Positive Unlabeled Learning Selected Not At Random (PULSNAR): class proportion estimation when the SCAR assumption does not hold

Anonymous authors
Paper under double-blind review

Abstract

Positive and Unlabeled (PU) learning is a type of semi-supervised binary classification where the machine learning algorithm differentiates between a set of positive instances (labeled) and a set of both positive and negative instances (unlabeled). PU learning has broad applications in settings where confirmed negatives are unavailable or difficult to obtain, and there is value in discovering positives among the unlabeled (e.g., viable drugs among untested compounds). Most PU learning algorithms make the *selected completely at random* (SCAR) assumption, namely that positives are selected independently of their features. However, in many real-world applications, such as healthcare, positives are not SCAR (e.g., severe cases are more likely to be diagnosed), leading to a poor estimate of the proportion, α, of positives among unlabeled examples and poor model calibration, resulting in an uncertain decision threshold for selecting positives. PU learning algorithms can estimate α or the probability of an individual unlabeled instance being positive or both. We propose two PU learning algorithms to estimate α, calculate calibrated probabilities for PU instances, and improve classification metrics: i) PULSCAR (positive unlabeled learning selected completely at random), and ii) PULSNAR (positive unlabeled learning selected not at random). PULSNAR uses a divide-and-conquer approach that creates and solves several SCAR-like sub-problems using PULSCAR. In our experiments, PULSNAR outperformed state-of-the-art approaches on both synthetic and real-world benchmark datasets.

1 Introduction

In a standard binary supervised classification problem, the classifier (e.g., decision trees, logistic regression, support vector machines, etc.) is given training instances with features x and their labels y=0 (negative) or y=1 (positive). The classifier learns a model $f : x \rightarrow y$, which can classify an unlabeled instance as positive or negative based on its features. It is often challenging, expensive, and even impossible to annotate large datasets in real-world applications Jaskie et al. (2019), and frequently only positive instances are labeled. Unlabeled instances with their features can be classified via positive and unlabeled (PU) learning Jaskie et al. (2019); Elkan & Noto (2008). Some of the PU learning literature focuses on improving classification metrics, and others focus on the problem of estimating the fraction, α, of positives among the unlabeled instances. Although this work focuses on the latter, calibration and enhancing classification performance are also addressed.

PU learning problems abound in many domains Jaskie & Spanias (2019). For example, in electronic healthcare records, diagnosis codes can establish patients have been evaluated and labeled positive for a given disease, yet the absence of a diagnosis code does not establish a patient is negative for a disease, and such patients are elements of an unlabeled mix of positives and negatives. Importantly, confirmed negatives are generally not codable, and thus traditional supervised learning is infeasible. Much medical literature is dedicated to estimating disease incidence and prevalence but contends with incomplete medical assessment and recording. The potential to assess the incidence of a given disease (without costly in-person assessment or chart reviews) in large unlabeled populations could have substantial public health benefits. In market research, one typically

has a modest set of positives, say of customers or buyers of a product, has a set of attributes over both the positives and a large population of unlabeled people of size N, and wishes to establish the size of the addressable market, αN.

The majority of PU learning algorithms use the *selected completely at random* (SCAR) assumption, which states that the labeled positive examples are randomly selected from the universe of positives. That is, the labeling probability of any positive instance is constant Elkan & Noto (2008). This assumption may fail in real-world applications. For example, in email spam detection, positive instances labeled from an earlier time period could differ from later spam due to adaptive adversaries.

Although some PU learning algorithms have shown promising performance on different machine learning (ML) benchmark SCAR datasets, to our knowledge, the development of PU learning algorithms to estimate the extent of undercoding in large and highly imbalanced *selected not at random* (SNAR) real-world data remains an active research area. Class imbalance in a PU setting generally means the number of unlabeled instances is large compared to the labeled positive examples. Also, current PU learning approaches have rarely explored how to calculate well-calibrated probabilities for PU examples in SCAR and SNAR settings. In addition, few PU algorithms have been assessed when α is small ($\leq 5\%$), where performance is expected to suffer.

In this paper, we propose a PU learning approach to estimate α when positives are SCAR or SNAR, and compare our estimation in simulated and real data versus other algorithms. We assess the performance with class imbalance in both modest and large datasets and over a rigorous α range. Our contributions are summarized as follows:

1. We propose PULSCAR, a PU learning algorithm for estimating α when the SCAR assumption holds. It uses kernel density estimates of the positive and unlabeled distributions of ML probabilities to estimate α. Innovations include:
   (a) Using the beta distribution for density estimation where the $[0 \ldots 1]$ support of the beta function matches that of the probability distributions generated from ML classifiers Chen (1999). A bandwidth heuristic is introduced for the beta distribution, with good empirical performance for α estimation.
   (b) Introducing an objective function whose derivative maximum provides a rapid, robust estimate of α.
2. We propose PULSNAR, a PU learning algorithm for estimating α when the positives are SNAR, that uses a novel clustering approach to divide the positives into several subsets that can have separate α estimates versus the unlabeled. These sub-problems are more SCAR-like and are solved with PULSCAR.
3. We propose a method to calibrate the probabilities of PU examples to their true (unknown) labels in SCAR and SNAR settings.
4. We propose a method to improve the classification performance in SCAR and SNAR settings.
5. We simulate PU learning data with large deviations from the SCAR assumption, and generate SCAR and SNAR problems from fully-labeled ML repository data.
6. We release an open-source Python package for our PULSNAR and PULSCAR algorithms that fully integrates with scikit-learn, enabling multiple state-of-the-art ML algorithms (e.g., XGBoost, catboost, etc.).

2 Related work

Early PU learning Yu et al. (2004); Yu (2005); Wang et al. (2006) generally followed a two-step heuristic: i) identify strong negative examples from the unlabeled set, and then ii) apply an ML algorithm to given positive and identified negative examples. Some recent work iteratively identifies better negatives Luo et al. (2021), or combines negative-unlabeled learning with unlabeled-unlabeled learning Hammoudeh & Lowd (2020). Instead of extracting only strong negative examples from the unlabeled set, Fung et al. (2005) extracted high-quality

positive and negative examples from the unlabeled set and then applied classifiers to those data. Lee & Liu (2003); Liu et al. (2008) assigned weights to the unlabeled examples to train a classifier. Elkan & Noto (2008) introduced the SCAR assumption. By partially matching the class-conditional density of the positive class to the input density under Pearson divergence, Du Plessis & Sugiyama (2014) estimated the class prior. Jain et al. (2016) proposed a nonparametric class prior estimation technique, AlphaMax, using two-component mixture models. The kernel embedding approaches KM1 and KM2 Ramaswamy et al. (2016) showed that the algorithm for mixture proportion estimation converges to the true prior under certain assumptions. Estimating the class prior through decision tree induction Bekker & Davis (2018) provides a lower bound for label frequency under the SCAR assumption. DEDPUL Ivanov (2020) assumes SCAR and uses probability densities to estimate α with a compute-intensive EM-algorithm; the method does not produce calibrated probabilities. Confident learning (CL) Northcutt et al. (2021) combines the principle of pruning noisy data, probabilistic thresholds to estimate noise, and sample ranking. Multi-Positive and Unlabeled Learning Xu et al. (2017) extends PU learning to multi-class labels. Oversampling the minority class Chawla et al. (2002); Yan et al. (2019) or undersampling the majority class are not well-suited approaches for PU data due to contamination in the unlabeled set; Su et al. (2021) uses a re-weighting strategy for imbalanced PU learning. Recent studies have focused on labeling/selection bias to address the SCAR assumption not holding. Bekker et al. (2019); Gerych et al. (2022) used propensity scores to address labeling bias and improve classification. Using the propensity score, based on a subset of features, as the labeling probability for positive examples, Bekker et al. (2019) reduced the Selected At Random (SAR) problem into the SCAR problem to learn a classification model in the PU setting. The "Labeling Bias Estimation" approach was proposed by Gong et al. (2021) to label the data by establishing the relationship among the feature variables, ground-truth labels, and labeling conditions.

3 Problem Formulation and Algorithms

In this section, we explain: i) the SCAR and SNAR assumptions, ii) our PULSCAR algorithm for SCAR data and PULSNAR algorithm for SNAR data, iii) bandwidth estimation techniques, and iv) method to find the number of clusters in the labeled positive set. Our method to calibrate probabilities and enhance classification performance using PULSCAR/PULSNAR is in Appendix B and C, respectively.

3.1 SCAR assumption and SNAR assumption

In PU learning settings, a positive or unlabeled example can be represented as a triplet (x, y, s) where "x" is a vector of the attributes, "y" the actual class, and "s" a binary variable representing whether or not the example is labeled. If an example is labeled (s=1), it belongs to the positive class (y=1) i.e., $p(y = 1|s = 1) = 1$. If an example is not labeled (s=0), it can belong to either class. Since only positive examples are labeled, $p(s = 1|x, y = 0) = 0$ Elkan & Noto (2008). Under the SCAR assumption, a labeled positive is an independent and identically distributed (i.i.d) example from the positive distribution, i.e., positives are selected independently of their attributes. Therefore, $p(s = 1|x, y = 1) = p(s = 1|y = 1)$ Elkan & Noto (2008).

For a given dataset, $p(s = 1|y = 1)$ is a constant and is the fraction of labeled positives. If $|P|$ is the number of labeled positives, $|U|$ is the number of unlabeled examples, and α is the unknown fraction of positives in the unlabeled set, then

$$
\begin{aligned}
p(s=1) = \frac{|P|}{|P|+|U|} \quad \text{and} \quad p(y=1) &= \frac{|P|+\alpha|U|}{|P|+|U|} \\
p(s=1|y=1) = \frac{p(y=1|s=1)p(s=1)}{p(y=1)} &= \frac{p(s=1)}{p(y=1)} \quad \text{, since } p(y=1|s=1)=1 \\
&= \frac{|P|}{|P|+\alpha|U|}, \text{ which is a constant.}
\end{aligned} \tag{1}
$$

On the contrary, under the SNAR assumption, the probability that a positive example is labeled is not independent of its attributes. Stated formally, the assumption is that $p(s = 1|x, y = 1) \neq p(s = 1|y = 1)$ i.e. $p(s = 1|x, y = 1)$ is not a constant, which can be proved by Bayes' rule (Appendix A).

The SCAR assumption can hold when: a) both labeled and unlabeled positives are not a mixture of subclasses (i.e. they have similar attributes); b) both labeled and unlabeled positives are from k subclasses $(1 \ldots k)$, and the relative proportion of those subclasses is the same in positive and unlabeled sets. Intra-subclass examples will have similar attributes, whereas the inter-subclass examples may not have similar attributes. E.g., in a dataset of patients positive for diabetes, type 1 patients will be in one subclass, and type 2 patients will be in another subclass.

The SCAR assumption can fail when labeled and unlabeled positives are from k subclasses, and the proportion of those subclasses is different in positive and unlabeled sets. Suppose both positive and unlabeled sets have subclass 1 and subclass 2 positives, and in the positive set, their ratio is 30:70. If the ratio is also 30:70 in the unlabeled set, the SCAR assumption will hold. If it was different, say 80:20, the SCAR assumption would not hold.

3.2 Positive and Unlabeled Learning Selected Completely At Random (PULSCAR) Algorithm

Given any ML algorithm, $\mathcal{A}(x)$, that generates $[0 \ldots 1]$ probabilities for the data based on covariates x, let $f_p(x)$, $f_n(x)$, and $f_u(x)$ be probability density functions (PDFs) corresponding to the probability distribution of positives, negatives, and unlabeled respectively. Let α be the unknown proportion of positives in the unlabeled, then

$$f_u(x) \equiv \alpha f_p(x) + (1 - \alpha) f_n(x) \quad \text{, Using the rule of total probability} \tag{2}$$

A key observation is that $f_p(x)$ should not exceed $\alpha f_u(x)$ anywhere, allowing one to place an upper bound on α. PULSCAR estimates α by finding the value α where the following objective function maximally changes:

$$\begin{aligned} f(\alpha) &= log(|\min(f_u(x) - \alpha f_p(x))| + \epsilon), \text{ where} \\ \epsilon &= |\min(f_p(x))|, \text{ if } \min(f_p(x)) \neq 0, \text{ else } \epsilon = 10^{-10}. \end{aligned} \tag{3}$$

The intuition behind the objective function is that $|\min(f_u(x) - \alpha f_p(x))|$ approaches zero at the point where $\alpha f_p(x)$ equals $f_u(x)$, see Figure 1D. This point is what we need to locate to place an upper bound on α. The reason we use the logarithm is it steeply approaches $-\infty$ as $|\min(f_u(x) - \alpha f_p(x))|$ approaches zero. Adding ϵ to prevent $log(0)$ and using finite differences to find the max change in slope gives a robust estimator that is resilient to noise. This also obviates the need for implementing an iterative solver technique, accounting in part for the speed of our algorithm.

We use beta kernel density estimates on ML-predicted class 1 probabilities of positives and unlabeled to estimate $f_p(x)$ and $f_u(x)$. We use a finite difference approximation of the slope of $f(\alpha)$ to find its maximum. The value of α can also be determined visually by plotting the objective function (Figure 1D); the sharp inflection point in the plot represents the value of α. Algorithm 1 shows the pseudocode of the PULSCAR algorithm to estimate α using the objective function based on probability densities. Algorithm 2 is a subroutine to compute the beta kernel bandwidth. Full source code for our algorithms is provided as a supplemental document.

3.3 Kernel Bandwidth estimation

A beta kernel estimator is used to create a smooth density estimate of both the positive and unlabeled ML probabilities, generating distributions over $[0 \ldots 1]$, free of the problematic boundary biases of kernels (e.g. Gaussian) whose range extends outside that interval, adopting the approach of Chen (1999). Another problem with (faster) Gaussian kernel density implementations is that they often use polynomial approximations that

can generate negative values in regions of low support, dramatically distorting α estimates which require non-negative probability distribution estimates. The beta PDF is as follows Virtanen et al. (2020):

$$f(x, a, b) = \frac{\Gamma(a+b)x^{a-1}(1-x)^{b-1}}{\Gamma(a)\Gamma(b)}, \tag{4}$$

for x ∈ [0,1], where Γ is the gamma function, $a = 1 + \frac{z}{bw}$ and $b = 1 + \frac{1-z}{bw}$, with z the bin location, and bw the bandwidth.

Kernel bandwidth selection can also significantly influence α estimates: too narrow of a bandwidth can result in outliers driving poor estimates, and too wide of a bandwidth prevents distinguishing between distributions. We use a histogram bin count heuristic to generate a histogram density, then optimize the beta distribution bandwidth to best fit that histogram density.

3.3.1 Bin count

Our implementation supports 4 well-known methods to determine the number of histogram bins: square root, Sturges' rule, Rice's rule, Scott's rule, and Freedman–Diaconis (FD) rule Alxneit (2020).

3.3.2 Bandwidth estimation

We compute a histogram density using a bin count heuristic and a beta kernel density estimate at those bin centers using the ML probabilities of both the positive and unlabeled examples. We find the global minimum of the mean squared error (MSE) between the histogram and beta kernel densities using the scipy *differential_evolution()* optimizer Virtanen et al. (2020), solving for the best bandwidth in the range [0.001...0.5]. That bandwidth is chosen for kernel density estimation in the PULSCAR algorithm. All experiments herein use MSE as the error metric, but alternatively, the Jensen-Shannon distance can be employed.

Algorithm 1 PULSCAR Algorithm

Input: X ($X_p \cup X_u$), y ($y_p \cup y_u$), n_bins
Output: estimated α

predicted_probabilities (p) ← $\mathcal{A}(X, y)$
p0 ← p[y == 0]
p1 ← p[y == 1]
estimation_range ← [0, 0.0001, 0.0002, ..., 1.0]
bw ← estimate_bandwidth_pu(p, n_bins)
D_u ← beta_kernel(p0, bw, n_bins)
D_p ← beta_kernel(p1, bw, n_bins)
$\epsilon \leftarrow |\min(D_p)|$
if $\epsilon = 0$ **then**
  $\epsilon \leftarrow 10^{-10}$
end if
len ← length(estimation_range)
selected_range ← estimation_range[2:len]
α ← estimation_range
f(α) ← $\log(|\min(D_u - \alpha D_p)| + \epsilon)$
d ← f'(α)
i ← where the value of d changes maximally
return selected_range[i]

Algorithm 2 estimate_bandwidth_pu

Input: predicted_probabilities, n_bins
Output: bandwidth

preds ← predicted_probabilities
bw ∈ [0.001, 0.5]
D_{hist} ← histogram(preds, n_bins, density=True)
D_{beta} ← beta_kernel(preds, bw, nbins)
return optimize(MeanSquaredError(D_{hist}, D_{beta}))

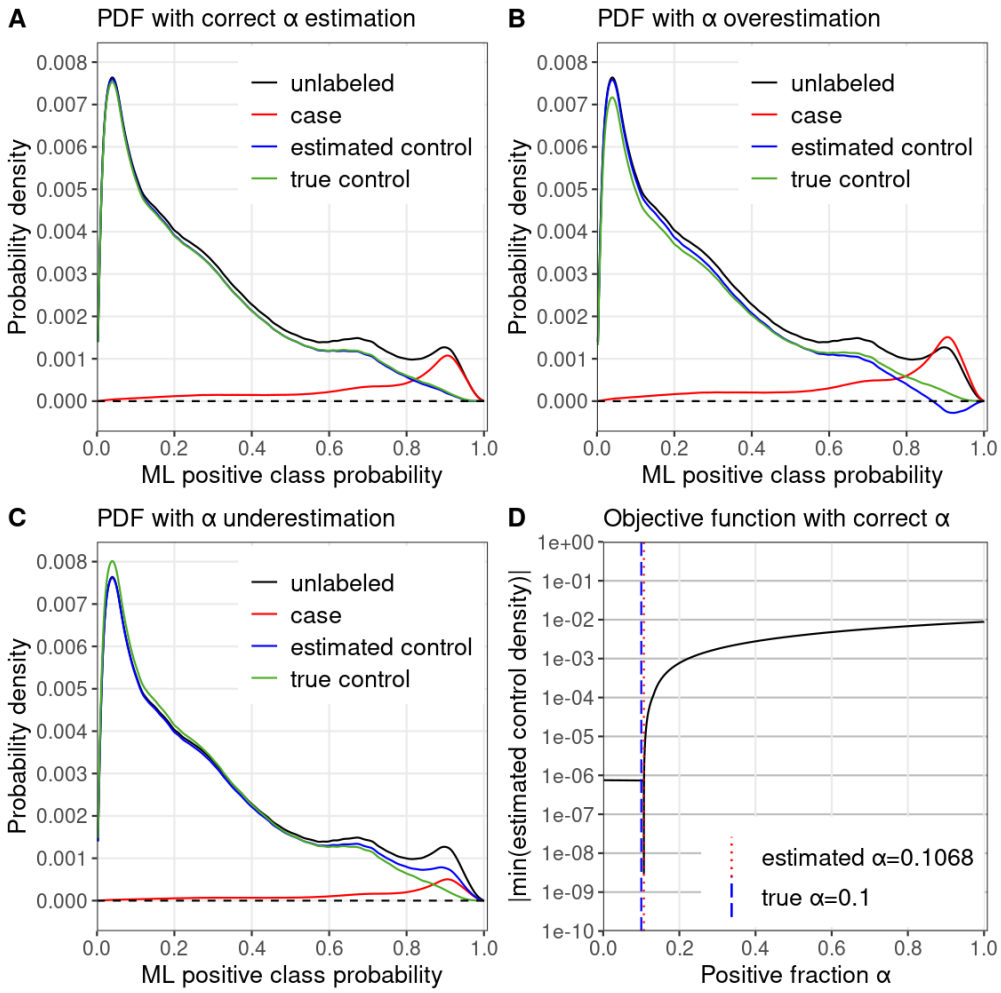

Figure 1: **PULSCAR algorithm visual intuition**. PULSCAR finds the smallest α such that $f_u(x) - \alpha f_p(x)$ is everywhere positive in $[0 \ldots 1]$. A) Kernel density estimates for simulated data with $\alpha = 10\%$ positives in the unlabeled set – estimated negative density (blue) nearly equals the ground truth (green). B) Overweighting the positive density by $\alpha = 15\%$ results in the estimated negative density (blue), $f_u(x) - \alpha f_p(x)$ dropping below zero. C) Underweighting the positive density by $\alpha = 5\%$ results in the estimated negative density (blue) being higher than the ground truth (green). D) Objective function with estimated $\alpha = 10.68\%$ selected where the finite-differences estimate of the slope is largest – very close to ground truth $\alpha = 10\%$.

3.4 Positive and Unlabeled Learning Selected Not At Random (PULSNAR) Algorithm

We propose a new PU learning algorithm (PULSNAR) to estimate the α in SNAR data, i.e., labeled positives are not selected completely at random. PULSNAR uses a divide-and-conquer strategy for the SNAR data. It converts a SNAR problem into several sub-problems using an unsupervised learning method (clustering), each of which better satisfies the SCAR assumption; then applies the PULSCAR algorithm to those sub-problems. The final alpha is computed by summing the alpha returned by the PULSCAR algorithm for each cluster.

$$\alpha = (|U|\alpha_1 + |U|\alpha_2 + ... + |U|\alpha_c)/|U|$$
$$\alpha = \alpha_1 + \alpha_2 + ... + \alpha_c, \quad c = \text{number of clusters} \tag{5}$$

Figure 2 visualizes the PULSNAR algorithm, and Algorithm 3 provides its pseudocode.

3.4.1 Clustering rationale

Suppose both positive and unlabeled sets contain positives from k subclasses $(1 \ldots k)$. With selection bias (SNAR), the subclass proportions will differ between the sets, and thus the PDF of the labeled positives cannot be scaled by a uniform α to estimate positives among the unlabeled. The smallest subclass would drive an α underestimate with PULSCAR. To address this, we apply clustering to the labeled positives to split them into c clusters. Clustering separates subclasses of positives, and if the assumption that subclass membership drives selection bias holds, PU models comprising examples from one cluster and the unlabeled set will be more likely to follow the SCAR assumption. Applying PULSCAR to each cluster of positives versus the unlabeled results in better estimates of the proportions of similar unlabeled positives (Figure 2).

3.4.2 Determining the number of clusters in the positive set

We build an XGBoost Chen & Guestrin (2016) model on all positive and unlabeled examples to determine the important features and their *gain* scores. A *gain* score measures the magnitude of the feature's contribution to the model. We select all labeled positives and then cluster them on those features scaled by their corresponding *gain* score, using scikit_learn's Gaussian mixture model (GMM) method. To establish the number of clusters (n_components), we iterate n_components over $1 \ldots 25$ and compute the Bayesian information criterion (BIC)Vrieze (2012) for each clustering model. We use max_iter=250, and covariance_type="full". The other parameters are used with their default values. We implemented the "Knee Point Detection in BIC" algorithm to find the number of clusters in the labeled positives Zhao et al. (2008).

Algorithm 3 PULSNAR Algorithm

Input: X $(X_p \cup X_u)$, y $(y_p \cup y_u)$, n_bins
Output: estimated α

1: feature_importance $(v_1 ... v_k)$, imp_features $(x_1 ... x_k) \leftarrow \mathcal{A}(X, y)$
2: $x'_1 ... x'_k \leftarrow x_1 v_1 ... x_k v_k$
3: $X'_p \leftarrow X_p[x'_1 ... x'_k]$
4: clusters $s_1 ... s_c \leftarrow$ GMM(X'_p)
5: $\alpha \leftarrow 0$
6: **for** c in $s_1 ... s_c$ **do**
7: X' $\leftarrow X_p[c] \cup X_u$
8: y' $\leftarrow y_p[c] \cup y_u$
9: $\alpha \leftarrow \alpha +$ PULSCAR(X', y', n_bins)
10: **end for**
11: **return** α

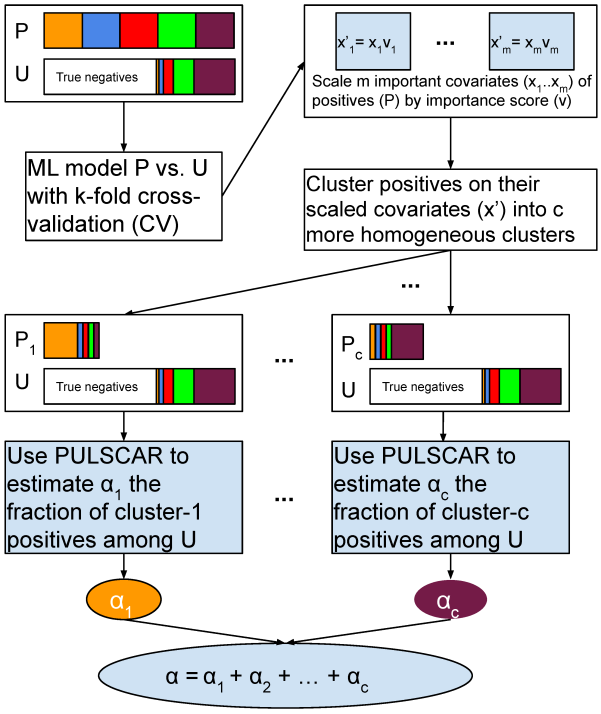

Figure 2: **Schematic of PULSNAR algorithm**. An ML model is trained and tested with 5-fold CV on all positive and unlabeled examples. The important covariates that the model used are scaled by their importance value. Positives are divided into c clusters using the scaled important covariates. c ML models are trained and tested with 5-fold CV on the records from a cluster and all unlabeled records. We estimate the proportions $(\alpha_1 ... \alpha_c)$ of each subtype of positives in the unlabeled examples using PULSCAR. The sum of those estimates gives the overall fraction of positives in the unlabeled set. P = positive set, U = Unlabeled set.

3.5 Calculating calibrated probabilities

The approach to calibrate the ML-predicted probabilities of positive and unlabeled examples in the SCAR and SNAR data is explained in Appendix B.

3.6 Improving classification performance

An approach to improving PULSCAR and PULSNAR classification, based on flipping the highest probability $\alpha|U|$ unlabeled examples to 1, is explained in Appendix C.

4 Experiments

We evaluated our proposed PU learning algorithms in terms of α estimates, six classification performance metrics, and probability calibration. We used real-world ML benchmark datasets and synthetic data for our experiments. For real-world data, we used Bank Moro et al. (2014) and KDD Cup 2004 particle physics Caruana et al. (2004) datasets as SCAR data and Statlog (Shuttle) UCI ML Repository (2022b) and Firewall datasets UCI ML Repository (2022a) as SNAR data. The synthetic (SCAR and SNAR) datasets were generated using the scikit-learn function *make_classification()* Pedregosa et al. (2011). We used XGBoost as a binary classifier in our proposed algorithms. To train the classifier on the imbalanced data, we used *scale_pos_weight* parameter of XGBoost to scale the weight of the labeled positive examples by the factor $s = \frac{|U|}{|P|}$. We also compared our methods with five recently published methods for PU learning: KM1 and KM2 Ramaswamy et al. (2016), TICE Bekker & Davis (2018), DEDPUL Ivanov (2020) and CleanLab Northcutt et al. (2021). KM1, KM2, and TICE algorithms were not scalable and failed to execute on large datasets, so we used smaller synthetic datasets to compare our method with these methods. We compared PULSNAR with only DEDPUL on large synthetic datasets (Appendix D). Also, Ivanov (2020) previously demonstrated that DEDPUL outperformed KM and TICE algorithms on several UCI (University of California Irvine) ML benchmark and synthetic datasets.

4.1 Synthetic data

We generated SCAR and SNAR PU datasets with different fractions of positives (1%, 5%, 10%, 20%, 30%, 40%, and 50%) among the unlabeled examples to test the effectiveness of our proposed algorithms. For each fraction, we generated 40 datasets using sklearn's *make_classification()* function with random seeds 0-39. The *class_sep* parameter of the function was used to specify the separability of data classes. Values nearer to 1.0 make the classification task easier; we used class_sep=0.3 to create difficult classification problems.

4.1.1 SCAR data

The datasets contained 2,000 positives (class 1) and 6,000 unlabeled (class 0) examples with 50 continuous features. The unlabeled set comprised k% positive examples with labels flipped to 0 and $(100 - k)$% negative examples.

4.1.2 SNAR data

We generated datasets with 6 labels (0-5), defining '0' as negative and 1-5 as positive subclasses. These datasets contained 2,000 positives (400 from each positive subclass) and 6,000 unlabeled examples with 50 continuous features. The unlabeled set comprised k% positive examples with labels (1-5) flipped to 0 and (100-k)% negative examples. The unlabeled positives were markedly SNAR, with the 5 subclasses comprising 1/31, 2/31, 4/31, 8/31, and 16/31 of the unlabeled positives. (e.g., in the unlabeled set with 20% positives, negative: 4,800, label 1 positive: 39, label 2 positive: 77, label 3 positive: 155, label 4 positive: 310, label 5 positive: 619).

4.2 SCAR ML Benchmark Datasets

4.2.1 UCI Bank dataset

The dataset has 45,211 records (class 1: 5,289, class 0: 39,922) with 16 features. This dataset is a good example of data with class imbalance and mixed features. Since the features contain both numerical and categorical values, they were one-hot encoded Hollaar (1982) using the scikit-learn function *OneHotEncoder()* Pedregosa et al. (2011). The encoder derives the categories based on the unique values in each feature, resulting in 9,541 features. The ML classifier was applied to the encoded features.

4.2.2 KDD Cup 2004 Particle Physics dataset

The dataset contains two types of particles generated in high-energy collider experiments; 50,000 examples (class 1: 24,861, class 0: 25,139) with 78 numerical attributes. This dataset is a good example of balanced data.

In both datasets, class 1 records were used as positive, and class 0 records were used as unlabeled for the ML model. To add k% positive examples to the unlabeled set, the labels of m randomly selected positive records were flipped from 1 to 0, where $m = \frac{k|U|}{100-k}$.

4.3 SNAR ML Benchmark Datasets

4.3.1 UCI Statlog (Shuttle) Dataset

The dataset contains 43,500 records (class 1: 34,108, class 2: 37, class 3: 132, class 4: 6,748, class 5: 2,458, class 6: 6, class 7: 11) with 9 numerical attributes. This dataset is an example of data with multiclass and class imbalance. We used class 1 as unlabeled examples and the rest of the records as subclasses of positive examples for the ML model (positive: 9,392, unlabeled: 34,108).

4.3.2 UCI Firewall dataset

It is a multiclass dataset containing 65,532 records ('allow': 37,640, 'deny': 14,987, 'drop': 12,851, 'reset-both': 54) with 12 numerical attributes. Class 'allow' was used as unlabeled examples, and the others ('deny', 'drop', 'reset-both') were used as subclasses of positive examples for the ML model (positive: 27,892, unlabeled: 37,640).

In both datasets, the majority of positives are from two classes (*shuttle: class 4, 5; firewall: 'deny', 'drop'*). So, to add k% positive examples to the unlabeled set, we randomly selected some examples from the minor positive classes and the remaining examples from two major positive classes in equal proportion. Thus, the proportion of positives in the positive set differed from the unlabeled set.

4.4 Estimation of fraction of positives among unlabeled examples

We applied the PULSCAR algorithm to both SCAR and SNAR data, and the PULSNAR algorithm only to SNAR data, to estimate α.

4.4.1 Using the PULSCAR algorithm

To find the 95% confidence interval (CI) on estimation, we ran XGBoost with 5-fold cross-validation (CV) for 40 random instances of each dataset generated (or selected from benchmark data) using 40 random seeds. Each iteration's class 1 predicted probabilities of positives and unlabeled were used to calculate the value of α.

4.4.2 Using the PULSNAR algorithm

The labeled positives were divided into c clusters to get homogeneous subclasses of labeled positives. The XGBoost ML models were trained and tested with 5-fold CV on data from each cluster and all unlabeled

records. For each cluster, α was estimated by applying the PULSCAR method to class 1 predicted probabilities of positives from the cluster and all unlabeled examples. The overall proportion was calculated by summing the estimated α for each cluster. To compute the 95% CI on the estimation, PULSNAR was repeated 40 times on data generated/selected using 40 random seeds.

5 Results

5.1 Synthetic datasets

Figure 3 shows the α estimated by PU learning algorithms for synthetic datasets. TICE overestimated α for all fractions in both SCAR and SNAR datasets. For SCAR datasets, only PULSCAR returned close estimates for all fractions; DEDPUL overestimated for 1%; KM1 and KM2 underestimated for 50%; CleanLab underestimated for larger α (10-50%). For SNAR datasets, only PULSNAR's estimates were close to the true α; other algorithms underestimated for larger α (40-50%). Figure 14 in Appendix D shows the α estimated by DEDPUL and PULSNAR on large SNAR datasets with different class imbalances. As the class imbalance increased, the performance of DEDPUL dropped, especially for larger fractions. The estimated α by the PULSNAR method was close to the true α for all fractions and sample sizes.

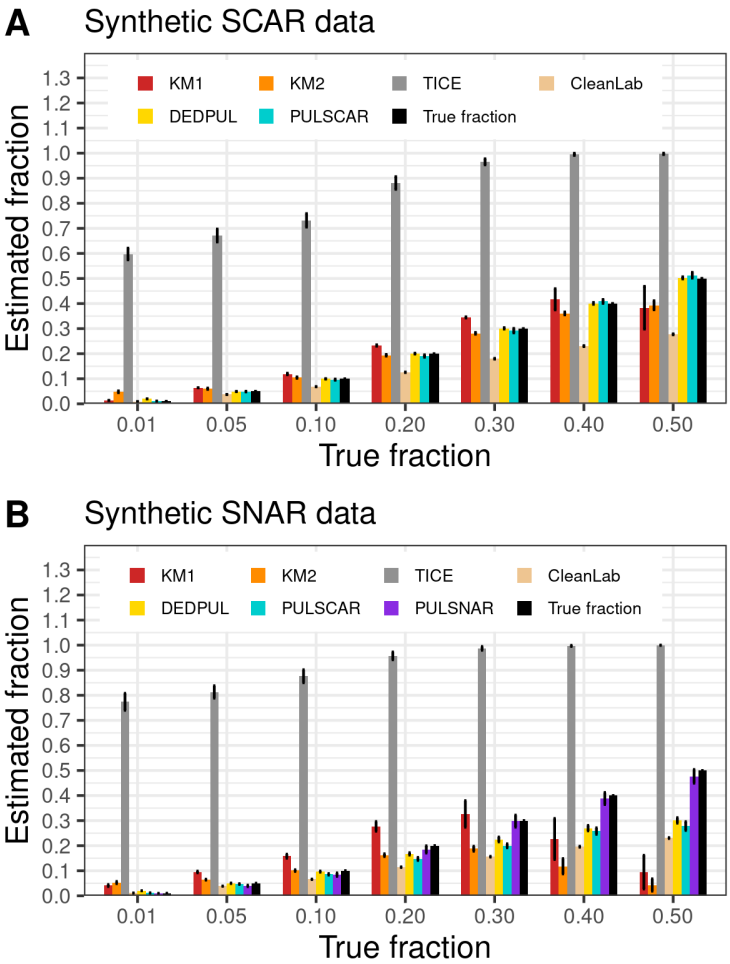

Figure 3: **KM1, KM2, TICE, CleanLab, DEDPUL, PULSCAR, and PULSNAR evaluated on SCAR and SNAR synthetic datasets**. The bar represents the mean value of the estimated α, with 95% confidence intervals for estimated α.

5.2 ML Benchmark datasets

5.2.1 SCAR data

Figure 4 shows the α estimated by PU learning algorithms for the KDD Cup 2004 particle physics and UCI bank datasets. For KDD Cup, estimates by PULSCAR and DEDPUL were close to the true answers for all fractions; TICE overestimated for all fractions; CleanLab overestimated for 1-30%. For Bank, only PULSCAR returned correct estimates for all fractions; other algorithms overestimated for all fractions.

5.2.2 SNAR data

Figure 5 shows the α estimated by PU learning algorithms for UCI Shuttle and UCI Firewall datasets. For the Shuttle dataset, only PULSNAR's estimates were close to the true fractions; other algorithms either overestimated or underestimated. For the Firewall dataset, TICE overestimated, and CleanLab underestimated for all fractions; PULSNAR's estimates were within $\pm 20\%$ of the true α; DEDPUL and PULSCAR underestimated for 40%.

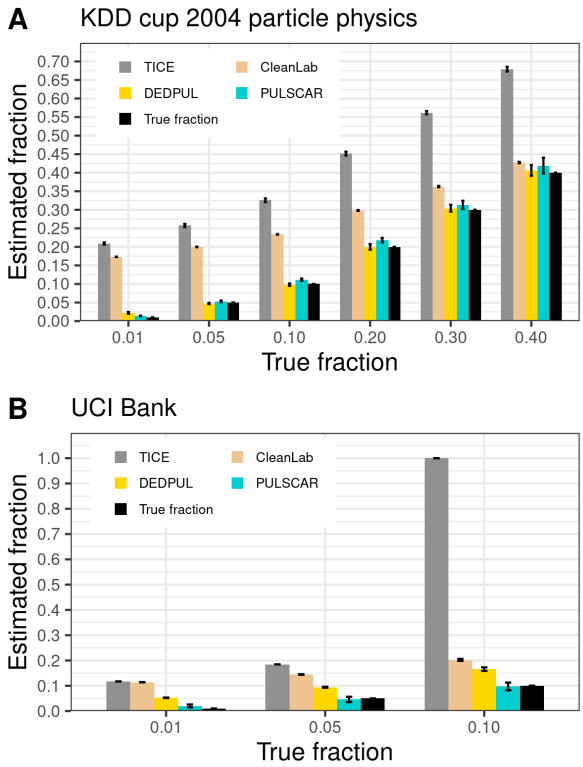

Figure 4: **TICE, CleanLab, DEDPUL, and PULSCAR evaluated on SCAR KDD cup 2004 particle physics and UCI Bank datasets**. The bar represents the mean value of the estimated α, with 95% confidence intervals for estimated α. KM1 and KM2 failed to execute on both datasets. As TICE was taking several hours to finish one iteration on the bank dataset, the mean α was computed using 5 iterations, and the standard error was set to 0.

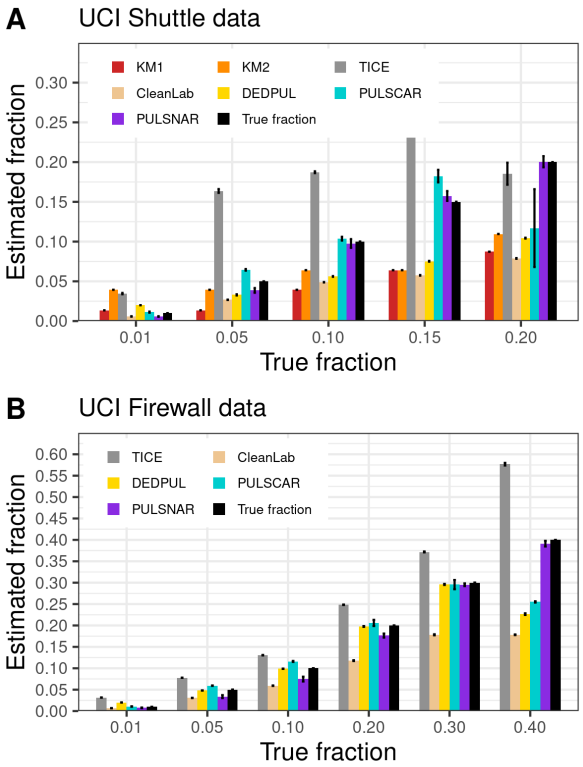

Figure 5: **KM1, KM2, TICE, CleanLab, DEDPUL, PULSCAR, and PULSNAR evaluated on SNAR UCI Shuttle and Firewall datasets**. The bar represents the mean value of the estimated α, with 95% confidence intervals for estimated α. KM1 and KM2 failed to execute on the Firewall dataset. As KM1 and KM2 were taking several hours to finish one iteration on the Shuttle dataset, the mean α was computed using 5 iterations, and the standard error was set to 0.

5.3 Probability calibration

Appendix B.1 shows the calibration curves generated using the unblinded labels and isotonically calibrated probabilities of positive and unlabeled examples or only unlabeled examples in the SCAR and SNAR data.

5.4 Classification performance metrics

Appendix C.1 shows substantial improvement in 6 classification performance metrics when applying PULSCAR and PULSNAR versus XGBoost alone.

6 Discussion and Conclusion

This paper presented novel PU learning algorithms to estimate the proportion of positives among unlabeled examples in both SCAR and SNAR data with/without class imbalance. Preliminary work (not shown) suggests PULSNAR α estimation is robust to overestimating the number of clusters. Our synthetic experiments were run on difficult classification tasks with low separability. For SNAR data, with true $\alpha = 1\%$, when we increased *class_sep* from 0.3 to 0.5 the PULSNAR α estimate improved from 1.6% (Figure 3) to 0.98% (data not shown). Experimentally, we showed that our proposed methods outperformed state-of-art methods on synthetic and real-world SCAR and SNAR datasets. PU learning methods based on the SCAR assumption generally give poor α estimates on SNAR data. We demonstrated that after applying PULSCAR/PULSNAR, classifier performance, including calibration, improved significantly. Since the SCAR assumption often does not hold in real-world data, better α estimates in a SNAR setting open up new horizons in PU Learning.

References

Ivo Alxneit. Particle size distributions from electron microscopy images: avoiding pitfalls. *The Journal of Physical Chemistry A*, 124(48):10075–10081, 2020.

Jessa Bekker and Jesse Davis. Estimating the class prior in positive and unlabeled data through decision tree induction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

Jessa Bekker, Pieter Robberechts, and Jesse Davis. Beyond the selected completely at random assumption for learning from positive and unlabeled data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 71–85. Springer, 2019.

Rich Caruana, Thorsten Joachims, and Lars Backstrom. KDD-Cup 2004: results and analysis. *SIGKDD Explor. Newsl.*, 6(2):95–108, December 2004.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Song Xi Chen. Beta kernel estimators for density functions. *Computational Statistics & Data Analysis*, 31(2): 131–145, 1999.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Marthinus Christoffel Du Plessis and Masashi Sugiyama. Class prior estimation from positive and unlabeled data. *IEICE TRANSACTIONS on Information and Systems*, 97(5):1358–1362, 2014.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 213–220, 2008.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Hongjun Lu, and Philip S Yu. Text classification without negative examples revisit. *IEEE transactions on Knowledge and Data Engineering*, 18(1):6–20, 2005.

Walter Gerych, Thomas Hartvigsen, Luke Buquicchio, Emmanuel Agu, and Elke Rundensteiner. Recovering the propensity score from biased positive unlabeled data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6694–6702, Jun. 2022. doi: 10.1609/aaai.v36i6.20624. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20624`.

Chen Gong, Qizhou Wang, Tongliang Liu, Bo Han, Jane You, Jian Yang, and Dacheng Tao. Instance-dependent positive and unlabeled learning with labeling bias estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4163–4177, 2021.

Zayd Hammoudeh and Daniel Lowd. Learning from positive and unlabeled data with arbitrary positive shift. *Advances in Neural Information Processing Systems*, 33:13088–13099, 2020.

Lee A. Hollaar. Direct implementation of asynchronous control units. *IEEE Transactions on Computers*, 31 (12):1133–1141, 1982.

Dmitry Ivanov. Dedpul: Difference-of-estimated-densities-based positive-unlabeled learning. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 782–790. IEEE, 2020.

Shantanu Jain, Martha White, Michael W Trosset, and Predrag Radivojac. Nonparametric semi-supervised learning of class proportions. *arXiv preprint arXiv:1601.01944*, 2016.

Kristen Jaskie and Andreas Spanias. Positive and unlabeled learning algorithms and applications: A survey. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pp. 1–8. IEEE, 2019.

Kristen Jaskie, Charles Elkan, and Andreas Spanias. A modified logistic regression for positive and unlabeled learning. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 2007–2011. IEEE, 2019.

Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pp. 448–455, 2003.

Zhigang Liu, Wenzhong Shi, Deren Li, and Qianqing Qin. Partially supervised classification: based on weighted unlabeled samples support vector machine. In *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications*, pp. 1216–1230. IGI Global, 2008.

Chuan Luo, Pu Zhao, Chen Chen, Bo Qiao, Chao Du, Hongyu Zhang, Wei Wu, Shaowei Cai, Bing He, Saravanakumar Rajmohan, and Qingwei Lin. Pulns: Positive-unlabeled learning with effective negative sample selector. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8784–8792, May 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/17064`.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Curtis Northcutt, Lu Jiang, and Isaac Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Harish Ramaswamy, Clayton Scott, and Ambuj Tewari. Mixture proportion estimation via kernel embeddings of distributions. In *International conference on machine learning*, pp. 2052–2060. PMLR, 2016.

Guangxin Su, Weitong Chen, and Miao Xu. Positive-unlabeled learning from imbalanced data. In *IJCAI*, pp. 2995–3001, 2021.

UCI ML Repository. University of california irvine (uci) machine learning repository: Internet firewall data set. `https://archive.ics.uci.edu/ml/datasets/Internet+Firewall+Data`, 2022a. Accessed: 2022-06-30.

UCI ML Repository. University of california irvine (uci) machine learning repository: Statlog (shuttle) data set. `https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)`, 2022b. Accessed: 2022-06-30.

Pauli Virtanen et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Scott I Vrieze. Model selection and psychological theory: a discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic). *Psychological methods*, 17(2):228, 2012.

Chunlin Wang, Chris Ding, Richard F Meraz, and Stephen R Holbrook. Psol: a positive sample only learning algorithm for finding non-coding rna genes. *Bioinformatics*, 22(21):2590–2596, 2006.

Yixing Xu, Chang Xu, Chao Xu, and Dacheng Tao. Multi-positive and unlabeled learning. In *IJCAI*, pp. 3182–3188, 2017.

Yuguang Yan, Mingkui Tan, Yanwu Xu, Jiezhang Cao, Michael Ng, Huaqing Min, and Qingyao Wu. Oversampling for imbalanced data via optimal transport. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5605–5612, Jul. 2019. doi: 10.1609/aaai.v33i01.33015605. URL `https://ojs.aaai.org/index.php/AAAI/article/view/4503`.

Hwanjo Yu. Single-class classification with mapping convergence. *Machine Learning*, 61(1):49–69, 2005.

Hwanjo Yu, Jiawei Han, and KC-C Chang. Pebl: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81, 2004.

Qinpei Zhao, Ville Hautamaki, and Pasi Fränti. Knee point detection in bic for detecting the number of clusters. In *International conference on advanced concepts for intelligent vision systems*, pp. 664–673. Springer, 2008.

Appendix

A Proof: positives are not independent of their attributes under the SNAR Assumption

Under the SNAR assumption, the probability that a positive example is labeled is not independent of its attributes. Stated formally, the assumption is that $p(s=1|x,y=1) \neq p(s=1|y=1)$ i.e. $p(s=1|x,y=1)$ is not a constant.

Proof:

$$\begin{aligned} p(s=1|x,y=1) &= p(y=1|(s=1|x))p(s=1|x) \\ &= p(y=1|(s=1|x))\frac{p(x|s=1)p(s=1)}{p(x)} \text{ , using Bayes' rule} \\ &= \frac{p(x|s=1)p(s=1)}{p(x)} \text{ , since } p(y=1|(s=1|x))=1 \\ &= \text{a function of } x. \end{aligned}$$

B Algorithm for calibrating probabilities

Algorithm 4 shows the complete pseudocode to calibrate the machine learning (ML) model predicted probabilities. Once α is known, we seek to transform the original class 1 probabilities so that their sum is equal to $\alpha|U|$ among the unlabeled or $|P|+\alpha|U|$ among positive and unlabeled, and that they are well-calibrated. Our approach is to probabilistically flip $\alpha|U|$ labels of unlabeled to positive (from 0 to 1) in such a way as to match the PDF of labeled positives across 100 equispaced bins over $[0\ldots1]$, then fit a logistic or

isotonic regression model on those labels versus the probabilities to generate the transformed probabilities. To determine the number of unlabeled examples that need to be flipped in each bin, we compute the normalized histogram density, D_hist, for the labeled positives with 100 bins and then multiply $\alpha|U|$ with D_hist.

The unlabeled examples are also divided into 100 bins based on their predicted probabilities. Starting from the bin with the highest probability(p=1), we randomly select k examples and flip their labels from 0 to 1, where k is the number of unlabeled examples that need to be flipped in the bin. If the number of records (n_1) that need to be flipped in a bin is more than the number of records (n_2) present in the bin, the difference ($n_1 - n_2$) is added to the number of records to be flipped in the next bin, resulting in $\alpha|U|$ flips.

After flipping the labels of $\alpha|U|$ unlabeled examples from 0 to 1, we fit an isotonic or sigmoid regression model on the ML-predicted class 1 probabilities with the updated labels to obtain calibrated probabilities.

The above calibration approach applies to both SCAR and SNAR data. For the SNAR data, the PULSNAR algorithm divides labeled positive examples into k clusters and estimates the α for each cluster. For each cluster, the ML-predicted class 1 probabilities of the examples (positives from the cluster and all unlabeled examples or only unlabeled examples) are calibrated using the estimated α for the cluster. Since, for each cluster, PULSNAR uses all unlabeled examples, each unlabeled example has k ML-predicted/calibrated probabilities. The final ML-predicted/calibrated probability of an unlabeled example is calculated using the following Equation 6:

$$p = 1 - (1 - p_1)(1 - p_2) \dots (1 - p_k) \tag{6}$$

where p_k is the probability of an unlabeled example from cluster k.

Algorithm 4 calibrate_probabilities

Input: predicted_probs, labels, n_bins, calibration_method, calibration_data, α
Output: calibrated_probs

p0 $\leftarrow$ predicted_probs[labels == 0]
p1 $\leftarrow$ predicted_probs[labels == 1]
y0 $\leftarrow$ labels[labels == 0]
y1 $\leftarrow$ labels[labels == 1]
D_{hist} $\leftarrow$ histogram(p1, n_bins, density=True)
unlab_pos_count_in_bin $\leftarrow$ α $|p0|$ D_{hist}
p0_bins $\leftarrow$ split unlabeled examples into n_bins using p0
for k $\leftarrow$ [n_bins ... 1] **do**
  n_1 $\leftarrow$ unlab_pos_count_in_bin[k]
  n_2 $\leftarrow$ p0_bins[k]
  if $n_1 > n_2$ **then**
    $\hat{y0}$ $\leftarrow$ flip labels (y0) of n_2 examples from 0 to 1 in bin k
    unlab_pos_count_in_bin[k-1] $\leftarrow$ unlab_pos_count_in_bin[k-1] + $(n_1 - n_2)$
  else
    $\hat{y0}$ $\leftarrow$ flip labels (y0) of random n_1 examples from 0 to 1 in bin k
  end if
end for
if calibration_data == 'PU' **then**
  p, y $\leftarrow$ $p1 \cup p0$, $y1 \cup \hat{y0}$
else if calibration_data == 'U' **then**
  p, y $\leftarrow$ p0, $\hat{y0}$
end if
if calibration_method is 'sigmoid' **then**
  $\hat{p}$ $\leftarrow$ LogisticRegression(p, y)
else if calibration_method is 'isotonic' **then**
  $\hat{p}$ $\leftarrow$ IsotonicRegression(p, y)
end if
return $\hat{p}$

B.1 Experiments and Results

We used synthetic SCAR and SNAR datasets and KDD Cup SCAR dataset to test our calibration algorithm.

SCAR datasets: After estimating the α using the PULSCAR algorithm, we applied Algorithm 4 to calibrate the ML-predicted probabilities. To calculate the calibrated probabilities for both positive and unlabeled (PU) examples, we applied isotonic regression to the ML-predicted class 1 probabilities of PU examples with labels of positives and updated labels of unlabeled (of which $\alpha|U|$ were flipped per Algorithm 4). We applied isotonic regression to the unlabeled's predicted probabilities with their updated labels to calculate the calibrated probabilities only for the unlabeled.

SNAR datasets: Using the PULSNAR algorithm, the labeled positive examples were divided into k clusters. For each cluster, after estimating the α, Algorithm 4 was used to calibrate the ML-predicted probabilities. To calculate the calibrated probabilities for positives from a cluster and all unlabeled examples, we applied isotonic regression to their ML-predicted class 1 probabilities with labels of positives from the cluster and updated labels of unlabeled (of which $\alpha_j|U|$ were flipped for cluster $j = 1 \ldots k$, see Algorithm 4). We applied isotonic regression to the unlabeled's predicted probabilities with their updated labels to calculate the calibrated probabilities only for the unlabeled. Thus, each unlabeled example had k calibrated probabilities. We computed the final calibrated probability for each unlabeled example using Formula 6.

Figures 6, 7, 8, 9, 10 and 11 show the calibration curves generated using the unblinded labels and isotonically calibrated (red)/ uncalibrated (blue) probabilities. When both positive and unlabeled examples were used to calculate calibrated probabilities, the calibration curve followed the y=x line (well-calibrated probabilities). When only unlabeled examples were used, the calibration curve for 1% did not follow the y=x line, presumably due to the ML model being biased toward negatives, given the small α. Also, the calibration curves for the SCAR data followed the y=x line more closely than the calibration curves for the SNAR data. It is due to the fact that the final probability of an unlabeled example in the SNAR data is computed using its k probabilities from k clusters. So, a poor probability estimate from even one cluster can influence the final probability of an unlabeled example.

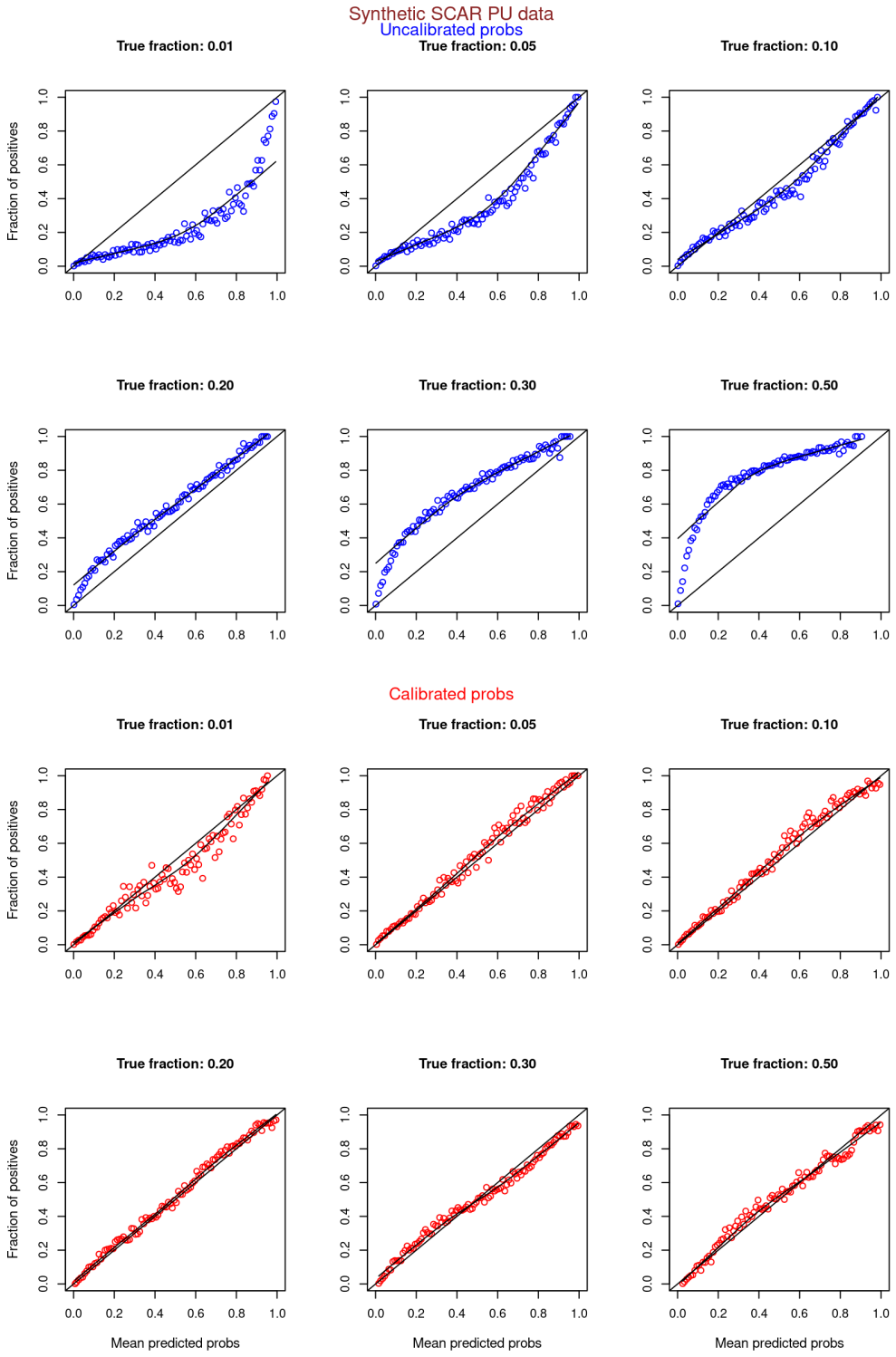

Figure 6: **Calibration curves for Synthetic SCAR datasets (both positive and unlabeled examples)**. Synthetic datasets were generated with different fractions of positives (1%, 5%, 10%, 20%, 30%, and 50%) among the unlabeled examples. class_sep=0.3, number of attributes=100, $|P| = 5,000$ and $|U| = 50,000$. Calibration curves were generated using both positive and unlabeled examples (Uncalibrated probabilities - blue, calibrated probabilities - red).

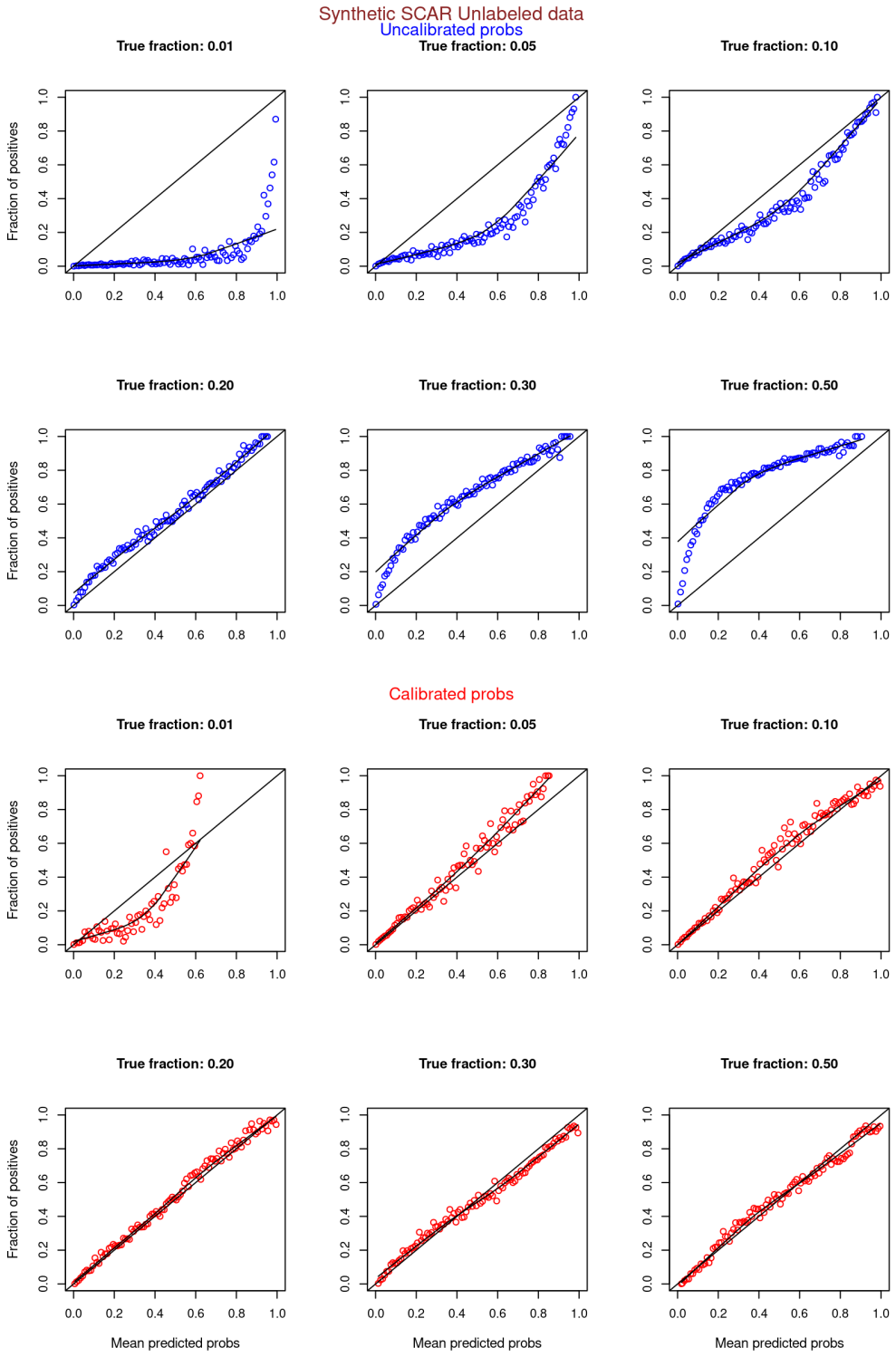

Figure 7: **Calibration curves for Synthetic SCAR datasets (only unlabeled examples)**. Synthetic datasets were generated with different fractions of positives (1%, 5%, 10%, 20%, 30%, and 50%) among the unlabeled examples. class_sep=0.3, number of attributes=100, $|P| = 5,000$ and $|U| = 50,000$. Calibration curves were generated using only unlabeled examples (Uncalibrated probabilities - blue, calibrated probabilities - red).

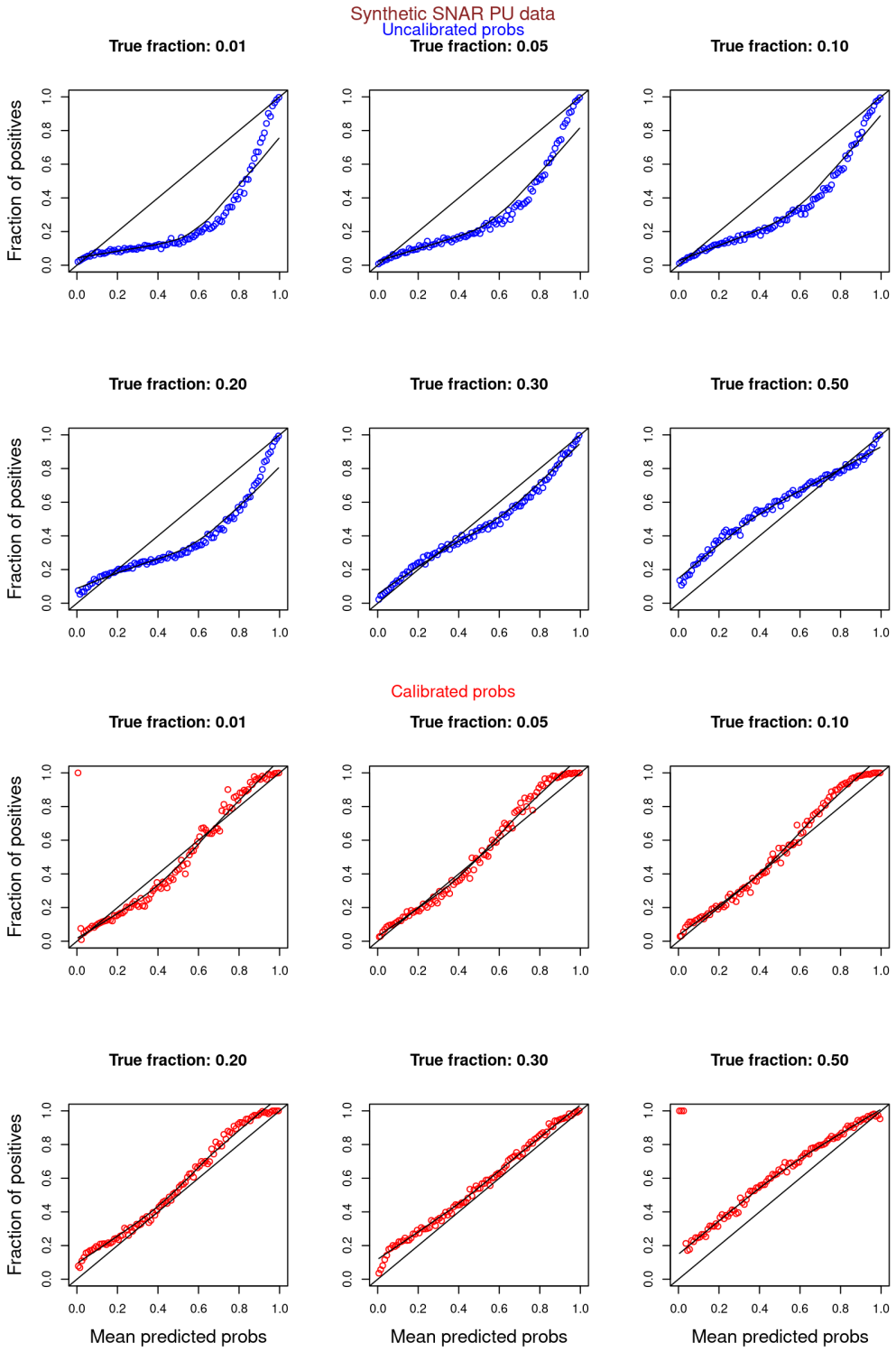

Figure 8: **Calibration curves for Synthetic SNAR datasets (both positive and unlabeled examples)**. Synthetic datasets were generated with different fractions of positives (1%, 5%, 10%, 20%, 30%, and 50%) among the unlabeled examples. class_sep=0.3, number of attributes=100, number of positive subclasses=5, $|P|$ = 20,000 (4,000 from each subclass) and $|U|$ = 50,000. Calibration curves were generated using both positive and unlabeled examples (Uncalibrated probabilities - blue, calibrated probabilities - red).

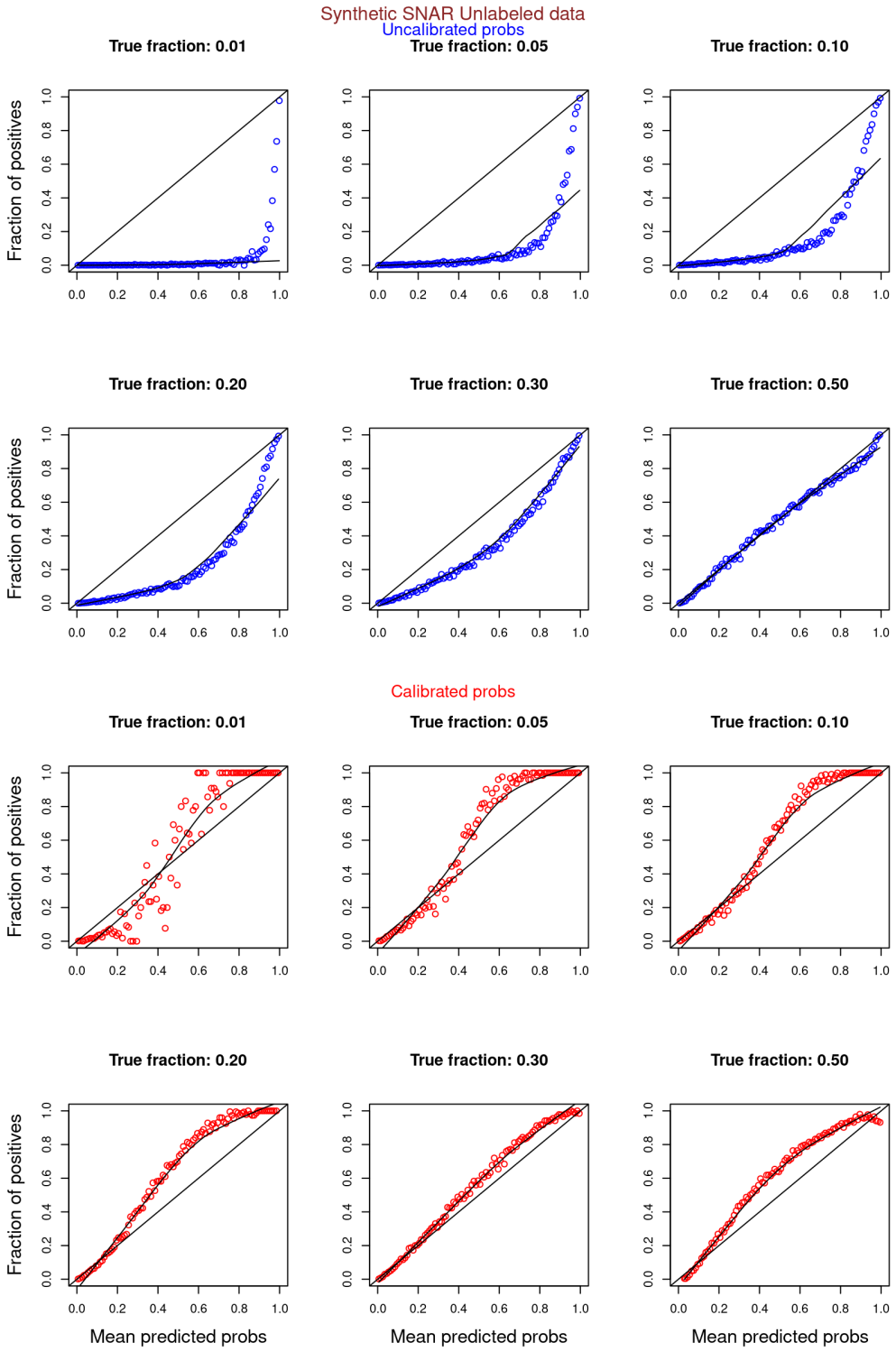

Figure 9: **Calibration curves for Synthetic SNAR datasets (only unlabeled examples)**. Synthetic datasets were generated with different fractions of positives (1%, 5%, 10%, 20%, 30%, and 50%) among the unlabeled examples. class_sep=0.3, number of attributes=100, number of positive subclasses=5, $|P|$ = 20,000 (4,000 from each subclass) and $|U|$ = 50,000. Calibration curves were generated using only unlabeled examples (Uncalibrated probabilities - blue, calibrated probabilities - red).

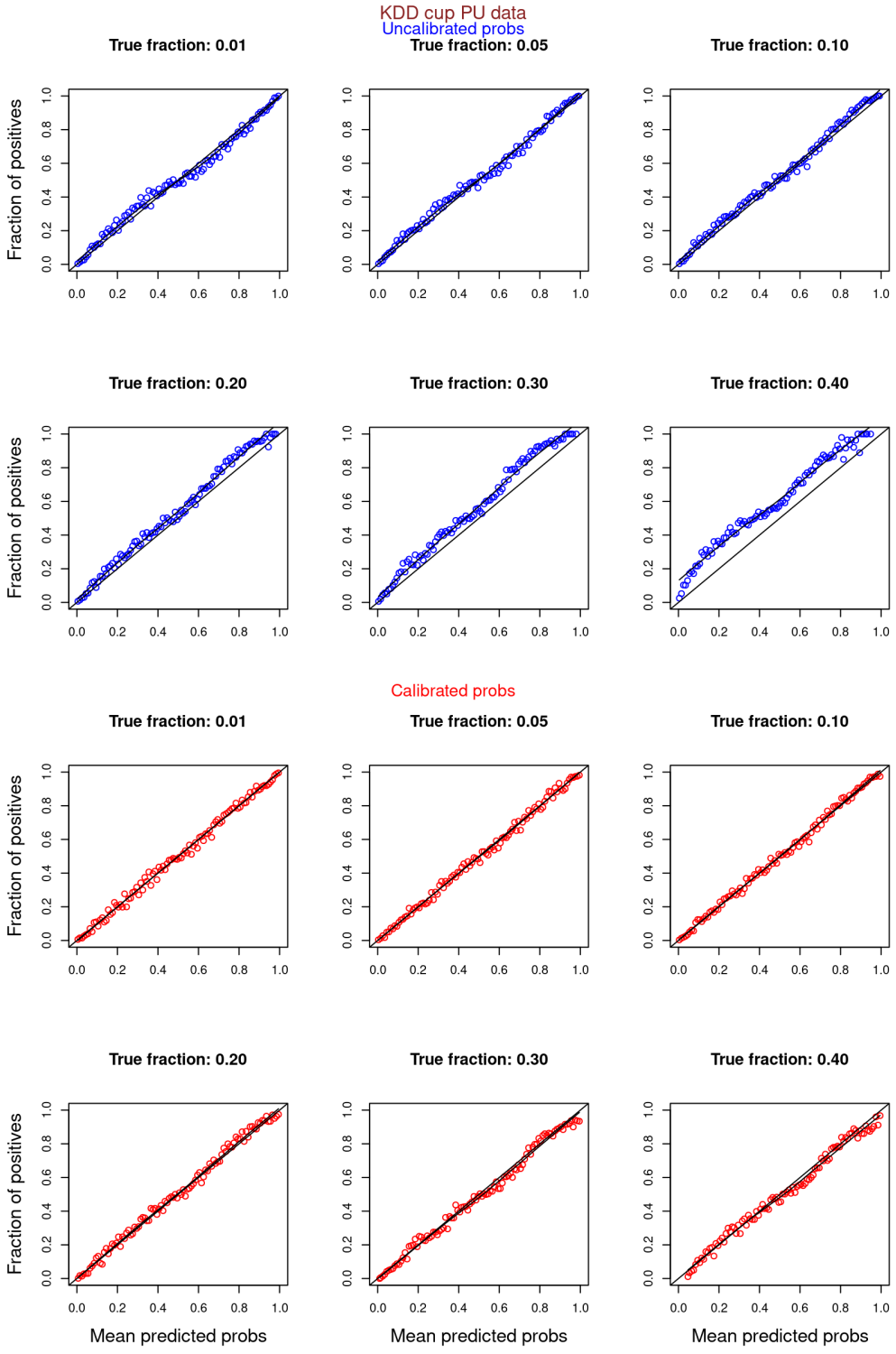

Figure 10: **Calibration curves for SCAR KDD Cup 2004 particle physics dataset (both positive and unlabeled examples)**. Unlabeled sets contained 1%, 5%, 10%, 20%, 30%, and 40% positive examples. Calibration curves were generated using both positive and unlabeled examples (Uncalibrated probabilities - blue, calibrated probabilities - red).

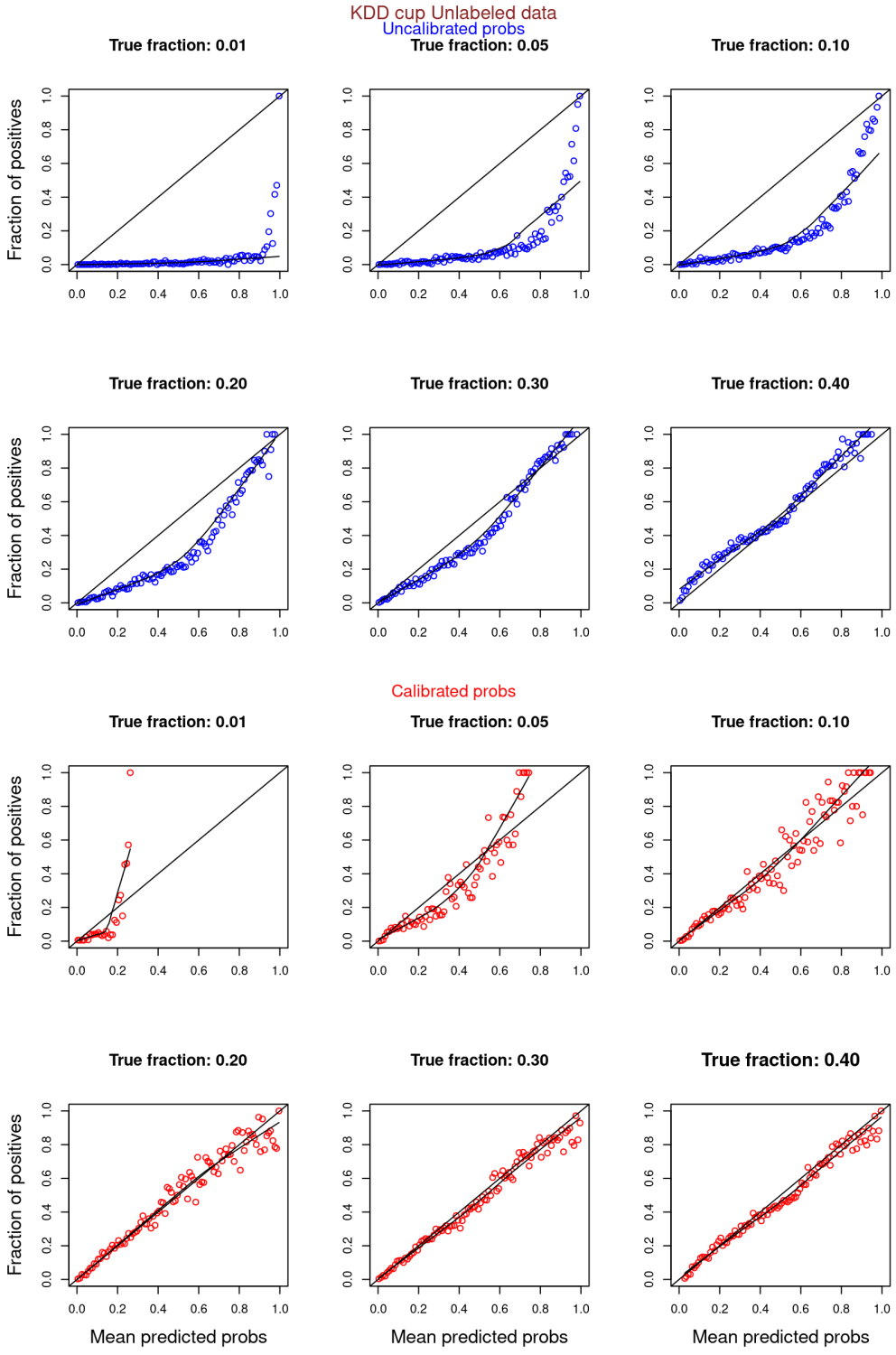

Figure 11: **Calibration curves for SCAR KDD Cup 2004 particle physics dataset (only unlabeled examples)**. Unlabeled sets contained 1%, 5%, 10%, 20%, 30%, and 40% positive examples. Calibration curves were generated using only unlabeled examples (Uncalibrated probabilities - blue, calibrated probabilities - red).

C Improving classification performance with PULSCAR and PULSNAR

Algorithm 5 shows the complete pseudocode to improve classification performance with PULSCAR and PULSNAR. The algorithm returns the following six classification metrics: *Accuracy, AUC-ROC, Brier score (BS), F1, Matthew's correlation coefficient (MCC)*, and *Average precision score (APS)*. The approach to enhancing the classification performance is as follows:

Using PULSCAR: After estimating the α, the class 1 predicted probabilities of only unlabeled examples are calibrated using Algorithm 4. The calibrated probabilities of the unlabeled examples are sorted in descending order, and the labels of top $\alpha|U|$ unlabeled examples with the highest calibrated probabilities are flipped from 0 to 1 (probable positives). We then train and test an ML classifier (XGBoost) with 5-fold CV using the labeled positives, probable positives, and the remaining unlabeled examples. The classification performance metrics are calculated using the ML predictions and the true labels of the data.

Using PULSNAR: The PULSNAR algorithm divides the labeled positive examples into k clusters. For each cluster, after estimating α_j for j in $1 \ldots k$, the class 1 predicted probabilities of only unlabeled examples are calibrated using Algorithm 4. Since each unlabeled example has k calibrated probabilities, we compute the final calibrated probability for each unlabeled example using the Formula 6. The final α is calculated by summing the α_j values over the k clusters. The final calibrated probabilities of the unlabeled examples are sorted in descending order, and the labels of top $\alpha|U|$ unlabeled examples with the highest calibrated probabilities are flipped from 0 to 1 (probable positives). We then train and test an ML classifier (XGBoost) with 5-fold CV using the labeled positives, probable positives, and the remaining unlabeled examples. The classification performance metrics are calculated using the ML predictions and the true labels of the data.

Algorithm 5 calculate_classification_metrics

Input: X ($X_p \cup X_u$), y ($y_p \cup y_u$), y_true, bin_method, n_bins, predicted_probabilities, α
Output: classification_metrics (accuracy, roc auc, brier score, f1, Matthew's correlation coefficient, average precision)

1: p $\leftarrow$ predicted_probabilities
2: $\hat{p} \leftarrow$ calibrate_probabilities(p, y, n_bins, calibration_method, 'U', α)
3: sort $\hat{p}$ in descending order
4: $\hat{y_u} \leftarrow$ flip labels of top $\alpha|U|$ unlabeled examples with highest $\hat{p}$
5: y $\leftarrow y_p \cup \hat{y_u}$
6: predicted_probabilities (p) $\leftarrow \mathcal{A}(X, y)$
7: **return** accuracy(p, y_true), auc(p, y_true), bs(p, y_true), f1(p, y_true), mcc(p, y_true), aps(p, y_true)

C.1 Experiments and Results

We applied Algorithm 5 to synthetic SCAR and SNAR datasets to get the performance metrics for the XGBoost model with PULSCAR and PULSNAR, respectively. The classification performance metrics were also calculated without applying the PULSCAR or PULSNAR algorithm, in order to determine the improvement in the classification performance of the model. The experiment was repeated 40 times by selecting different train and test sets using 40 random seeds to compute the 95% confidence interval (CI) for the metrics.

Figures 12 and 13 show the classification performance of the XGBoost model with/without the PULSCAR or PULSNAR algorithm on synthetic SCAR and SNAR data, respectively. The classification performance using PULSCAR or PULSNAR increased significantly over XGBoost alone. As the proportion of positives among the unlabeled examples increased, the performance of the model without PULSCAR or PULSNAR (blue) worsened significantly more than when using PULSCAR or PULSNAR.

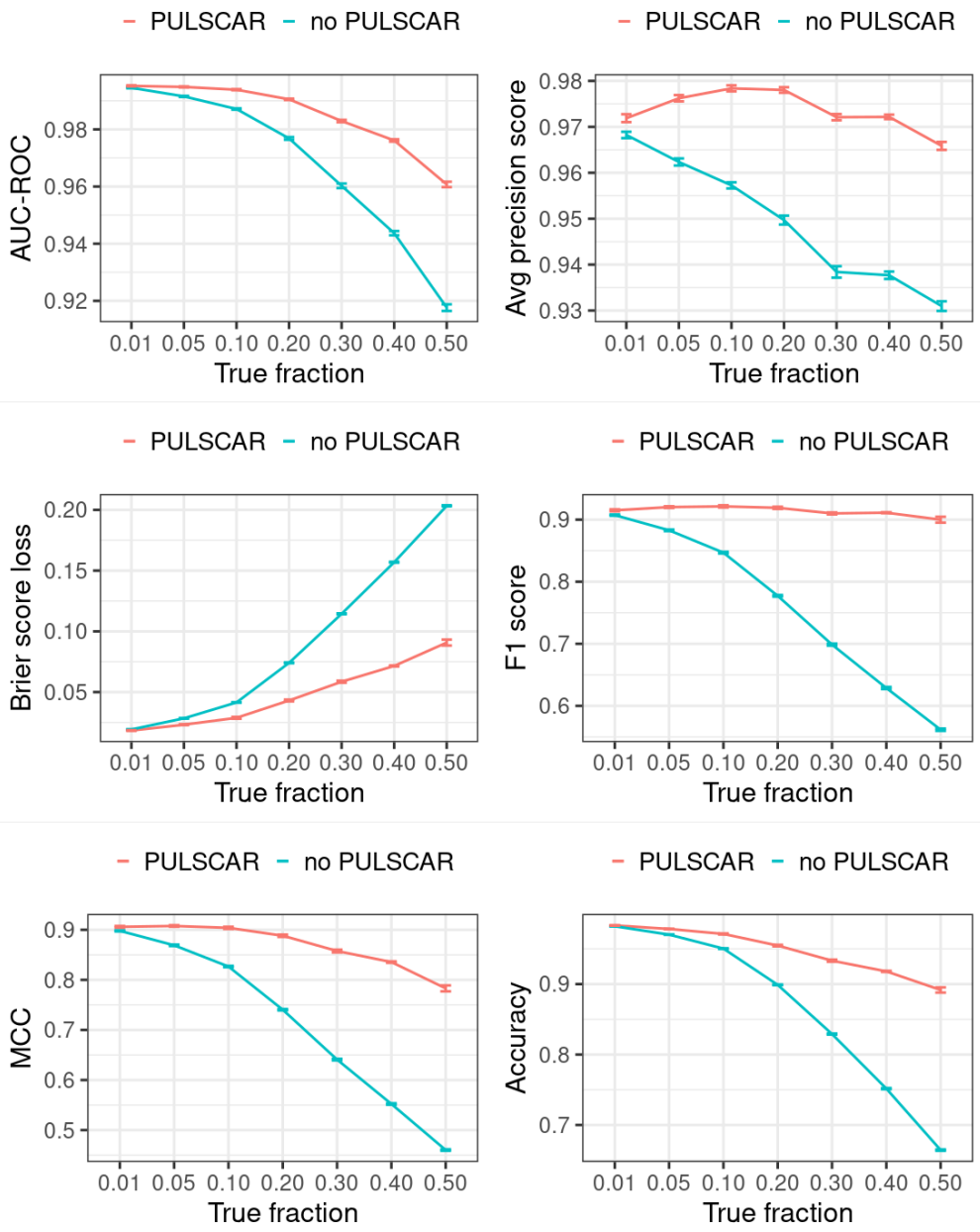

Figure 12: **Classification performance of XGBoost model on synthetic SCAR datasets with and without the PULSCAR algorithm**. Synthetic datasets were generated with different fractions of positives (1%, 5%, 10%, 20%, 30%, 40%, and 50%) among the unlabeled examples. class_sep=0.3, number of attributes=100, $|P| = 5,000$ and $|U| = 50,000$. *"no PULSCAR"* (blue): XGBoost model was trained and tested with 5-fold CV on the given data; the classification metrics were calculated using the model predictions and true labels. *"PULSCAR"* (red): PULSCAR algorithm was used to find the proportion of positives among unlabeled examples (α); using α, probable positives were identified; XGBoost model was trained and tested with 5-fold CV on labeled positives, probable positives, and the remaining unlabeled examples; classification metrics were calculated using the model predictions and true labels. The error bars represent 95% CIs for the performance metrics.

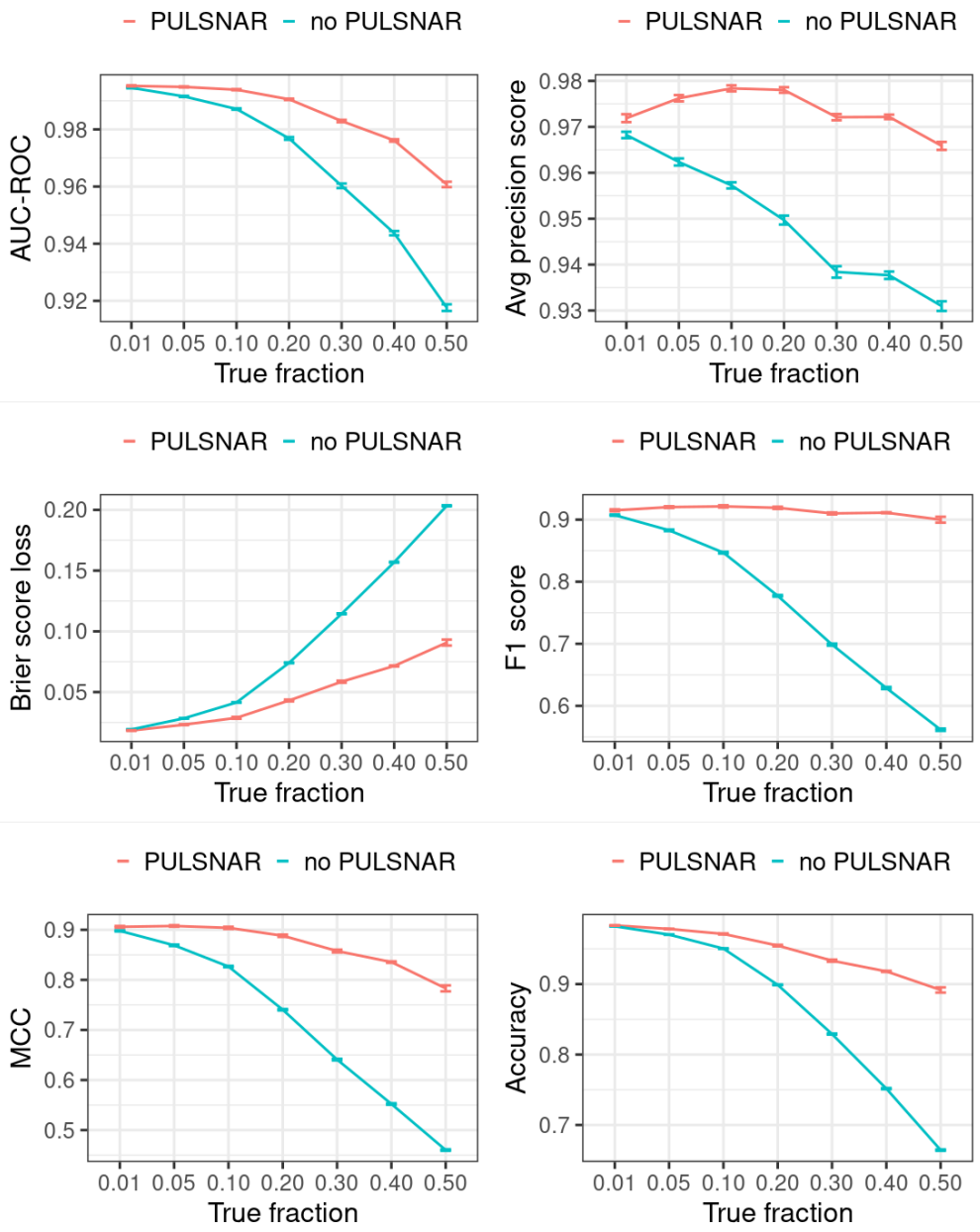

Figure 13: **Classification performance of XGBoost model on synthetic SNAR datasets with and without the PULSNAR algorithm**. Synthetic datasets were generated with different fractions of positives (1%, 5%, 10%, 20%, 30%, 40%, and 50%) among the unlabeled examples. class_sep=0.3, number of attributes=100, number of positive subclasses=5, $|P|$ = 20,000 (4,000 from each subclass) and $|U|$ = 50,000. *"no PULSNAR"* (blue): XGBoost model was trained and tested with 5-fold CV on the given data; the classification metrics were calculated using the model predictions and true labels. *"PULSNAR"* (red): PULSNAR algorithm was used to find the proportion of positives among unlabeled examples (α); using α, probable positives were identified; XGBoost model was trained and tested with 5-fold CV on labeled positives, probable positives, and the remaining unlabeled examples; classification metrics were calculated using the model predictions and true labels. The error bars represent 95% CIs for the performance metrics.

D DEDPUL vs. PULSNAR: Alpha estimation

Public implementations of the PU learning methods KM1, KM2, and TICE were not scalable; they either failed to execute or would have taken weeks to run the multiple iterations required to obtain confidence estimates for large datasets. We thus could not compare our method with KM1, KM2, and TICE on large datasets and used only DEDPUL for comparison. Importantly, it was previously demonstrated that the DEDPUL method outperformed these three methods on several UCI ML benchmark and synthetic datasets Ivanov (2020).

We compared our algorithm with DEDPUL on synthetic SNAR datasets with different fractions (1%, 5%, 10%, 20%, 30%, 40%, and 50%) of positives among unlabeled examples. In our experiments, we observed that class imbalance (ratio of majority class to minority class) could affect the α estimates. So, we used 4 different sample sizes: 1) positive: 5,000 and unlabeled: 5,000; 2) positive: 5,000 and unlabeled: 25,000; 3) positive: 5,000 and unlabeled: 50,000; 4) positive: 5,000 and unlabeled: 100,000. For each sample size and fraction, we generated 20 datasets using sklearn's *make_classification()* method with random seeds 0-19 to compute 95% CI. We used class_sep=0.3 for each dataset to create difficult classification problems. All datasets were generated with 100 attributes and 6 labels (0-5), defining '0' as negative and 1-5 as positive subclasses. The positive set contained 1000 examples from each positive subclass in all datasets. The unlabeled set comprised k% positive examples with labels (1-5) flipped to 0 and (100-k)% negative examples. The unlabeled positives were markedly SNAR, with the 5 subclasses comprising 1/31, 2/31, 4/31, 8/31, and 16/31 of the unlabeled positives.

Figure 14 shows the α estimates by DEDPUL and PULSNAR on synthetic SNAR data. For smaller true fractions (1%, 5%, 10%), DEDPUL returned close α estimates, but for larger fractions (20%, 30%, 40%, and 50%), it underestimated α. Also, as the class imbalance increased, the performance of DEDPUL dropped, especially for larger true fractions. The estimated α by the PULSNAR method was close to the true α for all fractions and sample sizes.

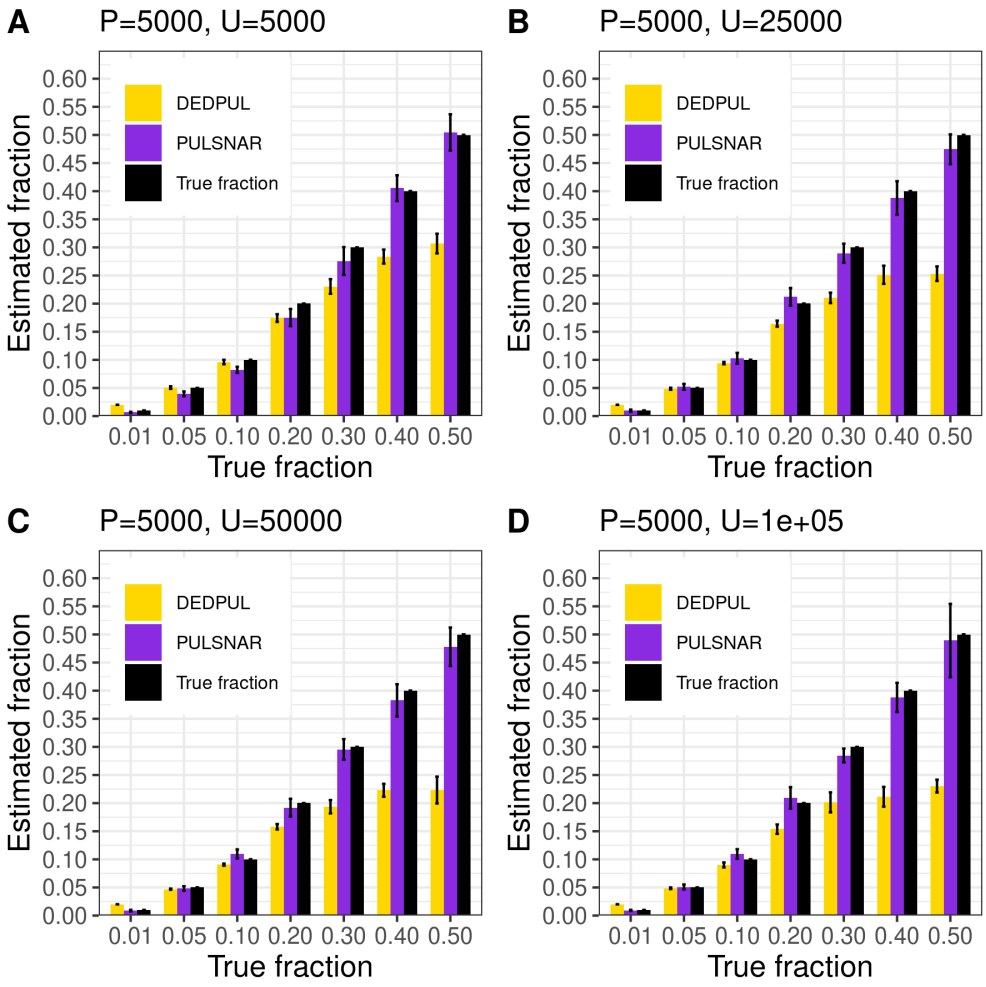

Figure 14: **PULSNAR and DEDPUL evaluated on synthetic SNAR datasets**. The bar represents the mean value of the estimated α, with 95% CI for estimated α.