Unknown Examples & Machine Learning Model Generalization

Yeounoh Chung  
Brown University  
yeounoh_chung@brown.edu

Eli Upfal  
Brown University  
eli_upfal@brown.edu

Peter J. Haas  
UMass Amherst  
phaas@cs.umass.edu

Tim Kraska  
MIT CSAIL  
kraska@mit.edu

ABSTRACT
Over the past decades, researchers and ML practitioners have come up with better and better ways to build, understand and improve the quality of ML models, but mostly under the key assumption that the training data is distributed identically to the testing data. In many real-world applications, however, some potential training examples are unknown to the modeler, due to sample selection bias or, more generally, covariate shift, i.e., a distribution shift between the training and deployment stage. The resulting discrepancy between training and testing distributions leads to poor generalization performance of the ML model and hence biased predictions. We provide novel algorithms that estimate the number and properties of these unknown training examples—unknown unknowns. This information can then be used to correct the training set, prior to seeing any test data. The key idea is to combine species-estimation techniques with data-driven methods for estimating the feature values for the unknown unknowns. Experiments on a variety of ML models and datasets indicate that taking the unknown examples into account can yield a more robust ML model that generalizes better.

1 INTRODUCTION
Over the past decades, researchers and Machine Learning (ML) practitioners have come up with better and better ways to build, understand and improve the quality of ML models, but mostly under the key assumption that the training data is distributed identically to the testing data. This assumption usually holds true in algorithm-development environments and data-science competitions, where a single dataset is split into training and testing sets, but does it hold more generally? If not, what are the consequences for ML?

Experience shows that the foregoing assumption can fail dramatically in many real-world scenarios, especially when the data needs to be collected and integrated over multiple sources and over a long period of time. This issue is well known, for example, to the Census Bureau. A 2016 Census Advisory Committee report [7] highlights the difficulties in reaching groups such as racial and ethnic minorities, poor English speakers, low income and homeless persons, undocumented immigrants, children, and more. Some of these groups do not have access to smartphones or the internet, or they fear interactions with authorities, so the prospects for data collection will remain difficult into the foreseeable future. Similarly, a recent report on fairness in precision medicine [8] documents bias in labeled medical datasets and asserts that “insofar as we still have a systematically describable group who are not in a health care system with data being collected upon them, from them, then that will be a source of bias.” In each of these cases, factors such as income and ethnicity can result in exclusion of items from a training set, yielding unrepresentative training data. We emphasize that the issue here is not just underrepresentation of classes of data items, but the complete absence of these items from consideration because they are unknown to the ML modeler.

Besides sampling bias, population shifts over time can lead to unrepresentative training data. For example, a regression model for predicting height based on weight that was trained on the US population in the 80’s may not be usable today, because the variables (population-wide height and weight distributions) have changed over time, so that the old training data do not represent the actual testing data of today.

In either case, the unrepresentativeness of the training data will adversely affect an ML model’s ability to handle unseen test data. Indeed, the better the fit to biased training data, the harder it becomes for the model to handle new test data. Clearly, mitigation of biased training data is crucial for achieving fair ML.

In this work, we focus on the impact of unknown training instances on ML model performance. The unknown examples during training can arise if the training distribution p(x) is different from the testing distribution p(x) due to sample selection bias—where some data items from the testing distribution are more or less likely to be sampled in the training data—and, more generally, covariate shift, where the training data and testing data distributions can be different for any reason. Although p(x) ≠ p′(x), we assume that p(y|x) = p′(y|x), so that the conditional distribution of the class variable y of interest is the same for both training and test data.

That is, the predictive relationship between x and y is the same; only the data distribution of the x-values differs.

Generalization is the ability of a trained ML model to accurately predict on examples that were not used for training [2]. Good generalization performance is a key goal of any practical learning algorithm. Ideally, we want to fit the model on a training set that well represents the hidden testing data or the target population—i.e., the training and testing data are drawn from the same distribution. If this is not the case, then we end up with unknown instances that are missing during the training. If the training data is biased, in a sense that some parts of the population are under-represented or missing in the training data (i.e., we have unknown instances), then the fitted model f∗ on that training data will be biased away from the optimal function f∗ and have poor generalization performance. This issue is orthogonal to the typical model complexity and generalization trade-off [26], where training and testing distributions are assumed to be identical. In Section 4, we show that both simple and complex models can suffer if the training data is biased.
1.1 Learning under covariate shift
Learning under covariate shift has been studied extensively [1, 14, 17, 24, 30]. An important observation from importance sampling states that the loss on the test distribution can be minimized by weighting the loss on the training distribution with the scaling factor, \( p(x)/p'(x) \) [22]. The previous work proposes many techniques to estimate the scaling factor or the training, testing or the conditional densities more accurately and efficiently, which in turn, require both training and “unlabeled” testing data.

Access to the unlabeled testing data (during training) is only feasible in a setting where the actual test data is provided, e.g., in a data science competition. However, using such a target dataset (or re-training the model after seeing the test data) may not be possible in many real applications. We therefore propose the first techniques for learning under covariate shift that require just the training data.

1.2 Our goal and approach
We aim to develop methods for mitigating unrepresentativeness in training data arising from sampling bias, or covariate shift more generally, thereby improving ML generalization performance. Our key idea is to exploit the fact that training data is typically created by combining overlapping data sets, so that instances often appear multiple times in the combined data. We first apply species-estimation techniques that use the multiplicity counts for the existing training instances to estimate the number of unknown instances. We then use this information to correct the sample by either weighting existing instances or generating synthetic instances. For the latter approach, we investigate both kernel density and interpolation techniques for generating feature values for the synthetic instances. As shown in our experiments over different types of ML models and datasets, correcting a training set by taking unknown instances into account can indeed improve model generalization.

2 THE IMPACT OF UNKNOWN EXAMPLES
In this section, we define unknown unknowns [5], and describe how common data collection procedures can produce a biased training data with unknown unknowns. Our goal is twofold: First, we are interested in training a model that generalizes better to the unseen examples (e.g., testing data); Second, we want to minimize the generalization gap between the training and the testing/validation scores. We formally state the problem and the learning objectives at the end of the section.

2.1 Problem Setup
A typical training data collection process involves sourcing and integrating multiple data sources, e.g., data crowdsourcing where each worker is an independent source [13, 16]. In this work, we assume that data sources are independent but overlapping samples \( S_j \), each obtained by sampling \( n_j = |S_j| \) data items from the underlying distribution \( p \); the sampling is without replacement, because a data source typically only mentions a data item once. \( p \) is also our target distribution for learning, and each data item \( x_i = <x_{i1}, x_{i2}, ..., x_{id}> \) has a sampling likelihood \( p'(x_i) \) and consists of \( d \) features/variables. The data sources are then integrated into a training data set \( S \) of size \( n_S = \sum_{j=1}^{J} n_j \); \( S \) contains duplicates because every data source is sampling from the same underlying population. If we integrate a sufficiently large number of sources, then \( S \) approximates a sample with replacement from \( p' \). The duplicate counts resulting from the overlap of the \( S_j \)'s enables the use of species estimation techniques to estimate the number of the missing, unseen test instances \( U \).

Ideally, we would like the integrated sample \( S \) to follow the target distribution \( p(x) \) for \( x \in S \). However, we assume covariate shift between \( p(x) \) the actual training data distribution of \( S \), denoted as \( p'(x) \). That is, \( p(x) \neq p'(x) \) but \( p(y|x) = p'(y|x) \) for all \( x \). As discussed previously, this situation may arise if (1) any of the integrated sources exhibits a strong sample selection bias or (2) the source is outdated, but the fundamental relationship between \( x \) and \( y \) is unchanged.

For training, we assume that, for \( S = \{(x_i, y_i)\} \), each class label \( y_i \in Y \) is perfectly curated. Neither testing data \( T \) nor the distribution \( p(x, y) = p(y|x)p(x) \) is available during training. We assume that the hidden test data comprises an i.i.d. sample from \( p(x, y) \), and well represents this distribution.

2.2 The Unknown Examples
We focus on the missing training examples that actually exist in the underlying data distribution or the test set, and now formally define such missing examples as unknown unknowns.

**Definition 1 (Unknown Unknowns).** Given an integrated data set \( S \sim p'(x, y) \) for training and a (hidden) testing data set \( T \sim p(x, y) \), the set \( U \) of unknown unknowns is defined as \( U = T - S \).

The terminology “unknown unknowns” stems from the fact that both the cardinality of \( U \) and the feature values of \( x \in U \) are both unknown. The existence of unknown unknowns critically impacts a model’s generalization ability.

2.3 Problem Statement
We quantify a model’s generalization ability via generalization error, the difference between the error (expected loss) with respect to the underlying joint probability distribution and the error (average loss) on the finite training data:

\[
G = \int L(f(x), y)p(x, y)dxdy - \frac{1}{n_S} \sum_{i \in S} L(f(x_i), y_i) \\
\approx \frac{1}{n_T} \sum_{i \in T} L(f(x_i), y_i) - \frac{1}{n_S} \sum_{i \in S} L(f(x_i), y_i), \quad (1)
\]

where \( n_T = |T| \) and \( L \) is a loss function such as \( L(x, y) = (x - y)^2 \).

Our goal is to minimize generalization error rather than training error in order to maximize the predictive ability on new data. However, \( \sum_{i \in T} L(f(x_i), y_i) \) is not available during training, and so ML training algorithms aim to minimize the empirical risk \((1/n) \sum_{i \in S} L(f(x_i), y_i)\), where \((x_i, y_i) \sim p'(x, y)\). Thus, any significant discrepancy between \( p(x) \) and \( p'(x) \) will be reflected in \( G \) through the average loss over the unknown examples. Training on \( S \) when \(|U| >> 0 \) can result in worse model performance on the actual testing data.

We now define the impact of unknown unknowns in the context of the generalization error.

**Definition 2 (The Impact of Unknown Unknowns).** Given an integrated data set \( S \sim p'(x, y) \) for training and a (hidden) testing data set \( T \sim p(x, y) \), the difference between the expected loss and the average loss over the unknown examples is defined as

\[
\Delta = \int L(f(x), y)p(x, y)dxdy - \frac{1}{n_T} \sum_{i \in T} L(f(x_i), y_i).
\]

We then show how to minimize this discrepancy.
data set \( T \sim p(x, y) \), the impact of unknown unknowns is defined as

\[
\Delta = \frac{1}{n_T} \sum_{i \in U} L(f(x_i), y_i) - \frac{1}{n_S} \sum_{i \in S} L(f(x_i), y_i)
\]

(2)

Under the sample selection bias model [30] where \( S \) is sampled from \( T \) (so that \( S \subseteq T \)), the generalization error is equivalent to the difference between the training and the testing scores over the unknown examples:

\[
G = \frac{1}{n_T} \sum_{i \in T} L(f(x_i), y_i) - \frac{1}{n_S} \sum_{i \in S} L(f(x_i), y_i)
\]

(3)

In a more general setting, in which training and testing data can differ arbitrarily, \( \Delta \) as defined in (2) only approximates the rightmost difference in (2) (and hence \( G \)) because \( U \) only contains missing testing instances from \( T \), not additional training instances in \( S - T \). In this work, we simply take \( G \) to approximate \( \Delta \) in the evaluation, and leave an estimation error bound (for the general setting) as future work.

Our goal is to capture the distributional difference between \( p(x) \) and \( p'(x) \) via unknown unknowns, and make the unknown examples part of model training \( (U \cup X) \). The challenge arises because neither \( T \) nor \( p(x) \) is available in training; we cannot directly compute \( U \). In Section 3, we explain how we Adding \( U \) to \( S \) can un-bias \( S \) or correct the training distribution \( p'(x) \) to resemble \( p(x) \).

### 3 LEARNING THE UNKNOWN

In this section, we focus on a simple regression problem to illustrate our techniques for learning the unknown examples. The proposed techniques can easily be extended to other problem types, such as classification (apply the same technique for each class label). In Section 4, we present the experimental results for both regression and classification problems.

We propose two approaches to model the unknown examples during training. Figure 1 provides an overview of different techniques and their results using a simple regression example.

#### 3.1 Weighting by Unknown Example Count

The first approach simply weights an existing training example by the estimated number of unknown unknowns that surround it. This is similar to the importance sampling-based techniques studied previously, where the instance-specific weights are computed based on \( p(x)/p'(x) \) [1, 22, 30]. Other techniques exist for learning the weights, directly or indirectly, from the training and testing distributions [14, 17], but they are not applicable if the target/test data \( T \) or the ground truth distribution \( p(x) \) is not available. We use a sample-coverage-based species estimation technique to estimate the size of the missing mass. The underlying assumption of the species estimator is that the rare species in a sample with replacement are the best indicators of the missing unknown species in the target distribution. In our case, we assume that the rare examples in our collected \( S \) are the best indicators of the unknown unknowns. We estimate a large number of unknown unknowns near a given training example, then we expect a large number of similar instances to be present in the actual test data or target population; thus, we attach more importance to such a training example during the training phase.

### 3.1.1 Chao92 Species Estimator

There are several species estimation techniques, and no single estimator performs well in all of the settings [11]. In this work, we use the popular Chao92 estimator, which is defined as:

\[
\hat{D}_{\text{Chao92}} = \frac{c}{C} + \frac{f_1 \cdot y^2}{C}
\]

(4)

where \( y \) is coefficient of variation and can be estimated as:

\[
y^2 = \max \left\{ \frac{\sum_i n_i (i - 1) f_i}{n(n - 1)} - 1, 0 \right\}
\]

(5)

Here \( c \) is the number of unique examples in the training data \( S \), \( \hat{C} \) the sample coverage estimate—i.e., the percentage of the \( T \) covered by \( S \) and \( D_{\text{Chao92}} \) our estimate of the total number of unique examples in \( T \). The sample coverage is estimated using the Good-Turing estimator [9]:

\[
\hat{C} = 1 - f_1/n
\]

(6)

The Good-Turing estimator leverages what are called the \( f \)-statistics, or the “frequencies of frequencies.” Specifically, \( f_1 \) denotes the number of singletons, the examples that appear exactly once in the integrated training data \( S \). Similarly, \( f_2 \) denotes the number of doubletons, the examples which appear exactly twice in \( S \), and so on.

Given the \( f \)-statistics, we can estimate the missing distribution mass of all the unknown unknowns as \( f_1/n \).

#### 3.1.2 Dynamic Bucketization

To estimate the number of unknown unknowns near each training example, we partition the data into buckets based on the values of a selected feature and then perform the estimation for each partition. Instead of partitioning the feature space statically, with fixed boundaries and sizes, we define the buckets dynamically, making sure that each partition contains enough examples and duplicates to permit high-quality estimation.

### ALGORITHM 1: Dynamic Bucketization

**Input**: Integrated training data \( S \), feature index \( v \), min sample coverage threshold \( \theta \)

**Output**: data partitions (buckets) \( B \)

1. \( B = [] \); /* buckets */
2. \( Q = \text{PriorityQueue}(S) \); /* priority queue sorted by ascending feature value \( x_{1U} \) of \( x_i \in S \) */
3. \( b = [] \); /* current bucket to fill */
4. while \( Q \) not empty do
   5.     \( x = Q.pop(); \) /* the next \( x_i \) by ascending \( x_{1U} \) */
   6.     if \( \text{sample_coverage}(b) \geq \theta \) then
      7.         \( B.append(b); \) /* if \( \hat{C} \geq \theta \), \( b \) has enough */
      8.     else
         9.         \( b = [x]; \) /* new bucket to fill */
   10.    end
   11. else
      12.     \( b.append(x); \)
   13. end

Algorithm 1 illustrates the mechanism. First, we push the training data \( S \) onto a priority queue sorted by ascending feature value.
with the highest entropy or variance. Afterwards, we group nearby examples by missing values in a less relevant feature dimension for the final ML task. Alternatively, we can also choose a feature with the highest entropy or variance. Afterwards, we group nearby examples \( x_i \in S \) in a way that each bucket has enough examples and duplicates, according to the sample coverage estimate, for the quality unknown unknowns species estimation (lines 4-13).

### 3.2 Synthetic Unknown Examples

The second approach tries to model the unknown unknowns more explicitly and generate synthetic training examples. Generating the synthetic examples for unknown unknowns requires estimating both the number of unknown unknowns and their feature values. For each bucket \( b \), we first use the species estimation technique and dynamic bucketization (Algorithm 1) to estimate a number \( n_b \) of unknown unknowns, and then we generate \( n_b \) synthetic unknown unknowns as described below.

The weighting approach in Section 3.1 is cleaner in the sense that it avoids the difficult feature-value estimation step, a source of additional uncertainty and error. Indeed, if we generate bad examples, then it might harm the final model performance against \( T \). A naive unknown value estimation would be using mean substitution [21], where the observed average feature values are used for any unknown unknowns. We can also try doing this at the bucket-level [5], but all in all, this does not add much value to the learner. Instead, we use a couple of data-driven oversampling techniques for unknown unknowns value estimation that are more aggressive than the weighting approach, but also do not generate values that depart arbitrarily far from the observed data distribution \( p'(x) \). As mentioned above, we hew to more conservative data-driven approaches to avoid adding bad examples that can harm the model’s generalization performance.

#### 3.2.1 KDE-Based Value Estimator

We use a Kernel Density Estimation (KDE) approach to estimate the probability density of each bucket and sample the missing unknown examples from it. This is effective, especially when covariate shift is mainly due to sample selection bias, i.e., \( S \subset T \), and unknown unknowns are similar to the observed training examples. On the other hand, the value estimator can actually mislead the training if \( p'(x, y) \) is very far apart from \( p(x, y) \).

For the purpose of this work, we used a Gaussian kernel and a “normal reference” rule of thumb [12] to determine the smoothing bandwidth, but other parameters are subject to study.

#### 3.2.2 SMOTE-Based Value Estimator

Synthetic Minority Oversampling Technique (SMOTE) is widely-accepted technique to balance a dataset [3]. A dataset is said to be imbalanced if different class examples are not equally represented. SMOTE generates extra training examples for the minority class in a very conservative way; the algorithm randomly generates synthetic examples in between minority class examples and their closest neighbors. Motivated by this class label-balancing algorithm, we generate synthetic unknown examples in a similar fashion.

Algorithm 2 illustrates how the synthetic unknown examples are generated. We generate exactly \( l \) unknown examples, where \( l \) is computed using the species estimation technique (line 2). We first initialize a dummy synthetic example with \( d \) features (an arbitrary example from the same feature space) (line 6). Next, we take \( k \) nearest neighbors \( N \) of a randomly picked example \( x_i \in S \), and also pick a neighbor \( x_j \in N \). Setting \( k \) high results in more aggressive data generation, since \( \delta \) (line 8) can be larger. Finally, we randomly generates \( d \) features in between \( x_i \) and \( x_j \) (lines 7-11).

### 4 EXPERIMENTS

We evaluate our unknown example learning techniques using real-world crowdsourced datasets as well as simulated ones. For the
We used Amazon Mechanical Turks (AMT) for real-world data. For each task, we provided a description of the task and the data to be processed. Each worker received $0.03 for each example/data item provided. Because we are collecting data from multiple workers, each treated as a data source, our final dataset contains duplicate information. Our crowdsourcing spans the following scenarios:

**U.S. Tech Employees & Revenue.** We used the crowd to collect a training dataset of 2722 records in order to see if there is a positive correlation between the size of a tech company and its revenue, and to build a model to predict a tech company’s revenue based on its number of employees. We do not have the ground truth as a complete list of U.S. tech companies. Instead, we use the entire crowdsourced data as testing data. In this case, the training data is more biased in the beggin of active NBA players, and the final dataset contains duplicate information. We have ground truth test data with all 439 active NBA players, which we hide during training.

**Hollywood Movie Budget & Revenue.** Do blockbusters always make lots of money? We want to build a classifier that predicts a movie’s success (i.e., make the production budget) based on the budget cost. We used the crowd to collect movie production budget and world-wide gross information (300 records with duplicates) for movies released from 1995 to 2018. The scale of movie production and gross income have changes a lot over the years; we use the Hollywood movies released between 1995 and 2015 as test data, which should cause more covariate shift between the training and testing data.

**U.S. College Ranking & Tuition.** We are also interested in the relationship between the U.S. college ranking and the school tuition. We use the crowd to collect this year’s U.S. college ranking and tuition from U.S. News Ranking2 (fixing the ranking system/the target population). We have a separate test dataset, which lists 220 U.S. college ranking and tuition for the same year. We have collected 300 college information with duplicates.

Workers sample from the same underlying population (for each question), and they are assumed to be independent. Yet, the combined datasets might still be biased (i.e., not a uniform random sample from the population) and contain more of the popular movies or NBA players, while missing many others due to this sample selection bias.

For each problem case, we train a simple ML model (e.g., linear regression) with default hyper-parameters. We focus on evaluating the learning techniques, given a blackbox ML algorithm.

Figure 2 compares testing error of the base model trained on the original training data (Original) and the models trained on a weighted training data (WeightByUnk, Section 3.1), and the models trained with synthetic unknown examples using two different unknown value estimators (SynUnk (KDE) and SynUnk (SMOTE), Section 3.2), on the real-world crowdsourcing problems. We use the mean absolute error metric:

$$\text{MAE} = \frac{1}{m} \sum_{i \in T} |y_i - f(x_i)|$$  \hspace{1cm} (7)

The scales on the y-axes are not normalized and dependent on the actual target variables for the problems. It is interesting to see that the different techniques win in different problem cases; Synthetic Unknown Examples approaches (SynUnk) seem to perform at least as well as the baseline (Original) in all four cases. Adding synthetic examples can be useful [10], and we generate feature values conservatively near the existing training points in order to minimize bad examples.

On the other hand, and surprisingly, the weighting by unknown count estimates is less consistent even though it does not require estimating the unknown example features; WeightByUnk performs the best in NBA Player Body Measurement, but also the worst in the other three problems. This is because the unknown example count estimates can be inaccurate, but more importantly, they are not necessarily the best weighting factors: there is no guarantee

---

1. https://archive.ics.uci.edu/ml/datasets.html
2. https://www.usnews.com/best-colleges/rankings
that putting more importance on rare items would correct the sample bias nor the sample selection bias actually existed from the beginning. Unlike the prior work, we do not assume \( p(x) \) or (unlabeled) \( T \) to figure this out. And yet, WeightByUnk can be useful under covariate shift.

### 4.2 Simulation Study

Next, we evaluate the proposed techniques in simulations using UCI ML repository datasets:

- **Adult Dataset** [15]. The prediction task is to predict if a person makes over 50K/year or not. The repository has separate training and testing data, and each example instance consists of 14 numerical and categorical features. We use only the original training data for training, and the original testing data for the evaluation (testing scores/errors).

- **Auto MPG Dataset** [19]. The prediction task is to predict city-cycle fuel consumption in miles per gallon. There are 8 features, both numerical and categorical. The original dataset is collected over three different cities; to simulate a sample selection bias, we sample examples mostly from city 1 for training and use the examples from all the cities (1, 2, and 3) for testing.

For all simulations, we permute/re-sample the training dataset to repeat the experiments \( r = 20 \). For Adult dataset, we have a separate training and testing datasets. We re-sample from the training data with replacement, so the new simulated training data contains duplicates. The sampling procedure is slightly biased to favor people with higher education backgrounds. For Auto MPG dataset, we inject a sampling bias in a way that our training dataset mostly consists of the examples from a particular location. We also sample the training data with replacement. This simulates the data collection process that combine multiple data sources.

Figure 3 shows the classification scores on Adult Dataset test dataset, using the proposed learning technique and different ML algorithms. For some ML algorithms, like Neural Network, training with instance-specific weights is not possible, so for such classifiers we do not show the scores. For the evaluation metric, we used F1 Score:

\[
F_1 = 2 \frac{precision \cdot recall}{precision + recall} \tag{8}
\]

In binary classification with positive and negative class labels, \( precision \) is the fraction of correct positive predictions among all positive predictions, and \( recall \) is the fraction of correct positive predictions among all true positives. It is interesting to see that a simple ML algorithm, like logistic regression, is more sensitive to unknown unknowns; SYNUnk (SMOTE) improves the model generalization by far. Random forest classifier is known for a better generalization [29], and we see that it performs better than the other classifiers even with the original data. As the algorithm is less sensitive to covariate shift, our techniques can help only so much. For neural network classifier, training with instance-specific weights is not applicable; thus, we do not show any result for WeightByUnk. Once again, SYNUnk (SMOTE) is a relatively safer technique in that it performs at least as well as the original with all three algorithms.

Figure 4 illustrates how our learning the unknown techniques compare to the existing techniques on Adult dataset. We take two existing techniques based on the importance-sampling result [22]. Two-Stage LR first learns a logistic regression model \( \hat{f} \) to classify if an example belongs to \( S \) and \( T \) (this requires an access to \( T \); next, the scale-factor is computed as follows [1]:

\[
p(x) = \frac{\eta S}{\eta T} \left( \frac{1}{f(x)} - 1 \right) \tag{9}
\]

\( \hat{f} \) outputs the likelihood that \( x \) belongs to \( S \). Two-Stage LR (SSB) is simpler in that it uses (normalized) \( 1/f(x) \) to scale \( x \). The existing techniques do not perform better for this particular example, but adding synthetic examples did improve the model generalization.

Figure 5 illustrates how our techniques can reduce the generalization error or the impact of unknown unknowns. We evaluated our techniques using the Auto MPG dataset, and we show the evaluation error on both \( S \) and \( T \) over the increasing training data size [5]. And the techniques do not perform better than the original in terms of the actual test error after training on the full training data. However, adding synthetic unknown examples (SynUnk (KDE))
Figure 3: UCI Adult Dataset classification test scores (F1 scores; the higher the better). Overall, the proposed techniques either improve or perform as well as the baseline (original), in terms of model generalization. It is interesting to see that a simple ML algorithm, like logistic regression, is more sensitive to unknown unknowns or the covariate shift. Random forest classifier is known for a better generalization, and we see that it performs better than the other classifiers even with the original data. For neural network classifier, training with instance-specific weights is not applicable; thus, we do not show any result for WeightByUnk.

Figure 4: Comparison with existing techniques (Two-Stage LR and Two-Stage LR (SBB)) on Adult Dataset. Both techniques look at $p(x)$ and use $p(x)/p'(x)$ to re-scale the original training data. Though given the target distribution (or its estimate) to adjust the importance of $x \in S$, the existing techniques do not perform any better than the original. We suspect that $S$ and $T$ provided by UCI repo are indeed similarly distributed. On the other hand, adding synthetic examples improve the generalization.

results in reduced generalization error. This means that the training error better represent what is to be expected on the actual testing data. This is much desirable in practice, since ML algorithm assumes that the training distribution follows the testing, and optimizes for the best training score/error. Under covariate shift or the sample selection bias, this can be problematic.

4.3 Which Technique To Use?
Learning under covariate shift is not an easy problem. Doing so even without any reference dataset (e.g., the unlabeled test data) to detect the distributional shift and un-bias the training data is more challenging. In this work, we have proposed learning techniques combined with species estimation, that allows us to reason about the unknown missing mass.

As we have seen in this evaluation section, no single technique always performs better than working with the original data. Worse yet, there are cases where all techniques fail and, sometimes, perform worse than the original. To this end, we have been more conservative about generating synthetic training examples, in a way that we would interpolate or re-sample near the existing examples.

We would need to study more to have more thorough guidelines, but as a rule-of-thumb, based on our experiences, SynUnk (SMOTE) worth the shot when any sampling bias or covariate shift is suspected.

5 RELATED WORK
To the best of our knowledge, this is the first work to consider learning under covariate shift without an access to the (unlabeled) test data, but instead using species estimation techniques.

Learning under covariate shift or sample selection bias has been studied extensively [1, 14, 17, 22, 24, 30], for training and test distributions diverge quite often and for many reasons in practice. As mentioned in Section 1.1, most known learning techniques under
covariate shift requires the (unlabeled) test data, which may not be available in many real applications.

The situation where training and test data follow different distributions is also related to transfer learning, domain adaptation and dataset-shift adaptation [6, 18, 23]. In transfer learning, knowledge or model built in one problem is transferred to a similar problem. Here, the conditional distribution $p(y|x)$ is not constant and the learner is even asked to predict different labels. In that case, model is re-trained, at least partially, to adapt to a new problem.

There are also techniques to detect covariate shift. The most intuitive and direct approach would be taking the two distributions, training and testing, and perform use Kullback-Leibler divergence model [27] or Wald-Wolffowitz test [28] to detect any significant data-shift. Researchers also have looked at covariate shift detection where the distribution is non-stationary [20]. In our case, we cannot test to see if a training distribution is really different from a testing distribution, which is hidden.

Species estimation techniques has been studied in prior work for distinct count estimation, data quality estimation, and crowdsourced data enumeration [4, 5, 11, 25]. In this work, we use species estimation techniques to model the unknown examples, missing from the training data. This allows us to correct biased training data without the test data.

6 CONCLUSION

Good model generalization is critical for any practical learning algorithm. And hoping that the training distribution closely follows the testing distribution, many ML algorithms simply optimizes for the best training score, taken as an expected testing score on a hidden target population (i.e., test data). Unfortunately, this is often risky under covariate shift (i.e., training and testing data are not sampled from the same distribution).

In this work, we have developed novel techniques for learning the unknown examples that account for the distributional shift. The key challenge is that we do not have an access to any test data (or the ground truth distribution), labeled or unlabeled. Prior work for learning under covariate shift compared the training and the (unlabeled) testing data to detect and correct the shift. Instead, we use the fact that training data is created by combining multiple sources, yielding duplicate instances, to apply the species estimation technique; we then either explicitly model the missing unknown examples or scale the existing examples, all without using the test data in training. Our experimental results using real-world data and simulations show that the proposed techniques can help improve model generalization.

There are a number of interesting directions for future work. So far, we have focused on blackbox approaches where we work directly with the data. Instead, we plan on working directly with the ML algorithms. We saw that not all ML models are created equal (i.e., some are more sensitive to unknown unknowns), and
by taking this to a further study, we hope to develop a more robust learning algorithm. We are also interested in developing species estimation based covariate shift detector, as well as better unknown example value estimators.

REFERENCES

[1] Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2007. Discriminative learning for differing training and test distributions. In Proceedings of the 24th international conference on Machine learning. ACM, 81–88.

[2] Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research 16 (2002), 321–357.

[4] Yeounoh Chung, Sanjay Krishnan, and Tim Kraska. 2017. A Data Quality Metric (DQM): How to Estimate the Number of Undetected Errors in Data Sets. Proc VLDB Endow 10, 10 (June 2017), 1094–1105. https://doi.org/10.14778/3115404.3115414

[5] Yeounoh Chung, Michael Lind Mortensen, Carsten Binnig, and Tim Kraska. 2018. Estimating the Impact of Unknown Unknowns on Aggregate Query Results. ACM Trans. Database Syst. 43, 1, Article 3 (March 2018), 37 pages. https://doi.org/10.1145/3167970

[6] Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. Journal of Artificial Intelligence Research 26 (2006), 101–126.

[7] J. Dowling et al. 2016. Final Report: National Advisory Committee on Racial, Ethnic, and Other Populations Administrative Records, Internet, and Hard to Count Population Working Group. Technical Report. U.S. Census Bureau. www2.census.gov/cac/nac/reports/2016-07-admin_internet-wg-report.pdf

[8] Kadija Ferryman and Mikaela Pitcan. 2018. Fairness in Precision Medicine. Technical Report. Data&Society. tinyurl.com/7g5xznz

[9] I. J. Good. 1953. The Population Frequencies of Species and the Estimation of Population Parameters. Biometrics 40, 3/4 (1953).

[10] Thien M. Ha and Horst Bunke. 1997. Off-Line, Handwritten Numeral Recognition by Perturbation Method. IEEE Trans. Pattern Anal. Mach. Intell. 19, 5 (May 1997), 535–539. https://doi.org/10.1109/34.589216

[11] Peter J. Haas, Jeffrey F. Naughton, S. Seshadr, and Lynne Stokes. 1995. Sampling-Based Estimation of the Number of Distinct Values of an Attribute. In PVLDB. 311–322. http://www.vldb.org/conf/1995/P311.PDF

[12] Daniel J Henderson and Christopher F Parmeter. 2012. Normal reference bandwidths for the general order, multivariate kernel density derivative estimator. Journal of statistical planning and inference 142, 11 (2012), 2501–2516.

[13] Wei-Yun Hsieh, Prem Melville, and Vikas Sindhwani. 2009. Data quality from crowdsourcing: a study of annotation selection criteria. In Proceedings of the NAACL HLT 2009 workshop on active learning for natural language processing. Association for Computational Linguistics, 27–35.

[14] Jae-yuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. 2007. Correcting sample selection bias by unlabeled data. In Advances in neural information processing systems. 601–608.

[15] Ron Kohavi. 1996. Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In KDD. Vol. 96. Citeseer, 202–207.

[16] Matthew Lease. 2011. On quality control and machine learning in crowdsourcing. Human Computation 11, 11 (2011).

[17] Anqi Liu and Brian Ziebarth. 2014. Robust classification under sample selection bias. In Advances in neural information processing systems. 37–45.

[18] Sinno Jialin Pan, Qiang Yang, et al. 2010. A survey on transfer learning. IEEE Transactions on knowledge and data engineering 22, 10 (2010), 1345–1359.

[19] J Ross Quinlan. 1993. Combining instance-based and model-based learning. In Proceedings of the tenth international conference on machine learning. 236–243.

[20] Haider Raza, Girijesh Prasad, and Yuhua Li. 2015. EWMA model based shift in machine learning. The MIT Press.

[21] Joseph L Schafer and John W Graham. 2002. Missing data: our view of the state of the art. Psychological methods 7, 2 (2002), 147.

[22] Masashi Sugiyama, Neil D Lawrence, Anton Schwaighofer, et al. 2017. Dataset shift in machine learning. The MIT Press.

[23] Masashi Sugiyama, Makoto Yamada, and Martthus Christoffel du Plessis. 2013. Learning under nonstationarity: covariate shift and class-balance change. Wiley Interdisciplinary Reviews: Computational Statistics 5, 6 (2013), 465–477.

[24] Beth Trushkowsky, Tim Kraska, and Purnamrita Sarkar. 2016. Answering enumeration queries with the crowd. Commun. ACM 59, 1 (2016), 118–127. https://doi.org/10.1145/2845644

[25] Wikipedia contributors. 2017. Overfitting — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Overfitting [Online; accessed 22-July-2018].

[26] Wikipedia contributors. 2018. Kullback-Leibler divergence — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence [Online; accessed 22-July-2018].

[27] Wikipedia contributors. 2018. Wald-Wolfowitz runs test — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Wald%E2%80%93Wolfowitz_runs_test [Online; accessed 22-July-2018].

[28] Wikipedia contributors. 2017. Overfitting — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Overfitting [Online; accessed 22-July-2018].

[29] Abraham J Wyner, Matthew Olson, Justin Bleich, and David Mease. 2017. Explaining the success of adaboost and random forests as interpolating classifiers. The Journal of Machine Learning Research 18, 1 (2017), 1588–1590.

[30] Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In Proceedings of the twenty-first international conference on Machine learning. ACM, 114.