# A Critical Reassessment of the Saerens-Latinne-Decaestecker Algorithm for Posterior Probability Adjustment

ANDREA ESULI, Consiglio Nazionale delle Ricerche, Italy
ALESSIO MOLINARI, Consiglio Nazionale delle Ricerche, Italy and Università di Pisa, Italy
FABRIZIO SEBASTIANI, Consiglio Nazionale delle Ricerche, Italy

We critically re-examine the Saerens-Latinne-Decaestecker (SLD) algorithm, a well-known method for estimating class prior probabilities ("priors") and adjusting posterior probabilities ("posteriors") in scenarios characterized by distribution shift, i.e., difference in the distribution of the priors between the training and the unlabelled documents. Given a machine-learned classifier and a set of unlabelled documents for which the classifier has returned posterior probabilities and estimates of the prior probabilities, SLD updates them both in an iterative, mutually recursive way, with the goal of making both more accurate; this is of key importance in downstream tasks such as single-label multiclass classification and cost-sensitive text classification. Since its publication, SLD has become the standard algorithm for improving the quality of the posteriors in the presence of distribution shift, and is still considered a top contender when we need to estimate the priors (a task that has become known as "quantification"). However, its real effectiveness in improving the quality of the posteriors has been questioned. We here present the results of systematic experiments conducted on a large, publicly available dataset, across multiple amounts of distribution shift and multiple learners. Our experiments show that SLD improves the quality of the posterior probabilities and of the estimates of the prior probabilities, but only when the number of classes in the classification scheme is very small and the classifier is calibrated. As the number of classes grows, or as we use non-calibrated classifiers, SLD converges more slowly (and often does not converge at all), performance degrades rapidly, and the impact of SLD on the quality of the prior estimates and of the posteriors becomes negative rather than positive.

Additional Key Words and Phrases: Text Classification, Probabilistic Classifiers, Posterior Probabilities, Prior Probabilities, Distribution Shift, Dataset Shift

**ACM Reference Format:**
Andrea Esuli, Alessio Molinari, and Fabrizio Sebastiani. 2020. A Critical Reassessment of the Saerens-Latinne-Decaestecker Algorithm for Posterior Probability Adjustment. *ACM Transactions on Information Systems* 1, 1 (November 2020), 33 pages. https://doi.org/0000001.0000001

The order in which the authors are listed is purely alphabetical; each author has given an equally important contribution to this work.

Authors' addresses: Andrea Esuli, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124, Pisa, Italy, andrea.esuli@isti.cnr.it; Alessio Molinari, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124, Pisa, Italy, alessio.molinari@isti.cnr.it, Dipartimento di Informatica, Università di Pisa, 56127, Pisa, Italy; Fabrizio Sebastiani, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124, Pisa, Italy, fabrizio.sebastiani@isti.cnr.it.


1046-8188/2020/11-ART $15.00
https://doi.org/0000001.0000001

## 1 INTRODUCTION

Single-label text classification is the task of training a text classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ that labels each document $\mathbf{x}_i \in \mathcal{X}$ with a class $h(\mathbf{x}_i) \in \mathcal{Y}$; $\mathcal{X}$ is a (possibly infinite) set of documents (the *domain*), while $\mathcal{Y} = \{y_1, ..., y_{|\mathcal{Y}|}\}$ is a finite set of classes (the *codeframe*, or *classification scheme*).

The classifiers trained by means of modern machine learning methods usually return, together with the class assigned to the document, a vector $(s(\mathbf{x}_i, y_1), ..., s(\mathbf{x}_i, y_{|\mathcal{Y}|}))$ of *confidence scores*, where $s(\mathbf{x}_i, y_j)$ by and large represents the confidence (or the strength of belief) that the classifier has in the fact that $\mathbf{x}_i$ belongs to $y_j$; the class $h(\mathbf{x}_i)$ assigned to document $\mathbf{x}_i$ is thus the one with the highest confidence score, i.e.,

$$h(\mathbf{x}_i) = \arg\max_{y_j \in \mathcal{Y}} s(\mathbf{x}_i, y_j) \tag{1}$$

Classifiers that return confidence scores are sometimes called *scoring classifiers* [12]. Without loss of generality[1] we may assume that these confidence scores are actual probabilities (if so, these are called *posterior probabilities*, or simply *posteriors*), i.e., we may assume that the vector being returned has the form $(\Pr(y_1|\mathbf{x}_i), ..., \Pr(y_{|\mathcal{Y}|}|\mathbf{x}_i))$, where $\sum_{j=1}^{|\mathcal{Y}|} \Pr(y_j|\mathbf{x}_i) = 1$ and $\Pr(y_j|\mathbf{x}_i)$ represents the probability that the classifier "subjectively" attributes to the fact that $x_i$ belongs to class $y_j$. Rather than simple classifiers, these models are full-blown *probability estimators*.

The posteriors play an important role in several tasks, a role that goes beyond allowing to take a classification decision by means of Equation 1. One of these tasks is *document ranking*, as when the documents are ranked in decreasing order of the probability $\Pr(y_j|\mathbf{x}_i)$ that they belong to a certain class $y_j$; ranking is useful, for instance, when performing active learning by means of relevance sampling [21], or when one needs to choose the best $k$ documents for a certain class, or when one needs to choose the best $k$ classes for a certain document. Another such task is *cost-sensitive classification*, where classification is performed in such a way that

$$h(\mathbf{x}_i) = \arg\min_{y_j \in \mathcal{Y}} \sum_{y_l \in \mathcal{Y}} \lambda_{jl} \cdot \Pr(y_l|\mathbf{x}_i) \tag{2}$$

where $\lambda_{jl}$ represents the "cost" of classifying a document in class $y_j$ when it should have been classified in class $y_l$ (this cost is equal to 0 when $j = l$ and higher than 0 when $j \neq l$); in other words, $\mathbf{x}_i$ is assigned to the class such that the expected cost (i.e., the risk) of assigning $\mathbf{x}_i$ to it is minimum. Example applications of cost-sensitive text classification may be found, for instance, in spam filtering [5], or in technology-assisted review [30].

Of course, in order to guarantee that single-label multiclass classification, ranking, cost-sensitive classification, and other such tasks, are executed with high accuracy, the posteriors must be accurate too. An intuition of what "accurate posteriors" means can be provided by the following example. For instance, if 10% (resp., 90%) of all the documents $\mathbf{x}_i$ for which $\Pr(y_j|\mathbf{x}_i) = 0.5$ indeed belong to $y_j$, we can say that the classifier has overestimated (resp., underestimated) the probability that these documents belong to $y_j$, and that their posteriors are thus inaccurate. Indeed, we say (see for instance [15]) that the posteriors $\Pr(y_j|\mathbf{x}_i)$, where $\mathbf{x}_i$ belongs to a set $S = \{\mathbf{x}_1, ..., \mathbf{x}_{|S|}\}$, are (perfectly) *calibrated* (i.e., accurate) when, for all $a \in [0, 1]$, it holds that[2]

$$\frac{|\{\mathbf{x}_i \in S \cap y_j | \Pr(y_j|\mathbf{x}_i) = a\}|}{|\{\mathbf{x}_i \in S | \Pr(y_j|\mathbf{x}_i) = a\}|} = a \tag{3}$$

[1] See the discussion on probability calibration mechanisms later in this section.

[2] Perfect calibration is usually unattainable on any non-trivial dataset; however, calibration comes in degrees (and the quality of calibration can indeed be measured – see Section 3.1.2), so efforts can be made to obtain posteriors which are as close as possible to their perfectly calibrated counterparts.

The classifiers trained by means of some learners (such as logistic regression) are known to return reasonably well calibrated probabilities. Those trained by means of some other learners (such as Naïve Bayes) return probabilities which are known to be not well calibrated [9]. Yet other learners (such as SVMs or AdaBoost) train classifiers that return confidence scores that are not probabilities (i.e., that do not range on [0,1] and/or that do not sum up to 1). In order to address these two latter cases, *probability calibration* mechanisms exist (see e.g., [28, 29, 31, 40, 41]) that convert the outputs of these classifiers into well calibrated probabilities.

However, even when using text classifiers that tend to return well calibrated probabilities, or even when using the probability calibration methods mentioned above, the accuracy of the posteriors tends to be low if the problem setting exhibits *dataset shift* (see e.g., [33]), i.e., if the joint distribution $p_L(\mathbf{x}, y)$ in the training set is different from the joint distribution $p_U(\mathbf{x}, y)$ in the unlabelled set. One particular type of dataset shift that interests us is *distribution shift* [2], which occurs when the distribution of the *prior probabilities* $\Pr(y_j)$ (or simply *priors*) in the training set (noted as $p_L(y)$) and that of the prior probabilities in the unlabelled set (noted as $p_U(y)$) differ. To see why this is the case, take the probabilistic classifier

$$\Pr(y_j|\mathbf{x}_i) = \frac{\Pr(\mathbf{x}_i|y_j)\Pr(y_j)}{\Pr(\mathbf{x}_i)} \tag{4}$$

and note that the posterior $\Pr(y_j|\mathbf{x}_i)$ on the left-hand side directly depends on the prior $\Pr(y_j)$ on the right-hand side. Since the prior $\Pr(y_j)$ has been estimated on the training set (i.e., its value has been set to $\Pr_L(y_j)$, which is distributed as $p_L(y)$), if $\Pr_L(y_j)$ is higher (resp., lower) than $\Pr_U(y_j)$ (which is distributed as $p_U(y)$), then the posteriors $\Pr(y_j|\mathbf{x}_i)$ of the documents in $U$ will be overestimated (resp., underestimated).[3]

Ideally, in order to have well calibrated posteriors even in the presence of distribution shift, we would need to set $\Pr(y_j)$ in Equation 4 to $\Pr_U(y_j)$, and not to $\Pr_L(y_j)$. But this is impossible, since $\Pr_U(y_j)$ is unknown at training time. The only known way out of this conundrum is provided by the Saerens-Latinne-Decaestecker algorithm (that we here call SLD for brevity)[4], an algorithm that iteratively re-estimates the priors $\Pr_U(y_j)$ of the unlabelled set and adjusts the posteriors $\Pr(y_j|\mathbf{x}_i)$, in a mutually recursive way [34]. This algorithm is essentially unique in its kind, and, to the best of our knowledge, no other algorithm that attempts to adjust the posteriors in the presence of distribution shift has been proposed since its publication. (An exception is the algorithm described in [38]; in Section 6 we discuss why we do not consider it as a contender.) As a result, SLD has become a standard, and is frequently used in scenarios characterized by distribution shift, either when the goal is improving the accuracy of the posteriors, or when the goal is obtaining estimates of the priors more accurate than can be obtained by the trivial "classify and count" method (the latter task is known as *supervised prevalence estimation*, or *quantification* [19]).

However, in recent experiments aimed at improving the quality of cost-sensitive text classification in technology-assisted review [22, 23], SLD has not delivered any measurable improvement in the quality of the posteriors. Since these experiments were limited in scope, we have then decided to engage in a large-scale experimentation of SLD, with the goal of reassessing its true ability at (i) accurately re-estimating the priors $\Pr_U(y_j)$ of the unlabelled set, and (ii) improving the quality of the posteriors $\Pr(y_j|\mathbf{x}_i)$ of the unlabelled documents. Note that goal (ii) is more important than goal (i), since the ability of SLD at estimating the priors has been systematically tested in previous

[3] In other words, that distribution shift brings about a low quality of the posteriors is due to the fact that distribution shift, as all types of dataset shift, invalidates the iid assumption (according to which the training examples and the unlabelled examples are drawn from the same distribution), on which probability calibration methods rely.

[4] In a number of other publications [17, 22, 23] the same algorithm was called EMQ, standing for "Expectation Maximization for Quantification"; in yet other publications [3] it is called RS, standing for "rescaling algorithm".

works (e.g., [10]), and since (as mentioned before) SLD is essentially the only known algorithm for improving the quality of already calibrated posteriors, while there are many alternatives to it (see the extensive review by González et al. [19]) when it comes to estimating the priors. We thus present systematic experiments involving different learners, different datasets, and different amounts of distribution shift, in which we try to assess the real benefits of using SLD.

The rest of the paper is structured as follows. Section 2 introduces and discusses the SLD algorithm in detail. In Section 3 we present the systematic experimentation to which we have subjected SLD, and in Section 4 we present its results, while in Section 5 we discuss exactly which kinds of distribution shift we target in our experiments. Section 6 discusses some related work, while Section 7 concludes.

## 2 THE SLD ALGORITHM

We assume a training set $L$ of labelled examples and a set $U = \{(\mathbf{x}_1, t(\mathbf{x}_1)), \ldots, (\mathbf{x}_{|U|}, t(\mathbf{x}_{|U|}))\}$ of unlabelled examples, i.e., examples whose true labels $t(\mathbf{x}_i) \in \mathcal{Y} = \{y_1, \ldots, y_{|\mathcal{Y}|}\}$ are unknown to the system.

SLD, proposed by Saerens et al. [34], is an instance of Expectation Maximization [8], a well-known iterative algorithm for finding maximum-likelihood estimates of parameters (in our case: the class prior probabilities) for models that depend on unobserved variables (in our case: the class labels). Pseudocode of the SLD algorithm is here included as Algorithm 1.

Essentially, SLD iteratively updates (Line 11) the class priors by using the posterior probabilities computed in the previous iteration, and updates (Line 13) the posterior probabilities by using the class priors computed in the present iteration, in a mutually recursive fashion. The main goal is to adjust the posteriors and re-estimate the priors in such a way that they are consistent with each other, where this "mutual consistency" means that they should be such that

$$\Pr_U(y_j) = \frac{1}{|U|} \sum_{\mathbf{x}_i \in U} \Pr(y_j|\mathbf{x}_i) \tag{5}$$

In Appendix A we show that Equation 5 is a necessary (albeit not sufficient) condition for the posteriors $\Pr(y_j|\mathbf{x}_i)$ of the documents $\mathbf{x}_i \in U$ to be calibrated. SLD may thus be viewed as making a step towards calibrating these posteriors.

The algorithm iterates until convergence, i.e., until the class priors become stable and Equation 5 is satisfied. The convergence of SLD may be tested by computing how the distribution of the priors at iteration $(s-1)$ and that at iteration $s$ still diverge; this can be evaluated, for instance, in terms of absolute error, i.e.,[5]

$$\mathrm{AE}(\hat{p}_U^{(s-1)}, \hat{p}_U^{(s)}) = \frac{1}{|\mathcal{Y}|} \sum_{j=1}^{|\mathcal{Y}|} |\hat{\Pr}_U^{(s)}(y_j) - \hat{\Pr}_U^{(s-1)}(y_j)| \tag{6}$$

In the experiments of Section 3, we decree that convergence has been reached when $\mathrm{AE}(\hat{p}_U^{(s-1)}, \hat{p}_U^{(s)}) < 10^{-6}$; we stop SLD when we have reached either convergence or the maximum number of iterations (that we set to 1000).

At each iteration of the algorithm, all the posteriors relative to class $y_j$ are multiplied by the same amount $\hat{\Pr}_U^{(s)}(y_j)/\hat{\Pr}_U^{(0)}(y_j)$. As a consequence, the net effect of SLD is to multiply all these posteriors by the same amount $\hat{\Pr}_U(y_j)/\hat{\Pr}_U^{(0)}(y_j)$ so that the resulting posteriors $\Pr(y_j|\mathbf{x}_i)$ are consistent with the resulting class prior $\hat{\Pr}_U(y_j)$, i.e., so that Equation 5 is satisfied; in other words, SLD is an iterative *rescaling* algorithm. The posteriors for different classes, though, do not get multiplied

[5]Consistently with most mathematical literature, we use the caret symbol (ˆ) to indicate estimation.

**ALGORITHM 1:** The SLD algorithm [34].

**Input** : Class priors $\Pr_L(y_j)$ on $L$, for all $y_j \in \mathcal{Y}$;
Posterior probabilities $\Pr(y_j|\mathbf{x}_i)$, for all $y_j \in \mathcal{Y}$ and for all $\mathbf{x}_i \in U$;
**Output** : Estimates $\hat{\Pr}_U(y_j)$ of class prevalence values on $U$, for all $y_j \in \mathcal{Y}$;
Updated posterior probabilities $\Pr(y_j|\mathbf{x}_i)$, for all $y_j \in \mathcal{Y}$ and for all $\mathbf{x}_i \in U$;

// Initialization
$s \leftarrow 0$;
**for** $y_j \in \mathcal{Y}$ **do**
  $\hat{\Pr}_U^{(s)}(y_j) \leftarrow \Pr_L(y_j)$; // Initialize the prior estimates
  **for** $\mathbf{x}_i \in U$ **do**
    $\Pr^{(s)}(y_j|\mathbf{x}_i) \leftarrow \Pr(y_j|\mathbf{x}_i)$; // Initialize the posteriors
  **end**
**end**
// Main Iteration Cycle
**while** *stopping condition = false* **do**
  $s \leftarrow s + 1$;
  **for** $y_j \in \mathcal{Y}$ **do**
    $\hat{\Pr}_U^{(s)}(y_j) \leftarrow \frac{1}{|U|} \sum_{\mathbf{x}_i \in U} \Pr^{(s-1)}(y_j|\mathbf{x}_i)$; // Update the prior estimates
    **for** $\mathbf{x}_i \in U$ **do**
      $\Pr^{(s)}(y_j|\mathbf{x}_i) \leftarrow \frac{\frac{\hat{\Pr}_U^{(s)}(y_j)}{\hat{\Pr}_U^{(0)}(y_j)} \cdot \Pr^{(0)}(y_j|\mathbf{x}_i)}{\sum_{y_j \in \mathcal{Y}} \frac{\hat{\Pr}_U^{(s)}(y_j)}{\hat{\Pr}_U^{(0)}(y_j)} \cdot \Pr^{(0)}(y_j|\mathbf{x}_i)}$ // Update the posteriors
    **end**
  **end**
**end**
// Generate output
**for** $y_j \in \mathcal{Y}$ **do**
  $\hat{\Pr}_U(y_j) \leftarrow \hat{\Pr}_U^{(s)}(y_j)$ ; // Return the prior estimates
  **for** $\mathbf{x}_i \in U$ **do**
    $\Pr(y_j|\mathbf{x}_i) \leftarrow \Pr^{(s)}(y_j|\mathbf{x}_i)$ // Return the adjusted posteriors
  **end**
**end**

by the same amount; this is somehow obvious, since at the end of the process the posteriors for document $\mathbf{x}_i$ must all sum up to 1, which means that if the posteriors for a class $y'$ all end up increasing, there must be at least a class $y''$ whose posteriors all end up decreasing.

SLD, as proposed by Saerens et al. [34] and as described here, addresses single-label classification, i.e., the task in which exactly 1 out of $|\mathcal{Y}|$ classes must be assigned to each document. This means that SLD can be used for *binary* classification (which is single-label classification with $|\mathcal{Y}| = 2$), for *single-label multiclass classification* (which is single-label classification with $|\mathcal{Y}| > 2$), and for *multi-label* classification (which is the task in which any number of classes in $\mathcal{Y}$ can be assigned

to a document), since multi-label classification can be trivially recast into $|\mathcal{Y}|$ independent binary classification tasks.

It is worth pointing out something which Saerens et al. [34] did not observe, i.e., that the combination of (i) a learner that trains classifiers to return posterior probabilities, and (ii) the SLD algorithm that improves the quality of the posterior probabilities for a given set of unlabelled documents $U$, might be called a *transductive* algorithm [39], since it uses training documents to infer posterior probabilities only for a specific, finite set of unlabelled documents known at training time. This is different from standard *inductive* algorithms, that use training documents to infer a general-purpose hypothesis that can later be applied to the entire domain. One aspect of this transductive nature is that SLD must operate "holistically", i.e., on entire *sets* of unlabelled documents, and cannot, for instance, update the posteriors of individual unlabelled documents in isolation of each other; another aspect is that, as Saerens et al. [34, p. 35] put it, "the model has to be completely refitted each time it is applied to a new data set".

Interestingly enough, SLD was originally designed with the goal of improving the posteriors, so as to improve the accuracy of classification (by means of Equation 1) in the presence of distribution shift. The fact that it also allows estimating the priors in a more accurate way than by just "classifying and counting" was considered a by-product by its authors. However, in the years that followed, thanks to increased interest in the "quantification" task (see Section 1), SLD became a popular baseline for algorithms whose goal was the estimation of the priors, and in recent extensive experimentation it has been found to be a top-notch performer for this task [25].

In [34], the quality of the posteriors generated by means of SLD was measured in terms of error rate, i.e., the fraction of classification decisions that are wrong. However, a major difference between error rate and the measures we will instead use for the same purpose (see Section 3.1.2) is that the former, unlike the latter, evaluates not the posterior probabilities *per se* but the classification decisions that are based on them. Error rate is thus only an "indirect" measure of the quality of the posteriors, and a coarse one too. To see this, let us assume we are dealing with binary classification, and let us consider a document $\mathbf{x}_i$ such that its true class is $y_1$. According to Equation 1, posteriors $\Pr(y_1|\mathbf{x}_i) = .51$ and $\Pr(y_2|\mathbf{x}_i) = .49$ would lead to $\mathbf{x}_i$ being correctly classified into $y_1$, and so would posteriors $\Pr(y_1|\mathbf{x}_i) = .99$ and $\Pr(y_2|\mathbf{x}_i) = .01$. The former set of posteriors is equivalent to the latter set as far as error rate is concerned; however, we intuitively consider the latter set "better" than the former set, and the measures we discuss in Section 3.1.2 indeed consider it as such. Note also that classification (as implemented by means of Equation 1), is just a downstream application of the posteriors, and there are many such potential applications, such as (as already recalled in the introduction) ranking and cost-sensitive classification; rather than evaluating the posteriors by evaluating one of their potential applications, it seems more sensible to evaluate them directly, which can be done by means of the measures of Section 3.1.2.

In [34], SLD was subjected to a small-scale experimentation, which involved the binary case only. The experiments we conduct in this paper are instead carried out on a very large scale, and involve both binary and multiclass classification.

## 3 EXPERIMENTS

In this section we report systematic experiments in which, using a variety of datasets, learners, and amounts of distribution shift, we compare the quality of the priors and (above all) of the posteriors *before* the application of SLD, with that *after* the application of SLD. This allows us to see when and in what conditions the application of SLD is beneficial.

### 3.1 Evaluation Measures

We evaluate SLD in terms of two main criteria, i.e., (i) the ability to improve the accuracy of the estimated class priors with respect to the trivial "Classify & Count" estimator, and (ii) the ability to improve the accuracy of the posterior probabilities with respect to the ones originally returned by the classifier.

*3.1.1 Evaluating the Priors.* For evaluating the quality of the estimated class priors we use *normalized absolute error* (NAE) (see e.g., [35, §4.2]), defined as

$$\text{NAE}(p_U, \hat{p}_U) = \frac{\sum_{j=1}^{|\mathcal{Y}|} |\Pr_U(y_j) - \hat{\Pr}_U(y_j)|}{2(1 - \min_{y_j \in \mathcal{Y}} \Pr(y_j))} \tag{7}$$

where $p_U$ and $\hat{p}_U$ indicate the true class distribution and the predicted class distribution, resp., on the set $U$ of unlabelled documents. The reason we use NAE is that, besides its simplicity, it is also (as argued in [35]) one of the theoretically most satisfying measures for evaluating the quality of class priors; NAE ranges between 0 (best) and 1 (worst). In all the tables of results that we include in Section 4, we compare the estimates of the class priors before applying SLD, computed by "classifying and counting", i.e., as

$$\hat{\Pr}_U(y_j) = \frac{1}{|U|}|\{\mathbf{x}_i \in U, h(\mathbf{x}_i) = y_j\}|$$

with the same estimates after applying SLD (which are the values of $\hat{\Pr}_U(y_j)$ resulting from Line 18 of Algorithm 1).

*3.1.2 Evaluating the Posteriors.* For evaluating the quality of the posterior probabilities, the measure we use is the *Brier score* [4]. Given a set $U = \{(\mathbf{x}_1, t(\mathbf{x}_1)), \ldots, (\mathbf{x}_{|U|}, t(\mathbf{x}_{|U|}))\}$ of unlabelled documents to be labelled according to codeframe $\mathcal{Y}$, the Brier score is defined as

$$\text{BS} = \frac{1}{|\mathcal{Y}| \cdot |U|} \sum_{j=1}^{|\mathcal{Y}|} \sum_{i=1}^{|U|} (I(t(\mathbf{x}_i) = y_j) - \Pr(y_j|\mathbf{x}_i))^2 \tag{8}$$

where $I(\cdot)$ is a function that returns 1 if its argument is true and 0 otherwise. The Brier score ranges between 0 (best) and 1 (worst), i.e., it is a measure of error, and not of accuracy. It rewards classifiers that return a high posterior for the true class of $\mathbf{x}_i$ and low posteriors for all classes other than the true class of $\mathbf{x}_i$. The Brier score is an example of so-called *strictly proper scoring rules* [18], defined as loss functions which are minimized only when $\Pr(y_j|\mathbf{x}_i)$ equals 1 for $y_j = t(\mathbf{x}_i)$.

It is useful to analyze the Brier score in a more fine-grained way. For class $y_j$, let the [0,1] interval be partitioned into an ordered sequence of $b$ intervals $I_{1j}, ..., I_{bj}$, and let us define bins $B_{1j}, ..., B_{bj}$ such that $\mathbf{x}_i \in B_{kj}$ iff $\Pr(y_j|\mathbf{x}_i) \in I_{kj}$. DeGroot and Fienberg [7, §4] show[6] that the Brier score can be written as

$$\text{BS} = \text{CE} + \text{RE} \tag{9}$$

[6]The formulation of the Brier score originally given in [7, §4] is slightly different since the authors assume that a posterior may only take up one of a small, fixed number of values, which makes intervals and bins not necessary for the formulation of BS. While this assumption is reasonable when posteriors are returned by human beings, this is not when they are returned by automatic probabilistic classifiers; as a result, we here reformulate BS by using intervals and bins.

with

$$\mathrm{CE} = \frac{1}{|\mathcal{Y}| \cdot b} \sum_{j=1}^{|\mathcal{Y}|} \sum_{k=1}^{b} \nu(B_{kj}, U) \cdot (\pi(B_{kj}) - \rho(y_j, B_{kj}))^2 \tag{10}$$

$$\mathrm{RE} = \frac{1}{|\mathcal{Y}| \cdot b} \sum_{j=1}^{|\mathcal{Y}|} \sum_{k=1}^{b} \nu(B_{kj}, U) \cdot \rho(y_j, B_{kj}) \cdot (1 - \rho(y_j, B_{kj})) \tag{11}$$

where

- $\nu(B_{kj}, U)$ is the prevalence of $B_{kj}$ in $U$, i.e., the fraction $\frac{|B_{kj}|}{|U|}$ of documents $\mathbf{x}_i$ in $U$ that are in $B_{kj}$;
- $\pi(B_{kj})$ is the expected value $\frac{1}{|B_{kj}|} \sum_{\mathbf{x}_i \in B_{kj}} \Pr(y_j|\mathbf{x}_i)$ of the posteriors for the documents in $B_{kj}$;
- $\rho(y_j, B_{kj})$ is the prevalence of $y_j$ in $B_{kj}$, i.e., the fraction $\frac{1}{|B_{kj}|} \sum_{\mathbf{x}_i \in B_{kj}} I(t(\mathbf{x}_i) = y_j)$ of documents $\mathbf{x}_i$ in $B_{kj}$ that belong to class $y_j$.

Here, CE is a measure of the *calibration error* of the posterior probabilities; in fact, it is easy to see that its value is 0 if and only if Equation 3 is verified for each $S \in \{B_{1j}, ..., B_{bj}\}$. RE is instead a measure of what DeGroot and Fienberg [7] call the *refinement error* of the classifier, i.e., of the lack of confidence of its predictions; its value is 0 if and only if all the posteriors it returns have a value of 0 or 1, while its value is 1 if and only it the classifier always "sits on the fence", i.e., if all the posteriors it returns have a value equal to the prevalence of $y_j$ in $U$.[7]

As an example, in a binary setting consider a perfectly balanced unlabelled set $U$, consider a ("perfect") classifier $h'$ that returns $\Pr(y_j|\mathbf{x}_i) = 1$ for all $\mathbf{x}_i$ whose true class is $y_j$ and $\Pr(y_j|\mathbf{x}_i) = 0$ for all $\mathbf{x}_i$ whose true class is not $y_j$, and consider a classifier $h''$ that returns $\Pr(y_j|\mathbf{x}_i) = .50$ for all $\mathbf{x}_i \in U$. Classifiers $h'$ and $h''$ are equivalent as far as CE is concerned (they both get a score of 0), but they are not for RE, which is equal to 0 for $h'$ and to .50 for $h''$. Conversely, consider the same set $U$ and the same ("perfect") classifier $h'$ of the previous example, and consider a ("perverse") classifier $h'''$ that returns $\Pr(y_j|\mathbf{x}_i) = 0$ for all $\mathbf{x}_i$ whose true class is $y_j$ and $\Pr(y_j|\mathbf{x}_i) = 1$ for all $\mathbf{x}_i$ whose true class is not $y_j$. Classifiers $h'$ and $h'''$ are equivalent as far as RE is concerned (they both get a score of 0), but they are not for CE, which is equal to 0 for $h'$ and to 1 for $h''$. Prediction power (which, in this case, manifests itself in the form of good-quality posteriors) thus requires calibration *and* refinement. Another way of saying this is that BS measures the classifier's *knowledge*, which is a combination of the classifier's *introspection*, or *self-awareness* (which is measured by CE), and of the classifier's *confidence* (which is measured by RE).

In this paper we define and use two variants of the Brier score, i.e.,

- the *Isometric Brier Score* (here shortened as $\mathrm{BS}_L$, where $L$ stands for "length"), which is obtained by partitioning $U$ into intervals $I_{1j}, ..., I_{bj}$ of equal length; for instance, if $b = 10$ then $I_{1j} = [.0, .1)$, $I_{2j} = [.1, .2)$,..., $I_{bj} = [.9, 1.0]$;
- the *Isomerous Brier Score* (here shortened as $\mathrm{BS}_N$, where $N$ stands for "number"), which is obtained by partitioning $U$ into intervals $I_{1j}, ..., I_{bj}$ such that the corresponding bins $B_{1j}, ..., B_{bj}$ have equal size, i.e., are such that (a) $\mathbf{x}' \in B_{sj}$ and $\mathbf{x}'' \in B_{tj}$ with $s < t$ implies that $\Pr(y_j|\mathbf{x}') \leq \Pr(y_j|\mathbf{x}'')$, and (b) $|B_{sj}| = |B_{tj}|$ for any $s, t \in \{1, ..., b\}$. Note that, when partitioning $U$ this way, $\nu(B_{kj}, U)$ is the same for all $1 \leq k \leq b$.

The advantage of $\mathrm{BS}_N$ over $\mathrm{BS}_L$ is that all bins are guaranteed to have a high enough number of elements, which reduces the risk that the difference between $\rho(y_j, B_{kj})$ and $\pi(B_{kj})$ is extreme due

[7] The decomposition of BS into CE and RE was originally introduced by Murphy [26], who actually used the terms *reliability* and *resolution* to denote CE and RE, respectively; the terminology we use in this paper is the one now current.

to sparsity. In this paper we use the $\text{BS}_L$ variant for compatibility with previous literature (which mostly uses the $\text{BS}_L$ variant – e.g., [3, 37]), and the $\text{BS}_N$ variant because, as argued, it seems to have superior formal properties.

In the experiments reported in this paper we use $b = 10$.

### 3.2 Dataset

As the dataset, in our experiments we use RCV1-v2, a dataset comprising 804,414 news stories published by Reuters from Aug 20, 1996, to Aug 19, 1997.[8] We here consider the "Topic" hierarchy, consisting of 101 classes. For text classification purposes, RCV1-v2 is traditionally split into a training set consisting of the (chronologically) first 23,149 documents (the ones written in Aug 1996), and a test set consisting of the last 781,265 documents (the ones written from Sep 1996 onwards).

RCV1-v2 is multi-label, i.e., a document may belong to several classes at the same time; since in this paper we are interested in single-label classification, we select its "single-label fragment", i.e., the subset of RCV1-v2 documents that have exactly 1 label. In order to do so, (a) we remove all "derived" labels, leaving only "primitive" labels[9], and (b) we remove from the collection all documents that do not have exactly one "primitive" label.

For reasons that will be clear in Section 3.2.1, in our experiments we consider only the 37 classes with at least 2000 (training or test) positive examples; of these, 31 are "leaf" classes while the remaining 6 classes correspond to internal nodes of the hierarchy.[10] We also remove all documents that do not belong to any of these 37 classes, which leaves us with 517,978 documents.

*3.2.1 Generating Samples with Controlled Amounts of Distribution Shift.* RCV1-v2 exhibits very little distribution shift between training set and test set. In fact, if we compute the normalized absolute error between $p_L$ (the class distribution in the training set) and $p_U$ (the class distribution in the unlabelled documents), i.e.,

$$\text{NAE}(p_L, p_U) = \frac{\sum_{j=1}^{|\mathcal{Y}|} |\Pr_L(y_j) - \Pr_U(y_j)|}{2(1 - \min_{y_j \in \mathcal{Y}} \Pr_L(y_j))} \tag{12}$$

for RCV1-v2 we obtain NAE = .0026, which is an extremely low value (since NAE always ranges between 0 – indicating no shift – and 1 – indicating maximum shift).

We instead want to test the SLD algorithm on a variety of distribution shift values, thus simulating a variety of possible application scenarios.[11] In order to do so, by using the protocol described below we extract from RCV1-v2 $k$ different samples, each consisting of a training set and a test set sampled from different class distributions; all the results of our experiments will thus be average values across these $k$ samples.

We run binary, "one-against-the-rest" classification experiments, i.e., experiments in which, for each class $y_j$, all the examples not belonging to $y_j$ are considered negative examples of $y_j$. For these experiments:

[8] Available from http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

[9] The RCV1-v2 codeframe has a hierarchical structure. As a result, when a document is labelled with class $y_j$, it is also labelled with all classes that are ancestors of $y_j$ in the RCV1-v2 tree. Whenever a document has two labels $y'$ and $y''$ such that $y'$ is an ancestor of $y''$, we remove this "derived" label $y'$ from its labels; we are thus left with "primitive" labels (i.e., labels $y_j$ such that the document has no label which is a descendant of $y_j$).

[10] Each of these latter 6 classes has at least 2000 positive examples "of its own", i.e., such that none of its descendant classes has any of these examples.

[11] Testing SLD against different values of distribution shift is of key importance also because one of the claims made by Saerens et al. [34, p. 31] is that their algorithm improves the prior estimates when distribution shift is substantial while it may actually worsen them in a "zero shift" setting.

(1) We generate two random vectors $\Pi_L = (\pi_1^L, \pi_2^L)$ and $\Pi_U = (\pi_1^U, \pi_2^U)$ of class priors, i.e., two vectors such that $0 \leq \pi_j^L, \pi_j^U \leq 1$ for each $1 \leq j \leq 2$ and such that $\sum_{j=1}^{2} \pi_j^L = \sum_{j=1}^{2} \pi_j^U = 1$;[12]
(2) We generate a training set $\sigma_L$ by drawing $m_L = |\sigma_L|$ different documents (with $m_L$ a parameter to be fixed beforehand), where at each draw we pick with probability $\pi_j^L$ a random document among those belonging to class $y_j$, and with probability $(1 - \pi_j^L)$ a random document among those not belonging to $y_j$. We then generate a test set $\sigma_U$ by first removing from the pool the documents drawn for $\sigma_L$, and then by drawing $m_U = |\sigma_U|$ different documents (with $m_U$ a parameter to be fixed beforehand), where at each draw we pick with probability $\pi_j^U$ a random document among those belonging to class $y_j$, and with probability $(1 - \pi_j^U)$ a random document among those not belonging to $y_j$. We thus obtain a sample $\sigma = (\sigma_L, \sigma_U)$ with which we run a train-and-test experiment.
(3) We repeat the two steps above $k$ times for each class $y_j \in \mathcal{Y}$ and average the results across these $37 \times k$ train-and-test experiments.

We also run single-label multiclass classification experiments, using varying number of classes. For these experiments

(1) given a desired number $n$ of classes, we randomly choose $n$ of our 37 RCV1-v2 classes, thus obtaining codeframe $\mathcal{Y}$, with $|\mathcal{Y}| = n$;
(2) we generate two random vectors $\Pi_L = (\pi_1^L, ..., \pi_{|\mathcal{Y}|}^L)$ and $\Pi_U = (\pi_1^U, ..., \pi_{|\mathcal{Y}|}^U)$ of class priors, i.e., two vectors such that $0 \leq \pi_j^L, \pi_j^U \leq 1$ for each $1 \leq j \leq |\mathcal{Y}|$ and such that $\sum_{j=1}^{|\mathcal{Y}|} \pi_j^L = \sum_{j=1}^{|\mathcal{Y}|} \pi_j^U = 1$;
(3) we generate a training set $\sigma_L$ (resp., a test set $\sigma_U$) by drawing $m_L = |\sigma_L|$ (resp., $m_U = |\sigma_U|$) different documents (with $m_L$ and $m_U$ two parameters to be fixed beforehand), where at each draw we pick with probability $\pi_j^L$ (resp., $\pi_j^U$) a document belonging to class $y_j$. We thus obtain a sample $\sigma = (\sigma_L, \sigma_U)$ with which we run a train-and-test experiment;
(4) we repeat the three steps above $k$ times and average the results across these $k$ train-and-test experiments.

In the experiments we run in this paper we use $m_L = m_U = 1000$, and $k = 500$. The fact that, as previously specified, we only consider classes with at least 2000 positive examples allows us to use $m_L = m_U = 1000$, i.e., there would be enough positive training examples even if, in some of the $k$ draws, $\pi_j^L$ and $\pi_j^U$ were both 1 for some $y_j$.[13] We run multiclass experiments for all values of $|\mathcal{Y}| \in \{5, 10, 20, 37\}$.

Thanks to the use of randomly generated drawing probabilities, the class distributions of both the training set and the test set of each sample are random, each class distribution is equiprobable, and the value of distribution shift (as measured by NAE) between the training set and the test set of each sample we generate is also random. The set of samples that we generate with this method is, since $k$ is large enough, fairly representative of the entire spectrum of shift values.

Note that this strategy for generating samples characterized by random values of distribution shift is radically different from the one adopted, for instance, in [10, 16]. In these latter works there is no random component in picking class distributions or distribution shift values, and an

[12]The method we use for generating each such vector is to pick two random real numbers in [0,1] and normalizing them so that they sum to 1. We have specified this since different methods to generate random vectors of class priors (for instance, picking a random number $x$ in [0,1] and using $(x, (1 - x))$ as the vector) are possible, and may yield different results. The same method is also used in Step 2 of the analogous process for multiclass experiments that we are going to describe next, of course using vectors with dimensionality equal to the number of classes considered.
[13]Note that documents are drawn from RCV1-v2 in its entirety, disregarding the "traditional" split of RCV1-v2 into 23,149 training documents and 781,265 test documents.

equal number of samples is generated for all possible class distributions such that each class prior belongs to a finite set of values (e.g., {.00, .01, ..., .99, 1.00}). However, those works deal only with the binary case, where the number of all possible such class distributions is small. In the general multiclass case (i.e., when $|\mathcal{Y}| > 2$) this number is much higher, since it grows exponentially with $|\mathcal{Y}|$; therefore, even generating a single sample for all possible class distributions such that each class prior is in {.00, .01, ..., .99, 1.00}, would be prohibitive even for small numbers of $|\mathcal{Y}|$. The random strategy we adopt in this paper thus allows us to avoid this pitfall.

### 3.3 Representing text

We preprocess text by using stop word removal and no stemming. As the weighting criterion we use a version of the well-known *tfidf* method, expressed as

$$tfidf(f, \mathbf{x}_i) = \log \#(f, \mathbf{x}_i) \times \log \frac{|L|}{|\mathbf{x} \in L : \#(f, \mathbf{x}) > 0|} \tag{13}$$

where $\#(f, \mathbf{x}_i)$ is the raw number of occurrences of feature $f$ in document $\mathbf{x}_i$; weights are then normalized by means of cosine normalization.

### 3.4 Learners

In our experiments we use four different learners, i.e., support vector machines (SVMs), logistic regression (LR), multinomial naive Bayes (MNB), and random forests (RFs). For all of them we rely on the implementations available from the `scikit-learn` package.[14] For all of them we use the default parameters of the `scikit-learn` implementation, since the possible accuracy improvements resulting from a parameter optimization based on $k$-fold cross-validation would be obtained at the expense of a very large computational cost.[15] This possible accuracy improvement would bring about no evident benefit to our study, since the goal of this work is not squeezing every possible drop of accuracy from our classifiers, but comparing the pre-SLD results with the post-SLD results in the same experimental conditions. The default values are as follows:

- SVMs: we use soft-margin SVMs with linear kernel, L2 regularization with $C = 1$;
- LR: we use L2 regularization with a regularization coefficient $C = 1$;
- MNB: We use Laplace smoothing, with $a = 1$ as the additive factor;
- RFs: we use 100 trees per forest, Gini impurity as the splitting function, no max depth, no pruning.

For each of these learners but SVMs we use two versions, one with post-calibration of the posteriors that the learner returns (Calib), and the other without calibration (NoCalib). SVMs are an exception because, as is well-known, the confidence scores they return are not probabilities, and the only way to have SVMs return probabilities in `scikit-learn` is to invoke a calibration routine; as a result, the only version of SVMs we experiment with is one with post-calibration.

We perform calibration using the method proposed by Platt [31], sometimes known as "Platt scaling".[16] Given a confidence score $s(\mathbf{x}_i, y_j)$ produced by a classifier, either in the form of a non-probabilistic score or of a non-calibrated probability, we transform it into a calibrated probability

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

[15] No parameter has been optimized because it would have been too expensive to do it individually for each of the 500 samples per dataset mentioned in Section 3.2.1, and because doing it on just one of the 500 samples and using the obtained parameter values for the other 499 would have been of dubious utility.

[16] We have implemented Platt scaling ourselves since the version available from `scikit-learn` turns out to be not a faithful implementation of Platt's algorithm; see https://github.com/scikit-learn/scikit-learn/issues/16145 for a discussion. The code of our implementation is available at https://github.com/aesuli/scikit-learn/tree/platt

$\Pr(y_j|\mathbf{x}_i)$ by applying the logistic transformation

$$\Pr(y_j|\mathbf{x}_i) = \frac{1}{1+\exp(\alpha \cdot s(\mathbf{x}_i, y_j) + \beta)} \tag{14}$$

where the parameters $\alpha$ and $\beta$ are determined by fitting a maximum-likelihood model on a set of scores $S_{\text{Calib}} = \{s(\mathbf{x}, y_j)|\mathbf{x} \in Tr_{\text{Calib}}\}$ produced by the classifier on some training documents $Tr_{\text{Calib}}$. If the same training documents that are used to train the classifier are also used for calibration, overfitting may happen. Held-out documents may be used, but this requires additional labelled documents. To avoid overfitting without requiring held-out documents, Platt suggests to collect the set of scores $S_{\text{Calib}}$ by performing cross-validation on the training documents. We have implemented this $k$-fold cross-validation procedure, performing 10-fold cross-validation on the training documents, using the same learning algorithm that is separately used on the entire training set in order to learn the actual classifier. We obtain the scores $S^f_{\text{Calib}}$ for each validation fold $f \in [1, \ldots, 10]$ and then optimize the parameters of Equation 14 on the resulting set of scores $S_{\text{Calib}} = \bigcup_f S^f_{\text{Calib}}$. We then apply the optimized Equation 14 to the scores of the classifier trained on the entire training set; we refer to this process as the Calib version of the learner.

## 4 RESULTS

This section presents the results of our experiments. The code for reproducing them is available at https://github.com/HLT-ISTI/SLD-reassessment. At https://hlt-isti.github.io/SLD-visualization/ we also make available a visualization tool that shows, for various combinations of (number of classes, sample, learner, class) from our experiments,

(1) how the prior of the chosen class as estimated by SLD evolves as a function of the number of iterations;
(2) as the values of the four evaluation metrics (as computed on this sample only) evolve as a function of the number of iterations.

Figure 1 shows sample plots as generated by our visualization tool.

### 4.1 Results of Binary Classification Experiments

The upper half of Table 1 reports, for our binary classification experiments, the values of NAE and of the isometric variants of BS, CE, RE, before ("Pre-SLD") and after ("Post-SLD") the application of SLD. In other words, Pre-SLD indicates the values computed directly on the outputs of the classifier, while Post-SLD indicates the values after these outputs have been updated by SLD. Since NAE, BS, CE, RE are all error measures, differences between Pre-SLD and Post-SLD are indicated in terms of *relative error reduction* $R_E = \frac{B-A}{B}$, where $E \in \{\text{NAE}, \text{BS}, \text{CE}, \text{RE}\}$ is the specific error measure, $B$ and $A$ are the Pre-SLD and Post-SLD values of $E$, respectively, and where the values of $R_E$ are reported (for simplicity) as percentages instead of as fractions.[17] Note that for $R_E$ a positive value indicates an improvement (i.e., that SLD had a beneficial effect) while a negative value indicates a deterioration. The rows of the table each correspond to one of the learners of Section 3.4, grouped into learners with post-calibration of the posteriors that the learner returns (Calib) and ones without such calibration (NoCalib). As indicated in Section 3.2.1, every row of this table is the result of 37×500=18,500 train-and-test runs; given that each of the 7 rows accounts for a different learner, this is a total of 18,500×7=129,500 train-and-test runs.

[17] In this table and in all the other tables in this paper, some values of $R_E$ might not appear to be completely justified; for instance, when the transition from a Pre-SLD value of .001 to a Post-SLD value of .000 is indicated to correspond to a value $R_E = +79.3\%$. Of course, this value of $R_E$ derives from using the real Pre-SLD and Post-SLD values in much higher precision. We use the standard notation (e.g., .027) rather than the more precise E notation (e.g., 2.7E-3) for higher legibility.

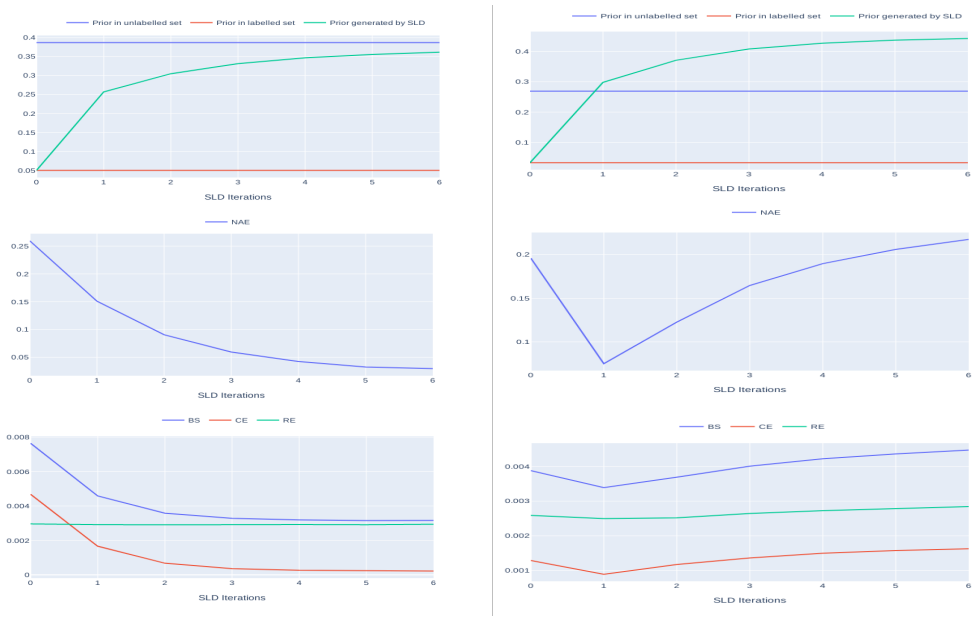


Fig. 1. Plots from the visualization tool at https://hlt-isti.github.io/SLD-visualization/, showing the evolution, as a function of the number of SLD iterations, (top) of the prior of a chosen class as estimated by SLD, (center) of NAE, and (bottom) of BS, CE and RE. The plots on the left show a successful application of SLD (3 of the 4 metrics improve and the 4th stays constant), and the distance between the prior generated by SLD and the true prior in the unlabelled set $U$ diminishes, while the plots on the right show an unsuccessful such application (all 4 metrics worsen, and the distance between the prior generated by SLD and the true prior in the unlabelled set $U$ increases).

There are a number of observations that can be derived from the top part of Table 1:

- In terms of the quality of the estimated priors (as measured by NAE), there is a very substantial difference between the performance of non-calibrated learners and that of calibrated learners: for the former, the application of SLD brings about an extremely large deterioration (an average of 72.7% across all tested learners), while it brings about a very good improvement (an average of 43.5% across all tested learners) for the latter.[18] This adds to the fact that calibrated learners have, on average, a much better NAE right from the start (.009, instead of .046 for the non-calibrated ones); this means that *calibrating one's learner is a win-win move*,

[18]This confirms an observation of Saerens et al. [34], according to whom "In order to obtain good a priori probability estimates [by means of SLD], it is necessary that the a posteriori probabilities relative to the training set are reasonably well approximated (i.e., sufficiently well estimated by the model)".

Table 1. Values of NAE, BS, CE, RE, before and after the application of SLD, for binary classification experiments.

| | | | Priors | | | Posteriors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NAE | | | BS | | | CE | | | RE | | |
| | | | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction |
| Isometric | NoCalib | LR | .005 | .013 | -136.9% | .011 | .011 | -4.2% | .008 | .006 | +21.4% | .003 | .005 | -68.8% |
| | | MNB | .116 | .186 | -60.3% | .020 | .025 | -26.0% | .013 | .015 | -15.6% | .007 | .011 | -44.4% |
| | | RF | .016 | .039 | -142.1% | .010 | .007 | +26.2% | .006 | .003 | +46.9% | .003 | .004 | -12.1% |
| | | Avg | .046 | .079 | -72.7% | .013 | .015 | -7.8% | .009 | .008 | +9.6% | .005 | .006 | -41.8% |
| | Calib | SVM | .005 | .004 | +21.5% | .003 | .002 | +31.7% | .001 | .000 | +79.3% | .002 | .002 | +8.6% |
| | | LR | .008 | .002 | +78.2% | .004 | .003 | +29.1% | .001 | .000 | +75.8% | .002 | .002 | +7.6% |
| | | MNB | .023 | .014 | +39.1% | .009 | .006 | +35.3% | .004 | .001 | +72.8% | .005 | .005 | +8.3% |
| | | RF | .002 | .002 | +17.5% | .005 | .003 | +32.7% | .002 | .000 | +81.6% | .003 | .003 | +7.7% |
| | | Avg | .009 | .005 | +43.5% | .005 | .004 | +33.0% | .002 | .000 | +76.1% | .003 | .003 | +8.1% |
| Isomerous | NoCalib | LR | .005 | .013 | -136.9% | .012 | .012 | -3.0% | .009 | .009 | -4.1% | .003 | .003 | +0.0% |
| | | MNB | .116 | .186 | -60.3% | .022 | .028 | -26.5% | .017 | .023 | -35.3% | .005 | .005 | -0.0% |
| | | RF | .016 | .039 | -142.1% | .011 | .008 | +28.0% | .007 | .004 | +44.4% | .004 | .004 | +0.0% |
| | | Avg | .046 | .079 | -72.7% | .015 | .016 | -7.0% | .011 | .012 | -9.8% | .004 | .004 | +0.0% |
| | Calib | SVM | .005 | .004 | +21.5% | .004 | .003 | +23.7% | .001 | .000 | +90.1% | .003 | .003 | +0.0% |
| | | LR | .008 | .002 | +78.2% | .004 | .003 | +22.8% | .001 | .000 | +86.5% | .003 | .003 | +0.0% |
| | | MNB | .023 | .014 | +39.1% | .010 | .007 | +33.1% | .004 | .001 | +73.5% | .005 | .005 | -0.0% |
| | | RF | .002 | .002 | +17.5% | .006 | .004 | +28.0% | .002 | .000 | +86.8% | .004 | .004 | +0.0% |
| | | Avg | .009 | .005 | +43.5% | .006 | .004 | +28.4% | .002 | .000 | +80.3% | .004 | .004 | -0.0% |

given that it brings about much better posterior probabilities *and* that these posteriors have much larger margins of improvement by means of the application of SLD.[19]

- The very large magnitude of these improvements / deteriorations is not mirrored by analogous magnitudes when it comes to the quality of the posteriors. It is still true that deteriorations are observed in the case of non-calibrated classifiers (7.8% on average) and improvements are instead observed for the calibrated classifiers (33.0% on average), but the magnitudes of these variations are smaller.
- SLD seems to have a much more beneficial effect in terms of calibration than in terms of refinement; in fact improvements in BS, when present, are largely the responsibility of CE, while deteriorations in BS, when present, are largely the responsibility of RE.

[19]Niculescu-Mizil and Caruana [29] state that "For learning methods that make well calibrated predictions such as neural nets, bagged trees, and logistic regression, neither Platt Scaling nor Isotonic Regression yields much improvement in performance even when the calibration set is very large. With these methods calibration is not beneficial, and actually hurts performance when the calibration sets are small." Our large-scale experimentation indicates that, while this might be true in the absence of distribution shift, when distribution shift is present any calibrated learner works better than its non-calibrated counterpart. In fact, note that the Pre-SLD values of NAE, BS and CE for the calibrated learners are always substantially better than the values of the corresponding non-calibrated learners, and this also includes logistic regression, a learner that is known to return well calibrated probabilities.

- While there are (even substantial) *quantitative* differences among learners belonging to the same category (non-calibrated or calibrated), there are very few *qualitative* differences, i.e., when a learner exhibits a deterioration in one of the measures, all other learners (with few exceptions) also exhibit a deterioration for the same measure. This seems to indicate that the results derive from inherent properties of the SLD algorithm, rather than from peculiarities of the individual learning algorithms.

The lower half of Table 1 presents the analogous results for the isomerous variants of BS, CE, RE. (The results for NAE are the same as in the upper half, since the distinction isometric/isomerous does not apply to NAE.) The observations that can be made by looking at the lower half the table are essentially the same as those derived from the upper half, since the results are qualitatively similar. There is one important difference, though, i.e., the fact that, when measured by means of the isomerous variant, RE is always 0 or very close to 0, which is far from being the case when using isometric RE. That this should be so is an obvious consequence of the definition of RE, as from Equation 11. In fact, since all bins are equally populated, it is clear from Equation 11 that RE only depends, for all bins $B_{kj}$ ($1 \leq k \leq b$) and for all classes $y_j \in \mathcal{Y}$, on the fraction $\rho(y_j, B_{kj})$ of documents in the bin that belong to the class. However, for all bins, that fraction is the same in the Pre-SLD and Post-SLD distributions, because, as observed in Section 2, SLD is just a *rescaling* algorithm, that multiplies all the posteriors for a given class for the same constant but does not change the composition of the bins. That RE is not 0 when using the isometric variant is thus due to the fact that rescaling changes the compositions of the bins; for instance, a document that was in the bin corresponding to the [.9, 1.0] interval before the application of SLD, after SLD has been applied might be in the [.8, .9) bin if SLD has multiplied all the posteriors for that class by a factor smaller than 1. In this case, rescaling not only changes the composition of the bins, but also changes the number of documents they contain, thus potentially generating also very sparse bins.

Interestingly, the fact that SLD could not reduce RE is reminiscent of an observation by DeGroot and Fienberg [7]:

> We then study the question of when an observer can use a forecaster's predictions to obtain a better score than the forecaster himself, and show that such an improvement can be achieved by the observer essentially if and only if the forecaster is not well calibrated.

Here, the forecaster is the classifier and the observer is SLD, who tries to obtain a better score (in terms of Brier score) than the classifier by "piggybacking" on the classifier's predictions. DeGroot and Fienberg [7] state that what SLD can at most hope for, is to improve on the classifier's *calibration* error, but not on its *refinement* error. This shows that SLD is, in essence, a *re-calibration algorithm*, i.e., an algorithm for re-calibrating the posterior probabilities of documents belonging to an unlabelled set $U$, where these posteriors have been returned by a classifier already calibrated on a training set $L$, and where the re-calibration is made necessary by the fact that a prior probability shift between $L$ and $U$ has occurred.

### 4.2 Results of Multiclass Classification Experiments

Tables 2 to 5 report the results of our experiments on multiclass classification. As indicated in Section 3.2.1, we run multiclass experiments with varying number of classes, starting from $|\mathcal{Y}| = 5$ classes (Table 2) and moving up to $|\mathcal{Y}| = 10$ (Table 3), $|\mathcal{Y}| = 20$ (Table 4), and $|\mathcal{Y}| = 37$ (Table 5), which is the total number of classes in our dataset. For $|\mathcal{Y}| \in \{5, 10, 20\}$, the classes are randomly sampled from the entire set of 37 RCV1-v2 classes. As indicated in Section 3.2.1, every row of these 4 tables is the result of 500 train-and-test runs; given that each of the 7 rows accounts for a different learner, this is a total of 4×500×7=14,000 train-and-test runs.

Table 2. As Table 1, but for multiclass classification (5 classes).

| | | | Priors | | | Posteriors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NAE | | | BS | | | CE | | | RE | | |
| | | | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction |
| Isometric | NoCalib | LR | .007 | .003 | +50.1% | .005 | .007 | -33.9% | .004 | .004 | -25.2% | .002 | .003 | -49.3% |
| | | MNB | .022 | .030 | -36.7% | .009 | .012 | -36.9% | .005 | .006 | -25.3% | .004 | .006 | -49.6% |
| | | RF | .014 | .036 | -154.9% | .005 | .005 | +11.0% | .003 | .002 | +31.9% | .002 | .003 | -22.4% |
| | | Avg | .014 | .023 | -61.6% | .007 | .008 | -22.9% | .004 | .004 | -8.6% | .003 | .004 | -42.7% |
| | Calib | SVM | .009 | .006 | +37.5% | .003 | .003 | +2.6% | .001 | .001 | -2.5% | .002 | .002 | +5.0% |
| | | LR | .004 | .009 | -133.0% | .002 | .002 | -31.6% | .001 | .001 | -48.2% | .001 | .001 | -22.4% |
| | | MNB | .015 | .017 | -15.1% | .005 | .004 | +21.7% | .002 | .001 | +40.1% | .003 | .003 | +10.7% |
| | | RF | .005 | .005 | -4.8% | .003 | .002 | +24.4% | .001 | .000 | +53.5% | .002 | .002 | +9.7% |
| | | Avg | .008 | .009 | -12.7% | .003 | .003 | +9.5% | .001 | .001 | +19.9% | .002 | .002 | +3.8% |
| Isomerous | NoCalib | LR | .007 | .003 | +50.1% | .006 | .008 | -28.2% | .004 | .006 | -42.5% | .003 | .003 | -5.7% |
| | | MNB | .022 | .030 | -36.7% | .010 | .014 | -35.4% | .006 | .009 | -51.7% | .004 | .004 | -10.7% |
| | | RF | .014 | .036 | -154.9% | .006 | .006 | +12.8% | .004 | .003 | +29.2% | .003 | .003 | -6.9% |
| | | Avg | .014 | .023 | -61.6% | .008 | .009 | -19.8% | .005 | .006 | -27.8% | .003 | .003 | -8.2% |
| | Calib | SVM | .009 | .006 | +37.5% | .004 | .004 | -0.5% | .001 | .001 | +4.2% | .003 | .003 | -2.0% |
| | | LR | .004 | .009 | -133.0% | .003 | .004 | -20.2% | .001 | .001 | -88.9% | .002 | .002 | -1.8% |
| | | MNB | .015 | .017 | -15.1% | .006 | .005 | +18.4% | .002 | .001 | +43.8% | .004 | .003 | +4.7% |
| | | RF | .005 | .005 | -4.8% | .004 | .003 | +16.5% | .001 | .000 | +60.6% | .003 | .003 | +2.7% |
| | | Avg | .008 | .009 | -12.7% | .004 | .004 | +6.4% | .001 | .001 | +19.7% | .003 | .003 | +1.3% |

There are several observations we can make by looking at these tables:

- The main fact that emerges is that *all quality indicators of SLD* (i.e., the values of error reduction, for each of the four error measures we consider) *drastically deteriorate when* $|\mathcal{Y}|$ *grows*, for all learners, calibrated or not. Table 5, that reports results for $|\mathcal{Y}| = 37$, indicates disastrous performance on the part of SLD on all counts.
- Concerning SLD's impact on the priors, while the binary experiments had indicated a very positive impact (at least: for the calibrated learners), the multiclass experiments indicate a negative impact for $|\mathcal{Y}| = 5$ (12.7% average deterioration across all calibrated learners) and an even more negative impact for $|\mathcal{Y}| \in \{10, 20, 37\}$, with the average deterioration across all calibrated learners reaching up to 251.0% for $|\mathcal{Y}| = 37$.
- Concerning SLD's impact on the posteriors, while the binary experiments had indicated a very positive impact (at least: for the calibrated learners), the multiclass experiments indicate that this impact is still mildly positive for $|\mathcal{Y}| = 5$ but becomes negative for $|\mathcal{Y}| = 10$ and deteriorates even more for $|\mathcal{Y}| \in \{20, 37\}$. For instance, BS in the isomerous variant has, thanks to SLD, an average improvement across the calibrated learners by 28.0% for $|\mathcal{Y}| = 2$ and 6.4% for $|\mathcal{Y}| = 5$, but this improvement becomes a deterioration for $|\mathcal{Y}| = 10$ (28.2%) and for $|\mathcal{Y}| \in \{20, 37\}$ (e.g., 72.2% for $|\mathcal{Y}| = 37$). This trend is even more marked for CE, and indicates an improvement for $|\mathcal{Y}| = 2$ (80.3%, average across the calibrated learners) and

Table 3. As Table 2, but with 10 classes.

| | | | Priors | | | Posteriors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NAE | | | BS | | | CE | | | RE | | |
| | | | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction |
| Isometric | NoCalib | LR | .014 | .030 | -107.5% | .005 | .007 | -61.0% | .002 | .004 | -55.1% | .002 | .004 | -67.4% |
| | | MNB | .014 | .035 | -143.5% | .006 | .009 | -51.8% | .002 | .004 | -59.9% | .004 | .005 | -46.5% |
| | | RF | .018 | .068 | -281.6% | .004 | .004 | +0.3% | .002 | .001 | +26.3% | .002 | .003 | -24.7% |
| | | Avg | .016 | .044 | -185.4% | .005 | .007 | -40.2% | .002 | .003 | -32.7% | .003 | .004 | -46.7% |
| | Calib | SVM | .015 | .022 | -41.3% | .002 | .004 | -43.1% | .001 | .001 | -97.5% | .002 | .002 | -21.9% |
| | | LR | .018 | .026 | -45.1% | .002 | .004 | -93.6% | .001 | .002 | -173.3% | .001 | .002 | -58.7% |
| | | MNB | .017 | .020 | -19.3% | .003 | .004 | -18.1% | .001 | .001 | -38.0% | .002 | .003 | -9.9% |
| | | RF | .008 | .020 | -143.7% | .002 | .003 | -22.4% | .001 | .001 | -52.3% | .002 | .002 | -12.3% |
| | | Avg | .015 | .022 | -50.4% | .003 | .003 | -39.6% | .001 | .001 | -83.3% | .002 | .002 | -22.4% |
| Isomerous | NoCalib | LR | .014 | .030 | -107.5% | .005 | .008 | -52.0% | .003 | .006 | -93.7% | .003 | .003 | -4.8% |
| | | MNB | .014 | .035 | -143.5% | .007 | .010 | -48.6% | .003 | .006 | -87.0% | .004 | .004 | -14.2% |
| | | RF | .018 | .068 | -281.6% | .005 | .005 | +3.0% | .002 | .002 | +16.5% | .003 | .003 | -6.6% |
| | | Avg | .016 | .044 | -185.4% | .006 | .008 | -34.9% | .003 | .004 | -63.4% | .003 | .003 | -9.1% |
| | Calib | SVM | .015 | .022 | -41.3% | .003 | .005 | -31.5% | .001 | .002 | -158.8% | .003 | .003 | -1.8% |
| | | LR | .018 | .026 | -45.1% | .003 | .005 | -58.8% | .000 | .002 | -319.4% | .002 | .003 | -6.1% |
| | | MNB | .017 | .020 | -19.3% | .004 | .005 | -14.0% | .001 | .002 | -65.0% | .003 | .003 | +1.4% |
| | | RF | .008 | .020 | -143.7% | .003 | .004 | -16.1% | .001 | .001 | -102.1% | .003 | .003 | +0.3% |
| | | Avg | .015 | .022 | -50.4% | .003 | .004 | -28.2% | .001 | .002 | -142.3% | .003 | .003 | -1.3% |

$|\mathcal{Y}| = 5$ (19.7%) but a deterioration for higher values of $|\mathcal{Y}|$, with the amount of deterioration reaching up to 937.2% for $|\mathcal{Y}| = 37$.

### 4.3 Analyzing the Results by Amount of Shift

In this section we analyze the relations between error and distribution shift. Our goal is that of highlighting, in the results of the experiments discussed in Sections 4.1 and 4.2, any noteworthy correlation between error reduction, for any of our four measures, and distribution shift.

In our analysis of the results, we have not been able to detect any significant correlation between NAE and distribution shift, or between RE and distribution shift. As a result, from here onwards we only concentrate on discussing BS and CE. Figure 2 plots the values of relative error reduction for the BS and CE measures (we here use the isomerous variants; the isometric variants return similar results) for the same experiments as discussed in Sections 4.1 and 4.2, for each of our 4 calibrated classifiers,[20] but with the samples binned into four quartiles according to how much distribution shift between the training set and test set the sample exhibits. The 1st quartile contains the samples characterized by the lowest amounts of distribution shift, and the 4th contains the samples characterized by the highest such amounts. The actual values of distribution shift (expressed

[20]We omit discussing the non-calibrated classifiers since the experiments of Sections 4.1 and 4.2 have clearly indicated that SLD requires, in order to perform well, calibrated classifiers.

Table 4. As Table 2, but with 20 classes.

| | | | Priors | | | Posteriors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NAE | | | BS | | | CE | | | RE | | |
| | | | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction |
| Isometric | NoCalib | LR | .017 | .081 | -382.2% | .003 | .006 | -72.3% | .001 | .002 | -95.6% | .002 | .003 | -59.2% |
| | | MNB | .021 | .075 | -260.7% | .004 | .006 | -64.6% | .001 | .002 | -130.9% | .003 | .004 | -39.0% |
| | | RF | .025 | .118 | -379.4% | .003 | .003 | -11.0% | .001 | .001 | +17.0% | .002 | .002 | -27.4% |
| | | Avg | .021 | .091 | -340.5% | .003 | .005 | -52.2% | .001 | .002 | -71.3% | .002 | .003 | -42.6% |
| | Calib | SVM | .030 | .062 | -110.5% | .002 | .004 | -86.3% | .001 | .001 | -175.3% | .002 | .002 | -54.4% |
| | | LR | .024 | .047 | -96.7% | .002 | .003 | -114.5% | .000 | .001 | -230.7% | .001 | .002 | -74.7% |
| | | MNB | .015 | .047 | -205.2% | .002 | .004 | -85.2% | .000 | .002 | -234.7% | .002 | .002 | -38.4% |
| | | RF | .015 | .040 | -162.8% | .002 | .003 | -87.2% | .000 | .001 | -265.7% | .001 | .002 | -40.7% |
| | | Avg | .021 | .049 | -133.4% | .002 | .004 | -92.3% | .000 | .001 | -222.4% | .001 | .002 | -50.9% |
| Isomerous | NoCalib | LR | .017 | .081 | -382.2% | .004 | .006 | -61.8% | .002 | .004 | -149.6% | .002 | .002 | -3.7% |
| | | MNB | .021 | .075 | -260.7% | .004 | .007 | -58.9% | .001 | .003 | -144.7% | .003 | .003 | -13.6% |
| | | RF | .025 | .118 | -379.4% | .003 | .004 | -6.1% | .001 | .001 | -9.8% | .003 | .003 | -4.8% |
| | | Avg | .021 | .091 | -340.5% | .004 | .006 | -44.1% | .001 | .003 | -115.7% | .003 | .003 | -7.6% |
| | Calib | SVM | .030 | .062 | -110.5% | .003 | .005 | -57.3% | .000 | .002 | -395.2% | .002 | .003 | -1.4% |
| | | LR | .024 | .047 | -96.7% | .003 | .004 | -62.9% | .000 | .002 | -502.6% | .002 | .002 | -4.3% |
| | | MNB | .015 | .047 | -205.2% | .003 | .005 | -56.2% | .000 | .002 | -407.9% | .003 | .003 | -0.6% |
| | | RF | .015 | .040 | -162.8% | .003 | .004 | -52.1% | .000 | .002 | -491.1% | .002 | .002 | -1.0% |
| | | Avg | .021 | .049 | -133.4% | .003 | .004 | -57.0% | .000 | .002 | -441.6% | .002 | .002 | -1.8% |

in terms of NAE) that characterize the samples in each quartile are reported in Table 6. Each result reported in the plots is the average across all samples that belong to the bin.

A clear pattern emerges from the analysis of BS and CE values: for both measures, for both the binary and multiclass cases, and for all numbers of classes considered in the multiclass experiments ($|\mathcal{Y}| \in \{5, 10, 20, 37\}$), *performance tends to improve monotonically with the amount of distribution shift*. (Exceptions do exist for individual classifiers, but the average values across the 4 classifiers exhibits strict monotonicity). This happens both for the cases (binary case + multiclass case with $|\mathcal{Y}| = 5$) in which SLD has a positive impact (i.e., error diminishes as a result of its application), and for the cases (multiclass case with $|\mathcal{Y}| > 5$) in which the impact of SLD is negative (i.e., error increases as a result of its application); in the former cases the magnitude of error reduction increases with the increase in shift, while in the latter cases the magnitude of error amplification diminishes with the increase in shift. Case $|\mathcal{Y}| = 5$ seems the threshold here, with SLD yielding a decrease in error for the two quartiles representing low shift and an increase in error for the two quartiles representing high shift.

Together with the analyses presented in Sections 4.1 and 4.2, this observation suggests that SLD should be used to improve the quality of the prior probability estimates and of the posterior

Table 5. As Table 2, but with 37 classes.

| | | | Priors | | | Posteriors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NAE | | | BS | | | CE | | | RE | | |
| | | | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction | Pre-SLD | Post-SLD | Error Reduction |
| Isometric | NoCalib | LR | .014 | .137 | -913.5% | .002 | .003 | -50.5% | .000 | .001 | -149.9% | .002 | .002 | -25.6% |
| | | MNB | .022 | .115 | -411.6% | .002 | .004 | -67.8% | .000 | .002 | -326.1% | .002 | .002 | -19.2% |
| | | RF | .028 | .159 | -458.5% | .002 | .002 | -19.6% | .000 | .000 | +3.7% | .001 | .002 | -28.4% |
| | | Avg | .021 | .137 | -537.7% | .002 | .003 | -47.9% | .000 | .001 | -140.3% | .002 | .002 | -23.9% |
| | Calib | SVM | .031 | .091 | -193.1% | .002 | .003 | -115.1% | .000 | .001 | -278.9% | .001 | .002 | -67.9% |
| | | LR | .026 | .067 | -160.4% | .001 | .003 | -112.8% | .000 | .001 | -269.0% | .001 | .002 | -69.5% |
| | | MNB | .015 | .071 | -368.2% | .001 | .003 | -135.0% | .000 | .002 | -447.1% | .001 | .002 | -56.9% |
| | | RF | .016 | .080 | -398.3% | .001 | .003 | -138.2% | .000 | .001 | -501.4% | .001 | .002 | -62.1% |
| | | Avg | .022 | .077 | -251.0% | .001 | .003 | -125.2% | .000 | .001 | -363.9% | .001 | .002 | -64.0% |
| Isomerous | NoCalib | LR | .014 | .137 | -913.5% | .002 | .004 | -45.7% | .001 | .002 | -184.1% | .002 | .002 | -1.7% |
| | | MNB | .022 | .115 | -411.6% | .003 | .004 | -64.3% | .001 | .002 | -271.6% | .002 | .002 | -7.5% |
| | | RF | .028 | .159 | -458.5% | .002 | .003 | -12.8% | .000 | .001 | -83.0% | .002 | .002 | -2.4% |
| | | Avg | .021 | .137 | -537.7% | .002 | .003 | -41.9% | .000 | .001 | -196.4% | .002 | .002 | -3.9% |
| | Calib | SVM | .031 | .091 | -193.1% | .002 | .004 | -74.0% | .000 | .002 | -803.6% | .002 | .002 | -0.7% |
| | | LR | .026 | .067 | -160.4% | .002 | .003 | -57.9% | .000 | .001 | -790.7% | .002 | .002 | -1.1% |
| | | MNB | .015 | .071 | -368.2% | .002 | .004 | -79.7% | .000 | .002 | -1009.1% | .002 | .002 | -0.9% |
| | | RF | .016 | .080 | -398.3% | .002 | .004 | -76.8% | .000 | .002 | -1206.9% | .002 | .002 | -0.9% |
| | | Avg | .022 | .077 | -251.0% | .002 | .004 | -72.2% | .000 | .002 | -937.2% | .002 | .002 | -0.9% |

Table 6. Values of distribution shift (expressed in terms of NAE) for the samples in each of the four quartiles in which all samples are binned; for each quartile we indicate minimum shift and maximum shift of the samples the quartiles actually contain.

| | 2 classes | | 5 classes | | 10 classes | | 20 classes | | 37 classes | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Min | Max | Min | Max | Min | Max | Min | Max | Min | Max |
| 1st quartile | .000 | .168 | .063 | .253 | .116 | .274 | .177 | .297 | .224 | .311 |
| 2nd quartile | .168 | .343 | .254 | .334 | .274 | .329 | .297 | .335 | .311 | .342 |
| 3rd quartile | .343 | .557 | .334 | .413 | .329 | .387 | .336 | .373 | .342 | .372 |
| 4th quartile | .557 | .993 | .414 | .746 | .387 | .644 | .374 | .534 | .372 | .483 |

probabilities, only (a) when the classifier has been calibrated, *and* (b) the number of classes in the codeframe $\mathcal{Y}$ is low (say, $|\mathcal{Y}| \leq 5$), *and* (c) when the amount of distribution shift is high enough.[21]

### 4.4 Analyzing the Distributions Produced by SLD

In the previous sections we have evaluated, among other things, the impact of SLD on the difference between the predicted class distribution and the true class distribution, by using the NAE measure.

[21] Statistical tests are indeed available that allow to detect how much distribution shift there is between the training documents and the unlabelled documents; one such test (a likelihood ratio test) is presented in [34, §3].

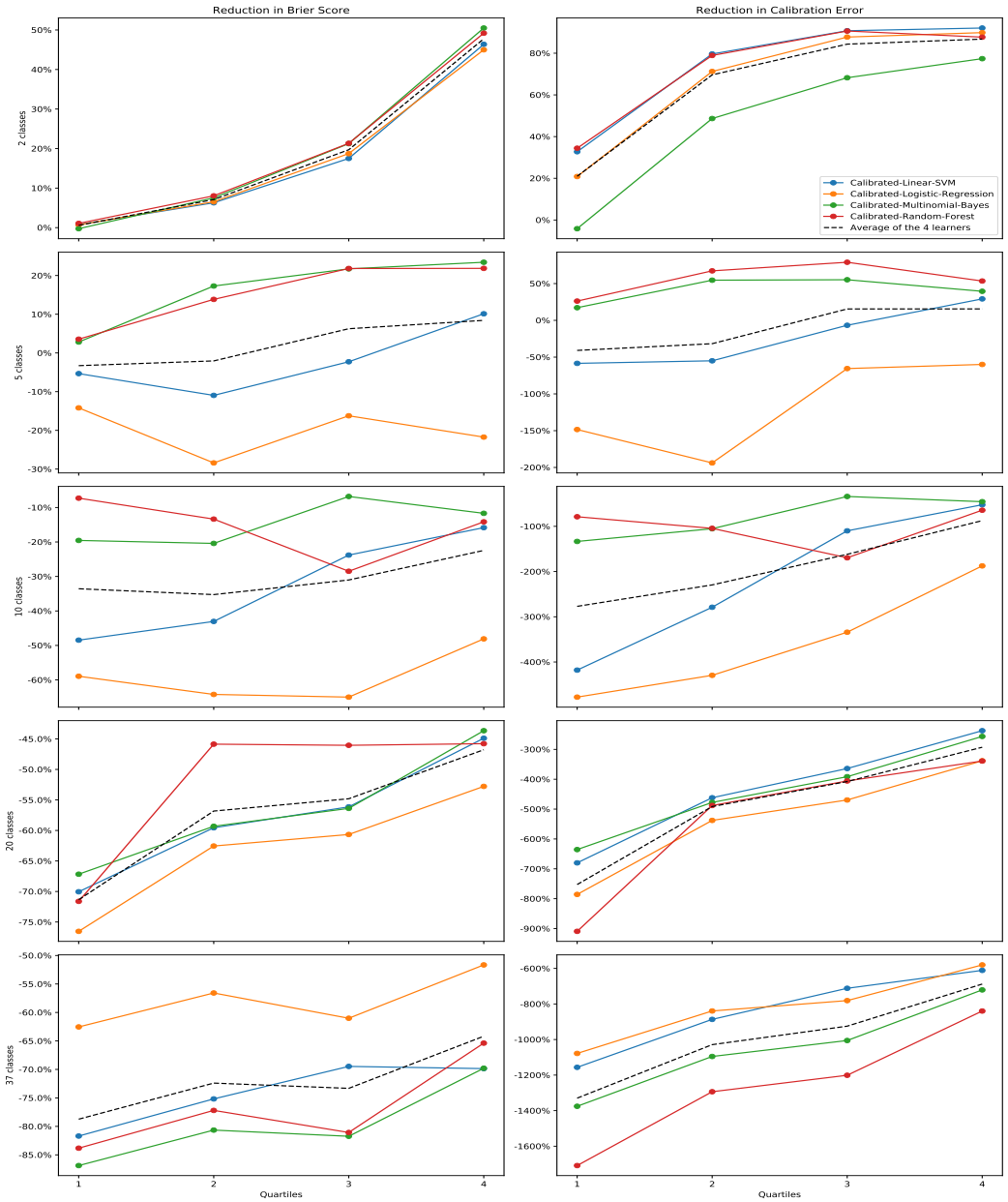


Fig. 2. Error reduction for the isomerous variants of Brier Score (left) and Calibration Error (right) for the four different quartiles into which samples have been binned; in each of the ten subfigures, quartiles are arranged with low-shift quartiles on the left and high-shift quartiles on the right. Subfigures are sorted top-to-bottom as a function of the number of classes considered, from $|\mathcal{Y}| = 2$ (top) to $|\mathcal{Y}| = 37$ (bottom).

In this section we instead look at the impact of SLD on two intrinsic characteristics of class

distributions, i.e., their entropy and their shape. In order to do so we compare, for a given train-and-test run, the four class distributions involved: (a) the true class distribution of $L$, (b) the true class distribution of $U$, (c) the class distribution of $U$ predicted by the classifier (which SLD receives as input), and (d) the class distribution of $U$ returned by SLD. This allows us to better understand the impact of SLD, and the reasons behind some of the patterns highlighted in Sections 4.1 through 4.3.

*4.4.1 Average Entropy of Class Distributions.* For each of the 129,500+14,000=143,500 train-and-test runs we have discussed in Sections 4.1 and 4.2, we measure the entropy

$$H(\mathcal{Y}) = -\sum_{i=1}^{|\mathcal{Y}|} \Pr(y_i) \log_{|\mathcal{Y}|} \Pr(y_i) \tag{15}$$

of the four class distributions (a) to (d) mentioned in the previous paragraph. In each case we set the base of the logarithm to the number $|\mathcal{Y}|$ of classes of the distribution being observed, so that entropy values always range between 0 and 1. A low entropy value means that most of the documents in the sample belong to one or few classes, while a high entropy value means that the documents are spread fairly evenly across the entire set of classes.

Table 7 shows the average value of the entropy of the four class distributions (a) to (d) (which in this and in the following tables will be indicated as $L$, $U$, Pre-SLD, and Post-SLD, respectively) across all the 143,500 train-and-test runs.

| | $L$ | $U$ | Pre-SLD | Post-SLD |
|---|---|---|---|---|
| Entropy | .902 | .902 | .906 | .638 |

Table 7. Values of the entropy of the four class distributions, averaged across all train-and-test runs.

From this table we can observe that the values for $L$ and $U$ are the same. This is intuitive, since, even though the training set and the test set of a given sample have in general two different class distributions, the sampling method for generating training sets and test sets is the same and the pool of documents from which to sample is the same, so training sets and test sets will exhibit, on average, the same class distribution. In the following we will not report the entropy values for $L$, since those for $U$ are always practically identical.

The average entropy value of Pre-SLD class distributions is slightly higher than those for $L$ and $U$, but not substantially so. However, what immediately jumps to the eye is that the average entropy value for Post-SLD is much lower than the values for $L$, $U$, and Pre-SLD. Similar distributions exhibit similar entropy values, so a difference in entropy values is a clear indicator of a dissimilarity between the two observed distributions. The sharp difference between the Pre-SLD and Post-SLD average entropy values thus unequivocally indicates that *SLD substantially alters the Pre-SLD class distribution*, and the sharp difference between the $U$ and Post-SLD values indicates that *this alteration is detrimental*. Since a high entropy value indicates a highly uniform distribution, the above results indicate that SLD has a tendency to sharply diminish this uniformity, and label most of the documents with one or few classes.

Table 8 reports again entropy values of class distributions, but averaged across all runs characterized by the same number of classes in the codeframe. An analysis of this table shows that values for $U$ tend to increase as the number $|\mathcal{Y}|$ of classes in the codeframe increases. This is due to the sampling method, that generates prior probabilities with mean equal to $\frac{1}{|\mathcal{Y}|}$ and variance equal to $\frac{1}{|\mathcal{Y}|^2}$ (as will also be evident from Figures 4 to 7). The values for Pre-SLD follow the same trend

Table 8. Values of the entropy of the four class distributions, averaged across all train-and-test runs with the same number of classes in the codeframe.

| # of classes | $U$ | Pre-SLD | Post-SLD |
|---|---|---|---|
| 2 | .821 | .819 | .704 |
| 5 | .894 | .891 | .787 |
| 10 | .919 | .917 | .721 |
| 20 | .935 | .935 | .574 |
| 37 | .943 | .946 | .405 |

as values for $U$, but the values for Post-SLD have an almost *opposite* trend: as the number $|\mathcal{Y}|$ of classes in the codeframe increases, SLD decreases the uniformity of class distributions.

Table 9 reports again entropy values of class distributions, but averaged across all runs that use the same learning algorithm. The application of SLD following the use of the No-Calib learners

Table 9. Values of the entropy of the four class distributions, averaged across all train-and-test runs obtained by means of the same learning algorithm.

| | | $U$ | Pre-SLD | Post-SLD |
|---|---|---|---|---|
| No-Calib | LR | .902 | .925 | .501 |
| | MNB | .902 | .802 | .395 |
| | RF | .902 | .923 | .751 |
| | Avg | .902 | .883 | .549 |
| Calib | SVM | .902 | .914 | .698 |
| | LR | .902 | .919 | .719 |
| | MNB | .902 | .906 | .687 |
| | RF | .902 | .921 | .719 |
| | Avg | .902 | .915 | .706 |

brings about an even stronger divergence between the $U$ values and the Post-SLD values than the corresponding Calib versions, thus confirming that non-calibrated classifiers are not fit for use with SLD. The application of SLD drastically reduces the average entropy for all learners, thus indicating that the decrease in uniformity of the distributions is less related to the chosen learning algorithm than to the number $|\mathcal{Y}|$ of classes in the codeframe (see also Section 4.4.2).

Table 10 shows the average entropy values of the class distributions for all the possible combinations of number of classes and learners. From this table we can identify the very few cases in which the $U$ class distributions have an average entropy value closer to the Post-SLD value than to the Pre-SLD value: this happens only for $|\mathcal{Y}| = 2$ with calibrated SVMs, MNB, and RF.

*4.4.2 Histogram-Based Representations of Class Distributions.* In this section we display and comment on histograms that indicate how class prevalence values are distributed in $U$, Pre-SLD, and Post-SLD class distributions resulting from the use of specific learners and with specific numbers of classes. Figure 3 does this for $|\mathcal{Y}| = 2$, i.e., the binary classification case. As an example, the histogram in its left bottom subfigure ("Pre-SLD - Random Forests") shows that, if we pool together all the 37×500=18,500 train-and-test runs where RF was used as the learner, the results returned by the classifier (i.e., Pre-SLD) are such that a high number of classes have a prevalence of about 50% (i.e., $\Pr(y) = .5$), a slightly lower number of classes have a prevalence of 40%, ..., and a very small number of classes have a prevalence of 0%. Every subfigure of Figure 3 is, of course, bilaterally

Table 10. Values of the entropy of the four class distributions, averaged across all train-and-test runs with the same number of classes *and* obtained by means of the same learning algorithm.

| # of classes | | No-Calib | | | Calib | | | |
|---|---|---|---|---|---|---|---|---|
| | | LR | MNB | RF | SVM | LR | MNB | RF |
| 2 | $U$ | .821 | .821 | .821 | .821 | .821 | .821 | .821 |
| | Pre-SLD | .866 | .631 | .868 | .840 | .828 | .846 | .851 |
| | Post-SLD | .577 | .409 | .693 | .815 | .799 | .820 | .813 |
| 5 | $U$ | .894 | .894 | .894 | .894 | .894 | .894 | .894 |
| | Pre-SLD | .921 | .772 | .922 | .905 | .913 | .896 | .909 |
| | Post-SLD | .708 | .567 | .804 | .858 | .851 | .854 | .869 |
| 10 | $U$ | .919 | .919 | .919 | .919 | .919 | .919 | .919 |
| | Pre-SLD | .939 | .821 | .934 | .931 | .940 | .914 | .938 |
| | Post-SLD | .552 | .479 | .789 | .809 | .780 | .798 | .842 |
| 20 | $U$ | .935 | .935 | .935 | .935 | .935 | .935 | .935 |
| | Pre-SLD | .947 | .874 | .943 | .946 | .953 | .932 | .951 |
| | Post-SLD | .351 | .321 | .762 | .635 | .657 | .614 | .678 |
| 37 | $U$ | .943 | .943 | .943 | .943 | .943 | .943 | .943 |
| | Pre-SLD | .953 | .912 | .949 | .951 | .959 | .942 | .958 |
| | Post-SLD | .318 | .198 | .705 | .370 | .507 | .346 | .393 |

symmetric, since we are in the binary case, in which $\Pr(y) = \alpha$ entails $\Pr(\overline{y}) = (1 - \alpha)$. The top row of the figure (orange colour) refers to the $U$ class distributions (the left and right histograms are the same); the other histograms in the left column refer to Pre-SLD class distributions, one for each of the 7 learning algorithms, while the other histograms in the right column refer to Post-SLD class distributions for the same algorithms. Figures 4 to 7 do the same for $|\mathcal{Y}| \in \{5, 10, 20, 37\}$; these histograms are, of course, not bilaterally symmetric.[22]

Figure 3 shows that all the methods produce class prevalence values that are distributed more uniformly than the true ones, i.e., many Pre-SLD or Post-SLD distributions generate many class prevalence values with very low or very high values. What is more important, though, is that for each learning method the difference between the $U$ histogram and the Post-SLD histogram is larger than the difference between the $U$ histogram and the Pre-SLD histogram; in other words, this confirms that SLD alters the Pre-SLD class distribution, and that this alteration is detrimental. However, what we learn from these histograms, and that we had not learned from the entropy study of the previous section, is *how* SLD alters this distribution: it does so by generating fewer class priors with mid values, i.e., close to 50%, and more class priors with extreme values, i.e., close to 0% or 1% (to see this better, note that the $Y$ axes of the left subfigure and the right subfigure are often not on the same scale).

It is evident from Figure 3 that SLD's impact in altering the distribution is substantial for each of the four calibrated learners (4th to 8th rows), and it is even more for the non-calibrated ones (2nd to 4th rows). When SLD is run on the posteriors generated by these latter learners, all class priors except 0 and 1 become much more frequent, and class priors equal to 0 and 1 increase dramatically with respect to the Pre-SLD case.

In Figures 4 to 7, which represent the multiclass case with $|\mathcal{Y}| \in \{5, 10, 20, 37\}$, these trends are increasingly evident, and the deterioration introduced by SLD reaches disastrous levels for

[22]Note that, for higher legibility, the $X$ axis displays a shorter interval than [0,1] when there are no classes with prevalence outside that interval.

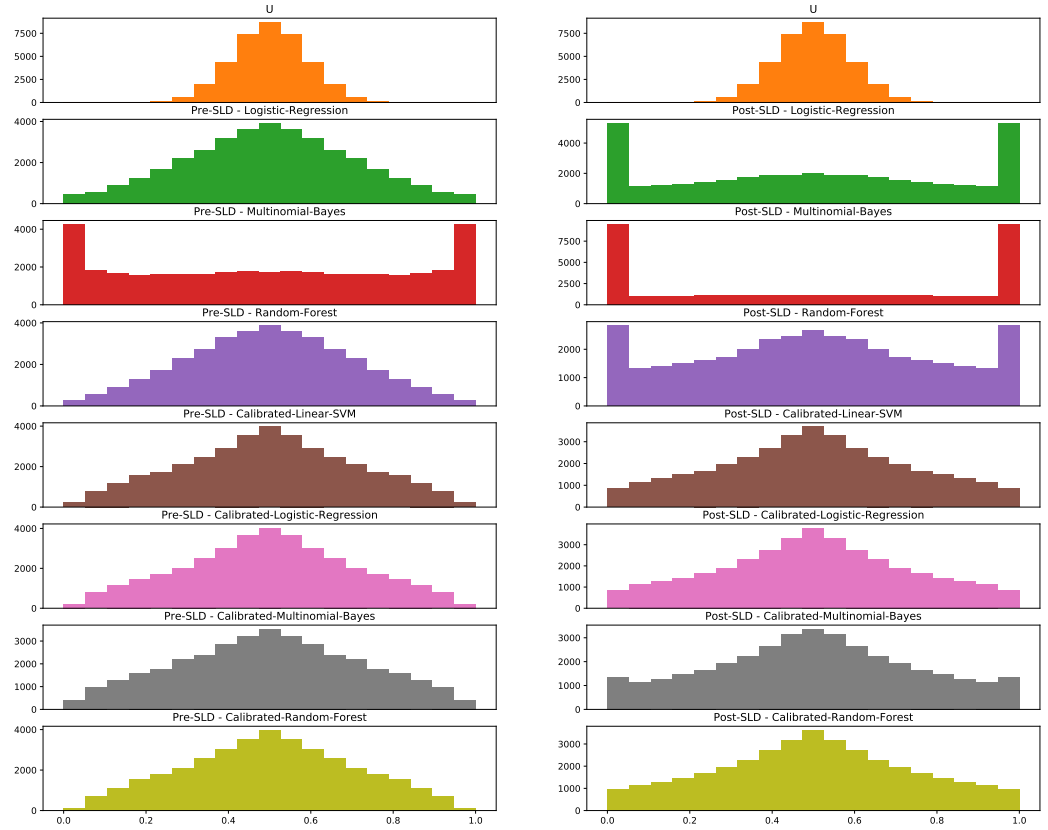


Fig. 3. Histograms showing various distributions of the class priors for $|\mathcal{Y}| = 2$ experiments. The two top subfigures show the true distribution in the unlabelled set $U$; the other subfigures show, for different classifiers, the distribution of predicted class priors before SLD is applied (i.e., as computed on the classifier output) and the distribution of predicted class priors after the application of SLD.

$|\mathcal{Y}| = 37$. Post-SLD average class distribution become increasingly skewed when $|\mathcal{Y}|$ grows, and this concerns both calibrated and non-calibrated learners (although the latter are impacted to a much higher degree). While for the $|\mathcal{Y}| = 2$ case class priors equal to 0 and class priors equal to 1 were both prevalent, in the multiclass cases class priors equal to 1 are practically absent and, as $|\mathcal{Y}|$ grows, the histogram becomes increasingly skewed and class priors equal to 0 become the overwhelming majority.

Overall, what all these histograms show, aligns very well with our experimental findings of Sections 4.1 and 4.2, i.e., with the facts that SLD works better with calibrated than with non-calibrated classifiers, and with the fact that it works better for small values of $|\mathcal{Y}|$ and (much) worse for high values of $|\mathcal{Y}|$. They also show something more, i.e., that the reason of the bad behaviour is the fact that SLD has, especially when $|\mathcal{Y}|$ is high and/or when classifiers are non-calibrated, a tendency to return many class priors equal to 0 and few class priors different from 0.

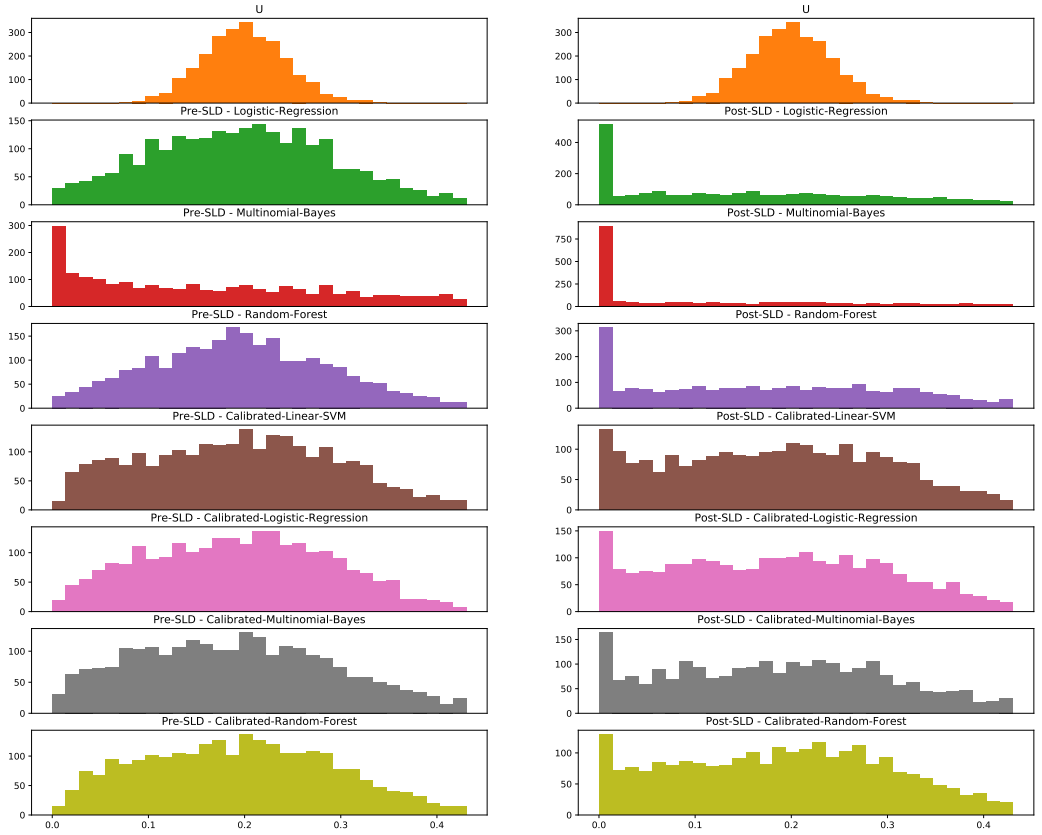


Fig. 4. As in Figure 3 but with $|\mathcal{Y}| = 5$.

### 4.5 On the Speed of Convergence of SLD

As indicated in Section 2, in our experiments we stop SLD when we have reached either convergence (which we take to mean that $\text{AE}(\hat{p}_U^{(s-1)}, \hat{p}_U^{(s)}) < 10^{-6}$) or the maximum number of iterations (that we set to 1000). Table 11 reports, for each learner and for each number $|\mathcal{Y}|$ of classes,

- the average number of iterations (column "#") SLD required to reach convergence (when convergence was actually reached – the value 1000 is used whan convergence was not reached), where the average is computed across all the train-and-test runs we have performed;
- the percentage of cases (column "%") in which convergence was not reached, and processing had to be stopped after 1000 iterations.

There are three conclusions that can be reached from this table, i.e.,

(1) For a given number of classes, convergence tends to be quicker when the Pre-SLD posteriors have been obtained by calibrated learners; this is always true for LR and RF, although it is always false for MNB. The difference between the two versions (non-calibrated and calibrated) of LR is somehow surprising, since LR is often presented as an algorithm that naturally returns calibrated probabilities, i.e., a classifier which does not need post-calibration; our results throughout this paper instead show that post-calibration is beneficial for LR too.

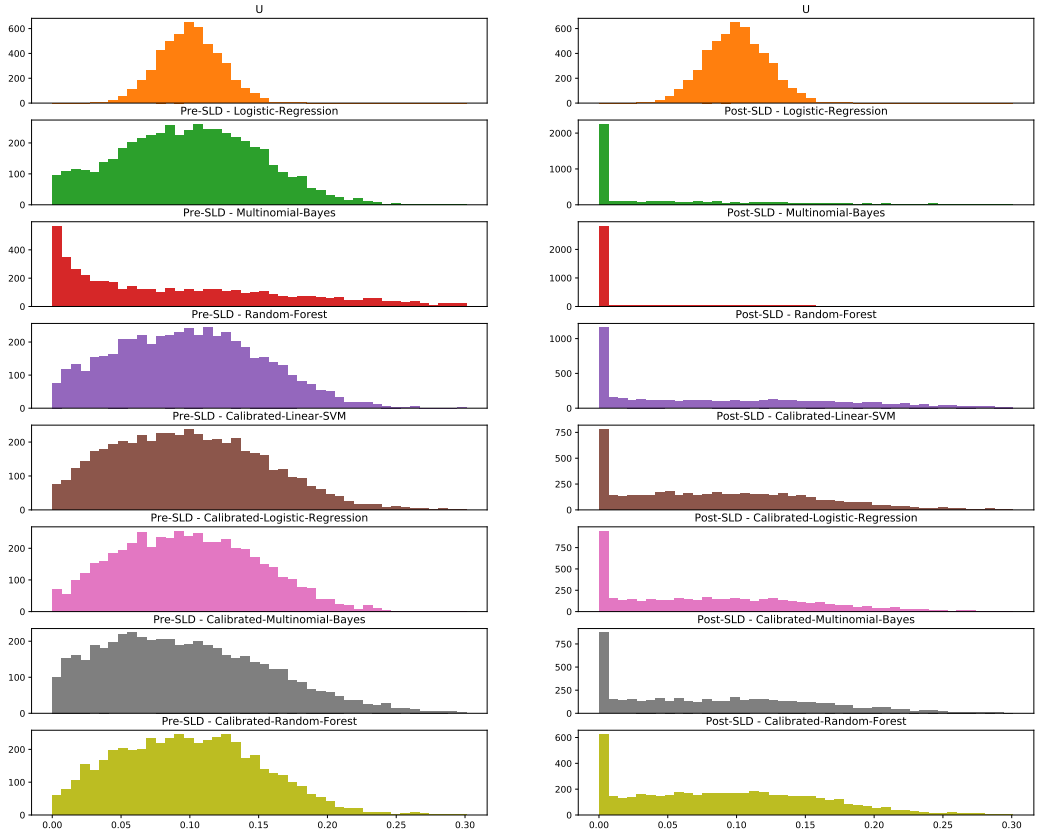


Fig. 5. As in Figure 3 but with $|\mathcal{Y}| = 10$.

Table 11. Average number of iterations needed to reach convergence ("#") and percentage of cases in which convergence was not reached ("%") for all combinations of learner and number $|\mathcal{Y}|$ of classes.

| | | 2 classes | | 5 classes | | 10 classes | | 20 classes | | 37 classes | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | # | % | # | % | # | % | # | % | # | % |
| No-Calib | LR | 60.86 | 0.18% | 138.92 | 0.20% | 476.69 | 12.40% | 939.06 | 77.60% | 999.62 | 99.80% |
| | MNB | 29.27 | 0.00% | 53.74 | 0.00% | 136.46 | 0.60% | 296.92 | 5.80% | 469.28 | 14.00% |
| | RF | 32.38 | 0.01% | 83.65 | 0.20% | 223.70 | 1.40% | 446.88 | 5.80% | 636.25 | 14.80% |
| | Avg | 40.84 | 0.06% | 92.10 | 0.13% | 278.95 | 4.80% | 560.95 | 29.73% | 701.72 | 42.87% |
| Calib | SVM | 9.29 | 0.00% | 29.02 | 0.00% | 109.75 | 0.00% | 288.73 | 0.80% | 353.37 | 1.40% |
| | LR | 9.39 | 0.00% | 27.68 | 0.00% | 95.87 | 0.00% | 249.06 | 1.00% | 368.32 | 4.00% |
| | MNB | 35.69 | 0.82% | 58.95 | 0.40% | 162.36 | 1.00% | 308.40 | 4.20% | 507.15 | 20.60% |
| | RF | 12.66 | 0.00% | 26.85 | 0.00% | 86.00 | 0.00% | 223.57 | 1.20% | 432.19 | 11.00% |
| | Avg | 16.76 | 0.20% | 35.62 | 0.10% | 113.49 | 0.25% | 267.44 | 1.80% | 415.26 | 9.25% |

(2) For a given learner, the number of iterations required to reach convergence grows monotonically with the number $|\mathcal{Y}|$ of classes considered.

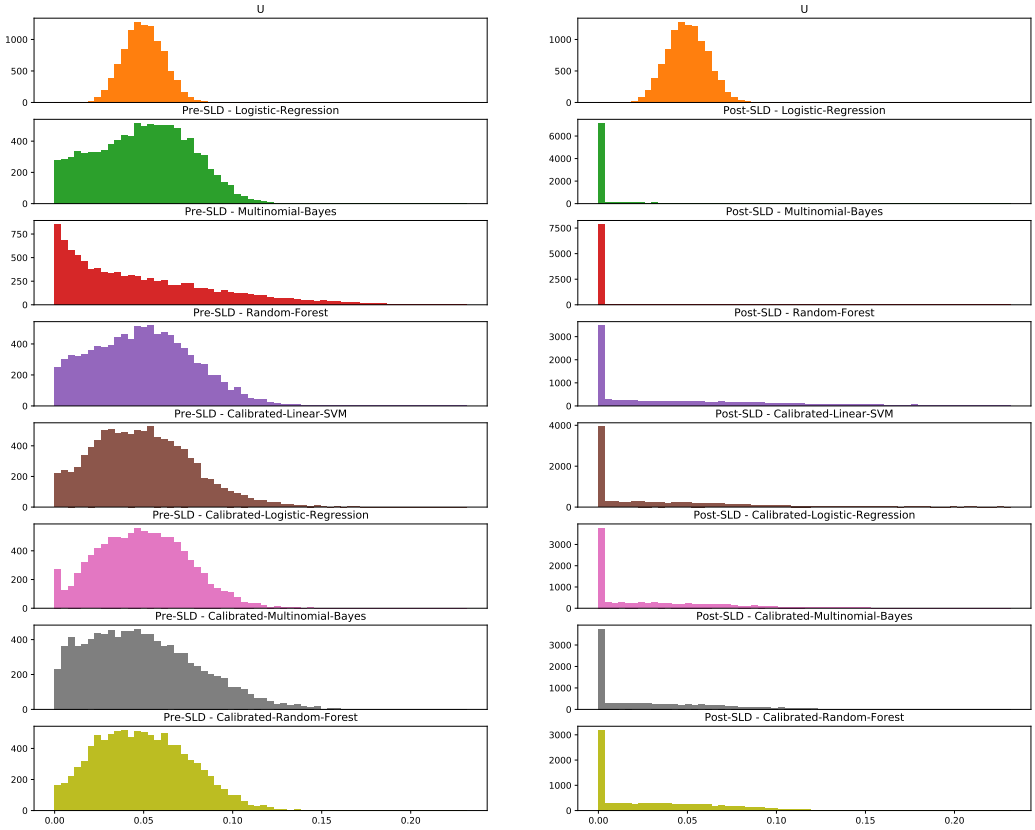


Fig. 6. As in Figure 3 but with $|\mathcal{Y}| = 20$.

(3) For a given learner, the percentage of cases in which convergence is not reached grows monotonically with the number $|\mathcal{Y}|$ of classes considered.

These findings constitute yet another argument in favour of calibrated learners, and yet another reason why the use of SLD should be contemplated only when the number $|\mathcal{Y}|$ of classes is small.

## 5 WHAT KIND OF DISTRIBUTION SHIFT DO WE SIMULATE IN OUR EXPERIMENTS?

In Section 3.2.1 we stated that, in extracting from RCV1-v2 a number of samples with trining sets and test sets characterized by random class distributions, our goal is to test the SLD algorithm on a variety of distribution shift values. But what kind of distribution shift are we simulating, exactly?

In order to distinguish different types of dataset shift (of which distribution shift is a type), Moreno-Torres et al. [24] distinguish (along with Fawcett and Flach [13]) between "$\mathcal{X} \rightarrow \mathcal{Y}$ problems" and "$\mathcal{Y} \rightarrow \mathcal{X}$ problems".

Problems of type $\mathcal{X} \rightarrow \mathcal{Y}$ are ones in which it is the values of the features in $\mathbf{x}$ that stochastically determine the class $y = t(\mathbf{x})$ to which $\mathbf{x}$ belongs. An example of a $\mathcal{X} \rightarrow \mathcal{Y}$ learning problem is weather forecasting, since it is a number of climatic conditions (for instance, pressure, temperature, humidity, etc., that can be represented in a feature vector $\mathbf{x}$) that determine whether it is going to snow or not (a fact that can be represented by a binary dependent variable $y$). In these cases, it is

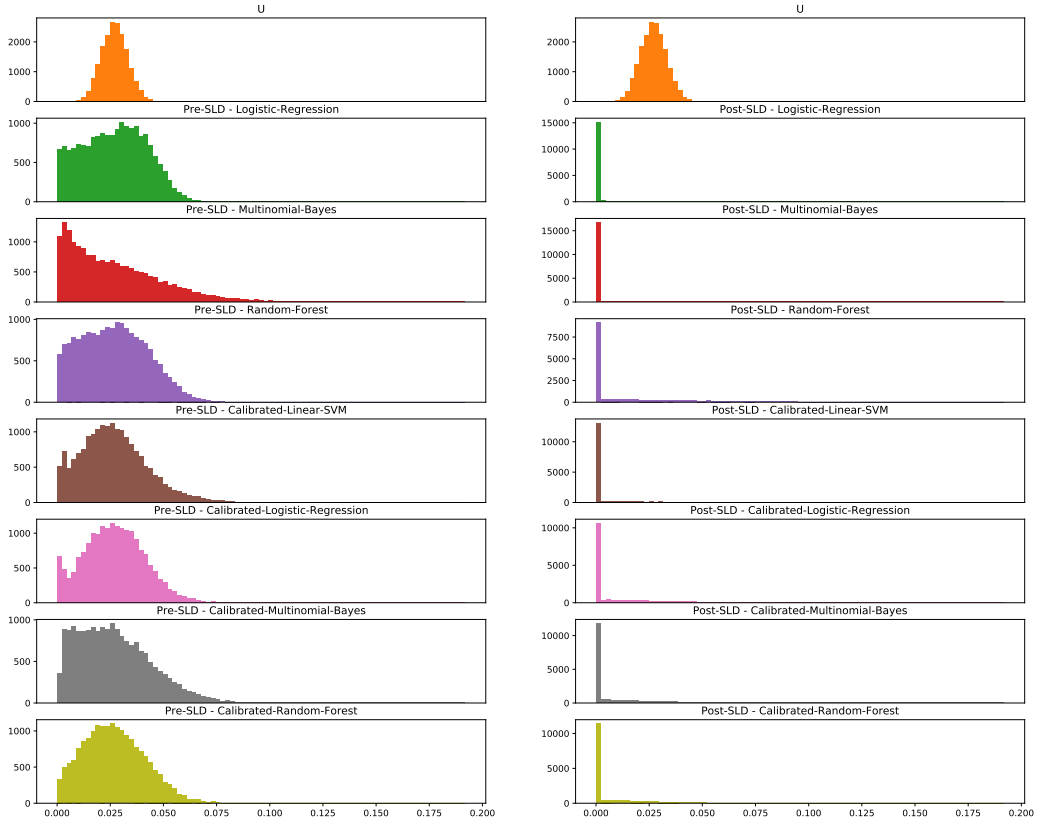


Fig. 7. As in Figure 3 but with $|\mathcal{Y}| = 37$.

useful to write the joint distribution $p(\mathbf{x}, y)$ as

$$p(\mathbf{x}, y) = p(y|\mathbf{x})p(\mathbf{x}) \tag{16}$$

Equation 16 suggests that there are two phenomena (or, of course, a combination of both) that can cause $p(y)$ to vary across $L$ and $U$, i.e.,

(1) *Covariate shift*, defined as the case in which $p_L(y|\mathbf{x}) = p_U(y|\mathbf{x})$ and $p_L(\mathbf{x}) \neq p_U(\mathbf{x})$;
(2) *Concept shift*, defined as the case in which $p_L(y|\mathbf{x}) \neq p_U(y|\mathbf{x})$ and $p_L(\mathbf{x}) = p_U(\mathbf{x})$.

For instance, in the example above, if the distribution of climatic conditions change, the probability that it is going to snow changes too; this is a case of covariate shift. Instead, if the causal relationship between climatic conditions and snowing were to change (an admittedly unlikely case), this would be a case of concept shift.

Problems of type $\mathcal{Y} \rightarrow \mathcal{X}$ are instead ones in which the class $y = t(\mathbf{x})$ to which document $\mathbf{x}$ belongs stochastically determines the values of the features in vector $\mathbf{x}$. An example of a $\mathcal{Y} \rightarrow \mathcal{X}$ learning problem is authorship attribution, i.e., the task of determining the author (from a set of $|\mathcal{Y}|$ candidate authors) of a text of unknown or disputed paternity [20], a task which is usually carried out by using as features a number of "stylistic" traits that tend to characterize an author's writing style. Authorship attribution is an $\mathcal{Y} \rightarrow \mathcal{X}$ problem, since it is the fact that a certain text is, say, Shakespeare's, that causes it to have certain stylistic characteristics, and not the other way

around. In these cases, the joint distribution $p(\mathbf{x}, y)$ can be usefully written as

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y) \tag{17}$$

Here, $p(y)$ can vary for independent reasons (since $y$ is a cause, and not an effect), a phenomenon which is usually called *prior probability shift*. For instance, in Stratford-upon-Avon's municipal library there might proportionally be more books by Shakespeare than in any other municipal library. (Note that $p(\mathbf{x}|y)$ may vary too, but this is not our concern since it would not cause distribution shift anyway.)[23]

So, what kind of distribution shift are we simulating with the sampling strategy of Section 3.2.1, exactly?

If our dataset is from a $\mathcal{X} \rightarrow \mathcal{Y}$ problem, we are certainly simulating covariate shift but *not* concept shift; in fact, we are selectively removing documents (which means that $p(\mathbf{x})$ changes) but we are not making the causal relationship between $\mathcal{X}$ and $\mathcal{Y}$ change (which means that $p(y|\mathbf{x})$ does not change), since the documents that are not removed still have the same class label. Conversely, if our dataset is from a $\mathcal{Y} \rightarrow \mathcal{X}$ problem, we are simulating prior probability shift, because by selectively removing documents we are making $p(y)$ change.

So, what we are simulating with the sampling strategy of Section 3.2.1 is covariate shift and/or prior probability shift, but not concept shift.

There are two reasons for this:

- While a strategy that also simulates concept shift might have been better, since it would have allowed us to test the SLD algorithm in a broader set of challenging situations, it is not clear how concept shift should be simulated, since this would involve changing the class labels of documents that are included in a sample, and it is unclear whether there are sensible policies for doing it.
- SLD was conceived for handling not concept shift but distribution shift; it would thus probably make no sense to simulate situations for which SLD is intentionally unequipped.[24]

## 6 RELATED WORK

Despite having been proposed more than 15 years ago, SLD remains an algorithm unique in its kind, since at the same time it updates the posterior probabilities *and* the class prior probability estimates returned by the classifier.

As discussed in the previous sections (and, especially, in Appendix A), SLD bears strong relations to probability calibration. While several calibration methods have been proposed in the last 20 years (e.g., [1, 3, 6, 27]), none of them actually deals with calibrating the posterior probabilities of the unlabelled set *in the presence of distribution shift*.

As already mentioned throughout the paper, dataset shift (and distribution shift in particular) is central to SLD's concerns. Dataset shift (the word "shift" and "drift" sometimes being used interchangeably) is a multifaceted phenomenon and a largely unexplored territory, and only in the

[23] Note also that it is not always easy to characterize with certainty a given problem as being of type $\mathcal{X} \rightarrow \mathcal{Y}$ or of type $\mathcal{Y} \rightarrow \mathcal{X}$; sometimes this question looks a bit akin to wondering which of chicken and egg came first. As a result, different types of dataset shift (covariate shift, concept shift, prior probability shift) that concur in causing distribution shift may be at play at the same time.

[24] Saerens et al. [34] explicitly assume

> "that the generation of the observations within the classes, and thus the within-class densities, $p(\mathbf{x}|y)$, does not change from the training set to the new data set (only the relative proportion of measurements observed from each class has changed). This is a natural requirement; it supposes that we choose the training set examples only on the basis of the class labels $y$, not on the basis of $\mathbf{x}$."

Our method to generate samples, detailed in Section 3.2.1, is indeed based on choosing the training set examples only on the basis of the class labels.

last ten years or so the machine learning community has started to address it systematically [33]. The task of estimating class prior probabilities in the presence of distribution shift has, since about 2005, evolved as a task of its own, called *quantification* [19], and many algorithms alternative to SLD have been proposed (see [11, 14, 32, 36] for a few recent examples). However, while these algorithms are interesting alternatives to SLD as far as estimating class prior probabilities goes, there are no current alternatives to SLD when it comes to *adjusting the posterior probabilities* in the presence of distribution shift. To the best of our knowledge, the only alternative to SLD that has ever been proposed for adjusting the posterior probabilities in the presence of distribution shift is the algorithm in [38], based on the idea of binning the unlabelled documents based on an invariance property of ROC curves. However, this algorithm *assumes that the true class priors in the unlabelled set are known*; this is an assumption which is not verified in practice (because, in the presence of distribution shift, these class priors are different from the ones in the training set), which means that this algorithm cannot be used in practice.[25]

## 7 CONCLUSIONS

We present a thorough reassessment of SLD, a well-known algorithm that, given a machine-learned single-label classifier and a set of unlabelled documents characterized by distribution shift with respect to the labelled documents the classifier has been trained on, adjusts the posterior probabilities and class prior probability estimates returned by the classifier, in an iterative, mutually recursive way, with the goal of making both more accurate. Since its publication more than 15 years ago, SLD has become the standard algorithm for improving the quality of posterior probabilities, and is still considered a contender when it comes to estimating the class prior probabilities on unlabelled sets. However, its real effectiveness at improving the quality of posterior probabilities has been questioned. Studying SLD is thus not just an academic exercise, and is still important, since no other algorithm for adjusting the posterior probabilities returned by a classifier in the presence of distribution shift is known, and since the quality of posterior probabilities is of key importance for a number of document management tasks, including document ranking and cost-sensitive text classification.

We here present the results of a large scale experimentation that uses multiple learners and a very large, publicly available dataset for text classification, on which multiple amounts of distribution shift (i.e., difference in the distribution of prior probabilities between the training and the unlabelled documents) have been simulated. In total, the experimentation consists of 129,500 train-and-test runs for the binary case and 14,000 such runs for the multiclass case. In these experiments we are especially interested in SLD's ability at improving the quality of posterior probabilities, something which Saerens et al. [34] evaluated only indirectly, i.e., in terms of the accuracy of (cost-insensitive) classification that results from using the posteriors SLD generates.

Our study allows three main conclusions. The first conclusion is that SLD is ineffective, and often detrimental, when the classifier has not been previously calibrated; in this latter case, an additional disadvantage is that the speed of convergence of SLD is slower, and the probability that the computation does not even converge is higher. The second conclusion is that, in any situation, the improvements that SLD brings about are higher (or the deterioration it brings about is lower) when distribution shift is higher. The third conclusion is that the improvements that SLD brings about are higher (or the deterioration it brings about is lower) when the number of classes in the codeframe is small; binary classification is thus the most apt context for the use of SLD, which should instead be used with prudence in multiclass classification with small numbers of classes,

[25] Indeed, the experiments reported in [38] use an oracle that provides to the algorithm the true class priors of the unlabelled set; but this oracle, as all oracles, is not available in practice, so the utility of this algorithm is extremely questionable.

and completely avoided in multiclass classification with high numbers of classes. An additional disadvantage of working with a high number of classes is that, as for non-calibrated classifiers, the speed of convergence of SLD is much slower, and the probability that the computation does not even converge is much higher.[26]

Our results also show that, concerning the improvements in the quality of the posteriors that have been found in the binary case (and, to a lesser extent, in the multiclass case when the codeframe is small), these are due to a reduction of the calibration error, and not to a reduction of the refinement error. This shows that SLD is, in essence, a re-calibration algorithm, i.e., an algorithm for re-calibrating the posterior probabilities of documents belonging to an unlabelled set $U$, where these posteriors have been returned by a classifier already calibrated on a training set $L$ and where the re-calibration is made necessary by the presence of prior probability shift. For this kind of use, and when the number of classes $|\mathcal{Y}|$ is small and the classifiers have been calibrated beforehand, the use of SLD is still recommended.

## ACKNOWLEDGMENTS

The present work has been supported by the ARIADNEplus project, funded by the European Commission (Grant 823914) under the H2020 Programme INFRAIA-2018-1, by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020, and by the SoBigData++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1. The authors' opinions do not necessarily reflect those of the European Commission. We thank Alejandro Moreo for useful discussions on the SLD algorithm.

## REFERENCES

[1] Tuomo Alasalmi, Jaakko Suutala, Heli Koskimäki, and Juha Röning. 2020. Better classifier calibration for small data sets. *ACM Transactions on Knowledge Discovery from Data* 14, 3 (2020), 1–19. https://doi.org/10.1145/3385656

[2] Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2014. Aggregative quantification for regression. *Data Mining and Knowledge Discovery* 28, 2 (2014), 475–518. https://doi.org/10.1007/s10618-013-0308-z

[3] Artem Bequé, Kristof Coussement, Ross W. Gayler, and Stefan Lessmann. 2017. Approaches for credit scorecard calibration: An empirical analysis. *Knowledge-Based Systems* 134 (2017), 213–227. https://doi.org/10.1016/j.knosys.2017.07.034

[4] Glenn W. Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly Weather Review* 78, 1 (1950), 1–3. https://doi.org/10.1175/1520-0493(1950)078<0001:vofeit>2.0.co;2

[5] Gordon V. Cormack. 2008. Email spam filtering: A systematic review. *Foundations and Trends in Information Retrieval* 1, 4 (2008), 335–455. https://doi.org/10.1561/9781601981479

[6] Kristof Coussement and Wouter Buckinx. 2011. A probability-mapping algorithm for calibrating the posterior probabilities: A direct marketing application. *European Journal of Operational Research* 214, 3 (2011), 732–738. https://doi.org/10.1016/j.ejor.2011.05.027

[7] Morris H. DeGroot and Stephen E. Fienberg. 1983. The comparison and evaluation of forecasters. *The Statistician* 32, 1/2 (1983), 12–22. https://doi.org/10.2307/2987588

[8] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B* 39, 1 (1977), 1–38.

[9] Pedro M. Domingos and Michael J. Pazzani. 1996. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*. Bari, IT, 105–112.

[10] Andrea Esuli, Alejandro Moreo, and Fabrizio Sebastiani. 2018. A recurrent neural network for sentiment quantification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM 2018)*. Torino, IT, 1775–1778. https://doi.org/10.1145/3269206.3269287

[11] Andrea Esuli, Alejandro Moreo, and Fabrizio Sebastiani. 2020. Cross-lingual sentiment quantification. *IEEE Intelligent Systems* 35, 3 (2020), 106–114. https://doi.org/10.1109/MIS.2020.2979203

[12] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27 (2006), 861–874.

[26] Note that our results do not contradict the original results of Saerens et al. [34], since these authors, while presenting SLD as a general-purpose multiclass algorithms, only run (900) *binary* classification experiments.

[13] Tom Fawcett and Peter Flach. 2005. A response to Webb and Ting's 'On the application of ROC analysis to predict classification performance under varying class distributions'. *Machine Learning* 58, 1 (2005), 33–38. https://doi.org/10.1007/s10994-005-5256-4

[14] Afonso Fernandes Vaz, Rafael Izbicki, and Rafael Bassi Stern. 2019. Quantification under prior probability shift: The ratio estimator and its extensions. *Journal of Machine Learning Research* 20 (2019), 79:1–79:33.

[15] Peter A. Flach. 2017. Classifier Calibration. In *Encyclopedia of Machine Learning* (2nd ed.), Claude Sammut and Geoffrey I. Webb (Eds.). Springer, Heidelberg, DE, 212–219.

[16] George Forman. 2008. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery* 17, 2 (2008), 164–206. https://doi.org/10.1007/s10618-008-0097-y

[17] Wei Gao and Fabrizio Sebastiani. 2016. From classification to quantification in tweet sentiment analysis. *Social Network Analysis and Mining* 6, 19 (2016), 1–22. https://doi.org/10.1007/s13278-016-0327-z

[18] Tilmann Gneiting and Adrian E. Raftery. 2007. Strictly proper scoring rules, prediction, and estimation. *J. Amer. Statist. Assoc.* 102, 477 (2007), 359–378. https://doi.org/10.1198/016214506000001437

[19] Pablo González, Alberto Castaño, Nitesh V. Chawla, and Juan José del Coz. 2017. A review on quantification learning. *Comput. Surveys* 50, 5 (2017), 74:1–74:40. https://doi.org/10.1145/3117807

[20] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for Information Science and Technology* 60, 1 (2009), 9–26. https://doi.org/10.1002/asi.20961

[21] David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1994)*. Dublin, IE, 3–12. https://doi.org/10.1007/978-1-4471-2099-5_1

[22] Alessio Molinari. 2019. Leveraging the transductive nature of e-discovery in cost-sensitive technology-assisted review. In *Proceedings of the 8th BCS-IRSG Symposium on Future Directions in Information Access (FDIA 2019)*. Milano, IT, 72–78.

[23] Alessio Molinari. 2019. *Risk minimization models for technology-assisted review and their application to e-discovery*. Master's thesis. Department of Computer Science, University of Pisa, Pisa, IT.

[24] Jose G. Moreno-Torres, Troy Raeder, Rocío Alaíz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. 2012. A unifying view on dataset shift in classification. *Pattern Recognition* 45, 1 (2012), 521–530. https://doi.org/10.1016/j.patcog.2011.06.019

[25] Alejandro Moreo and Fabrizio Sebastiani. 2020. Tweet sentiment quantification: An experimental re-evaluation. Submitted for publication.

[26] Allan H. Murphy. 1973. A new vector partition of the probability score. *Journal of Applied Meteorology* 12, 4 (1973), 595—-600.

[27] Mahdi P. Naeini, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*. Austin, US, 2901–2907.

[28] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Obtaining calibrated probabilities from boosting. In *Proceedings of the 21st Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI 2005)*. Arlington, US, 413–420.

[29] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*. Bonn, DE, 625–632. https://doi.org/10.1145/1102351.1102430

[30] Douglas W. Oard, Fabrizio Sebastiani, and Jyothi K. Vinjumur. 2018. Jointly minimizing the expected costs of review for responsiveness and privilege in e-discovery. *ACM Transactions on Information Systems* 37, 1, Article 11 (2018), 11:1–11:35 pages. https://doi.org/10.1145/3268928

[31] John C. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, Alexander Smola, Peter Bartlett, Bernard Schölkopf, and Dale Schuurmans (Eds.). The MIT Press, Cambridge, MA, 61–74.

[32] Pablo Pérez-Gállego, Alberto Castaño, José Ramón Quevedo, and Juan José del Coz. 2019. Dynamic ensemble selection for quantification tasks. *Information Fusion* 45 (2019), 1–15. https://doi.org/10.1016/j.inffus.2018.01.001

[33] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence (Eds.). 2009. *Dataset shift in machine learning*. The MIT Press, Cambridge, US. https://doi.org/10.7551/mitpress/9780262170055.001.0001

[34] Marco Saerens, Patrice Latinne, and Christine Decaestecker. 2002. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation* 14, 1 (2002), 21–41. https://doi.org/10.1162/089976602753284446

[35] Fabrizio Sebastiani. 2020. Evaluation measures for quantification: An axiomatic approach. *Information Retrieval Journal* 23, 3 (2020), 255–288. https://doi.org/10.1007/s10791-019-09363-y

[36] David Spence, Christopher Inskip, Novi Quadrianto, and David Weir. 2019. Quantification under class-conditional dataset shift. In *Proceedings of the 11th International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2019)*. Vancouver, CA, 528–529. https://doi.org/10.1145/3341161.3342948

[37] D. B. Stephenson, C. A. S. Coelho, and I. T. Jolliffe. 2008. Two extra components in the Brier score decomposition. *Weather and Forecasting* 23, 4 (2008), 752–757. https://doi.org/10.1175/2007WAF2006116.1

[38] Meesun Sun and Sungzoon Cho. 2018. Obtaining calibrated probability using ROC binning. *Pattern Analysis and Applications* 21, 2 (2018), 307–322. https://doi.org/10.1007/s10044-016-0578-3
[39] Vladimir Vapnik. 1998. *Statistical learning theory.* Wiley, New York, US.
[40] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. 2004. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research* 5 (2004), 975–1005.
[41] Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2002).* Edmonton, CA, 694–699. https://doi.org/10.1145/775107.775151

## A SLD'S GOAL IS TO ENFORCE THE MUTUAL CONSISTENCY OF THE POSTERIORS AND THE PRIORS OF $U$

We here show that SLD may be viewed as an attempt to enforce a necessary condition for the posteriors $\Pr(y_j|\mathbf{x}_i)$ of the documents $\mathbf{x}_i \in U$ to be calibrated. In order to show this, let us define

- $U_a$ to be the set of documents $\mathbf{x}_i \in U$ such that $\Pr(y_j|\mathbf{x}_i) = a$;
- $U^j$ to be the set of documents $\mathbf{x}_i \in U$ such that $\mathbf{x}_i \in y_j$;
- $U_a^j$ to be the set of documents $\mathbf{x}_i \in U_a \cap U^j$.

Recall from Section 1 that the posteriors $\Pr(y_j|\mathbf{x}_i)$, with $\mathbf{x}_i \in U$, are perfectly calibrated when, for all $a \in [0, 1]$, it holds that $\frac{|U_a^j|}{|U_a|} = a$. If so, then it holds that

$$\begin{aligned} |U_a^j| &= |U_a| \cdot a \\ &= \sum_{\mathbf{x}_i \in U_a} a \\ &= \sum_{\mathbf{x}_i \in U_a} \Pr(y_j|\mathbf{x}_i) \end{aligned} \tag{18}$$

Since $U$ is finite, there is a finite set $A$ of values that the posteriors of the documents in $U$ take. From Equation 18 it follows that

$$\sum_{a \in A} |U_a^j| = \sum_{a \in A} \sum_{\mathbf{x}_i \in U_a} \Pr(y_j|\mathbf{x}_i) \tag{19}$$

which can be rewritten as

$$|U^j| = \sum_{\mathbf{x}_i \in U} \Pr(y_j|\mathbf{x}_i) \tag{20}$$

By multiplying both sides by $\frac{1}{|U|}$ we obtain

$$\Pr_U(y_j) = \frac{1}{|U|} \sum_{\mathbf{x}_i \in U} \Pr(y_j|\mathbf{x}_i) \tag{21}$$

which is exactly the condition on the "mutual consistency" of priors and posteriors that SLD tries to enforce (see Equation 5 and Step 11 of Algorithm 1), and that holds after SLD has converged.

In sum, for the posteriors $\Pr(y_j|\mathbf{x}_i)$ of the documents $\mathbf{x}_i \in U$ to be calibrated, Equation 21 must hold. While SLD is not a full-fledged attempt to calibrate the posteriors in $U$ (which would be impossible, since we do not know the label of any document in $U$), it may nevertheless be seen as a step in that direction.

Received September 2020; accepted November 2020