Estimating Precisions for Multiple Binary Classifiers Under Limited Samples

Rahul Tripathi(✉), Srinivasan Jagannathan, and Balaji Dhamodharaswamy

Amazon, Seattle, USA
{rahtripa,sjaganna,dhbalaji}@amazon.com

Abstract. Machine learning classifiers often require regular tracking of performance measures such as precision, recall, F1-score, *etc.*, for model improvement and diagnostics. The population over which accuracy metrics are evaluated can be too large for a full ground-truth assessment and so only small random samples are chosen for estimation. Ground-truthing often requires human review, which is expensive. Moreover, in some business applications, it may be preferable to minimize human contact with the data in order to improve privacy safeguards. Thus, sampling methods that can provide estimates with low margin of error, high confidence, and small sample size are highly desirable. With an ensemble of multiple binary classifiers, choosing the right sampling method with these desired properties and small size for the collective sample becomes even more important. We propose a sampling method to estimate the precisions of multiple binary classifiers that exploits the overlaps between their prediction sets. We provide theoretical guarantees that our estimators are unbiased and empirically demonstrate that the precision metrics estimated from our sampling technique are as good (in terms of variance and confidence interval) as those obtained from a uniform random sample.

We applied our sampling technique to performance evaluation of an ensemble of binary classifiers. The reduction in sample size depends on the extent of overlap between the predicted positive set of the ensemble and that of the individual classifiers. Since we do not have a closed form solution for quantifying the impact of the overlap, we relied on simulations to investigate how the overlap between an ensemble (parent) and component (child) classifier affects the overall sample size. We found that for every combination of parent and child intersection ratio we tested on, there were significant savings in sample size. Moreover, across all these simulations, we found a mean reduction of 33% in the sample size needed from a child. Our simulations also confirm that the precision metrics estimated from the samples generated using our sampling technique have accuracy comparable to those estimated from uniform random sampling.

Keywords: model precision · crowd-sourcing · sampling.

1 Introduction

Machine learning (ML) models rely on the assumption that the target data distribution is close (in statistical sense) to the training data distribution. While the latter is static when the models are being developed and trained, in many applications, the target data distribution may vary over time due to the dynamic nature of production workloads that are classified by the models. In order to continuously evaluate ML models, accuracy metrics such as *precision* and *recall* need to be measured on a regular basis. For a binary classifier that classifies any instance into either $\mathcal{P}$ (positives) or $\mathcal{N}$ (negatives), precision is defined as the fraction of instances predicted as positive that are in fact positive whereas recall is defined as the fraction of positive instances that are correctly predicted as positive. More precisely, if TP, FP, FN denotes True-Positive, False-Positive, and False-Negative instances respectively based on the classification decisions by the model, then precision = TP/(TP + FP) and recall = TP/(TP + FN).

One of the main bottlenecks in tracking these model performance metrics is the need for labeling of the target data used in the evaluation. The label assignment process, called *annotation* or *ground-truthing*, in ML applications is often done manually, which is not scalable. In particular, the cost of annotating a dataset increases significantly with the size of the dataset. Additionally, in some applications, it is preferable to minimize the exposure of data to manual reviewers, for example, to improve privacy safeguards and increase security.

Quite often, an ML application is composed of an ensemble of multiple classifiers. As a result, for model performance diagnostics and tracking, it becomes important to evaluate accuracy metrics of not only the ensemble but also each individual ML classifiers. Therefore, a challenging problem is how to estimate the performance metrics (e.g., precision) of multiple (binary) classifiers with low error, high confidence, and minimal ground truth cost. In this paper, we focus only on the precision performance metric, however, our techniques can be generalized to other measures.

There are two main approaches to estimating the precision of a classifier: *simple random sampling* and *stratified sampling*. These sampling approaches select a small, but statistically relevant, number of instances, called a *sample* from the underlying population (i.e., the predicted positive set of a classifier). Based on the ground-truth assignment of labels to instances in the sample, the precision is estimated using the formula, discussed earlier, but applied to the sample instead of the population.

In simple random sampling, one chooses a uniformly random sample from the population. The main parameter here is the sample size, which as explained in Section 3.2, depends on the desired level of accuracy and confidence. Simple random sampling is quite effective in that it yields an unbiased estimator for the precision. However, it can result in a larger sample size than possible with a stratified sampling.

Stratified sampling divides the population into k disjoint strata or bins, for some fixed k. It requires two important considerations: (a) stratification method - how the the bins/strata are constructed and (b) allocation method - how the sample size is split across all the bins. It is expected that stratification results in near homogenous bins, i.e., bins containing high concentration of instances with same ground-truth labels, and therefore it lowers the variance of the precision within each bin. By giving different weights to bins and taking a weighted average of the precision estimation from each bin, we can get an unbiased estimator for the precision of the classifier. Also, if the variance in each bin is low, the resulting estimator will also have low variance over the population.

In this work, we use the observation that if a random sample for one classifier overlaps with the prediction set of another, then we can reuse the common instances so that only a smaller sample size is needed for the other classifier. Large-scale production systems often consist of multiple binary classifiers whose individual predictions contribute to the final decision of an ensemble composed of individual classifiers. This observation is particularly useful in such systems since the classifiers are expected to have overlaps in their prediction sets. We give theoretical justification and share experimental findings to show that the new sampling scheme, based on this observation, reaches the same accuracy at a significantly reduced sample size.

We describe our algorithms for estimating the precisions of multiple binary classifiers in Section 4. We address the case of an ensemble model and its constituent binary classifiers (Section 4.1). We present both theoretical and experimental results to demonstrate that our solution achieves the desired objectives: low error, high confidence, and low ground truth sample size compared to the baseline (Sections 4.1 and 5). Generalization of our method to other accuracy metrics (e.g., recall) is explained in Section 6. Finally, we conclude with a summary of the main results (Section 7).

2 Related Work

Bennett et al. [1] adapts stratified sampling techniques to present an online sampling algorithm to evaluate the precision of a classifier. They experimentally demonstrate that their algorithm achieves an average reduction of 20% in sample size compared to simple random sampling and other types of stratified sampling to get the same level of accuracy and confidence. Similarly, Kumar [6] proposes strategies based on stratified sampling to estimate the accuracy of a classifier. They also experimentally show that their methods are more precise compared to simple random sampling for accuracy estimation under constrained annotation resources. In Kataria et al. [4], an iterative stratified sampling strategy is presented that continuously learns a stratification strategy and provides improved accuracy estimates as more labeled data is available. However, for more than one classifier, it is unclear whether these stratified sampling based methods applied individually to the constituent classifiers would give a similar saving on the size of the collective sample set. Our proposed sampling algorithm relies on uniform random sampling and achieves significant saving (e.g., on average $\approx 33\%$ and $\approx 85\%$ average reduction in sample size for any individual classifier in two different experimental settings) compared to the baseline of simple random sampling when applied individually on multiple classifiers for estimating their precisions.

For multiple classifiers, unsupervised methods for estimating classifier accuracies, ranking them, and constructing a more accurate ensemble classifier based solely on classifier outputs over a large unlabeled test data are presented in [3, 7, 8]. However, these methods rely on assumptions such as conditional independence of classifiers or certain constraints on classifier errors, which limits their practical applicability in many situations. Our work makes no assumption regarding the classifiers.

3 Preliminaries

3.1 Notation

We consider binary classifiers that map instances from some universe Ω to either *positives* ($\mathcal{P}$) or *negatives* ($\mathcal{N}$) label. The *predicted positive set* (*predicted negative set*) of a classifier is the set of all instances that it maps to $\mathcal{P}$ (resp., $\mathcal{N}$). Let sequence $S \subseteq \Omega$ be an ordered multi-set in Ω and denote its length by $|S|$, which includes duplicity. A subsequence of a sequence S contains a subset of elements and preserves their ordering in S. The notation $A - B$ denotes the set difference between any two sets A and B in Ω. For any sequence S and set A, we denote $S \cap A$ to denote the subsequence of S that contains all and only those elements that are in A. If S and T are sequences, then $S{+}T$ denotes the sequence obtained by appending T to S to the right of S. If a is any instance and S is a sequence, then $\mathrm{count}(a, S)$ denotes the number of occurrences of a in S.

3.2 Sample Size to Estimate Precision

The precision of a classifier C for positives $\mathcal{P}$ can be estimated by uniformly sampling instances from the predicted positive set of C. Given a sample with sufficient number of such instances, we can label each instance to determine the number of True-Positives (TPs) and False-Positives (FPs) in the collection. A point estimate $\hat{p}$ for the precision p is: $\hat{p} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$.

To determine a $(1 - \delta)$-confidence-interval with $\pm\epsilon$ additive margin of error, the sample size needed is given by $\epsilon \geq z_{1-\delta} \times s/\sqrt{n}$, where s is the standard deviation of each random instance, n is the number of samples, and for any $0 < \alpha < 1$, z_α is the α'th quantile[1] of the standard normal distribution. Since a sample instance being in $\mathcal{P}$ is a Bernoulli trial with success probability equal to precision p, its variance is $s^2 = p(1-p)$. Plugging into the earlier equation gives

$$\epsilon \geq z_{1-\delta} \times \sqrt{\frac{p(1-p)}{n}}. \qquad (1)$$

Thus, for $\epsilon = 0.03$, $\delta = 0.05$, and the maximum variance assumption ($p = 1/2$), the sample size estimate is 1068. If one is willing to make stronger assumptions, e.g. precision is guaranteed to be at least some threshold p_0, then the sample size estimate can be considerably reduced (e.g., 385 if $p_0 \geq 90\%$ and 278 if $p_0 \geq 93\%$). We denote the sample size needed to estimate precision within $\pm\epsilon$ additive error and $1 - \delta$ confidence by $n_{\epsilon,\delta}$.

[1] z_α is a factor such that a normal r.v. $N(\mu, \sigma)$ lies inside the interval $\mu \pm z_\alpha \sigma$ with probability α.

4 Optimized Precision Estimation By Recycling Samples

Given a collection of binary classifiers, estimating the precision for each one requires generating samples and assigning a label to each instance. The label assignment (*annotation*), is typically a manual, laborious process whose cost is proportional to the size of a sample. The baseline approach to estimate the precisions of a collection of k binary classifiers requires labeling individual sample sets for each of the classifiers. Anchoring on one of the classifiers (called *parent* in the follow up discussion), we consider the remaining classifiers as its children. Any classifier whose predicted positive set is presumed to have significant overlap with those of the remaining classifiers is a good choice for the parent. For example, an ML system may be composed of multiple binary classifiers with an ensemble of them as the authoritative classifier. In this setting, the ensemble could be considered a parent classifier because we expect the predicted positive set of the ensemble to overlap with that of each individual classifier. In this section, we explain how samples from a parent classifier can be recycled to generate subsamples of each child classifier, and thereby reduce the combined sample size.

4.1 Classifiers With Overlapping Predicted Positive Sets

Suppose we have a parent classifier P with predicted positive set, denoted A_P, and a sample S_P generated from A_P. Given a child classifier C with predicted positive set, denoted A_C, we exploit the overlap between A_P and A_C to generate a sample S_C using S_P. We show the the sample S_C retains the statistical property needed for an unbiased estimator of the precision of C, provided S_P possessed the same. Thus, it results in a smaller sample size to estimate the precisions of both C and P compared to the baseline. We demonstrate empirically that our estimates are within the desired margin of error and acceptable confidence.

Algorithm 1, called RecycleSamplesForPrecision, takes the input (a) the predicted positive sets A_P and A_C, (b) the size n_C of the sample needed to estimate the precision of C, (c) the sample S_P, and (d) an option UniformSample or UniformShuffle. It generates a sample S_C and estimates precision $\hat{p}_C$ for C. In Line 1, S^+ equals the subsequence (with repetitions) of all elements in S_P that belong to A_C. In Line 2, a call to a function UniformSample is made, which generates a uniform sample with replacement from a population. The function takes three arguments (a) the population to sample from, (b) the size of the sample, (c) and whether (or not) to sample with replacement. The subsequence S^- is a uniformly generated from $A_C - A_P$ and its size is required to satisfy: $|S^-|/|S^+| = |A_C - A_P|/|A_P \cap A_C|$. This is needed because any uniformly generated sample S from A_C of size $|S^-| + |S^+|$ is expected to contain instances from the disjoint sets $A_P \cap A_C$ and $A_C - A_P$ in proportion to their sizes. S^{remain} in Line 3 includes uniformly generated instances from A_C to backfill any shortage from just S^+ and S^- combined.

Any single instance in $S^+ + S^-$ is not uniformly distributed over A_C. To see this, if e_1 is the first instance and e_l is the last instance in this subsequence, then e_1 is likely to come from S^+ and e_l from S^-. Therefore, in such a case, e_1's distribution is over $A_P \cap A_C$ whereas e_l's over $A_C - A_P$, and so they are not uniform over A_C. The function MixSequence, defined in Algorithm 2, ensures that $S^+ + S^-$ is a uniformly random sample from A_C, and so the estimator $\hat{p}_C$ in Line 8 is an unbiased estimator of the precision of C.

```
Algorithm 1: RecycleSamplesForPrecision
  Data: Classifiers $C$ and $P$ with predicted positives sets $A_C$ and $A_P$,
        respectively; sample size $n_C$ for $C$; a sequence $S_P$ of uniformly random
        instances from $A_P$; and a parameter
        option $\in$ [UniformSample, UniformShuffle].
  Result: Estimated precision $\hat{p}_C$ of $C$ and a sequence $S_C$ of uniformly random
          instances from $A_C$ of size $n_C$.
1 $S^+ \leftarrow S_P \cap A_C$ .
2 $S^- \leftarrow$ UniformSample($A_C - A_P$, $|A_C - A_P| \times |S^+|/|A_P \cap A_C|$, replace=True)
3 $S^{\text{remain}} \leftarrow$ UniformSample($A_C$, $\max(0, n_C - (|S^+| + |S^-|)$, replace=True)
4 $S_C \leftarrow \text{MixSequence}(S^+ + S^-, \text{option}) + S^{\text{remain}}$
5 if $|S_C| > n_C$ then
6     $S_C \leftarrow S_C[0 : n_C]$
7 end
8 $\hat{p}_C \leftarrow$ fraction of positives instances in $S_C$
9 return $\hat{p}_C$, $S_C$
```

Figure 1 shows the child-parent relationship and the sets involved in generating the final sample S_C. We transform the sequence $S^+ + S^-$ into a sequence S of uniformly random instances in Algorithm 2. Two possible ways are considered: (a) uniform sampling and (b) uniform shuffling. The former is nothing but sampling with replacement and the latter is without replacement. In both options, we show that the new sequence consists of uniformly random instances from A_C, and so the average $\hat{p}_C$ is unbiased. We distinguish between these two options because only uniform sampling would result in total independence of instances. This observation is useful in situations where C itself is an ensemble of other classifiers. In such a situation, we may want to start with a uniformly *independent* sample S_C of C (i.e., sample with replacement), then recursively apply Algorithm 2 by treating C as a parent classifier and its constituent classifiers as children of C.

In Lemma 1, we show that with uniform sampling as the option in the function MixSequence, the original sequence $S^+ + S^-$ is transformed into a sequence composed of independent and uniformly distributed instances from A_C.

Lemma 1. *Let* $S =_{df} \text{MixSequence}(S^+ + S^-, \text{option} = \text{UniformSample})$ *in Algorithm 1. Then* S *is a sequence of i.i.d. uniformly random instances from* A_C.

Proof. Let s_i denote the i'th random instance in S. Since each s_i is chosen uniformly from $S^+ + S^-$ with replacement, s_i's are independent and identically

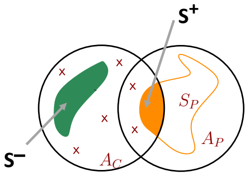


Fig. 1. A parent classifier P overlapping with a child classifier C on their predicted positive sets A_P and A_C, respectively. The right circle represents A_P and the left A_C. A random sample S_P, shown as a closed curve, intersects with A_C as shown in the shaded region. This shaded region represents S^+, the shaded closed curve in $A_C - A_P$ represents S^-, and the sprinkled tick-marks represent S^{remain} in the description of Algorithm 1.

Algorithm 2: MixSequence

```
Data: A sequence $S$ and a parameter
      option $\in$ [UniformSample, UniformShuffle].
Result: Rearranged sequence $S$, where option determines the rearrangement
        method.
if option *equals* UniformSample then
    $S \leftarrow$ UniformSample($S$, $|S|$, replace=True)
else if option *equals* UniformShuffle then
    $S \leftarrow$ UniformShuffle($S$, $|S|$)
return $S$
```

distributed. We now show that, for any $a \in A_C$ and any i, it holds that $\Pr[s_i = a] = 1/|A_C|$. Fix some $a \in A_P \cap A_C$. Let $S^+ + S^- =_{df} Y_1, Y_2, \ldots, Y_{|S|}$. Then

$$\begin{aligned}
\Pr[s_i = a] &= \sum_{\ell > 0} \Pr[s_i = a \mid |S^+| = \ell] \times \Pr[|S^+| = \ell] \\
&= \sum_{\ell > 0} \sum_{k=1}^{|S|} \frac{1}{|S|} \Pr[Y_k = a \mid |S^+| = \ell] \times \Pr[|S^+| = \ell] \\
&= \sum_{\ell > 0} \frac{|A_P \cap A_C|}{|A_C|\ell} \sum_{k=1}^{|S^+|} \Pr[Y_k = a \mid |S^+| = \ell] \times \Pr[|S^+| = \ell] \\
&= \frac{|A_P \cap A_C|}{|A_C|} \sum_{\ell > 0} \frac{1}{\ell} \times \sum_{k=1}^{\ell} \frac{1}{|A_P \cap A_C|} \times \Pr[|S^+| = \ell] = \frac{1}{|A_C|}.
\end{aligned}$$

Here, the second equality uses the fact that s_i equals Y_k (for any k) with probability $1/|S|$. The third equality uses $|S| = |S^+| + |S^-| = \frac{|A_C| \times |S^+|}{|A_P \cap A_C|}$, $|S^+| = \ell$, and the fact that since $a \in A_P \cap A_C$, the terms are zero for $k \in [|S^+| + 1, |S|]$. The fourth equality uses the fact that each element of $A_P \cap A_C$ is equally likely to be the k'th element of S^+, therefore $\Pr[Y_k = a \mid |S^+| = \ell]$ is equal to $\frac{1}{|A_P \cap A_C|}$. Additionally, $\sum_{k=1}^{\ell} \Pr[|S^+| = \ell]$ equals $l \times \Pr[|S^+| = \ell]$.

For the the case where $a \in A_C - A_P$, the analysis is analogous, with minor differences. Instead of k varying over $[1, |S^+|]$ in the third and the fourth equality, we now have k vary over $[|S^+| + 1, |S|]$, and the term $\frac{1}{|A_P \cap A_C|}$ is replaced by $\frac{1}{|A_C - A_P|}$ inside the second summation in the fourth equality. □

In Lemma 2, we show that with uniform shuffling as the option in the function MixSequence, the original sequence $S^+ + S^-$ is transformed into a sequence composed of uniformly distributed instances from A_C, but the instances are not independent.

Both Lemmas 1 and 2 appear identical in that they generate a uniform random sample. However, the main distinction is in the *independence* of the resulting sequence: uniform sampling results in an independent sequence whereas uniform shuffle in Lemma 2 does not imply independence. Nevertheless, we explain below that Lemma 2 gives rise to a stratified sampling procedure. We empirically show that the mean, the std dev., and the 95% confidence interval of precision errors are comparable to that of the simple random sample (see Table 1 and Figures 2(b) and 2(c)).

Lemma 2. *Let* $S =_{df} \text{MixSequence}(S^+ + S^-, \text{option} = \text{UniformShuffle})$ *in Algorithm 1. Then* S *is a sequence of identically and uniformly distributed instances from* A_C*.*

Proof. Let s_i denote the i'th random instance in S. Fix an element a of $A_P \cap A_C$. (A similar argument will apply if $a \in A_C - A_P$.) We will show that, for any $1 \leq i \leq |S|$, $\Pr[s_i = a] = 1/|A_C|$, and so the lemma would follow. By the law of total probability,

$$\begin{aligned}
\Pr[s_i = a] &= \sum_{S^+, S^-} \Pr[s_i = a \mid S^+, S^-] \times \Pr[S^+, S^-] \\
&= \sum_{S^+, S^-} \frac{\text{count}(a, S^+ + S^-)}{|S|} \times \Pr[S^+, S^-] \\
&= \sum_{S^+, S^-} \frac{|A_P \cap A_C|}{|A_C|} \times \frac{\text{count}(a, S^+)}{|S^+|} \times \Pr[S^+, S^-] \\
&= \frac{|A_P \cap A_C|}{|A_C|} \sum_{S^+} \frac{\text{count}(a, S^+)}{|S^+|} \times \sum_{S^-} \Pr[S^+, S^-] \\
&= \frac{|A_P \cap A_C|}{|A_C|} \sum_{S^+} \frac{\text{count}(a, S^+)}{|S^+|} \times \Pr[S^+] \\
&= \frac{|A_P \cap A_C|}{|A_C|} E[X_a], \qquad (2)
\end{aligned}$$

where X_k, for any $k \in A_P \cap A_C$, is a random variable that equals the fraction of times k occurs in S^+ when each element in S (and so S^+) is chosen uniformly at random. Here, the second equality follows since S is a uniformly random shuffle of $S^+ + S^-$. In the third equality, we use the fact that $a \in A_P \cap A_C$ implies $a \in S^+$, and so $\text{count}(a, S^+ + S^-) = \text{count}(a, S^+)$. We also use $|S| = \frac{|A_C| \times |S^+|}{|A_P \cap A_C|}$

there. Note that $\sum_{k \in A_P \cap A_C} X_k = 1$ and, by symmetry, $E[X_k] = E[X_{k'}]$ for any $k, k' \in A_P \cap A_C$. Hence, by the linearity of expectation, $E[X_a] = 1/|A_P \cap A_C|$. It follows from Eq. (2) that $\Pr[s_i = a] = 1/|A_C|$. □

Lemma 2 shows that the resulting sequence is composed of uniformly distributed instances over A_C. This shows that the estimator $\hat{p}_C$ in Algorithm 1 with option = UniformShuffle is unbiased. Note that Algorithm 1 with option = UniformShuffle is just a special case of *stratified sampling* with *proportional allocation* [2] involving the two strata $A_C - A_P$ and $A_C \cap A_P$. This is true because we maintained the ratio of $|S^+|$ to $|S^-|$ as that of $|A_C \cap A_P|$ to $|A_C - A_P|$. Since $A_C \cap A_P$ is expected to be more homogeneous than A_C and likewise for $A_C - A_P$, the stratification should reduce the variance of $\hat{p}_C$.

Lemma 3 expresses the amount of saving in sample size in terms of various probability events. As evident from Lemma 3, the saving in the sample size depends on the extent of the overlap of $A_P \cap A_C$ relative to A_P and to A_C. We refer to the ratios $|A_P \cap A_C|/|A_P|$ as PIR (parent intersection ratio) and $|A_P \cap A_C|/|A_C|$ as CIR (child intersection ratio).

Lemma 3. *Let* $S =_{df}$ MixSequence$(S^+ + S^-, \text{option})$, $n_P =_{df} |S_P|$, *and* $X =_{df} |S^+|$ *in Algorithm 1. Let* Savings *denotes the number of sample instances saved by Algorithm 1 relative to the baseline (simple random sampling) of sample size* n_C*. Then the following statements hold:*

(a) $X \mid n_P \sim B(n_P, \frac{|A_P \cap A_C|}{|A_P|})$.

(b) Savings $= X$ *when* $\frac{X}{n_C} \leq \frac{|A_P \cap A_C|}{|A_C|}$.

(c) Savings $= |S[0 : n_C] \cap S^+|$ *when* $\frac{X}{n_C} > \frac{|A_P \cap A_C|}{|A_C|}$.

Here, $B(n, p)$ *denotes the binomial distribution with parameters* n *and* p.

Proof. Part (a) follows because each instance in S^+ arises because of the successful Bernoulli trial of choosing an element in $A_P \cap A_C$ uniformly and independently from A_P. Hence, the distribution of $X = |S^+|$ given n_P is binomial with number of trials n_P and success probability $\frac{|A_P \cap A_C|}{|A_P|}$. For Part (b), we note that if $\frac{X}{n_C} \leq \frac{|A_P \cap A_C|}{|A_C|}$, then $|S| = |S^+| + |S^-| \leq n_C$. Hence, in this case, we can reuse all of S^+ and so Savings equals X. The condition in Part (c) implies that $|S| > n_C$ and so S^{remain} would equal the empty set. Therefore, any saving we get would be due to only those instances in S^+ that also occur in the first n_C instances in S. □

Using Lemma 3, we can describe the distribution of savings as a function of PIR, CIR, n_P, and n_C.

5 Experiments and Results

5.1 Metrics for comparison

We consider the below metrics for comparing Algorithm 1 against simple random sampling. Our simulations involve multiple trials in which each trial requires a distinct seed for randomly selecting parameters of the simulation. For trial i,

- %**Savings:** this is the percentage savings in the sample size achieved by our algorithm against a simple random sample. Formally, if $s_{i,a}$ denotes the sample size required by our algorithm and $s_{i,r}$ denotes the sample size required by a simple random sample in trial i, then this equals $\frac{s_{i,r}-s_{i,a}}{s_{i,r}} \times 100$.
- %**PrecisionError:** this is the percentage absolute deviation of the point estimate from actual precision. Formally, if $\hat{p}_i$ and p_i denote the estimated and the actual precisions in trial i, respectively, then this equals $\frac{|\hat{p}_i-p_i|}{p_i} \times 100$.
- %**MaxCIError:** this is percentage width of the 95% confidence interval relative to the actual precision. Formally, if $[l_i, u_i]$ denotes the 95% confidence interval and p_i is the actual precision in trial i, then this equals $\frac{\max\{|l_i-p_i|,|u_i-p_i|\}}{p_i} \times 100$.

5.2 Simulations

We compare the sample sizes required by Algorithm 1 against that of simple random samples for an ensemble of three classifiers whose properties and performance are randomly chosen. We consider a majority vote ensemble model (denoted MVE) as parent of three models (denoted ML1, ML2, ML3), which are its children. We compare three different sampling algorithms for precisions: (1) **Simple Random Sample (SRS):** each of MVE, ML1, ML2, and ML3 requires a separate sample of size 1100. As noted in Section 3.2, 1100 samples are sufficient to estimate precision within ± 0.03 additive error and 95% confidence. The collective sample size in this case is at most 4400 and can be lower if sample instances repeat across the collective samples. (2) **RecycleSampleForPrecision(RSFP)-Shuffle** This is Algorithm 1 with *option*=UniformShuffle. Here, MVE requires 1100 samples, but ML1, ML2, and ML3 have reduced sampling requirements because of overlap between predicted positive sets of MVE and ML1, MVE and ML2, and MVE and ML3. (3) **RecycleSampleForPrecision(RSFP)-Sample:** This is Algorithm 1 with *option*=UniformSample.

We choose the parameters of our simulation as follows: (1) *population size of positives*: set to one million; (2) *number of trials*: 200 (a separate random seed is used in each trial); (3) *class ratio of positives to the size of the entire population*: randomly chosen from $[0.01, 0.5]$ range; (4) *precision of model ML1*: randomly chosen from $[0.70, 1.0]$ range; (5) *recall of model ML1*: randomly chosen from $[0.1, 1.0]$ range; and (6) *precisions of models ML2 and ML3*: for each, randomly chosen between 0.5 and that of ML1.

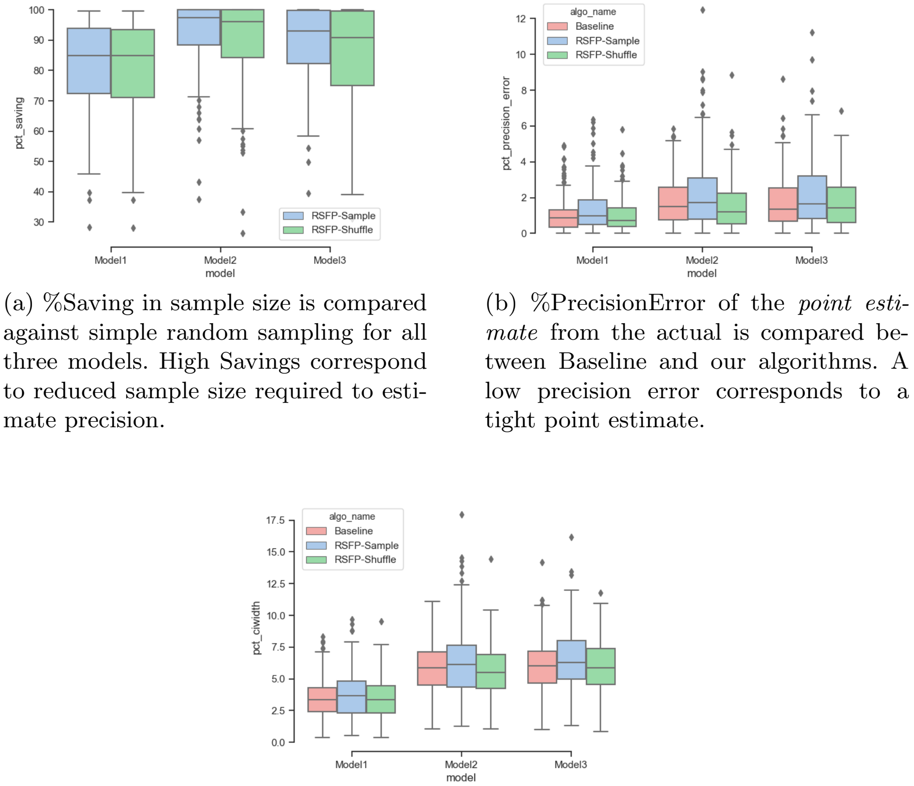
These random choices range over almost all permissible values of these parameters. Thus, our simulations were designed to cover arbitrary model performance characteristics, and empirically illustrate the validity of our algorithms. The data within each trial is generated independently as follows. First, the models ML1, ML2, and ML3 are simulated by independently assigning each one scores between 0 and 1 using a truncated exponential distribution with shape parameter 0.5. In order to de-correlate the scores across models, we divide the scores into blocks of size 0.03 and randomly shuffle the scores of ML2 and ML3 within each block. Next, for any fixed choice of (a) the class ratio of positives to the population size, (b) precisions of the models, and (c) their recalls, we determine the right decision threshold for each of the models so that their predicted positive sets satisfy the precision and recall constraints. Next, fixing one of the models, say ML1, we uniformly assign true-positive labels over the predicted positive set and randomly assign the remaining false-negative labels over the predicted negative set of ML1. Here, we considered different variations of random assignment of false-negative labels: uniform selection and exponentially decaying selection. Once the scores and the thresholds of ML1, ML2, and ML3 are determined, MVE is also uniquely defined. Together with the label assignments, we use the data within each trial to estimate precisions using these sampling algorithms.

Figure 2 reports the metrics for comparison between our algorithms (RFSP-Sample and RFSP-Shuffle) and the SRS (baseline) algorithm in the simulation. In all trials and for each model, we compare %Savings in sample size, %PrecisionError of the point estimate from actual precision, and %MaxCIError of the 95% confidence interval relative to the actual precision. All three algorithms use a simple random sample of size 1100 for MVE. Thereafter, the baseline algorithm draws independent random samples of size 1100 for each of three ML models whereas RFSP-Sample and RFSP-Shuffle require the same sample size but different option setting. Since sampling for MVE is done similarly in all three algorithms, we report comparison results only for ML1, ML2, and ML3.

Figure 2(a) shows that both RSFP-Sample and RSFP-Shuffle can lead to significant %Savings in sample size compared to SRS. The savings is $> 85\%$ in at least half of the trials. The higher savings occur because of large overlap in the predicted positives sets of the majority ensemble with each of ML1, ML2, and ML3 in various trials. The overlap is possibly because the model scores were somewhat positively correlated with each other during the simulation. The simple random sample for any one model, say ML1, may also save on sample size because of possible overlap with the random sample for MVE. However, if the size of predicted positive sets of both ensemble and ML1 are extremely large compared to the the sample size (1100), the overlap is generally low.

In Figure 2(b), we can see that %PrecisionError of all the algorithms are all close to each other for each of the models (ML1, ML2, and ML3) and that RSFP-Shuffle and baseline show slightly lower error than RSFP-Sample. For example, the median, quartliles, and the inter-quartlile ranges for all three models are lower for both RSFP-Shuffle and baseline than for RSFP-Sample. This shows that both RSFP-Shuffle and SRS allow to produce an equally tight point estimate of the precision with RSFP-Sample a little behind these two in accuracy. The extreme outlier instance for Model 2 corresponds to a trial run in which the actual precision was 0.539 and the estimated precision was 0.472, and so the %PrecisionError turned out to be 12.43%.

In Figure 2(c), we notice that %MaxCIError of each of the algorithms are again close to each other. This shows that the confidence intervals produced from RSFP-Sample and RSFP-Shuffle are almost as narrow as those from the simple random sampling. The extreme outlier for ML2 is for the case when the actual precision was 0.539 and the estimated confidence interval from RSFP-Sample was $[0.442, 0.501]$, and so %MaxCIError turned out to be 17.92%.



(a) %Saving in sample size is compared against simple random sampling for all three models. High Savings correspond to reduced sample size required to estimate precision.

(b) %PrecisionError of the *point estimate* from the actual is compared between Baseline and our algorithms. A low precision error corresponds to a tight point estimate.

(c) %MaxCIError in the width of the *confidence interval* relative to the actual is compared between Baseline and our algorithms. A low width corresponds to narrow confidence interval around the actual.

Fig. 2. The boxplots are based on 200 trials with random selection of parameters of the experiment. (a) Both RSFP-Sample and RSFP-Shuffle show significant savings in sample size compared to simple random sampling for all the models. The median %Savings reach above 85% in all cases because of high overlap between predictive positives sets. Notice also that %Savings is low (< 40%) in certain trials in which this overlap is small. (b) The %PrecisionError of all three algorithms are low (e.g., the third quartile is around 3% or less), which suggests that the *point estimates* from RSFP-Sample and RSFP-Shuffle are tight. The outliers correspond to those trials where the estimate deviates too far from actual, which is possible in up to 5% of trials. (c) The %MaxCIError of RSFP-Sample and RSFP-Shuffle are also close to the Baseline for all three models, which suggests that the samples generated from RSFP-Sample and RSFP-Shuffle produce as narrow *confidence interval* as the simple random sampling. The outliers correspond to the trials in which one of the (upper or lower) limits of the generated confidence interval deviates too far from the actual, which is possible in a small % of trials.

5.3 Savings in sample size as a function of PIR and CIR

We ran a simulation of Algorithm 1 to evaluate the amount of savings for different PIR and CIR values. Here, we present results for *option*=UniformShuffle because, as reported in Figure 2 (see Section 5.2), the random shuffle provides more accurate precision estimates. Specifically, we ran 200 trials each for 361 combinations of PIR and CIR (values in the range $[0.05, 0.95]$ in increments of 0.05). In each trial of the simulations we randomly chose $|A_P \cap A_C|$ in the range $[10K,\ 100K]$ and performed a random selection of samples in S_P. In Table 1, we report the percentage saving in sample sizes (mean as well as 95% confidence interval) for a representative subset of PIR and CIR values we used in the simulations. We found a mean savings of 33.68%, and that in 95% of the simulation runs, the savings varied between 4.43% and 82.98%. In Fig 3, we present a surface plot of the mean savings in the simulation runs as a function of PIR and CIR. Our simulations confirm that significant amount of savings is achieved as the amount of overlap between parent and child classifiers increases. Furthermore, the error in precision due to our sampling method is extremely low (both mean and std. dev. of %PrecisionError $\leq 1.51\%$) across all choices considered.

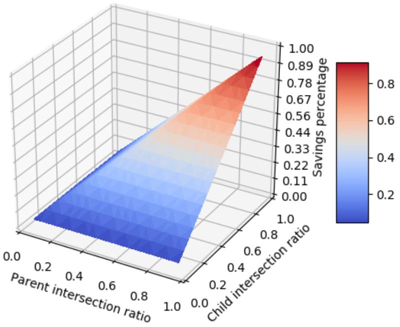


Fig. 3. A surface plot of %Savings as a function of parent intersection ratio (PIR) and child intersection ratio (CIR). Notice that %Savings increase as both PIR and CIR increase with %Savings can reach as high as 90%.

5.4 Practical Application of Algorithm 1

We applied our sampling algorithm to evaluate precision of binary classifiers for the offensive content detection problem, studied in [5]. We start with the labeled dataset considered in that work, take random subsets of it to create a train set with 4.5M texts and a distinct test set with 2M texts. The ratio of positives and negatives is kept 1:1 in both train and test sets. We implemented three binary

Table 1. Percentage saving in sample size as a function of Parent-Intersection-Ratio and Child-Intersection-Ratio is shown. For each combination of PIR and CIR below, we report over 200 trials the mean and the 95% confidence interval of %Saving in sample size and the mean and the std. dev. of %PrecisionError when applying Algorithm 1. As seen below, the savings increase with increasing values for PIR and CIR, i.e., with the amount of overlap between parent and child classifiers. Also, the mean, the std. dev, and the 95% confidence interval of %precision errors from our algorithm are 1.367%, 1.269%, and $[0.0379, 4.697]$%, resp., which match closely with those from *simple random sampling* that are 1.368%, 1.267%, and $[0.0385, 4.678]$%, resp.

PIR	CIR	mean%	2.5^{th}%	97.5^{th}%	mean%	std. dev%
		Savings	Savings	Savings	precision error	precision error
0.05	0.05	4.72	3.64	5.38	1.50	1.46
0.05	0.25	4.91	3.39	6.42	1.39	1.27
0.05	0.45	4.95	3.23	6.54	1.36	1.31
0.05	0.65	4.92	3.06	6.58	1.43	1.31
0.05	0.85	4.91	2.79	6.70	1.30	1.16
0.25	0.05	4.93	3.65	6.11	1.51	1.33
0.25	0.25	24.21	20.60	25.57	1.45	1.32
0.25	0.45	24.73	19.73	27.79	1.20	1.12
0.25	0.65	24.61	18.70	27.73	1.44	1.36
0.25	0.85	24.61	18.70	27.75	1.26	1.28
0.45	0.05	4.89	3.55	6.29	1.37	1.31
0.45	0.25	24.58	20.63	26.72	1.45	1.28
0.45	0.45	43.85	36.15	46.03	1.31	1.17
0.45	0.65	44.45	33.75	48.56	1.46	1.28
0.45	0.85	44.47	32.82	48.74	1.37	1.42
0.65	0.05	4.83	3.28	6.20	1.29	1.11
0.65	0.25	24.34	18.44	26.88	1.34	1.19
0.65	0.45	44.17	33.15	47.19	1.50	1.50
0.65	0.65	63.68	48.02	67.07	1.37	1.16
0.65	0.85	64.39	49.58	69.21	1.41	1.39
0.85	0.05	4.82	3.19	6.20	1.50	1.43
0.85	0.25	24.29	18.98	26.95	1.43	1.40
0.85	0.45	44.00	32.28	47.36	1.39	1.34
0.85	0.65	64.14	48.59	67.95	1.33	1.24
0.85	0.85	84.04	55.49	88.83	1.30	1.32

classifiers, namely Bi-LSTM, CNN, and LogReg, described in [9], and trained them on our train dataset. We also created a majority vote ensemble (MVE) of them. As in Section 5.3, we present results for *option*=UniformShuffle only.

We fed the Reddit test dataset as input to the classifiers discussed above, and applied Algorithm 1 on the predicted positive sets to sample for precision. We ran 1000 trials to generate samples and calculated %PrecisionError and %Savings from the samples in each trial. In Table 2, we observe that our algorithm can save > 88% in the number of samples needed to estimate precision, while obtaining very low errors in the precision estimates derived from the smaller sample sizes. Note that there is no saving for MVE since in this case our algorithm defaults to a simple random sample. Moreover, the percentage precision errors of our algorithm closely matches that of simple random sample for each model.

Table 2. The sizes of predicted positive sets, and the min/max of %PrecisionError and %Savings over 1000 trials for each of the component models in MVE (majority vote ensemble) are reported here. Note that there are no metrics for MVE alone, as we would use simple random sample for MVE—the parent classifier.

Model	predicted	min/max %	min/max %	min/max %
	positive	precision error	precision error	savings
	set size	(Algorithm 1)	(Simple Random Sample)	
BiLSTM	776653	0.03/0.25 %	0.03/0.34 %	88.96/94.73 %
CNN	848576	0.04/0.50 %	0.04/0.41 %	97.52/97.90 %
LogReg	884272	0.01/0.56 %	0.01/0.65 %	90.22/91.72 %

6 Generalizing to Other Performance Measures

The focus of this paper has been on the precision metric. However, as previously stated, our approach can be generalized to other metrics such as recall. In this section, we illustrate how we can generalize to recall calculations.

Suppose C_1 and C_2 are classifiers with their precisions p_1 and p_2, recalls r_1 and r_2, predicted positive sets A_1 and A_2, and predicted negative sets B_1 and B_2, respectively. Then the number of true-positives for C_1 and C_2 are given by $p_1|A_1|$ and $p_2|A_2|$, respectively. Since the total size of positives in the population (say N) is fixed and independent of the classifiers, it follows that $\frac{r_1}{r_2} = \frac{p_1|A_1|/N}{p_2|A_2|/N} = \frac{p_1|A_1|}{p_2|A_2|}$. Generally, sizes of A_1, A_2, B_1, and B_2 are known. Therefore, if tight estimates $\hat{p}_1$ on p_1, $\hat{p}_2$ on p_2, and $\hat{r}_1$ on r_1 are known, then we can obtain a tight estimate $\hat{r}_2$ on r_2. Also, since recall equals TP/(TP + FN) and false omission rate (FOR) equals FN/(TN + FN), we get recall $r_1 = \frac{p_1|A_1|}{p_1|A_1|+f_1|B_1|}$ where f_1 is the FOR of C_1. It follows that the recall estimation problem for multiple classifiers is reducible to obtaining tight estimates on their precisions and a tight estimate on the FOR of a *single* classifier, say C_1.

Sampling for estimating the FOR and for estimating the precision of a classifier C_1 are over disjoint populations B_1 and A_1, respectively. So, applying Algorithm 1 to estimate FOR (by treating B_1 and A_1 as child-parent relationship) will not result in reduced sample size. The overlap between B_1 (predicted negative set of C_1) and predicted positive sets of other classifiers are also expected to be weak. Therefore, it is unlikely that Algorithm 1 will help. In this case, estimating FOR for classifier C_1 under limited annotations should ideally be done using stratified sampling approaches suggested in [1, 6, 4]. In other words, a combination of Algorithm 1 for estimating the precisions of multiple classifiers and stratified sampling method for estimating the FOR of a *single* classifier would suffice to estimate the recalls of multiple classifiers to achieve an overall reduction in number of samples.

7 Conclusion

We presented a sampling algorithm RecycleSamplesForPrecision to estimate precisions of multiple binary classifiers with minimal sample size. Our algorithm makes use of two properties: (a) the predicted positive sets of classifiers quite often have significant overlaps and (b) if a random sample for estimating precision of one classifier overlaps with the predicted positive set of another classifier, then we can reuse the common instances to reduce the sample size. We showed that our algorithm results in uniformly distributed random samples. We ran experiments with an ensemble of three classifiers (with randomly assigned accuracy metrics) and observed (in Fig 2) that, for each individual classifier in the ensemble, (a) the mean %savings is $> 80\%$ and (b) the distribution of %PrecisionError and %MaxCIError are all close to the baseline (simple random sample). In particular, our algorithm with option=UniformShuffle gives a slightly tighter estimate compared to option=UniformSample. Next, focusing only on RecycleSamplesForPrecision with option=UniformShuffle, we ran experiments over a wide range of possible ratios of intersections for parent and child classifiers, and observed consistent savings in samples sizes across all these scenarios, where the amount of savings increases with the amount of intersection ratios. Over all the runs of this experiment (see Table 1), we observe (a) a mean %savings of $\approx 33\%$ and (b) the mean, the std. dev., and the 95% confidence interval of %PrecisionError are 1.367%, 1.269%, and $[0.0379, 4.697]\%$, respectively, which are comparable to those of simple random sampling.

References

1. Bennett, P.N., Carvalho, V.R.: Online stratified sampling: evaluating classifiers at web-scale. In: Proceedings of the 19th ACM CIKM. pp. 1581–1584. ACM (2010)
2. Cochran, W.G.: Sampling techniques. John Wiley & Sons (2007)
3. Jaffe, A., Nadler, B., Kluger, Y.: Estimating the accuracies of multiple classifiers without labeled data. In: Artificial Intelligence and Statistics. pp. 407–415 (2015)
4. Katariya, N., Iyer, A., Sarawagi, S.: Active evaluation of classifiers on large datasets. In: IEEE 12th International Conference on Data Mining. pp. 329–338. IEEE (2012)
5. Khatri, C., Hedayatnia, B., Goel, R., Venkatesh, A., Gabriel, R., Mandal, A.: Detecting offensive content in open-domain conversations using two stage semi-supervision. CoRR **abs/1811.12900** (2018)
6. Kumar, A., Raj, B.: Classifier risk estimation under limited labeling resources. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 3–15. Springer (2018)
7. Parisi, F., Strino, F., Nadler, B., Kluger, Y.: Ranking and combining multiple predictors without labeled data. Proceedings of the National Academy of Sciences **111**(4), 1253–1258 (2014)
8. Platanios, E.A., Blum, A., Mitchell, T.: Estimating accuracy from unlabeled data. In: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI). pp. 682–691 (2015)
9. Tripathi, R., Dhamodharaswamy, B., Jagannathan, S., Nandi, A.: Detecting sensitive content in spoken languages. In: Proceedings of the 6th IEEE International Conference on Data Science and Advanced Analytics (DSAA) (2019)