II. CONFUSION MATRIX ON TEST SET OF A QUANTIFIER.

better quantifier can be obtained using an ad-hoc classifier optimized for quantification, such as one which yields the confusion matrix in Table II. In this case $freq^{CC}_{\mathcal{T}_e}(unemployed) = \frac{75+30}{145} = 72.41\%$, which perfectly estimates the real frequency of the class. This is due to the fact that false positives and false negatives compensate, and therefore the column totals coincide with those of a perfect classifier with $FP = FN = 0$. Notice that the classifier in Table II is less accurate than that in Table 1, albeit the latter is an optimal quantifier.

Moreover, another problem in the use of standard classifiers for quantification is due to the fact that usually supervised learning algorithms are based on the assumption that both training set and test set have the same distribution. However, often in real-world scenarios this assumption is not true and clearly, the quantification task is particularly useful in cases where the distribution changes.

III. DECISION TREES FOR QUANTIFICATION

In this paper, we radically depart from the use of standard classification models for quantification and propose to directly learn a model optimized for quantification. To this purpose, we customize decision tree learning [3], [18] for quantification.

We call this type of trees *quantification trees*. Our idea is to use a quantification tree instead of a standard classifier in the methods for quantification, presented in Section II-B.

Providing a method for building a decision tree directly optimized for quantification means: (1) defining a measure for evaluating the goodness of a split on attribute at a decision node in order to select the best split; and (2) defining a stopping condition that is necessary to terminate the tree-growing process.

A. Measures for Selecting the best split

The definition of a measure of evaluation of a split is the most important point for obtaining a decision tree optimized for quantification. Intuitively, we need to develop a new feature selection method for the split at a decision node that selects the attribute providing the best balance between false positive and false negative errors in the tree. We call this measure *quantification accuracy* and we can measure it in different ways. In this paper we propose two approaches.

a) Classification Error Balancing: Given a possible split, for each class $c_i \in C$ we compute the difference between the number of false positive (FP_{c_i}) and the number of false negative (FN_{c_i}):

$$E_{c_i} = |FP_{c_i} - FN_{c_i}|.$$

This measure describes the total number of instances classified in a wrong way with respect to class c_i. For the quantification goal $E_{c_i} = 0$ is the best result. After the split we have a new vector QE_1 for the tree, where each element i represents the error of quantification for the class c_i: $QE_1 = [E_{c_1}, E_{c_2}, \ldots, E_{c_n}]$. A measure of *quantification accuracy* is the L2-norm $||QE_1||_2$ of the error vector QE_1.

b) Classification-Quantification Balancing: The previous method takes into consideration only the goal of the quantification task. An alternative measure may try to find a trade-off between classification and quantification. To this end, given a possible split, for each class $c_i \in C$ we compute the following equation

$$\overline{E}_{c_i} = |FP_{c_i}^2 - FN_{c_i}^2| = (|FP_{c_i} - FN_{c_i}|) \times (|FP_{c_i} + FN_{c_i}|)$$

where $\overline{E}_{c_i}$ represents the error of classification with respect to class c_i. Clearly, we want low values for $\overline{E}_{c_i}$ that intuitively means a low value for $(|FP_{c_i} - FN_{c_i}|)$ (i.e., a good quantifier) and a low value for $(|FP_{c_i} + FN_{c_i}|)$ (i.e., a good classifier). After the split we have a new vector QE_2 for the tree, $QE_2 = [\overline{E}_{c_1}, \overline{E}_{c_2}, \ldots, \overline{E}_{c_n}]$. Also in this case we use the L2-norm $||QE_2||_2$ of the error vector QE_2 for measuring quantification accuracy.

B. Goodness of a split & Stopping Criterion

To determine the goodness of the split we need to compare the quantification accuracy before splitting (at parent node) with the quantification accuracy after splitting (at child node). The larger their difference, the better the test condition. The *gain*, Δ, is a criterion that can be used to determine the goodness of a split:

$$\Delta = ||QE_r^{parent}||_2 - ||QE_r^{child}||_2.$$

where QE_r^{parent} denotes the quantification error before the split and QE_r^{child} the quantification error after the split. Here, r is 1 if we use the *Classification Error Balancing* method and 2 if we use the *Classification-Quantification Balancing* method. Clearly, for building a good decision tree the best choice is the selection of a test condition that maximizes the gain Δ.

As explained above, to terminate the tree-growing process we need to define a stopping condition. We base our stopping decision on the gain value: if the gain Δ is not greater than zero there is no possible split for the decision node, therefore the node is a leaf of the final tree.

IV. Quantification Tree Learning

In the following we provide two methods for solving the quantification task by quantification trees. The first one constructs a decision tree by recursively selecting the best attribute to split the data and expanding the leaf nodes of the tree until the stopping criterion is met (Section IV-A). The second method adopts the *Random Forests* technique [2] introduced for classification, making it targeted to quantification (Section IV-B).

A. Recursive Quantification Tree Learning

The recursive method for quantification tree learning is detailed in Algorithms 1 & 2. Algorithm 1 takes as input the training set $\mathcal{T}_r$ and returns a quantification tree. It recursively selects the best attribute to split the data and expands the leaf nodes of the of the tree until there is no split that leads to a positive gain. In particular, Algorithm 1 computes the initial vector

Algorithm 1 BuildQuantifier($\mathcal{T}_r$)

Require: $\mathcal{T}_r$: training set
Ensure: the quantification tree
1: **for each** c in {Classes} **do**
2: $QE_r[c] \leftarrow |FP_c - FN_c|$;
3: **end for**
4: QAccuracy←UPDATEQACCURACY(QE_r);
5: //build the tree
6: **return** root←BUILDTREE($\mathcal{T}_r$, QE_r,QAccuracy)

of quantification errors QE_r and hence, the corresponding quantification accuracy $||QE_r||_2$. Then, it calls the *BuildTree* Algorithm (Algorithm 2) for creating the tree.

The first action of the *BuildTree* algorithm is to determine which attribute should be selected as test condition for splitting the training set. The *findBestSplit()* function performs that task. As previously stated, the choice of attribute depends on the evaluation of quantification accuracy measure that determines the goodness of a split. Clearly, if the split does not lead to a positive gain ($\Delta \leq 0$) then the node becomes a leaf of the tree and the algorithm assigns to it the majority class (*Classify()* function); otherwise, after splitting the data, it updates the vector describing the quantification errors (QE_r), computes the new quantification accuracy $||QE_r||_2$ and recursively calls the function.

Algorithm 2 BuildTree($\mathcal{T}_r$, QE_r, QAccuracy)

Require: $\mathcal{T}_r$: training set, QE_r: quantification error vector of classes, QAccuracy: 2-norm of the error vector QE_r
Ensure: Node
1: FINDBESTSPLIT( );
2: **if** $\nexists$ split **then**
3: leaf←CREATENODE( );
4: leaf.label← CLASSIFY($\mathcal{T}_r$)
5: **return** leaf;
6: **else**
7: Data←split the dataset;
8: QE_r ←UPDATEQE_r( )
9: QAccuracy←UPDATEQACCURACY(QE_r);
10: root←CREATENODE( );
11: let V be a possible outcome of FINDBESTSPLIT( );
12: **for each** v in {V} **do**
13: child←BUILDTREE(v.Data,QE_r,QAccuracy);
14: add child as a descendent of root and label the edge (root→child) as v
15: **end for**
16: **end if**
17: **return** root;

B. Random Forests

We propose an alternative quantification tree learning algorithm for quantification by decision trees. The basic idea is to use a variant of *Random Forests* [2] for applying the well-known *Wisdom of the crowd* theory [12], [23]. *Wisdom of the crowd* is a sociological theory stating that to answer a question is better the collective opinion of a group of individuals rather than a single expert. An intuitive explanation for this phenomenon is that there is noise associated with each individual judgment, and taking the average over a large number of responses tends to cancel the effect of this noise. Thus, achieving better predictions. The work in [23] provides some criteria that guarantee that the theory works:

- each individual has a different opinion;
- the opinion of a person does not have to influence the other individuals;
- nobody has to be a leader;
- the judgments have to be aggregated for obtaining a final result.

Now the question is: *how do we apply the Wisdom of the crowd theory in our setting?* We propose to use the basic idea of random forests for decision trees. This class of methods combines the prediction made by different decision trees, where each tree is generated by using a set of independent random vectors. These vectors are generated by a fixed probability distribution. Typically, each tree gives a specific prediction and the final result is the prediction with the major number of votes. Our idea instead is to use quantification trees instead decision trees and the final prediction is given by averaging the estimations of all trees.

A skeleton of the algorithm that combines a *Random Forests* approach with the *Wisdom of the Crowd* theory is presented in Algorithm 3. The input of the this algorithm consists in: the whole training set $\mathcal{T}_r$, the number of *quantification trees* k to be built and the number of records of each training set for constructing each tree. This dimension of each training set is expressed as a percentage p of $\mathcal{T}_r$. The algorithm returns the set of quantification trees that can be used for the quantification task.

The procedure for selecting the data for learning each tree is as follows. Suppose that the number of features of the original training set is d. We construct k different quantification trees and for the construction of each tree we randomly take $\log_2 d + 1$ features from the original training set. In other words, first we select $p\%$ of records from the original training set and then, we select only $\log_2 d + 1$ random features. In this way, we obtain the training set $\mathcal{T}_{r_i}$, that is used for building a quantification tree as described in the previous section. Intuitively, the data $\mathcal{T}_{r_i}$ simulate the knowledge of an individual that uses it for expressing an opinion. The different $\mathcal{T}_{r_i}$ express the diversity of knowledge that each participant in the crowd of quantifier has.

Finally, given a test set $\mathcal{T}_e$, each quantifier Φ_i returns for each class $c_j \in C = \{c_1, c_2, \ldots, c_n\}$ an estimation of its frequency ($freq^{\Phi_i}_{\mathcal{T}_e}(c_j)$) using the formula (1) and the recursive algorithm; then, it returns the frequency estimation as the average of the estimations of all quantifiers; in formula $\forall j = 1, \ldots, n \; freq_{\mathcal{T}_e}(c_j) = \frac{\sum_{i=1}^{k} freq^{\Phi_i}_{\mathcal{T}_e}(c_j)}{k}$.

Note that, the building of the quantification tree set composing the forest is completely parallelizable in $m \leq k$ sites. That interesting property allows generating an high number of trees (opinions) exploiting in this way as much as possible the *Wisdom of the Crowd* theory.

Algorithm 3 Random Forest Quantifier

Require: $\mathcal{T}_r$: training set, k: number of quantification trees, p: percentage of records of the training set $\mathcal{T}_r$.
Ensure: Set of quantification trees
1: **for** i=1 to k **do**
2: Generate a training set $\mathcal{T}_{r_i}$ from $\mathcal{T}_r$ with size p w.r.t. $\mathcal{T}_r$;
3: Select randomly $\log_2 d+1$ features F from the original data ;
4: generate a quantifier Φ_i from $\mathcal{T}_{r_i}$ and F;
5: **end for**

V. EXPERIMENTS

We now present our experimental findings. We used three datasets: Adult, Bank and Magic[2].

Adult is composed by a clean set of records extracted from the 1994 U.S. census data. It contains 32561 instances defined by 14 attributes. The binary class label represents the income level of the individual respondents ($<= 50K$ or $> 50K$); over the whole dataset the latter class (the rich) has a frequency of 24.08%.

Bank Marketing is composed by data from a telephone marketing campaigns of a Portuguese bank: clients were

[2]Datasets are available in the UCI Machine Learning repository website: http://archive.ics.uci.edu/ml/

contacted to asses their intentions regarding the subscription of a proposed bank deposit. The dataset is composed by 45211 records and 17 attributes: 88.30% of the contacted clients fall in the "no" class.

MAGIC Gamma Telescope: is composed by data of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope: the class variable defines whether an energy particle is a gamma (signal) or an hadron (background). The dataset is composed by 19020 records and 11 attributes: 35.16% of the energy particles fall in the "hadron" class.

Table III summarizes the main information about the used datasets.

Name	Record $\mathcal{T}_r$	Record $\mathcal{T}_e$	Class label	Frequency
Adult	8000	4000	$> 50K$,$<= 50K$	$> 50K$ 24.08%
Bank	4450	1100	yes, no	no 88.3%
Magic	5630	1408	gamma, hadron	hadron 35.16%

TABLE III. DATASETS.

A. Evaluation

The methodology used to evaluate the accuracy of a quantifier differs substantially from that ones used to evaluate a classifier: the main reason is that, for the latter, the prediction given for a single instance could be considered proper or not independently from the ones of others instances belonging to the same test set whereas quantification requires a collective evaluation. In our case the focus is on the distribution of the classes: while a classifier is well suited test and the training sets with similar class distribution, a quantifier needs to be resilient to distribution changes. To evaluate the stability of our proposed methods w.r.t. distribution drift we set up the following experimental protocol: for each dataset we fixed the cardinality of the training and test set (i.e. Adult 8000 and 4000, Bank 4450 and 1100, Magic 5830 and 1408 respectively) and built 19 new training and test sets by varying the class distribution (from 0.05 to 0.95), preserving the fixed cardinality; in the final step each test set was used to evaluate the quantifiers learned upon all training sets.

In order to evaluate the accuracy of a quantifier we need to compare $\widehat{freq_{\mathcal{T}_e}}(c_i)$, the frequency computed for the class c_i, with $freq_{\mathcal{T}_e}(c_i)$, its actual frequency. Different measures have been used in the literature for measuring quantification accuracy: the most convincing among the ones proposed so far is the one used by Forman in [9], which uses normalized cross-entropy, better known as Kullback-Leibler Divergence (KLD), defined as:

$$KLD\left(freq_{\mathcal{T}_e}(c_i)||\widehat{freq_{\mathcal{T}_e}}(c_i)\right) = \sum_{i=1}^{n} freq_{\mathcal{T}_e}(c_i)\log\frac{freq_{\mathcal{T}_e}(c_i)}{\widehat{freq_{\mathcal{T}_e}}(c_i)}.$$

KLD is introduced in order to evaluate the information loss when $\widehat{freq_{\mathcal{T}_e}}(c_i)$ is used as approximation of $freq_{\mathcal{T}_e}(c_i)$. It ranges in the interval $[0, +\infty)$: assumes value 0 when the two frequencies are equal for each C_i and tends to $+\infty$ when their values diverge. If $\widehat{freq_{\mathcal{T}_e}}(c_i) = 0$ for at least one class, KLD is not defined: for this reason, as in [9], we add a small amount ϵ (set to $\frac{|\mathcal{T}_e|}{0.5}$) to both nominator and denominator in the log function.

The experiments were performed on a Quad Core Intel i7 64 bits @ 3.2 GHz, equipped with 8 GB of RAM and with a kernel Linux 3.0.0-12-generic (Ubuntu 12.04). The code, both for the introduced quantifiers and the baseline (i.e., Forman's CC and AC) was developed in Java using Weka (ver. 3.6.9). Table IV reports the legend for the acronyms of the various quantifiers used in the following discussion. Note that, we apply the AC post-processing also to our quantifiers because this post-processing, able to improve the quantification accuracy of a general classifier, in our case permits to reach more and more precise estimations.

We have used 10-fold cross-validation to estimate the two quantities $fpr_{\mathcal{T}_r}$ and $tpr_{\mathcal{T}_r}$. We have run experiments with $C4.5$, $RF_{C4.5}$ and SVM with the parameters set at their default values.

For each dataset, given a training set, we evaluated the accuracy of the quantifiers over all 19 test sets: results are shown, in an aggregate, in Table V (Adult), VI (Bank) and VII (Magic). We reported, for each quantifier, mean μ and variance σ^2 of KLD of predicted and actual quantification, obtained by fixing a training (resp., test) and varying the test (reps., training). Three cases H, M and L of different degrees of class distribution unbalance were defined, adopting the following criterion:

- **H**: training (resp., test) sets with highly unbalanced class distributions (i.e. [0.05,.15] and [0.85,0.95]);
- **M**: training (resp., test) sets with moderate unbalanced distributions (i.e. [0.2,0.3] and [0.7,0.8]);
- **L**: training (test) set with low unbalanced distribution (i.e. [0.35,0.65]).

The idea is to analyze two different characteristics of the compared quantifiers: (i) by fixing the training and computing the average of the KLD w.r.t. test sets with different distributions, we want to understand which quantifier build the more resilient model given a specific distribution of the initial data, (ii) by fixing the test and computing the average of the KLD obtained varying the models used to quantify it, we want to identify the overall accuracy for each quantifier given the real class distribution of the target data.

1) Comparison of Decision-Tree Based Models: A first analysis regards the accuracy of the proposed methodologies. Based on the values reported for both Adult and Bank datasets we observe how, in general, all our quantifiers that exploit an adjusted classify and count (AC) strategy behave better than the ones relying only on classify and count (CC). In particular we can notice how random forest boosting, applied both to EB and CQB, led to the highest accuracy and stability when variations in distribution of the data occur (either when the training or the test were fixed). Comparing our quantifiers with C4.5 reveals that neither exploiting an AC post-processing nor random forest boosting are able to reduce the gap from our top methods. Classification trees become more reliable when class distribution tends to be balanced: even in this case, however, trees optimized to reduce quantification error led to considerably lower values of KLD.

Identifier	Description	Identifier	Description
$AC(Q_{EB})$	Tree with Classification Error Balancing	$AC(RF_{EB})$	Random Forest of $AC(Q_{EB})$
$AC(Q_{CQB})$	Tree with Classification & Quantification Balancing	$AC(RF_{CQB})$	Random Forest of $AC(Q_{CQB})$
$AC(C4.5)$	Classification Tree	$AC(RF_{C4.5})$	Random Forest of $AC(C4.5)$
$AC(SVM)$	Forman's AC SVM		

TABLE IV. ACRONYMS USED FOR THE DIFFERENT QUANTIFIERS. FOR OUR QUANTIFIERS WE REPORT ONLY THE ADJUSTED CLASSIFY AND COUNT LABELS (AC): CC ONES FOLLOWS THE SAME NAMING LOGIC (I.E., CC(QUANTIFIER NAME)).

	Training fixed						Test fixed					
	H		M		L		H		M		L	
Method	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
$AC(Q_{EB})$	0.0156	0.0033	0.0045	2.02E-5	0.0897	3.38E-4	0.0409	0.003	0.05	6.77E-5	0.0292	8.66E-5
$AC(Q_{CQB})$	0.0272	0.0031	0.0073	1.69E-5	0.0098	6.56E-7	0.0297	6.54E-7	0.0097	1.46E-5	0.0055	6.54E-7
$AC(RF_{EB})$	**0.002**	8.98E-7	**0.0003**	1.98E-8	**0.0002**	1.04E-8	0.0013	6.48E-10	**0.0006**	7.90E-9	**0.0005**	1.01E-6
$AC(RF_{CQB})$	**0.002**	1.39E-7	0.0004	3.84E-9	0.003	9.77E-10	**0.0012**	1.37E-7	0.0007	5.00E-9	0.0006	8.47E-9
$CC(Q_{EB})$	0.3534	0.0167	0.12	0.0115	0.1147	0.006	0.2938	0.0183	0.1729	0.00972	0.1204	0.0042
$CC(Q_{CQB})$	0.3098	0.0178	0.08	0.0131	0.044	0.0087	0.201	0.0056	0.1266	0.0113	0.0972	0.0056
$CC(RF_{EB})$	0.4356	0.029	0.1506	0.0197	0.0907	0.0125	0.3245	0.0079	0.1989	0.0176	0.1445	0.033
$CC(RF_{CQB})$	0.461	0.0295	0.1697	0.0195	0.0896	0.013	0.3369	0.0339	0.2127	0.0177	0.1591	0.0077
$AC(C4.5)$	0.014	3.79E-5	0.0052	8.04E-6	0.0033	2.66E-6	0.0125	3.95E-5	0.006	7.55E-6	0.0039	1.38E-6
$CC(C4.5)$	0.31	0.0123	0.1051	0.0082	0.0452	0.006	0.2014	0.015	0.1369	0.0073	0.111	0.0033
$AC(RF_{C4.5})$	0.2939	0.0083	0.1007	0.0051	0.0412	0.0036	0.1936	0.01	0.1285	0.0045	0.1033	0.0017
$CC(RF_{C4.5})$	0.2973	0.0085	0.1019	0.0053	0.0419	0.0037	0.1955	0.0103	0.13	0.0047	0.1047	0.0018
$AC(SVM)$	0.0156	1.15E-5	0.0006	3.24E-6	0.0003	8.04E-7	0.0054	1.47E-5	0.0048	1.83E-6	0.0056	2.41E-7
$CC(SVM)$	0.6367	0.0118	0.1096	0.0084	0.0427	0.0062	0.3054	0.0129	0.2406	0.0052	0.2144	0.0014

TABLE V. ADULT DATASET: MEAN μ AND VARIANCE σ^2 OF KLD OF PREDICTED AND ACTUAL QUANTIFICATION FOR ALL TESTED QUANTIFIERS W.R.T. HIGH, MODERATE AND LOW CLASS DISTRIBUTION UNBALANCE IN THE TRAINING SETS (LEFT) OR THE TEST SETS (RIGHT) OF FIXED SIZE. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

2) Comparison with Forman's SVM: Using our implementation of AC and CC Forman's strategies for SVM as baseline, we obtain interesting results. Our top quantifiers ($AC(RF_{EB})$ and $AC(RF_{CQB})$) show, for both datasets, greater accuracy in case of highly unbalanced training than SVM (H class). When the unbalance in the training distribution decreases (classes M and L), the average accuracy for the two classes of methods becomes alike: however, is worth noting how in such cases the σ^2 for SVMs is higher w.r.t. that of quantification trees (see Table VI) meaning that the latter shows fewer dispersion (higher accuracy stability) inside the considered classes.

A discussion apart is needed when analyzing the average accuracy outcome when the test class distribution is fixed. In this case the overall accuracy for quantification trees is consistently better w.r.t. SVM's. On the basis of these empirical results, we can conclude that, the proposed quantifiers lead to a higher average accuracy (regardless the distribution of the training set used to build the model) and to a comparable resilience to changes in the distribution of the test w.r.t. the best baselines. This is a remarkable outcome, given the superior classification accuracy of the SVM model compared to decision trees. Despite this, Our variants of decision-tree based techniques yield simpler, more accurate and more resilient quantifiers.

VI. RELATED WORK

The earliest mention of the quantification problem we are aware of is to be found in [17], where the task is simply called *counting*. However, it is not until 2005 that quantification was addressed as a task in its own right, in the work of Forman and colleagues [7], [8], [9], [10]; in this series or papers, they propose several quantification methods and the measure (KLD) which is now the standard evaluation measure for quantification. Bella et al. [1] later introduced probabilistic version of Forman's methods discussed in Section II-B. Esuli and Sebastiani [6] use a variant of SVMs that uses KLD as a loss, and were thus the first to propose the use of a learning algorithm specifically devised for quantification.

Quantification has been applied to several domains. For example, [9] uses it to determine the prevalence of support-related issues in incoming telephone calls received at customer support desks, while [4] use it to estimate the prevalence of response classes in open-ended answers obtained in the context of market research surveys. [14] applies quantification for estimating the distribution of support for different political candidates within blog posts, while [13], [20] apply it for estimating the prevalence of damaged cells in sperm samples for veterinary applications. Differently from all of the above, Xue and Weiss [26] use quantification with the goal of improving the accuracy of classification. Quantification has been studied also in the context of networked data [24], where the goal is estimating class prevalence among a population of nodes in a network.

We should also stress that no work in the (small) published literature proposes learning models explicitly optimized for quantification; *a fortiori*, our paper is also the first to proposes

Training fixed							Test fixed					
	H		M		L		H		M		L	
Method	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
$AC(Q_{EB})$	0.0185	1.61E-4	0.006	3.88E-5	0.0984	3.26E-4	0.0595	1.26E-4	0.0669	1.17E-4	0.011	1.17E-4
$AC(Q_{CQB})$	0.0281	1.73E-4	0.009	4.49E-5	0.007	8.92E-6	0.026	1.78E-4	0.011	4.14E-5	0.007	4.68E-6
$AC(RF_{EB})$	**0.008**	1.82E-4	0.0036	3.22E-5	0.0036	1.08E-5	**0.0085**	1.80E-4	**0.0038**	3.18E-5	0.0029	1.06E-5
$AC(RF_{CQB})$	0.0092	1.83E-4	0.0037	4.5E-5	0.0027	1.16E-5	0.0091	1.83E-4	0.0039	4.45E-5	**0.0026**	9.97E-6
$CC(Q_{EB})$	0.3285	0.0206	0.006	0.0133	0.1004	0.0074	0.2763	0.0222	0.1584	0.0118	0.1108	0.0054
$CC(Q_{CQB})$	0.3398	0.0222	0.1139	0.0149	0.0606	0.0098	0.239	0.0258	0.15	0.0134	0.116	0.0063
$CC(RF_{EB})$	0.5328	0.0455	0.2197	0.0279	0.1386	0.0158	0.4362	0.0496	0.2619	0.0257	0.1852	0.010
$CC(RF_{CQB})$	0.5368	0.0445	0.2165	0.028	0.1255	0.0165	0.4217	0.0493	0.2589	0.0258	0.1878	0.0101
$AC(C4.5)$	0.0672	1.46E-4	0.0126	3.27E-5	0.0064	1.75E-5	0.0441	1.26E-4	0.0239	1.92E-5	0.0164	9.51E-7
$CC(C4.5)$	0.3403	0.0038	0.0571	0.0023	0.0323	0.0013	0.1859	0.004	0.1272	0.00138	0.1046	2.92E-4
$AC(RF_{C4.5})$	0.3883	0.0078	0.0807	0.006	0.0295	0.0045	0.2068	0.0105	0.1473	0.0043	0.1279	0.0017
$CC(RF_{C4.5})$	0.3883	0.0078	0.0807	0.006	0.0295	0.0045	0.2068	0.0105	0.1473	0.0043	0.1279	0.0017
$AC(SVM)$	0.2576	0.0046	**0.0024**	0.1018	**0.0019**	0.34	0.0307	0.012	0.0745	0.0916	0.1346	0.3213
$CC(SVM)$	0.6912	0.0055	0.1069	0.0873	0.0402	0.33	0.2657	0.0044	0.2474	0.0697	0.2846	0.2784

TABLE VI. BANK DATASET: MEAN μ AND VARIANCE σ^2 OF KLD OF PREDICTED AND ACTUAL QUANTIFICATION FOR ALL TESTED QUANTIFIERS W.R.T. HIGH, MODERATE AND LOW CLASS DISTRIBUTION UNBALANCE IN THE TRAINING SETS (LEFT) OR THE TEST SETS (RIGHT) OF FIXED SIZE. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Training fixed							Test fixed					
	H		M		L		H		M		L	
Method	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
$AC(Q_{EB})$	1.2414	0.9081	0.0023	0.7133	0.0512	0.7902	0.4438	1.1576	0.399	0.8	0.3947	0.5884
$AC(Q_{CQB})$	1.2419	0.908	8.41E-4	0.7323	0.0011	0.7845	0.4043	1.1717	0.3908	0.8035	0.3847	0.5912
$AC(RF_{EB})$	0.008	1.34E-5	9.17E-4	1.06E-5	5.16E-4	0.0576	0.0049	1.68E-5	0.0021	8.25E-6	0.0021	8.43E-6
$AC(RF_{CQB})$	**0.0078**	2.62E-5	**5.53E-4**	9.08E-6	**2.34E-4**	0.0588	**0.0048**	3.06E-5	**0.0019**	1.15E-5	**0.0016**	7.05E-6
$CC(Q_{EB})$	1.4482	0.9471	0.1109	0.7981	0.1173	0.5675	0.6481	1.0154	0.5117	0.6592	0.4506	0.4435
$CC(Q_{CQB})$	1.4482	0.7956	0.1272	0.7956	0.0463	0.5915	0.5969	1.0317	0.4983	0.6633	0.458	0.4475
$CC(RF_{EB})$	0.6447	9.9E-6	0.2256	0.0789	0.1082	0.0411	0.4373	0.0927	0.2921	0.0523	0.2289	0.0266
$CC(RF_{CQB})$	0.6613	1.36E-5	0.2321	0.0796	0.0939	0.0434	0.4298	0.0951	0.2958	0.0537	0.2378	0.0271
$AC(C4.5)$	0.055	0.0021	0.0176	3.21E-4	0.0126	0.0144	0.049	0.0022	0.0222	8.15E-4	0.0137	2.52E-4
$CC(C4.5)$	0.29	8.59E-4	0.0806	0.0211	0.0806	0.0075	0.1873	0.0249	0.1207	0.0127	0.0949	0.0061
$AC(RF_{C4.5})$	0.2862	0.0303	0.0698	0.022	0.0282	0.0151	0.1689	0.0361	0.1128	0.0197	0.0918	0.0108
$CC(RF_{C4.5})$	0.2862	0.0303	0.0698	0.022	0.0282	0.0151	0.1689	0.0361	0.1128	0.0197	0.0918	0.0108
$AC(SVM)$	2.3009	0.6356	0.5772	0.6071	0.0012	0.5965	0.9661	0.9199	0.8997	0.5486	0.8689	0.3376
$CC(SVM)$	2.5264	0.6394	0.7248	0.606	0.0574	0.5877	1.1509	0.8694	1.0299	0.515	0.9748	0.3056

TABLE VII. MAGIC: MEAN μ AND VARIANCE σ^2 OF KLD OF PREDICTED AND ACTUAL QUANTIFICATION FOR ALL TESTED QUANTIFIERS W.R.T. HIGH, MODERATE AND LOW CLASS DISTRIBUTION UNBALANCE IN THE TRAINING SETS (LEFT) OR THE TEST SETS (RIGHT) OF FIXED SIZE. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

the direct optimization of decision trees for quantification purposes.

A research field that is only apparently related to quantification is *collective classification* (CC) [21]. Similarly to quantification, here the classification of instances is not viewed in isolation. However, CC is radically different from quantification in that its focus is on improving the accuracy of classification by exploiting relationships between the objects to classify (e.g., hypertextual documents that link to each other). The accuracy of CC is evaluated at the individual level, rather than at the aggregate level as for quantification.

Quantification has also relations with *prevalence estimation from screening tests*, an important task in epidemiology ([15], [16], [19], [27]). A screening test is a test that a patient undergoes in order to check if s/he has a given pathology. Tests are often imperfect, i.e., they may give rise to false positives (the patient is incorrectly diagnosed with the pathology) and false negatives (the test wrongly diagnoses the patient to be free from the pathology). Therefore, testing a patient is akin to classifying a document, and using these tests for estimating the prevalence of the pathology in a given population is akin to performing quantification via classification. The main difference between this task and quantification is that a screening test typically has known and fairly constant recall (that epidemiologists call "sensitivity") and specificity (i.e., recall on the complement of the class), while the same usually does not happen for a classifier. Another related field in statistics is the "randomized response" methodology for conducting privacy-preserving tests, which uses a correction statistics similar to

adjusted count post-processing [25].

VII. Conclusion

In this paper we have proposed how to build decision trees geared towards the quantification task. Specifically, we have presented two measures for quantification accuracy, one focused on the optimization of quantification accuracy only, and the other aiming at a trade-off between classification and quantification accuracy. We have defined two methods for solving the quantification task via decision tree learning based on the two proposed measures. The thorough experimental evaluation that we have carried out shows that our methods outperform state-of the-art quantifiers optimized for classification accuracy, such as SVMs.

Solving the quantification problem by machine learning techniques opens up new avenues to the estimation of social indicators based on big data, provided that we can rely on relatively small surveys of labelled data. Clearly, an important aspect to be further investigated is the analysis of the confidence of our quantification results, that would make our methodology stronger. Future investigations will be directed also to explore other methods for optimizing quantification accuracy. For example, we plan to study the effect of directly optimizing the KLD function inside the learning process. Different strategies for selecting the best node for splitting should also be evaluated more in depth.

An additional, interesting open question is related to the combination of random forests with the wisdom-of-the-crowd theory. In our current solution every quantification tree represents the opinion of an expert because all trees are learnt on a training set with a fixed set of features. It will be interesting to study the effect of learning individual quantification trees with varying numbers of features representing the various levels of expertise of the participants.

Acknowledgments

This work has been supported by EU FET-Open project ICON (FP7-ICT-2011-C n. 284715).

References

[1] A. Bella, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Quantification via probability estimators. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2010)*, pages 737–742, Sydney, AU, 2010.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] W. Buntine and W. Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.

[4] A. Esuli and F. Sebastiani. Machines that learn how to code open-ended survey data. *International Journal of Market Research*, 52(6):775–800, 2010.

[5] A. Esuli and F. Sebastiani. Sentiment quantification. *IEEE Intelligent Systems*, 25(4):72–75, 2010.

[6] A. Esuli and F. Sebastiani. Optimizing text quantifiers for multivariate loss functions. Technical Report 2013-TR-005, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT, 2013.

[7] G. Forman. Counting positives accurately despite inaccurate classification. In *Proceedings of the 16th European Conference on Machine Learning (ECML 2005)*, pages 564–575, Porto, PT, 2005.

[8] G. Forman. Quantifying trends accurately despite classifier error and class imbalance. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 157–166, Philadelphia, US, 2006.

[9] G. Forman. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206, 2008.

[10] G. Forman, E. Kirshenbaum, and J. Suermondt. Pragmatic text mining: Minimizing human effort to quantify many issues in call logs. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 852–861, Philadelphia, US, 2006.

[11] B. Furletti, L. Gabrielli, C. Renso, and S. Rinzivillo. Identifying users profiles from mobile calls habits. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing (UrbComp 2012)*, pages 17–24, Beijing, CN, 2012. ACM.

[12] F. Galton. Vox Populi. *Nature*, 75:450–451, 1907.

[13] V. González-Castro, R. Alaiz-Rodríguez, L. Fernández-Robles, R. Guzmán-Martínez, and E. Alegre. Estimating class proportions in boar semen analysis using the Hellinger distance. In *Proceedings of the 23rd International Conference on Industrial Engineering and other Applications of Applied Intelligent Systems (IEA/AIE 2010)*, pages 284–293, Cordoba, ES, 2010.

[14] D. Hopkins and G. King. A method of automated. nonparametric content analysis for social science. *American Journal of Political Science*, 54(1):229—247, 2010.

[15] P. S. Levy and E. H. Kass. A three-population model for sequential screening for bacteriuria. *American Journal of Epidemiology*, 91(2):148—154, 1970.

[16] R. A. Lew and P. S. Levy. Estimation of prevalence on the basis of screening tests. *Statistics in Medicine*, 8(10):1225–1230, 1989.

[17] D. D. Lewis. Evaluating and optmizing autonomous text classification systems. In *Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1995)*, pages 246–254, Seattle, US, 1995.

[18] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[19] E. Rahme and L. Joseph. Estimating the prevalence of a rare disease: Adjusted maximum likelihood. *The Statistician*, 47:149–158, 1998.

[20] L. Sánchez, V. González, E. Alegre, and R. Alaiz. Classification and quantification based on image analysis for sperm samples with uncertain damaged/intact cell proportions. In *Proceedings of the 5th International Conference on Image Analysis and Recognition (ICIAR 2008)*, pages 827–836, Póvoa de Varzim, PT, 2008.

[21] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[22] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall, London, UK, 1986.

[23] J. Surowiecki. *The Wisdom of Crowds*. Anchor, 2005.

[24] L. Tang, H. Gao, and H. Liu. Network quantification despite biased labels. In *Proceedings of the 8th Workshop on Mining and Learning with Graphs (MLG 2010)*, pages 147–154, Washington, US, 2010.

[25] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[26] J. C. Xue and G. M. Weiss. Quantification and semi-supervised classification methods for handling changes in class distribution. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2009)*, pages 897–906, Paris, FR, 2009.

[27] X.-H. Zhou, D. K. McClish, and N. A. Obuchowski. *Statistical Methods in Diagnostic Medicine*. Wiley, New York, US, 2002.